ALBERT-LUDWIGS-UNIVERSITÄT
FREIBURG
INSTITUT FÜR INFORMATIK

Arbeitsgruppe Autonome Intelligente Systeme

Prof. Dr. Wolfram Burgard

Performanceanalyse lokaler Feature Detektoren für das Tracking mit KLT, SIFT und SURF

Studienarbeit

Alexander Streicher

Januar 2008

ALBERT-LUDWIGS-UNIVERSITÄT FREIBURG
FAKULTÄT FÜR ANGEWANDTE WISSENSCHAFTEN

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

Performanceanalyse lokaler Feature Detektoren für das Tracking mit KLT, SIFT und SURF

Studienarbeit

Verfasser: Alexander Streicher

Abgabedatum: 15. Januar 2008

Betreuer: Prof. Dr. Wolfram Burgard
Dipl.-Inf. Axel Rottmann

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe.

(Alexander Streicher)

Freiburg, den 16. Januar 2008

Kurzfassung

Diese Arbeit vergleicht die Feature Detektoren KLT, SIFT und SURF im Hinblick auf ihre Performance beim Tracking. Für das Tracking bei mobilen Robotern liefern lokale Features vielversprechende Ergebnisse und werden immer häufiger zur kamerabasierten Lokalisierung eingesetzt. Drei Vertreter lokaler Feature Detektoren werden im Hinblick auf ihre Qualität zur Spurverfolgung untersucht und verglichen. Neben dem KLT Tracker (KLT) werden die bekannten, skalierungsinvarianten Feature Deskriptoren SIFT und die neueren SURF Features betrachtet. Das Tracking wurde mit künstlich erzeugten Bildsequenzen und auf realen Bildsequenzen mit planaren Bildtransformationen durchgeführt. Spezielles Interesse wurde auf das Verhalten der Feature Detektoren bei rauschbehafteten und sich in der Helligkeit ändernden Bildsequenzen mit Stillstand gelegt.

Inhaltsverzeichnis

1. Einleitung

Der vermehrte Einsatz von Computer Vision in der Robotik hat die Bedeutung visueller Merkmale (engl. *Features*) erheblich steigen lassen. Aktuelle Applikationen arbeiten heute nicht mehr nur alleine mit Laser- oder Ultraschallsensoren, sondern setzten auf eine Kombination mit visuellen Daten [8], oder vertrauen allein auf Kameradaten [29].

Roboter müssen für die Interaktion mit anderen Objekten und die selbstständige Navigation genaues Wissen über ihre Umwelt sammeln. Wie beim biologischen Vorbild, der optischen Wahrnehmung der Welt mit den Augen, wird versucht, die Umgebung so präzise wie möglich visuell zu erfassen und relevante Informationen aus dem Bild zu extrahieren. Für spezielle Problemstellungen, zum Beispiel bei der selbstständigen Roboternavigation oder der Erkennung von Personen durch mobile Roboter [3], können visuelle Merkmale wegen ihrer hohen Informationsdichte und Dimensionalität mehr Informationen liefern als etwa Sonar- oder Laserabtastungen. Es liegt also nahe, Kameras zur Erfassung der Umwelt einzusetzen und aus den Bildern Merkmale zu extrahieren, die für autonom agierende Roboter von Interesse sind.

Die Merkmale können in der Roboternavigation zum Generieren von Karten eingesetzt werden, indem eine Menge von Merkmalen als Landmarken in Karten verzeichnet werden. Anhand dieser Landmarken können autonom agierende Roboter sich in ihrer Umwelt orientieren. Im Zusammenhang mit visuellen Merkmalen als Landmarken und der automatischen Kartenerzeugung spricht man in der Robotik von *vSLAM* (*visual Simultaneous Localization and Mapping)*. Üblicherweise werden bei vSLAM die visuellen Daten mit den Daten der Odometrie verknüpft, so dass die Merkmale nur ein Teil des Systems ausmachen. Für manche Anwendungsgebiete, etwa im Bereich der eingebetteten Systeme, wo optische Sensoren wegen ihres geringen Gewichts bei höherem Informationsfluss anderen Sensoren vorgezogen werden, kommt den visuellen Merkmalen dagegen besondere Bedeutung zu, insbesondere wenn die Lokalisierung und Navigation überwiegend mit ihnen realisiert wird [29, 28, 31]. Visuelle Merkmale für die Navigation müssen stabil sein, damit gültige Landmarken daraus erzeugt werden können. Naturgemäß sind optische Daten keine identische Projektion der Welt, sondern variieren wegen vielerlei Einflüsse, wie Verzerrungen, Helligkeits- oder Perspektivänderungen, oder auch Rauschen. Gefordert wird, dass die visuellen Merkmale diese Änderungen möglichst exakt wiedergeben, so dass eine Wiedererkennung gleicher Merkmale im Bild möglich ist. Anhand dieser Merkmale lässt sich dann die Spur eines Roboters verfolgen.

1.1. Zielsetzung

Die Aufgabe dieser Arbeit war die Entwicklung eines Systems für den automatisierten Vergleich von verschiedenen visuellen Merkmalsdetektoren, respektive deren erzeugten Merkmalen, in Hinblick auf ihre Güte zur Spurverfolgung bei translatorischen Bewegungen. Vergleichskriterium ist dabei die Qualität der Spurverfolgung, indem anhand der Abweichung die gemessene Spur mit der tatsächlichen verglichen wird. Als Merkmalsdetektoren werden in dieser Arbeit die *Scale Invariant Features Transform, SIFT* [19], die *Speeded Up Robust Features, SURF* [2] und der *KLT Tracker* [30] im Hinblick auf ihre Performance für das Tracking untersucht. Hierbei ist von Interesse, wie sich die Performance der Merkmalsdetektoren bei qualitativ unterschiedlichen Bildfolgen hinsichtlich Bildstruktur, -textur und -transformation darstellt. Ein besonderes Augenmerk wird auf die Spurverfolgung bei Stillstand und auf den dabei entstehenden Abweichungsfehler gelegt.

Die so gewonnenen Ergebnisse sind für die Roboternavigation von Bedeutung, weil sie bei der Auswahl der Merkmalsdetektoren helfen können. Die Entscheidung für einen Typ von Merkmalen kann, abhängig vom Anwendungsbereich, von enormer Bedeutung für die Performance sein. Werden für einen spezifischen Bereich unpassende Merkmale gewählt, so kann keine robuste Spurverfolgung ausgeführt werden und die Selbstlokalisierung und Kartenerzeugung schlägt fehl.

Diese Arbeit nimmt Bezug auf Anwendungen mit translatorischen Bewegungsmodellen. Beispiele hierfür sind Applikationen mit Draufsichtkameraaufnahmen von Böden, wie sie etwa bei autonom navigierenden Miniaturluftschiffen oder Helikoptern entstehen [25, 26, 35]. Anwendungen mit Luftschiffen setzten neben den Odometriedaten von Höhenmesser und Kompass visuelle Merkmale unterstützend bei der Lokalisierung und Navigation ein [23]. Es gibt aber auch Ansätze, die nur mit visuellen Merkmalen arbeiten [1].

1.2. Aufbau dieser Arbeit

Diese Arbeit ist wie folgt gegliedert:

In Kapitel 2 werden bisherige Arbeiten vorgestellt, die Überschneidungspunkte mit dieser Arbeit aufweisen, beziehungsweise ähnliche Ziele mit anderen Ansätzen verfolgen. In Kapitel 3 werden die in dieser Arbeit verwendeten Merkmalsdetektoren KLT, SIFT und SURF vorgestellt. Der Zusammenhang zwischen den Merkmalen und der Spurverfolgung wird in Kapitel 4 dargestellt sowie die Konzepte für die Spurverfolgung erläutert. Die Details zur Implementierung liefert Kapitel 5. Die durchgeführten Experimente und deren Ergebnisse werden in Kapitel 6 beschrieben. Abschließend werden in Kapitel 7 die erzielten Ergebnisse zusammengefasst und mögliche Erweiterungen und Verbesserungen des Systems diskutiert.

2. Verwandte Arbeiten

Die Bewertung der Performance lokaler Feature Detektoren hat seit ihrem verstärkten Einsatz in kamerabasierten Anwendungen erheblich an Bedeutung gewonnen. Eine Vielzahl von Arbeiten beschäftigen sich mit dem Vergleich lokaler Merkmalsdetektoren. Mikolajczyk und Schmid [21, 22] vergleichen auf großer Breite lokale Merkmalsdeskriptoren im Hinblick auf die Art ihrer Deskriptoren, wie die Merkmale gefunden werden, wie hoch ihre Wiederholungsrate ist, und wie gut sie sich zum Matching eignen. Schmid *et al.* [27] vergleichen die Merkmale im Hinblick auf ihre Wiederholbarkeit und den Informationsgehalt. Merkmale, die stabil auch unter variierenden Bildinhalten in einer Bildsequenz sind, wird eine hohe Wiederholbarkeit zugesprochen, und mittels der Entropie auf den Deskriptoren deren Informationsgehalt beschrieben. Brand und Mohr [5] nehmen die Lokalisationsgenauigkeit als Evaluationsskriterium. Diese gibt an, wie exakt ein Merkmal an einer erwarteten Position gesetzt wird. Eine andere Methode zur Evaluation wird in den Arbeiten von Heath *et al.* und Lopez *et al.* [13, 16] verfolgt, wo die Merkmale mittels visueller Kontrollen miteinander verglichen werden, was naturgemäß subjektiv geprägt ist.

Die vorliegende Arbeit geht auf die Vergleichsmethode mittels Bodenwahrheit ein. Eine andere Arbeit, die dieses Konzept zur Evaluation eingesetzt hat, wird von Bowyer *et al.* [4] vorgestellt, wo anhand der Bodenwarheit Kantendetektoren im Hinblick auf ihre Performance zur Spurverfolgung beurteilt werden.

3. Visuelle Merkmale und Merkmalsextraktion

Visuelle Merkmale kennzeichnen interessante Stellen in Bildern. In der Computergrafik spricht man von *visuellen Features* oder auch Feature Punkten. Neben globalen visuellen Merkmalen, die sich durch eine Beschreibung der Gesamtcharakteristik eines Bildes auszeichnen, zum Beispiel Wavelets [33, 24], gibt es lokale Merkmale, die eine Bildregion charakterisieren [12, 17, 18]. Lokale Features sind seit Jahren aktiver Forschungsgegenstand, die auch in der mobilen Robotik in vielen Szenarien Anwendung fanden [29, 23, 1].

Visuelle Merkmale (Features) parametrisieren Bilder, indem sie Objekte oder Strukturen im Bild beschreiben und so die Bilddaten abstrahieren. Durch den Abstraktionsprozess gehen zwar viele Bildinformationen verloren, andererseits können so die komplexen Datenströme (Bilder) auf für Computer effizient berechenbare Datenstrukturen abgebildet werden. Die berechneten Features müssen wohldefiniert sein, eine möglichst hohe Wiederholungsrate aufweisen, und robust gegenüber Bildtransformationen sein. Ein weiterer Anspruch an visuelle Merkmale ist, dass mit ihnen das Korrespondenzproblem optimal gelöst werden kann. Als Korrespondenzproblem wird das Problem bezeichnet, in zwei Aufnahmen diejenigen Bildpunktepaare zu finden, die auf dasselbe Objekt zeigen. Dazu müssen die Bildobjekte derart durch die Merkmalsdetektoren beschrieben werden, dass sie nachträglich eindeutig identifiziert werden können.

Um visuelle Merkmale zu finden, gibt es verschiedene Methoden. Neben Verfahren, die Feature Punkte an einfachen Bildcharakteristiken, wie Ecken und Kanten [12], oder mittels Textur und Farbe finden, nehmen andere Verfahren den Skalierungsraum als Grundlage für die Extraktion von Features [7, 2, 19, 22].

In diesem Kapitel wird die Merkmalsextraktion mit lokalen Feature Detektoren und die Algorithmen zu den Detektoren KLT, SIFT und SURF beschrieben.

3.1. Merkmalsextraktion

Bei der Merkmalsextraktion werden Feature Punkte aus Bilddaten extrahiert. Ein Feature setzt sich zusammen aus der Information, wo es im Bild lokalisiert ist und einem Feature Deskriptor, der das Feature mit seiner Umgebung charakterisiert. Die einzelnen Feature Detektoren unterscheiden sich sowohl in der Art und Weise wie sie den Bildpunkt finden, als auch wie sie die Bildpunktregion beschreiben.

Für die in dieser Arbeit vorgestellten Feature Detektoren ist die Merkmalsextraktion zweistufig:

1. Erkennung von interessanten Stellen im Bild (*interest points*).

2. Beschreibung der Merkmale durch einen Merkmalsdeskriptor (*feature descriptor*).

Mit *interest points*, auch *local interest points* oder *key point*s, werden bei der Merkmalsextraktion die Stellen im Bild bezeichnet, die in besonderer Weise auffällig sind. Diese Auffälligkeit lässt sich so beschreiben, dass das Bild eine charakteristische Bilddynamik an der gefundenen, interessanten Stelle aufweist. Kontrastreiche Bereiche, in denen sich Strukturen zeigen, stellen für die Merkmalsdetektoren ein Gebiet hoher Resonanz dar, und sind damit eine interessante Stelle. Homogene und konstrastarme Flächen dagegen sind keine interessanten Stellen, da sie wenig Information beinhalten.

Im Zweiten Schritt werden mit den interessanten Stellen zusätzliche Informationen assoziiert, die sich durch die Strukturen der unmittelbaren Umgebung des *interest points* definieren. Die zusätzlichen Informationen werden im so genannten Feature Deskriptor abgelegt und mit der interessanten Stelle verknüpft. Unter dieser Kombination versteht man in der Computergrafik die Bezeichnung *Feature*. Ein Feature drückt also nicht nur die Position eines Merkmals aus, sondern beinhaltet außerdem das Merkmal beschreibende Informationen in Form des Deskriptors. Dieser Deskriptor spielt eine maßgebliche Rolle bei der Zuordnung von zwei Feature Punkten. Mittels Vergleichsmethoden auf den Deskriptoren lassen sich korrespondierende Feature Punkte in zwei unterschiedlichen Bilder finden. Bezogen auf eine Bildsequenz kann somit die Bewegung von Objekten verfolgt werden.

