Entwicklung eines Indoor-Assistenzsystems für Multicopter

mit Hilfe von Monocularer Tiefenbild Rekonstruktion

Studienarbeit

Studiengang Angewandte Informatik

Duale Hochschule Baden-Württemberg Karlsruhe

von

Christoph Meise, Max Lenk

Datum der Abgabe:	15.05.2017
Bearbeitungszeitraum:	2 Semester
Matrikelnummern und Kurse:	4050853, 3460046, TINF14B2, TINF14B1
Betreuer:	Markus Strand

Ehrenwörtliche Erklärung

"Ich erkläre ehrenwörtlich:

1. dass ich meine Projektarbeit mit dem Thema

   Entwicklung eines Indoor-Assistenzsystems für Multicopter

   ohne fremde Hilfe angefertigt und selbstständig verfasst habe;
2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe;
3. dass ich meine Projektarbeit bei keiner anderen Prüfung vorgelegt habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird."

Walldorf, der 16.09.2016

Christoph Meise, Max Lenk

Restriction notice

This report contains confidential information of

SAP SE

Dietmar-Hopp-Allee 16

69190 Walldorf, Germany

It may be used for examination purposes as a performance record of the department of Applied Computer Science at the Cooperative State University Karlsruhe. The content has to be treated confidentially.

Duplication and publication of this report - as a whole or in extracts - is not allowed.

Sperrvermerk

Der Inhalt dieser Arbeit darf weder als Ganzes noch in Auszügen Personen außerhalb des Prüfungsprozesses und des Evaluationsverfahrens zugänglich gemacht werden, sofern keine anders lautende Genehmigung der Ausbildungsstätte vorliegt.

Abstrakt

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Autonomes Fahren, Machine Learning und Industrie 4.0. Hinter diesen aktuellen Themen steckt das Ziel, Abläufe und Zusammenhänge kontinuierlich zu automatisieren und für den Menschen zu vereinfachen. Das Thema der Automation kann vor allem in der Interaktion zwischen Mensch und Maschine hilfreich sein. Auf Grund der geringen Kosten und der hohen Anwendungsvielfalt bieten sich vor allem Drohnen für die Forschung zu Automation in der Robotik an.
Mit Modellen ab 30 und bis zu mehreren tausend Euro gibt es Ausführungen für nahezu jeden Anwendungsfall. Meist mit mehreren Sensoren und Kameras ausgestattet, stellen sie nicht nur Spielzeug dar, sondern sind essentiell für reale Anwendungsgebiete.
So werden heute schon Drohnen genutzt, um Katastrophengebiete und Kriegsregionen aus sicheren Standorten aufzuklären, oder um die Feuerwehr bei der Branderkundung und Menschensuche zu unterstützen.
Natürlich können sie auch genutzt werden, um alltägliche Probleme zu lösen, wie die schnelle und direkte Lieferung von Paketen.
Da diese große Zahl an Drohnen nicht mehr manuell gesteuert werden kann, müssen sich diese größtenteils autonom bewegen. Dabei treten eine Vielzahl von komplexen Problemen auf, wie das zurechtfinden in einem unbekannten Raum und die Objekterkennung.
Außerdem ist bei Fluggeräten das Problem, dass die verwendete Ausrüstung ten-

denziell leicht und klein sein muss, damit die Flugeigenschaften nicht eingeschränkt werden bzw. die Drohne nicht zu groß wird.
Im Umfang dieser Arbeit soll eine Vorstufe zum autonomen Fliegen betrachtet werden: ein Assistenzsystem für den manuellen Flug. Dies ist Vergleichbar mit den Assistenzsystemen in PKWs, bei denen ein Tempomat, Licht- und Regensensoren oder Spurhalteassistenten den Fahrer unterstützen, jedoch das Fahren nicht abnehmen.

