ALBERT-LUDWIGS-UNIVERSITÄT
FREIBURG

INSTITUT FÜR INFORMATIK

Arbeitsgruppe Autonome Intelligente Systeme

Prof. Dr. Wolfram Burgard

Techniken für bildbasiertes SLAM unter Verwendung von Lagesensoren

Diplomarbeit

Bastian Steder

April 2007

ALBERT-LUDWIGS-UNIVERSITÄT FREIBURG
FAKULTÄT FÜR ANGEWANDTE WISSENSCHAFTEN

Institut für Informatik
Arbeitsgruppe Autonome Intelligente Systeme
Prof. Dr. Wolfram Burgard

Techniken für bildbasiertes SLAM unter Verwendung von Lagesensoren

Diplomarbeit

Verfasser: Bastian Steder

Abgabedatum: 10. April 2007

Erstgutachter: Prof. Dr. Wolfram Burgard
Zweitgutachter: Prof. Dr. Bernhard Nebel
Betreuer: Dr. Giorgio Grisetti

Erklärung

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

(Bastian Steder)
Freiburg, den 10. April 2007

Danksagung

An dieser Stelle möchte ich mich bei all denen bedanken, die mich bei der Erstellung dieser Diplomarbeit unterstützt haben. Großer Dank geht an Prof. Dr. Wolfram Burgard dafür, dass er mir die Chance zu dieser Arbeit gegeben hat. Ich danke Prof. Dr. Bernhard Nebel dafür, dass er sich als Zweitgutachter zur Verfügung gestellt hat. Weiterhin danke ich Dr. Giorgio Grisetti vielmals für die Betreuung meiner Arbeit und dafür, dass er stets mit Rat und Tat zur Verfügung stand. Vielen Dank an Slawomir Grzonka für seine enorme Hilfsbereitschaft und für die Durchsicht der schriftlichen Ausarbeitung. Auch bei Axel Rottmann, Cyrill Stachniss, Rainer Kümmerle und Patrick Pfaff möchte ich mich bedanken, für allgemeine Anregungen oder Unterstützung bei Experimenten, entweder durch tatkräftige Hilfe oder durch das zur Verfügung stellen von benötigter Hardware. Insgesamt verdient der komplette Lehrstuhl meinen Dank, für die freundliche und lockere Atmosphäre und allgemeine Hilfsbereitschaft.

Kurzfassung

Eine der Grundvoraussetzungen für ein wirklich autonom agierendes System ist die Fähigkeit, eine Karte seiner Arbeitsumgebung autonom zu erstellen. Für die Kartographierung und zeitgleiche Lokalisierung (*SLAM: Simultaneous Localization and Mapping*) werden in der Regel Lasersensoren oder hochauflösende Kameras in Kombination mit Odometrie verwendet. Im Kontext von autonom agierenden Flugrobotern, wie beispielsweise einem Miniaturhubschrauber oder -zeppelin ist hingegen keine Odometrie verfügbar. Bedingt durch die geringe Nutzlast lassen sich weiterhin nur wenige und zudem oftmals qualitativ schlechtere Sensoren im Roboter unterbringen.
In dieser Arbeit wird aufbauend auf den Informationen eines nach unten gerichteten Stereokamerasystems und eines Lagesensors ein Verfahren zum 3D-SLAM entwickelt. Dazu werden Techniken zur inkrementellen Schätzung der Bewegung des Fluggerätes, sowie zum Finden von geschlossenen Schleifen (*loop-closings*) vorgestellt. Die wahrscheinlichste Trajektorie des Roboters wird dann mittels eines Graph-basierten Optimierungsalgorithmus geschätzt. Diese wird anschließend für die Konstruktion einer texturierten Oberflächenkarte des überflogenen Gebietes verwendet. In verschiedenen Experimenten wird gezeigt, dass dieses System auch unter schwierigen Bedingungen und eingeschränkter Sensorenverfügbarkeit konsistente Trajektorien und Karten erzeugt.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Zielsetzung . . . 2
- 1.2 SLAM . . . 3
- 1.3 Visueller SLAM . . . 5
- 1.4 Notationen . . . 5
- 1.5 Aufbau . . . 6

2 Verwandte Arbeiten **7**

3 Grundlagen zu optischen Abbildungen **11**
- 3.1 Die Lochkamera . . . 11
- 3.2 Interne Kameraparameter . . . 12
- 3.3 Reale Kameras . . . 13
- 3.4 Externe Kameraparameter . . . 15
- 3.5 3D-Daten aus Kamerabildern . . . 16
- 3.6 Kameradaten aus 3D-Punkten . . . 17

4 Merkmalsextraktion **19**
- 4.1 Der Moravec-Detektor . . . 21
- 4.2 Der Harris-Detektor . . . 22
- 4.3 Kanade-Lucas-Tomasi . . . 25
- 4.4 Der Skalierungsraum . . . 25
- 4.5 Der Harris-Detektor für multiple Skalierungen . . . 26
- 4.6 Hesse-Matrix-basierte Detektoren . . . 27
- 4.7 Der SIFT-Detektor . . . 28
- 4.8 Der SURF-Detektor . . . 30
- 4.9 Beschreibung und Vergleich von Merkmalen . . . 31
- 4.10 SIFT-Deskriptorvektoren . . . 32
- 4.11 SURF-Deskriptorvektoren . . . 33

5 Die Merkmalskarte **37**
- 5.1 Merkmale im Weltsystem . . . 37
- 5.2 Erstellung und Visualisierung der Karte . . . 38
- 5.3 Kandidaten für den Merkmalsvergleich . . . 38

6 Rekonstruktion der Kameraposition **41**
6.1 RANSAC & PROSAC 42
6.2 Der Algorithmus - Idee 43
6.3 Der Algorithmus im Detail 44
6.4 Komplexität 50

7 Optimierung der Trajektorie **53**
7.1 Definitionen 54
7.2 Die Parametrisierung 55
7.2.1 Inkrementelle Posen 55
7.2.2 Baum-basierte inkrementelle Posen 56
7.3 Optimierung bezüglich inkrementeller Posen 57
7.4 Optimierung mit Baum-basierter Parametrisierung 61
7.5 Anwendung für visuellen SLAM 62

8 Experimente **65**
8.1 Anwendung im Außenbereich 65
8.2 Anwendung im Innenbereich 68
8.3 Anwendung auf einem echten Fluggerät 74
8.4 Leistungsanalyse 76

9 Zusammenfassung **79**
9.1 Mögliche Problemquellen 79
9.2 Ausblick 80

A Berechnung der Kameraposition **83**
A.1 Berechnung der Kamerahöhe 83
A.2 Berechnung des Drehwinkels 85
A.3 Berechnung der X,Y-Koordinate der Kamera 86

B Berechnung von Tiefeninformationen **87**

C Verwendete Hardware **91**
C.1 Stereokamerasystem 91
C.2 Lagesensor 92
C.3 Miniaturzeppelin 93

Quellenangabe **94**

1 Einleitung

Getrieben durch immerwährenden technologischen Fortschritt gewinnt die Robotik im alltäglichen Leben mehr und mehr an Bedeutung. Dabei stehen verschiedene Anwendungen im Blickwinkel der Forschung. Roboter werden als Museumsführer genutzt, spielen Fußball, mähen Rasen oder sind auch für sehr ernste Anwendungen wie das Entschärfen von Minen oder die Suche nach Überlebenden in Unglückssituationen konzipiert.

Beispiele verschiedener mobiler Roboter

Die Fähigkeit sich zu orientieren ist dabei eine der elementarsten Grundvoraussetzungen für einen autonom agierenden Roboter. Hierzu ist eine interne Repräsentation seiner Umgebung in Form einer Karte nötig. Diese ist jedoch in vielen praktischen Anwendungen nicht im Voraus verfügbar. Der Roboter startet also auf einer unbekannten Position in einer unbekannten Umgebung und muss die Karte auf dem Weg selbstständig aufbauen. Dieses Problem, eine Karte seiner Umgebung aufzubauen und sich gleichzeitig darin zu lokalisieren, ist als *SLAM, Simultaneous Localization and Mapping (Simultane Lokalisierung und Kartengenerierung)* bekannt.

Dank des technischen Fortschritts werden immer kleinere und leichtere Roboter möglich. Die Idee, auch flugfähige Roboter zu entwickeln ist daher naheliegend. Auf Miniaturhubschraubern oder -zeppelinen beruhende Robotersysteme könnten auch in unwegsame Gebiete vordringen oder zur Überwachung eingesetzt werden.


Beispiele möglicher Plattformen für flugfähige Roboter

Für solche Fluggeräte stellt das SLAM-Problem eine besondere Herausforderung dar. Durch ihre geringe Nutzlast sind diese in der Wahl der verwendeten Sensoren stark eingeschränkt. Außerdem weisen sie komplexere Bewegungsmodelle als beispielsweise Fahrzeuge auf, da sie sich in allen drei Dimensionen frei bewegen können. Auch auf eine Odometrie, wie sie bei Radrobotern üblicherweise zur Verfügung steht, muss hier verzichtet werden.

1.1 Zielsetzung

Ziel dieser Arbeit war es, für ein solches Fluggerät ein System zum dreidimensionalen SLAM zu entwickeln. Das behandelte Szenario zieht als verwendete Sensoren ein nach unten gerichtetes (Stereo-)Kamerasystem, sowie einen Lagesensor, der die dreidimensionale Orientierung des Vehikels misst, in Betracht.

Wichtige Fragen, die auf dem Weg zu einem solchen System beantwortet werden müssen sind beispielsweise:

- Mit welchen Mitteln werden Informationen aus den Rohdaten extrahiert?
- Wie werden diese Daten verarbeitet um eine robuste und fehlertolerante Schätzung der Posen zu erhalten?
- Welchen Aufbau soll die resultierende Karte haben?
- Welcher Ansatz zur Lösung des SLAM-Problems soll eingesetzt werden?

Im Verlauf dieser Arbeit werden unter anderem diese Fragen beantwortet und die hieraus entstandene Implementation vorgestellt.

Der verwendete Ansatz setzt moderne Merkmalsextraktion zur inkrementellen Schätzung der Bewegung des Fluggerätes, sowie zur Erkennung von geschlossenen Schleifen (*loop-closings*) ein. Die wahrscheinlichste Konfiguration des Gesamtsystems wird dann mittels eines Optimierungsalgorithmus erlangt. Ausgabe ist der genommene

Pfad sowie eine texturierte Oberflächenkarte des überflogenen Gebietes. In Experimenten wird nachgewiesen, dass dieses System auch unter schwierigen Bedingungen und eingeschränkter Sensorenverfügbarkeit konsistente Trajektorien und Karten erzeugen kann.

Im Folgenden wird nun zuerst das SLAM-Problem im Allgemeinen erläutert und dann zum visuellen SLAM übergegangen.

1.2 SLAM

Das größte Problem beim SLAM ist, dass Lokalisierung und Kartenbau jeweils voneinander abhängig sind. Um sich zu lokalisieren wird eine Karte benötigt, aber um eine Karte aufzubauen muss die eigene Position bekannt sein, da sonst unklar ist, wo neue Sensorenmessungen in der Karte zu integrieren sind. Hierdurch spiegeln sich kleine Fehler in den Sensorwahrnehmungen in Fehlern in der Positionsbestimmung anhand der Karte wieder, was wiederum zu Fehlern in neuen Elementen der Karte führt. Insbesondere bei Sensoren, die keine hohe Genauigkeit aufweisen, summieren sich die Fehler in Position und Karte hierdurch schnell auf. Abhilfe bieten hier Verfahren, die versuchen, den entstandenen Fehler zu korrigieren, sobald ein bereits bekanntes Gebiet erneut betreten wird (*Schleifenschluss - loop closing*).

Bezüglich des Kartenbaus ist der Begriff *Landmarken* von großer Bedeutung. Hierunter versteht man Merkmale in der Welt, die mittels der verwendeten Sensoren wahrgenommen und wiedererkannt werden können und mit deren Wahrnehmung Rückschlüsse auf die Position des Roboters möglich sind. Mit Hilfe dieser Orientierungspunkte können landmarkenbasierte Karten aufgebaut werden. Diese können beispielweise aus den (vermuteten) Positionen dieser Orientierungspunkte bestehen. Andere Karten diskretisieren die Umgebung in Zellen gleicher Größe. Eine Zelle beinhaltet hierbei die Wahrscheinlichkeit, dass diese belegt und somit für den Roboter unpassierbar ist. Beide Repräsentationen können hierbei zwei- oder dreidimensionale Informationen speichern. Zwischenlösungen zwischen 2D und 3D sind ebenfalls möglich. Hier sind beispielsweise Höhen-, bzw. Oberflächenkarten einzuordnen, die jeder 2D-Zelle innerhalb der Karte eine Höhe zuordnen.

Im Laufe der Entwicklung der mobilen Robotik entstanden zahlreiche Lösungsansätze für das SLAM-Problem. Hierbei sind probabilistische Verfahren, die sowohl die Position des Roboters, als auch die Elemente der Karte mittels parametrisierter Wahrscheinlichkeitsverteilungen repräsentieren, am stärksten verbreitet.
Hierzu gehört z.B. die Methode des *erweiterten Kalman-Filters (EKF)* [1, 2, 3]. Dieser dient wie der normale *Kalman-Filter (KF)*[1, 4, 5, 6] der Rekonstruktion des Zustandsvektors eines dynamischen Systems unter Verwendung von verrauschten

Observationen. Er kann jedoch auch bei nicht-linearen Prozessen eingesetzt werden. Hierbei geht allerdings durch die Linearisierung die Optimalität verloren. Im Kontext der SLAM-basierten Anwendung werden sowohl Roboter-, als auch Landmarkenpositionen in einem großen Zustandsvektor mit kompletter Kovarianzmatrix verwaltet. Nach Aktionen, bzw. Sensorwahrnehmungen wird dieses System anhand des Bewegungsmodells, bzw. des Sensormodells aktualisiert.
Die Komplexität des EKF pro Updateschritt ist quadratisch in der Dimension des Zustandsvektors, was die maximal mögliche Zahl der Landmarken in der Karte stark einschränkt.
Es gibt einige Alternativen zum EKF, die den Zustandsvektor ebenfalls als parametrisierte Wahrscheinlichkeitsverteilungen repräsentieren. Einige versuchen hierbei die Komplexität einzugrenzen (z.B. SEIF: Sparse Extended Information Filter [1, 7]) und/oder Probleme durch die Linearisierung zu minimieren (z.B. UKF: Unscented Kalman Filter [1, 8, 9]).

Neben diesen Methoden sind auch Stichproben-basierte Ansätze sehr verbreitet. So z.B. der Rao-Blackwellized Partikelfilter[1, 10, 11]. Hier werden multiple Hypothesen über die aktuelle Konfiguration des Systems (Position und Karte) mittels einzelner Partikel repräsentiert, deren Positionsänderungen und zugeordnete Wahrscheinlichkeiten aus dem Bewegungs- und Sensormodell hervorgehen. Wie der EKF sind sie nicht an lineare Prozesse gebunden. Zusätzlich müssen sie nicht auf Gaussverteilungen basieren, sondern können beliebige (insbesondere multi-modale) Wahrscheinlichkeitsverteilungen repräsentieren. Für eine hohe Anzahl von Partikeln nähert sich das Ergebnis an das eines optimalen Bayes-Filters an. Allerdings steigt mit der Anzahl der Partikel der Rechen- und Speicheraufwand linear an. Es muss demnach ein Kompromiss zwischen der Genauigkeit und dem Rechen- bzw. Speicheraufwand geschlossen werden.

Eine weitere Möglichkeit zu Lösung des SLAM-Problems bieten Graphen-, bzw. Netzwerkbasierte Systeme. Bei diesen wird versucht, die bezüglich den gemachten Sensorwahrnehmungen wahrscheinlichste Trajektorie/Karte zu finden (*maximum likelihood approach*). Dies ist die Methode zur Lösung des SLAM-Problems, die für diese Arbeit eingesetzt wurde. Weitere Informationen zu Verfahren dieser Art sind in Kapitel 7 zu finden.

Die geläufigsten Sensoren, die zur Lösung des SLAM-Problems eingesetzt werden, sind z.B. Laserscanner, die direkt den Abstand zu Hindernissen messen. Hierbei handelt es sich um aktive Sensoren, die Signale aussenden und deren Reflexionen bewerten. Ihre Beliebtheit im Bereich der Robotik verdanken diese Sensoren ihrer enormen Genauigkeit, sowie ihrer hohen Reichweite.

1.3 Visueller SLAM

Obwohl Kameras in der Robotik alles andere als unüblich sind, so wurden sie doch bis vor kurzem nur selten zur Lösung des SLAM-Problems eingesetzt. Der Hauptgrund hierfür ist sicherlich, dass der geometrische Aufbau der Umgebung nur indirekt in den Bilddaten beschrieben wird. Daher können nur mittels aufwendiger Nachverarbeitung für den Roboter relevante Informationen gewonnen werden. Dem gegenüber stehen jedoch einige starke Argumente, die für den Einsatz von Kameras sprechen. Sie sind kompakt, passiv, d. h. sie senden keine Signale aus, die andere Sensoren stören könnten, bieten eine hohe Auflösung bei hohen Frameraten, haben eine niedrige Stromaufnahme und sind heutzutage sehr günstig und qualitativ hochwertig verfügbar. Und nicht zuletzt ist ihr Potenzial allein dadurch belegt, dass Menschen sich primär mittels ihrer Augen zu orientieren vermögen.

Hierbei ist noch zu unterscheiden, ob eine (*Monovision*) oder zwei Kameras (*Stereovision*) vorhanden sind. Bei einer Kamera fehlt für einzelne Bilder jegliche Tiefeninformation. Diese ließe sich nur bei bekannter Größe eines Objektes im Bild berechnen. Hingegen lässt sich bei zwei Kameras (Stereovision) die relative dreidimensionale Position eines Punktes, der in beiden Bildern zu sehen ist, berechnen (siehe Kapitel B im Anhang).

Ein Lagesensor, wie er auch in dieser Arbeit verwendet wird (siehe Kapitel C.2 im Anhang), kann genutzt werden, um den Problemraum zu verkleinern, da er Informationen über den Aufnahmewinkel direkt bereitstellt.

1.4 Notationen

Im Folgenden einige Notationen, wie sie im Verlauf dieser Arbeit verwendet werden:

Matrizen

Matrizen werden durch Großbuchstaben oder auch große griechische Buchstaben dargestellt.
Beispiele: $A, R, \Omega, ...$

Vektoren

Vektoren werden durch fett gedruckte Kleinbuchstaben dargestellt.
Beispiele: $\mathbf{a}, \mathbf{p}, \mathbf{x}, ...$

Winkel

Winkel werden durch kleine griechische Buchstaben dargestellt.
Beispiele: $\alpha, \beta, \gamma, \theta, ...$
Weiterhin gelten die folgenden Konventionen:

- Zur besseren Lesbarkeit werden Winkel im Folgenden nicht im Bogenmaß, sondern im Gradmaß $(-180°, 180°]$ angegeben.
- Alle Winkel, auch Ergebnisse von Berechnungen, werden als korrekt auf $(-180°, 180°]$ normalisiert angenommen, ohne dass dies nochmals erwähnt wird.
- Die Subtraktion zweier Winkel α, β ist definiert als die jeweils kleinere Distanz zwischen den beiden Winkeln, d.h:
  Es sei $\delta = \alpha - \beta$ die normale Subtraktion der reellen Werte der Winkel.
  Dann gilt im Folgenden: $\alpha - \beta := \begin{cases} \delta & \text{falls } \delta \in (-180°, 180°] \\ \delta + 360° & \text{falls } \delta <= -180° \\ \delta - 360° & \text{falls } \delta > 180° \end{cases}$

1.5 Aufbau

Diese Arbeit ist wie folgt gegliedert:
In Kapitel 2 werden bisherige Arbeiten zum visuellen SLAM vorgestellt, die ähnliche Ziele mit anderen Ansätzen verfolgen. Anschließend werden in Kapitel 3 die Grundlagen der Abbildungseigenschaften für Kameras behandelt, sowie Verfahren zur Merkmalsextraktion (Kapitel 4) beschrieben. Basierend darauf wird in Kapitel 5 auf den Aufbau einer Karte aus diesen Merkmalen eingegangen und anschließend in Kapitel 6 das entwickelte Verfahren zur Rekonstruktion der Kameraposition erläutert. Kapitel 7 beschreibt den Algorithmus, der für die nachträgliche Optimierung des Pfades verwendet wurde. In Kapitel 8 wird auf die durchgeführten Experimente, ihre Ergebnisse und Deutungen eingegangen. Abschließend werden in Kapitel 9 die erzielten Ergebnisse zusammengefasst und mögliche Erweiterungen und Verbesserungen des Systems diskutiert.

Im Anhang sind einige Berechnungen im Detail, sowie Beschreibungen der verwendeten Hardware zu finden.

2 Verwandte Arbeiten

In diesem Kapitel sollen beispielhaft einige Projekte, die sich mit visueller Lokalisierung befassen, genannt werden, um einen Eindruck über die Anwendungsgebiete zu verschaffen. Außerdem soll ein Überblick über den aktuellen Stand der Technik bezüglich visuellem SLAM gegeben werden und aufgezeigt werden, auf welchen Prinzipien diese Ansätze üblicherweise beruhen und welche Techniken zum Einsatz kommen.

Es gibt einige Ansätze, bei denen der Mangel einer herkömmlichen Odometrie durch eine visuelle Odometrie kompensiert wird. Bei solchen Systemen wird lediglich eine feste Zahl von Kamerabildern (meist nur die letzten zwei) genutzt, um die Bewegungen des Roboters zu schätzen. Eine solche visuelle Odometrie weist die selben Probleme auf, wie normale Rad-Odometrie. Während auf kurze Sicht die Bewegung meist korrekt wiedergegeben werden kann, summieren sich über längere Zeit kleine Fehler auf, so dass die geschätzte Position immer mehr von der tatsächlichen Position abweicht.

Takaoka et al [12] nutzten beispielsweise ein Stereokamerasystem zur Erstellung einer hochauflösenden 3D Karte und zur Navigation eines humanoiden Roboters. Es wurde lediglich ein visuelles Odometriesystem zur Schätzung der Positionänderungen des Roboters genutzt. Zur Extraktion von Merkmalen aus den Bildern wurde der Kanade-Lucas-Tomasi-Detektor (siehe Kapitel 4.3) eingesetzt. In dem vorgestellten Verfahren wird keine Methode zum Finden von geschlossenen Schleifen verwendet. Das Team schlägt jedoch den Einsatz eines ICP-basierten Systems hierfür vor (ICP = *Iterative Closest Point*). Also eine Methode, die versucht, Wolken von 3D-Punkten in Übereinstimmung zu bringen und nicht auf den Kamerabildern selbst arbeitet.

Dornhege und Kleiner [13] verwendeten ein System zur visuellen Odometrie für den Einsatz auf einem Kettenfahrzeug. Es wurde ebenfalls die KLT-Methode zur Merkmalsextraktion eingesetzt. Mittels einer zur Seite zeigenden Standard-Webcam sowie eines Lagesensors wurden die Bewegungen des Vehikels auf komplexem Untergrund nachvollzogen. Eine herkömmliche Odometrie war auf dem verwendeten Fahrzeug nicht verfügbar und wäre auch beim Überwinden von Hindernissen auf Grund von durchrutschenden Ketten sehr fehleranfällig gewesen. Als vereinfachte Annahmen wurde für dieses System davon ausgegangen, dass das Kettenfahrzeug

sich nur mit einer festen Geschwindigkeit vorwärts bewegt. Eine solche Annahme war nötig, da mit nur einer Kamera keine Beurteilung des Abstandes der beobachteten Objekte möglich war.

Neben solchen System zur visuellen Odometrie sind auch zahlreiche Systeme zum visuellen SLAM in der Literatur der letzten Jahre zu finden.

Hierbei gibt es einige durchaus beeindruckende Ergebnisse. So präsentierten z.B. *Davison et al* [14] einen EKF-basierten Ansatz, der mit einer einzelnen Kamera in Echtzeit bei hohen Frameraten (30Hz) 3D-SLAM betreibt. Es wird mit einer dreidimensionalen Karte mit wenigen hochqualitativen Merkmalen gearbeitet, deren Entsprechungen in den folgenden Kamerabildern gesucht werden. Durch das Wiedererkennen von älteren Landmarken in der Karte (*loop closings*) wird ein Drift von der tatsächlichen Position verhindert, sofern oft genug zu alten Positionen zurückgekehrt wird. Es wird ein sehr allgemeines Bewegungsmodell für die Kamera verwendet um die Merkmalspositionen vorauszusagen und so den Suchraum einzuschränken. Zur Extraktion von möglichen Merkmalspositionen wird auch hier der KLT-Detektor eingesetzt. Der Vergleich von möglichen übereinstimmenden Merkmalen erfolgt über den direkten Vergleich von Bildausschnitten, transformiert entsprechend dem Unterschied in der Betrachtung. Die Kartengröße ist bei diesem System auf eine relativ geringe maximal mögliche Anzahl von Landmarken (etwa 100 Stück) eingeschränkt, was unter anderem aus der Komplexität des EKF resultiert. Die Anwendung ist hauptsächlich für Bewegungen in Räumen gedacht, bei denen ab einem gewissen Punkt keine neuen Gebiete mehr hinzukommen.

In einer früheren Arbeit von *Davison et al* [15] wurde ein Stereovisionsystem für die Navigation und den Bau einer dreidimensionalen Merkmalskarte für einen mobilen Roboter auf unebenem Untergrund in Echtzeit verwendet. Hier wurde ebenfalls ein EKF-Ansatz zum Bau einer Karte von wenigen guten Merkmalen genutzt. Es wurde aktive visuelle Wahrnehmung eingesetzt. Die Kameras waren also beweglich, so dass ein Bildbereich für eine gewisse Zeit fixiert werden konnte, während der Roboter sich bewegte. Dadurch müssen weniger Landmarken gespeichert werden. Neben den visuellen Daten waren auf dieser Plattform Daten eines Lagesensors und die Odometrie aus den Radbewegungen verfügbar, was die Unsicherheiten im System vermindert. Neben der Lokalisierung des Roboters wurde außerdem die Bodenoberfläche als Anordnung von Rampen/Flächen rekonstruiert.

Ein Ansatz zum visuellen SLAM für Fluggeräte wurde 2003 von *Kim et al* [16] beschrieben. Ziel war die Positionsbestimmung eines unbemannten Flugzeugs (ugf. 45 kg). Zur Lokalisierung des UAVs (uninhabitated air vehicle - unbemanntes Flugobjekt) wurde ein EKF eingesetzt. Im Unterschied zu dem in dieser Arbeit betrachteten Szenario standen hier qualitativ hochwertige Sensoren zu Verfügung und die

verwendeten visuellen Landmarken waren künstlich, mit bekannter Geometrie angebracht. Außerdem konnte angenommen werden, dass sich diese auf einer flachen Ebene befinden. Ziel des Experimentes war es, Lokalisierungsmethoden basierend auf passiven Sensoren zu testen. Über mehrere geflogene Schleifen hinweg konnte die Position erfolgreich extrahiert werden.

In einem dem Szenario in dieser Arbeit sehr ähnlich kommenden Aufbau verarbeiteten *Eustice et al* [17] einen Satz von Unterwasseraufnahmen der RMS-Titanic, inklusive IMU-Daten (Inertial Measurement Unit - Inertiales Navigationssystem), zur Rekonstruktion des Pfades des verwendeten Unterwasserfahrzeuges sowie einer detaillierten Darstellung des Meeresbodens, bzw. des Wracks. Wie auch bei dem für diese Arbeit vorhandenen Aufbau sind also Bilder einer nach unten ausgerichteten Kamera, sowie 3D-Orientierungen durch die IMU vorhanden. Allerdings standen *Eustice et al* weit hochwertigere Sensoren zur Verfügung. Die hochauflösenden Bilder der nach unten gerichteten Kamera wurden in einem SEIF-ähnlichen Filter verarbeitet. Dieser nutzt die Eigenschaft der Informationsmatrix, viele Nulleinträge (bzw. sehr kleine Einträge) aufzuweisen, zur Verringerung des Rechenaufwands für Updates. Dies wird dem dort behandelten Szenario eines großen Gebietes mit nur wenigen geschlossenen Schleifen gerecht.

Ebenfalls ähnlich zu dem in dieser Arbeit behandelten Szenario ist ein Projekt von *Jung et al* [18]. Hier wurde ein Zeppelin mit einem hochwertigen, nach unten gerichteten Stereokamerasystem ausgerüstet. Ziel war die Erstellung einer hochauflösenden Höhenkarte des überflogenen Terrains. Ein Lagesensor wurde hier nicht eingesetzt. Wieder wurde ein EKF-Ansatz verfolgt, bei dem die Odometriedaten jedoch ebenfalls aus den Kamerabildern stammten, indem die extrahierten Merkmale in zwei Mengen aufgeteilt wurden, von denen jeweils eine für den Kartenbau und eine für die Bild-zu-Bild-Verfolgung verwendet wurde. Die entstandenen Karten sind von beachtlicher Qualität, allerdings lief die Implementierung nicht in Echtzeit.

Wie bei diesen Beispielen zu sehen ist, ist ein probabilistisches Vorgehen, wie z.B. die Methode des EKF, bei dem sowohl Kamera-, als auch Merkmalspositionen in einer Kovarianzmatrix verwaltet werden, sehr verbreitet in diesem Gebiet.
Der EKF schränkt jedoch die maximal verwaltbare Anzahl von Merkmalen (Landmarken) stark ein.

Elinas et al [19] stellten einen Ansatz vor, der zwar ebenfalls probabilistisch arbeitet, jedoch nicht mit einem EKF, sondern stichprobenbasiert mit einem Rao-Blackwell Partikelfilter arbeitete. Als Sensor für einen mobilen Roboter stand hier nur eine (nach vorne ausgerichtete) Stereokamera zur Verfügung, aus deren Daten jeweils visuelle Odometrie und wiedererkannte Landmarken in den Partikelfilter einflossen. Ähnlich zu dieser Arbeit wurde RANSAC (siehe Kapitel 6.1) verwendet um

das Problem von falsch erkannten Landmarken zu minimieren. Zwecks Merkmalsextraktion und -vergleich wurde die SIFT-Technologie eingesetzt (siehe Kapitel 4.7 und 4.10). Hierbei werden die Merkmale mittels eines Deskriptorvektors beschrieben, was einen effektiven Vergleich ohne die tatsächlichen Bilddaten möglich macht. Das SLAM-System arbeitete offline auf gespeicherten Daten, da nur ungefähr 1.5 Bilder pro Sekunde verarbeitet werden konnten.

In dieser Arbeit wird das Ziel verfolgt, ein robustes System zum visuellen SLAM zu erhalten. Dies auch unter Verwendung von günstigen, qualitativ nicht sehr hochwertigen Kameras (Standardwebcams), aus denen ein einfaches Stereokamerasystem gebildet wird. Das betrachtete Szenario ähnelt einigen der oben genannten Ansätze, bei denen ein Fluggerät (oder äquivalent ein Unterwasserfahrzeug) sich anhand visueller Aufnahmen des Bodens orientiert. Als zusätzlicher Sensor steht ein vergleichsweise günstiger Lagesensor zur Verfügung, der die dreidimensionale Orientierung im Raum misst. Zur Merkmalsextraktion wird SURF (siehe Kapitel 4.8 und 4.11) eingesetzt, eine Technologie, die ähnlich zu der von SIFT, jedoch weniger rechenintensiv ist. Auch hier werden die Merkmale durch einen Deskriptorvektor beschrieben, was effektiven Vergleich zwischen Merkmalen ermöglicht und gleichzeitig das Erstellen von Landmarkenkarten erleichtert. Im Vergleich zu den oben genannten Autoren wird in dieser Arbeit auf einen probabilistischen Ansatz verzichtet. Statt dessen wird das Prinzip des Graph-basierten SLAMs (siehe Kapitel 7) mit visuellen Lokalisierungsmethoden verbunden. Hierdurch erhalten wir ein System, dass auch Karten mit einer hohen Anzahl von Landmarken (im fünfstelligen Bereich) verwalten kann.
Das Ergebnis sind hierbei jeweils die Trajektorie, sowie eine texturierte Höhenkarten des überflogenen Gebietes.

