Masterarbeit

Analyse von ORB SLAM 2 im verkehrsähnlichen Kontext

Verfasser:
Jan Bernoth

Betreuer:
Prof. Dr. Daniel Göhring

03.11.2016

Freie Universität Berlin
Mobile Robotics and Autonomous Vehicles

Kurzfassung

In dieser Arbeit wurde untersucht, inwiefern die montierten Monokameras in dem autonom fahrenden Auto von AutoNOMOS Labs Tiefeninformationen extrahieren können. Die Tiefeninformationen können durch den Umstand erzeugt werden, dass mehrere Bilder aus verschiedenen Perspektiven von statischen Elementen einer Szene während einer Autofahrt erzeugt werden und dadurch sowohl die relative Pose der Kamerabilder zueinander bestimmt werden kann, als auch die Position der erkannten Merkmale im Raum projiziert werden.

Bislang wird häufig eine Radar Technologie verwendet, um eine digitale räumliche Darstellung der Umgebung zu erzeugen. Im Gegensatz zu dieser Technologie ist die handelsübliche, hochauflösende Kamera wesentlich günstiger und kann mit den Prinzipien von Structure from Motion mitunter ähnliche Informationen extrahieren.

Zu diesem Zweck wurde ORB SLAM 2 in das Projekt integriert und auf Benutzbarkeit und Effektivität überprüft. ORB SLAM 2 wurde 2015 veröffentlicht und ist als Open Source Projekt verfügbar. Mittels Bundle Adjustment kann dieser Algorithmus erkannte ORB Merkmale in einem eigens abgewandelten SLAM Algorithmus einbeziehen und in einer lokalen Karte darstellen. In dieser Arbeit wurde vor allem untersucht, wie konkret ORB SLAM 2 eingebunden werden kann und wie der Algorithmus vorrangig die Informationen der eingebauten Weitwinkelkameras wirksam verarbeitet.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Autonomes Fahren ist eine Disziplin der Robotik, die viele Herausforderungen bereit hält. Dabei ist nicht nur die Planung der Fahrtwege nötig, sondern beispielsweise auch die Positionierung in der Welt, das Erkennen von Hindernissen, Wechselwirkungen mit dem Umfeld und die Interaktion mit den Mitfahrern. Diese Arbeit beschäftigt sich mit der relativen visuellen Positionierung des Autos in Bezug auf umliegende Merkmalspunkte.

Für die umgebungsspezifische Lokalisierung und Erkennung der Welt wird häufig eine Art Radar verwendet, das die Umgebung wahrnimmt und mit Hilfe einer Punktwolke digital abbildet, wobei jeder Punkt der Wolke eine Abstandsmessung zwischen Radar und wahrgenommenem Gegenstand ist. Auf dieser Punktwolke kann spezifische Software Objekte erkennen und diese gegebenfalls auch verfolgen und klassifizieren. In der Regel wurden die Kameras für diese Aufgaben außen vor gelassen, da visuelle Verfahren im Vergleich zu der Radartechnologie nicht effizient genug eingesetzt werden konnten.

Die Hardware von Kameras und Computern entwicklelt sich stetig weiter und somit ist es dem Normalverbraucher schon möglich, hochauflösende Bilder aufzunehmen und der Welt zur Verfügung zu stellen. Das animierte mehrere Forscherteams dazu, die große Bandbreite an Bildern von beudeutenden Sehenswürdigkeiten, zum Beispiel aus Rom, dazu zu verwenden, um die Umgebung digital zu Rekonstruieren [Agarwal et al., 2009]. Diese Rekonstruktion von Szenen mittels verschiedener Bilder aus unterschiedlichen Perspektiven wird unter den Sammelbegriff *Structure from Motion*, also Strukturen durch Bewegung, zusammengefasst.

Nahezu unmöglich ist es, einen Urheber für das Thema Structure from Motion auszumachen, weil die mathematischen Grundkonzepte im Wesentlichen schon im 19 Jahrhundert entwickelt wurden [Konderlink and van Doorn, 1991], die Namensprägung jedoch erst Ende des 20. Jahrhunderts stattfand. Im 21. Jahrhundert hat der Reifegrad der Hardware und Software einen Punkt erreicht, in dem Structure from Motion effizient eingesetzt werden kann. Dementsprechend werden, neben der schon erwähnten Rekonstruktion Roms, auch viele andere Projekte besonders im urbanen Umfeld durchgeführt, die auf der Structure from Motion Technik basieren. Zum Beispiel die Rekonstruktion urbaner Fassaden durch die Auswertung mehrerer Bilder, die entlang einer Straßenfahrt aufgenommen wurden [Xiao et al., 2008] oder die urbane Rekonstruktion in Echtzeit an Hand eines aufgenommenen Videos [Pollefeys et al., 2008].

Die Insipiration für diese Arbeit ist ein Projekt, das es zum Ziel hat, mittels Structure from Motion in einem Parkhaus oder auf einem Parkplatz automatisch nach freien Parkplätzen zu suchen und das Auto dann hinein manövrieren soll. [Hane et al., 2015] Das Projekt "V-Charge" ist nicht nur ein elektronisch fahrendes Auto, sondern verlässt sich vollkommen auf vier

montierte Kameras, welche die nächste Umgebung des Autos im Rundumblick beobachten und analysieren. Es wird dabei lediglich mit Computer Vision gearbeitet, ohne Absicherungen oder Anreicherungen durch andere Sensoren.

Um diesen Ansatz weiter zu verfolgen, wurde im Rahmen dieser Arbeit untersucht, inwiefern sich eine Structure from Motion Technik im Projekt AutoNOMOS Labs integrieren lässt. Dabei besteht der Anspruch darin, dass die Technik während einer Fahrt benutztbar ist, das heißt eine effiziente Lösung für die Benutzung in Echtzeit geschaffen wird und eine fortlaufende Benutzung möglich ist. Für diese Anforderungen ist ORB SLAM 2 [Mur-Artal et al., 2015] ein gute Option. Dieser Algorithmus kann durch ein Video Merkmale mit Oriented FAST and Rotated BRIEF (ORB) extrahieren, welche durch den iterativ fortlaufenden Prozess relativ zu den Kamerapositionen im Raum dargestellt werden können. Zusätzlich ist der Algorithmus freiverfügbar. [1] Im wissenschaftlichen Beitrag und auch in Präsentationen wird gezeigt, wie ORB SLAM 2 aus einem Video eine Punktwolke generiert. Dieser Anwendungsfall könnte für das AutoNOMOS Projekt dazu verwendet werden, um zum Beispiel freie Parkplätze zu finden und daraufhin ein Parkmanöver einzuleiten.

Anhand dieser Zielstellung gliedert sich die folgende Arbeit: Im Kapitel 2 werden die Grundlagen zur relativen Ausrichtung von Kameras an Hand von visuellen Kernmerkmalen dargelegt. Ein hervorzuhebendes Konzept stellt *Bundle Adjustment* im Abschnitt 2.4 dar, welches die Basis aller Structure from Motion Algorithmen bildet. Im darauffolgenden Kapitel 3 werden detailliert ORB SLAM 2 Konzepte vorgestellt und die Funktionsweise des Algorithmus näher erläutert. Die praktische Arbeit mit ORB SLAM 2 im Projekt AutoNOMOS Labs ist im Kapitel 4 aufgeführt. In diesem Kapitel wird im Abschnitt 4.1 die Erstanbindung von ORB SLAM 2 an die vorhandenen Strukturen im AutoNOMOS Labs Projekt dargestellt, danach werden im Abschnitt 4.2 die programmatischen Vorteile eines eingeschränkten Sichtfelds erläutert. Im Anschluss daran wird im Abschnitt 4.3 näher untersucht, warum es Komplikationen mit der Fischaugenlinse gab. Abschließend wird im Abschnitt 4.4 gezeigt, wie das Schließen eines gefahrenen Kreises praktisch funktioniert. Im darauffolgenden Kapitel 5 werden Möglichkeiten zu Verbesserungen des Algorithmus vorgestellt, die besonders im Kontext eines autonom fahrenden Autos wichtig sind. Kapitel 6 beinhaltet schließlich eine Zusammenfassung und ein abschließendes Fazit.

[1] `https://github.com/raulmur/ORB_SLAM2`

2 Grundlagen der visuellen Rekonstruktion

Dieses Kapitel beschäftigt sich mit den Grundlagen der maschinellen Rekonstruktion einer Szene mittels verschiedener Perspektiven und korrespondierender Merkmalenspunkte. Das sind Punkte, die in mehreren Ansichten gleich sind. Aufbauend auf der Grundlagen zur Kameramatrix aus Abschnitt 2.1, wird im Abschnitt 2.2 die Epipolargeometrie thematisch eingeführt, die sich ausschleißelich mit einem Szenario aus zwei Sichten beschäftigt. Erweiternd dazu wird im Abschnitt 2.4 Bundle Adjustment vorgestellt, welches die Geometrie n-viele Perspektiven ausbaut, die iterativ optimiert werden. Der letzte Abschnit dieses Kapitels behandelt das Triangulieren zwischen n-vielen Punkten, was durch die verschiedenen Blickpunkte und die nicht perfekte Rekonstruktion der korrespondierenden Punkte im Raum eine einfache Lösung bietet, um aus den verschiedenen Vorschlägen einen finalen Punkt im Raum zu errechnen.

2.1 Kameramatrix und Verzerrungskoeffizient

Um Punkte der Umgebung abzubilden, bedarf es der Projektion von 3D Punkten auf eine 2D Fläche der Kamera. Es gibt eine Vielzahl von benutzbaren Kameras, wovon die einfachste Variante die Lochkamera ist, in der sich keine Linse befindet und die Abbildung der Merkmale auf der Bildfläche sich mittels Strahlensatz der Optik analysieren lassen. Ist hingegen eine Linse vorhanden, wie es heutzutage in handelsüblichen Kameras der Fall ist, muss die Linse als Teil des Projektionsprozesses mit betrachtet werden. Da die Linse nie parallele zur Projektionsfläche ausgerichtet ist, wird das Bild radial verzerrt. Somit muss die Linse differenziert in die Betrachtung der Bilder einbezogen werden, was im folgenden näher erläutert wird.

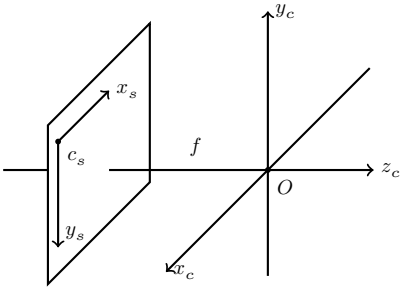


Abbildung 1: Schematische Darstellung der intrinsischen Parameter einer Kamera. Dabei ist f die Fokallänge und O der Kameraursprung. Beim Punkt c_s wird das Koordinatensystem der Sensorenfläche aufgespannt, die durchaus zum Kameraursprung verschoben sein kann.[Szeliski, 2010, S. 50]

Um ein Bild im Kontext der Kamera auszuwerten, werden intrinsische

Werte bei der rechnerischen Abbildung jedes Pixels berücksichtigt. Dazu werden die Pixel im Bild (x_s, y_s) mit Werten der Pixelräume skaliert (s_x, s_y). Diese Punkte werden abhängig von der Rotation R_s und der Translation c_s der Kamera in Bezug auf dem Kameraursprung (O) aus der 2D Ebene in den 3D Raum projiziert:

$$p = [R_s \mid c_s] \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_s \\ y_s \\ 1 \end{pmatrix} = M_s \overline{x}_s. \tag{1}$$

In Abbildung 1 wird diese Rechnung schematisch skizziert. Wie aus der Formel ersichtlich, sind $\overline{x}_s$ die Koordinaten der 2D Abbildung. Im Kameramodell M lassen sich demnach kamerazentrierte Punkte p_c mittels Skalierungsfaktoren s auf 3D Pixel p abbilden: $p = sp_c$. Die komplette Abbildung der homogenisierten Version des Pixels $\tilde{x}_s$ ist folglich:

$$\tilde{x}_s = \alpha M^{-1} p_c = K p_c. \tag{2}$$

K wird im Allgemeinen als Kalibrierungsmatrix bezeichnet, da diese alle intrinsischen Werte, die spezifisch auf die Kamera bezogen sind, vereint. Diese intrinsischen Werten müssen mit Hilfe einer Kalibrierung ermittelt werden. Wenn diese Parameter mit der extrinsischen Rotation R und Translation t kombiniert werden, kann eine Beziehung zwischen den Bildpunkten im 2D Raum $\tilde{x}_s$ und den 3D Weltkoordinaten p_w hergestellt werden:

$$\tilde{x}_s = K[R \mid t] p_w = P p_w. \tag{3}$$

Ohne Koordinaten ergibt sich folgender Zusammenhang:

$$P = K[R \mid t]. \tag{4}$$

P wird allgemein als Kameramatrix bezeichnet. Diese Herleitung folgt den Ausführungen von Szeliski [Szeliski, 2010, S. 50ff.].

Für die Kalibrierungsmatrix K gibt es mehrere Möglichkeiten der Formulierung, zum Beispiel:

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{5}$$

benutzt die Fokallänge in voneinander unabhängigen Variablen f_x und f_y und die vermeintliche Schräglage s zur Achse der Kamera. Die Werte c_x, c_y sind die Verschiebungen zum optischen Zentrum. Eine andere Möglichkeit ist:

$$K = \begin{pmatrix} f & s & c_x \\ 0 & af & c_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{6}$$

Dabei ist die Veränderung darin zu sehen, dass der Unterschied zwischen der Fokallänge in x und y Richtung lediglich durch einen Multiplikator a dargestellt wird. Wenn hingegen die Fokallängen gleich sind und keine Schräglagen der Achsen erkennbar sind, sieht die Kalibrierungsmatrix wie folgt aus:

$$K = \begin{pmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{7}$$

Bei dieser Projektion wird im Allgemeinen davon ausgegangen, dass ein lineares Modell zugrunde liegt, also die Bildebene nicht verzerrt ist. Dies ist, wie schon oben erwähnt, bei Kameras mit Linsen nicht ausschließlich anwendbar, da hierbei eine radiale Verzerrung des Bildes entsteht. Das unverzerrte Bild mit den Pixeln x_d, y_d wird folgendermaßen berechnet:

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = (1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6) \begin{pmatrix} x \\ y \end{pmatrix}, \tag{8}$$

wobei κ_1, κ_2 und κ_3 die radialen Verzerrungskoeffizienten sind und es gilt: $r^2 = x^2 + y^2$. Zusätzlich zu der radialen Verzerrung kann die Linse tangential das Bild verzerren, zum Beispiel durch eine Weitwinkelkamera mit einem Fischaugenobjektiv. In modernen Frameworks, wie zum Beispiel OpenCV, wird die tangentiale Verzerrung folgendermaßen mit einbezogen[2]:

$$x_d = x + [2p_1xy + p_2(r^2 + 2x^2)] \tag{9}$$
$$y_d = y + [p_1(r^2 + 2y^2) + 2p_2xy]. \tag{10}$$

Zusammengefasst ist zusätzlich neben der Fokallänge f und der Verschiebung des Kameraursprung zum Bildzentrum c_x, c_y zu beachten, dass die Linse durch ihre Positionierung zum Sensor und eventueller tangentialer Verzerrung noch mit weiteren 5 Verzerrungparametern $\kappa_1, \kappa_2, \kappa_3, p_1, p_2$ kalibriert werden muss.