1.1 Motivation

Das Ziel besteht darin, dass die Drohne aktiv die Umgebung auswertet und dabei Objekte wie Türen, oder Wände erkennt. Der Unterschied zu bereits bestehenden Projekten in diesem Themengebiet besteht darin, dass nur eine einzelne monokulare Kamera verwendet werden soll, anstatt das externe Tiefenbildkameras montiert werden.
Dadurch trifft man auf eine Vielzahl von komplexen Problemen, welche im weiteren Verlauf dieser Arbeit dargestellt werden. Auf der anderen Seite könnte man dadurch teure Hardware sparen und somit auch auf andere Projekte anwenden.
Anhand der Tiefenbilder soll die Drohne anschließend in der Lage sein, entgegen der Entscheidungen des Nutzers zu fliegen, um somit beispielsweise Kollisionen zu vermeiden.
Da das Problem nur durch die integrierte Hardware gelöst werden soll, kann nur auf einfache mittelmäßige Kameras zurückgegriffen werden, welche nach Vorne und zum Boden gerichtet sind. Dies soll die Drohne weiterhin auch in einer vollständig simulierten Umgebung können.

1.2 Aufbau

Diese Studienarbeit besteht aus 3 Kapiteln. Im ersten Abschnitt der Arbeit sollen die Grundlagen erklärt werden. Zuerst wird dabei die genutzte Drohne und deren Spezifikationen dargestellt.
Anschließend wird das Software Framework Roboter Operating System *(ROS)* eingeführt, welches ein Hauptbestandteil der Projektarchitektur ausmacht.
Im weiteren Verlauf wird dann die Simulationsumgebung beschrieben, da das Projekt sowohl unter realen Bedingungen, als auch in einer simulierten Welt soll. Im letzten Teil dieses Kapitels wird die später verwendete Fuzzylogik erklärt.
Der zweite Abschnitt dieser Arbeit umfasst die Software Architektur. Hierbei sollen sowohl die Anforderungen an das Projekt, als auch die Implementierungsstechnischen Spezifikationen dargelegt werden. Im dritten Teil besteht aus dem erzielten Ergebnis, sowie aus dem Ausblick für weitere Betrachtungen dieser Problematik.

2 Grundlagen

2.1 AR.Drone 2.0

Bei der AR.Drone 2.0 handelt es sich um einen ferngesteuerten Quadrocopter des französischem Herstellers Parrot SA. [1] Die Drohne ist standardmäßig steuerbar mit einer mobilen Applikation für Android und iOS Geräte. Dafür baut sie ein WLAN Netzwerk auf, mit dem sich die Geräte verbinden können. Zur Steuerung stellt die AR.Drone ein Interface zur Verfügung, mit dem sie ferngesteuert werden kann.
Im Umfang der Studienarbeit wird die aktuellste Version der AR.Drone 2.0 verwendet. Diese zeichnet sich unter Anderem durch eine Frontkamera mit einer Auflösung von 1280×720 Pixeln und einer Bildrate von 30 fps aus. Weiterhin ist Sie mit einer zum Boden gerichteten QVGA Kamera ausgerüstet, welche 60 Bilder pro Sekunde aufnimmt.
Die Drohne orientiert sich beim Fliegen mit Hilfe einer Vielzahl von Sensoren. Dazu gehören ein dreiachsiges Gyroskop und ein Magnetometer. Weiterhin nutzt sie Beschleunigungs-, Ultraschall- und Luftdrucksensoren.
Der Grund für die Wahl der Drohne ist vor allem der vergleichsweise niedrige Preis von ca. 200€ und der starken Verbreitung in der Forschung. Dadurch gibt es bereits eine Vielzahl von Projekten, die dazu führen, dass die Drohne und das dazugehörige Interface zu einem großen Umfang fehlerfrei funktionieren.
Weiterhin gibt es schon ROS Nodes (siehe 2.2) und konfigurierte Modelle in Simulationsumgebungen, welche die Arbeit an dem Projekt beschleunigen.