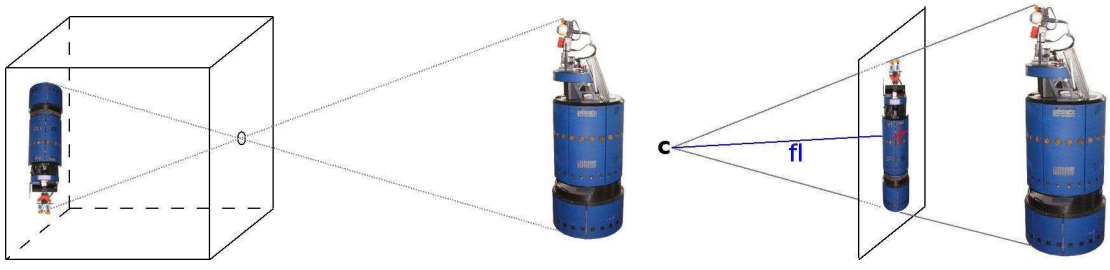
3 Grundlagen zu optischen Abbildungen

Eine Kamera erzeugt zweidimensionale Abbilder dreidimensionaler Strukturen. Was hierbei zu beachten ist, und wie sich geometrisch damit arbeiten lässt, soll im Folgenden näher erläutert werden. Dazu wird zunächst das Prinzip einer idealisierten Lochkamera erläutert und dann darauf eingegangen, wie sich dieses von realen Kameras unterscheidet.

3.1 Die Lochkamera

Zur Berechnung dreidimensionaler Daten aus Kamerabildern ist es sinnvoll, zuerst das vereinfachte System einer *idealen Lochkamera (pinhole camera)* zu betrachten, bei der Punkte im Raum mittels perspektivischer Projektion auf eine Bildebene (bzw. Projektionsebene) abgebildet werden. Hierdurch wird ein auf dem Kopf stehendes, unverzerrtes Abbild geschaffen (siehe Abb. 3.1 (a)).

Es ist auch möglich, sich die Projektionsebene zwischen Kamera und Objekt vorzustellen, was prinzipiell äquivalent ist, aber geometrische Überlegungen oft vereinfacht. Insbesondere steht hierbei das Bild nicht auf dem Kopf (siehe Abb. 3.1 (b)).



(a) (b)

Abbildung 3.1: **(a)** Skizze einer Lochkamera **(b)** Lochkameraskizze mit Projektionsebene zwischen Objekt und Kamera

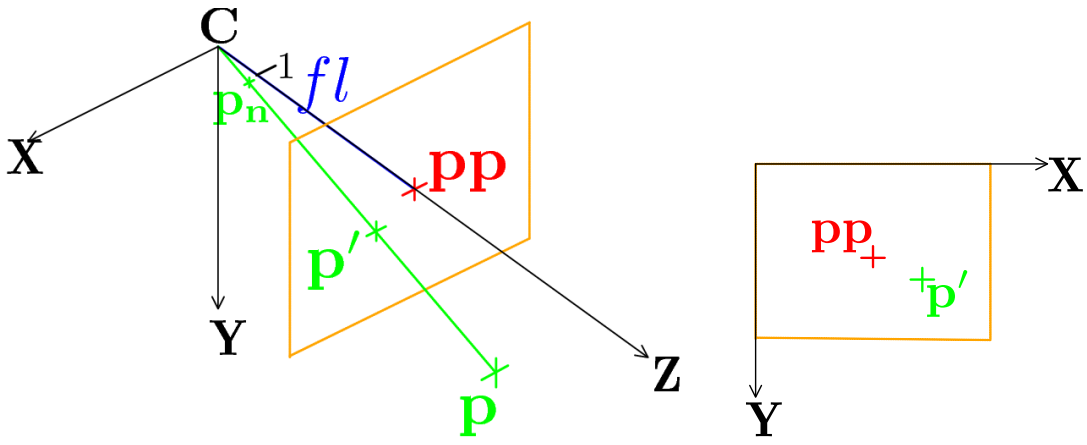


Abbildung 3.2: **(a)** Beispiel für die Abbildung eines Punktes $\mathbf{p}$ im Kamerasystem auf den Bildpunkt $\mathbf{p}'$, sowie den normalisierten Bildpunkt $\mathbf{p_n}$. **(b)** Das zu (a) zugehörige Bildsystem.

3.2 Interne Kameraparameter

Die bauartbedingten Eigenschaften einer Kamera werden mittels der sogenannten *internen*, oder auch *intrinsischen Kameraparameter* beschrieben.
Hierzu gehört beispielsweise der Abstand zwischen Projektionsebene und dem Lichteinlass C, die sogenannte *Brennweite (focal length)* fl (siehe Abb. 3.1 (b)). In Kombination mit der Auflösung des Kamerabildes ergibt sich aus der Brennweite der Aufnahmewinkel und damit die Größe des Sichtfeldes.
Die Brennweite wird in dieser Arbeit als konstanter Wert für die jeweils verwendete Kamera angenommen. Dies gilt im Allgemeinen nicht. Eine Kamera mit verstellbarem Zoom hat beispielsweise eine variable Brennweite.
Weitere interne Kameraparameter werden im weiteren Verlauf dieses Kapitels erläutert werden.

Welt-, Kamera- und Bildkoordinatensystem

Im Folgenden werden drei verschiedene Koordinatensysteme von Bedeutung sein, das dreidimensionale Weltsystem, das dreidimensionale Kamerasystem und das zweidimensionale Bildkoordinatensystem. Das Weltsystem ist das System, in dem die Karte mit Kameraposition und Positionen der Landmarken dargestellt wird. Der Nullpunkt sowie die Ausrichtung dieses Systems sind prinzipiell willkürlich wählbar, wobei üblicherweise die X-Achse nach Norden zeigt und die Z-Achse nach oben, der Gravitation entgegen, gerichtet ist.

Im Kamerasystem bildet die Position der Kamera stets den Ursprung (siehe Abb. 3.2 (a)). Diese Position ist bei der Betrachtung als Lochkamera die Position des Lichteinlasses C.
Die Projektionsebene ist parallel zur X-Achse, sowie der Y-Achse und damit senkrecht zur Z-Achse. Ein Punkt $\mathbf{p} = \begin{pmatrix} x_p \\ y_p \\ z_p \end{pmatrix}$ im Raum wird nach dem Strahlensatz auf den Punkt

$$\mathbf{p'} = \begin{pmatrix} x_{p'} \\ y_{p'} \\ z_{p'} \end{pmatrix} = \begin{pmatrix} fl \cdot x_p/z_p \\ fl \cdot y_p/z_p \\ fl \end{pmatrix} \tag{3.1}$$

auf der Projektionsebene abgebildet (siehe Abb. 3.2 (a)).

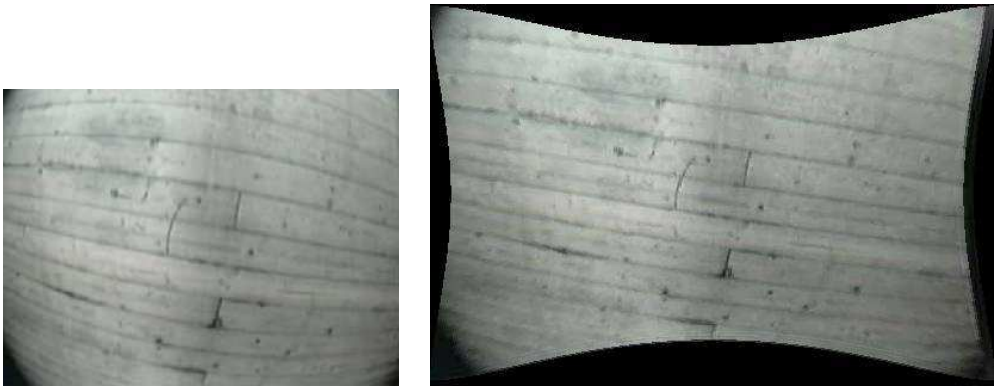
Das Bildkoordinatensystem repräsentiert die tatsächlichen Pixel, aus denen sich das Kamerabild zusammensetzt. Es ist die zweidimensionale Darstellung der Projektionsebene aus dem Kamerasystem (siehe Abb. 3.2 (b)). Hierbei ist zu beachten, dass die einzelnen Pixel eines Kamerabildes normalerweise mit $\mathbf{0_B} = (0\ 0)^T$ in der linken oberen Ecke beginnen. Der Punkt $(0\ 0\ fl)^T$ im Kamerasystem entspricht im Normalfall jedoch nicht dieser Ecke sondern dem Pixel $\mathbf{pp} = (x_{pp}\ y_{pp})^T$ im Bildsystem. Dieser Punkt wird *Bildhauptpunkt (principle point)* genannt. Er gehört wie die Brennweite zu den internen Kameraparametern.
Der zu $\mathbf{p'}$ zugehörige Pixel im Bild $\mathbf{p_B} = (x_{p_B}\ y_{p_B})^T$ ist demnach gerade um die Position des Bildhauptpunktes verschoben:
$\mathbf{p_B} = \begin{pmatrix} x_{p'} + x_{pp} \\ y_{p'} + y_{pp} \end{pmatrix}$

Ein wichtiger Begriff ist der des *normalisierten Bildpunktes*. Dieser entspricht der Projektion eines Punktes im Raum auf eine Projektionsebene mit einem Pixel Abstand zur Kamera. D. h. er entspricht demjenigen Bildpunkt, der von einer idealen Lochkamera mit Brennweite 1.0 aufgenommen würde.
Für den Punkt $\mathbf{p}$ ist der zugehörige normalisierte Bildpunkt im Kamerasystem

$$\mathbf{p_n} = \begin{pmatrix} x_p/z_p \\ y_p/z_p \\ 1 \end{pmatrix} = \begin{pmatrix} x_{p'}/fl \\ y_{p'}/fl \\ 1 \end{pmatrix} = \begin{pmatrix} (x_{p_B} - x_{pp})/fl \\ (y_{p_B} - y_{pp})/fl \\ 1 \end{pmatrix}. \tag{3.2}$$

3.3 Reale Kameras

Reale Kameras können in den meisten Fällen nicht hinreichend genau durch eine Lochkamera beschrieben werden. Verzerrungen durch die Linse, wie sie insbesondere



(a) (b)

Abbildung 3.3: **(a)** Bild eines Holzbodens durch eine Weitwinkelkamera. Die Linien auf dem Boden sind in Wirklichkeit gerade und parallel zueinander. **(b)** Das selbe Bild wie in (a), jedoch entzerrt.

bei Verwendung von Weitwinkelobjektiven auftreten, können nicht einfach ignoriert werden. Ein solch verzerrtes Bild ist in Abb. 3.3 zu sehen.
Brown veröffentlichte 1966 [20] ein mathematisches Modell (das sogenannte *Plumb Bob model*) für die Eigenschaften verschiedener Linsen. Es beschreibt die auftretenden Verzerrungen durch fünf Parameter, die radiale und tangentiale Verzerrung (d. h. Abweichungen des Zentrums der radialen Verzerrung) angeben. Mittels dieser Werte ist es möglich, die Verzerrung als Funktion anzugeben [21]. Für die Inverse existiert zwar kein allgemeiner algebraischer Ausdruck, eine numerische Implementation der Umkehrung ist jedoch möglich, so dass für bekannte Parameter der Kamera und Bildpunkte im verzerrten Bild die unverzerrten Bildpunkte, wie sie eine Lochkamera geliefert hätte, berechnet werden können. In der Implementation für diese Arbeit wird jeweils eine Tabelle mit vorberechneten Werten verwendet, um den hohen rechnerischen Aufwand für das Entzerren zu verringern.
Die fünf Parameter zur Beschreibung der Linsenverzerrungen gehören wie die Brennweite und die Position des Bildhauptpunktes zu den internen Kameraparametern.

Im allgemeinen sind noch zwei weitere interne Kameraparameter zu beachten, der *Scherungskoeffizient (skew coefficient)*, der Abweichungen des Winkels zwischen X- und Y-Achse von den normalen 90° beschreibt, sowie das Verhältnis der Pixelgröße in X-Richtung zu der in Y-Richtung (*aspect ratio*), welche in Betracht zieht, dass Pixel nicht perfekt quadratisch sondern rechteckig sein können. Für die meisten Kameras, insbesondere alle in dieser Arbeit verwendeten, können diese Werte vernachlässigt werden, weshalb sie hier nicht weiter beachtet werden.

Zur Bestimmung der internen Parameter einer Kamera ist diverse Software im Internet verfügbar. Für diese Arbeit wurde die *Matlab Camera Calibration Toolbox*[22]

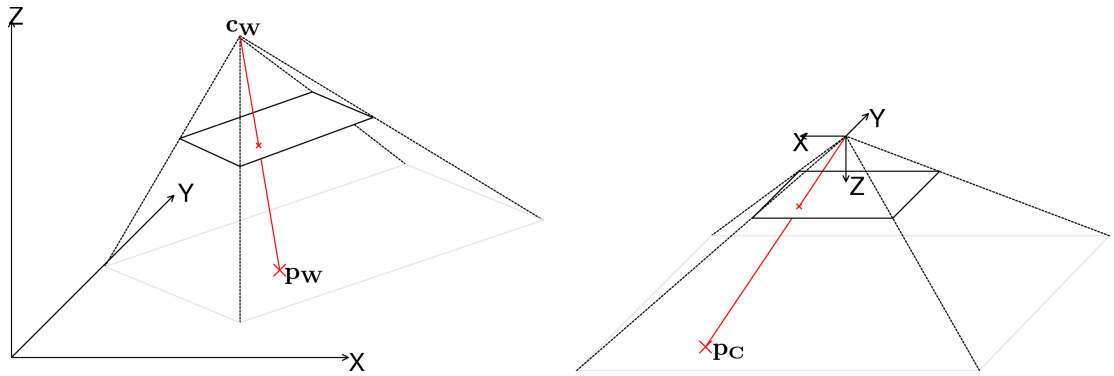


Abbildung 3.4: **(a)** Weltsystem mit Kamera auf Position $\mathbf{c_W}$ **(b)** Entsprechendes Kamerasystem zu (a), mit der Kamera im Ursprung und der Projektionsebene senkrecht zur Z-Achse

verwendet. Wie bei den meisten Programmen dieser Art werden Aufnahmen eines Schachbrettmusters bekannter Größe aus verschiedenen Perspektiven genutzt, um die internen Parameter der verwendeten Kamera zu schätzen.

Mittels der bekannten internen Parameter einer Kamera lassen sich zu jedem Bildpunkt die normalisierten Bildpunkte berechnen, die dann ohne das Wissen über die Kamera genutzt werden können. Daher werden im Folgenden o.B.d.A. alle geometrischen Überlegungen bezüglich einer Lochkamera gemacht werden.

3.4 Externe Kameraparameter

Die sogenannten externen oder extrinsischen Kameraparameter bestimmen die Transformation zwischen Kamerasystem und Weltsystem. Sie beinhalten die 3D-Position der Kamera $\mathbf{c_W} = (x_{c_W}\ y_{c_W}\ z_{c_W})^T$ im Weltsystem (also die Translation), sowie die Rotation, die nötig ist, um das eine System in das andere zu überführen (siehe Abb. 3.4). Für die Rotation gibt es verschiedene Möglichkeiten der Repräsentation, wie z.B. Rotationsmatrizen, Quaternionen[23] oder Eulerwinkel. Letztere sind mit drei Parametern die kompakteste Form der Repräsentation. Die drei Parameter sind der *Roll* γ (Rollwinkel, Drehung um die X-Achse), der *Pitch* β (Neigungswinkel, Drehung um die Y-Achse) und der *Yaw* α (Drehwinkel, Drehung um die Z-Achse) (siehe Abb. 3.5).
Die Reihenfolge, in der diese Drehungen angewandt werden, muss beachtet werden. Als Standard gilt hier die sogenannte XYZ-Konvention, d. h. zuerst wird um die X-Achse gedreht (Roll), dann um die Y-Achse (Pitch) und dann um die Z-Achse (Yaw). Es ergibt sich also die Rotationsmatrix R als Produkt der Rotationsmatrizen

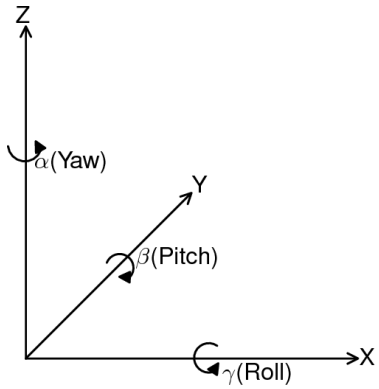


Abbildung 3.5: Skizze zur Erklärung der Winkel Roll, Pitch und Yaw.

der einzelnen Winkel:

$$R = R_\alpha \cdot R_\beta \cdot R_\gamma \qquad \text{mit } R_\phi = \begin{pmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{pmatrix}$$

Es sei $\mathbf{p_C}$ ein Punkt im Kamerasystem. Mittels der sechs externen Kameraparameter x_{c_W}, y_{c_W}, z_{c_W}, γ, β und α ergibt sich die Transformation in den entsprechenden Punkt $\mathbf{p_W}$ im Weltsystem als:

$$\begin{aligned} \mathbf{p_W} &= R_\alpha \cdot R_\beta \cdot R_\gamma \cdot \mathbf{p_C} + \mathbf{c_W} = R \cdot \mathbf{p_C} + \mathbf{c_W} \\ &= \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \cdot \begin{pmatrix} x_{p_C} \\ y_{p_C} \\ z_{p_C} \end{pmatrix} + \begin{pmatrix} x_{c_W} \\ y_{c_W} \\ z_{c_W} \end{pmatrix} \end{aligned} \tag{3.3}$$

3.5 3D-Daten aus Kamerabildern

Angenommen die Position der Kamera $\mathbf{c_W}$, sowie die Rotationsmatrix R sind bekannt und es soll von den Punkten aus dem aktuellen Kamerabild auf den Aufbau der Welt geschlossen werden. Die Berechnung, welchem Punkt $\mathbf{p_W}$ in der Welt, bzw. $\mathbf{p_C}$ im Kamerasystem, ein normalisierter Bildpunkt $\mathbf{p_n} = (x_{p_n}, y_{p_n})^T$ aus dem aktuellen Kamerabild entspricht, ist nicht ohne weitere Informationen möglich.
Aus den Gleichungen 3.2 und 3.3 erhält man:

$$\begin{aligned} & \mathbf{p_W} \overset{3.3}{=} R \cdot \mathbf{p_C} + \mathbf{c_W} \overset{3.2}{=} R \cdot z_{p_C} \mathbf{p_n} + \mathbf{c_W} \\ \iff & \begin{pmatrix} x_{p_W} \\ y_{p_W} \\ z_{p_W} \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \cdot \begin{pmatrix} x_{p_n} \cdot z_{p_C} \\ y_{p_n} \cdot z_{p_C} \\ z_{p_C} \end{pmatrix} + \begin{pmatrix} x_{c_W} \\ y_{c_W} \\ z_{c_W} \end{pmatrix} \end{aligned} \tag{3.4}$$

Hierbei handelt es sich um ein Gleichungssystem mit drei Gleichungen, jedoch vier Unbekannten: x_{p_W}, y_{p_W}, z_{p_W} und z_{p_C}.

Einer dieser Werte muss aus einer anderen Quelle bekannt sein, um die anderen berechnen zu können. Bei einem Stereokamerasystem ist es beispielsweise möglich, z_{p_C} zu berechnen (siehe Kapitel B im Anhang), wodurch $\mathbf{p_W}$ direkt aus obigem Gleichungssystem hervorgeht.
In unserer Anwendung, bei der die Daten von einer unter einem Fluggerät angebrachten, nach unten gerichteten Kamera stammen, ist alternativ die Annahme möglich, dass z_{p_W} ein bekannter, für alle Bildpunkte konstanter Wert ist. Dies entspricht der Annahme, dass wir uns über einem flachen Boden befinden. In diesem Fall können wir z_{p_C} mittels der dritten Gleichung aus 3.4 berechnen:

$$\begin{aligned} & z_{p_W} \overset{3.4}{=} \begin{pmatrix} r_{31} & r_{32} & r_{33} \end{pmatrix} \cdot \begin{pmatrix} x_{p_n} \cdot z_{p_C} \\ y_{p_n} \cdot z_{p_C} \\ z_{p_C} \end{pmatrix} + z_{c_W} \\ \iff \quad & z_{p_W} = z_{p_C} \cdot (r_{31} x_{p_n} + r_{32} y_{p_n} + r_{33}) + z_{c_W} \\ \iff \quad & z_{p_C} = \frac{z_{p_W} - z_{c_W}}{r_{31} x_{p_n} + r_{32} y_{p_n} + r_{33}} \end{aligned} \tag{3.5}$$

Eingesetzt in 3.4 lassen sich nun x_{p_W} und y_{p_W} berechnen.

3.6 Kameradaten aus 3D-Punkten

Angenommen es ist ein 3D-Punkt $\mathbf{p_W}$ in der Welt bekannt und es soll berechnet werden, wo dieser im Kamerabild auftaucht. Hierzu wird zuerst seine Position im Kamerabild mittels der Umkehrung von Gleichung 3.3 berechnet. Hierfür ist zu beachten, dass Rotationsmatrizen orthogonale Matrizen sind, d. h. $R^{-1} = R^T$.

$$\begin{aligned} & \mathbf{p_C} \overset{3.3}{=} R^{-1}(\mathbf{p_W} - \mathbf{c_W}) \\ \iff \quad & \mathbf{p_C} \overset{R^{-1} = R^T}{=} \begin{pmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{pmatrix} \cdot \begin{pmatrix} x_{p_W} - x_{c_W} \\ y_{p_W} - y_{c_W} \\ z_{p_W} - z_{c_W} \end{pmatrix} \end{aligned} \tag{3.6}$$

Mittels Gleichung 3.2 lässt sich nun der normalisierte Bildpunkt berechnen:

$$p_n = \begin{pmatrix} x_{p_C} / z_{p_C} \\ y_{p_C} / z_{p_C} \end{pmatrix}$$

Und hieraus lässt sich mittels der internen Kameraparameter der tatsächliche Bildpunkt p_B berechnen. Vorausgesetzt, der Punkt ist von der Kamera aus gesehen nicht verdeckt und befindet sich innerhalb des Sichtwinkels der Kamera.

4 Merkmalsextraktion

Im letzten Kapitel wurde behandelt, wie Punkte in der Welt in ein Kamerabild abgebildet werden und in welchem geometrischen Zusammenhang Bildpunkt und Weltpunkt stehen. Um diese Daten für SLAM nutzen zu können, muss das System in der Lage sein, Weltelemente in verschiedenen Kamerabildern wiederzuerkennen. Dieses Kapitel beschreibt technische Möglichkeiten, die solche Identifizierungen möglich machen.
In diesem Zusammenhang wird der Begriff *Merkmal (feature)* benötigt. Ein Merkmal ist prinzipiell ein Punkt in der Welt, der jedoch innerhalb eines Kamerabildes durch seine nähere Umgebung definiert ist, also durch einen Bereich, der größer als ein Punkt ist.

Die Arbeitsweise mit solchen Merkmalen lässt sich grob in zwei Schritte gliedern:

- Die Erkennung informationsreicher/interessanter Stellen im Bild (*interest point detector*) [24], die als Merkmal nutzbar sind. Dies sind Bereiche im Bild, deren Positionen wohldefiniert sind und für die eine robuste Erkennung möglich ist. Letztere Bedingung ist auch als *Wiederholbarkeit (repeatability)* bekannt. D. h., dass ein solches Merkmal, das in einem Bild gefunden wurde, auch in anderen Bildern als interessanter Punkt klassifiziert wird.

- Die Beschreibung der Merkmale. Jedes der gefundenen Merkmale muss nun in einer geeigneten Weise parametrisiert werden, so dass ein Ähnlichkeitsvergleich von gefundenen Merkmalen möglich wird. Diese Beschreibung sollte möglichst eindeutig, aber dennoch robust bezüglich Rauschen in den Bildern sein. Außerdem sollte sie invariant gegenüber typischen, aus der Kameraposition resultierenden Transformationen sein. Hierzu gehören z.B. Größe und Orientierung des Merkmals im Bild.

Alle Überlegungen hierzu werden im Folgenden bezüglich Graustufenbildern gemacht. Ein Kamerabild wird durch eine Matrix von Intensitäten I dargestellt. Dabei gibt $I(x, y)$ den Grauwert des Pixels an der Stelle (x, y) wieder.

Für den ersten Schritt, das Finden interessanter Bereiche im Bild, sind einige Grundüberlegungen nötig, was solche Stellen charakterisieren könnte. Offensichtlich sollte es sich um kontrastreiche Stellen handeln, an denen sich etwas im Bild verändert.

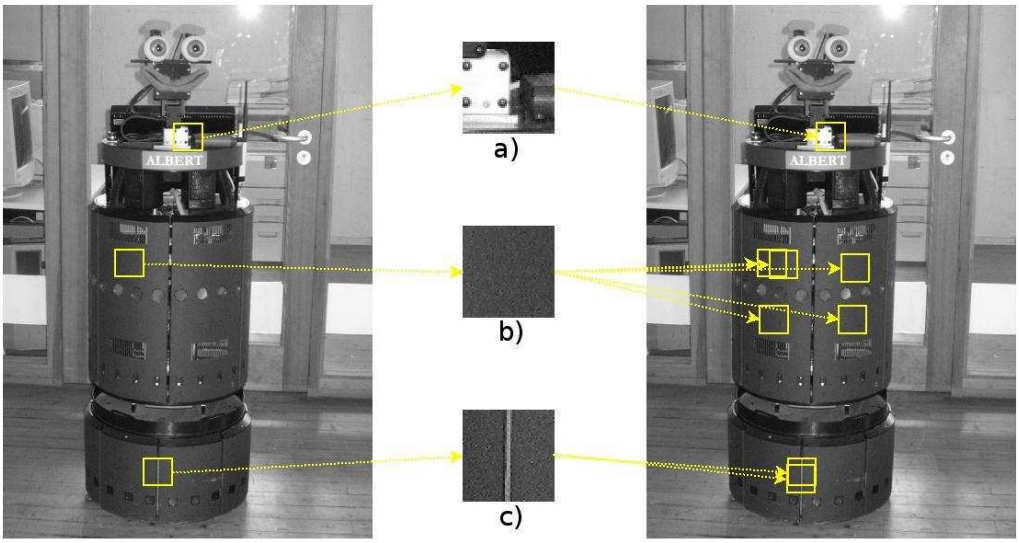


Abbildung 4.1: **Links**: Beispielbild mit möglichen Merkmalen. **Mitte**: Vergrößerte Darstellung dieser Merkmale. **Mitte**: Beispiele für Positionen, an denen diese Merkmale wiedererkannt werden könnten. Merkmal **a)** kann hierbei eindeutig zugeordnet werden. Die Merkmale **b)** und **c)** hingegen besitzen keine eindeutige Struktur, weshalb für diese Merkmale mehrere Vergleichskandidaten vorhanden sind.

Betrachtet man beispielsweise Abb. 4.1, so ist der markierte Bereich b) offensichtlich kein gutes Merkmal, da es sich nur um ein einheitlich graues Gebiet handelt. Wie im rechten Bereich der Abbildung zu sehen ist, könnte dieser Bereich an verschiedensten Stellen im Bild gefunden werden. Insbesondere in der unmittelbaren Umgebung seiner tatsächlichen Position, die demnach nicht wohldefiniert ist. Kleine Verschiebungen in X- oder Y-Richtung liefern optisch nicht unterscheidbare Ausschnitte. Der markierte Bereich c) ist etwas besser, da der einheitlich graue Bereich zumindest durch eine hellere Linie/Kante unterbrochen ist. Jedoch ist auch hier die Position in der unmittelbaren Umgebung nicht eindeutig. Kleine Verschiebungen in X-Richtung sind zwar leicht zu erkennen, in Y-Richtung jedoch ist die Position optisch mehrdeutig. Der markierte Bereich a) scheint ein gutes Merkmal zu sein, da seine Position, insbesondere in seiner unmittelbaren Umgebung, eindeutig erscheint. Übergänge von dunkel nach hell, bilden hier Ecken miteinander und es sind dunkle Punkte zu sehen, die sich eindeutig von ihrem helleren Untergrund abgrenzen. Würde man als Mensch versuchen, den Bildausschnitt „an die richtige Stelle zu schieben“, so wären dies vermutlich die Elemente, an denen man sich orientieren würde.

In der englischen Literatur werden die Begriffe *interest point detector* und *corner detector* oftmals als äquivalent behandelt. Jedoch sind Ecken (*corner*) (Schnittpunk-

te zweier Kanten, oder allgemeiner: Punkte, die eindeutig durch zwei umgebende Kanten verschiedener Richtung definiert sind) nur eine bestimmte Art von *interest points*. Andere Möglichkeiten sind beispielsweise isolierte Intensitätsextrema (also dunkle oder helle Bereiche auf andersartigem Hintergrund - *blobs*), Linienenden oder Punkte auf Kurven, an denen die Krümmung lokal maximal ist.

Es stellt sich nun die Frage, wie sich solche Eigenschaften von Bildausschnitten mathematisch beschreiben, bzw. algorithmisch im Bild finden lassen. Im Folgenden werden nun zuerst einige Methoden zum Finden solcher interessanter Stellen im Bild vorgestellt. Danach wird auf Möglichkeiten zum Vergleich der gefundenen Merkmale eingegangen.

4.1 Der Moravec-Detektor

Einer der frühesten Ansätze zur Extraktion von informationsreichen Merkmalen aus Kamerabildern ist der Moravec-Detektor [25, 24, 26] (Hans Peter Moravec 1980). Er definiert diese als Zentren von Bereichen mit geringer Selbstähnlichkeit. Also Bereiche, die möglichst unähnlich zu nahen, stark überlappenden Bereichen sind.
Konkret wird dies für einen bestimmten Bildausschnitt geprüft, indem dieser mehrmals leicht innerhalb des Bildes verschoben wird und dann mit dem „neuen Untergrund“ verglichen wird. Es sei

- $I(x, y)$ die Intensität des betrachteten Graubildes an der Stelle $(x\ y)^T$.
- $(x_0\ y_0)^T$ die linke obere Ecke des s_x Pixel breiten und s_y Pixel hohen rechteckigen Bildausschnittes.
- $(\Delta x, \Delta y)$ der Positionsunterschied zwischen verschobenem und unverschobenem Bildausschnitt.

Für den Unterschied zwischen verschobenem und unverschobenem Bildausschnitt verwendet Moravec die Summe der quadratischen Unterschiede:

$$S = \sum_{x=x_0}^{x_0+s_x} \sum_{y=y_0}^{y_0+y_x} \left(I(x + \Delta x, y + \Delta y) - I(x, y)\right)^2 \tag{4.1}$$

Je kleiner dieser Wert, um so ähnlicher sind die beiden Bildausschnitte. Für alle geprüften Verschiebungen des Bildausschnittes wird das minimale Ergebnis von S zur Bewertung des Ausschnittes verwendet. Moravec betrachtete jeweils die Verschiebungen $(\Delta x, \Delta y) \in \{(1, 0), (1, 1), (0, 1), (-1, 1)\}$.

Betrachten wir noch einmal Abb. 4.1: Ausschnitt b) würde offensichtlich für alle Verschiebungen $(\Delta x, \Delta y)$ einen sehr geringen Wert von S erhalten, da er sich in keinster Weise von seiner näheren Umgebung abhebt. Ausschnitt c) würde höhere Werte für $(\Delta x, \Delta y) \in \{(1,0),(1,1),(-1,1)\}$ erhalten, jedoch einen sehr kleinen Wert für $(\Delta x, \Delta y) = (0,1)$, da dies einer Verschiebung in Richtung der Linie entspricht. Da das Minimum zur Bewertung herangezogen wird, hätte also auch dieser Ausschnitt einen sehr geringen Wert. Ausschnitt a) würde für alle $(\Delta x, \Delta y)$ eine hohen Wert von S erhalten und somit korrekt als gutes Merkmal angesehen werden.

Um nun ein Bild auf Merkmale zu überprüfen kann beispielsweise eine feste Ausschnittsgrösse gewählt werden und jeder Pixel im Bild, der Zentrum für solch einen Ausschnitt sein kann, mit obigem Verfahren geprüft werden.

Diesem Verfahren sind seine Einfachheit und intuitive Verständlichkeit zugute zu halten, es hat jedoch auch einige Nachteile. So ist es z.B. anisotropisch, d. h. es ist abhängig von der Drehung der potenziellen Merkmale im Bild. So würde z.B. eine Kante, die parallel zu einer der geprüften Verschiebungen ist niedrig bewertet werden, anders gedrehte Kanten können hingegen fälschlich als gute Merkmale erkannt werden.
Außerdem ist das Verfahren abhängig von den gewählten Ausschnittgröße, so dass nur Merkmale einer bestimmten Größe gefunden werden.