2.2 Epipolargeometrie

Die Epipolargeometrie ist die Grundlage zur Berechnung der Entfernung eines bestimmten Punktes im Bezug zum Standpunkt der Kamera. Um die

[2] `http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html`, Stand: 21.10.2016.

Entfernung berechnen zu können, werden zwei Blickpunkte aus unterschiedlichen Perspektiven auf einen wiedererkennbaren Fixpunkt benötigt. Schematisch wird dies in Abbildung 2 dargestellt.

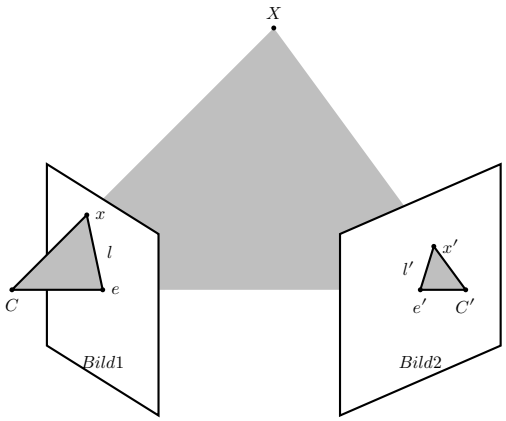


Abbildung 2: Darstellung der Berechnung der Tiefe vom Punkt X. Auf Bild 1 wird X auf x abgebildet und auf Bild 2 auf x'. Die Basisline der Kameras führt vom Blickpunkt C zu C' und der Schnittpunkt ist auf dem jeweiligen Bild mit e und e' gekennzeichnet. l und l' sind die Epipolarlinien, die durch x, x' und e, e' definiert werden, angelehnt an [Hartley and Zisserman, 2003, S. 240].

Dabei ist der Fixpunkt X im Raum und spannt mit den Kamerastandpunkten C und C' die epipolare Fläche auf. Der Schnittpunkt x oder x' wird von der Ebene (Bild1 oder Bild2) und der Geraden $\overline{CX}$ oder $\overline{C'X}$ definiert. Weiterhin ist l und l' zwischen x, x' und e, e' aufgespannt. Zusammengefasst ist die Abhängigkeit

$$l' = e' \times x' = [e']_\times x' \tag{11}$$

gegeben. Durch das Gleichnis $a \times b = [a]_\times b$ kann die schiefsymmetrische Matrix $[a]_\times = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}$ zu $a \times b = \begin{pmatrix} a_2b_3 - a_3b_2 \\ a_3b_1 - a_1b_3 \\ a_1b_2 - a_2b_1 \end{pmatrix} = [a]_\times b$ vereinfacht werden.

Von x auf x' wird mittels Homographie über die Fläche π abgebildet: $x' = H_\pi x$, siehe Abbildung 3. Durch Einsetzen dieser Homographie in die Formel 11 entsteht:

$$l' = [e']_\times H_\pi x = Fx, \tag{12}$$

wobei F die fundamentale Matrix ist. Die fundamentale Matrix entspricht der algebraischen Repräsentation der Epipolargeometrie und ist vom Rang 2:

$$F = [e]_\times H_\pi. \tag{13}$$

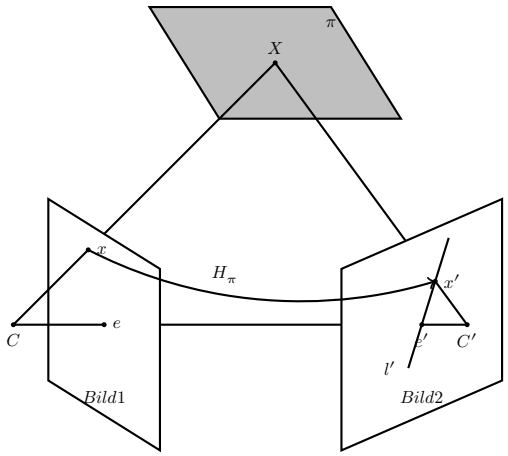


Abbildung 3: Darstellung der Homographie, angelehnt an [Hartley and Zisserman, 2003, S. 243]

Um die fundamentale Matrix durch die Projektionsmatrizen P, P' darzustellen, wird eine algebraische Ableitung nach dem Ansatz von Xu und Zhang [Xu and Zhang, 1996] durchgeführt. Mittels der Projektionsmatrix können Punkte im Raum X auf Punkte auf der Bildfläche der Kamera abgebildet werden: $PX = x$. Die dazugehörige Gleichung wird folgendermaßen formuliert:

$$X(\lambda) = P^+x + \lambda C, \tag{14}$$

wobei P^+ die Pseudoinverse ist, welche ein Ersatz für die Inverse der Matrix ist, da diese bei Ungleichheit der Reihen und Spalten der Matrix nicht vollständig definiert ist. Die Funktion wird durch ein Skalar λ parametrisiert. Dieses Skalar hat zwei fest definierte Punkte: Zum einen P^+x bei $\lambda = 0$ und das erste Kamerazentrum C bei $\lambda = \infty$.[3] Als Ergebnis der Herleitung wird die Definition der Pseudoinversen aufgestellt:

$$P^+ = P^T(PP^T)^{-1}. \tag{15}$$

Dabei ist die Projektionsmatrix so definiert, dass stets gilt $PP^+ = I$. Wenn die Definition der Pseudoinversen der Projektionsmatrix in die Formel 14 eingesetzt wird, ergibt sich daraus die Eigenschaft, dass $PC = 0$ ist. Aus der Grenzwertüberlegung von λ resultiert eine Umformung der Definition von l': $l' = (P'C) \times (P'P^+x)$. In diesem Zusammenhang ist $P'C$ die Epipolarlinie und wird als e' bezeichnet. Somit ergibt sich $l' = [e']_\times(P'P^+)x = Fx$, wobei F folgendermaßen definiert ist:

$$F = [e']_\times P'P^+. \tag{16}$$

[3]Eine ausführliche Herleitung ist beispielsweise bei G. Xu und Z. Zhang[Xu and Zhang, 1996, S. 75] ausgeführt.

Somit kann die vorher herangezogene Homographie durch $H_\pi = P'P^+$ beschrieben werden. Diese Herleitung beruht auf der Annahme, dass beide Kamerazentren C und C' ungleich sind, ansonsten wäre F eine Nullmatrix. Die ganze Herleitung ist angelehnt an "Multiple View Geometry" [Hartley and Zisserman, 2003, S. 259ff.]. Zusammengefasst bedeutet dies, dass mit der vorgestellten Methodik ein prinzipieller Zusammenhang zwischen zwei Kameras und Punkten im Raum hergestellt wird. Hingegen kann mit diesen Mitteln noch keine Rekonstruktion von Tiefeninformationen ermöglicht werden. Die aufbauenden Schritte dazu werden im folgenden Abschnitt beschrieben.

2.3 Tiefenrekonstruktion

Eine Methode um die Tiefeninformationen zu rekonstruieren ist nach [Hartley and Zisserman, 2003, S. 262] in drei Punkte unterteilt:

1. Berechne die Fundamentale Matrix an Hand von Korrespondenzpunkten.
2. Berechne die Kameramatrix mittels der Fundamentalmatrix.
3. Jeder der Korrespondenzpunkt $x_i \leftrightarrow x_i'$ wird in den Raum projiziert.

Berechnen der Fundamentalmatrix
Die Basis der Berechnung der Fundamentalmatrix ist die Gleichung:

$$x^{'T} F x = 0 \tag{17}$$

für alle übereinstimmenden Paare $x \leftrightarrow x'$. Insgesamt müssen mindestens sieben übereinstimmende Punkte ausgewählt werden, um die Formel vollständig lösen zu können. Wenn die Punkte $x = (x, y, 1)^T$ und $x' = (x', y', 1)^T$ in die Formel 17 eingesetzt werden, ist das Ergebnis:

$$(x'x, x'y, x', y'x, y'y, y', x, y, 1)f = 0 \tag{18}$$

Da mehrere Punkte benötigt werden, um F zu lösen, wird eine Matrix A erstellt, die mit f multipliziert wird:

$$Af = \begin{pmatrix} x_1'x_1 & x_1'y_1 & x_1' & y_1'x_1 & y_1'y_1 & y_1' & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ x_n'x_n & x_n'y_n & x_n' & y_n'x_n & y_n'y_n & y_n' & x_n & y_n & 1 \end{pmatrix} f = 0 \tag{19}$$

In Hinblick auf die Matrix A ist festzustellen, dass diese mindestens den Rang 8 haben muss, damit alle Unbekannten in der Gleichung determinierbar sind. Beim genauen Rang 8 ist die Matrix mittels einer linearen Methode lösbar. Falls die Matrix im Rang 9 ist, muss die Lösung mittels einer "Least Squares Methode" ermittelt werden. Die Lösung für f wäre eine

Singulärwertzerlegung (SVD) von A durchzuführen und davon den kleinen singulären Wert zu nehmen, also die letzte Spalte von V in $A = UDV^T$. Das ist die Essenz vom *8 Punkte Algorithmus*.

Der normalisierte 8 Punkte Plan ist die einfachste Möglichkeit die fundamentale Matrix zu berechnen [Hartley and Zisserman, 2003], weil lediglich die Konstruktion und Lösung eines linearen Gleichungssystems (mit der "Least Squares" Methode) durchgeführt werden muss. Der ursprüngliche Algorithmus ist auf [Longuet-Higgins, 1987] zurückzuführen, wobei die Idee der fundamentalen Matrix nicht impliziert wurde. Angenommen, es existieren mindestens 8 Korrespondenzpunkte, dann müssen nach [Hartley and Zisserman, 2003, S. 282] folgende Schritte durchgeführt werden:

- **Normalisieren:** Transformiere die Bildkoordinaten mittels $\hat{x}_i = Tx_i$, sowie dessen Korrespondenzpunkt $\hat{x'}_i = T'_i x'_i$, wobei T, T' die normalisierten Transformationen sind, die durch eine Translation und Skalierung definiert werden. Die Normalisierung ist wichtig, damit der geometrische Schwerpunkt der Referenzpunkte in der Mitte der Koordinaten ist.

- Finde die Fundamentalmatrix $\hat{F}'$:

  - Finde $\hat{F}$ durch die Berechnung des kleinsten singulären Werts von $\hat{A}$ in der Gleichung 19.
  - Ersetze $\hat{F}$ durch $\hat{F}'$, damit $det\hat{F}' = 0$ gilt. Dazu muss, wie im vorherigen Abschnitt beschrieben, das SVD gebildet werden.

- **Denormalisieren:** Löse $F = T'^T \hat{F}' T$, damit die Matrix F als Fundamentale Matrix für die Korrespondenzpunkte herangezogen werden kann.

Ein weiterer wichtiger Unterschied zwischen dem ursprünglichen Algorithmus von Longuet-Higgins und dem oben vorgestellten Algorithmus von Hartley und Zisserman ist, dass der Normalisierungs- und Denormalisierungsprozess ursprünglich nicht vorgesehen war. In einem wissenschaftlichen Beitrag von [Hartley, 1997] wurde festgestellt, dass der 8 Punkte Algorithmus nicht ohne Einschränkungen anwendbar ist und dass es Fälle gibt in denen dieser nicht funktioniert. Deswegen wurde an iterativen Verfahren geforscht, welche besondere Schwachstellen des Algorithmus verbessern sollen:

1. Die Korrespondenzpunkte müssen verschoben werden, so dass dessen geometrischer Schwerpunkt im Ursprung des Bildes ist. Anschließend gilt: $\sum_i \tilde{x}_i = \sum_i \tilde{y}_i = 0$.

2. Die isotrope Skalierung der Korrespondenzpunkte muss hergestellt werden damit alle Punkte eine gleichen Abstand von $\sqrt{2}$ zum Mittelpunkt haben. Dabei ist die rechnerisch bessere Variante $\sum_i \tilde{x}_i^2 +$

$\sum_i \tilde{y}_i^2 = 2n$, weil die Berechnung von Quadraten effizienter ist als die Berechnung einer Quadratwurzel.[Szeliski, 2010]

Für beide Punkte wird folgende Definition vorausgesetzt:

$$\tilde{x}_i = s(x_i - \mu_x) \text{ und } \tilde{y}_i = s(y_i - \mu_y) \tag{20}$$

Diese Schritte werden unabhängig voneinander mit jedem Bild durchgeführt, um mit den normalisierten Koordinaten die Fundamentalmatrix bestimmen zu können und nach dem Finden wieder für weitere Bearbeitungsschritte die denormalisierten Koordinaten nutzen zu können.

Berechne die Kameramatrix mittels der Fundamentalmatrix
Um den Zusammenhang zwischen essentieller Matrix und Fundamentalmatrix explizit erläutern zu können, müssen nach [Hartley and Zisserman, 2003, S. 257] folgende Annahmen getroffen werden. Sei $P = K[R \mid t]$ die Kameramatrix, die aus der Kalibrierungsmatrix K, der Rotationsmatrix R und des Translationsvektors t besteht, dann gilt für jeden Punkt im Bild: $x = PX$, wie in den vorangegangenen Abschnitten schon festgestellt wurde. Wenn die Kalibrierung K bekannt ist, dann ist Inverse vom Punkt x ein neuer Punkt $\hat{x} = K^{-1}x$. Dann gilt für $\hat{x}$ in normalisierten Koordinaten: $\hat{x} = [R \mid t]X$. Folglich wird die Kameramatrix $K^{-1}P = [R \mid t]$ als normalisierte Kameramatrix bezeichnet.