2.2 ROS

2.2.1 Allgemeines

Das Robotic Operating System, kurz ROS, ist eine Sammlung von Softwareframeworks für die Entwicklung von Software für persönliche Roboter. Es stellt entsprechende Bibliotheken und Werkzeuge zur Verfügung, um Entwicklern die Programmierung zu vereinfachen. Dabei bietet ROS einem Betriebssystem ähnliche Funktionalitäten auf Basis eines homogenen Computercluster. Dazu gehören Hardwareabstraktion, low-level Steuerung, Nachrichtevermittlung zwischen verschiedenen Prozessen und Paketmanagement. Trotz der Notwendigkeit hoher Reaktivität und geringer Latenz bei der Steuerung von Robotern handelt es sich es de facto nicht um ein richtiges Betriebssystem , obwohl es durch den Namen(“Operating System“) suggeriert wird. Dennoch ist es möglich Echtzeitcode (“realtime code“) in ROS zu integieren [2]. ROS ist eins der am meisten genutzten Frameworks und hat eine stark wachsende Gemeinschaft, was es in Kombination mit dessen Feature zu einer enorm wichtigen Technologie macht.

2.2.2 ROS Nodes

ROS baut auf einem einfachen Konzept auf, dem Publish-Subscribe Pattern, bei welchem ein Publisher ein Nachricht mit einem festem Thema verschicken kann und ein beliebiger Subscriber, der sich für das Thema interessiert, ist in der Lage diese zu empfangen, zu verarbeiten und unter Umständen erneut zu versenden. Somit besteht eine ROS Anwendung aus der Regel aus vielen kleinen Teilen, den sogenannte Nodes. Jede Node hat ihre eigenen Aufgabe und registriert sich auf ein bestimmtes Thema(Topic), verarbeitet die empfangen Daten und publiziert sie für andere Nodes. Somit kann eine Node sowohl Publisher, als auch Subscriber sein. Topics werden in ROS durch einen festen Nachrichtentyp definiert. Dessen exakter Aufbau muss vor

Verwendung deklariert werden um einen einheitlichen Nachrichtenaustausch zu gewähren. Wie in der Abbildung unterhalb zu sehen, wird der Nachrichtentransfer durch eine zentrale Einheit geregelt, sodass Daten wirklich nur die Nodes erreichen, die sich auch dafür interessieren. Dieser zentrale Bestandteil ist in ROS der Roscore, bei welchem sich alle Nodes zur Erstellung registrieren. Sozusagen eine Masternode, welche sich um die Verwaltung der anderen Nodes kümmert. Im Unterschied zum Pattern kümmert sich der Roscore nur um die anfängliche Vermittlung der Nodes untereinander.