Im Allgemeinen ist der Feature Deskriptor D ein Vektor mit reellen Zahlen und einer spezifischen Länge n

$$D = (\lambda_1, \lambda_2, \ldots, \lambda_n), \qquad \lambda \in \mathbb{R}, n \in \mathbb{N}.$$

Die Länge eines Deskriptors D variiert je nach Detektor. Bei SIFT hat er eine Größe von 128 Einträgen, bei SURF liegt er in der nativen Version bei 64. SURF bietet aber die Möglichkeit, einen erweiterten Deskriptor berechnen zu lassen, der eine Länge von 128 hat (SURF-128).

Abhängig vom verwendeten Merkmalsextraktor werden zusätzliche (Meta-)Informationen zur Deskription eines Merkmal bereitgestellt. Dies sind beispielsweise Angaben zur Orientierung, Stärke, Skalierung oder andere spezifische Informationen, die bei der Merkmalsextraktion ohnehin berechnet und dann zur Verfügung gestellt werden.

Von großer Bedeutung ist, dass ein möglichst eindeutiger Feature Deskriptor erzeugt wird, damit eine robuste Wiedererkennung derselben Feature Punkte in anderen Bildern möglich ist. Dies ist unter dem Begriff der Wiederholbarkeit bekannt. Auch muss gelten, dass die Positionen der Feature Punkte wohldefiniert sind. Gleiche Features müssen in zwei Bildern derselben Szene an denselben Stellen wiedergefunden werden. Anhand den derart gekennzeichneten Bildpunkten lässt sich auf die Bewegung von Bildobjekten schließen, oder, in umgekehrter Betrachtungsweise, auf die Bewegung der Kamera.

Detector	*x/y*	θ	*Scale*	*Strength*	*Sign of Laplacian*	*Descriptor Vector*	*Tracking Status Information*
SIFT	✓	✓	✓	-	-	✓	-
SURF	✓	✓	✓	✓	✓	✓	-
KLT	✓	-	-	-	-	-	✓
U-SURF	✓	-	✓	✓	✓	✓	-
SURF-128	✓	✓	✓	✓	✓	✓*	-

Tabelle 3.1.: Vergleich der Features von KLT, SIFT, SURF und SURF-Varianten. Feature Parameter sind, abhängig vom Detektor, die kartesischen Koordinaten, Rotation, Skalierung, Stärke, Laplace-Vorzeichen und, nur bei KLT, Statusinformationen zum Tracking.
*SURF-128 hat einen 128-stelligen Feature Deskriptor gegenüber 64 Stellen bei SURF.

3.2. Feature Detektoren

Betrachtet man die zeitliche Abfolge der im Folgenden genannten Merkmalsdetektoren, so sind die Features des *Kanade-Lucas-Tomasi Trackers, KLT* (1994) [30] zeitlich zuerst anzuordnen. Die *SIFT*, *Scale Invariant Feature Transform* wurde erstmals 1999 beschrieben [18, 19] und hat seit der Erstbeschreibung einige Ableger hervorgebracht [14, 21, 10], die das grundlegende Prinzip der SIFT verfeinern oder dieses als Grundlage für eigene Features nutzen, wie die 2006 vorgestellten *SURF*, *Speeded Up Robust Features* [2]. Die genannten Verfahren werden jetzt vorgestellt. Auch werden die Unterschiede der Detektoren in Tabelle 3.1 zusammengefasst.

3.2.1. Kanade-Lucas-Tomasi Feature Tracker (*KLT*)

Der *Kanade-Lucas-Tomasi Feature Tracker (KLT)* [30] ist ein Feature Detektor mit integriertem Tracker, d.h. er ermittelt nicht nur die Feature Punkte, sondern nimmt gleichzeit eine Verfolgung dieser Punkte in einer Bildsequenz vor. Das Verfahren basiert auf der 1981 von Lucas und Kanade entwickelten Methode [20] zum Auffinden von Punktkorrespondenzen und Berechnung des optischen Flusses. Daraus entwickelten Tomasi und Kanade 1991 einen Algorithmus [32] zur Verfolgung von Feature Punkten, der 1994 von Shi und Tomasi [30] zu einem Tracker weiter entwickelt wurde. Die Idee ist nur noch diejenigen Feature Punkte auszuwählen, die für die Verfolgung durch den Tracker am geeignetsten sind. Shi und Tomasi sprechen hier von einer durch Konstruktion optimale Auswahl von Features.

Die in dieser Arbeit verwendeten Features und der zugehörige Tracker beziehen sich auf den Ansatz von Shi und Tomasi [30]. Der folgende Abschnitt behandelt die zugrunde liegende Idee zur Auswahl von KLT Feature Punkten und die mathematischen Grundlagen. Die Notation wurde weitgehend aus [32] übernommen.

Um die Bewegung zwischen zwei aufeinander folgenden Bildern zu finden, berücksichtigt KLT die unmittelbare Umgebung eines Punktes. Ein einziger Bildpunkt (*Pixel*) lässt sich nicht zuverlässig verfolgen, da der Pixelwert abhängig ist von auf das Bild einwirkenden Faktoren, wie Rauschen oder Helligkeitsänderungen. Deshalb werden kleine Bildausschnitte (Bildfenster) untersucht. Das Ziel ist, den Versatz dieser Bildfenster zu verfolgen. In einem Fenster $\mathcal{W}$ wird der Versatz durch den Vektor $\mathbf{d}$ ausgedrückt, der den quadratischen Unterschied ϵ zweier Bildfenster minimiert:

$$\epsilon = \int \int_{\mathcal{W}} [J(\mathbf{x}+\mathbf{d}) - I(\mathbf{x})]^2 \, w(\mathbf{x}) \, d\mathbf{x}. \tag{3.1}$$

Für den quadratischen Fehler ϵ wird jeder Punkt $\mathbf{x} = (x, y)^\top$ im betrachteten Fenster $\mathcal{W}$ berücksichtigt, wobei $\mathbf{d} = (d_x, d_y)^\top$ den Versatz darstellt. Die Gewichtungsfunktion $w(\mathbf{x})$ wird üblicherweise auf 1 gesetzt, so dass jedes Pixel im Bildfenster gleich gewichtet wird. Alternativ ließen sich die Zentren der Fenster stärker betonten, indem eine Gauß-Funktion für w verwandt wird.

Um den Vektor $\mathbf{d}$ zu finden wird die Gleichung 3.1 bezüglich $\mathbf{d}$ abgeleitet und gleich Null gesetzt, so dass man die Gleichung

$$\frac{\partial \epsilon}{\partial \mathbf{d}} = 2 \int \int_{\mathcal{W}} [J(\mathbf{x}+\mathbf{d}) - I(\mathbf{x})] \frac{\partial J(\mathbf{x}+\mathbf{d})}{\partial \mathbf{d}} w(\mathbf{x}) \, d\mathbf{x} = 0. \tag{3.2}$$

erhält. Eine Lösung dieser Gleichung lässt sich über die Taylorreihenentwicklung bestimmen. Der lineare Teil der Taylorreihenentwicklung von J bezüglich x ist

$$J(\mathbf{x}+\mathbf{d}) \approx J(\mathbf{x}) + d_x \frac{\partial J(\mathbf{x})}{\partial \mathbf{x}} + d_y \frac{\partial J(\mathbf{x})}{\partial y}.$$

Die lineare Approximation der Taylorreihenentwicklung eingesetzt in Gleichung 3.2 führt mit $\mathbf{g} = \left[\frac{\partial}{\partial x} J, \frac{\partial}{\partial y} J\right]^\top$ zu

$$\frac{\partial \epsilon}{\partial \mathbf{d}} = 2 \int \int_{\mathcal{W}} [J(\mathbf{x}) - I(\mathbf{x}) + \mathbf{g}(\mathrm{x})^\top \mathbf{d}] \mathbf{g}(\mathbf{x}) w(\mathbf{x}) \, d\mathbf{x} = 0.$$

Da $(\mathbf{g} \cdot \mathbf{d}) = (\mathbf{g}\mathbf{g}^\top)\mathbf{d}$, und angenommen wird, dass der Versatz-Vektor $\mathbf{d}$ konstant im Bildfenster $\mathcal{W}$ bleibt, kann man die Gleichung umformen zu

$$\left(\int \int_{\mathcal{W}} g g^\top w(\mathbf{x}) \, d\mathbf{x} \right) \mathbf{d} = \int \int_{\mathcal{W}} (J(\mathbf{x}) - I(\mathbf{x}) \mathbf{g} \, d\mathbf{x}.$$

Dies ist ein lineares 2x2 System mit zwei Unbekannten

$$G\mathbf{d} = \mathbf{e}, \tag{3.3}$$

wobei G die symmetrische 2x2 Matrix

$$G = \int \int_{\mathcal{W}} \mathbf{g}\mathbf{g}^{\top} w(\mathbf{x})\, d\mathbf{x}$$

ist, und $\mathbf{e}$ der zweidimensionale Vektor

$$\mathbf{e} = \int \int_{\mathcal{W}} [I(\mathbf{x}) - J(\mathbf{x})]\mathbf{g} w(\mathbf{x})\, d\mathbf{x}.$$

Die Gleichung 3.3 ist der grundlegende Schritt des KLT Trackings. Um ein Feature verfolgen zu können, muss das lineare System 3.3 gelöst werden können. Dies kann mit der inversen Matrix G^{-1} erreicht werden, deren Eigenwerte λ_1, λ_2 die Intensitätenänderungen in den beobachteten Bildfenstern beschreiben.

Statt von vornherein die Eigenschaften für gute Feature-Fenster zu definieren, basiert die Definition für gute Features auf der Fähigkeit des Tracking Verfahrens: Es werden nur die Features berücksichtigt, die in Fenstern liegen, die gute, sprich nachverfolgbare Features enthalten. Die Güte der Features für das Tracking orientiert sich an den Eigenwerten λ_1, λ_2. Gute Feature Punkte haben große Eigenwerte. Es werden drei Fälle unterschieden:

- $\lambda_1 \approx 0$ und $\lambda_2 \approx 0$, d.h. beide Eigenwerte sind sehr klein. Dies gilt für homogene Flächen mit einheitlichen Intensitäten.
- Ein großer und ein kleiner Eigenwert, $\lambda_1 \gg \lambda_2$ und $\lambda_2 \approx 0$ beziehungsweise umgekehrt, entsprechen einer Kante. Im Bezug zu den Intensitäten, bedeutet dies, dass die Richtung der Kante orthogonal zum Eigenvektor des größten Eigenwerts steht.
- $\lambda_1 \gg 0$ und $\lambda_2 \gg 0$, d.h. beide Eigenwerte sind positiv. Dies spiegelt Texturen wider, die zuverlässig für das Tracking benutzt werden können.

Ein Bildausschnitt wird für Auswahl von Feature Punkten berücksichtigt, wenn

$$min(\lambda_1, \lambda_2) > t$$

wobei t ein vordefinierter Schwellenwert ist. Dieser kann zu Beginn eingegrenzt werden durch die Ermittlung zweier Bildbereiche, bei denen der eine Bereich einheitlich hell ist, und der andere sich durch Strukturen wie Ecken oder andere Texturen auszeichnet. Dies liefert eine untere und eine obere Grenze für t. Es hat sich gezeigt, dass die zwei Grenzen meist weit genug auseinander liegen, so dass als t der Wert in der Mitte der Grenzen gesetzt werden kann [32]. Je größer $min(\lambda_1, \lambda_2)$ ist, desto besser ist ein Bildfenster für das Tracking geeignet.

Die Features bekommt man letztlich aus der Liste der möglichen Features Kandidaten, indem man die Kandidatenliste nach deren Eigenwerten sortiert und die besten n Features auswählt. Der kleinere der beiden Eigenwerte λ_1, λ_2 wird zur Bewertung der Features benutzt. Den Versatz der Feature Punkte erhält man durch das Lösen von Gleichung 3.3 und damit der Bestimmung des zweidimensionalen Vektors $\mathbf{e}$.

Da das Verfolgen von Feature Punkten der KLT Tracker selbst übernimmt, sind direkt keine weiteren Parameter abrufbar als die Angaben zu den x, y-Koordinaten und einer Statusinformation zum Tracking. Ein Matching von Features etwa mittels eines Deskriptors ist nicht sinnvoll möglich, da die Statusinformation zum Tracking nur eine konstante Ganzzahl ist.

3.2.2. Scale Invariant Feature Transform (*SIFT*)

Die *SIFT, Scale Invariant Feature Transform* wurde 1999 von Lowe vorgestellt und 2004 von ihm verbessert [18, 19]. SIFT basiert auf dem Prinzip des Skalierungsraums, es extrahiert Features mit einem 128-stelligem Deskriptor, die weitgehend invariant gegenüber Skalierung und Rotation, affinen Verzerrungen, Rauschen, Änderungen der Perspektive und Helligkeitsänderungen sind. SIFT ist aktueller Forschungsgegenstand und Basis vieler heute eingesetzter Verfahren im Bereich der Computergrafik, wie zum Beispiel die automatische Panoramabilderstellung aus mehreren Fotos [6], und auch in der mobilen Robotik zur Navigation und Kartenerstellung [29, 28].

Die SIFT transformiert Bilddaten in skalierungsinvariante Features. Das Verfahren ist mehrstufig aufgebaut. Teure Rechenoperationen werden nur auf Punkten ausgeführt, die das Verfahren in vorherigen Schritten als interessante Punkte (*keypoints*) deklariert hat. Die SIFT lässt sich nach [19] grob in vier Stufen beschreiben:

1. **Ermittlung potentieller Keypoints** über Extremstellen im Skalierungsraum. Es wird eine *Difference of Gaussian* oder *DoG*-Pyramide aufgebaut und in ihr nach Extremstellen gesucht.

2. **Keypoint Lokalisation.** Die Keypoint-Kanditen werden anhand ihrer Stabilität gefiltert. Für die verbleibenden Punkte wird eine subpixel-genaue Position berechnet.

3. **Erzeugung der Orientierungsinvarianz**. Für jeden Keypoint wird eine Orientierung ermittelt. Kann einem Punkt mehr als eine dominante Richtung zugeordnet werden, so werden weitere Keypoints an dieser Position mit unterschiedlichen Orientierungen angelegt.

4. **Berechnung der Deskriptoren**. Im letzten Schritt wird für die verbliebenen Keypoints ein einzigartiger, charakteristischer Feature-Vektor erzeugt.