Die essentielle Matrix wird ähnlich wie die Fundamentalmatrix definiert:

$$E = [t]_\times R. \tag{21}$$

Dieses Gleichnis angewandt, ergibt es folgende Gleichung:

$$\hat{x}'^T E \hat{x} = 0. \tag{22}$$

Und nachdem x mit $\hat{x}$ wieder substituiert wurde:

$$x'^T K'^{-T} E K^{-1} x = 0. \tag{23}$$

Beim Vergleich mit der Gleichung 17 resultiert folgender Zusammenhang:

$$E = K'^T F K \tag{24}$$

Diese essentielle Matrix ist nur gültig, wenn zwei Singulärwerte gleich sind und die dritte 0 ist [Hartley and Zisserman, 2003, S. 257]. Somit gilt für die essentielle Matrix folgende SVD:

$$E = U diag(1, 1, 0) V^T \tag{25}$$

Nach dieser Formel wird klar, dass E vom Rang defekt ist und deswegen nur 7 Korrespondenzen benötigt werden. Ausgehend von dem Fakt, dass die erste Kameramatrix im Ursprung liegt, also bei $P = [I \mid 0]$, muss nur

noch die zweite Kameramatrix definiert werden. Um zu zeigen, dass diese Bestimmung 4 Möglichkeiten zulässt, wird erst W und Z definiert:

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ und } Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}. \tag{26}$$

Da $E = Udiag(1, 1, 0)V^T$ gilt, sind zwei, wenn man die Vorzeichen ignoriert, der folgende Faktorisierungen für $E = SR$ möglich:

$$S = UZU^T,\ R = UWV^T \text{ oder } R = UW^TV^T. \tag{27}$$

Die dadurch entstandene Formel ermöglicht es, die Translation t aus der Projektionsmatrix $P = K[R \mid t]$ zu bestimmen, da $S = [t]_\times$ gilt. Die Forbeniusnorm von $S = UZU^T$ ist $\sqrt{2}$, was zum Resultat hat, dass $\|t\| = 1$ gilt, wenn die Skalierung von $S = [t]_\times$ mit einbezogen wird. Somit folgt aus $St = 0$, dass t lediglich die letzte Spalte von U darstellt: $t = U(0, 0, 1)^T = u_3$. Letztendlich kann nicht ohne Vorwissen über die Stellung der Kameras zueinander bestimmt werden, welche Vorzeichen bei u_3 zutreffen und ob W transponiert wird oder nicht. Daraus folgen 4 mögliche Varianten um P' mittels Essentieller Matrix zu definieren:[Hartley and Zisserman, 2003, S. 259]

$$P' = [R \mid \pm u_3] \tag{28}$$

Jeder der Korrespondenzpunkte wird in den Raum projiziert

Durch die Ermittlung der Position und Ausrichtung der einzelnen Kameras und deren Punkte auf der Bildflächen können nun die Punkte in den Raum projiziert werden. Dazu kann zum Beispiel die Triangulation verwendet werden, die im Detail im Abschnitt 2.5 erläutert wird.

Aus diesem Abschnitt geht hervor, wie mittels korrespondierender Punkte, sowohl die relative Position zweier Kameras zueinander bestimmt werden kann, als auch die korrespondierenden Punkte in den Raum projiziert werden können. Für spezielle Anwendungsfälle kann dieser allgemeine Algorithmus angepasst oder sogar verbessert werden. Hingegen für unspezifische Fälle gilt der Goldene Standard als Grundlage für die Projektion zweier Kameras im Raum und die damit einhergehende Rekonstruktion von korrespondierenden Punkten.

2.4 Bundle Adjustment

In den vorherigen Kapiteln wurde erläutert, wie mittels von zwei oder drei Bildern die Kamera sowohl kalibriert als auch in einem fixen System ausgerichtet wird. Dabei wird einfachheitshalber angenommen, dass eine Kamera sich im Ursprung dieses Systems befindet und sich die übrigen Kameras relativ zur Ursprungskamera ausrichten. Eine Erweiterung dieses Szenarios ist die Bezugnahme auf mehrere Kameras.

Dieser Algorithmus heißt Bundle Adjustment und kann aus beliebig vielen Kameraperspektiven die visuelle Rekonstruktion errechnen. Zu diesem Zweck wird ein Modell erstellt, dessen Fehlerfunktion minimiert wird, um eine optimale Lösung zu erhalten.[Triggs et al., 2000, S. 298]

Szeliski beschreibt die Grundlagen von Bundle Adjustment als einen iterativen Optimierungsprozess der Kamerapose.[Szeliski, 2010, S. 363] Wobei der generisch präsentierte Algorithmus lediglich auf mehreren Punkten basiert[Szeliski, 2010, S. 324ff.], dazu aber noch die unterschiedlichen Kamerapositionen betrachtet werden müssen. Ausgehend einer unbekannten Kamerapose(R, t) und Kalibrierungsmatrix (K) ist die akkurateste Variante eine Optimierungsfunktion mit den erkannten Punkten auf die quadrierten Reproduktionsfehler durchzuführen:

$$x_{ij} = f(p_i, R_j, C_j, K_j). \tag{29}$$

Und iterativ wird der robustifizierte, linearisierte Reproduktionsfehler minimiert:

$$E_{NLP} = \sum_i \rho = (\frac{\partial f}{\partial R_j}\Delta R_j + \frac{\partial f}{\partial t_j}\Delta t_j + \frac{\partial f}{\partial K_j}\Delta K_j - r_{ij}), \tag{30}$$

wobei $r_{ij} = \tilde{x}_{ij} - \hat{x}_{ij}$, also der Fehler zur vorhergesagten Position und der partiell abgeleitete Vektor abhängig der unbekannten Pose und Kalibrierung sind.

Eine einfachere Herangehensweise ist, die Ausformulierung der Formel in verschiedene Teilschritte:

$$y^{(1)} = f_T(p_{ij}; c_{jn}) \qquad = p_{ij} - c_{jn}, \tag{31}$$
$$y^{(2)} = f_R(y^{(1)}; q_{jn}) \qquad = R_j(q_{jn})y^{(1)}, \tag{32}$$
$$y^{(3)} = f_P(y^{(2)}) \qquad = \frac{y^{(2)}}{z^{(2)}}, \tag{33}$$
$$x_i = f_C(y^{(3)}; k_j) \qquad = K_j(k_j)y^{(3)}. \tag{34}$$

In den vorgestellten Schritten wird der Index n als Unterscheidung für die verschiedenen Kameras verwendet und j ist der Index für die eventuelle Nutzung mehrere Kamerazentren oder -rotationen. q sind in diesem Fall Quaternionen der Kameras. Die einzelnen Schritte sind aufgeschlüsselt, sodass im ersten Schritt die Translation, im zweiten die Rotation und im dritten Schritt die Perspektive berechnet wird. Ein Vorteil dieser Ausführungen [Szeliski, 2010, S. 325ff.] ist, dass sobald ein $\tilde{x}_i$ berechnet wurde, daraus alle Poseparameter, wie c_n, q_n und k, durch eine Kettenregel partieller Ableitungen berechnet werden:

$$\frac{\partial r_i}{\partial p^{(n)}} = \frac{\partial r_i}{\partial y^{(n)}} \frac{\partial y^{(n)}}{\partial p^{(n)}}, \tag{35}$$

wofür $p^{(n)}$ einen der Parametervektoren darstellt, der optimiert wird.

Eine weitere Unterscheidung zum iterativen Algorithmus ist ein zusätzlicher Schritt, der die einzelnen Kameras zusammenführt und somit untereinander optimiert. Dazu wird die Kovarianzmatrix $\sum_{ij}$, aus den Messfehlern der einzelnen Kameraposition und Punkten mit berücksichtigt und es entsteht folgende Rechnung:

$$r_{ij} = \tilde{x}_{ij} - \hat{x}_{ij}, \tag{36}$$

$$s_{ij}^2 = r_{ij}^T \sum_{ij}^{-1} r_{ij}, \tag{37}$$

$$e_{ij} = \hat{\rho}(s_{ij}^2), \tag{38}$$

dabei gilt $\hat{\rho}(s_{ij}^2) = \rho(r)$. Die dazugehörige Jakobi Matrix wird folgendermaßen definiert:

$$\frac{\partial e_{ij}}{\partial s_{ij}^2} = \hat{\rho}'(s_{ij}^2), \tag{39}$$

$$\frac{\partial s_{ij}^2}{\partial \tilde{x}_{ij}} = \sum_{ij}^{-1} r_{ij}. \tag{40}$$

Der Vorteil dieser Aufteilung innerhalb kleiner Teilschritte ist nach [Szeliski, 2010, S. 364], dass sowohl die Rechnung mit den partiellen Ableitungen und der Jakobi Matrix vereinfacht wird, als auch an jeder individuellen Kameraeinstellung angepasst werden kann.

Zusammengefasst ist Bundle Adjustment ein dünnbesetztes geometrisches Parameterschätzungsproblem, wobei ein Parameter die Kombination aus 3D Merkmalspunkten, Kamerapositionen und -kalibrierungen ist. [Triggs et al., 2000, S. 299] Für diese Aufgabe bietet das System mehrere Vorteile:

- **Flexibilität**: Bundle Adjustment kann eine sehr breite Facette an 3D Merkmalen und Kameras, Typen von Szenen, Informationsquellen und Fehlermodelle verarbeiten.
- **Genauigkeit**: Durch das akkurate, statistische Fehlermodell wird ein genaues und leicht zu interpretierendes Ergebnis erstellt
- **Effizienz**: Der Einsatz von ökonomischen und schnell konvergierenden, numerischen Methoden ermöglicht es, dass die ausgereiften Bundle Adjustment Algorithmen nahezu optimal große Daten verarbeiten können. [Triggs et al., 2000, S. 300]

2.5 Triangulierung

Wenn aus mehreren Perspektiven der gleiche Punkt gefunden wird, ist es selten der Fall, dass innerhalb der Perspektiven die gleiche Annahme über

die Position des Punktes im Raum getroffen wird. Somit muss eine algorithmische Lösung gefunden werden, um die unterschiedlichen Annahmen zu einer gemeinsamen Aussage über den Punkt zu vereinen. Die einfachste Form ist die Triangulation.

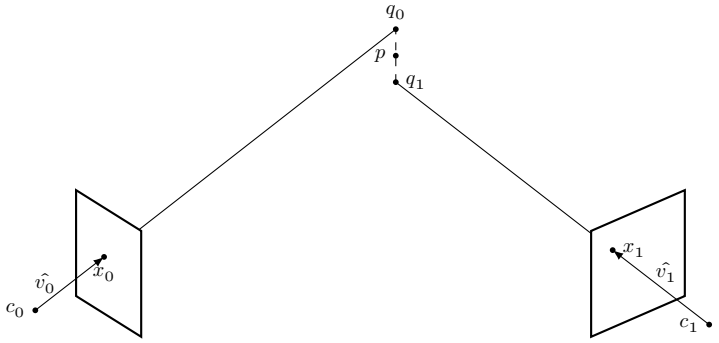


Abbildung 4: Schematische Darstellung der Triangulation aus zwei Kameraperspektiven. c_i ist das Kamerazentrum, x_i sind die Bildpunkte in der Perspektive und q_i sind die Punkte im Raum die am nähsten zum Punkt p sind. Diese Grafik ist angelehnt an [Szeliski, 2010, S. 346].

Angenommen aus mehreren Kameraperspektiven $\{P_j = K_j[R_j|t_j]\}$, wobei $K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$ die Kameramatrix, R die Rotationsmatrix und t der Translationsvektor der j-ten Kamera ist, wird in allen Kameras der gleiche Punkt $\{x_j\}$ gesehen. Schematisch wird das in Abbildung 4 dargestellt. Aus der oberen Gleichung folgt $t_j = -R_j c_j$, wobei c für das Kamerazentrum steht. Wenn $\hat{v}$ der Vektor zwischen dem Bildpunkt und dem Kamerazentrum ist, dann wird der nächste Punkt q zum tatsächlichen Punkt p folgendermaßen berechnet:

$$q_j = c_j + (\hat{v_j}\hat{v_j}^T)(p - c_j) = c_j + (p - c_j)_{\parallel}. \tag{41}$$

Dafür ist allgemein $v_{\parallel}$ definiert

$$v_{\parallel} = \hat{n}(\hat{n}v) = (\hat{n}\hat{n}^T)v. \tag{42}$$

Die minimale Distanz zwischen dem aufgespannten Vektor $\hat{v}$ und dem Punkt p ist:

$$\|c_j + d_j\hat{v_j} - p\|^2, \tag{43}$$

welches als Minimum $d_j = \hat{v_j}(p - c_j)$ hat und die quadrierte Abstandsfunktion zwischen p und q_j ist:

$$r_j^2 = \|(I - \hat{v}\hat{v}^T)(p - c_j)\|^2 = \|(p - c_j)\|^2. \tag{44}$$

Die optimale Lösung, was der kürzesten Strecke zwischen p und q_i entspricht, kann mit der Least Square Methode errechnet werden, indem alles aufsummiert wird und für p gilt:

$$p = [\sum_j (I - \hat{v_j}\hat{v_j}^T)]^{-1}[\sum j(I - \hat{v_j}\hat{v_j^T})c_j] \tag{45}$$

Dieser Ansatz entspricht dem Gedankengang von Szeliski [Szeliski, 2010, S.345ff.].

Durch die Triangulation ist es realisierbar zwischen beliebig vielen Perspektiven einen Punkt im Raum zu ermitteln, obwohl sich die vom Bild ausgehenden Strahlen nicht in einem Punkt vereinigen.

3 Konzepte und Funktionsweise von ORB SLAM 2

ORB-Slam wurde entwickelt von Raúl Mur-Artal et al. [Mur-Artal et al., 2015] und bietet ein Framework, das aus fortlaufenden Kamerabildern, sowohl die eigene Position bestimmt, als auch gefundene Merkmalspunkte aus den Bildern in den 3D Raum projizieren kann. Die Merkmalspunkte werden durch einen ORB Algorithmus gefunden. Folgend wird zuerst ORB und danach werden SLAM Prinzipien vorgestellt. Darauf folgt die Darstellung der Architektur vom ORB-SLAM und die Anwendung im AutoNOMOS Lab Projekt der FU Berlin.

3.1 Merkmalserkennung mit ORB

Oriented FAST and Rotated BRIEF (ORB) ist eine Entwicklung aus dem Willow Garage von Rublee et. al [Rublee et al., 2011]. In der Entwicklung des Algorithmus stand es im Vordergrund vorhandene Algorithmen, wie FAST und BRIEF, weiterzuentwickeln und deren Stärken auszunutzen, indem die Schwächen ausgebessert werden. Daraus entstand ein schneller und robuster Algorithmus, der frei verfügbar und zum Beispiel in Open Source Bibliotheken wie OpenCV enthalten ist.

Ein Algorithmus zur schnellen Kanten- und Eckendetektion ist FAST [Rosten and Drummond, 2006] [Rosten et al., 2010]. Durch eine Bildpyramide werden multiskalare Merkmalspunkte gefunden. Jedoch wird die Ausrichtung der gefundenen Merkmalspunkte nicht weiter berücksichtigt. Dies wird mittels Bildmoment verbessert:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y), \tag{46}$$

dabei ist x und y die jeweilige Koordinate und die Funktion $I(x,y)$ gibt die Intensität des Bildes am Punkt x, y wieder. Daraus wird schließlich der geometrische Schwerpunkt des Ausschnitts berechnet:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \tag{47}$$

Durch die Berechnung des Schwerpunktes des Bildes kann ein Bezug zum eigentlichen Mittelpunkt hergestellt werden:

$$\theta = atan2(m_{01}, m_{10}), \tag{48}$$

wobei *atan2* die Umrechnung von zwei Koordinaten in einem Winkel ist. Diese Erweiterung wird **Oriented FAST** genannt[Rublee et al., 2011]. Dieser Schwerpunkt wird bei gleichartigen Patches sehr ähnlich ausfallen, unabhängig von der Drehung des Patches. Die Rotationsinvarianz ist folglich

gewährleistet, da bei gleichen Patches mit unterschiedlicher Drehung die Vergleichbarkeit hergestellt werden kann, indem die Ausschnitte gleich zum Schwerpunkt ausgerichtet werden.