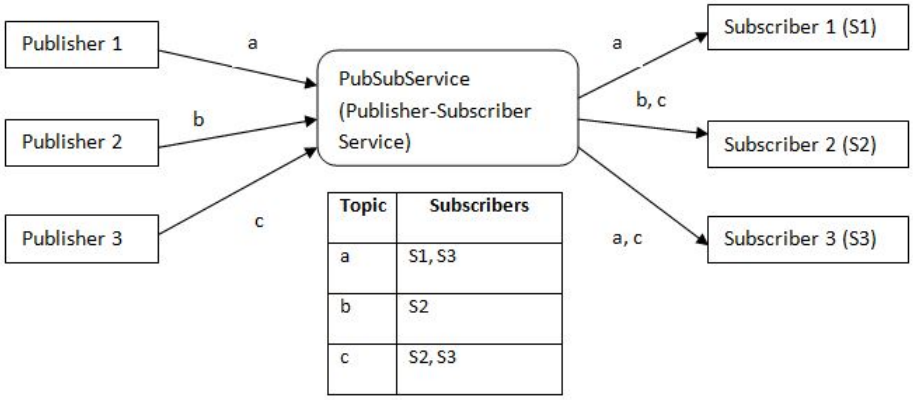


Abbildung 2.1: Publish-Subscribe Pattern

ROS Nodes können in verschiedenen Sprachen implementiert werden, da die Kommunikation über die festen Nachrichtentypen geschieht, welche über den Roscore ausgetauscht werden. Dadurch ist es möglich Nodes in C++, Python und ebenfalls Lisp zu programmieren. Wodurch das ganze Konzept enorm flexibel gestaltet wird. Die fest definierten Nachrichtentypen verhindern, dass es an den Schnittstellen zu Problemen kommt und Nachrichten von allen Nodes einheitlich empfangen und versendet werden.

Durch diese Modularität ist einfacher Komponenten und Funktionalitäten sowohl zu verwalten, als auch zu warten. Ebenfalls sind durch einheitliche Schnittstellen

der Austausch von Nodes einfach und ermöglicht ein flexibles Ökosystem.

2.3 Simulation

Da es besonders beim Flug von Quadrocoptern schnell zu Schäden kommen kann und das sowohl in langen Ausfallzeiten, als auch zu erhöhten Materialkosten führen. Speziell beim Test von autonomen Verhalten ist der Test der Features in realer Umgebung somit mit hohem Risiko verbunden. Um dies zu vermeiden ist die Simulation von Quadrocoptern und verschiedener Umgebungen unabdingbar. Ebenfalls erleichtert eine simulierte Version der Drohne die Entwicklung, da sie nicht immer physikalisch vorhanden sein muss. Dabei ist es allerdings notwendig, insbesondere bei Bilddaten, dass die reale Situation möglichst realitätsnah abgebildet wird, sodass das Verhalten im realen Umfeld entsprechend ähnlich ist.
Durch die Integration von ROS kommt die Simulationsumgebung Gazebo als Bestandteil mit. Anfänglich handelte es sich um ein ROS Paket, mittlerweile ist es allerdings ein eigenständiges Ubuntupaket und benötigt de facto kein ROS.

2.4 Fuzzylogik

2.5 Kinect

Um Gesten des Benutzers zur erkennen und entsprechend auszuwerten wird der visuelle Sensor Microsoft Kinect verwendet. Normalerweise wird er zur Steuerung der Videospiel Konsole Xbox360/Xbox One verwendet. Das System ist in der Lage Tiefenbilder zu erstellen, besitzt sowohl eine 1080p Farbkamera, als auch einen Infrarotsensor und mithilfe mehrere Mikrofone Sprache und Bewegung im Raum zu erkennen.

3 Software Architektur

3.1 Anforderungen

Im Folgenden sollen die Anforderungen an die Studienarbeit festgelegt werden.
Ein Hauptbestandteil der Arbeit besteht darin, ein Vorgängerprojekt mit einer AR.Drone 2.0 in das ROS Framework zu portieren. Dies beinhaltet eine Modularisierung der Projektbestandteile in ROS Nodes. Ziel soll sein, den Quadrocopter mit Hilfe einer Kinect und Gestenerkennung steuern zu können. Anzumerken ist, dass ROS nur unter UNIX basierten Betriebssystemen läuft und das bestehende Projekt Libraries verwendet, welche nur unter Windows verfügbar sind.
Eine weitere Vorgabe besteht darin, dass man neben der realen Drohne jegliche Funktionalität auch in einer Simulation laufen soll.
Somit kann man für Präsentationen, in denen der Flug einer Drohne nicht möglich ist, den Quadrocopter in einer frei gestaltbaren simulierten Umgebung fliegen lassen.
Ein weiterer Bestandteil dieser Arbeit besteht darin, Ansätze und Limitationen der Implementierung eines Assistenzsystems zu testen und zu bewerten. Dafür soll ein externes Projekt, zur Gewinnung von Tiefenbildern aus einer monokularen Kamera, in die Projektumgebung integriert werden. Hierbei soll es wiederum möglich sein, dass die Videostream sowohl von der realen, als auch von der simulierten Drohne gesendet werden kann.
Es soll dabei ermittelt werden, ob die Nutzung der Software für den Anwendungs-

zweck praktikabel und sinnvoll ist.