4.2 Der Harris-Detektor

Der Harris-Detektor[26, 24] (Harris&Stephens 1988) gehört zu den am häufigsten genutzten Methoden, informationsreiche Merkmale aus Bildern zu extrahieren. Die Idee des Moravec-Detektors wird hierbei aufgegriffen, wobei auf das tatsächliche Rechnen mit verschobenen Bildausschnitten verzichtet wird und stattdessen die Gradienten der Pixelintensitäten herangezogen werden.

Betrachten wir hierzu erneut die Summe der quadratischen Unterschiede zweier verschobener Bildausschnitte (Gleichung 4.1):

$$S = \sum_{x=x_0}^{x_0+s_x} \sum_{y=y_0}^{y_0+y_x} \left(I(x+\Delta x, y+\Delta y) - I(x,y)\right)^2$$

Um einen von der Rotation des Merkmals unabhängigen Detektor zu erhalten kann offensichtlich das rechteckige Suchfenster nicht beibehalten werden und müsste durch einen radialen Bereich ersetzt werden. Außerdem scheint es sinnvoll, nahe am Zentrum (also der Position des Merkmalpunktes) gelegene Änderungen stärker zu bewerten als weiter außen liegende. Dies kann helfen, Bildrauschen aus der weiteren Umgebung schwächer einzubeziehen und so die Robustheit des Merkmals zu

erhöhen. Die Verwendung einer Gaussverteilung zur Gewichtung bietet sich hierfür an:

$$G(x, y, \sigma) = \frac{e^{-(x^2+y^2)/2\sigma^2}}{2\pi\sigma^2} \tag{4.2}$$

Es ergibt sich so die neue gewichtete Summe der quadratischen Unterschiede:

$$S_G(x, y, \sigma) = \sum_{x',y'} G(x' - x, y' - y, \sigma) \cdot (I(x' + \Delta x, y' + \Delta y) - I(x', y'))^2 \tag{4.3}$$

Dabei ist (x, y) das Zentrum des zu beurteilenden Bildbereiches, also die vermutete Merkmalsposition. Die Standardabweichung σ bestimmt, wie stark das weitere Umfeld mit einbezogen werden.

Für kleine $\Delta x, \Delta y$ lässt sich $I(x + \Delta x, y + \Delta y)$ über die Gradienten (partiellen Ableitungen) der Intensitäten $\delta I/\delta x$ und $\delta I/\delta y$ linear approximieren (Taylor-Reihe erster Ordnung):

$$I(x + \Delta x, y + \Delta y) \approx I(x, y) + \left(\frac{\delta I}{\delta x}(x, y) \cdot \Delta x + \frac{\delta I}{\delta y}(x, y) \cdot \Delta y\right)$$

Eingesetzt in Gleichung 4.3 ergibt dies:

$$\begin{aligned}
S_G &\approx \sum_{x',y'} G(x' - x, y' - y, \sigma) \cdot \left(I(x', y') + \frac{\delta I}{\delta x}(x', y') \cdot \Delta x + \frac{\delta I}{\delta y}(x', y') \cdot \Delta y - I(x', y')\right)^2 \\
&= \sum_{x',y'} G(x' - x, y' - y, \sigma) \cdot \left(\frac{\delta I}{\delta x}(x', y') \cdot \Delta x + \frac{\delta I}{\delta y}(x', y') \cdot \Delta y\right)^2 \\
&= \sum_{x',y'} G(x' - x, y' - y, \sigma) \\
&\quad \cdot \left(\left(\frac{\delta I}{\delta x}(x', y') \cdot \Delta x\right)^2 + 2\frac{\delta I}{\delta x}(x', y')\frac{\delta I}{\delta y}(x', y')\Delta x\Delta y + \left(\frac{\delta I}{\delta y}(x', y') \cdot \Delta y\right)^2\right) \\
&= \Delta x^2 \sum_{x',y'} \left(G(x' - x, y' - y, \sigma) \cdot \left(\frac{\delta I}{\delta x}(x', y')\right)^2\right) \\
&\quad + \Delta x\Delta y \sum_{x',y'} \left(G(x' - x, y' - y, \sigma) \cdot 2\frac{\delta I}{\delta x}(x', y')\frac{\delta I}{\delta y}(x', y')\right) \\
&\quad + \Delta y^2 \sum_{x',y'} \left(G(x' - x, y' - y, \sigma) \cdot \left(\frac{\delta I}{\delta y}(x', y')\right)^2\right) \\
&= \begin{pmatrix}\Delta x & \Delta y\end{pmatrix} \cdot M_H \cdot \begin{pmatrix}\Delta x \\ \Delta y\end{pmatrix}
\end{aligned} \tag{4.4}$$

Wobei M_H die Harris-Matrix ist, mit

$$M_H(x,y,\sigma) = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \overset{m_{12}=m_{21}}{=} \begin{pmatrix} m_{11} & m_{12} \\ m_{12} & m_{22} \end{pmatrix} \tag{4.5}$$

$$= \sum_{x',y'} \left(G(x'-x, y'-y, \sigma) \cdot \begin{pmatrix} \left(\frac{\delta I}{\delta x}(x',y')\right)^2 & \frac{\delta I}{\delta x}(x',y') \cdot \frac{\delta I}{\delta y}(x',y') \\ \frac{\delta I}{\delta x}(x',y') \cdot \frac{\delta I}{\delta y}(x',y') & \left(\frac{\delta I}{\delta y}(x',y')\right)^2 \end{pmatrix} \right)$$

Dabei können die Gradienten auf dem diskreten Bild z.B. wie folgt approximiert werden:

$$\frac{\delta I}{\delta x}(x,y) = I(x+1,y) - I(x-1,y) \text{ und } \frac{\delta I}{\delta y}(x,y) = I(x,y+1) - I(x,y-1)$$

Die Matrix H beschreibt die Struktur der Intensitäten im Umfeld des potentiellen Merkmals. Die Stärke der Intensitätsveränderungen in diesem Bereich wird durch die Größe der Eigenwerte λ_1, λ_2 dieser 2x2-Matrix beschrieben. Sie bilden, dank der Verwendung einer radialen Gewichtungsfunktion wie der Gaussverteilung, eine rotationsinvariante Beschreibung der Merkmalsgüte.

Drei verschiedene Fälle sind hierbei von Bedeutung:

- Beide Eigenwerte sind sehr klein, d. h. $\lambda_1 \approx 0$ und $\lambda_2 \approx 0$. Dies spricht für einen Bereich mit einheitlicher Intensität (wie in Abb. 4.1 Ausschnitt b))
- Einer der Eigenwerte ist sehr klein, während der andere ein großer positiver Wert ist, d. h. $\lambda_1 \approx 0 \wedge \lambda_2 >> 0$ oder $\lambda_1 >> 0 \wedge \lambda_2 \approx 0$. Dies spricht für eine Kante (wie in Abb. 4.1 Ausschnitt b)). Die Richtung der Kante ist orthogonal zum Eigenvektor des größeren Eigenwertes.
- Beide Eigenwerte sind große positive Werte, d. h. $\lambda_1 >> 0 \wedge \lambda_2 >> 0$. Dies spricht für ein gutes Merkmal (wie in Abb. 4.1 Ausschnitt a))

Harris und Stephens schlagen darauf basierend folgende Funktion zur Beurteilung der Qualität eines Merkmals vor:

$$Q = \lambda_1\lambda_2 - k(\lambda_1+\lambda_2)^2 = \det(M_H) - \text{trace}^2(M_H) \tag{4.6}$$

Dabei ist $\det(M_H) = m_{11}m_{22} - m_{12}^2$ die Determinante der Harris-Matrix und $\text{trace}(M_H) = m_{11} + m_{22}$ deren Spur. So sind die tatsächlichen Berechnungen der Eigenwerte nicht nötig, was rechnerisch effektiver ist. k ist hierbei ein Parameter, der bestimmt, wie negativ auf Kanten reagiert wird. Negative Werte sprechen für Kanten, Werte nahe Null für Bereiche mit einheitlicher Intensität und hohe positive Werte für gute

Merkmale.

Angenommen M_H wurde für jeden Bildpunkt und ein festes σ berechnet. Die tatsächliche Extraktion von Merkmalen könnte dann auf der Auswahl lokaler Maxima von Q (d. h. die umgebenden acht Pixel haben niedrigere Werte) basieren.

4.3 Kanade-Lucas-Tomasi

Sehr eng verwandt zu dem Harris-Detektor ist der *Kanade-Lucas-Tomasi-Detektor (KLT)* [27, 28, 29, 24]. Hier wird der kleinere der beiden Eigenwerte λ_1, λ_2 von M_H (siehe Gleichung 4.5) zur Bewertung von Merkmalen herangezogen. Hat dieser einen großen positiven Wert, so wird die betrachtete Position als gutes Merkmal angesehen. Dies entspricht der selben Idee, die auch der Bewertungsfunktion Q (siehe Gleichung 4.6) zugrunde liegt.

4.4 Der Skalierungsraum

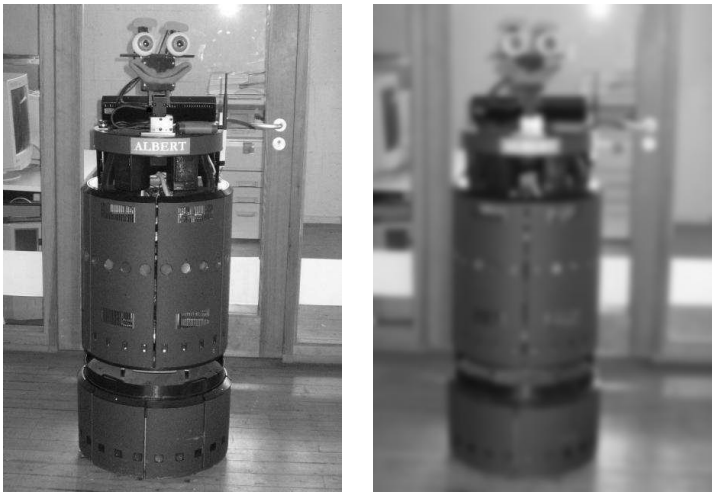
Bei der Merkmalsextraktion ist Größeninvarianz (=Skalierungsinvarianz) für SLAM-basierte Anwendungen sehr wichtig. Mit der Entfernung zu dem aufgenommenen Objekt in der Welt ändert sich dessen Größe im Bild. Um als Landmarke sinnvoll zu sein muss es aber dennoch in beiden Fällen als Merkmal erkannt werden.

In diesem Zusammenhang arbeiten viele Detektoren nicht direkt auf dem Bild $I(x, y)$, sondern auf der sogenannten *linearen Skalierungsraum-Repräsentation (linear scale-space representation)* [30, 31, 24]:

$$L(x, y, \sigma) = \sum_{x',y'} G(x' - x, y' - y, \sigma) \cdot I(x', y') \tag{4.7}$$

Dies entspricht der Faltung von G mit I: $L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$
Dabei ist G eine Gaussverteilung (siehe Gleichung 4.2).

L entspricht optisch einer weichgezeichneten/unscharfen Version des Originalbildes (siehe Abb. 4.2). Die Stärke der Unschärfe hängt von der Größe von σ ab.
Arbeitet man nun auf L statt auf I, so existiert mit σ ein Parameter, der bestimmt, mit welcher Skalierung gerade gearbeitet wird.
Zum intuitiven Verständnis: Der Punkt (x, y) in L ist eine Kombination der Bildpunkte in der Umgebung von (x, y) in I. Je größer σ, desto größer ist diese Umgebung. Betrachtet man also einen kleinen Ausschnitt von L, so enthält dieser Daten



(a) (b)

Abbildung 4.2: **(a)** Graubild mit Intensitäten $I(x, y)$ **(b)** Beispiel für eine zu (a) zugehörige Skalierungsraum-Repräsentation $L(x, y, \sigma)$

eines größeren Ausschnitts vom Originalbild.

4.5 Der Harris-Detektor für multiple Skalierungen

Mit der Definition des Skalierungsraumes ist es nun möglich, den Harrisdetektor für verschiedene Skalierungen zu definieren [32, 24]. Statt auf $I(x, y)$ arbeitet dieser auf $L(x, y)$ (siehe Gleichung 4.7).
Hierfür ergibt sich nun das Äquivalent zur normalen Harris-Matrix (siehe Gleichung 4.5):

$$M_H L(x, y, \sigma_1, \sigma_2) = \sum_{x',y'} G(x' - x, y' - y, \sigma_2) \tag{4.8}$$
$$\cdot \begin{pmatrix} \left(\frac{\delta L}{\delta x}(x', y', \sigma_1)\right)^2 & \frac{\delta L}{\delta x}(x', y', \sigma_1) \cdot \frac{\delta L}{\delta y}(x', y', \sigma_1) \\ \frac{\delta L}{\delta x}(x', y', \sigma_1) \cdot \frac{\delta L}{\delta y}(x', y', \sigma_1) & \left(\frac{\delta L}{\delta y}(x', y', \sigma_1)\right)^2 \end{pmatrix}$$

Dabei wird σ_1 auch als lokaler Skalierungsparameter (local scale parameter) bezeichnet. Er beschreibt die Stärke des im Voraus erfolgten Gaussschen Weichzeichnens.

σ_2 wird auch als Integrationsskalierungsparameter (integration scale parameter) bezeichnet und beschreibt die Fenstergröße. Diese beiden Parameter werden üblicherweise in ein Verhältnis $\gamma^2 \in [2, 4]$ zueinander gesetzt, d. h.: $\sigma_2 = \gamma^2 \sigma_1$.
So ergibt sich die zur normalen Bewertungsfunktion Q (siehe Gleichung 4.6) entsprechende Funktion $Q_L(x, y, \sigma_1)$, die für jeden Punkt x, y im Bild und σ_1 im Skalierungsraum beurteilt, wie gut dies als Merkmal geeignet wäre.

4.6 Hesse-Matrix-basierte Detektoren

Statt der Harris-Matrix arbeiten viele Detektoren auf der Hesse-Matrix von L.

Die Hesse Matrix ist für eine n-dimensionale Funktion $f(x_1, ..., x_n)$ definiert als

$$H(f) = \begin{pmatrix} \frac{\delta^2 f}{\delta x_1 \delta x_1} & \frac{\delta^2 f}{\delta x_1 \delta x_2} & \cdots & \frac{\delta^2 f}{\delta x_1 \delta x_n} \\ \frac{\delta^2 f}{\delta x_2 \delta x_1} & \frac{\delta^2 f}{\delta x_2 \delta x_2} & \cdots & \frac{\delta^2 f}{\delta x_2 \delta x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\delta^2 f}{\delta x_n \delta x_1} & \frac{\delta^2 f}{\delta x_n \delta x_2} & \cdots & \frac{\delta^2 f}{\delta x_n \delta x_n} \end{pmatrix}$$

Die Hesse-Matrix von L ist $H(L) = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix} = \begin{pmatrix} \frac{\delta^2 L}{\delta x \delta x} & \frac{\delta^2 L}{\delta x \delta y} \\ \frac{\delta^2 L}{\delta x \delta y} & \frac{\delta^2 L}{\delta y \delta y} \end{pmatrix}$

Unter Betrachtung von Gleichung 4.7 lässt sich dies auf die Ableitungen zweiten Grades der Gauss-Verteilungen zurückführen:

$$\frac{\delta^2 L}{\delta x \delta x} \overset{4.7}{=} \frac{\delta^2}{\delta x \delta x} \sum_{x',y'} G(x' - x, y' - y, \sigma) \cdot I(x', y')$$
$$= \sum_{x',y'} \frac{\delta^2 G}{\delta x \delta x}(x' - x, y' - y, \sigma) \cdot I(x', y') = \frac{\delta^2 G}{\delta x \delta x}(x, y, \sigma) * I(x, y)$$

Wobei $*$ der Faltungsoperator ist. Äquivalentes gilt für $\frac{\delta^2 L}{\delta x \delta y}$ und $\frac{\delta^2 L}{\delta y \delta y}$.

Damit gilt für die Hesse-Matrix:

$$H(L) = \begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix} = \begin{pmatrix} \frac{\delta^2 G(x,y,\sigma)}{\delta x \delta x} * I(x, y) & \frac{\delta^2 G(x,y,\sigma)}{\delta x \delta y} * I(x, y) \\ \frac{\delta^2 G(x,y,\sigma)}{\delta x \delta y} * I(x, y) & \frac{\delta^2 G(x,y,\sigma)}{\delta y \delta y} * I(x, y) \end{pmatrix} \tag{4.9}$$

Basierend auf der Hesse-Matrix stellte Lindeberg [30] zwei Methoden zur Merkmalsextraktion vor:

- Basierend auf der Spur: $\text{trace}(H(L)) = h_{11} + h_{22}$. Diese ist identisch mit dem Laplace-Operator $\nabla^2 L(x, y, \sigma)$. Dieser Wert reagiert mit stark positiven

(negativen) Werten auf dunkle (helle) Bereiche vor hellem (dunklem) Hintergrund, die ungefähr das Ausmaß σ haben. Allerdings sinkt diese Reaktion mit der Größe von σ, weshalb der normalisierte Laplace-Operator

$$\nabla^2_{norm} L(x, y, \sigma) = \sigma^2 \nabla^2 L(x, y, \sigma) \overset{4.9}{=} \sigma^2 \nabla^2 G(x, y, \sigma) * I(x, y) \tag{4.10}$$

als skalierungsinvariantes Maß für die Qualität als Merkmal verwendet wird. Diese Methode ist in der Literatur als *Laplacian of the Gaussian (LoG)* bekannt. Wie die Spur der Harris-Matrix reagiert dieser Operator allerdings auch stark auf Kanten, welche als Merkmal nicht unbedingt nutzbar sind (siehe Anfang dieses Kapitels).

- Basierend auf der Determinante: $\det(H(L)) = h_{11}h_{22} - h_{12}^2$. Dieser Wert kann nach einer Normalisierung ebenfalls zur Bewertung von Merkmalen genutzt werden:
  $\det_{norm}(H(L(x, y, \sigma))) = \sigma^4(\frac{\delta^2 L}{\delta x \delta x}(x, y, \sigma) \cdot \frac{\delta^2 L}{\delta y \delta y}(x, y, \sigma)) - \left(\frac{\delta^2 L}{\delta x \delta y}(x, y, \sigma))\right)^2$
  Diese Methode ist in der Literatur als *Determinant of the Hessian (DoH)* bekannt. Dieser Operator reagiert jedoch unempfindlicher auf Kanten. Er ergibt nur stark positive/negative Werte für signifikante Änderungen der Intensitäten in X- und Y-Richtung.

Beide Methoden zeigen gute Merkmale an den Stellen an, an denen Extrema der Werte bzgl. räumlicher Position und Skalierung, d. h. aller drei Parameter x, y und σ, auftreten.

4.7 Der SIFT-Detektor

SIFT steht für *Scale-invariant feature transform* [31]. Es extrahiert skalierungs- und rotationsinvariante Merkmale und beschreibt diese mittels eines 128-stelligen Deskriptorvektors. Dieses Verfahren gehört aktuell zu den für neue Anwendungen am häufigsten eingesetzten Methoden. Im Folgenden wird vorerst nur auf die Merkmalsextraktion eingegangen. Das Zustandekommen des Deskriptorvektors wird in Kapitel 4.10 beschrieben.

Die Merkmalsextraktion von SIFT basiert auf einer Approximation der LoG-Methode (siehe Kapitel 4.6). Diese basiert auf der Faltung der Differenz von Gaussverteilungen mit verschiedener Standardabweichung und dem Bild (*difference-of-Gaussian - DoG*), bzw. der Differenz zweier Skalierungsraum-Repräsentationen:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \tag{4.11}$$
$$= L(x, y, k\sigma) - L(x, y, \sigma) \tag{4.12}$$

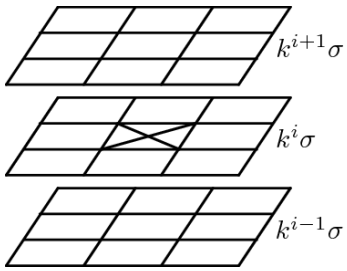


Abbildung 4.3: Zur Extremabestimmung zu überprüfende Stellen in den Bildern. Für die Position in der Mitte auf Skalierungsstufe $k^i\sigma$ müssen die umliegenden acht Punkte in der selben Stufe, sowie die neun Punkte in der jeweils höheren und niedrigeren Skalierungsstufe geprüft werden.

Lowe [31] zeigte, dass die folgende Approximation gilt:

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2\nabla^2 G \tag{4.13}$$

Und damit mit Gleichungen 4.10 und 4.11:

$$D(x, y, \sigma) \approx (k-1)\nabla^2_{norm} L(x, y, \sigma) \tag{4.14}$$

Sofern der Faktor k konstant für alle durchgeführten Skalierungsschritte ist, ändert der Faktor $(k-1)$ nichts an den Extrema von $\nabla^2_{norm} L(x, y, \sigma)$. Dadurch können Extrema von $D(x, y, \sigma)$ (bezüglich allen Parametern x, y, σ) als gute Merkmale angesehen werden.

In der Praxis wird dies wie folgt berechnet:
D wird für σ, $k\sigma$, $k^2\sigma$, ..., $k^n\sigma$ berechnet. k ist so gewählt, dass $k^n = 2$. Nun wird die Auflösung des Originalbildes halbiert und basierend auf diesem $I'(x, y)$ von vorne begonnen. Es wird also in einer Pyramidenstruktur gearbeitet, bei der die Bildgröße nach einer festen Anzahl von Schritten halbiert wird. Gestoppt wird bei einer Mindestgröße des Bildes.
Die besten Werte für n und σ wurde von den Autoren mittels Versuchen bestimmt.

Extrema liegen an den Stellen vor, an denen der Wert von D in einem Punkt (x, y) mit Skalierung $k^i\sigma$ minimal/maximal ist. Dies bezüglich der acht umliegenden Punkte im selben Bild und der jeweils neun umliegenden Punkte in den Bildern mit Skalierung $k^{i-1}\sigma$, bzw. $k^{i+1}\sigma$ (siehe Abb. 4.3). Sofern $|D|$ an diesen Stellen ein gewisses Maximum übersteigt wird die Position in Bild/Skalierungsraum als mögliches Merkmal in Betracht gezogen. Wie bereits vorher erwähnt ist der LoG und damit natürlich auch die hier verwendete Approximation, DoG, anfällig für Kanten. Daher wird für die gefundenen Punkte jeweils noch die Determinante der Hesse-Matrix berechnet und geprüft (siehe Kapitel 4.6). Punkte die diesen Test bestehen werden als gute Merkmale akzeptiert.

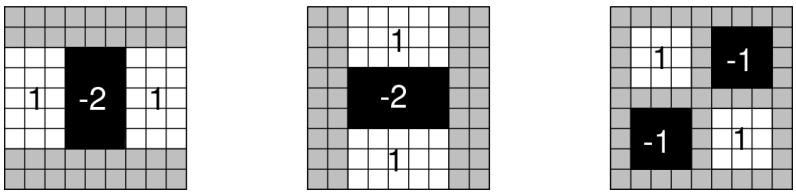


Abbildung 4.4: Beispiel für die Approximationen für (von links nach rechts) $\frac{\delta^2 G(x,y,\sigma)}{\delta x \delta x}$, $\frac{\delta^2 G(x,y,\sigma)}{\delta y \delta y}$ und $\frac{\delta^2 G(x,y,\sigma)}{\delta x \delta y}$ mit $\sigma = 1.2$. Alles außerhalb der weißen/schwarzen Bereiche ist gleich Null.

Weiterhin werden noch Interpolationsmethoden genutzt, um die Position der Merkmale mit Genauigkeiten kleiner einem Pixel zu bestimmen.

4.8 Der SURF-Detektor

SURF steht für *Speeded Up Robust Features* und ist ein Verfahren, dass 2006 von *Bay et al* [33] vorgestellt wurde.
Wie bei SIFT werden hier skalierungs- und rotationsinvariante Merkmale extrahiert, für die jeweils ein Deskriptorvektor berechnet wird (siehe Kapitel 4.11).
Wie der Name schon sagt, sollen hier Merkmale berechnet werden, die in ihrer Robustheit mit aktuellen Verfahren wie SIFT vergleichbar sind, dabei aber bedeutend schneller berechnet werden können. Hierzu wird eine Approximation der Hesse-Matrix verwendet, die auf Basis von Integralbildern berechnet wird. Das Integralbild I_Σ eines Intensitätsbildes I ist definiert als $I_\Sigma(x,y) = \sum_{i=0}^{x}\sum_{j=0}^{y} I(i,j)$.
Diese ermöglichen die Berechnung von Summen über rechteckige Bereiche beliebiger Größe in I mit nur vier Additionen:
Für ein Rechteck mit linker oberer Ecke (l,o) und rechter unterer Ecke (r,u) gilt $\sum_{i=l}^{r}\sum_{j=o}^{u} I(i,j) = I_\Sigma(r,u) - I_\Sigma(r,o) - I_\Sigma(l,u) + I_\Sigma(l,o)$.

Zur effektiven Berechnung der Hesse-Matrizen werden die partiellen Ableitungen zweiter Ordnung der Gaussverteilung meist in einer begrenzten und diskretisierten Form verwendet. *Bay et al* verwenden eine Form hiervon, die sie als *box filters* bezeichnen (siehe Abb. 4.4).
Diese machen den Einsatz der Integralbilder sinnvoll, da die einzelnen Elemente der Hessematrix so als Summen über Rechtecke dargestellt werden können, die jeweils mit den Werten aus der Approximation gewichtet werden. Auf der so erlangten Hesse-Matrix wird die Determinante berechnet und zur Bewertung der Merkmale eingesetzt. Bei der SURF-Merkmalsextraktion handelt es sich also um eine Ap-

proximation der DoH-Methode (siehe Kapitel 4.6). Die Autoren bezeichnen dieses Verfahren als *Fast-Hessian*.
Das stufenweises Verkleinern des Bildes für jede Verdopplung von σ, also eine Pyramidenstruktur wie bei SIFT, ist nicht notwendig zur Berechnung der einzelnen Skalierungsstufen, da die Anwendung des Filters unabhängig von der Skalierung in konstanter Zeit verläuft.
Entsprechend der DoH-Methode werden Extrema bzgl. der normalisierten Determinante als Merkmale extrahiert. Mittels Interpolationsmethoden werden danach deren Positionen in Bild und Skalierungsraum noch genauer bestimmt.

4.9 Beschreibung und Vergleich von Merkmalen

Am Anfang dieses Kapitels wurden die zwei Schritte zur Nutzung von Bildmerkmalen erläutert. Der erste Schritt war die Extraktion informationsreicher Bereiche im Bild. Der zweite die Beschreibung dieser Merkmale, so dass ein Ähnlichkeitsvergleich möglich wird. Letzteres soll nun im Folgenden behandelt werden.

Die einfachste Methode ist es, direkt die Bilddaten miteinander zu vergleichen. D. h., das Bildfenster mit dem Merkmal kann über das Vergleichsbild gelegt und schrittweise verschoben werden, bis die Intensitäten an jeder Stelle etwa übereinstimmen. Selbiges kann auch in Form einer stufenweisen Suche implementiert werden. Hierzu wird die Suche erst auf Bildern mit verkleinerter Auflösung durchgeführt. Im jeweils besten Bereich wird dann die Auflösung erhöht und der Vergleich wiederholt bis die Originalauflösung der Bilder erreicht ist. Dies ist laufzeittechnisch effektiver und führt zu einem robusteren Merkmalsvergleich [25]. Zusätzliche Informationen können hierbei leicht integriert werden. Sind beispielsweise Betrachtungswinkel (näherungsweise) bekannt, können die Bilder vor dem Vergleich dementsprechend transformiert werden. Vorwissen über die Kamerabewegung kann außerdem genutzt werden, um den Suchbereich innerhalb der Bilder erheblich einzuschränken [14].
Ergebnisse aus dem Merkmalsdetektor können hierfür natürlich auch genutzt werden, so dass beispielsweise nur Bereiche verglichen werden die einen ähnlichen Wert in der Bewertung des Harrisdetektors hatten. Oder Wissen über die Skalierung der Merkmale kann für die vorherige Transformation der Bildbereiche genutzt werden.

Andere Methoden beschreiben Merkmale in einer Art und Weise, dass diese effektiv miteinander verglichen werden können. So kann darauf verzichtet werden, die Bilddaten zwecks weiterem Vergleich zu behalten.

So werden SIFT-Merkmale[31] mittels eines 128-elementigen *Deskriptorvektor* beschrieben, der invariant bezüglich Skalierung und Rotation ist. Der Vergleich von zwei Merkmalen wird reduziert auf die Berechnung der euklidischen Distanz zwi-

schen den beiden Deskriptorvektoren. Diese Distanz wird im Folgenden als Deskriptorendistanz bezeichnet. Je größer diese ist, desto verschiedener sind die Merkmale.

4.10 SIFT-Deskriptorvektoren

Diese Deskriptoren werden wie folgt berechnet:

Nachdem ein Merkmal M extrahiert wurde, ist dessen Position im Bild (x_M, y_M), sowie seine Skalierungsstufe σ_M bekannt (siehe Kapitel 4.7). Anhand der Skalierung wird das zugehörige Gauss-weichgezeichnete Bild aus dem Extraktionsschritt ausgewählt und für die weiteren Berechnungen verwendet. Hierdurch wird die Berechnung des Deskriptors skalierungsinvariant.

Um Rotationsinvarianz zu erhalten muss nun zuerst eine Orientierung für das Merkmal bestimmt werden. Für einen gewissen Bereich um das Merkmalszentrum werden hierzu für jeden Punkt die Gradientenmagnitude

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y,+1) - L(x,y-1))^2}, \quad (4.15)$$

sowie die Orientierung

$$\Omega(x,y) = \arctan((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))) \quad (4.16)$$

berechnet.

Diese Orientierungen werden in ein 36-stelliges Histogramm eingeordnet, wobei jede Stelle 10° des 360°-Spektrums abdeckt. Beim Einfügen werden die Orientierungen anhand ihrer Gradientenmagnitude, sowie einer Gaussverteilung mit Standardabweichung $1.5\sigma_M$ gewichtet.

Die höchste Spitze im Histogramm repräsentiert die Orientierung. Mehrere Orientierungen sind möglich, wenn weitere Spitzen im Bereich von 80% der höchsten Spitze vorhanden sind. In diesen Fällen wird ein weiteres Merkmal mit selber Position und Skalierung aber anderer Orientierung eingeführt.

Die Berechnung des Deskriptors erfolgt ebenfalls auf den Gradientenmagnituden und Orientierungen der das Zentrum umgebenden Punkte. Die Gradienten, sowie deren Positionen, werden anhand der soeben berechneten Merkmalsorientierung gedreht, so dass das Ergebnis rotationsinvariant ist.

Um das Zentrum wird ein Gebiet von 16x16 Gradienten und Magnituden berechnet. Die Elemente von jeweils 4x4er Blöcken werden zu einem 8-stelligen Histogramm

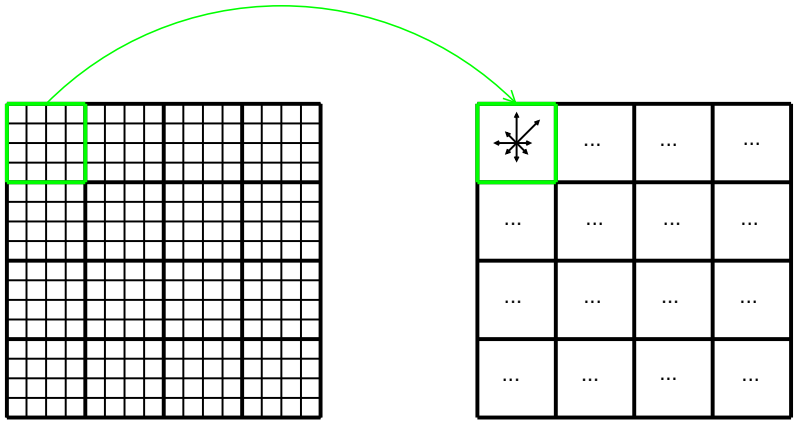


Abbildung 4.5: Skizze für die Berechnung des SIFT-Deskriptors. Der 16x16-Bereich links wird in 4x4 Blöcke unterteilt. Die einzelnen Orientierungen innerhalb dieser Blöcke tragen jeweils (gewichtet) zu einem 8-stelligen Histogramm (45°-Schritte) bei (rechts)

(45°-Schritte) hinzugefügt, gewichtet anhand ihrer Magnituden und einer Gaussverteilung um das Merkmalszentrum. Damit kleine Verschiebungen im Bild keine Sprünge in den Histogrammen auslösen, erfolgt außerdem eine Interpolation, so dass einzelne Gradienten sich auch auf die Histogramme der umliegenden Bereiche auswirken.