Keypoint-Kandidaten

Die SIFT arbeitet nicht direkt auf dem Bild, sondern nutzt eine Repräsentation des Bildes im linearen Skalierungsraum. Dies erlaubt es Features zu generieren, die gegenüber Bildskalierungen invariant sind. Die Suche der Keypoint-Kandidaten erfolgt in zwei Schritten. Zunächst erfolgt der Aufbau einer so genannten Gauß-Pyramide. Diese wird verwendet, um die *Difference-of-Gaussians* (DoG) zu berechnen, und dort über die Differenzbilder die Extremstellen zu bestimmen.

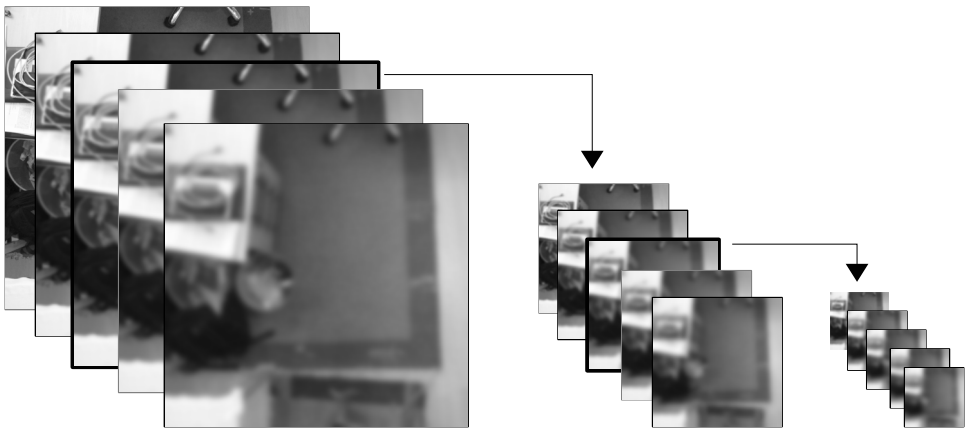


Abbildung 3.1.: Gauß-Pyramide mit drei Oktaven und fünf Stufen pro Oktave.

Zum Aufbau der Gauß-Pyramide wird das Eingangsbild mit verschiedenen Gauß-Kerneln $G(x, y, \sigma)$ gefaltet. Diese so geglätteten Bilder bilden die Stufen der ersten Oktave der Pyramide. Anschließend wird eines der geglätteten Bilder auf ein Viertel seiner ursprünglichen Größe verkleinert und mit dem Gauß-Kernel gefaltet. Ausgeführt auf alle Bilder der Oktave ergibt dies die nächste Oktave der Gauß-Pyramide. Der Vorgang stoppt, wenn die Bilder eine minimale Größe erreicht haben. Zur Veranschaulichung ist in Abbildung 3.1 eine Gauß-Pyramide mit drei verschiedenen Oktaven und fünf Stufen abgebildet.

Die Gauß-Pyramide wird erzeugt, indem die Eingangsbilder mit einem Gauß-Kernel $G(x, y, \sigma)$ gefaltet und in den *Scale Space* $L(x, y, \sigma)$ überführt werden:

$$L(x, y, \sigma) = G(x, y, \sigma) \star I(x, y)$$

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma} e^{-(x^2+y^2)/2\sigma}$$

$I(x, y)$ sind die Grauwertintensitäten des Bildpunktes an der Stelle x, y. Der Parameter σ gibt die Skalierungsebene an, wobei $\sigma = 0$ das Originalbild ist. Für mehrere Werte von σ wird mit der Faltung der Skalierungsraum L aufgespannt. Je größer σ ist, desto unschärfer wird das Bild. Ein Punkt (x, y) in L repräsentiert die Kombination von Bildpunkten einer Umgebung in I. Ein Ausschnitt in einer Skalierungsebene repräsentiert den größeren Ausschnitt der vorherigen Ebene.

Aus den Stufen der Gauß-Pyramide wird über die Differenz zweier benachbarter Gauß-Bilder eine neue Pyramide berechnet (siehe Abbdildung 3.2). Dies ist die *Difference-of-Gaussians* (DoG) Pyramide. Für einen konstanten Faktor k wird die DoG-Pyramide $D(x, y, \sigma)$ über die Differenzen benachbarter Stufen berechnet

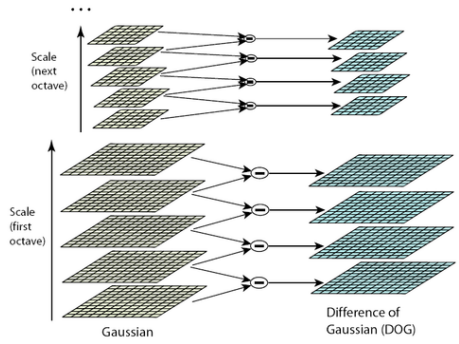


Abbildung 3.2.: Die Differenz benachbarter Gauß-Bilder links liefert die Difference-of-Gaussians Pyramide rechts. [19]

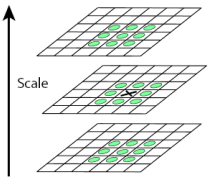


Abbildung 3.3.: Die Extremstellen der DoG werden über den Vergleich eines Pixels mit seinen 26 Nachbarn in derselben und benachbarten Skalierungen gesucht. [19]

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma).$$

Die DoG stellen eine gute Approximation der *Laplacian-of-Gaussians* (LoG), $\sigma^2\nabla^2 G$, wie von Lindeberg [15] untersucht wurde.

Die Positionen der Keypoints ergeben sich aus den Maxima und Minima der DoG-Pyramide. Jeder Pixel auf der DoG-Pyramide wird mit seinen acht Nachbarn in gleicher Stufe und mit jeweils neun Pixeln in der darüber- und darunterliegenden Stufe verglichen (siehe Abbildung 3.3). Er wird nur dann ausgewählt, wenn sein Wert echt größer oder echt kleiner als all die Werte dieser Nachbarn ist.

Filterung und Lokalisierung

Einige der Keypoints sind als Kandidaten ungeeignet, wenn sie in an Stellen mit niedrigem Kontrast oder immer entlang einer Kante gefunden werde. Die Filterung der Keypoints mit niedrigem Kontrast wird mit einem Schwellenwert auf den Difference-of-Gaussians realisiert. Alle Diffe-

renzen, die größer als dieser Schwellenwert sind, haben einen niedrigen Kontrast und werden eliminiert. Die Filterung an den Kanten lässt sich mit Hilfe der 2x2 Hesse-Matrix H ausführen

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix},$$

wobei D_{xx}, D_{xy}und D_{yy} die zweifachen Ableitungen des Differenzbildes auf einer Stufe in x-, in y- bzw. in x- und y-Richtung darstellen. Die Eigenwerte der Hesse-Matrix sind proportional zu den Hauptkrümmungen von D, und das Verhältnis der beiden Hauptkrümmungen ist gleich dem Verhältnis der beiden Eigenwerte. Anhand dieses Kriteriums können die Keypoints entlang einer Kante eliminiert werden. Zur Subpixellokalisation wird versucht eine dreidimensionale Parabel durch die Nachbarpunkte des Keypoints in der DoG-Pyramide zu legen. Dies liefert eine 3x3 Hesse-Matrix, in der die DoG unmittelbarer Punktnachbarn berücksichtigt werden.

Orientierung

Für die Rotationsinvarianz der Keypoints werden zunächst die Gradienten jedes Pixels berechnet. Diese setzten sich aus einer Magnitude m und einer Richtung θ zusammen

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

$$\theta(x,y) = tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y)))$$

Aus den Gradienten der Regionen um den Keypoint werden im zweiten Schritt Orientierungshistogramme erstellt. Die 360 Grad der Gradientenrichtungen werden in 36 Bins eingeteilt. Der Einfluss jedes Eintrags wird mit seiner Magnitude gewichtet. Die Magnituden werde zusätzlich mittels einer Gauß-Funktion gelättet, so dass eine Gewichtung der Magnituden zur Mitte hin hergestellt wird (siehe linkes Bild in Abbildung 3.4).

SIFT Deskriptor

Im letzten Schritt wird den Keypoints charakteristische Feature-Vektoren zugeordnet, die über die Umgebung der Keypoints definiert werden. In Abbildung 3.4 ist dies veranschaulicht. In einer Umgebung um den gefundenen Keypoint werden wie zuvor die Gradienten, bestehend aus Magnitude m und Richtung θ, eines Gauß-geglättenen Bildes ermittelt. Diese Umgebung umfasst üblicherweise 16x16 Pixel und ist entsprechend der Orientierung des Keypoints gedreht. Sie wird unterteilt in 16 Unterregionen der Größe 4×4, und für jede der 16 Unterregionen wird ein Gradientenhistogramm erstellt. Ein Histogramm besteht aus acht Bins für die verschiedenen Gradientenrichtungen. Wie bei der Berechnung der Orientierung wird hier jeder Eintrag mit seiner Magnitude gewichtet. Zusätzlich werden mittels einer Gauß-Funktion Veränderung in weit vom Keypoint entfernten Punkten weniger stark gewichtet, damit diese den Deskriptor nur minimal beeinflussen. Die insgesamt sechzehn Histogramme mit Einträgen in jeweils acht Bins

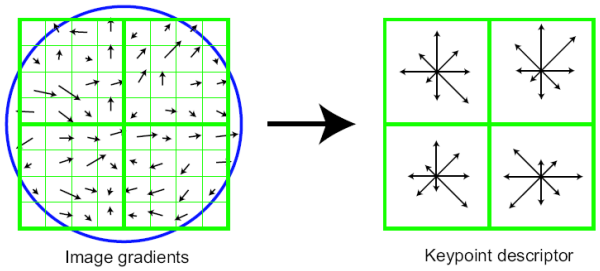


Abbildung 3.4.: Der SIFT Deskriptor berechnet sich aus den Gradientenmagnituden und -orientierungen einer 16x16 Pixel großen Umgebung um den Keypoint. Die Magnituden werden mit einer Gauß-Funktion zur Mitte hin gewichtet, wie mit dem blauen Kreis angedeutet ist. Die so ausgewählten Einträge werden in Orientierungshistogrammen zusammengefasst. [19]

bilden den 128-dimensionalen Feature Deskriptor. Dieser wird normalisiert, um ihn invariant gegenüber Helligkeitsänderungen zu gestalten.

3.2.3. Speeded Up Robust Features (*SURF*)

Die *Speeded Up Robust Features, SURF* ist ein Verfahren, das 2006 von Bay *et al.* [2] vorgestellt wurde. Es basiert auf den Ideen von SIFT, ist aber eine beschleunigte Version, die robustere Features erzeugt. Dabei faltet SURF ein Bild nicht mit einer Gauß-Funktion sondern mit einer diskreten Maske. Durch ein Integralbild, das einer Summentabelle entspricht, kann die Faltung beschleunigt und nebenher robuste Features erzeugt werden. Die Robustheit der Features zeichnet sich durch Invarianzen gegenüber vielen Bildänderungen aus. Die Features sind invariant gegenüber Translation, Rotation, Skalierung und Beleuchtung, affinen Verzerrungen, und kleinen Abweichungen der Perspektive. Der Deskriptorvektor ist in der nativen Version 64-stellig, mit der Erweiterung *SURF-128* verdoppelt er sich auf 128 Stellen. Eine weitere Version, die *Upright SURF (U-SURF)*, lässt die Rotationsinvarianz unberücksichtigt und fokussiert auf rein translatorischen Bewegungsmodellen, was einen Geschwindigkeitsvorteil verspricht. Die Features der Detektoren SIFT und SURF unterscheiden sich in den zusätzlichen Parametern *Strength* und *Sign of Laplacian*. Das Laplace-Vorzeichen gibt an, ob ein Feature an einer hellen Bildstelle mit dunklem Hintergrund oder umgekehrt gefunden wurde. Dies ermöglicht ein schnelleres Matching, wie es von Valgren *et al.* [34] demonstriert wird.

3.2.4. Verwandte Deskriptoren

Als Alternative stellen Ke und Sukthankar zu Lowes SIFT die *PCA-SIFT* vor [14]. Dieser Algorithmus verwendet die Principal Component Analysis (PCA) zur Berechnung der Deskriptoren und liefert einen mittels PCA komprimierten Feature Deskriptor. Die geringere Größe bei

dennoch hohem Informationsgehalt ist beim Matching und Speicherverbrauch von Vorteil, aber wegen der hohen Berechnungszeit der PCA langsamer als etwa SIFT.

Mikolajczyk und Schmid erweiterten den PCA Ansatz zum *Gradient Location and Orientation Histogramm, GLOH* Deskriptor *[21],* bei dem zwar der Deskriptor wie bei SIFT aus Gradientenhistogrammen erstellt wird, die Einteilung der Unterregionen aber in radiale Zonen statt in ein Gitter vorgenommen wird. Die Anzahl der Bins erhöht sich bei GLOH auf 272, was einen stabileren Deskriptor hervorbringt. Es hat sich gezeigt, dass GLOH stabilere Features findet als SIFT, allerdings sprechen die Autoren von einem erheblich höheren Rechenaufwand.

In [10] wird eine Beschleunigung von SIFT vorgestellt, welche auf einer Approximation des Skalierungsraumes durch Integralbilder basiert, die als *Fast Approximated SIFT* bezeichnet wird.

4. Spurverfolgung

Bei der Spurverfolgung (engl. *Tracking*) wird die sich ändernde Spur eines sich bewegenden Objektes verfolgt. Die Berechnung einer Spur mittels Bildern basiert in der Ausnutzung der visuellen Merkmale. Es wird der Versatz zwischen den Merkmalen bestimmt und damit die Spur geschätzt.