3.2 Überblick

3.3 Bildverarbeitung

In diesem Abschnitt wird beschrieben, wie Drohne aus den Bildern der Kamera Informationen gewinnen kann, die später für die Betrachtung des Assistenzsystems relevant sind.

3.3.1 Semi-Direct Monocular Visual Odometry - SVO

Eine Grundanforderung an das Projekt ist die Nutzung einer nicht modifizierten AR.Drone. Dadurch entsteht die Problematik, dass keine Tiefenbildkamera genutzt werden kann, um in Echtzeit Tiefenbilder zu erhalten. Die Drohne ist lediglich mit einer monokularen [1] Frontkamera ausgestattet.
Um Tiefeninformationen aus den Bildern einer solchen Kamera zu gewinnen, wird eine Szene aus verschiedenen Perspektiven aufgenommen. Anschließend gibt es unterschiedliche Ansätze um aus den aufeinanderfolgenden Bildern Kamerapositionen und Umgebungsstrukturen zu ermitteln.
Feature basierte Ansätze sind der aktuelle Standard zur Berechnung der Kamerapostion. Diese versuchen die wichtigsten Merkmale eines Bildes, die Features, zu extrahieren. Mit Hilfe von Feature Deskriptor Algorithmen werden Vektoren mit Informationen zu invarianten Bildbereichen berechnet. Diese Vektoren verhalten sich wie ein einzigartiger Fingerabdruck, der die Merkmale repräsentiert.
Aufeinanderfolgende Bilder werden dann mit Hilfe dieser Deskriptoren abgeglichen und sowohl Kamerabewegungen, als auch Strukturen werden rekonstruiert. Zur Op-

[1] Monokular ist die Bezeichnung für Kameras mit einer einzelnen Linse

timierung sind abschließend die ermittelten Kamerapositionen anzugleichen. Dies geschieht mit Hilfe von Algorithmen zur Minimierung des Reprojektionsfehlers. [2]
Ein weiterer Ansatz ist die direkte Methode. Hierbei werden die Features nicht über Deskriptor Algorithmen bestimmt, sondern das Problem wird über die Intensitäten der Pixel gelöst. Bei einem Graustufenbild entspricht diese Intensität der Helligkeit von Bildbereichen.
Somit kann bei der Rekonstruktion im Gegensatz zum Feature basierten Ansatz auch die Richtung der Gradienten von Intensitäten genutzt werden. Dadurch funktioniert diese Methode auch bei Bildern mit sehr wenig Textur, Bewegungsunschärfe und fehlerhaftem Kamerafokus.
Das für diese Arbeit relevante Vorgehen kombiniert die Vorteile der beschriebenen Methoden. Die semi-direkte Odometrie verwendet einen Algorithmus der ebenfalls auf Zusammenhängen von Features basiert. Diese werden jedoch implizit aus einer direkten Bewegungsabschätzung bezogen, anstatt explizit durch Algorithmen mit Feature Deskriptoren berechnet zu werden.
Dadurch müssen Features nur extrahiert werden, wenn diese noch nicht auf einem der vorherigen Bildern vorhanden waren. Insgesamt ist dieser Ansatz somit schnell, da wenig Berechnungen pro Bild stattfinden und auf Grund der Verwendung von Intensitätsgradienten äußerst genau und robust.
Diese Eigenschaften sind für die Anforderungen der Studienarbeit essentiell, da die Drohne sich sehr schnell bewegen kann und somit in möglichst kurzer Zeit neue Bilder auswerten muss. Das beschrieben Verfahren minimiert damit die Auswirkungen der typischen Probleme von Drohnen. Diese sind einerseits die niedrige Texturierung der Umgebung, welche hauptsächlich in Innenräumen auftritt und andererseits kameraspezifische Probleme wie Bewegungsunschärfe und der Verlust des Kamerafokus.