Die 16 jeweils 8-stelligen Histogramme bilden nun den 128-stelligen ($= 16 \cdot 8$) Deskriptorvektor. Jeder Eintrag entspricht der Stärke der einzelnen Orientierungen (der Länge der Pfeile). Da nur mit Gradienten gearbeitet wird ist das Ergebnis invariant bezüglich Helligkeitsänderungen (ein konstantes Intensitätsoffset: $I'(x, y) = I(x, y) + c$) und um Invarianz bezüglich Kontraständerungen (Multiplikation mit einem konstanten Faktor: $I'(x, y) = k \cdot I(x, y)$) zu erreichen wird der Deskriptorvektor normalisiert.
In der Realität machen Beleuchtungsverhältnisse trotzdem einen Unterschied. Bei realen Kameras kann z.B. eine Sättigung der Intensitäten erreicht werden (komplett schwarz oder weiß). Außerdem ändert sich die Beleuchtung meist nicht-linear mit der Position der Kamera relativ zu der Lichtquelle.

4.11 SURF-Deskriptorvektoren

SURF [33] arbeitet wie SIFT mit Deskriptorvektoren, die das Merkmal rotations- und skalierungsinvariant beschreiben. Deren Berechnung erfolgt ähnlich wie die der SIFT-Deskriptoren. Für die zuvor berechneten Merkmalspositionen (siehe Kapi-

tel 4.8) wird wieder zuerst eine Orientierung berechnet. Diese Berechnung ist sehr ähnlich zu der von SIFT. Es werden allerdings wie schon bei der Extraktion der Merkmale die Integralbilder eingesetzt um Bereiche anhand der Skalierung zusammenzufassen. Dies im Gegensatz zu SIFT, wo auf Bildern mit verschiedener Skalierungsstufen gearbeitet wird. Die Gradientenmagnituden und Orientierungen werden somit nicht auf einzelnen Pixeln berechnet, sondern auf Summen von rechteckigen Bereichen, deren Größen jeweils von der Skalierung des Merkmals abhängen. Diese Berechnung kann Dank der Integralbilder wieder sehr effektiv und unabhängig von der Skalierung in konstanter Zeit berechnet werden.

Zur Berechnung des Deskriptors wird nun ein quadratischer Bereich, mit einer von der Skalierung abhängigen Größe, um das Merkmalszentrum gelegt. Zuvor wurde das Merkmal anhand der Orientierung gedreht, so dass der Deskriptorvektor rotationsinvariant ist. Dieses Quadrat wird in 4x4 gleichgroße Regionen aufgeteilt. Für diese Regionen werden jeweils 5x5 approximierte Gradienten in X- und Y-Richtung berechnet. Die Abschätzung dieser Gradienten erfolgt jeweils durch die Subtraktion der Intensitätssummen zweier angrenzender Rechtecke. Die Gradienten, jeweils für X- und Y-Richtung, werden nun anhand einer Gaussverteilung gewichtet. Diese gewichteten Gradienten seien $\{dx_1, ..., dx_{25}\}$, bzw. $\{dy_1, ..., dy_{25}\}$. Jedes der 4x4 Quadrate trägt nun mit vier Elementen zu dem Deskriptorvektor bei:
$(\sum_i dx_i, \sum_i dy_i, \sum_i |dx_i|, \sum_i |dy_i|)$.

So ergibt sich ein 64-elementiger Deskriptorvektor. Wie der SIFT-Deskriptor ist dieser bereits invariant bezüglich Helligkeitsänderungen und wird normalisiert um invariant bezüglich Kontraständerungen zu sein. Die selben Einschränkungen wie bei SIFT bezüglich realen Lichtquellen und Kameras gelten hier natürlich auch.

Die Verwendung von Deskriptorvektoren vereinfacht den Aufbau von Merkmalskarten enorm, da die eigentlichen Bilddaten nicht weiter gespeichert werden müssen. Bezüglich der Wiedererkennungsrate ist SURF mit SIFT vergleichbar, die Extraktion der Merkmale ist jedoch weniger rechenaufwendig. Durch den nur 64-elementigen Deskriptorvektor ist außerdem der Aufwand zum Vergleichen zweier Merkmale geringer als für die 128-elementigen SIFT-Deskriptoren. Aus diesen Gründen wurde für diese Arbeit entschieden, mit SURF-Merkmalen zu arbeiten. Die Software hierzu ist für nicht-kommerzielle Anwendungen frei verfügbar. Für jedes Merkmal werden unter anderem Skalierung, Orientierung und Deskriptorvektor ausgegeben.
Durch Angabe von Mindestwerten für die approximierte Determinante der Hessematrix lässt sich die Anzahl der extrahierten Merkmale einschränken. In der Implementierung zu dieser Arbeit wird dieser Wert automatisch angepasst, so dass pro Bild zwischen 50 und 100 Merkmale extrahiert werden.

Die *Deskriptorendistanz* bezeichnet im Folgenden die euklidische Distanz zwischen

den Deskriptorvektoren $\mathbf{d} = (d_1 \ldots d_n)^T$ und $\mathbf{d}' = (d'_1 \ldots d'_n)^T$ zweier SURF-Merkmale: $\sqrt{\sum_1^n (d_i - d'_i)^2}$. In der Implementation zu dieser Arbeit wurde auf das Ziehen der Wurzel verzichtet und statt dessen mit der quadrierten euklidischen Distanz gerechnet um mögliche Übereinstimmungen zu finden. Dies ist für diesen Zweck äquivalent, jedoch weniger aufwendig zu berechnen. Eine zusätzliche Optimierung des Programmcodes, wie z.B. Aufrollen der Schleife, ist hier besonders wichtig, da diese Vergleiche enorm häufig berechnet werden müssen.

Im Folgenden werden die Begriffe Landmarken und (Bild-)Merkmale als äquivalent verwendet werden.

Es ist noch zu erwähnen, dass keine der hier erläuterten Merkmalstypen invariant bezüglich affinen Verzerrungen sind. Diese können beispielsweise aus verschiedenen Aufnahmewinkeln resultieren. Es gibt Ansätze hierzu (siehe z.B. [34]), allerdings hat die zusätzliche Komplexität von affin-invarianten Merkmalen oft eine negative Auswirkung auf die Robustheit und zahlt sich nur für sehr große Änderungen des Sichtwinkels aus [31].
Die allgemeine Robustheit von z.B. SIFT oder SURF ermöglicht das Wiedererkennen von Merkmalen auch unter verschiedenen Perspektiven, sofern diese sich nicht zu extrem unterscheiden.

5 Die Merkmalskarte

Im letzten Kapitel wurde dargestellt, wie Merkmale aus dem Kamerabild extrahiert werden. Weiterhin wurden Verfahren vorgestellt, die jedem Merkmal einen Deskriptorvektor zuordnen und damit effektive Ähnlichkeitsvergleiche zwischen Merkmalen ermöglichen. Basierend darauf soll in diesem Kapitel die Idee beschrieben werden, wie man mit Hilfe der bisher vorgestellten Techniken eine (Landmarken-) Karte aufbauen kann. In den hierauf folgenden Kapiteln wird dann detailliert auf die Verfahren eingegangen, die für eine konkrete Realisierung des Gesamtsystems notwendig sind.

5.1 Merkmale im Weltsystem

Wie in Kapitel 3.5 erläutert, können die aus den Bildern extrahierten SURF-Merkmale unter jeweils bekannter Kameraposition in das Weltsystem projiziert werden.

Neben der Position der Merkmale ist es auch sinnvoll, die Orientierung der SURF-Merkmale zu verwenden. So können ähnlich aussehende Bereiche in der Welt, die jedoch verschiedene Orientierungen haben, besser unterschieden werden. Hierzu muss die Orientierung ebenfalls in das Weltsystem projiziert werden. Es ist also die Orientierung des Merkmals in der XY-Ebene der Welt gesucht. Hierbei wird die Annahme gemacht, dass das Merkmal auf einer Fläche parallel zum Boden liegt.
Sei α die Orientierung des Merkmals im Bild. Wir betrachten nun den Punkt $\mathbf{a} = (\cos\alpha \ \sin\alpha \ 1)^T$ im Kamerasystem. Dieser bildet mit dem Punkt $\mathbf{o} = (0\ 0\ 1)^T$ eine Strecke mit der Orientierung α. Nun werden a und o anhand der bekannten Kameraorientierung R im Raum rotiert:

$$\mathbf{a}' = \begin{pmatrix} x_{a'} \\ y_{a'} \\ z_{a'} \end{pmatrix} = R \cdot \mathbf{a} \text{ und } \mathbf{o}' = \begin{pmatrix} x_{o'} \\ y_{o'} \\ z_{o'} \end{pmatrix} = R \cdot \mathbf{o}$$

Diese Punkte normalisieren wir nun, so dass sie auf der selben Höhe liegen:

$$\mathbf{a}'' = \begin{pmatrix} x_{a''} \\ y_{a''} \end{pmatrix} = \begin{pmatrix} x_{a'}/z_{a'} \\ y_{a'}/z_{a'} \end{pmatrix} \text{ und } \mathbf{o}'' = \begin{pmatrix} x_{o''} \\ y_{o''} \end{pmatrix} = \begin{pmatrix} x_{o'}/z_{o'} \\ y_{o'}/z_{o'} \end{pmatrix}$$

Die Orientierung der sich aus diesen Punkten ergebenden Strecke entspricht nun der gesuchten Merkmalsorientierung in der Welt:

$$\alpha' = \arctan2(y_{a''} - y_{o''},\ x_{a''} - x_{o''})$$

5.2 Erstellung und Visualisierung der Karte

Nachdem nun Positionen und Orientierungen der Merkmale in der Welt bekannt sind, können sie abgespeichert werden. In der Implementierung zu dieser Arbeit wird der Boden hierzu in quadratische Bereiche konstanter Größe aufgeteilt (Kästchenwelt). Die Merkmale werden unabhängig von ihrer z-Koordinate in dem jeweils zugehörigen Feld abgelegt. Dies erleichtert die spätere Suche nach Merkmalen in einem lokal eingeschränkten Bereich.

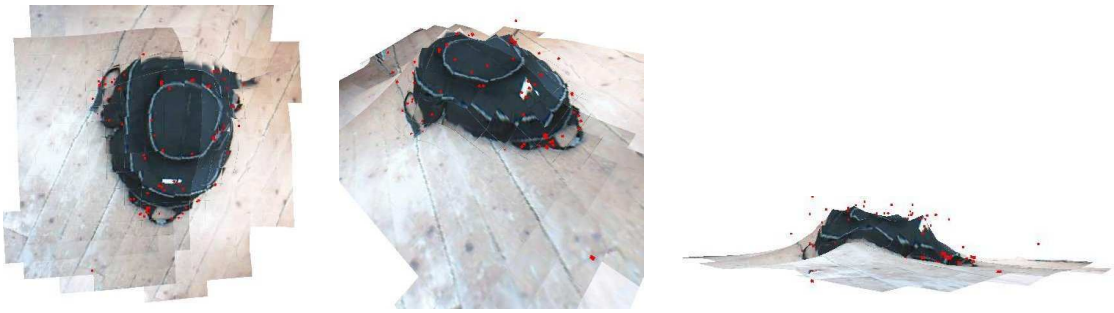
Außerdem ermöglicht dieser Aufbau die Erstellung einer texturierten Oberflächenkarte. Den einzelnen Quadraten wird hierzu die durchschnittliche Höhe der beinhalteten Merkmale zugeordnet. Die Höhe von Feldern, die keine Merkmale beinhalten, wird aus den umliegenden Feldern interpoliert. Aus den zugehörigen Kamerabildern kann durch Projektion der einzelnen Pixel in die Welt eine Textur für die Felder berechnet werden. In Abbildung 5.1 ist ein Beispiel für die sich so ergebende Visualisierung der Karte zu sehen. Weitere Beispiele sind bei den Experimenten in Kapitel 8 zu finden.

Der schrittweise Aufbau der Karte geschieht wie folgt:

- Beim Start der Exploration wird die Kameraposition mit $(0\ 0\ 0)^T$ für die Translation und den Daten des Lagesensors als Orientierung initialisiert. Die Merkmale des Bildes werden in die Welt projiziert und dementsprechend in der Karte abgespeichert.
- Für jedes weitere Kamerabild:
  - Die Berechnung der Kameraposition erfolgt durch das Finden von übereinstimmenden Merkmalen (*matches*) zwischen Bildmerkmalen und den gespeicherten Merkmalen in der Karte. Siehe hierzu Kapitel 5.3 und Kapitel 6.
  - Mittels dieser geschätzten Position werden die Merkmale des Bildes in die Welt projiziert und dementsprechend in der Karte abgelegt. Es werden nur Merkmale abgespeichert, für die keine Übereinstimmung in der Karte gefunden wurde. Hierdurch soll verhindert werden, dass das selbe Merkmal mehrmals in der Karte vorkommt.
  - Sofern die Visualisierung aktiviert ist, werden neue Texturen für die Oberflächenkarte aus dem Kamerabild extrahiert.

5.3 Kandidaten für den Merkmalsvergleich

Die Wahl des Verfahrens zur Berechnung der globalen Position ist abhängig von der Wahl der Merkmale in der Karte, die für den Vergleich mit den Bildmerkmalen



(a) (b) (c)

Abbildung 5.1: Beispiel für die sich ergebenden texturierten Oberflächenkarten. Hier ist ein Rucksack auf einem Holzboden zu sehen. Die Höhe der Merkmale wurde mittels einer Stereokamera ermittelt. Durch Aufteilung der Texturen in verbundene Dreiecke wird eine geschlossene Fläche gebildet. Die Visualisierung erfolgt mittels OpenGL. Die roten Punkte markieren die Positionen der Merkmale. Die Texturen für den dargestellten Bereich stammen teilweise aus verschiedenen Kamerabildern. Die Übergänge von dunkel nach hell resultieren aus der automatischen Belichtungsanpassung der Kamera. **(a)** Ansicht von oben **(b)** Ansicht von schräg oben **(c)** Ansicht von der Seite.

verwendet werden. In dieser Arbeit werden für jedes Bild drei verschiedene Mengen von Merkmalen für den Vergleich herangezogen:

- **Visuelle Odometrie:** Hierfür werden nur Merkmale aus zeitlich nahe gelegenen Bildern für den Vergleich genutzt. In der Implementierung zu dieser Arbeit typischerweise die letzten fünf Bilder. Dabei ist anzumerken, dass für diesen Vergleich alle aus diesen Bildern extrahierten Merkmale genutzt werden, nicht nur die, die wirklich in die Karte übernommen wurden. Vorteil dieser Methode ist es, dass mit einer hohen Wahrscheinlichkeit die Veränderung der Kameraposition errechnet werden kann, da die Bilder sehr ähnlich sind. Nachteilig ist, dass sich über längere Zeit kleine Fehler aufaddieren, was zu einem Drift weg von der tatsächlichen Position führt.

- **Lokale Lokalisierung:** Hier wird in einem gewissen Umkreis um die aus der Odometrie bekannten Position nach übereinstimmenden Merkmalen gesucht. Hierdurch können kleine Schleifen geschlossen werden und beim Zurückkehren in bekannte Gebiete wird so der Drift-Fehler korrigiert. Es kann natürlich vorkommen, dass hier einfach die selbe Position gefunden wird wie für die visuelle Odometrie. Dies ist sogar wahrscheinlich, da die Merkmale der letzten paar Bilder auch in diesem Bereich zu finden sind und typischerweise die besten Übereinstimmungen liefern. Daher muss noch ein Kriterium angelegt werden, dass nur ältere Merkmale in Betracht gezogen werden. In der Implementierung zu dieser Arbeit wurde hierfür jeweils geprüft, welches die ältesten Merkmale

in dem lokalen Bereich der Karte sind. Angenommen, diese Merkmale stammen aus dem m-ten verarbeiteten Kamerabild und das aktuelle Kamerabild hat die Nummer n. Dann werden nur diejenigen Merkmale aus dem Kartenausschnitt für den Vergleich genutzt, die älter als das Kamerabild mit der Nummer $(n + m)/2$ sind.

- **Globale Lokalisierung:** Angenommen, die Kamera erreicht nach einem langen Weg einen bereits bekannten Bereich. Dann kann es passieren, dass der aufaddierte Fehler der Odometrie die Größe des Suchbereichs für die lokale Lokalisierung übersteigt. Es wäre jedoch zu aufwendig, einen kompletten paarweisen Vergleich aller Bildmerkmale mit allen Kartenmerkmalen durchzuführen. Wir reduzieren diesen Aufwand durch Einsatz einer Zwei-Phasen-Suche. In der ersten Phase wird nur ein einzelnes Bildmerkmal (das Referenzmerkmal) mit allen Kartenmerkmalen verglichen. Als Referenzmerkmal wird das Merkmal gewählt, das in der visuellen Odometrie die beste Übereinstimmung aufwies. Wird für dieses Merkmal eine mögliche Übereinstimmung gefunden, so wird eine lokale Suche in diesem Bereich ausgeführt.
  Kartenmerkmale die bereits in der Odometrie oder der lokalen Suche betrachtet wurden, werden hier außer Acht gelassen.

In Kapitel 6 wird beschrieben, wie mittels solcher Teilmengen der Kartenmerkmale die Kameraposition geschätzt wird. Hierbei wird außerdem eine Güte für die geschätzte Position bestimmt.

In unserer Implementierung wird jeweils zu den Ergebnissen der lokalen Lokalisierung, sofern vorhanden, „gesprungen“. Ansonsten wird das Ergebnis der Odometrie verwendet. Es werden aber alle Daten der Odometrie, sowie der lokalen und globalen Lokalisierung gespeichert. In Kapitel 7 wird ein Optimierungsalgorithmus beschrieben, der versucht, die für diese Daten wahrscheinlichste Trajektorie zu rekonstruieren.
Sollte es passieren, dass keine der drei Möglichkeiten eine ausreichend gute Positionsschätzung liefert, z.B. weil das Kamerabild stark gestört ist oder eine zu schnelle Bewegung gemacht wurde, so wird von keiner Positionsänderung gegenüber dem letzten verarbeiteten Bild ausgegangen. Für die Orientierung werden die Daten des Lagesensors genutzt.

6 Rekonstruktion der Kameraposition

In den bisherigen Kapiteln wurde zuerst erläutert, wie sich Aufnahmen einer Kamera geometrisch beschreiben lassen. Im Weiteren wurde dargestellt, wie aus solchen Bildern Merkmale extrahiert werden können, die es dem System ermöglichen, Bereiche im Bild wiederzuerkennen. Außerdem wurde beschrieben, wie mittels solcher Merkmale eine Karte aufgebaut werden kann, in der diese als Landmarken verwendet werden. Im Folgenden wird nun erläutert, wie mit diesen Mitteln eine unbekannte Kameraposition geschätzt werden kann.

Für jedes aufgenommene Kamerabild muss die neue Position der Kamera auf Basis der Landmarken in der Karte, sowie der Daten des Lagesensors rekonstruiert werden. Der Lagesensor (siehe Kapitel C.2 im Anhang) liefert Yaw, Pitch und Roll mit einer Rate von 100 Hz. Dadurch stehen zu jedem Kamerabild zeitnahe Messungen bereit. Im Folgenden sei ($\text{Yaw}_{\text{Sensor}}$, $\text{Pitch}_{\text{Sensor}}$, $\text{Roll}_{\text{Sensor}}$) die durch Interpolation der Messungen erlangte Orientierung zum Zeitpunkt des aktuellen Kamerabildes. Es hat sich herausgestellt, dass die Messungen von Pitch und Roll hinreichend genau sind. Deshalb können diese Werte ungefiltert direkt übernommen werden. Da der Yaw-Winkel sensorintern mit Hilfe des Erdmagnetfeldes gemessen wird, ist dieser Wert anfällig gegenüber Störungen, die sich in einem veränderten Magnetfeld widerspiegeln. Dies ist beispielsweise in der Nähe von elektrischen Geräten der Fall. In solchen Bereichen wird eine Messverschiebung in Richtung des fehlerhaften Nordpols bewirkt. Daher ist dieser Wert nicht verlässlich genug, weshalb eine Korrektur mittels der Bilddaten notwendig ist. Allerdings sind die gemessenen Änderungen des Yaws zwischen zwei Kamerabildern gut genug, um sie als Schätzung für die neue Position zu übernehmen. D. h. für zwei aufeinanderfolgende Kamerabilder zu den Zeitpunkten $t-1$ und t kann $\text{Yaw}'_{\text{Sensor}}(t) = \text{Yaw}(t-1)+(\text{Yaw}_{\text{Sensor}}(t)-\text{Yaw}_{\text{Sensor}}(t-1))$ als Basis für die Berechnung von $\text{Yaw}(t)$ genutzt werden.

Sind noch keine Landmarken in der Karte vorhanden, so wird das System mit der Position (0, 0, Starthöhe), sowie den Lagesensordaten zum Zeitpunkt dieses ersten Kamerabildes initialisiert. Die Starthöhe muss bei Monovision vorgegeben sein. Bei Stereovision kann sie aus den Bildern berechnet werden (siehe Kapitel B).

Für den in dieser Arbeit verwendeten Algorithmus zur Berechnung der Kamerapo-

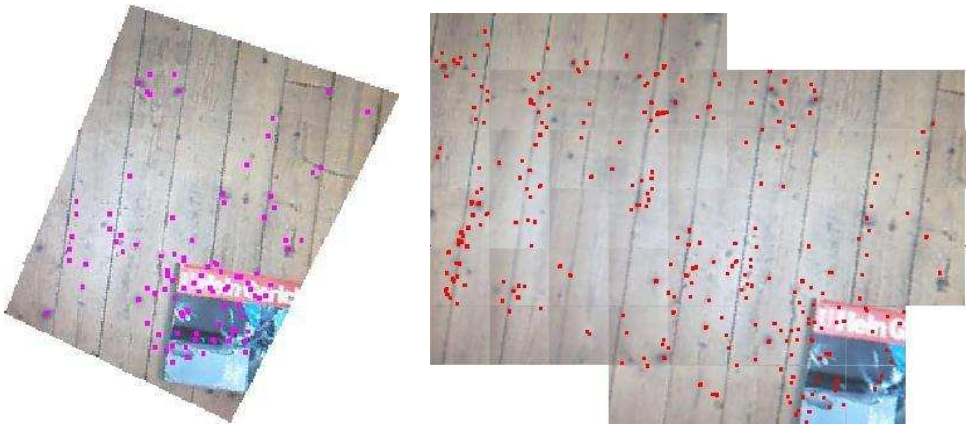
sition anhand von gefundenen Merkmalsübereinstimmungen wird das Prinzip von *RANSAC*, bzw. *PROSAC* aufgegriffen. Diese Methoden werden im Folgenden kurz dargestellt, um die zu Grunde liegende Idee zu vermitteln. Danach wird unsere Anwendung hiervon zuerst in einer Kurzfassung beschrieben und danach im Detail und anhand eines Beispiels erläutert.

6.1 RANSAC & PROSAC

RANSAC[35] steht für *'Random Sample Consensus'*. Es handelt sich hierbei um eine Methode, eine geometrische Transformation zwischen zwei Punktmengen zu finden. Dies auch dann, wenn die Zuordnung zueinandergehöriger Punkte stark mit Ausreißern belastet ist. Sei m die Mindestanzahl an übereinstimmenden Punkten, die notwendig ist, um die Transformation zu berechnen. Es wird zufällig eine Menge dieser Größe ausgewählt und anhand dieser die Tranformation berechnet. Diese Tranformation wird nun genutzt, um die beiden Punktemengen ineinander zu überführen. Die Güte der Transformation wird daran bestimmt, wie gut die Punktemengen nach der Transformation aufeinander passen. Ist dieses Ergebnis zufriedenstellend wird abgebrochen. Ansonsten wird eine neue Menge von m Punkten ausgewählt und von vorne begonnen.

PROSAC[36] steht für *Progressive Sample Consensus*. Dieses Verfahren greift die Idee von RANSAC auf, verwendet jedoch zusätzliches Wissen über die Güte der bekannten Punkteübereinstimmungen. Angenommen es kann eine Wahrscheinlichkeit für Paare aus den beiden Punktmengen gegeben werden, dass diese zusammengehörig sind. Dann werden die m Paare nicht rein zufällig gewählt, sondern anhand einer Wahrscheinlichkeitsverteilung gezogen, die diesem Vorwissen entspricht. So erhöht sich die Chance, früher eine korrekte Transformation zwischen den Punktemengen zu finden.

In unserer Anwendung sind die beiden Punktemengen die Merkmale im Kamerabild und in der Karte. Die gesuchte Transformation entspricht dem Wissen über die Kameraposition. Dank der Daten des Lagesensors genügen zwei übereinstimmende Merkmale aus Karte und Bild um die Kameraposition zu berechnen (siehe Kapitel A im Anhang). Bei uns ist also $m = 2$. Die Güte der Merkmalsübereinstimmung kann unter anderem an der Deskriptorendistanz der Merkmale gemessen werden, so dass sich die Anwendung von PROSAC anbietet. Wir verzichten allerdings auf das probabilistische Ziehen, sondern ordnen die Merkmalsübereinstimmungen anhand ihrer Güte und wählen die Paare in der sich so ergebenden Reihenfolge.



(a) (b)

Abbildung 6.1: **(a)** Alle Positionen der in einem Kamerabild gefundenen Merkmale. Das Bild zeigt einen Holzboden, sowie einen Teil eines Buches und wurde bereits entsprechend den Daten des Lagesensors auf den Boden projiziert. **(b)** Karte mit gespeicherten Merkmalen zum Zeitpunkt des Bildes in (a).

6.2 Der Algorithmus - Idee

- Finde eine Menge von möglichen Übereinstimmungen von Merkmalen aus dem aktuellen Kamerabild (siehe Abb. 6.1 (a)) und gespeicherten Merkmalen aus der Karte (siehe Abb. 6.1 (b)). Falsche Übereinstimmungen von ähnlich aussehenden Merkmalen können in dieser Menge häufig sein.
- Berechne eine Fehlerabschätzung für jede dieser möglichen Übereinstimmungen, basierend auf der Deskriptorendistanz (siehe Kapitel 4.11), sowie möglichem Wissen über die Orientierung.
- Ordne die möglichen Übereinstimmungen ansteigend anhand dieses Fehlers
- Wähle zwei dieser möglichen Übereinstimmungen (d. h. zwei Merkmale aus dem Bild, mit jeweils möglichen Partnern aus der Karte). Beginne mit Paaren mit geringem Fehler.
  - Berechne die externen Kameraparameter anhand dieses Paares von Übereinstimmungen (siehe Kapitel A im Anhang), unter der Annahme, dass sie korrekt sind.
  - Betrachte die so erhaltene Kameraposition als korrekt und projiziere dementsprechend alle Merkmale aus der Karte in das Kamerabild. Berechne für alle möglichen Übereinstimmungen anhand dieser Reprojektion eine Güte.

  - Summiere die Gütewerte auf (für jedes Bildmerkmal die Güte der jeweils besten Übereinstimmung) und betrachte das Ergebnis als Güte der betrachteten Kameraposition.

  Wiederhole dies, bis kein Paar mehr übrig ist, eine Kameraposition eine ausreichend hohe Güte aufweist oder eine andere Abbruchbedingung gültig ist.

- Nutze das Ergebnis des besten Paares als Kameraposition

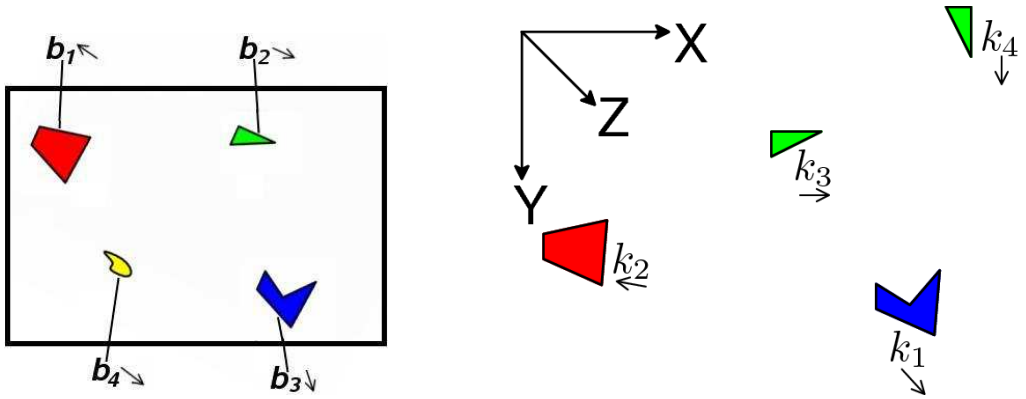
6.3 Der Algorithmus im Detail

Es sei $C \subseteq \mathbb{R}^6$ die Menge aller möglichen Kamerapositionen, bestehend aus $(x, y, z, \alpha, \beta, \gamma)$ (siehe Kapitel 3.4). Gesucht ist ein Element dieser Menge, wobei Pitch und Roll von dem Lagesensor bekannt sind und für den Yaw angenommen werden kann, dass er ähnlich zu dem am Anfang dieses Kapitels genannten $\text{Yaw}'_{\text{Sensor}}$ ist.

Die Menge aller möglichen SURF-Merkmale (siehe Kapitel 4.11) sei S. Für jedes dieser Merkmale ist seine 2D-Position und Orientierung im Kamerabild, sowie sein Deskriptorvektor bekannt.
S' sei die Menge aller ins dreidimensionale transformierten SURF-Merkmale. D. h. SURF-Merkmale, für die neben ihren 2D-Daten im Bild bereits 3D-Position und Orientierung in der Karte bekannt sind (siehe Kapitel 5).

$B = \{b_1, ..., b_n\} \subseteq S$ sei die Menge der SURF-Merkmale, die aus dem aktuellen Bild extrahiert wurden (siehe Abb. 6.1 (a)).
$K = \{k_1, ..., k_m\} \subseteq S'$ sei eine Teilmenge der SURF-Merkmale aus der Karte (siehe Abb. 6.1 (b)). Nach welchen Kriterien diese Teilmenge ausgewählt werden kann, wurde bereits im Kapitel 5.3 behandelt.

Betrachten wir hierzu folgendes Beispiel für eine Merkmalskarte, die bisher nur vier Merkmale enthält, sowie ein fiktives Kamerabild, das drei dieser Merkmale und ein bisher unbekanntes Merkmal im Sichtfeld hat. Die Pfeile beschreiben die Orientierungen der Merkmale:



Kamerabild **Karte**

Es sei $d: S \times S \rightarrow \mathbb{R}^+$: Die Funktion, die für zwei SURF-Merkmale deren Deskriptorendistanz zurückliefert (siehe Kapitel 4.11). Und es sei $maxD$ die Obergrenze für die Deskriptorendistanz, für die noch möglich ist, dass es sich um ein übereinstimmendes Merkmal handelt. In der Implementierung zu dieser Arbeit wurde dieser Wert auf 0.3 für den 64-elementigen Deskriptorvektor festgelegt.

Für das obige Beispiel könnten die Deskriptorendistanzen wie folgt sein:

d	k_1	k_2	k_3	k_4
b_1	2.32	0.09	3.14	3.21
b_2	3.04	1.87	0.11	0.10
b_3	0.21	2.18	3.19	3.21
b_4	4.03	3.84	2.86	2.76

Für $maxD = 0.3$ könnten demnach (b_1, k_2), (b_2, k_3) und (b_3, k_1) gleiche Merkmale sein, was korrekt ist. Allerdings wäre auch (b_2, k_4) auf Grund des ähnlichen Aussehens möglich, was eine fehlerhafte Übereinstimmung (*ein Mismatch*) wäre.

Sei $M \subseteq B \times K$ die Menge aller möglichen Übereinstimmungen aus Bildmerkmalen und Kartenmerkmalen.
D. h.: $M = \{(b,k) \mid d(b,k) \leq maxD \quad \forall b \in B, \forall k \in K\}$

Also $M = \left\{ \begin{array}{l} (b_1, k_2), \\ (b_2, k_3), (b_2, k_4), \\ (b_3, k_1) \end{array} \right\}$ für unser Beispiel.