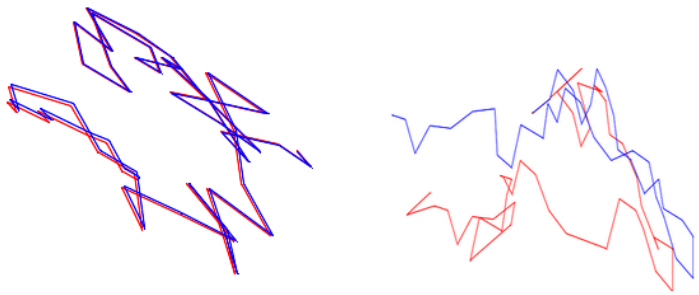
Beim Menschen spielt die optische Verfolgung von Objekten zur Ortsbestimmung eine enorme Rolle; wir orientieren uns an besonderen Punkten in der Welt, vornehmlich solche Punkte, die über unsere eigene Bewegung hinweg stabil bleiben. Ähnlich ist die Spurverfolgung beim Roboter. Mittels der Merkmalsextraktoren werden besondere Punkte zur Lokalisation (*Landmarken*) bestimmt, die dem Roboter eine relative Positionsbestimmung nach einer Eigenbewegung ermöglichen. Hierbei sei angemerkt, dass die Merkmale so geartet sein müssen, dass auch sich wenig unterscheidende Objekte unterschiedliche Merkmale aufweisen müssen. Auch müssen die Landmarken statisch für Objekte mit stabilen Positionen sein. Bei einer Null-Eigenbewegung (Stillstand), wird außerdem gefordert, dass die Merkmale statisch sind und an den erstmals gefundenen Positionen bleiben. Im Idealfall muss bei einem Stillstand die mit den visuellen Merkmalen geschätzte Bewegung Null betragen.

4.1. Problembeschreibung

Für aufeinander folgende Bilder müssen die berechneten Features mittels eines Abgleichs (engl. *Matching*) zueinander in Relation gebracht werden. Es werden Übereinstimmungen zwischen paarweise verschiedenen Features gesucht. Aus korrespondierenden Features wird der Versatz bestimmt und daraus die Bewegung geschätzt. Ziel ist es, das Problem der Zuordnungen bestmöglich im Hinblick auf das erwartete Bewegungsmodell zu lösen.

Probleme tauchen auf, wenn keine korrespondierenden Features gefunden werden können, oder gefundene Zuordnungen fehlerhaft in der Art sind, dass sich aus ihnen keine geeignete Bewegung berechnen lässt. Falsche Zuordnungen werden überwiegend bei gleichartigen Bildmerkmalen hergestellt, wie sich wiederholenden Mustern. Sich stark gleichende Objekte erzeugen ähnliche Features. Stehen diese Objekte im Bild voneinander entfernt, kann es sein, dass bei einer Bildbewegung das Matching zwar richtige Zuordnungen in Bezug auf die Features herstellt, aber nicht in Bezug auf die durch die Features jeweils repräsentierten Objekte.

In Abbildung 4.1 ist das Problem in Form zweier Trajektorien dargestellt. Beim Tracking wird versucht die tatsächliche Bewegung so exakt zu rekonstruieren, dass die errechnete Bewegung diese vollständig überdeckt. In Kombination mit Verfahren zur Vorhersage möglicher



(a) 2 Spuren, die sich nahezu überdecken.

(b) Stark abweichende Trajektorien.

Abbildung 4.1.: Beispiel Spurverfolgung. Die tatsächliche Spur (blau) in Relation zur geschätzten Spur (rot) des Trackings. Im Idealfall überdecken sich die Spuren vollständig.

Positionen, zum Beispiel einem Kalman Filter, lassen sich Korrekturen an der errechneten Spur vornehmen, so dass etwaige Ausreißer während des Trackings korrigiert werden. In dieser Arbeit werden nur die mittels der Features errechneten Positionen berücksichtigt, so dass über den Vergleich der Abweichungen der errechneten und tatsächlichen Spur, auf die Performance der Features für das Tracking geschlossen werden kann. Auch von Interesse ist, wie sich die Features im Subpixelbereich verhalten. Die betrachteten Feature Detektoren liefern die Punktpositionen derart, dass Verschiebungen auch im Subpixelbereich bestimmt werden können.

4.2. Bewegungsmodell

Für das Tracking ist wichtig zu wissen, in welchem Raum die Bewegung stattfindet. Im dreidimensionalen Raum mit sechs Freiheitsgraden sind Bewegungen in alle Richtungen möglich. Drei Freiheitsgrade für die Bewegung und drei für die Rotationen. Um solche Bewegungen zu beschreiben, bedarf es eines komplexeren Bewegungsmodells als es etwa in einem zweidimensionalen Raum nötig ist. Für die zweidimensionale Bewegung auf Bildern ist die Betrachtung eines einfachen Bewegungsmodells ausreichend, das nur die drei Freiheitsgrade zu Bewegungen in x, y-Richtungen und Rotation θ berücksichtigt. Die Spur von sich auf einer Ebene bewegenden Objekten lässt sich durch solch ein Bewegungsmodell abbilden.

Im Folgenden wird von einer planaren Bewegung mit Rotation in einer Ebene ausgegangen. Ein Roboter, zum Beispiel ein autonom navigierendes Luftschiff, bewegt sich in einer parallel zum Boden zweidimensionalen Ebene XY (siehe Abbildung 4.2).

Die Bewegung spiegelt sich in den Bildern als Versatz wider, bei der die Distanz und die Ausrichtung der Bilder zueinander die tatsächliche Bewegung ausdrücken. Die Abfolge der Bewegung auf Bildern lässt sich mit folgender Gleichung modellieren:

$$I(x + \xi, y + \eta, t + \tau) = I(x, y, t) \tag{4.1}$$

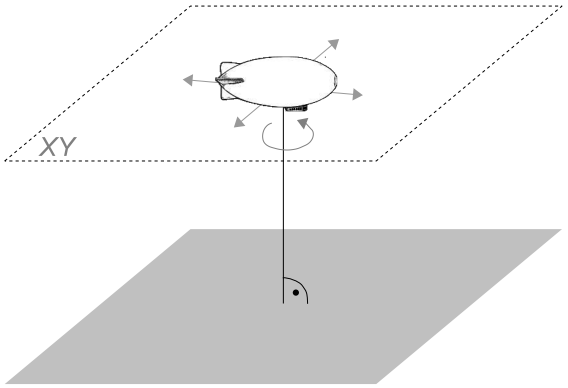


Abbildung 4.2.: Bewegung in planarer Ebene parallel zum Boden.

Diese Gleichung beschreibt den so genannten optischen Fluss. Mit diesem wird ein Vektorfeld bezeichnet, in dem die Bewegungen beobachteter Objekte auf Vektoren projiziert werden. Die Bewegungsrichtungen und Geschwindigkeiten der Objekte lassen sich an der Ausrichtung und der Länge der Vektoren ablesen. Offensichtlich ist diese Projektion kein genaues Abbild der realen Bewegung, da hier stetige Vorgänge diskretisiert werden. Wählt man aber die Frequenz der Bildaufnahmen genügend groß, so lassen sich auch Kurvenbewegungen durch den optischen Fluss approximieren. Mit Bezug auf Gleichung 4.1 lässt sich sagen, dass sich in einem Bild $I(x, y, t)$ nach der Zeit $t+\tau$ jeder Bildpunkt x, y um einen bestimmten Wert ξ in x-Richtung und η in y-Richtung bewegt hat. Die Bewegung $m = [\xi, \eta]^\top$ wird als Verschiebung (*displacement)* bezeichnet.

Unter der Berücksichtigung, dass die Kamera das sich bewegende Element ist, ist bei einer statischen Szene die Verschiebung aller Pixel gleich. In dynamischen Szenen unterscheiden sich lokale Felder an den Stellen, wo Objekte in der Szene sich bewegt hatten. Überträgt man dies auf Luftschiff, bei dem die Kamera senkrecht zum Boden ausgerichtet ist, und geht man weiterhin davon aus, dass am Boden keine dynamischen Objekte (Fußgänger, Autos, etc.) die Szene verändern, so zeigen die Verschiebungen der Bilder zueinander die Änderungen des Luftschiffes in der zum Boden parallelen Ebene XY. Im Folgenden wird von solch einem Modell ausgegangen. Änderungen der Bilder zueinander stellen Änderungen der Kamerapositionen dar. Bezogen auf Features bedeutet dies, dass die Positionen der Feature Punkte sich ändern. Die Verschiebung zweier Feature Punkte $\mathcal{F}_1$ und $\mathcal{F}_2$ ergibt sich aus der Distanz und der Lage der Features zueinander.

$$\mathbf{d}(\mathcal{F}_1, \mathcal{F}_2) = |\mathcal{F}_1 - \mathcal{F}_2| = dist(\mathcal{F}_1^{xy}, \mathcal{F}_2^{xy})$$

$$\theta(\mathcal{F}_1, \mathcal{F}_2) = atan2(\mathcal{F}_1^{xy}, \mathcal{F}_2^{xy})$$

$\mathcal{F}^{xy}$ meint die kartesischen Koordinaten x, y des Feature Punktes $\mathcal{F}$. Die euklidische Distanz $\mathbf{d}$ (siehe Abschnitt A.1) entspricht der Distanz zwischen zwei Feature Punkten, während die Lage θ der Winkel zweier Feature Punkte zueinander ist. Details zur Berechnung sind im Anhang zu finden.

Das verwendete Transformationsmodell tm (*transformation model*) enthält die Parameter Distanz $\mathbf{d}$ und Richtung θ

$$tm = [\mathbf{d}, \theta].$$

Die Verschiebung m kann man auch durch Projektion von tm in kartesische Koordinaten dargestellen

$$m = \begin{pmatrix} \xi \\ \eta \end{pmatrix} = \mathbf{d} \cdot \begin{pmatrix} cos\theta \\ sin\theta \end{pmatrix}.$$

Ausgehend von einer vorherigen Position $\vec{X}_{previous} = (x, y)$ ist die neue Position $\vec{X}_{new}$

$$\vec{X}_{new} = \vec{X}_{previous} + m$$

Sind die Kameraparameter bekannt, dann kann mittels 2D-3D Projektion die tatsächliche Bewegung in der realen Welt geschätzt werden [11].

4.3. Spurverfolgung mit visuellen Merkmalen

Die Spurverfolgung ergibt sich aus der Bewegung der visuellen Merkmale. Mittels eines Abgleichs (engl. *Matching*) werden Zuordnungen zwischen Features in zwei aufeinander folgenden Bildern hergestellt, anhand derer das bestmögliche Bewegungsmodell bestimmt werden kann. Da die tatsächliche Bewegung nicht bekannt ist, muss die Information aus den berechneten Merkmalen geschätzt werden.

4.3.1. Matching

Das Matching von Feature Punkten lässt sich auf das in der Computergrafik bekannte Problem der Bildregistrierung zurückführen. Bei der Bildregistrierung müssen zwei oder mehrere Bilder derselben Szene mit einem Referenzbild bestmöglich in Übereinstimmung gebracht werden. Bei den Feature-Mengen zweier Bilder muss das Transformationsmodell gefunden werden, das die Menge der Feature Punkte im ersten Bild auf die Menge der Feature Punkte im zweiten Bild abbildet. Es werden Punktkorrespondenzen gesucht, aus denen auf die ausgeführte Transformation geschlossen werden kann.

Der Vergleich zweier Feature Punkte wird über den Feature Deskriptor bewerkstelligt. Dazu werden mittels einer gewichteten Nächste-Nachbar-Klassifikation [19, 21] die Features anhand der Vektorendistanz ihrer Deskriptorvektoren klassifiziert. Die Vektorendistanz berechnet sich

über die euklidische Distanz zwischen jeder Komponente der zwei Deskriptorvektoren. Zwei Features korrespondieren, wenn ihre Deskriptordistanz kleiner ist als 0.7mal die Distanz des zweiten nächsten Nachbars.

Für jedes Feature $\mathcal{F}^A$ aus einer Feature-Menge A wird ein bestmöglich passendes Feature $\mathcal{F}^B$ aus einer zweiten Feature-Menge B gewählt

$$matching: \ \mathcal{F}^A \rightarrow \mathcal{F}^B.$$

Die Menge A stellt die Menge der Feature Punkte aus einem vorherigen Bild dar, während die Menge B die Features des aktuellen Bildes sind. Wird die Funktion $matching$ auf alle Features im aktuellen Bild angewandt, dann ist das Ergebnis eine Menge $\mathcal{M}$, die eine Zuordnungsliste korrespondierender Punkte enthält.

$$\mathcal{M} = \left\{(\mathcal{F}^A, \mathcal{F}^B) \in \{A \times B\} \,|\mathcal{F}^A \rightarrow F^B\right\}$$

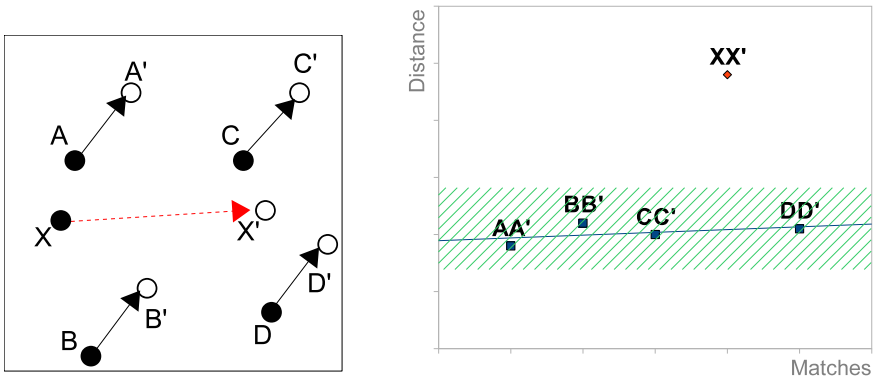
Ist $\mathcal{M}$ leer, so wurden keine übereinstimmenden Punkte zwischen den zwei Bildern gefunden und es muss mit alternativen Ansätzen Punktzuordnungen gesucht werden. Eine mögliche Heuristik ist, die Merkmale vorangegangener Bilder zu berücksichtigen. Dies setzt voraus, dass eine Historie mit Merkmalen bereit gehalten wird. Diese kann in chronologischer Reihenfolge die früheren Features enthalten, oder nach einem anderen Kriterium ausgewählte Features, etwa solche, die sich als besonders stabil herausgestellt hatten (*stabile Landmarken*). Se *et al.* [29] bestimmen stabile Landmarken indem für jedes Feature ein Gütemaß eingeführt wird, das die Gültigkeit eines Features anhand dessen Trefferquote (*hits and misses*) angibt.

4.3.2. Mittelwertklassifikation

Eine Möglichkeit, die Verschiebung der Feature Punkte in zwei aufeinanderfolgenden Bildern zu ermitteln, ist, anhand einer Mittelwertklassifikation nur solche Punkte zu berücksichtigten, die in einem Bereich um einen „stabilen Mittelwert“ angesiedelt sind, und Ausreißer auszusortieren. Das Bewegungsmodell ergibt sich dann über den Durchschnittswert der berücksichtigten Punkte. Der stabile Mittlwert folgt nach einer endlichen Anzahl Iterationen, nachdem die Varianz aller Punkte in einem gegebenen Toleranzbereich bleibt.