Nachdem das Merkmal gefunden wurde, muss die Umgebung wiedererkennbar und performant beschrieben werden. Zur Beschreibung wird der Algorithmus BRIEF [Calonder et al., 2010] verwendet. Mittels BRIEF wird der Bildausschnitt, in dem das Merkmal gefunden wurde, pixelweise in einem bestimmten Muster verglichen, und anschließend wird das Ergebnis des Vergleichs binär abgespeichert. Wenn p einen Patch abbildet, und somit $p(x)$ die Intensität des Patchs an der Stelle x ist, dann ist der binäre Test τ folgendermaßen definiert:

$$\tau(p; x, y)) = \begin{cases} 1, & p(x) < p(y) \\ 0, & p(x)\ p(y) \end{cases} \tag{49}$$

Die folgende Funktion f dient zur Darstellung des Vektors des binären Tests:

$$f_n(p) = \sum_{1 \leq i \leq n} 2^{i-1} \tau(p; x_i, y_i). \tag{50}$$

Durch den Binärtest ist BRIEF sehr stark auf die Rotation des Ausschnittes angewiesen. Es gibt in dem bisherigen Algorithmus keine Möglichkeit die Rotation mit in die Berechnung einzubeziehen. Dieses Problem wird innerhalb der Weiterentwicklung für ORB behoben. Wenn das Merkmalsset von n binären Tests in einer $2 \times n$ Matrix $S = \begin{pmatrix} x_1 & , \dots & , x_n \\ y_1 & , \dots & , y_n \end{pmatrix}$ dargestellt wird, dann kann die Orientierung des Patches mit einbezogen werden:

$$S_\theta = R_\theta S \tag{51}$$

Daraus entsteht der steered Brief Operator[Rublee et al., 2011]:

$$g_n(p, \theta) = f_n(p)(x_i, y_i) \in S_\theta \tag{52}$$

Die Winkel werden auf $2\pi/30$ diskretisiert und daraus wird eine Lookup Tabelle von vordefinierten BRIEF Mustern konstruiert.

Weiterhin wurden die Eigenwerte von BRIEF und steered BRIEF unter Einsatz der Hauptkomponentenanalyse untersucht. Dabei ist festgestellt worden, dass die ersten 10 bis 15 Komponenten die essentiellen Informationen enthalten. Das Ziel mittels weniger Informationen eine hohe Varianz zu erzeugen und wenig Korrelation auftreten zu lassen, wurde durch ein offline Training über eine große Menge an Trainingsdaten erreicht. Dieses Set an Tests wird robust BRIEF (rBRIEF) genannt. [Rublee et al., 2011]

Die beiden wichtigsten Bestandteile wurden somit weiterentwickelt, um robuste und rotationsinvariante Merkmale extrahieren zu können. Trotzdem

Detektoren	**Laufzeit[ms]**	**Speed-up**	**# Merkmale**
SURF	176	1.9	2 911
FAST	2	169.0	5 158
STAR	17	19.9	849
MSER	60	5.6	483
BRISK	10	33.8	1 874
ORB	7	48.3	594

Tabelle 1: **Durchschnittliche Laufzeit** In dieser Tabelle wird die absolute Laufzeit, der "Speed-up", welcher ein Quotient aus der Zeit der sequentiellen Suche T_S und der Zeit des effizienten Matching T_E ist, und die gefundene Anzahl der Merkmale pro Algorithmus, aufgeführt.[Miksik and Mikolajczyk, 2012]

Detektoren	**Deskriptor**	**Präzision**	**Recall**
SURF	SIFT	0.525	0.533
BRISK	BRISK	0.504	0.527
ORB	ORB	0.493	0.495
FAST	SIFT	0.366	0.376

Tabelle 2: **Präzision/Recall** Die Präzision ist das Verhältnis zwischen dem korrekt gefundenen Merkmal und den möglichen Merkmalen, verglichen mit der Merkmalsdatenbank. Recall ist das Verhältnis aus dem gefundenen Merkmal und deren Korrespondenzen.[Miksik and Mikolajczyk, 2012]

bedarf es noch eines schnellen Algorithmus, um die binären rBRIEF Muster zu vergleichen. Dazu wird Locality Sensitive Hashing verwendet. Dieser Algorithmus teilt die Deskriptoren in einzelne Hashpakete auf, welche die Suche eines Musters auf eine Nearest Neighbor Suche der entsprechenden Deskriptor reduziert. [Rublee et al., 2011]

In der Tabelle 1 und 2 sind die Ergebnisse aus einer Studie dargestellt, die verschiedene Detektoralgorithmen miteinander vergleicht. [Miksik and Mikolajczyk, 2012] Zu Erkennen ist, dass ORB im Vergleich zu anderen bekannten Algorithmen, wie zum Beispiel SURF/SIFT, wenige Merkmale findet, allerdings sehr schnell und trotzdem ähnlich präzise arbeitet.

3.2 SLAM

Durch ORB werden visuelle Merkmale in Bildern oder Videos gefunden. Diese in Bezug zu setzen zu weiteren gefundenen Merkmalen und somit die Umgebung zu kartografieren, ist ein Problem, das als Simultaneous Localization and Mapping (SLAM) bezeichnet wird.

Während des laufenden Prozesses eines Roboters können in vordefinierten Zeitabständen Umgebungsvariablen erfasst werden. Zu den Variablen

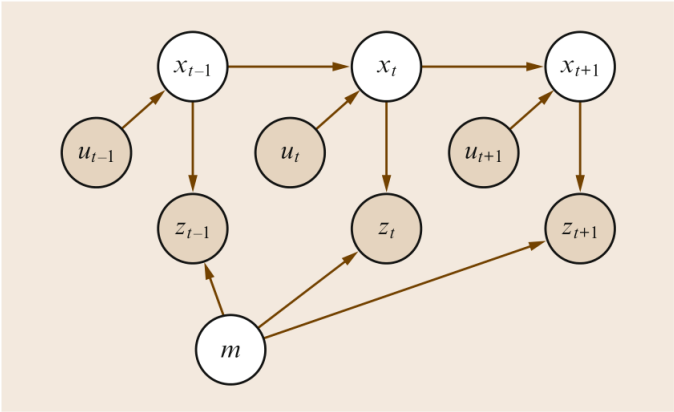


Abbildung 5: Graphisches Modell des SLAM Problems. x_t ist der Pfad, u_t ist die Bewegung während der Zeit t, z_t sind die erkannten Merkmale in der Umgebung m. [Thrun and Leonard, 2008]

gehört unter anderem die absolute Position $X_T = \{x_1, x_2, \dots, x_T\}$, die bedingt durch die relative Bewegung $U_T = \{u_1, u_2, \dots u_T\}$ verändert werden kann. Die Umwandlung von relativer Bewegung in absoluter Position wird unter dem Begriff Odometrie zusammengefasst. T ist in diesem Szenario der mögliche Endzeitpunkt, der aber auch unendlich sein kann. Wenn man annimmt, dass die Messungen der Umgebung m diskret durchgeführt werden, dann wird die Bestimmung der Umgebung mit $Z_T = \{z_1, z_2, \dots, z_T\}$ definiert. Zusammengefasst kann das Problem folgendermaßen definiert werden: [Thrun and Leonard, 2008]

$$p(X_T, m | Z_T, U_T) \tag{53}$$

Unter der Bedingung der veränderlichen absoluten Bewegung und der diskret messbaren Umgebung, ergeben sich relative Bewegungen und bestimmbare Merkmale der Umgebung. In Abbildung 5 wird schematisch das SLAM Problem dargestellt. Im Detail wird SLAM in zwei prinzipielle Ansätze unterteilt: online und offline. Im online Ansatz werden die aktuellen Daten mit den vergangenen verarbeitet und in Relation gesetzt. Gegensätzlich dazu ist der offline Ansatz, welcher die komplette Erkundung abschließen muss, ehe die Daten vollumfänglich analysiert werden.

SLAM wird hauptsächlich in drei Hauptparadigmen unterteilt [Thrun and Leonard, 2008]:

- **Kallmann Filter:** Durch statistische Methoden wird eine Kovarianz über die aktuelle lokale Position erstellt, die mit Hilfe von Messungen der Umgebung stetig aktualisiert wird.

- **Graphbasierte Optimierungstechnik:** Mittels eines Graphen wird der optimale Weg des Roboters bestimmt. Dafür ist jede Position zu einem festen Zeitpunkt ein Knoten im Graph. Sowie die gefundenen Merkmale zu dem Knoten verbunden werden, werden die aufeinanderfolgenden Knoten auch verbunden. Jede Verbindung ist eine Bedingung für das Optimierungsproblem.

- **Partikelmethode:** Ein Partikelfilter beruht auf beliebig vielen Annahmen (Partikel) über die aktuelle Position im Raum. Für die Umsetzung werden am Anfang die Anzahl der Partikel, bedingt durch den vorherigen Schritt, gestreut und danach mit Hilfe der Umgebungsbeobachtung korrigiert.

ORB SLAM ist nach den Paradigmen als eine graphbasierte Optimierungstechnik einzuordnen. Wie im nächsten Kapitel näher beschrieben wird, baut der Algorithmus einen Graphen zwischen besonderen Bildern, genannt KeyFrames, innerhalb der Laufzeit auf. Zwischen diesen KeyFrames wird eine Ähnlichkeit festgestellt, die ab einem bestimmten Schwellwert eine Verbindung darstellt. Am Ende wird ein minimaler Spannbaum aufgebaut, dessen Basis die stärksten Bindungen zwischen den einzelnen KeyFrames darstellt. Somit wird ein Optimierungsproblem aufgestellt, dass durch die Ähnlichkeit der KeyFrames gelöst wird.

Zwischen SLAM und Structure from Motion gibt es keine klare Abgrenzung. In beiden Themengebieten wird die Umwelt mit Hilfe von Merkmalen abgebildet und im Raum dargestellt. Hinsichtlich der Ausführung scheint es einen Unterschied zu geben. Structure from Motion scheint nur offline die Daten zu verarbeiten, also nachdem vollständig alle Bilder der Szene aufgenommen wurden. Das ist hingegen ein sehr kleiner Unterschied, der keinen Anspruch auf Allgemeinheit hat. Folglich kann in diesem Rahmen Structure from Motion mit SLAM gleich gesetzt werden.

3.3 Funktionsweise von ORB SLAM

ORB SLAM wurde entwickelt von Mur-Artal, Montiel und Tardós [Mur-Artal et al., 2015]. Dieser Algorithmus spezialisiert sich darauf, keine Odometriedaten aus dem Roboter zu verarbeiten, sondern lediglich aus dem gegebenen Bildmaterial eine eigene visuelle Odometrie zu berechnen. Dazu wird hauptsächlich ein lokales und globales Bundle Adjustment durchgeführt, dass, wie in Kapitel 2.4 beschrieben, verschiedene Bildperspektiven auf eine Szene an Hand von extrahierten Merkmalspunkten ausrichtet.

Die Aufteilung der einzelnen Teilabschnitte von ORB SLAM ist in der Abbildung 6 dargestellt. Der Algorithmus ist in drei Gebiete unterteilt: Tracking, Local Mapping und Loop Closing.

Der erste Part, das Tracking, fokussiert die Erkennung von Merkmalspunkten und die Entscheidung, ob das Bild, hier als Frame bezeichnet, als

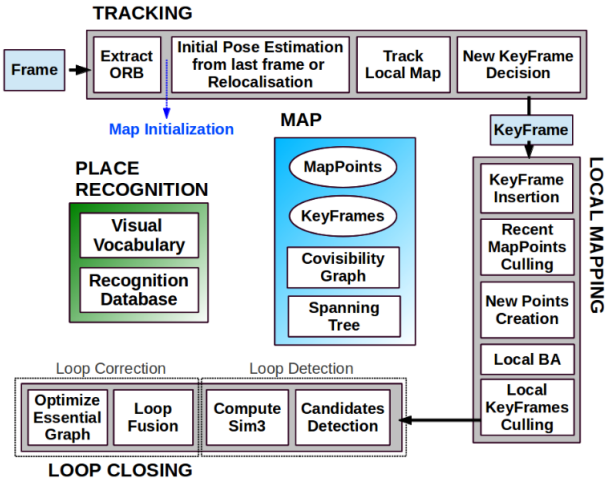


Abbildung 6: Schematische Darstellung der Programmstruktur von ORB SLAM 2 [Mur-Artal et al., 2015].

aussagekräftig genug erscheint, um als KeyFrame für den Algorithmus weiter verwendet werden zu können. Die Merkmalspunkte werden durch den ORB Algorithmus erfasst. Dieser wird in Vergleich zu anderen Merkmalserkennungsalgorithmen präferiert, weil ORB schnell robuste Merkmale extrahiert. Im Abschnitt 3.1 wurde näher auf die Funktionsweise von ORB eingegangen und ein Vergleich zu anderen aktuellen Algorithmen präsentiert.

Im Folgenden werden detailliert die einzelnen Schritte der Programmstruktur von ORB SLAM vorgestellt. Dabei wird weitestgehend darauf geachtet, die Reihenfolge des Schemas beizubehalten, obwohl teilweise Bemerkungen zu außerstrukturellen Vorgängen nötig sind, um die Arbeitsschritte zu erklären.

Initialisierung

Zur Initialisierung wird ein Referenzframe mit den folgenden Frames verglichen. Bei genügend übereinstimmenden Features wird der Referenzframe als Initialisierung verwendet. Tritt dieses Ereignis nach mehreren Frames nicht ein, ersetzt der aktuelle Frame den Referenzframe und der Initialisierungsprozess wird neu begonnen. Im nächsten Schritt des Initialisierungsprozesses werden parallel aus den beiden Frames die Homographie und Fundamentalmatrix berechnet. Ein Score, der sich aus den jeweiligen Fehlerraten beider Berechnungen ergibt, bestimmt, welche Form der Rekonstruktion gewählt wird. Je weniger Fehler auftreten, desto besser ist der Score. Näheres zur allgemeinen Bestimmung der Homographie und Fundamentalmatrix ist im Abschnitt 2.2 und 2.3 zusammengefasst. Stellt sich heraus, dass die Fehlerrate zu groß ist, wird keine von beiden Möglichkeiten verwendet und der Initialisierungsprozess beginnt von Neuem, wobei der Referenzframe vorerst

gleich bleibt.

Ist das Modell gewählt, werden 8 Bewegungshypothesen erstellt, mittels der Methode von Faugeras et. al [Faugeras and Lustman, 1988]. Die Triangulation der Hypothesen wird anschließend mit jeder Hypothese verglichen, woraufhin die passendste gewählt wird. Wenn es keinen eindeutigen Favoriten gibt, wird der Initialisierungsprozess neu gestartet. Durch die vielen Prüfschritte im Initialisierungsprozess ist der Algorithmus beständig gegen vermeintliche Doppeldeutigkeiten.