[2] Reprojektionsfehler sind geometrische Fehler die im Zusammenhang zwischen abgebildeten und berechneten Bildpunkten entstehen.

3.3.2 Kamerakalibrierung

Ein Grundproblem der Bildverarbeitung ist die Verzerrung des Bildes. Da die Bestimmung der Kameraposition möglichst genau sein soll, muss die Kamera vorher kalibriert werden. Dies bedeutet, dass Ungenauigkeiten der Linse erkannt und softwareseitig ausgeglichen werden. Dafür werden die intrinsischen Parameter der Kamera bestimmt

Kalibrierungsmodelle

Hierbei unterstützt SVO drei Kamera Modelle: ATAN, Ocam und Lochkamera. [3]
Das ATAN Modell basiert auf dem *Field of View* (FOV) Verzerrungsmodell SStraight lines have to be straight [...]"von Devernay und Faugeras.
Der Vorteil dieser Kalibrierungsmethode ist die äußerst schnelle Berechnung der Projektion des Bildes. Das Modell vernachlässigt jedoch tangentiale Verzeichnung, welche auftritt, wenn optische und mechanische Bestandteile des Objektives, sowie der CCD-Sensor [3] nicht perfekt zueinander ausgerichtet sind. Weiterhin sollte die Kamera mit einem globalem Shutter ausgestattet sein, um die Extraktion von Bildmerkmalen bei Bewegungen zu gewährleisten. Kameras mit Global-Shutter-CMOS [4] Sensoren und CCD-Sensoren nehmen Bild nicht zeilen- und spaltenweise, sondern vollständig auf und sind daher für das Verfahren geeignet.
Die Drohne besitzt eine veraltete und günstige Kamera mit einem CMOS Sensor, wodurch sowohl tangentiale Verzerrung, als auch der Rolling-Shutter-Effekt auftreten können.
Daher ist das ATAN Modell zwar eine der besten Kalibrierungsmethoden für teure

[3] CCD steht für *charge-coupled device*, was übersetzt ladungsgekoppeltes Bauteil bedeutet. Dieses lichtempfindliche elektronische Bauteil wird zur Bildaufnahme verwendet.
[4] CMOS steht für *Complementary metal-oxide-semiconductor* und ist ein spezieller Halbleiter der zur Bildaufnahme verwendet wird.

Hochleistungskameras, jedoch ist es für die Betrachtungen dieser Arbeit nicht optimal.
Der zweite Ansatz zur Kalibrierung ist das Ocam Modell von Davide Scaramuzza. Diese Methode sollte für Kameras mit sehr weitem Sichtfeld, oder omnidirektionalen Kameras genutzt werden. Damit ist es für die Drohne nicht geeignet.
Die dritte unterstützte Kalibrierungsmethode ist das Modell der Lochkamera. Hierbei handelt es sich um den aktuellen Standard in OpenCV [5] und ROS. Hierbei wird die Verzerrung mit Hilfe von fünf intrinsischen Parametern beschrieben, welche im Rahmen der Kalibrierung bestimmt werden müssen.

$$\mathrm{Distortion}_{\mathrm{coefficients}} = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

OpenCV betrachtet dabei radiale und tangentiale Faktoren. Die Formel für radiale Verzeichnung ist die Folgende:

$$x_{\mathrm{corrected}} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{\mathrm{corrected}} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Hierbei wird aus einem alten Bildpunkt (x, y) des Eingabebildes die korrigierte Position $x_{corrected} y_{corrected}$ bestimmt.
Die Berechnung der tangentialen Verzerrung erfolgt durch die Formel:

$$x_{\mathrm{corrected}} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$
$$y_{\mathrm{corrected}} = y + [p_1(r^2 + 2y^2) + 2p_2 xy]$$