In Abbildung 4.3 ist die Idee dargestellt. Bei einer Menge von Punktkorrespondenzen können sich Ausreißer darunter befinden, die eine stark abweichende Distanz oder Orientierung haben (gestrichelte Linie). Diese Punkte werden aussortiert, und es werden nur die Menge der Punkte im schraffierten Bereich für die Berechnung des Bewegungsmodells berücksichtigt. Der Algorithmus dazu wird im Folgenden erklärt.

Iterativ wird in jedem Iterationsschritt i für jede Zuordnung aus der Punktmenge $\mathcal{M}$ (siehe Abschnitt 4.3.1) das zugehörige Transformationsmodell tm_i (siehe Abschnitt 4.2) bestimmt. Im ersten Schritt wird auf der Menge der Transformationsmodelle der Mittelwert $\bar{tm}_i$ berechnet



(a) Verschiebungen mit Ausreißer XX'.

(b) Distanzen zu den Verschiebungen. Die Punkte im schraffierten Bereich sind in der Inliers-Menge.

Abbildung 4.3.: Idee zur Mittelwertklassifikation.

$$\bar{tm}_i = [\bar{\mathbf{d}}_i, \bar{\theta}_i] \tag{4.2}$$

$$= \left[\frac{1}{n}\sum_{i=1}^{n}\mathbf{d}_i, \frac{1}{n}\sum_{i=1}^{n}\theta_i\right] \qquad i,n \in \mathbb{N}, n \leq |\mathcal{M}| \tag{4.3}$$

Zur Berechnung der Winkeldurchschnitte $\bar{\mathbf{d}}_i$ wird eine Projektion von Polarkoordinaten in kartesische Koordinaten vorgenommen und darüber gemittelt (siehe Abschnitt A.3). n ist die Anzahl korrespondierender Merkmale aus der Menge $\mathcal{M}$.

Ziel ist es diejenigen Punkte auszufiltern, die pro Iterationsschritt eine große Varianz aufweisen. Dazu werden alle Punkte, die außerhalb eines Toleranzbereiches $\pm\Delta$ um den Mittelwert liegen, aussortiert. Die übrigen Punkte ergeben die Menge der $inliers$

$$inliers_i = \{\mathcal{F} \in \mathcal{M} | \, \mathbf{d}_i < \mathbf{d}_i \pm \Delta \wedge \theta_i < \theta_i \pm \Delta\} \, .$$

Die Menge $inliers_i$ im Iterationsschritt i wird als Basis für die Bestimmung des Mittelwerts tm_{i+1} im nächsten Schritt herangezogen. Der Algorithmus bricht ab, wenn die Änderungen des Mittelwertes unterhalb eines Schwellenwertes liegen. Das Resultat ist das Transformationsmodell

Die Mittelwertklassifikation ist anfällig für Fehler, bei denen die Varianz vieler Werte $\mathbf{d}, \theta$ sehr groß ist. Gibt es genügend viele Ausreißer, deren Varianzen schrittweise abfallen, so wird die Filterung über den Mittelwert durch diese verfälscht. Große Varianz entsteht, wenn beim Matching weit entfernte Feature Punkte einander zugeordnet werden. Dies kann der Fall sein, wenn die Bilder sich stark ähnelnde Merkmale an gegenüberliegenden Bildstellen enthalten, oder wenn

sich in einer Szene Änderungen durch dynamische Objekte ergeben haben. In so einem Fall sind die Distanzen oder die Rotationen zwischen einigen korrespondierenden Merkmalen groß

4.3.3. Random Sample Consensus (*RANSAC*)

Für eine robustere Bestimmung eines Bewegungsmodell eignet sich der Algorithmus *random sample consensus* [9]. Der Algorithmus ermittelt ein passendes Modell aus einer gegebenen Menge von Datenpunkten, die Ausreißer enhalten kann. Zur Berechnung des Modells werden nur diejenigen Punkte berücksichtigt, die ein bestmögliches Modell liefern.

Beim Matching der Features (siehe Abschnitt 4.3.1) kann es passieren, dass Punkte aus einer Menge A wegen großer Ähnlichkeit der Feature Deskriptoren auf Punkte in Menge B so zugeordnet werden, dass Ausreißer dabei sind (siehe Abbildung 4.3a). Diese Ausreißer dürfen bei der Bestimmung des Bewegungsmodells nicht berücksichtigt werden.

Der RANSAC Algorithmus wählt aus der Menge der Punktkorrespondenzen eine zufällige Anzahl Punkte aus und bestimmt für diese ein mögliches Bewegungsmodell. Dieses Modell wird nun genutzt, um die Ursprungsmenge A auf die Zielmenge B zu transformieren. Ist das gewählte Modell gut, liegen nach der Transformation die Punktmengen sehr nah aufeinander. Der Algorithmus wird abgebrochen, wenn die Überdeckung ausreichend ist, ansonsten werden neue Kandidaten gezogen und der Algorithmus beginnt von vorne.

Ein einfaches Beispiel für die Anwendung von RANSAC ist die Bestimmung einer Regressionsgeraden für eine Punktemenge mit Ausreißern (*Outliers*). In Abbildung 4.4) ist dies dargestellt. *Inliers* sind die Punkte, die durch eine Gerade approximiert werden können, *Outliers* dagegen liegen verstreut weit entfernt von dieser Geraden. Die Anwendung der Methode der kleinsten Fehlerquadrate liefert kein sinnvolles Ergebnis, da hier alle Punkte, auch die Ausreißer, in die Berechnung miteinfließen. Werden stattdessen nur die Inliers berücksichtigt, wie es bei RANSAC sein kann, so ist das Ergebnis ein gute Approximation der gesuchten Geraden.

Das Ergebnis von RANSAC ist davon abhängig, ob die zufällig gewählten Punkte geeignete Modellkandidaten erzeugt haben. Welche Parameter von RANSAC hierfür von Bedeutung sind und die Beschreibung zum Algorithmus wird im Folgenden aufgezeigt.

RANSAC Algorithmus

Als Eingabedaten für den RANSAC Algorithmus wird die Menge der Punktepaare $\mathcal{D}$ nach dem Matching benutzt. Ein Datenpunkt enthält die Zuordnung der kartesischen Koordinaten x, y für jedes Punktepaar:

$$(x_1, y_1) \rightarrow (x_2, y_2)$$

Der RANSAC Algorithmus angewandt zur Suche des bestmöglichen Transformationsmodells tm, ist wie folgt:

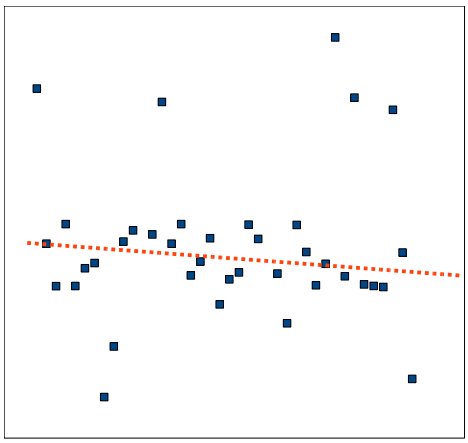


Abbildung 4.4.: Beispiel zu RANSAC. Approximation einer Regressionsgeraden.

1. Auswahl von n zufälligen Punktepaaren, die eine *Inliers*-Kandidatenmenge $\mathcal{K}$ ergeben. Im translatorischen Bewegungsmodell ist $n = 1$ hinreichend für die Bestimmung eines gültigen Transformationsmodells.

2. Berechnung eines hypothetischen Transformationsmodells tm_0. Im weiteren versucht der Algorithmus diese Hypothese zu verifizieren.

3. Selektion der Datenpunkte aus den Eingabedaten, die nicht in der Menge $\mathcal{K}$ sind, also die Menge $\mathcal{I} = \mathcal{D} \backslash \mathcal{K}$. Auf jeden Punkt $\mathbf{p}$ in der Menge I wird nun die Hypothese tm_0 angewandt und der Fehler t' bestimmt. Ist $t' < t$, für einen gegebenen Schwellenwert t, so wird der Punkt zu einer Konsens-Menge $\mathcal{C}$ (*consensus set*) hinzugefügt. Der Parameter t gibt an, wann ein Punkt das Modell erfüllt. Alle Punkte, die von der aktuellen Hypothese eine große Abweichung aufweisen, werden nicht berücksichtigt. Im benutzten Bewegungsmodell kann t als die Distanz in Pixel angegeben werden; es wurde $t = 30$ Pixel gesetzt.

4. Ist die Anzahl der Punkte in der Konsens-Menge $\mathcal{C}$ größer als ein gegebener Parameter d, so bedeutet dies, dass eventuell bereits ein gutes Modell gefunden wurde. Dieses wird nun verifiziert. Basierend auf den Punkten der Menge $\mathcal{C}$, die sich aus dem Transformationsmodell tm_0 ergab, wird eine neue Hypothese $\hat{tm}_0$ bestimmt. Für $\hat{tm}_0$ wird nun der Gesamtfehler $\hat{\epsilon}$ auf den Punkten in $\mathcal{C}$ berechnet. $\hat{\epsilon}$ kann als Gütemaß betrachtet werden, wie gut die Hypothese aus 2. ist.
   Ist der aktuelle Gesamtfehler $\hat{\epsilon}$ kleiner als der Gesamtfehler vorheriger Iterationsschritte, so wurde mit $\hat{tm}_0$ ein besseres Modell gefunden. Das Modell tm, das der Algorithmus am Ende liefern wird, auf die aktuelle Hypothese gesetzt: $tm = \hat{tm}_0$.

5. Iteriere k-mal von 1.-4.

Am Ende der Iterationen liefert der Algorithmus dasjenige Transformationsmodell, das die Punktzuordnungen bestmöglich abbildet. Der Algorithmus hängt im Wesentlichen von den drei Parametern k, t, d ab. Sie wurden empirisch bestimmt und sind $k = 5$, $t = 30$, $d = 10$. Im Gegensatz zur Mittelwertklassifikation (siehe Abschnitt 4.3.2), lieferte der RANSAC Algorithmus robuste Ergebnisse.

5. Details zur Implementierung

Um die verschiedenen Merkmalsextraktoren zu vergleichen wurde ein System in C++ implementiert, das anhand von Bilddatensequenzen und Referenzbewegungsinformationen eine Spurverfolgung vornimmt, die berechnete Spur mit der tatsächlichen vergleicht und die Abweichungen protokolliert. Im Folgenden wird dieses System als *Feature Quality Analysis (FQA)* bezeichnet.

Der Eingabedatenstrom wird zerteilt in die Bilddaten und die zugehörigen Referenzinformationen der Transformationen. Auf den Bilddaten werden die Features für einen Feature Detektor berechnet und mittels Matching Punktkorrespondenzen hergestellt. Anhand der Punktkorrespondenzen kann ein Bewegungsmodell geschätzt werden und der Fehler zu den Referenzdaten ermittelt werden.

Das Schema der Prozessabfolge ist in Abbildung 5.1 dargestellt und wird im Folgenden erklärt.

5.1. Eingabedaten

Die zu verarbeitenden Eingabedaten bestehen aus

- den Originalbildern (Kamerabilder) und
- den Referenztransformationen der tatsächlichen Spur (gemessene Bewegung/Odometrie).

Die Bilddaten werden keiner Vorverarbeitung unterzogen, die den Vergleich der Detektoren zu Gunsten eines Verfahrens beeinflussen könnte. Die Referenztransformationen geben die tatsächlich ausgeführte Bewegung an. Sie stellen eine Form von Bodenwahrheit dar, mit der im Verlauf der Anwendung die Abweichung der berechneten Spur ermittelt wird.

Die Bilddatenströme können Bilder mit nahezu beliebiger Kompression enthalten. Einzige Bedingung ist, dass die Bilddaten durch die Bildverarbeitungsbibliothek ImageMagick[1] gelesen werden können. Die Bildsequenzen als Video mittels gängiger Video-Codecs (z.B. MPEG2) zu kodieren ist im Hinblick auf das Tracking wenig sinnvoll, da die bei der Kompression entstehenden Artefakte die Feature Extraktion nachhaltig stören würden. Bei den Bildern wurde darauf geachtet, dass die Tests des Systems mit Eingabebildern erfolgten, die mit ververlustfreien Kompressionen kodiert wurden; das PNG-Format wurde in der Regel bevorzugt. Auf Bilder im JPEG-Kompressionsverfahren wurde weitgehend verzichtet, zeigen doch Mikolajczyk und Schmid [22], dass die Feature Detektoren empfindlich auf Kompressionsartefakte reagieren.

[1] http://www.imagemagick.org

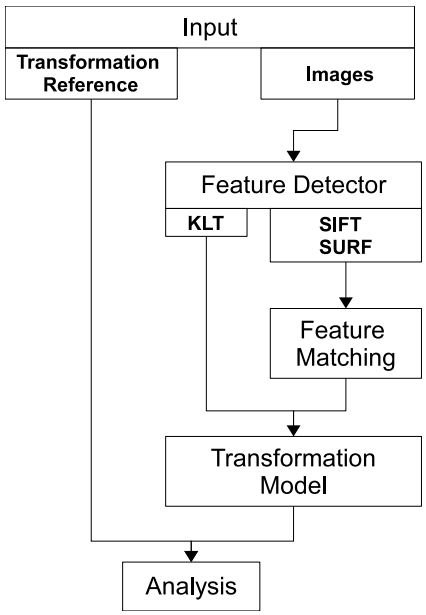


Abbildung 5.1.: Schema Prozessabfolge des implementierten Systems *FeatureQualityAnalysis*.

5.2. Merkmalsextraktion

Die Eingabebilddaten werden in Graustufenbilder im PGM Format umgerechnet und den Feature Detektoren zugeführt. Dabei muss zwischen der Arbeitsweise mit dem KLT Tracker und der mit SIFT und SURF unterschieden werden. Während der KLT Tracker das Tracking selbst vornimmt, muss dies bei SIFT und SURF über eigene Verfahren bewerkstelligt werden.