Weiterhin seien die folgenden Funktionen bezüglich der Merkmalsorientierungen definiert:

- $o: S \rightarrow \mathbb{R}$: Die Orientierung eines Merkmals im Kamerabild (siehe Kapitel 4.11)

- $o_K : K \to \mathbb{R}$: Die Orientierung eines Karten-Merkmals (siehe Kapitel 5)
- $o_B : B \to \mathbb{R}$: Die Karten-Orientierung eines SURF-Merkmals aus dem aktuellen Bild, berechnet (siehe Kapitel 5) mittels $\text{Yaw}'_{\text{Sensor}}, \text{Pitch}_{\text{Sensor}}$ und $\text{Roll}_{\text{Sensor}}$ (siehe Anfang dieses Kapitels).

In unserem Beispiel sind die Orientierungen:

	k_1	k_2	k_3	k_4
o_K	49°	192°	0°	90°

	b_1	b_2	b_3	b_4
o	220°	22°	73°	65°

Die Daten des Lagesensors seien $\text{Yaw}'_{\text{Sensor}} = 23°,\ \text{Pitch}_{\text{Sensor}} = \text{Roll}_{\text{Sensor}} = 0°$. D. h. die Kamera blickt gerade nach unten. In solchen Fällen vereinfacht sich die Berechnung von o_B auf $o_B(b_i) = o(b_i) - \text{Yaw}_{\text{Sensor}}$. Somit ergeben sich die folgenden geschätzten Kartenorientierungen für die Bildmerkmale:

	b_1	b_2	b_3	b_4
o_B	197°	−1°	50°	42°

Es wird nun eine Heuristik benötigt, die bewertet, wie gut oder schlecht ein Element aus M nach aktuellem Wissen ist. Hierzu sei die folgende Funktion definiert, die den Fehler zwischen Bild- und Kartenmerkmal bezüglich Deskriptoren und Orientierungen beschreibt:

$$e_{do_B} : M \to \mathbb{R}^+ \text{ mit } e_{do_B}((b,k)) = g_o \cdot |o_B(b) - o_K(k)| + d(b,k) \quad \forall (b,k) \in M.$$

Der Gewichtungsfaktor $g_o \in \mathbb{R}^+$ bestimmt, wie stark der Fehler in der Orientierung der Merkmale im Vergleich zu der Deskriptorendistanz bewertet wird. In der Implementierung zu dieser Arbeit wurde dieser so gewählt, dass eine Abweichung der Orientierung um 15° so stark bewertet wird wie $maxD$. D. h. $g_o \cdot 15° = maxD \Leftrightarrow g_o = \frac{0.3}{15°} = 0.02\frac{1}{°}$

Für unser Beispiel ergibt sich:

	(b_1,k_2)	(b_2,k_3)	(b_2,k_4)	(b_3,k_1)
e_{do_B}	0.19	0.13	1.02	0.23

Man sieht bereits, dass Fehler des Mismatches (b_2, k_4) auf Grund der falschen Orientierung recht hoch ist.

Es seien nun $m_1, m_2, ..., m_{|M|}$ die anhand des Fehlers aufsteigend geordneten Elemente aus M, d. h.:
$M = \{m_1, m_2, ..., m_{|M|}\}$ mit $e_{do_B}(m_i) \leq e_{do_B}(m_j)\ \forall i < j$ und $i, j \in \{1, ..., |M|\}$

In unserem Beispiel ergibt dies:
$m_1 = (b_2, k_3)$, $m_2 = (b_1, k_2)$, $m_3 = (b_3, k_1)$, $m_4 = (b_2, k_4)$

Zur Berechnung der Kameraposition sind mindestens zwei übereinstimmende Merkmale aus Bild und Karte nötig (siehe Kapitel A im Anhang). Die jeweiligen Bildmerkmale sollten einen gewissen Mindestabstand im Kamerabild haben, um eine korrekte Berechnung zu gewährleisten. Daher werden nun Paare aus M wie folgt gewählt: $P \subseteq M \times M$ mit

$$P = \{((b,k),(b',k')) \mid b \neq b' \wedge k \neq k' \wedge\ euk(b,b') \geq minEuk \\ \forall (b,k),(b',k') \in M\}$$

Wobei $euk:\ S \times S \rightarrow \mathbb{R}^+$ die euklidische Distanz der 2D-Positionen zweier Merkmale im Kamerabild ist und $minEuk$ der geforderte Mindestabstand der Merkmale. In der Implementierung zu dieser Arbeit betrug dieser Mindestabstand 30 Pixel.

Es sollten nun möglichst zuerst Paare mit einem geringen Fehlerwert gewählt werden. Elemente mit einem hohen Wert sollten erst ganz zum Schluss integriert werden. Dies kann z.B. über folgende Schleifenstruktur realisiert werden:

Algorithm 1 Schleifenstruktur zur Auswahl der Paare

```
for (j = 2; j ≤ |M|; j = j + 1) do
  for (i = 1; i < j; i = i + 1) do
    if ((m_i, m_j) ∈ P) then
      p = (m_i, m_j)
      ⋮
    end if
  end for
end for
```

Unsere Fehlerheuristik bewirkt also, dass wir mit denjenigen Paaren beginnen, die nach aktuellem Wissen am Besten sind.
Die so erlangte Reihenfolge sei p_1, ..., $p_{|P|}$.

Für unser Beispiel ergibt dies:
$p_1 = (m_1, m_2), p_2 = (m_1, m_3), p_3 = (m_2, m_3)$,
$p_4 = (m_2, m_4), p_5 = (m_3, m_4)$
Wie man sieht, taucht das Mismatch m_4 erst zuletzt auf.

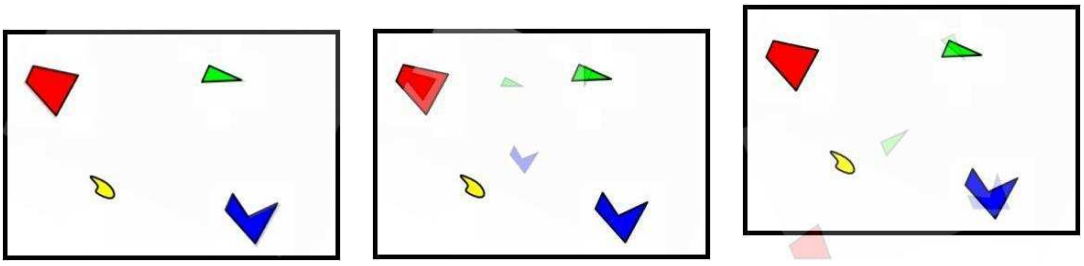
Nun werden die jeweils resultierenden Kameraparameter $c_1, ..., c_{|P|} \in C$ anhand dieser Paare berechnet (siehe Kapitel A im Anhang).

Für unser Beispiel wären c_1, c_2 und c_3 die (im Rahmen der Genauigkeit der Merkmale) korrekte Kameraposition. c_4 und c_5 hingegen basieren auf dem Mismatch m_4 und sind demnach fehlerhaft.

Gemäß dem zuvor erläuterten Prinzip von RANSAC, bzw. PROSAC, müssen diese Kamerapositionen nun bezüglich ihrer Güte bewertet werden. Hierzu soll nun unter anderem bestimmt werden, wie gut diese möglichen Kamerapositionen jeweils die Positionen der Merkmale in dem Kamerabild erklären. Hierzu gibt es grundlegend zwei Möglichkeiten: Es können entweder die Bildmerkmale in die Karte projiziert werden und dann die Distanzen zu ihren Gegenstücken berechnet werden, oder es können umgekehrt die Kartenmerkmale in das Kamerabild projiziert werden, um den Positionsfehler hierin zu berechnen.
Letzteres ist hierbei sinnvoller. Die Genauigkeit des verwendeten Sensors (der Kamera) ist abhängig von der Auflösung in Pixeln. Demnach sollte der Fehler in der Rückprojektion auch relativ hierzu gemessen werden. Ein Fehler in der Merkmalsposition von einigen Zentimetern in der Welt sagt für sich genommen nichts aus. Aus einem Meter Entfernung gemessen wäre dies ein relativ hoher Fehler, während es aus einigen hundert Metern Entfernung eine nahezu perfekte Übereinstimmung wäre. Hingegen ist die Abweichung im Kamerabild (in Pixeln) unabhängig davon.

Es sei also $proj : C \times K \rightarrow S$ die Funktion, die ein Kartenmerkmal anhand der gegebenen Kameraposition in das Bild projiziert (siehe Kapitel 5).

In unserem Beispiel sehen die Rückprojektionen der Kartenmerkmale in das Kamerabild wie folgt aus:



Projektion für c_1, c_2, c_3 **Projektion für c_4** **Projektion für c_5**

Das erste Bild stellt eine nahezu perfekte Überlagerung von Bild- und Kartenmerkmalen dar. In den letzten beiden Bildern ist anhand Position, Orientierung und Größe der Merkmale deutlich sichtbar, dass c_4 und c_5 fehlerhaft sind. Der Größenunterschied resultiert aus einer angenommenen Kameraposition, die im Vergleich zur tatsächlichen Position deutlich zu niedrig ist. Dieser Größenfehler wird jedoch im Folgenden nicht verwendet. SURF-Merkmale liefern zwar einen Größenfaktor innerhalb des Kamerabildes (siehe Kapitel 4.11), in Versuchen hat sich dieser Wert jedoch als zu unzuverlässig erwiesen, weshalb auf dessen Nutzung verzichtet wird.

Es wird nur mit den Positionen (der Zentren) der Merkmale, sowie deren Orientierungen gearbeitet.

Dementsprechend definieren wir die folgende Fehlerfunktion für jedes $m \in M$ bzgl. der Kameraposition $c \in C$:

$e: C \times M \to \mathbb{R}^+$ mit
$e(c,(b,k)) = g_e \cdot euk(b, proj(k)) + g_o \cdot |o(b) - o(proj(c,k))| + d(b,k)$
$\forall (b,k) \in M,\ c \in C.$

$g_o \in \mathbb{R}^+$ ist der bereits bekannte Gewichtungsfaktor für Differenzen in Orientierungen. $g_e \in \mathbb{R}^+$ ist das Äquivalent hierzu für Differenzen in der Position der Merkmale. In der Implementierung zu dieser Arbeit ist dieser so gewählt, dass eine Differenz von 2.5 Pixeln gleich stark gewichtet ist wie $maxD$. D. h. $g_e \cdot 2.5 = maxD \Leftrightarrow g_e = \frac{0.3}{2.5} = 0.12$
Es sei $maxE$ der maximale Fehler, für den eine Übereinstimmung noch als möglich angesehen werden kann. In der Implementierung zu dieser Arbeit ist dieser als $3 \cdot maxD$ definiert.

Die Güte einer Übereinstimmung sei definiert als $guete_m : C \times M \to [0,1]$ mit
$$guete_m(c,m) = \begin{cases} 1 - \frac{e(c,m)}{maxE} & \text{falls } e(c,m) \leq maxE \\ 0 & \text{sonst} \end{cases} \quad \forall c \in C, m \in M$$

Und $best: C \times B \to [0,1]$ mit
$$best(c,b) = \begin{cases} \max\limits_{\forall (b,k) \in M, k \in K} (guete_m(c,(b,k))) & \text{falls } \exists (b,k) \in M \\ 0 & \text{sonst} \end{cases}$$
sei die Güte der besten Übereinstimmung für ein Bildmerkmal.

Die Güte einer Kameraposition sei hieraus folgend definiert als
$guete: C \to \mathbb{R}^+$ mit $guete(c) = \sum\limits_{b \in B} best(c,b) \qquad \forall c \in C$

Die Kameraposition mit der höchsten Güte kann nun als aktuelle Positionsschätzung genutzt werden, sofern der Wert ein gewisses Minimum übersteigt. Für diese Arbeit wurde das Minimum als 3.0 gewählt. Dies bedeutet, dass als Minimum drei absolut perfekte Übereinstimmungen nötig sind. In der Praxis bedeutet dies jedoch eher ein Minimum von fünf bis sieben guten Übereinstimmungen.

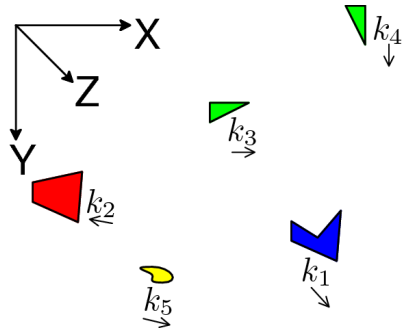
Wird dieses Minimum nicht überschritten, so kann z.B. das Bild zu stark gestört sein, oder K enthält nicht genug entsprechende Bildmerkmale.
Wie K gewählt wird und wie Fälle behandelt werden, in denen keine passende Po-

sition gefunden werden kann, wurde im Kapitel 5.3 behandelt.

Normalerweise ist es nicht notwendig, wirklich alle Paare zu testen. Ansonsten wäre die Heuristik, die wir zur Bestimmung der Reihenfolge der Paarwahl nutzen ohne Sinn. Es kann abgebrochen werden, wenn ein Paar gefunden wurde, dessen Ergebnis gut genug ist, nach einer gewissen Anzahl von getesteten Paaren oder nach einem Zeitlimit. In der Implementierung zu dieser Arbeit werden mindestens 50 Paare geprüft (sofern genug Merkmale im Bild gefunden wurden) und maximal 500. Im Bereich zwischen den 50 und 500 Paaren wird abgebrochen, wenn die Güte einer Kameraposition 10.0 überschreitet.

In unserem Beispiel hätten nur c_1, c_2 und c_3 eine Güte > 0 haben können, da die Abweichungen der Orientierungen und Positionen der ins Kamerabild projizierten Merkmale für c_4 und c_5 zu groß sind. Außerdem hätte nach der Berechnung der Güte von c_1 abgebrochen werden können, da für die errechnete Kameraposition bereits alle Bildmerkmale (mit mindestens einem zugehörigen Kandidaten in der Karte) einen Partner mit guter Übereinstimmung hatten. Das Minimum der Güte von 3.0 zu erreichen ist für ein Beispiel dieser Größe unrealistisch.

Nachdem nun die gefundene Kameraposition als korrekt angenommen wird, kann die Karte mit den neu hinzugekommenen Merkmalen aktualisiert werden (siehe Kapitel 5).
Für unser Beispiel würde dies eine Integration von b_4 bedeuten, da für dieses Bildmerkmal keine Übereinstimmung gefunden wurde. Dies ergibt folgende neue Karte:



Karte nach Update

6.4 Komplexität

Die Berechnung der möglichen Partner für jedes Bildmerkmal hat die Komplexität $O(|B|\cdot|K|\cdot dimD)$, wobei $dimD$ die Dimension der Deskriptorvektoren ist (hier üblicherweise 64). Da dieser Schritt demnach sehr aufwendig ist, ist die Auswahl, welche Elemente in K aufgenommen werden, sehr kritisch (siehe Kapitel 5.3). Außerdem

ist eine sorgfältige Low-Level-Optimierung der Berechnung der Deskriptorendistanz innerhalb des Programmcodes sinnvoll (siehe Kapitel 4.11).

Die Berechnung der Kamerapositionen ist in ihrem Aufwand linear in der Anzahl der geprüften Paare von Übereinstimmungen und hat somit eine obere Grenze von $O(|M|^2)$. Dank der beschriebenen Heuristik genügt es meist jedoch, nur einige wenige Paare zu prüfen. Tests haben gezeigt, dass nach 500 geprüften Paaren ohne gute Ergebnisse auch für den Rest angenommen werden kann, dass keine korrekte Position errechnet werden kann.

Der Aufwand zur Berechnung der Güte der Kamerapositionen ist in $O(|P| \cdot |M|)$, da für jede Position die Güte aller möglichen Übereinstimmungen berechnet werden muss. Die Größe von M ist abhängig von der Anzahl der gefundenen Bildmerkmale, sowie der durchschnittlichen Anzahl von möglichen übereinstimmenden Kartenmerkmalen für jedes von diesen. In der Implementierung zu dieser Arbeit wurde die Merkmalsextraktion so eingestellt, dass die Anzahl der Bildmerkmale zwischen 50 und 100 liegt. Die Anzahl der pro Bildmerkmal gefundenen möglichen Übereinstimmungen hängt von der Größe von K, von der Wahl von $maxD, g_o$ und von dem optischen Aufbau der Umgebung ab. Übliche Werte in unseren Versuchen sind z.B. bei Aufnahmen eines Holzbodens 0-5 mögliche Übereinstimmungen pro Bildmerkmal, wobei im Allgemeinen nur für vereinzelte Merkmale mehr als ein möglicher Partner gefunden wird. Diese Anzahl kann jedoch für ungünstige (z.B. stark symmetrische) Bilder stark ansteigen. So häufen sich z.B. bei Aufnahmen eines Blattes Papier auf dunklem Untergrund an den Rändern viele ähnliche Merkmale. Oder aber auf einem Boden mit einheitlichem Muster wie z.B. einzelnen quadratischen Kacheln, die für sich genommen alle gleich aussehen. Es ist sinnvoll, Merkmale mit zu vielen potenziellen Partnern in der Berechnung nicht zu berücksichtigen, da diese den Rechenaufwand unverhältnismäßig erhöhen.

Zu der Berechnungsdauer pro Kamerabild in praktischer Anwendung siehe Kapitel 8.4.

7 Optimierung der Trajektorie

In den letzten Kapiteln haben wir den Grundstein für eine Lösung des SLAM-Problems gelegt. Es wurde die Merkmalsextraktion erläutert und erklärt, wie hiermit eine Karte aufgebaut wird. Weiterhin wurde dargestellt, wie aus bekannten Übereinstimmungen von Merkmalen aus dem aktuellen Bild und der Karte die Kameraposition extrahiert werden kann. Hierbei wurde jedoch immer die Annahme gemacht, dass die aktuelle Karte konsistent mit der tatsächlichen Situation ist. Dies ist natürlich nicht realistisch. Die Positionsschätzung unterliegt verschiedenen möglichen Fehlern, unter anderem der Ungenauigkeit der verwendeten Sensoren. Hierdurch kann bei längeren Bewegungen ein Drift weg von der tatsächlichen Position entstehen. Es wurde bereits erläutert, wie versucht wird, Schleifen zu schließen, wenn ein bereits bekanntes Gebiet wieder besucht wird (siehe Kapitel 5.3). Bisher wird beim Finden eines solchen *loop-closings* allerdings nur die aktuelle Position korrigiert. Der bisherige Pfad und die hieraus resultierende Karte bleiben unverändert.
Im Folgenden soll nun erläutert werden, wie aus den gesammelten Daten ein Pfad rekonstruiert wird, der möglichst optimal zu diesen Daten passt. Es wird also ein Ansatz verfolgt, der versucht, die bezüglich den gemachten Beobachtungen wahrscheinlichste Karte zu finden (*maximum likelihood approach*).

Hierzu wird der Explorationsprozess als Graph repräsentiert. Die Knoten dieses Graphen sind die Roboterposen, sowie die Positionen der beobachteten Landmarken. Kanten stellen relative Positionsunterschiede dar, die mittels der verwendeten Sensoren gemessen wurden. So resultieren Kanten zwischen Roboterposen üblicherweise aus der Odometrie und Kanten zwischen Posen und Landmarken aus den Sensoren mittels derer die Landmarken erkannt werden (siehe Abb. 7.1(a)). Um die Komplexität des Graphen zu verringern können die Landmarken entfernt werden, indem die mit ihrer Hilfe gemessenen Informationen auf Kanten zwischen Roboterposen übertragen werden. Hierzu werden Kanten zwischen all denjenigen Posen eingefügt, die ein und die selbe Landmarke beobachtet haben und daher Wissen über ihre relativen Positionen zueinander vorhanden ist (siehe Abb. 7.1(b)). Mehrere Kanten zwischen zwei Knoten können zu einer zusammengefasst werden. Die Positionen der Knoten sollen nun so optimiert werden, dass die Sensorwahrnehmungen möglichst wahrscheinlich werden.

Lu und Milios nutzten 1997 [37] zuerst einen solchen Ansatz zur Lösung des SLAM-

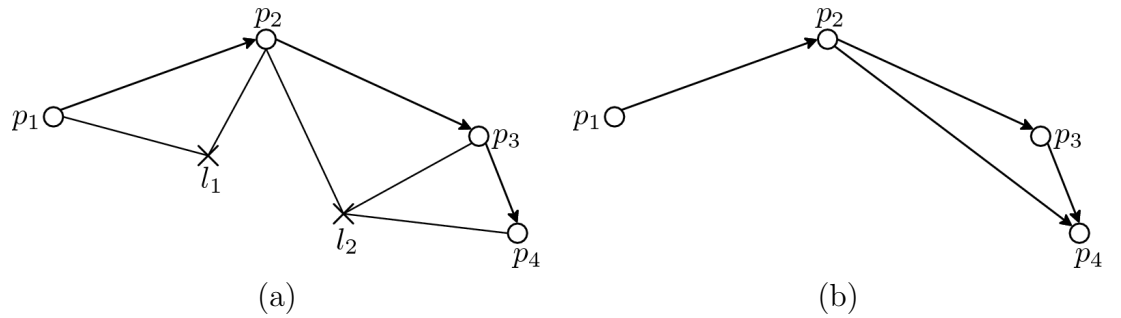


Abbildung 7.1: **(a)** Graph der einen Pfad von p_1 nach p_4 beschreibt, wobei l_1, l_2 jeweils Landmarken sind, die aus den verbundenen Posen beobachtet wurden. **(b)** Aus (a) resultierender Graph ohne explizite Angabe der Landmarken.

Problems. Bei diesem wird versucht, das gesamte Netzwerk auf einmal zu optimieren. In den folgenden Jahren wurden verschiedene Ansätze präsentiert, die beispielsweise schrittweise Positionen einzelner Knoten im Netzwerk anpassen [38, 39] oder jeweils den Fehler einzelner Kanten verringern. Zu Letzterem gehört auch ein Ansatz, der 2006 von Olson *et al.* [40] vorgestellt wurde. Dieser gehört aktuell zu den erfolgreichsten Methoden dieser Art. In dieser Arbeit wird eine Weiterentwicklung dieses Ansatzes von Grisetti *et al.* [41, 42] verwendet. Im Folgenden wird nun die Methode von Olson, sowie die Verbesserungen von Seiten Grisetti *et al.* erläutert.

7.1 Definitionen

Sei

- $\mathbf{p}$ der Vektor, der die globalen Positionen $(\mathbf{p_1} \dots \mathbf{p_n})^T$ der Knoten des aktuellen Graphen beinhaltet. Also die vermuteten Posen innerhalb des Pfades der Länge n.
  Wir werden vorerst von einem zweidimensionalen Szenario ausgehen. Die Position $\mathbf{p_i}$ des Roboters wird also durch drei Parameter bestimmt: $(x_{p_i}\ y_{p_i}\ \theta_{p_i})^T$ (seine Position und Orientierung in der XY-Ebene).

- $\delta_{\mathbf{ji}}$ der Vektor, der die Beobachtung des Knoten j aus Sicht des Knoten i repräsentiert. In unserem Fall ist dies die relative Position des Knoten j, gegeben das Referenzsystem von Knoten i. Dies entspricht jeweils einer Kante innerhalb des Graphen.

- Ω_{ji} die zu $\delta_{\mathbf{ji}}$ zugehörige Informationsmatrix. Eine Angabe darüber, wie sicher $\delta_{\mathbf{ji}}$ ist.

- $R_i = \begin{pmatrix} \cos\theta_{p_i} & -\sin\theta_{p_i} & 0 \\ \sin\theta_{p_i} & \cos\theta_{p_i} & 0 \\ 0 & 0 & 1 \end{pmatrix}$ die Matrix, die eine Position im Referenzsystem

des Knotens i in einen relativen Positionsunterschied zwischen dieser Position und der Position von i im globalen Referenzsystem überführt.

- f_{ji} eine Funktion, die eine rauschfreie Observation liefert, wie sie von Knoten i über Knoten j hätte gemacht werden können. Dies bezüglich der aktuellen Posen innerhalb des Graphen, bzw. bezüglich der aktuellen Karte.

7.2 Die Parametrisierung

Es stellt sich die Frage, in welcher Form die Positionen des Graphen nun verarbeitet werden sollen. Die einfachste Möglichkeit ist es natürlich, direkt die soeben genannten globalen Positionen $\mathbf{p_1}, ..., \mathbf{p_n}$ zu verwenden. Andere Parametrisierungen können jedoch sinnvoll sein. Laufzeit und Konvergenzgeschwindigkeit eines Optimierungsalgorithmus hängen stark von der gewählten Parametrisierung ab.
Eine Parametrisierung $\mathbf{x} = (\mathbf{x_1} \ ... \ \mathbf{x_n})^T$ ist gültig, falls eine bijektive Abbildung g existiert, so dass gilt: $\mathbf{x} = g(\mathbf{p})$, bzw. $\mathbf{p} = g^{-1}(\mathbf{x})$.

Basierend auf der jeweilig verwendeten Parametrisierung lässt sich nun die oben genannte Funktion $f_{ji}(\mathbf{x})$ definieren, die die gegebenen Positionen in eine Observation überführt. Für $\mathbf{x} = \mathbf{p}$, also unter direkter Verwendung der globalen Positionen würde beispielsweise gelten:

$$f_{ji}(\mathbf{x}) \quad = \quad R_i^T(\mathbf{p_j} - \mathbf{p_i}) \tag{7.1}$$

Im Folgenden werden nun zwei weitere mögliche Parametrisierungen vorgestellt.

7.2.1 Inkrementelle Posen

Eine Möglichkeit zur Repräsentation einer Pose ist es, jeweils die Änderung bezüglich der letzten Pose anzugeben:

$$\begin{aligned} \mathbf{x_0} &= \mathbf{p_0} \\ \mathbf{x_i} &= \mathbf{p_i} - \mathbf{p_{i-1}} \qquad \forall i \in \{1, ..., n\} \\ \Rightarrow \quad \mathbf{p_i} &= \sum_{k=0}^{i} \mathbf{x_i} \qquad \forall i \in \{0, ..., n\} \end{aligned}$$

Für f_{ji} ergibt sich:

$$f_{ji}(\mathbf{x}) \overset{7.1}{=} R_i^T(\mathbf{p_j} - \mathbf{p_i}) = R_i^T \cdot \left(\sum_{k=0}^{j} \mathbf{x_k} - \sum_{k=0}^{0} \mathbf{x_k} \right) = R_i^T \cdot \left(\sum_{k=i+1}^{j} \mathbf{x_k} \right) \tag{7.2}$$

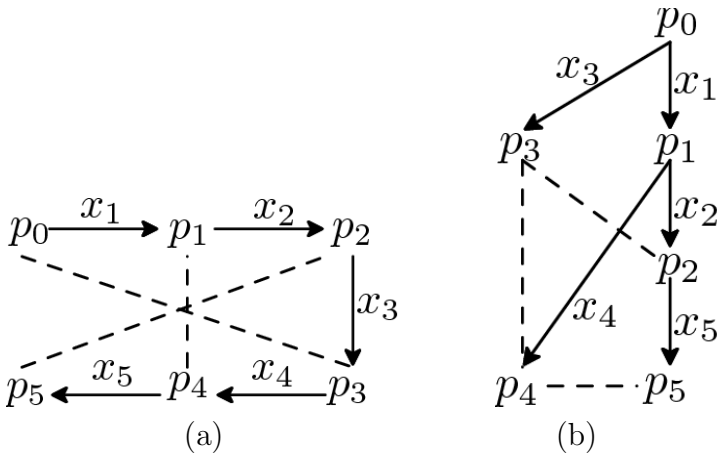


Abbildung 7.2: (a) Beispielgraph für die inkrementellen Posen. Pfeile stellen die Schritte der inkrementellen Parametrisierung dar. Gestrichelte Linien markieren weitere Kanten im Graph. **(b)** Zu (a) zugehörige Baumstruktur. Die gestrichelten Linien repräsentieren Kanten im Graph, die nicht zu der Baumstruktur zugehörig sind.

Abbildung 7.2(a) gibt ein visualisierendes Beispiel hierfür.
Bezüglich einer möglichen Optimierung des Graphen ist es einsichtig, dass sich eine Anpassung der Position des Knotens i automatisch auf alle seine Nachfolger auswirkt.
Es ist anzumerken, dass diese Methode voraussetzt, dass Kanten jeweils zwischen aufeinanderfolgenden Pfadpositionen existieren. In vielen Anwendungen kann diese Eigenschaft durch die Odometrie des verwendeten Vehikels gewährleistet sein. Bei rein Landmarken basierten Prozessen, wie dem visuellen SLAM in dieser Arbeit, sind zwar Kanten zwischen aufeinanderfolgenden Positionen sehr wahrscheinlich, jedoch nicht zwingend vorhanden.

Dies ist die Parametrisierung, wie sie Olson *et al.*[40] für ihren Optimierungsalgorithmus verwenden.

7.2.2 Baum-basierte inkrementelle Posen

Grisetti *et al.* [41, 42] schlagen eine alternative Parametrisierung vor um ein besseres Konvergenzverhalten zu erreichen. Hierbei werden die zusätzlichen Kanten genutzt, um eine Baumstruktur aufzubauen. Es wird für jeden Knoten i der älteste verbundene Knoten, im Folgenden bezeichnet als parent(i), ausgewählt und zum direkten Vorgänger erklärt. In Abbildung 7.2(b) ist ein Beispiel für einen solchen Baum zu sehen.

Die Parametrisierung ist nun ähnlich der oben beschriebenen, jedoch nicht auf die

Reihenfolge im Pfad, sondern auf diesen Baum bezogen:

$$\begin{aligned} \mathbf{x_0} &= \mathbf{p_0} \\ \mathbf{x_i} &= \mathbf{p_i} - \mathbf{p}_{\text{parent}(\mathbf{i})} \qquad \forall i \in \{1, ..., n\} \\ \mathbf{p_i} &= x_i + x_{\text{parent}(i)} + x_{\text{parent}(\text{parent}(i))} + ... + x_0 \end{aligned}$$

Sei $\mathcal{P}(i,j)$ die Menge der Strecken auf dem Pfad von Knoten i zu Knoten j im Baum. In dem Beispiel aus Abbildung 7.2(b) wäre beispielsweise $\mathcal{P}(4,5) = \{\mathbf{x_4}, \mathbf{x_2}, \mathbf{x_5}\}$. Sei $c(i,j)$ der erste gemeinsame Vorfahre der Knoten i und j (In obigem Beispiel der Knoten 1). Die Menge $\mathcal{P}(i,j)$ sei nun aufgeteilt in die Strecken, die zwischen $c(i,j)$ und i verlaufen $\mathcal{P}^{[-]}(i,j)$ und die Strecken, die zwischen $c(i,j)$ und j verlaufen $\mathcal{P}^{[+]}(i,j)$. In obigem Beispiel bedeutet dies $\mathcal{P}^{[-]}(i,j) = \{\mathbf{x_4}\}$ und $\mathcal{P}^{[+]}(i,j) = \{\mathbf{x_2}, \mathbf{x_5}\}$

Es ergibt sich für f_{ji}:

$$f_{ji}(\mathbf{x}) = R_i^T \cdot \left(\sum_{\mathbf{x}^{[+]} \in \mathcal{P}^{[+]}(i,j)} \mathbf{x}^{[+]} - \sum_{\mathbf{x}^{[-]} \in \mathcal{P}^{[-]}(i,j)} \mathbf{x}^{[-]} \right) \tag{7.3}$$

Für diese Parametrisierung wirkt sich ein Update einer Knotenposition auf alle Nachfolger im Baum aus. Die Komplexität der Berechnung der globalen Position eines Knotens hängt nicht mehr von dem kompletten vorherigen Pfad ab, sondern von der Tiefe des Knotens im Baum. Diese Tiefe basiert auf der Größe des erforschten Gebietes. Ein langer Pfad, der nur ein kleines Gebiet abdeckt, wird immer wieder zu bekannten Punkten zurückkehren und somit Kanten zu frühen Knoten erlangen. Hierdurch ergeben sich kürzere Pfade, was sich positiv auf Geschwindigkeit und Konvergenz auswirkt.
Weiterhin fällt bei dieser Methode die Einschränkung weg, dass aufeinanderfolgende Posen im Pfad durch eine Kante verbunden sein müssen.

7.3 Optimierung bezüglich inkrementeller Posen

Im Folgenden werden wir nun zuerst auf einen Optimierungsalgorithmus basierend auf den inkrementellen Posen eingehen, wie er 2006 von Olson *et al.* [40] vorgestellt wurde.