Abschließend werden die Einheiten angepasst:

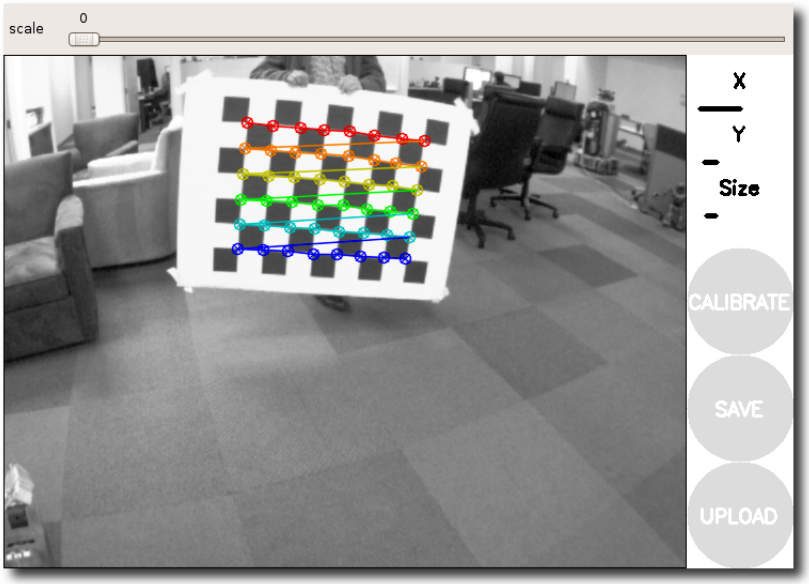
Dabei entspricht f_x und f_y der Brennweite der Linse und c_x, sowie c_y beschreiben die optische Bildmitte in Pixelkoordinaten.
Dieser Ansatz ist der einfachste und funktioniert grundsätzlich mit jeder Kamera.

[5] "OpenCV is the leading open source library for computer vision, image processing and machine learning, and now features GPU acceleration for real-time operation."

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

Umsetzung der Kalibrierung

Um die intrinsischen Parameter der Kamera zu bestimmen, wird ein bekanntes Bild oder Muster aufgenommen. Dazu wird ein Vergleich zwischen den theoretischen und tatsächlichen Abmessungen angestellt. Hierzu wird meist ein einfaches Schachbrettmuster genutzt, welches im möglichst vielen verschiedenen Perspektiven aufgenommen wird. Bei der realen Drohne wird dazu das Muster ausgedruckt, bei der Simulation muss hingegen ein muss ein solches Objekt in die Welt eingefügt werden. Da die Kamera in der Simulation ohnehin keine intrinsischen Fehler aufweisen sollte, kann auf eine Kalibrierung verzichtet werden.



Die Kalibrierung wurde mit dem frei verfügbaren ROS Node *camera_calibration* umgesetzt. Das Ergebnis ist abhängig von der Anzahl der Perspektiven und der

Qualität der Aufnahmen. Aufgegeben wird dann die List der Parameter, die für das Lochkamera Modell notwendig sind.

3.3.3 Regularized Monocular Depth Estimation - REMODE

3.4 Bildverarbeitung

3.4.1 Featureerkennung

3.4.2 Kinect

3.4.3 Die einzelnen ROS Nodes

3.4.4 Architektur

4 Evaluation

4.1 Ergebnis

4.2 Ausblick

Literaturverzeichnis

[1] "Parrot AR.Drone." https://de.wikipedia.org/wiki/Parrot_AR.Drone, 2016. visited: 01.22.2017.

[2] "Realtime code in ROS." http://www.willowgarage.com/blog/2009/06/10/orocos-rtt-and-ros-integrated, 2016. visited: 01.22.2017.

[3] Robotics and Perception Group, "Camera Calibration." https://github.com/uzh-rpg/rpg_svo/wiki/Camera-Calibration, 2014. visited: 02.19.2017.