5.2.1. KLT

Für den KLT Tracker werden die Bilder des Eingabedatenstroms in das PGM Format konvertiert, das nur noch Graustufenwerte enthählt. Als Implementierung des KLT Trackers wurde die Version von Stan Birchfield[2] gewählt. Das Vorgehen des Trackers wird im Folgenden beschrieben.

Der Tracker wird mit dem ersten Bild initialisiert. Aus den Bildern wählt der Tracker die n besten Feature Punkte aus (siehe Abschnitt 3.2.1), die dann für die weitere Verfolgung zwischengespeichert werden. Es werden eine Anzahl guter Feature Punkte mittels der Gradientenwerte extrahiert, wie in Abschnitt 3.2.1 beschrieben. Die Güte eines Feature Punktes wird dadurch bestimmt, wie gut sich dieser tracken lässt. Es wird der kleinste Eigenwert der 2×2 Gradientenmatrix für alle Bildausschnitte betrachtet und daraus auf die Güte des Feature Punktes geschlos-

[2] http://www.ces.clemson.edu/~stb/klt/

sen. Die Eigenwerte aller Punkte werden der Güte nach sortiert und diejenigen ausgewählt, deren kleinster Eigenwert über einem Schwellwert liegt. Als weiteres Filterkriterium werden nur die Feature Punkte übernommen, die einen Mindestabstand zu anderen Feature Punkten haben. Damit wird sichergestellt, dass nicht nur eine lokale Punktwolke betrachtet, sondern eine gute Abdeckung des Bildes erreicht wird.

Da die KLT Bibliothek keinen Feature Deskriptor liefert, kann hier kein Matching durch das implementierte System (FQA) stattfinden. Das Matching der Features läuft im KLT Tracker intern ab. Rückgabewert ist nur eine Information zum Tracking-Status in Form einer Konstanten, die angibt ob und wie das Matching für ein Feature in der internen KLT Feature-Liste erfolgreich war. Der Tracker bietet die Option verlorene Features durch neue zu ersetzen; diese Option wird genutzt.

5.2.2. SIFT und SURF

Da bei der SIFT Version von Lowe nur eine Binär-Version zur Verfügung steht, wurde zusätzlich die Implementierung SIFT++[3] von Andrea Vivaldi berücksichtigt. Für den SURF Detektor existiert bislang nur die original Implementierung der Autoren von SURF[4]. Beide Detektoren akzeptieren als Eingabedaten Grauwertbilder im PGM-Format.

Die Feature Detektoren SIFT und SURF liefern ähnliche Datenstrukturen für die Features. Benutzt werden die Koordinaten x, y, die Feature-Rotationen θ und der Deskriptorvektor. Letzterer wird zum Matching genutzt. Zur Verfolgung der Features bei Bildbewegung müssen die Punktzuordnungen auf den Feature-Mengen von zwei aufeinander folgenden Bildern mittels Matching erstellt werden (siehe Abschnitt 4.3.1). Dazu werden die Feature Punkte anhand ihrer Deskriptorvektoren zugeordnet. Die Deskriptoren sind Vektoren mit reellen Zahlen, die ein Feature identifizieren. Mit einer Nächsten-Nachbar-Klassifikation werden die Zuordnungen der Feature Punkte bestimmt. Für jeden Feature Deskriptor aus einer Feature Menge wird ein korrespondierender Punkt in einer zweiten Menge über die Vektordistanz gesucht. Die so erzeugte Menge von Zuordnungen ist die Menge der *Matches*.

Ein Problem ist, wenn die Feature Mengen des aktuellen und des vorherigen Bildes eine leere Menge von Matches ergibt. Um das Matching robuster zu gestalten wurde eine Historie für die Feature-Mengen implementiert. Wird zwischen dem aktuellen Bild und seinem Vorgänger kein Match gefunden, werden die Feature-Mengen vorheriger Bilder berücksichtigt, und es wird mit diesen versucht eine Menge mit Punktzuordnungen zu finden.

5.3. Bestimmung der Bildänderung

Die Bildänderungen spiegeln sich im Versatz der zugeordneten Feature Punkte wider. Anhand der Punktkorrespondenzen wird das bestmögliche Transformationsmodell ermittelt, das den Ver-

[3] http://vision.ucla.edu/~vedaldi/code/siftpp/siftpp.html
[4] http://www.vision.ee.ethz.ch/~surf/

satz beschreibt. Versuche das Transformationsmodell mit einer gewichteten Nächste-Nachbar-Klassifikation oder einer Klassifikation über den Mittelwert zu finden, hatten keine guten Ergebnisse erzielt. Stattdessen wurde der RANSAC Algorithmus (siehe Abschnitt 4.3.3) implementiert. Dieser liefert das Transformationsmodell tm:

$$tm = [\mathbf{d}_{trans}, \theta_{trans}]$$

$\mathbf{d}_{trans}$ ist die Pixel-Distanz der Verschiebung zwischen den Feature-Mengen zweier Bilder, θ_{trans} ist die Richtung der Verschiebung. Mit den Winkelfunktionen Sinus und Kosinus lassen sich die Bewegungsrichtungen x_{trans}, y_{trans} berechnen.

$$\begin{pmatrix} x_{trans} \\ y_{trans} \end{pmatrix} = \mathbf{d}_{trans} \cdot \begin{pmatrix} cos\theta_{trans} \\ sin\theta_{trans} \end{pmatrix}$$

Das so berechnete Transformationsmodell tm wird für den Vergleich mit den Refernzdaten (siehe Abschnitt 5.1) eingesetzt.

5.4. FQA - Feature Analyse

Das basierend auf den Features und RANSAC geschätzte Bewegungsmodell wird mit den Referenzdaten verglichen. Bewertungskriterium ist die Abweichung von den Referenzdaten. Sie ergibt sich aus der Differenz zwischen den Bewegungen der Referenzdaten und der geschätzten Bewegung:

$$\epsilon = \begin{pmatrix} x_{err} \\ y_{err} \\ \theta_{err} \end{pmatrix} = \left| \begin{pmatrix} x_{ref} \\ y_{ref} \\ \theta_{ref} \end{pmatrix} - \begin{pmatrix} x_{trans} \\ y_{trans} \\ \theta_{trans} \end{pmatrix} \right|$$

$x_{ref}, y_{ref}, \theta_{ref}$ sind die x, y Koordinaten und die Rotation θ aus den Referenzdaten $x_{trans}, y_{trans}, \theta_{trans}$ ist das Transformationsmodell, das mit Hilfe des RANSAC Algorithmus (siehe Abschnitt 4.3.3) bestimmt wurde.

Neben der Zeit, die für die Berechnung der Features benötigt wird und den statistischen Daten zu der Anzahl der Features und den Übereinstimmungen beim Matching, wird die Abweichung der berechneten Transformationen zu den Referenzdaten für die spätere Auswertung protokolliert und zur Analyse bereitgestellt. Die berechneten Bewegungen und Rotationen für jeden Schritt in der Bildsequenz werden in ein Bild übertragen, das am Ende die Trajektorie zeigt.

6. Experimente

Im Folgenden werden die Experimente zum Vergleich der Feature Detektoren sowohl auf künstlich simulierten als auch auf realen Bildsequenzen beschrieben. Zur Extraktion der Features wurden die Merkmalsextraktoren *KLT*, *SIFT* und *SURF* mit ihren Standardparametern auf die Eingabedaten angewandt. Auf den extrahierten Features wurde, wie in Kapitel 5.4 beschrieben, ein Matching und Tracking ausgeführt und die Abweichungen von den Referenzdaten ermittelt. Die im ersten Schritt ausgeführte Extraktion der Features war wie folgt:

- Mit dem KLT Tracker wurden die 100 besten Feature Punkte bestimmt und durch den KLT Tracker in den nachfolgenden Bildern verfolgt. Verloren gegangene Features, also die, die nicht mehr auffindbar waren, ersetzte der Tracker durch neue Feature Punkte. Die Anzahl der verloren gegangener Feature Punkte wurde protokolliert und bei der Auswertung der Statistiken berücksichtigt.
- SIFT-Features wurden sowohl mit der Bibliothek von Andrea Vivaldi als auch mit der Original-Implementierung von Lowe erzeugt (siehe Abschnitt 5.2.2).
- SURF-Features wurden mit der Bibliothek der Autoren von SURF erzeugt. Neben der nativen Version mit 64-stelligem Deskriptorvektor wurden auch die Varianten SURF-128, die einen 128-stelligen Deskriptorvektor berechnet, und U-SURF (siehe Abschnitt 3.2.3) getestet.

Alle Detektoren arbeiteten mit den gleichen Eingabedaten. Wie die Bildsequenzen konstruiert wurden wird im folgenden Abschnitt erläutert. Ein Abbruch des Vorgangs wurde eingeleitet, wenn entweder keine Feature Punkte gefunden wurden, oder das Matching keine Übereinstimmungen lieferte, selbst wenn die Feature Mengen der drei letzten Bilder berücksichtigt wurden. Der Abbruch ist notwendig, weil zum Vergleich der Feature Detektoren nur diejenigen zugelassen werden, die auf der gesamten Bildsequenz von sich aus gültige Daten liefern. Weiterverarbeitende Schritte oder erweiterte Filterungen und Korrekturen (zum Beispiel mit einem Kalman Filter) werden nicht durchgeführt, weil einzig die nativen Daten der Feature Detektoren interessieren.

6.1. Eingabedaten

Die Merkmalsextraktoren wurden auf Bildsequenzen getestet. Berücksichtigt wurden künstlich erzeugte Bildsequenzen, die Bildtransformationen simulieren, und die Bildsequenzen von Videos mit realen Bewegungen. Die Refernzdaten der simulierten Bildtransformationen stellen

hier eine Form der Bodenwahrheit (*ground truth*) dar. Diese Daten werden zur Bestimmung der Abweichungsfehler herangezogen.

Für die künstlich erzeugten Bildsequenzen wurden Draufsichtaufnahmen von unterschiedlichen Böden als Basis genommen. Die Entscheidung zu Draufsichtaufnahmen begründete sich durch das verwendete, translatorische Bewegungsmodell. Mit einer handelsüblichen Digitalkamera (Canon Ixus 70*)* wurden Fotos mit unterschiedlichen Draufsichtaufnahmen gemacht (siehe Abbildung 6.1):

- **Asphalt**: Aufgenommen aus 2,20 Metern Höhe. Typisch für Asphalt ist, dass er nie homogen ist und die Detailstrukturen sich wegen ihrer Zufälligkeit nicht wiederholen. Weiter fügen äußere Einflüsse dem Asphalt Veränderungen zu, wie Löcher, Farb- und Reifenspuren usw.
- **Gras mit Pflastersteinen**: Aufgenommen aus ca. 4 Meter Höhe. Die natürliche, zufällige Struktur eines Rasens erzeugt, neben dem Muster des Grases an sich, bei entsprechendem Sonnenstand Lichter und Schatten auf den Grashalmen. Weiterhin ist am unteren Bildrand der Rand eines Pflastersteinbodens mit regelmäßigem Muster enthalten.
- **Gras mit Erdboden und Pflastersteinen**: Aufgenommen aus ca. 4 Meter Höhe. Ähnlich dem Foto davor ist hier am oberen Bildrand zusätzlich ein Erdboden mit Geröll abgebildet.
- **Pflastersteine mit regelmäßiger Struktur**: Aufgenommen aus ca. 2 Meter Höhe. Boden mit regelmäßig angeordneten Pflastersteinen mit rechteckiger Form.
- **Holzdielenboden**: Aufgenommen aus ca. 2,50 Meter Höhe. Längliche Holzlatten mit Maserung, Astlöchern und Gebrauchsspuren (Kratzer, Markierungen, u.a.).
- **Wohnzimmer Draufsicht**: Aufgenommen aus 2,40 Metern Höhe, unter anderem mit Teppich, Schreibtisch, Mülleimer und Laminat-Holzboden. Trotz Unebenheiten hat der Teppich eine nahezu homogene Struktur, was eine Herausforderung an die Features darstellt.

Für alle Fotos gilt, dass sie bei nahezu idealer horizontaler Ausrichtung der Kamera gemacht wurden. Außenaufnahmen wurden bei Sonnenlicht aufgenommen, die Fotos in Innenräumen (Holzdielen- und Wohnzimmerboden) wurden jeweils in Höhe der Decke ohne Blitz gemacht. Alle Bilder haben eine Auflösung von 3072x2304 Pixeln und sind im JPEG-Format kodiert.