Tracking

Nach der Initialisierung werden die weiteren Frames in den Trackingprozess aufgenommen. Dazu wird anhand der letzten Frames ein Bewegunsmodell angenommen, um ausgehend von der Vermutung, über Position und Ausrichtung der Kamera, besser die tatsächliche Pose bestimmen zu können. Abhängig von dieser Position wird in einem Ausschnitt des Bild nach Merkmalen gesucht, die in anderen Frames als signifikant erkannt wurden. Werden nicht genügend Merkmale gefunden, wird die Suche auf einen größeren Ausschnitt ausgedehnt. Werden ausreichend Merkmale gefunden, wird die Position und Ausrichtung mittels der gefundenen Merkmale angeglichen.

Um die Komplexität gegenüber einer globalen Karte zu minimieren, werden die Frames und die dazugehörigen Merkmale nur lokal betrachtet. Dazu wird eine lokale Karte aus KeyFrames und deren lokalen Nachbarn aufgebaut, welche mit dem aktuellen Frame ausreichend Merkmale teilen. Aus der Sammlung der KeyFrames werden die einzelnen Merkmale in den aktuellen Frame projiziert. Dafür muss nur gewährleistet sein, dass diese Merkmale innerhalb der aktuellen Perspektive sichtbar sind. Konkret wird an den einzelnen Merkmalen der Winkel und der Abstand zum Frame berechnet, wobei bei einem zu großen Fehler die Merkmale verworfen werden können. Merkmale, die bislang keine Übereinstimmung in anderen KeyFrames vorweisen, werden nun mit den naheliegendsten Merkmalen verglichen und gegebenenfalls können dadurch Treffer erzielt werden. Abschließend kann daran die Ausrichtung der Kamera optimiert werden.

Zuletzt wird im Trackingprozess entschieden, ob der aktuelle Frame ein KeyFrame ist. Dazu müssen folgende Bedingungen erfüllt sein:

- Zum letzten KeyFrame müssen mindestens 20 Frames vergangen sein.
- Mehr als 20 Frames müssen nach der letzten globalen Relokalisierung vergangen sein oder der lokale Mappingprozess ist derweil unbeschäftigt.
- Der aktuelle Frame beinhaltet mindestens 50 Merkmale.
- Im Frame sind weniger als 90% der Merkmale aller naheliegenden KeyFrames.

Durch die Sicherstellung, dass ausreichend neue Merkmale im KeyFrame gefunden wurden, kann die exakte Distanz zwischen den KeyFrames unberücksichtigt bleiben. Zum Beispiel, wenn ein Roboter ausschließlich die Kamera nach vorn gerichtet hat und rückwärts fährt, kann es passieren, dass ORB SLAM keine neuen KeyFrames in den Prozess aufnimmt, da nicht genügend neue Features durch die Frames hinzugefügt werden. Dies ist eine explizite Abgrenzung zum Parallel tracking and mapping (PTAM) Algorithmus [Klein and Murray, 2007], welcher auch merkmalsbasiert einen SLAM Algorithmus implementiert hat, jedoch die Frames nicht auf Grund der Ähnlichkeit auswählt, sondern den Abstand zwischen den Frames als Kriterium zur Auswahl der KeyFrames benutzt. [Mur-Artal et al., 2015]

Die globale Relokalisierung ist erforderlich, sobald der aktuelle Frame nicht ausreichend übereinstimmende Merkmale findet und somit nicht mehr sicher gestellt werden kann, wo sich der Standpunkt des Frames relativ zur aufgebauten Karte befindet. Diese Relokalisierung basiert auf einem Bag of Words Algorithmus, namentlich DBoW2, der mittels Diskretisierung visueller Wörter, auch visuelles Vokabular genannt, schon bekannte Bereiche wiedererkennt.[Gálvez-López and Tardós, 2012]

Das Vokabular wird im Vorfeld offline aufgebaut, wobei sichergestellt werden muss, dass die Bilder einzigartig genug sind und eine sehr große Menge an möglichen Szenen abgedeckt wird, um im konkreten Einsatz gezielte Szenen beschreiben zu können. [Mur-Artal and Tardós, 2014] Unter der Voraussetzung, dass die Trainingsphase allgemein gehalten ist, kann das gleiche Vokabular für unterschiedliche Szenen eingesetzt werden. Wie weiter oben erwähnt, sind die bereits integrierten KeyFrames zu einem gewissen Teil ähnlich zueinander. Folglich ist es unwahrscheinlich, dass beim Vergleich eines neuen Frames zu den vorhandenen KeyFrames, ausschließlich ein KeyFrame passt. Somit wird zusätzlich zu dem Ähnlichkeitswert zwischen KeyFrame und aktuellem Frame auch noch die lokale Nähe betrachtet. Das ist eine Verbesserung zum ursprünglichen DBoW2 Algorithmus, der eigentlich nur die zeitliche Nähe betrachtet. Schlussendlich wird zum Finden der Position und Ausrichtung des Frames ein Perspective-n-Point (PnP) Algorithmus verwendet, der bei ausreichend gefundenen Inliers zum Finden und Optimieren der Frameposition verwendet wird.

Covisibility Graph

Der Covisibility Graph wird benutzt, um die lokale Anhängigkeit von KeyFrames darzustellen. Dazu werden die KeyFrames in einem ungerichteten Graphen miteinander verbunden, welche die gleichen Merkmale teilen. Je mehr Merkmale sich die KeyFrames teilen, desto höher ist das Kantengewicht. Nachdem ein KeyFrame eingefügt wurde, wird nur die Kante mit dem größten Gewicht in einem *Essential Graph* gespeichert. Somit wird ein minimaler Spannbaum aufgebaut. Durch die Untermenge des Covisibility Graphs wird ein geringer Speicheraufwand benötigt. Falls ein KeyFrame ausgesondert wird, weil die Informationen redundant oder fälschlich sind,

dann wird der Covisibility Graph und somit auch der Essential Graph aktualisiert. Laut den Entwicklern ist dadurch der Graph effizient genug, dass selbst ein globales Bundle Adjustment nur wenig Verbesserungen ermöglicht. Des weiteren werden durch die Technik des Covisibility Graphs Schleifen schnell gefunden, weil erkannt wird, wenn Frames eine Ähnlichkeit aufweisen, obwohl diese nach der bisherigen Karte möglicherweise nicht räumlich beieinander liegen.

Local Mapping

Zum Local Mapping wird der aktuelle KeyFrame in den Covisibility Graph eingefügt. Der Graph wird anschließend aktualisiert und schließlich ein minimaler Spannbaum, der Essential Graph, aufgebaut. Mittels des oben erwähnten DBoW2 Algorithmus wird der KeyFrame zu einer Bag of Words Repräsentation umgewandelt, das unter anderem zur Triangulierung beim Einfügen neuer Merkmale verwendet wird.

Diese durch die Triangulierung errechneten Kartenpunkte müssen mehrere Tests zur Aufnahme in die lokale Karte bestehen:

- Die Punkte müssen eine Erkennungsrate von mehr als 25% in den KeyFrames, in denen diese rechnerisch sichtbar sein müssten, vorweisen. Unabhängig vom Rauschen oder Merkmalserkennungsraten können Merkmale zum Beispiel durch wechselnde Perspektiven verdeckt werden. Infolge der Berücksichtigung dieses Aspekts ist die Erkennungsrate eher gering.

- Mindestens in drei KeyFrames müssen die Punkte gesehen und erkannt werden.

Sind diese Tests einmal bestanden, kann ein Punkt nur noch während der Laufzeit von der Karte gelöscht werden, wenn sich dieser bei weniger als drei KeyFrames im Verfolgungsprozess befindet. Vorerst scheint das wie ein immer zutreffendes Kriterium, weil es schon eingangs geprüft wurde. Jedoch können KeyFrames ausgeschlossen und gelöscht werden und somit ist diese Kriterium auch nach dem Einfügen des Punktes erfüllbar.

Um neue Punkte zur Karte hinzuzufügen, werden ORB Paare zwischen den Nachbar-KeyFrames im Covisibility Graphen gesucht und diese dann trianguliert. Eine detailierte Beschreibung zur Triangulation zwischen mehreren Punkten befindet sich im Abschnitt 2.5. Des weiteren werden Merkmale ohne eine gefundene Übereinstimmung in “Bag of Words”-ähnlichen KeyFrames gesucht. Im letzten Abschnitt des Trackings wird dieses Verfahren näher beschrieben. Dabei werden nur triangulierte Punkte akzeptiert, wenn sich diese in der positiven Tiefe beider KeyFrames befinden und die Parallaxe, der Reproduktionsfehler und die Skalierungskonsistenz untersucht und akzeptiert wurden.

Ein lokales Bundle Adjustment wird lediglich mit den KeyFrames ausgeführt, die durch den Covisibility Graphen mit dem aktuellen KeyFrame

verbunden werden. Die übrigen KeyFrames bleiben fix und werden nur noch im Optimierungsprozess berücksichtigt.

Im letzten Schritt des Local Mapping Prozesses werden redundante KeyFrames gefunden und aus dem Mapping ausgeschlossen. Hierbei wird festgestellt, ob 90% der dargestellten Punkte eines KeyFrames durch mindestens drei anderen KeyFrames repräsentiert wird. Wenn dies der Fall ist, wird der KeyFrame gelöscht. Durch die Eliminierung von Redundanzen wird der Datensatz in einem stetig komplexer werdenden Mappingprozess effizient im System angelegt.

Loop Closing

Zum Schließen eines gefundenen Kreises (Loop) muss zum einen die Erkennung des Loops durchgeführt werden und zum anderen der vorhandene Loop korrigiert werden.

Als erstes wird mittels DBoW2 eine Ähnlichkeit der nächsten KeyFrames im Covisibility Graphen berechnet. Der geringste Ähnlichkeitswert s_{min} wird dazu verwendet, um die angeforderten unähnlicheren Ergebnisse aus der Bildwiedererkennungsdatenbank zu filtern. Alle KeyFrames, die direkt mit dem aktuellen KeyFrame verbunden sind, werden ausgeschlossen. Um sicher zu stellen, dass die Ähnlichkeit kein Zufall ist, müssen mindestens drei zusammenhängende KeyFrames diesem Kriterium entsprechen, damit diese als Loop Kandidaten angesehen werden können.

Im nächsten Schritt wird der Kandidaten-KeyFrame dem aktuellen KeyFrame angeglichen. Dazu wird die aktuelle Abweichung und somit der fortlaufende Fehler bestimmt. Vorerst werden, wie im Prozess vorher zum Umwandeln des Frames in einen KeyFrame, die ORB Merkmale zwischen den KeyFrames abgeglichen. Als Ergebnis entsteht ein 3D zu 3D Punkteabgleich zwischen den beiden KeyFrames. Alternativ dazu wird ein RANSAC Verfahren eingesetzt, dass durch mehrere Iterationen die Ähnlichkeit beider KeyFrames bestimmt, nach dem Ansatz von [Horn, 1987]. Danach wird nach Inliers gesucht und mit der gefundenen Perspektive wird die aktuelle Ausrichtung des KeyFrames optimiert. Das wird solange fortgesetzt, bis die aktuelle Ausrichtung akzeptiert werden kann.

Um den Kreis vollständig schließen zu können, müssen alle KeyFrames neu ausgerichtet werden. Dazu werden ausgehend vom aktuellen KeyFrame und dessen Pendant die Nachbarn im Covisibility Graphen untersucht, um anschließend die übereinstimmenden Punkte neu zu positionieren und daran die KeyFrames neu auszurichten. Abschließend wird mittels über eine die Optimierung des Covisibility Graphen der Essential Graphs verbessert und somit die aktualisierte Karte abgespeichert.

4 Praktischer Einsatz von ORB SLAM

Der ORB SLAM wurde im Rahmen dieser Arbeit im Projekt AutoNOMOS Labs praktisch eingesetzt. Das zur Verfügung gestellte Auto besitzt vier Kameras, die einen 360° Blick ermöglichen. Dazu sind zwei Kameras in den Seitenspiegeln verarbeitet, wobei eine in und eine gegen die Fahrtrichtung ausgerichtet ist. Getestet wurde mit einem Laptop, der einen Intel i5-4300U Prozessor und eine integrierte Graphikkarte verbaut hat. Der RAM Speicher beträgt 8 GB. Für den praktischen Test wurden Bilder mit weiteren Daten, wie zum Beispiel der Odometrie, aufgenommen, damit der Algorithmus dann offline die Daten verarbeiten kann. Obwohl die Odometriedaten per se nicht benötigt werden, gab es Versuche diese in dem Algorithmus mit zu verwerten, um vermeintliche Schwachstellen auszubessern.

Für die Aufnahme und das Abspielen der Daten wurde ROS verwendet. Robot Operating System (ROS) ist ein Framework, das Bibliotheken und Werkzeuge zur Verfügung stellt, welche die Entwicklung von Robotern unterstützen.[4] Innerhalb des Systems können klar abgegrenzt über verschiedene Datentransferkanäle Nachrichten mittels eines First In First Out (FIFO) Prinzips zwischen den Modulen ausgetauscht werden. Da alle Nachrichten mit einem internen zeitlichen Stempel versehen sind, können einzelne Nachrichtentypen miteinander synchronisiert werden, wie zum Beispiel die Odometriedaten und die aufgenommenen Bilder. Module sind abgegrenzte Einheiten, die zweckgebunden einkommende Daten bearbeiten und in einen oder mehreren dedizierten Kanälen versenden. Folglich können einzelne Module ausgetauscht oder gar frei zur Verfügung gestellt werden, um in anderen Modulketten eingesetzt zu werden.

Das AutoNOMOS Labs Projekt ist mit vielen Modulen auf ROS eingestellt. Somit wurde die Aufnahme der Daten mit ROS realisiert und als ein ROS eigenes Archiv namens Rosbag verpackt. Dieses Archiv speichert in vordefinierten Zeitabständen (Ticks) alle Daten, die über die ausgewählten Kanäle zur Verfügung gestellt werden. ORB SLAM stellt außerdem auch eine Verbindung zu ROS bereit. Dadurch lassen sich mit wenig Aufwand die aufgenommenen Daten in den Algorithmus einspeisen.

4.1 Erstanbindung an ORB SLAM 2

Der Einsatz vom ORB SLAM 2 erschien vorerst problemlos. Sobald der Videoausgang aus dem Archiv an den passenden Eingangskanal des Programms anlag, konnten in den Frames Merkmale gefunden und auch eine lokale Karte aufgebaut werden. Jedoch war es auffällig, dass bei Erhöhung der Geschwindigkeit der Algorithmus den Bezug zur bisher aufgebauten Karte verlor und sich danach, durch die sich stetig verändernde Umgebung, nicht mehr relokalisieren konnte.

[4] Nähere Informationen findet man zu ROS unter `www.ros.org`.