Gegeben eine Kante δ_{ji} im Graphen, also eine Sensorwahrnehmung zwischen den Posen i und j, sei

$$e_{ji}(\mathbf{x}) = f_{ji}(\mathbf{x}) - \delta_{ji} \tag{7.4}$$

der Fehler dieser Kante innerhalb des aktuellen Graphen. $e_{ji}(\mathbf{x}) = \mathbf{0}$ bedeutet dabei, dass die aktuelle Konfiguration der Knoten i und j perfekt der Kante (=Observation der Sensoren) δ_{ji} entspricht.

Dementsprechend ist das Residuum der negative Fehler:

$$r_{ji}(\mathbf{x}) = \delta_{ji} - f_{ji}(\mathbf{x}) = -e_{ji}(\mathbf{x}). \tag{7.5}$$

Unter der Annahme, dass der Fehler der Sensorwahrnehmungen einer Gaussverteilung entspricht, lässt sich die negative logarithmische Wahrscheinlichkeit der Observation ausdrücken als [41, 42]:

$$\begin{aligned} F_{ji}(\mathbf{x}) &= \frac{1}{2} e_{ji}(\mathbf{x})^T \Omega_{ji} e_{ji}(\mathbf{x}) \\ &= \frac{1}{2} r_{ji}(\mathbf{x})^T \Omega_{ji} r_{ji}(\mathbf{x}) \end{aligned} \tag{7.6}$$

Dieser Wert bewegt sich zwischen Null und Unendlich, wobei ein niedriger Wert für eine hohe Wahrscheinlichkeit der Sensormessungen steht und ein hoher entsprechend für eine niedrige Wahrscheinlichkeit.
Unter der Annahme, dass die Observationen unabhängig voneinander sind, lässt sich die negative logarithmische Wahrscheinlichkeit des Gesamtsystems ausdrücken als

$$\begin{aligned} F(\mathbf{x}) &= \sum_{<i,j>\in\mathcal{C}} F_{ji}(\mathbf{x}) \\ &= \frac{1}{2} \sum_{<i,j>\in\mathcal{C}} r_{ji}(\mathbf{x})^T \Omega_{ji} r_{ji}(\mathbf{x}) \end{aligned} \tag{7.7}$$

Wobei $\mathcal{C}$ die Menge der Kanten im Graph ist, d.h. $< i, j > \in \mathcal{C}$ genau dann wenn δ_{ji} existiert.

Dieser Wert wird auch als die Energie des Systems bezeichnet. Hierzu kann man sich physikalisch ein Federsystem vorstellen, bei dem jede Feder eine der Kanten darstellt und versucht die Knoten in die zu der Sensorwahrnehmung passende Position zu ziehen. Je weniger Spannung auf den Federn ist (also je weniger Energie in dem System steckt), desto besser passt der Graph zu den Observationen. Ziel einer Optimierung des Graphen ist es demnach, die Energie des Systems zu minimieren und damit die Wahrscheinlichkeit der Sensorwahrnehmungen zu maximieren. Es wird also die Knotenkonfiguration $\mathbf{x}^*$ gesucht, für die gilt:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}}\, F(\mathbf{x}) \tag{7.8}$$

Diese Minimierung geschieht üblicherweise iterativ. Olsons Ansatz [40] wählt hierzu jeweils eine Observation δ_{ji} aus und verringert durch Positionsanpassungen der

Knoten im Netzwerk den Fehler dieser Kante.

Eine Möglichkeit hierzu ist es, jeweils dem abfallenden Gradienten zu folgen (*Gradient Descent*):

$$\begin{aligned} \mathbf{x}^{t+1} &= \mathbf{x}^t - \lambda \frac{\partial F_{ji}}{\partial \mathbf{x}} \\ &= \mathbf{x}^t + \lambda \cdot J_{ji}^T \Omega_{ji} \mathbf{r_{ji}} \end{aligned} \tag{7.9}$$

Hierbei ist J_{ji} die Jacobi-Matrix von f_{ji} bezüglich $\mathbf{x}$.
Wir bewegen uns also in der Konfiguration des Netzwerks in die Richtung, in der die Energie am stärksten abfällt. Das heißt, das Residuum $\mathbf{r_{ji}}$ der betrachteten Kante wird erhöht und damit der Fehler verringert.
Die Informationsmatrix Ω_{ji} skaliert hierbei die Komponenten des Residuums entsprechend der Sicherheit der Messung, auf der dieses basiert.
λ ist die Lernrate. Die Updates verschiedener Kanten können einander entgegen wirken. Die Lernrate nimmt mit der Anzahl der durchgeführten Iterationen ab, wodurch die Wirkung der Updates auch entsprechend sinkt, so dass sich ein Gleichgewicht einpendeln kann.

Die Rolle der Jacobi-Matrix ist es, das Residuum in den Raum der inkrementellen Posen zu transformieren, so dass der Fehler verringert wird.

$$J_{ji} = \frac{\partial f_{ji}}{\partial \mathbf{X}} = \begin{pmatrix} \frac{\partial f_{ji}}{\partial \mathbf{x_1}} & \frac{\partial f_{ji}}{\partial \mathbf{x_2}} & \dots & \frac{\partial f_{ji}}{\partial \mathbf{x_n}} \end{pmatrix} \tag{7.10}$$

Für inkrementelle Posen gilt nach Gleichung 7.2: $f_{ji}(\mathbf{x}) = R_i^T \cdot (\sum_{k=i+1}^{j} \mathbf{x_k})$
Zu beachten ist hierbei, dass R_i nur abhängig ist von $\theta_{p_i} = \sum_{k=0}^{i} \theta_{x_k}$, also eine Funktion von $\mathbf{x_{0:i}}$ und nicht von $\mathbf{x_{i+1:j}}$ ist.
Damit erhält man für die Jacobi-Matrix:

$$J_{ji}(\mathbf{x}) = \left(\underbrace{A_0 \ \cdots \ A_i}_{0,\dots,i} \ \underbrace{R_i^T \ \cdots \ R_i^T}_{i+1,\dots,j} \ \underbrace{0 \ \cdots 0}_{j+1,\dots,n} \right) \tag{7.11}$$

wobei

$$A_n = \frac{\partial (R_i^T \sum_{k=i+1}^{j} \mathbf{x_k})}{\partial \mathbf{x_n}} \tag{7.12}$$

$$= \begin{pmatrix} 0 & 0 & \\ 0 & 0 & \tilde{R}_{0:n-1}^T \tilde{R}_n'^T \tilde{R}_{n+1:i-1}^T \sum_{k=i+1}^{j} \tilde{\mathbf{x}}_\mathbf{k} \\ 0 & 0 & 0 \end{pmatrix}. \tag{7.13}$$

Hierbei sind $\tilde{\mathbf{x}}_\mathbf{k} = \begin{pmatrix} x_{x_k} \\ y_{x_k} \end{pmatrix}$ und $\tilde{R}_k = \begin{pmatrix} \cos\theta_{p_k} & -\sin\theta_{p_k} \\ \sin\theta_{p_k} & \cos\theta_{p_k} \end{pmatrix}$ die zweidimensionalen Gegenstücke zu $\mathbf{x_k}$ und R_k.

Es ist anzumerken, dass in den A_n die Winkelkoordinate der $\mathbf{x_{i+1:j}}$ nicht vorkommt. Ist $\sum_{k=i+1}^{j} \tilde{\mathbf{x}}_\mathbf{k}$, also die Translation zwischen den Positionen i und j, klein, so kann die Jacobi-Matrix wie folgt approximiert werden:

$$J_{ji}(\mathbf{x}) \approx R_i^T \left(\underbrace{0 \cdots 0}_{0,\ldots,i} \underbrace{I \cdots I}_{i+1,\ldots,j} \underbrace{0 \cdots 0}_{j+1,\ldots,n} \right) \tag{7.14}$$

Hierbei ist I eine 3x3 Identitätsmatrix.

Die Annahme, dass die Translation zwischen den Posen i und j klein ist, ist generell gewährleistet, wenn die verwendeten Sensoren keine hohe Reichweite haben. Denn, damit es die Kante zwischen i und j gibt, mussten von beiden Positionen die selben Landmarken sichtbar gewesen sein. Für das hier behandelte Szenario des visuellen SLAMs ist dies gewährleistet. Posen, in denen sich die aufgenommenen Bilder des Bodens überschneiden, müssen (im Verhältnis zur Höhe der Kamera) nahe beieinander liegen.

In der Praxis hat sich außerdem gezeigt, dass trotz dieser Approximation die Optimierung von Graphen auch problemlos funktioniert, wenn die Daten auf Sensoren mit hoher Reichweite beruhen.

Eine Variation der Update-Regel 7.9 ist für ein effizientes Konvergenzverhalten noch nötig [41, 42]. Es ist sinnvoll, dass die Iterationsschritte groß sind, wenn der Gradient an der aktuellen Position klein ist um die Konvergenzgeschwindigkeit zu erhöhen. Umgekehrt sollten die Iterationsschritte klein sein, wenn der Gradient groß ist, um das gesucht Minimum nicht zu verpassen. Olson [40] verwendet daher die folgende Update-Regel:

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \underbrace{\lambda \cdot (diag(J^T \Omega J))^{-1} * J_{ji}^T \Omega_{ji} \mathbf{r_{ji}}}_{\Delta \mathbf{x}} \tag{7.15}$$

wobei *diag* die Diagonalelemente als Vektor extrahiert, $*$ die elementweise Multiplikation ist und

$$J^T = \begin{pmatrix} J_{j_1,i_1}^T & J_{j_2,i_2}^T & \cdots & J_{j_M,i_M}^T \end{pmatrix} \qquad \text{mit } \mathcal{C} = \{< i_1, j_1 >, ..., < i_M, j_M >\}$$

$$\Omega = \begin{pmatrix} \Omega_{j_1,i_1} & 0 & 0 & 0 \\ 0 & \Omega_{j_2,i_2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \Omega_{j_M,i_M} \end{pmatrix}$$

$J^T \Omega J$ ist eine Approximation der Hesse-Matrix $H = \frac{\partial^2 F}{\partial^2 \mathbf{x}}$ des Systems. Und dementsprechend $H^{-1} \approx (J^T \Omega J)^{-1}$. Die Hesse-Matrix stellt den Krümmungsverlauf der Fehlerfunktion dar. Die Multiplikation mit obigem Ausdruck führt zu dem angesprochenen Ziel, dass kleine Schritte gemacht werden, wo die Krümmung groß ist und große Schritte, wo die Krümmung klein ist. Die tatsächliche Berechnung von H^{-1} ist aufwendig, weshalb diese Approximation genutzt wird.

Durch mehrfaches Optimieren aller Bedingungen (Kanten) konvergiert der Graph mit dieser Methode zu einem Graphen mit maximaler Wahrscheinlichkeit. Oder zumindest einem lokalen Maxima der Wahrscheinlichkeit.
Unter Betrachtung der Jacobi-Matrix sieht man, dass durch die inkrementelle Parametrisierung bei der Betrachtung von δ_{ji} die Werte von $\mathbf{x_{i+1:j}}$ aktualisiert werden.

7.4 Optimierung mit Baum-basierter Parametrisierung

Grisetti *et al.* [41, 42] griffen Olsons Algorithmus zur Graphenoptimierung auf, verwendeten jedoch die Baum-basierten inkrementellen Posen. Dies erhöht sowohl die Geschwindigkeit der einzelnen Iterationen, als auch die Konvergenzgeschwindigkeit enorm.

Grundlegend ändert sich die Berechnung nur gering. Nach Gleichung 7.3 gilt:

$$f_{ji}(\mathbf{x}) = R_i^T \left(\sum_{\mathbf{x}^{[+]} \in \mathcal{P}^{[+]}(i,j)} \mathbf{x}^{[+]} - \sum_{\mathbf{x}^{[-]} \in \mathcal{P}^{[-]}(i,j)} \mathbf{x}^{[-]} \right)$$

Hieraus folgt für die Jacobi-Matrix:

$$J_{ji}(\mathbf{x}) \approx R_i^T \left(0 \cdots 0 \underbrace{-I \cdots -I}_{k \in \mathcal{P}^{[-]}(i,j)} 0 \cdots 0 \underbrace{I \cdots I}_{k \in \mathcal{P}^{[+]}(i,j)} 0 \cdots 0 \right) \tag{7.16}$$

Es werden also nur Posen aktualisiert, die sich im Baum auf dem Weg von i nach j befinden.
Bei Pfaden, die nur aus einzelnen großen Schleifen bestehen macht dies keinen Unterschied. Jedoch bei Pfaden, die oft zu bekannten Orten zurückkehren ist ein Iterationsschritt mit dieser Parametrisierung bedeutend weniger aufwendig. Das Update ist auf ein Gebiet eingeschränkt, das besonders von dem Schleifenschluss betroffen ist. Die Anpassung der umliegenden Bereiche erfolgt automatisch während der Betrachtung der restlichen Kanten.
Insgesamt ist die Komplexität des Updates einer Kante nicht mehr abhängig von der Länge des Pfades, sondern von der Größe des erforschten Gebietes (also der maximalen Tiefe des Baumes).

7.5 Anwendung für visuellen SLAM

Nachteilig im Vergleich zu probabilistischen Ansätzen ist bei solchen Graphen-basierten SLAM-Systemen beispielsweise der Mangel an Informationen über die Güte der aktuellen Pfadschätzung. Diese Information kann normalerweise genutzt werden, um die Größe von Suchbereichen für beobachtete Landmarken zu bestimmen. Ohne diese Information muss dieser Bereich meist unnötig groß gewählt werden.
Außerdem ist die Grundidee dieser Algorithmen zur nachträglichen Pfadkorrektur nur für offline-Anwendungen gedacht, was ihre Verwendbarkeit in der mobilen Robotik einschränkt.

Es sprechen jedoch auch einige Punkte für die Graphen-basierte Realisierung eines visuellen SLAM-Systems:

- Im Vergleich zu Rao-Blackwellized Partikelfiltern oder erweiterten Kalman-Filtern kann eine bedeutend größere Anzahl von Landmarken verwaltet werden, da nur eine Karte vorhanden ist, bzw. keine Kovarianzmatrix gepflegt werden muss.

- Die Verwaltung der Karte und Trajektorie ist einfacher, da keine zusätzlichen probabilistischen Informationen gepflegt werden müssen.

- Das Prinzip passt allgemein gesehen sehr gut zum visuellen SLAM: Die Knoten des Graphen sind die Kamerapositionen und Kanten repräsentieren sich überlappende Bildbereiche.

Außerdem bieten sich weitere Verwendungsmöglichkeiten an, die zwar in dieser Arbeit nicht behandelt wurden, jedoch für zukünftige Weiterführungen denkbar sind:

- Es ist einfach, nachträglich Korrekturen an der Karte vorzunehmen, indem beispielsweise fehlerhaft erkannte loop-closings per Hand gelöscht werden und dann lediglich der Optimierungsschritt zu wiederholen ist. Oder aber umgekehrt nicht erkannte loop-closings per Hand eingefügt werden können, um die Qualität der Karte zu erhöhen.

- Eine Online-Variante erscheint unter Beachtung der sehr guten Performanz der von Grisetti *et al.* vorgestellten Version durchaus möglich. Es ist denkbar, einige Iterationsschritte zwischen den Kamerabildern durchzuführen. Außerdem müssten nur gelegentlich Optimierungen durchgeführt werden, wenn (größere) Schleifen geschlossen wurden.

Für diese Arbeit wurde daher entschieden, eine solche Methode einzusetzen. Es wurde die Implementierung von Grisetti *et al.* verwendet. Jedoch war hierzu noch

eine geringfügige Anpassung notwendig, da wir nicht nur zweidimensional arbeiten. Grisetti *et al.* entwickelten auch eine dreidimensionale Version dieses Algorithmus [43], der jeweils alle sechs Parameter $(x, y, z, \theta, \beta, \gamma)$ verarbeitet. Dies ist jedoch für die Anwendung in dieser Arbeit auf Grund der vom Lagesensor gegebenen Werte unnötig komplex. Da wir Neigungs- und Rollwinkel direkt vom Lagesensor nehmen, genügt es, den dreidimensionalen Vektor $(x_{p_i}\ y_{p_i}\ \theta_{p_i})^T$ zur Beschreibung der Posen in 2D durch einen vierdimensionalen Vektor $(x_{p_i}\ y_{p_i}\ z_{p_i}\ \theta_{p_i})^T$ zu ersetzen. Da die z-Koordinate unabhängig vom Winkel θ (der nun dem Drehwinkel/Yaw entspricht) ist, ändert sich an den Optimierungsschritten kaum etwas. R_i hat nun lediglich die Form $R_i = \begin{pmatrix} \cos\theta_{p_i} & -\sin\theta_{p_i} & 0 & 0 \\ \sin\theta_{p_i} & \cos\theta_{p_i} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$.
Ansonsten bleibt der oben beschriebene Weg identisch.

Die Kanten des Graphen bestehen in unserem System aus den Ergebnissen der visuellen Odometrie, sowie der lokalen und globalen Lokalisierung (siehe Kapitel 5.3). Um zu entscheiden, zwischen welchen Knoten diese Kanten verlaufen, wird das zur Berechnung genutzte Paar von Übereinstimmungen herangezogen (siehe Kapitel 6). Die Kanten verlaufen zwischen der aktuellen Pose und den Posen der Merkmale in der Karte, die tatsächlich für die Berechnung der Kameraposition genutzt wurden. Die zugehörigen Informationsmatrizen sind Diagonalmatrizen, deren Einträge jeweils der ermittelten Güte der Kante (siehe ebenfalls Kapitel 6) entsprechen. Die Ausgabe des Optimierungssystems ist ein korrigierter Pfad, dessen einzelne Posen dann genutzt werden, um die Karte zu aktualisieren. Dazu werden die Merkmale entsprechend Kapitel 5.1 neu in die Karte projiziert.
Beispielbilder für durchgeführte Graphenoptimierungen sind unter den Experimenten in Kapitel 8 zu finden.

8 Experimente

In diesem Kapitel werden die experimentellen Ergebnisse des implementierten Systems zum visuellen SLAM vorgestellt. Es soll gezeigt werden, dass das System unter verschiedenen und teilweise schwierigen Bedingungen gute Ergebnisse liefert.
Im ersten Versuch wurde die Anwendbarkeit für lange Trajektorien im Außenbereich getestet. Im zweiten Versuch soll die Verwendung innerhalb von Gebäuden, sowie die Genauigkeit der erlangten Karten überprüft werden. Der dritte Versuch wurde mit einem tatsächlichen Fluggerät, einem Miniaturzeppelin, unter eingeschränkter Sensorverfügbarkeit durchgeführt.

8.1 Anwendung im Außenbereich

Das folgende Experiment soll die Robustheit des vorgestellten Systems für lange Pfade mit schwer unterscheidbaren Texturen demonstrieren. Hierzu wurde unser Stereokamerasystem (siehe Kapitel C.1 im Anhang) zusammen mit dem Lagesensor (siehe Kapitel C.2 im Anhang) beweglich an der Spitze eines Stockes angebracht um ein frei schwebendes System zu simulieren (siehe Abb. 8.1). Die Kameras waren jeweils mit Weitwinkelobjektiven ausgerüstet.

Mit diesem System wurde ein Pfad um ein Gebäude auf dem Gelände der Albert-Ludwigs-Universität aufgenommen. Die von den Kameras aufgenommene Oberfläche bestand hauptsächlich aus Gras und Steinboden. Beispiele für Kamerabilder aus diesem Experiment sind in Abbildung 8.2 zu sehen. Hierbei ist besonderes Augenmerk darauf zu legen, dass teilweise nur schwer unterscheidbare Texturen sichtbar waren. Außerdem traten Effekte wie zu starke Beleuchtung und Unschärfe durch zu schnelle Bewegungen auf wodurch feine Unterschiede in den aufgenommenen Strukturen verschwimmen. Dies erschwert die Unterscheidung ohnehin sehr ähnlich aussehender Bereiche, wie beispielsweise den Platten des Steinbodens.

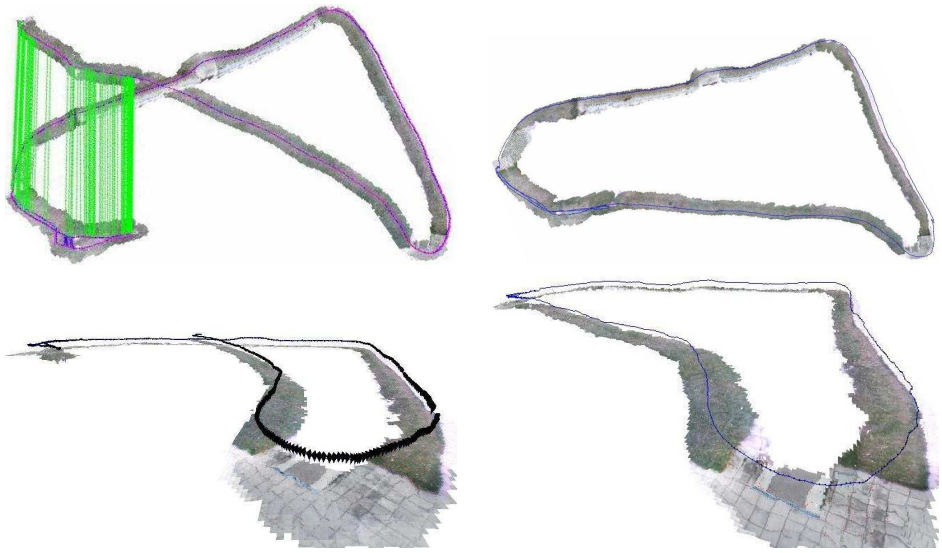
Die tatsächliche Trajektorie hatte eine Länge von ungefähr 188 m (geschätzt mittels Google Earth). Von dem Eingangsvideo wurden jeweils 5 Bilder pro Sekunde mit einer Auflösung von 320x240 Pixeln verwendet um die Trajektorie zu schätzen. Insgesamt wurden hierzu 1443 Stereo-Bilder verarbeitet. Abbildung 8.3(a) zeigt den aus der visuellen Odometrie resultierenden Pfad. Die hieraus extrahierte Länge des Weges beträgt 220.4 m.

(a) (b)

Abbildung 8.1: **(a)** Stereokamerasystem bestehend aus zwei Webcams in Kombination mit dem verwendeten Lagesensor. **(b)** Verwendete Methode zur Simulation eines freischwebenden Systems. An der Spitze des Stockes ist das Sensorsystem aus (a) angebracht.

(a) (b) (c)

Abbildung 8.2: Beispielbilder, die während diesem Experiment verarbeitet wurden. Allgemein zu beachten sind unter anderem die teilweise ungleichmäßige Beleuchtung und die Unschärfe, die durch schnelle Bewegungen der Kamera entstehen kann. **(a)** Übergang von Rasen zu Steinboden. **(b)** Stark symmetrischer Steinboden. **(c)** Rasen.



(a) (b)

Abbildung 8.3: **(a)** Aus der Odometrie resultierender Pfad. Ansicht von oben und geneigt. Die Linien links im oberen Bild verbinden Stellen, die das System als identisch erkannt hat (loop-closings). Es wurden selbst Stellen korrekt zugeordnet, an denen nur Gras zu sehen war, ohne weitere markante Objekte im Bild. **(b)** Anhand der in (a) sichtbaren loop-closings optimierter Pfad. Ansicht von oben und geneigt.

Es ist anzumerken, dass es in Bereichen, die optisch sehr ähnlich waren, teilweise zu falschen Erkennungen innerhalb der globalen Lokalisierung kam. Dies führt zu fehlerhaften Kanten innerhalb des resultierenden Graphen. Ein Beispiel für Stellen an denen dies vorkam ist in Abbildung 8.4 zu sehen. Solche Fehlerkennungen werden unter anderem durch die aus der Bewegung resultierende Unschärfe begünstigt. Durch diese verschwimmen die ohnehin schon schwer zu erkennenden Unterschiede zwischen den Bereichen noch zusätzlich. Obwohl solche Fälle vergleichsweise selten auftraten (ugf. an zehn Stellen des kompletten Pfades) führte dies zu Fehlern in der Optimierung, die das Ergebnis stark verschlechterten. Die fehlerhaften Zuordnungen traten allerdings nur bei Bildern auf, in denen überwiegend Pflastersteine zu sehen waren. Auf dem Rasen sind offensichtlich trotz der Bewegungsunschärfe genug eindeutige Merkmale vorhanden.
Um das Auftreten dieses Problems zu umgehen wurde in diesem Experiment die Funktion der globalen Lokalisierung erst aktiviert, als bereits bekannte Punkte in der Nähe waren.

Die anhand von korrekt wiedererkannten Punkten optimierte Karte und Trajektorie (siehe Kapitel 7) ist in Abbildung 8.3(b) zu sehen. In dieser Version ist die Schleife korrekt geschlossen, sodass der Pfad die Umrundung des Gebäudes beschreibt. Die Länge dieses Pfades beträgt 208.1 m, was im Vergleich mit der Länge der tat-

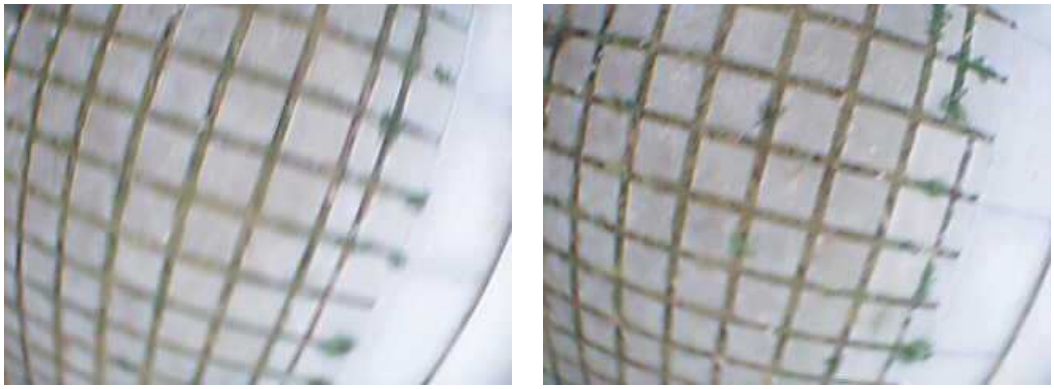


Abbildung 8.4: Beispiel für Bilder, die fälschlich als sich überschneidend angenommen werden können. Diese Aufnahmen liegen tatsächlich einige Meter auseinander.

sächlichen Trajektorie einen Fehler von etwa 10% bedeutet. Die Skalierung dieser Trajektorie stammt ausschließlich von den extrahierten Tiefen des Stereokamerasystems. Ein Teil des Fehlers kann demnach aus Ungenauigkeiten in der Kalibrierung dieses Systems resultieren. Eine weitere Fehlerquelle für das Stereokamerasystem ist außerdem, dass die verwendete Hardware keine Möglichkeit gibt, die Bilder der beiden Kameras zeitlich genau zu synchronisieren. Bei schnellen Bewegungen können so die Bilder der Kameras leicht gegeneinander verschoben sein, was die Erkennung der Distanz enorm stört.
Insgesamt ist ein Fehler von unter 10% unter Beachtung der verwendeten Sensoren jedoch durchaus als ein gutes Ergebnis zu betrachten. Insbesondere unter dem Gesichtspunkt, dass die aufgenommenen Bilder oftmals nur schwer unterscheidbare Texturen aufwiesen (Gras oder optisch zueinander sehr ähnliche Steinplatten). Obwohl es hierdurch wie oben erwähnt zu Fehlern in den loop-closing-Prozessen kam, arbeitete die Odometrie in diesen Bereichen absolut zufriedenstellend.

In Abb. 8.5 ist die extrahierte Trajektorie innerhalb eines Satellitenbildes des betreffenden Gebäudes zu sehen. Dies entspricht bis auf kleine Abweichungen dem tatsächlich genommenen Pfad.

8.2 Anwendung im Innenbereich

Das folgende Experiment sollte die Nutzbarkeit des SLAM-Systems im Innenbereich prüfen und gleichzeitig Daten über die Genauigkeit der Karten liefern. Zu diesem Zweck wurden Marken in bekannten Abständen, sowie Objekte mit bekannter Höhe in einem Korridor unseres Gebäudes ausgelegt (siehe Abb. 8.6 (a,b)). Der gewählte Pfad beschrieb jeweils eine Schleife, in der diese Objekte zu sehen waren. Neben den künstlich platzierten Objekten dienten die Texturen des Holzbodens als mögliche

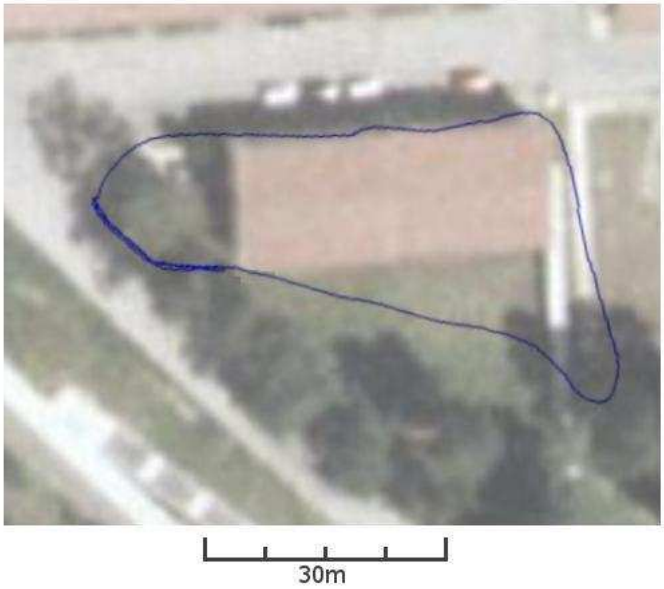
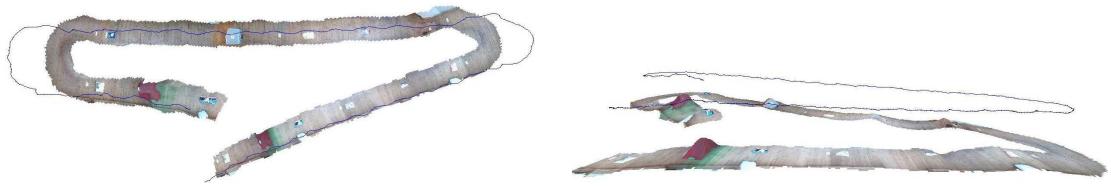


Abbildung 8.5: Pfad projiziert auf ein Satellitenbild des umrundeten Gebäudes (Quelle: Google Earth).

(a) (b) (c)

Abbildung 8.6: Versuchsaufbau zur Bestimmung der Genauigkeit der erlangten Karten, sowie der Nutzbarkeit des Systems im Innenbereich. **(a,b)** Aufnahmen des Aufbaus von gegenüberliegenden Seiten. Die Zettel auf dem Boden sind Markierung mit bekanntem Abstand zueinander. Die anderen dreidimensionalen Objekte dienen der Prüfung der Höhendaten innerhalb der Karte. **(c)** Methode zum Aufnehmen der Daten. Das Kamerasystem mitsamt Lagesensor wurde in der Hand gehalten.



(a) (b)

Abbildung 8.7: Aus der Odometrie resultierender Pfad. **(a)** Ansicht von oben. **(b)** Schrägansicht.

Landmarken. Diese liefern bereits ausreichend unterscheidbare Merkmale für unser System. Die künstlich platzierten Objekte dienten der Statistikerstellung, sowie als dreidimensionale Elemente innerhalb des kartographierten Bereichs.

Die Daten wurden in ähnlicher Weise aufgenommen wie im ersten Experiment. Es wurde jedoch auf den Stock verzichtet und die Sensoren direkt in der Hand gehalten. (siehe Abb. 8.6(c)). Außerdem wurden die Weitwinkelobjektive aus dem ersten Experiment durch Objektive mit einer höheren Brennweite (also einem geringeren Sichtwinkel) verwendet. Diese sind für den begrenzten Raum innerhalb des Korridors besser geeignet.

Es wurden insgesamt zehn Schleifen gelaufen, die jeweils getrennt voneinander verarbeitet wurden. Diese werden im Folgenden genutzt um den Fehler in der Position der bekannten Landmarken zu schätzen. Bei jedem Durchgang wurden zwischen 900 und 1500 Bilder bei einer Auflösung von 320x240 verarbeitet. Abbildung 8.7 zeigt die Ergebnisse der visuellen Odometrie eines dieser Durchläufe. Durch den Drift in der Odometrie ist die Schleife nicht korrekt geschlossen. Es wurden jedoch korrekte Schleifenschlüsse vom System erkannt, was eine Korrektur des Pfades entsprechend Kapitel 7 möglich macht. Die zugehörige optimierte Version ist in Abbildung 8.8 zu sehen. Dem optischen Eindruck nach ist die erlangte Karte konsistent. Wie in der Schrägansicht zu sehen ist, heben sich die dreidimensionalen Objekte, abhängig von ihrer tatsächlichen Höhe, deutlich vom Untergrund ab.