6.2. Künstlich erzeugte Bildsequenzen

Mit einem Eingabedatengenerator wurden zur Simulation künstliche Transformationen auf den Bilddaten ausgeführt. Da die Transformationen künstlich auf einem Ausgangsbild erzeugt werden, ist die Einschränkung auf Draufsichtaufnahmen hier nicht zwingend. Es kann ein Foto

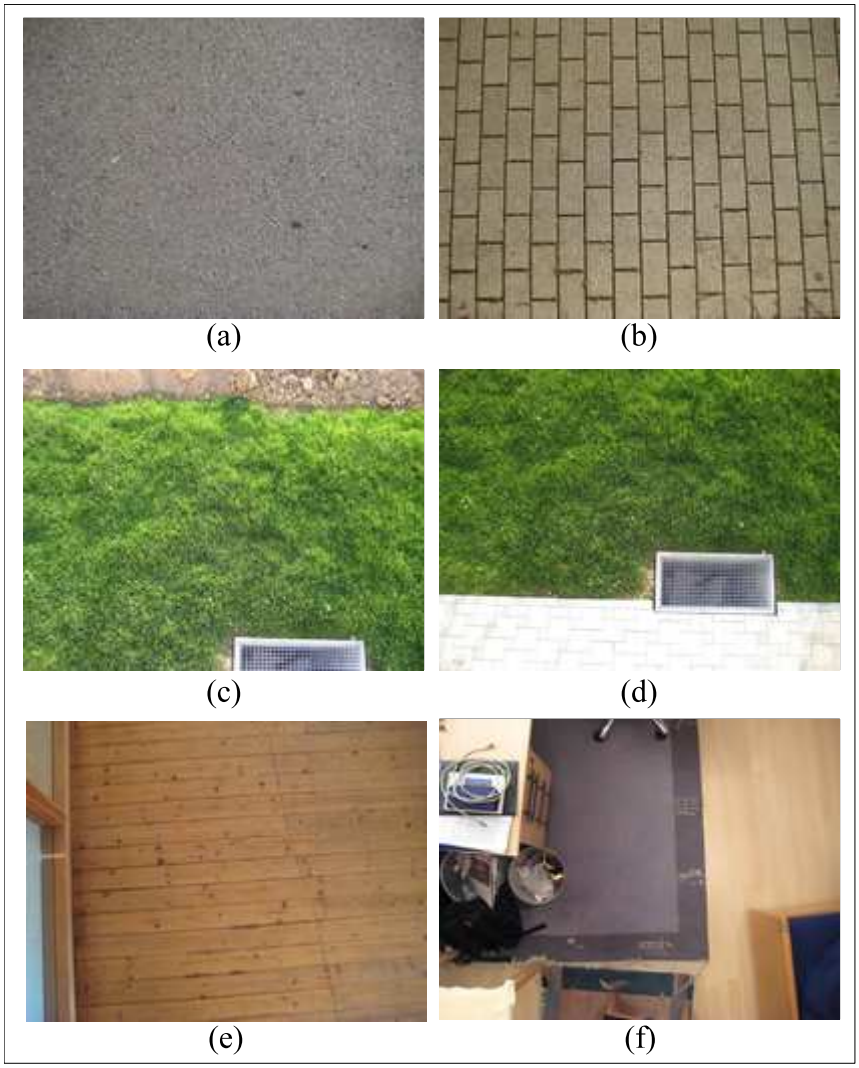


Abbildung 6.1.: Fotos von Böden. (a) Asphalt, (b) Pflastersteine, (c) Rasen mit Erdboden, (d) Rasen mit Pflastersteinboden, (e) Holzdielenboden, (f) Wohnzimmer

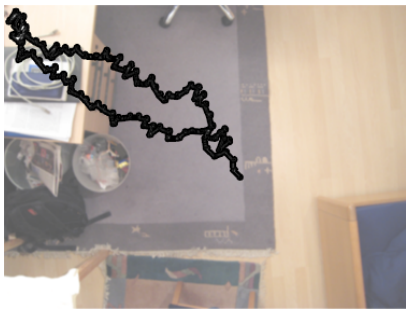


Abbildung 6.2.: Mögliche Trajektorie für eine zufällige Abfolge von Translationen. Startpunkt und Endpunkt sind in der Bildmitte. Die Bewegung ist so, dass sie sich kurvenförmig bis zum Bildrand und wieder zurück bewegt.

beliebiger Perspektive oder ein künstlich erzeugtes Testbild als Basis für die Erzeugung einer Bildsequenz genommen werden.

Die Bildtransformationen wurden nicht auf das gesamte Bild angewandt, sondern auf ein Fenster der Größe 320x240 Pixel. Als Position des Fensters wurde in den vorgestellten Ergebnissen die Bildmitte gewählt, die bei den Fotos aus Abbildung 6.1 die für die Feature Detekoren schwierigeren Stellen darstellen. In der Aufnahme 6.1f zum Beispiel, wurde der Teppich als Bildfenster ausgewählt.Es wurde darauf geachtet, dass die Bilder für die Bildsequenzen mit einem verlustfreien Kompressionsverfahren kodiert wurden (PNG), so dass keine Störung der Feature Extraktion durch Kompressionsartefakte entstand.

Die Menge der Bildtransformationen bestand aus drei Operationen:

- Keine Aktion (*NoAction)*: Keine Bildtransformation; das Bild bleibt unverändert.
- Rauschen *(Blur)*: Das Bild wird mit einem Gauß-Kernel gefaltet. Für den Radius der Filtermaske wurde 5 gewählt.
- Zufällige Helligkeitsänderung *(BrightnessRandom),* im Bereich $[0, 5; 1]$. wobei bei 0 jeder Bildpixel schwarz ist, und bei 1 das Originalbild ausgegeben wird.
- Zufällige Translation *(TranslateRandom)*: Das Bildfenster wird in eine zufällige Richtung und Entfernung verschoben. Bei der Generierung einer Bildsequenz wurde darauf geachtet, dass die Bewegung kurvenförmig verläuft und wieder zum Richtung Startpunkt zurückkehrt, und dass die Bewegung Richtung Bildrand verläuft. In Abbildung 6.2 ist eine mögliche Trajektorie dargestellt.

Der Eingabedatengenerator erzeugte für ein gegebenes Bild eine künstliche Abfolge von Transformationen und speicherte die Kombination aus Bildzuständen und Transformationsdaten in einer Protokolldatei. Die Daten in der Protokolldatei enthalten die für den Vergleich benötigten Referenzdaten.

6.2.1. Stillstand

Für den Stillstand wurden solche Bildsequenzen berücksichtigt, bei denen das Bild stillsteht und nur durch die Operationen Rauschen und zufällige Helligkeitsänderung verändert wird. Davon ausgehend wird die Menge der untersuchten Bildsequenzen durch vier Operationen charakterisiert:

Keine Aktion
Rauschen
Helligkeitsänderung
Rauschen und Helligkeitsänderung

Die Ergebnisse des 2D-Trackings und die ermittelten Fehlerabweichungen werden im Folgenden getrennt behandelt. Eine tabellarische Übersicht zu der Anzahl gefundener und zugeordneter Features, der Detektor Zeit und den Abweichungsfehlern für x, y, θ ist im Anhang zu finden (siehe Abbildung B.1). Im Folgenden wird auf diese Tabelle Bezug genommen.

- **Keine Aktion**: Erwartungsgemäß beträgt die Fehlerabweichung Null für jeden Feature Detektor auf den stillstenden Ausgangsbildern. Beim Matching werden nahezu alle Features zugeordnet (Mittelwert 98,91%). Dass nicht alle Features korrekt zugeordnet werden liegt daran, dass manche Feature Vektoren sich zu stark ähneln und bei der gewichteten Nächste-Nachbar-Klassifikation (siehe 4.3.1) eliminiert werden.

- **Rauschen**: Hier zeigte sich erstmals die Sensibilität der KLT auf Bildstörungen. Für die Innenaufnahmen schafft es der KLT Tracker nicht, Features zu finden, die er weiterverfolgen kann. Sobald in der Bildsequenz Rauschen auftrat brach das Tracking bei KLT ab. Bei SIFT bewegt sich der mittlere Fehler über die gemittelte Distanz in einem Bereich von 0 bis 0,43 Pixel bei der freien SIFT-Implementierung, und zwischen 0 und 0,26 Pixeln bei der Lowe Implementierung. Die Lowe Implementierung lieferte in einem Fall aber kein Ergebnis, sondern arbeitete nur auf fünf der sechs Bilder zuverlässig. Für das Bildfenster Teppichboden vom Wohnzimmer-Bild brach das Programm nach 40 Schritten ab, als mehrere verrauschte Bilder kamen. In Kombination mit dem nahezu homogenen Teppich konnten keine *interest points* mehr extrahiert werden. Da die interne Arbeitsweise der original Lowe Implementierung nicht bekannt ist (liegt nur im Binär-Format vor), kann für diesen Effekt kein Grund angegeben werden. Es bleibt das Resultat, dass die Lowe Implementierung von SIFT sensibel auf stark verrauschte Bilder mit wenig Textur reagieren kann, während die freie SIFT Implementierung und SURF davon unbeeindruckt bleiben. Im Mittel konnten 64,35% aller Features zugeordnet werden. Die Distanzabweichungen variierten zwischen 0,18 Pixel bei Lowe-SIFT und 1,16 Pixel bei SURF-128. Bei SURF kam die Variante U-SURF mit den verrauschten Bildern am besten zurecht (1,01 Pixel Abweichung).

Fehler	SIFT	SIFT Lowe	SURF	SURF-128	U-SURF
Distanz	0,16	0,26	0,74	0,72	0,65
Orientierung	-0,05	-0,11	-0,06	0,03	0,08

Tabelle 6.1.: Mittlere Abweichung der Distanz (in Pixel) und Orientierung (in Radiant) für die vier Bildsequenzen mit Rauschen und Helligkeitsänderungen.

- **Helligkeitsänderungen**: Wie bei vorheriger Bildsequenz, hatte auch hier die SIFT Implementierung von Lowe Probleme mit dem Bildausschnitt Teppichboden. In Bildern mit sofort aufeinander folgenden Helligkeitswechseln von Hell nach Dunkel konnte mangels Matches kein Tracking mehr vorgenommen werden. Auch die KLT hatte Probleme und brach schon nach wenigen Schritten ab. Für die anderen Features liegt der Fehler im Subpixelbereich zwischen durchschnittlich 0,13 und 0,54 Pixeln, bezogen auf die Distanz als Fehlermaß. Die SIFT-basierten Verfahren waren zu etwa einem Fünftel besser (beide 0,13 Pixel) als die SURF-Varianten (0,48 bis 0,54 Pixel).
- **Rauschen und Helligkeitsänderung**: Insgesamt zeigte sich diese Kombination anspruchsvoller für die Feature Extraktion. Mit durchschnittlich 63% konnten hier weniger Features als zuvor beim Matching zugeordnet werden. Die durchschnittliche Fehlerdistanz lag im Bereich von 0,29 bis 1,27 Pixel. Bestes Ergebnis lieferte die freie SIFT Implementierung (0,29 Pixel), gefolgt von der Lowe-SIFT (0,71 Pixel). Bestes Ergebnis bei SURF lieferte die U-SURF Variante (1,12 Pixel).

Bezogen auf alle Bildsequenzen zeigten die SIFT-basierten Verfahren eine bessere Performance im Hinblick auf den Abweichungsfehler bei der Distanz als SURF (siehe Tabelle 6.1). Die durchschnittliche Distanzabweichung bei SIFT war mit 0,22 Pixel etwa dreimal geringer als die 0,74 Pixel bei SURF. Im Hinblick auf die Zeit, die für die Extraktion der Features benötigt wurde, war SURF etwa viermal schneller als SIFT. Die native SURF-Variante benötigte im Schnitt 0,29 Sekunden gegenüber 1,34 Sekunden bei SIFT, wobei die SIFT-Implementierung von Lowe im Mittel zirka 20% schneller war als die freie Implementierung (durchschnittlich 1,48 Sekunden zu 1,2 Sekunden). Trotz wesentlich kürzerer Berechnungszeit waren die SURF-Features besser beim Matching. Mit SURF konnten durchschnittlich etwa 81% der gefunden Features zugeordnet werden, als 72% bei SIFT. Die KLT erfüllte die Erwartungen beim einfachsten Fall der Bildsequenzen ohne Bildtransformationen, versagte aber bei den anderen Bildsequenzen mit Rauschen oder Helligkeitsänderungen.

6.2.2. Bewegung

Für die Erzeugung von Bildsequenzen mit Bewegung wurden die gleichen Bildoperationen wie beim Stillstand plus die Operation zufällige Translation (siehe Abschnitt 6.1) berücksichtigt. Die Menge möglicher Bildoperationen war:

Zufällige Translation
Zufällige Translation und Rauschen
Zufällige Translation und Helligkeitsänderung
Zufällige Translation, Rauschen und zufällige Helligkeitsänderung

Die Ergebnisse sind in Tabelle B.2 zusammengefasst. SURF zeigt sich insgesamt als das schnellste und robusteste Verfahren. Für alle Bildsequenzen basierend auf den sechs Ausgangsbildern war ein ausreichend gutes Tracking mit den SURF-Features möglich. Die SIFT-basierten Verfahren hatten Probleme mit Innenaufnahmen. Bei der Operationsmenge zufällige Translation und Helligkeitsänderung wurde für beide SIFT-Implementierungen an der gleichen Stelle abgebrochen. Zu diesem Zeitpunkt gab es einen starken Helligkeitsabfall. Es wurden zwar weiterhin Feature Punkte extrahiert, diese konnten aber nicht mehr den Features vorheriger Bilder zugeordnet werden. Der KLT Tracker konnte nur in den Fällen „zufälligen Translationen“ und „zufällige Translation und Rauschen“ bei den Bildern mit Innenaufnahmen ein Tracking durchführen, dort aber mit einem sehr hohen Abweichungsfehler von 32,95 Pixeln beziehungsweise 17,34 Pixeln.

Die Ergebnisse von Tabelle B.2 getrennt betrachtet für die einzelnen Operationen:

- **Zufällige Translation**: Die besten Ergebnisse bezogen auf die Fehlerdistanz lieferte hier der SURF Detektor bei SURF-128 (0,66 Pixel), dann U-SURF (0,70 Pixel) und SURF (0,79 Pixel). SIFT (2,77 Pixel) und Lowe-SIFT (1,74 Pixel) zeigen deutlich höhere Abweichungen bei dreimal so hoher Rechenzeit (durchschnittlich 1,4 Sekunden) gegenüber SURF (0,29 Sekunden). Die Anteil gefundener Matches war bei SIFT (ca. 52%) doppelt so groß als bei SURF (ca. 26%). Dies liegt aber darin begründet, dass SIFT grundsätzlich eine größere Features-Menge produziert, die für das Matching genutzt werden kann. Mit der Lowe-SIFT gab es aber bei der Innenaufnahme *Wohnzimmerboden* Probleme beim Matching . Dort brach die Analyse gleich zu Beginn ab, weil auf für den Teppichboden keine Features gefunden wurden. Der KLT Tracker konnte nur auf den Innenarbeiten ein Tracking ausführen. Dort hatte er eine vergleichsweise sehr hohe Abweichung von 32,95 Pixeln, konnte das Tracking aber am schnellsten ausführen (0,16 Sekunden).

- **Translation und Rauschen**: Auch hier lieferten die SURF-Features für das Tracking die besten Ergebnisse (durchschnittlich 0,63 Pixel Distanzfehler) bei einer mittleren Rechenzeit von 0,26 Sekunden. Der Anteil der Matches lag im selben Bereich wie der von SIFT (36-41%). Bei der Rechenzeit benötigten SIFT und Lowe-SIFT etwa fünfmal so lange als SURF (durchschnittlich 1,25 Sekunden). Die Lowe-SIFT hatte Probleme mit der Innenaufnahme *Wohnzimmerboden.* Schon schon zu Beginn wurden keine Features für Teppichboden gefunden, was zu einem Abbruch der Analyse führte. Für die restlichen Bildsequenzen war die Fehlerdistanz 1,31 Pixeln für SIFT und 1,19 Pixeln bei Lowe-SIFT. Der KLT Tracker lieferte wie beim ersten Fall mit nur zufälliger Translation nur Tracking-Ergebnisse für die Innenaufnahmen. Die Abweichungen waren mit 17,34 Pixeln zwar besser, aber immer noch sehr groß.