Abbildung 7: Aufnahme einer Fahrt mit schwacher Außenbelichtung. Dadurch ist die Belichtungszeit der Kamera so lang, dass, wie hier dargestellt, ein Verwischungseffekt schon bei geringen Geschwindigkeiten auftritt. Deutlich erkennbar ist dies bei dem Muster der Steine und den Stufenkanten.

Die erste Vermutung war, dass durch die erhöhte Geschwindigkeit die Bilder verwischen und ein sogenannter Blur Effekt auftritt. Diese Verwischung der Bilder ist abhängig von der Belichtungszeit und der relativen Bewegung der Kamera zu den aufgenommenen Objekten. Bei der ersten Aufnahme war die Verwischung deutlich sichtbar, wie auf dem beispielhaften Ausschnitt des Videomaterials in Abbildung 7 zu sehen ist. Der Blureffekt kann bei vergleichbaren Kameras und Geschwindigkeiten unterschiedlich auftreten, wenn die Kamera, wie das verbaute Modell am Auto von AutoNOMOS Labs, automatisch die Belichtungszeit an der Helligkeit der Umgebung anpasst.[Szeliski, 2010, S. 75ff] Mit Hilfe einer neuen Aufnahme bei helleren Lichtverhältnissen konnte diese Vermutung widerlegt werden, da hier selbst bei höherer Geschwindigkeit kein Blur Effekt auftrat und ORB SLAM 2 sich trotzdem nicht dauerhaft lokalisieren konnte.

Eine weitere Möglichkeit, warum der Algorithmus die Lokalisierung verliert, ist, dass die Bilder nicht in Echtzeit verarbeitet werden konnten. Die Idee kam dadurch, dass unter mehr Last, wie zum Beispiel bei der gleichzeitigen Aufnahme eines neuen Archivs und die Nutzung von ORB SLAM 2, der Algorithmus beobachtbar weniger Punkte fand und somit auch das Aufbauen einer lokalen Karte schlechter funktionierte.

Um das zu Überprüfen wurde ein Testszenario ausgewählt in dem das Auto aus einem Parkplatz herausrangiert und danach in die Anfahrt übergeht. Beim Rangieren ist die Szene, auf dessen Blick sich die Kamera richtet, vorwiegend unverändert, lediglich die Perspektive verändert sich stetig. Das Ergebnis wird in Abbildung 8 dargestellt. Dabei wurden in 0.1 Schritten von

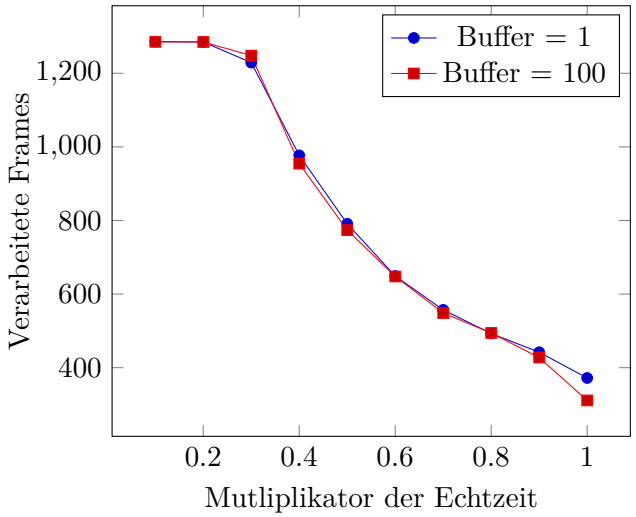


Abbildung 8: Die Gegenüberstellung der Abspielgeschwindigkeit des ROS Archivs und der verarbeiteten Frames. Man sieht, dass ab einen Faktor von 0.2 der Abspielgeschwindigkeit, die Verarbeitungsrate konvergiert. Die zwei unterschiedlichen Plots kennzeichnen die Größe des ROS Buffers beim Abonnenten des Bildkanals. Je größer der Buffer ist, desto mehr Bilder können im Buffer zur späteren Verarbeitung gelagert werden. Jedoch ist erkennbar, dass sich durch einen größeren Buffer keine signifikanten Unterschiede ergeben.

0.1 bis 1.0 der Echtzeit gemessen, wie viele Frames in ORB SLAM 2 verarbeitet wurden. Unabhängig von der Güte der Verarbeitung wird deutlich, dass je kleiner der Multiplikator wird, desto mehr Frames können verarbeitet werden. Dieses Szenario zeigt, dass die Verarbeitung der Bilder den fortlaufenden Prozess blockiert und somit bei nicht ausreichender Leistungsstärke des Prozessors oder der Graphikkarte das Framework nicht seine volle Leistung erzielen kann.

In ROS ist ein Buffersystem für Abonnenten eines Kanals integriert, dass eine vorher spezifizierte Anzahl an Elementen speichert, falls das aktuelle Modul blockiert. Um auszuschließen, dass der Grund für die geringe Verarbeitungsanzahl der Frames bei den echtzeitnahen Abspielgeschwindigkeiten ein zu klein bemessener Buffer ist, wurde dies ebenfalls untersucht und in der Grafik 8 eingetragen. Dabei ist zu erkennen, dass Abweichungen vorhanden sind, wohingegen der größere Buffer nicht deutlich besser abschneidet. Das lässt die Vermutung zu, dass die Verarbeitungszeit pro Frame ähnlich ist und der Buffer nicht mit eventuell unaufwendigen Frames abgebaut werden kann.

Trotz gleichem Bildmaterial und keinen Veränderungen der Abspielge-

schwindigkeiten, konnten Unterschiede in der Anzahl der verarbeiteten Frames festgestellt werden. Jedoch sind diese Unterschiede so gering, dass dadurch keine deutlichen Veränderungen zum hier präsentierten Ranking entstehen würden.

Abschließend ist die Framerate dafür ausschlaggebend gewesen, dass bei einer Fahrt entlang einer Straße nicht genügend Merkmale gefunden werden konnten, um das lokale Kartographieren vollständig durchführen zu können. Wenn die Abspielrate von ROS Archiven minimiert wird, kann die Qualität des Algorithmus vollständig entfaltet werden. Mit der oben genannten Spezifikation für diese Arbeit ist es angemessen, lediglich mit $\frac{1}{5}$ der Echtzeit zu arbeiten. Bei einer anderen Rechnerkonfiguration müsste neu bestimmt werden, ob eventuell die Echtzeit die volle Leistungskraft des Algorithmus entfaltet oder ob auch da die Abspielgeschwindigkeit herunter geregelt werden müsste.

4.2 Eingrenzung des Sichtfelds

Durch die Anbringung der Kamera im Seitenspiegel ist nicht nur die Umgebung im Bild zu erkennen, sondern auch das Auto per se. Da das Auto so viel Struktur aufweist, sodass der Algorithmus Merkmalspunkte darin erkennt, kommt es zu einem falschen Rückschluss über die Position der Merkmale im Raum: Die Kameraposition verändert sich, die Position der Merkmale im Auto hingegen nicht, somit müssen die gefundenen Merkmale in einer großen Entfernung liegen. Die Punkte im Raum, die sich nun unterhalb der Grundebene befinden, können sowohl nachfolgende Ausrichtungen neuer KeyFrames, als auch in der Weiterverarbeitung der Punkte, zum Beispiel zum Finden der Bodenebene, stören.

Um dies zu umgehen, wurde das Sichtfeld der Kamera eingeschränkt. Da sich das Auto im unteren rechten Bereich des Bildes befindet, wurde ausgehend vom linken oberen Eck das Sichtfeld auf $\frac{2}{3}$ der Höhe und der Breite verkürzt, wie in der Abbildung 9 beispielhaft dargestellt wurde. Durch diese Einschränkungen werden Informationen im roten und im grünen Abschnitt von der Straße nicht weiter beachtet. Der rote Part hindert den Algorithmus nicht, weil dieser Abschnitt größtenteils einen nahegelegenen Straßenabschnitt abbildet und dieser wenig Informationen über die Umgebung beinhaltet. Durch das Auslassen der grünen Felder werden wesentlich mehr Informationen eingebüßt, da gut erkennbare Merkmale nicht weiter zur Ausrichtung der Kamera dienen können. Als problematisch erweist es sich, wenn die Kamera nah an großen Objekten wie zum Beispiel parkenden Autos vorbei fährt. Dabei wird ein wesentlicher Teil des Bildes durch ein merkmalsarmes Objekt verdeckt, was dazu führen kann, dass der Algorithmus die Lokalisierung verliert und diese nicht mehr wiederfindet, weil alle bislang gefundenen Merkmale sich in einem Bereich befinden, der nicht mehr klassifiziert wird.

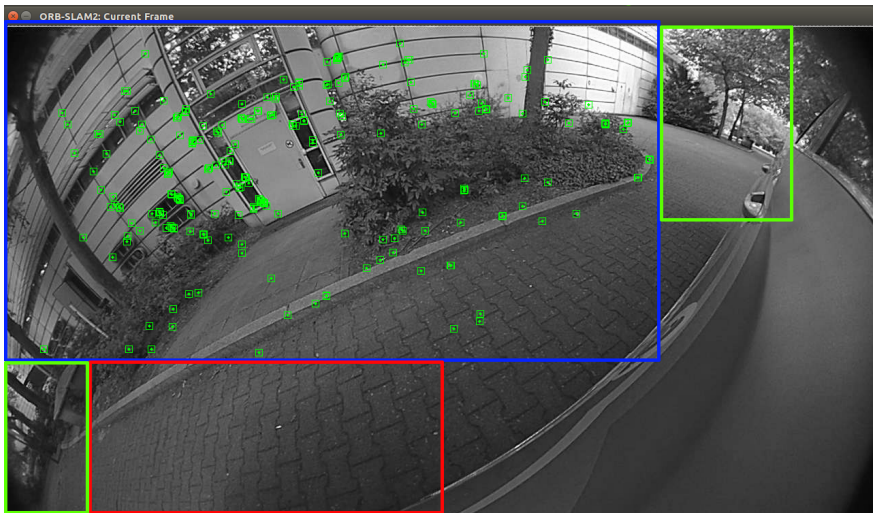


Abbildung 9: Ansicht des Sichtfeldes, in dem die Merkmale gesucht werden. Im blauen Part wird gesucht, der rote Part enthält wenig Informationen für den Algorithmus und die grünen Felder werden nicht berücksichtigt, obwohl diese wichtige Informationen enthalten.

Zur Verbesserung muss das Sichtfeld näher an die Kontur des Autos angeglichen werden. Auf Grund der Auslastung der Rechenleistung ist in diesem Projekt die Effektivität wichtiger als die Qualität. Deswegen wurden für weitere Tests der rechteckige, in Abbildung 9 blau dargestellte, Ausschnitt verwendet.

4.3 Krümmung durch tangentiale Verzerrung

Dieser Test fokussiert die Umgänglichkeit des Frameworks mit einer gekrümmten Kameralinse, auch Fischaugenoptik genannt. Durch die Krümmung der Linse wird ein größerer Öffnungswinkel der Kamera erzeugt, was dazu führt, dass die nahe Umgebung verzerrt vergrößert wird und die Ferne gestaucht wird. Weil die Bewegung zu den Seitenkameras nahezu orthogonal zur Ausrichtung ist, werden zwei Fluchtpunkte erfasst, aus denen sowohl die neue Umgebung auf der Bildfläche erscheint, als auch in einem anderen Fluchtpunkt verschwindet.

Die Kalibrierung der eingebauten Seitenkameras erwies sich als schwer, da Teile des Bildes vom Auto verdeckt werden oder durch die Autokontur unerreichbar waren. Somit entstand nur eine unzureichende Kalibrierung, die beim Einsatz die abgebildete Karte mutmaßlich verzerrte. Trotz der Probleme wurden zwei maßgebliche Kalibrierungen, die augenscheinlich nach der Rektifizierung unverzerrte Bilder hervorbrachten, mittels ORB SLAM 2 getestet. In der Tabelle 3 sind die verwendeten Kalibrierungsdaten zusam-

Kalibrierung	f_x	f_y	c_x	c_y	
Cal_1	394.999966	396.759510	603.059617	396.046422	
Cal_{12}	394.999966	396.759510	603.059617	396.046422	
Cal_{21}	279.477672	278.898255	636.855944	402.275101	
Cal_2	279.477672	278.898255	636.855944	402.275101	
Kalibrierung	k_1	k_2	p_1	p_2	k_3
Cal_1	-0.113782	0.007773	0.003508	0.002119	0.000000
Cal_{12}	0.261652	-0.082850	0.028070	-0.005376	0.000000
Cal_{21}	-0.113782	0.007773	0.003508	0.002119	0.000000
Cal_2	0.261652	-0.082850	0.028070	-0.005376	0.000000

Tabelle 3: Tabellarische Ansicht der verwerteten Kalibrierungen. Cal_1 und Cal_2 sind Werte aus vordefinierten Kalibrierungsprozessen und Cal_{12} und Cal_{21} sind Kombinationen aus Cal_1 und Cal_2. Die erste Tabelle enthält Werte für die Kameramatrix, wobei in der zweiten Tabelle die Kalibrierungskoeffizienten dargestellt werden.

mengefasst. Dabei sind die Kalibrierung Cal_1 und Cal_2 aus Kalibrierungsalgorithmen entstanden und Cal_{12} und Cal_{21} sind vom Autor kombinierte Kalibrierungen, um die Entzerrung der Bilder näher zu untersuchen.

In der ersten Tabelle aus der Tabellenansicht 3 ist die Kameramatrix der einzelnen Kalibrierungen aufgeführt. Die Zusammensetzung der Kamera aus den Werten stellt sich folgendermaßen dar:

$$\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{54}$$

In ORB SLAM 2 wird für die Entzerrung der Punkte die Standardbibliothek von OpenCV verwendet. Um die radiale und tangentiale Verzerrung aus der zweiten Tabelle aus der Tabellenansicht 3 zu berücksichtigen, werden in der Bibliothek die Parameter folgendermaßen verwertet[5]:

$$x_{\text{corrected}} = x + [2p_1xy + p_2(r^2 + 2x^2)] \tag{55}$$
$$y_{\text{corrected}} = y + [p_1(r^2 + 2y^2) + 2p_2xy]. \tag{56}$$

Nähere Erläuterungen zu den 5 Entzerrungsparametern $(\kappa_1, \kappa_2, \kappa_3, p_1, p_2)$ und den intrinsischen Parametern (f_x, f_y, c_x, c_y) sind im Abschnitt 2.1 zu finden.

Die Umgebung, in der die Verzerrungen getestet wurden, ist verkehrsähnlich, um gleichzeitig festzustellen, ob der Algorithmus auch auf die reale Praxis eines autonom fahrenden Autos angewendet werden kann. Im Verlauf

[5] `http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html`, Stand: 21.10.2016.