Abbildung 8.9 zeigt die Positionen der platzierten Landmarken mit ihren im Voraus gemessenen Abständen zueinander. Diese Abstände wurden per Maßband ermittelt und haben eine Genauigkeit von ungefähr 3 cm.

Für jeden der zehn Durchläufe wurde innerhalb der resultierenden Karten die Längen der Verbindungslinien a, b, c, d, e, f ermittelt. Die Ergebnisse hierzu sind in Abb. 8.10 und Tabelle 8.1 aufgetragen. Die durchschnittliche prozentuale Abweichung der relativen Positionsschätzungen der Landmarken zueinander vom tatsächlichen Wert ist demnach stets unter 10%. Dieser Fehler resultiert allgemein aus der Ungenau-

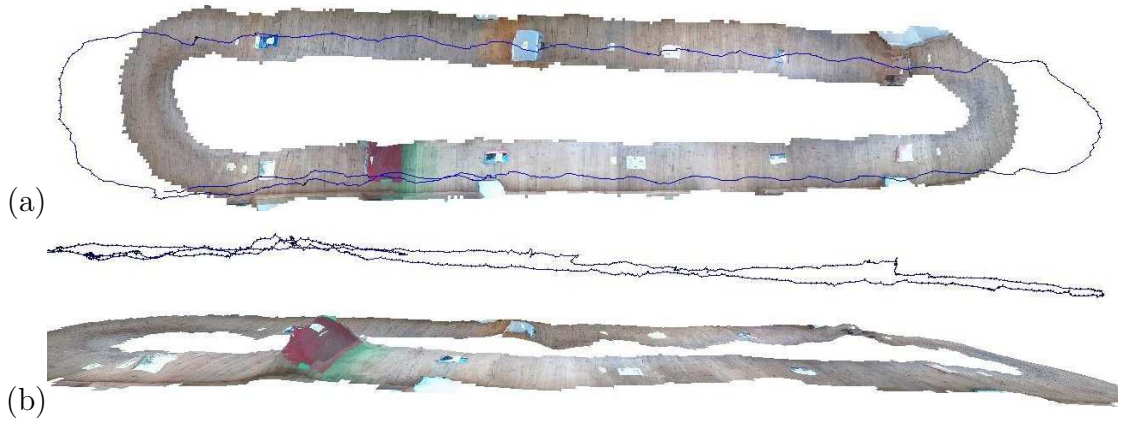


Abbildung 8.8: Anhand von Zirkelschlüssen optimierter Pfad und Karte. **(a)** Ansicht von oben. Dass die Trajektorie breiter erscheint als die darunter liegende Karte resultiert lediglich aus der Perspektive, da die Trajektorie näher am Betrachtungspunkt liegt. **(b)** Schrägansicht.

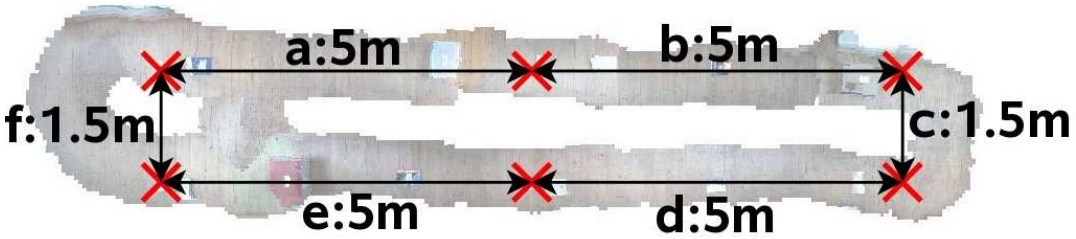


Abbildung 8.9: Die sechs Zettel auf dem Boden markieren die hier hervorgehobenen Positionen mit bekannten Abständen. Diese werden genutzt um die Genauigkeit der Karte zu überprüfen.

igkeit der verwendeten Sensoren, aus falsch erkannten Merkmalen und Ungenauigkeiten in der Kamerakalibrierung. Insbesondere, dass die Messwerte in Abb. 8.10 fast ausschließlich oberhalb der tatsächlichen Werte verlaufen, impliziert einen systematischen Fehler durch das Stereokamerasystem, da dieses das einzige Maß für die Skalierung der Karte liefert. Wie bereits im ersten Experiment erwähnt, können außerdem leichte Fehler in der Synchronität der Stereobilder in Kombination mit schnellen Bewegungen zu Fehlern in der Tiefenerkennung führen.
Unter der Betrachtung, dass für die Gewinnung der Daten weder ein Laser-, bzw. Sonarsensor, noch herkömmliche Odometrie zur Verfügung stand und die verwendeten Kameras einen vergleichsweise niedrigen Qualitätsstandard aufweisen (Standardwebcams), sind diese Werte insgesamt als ein gutes Ergebnis anzusehen.

Neben den Distanzen der Landmarken zueinander wurde außerdem die Genauigkeit der extrahierten Höhen von dreidimensionalen Objekten auf dem Boden überprüft.

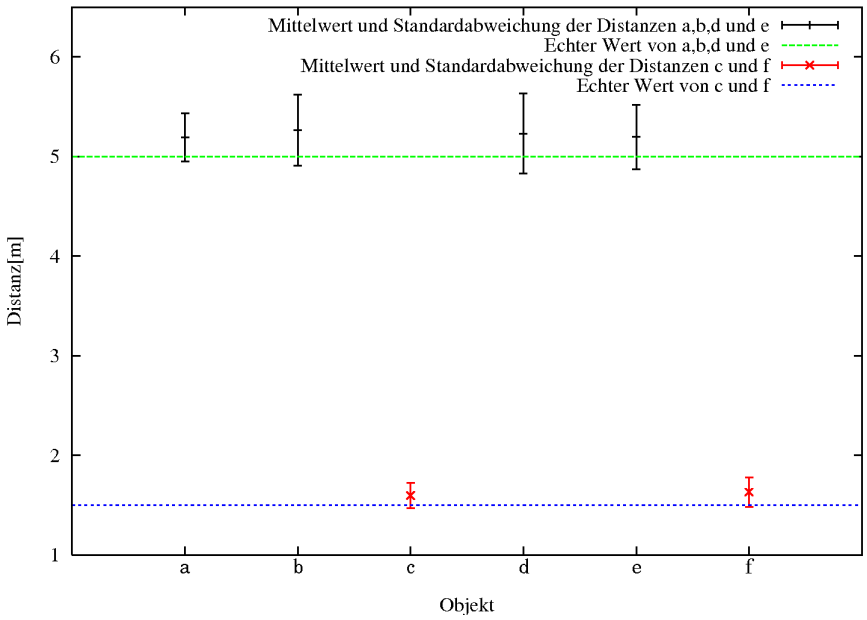


Abbildung 8.10: Darstellung der für die Strecken a, b, c, d, e, f gemessenen Distanzen (siehe Abb. 8.9), jeweils als Mittelwert mit zugehöriger Standardabweichung über den zehn Schleifendurchläufe. Der tatsächliche Wert für die Strecken a, b, d, e betrug $5m$. Der tatsächliche Wert für die Strecken c, f betrug $1.5\,m$.

Strecke	a	b	c	d	e	f	ges.
Tatsächliche Länge [m]	5.00	5.00	1.50	5.00	5.00	1.50	23.00
Durchschn. Kartenlänge [m]	5.19	5.27	1.60	5.23	5.20	1.63	24.11
Durchschn. Fehler [m]	0.19	0.27	0.1	0.23	0.2	0.13	1.11
Standardabweichung [m]	0.24	0.35	0.12	0.4	0.32	0.15	1.32
Durchschn. Abweichung [%]	4.2	6.1	8.1	5.7	4.5	8.6	5.2

Tabelle 8.1: Genauigkeit der gemessenen Abstände der Landmarkenpositionen zueinander. a, b, c, d, e, f sind die Strecken aus Abb. 8.9. Die letzte Spalte beschreibt das komplette Rechteck (23m), das aus den anderen Strecken gebildet wird und gibt grob den genommenen Pfad wieder.

(a) (b)

Abbildung 8.11: **(a)** Eine Kiste bekannter Höhe, die innerhalb der zehn Durchläufe im Bild war. **(b)** Die entsprechende Stelle in einer der erlangten Karten. Hierbei ist zu beachten, dass das Bild in der Karte aus einer Aufnahme von oben resultiert, wodurch hauptsächlich der rote Deckel als Textur zur Verfügung steht. Die runde Form hat ihre Ursache in einer Interpolation zwischen Bereichen verschiedener Höhen in der Visualisierung.

Exemplarisch wird im Folgenden das Ergebnis für das höchste in dem Szenario vertretene Objekt, die Kiste, die in Abb. 8.11 zu sehen ist, angegeben. Dieses Ergebnis ist auch repräsentativ für die restlichen Objekte innerhalb des kartographierten Bereichs.

Die gemessenen Werte für die Höhe sind in Abbildung 8.12 zu sehen.

Diese Werte ergeben im Durchschnitt eine Höhe von 39.3 cm. Die tatsächliche Höhe war 35.5 cm. Der durchschnittliche Fehler beträgt demnach 3.8 cm mit einer Standardabweichung von 7.0 cm. Die durchschnittliche prozentuale Abweichung vom tatsächlichen Wert ist 16.9%.
Die gemessenen Höhen innerhalb der Karte sind tendenziell eher zu hoch als zu niedrig, was insbesondere aus dem Diagramm und dem durchschnittlichen Fehler hervorgeht. Die Berechnung der Bewegung der Kamera in der z-Achse wird durch Änderungen in der Höhe der aufgenommenen Oberfläche erschwert. Dies führt innerhalb der Messungen oft zu leichten Bewegungen der Kamera in Richtung der Änderung der wahrgenommenen Oberfläche, obwohl die Kamera ihre Höhe eigentlich beibehalten hat. Allgemein ausgedrückt ist es schwierig für das System zu unterscheiden, ob sich die Kamera in der Höhe bewegt hat oder sich die Höhe des Bodens verändert hat. Da die Distanz zwischen Kamera und Oberfläche durch die Stereokamera bestimmt wird äußert sich dies dann in einer 'Anhebung' der 3D-Objekte innerhalb der Karte. Dies erklärt eine Tendenz zur Überschätzung der

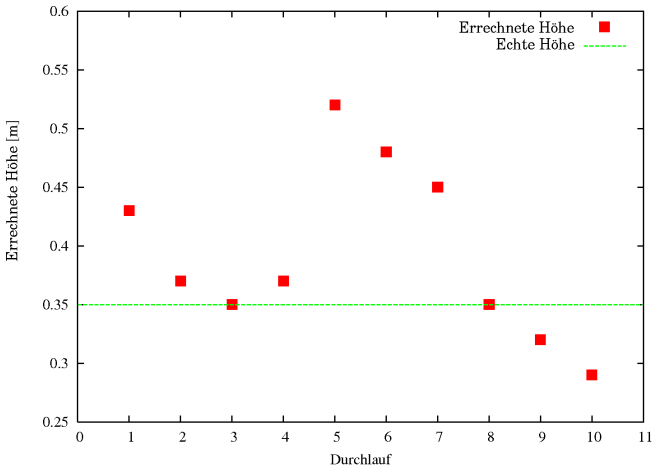
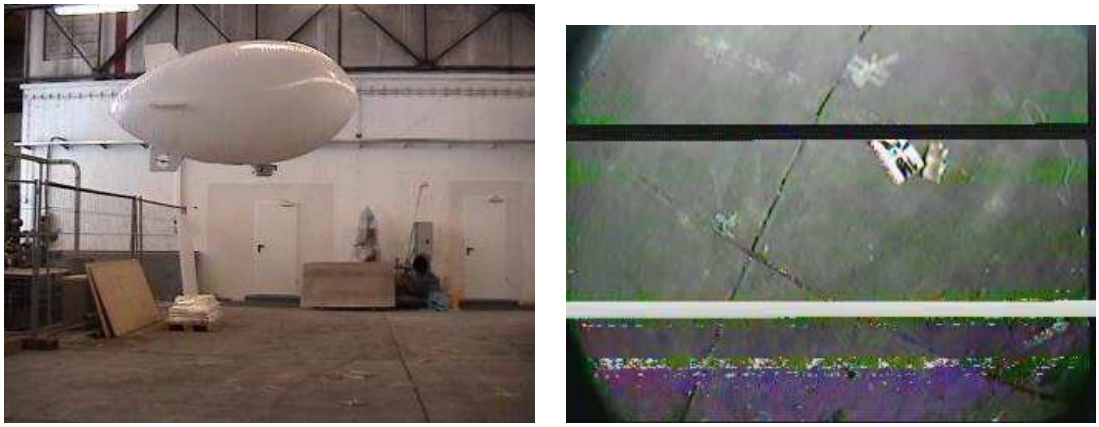


Abbildung 8.12: Darstellung der gemessenen Höhen für das höchste Objekt innerhalb des kartographierten Bereichs. Hierbei handelte es sich um eine Kiste mit einer tatsächlichen Höhe von 35.5 cm.

Höhe von wahrgenommenen Objekten. Außerdem ist natürlich ein systematischer Fehler durch die Kalibrierung des Stereokamerasystems möglich, was konsistent mit den Beobachtungen des ersten Teils dieses Experiments wäre.
Insgesamt ist der beobachtete Fehler zwar höher als in der XY-Ebene, aber unter Beachtung der verwendeten Sensoren sind die Ergebnisse dennoch als gut zu betrachten. Insbesondere wird der grobe Aufbau der Umgebung in den Karten korrekt widergespiegelt.

8.3 Anwendung auf einem echten Fluggerät

Das folgende Experiment sollte die Nutzbarkeit des vorgestellten Ansatzes zum visuellen SLAM in einer realen Anwendung auf einem echten Fluggerät testen. Der Miniaturzeppelin, der in Abbildung 8.13(a) (näher beschrieben in Kapitel C.3 im Anhang) zu sehen ist, trägt eine nach unten gerichtete analoge Funkkamera. Auf Grund der geringen Nutzlast des Zeppelins ist diese Kamera der einzige Sensor, der zur Verfügung steht. Es wurde die Annahme eines flachen Bodens gemacht (siehe Kapitel 3.5), um die fehlenden Stereokameradaten zu kompensieren. Die Daten des Lagesensors wurden durch konstante Nullsignale ersetzt, was der Annahme entspricht, dass die Kamera stets genau nach unten blickt. Leichte Schwankungen des Zeppelins werden zwar so als Bewegungen im Raum interpretiert, aber für die in diesem Experiment durchgeführten Flugmanöver waren diese klein genug, um die Trajektorie nicht übermäßig zu stören. Da die Daten der Kamera analog per Funk übertragen wurden traten öfters Bildstörungen auf (siehe Abb. 8.13(b)), die die Robustheit des vorgestellten Ansatzes auf die Probe stellen. Das Experiment



(a) (b)

Abbildung 8.13: **(a)** Der verwendete Miniaturzeppelin in der Halle, die für das Experiment genutzt wurde. **(b)** Beispiel für die Störungen im Bild der analogen Funkkamera. Problematisch ist insbesondere, dass Merkmale an den Rändern der Streifen im Bild gefunden werden, die zur Erkennung von fehlerhaften Bewegungen führen können.

wurde in einer Fabrikhalle durchgeführt (siehe Abb. 8.13(a)). Diese besitzt einen Betonboden, dessen gleichmäßige und geringe Textur eine weitere Schwierigkeit für das System darstellt. Zur Markierung von Start- und Endpunkten wurde ein Buch auf dem Boden platziert, das gleichzeitig gewährleisten sollte, dass genug Merkmale für das Schließen von Schleifen vorhanden waren. Der Zeppelin wurde ferngesteuert und flog in Bahnen, die ihn öfters zu diesem Punkt zurück brachten. 590 Bilder mit einer Auflösung von 320x240 Pixeln wurden verarbeitet.

Die extrahierte Trajektorie, sowie die Karte in optimierter wie nicht optimierter Form sind in Abbildung 8.14 zu sehen.
Das etwas chaotische Aussehen der nicht optimierten Form kommt teilweise daher, dass jeweils zu bereits bekannten Positionen 'gesprungen' wurde, sofern diese wiedererkannt wurden. Dies ist in Szenarien in denen oft zu bekannten Orten zurückgekehrt wird sinnvoll um einen übermäßigen Drift vor dem Optimierungsschritt zu verhindern. So bleibt der Fehler außerdem eher im Bereich der lokalen Suche, wodurch die globale Lokalisierung nicht so stark gefordert ist.
Die kleinen Schleifen und Unregelmäßigkeiten, die auch in der optimierten Version vorhanden sind resultieren aus Schwankungen des Zeppelins (Änderungen von Pitch und Roll), die ohne einen Lagesensor, wie oben erwähnt, als Bewegungen in x- und y-Achse gewertet werden.
Mit nur einer Kamera fehlen Tiefeninformationen und somit jegliches Wissen über die Skalierung der erlangten Karte. Daher können hier keine konkreten Größenverhältnisse der Karte oder Trajektorie angegeben werden. Der optische Eindruck der resultierenden Karte lässt jedoch Rückschlüsse auf die Korrektheit der extrahierten

Trajektorie zu. Betrachtet man beispielsweise die Linien auf dem Boden innerhalb der optimierten Karte, so kann man sehen, dass die Bilder bis auf kleine Störungen korrekt zusammengesetzt wurden. Außerdem entspricht der gezeigte Pfad den durchgeführten Flugmanövern. Selbst unter den schwierigen Bedingungen dieses Versuchs (wenig strukturierter Boden, eingeschränkte Sensoren) war unser System demnach in der Lage, die Bewegungen des Fluggerätes nachzuvollziehen.

8.4 Leistungsanalyse

Der für diese Experimente genutzte Computer war ein 1.8 GHz Pentium mit Doppelkernprozessor. Die durchschnittliche Framerate für die Berechnung von Odometrie und lokaler Lokalisierung bewegt sich zwischen fünf und zehn Bildern pro Sekunde (Für Bilder mit einer Auflösung von 320x240 Pixeln und 50-100 extrahierten Merkmalen pro Bild). Dieser Wert hängt jedoch von der Qualität der Bilder sowie dem beobachteten Untergrund ab. Die benötigte Zeit pro Bild steigt an, wenn es schwerer ist, eine passende Kameraposition zu finden, da mehr Merkmalspaare geprüft werden müssen (siehe Kapitel 6). In unserer Implementierung wurde die Suche nach 500 ms abgebrochen und das aktuelle Bild als erfolglos verworfen, bzw. keine Änderung in der Position angenommen.
Die durchschnittliche Zeit, die für die globale Lokalisierung benötigt wird, steigt linear mit der Größe der Karte. Im ersten hier vorgestellten Experiment (im Außenbereich) sank die Bildrate in der Offline-Berechnung hierdurch gegen Ende bis auf ungefähr ein Bild pro Sekunde ab.
Für große Karten muss demnach für Online-Berechnungen auf die globale Lokalisierung verzichtet werden. Eine Bildrate von 5Hz wäre ansonsten für die hier durchgeführten Experimente ausreichend gewesen, um diese auch online durchzuführen.

Videos der durchgeführten Experimente sind unter
`http://www.informatik.uni-freiburg.de/~steder/diplomarbeitVideos`
zu finden.

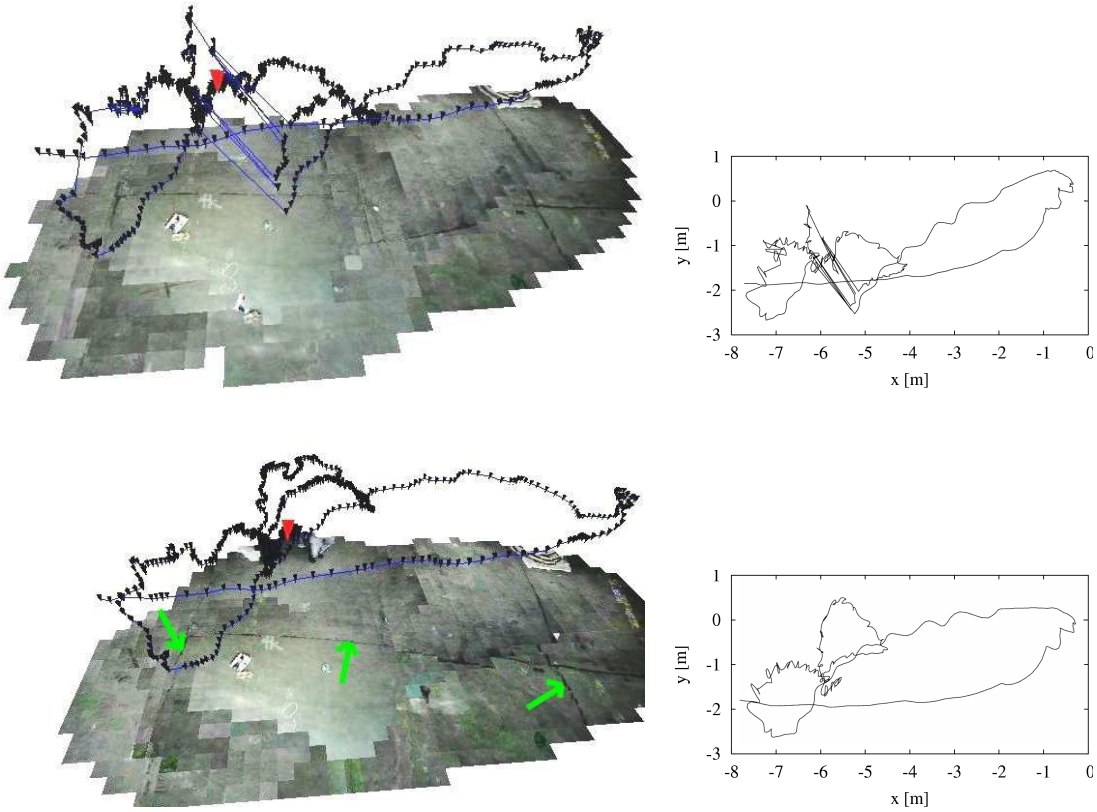


Abbildung 8.14: Oben links: Unoptimierte Karte mit Trajektorie in Schrägansicht. **Oben rechts:** Unoptimierte Trajektorie in Ansicht von oben. **Unten links:** Optimierte Karte mit Trajektorie in Schrägansicht. Die Pfeile markieren den Linienverlauf auf dem Boden, der aus den Übergängen zwischen den Betonplatten resultiert. **Unten rechts:** Optimierte Trajektorie in Ansicht von oben.

9 Zusammenfassung

In dieser Arbeit wurde ein System zum visuellen SLAM vorgestellt, das mittels eines Stereokamerasystems und eines Lagesensors Trajektorien von Fluggeräten nachvollziehen kann und texturierte Höhenkarten des überflogenen Gebietes liefert.
Es wurden verschiedene Methoden der Merkmalsextraktion vorgestellt und erläutert, wie mittels einer dieser Methoden die inkrementelle Schätzung der Bewegung des Fluggerätes, sowie die Erkennung von geschlossenen Schleifen möglich ist. Zur Schätzung der wahrscheinlichsten Trajektorie und Karte wurden Graph-basierte Optimierungsverfahren vorgestellt.
In Experimenten wurde die Robustheit des vorgestellten Systems für verschiedene Anwendungen gezeigt. Auch unter schwierigen Bedingungen und eingeschränkter Sensorenverfügbarkeit konnten konsistente Trajektorien und Karten erzeugt werden.

9.1 Mögliche Problemquellen

Verschiedene Faktoren beeinflussen die Genauigkeit des Systems.
Je stärker und eindeutiger ein Bereich texturiert ist, desto bessere Merkmale können gefunden und verarbeitet werden. Sehr einheitliche Bereiche, wie beispielsweise einfarbige Flächen oder symmetrisch angeordnete Strukturen (z.B. gekachelter Boden) erschweren hingegen die Extraktion unterscheidbarer Merkmale. Ab einem gewissen Punkt ist dann keine Positionsschätzung mehr möglich. Dieses Problem kann vermutlich ohne Änderungen im Aufbau der Hardware (beispielsweise durch zusätzliche Kameras mit anderer Blickrichtung) nicht gelöst werden. Die Verarbeitung von Bildern mit höherer Auflösung könnte jedoch vermutlich helfen, feinere Strukturen auf dem überflogenen Boden zu erkennen.

Weitere Probleme resultieren aus der Qualität der verwendeten Kameras. Das Wiedererkennen von Merkmalen kann durch Probleme bei der Belichtung oder durch Bewegungsunschärfe verhindert werden. Kameras, die weniger empfindlich bezüglich der Belichtung und schnellen Bewegungen sind, könnten hier Abhilfe schaffen. Auch die Verwendung eines Stereokamerasystems, bei dem die Synchronisierung der Bilder besser möglich ist, könnte für den Aufbau genauerer Karten hilfreich sein. Wie bei den ersten beiden Experimenten angesprochen, können leichte Zeitunterschiede

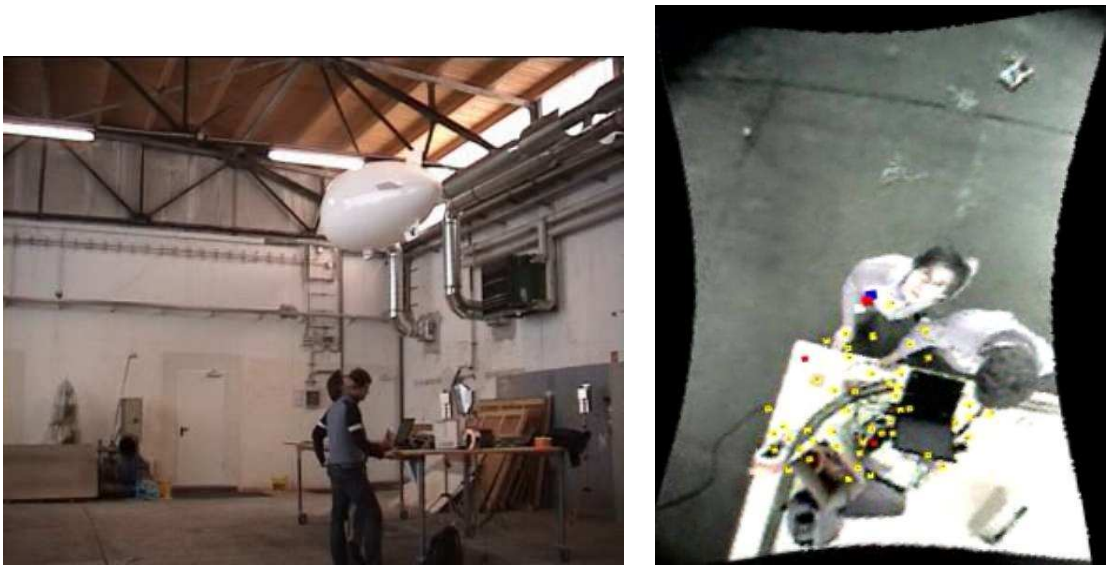
der Stereobilder in Kombination mit schnellen Bewegungen zu Ungenauigkeiten in der Tiefenerkennung führen.

Wie bereits in Kapitel 8.1 angesprochen, können ähnlich aussehende Bereiche zu fehlerhaften Erkennungen geschlossener Schleifen führen. Hierzu sind verschiedene Lösungsansätze denkbar. So könnte beispielsweise eine als geschlossene Schleife erkannte Position für eine gewissen Zeit weiter verfolgt werden, bevor sie als solche akzeptiert wird. Es wäre also mehr als ein einzelnes Kamerabild nötig, um ein loop-closing zu markieren. Dies würde prinzipiell auf einen Partikelfilter mit nur wenigen Partikeln hinauslaufen, die jeweils verschiedene Hypothesen über die Position eine gewisse Zeit verfolgen.
Eine weitere Idee hierzu wäre es, die Erkennung solcher falschen Wahrnehmungen dem Optimierungsalgorithmus zu überlassen. Solche Kanten würden innerhalb des Graphen einen besonders hohen Fehler induzieren und die Optimierung erschweren. Wie sich dies von tatsächlichen Schleifenschlüssen unterscheidet, erfordert jedoch weitere Untersuchungen.

Noch nicht angesprochen wurden in dieser Arbeit dynamische Elemente. Die Welt wurde als statisch angenommen. Allgemein gesehen ist das vorgestellte System relativ robust gegen Bewegungen von Objekten im Bild. Dynamische Elemente sind nur dann ein Problem, wenn sie einen größeren Anteil an Merkmalen auf sich ziehen, als der statische Anteil im Bild. Wenn dies passiert, können Bewegungen der Objekte im Bild als Bewegungen des Flugobjekts interpretiert werden. Ansonsten werden die restlichen Merkmale zur Positionsbestimmung verwendet und die Merkmale auf den dynamischen Objekten werden (sofern sich der dynamische Anteil seit dem letzten Bild bewegt hat) als Ausreißer behandelt. Ein Beispiel für eine Situation, in der die dynamischen Elemente die Posenschätzung nicht gestört haben, ist in Abbildung 9.1 zu sehen. Hier war die Mehrzahl der Merkmale auf dem Tisch und nicht auf den Personen im Bild.
Diese Robustheit bezieht sich allerdings nur auf die visuelle Odometrie. Es kann durchaus passieren, dass ein dynamisches Objekt, dass sich auf eine andere Position bewegt hat als geschlossene Schleife zu seiner früheren Position erkannt wird, was im Optimierungsschritt zu Fehlern führen kann.

9.2 Ausblick

Für die Weiterführung dieser Arbeit sind einige Punkte denkbar, die deren Anwendbarkeit in der Robotik verbessern könnten. So ist beispielsweise eine allgemeine Erhöhung der Performanz wünschenswert um für online-Anwendungen höhere Bildraten zu ermöglichen. Hierzu sind neben low-level-Optimierungen des Programmcodes auch bessere Heuristiken für die Merkmalsauswahl denkbar. Außerdem könnten



(a) (b)

Abbildung 9.1: Beispiel für eine Situation mit dynamischen Elementen im Bild. Auf dem Tisch sind genug statische Elemente, deren Merkmale zur Positionsbestimmung genutzt werden können. Merkmale auf den Personen werden nur dort angezeigt, wo keine Bewegung seit dem letzten Bild stattfand. **(a)** Situation von außen gesehen. Der Zeppelin über einem Tisch an dem zwei Personen arbeiten. **(b)** Bild der Kamera des Zeppelins. Die gelben Punkte markieren Merkmalspositionen, die erfolgreich mit Merkmalen aus dem letzten Bild in Übereinstimmung gebracht wurden.

intelligentere Datenstrukturen zur Speicherung der Karte den Abgleich der Bildmerkmale mit den Kartenmerkmalen enorm beschleunigen.
Auch die Verwendung einer inkrementellen Version des Optimierungsalgorithmus zum Einsatz in online-Anwendungen wäre ein denkbares Arbeitsgebiet.

Wie in Kapitel B im Anhang angesprochen, wurden im Rahmen dieser Arbeit auch Experimente zur Extraktion von Tiefeninformationen mit nur einer Kamera durchgeführt. Weitere Versuche in diesem Bereich könnten es möglich machen, auch in Anwendungen, in denen keine Stereokamera zur Verfügung steht, Karten mit Höheninformationen zu erlangen. Hierzu sind beispielsweise probabilistische Ansätze zur Schätzung der Tiefeninformationen denkbar.

A Berechnung der Kameraposition

Die Berechnung der Kameraposition erfolgt unter der Annahme, dass zwei übereinstimmende Merkmale aus dem aktuellen Bild und der Karte, sowie Rollwinkel (Roll) und Neigungswinkel (Pitch) von dem Lagesensor bekannt sind.
Die normalisierten Bildpunkte $\mathbf{i_1} = (x_{i_1}, y_{i_1}, 1)^T$ und $\mathbf{i_2} = (x_{i_2}, y_{i_2}, 1)^T$ im Kamerasystem entsprechen jeweils den Kartenmerkmalen
$\mathbf{f_1} = (x_{f_1}, y_{f_1}, z_{f_1})^T$ und $\mathbf{f_2} = (x_{f_2}, y_{f_2}, z_{f_2})^T$ im Weltsystem.