Abbildung 10: Ansicht der ersten Teststrecke in der Nähe der FU Berlin. Die Ansicht entstammt Google Maps .

des Tests wird eine verkehrsberuhigte Straße abgefahren, die mit Bäumen und parkenden Autos begrenzt ist. Die Bilder der Umgebung sind in der Abbildung 10 zu sehen.

Diese Strecke wurde wieder mittels ROS aufgenommen und mit den einzelnen Kamerakalibrierungen in ORB SLAM 2 verarbeitet. Die Ergebnisse können in den Plots den Abbildungen 11 und 12 eingesehen werden. Die Verzerrung aller lokalen Streckenaufnahmen ist deutlich sichtbar. Jedoch sind Cal_1 und Cal_{12} die besten Kalibrierungen, weil diese zumindest den Kreis schließen konnten und somit ein vermeintlicher Drift behoben werden konnte. Wie in der Abbildung 10 zu sehen ist, kann die Strecke nicht die eingetragene Wölbung vorweisen. In Cal_2 wurde kurz vor dem Schließen des Kreises der relative Bezug verloren. Die Cal_{21} baut eine nicht nachvollziehbare Karte auf.

Wie schon erwähnt, stellt die lokale Karte der KeyFrames eine sehr verzerrte Wirklichkeit dar. Das kann sowohl an der Kalibrierung liegen, welche durch Veränderungen ein signifikant verschiedenes Bild liefert, als auch an der Verarbeitung der Punkte in das eigene Kartografierungssystem. Um die Punkte zu entzerren wird eine OpenCV Funktion verwendet.[6] Diese Funktion transformiert Punkte einer verzerrten Ebene mittels der Kameramatrix und Verzerrungskoeffizienten in eine unverzerrte Ebene. Da ORB SLAM 2 noch mit OpenCV 2.x arbeitet, kann eine mögliche Verbesserung sein, das Projekt von OpenCV 2.x auf OpenCV 3.x umzustellen.

Durch die eigene Programmierung von vielen Modulen innerhalb von ORB SLAM 2 sind nicht viele OpenCV Klassen vorhanden, die von der Umstellung von OpenCV 2 zu 3 berücksichtigt werden müssten. Somit ist die Umstellung des Projekts mit einem kleinen Wegweiser, der von OpenCV vorbereitet wurde, schnell durchführbar.[7] Wenn man jedoch weiterhin ROS benutzen möchte, muss man beachten, dass nicht nur das OpenCV 3 Paket in ROS installiert ist, sondern auch, dass die Funktionen wie zum Beispiel *cv_bridge* auch nach OpenCV 3 portiert werden.

[6] `http://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html#undistort`

[7] `http://docs.opencv.org/3.1.0/db/dfa/tutorial_transition_guide.html`

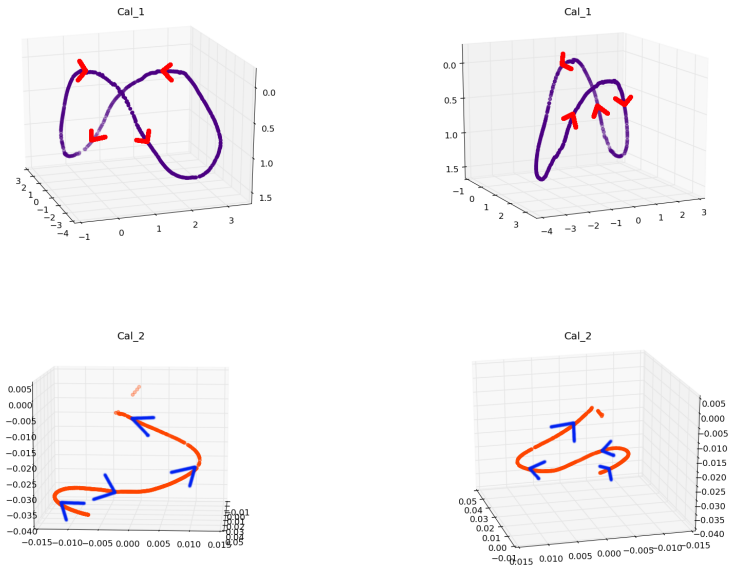


Abbildung 11: Die geplotteten KeyFrames der einzelnen Durchläufe mit den programmatischen Kalibrierungen aus den Tabellen 3. Die Pfeile zeigen die Fahrtrichtung.

Trotz der konzeptionellen Umstellungen innerhalb von OpenCV in Bezug auf das Fischaugen Modell[8] und der Portierung des Projekts nach OpenCV 3 konnte das lokale Kartographieren nicht verbessert werden. Das Gegenteil trat ein: der Algorithmus verlor kurz nach dem Start die Orierntierung, hat im Allgemeinen weniger Merkmale gefunden und verwarf frühzeitig schon aufgenommene KeyFrames. Vor allem beim letzten Punkt ist der Grund nicht klar erkennbar. Eine Möglichkeit ist, dass Merkmale aussortiert werden, wenn der Algorithmus konstant die aktuelle Position falsch bemisst und somit übereinstimmende Merkmale als falsch positioniert betrachtet. Diese für den Algorithmus falsch positionierten Merkmale werden als Fehler angesehen und gelöscht. Dadurch entfallen auch KeyFrames, die nicht mehr genug neue Merkmale zur lokalen Karte beitragen.

Eine weitere Erklärung für die sehr verzerrte Kartographierung ist, dass die Kalibrierung der Kamera, besonders im Bezug auf die Fokallänge, nicht mit der Längeneinheit des Algorithmus abgestimmt ist. Szelski erwähnt das als eine der größten Fehleranfälligkeiten in der Kalibrierung von Kameras in der Computer Vision. [Szeliski, 2010, S. 53] Dahingehend ist auch ein Fehler zu beobachten: Je länger der Algorithmus im Betrieb ist, desto größer wird die Skalierung der Umgebung. Daraus kann folgen, dass die Kalibrierung

[8] `http://docs.opencv.org/3.1.0/db/d58/group__calib3d__fisheye.html`

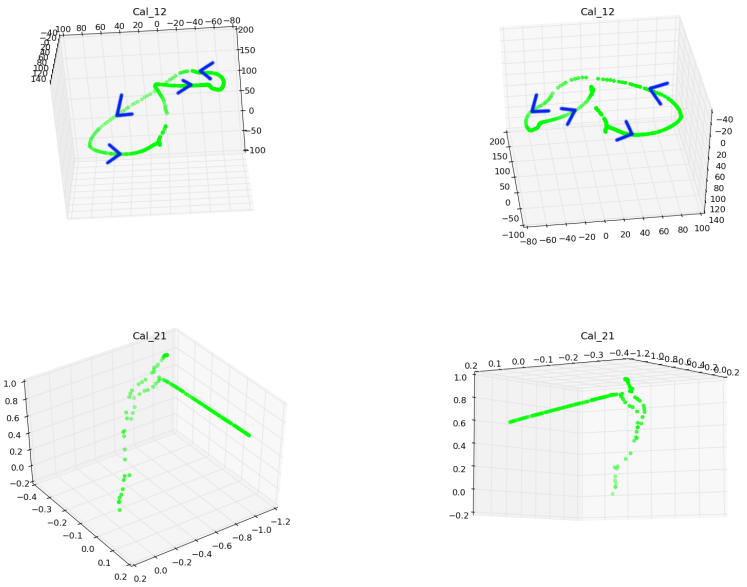


Abbildung 12: Der zweite Teil der geplotteten KeyFrames aus den Tabellen 3. Die Pfeile zeigen die Fahrtrichtung an, außer beim letzten Durchlauf, wo die Fahrt nicht mehr zu rekonstruieren ist.

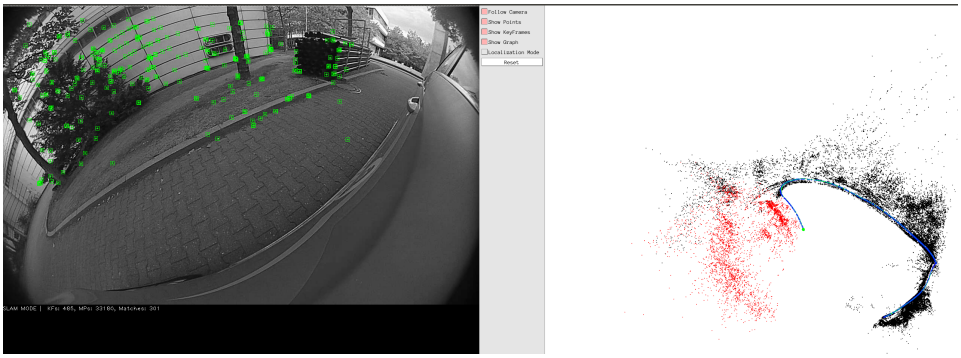
aus der Sicht von ORB SLAM 2 immer weniger mit der aktuellen Kameraeinstellung übereinstimmt und schließlich die eigene Karte noch verzerrter dargestellt wird.

Schlussendlich konnte nicht ermittelt werden, warum die Kartographierung dermaßen verzerrt ausfällt. Im veröffentlichten wissenschaftlichen Beitrag [Mur-Artal et al., 2015] wird dieses Thema nicht im Detail behandelt und auf der Open Source Seite des Projekts äußern sich lediglich Nutzer von ORB SLAM 2 über die Vereinbarkeit von Algorithmus und Fischaugenkameras. Dem Gegenüber steht ein Projekt von [Urban et al., 2016], welches ORB SLAM 2 mit mehreren Kameras gleichzeitig ausführt und somit eine rundum Kartographierung der Umgebung ermöglicht. Dafür wurden auch Fischaugenkameras verwendet, die in der Videodemonstration [9] eine unverzerrte Umgebung abbildeten. Allerdings konnte dieses Projekt im Rahmen dieser Arbeit nicht weiter untersucht werden.

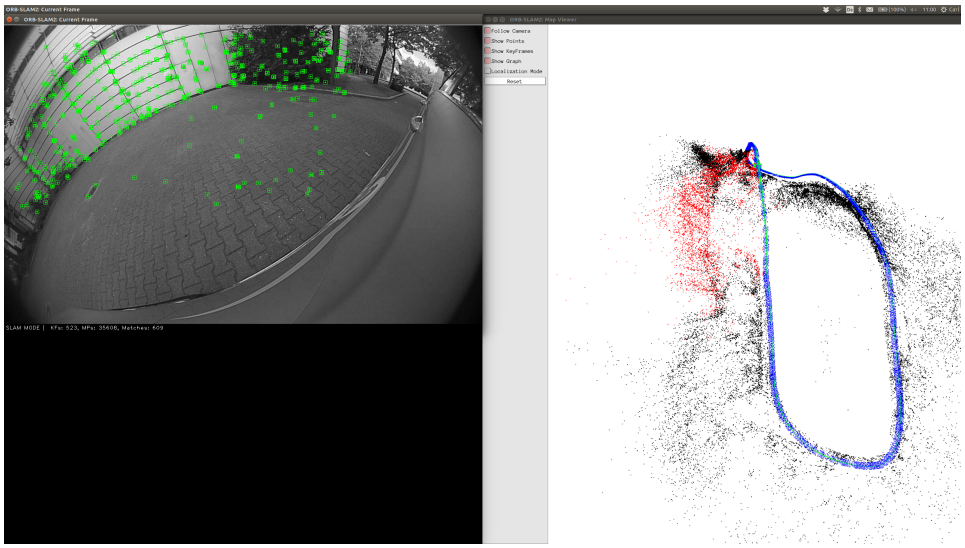
4.4 Schließen von Kreisen

Das Schließen eines Kreises funktioniert sehr gut. Selbst im verzerrten Szenario, in dem sich ORB SLAM 2 im Test aus dem vorherigen Abschnitt

[9] `https://youtu.be/ggZqsiePUq8`, Stand 21.10.2016



(a) Die abgefahrene Strecke wird in gekrümmter Form, bedingt durch die intrinsische Kalibrierung der Kamera, dargestellt und in wenigen KeyFrames wird erkannt, dass ein Kreis gefahren wurde.



(b) Der Kreis wurde erkannt und geschlossen. Daraufhin wird nahezu der ganze Graph optimiert.

Abbildung 13: Prozedur des Kreisschließens im ORB SLAM 2 Algorithmus.

befindet, können zwei KeyFrames als gleich erkannt werden, obwohl beide auf der lokalen Karte weit voneinander entfernt sind. Auf der Abbildung 13 sieht man den angezeigten Vorgang kurz vor und nach dem Kreisschließen. Der genaue interne Ablauf wird in Abschnitt 3.3 erläutert. Es werden, um den Kreis zu schließen, nicht alle KeyFrames optimiert, sondern lediglich solche, die Ähnlichkeit mit den KeyFrames haben, welche die Schließung initiieren. Dadurch entsteht vermutlich der markante Knick in der Mitte der Strecke.

5 Ausblick

ORB SLAM 2 funktioniert sehr gut in einer statischen, unveränderten Umgebung und ist, im Gegensatz zu PTAM, darauf ausgelegt, durch die effiziente und nicht redundante Speicherung von KeyFrames, über einen längeren Zeitraum performant zu arbeiten. Um in realen Umgebungen eingesetzt zu werden, bedarf es einigen Verbesserungen hinsichtlich der Einbindung in komplexen Werkzeugketten und der Umstellung von einem lokalen in einen globalen Kontext. Beispielhaft werden in diesem Kapitel Verbesserungsvorschläge diskutiert.

5.1 Teil eines komplexen Arbeitsablaufs

Damit der angewandte Algorithmus eine Komponente eines größeren Arbeitsprozesses wird, wie zum Beispiel als Teil des Prozesses einen Parkplatz für ein Auto zu finden, bedarf es neben der Funktionalität, die Umgebung zu Kartographieren und sich relativ zu positionieren, zusätzlich an Möglichkeiten, sowohl die gemessenen Daten in ORB SLAM 2 einzuspeisen, als auch die extrahierten Erkenntnisse leicht weiter zu verarbeiten.

Der Videoeingang, wie im Abschnitt 3.3 näher dargestellt, wird durch die Anschlussmöglichkeit mittels ROS gewährleistet. Leider basiert die ROS Schnittstelle auf einer sehr alten Version und wird nicht aktualisiert. Folglich ist der Build Prozess mit catkin, das neuste Build System in ROS, noch nicht im Hauptentwicklungsstrang integriert. Hingegen, und das ist bei einigen Kritikpunkten an ORB SLAM 2 ein Vorteil, ist der Code öffentlich verfügbar. Das heißt die Umstellung auf catkin wurde schon von anderen Nutzern zur Verfügung gestellt.[10] Falls noch weitere Daten in den ORB SLAM 2 Algorithmus einfließen sollen, wie zum Beispiel GPS Daten, um die lokale Karte im globalen Kontext einzubetten, muss diese Schnittstelle noch hinzugefügt werden. Im Gegensatz zu den sich stetig aktualisierenden Daten, wird die intrinsische Kameraeinstellungen von einer separaten Datei ausgelesen.