A.1 Berechnung der Kamerahöhe

Die Z-Koordinate der Kameraposition im Weltsystem lässt sich mit den bisher bekannten Daten wie folgt berechnen:

Sei R die Rotationsmatrix, die sich aus dem bekannten Roll γ und Pitch β des Lagesensors und einem beliebigen Drehwinkel (Yaw) ergibt.
Es sollen nun normalisierte Bildpunkte berechnet werden, wie sie entstanden wären, wenn die Kamera gerade nach unten gezeigt hätte, d.h., wenn $\gamma = \beta = 0°$ gewesen wären. Hierzu rotieren wir die normalisierten Bildpunkte $\mathbf{i_1}$ und $\mathbf{i_2}$ entsprechend, um sie dann wieder anhand ihrer Z-Koordinate im Kamerasystem zu normalisieren:

$$\mathbf{i_R} = \begin{pmatrix} x_{i_R} \\ y_{i_R} \\ z_{i_R} \end{pmatrix} = R \cdot \mathbf{i}; \qquad \mathbf{i'} = \begin{pmatrix} x_{i'} \\ y_{i'} \\ 1 \end{pmatrix} = \mathbf{i_R} / z_{i_R}$$

Mit den so erhaltenen transformierten Bildpunkten i'_1 und i'_2 lässt sich nun die unbekannte Kamerahöhe h wie folgt berechnen (siehe Abb. A.1):

$$d_1 = \sqrt{x_{i'_1}^2 + y_{i'_1}^2}$$
$$d_2 = \sqrt{x_{i'_2}^2 + y_{i'_2}^2}$$
$$d_3 = \sqrt{(x_{i'_2} - x_{i'_1})^2 + (y_{i'_2} - y_{i'_1})^2}$$
$$\Delta f_z = z_{f_2} - z_{f_1}$$
$$\Delta f_{xy} = \sqrt{(x_{f'_2} - x_{f'_1})^2 + (y_{f'_2} - y_{f'_1})^2}$$

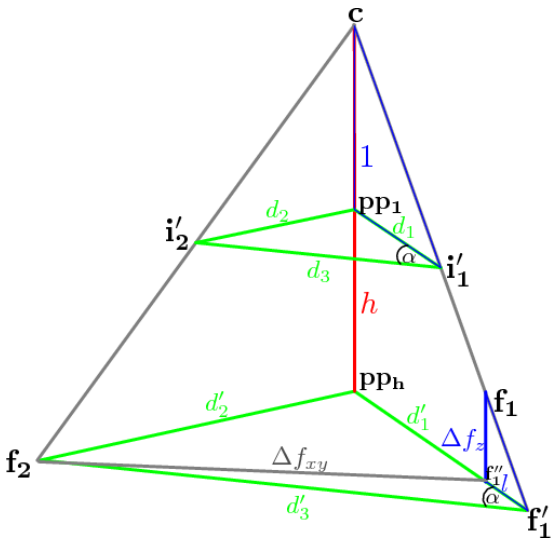


Abbildung A.1: Skizze zur Berechnung der Kamerahöhe h, mit der nach unten gerichteten Kamera im Punkt c und der Projektionsebene im Abstand 1, auf der $\mathbf{i'_1}$ und $\mathbf{i'_2}$ und der normalisierte Bildhauptpunkt $\mathbf{pp_1}$ liegen.

Das Dreieck aus ($\mathbf{c}$, $\mathbf{pp_1}$, $\mathbf{i'_1}$) ist ähnlich zum Dreieck ($\mathbf{f_1}$, $\mathbf{f''_1}$, $\mathbf{f'_1}$). Daher gilt:
$$\frac{l}{d_1} = \frac{\Delta f_z}{1} \quad \Longleftrightarrow \quad l = \Delta f_z \cdot d_1$$

Nach dem Kosinussatz gilt:
$$\cos\alpha = \frac{d_1^2 + d_3^2 - d_2^2}{2 d_1 d_3}$$

Und ebenfalls nach dem Kosinussatz gilt:
$2 l d'_3 \cos\alpha = l^2 + d'^2_3 - \Delta f^2_{xy} \quad \Longleftrightarrow \quad d'^2_3 - 2l(\cos\alpha) d'_3 + l^2 - \Delta f^2_{xy}$
Auflösen der quadratischen Gleichung nach d'_3 ergibt:
$$d'_{3_{1,2}} = l(\cos\alpha) \pm \sqrt{l^2(\cos\alpha)^2 - l^2 + \Delta f^2_{xy}}$$
Sofern $l \leq f_{xy}$ bleibt nur die Lösung $d'_3 = l(\cos\alpha) + \sqrt{l^2(\cos\alpha)^2 - l^2 + \Delta f^2_{xy}}$
Der andere Fall bleibt unbehandelt, da die Merkmale der entsprechenden Paare hierzu im Bild sehr nahe zusammen liegen müssen. Solche Merkmalspaare sind für die Berechnung von vornherein ungeeignet und werden bereits im Voraus herausgefiltert.

Das Dreieck aus ($\mathbf{pp_1}$, $\mathbf{i'_2}$, $\mathbf{i'_1}$) ist ähnlich zum Dreieck ($\mathbf{pp_h}$, $\mathbf{f_2}$, $\mathbf{f'_1}$).
Daher gilt: $\frac{d'_3}{d_3} = \frac{d'_2}{d_2}$
Nach dem Strahlensatz gilt: $\frac{d'_2}{d_2} = \frac{h}{1}$

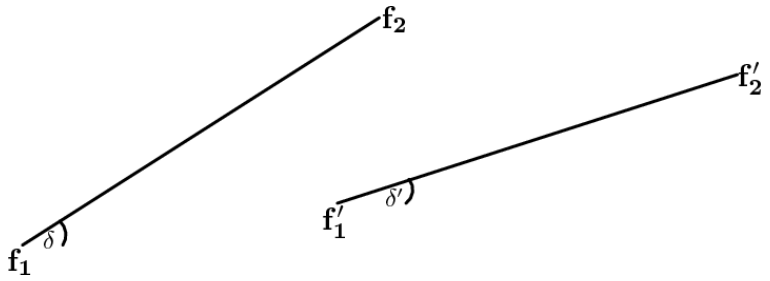


Abbildung A.2: Skizze zur Berechnung des Yaws α der Kamera (von oben betrachtet). $\mathbf{f_1}$ und $\mathbf{f_2}$ sind die tatsächlichen Landmarkenpositionen in der Karte, $\mathbf{f'_1}$ und $\mathbf{f'_2}$ die nach aktuellem Wissen über die Kameraposition ins Weltsystem transformierten Bildpunkte.

Und somit: $\frac{d'_3}{d_3} = \frac{h}{1} \quad \Longleftrightarrow \quad h = \frac{d'_3}{d_3}$

Somit ist die Kamera um h höher als $\vec{f_2}$.
Daraus folgt für die Kamerahöhe: $z_{Cam} = h + z_{f_2}$

A.2 Berechnung des Drehwinkels

Mit den bis jetzt bekannten Daten lässt sich nun der Drehwinkel (Yaw) berechnen. Hierzu werden die beiden Merkmale nach aktuellem Wissen in die Welt projiziert und dann so gedreht, dass ihre Verbindungslinie parallel zu ihrem Gegenstück in der Karte liegt. Dies wird berechnet wie folgt:

Sei R die Rotationsmatrix, die sich aus dem bekannten Roll und Pitch des Lagesensors und einem beliebigen Yaw α_R ergibt. Sei $\mathbf{c}$ die bisher bekannte Kameraposition, also die soeben berechnete Z-Position, sowie beliebige X,Y-Positionen.

Anhand dieser Werte berechnen wir die Weltkoordinaten $\mathbf{f'_1} = (x_{f'_1}\ y_{f'_1}\ z_{f_1})^T$ und $\mathbf{f'_2} = (x_{f'_2}\ y_{f'_2}\ z_{f_2})^T$ von i_1 und i_2 mittels der bekannten Z-Koordinaten der Landmarken (siehe Kapitel 3.5).

Es ergibt sich die Situation in Abb. A.2, aus der der Kamera-Yaw wie folgt berechnet werden kann:

$\delta = \arctan 2(y_{f_2} - y_{f_1}, x_{f_2} - x_{f_1})$
$\delta' = \arctan 2(y_{f'_2} - y_{f'_1}, x_{f'_2} - x_{f'_1})$

Der Unterschied $\Delta\delta$ zwischen δ und δ' entspricht gerade dem Fehler des aktuel-

len Yaws in R. D.h. der gesuchte Yaw α entspricht $\alpha_R + \Delta\delta$ (normalisiert auf $(-180°, 180°]$).

A.3 Berechnung der X,Y-Koordinate der Kamera

Für die Berechnung der noch unbekannt verbliebenen X- und die Y-Koordinaten wird nur noch eine erkannte Landmarke benötigt. Diese wird nach aktuellem Wissen in das Weltsystem transformiert, um dann die Verschiebung zu ihrem Gegenstück in der Karte zu berechnen.

Sei R die bekannte Rotationsmatrix und $\mathbf{c}$ die bisher bekannte Kameraposition, bestehend aus der bekannten Z-Position, sowie beliebigen X,Y-Positionen x_c und y_c.

Anhand dieser Werte berechnen wir die Weltkoordinaten $\mathbf{f_1'} = (x_{f_1'}\ y_{f_1'}\ z_{f_1})^T$ von i_1 mittels der bekannten Z-Koordinaten der zugehörigen Landmarke (siehe Kapitel 3.5).

Die Verschiebung zwischen $\mathbf{f_1'}$ und $\mathbf{f_1}$, $\mathbf{\Delta f} = \begin{pmatrix} x_{f_1} - x_{f_1'} \\ y_{f_1} - y_{f_1'} \end{pmatrix}$, entspricht dem Fehler in der X,Y-Position der Kameraposition. D.h. die gesuchte X,Y-Position ist $\begin{pmatrix} x_{Cam} \\ y_{Cam} \end{pmatrix} = \begin{pmatrix} x_c + x_{\Delta f} \\ y_c + y_{\Delta f} \end{pmatrix}$.

B Berechnung von Tiefeninformationen

Sofern ein Merkmal aus zwei verschiedenen bekannten Kamerapositionen beobachtet wurde, kann die ansonsten fehlende Tiefeninformation dieses Merkmals berechnet werden. Dieses Prinzip kommt bei einem Stereokamerasystem zum Einsatz. Hierbei handelt es sich um zwei Kameras, die fest miteinander verbunden sind, so dass sich deren relative Position zueinander nicht ändern kann.

Diese Berechnung kann wie folgt angegangen werden:

Sei $\mathbf{p_W}$ der gesuchte Punkt im Weltsystem. Dieser sei aus zwei verschiedenen Kamerapositionen $\mathbf{c_1}, R_1$ und $\mathbf{c_2}, R_2$ beobachtet worden. Die resultierenden normalisierten Bildpunkte seien $\mathbf{i_1} = (x_{i_1}\ y_{i_1})^T$ und $\mathbf{i_2} = (x_{i_2}\ y_{i_2})^T$, bzw. $\mathbf{i_{c1}} = (x_{i_1}\ y_{i_1}\ 1)^T$ und $\mathbf{i_{c2}} = (x_{i_2}\ y_{i_2}\ 1)^T$ im Kamerasystem.

Die Positionen $\mathbf{p_{n1}}$ und $\mathbf{p_{n2}}$ der normalisierten Bildpunkte im Weltsystem lassen sich mittels Gleichung 3.3 berechnen:

$\mathbf{p_{n1}} = R_1\mathbf{i_{c1}} + \mathbf{c_1}$ und $\mathbf{p_{n2}} = R_2\mathbf{i_{c2}} + \mathbf{c_2}$

Diese Punkte bilden gemeinsam mit ihren zugehörigen Kamerapositionen jeweils eine Gerade, auf der auch $\mathbf{p_W}$ liegen muss (siehe Abb B.1).

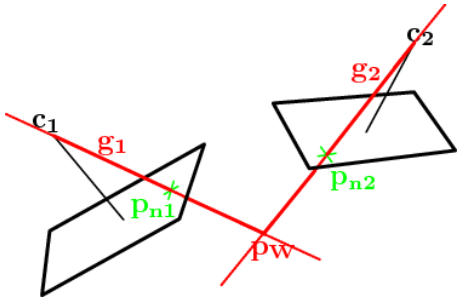


Abbildung B.1: Skizze zu zwei verschiedenen Kamerapositionen c_1 und c_2, die beide den Weltpunkt $\mathbf{p_W}$ im Bild haben.

Für $\mathbf{d_1} = (x_{d_1}\ y_{d_1}\ z_{d_1})^T = \mathbf{p_{n1}} - \mathbf{c_1}$, $\mathbf{d_2} = (x_{d_2}\ y_{d_2}\ z_{d_2})^T = \mathbf{p_{n2}} - \mathbf{c_2}$
und $\lambda_1, \lambda_2 \in \mathbb{R}$ sind diese Geraden:

$\mathbf{g_1}(\lambda_1) = \lambda_1\mathbf{d_1} + \mathbf{c_1}$ und $\mathbf{g_2}(\lambda_2) = \lambda_2\mathbf{d_2} + \mathbf{c_2}$

Gesucht ist hierbei der Schnittpunkt dieser beiden Geraden, d.h. gesucht sind λ_1^*, λ_2^*, so dass $\mathbf{g_1}(\lambda_1^*) = \mathbf{g_2}(\lambda_2^*)$. Dies ergibt ein Gleichungssystem mit drei Gleichungen und zwei Unbekannten:

$$\begin{aligned} & g_1(\lambda_1^*) = g_2(\lambda_2^*) \\ \Leftrightarrow\quad & \lambda_1^*\mathbf{d_1} + \mathbf{c_1} = \lambda_2^*\mathbf{d_2} + \mathbf{c_2} \\ \Leftrightarrow\quad & \mathbf{d_1}\lambda_1^* - \mathbf{d_2}\lambda_2^* = \mathbf{c_2} - \mathbf{c_1} \\ \Leftrightarrow\quad & \begin{pmatrix} x_{d_1} & x_{d_2} \\ y_{d_1} & y_{d_2} \\ z_{d_1} & z_{d_2} \end{pmatrix} \begin{pmatrix} \lambda_1^* \\ -\lambda_2^* \end{pmatrix} = \begin{pmatrix} x_{c_2} - x_{c_1} \\ y_{c_2} - y_{c_1} \\ z_{c_2} - z_{c_1} \end{pmatrix} \end{aligned}$$

Für den Fall, dass sich die beiden Geraden schneiden, hat dieses Gleichungssystem eine eindeutige Lösung für λ_1^* und λ_2^*. In der Realität haben wir es jedoch mit verrauschten Werten zu tun. Die Geraden werden sich demnach höchstwahrscheinlich nicht genau schneiden, sondern lediglich nahe aneinander vorbei gehen. Wir suchen nun die Punkte auf den Geraden, die am nächsten beieinander liegen. Betrachtet man obiges Gleichungssystem, so hat dieses die Form $A\mathbf{x} = \mathbf{b}$. Und da diese Gleichheit nicht gilt, soll statt dessen das folgende Minimierungsproblem gelöst werden: $\min_x ||Ax - b||_2$

Eine Lösung für Probleme dieser Form ist in der Literatur unter dem Begriff *Methode der kleinsten Quadrate (Least Squares Method)* zu finden [44]. Hiernach erfüllt dies die folgende Gleichung:

$$A^TA\mathbf{x} = A^T\mathbf{b} \Leftrightarrow \mathbf{x} = (A^TA)^{-1}A^T\mathbf{b}$$

Die Invertierung von A^TA kann mittels der normalen Invertierung von 2x2-Matrizen erfolgen:

$$M^{-1} = \frac{1}{m_{11}m_{22} - m_{12}m_{21}} \begin{pmatrix} m_{22} & -m_{12} \\ -m_{21} & m_{11} \end{pmatrix} \text{ für } M = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix}$$

Somit lassen sich λ_1^*, λ_2^* und damit $\mathbf{p_{W1}} = \mathbf{g_1}(\lambda_1^*)$, $\mathbf{p_{W2}} = \mathbf{g_2}(\lambda_2^*)$ berechnen, wobei im Idealfall $\mathbf{p_{W1}} = \mathbf{p_{W2}} = \mathbf{p_W}$ gilt.

Diese Berechnung kann unter anderem aus zwei Gründen scheitern:

- $\mathbf{i_1}$ und $\mathbf{i_2}$ markieren keine übereinstimmenden Merkmale (ein Mismatch). Fälle wie diese äußern sich üblicherweise darin, dass sich $\mathbf{p_{W1}}$ und $\mathbf{p_{W2}}$ stark

voneinander unterscheiden. Wie bereits in Kapitel 6.3 erläutert, ist jedoch ein Fehler im Kamerabild gegenüber einem Fehler im Weltsystem aussagekräftiger. Daher projizieren wir den Punkt $\mathbf{p_{W1}}$ in das Kamerabild von $\mathbf{i_2}$, bzw. umgekehrt. Das Maß der Abweichung in Pixeln kann nun als Fehlerwert verwendet werden. Eine Abweichung von mehr als 2.5 Pixeln wurde in der Implementierung zu dieser Arbeit als Zeichen für eine falsche Übereinstimmung behandelt. Selbiges gilt für Ergebnisse, bei denen die Merkmalspositionen hinter der Kamera liegen ($\lambda_1^* < 0 \vee \lambda_2^* < 0$)

- Der Abstand des Merkmals zu den Kameras ist im Verhältnis zum Abstand der beiden Kamerapositionen zu groß. Je weiter c_1 und c_2 voneinander entfernt sind, desto weitere Abstände zum Merkmal können gemessen werden. Allerdings verkleinert sich so natürlich auch die Überlappung der Bilder und damit die Anzahl der aus beiden Positionen beobachteten Merkmale. Um ein Maß dafür zu erhalten, wie genau die Wahrnehmung des Abstandes ist, können die Bildpunkte betrachtet werden, die für $R_1 = R_2$ entstanden wären. Wie diese berechnet werden, wurde bereits in Kapitel A.1 angegeben. Die aus diesen Punkten ablesbare Verschiebung des Merkmals im Kamerabild kann bereits im Voraus verwendet werden, um zu beurteilen, ob die Merkmalsposition zuverlässig bestimmt werden kann. Je kleiner der Abstand vom Merkmal zur Kamera, desto größer die Verschiebung im Bild und desto genauer kann die Merkmalsposition berechnet werden. In der Implementierung zu dieser Arbeit werden Berechnungen verworfen, wenn die Verschiebung im Bild kleiner als 15 Pixel ist.

Mit Hilfe eines Stereokamerasystems, für das die relativen Positionen der Kameras zueinander bekannt sind, lässt sich also für einige der extrahierten Merkmale deren 3D-Position berechnen. Bedingung hierfür ist, dass das Merkmal in beiden Bildern sichtbar ist, erfolgreich wiedererkannt wurde und nicht zu weit entfernt ist. Die Merkmale, für die keine Tiefeninformationen berechnet werden konnten, wurden in dieser Arbeit dennoch in die Karte aufgenommen. Ihnen wird die Z-Position ihres nächsten Nachbarn mit verfügbarer 3D-Information zugeteilt. Zu einem späteren Zeitpunkt kann diese Information aktualisiert werden, wenn das Merkmal erfolgreich wiedererkannt wurde.

Im Rahmen dieser Arbeit wurde kurzzeitig mit einem Ansatz experimentiert, mit nur einer Kamera Tiefeninformationen zu berechnen. Hierzu wurde mit Hilfe von Aufnahmen aus verschiedenen Positionen versucht, die Tiefeninformation zu extrahieren. In obiger Berechnung wurde dazu jeweils die SLAM-Schätzung für die Kamerapositionen verwendet. Diese Versuche verliefen nicht zufriedenstellend. Die Berechnung von Tiefeninformationen erfordert, dass die Kamerapositionen möglichst genau bekannt sind. Dies ist in diesem Fall leider nicht gewährleistet, da die

Berechnung der Kamerapositionen zuerst auf Merkmalen mit unbekannter Tiefe erfolgt. Daher wurde sich für diese Arbeit auf die Verwendung von Stereokameras beschränkt. Monokameras wurden nur unter der Annahme verwendet, dass der Boden flach ist, also eine einheitliche Höhe hat.

Die relativen Positionen der Kameras im Stereokamerasystem wurden für diese Arbeit, wie schon die internen Kameraparameter, mit der *Matlab Camera Calibration Toolbox*[22] berechnet.

C Verwendete Hardware

C.1 Stereokamerasystem

Abbildung C.1: Das verwendete Stereokamerasystem.

Für das Stereokamerasystem wurden zwei Webcams der Marke *Logitech QuickCam Communicate STX Plus* verwendet. Diese haben eine Maximalauflösung von 640x480 Pixeln bei einer Framerate von 30Hz. Die Bilder werden im JPEG-Format per USB an den Computer übermittelt. Diese Kameras sind hauptsächlich für den Inneneinsatz und nur langsame Bewegungen geeignet. Im Außenbereich sind die Bilder schnell überbelichtet und bei schnellen Bewegungen verschwimmen die Bildelemente.

Für die Experimente zu dieser Arbeit wurden die Gehäuse der Kameras entfernt und sie wurden gemeinsam auf einer Kunststoffschiene angebracht, so dass der Abstand variiert werden konnte (siehe Abb. C.1). Teilweise wurden hierbei die Originalobjektive ($\approx 35°$ Sichtwinkel) und teilweise Weitwinkelobjektive ($\approx 73°$ Sichtwinkel) verwendet.

C.2 Lagesensor

Abbildung C.2: Der *MTi* von *Xsens Motion Technologies*

Der verwendete Lagesensor ist der *MTi* von *Xsens Motion Technologies* (siehe Abb. C.2). Dieser Sensor liefert seine Orientierung in 3D mit bis zu 100Hz. Diese können in Form von einer Rotationsmatrix, als Quaternion oder als Euler-Winkel abgefragt werden. In der Implementierung zu dieser Arbeit wurden die Daten als Quaternionen abgefragt und dann jeweils in die zur Anwendung passendste Form umgewandelt. Die Fehler von Neigungswinkel und Rollwinkel sind $< 1°$, da diese hauptsächlich mittels der Erdbeschleunigung gemessen werden. Der auf dem Erdmagnetfeld beruhende Drehwinkel hingegen ist anfällig gegenüber magnetischen Störungen.

Intern wird die Orientierung durch Integration von Daten aus Gyroskopen, Beschleunigungssensoren und Magnetometern in allen drei Dimensionen berechnet. Die Rohdaten der einzelnen internen Sensoren können ebenfalls ausgelesen werden. Im Rahmen dieser Arbeit wurden Versuche mit einer auf den gemessenen Beschleunigungen beruhenden Odometrie durchgeführt. Durch doppelte Integration kann hier ein Rückschluss auf den zurückgelegten Weg gemacht werden. Die Ergebnisse hiervon waren jedoch nicht zufriedenstellend, da sich Fehler selbst in dem kurzen Zeitraum zwischen zwei Kamerabildern zu schnell aufsummierten. Daher wurde diese Methode nicht weiter verfolgt.

C.3 Miniaturzeppelin

(a) (b) (c)

Abbildung C.3: **(a)** Für eines der Experimente eingesetzter Miniaturzeppelin. **(b)** Gondel mit Elektronik und drehbaren Motoren. **(c)** Verwendete analoge Funkkamera.

Der in den Experimenten verwendete Miniaturzeppelin (siehe Abb. C.3(a)) besteht aus einer 1.8 m langen Kunststoffhülle. Er wird von drei Motoren angetrieben, einer in der Heckflosse zur Kontrolle der Rotation und zwei an der Gondel (siehe Abb. C.3(b)) zum Beschleunigen. Letztere sind drehbar, so dass die Richtung der Beschleunigung geregelt werden kann. Die Gondel enthält die für die Steuerung nötige Elektronik sowie einen Lithium-Polymer-Akku. In unserem Experiment war der Zeppelin mit einer analogen Funkkamera (siehe Abb. C.3(c)) mit einem Gewicht von ungefähr 9 g ausgerüstet. Das Gesamtgewicht des Zeppelins beträgt ungefähr 350 g.

Literaturverzeichnis

[1] Thrun, S. ; Burgard, W. ; Fox, D.: *Probabilistic Robotics*. MIT Press, 2005

[2] Leonard, J.J. ; Durrant-Whyte, H.F.: Mobile robot localization by tracking geometric beacons. In: *IEEE Transactions on Robotics and Automation* 7 (1991), Nr. 4, S. 376–382

[3] Smith, R. ; Self, M. ; Cheeseman, P.: Estimating Uncertain Spatial Relationships in Robotics. In: *Proceedings of the 2nd Annual Conference on Uncertainty in Artificial Intelligence (UAI-86)*. New York, NY : Elsevier Science, 1986, S. 435–461

[4] Kalman, R. E.: A New Approach to Linear Filtering and Prediction Problems. In: *Transactions of the ASME, Part D, Journal of Basic Engineering* 82 (1960), Nr. 1, S. 35–45

[5] Maybeck, Peter S.: *Mathematics in Science and Engineering*. Bd. 141: *Stochastic models, estimation, and control*. 1979. – Kapitel 1

[6] Dissanayake, M. ; Newman, P. ; Clark, S. ; Whyte, Durrant H. F. ; Csorba, M.: A solution to the simultaneous localization and map building (SLAM) problem. In: *IEEE Transactions on Robotics and Automation* 17 (2001), Nr. 3, S. 229–241

[7] Thrun, S. ; Liu, Y. ; Koller, D. ; Ng, A.Y. ; Ghahramani, Z. ; Durrant-Whyte, H.: Simultaneous localization and mapping with sparse extended information filters. In: *International Journal of Robotics Research* 23 (2004), Nr. 7/8, S. 693–716

[8] Julier, S. ; Uhlmann, J.: A new extension of the Kalman filter to nonlinear systems. In: *Proceeding of AeroSence. The 11th International Symposium on Aerospace Defence Sensing, Simulation and Controls*, 1997

[9] Wan, Eric A. ; Merwe, Rudolph van d.: The Unscented Kalman Filter for Nonlinear Estimation. In: *Proceedings of Symposium 2000 on Adaptive Systems for Signal Processing, Communication and Control (AS-SPCC)*, 2000

[10] Grisetti, G. ; Stachniss, C. ; Burgard, W: Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters. In: *IEEE Transactions on Robotics* 32 (2006), S. 16–25

[11] Murphy, Kevin: Bayesian Map Learning in Dynamic Environments. In: *Online Proceedings of the Workshop on Adaptive Spatial Representations of Dynamic Environments at the Sixteenth International Joint Conference on Artificial Intelligence*, 1999

[12] Takaoka, Yutaka ; Kida, Yusuke ; Kagami, Satoshi ; Mizoguchi, Hiroshi ; Kanade, Takeo: 3D map building for a humanoid robot by using visual odometry. In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 2004, S. 4444–4449

[13] Dornhege, C. ; Kleiner, A.: Visual Odometry for Tracked Vehicles. In: *Proc. of the IEEE Int. Workshop on Safty, Security and Rescue Robotics (SSRR)*, 2006

[14] Davison, Andrew J. ; Reid, Ian D. ; Molton, Nicholas ; Stasse, Olivier: MonoSLAM: Real-Time Single Camera SLAM. In: *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 2007

[15] Davison, Andrew J. ; Kita, Nobuyuki: 3D Simultaneous Localisation and Map-Building Using Active Vision for a Robot Moving on Undulating Terrain. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* 01 (2001), S. 384–391

[16] Kim, Jong-Hyuk ; Sukkarieh, Salah: Airborne simultaneous localisation and map building. In: *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, 2003, S. 406–411

[17] Eustice, R. ; Singh, H. ; Leonard, J. ; Walter, M. ; Ballard, R.: Visually Navigating the RMS Titanic with SLAM Information Filters. In: *Proceedings of Robotics Science and Systems*, MIT Press, 2005, S. 57–64

[18] Jung, Il-Kyun ; Lacroix, Simon: High resolution terrain mapping using low altitude aerial stereo imagery. In: *9th IEEE International Conference on Computer Vision (ICCV)* 02 (2003), S. 946–951

[19] Elinas, Pantelis ; Sim, Robert ; Little, James J.: σSLAM: Stereo Vision SLAM using the Rao-Blackwellised Particle Filter and a Novel Mixture Proposal Distribution. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006, S. 1564–1570

[20] Brown, D. C.: Decentering Distortion of Lenses. In: *D.C. Brown, Photometric Engineering* 32 (1966), Nr. 3, S. 444–462

[21] *Camera Calibration Toolbox for Matlab - Calibration parameters.* `http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/parameters.html`, Abruf: 08.04.2007

[22] *Camera Calibration Toolbox for Matlab.* `http://www.vision.caltech.edu/bouguetj/calib_doc`, Abruf: 08.04.2007

[23] Shoemake, Ken: *Quaternions.* 2001. – Lecture notes for COMP9018

[24] *Wikipedia - Corner detection.* `http://en.wikipedia.org/wiki/Corner_detection`, Abruf: 08.04.2007

[25] Moravec, Hans: Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover. In: *Tech. report CMU-RI-TR-80-03, Robotics Institute, Carnegie Mellon University and doctoral dissertation, Stanford University.* 1980, Kapitel 5

[26] Harris, C. ; Stephens, M.: A Combined Corner and Edge Detector. In: *Proceedings of The Fourth Alvey Vision Conference*, 1988, S. 147–151

[27] Tomasi, Carlo ; Kanade, Takeo: Detection and Tracking of Point Features / Carnegie Mellon University. 1991 (CMU-CS-91-132). – Forschungsbericht

[28] Shi, Jianbo ; Tomasi, Carlo: Good Features to Track. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1994

[29] Dornhege, Christian: *Untersuchung verschiedener Ansätze zur visuellen Odometrie.* Freiburg, Albert-Ludwigs-Universität, Studienarbeit, 2006

[30] Lindeberg, Tony: Feature Detection with Automatic Scale Selection. In: *International Journal of Computer Vision* 30 (1998), Nr. 2, S. 77–116

[31] Lowe, D.: Distinctive image features from scale-invariant keypoints. In: *International Journal of Computer Vision* Bd. 20, 2003, S. 91–110

[32] Baumberg, Adam: Reliable Feature Matching across Widely Separated Views. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000, S. 1774–1781

[33] Bay, Herbert ; Tuytelaars, Tinne ; Gool, Luc V.: SURF: Speeded Up Robust Features. In: *Proceedings of the ninth European Conference on Computer Vision*, 2006

[34] Mikolajczyk, K. ; Schmid, C.: An Affine Invariant Interest Point Detector. In: *Proceedings of the 7th European Conference on Computer Vision (1) (ECCV)*, 2002, S. 128–142

[35] Fischler, Martin A. ; Bolles, Robert C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: *Commun. ACM* 24 (1981), Nr. 6, S. 381–395

[36] Chum, Ondřej ; Matas, Jiří: Matching with PROSAC - Progressive Sample Consensus. In: *Proc. of Conference on Computer Vision and Pattern Recognition (CVPR)* Bd. 1, IEEE Computer Society, 2005, S. 220–226

[37] Lu, F. ; Milios, E.: Globally Consistent Range Scan Alignment for Environment Mapping. In: *Autonomous Robots* 4 (1997), Nr. 4, S. 333–349

[38] Howard, Andrew ; Matarić, Maja J. ; Sukhatme, Gaurav S.: Localization for Mobile Robot Teams: A Distributed MLE Approach. In: *Experimental Robotics VIII*. Springer-Verlag, 2003 (Advanced Robotics Series), S. 146–155

[39] Frese, U. ; Larsson, P. ; Duckett, T.: A Multilevel Relaxation Algorithm for Simultaneous Localisation and Mapping. In: *IEEE Transactions on Robotics* 21 (2005), Nr. 2, S. 1–12

[40] Olson, E. ; Leonard, J.J. ; Teller, S.: Fast Iterative Optimization of Pose Graphs with Poor Initial Estimates. In: *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, 2006, S. 2262–2269

[41] Grisetti, G. ; Stachniss, C. ; Grzonka, S. ; Burgard, W.: A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent. In: *Proceedings of Robotics: Science and Systems (RSS)*, 2007. – Accepted for publiction

[42] Grisetti, Giorgio ; Grzonka, Slawomir ; Stachniss, Cyrill: *A Tutorial on Constraint-based SLAM*. 2006

[43] Grisetti, G. ; Grzonka, S. ; Stachniss, C. ; Pfaff, P. ; Burgard, W.: Efficient Estimation of Accurate Maximum Likelihood Maps in 3D. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007. – Under review

[44] *Wikipedia - Methode der kleinsten Quadrate*. `http://de.wikipedia.org/wiki/Least_Square`, Abruf: 08.04.2007