Im Kontrast dazu ist die Weiterverarbeitung der SLAM Erkenntnisse wesentlich schwieriger. Nach dem Schließen des Programms wird eine Datei angelegt, die ausschließlich die Trajektorie der KeyFrames speichert. Das wirft mehrere Probleme auf:

- Die Weiterverarbeitung der Trajektorie kann nur offline, nachdem alle Daten verarbeitet wurden, vollzogen werden.
- Die Merkmale zu den einzelnen Frames werden nicht persistent gespeichert und auch nicht für eine Weiterverarbeitung aufbereitet. Die Weiterverarbeitung der erkannten Merkmalspunkte beispielsweise zur

[10] `https://github.com/raulmur/ORB_SLAM2/pull/52`

Objekterkennung kann nicht ohne eine Weiterentwicklung des Projekts durchgeführt werden.

In der Gemeinschaft, die sich um das Open Source Projekt versammelt hat, wurde der Ansatz zur persistenten Speicherung der Merkmale schon umgesetzt[11]. Jedoch ist lediglich angedacht, dass die Daten wieder mit dem ORB SLAM 2 Programm geöffnet werden und die gewonnenen Daten nur innerhalb des Programms wiederverwendbar sind.

Der Ansatz, dass ORB SLAM 2 als Modul einer Werkzeugkette eingebettet wird, ist in diesem Projekt nicht geplant. Die Realisierung ist dementsprechend auch schwer, da mehrere Kanäle als Ausgang aufgebaut werden müssten. Zum einen wird ein Kanal benötigt, der alle neuen Erkenntnisse aufnimmt, wie zum Beispiel die neuen KeyFrames und die angrenzenden Merkmalspunkte, die danach weiter verarbeitet werden könnten. Zum anderen müsste ein Kanal aufgebaut werden, der die Aktualisierungen überträgt, zum Beispiel Optimierungen an den KeyFrames und gelöschte KeyFrames.

Eine weitere Entscheidung, die man in einem Arbeitsprozess mit bedenken muss, ist, ob die Datenaustauschrate zwischen den Modulen gering zu halten ist oder ob die Verarbeitung so wenig redundant wie möglich sein soll. Ist die Prämisse die Datenraten zwischen den Modulen gering zu halten, zum Beispiel indem nur die Veränderungen der KeyFrames übertragen werden, dann müssen in dem aufnehmenden Modul Möglichkeiten implementiert sein, Schritte zur Auswertung dieser Daten selbst vornehmen zu können, wie unter anderem die Ausrichtung der angrenzenden KeyFrames oder der Umgang mit den bereits gewonnenen Merkmalen. Das bedeutet, dass Teile des ORB SLAM 2 Algorithmus doppelt benutzt werden. Wenn die Verarbeitung effizient realisiert werden soll, dann muss jede einzelne Anweisung über die Datenkanäle übertragen werden, das heißt sowohl die Veränderungen der KeyFrames, als auch der sich verändernde Covisibility Graph und die Veränderungen an den gefundenen Merkmalen.

5.2 Einbringen von globalen Informationen

In einem sich stetig ändernden Szenario, wie zum Beispiel einer Autofahrt, kann es passieren, dass ORB SLAM 2 die Lokalisierung verliert und sich nicht mehr relokalisieren kann. Wie im Abschnitt 4.2 analysiert wurde, ist das in den Experimenten häufiger aufgetreten, wenn zum Beispiel Autos einen großen Teil der Sichtfläche verdecken. Die Relokalisierung ist darauf ausgelegt, dass zu einem unbestimmten Zeitpunkt schon erkannte Merkmale wiedergefunden werden, an Hand derer sich der Algorithmus ausrichten kann. Jedoch tritt bei einer Autofahrt in einer nicht vorher kartographierten Umgebung dieser Ereignis nicht ein.

[11] `https://github.com/raulmur/ORB_SLAM2/issues/19`

Um dem entgegenzuwirken muss die lokal aufgebaute Karte mit globalen Informationen, wie zum Beispiel GPS Daten verknüpft werden. Dies kann dazu führen, dass die Relokalisierung bei einer zu weiten Entfernung der bereits verarbeiteten KeyFrames nicht weiter berücksichtigt wird, sondern der Algorithmus sich in den Zustand des Neustarts versetzt und den aktuellen Frame als Initialisierungsframe verwendet. In der aktuellen Funktionsweise von ORB SLAM 2 ist es nicht vorgesehen, dass der Algorithmus sich selbstständig neustartet, wenn erst einmal KeyFrames in den Prozess des Trackings aufgenommen wurden. Ohne globale Informationen ist es eine schwierige Aufgabe, die bereits gesammelten Informationen abzuspeichern und gegebenenfalls effizient wiederzuverwenden.

Im Bezug auf die Speicherung der vorhandenen KeyFrames können globale Informationen helfen, über einen längeren Zeitraum große Mengen an Daten zu speichern. Wenn in der Datenbank für die Ähnlichkeiten zusätzlich globale Informationen verarbeitet werden, können viele KeyFrames unberücksichtigt bleiben, die durch eine größere Entfernung zum aktuellen Bild nicht relevant sind. Somit würden globale Informationen vermutlich den lokalen Kontext komplexer werden lassen, hingegen könnten Probleme der Langzeitnutzung dadurch verhindert werden.

5.3 Gemeinsame Verarbeitung mehrerer Kameras

Besonders im Szenario eines Autos ist die Idee naheliegend, mehrere Kameras miteinander zu verbinden und die unterschiedlichen gewonnenen Erkenntnisse aus den verschiedenen Kameras in einer lokalen Karte zu verbinden. Dafür hat ORB SLAM 2 wenige Strukturen bereits integriert. Zum Beispiel ist an jedem KeyFrame im Programmcode neben der Ausrichtung und Position auch die Kamerakalibrierung angeheftet. Das erleichtert den Prozess jeden KeyFrame individuell zu behandeln. Jedoch ist dieses Vorgehen nicht konsequent im Algorithmus wiederzufinden.

Im Jahr 2016 wurde in einer wissenschaftlichen Arbeit von [Urban et al., 2016] diese Idee der Benutzung mehrerer Kameras umgesetzt. Dabei geht es konkret um ein festes System aus mehreren Fischaugenkameras, die in einer weitestgehend starren Umgebung mit ORB SLAM 2 kartographieren. Dieses Projekt ist auch auf Github verfügbar.[12] Auch wenn es in dieser Arbeit nicht weiter berücksichtigt werden konnte, wäre es sowohl im Einsatz der Fischaugentechnologie als auch in der Kombination der verschieden ausgerichteten Kameras ein spannendes Projekt für das autonome Fahren.

[12] `https://github.com/urbste/MultiCol-SLAM`.

6 Fazit

Im Rahmen dieser Arbeit wurde untersucht, wie praktikabel einsetzbar ORB SLAM 2 im Projekt AutoNOMOS Lab ist. Ziel dabei war es, dass der eingesetzte Algorithmus sowohl in Echtzeit anwendbar ist und iterativ mit Hilfe der eingehenden Daten die Umgebung analysieren kann.

ORB SLAM 2 ist ein Algorithmus, der basierend auf Bundle Adjustment, die Kameras iterativ ausrichtet, indem die relative Position zum gefundenen Merkmal aus der Umgebung bestimmt wird. Bundle Adjustment ist ein Prinzip, dass der Epipolargeometrie entstammt und die relative Positionierung und Ausrichtung von n-Kameras ermöglicht. Um aus den Kameraeinstellungen die Punktvermutungen zu vereinen, werden diese Punkte im Raum trianguliert.

ORB SLAM 2 setzt sich schon im Namen aus zwei Themengebieten zusammen, zum einen ORB und zum anderen SLAM. ORB ist ein Algorithmus zum schnellen Finden von robusten Merkmalen. SLAM hingegen ist ein weites Problemfeld, dass sich mit der gleichzeitigen Lokalisierung und Kartographierung der Umgebung beschäftigt. Zusammengefasst ermöglichen beide Prinzipien, dass ORB SLAM 2 ausschließlich auf Kamerabildern basierend eine visuelle Odometrie aufbaut und die Umgebung mit ORB Merkmalen in einer lokalen Karte abbildet.

Es wurde gezeigt, dass es nicht möglich war, den Algorithmus in Echtzeit einzusetzen. Das heißt aber nicht, dass ORB SLAM 2 prinzipiell nicht echtzeitfähig ist. Mit einer besseren Hardware und eventuell einer teilweisen Auslagerung des Algorithmus auf die Grafikkarte, kann eine Echtzeitfähigkeit unter Umständen erreicht werden. Weiterhin ist klar geworden, dass durch die Eingrenzung des Sichtfelds der Algorithmus zuverlässiger die Umgebung kartographiert, da Elemente vom Auto das Bild somit nicht verdecken und die Ergebnisse des Algorithmus nicht verfälscht werden. In der praktischen Anwendung von ORB SLAM 2 konnte außerdem festgestellt werden, dass die Darstellung der Karte durch scheinbare Umstimmigkeiten zwischen Kalibrierung und ORB SLAM 2 verzerrt dargestellt wird. Trotz verschiedener Ansatzpunkte, wie zum Beispiel unterschiedlicher Kalibrierungen einzusetzen oder das Projekt von OpenCV 2.x auf OpenCV 3.x zu portieren, wurde im Rahmen dieser Arbeit keine zufriedenstellende Lösung oder Erklärung für dieses Problem gefunden. Abschließend wurde gezeigt, dass trotz verzerrter Darstellung das Schließen eines Kreises durchgeführt werden kann.

Die Frage nach der tatsächlichen Einsetzbarkeit von ORB SLAM 2 muss also nach derzeitigem Stand verneint werden. Die vom Algorithmus erstellte Karte ist fehlerhaft und kann nicht, wegen der mangelnden Extrahierbarkeit, weiterverarbeitet werden. Im Ausblick dieser Arbeit wird herausgearbeitet, wie der Ausbau des Dateneingangs und -ausgangs von ORB SLAM 2 realisiert werden müsste. In diesem Punkt ist festzuhalten, dass sich die Karte stetig aktualisiert und dahingehend Verarbeitungsstrategien für die

angrenzenden Module entwickelt werden müssen. Weiterhin ist es notwendig, globale Informationen in ORB SLAM 2 einfließen zu lassen, um zum Beispiel sich trotz nicht vorhandener Informationen über die Umgebung relokalisieren zu können. Außerdem wird ein Projekt kurz vorgestellt, welches die karthographierten Daten mehrerer Kameras verbindet und aus allen Eingaben eine ORB SLAM 2 Karte aufbaut. Das lässt die Vermutung zu, dass das vorgestellte Programm nicht gänzlich unkompatibel in Bezug auf Weitwinkelkameras ist.

Trotz der Vielzahl der aufgeführten Probleme sind grundlegende Errungenschaften hervorzuheben: Die Kartographierung der Umgebung funktioniert und die visuelle Odometrie ist, abgesehen von der räumlichen Verzerrung, akkurat. Das Schließen des Kreises konnte trotz der großen Entfernung zwischen den Kandidaten vollzogen werden. Schlussendlich ist das größere Ziel hinter diesem Projekt, mittels günstiger Kameratechnik aufwendige Radartechnologie zu ersetzen, sehr nah.

Literatur

[Agarwal et al., 2009] Agarwal, S., Snavely, N., Simon, I., Seitz, S. M., and Szeliski, R. (2009). Building rome in a day. In *Twelfth IEEE International Conference on Computer Vision (ICCV 2009)*, Kyoto, Japan. IEEE.

[Calonder et al., 2010] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. In *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, pages 778–792, Berlin, Heidelberg. Springer-Verlag.

[Faugeras and Lustman, 1988] Faugeras, O. D. and Lustman, F. (1988). Motion and structure from motion in a piecewise planar environment. *Intern. J. of Pattern Recogn. and Artific. Intelige.*, ??(3):485–508.

[Gálvez-López and Tardós, 2012] Gálvez-López, D. and Tardós, J. D. (2012). Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197.

[Hane et al., 2015] Hane, C., Sattler, T., and Pollefeys, M. (2015). Obstacle detection for self-driving cars using only monocular cameras and wheel odometry. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 5101–5108.

[Hartley and Zisserman, 2003] Hartley, R. and Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition.

[Hartley, 1997] Hartley, R. I. (1997). In defense of the eight-point algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(6):580–593.

[Horn, 1987] Horn, B. K. P. (1987). Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642.

[Klein and Murray, 2007] Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan.

[Konderlink and van Doorn, 1991] Konderlink, J. J. and van Doorn, A. J. (1991). Affine structure from motion. *Journal of the Optical Society of America*, 8(2):377–385.

[Longuet-Higgins, 1987] Longuet-Higgins, H. C. (1987). Readings in computer vision: Issues, problems, principles, and paradigms. In Fischler, M. A. and Firschein, O., editors, *Nature*, volume 293, pages 61–62, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Miksik and Mikolajczyk, 2012] Miksik, O. and Mikolajczyk, K. (2012). Evaluation of local detectors and descriptors for fast feature matching. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 2681–2684.

[Mur-Artal et al., 2015] Mur-Artal, R., Montiel, J. M. M., and Tardós, J. D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163.

[Mur-Artal and Tardós, 2014] Mur-Artal, R. and Tardós, J. D. (2014). Fast relocalisation and loop closing in keyframe-based slam. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 846–853.

[Pollefeys et al., 2008] Pollefeys, M., Nistér, D., Frahm, J.-M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S.-J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Yang, Q., Stewénius, H., Yang, R., Welch, G., and Towles, H. (2008). Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2):143–167.

[Rosten and Drummond, 2006] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*, ECCV'06, pages 430–443, Berlin, Heidelberg. Springer-Verlag.

[Rosten et al., 2010] Rosten, E., Porter, R., and Drummond, T. (2010). Faster and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(1):105–119.

[Rublee et al., 2011] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2564–2571, Washington, DC, USA. IEEE Computer Society.

[Szeliski, 2010] Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition.

[Thrun and Leonard, 2008] Thrun, S. and Leonard, J. J. (2008). *Simultaneous Localization and Mapping*, chapter 37, pages 13–41. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Triggs et al., 2000] Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W. (2000). Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, pages 298–372, London, UK, UK. Springer-Verlag.

[Urban et al., 2016] Urban, S., Wursthorn, S., Leitloff, J., and Hinz, S. (2016). Multicol bundle adjustment: A generic method for pose estimation, simultaneous self-calibration and reconstruction for arbitrary multi-camera systems. *International Journal of Computer Vision*, pages 1–19.

[Xiao et al., 2008] Xiao, J., Fang, T., Tan, P., Zhao, P., Ofek, E., and Quan, L. (2008). Image-based faÇade modeling. In *ACM SIGGRAPH Asia 2008 Papers*, SIGGRAPH Asia '08, pages 161:1–161:10, New York, NY, USA. ACM.

[Xu and Zhang, 1996] Xu, G. and Zhang, Z. (1996). *Epipolar Geometry in Stereo, Motion, and Object Recognition: A Unified Approach*. Kluwer Academic Publishers, Norwell, MA, USA.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben. Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

Datum Unterschrift