Fehler	SIFT	SIFT Lowe	SURF	SURF-128	U-SURF
Distanz	2,08	2,49	1,02	0,86	0,93
Orientierung	-0,01	0,05	0	0	0,02

Tabelle 6.2.: Mittlere Abweichung der Distanz (in Pixel) und Orientierung (in Radiant) für die vier Bildsequenzen mit Bewegung.

- **Translation und Helligkeitsänderung**: Hier zeigte sich bei SIFT und SURF ein ähnliches Schema wie im vorigen Fall. Die Fehlerdistanzen sind in etwa gleich, während die Berechnungszeit signifikant zunahm (Steigerung um 24%) und beim Matching mehr Zuordnungen gefunden wurden (durchschnittlich 58% gegenüber 38% bei nur Translation). Beim Bild *Wohnzimmerboden* wurden etwa an derselben Stelle in der Bildsequenz mit den Detektoren SIFT und Lowe-SIFT keine Features mehr gefunden, was in einem Abbruch der Analyse resultierte. Die KLT konnte auf keinen Bildsequenzen ein Tracking vornehmen.

- **Translation, Rauschen und Helligkeitsänderung**: Wie im vorigen Fall hatten SIFT und Lowe-SIFT hier Probleme mit dem Bildtyp *Wohnzimmerboden*. Die SURF-Varianten dagegen zeigten auch hier das beste Resultat (Fehlerdistanz 0,77-0,9 Pixel) bei schneller Berechnungszeit (0,17-0,32 Sekunden). Wiederum konnte mit dem KLT Tracker auf allen sechs Bildsequenzen kein Tracking ausführt werden, was zu einem vorzeitigem Abbruch der Analyse bei KLT führte.

In Tabelle 6.2 sind die kumulierten Abweichungen zu den Distanzen und Orientierungen zusammengefasst. Die Abweichung bei der Orientierung ergibt sich aus dem Parameter Θ des Transformationsmodells (siehe Abschnitt 4.2), also aus der Abweichung der Bewegungsrichtung in Radiant. Es zeigt sich, dass die SURF-Features deutlich besser als die SIFT-Features sind. Dies sowohl bei der Distanz als auch bei den Orientierungen. Die besten Resultate liefert SURF-128 mit durchschnittlich 0,86 Pixel Abweichung in der Distanz und 0 Grad Abweichung bei der Orientierung. Berücksichtigt man aber, dass die Ergebnisse die durchschnittlichen Werte für 100 Bilder sind, so sind die Abweichungen verhältnismäßig groß. Bei 100 Bildern und 0,86 Pixeln.

6.3. Reale Bildsequenzen

Für die Bildszene vom Typ Holzdielenboden wurden Bildsequenzen mit Stillstand und konstanter Bewegung erzeugt. Eine Kamera (Webcam Logitech Quickcam Communicate STX Plus) wurde an einem Tisch befestigt, die Aufnahmehöhe betrug 70cm. Die bei 25 Bildern pro Sekunde aufgenommenen Bildsequenzen zeigen einen Holzdielenboden (siehe Abbildung 6.3) mit Maserung, ähnlich dem Bodentyp (e) in Abbildungsich 6.1. Die Aufnahmen weisen Rauschen

(a) Kameraaufnahme mit Lichtverhältnissen bei aktivierter Raumbeleuchtung.

(b) Starker Helligkeitsabfall beim Ausschalten der Raumbeleuchtung.

Abbildung 6.3.: Kameraaufnahme bei den realen Bildsequenzen. Zu sehen ist ein Holzboden mit Maserung und Helligkeitsänderungen.(a) zeigt den Boden bei aktivierter Deckenbeleuchtung. (b) zeigt das erste Frame nach Ausschalten der Beleuchtung; die Kamera hat sich noch nicht an die neuen Bedingungen angepasst.

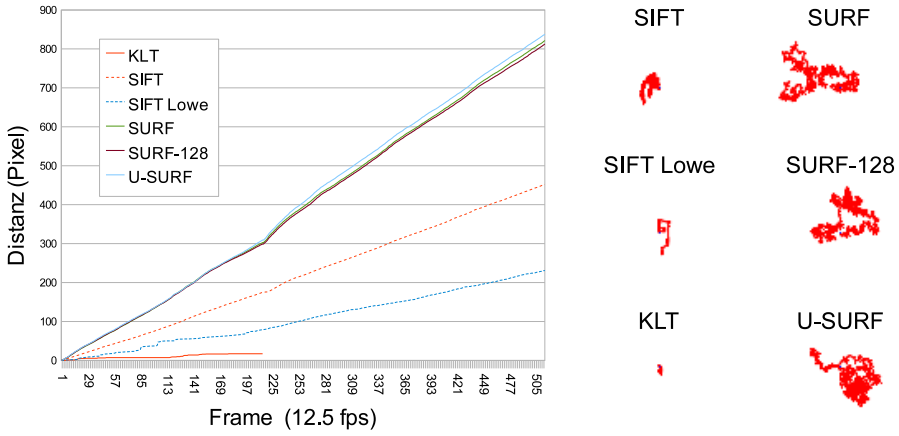
und Fluktuationen bei der Helligkeit auf, was sich auf das verwendete Kamaramodell zurückführen lässt. Von den 25 Bildern pro Sekunde wurde bei der Anwendung nur jedes zweite Frame berücksichtigt, so dass die effektive Anzahl Bilder pro Sekunde 12,5 betrug.

6.3.1. Stillstand

Die Bildsequenz zum Stillstand setzt sich aus 518 Einzelbildern zusammen, die sich nur durch Änderungen in der Helligkeit unterscheiden. Auf diesen Bildern wurden Features erzeugt, und die Abweichungen von der Referenzbewegung ermittelt. Die Referenzbewegungen bei Stillstand betragen in allen Parametern des Bewegungsmodells den Wert Null.

In Abbildung 6.4 sind die kumulierten Distanzen der einzelnen Feature Detektoren grafisch dargestellt. Man sieht, dass der KLT Tracker nach zirka dem ersten Drittel keine Feature Punkte mehr findet. Idealerweise sollten die berechneten Distanzen Null betragen. Die SIFT- und SURF-basierten Verfahren zeigen hier das beste Ergebnis. Die Implementierung von Lowe liefert geringere Distanzabweichungen gegenüber der frei verfügbaren Implementierung. Anhand der gezeichneten Trajektorien (Abbildung 6.4) wird deutlich, dass SIFT die verlässlichsten Ergebnisse lieferte und die SURF-Varianten größere Ablenkungen produzierten.

Bis auf den KLT Tracker arbeiteten alle Feature Detektoren auf den realen Bildsequenzen mit Stillstand gut. Das Tracking wurde auf allen 518 Bildern ausgeführt, nur der KLT Tracker konnte bei Frame 215 (ca. 41%) keine Feature Punkte mehr zum Tracking finden. Zu diesem Zeitpunkt wurde im Raum das Licht ausgeschaltet (siehe Abbildung 6.3b). Dies zeigt deutlich, dass KLT stark von schnellen Helligkeitsänderungen beeinflusst wird, während die SIFT- und SURF-basierten Verfahren wegen der Normalisierung des Deskriptorvektors nicht durch



(a) Kummulierte Distanzabweichungen.

(b) Trajektorie

Abbildung 6.4.: Distanzabweichungen bei realen Bildsequenzen mit Stillstand. Der KLT Tracker bricht bei ca. 41% ab, zeigt aber für diesen Bereich gute Ergebnisse. Die SURF-basierten Verfahren SURF, SURF-128 und U-SURF zeigen alle ähnliche Resultate.

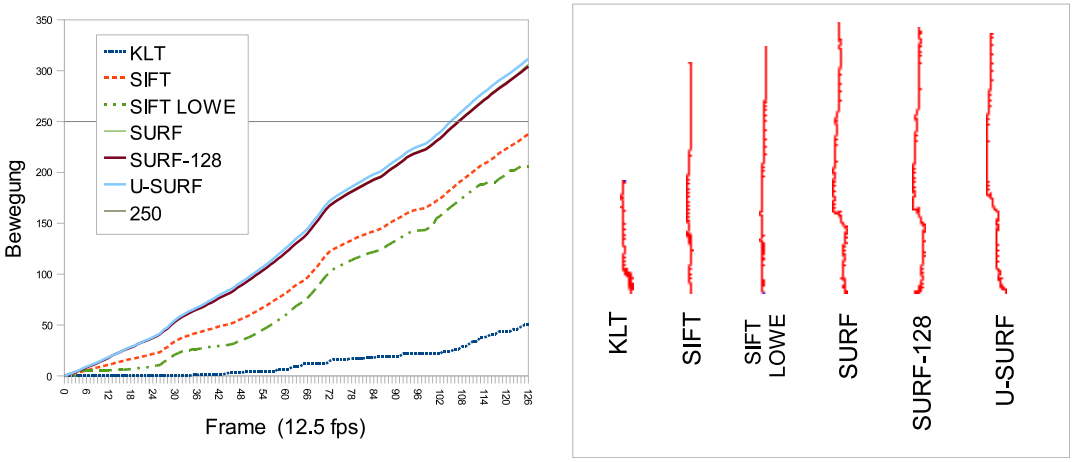
Helligkeitsänderungen beeinflusst werden. Beispiele für gravierende Helligkeitsänderungen im Außenbereich sind etwa Tunnel, wo harte Schnitte bei den Intensitätswerten der Bilder erfolgen.

Tabelle 6.3 listet die Summen, Mittelwerte und Varianzen bei Stillstand für die Distanzabweichungen und Orientierungen auf. Die Distanzabweichungen geben den Versatz pro Frame an, der anhand eines Features berechnet wurde und bei Stillstand Null betragen müsste.

Dass die Mittelwerte der Distanzänderungen nicht sehr nahe bei Null liegen ist dadurch begründet, dass die Bilder Einflüssen entweder durch die Kamera selbst oder von Außen ausgesetzt sind. Ersteres wird durch die bereits erwähnten, ständigen Bildhelligkeitsanpassungen der

	Distanzabweichungen			Orientierungen	
Detektor	Summe	Mittelwert	Varianz	Mittelwert	Orientierung
SIFT	452,1	0,88	0,04	0,04	2,88
SIFT (Lowe)	230,9	0,36	0,72	0,03	2,25
SURF	821,6	1,6	0,16	-0,04	3,33
SURF-128	812,6	1,58	0,16	-0,07	3,17
U-SURF	837,5	1,63	0,17	0,01	3,22

Tabelle 6.3.: Mittelwert und Varianz der Feature Detektoren bei realen Bildsequenzen mit Stillstand. Der KLT Tracker konnte ab ca. 41% kein Tracking mehr ausführen, und ist deshalb nicht aufgeführt.



(a) Bewegung pro Frame

(b) Trajektorien

Abbildung 6.5.: Kumulierte berechnete Bewegungen bei der realen Bildsequenz.Grundlage war die tatsächliche Bewegung in eine Richtung auf einer Länge von 50cm .

Kamera oder Bildrauschen verursacht. Einflüsse von Außen sind zum Beispiel Vibrationen nebenstehender Geräte (Computer).

6.3.2. Bewegung

Für die Bildsequenz mit Bewegung, wurde die Kamera mit nahezu linearer Geschwindigkeit in eine konstante Richtung bewegt, während die zurückgelegte Strecke mit Hilfe eines Maßbands kontrolliert wurde. Die so erzeugte Bildsequenz hat eine Länge von 160 Einzelbildern und repräsentiert eine Bewegung um ca. 50cm. Da dieselbe Kamera wie beim Stillstand benutzt wurde, sind auch hier Helligkeitsfluktuationen zwischen den Bildern vorhanden. Starke Helligkeitsänderungen wurden über Zustandswechsel der Raumbeleuchtung simuliert.

In Abbildung 6.5 sind die kumulierten Bewegungen und die zugehörigen Trajektorien grafisch dargestellt. Die Bewegung ist die Distanz zwischen zwei Frames. Visualisiert man die Bewegung (Abbildung 6.5b) ergeben sich zwar für alle Detektoren ein einheitliches Bild bezüglich der Richtung, die Länge der Bewegung ist aber stark unterschiedlich. In der Steigung der Kurven zeigt sich nicht konstante Geschwindigkeit der Bewegung, da diese „von Hand“ ausgeführt wurde. Interessant ist, dass für für unterschiedliche Feature Detektoren auch unterschiedlich große Bewegungsänderung verzeichnet worden sind. Während der KLT Tracker anfangs überhaupt keine Bewegung anzeigt, sind SIFT und SURF relativ gleich, driften aber zum Ende hin stark auseinander. Durch einen visuellen Vergleich auf den Ausgangsdaten konnte empirisch der Wert 250 für die Gesamtbewegung für 50cm bestimmt werden. Die freie SIFT Implementierung lag diesem Wert am nächsten.

7. Zusammenfassung

In dieser Arbeit wurde ein System zum automatisierten Vergleich von Feature Detektoren vorgestellt, das Features anhand ihrer Abweichungsquote beim Stillstand im Hinblick auf die Performance zum Tracking vergleicht. Das implementierte System arbeitet sowohl auf künstlich erzeugten Bildsequenzen, für die es unterschiedliche Bildtransformationen simuliert und die Abweichungen subpixelgenau bestimmt, als auch auf realen Bildsequenzen, bei denen die Auswertung in Relation zur visualisierten Trajektorie erfolgen kann.

Überraschend war die unterschiedlichen Reaktionen der Merkmalsextraktoren KLT, SIFT und SURF in der Simulation mit künstlich erzeugten Bildsequenzen bei Stillstand. Alle Feature Detektoren erfüllen bei Stillstand und keinerlei Bildtransformationen die Erwartungen. Bei rauschbehafteten und sich in der Helligkeit verändernden Bildsequenzen brachte die KLT keine Ergebnisse, während SIFT- und SURF-basierte Verfahren die aus der Literatur bekannten Erwartungen erfüllten. Während die SIFT Features deutlich mehr Features und Feature-Matches bei einem erhöhten Rechenaufwand liefern, zeigen sich die SURF Features tatsächlich als schnell zu berechnen und robust. Die Abweichungsfehler auf den Bildsequenzen mit simulierten Bildtransformationen sind bei SURF bis zu 50% besser im Vergleich mit SIFT - bei wesentlich kürzerer Berechnungszeit. Dies spricht für den Einsatz von SURF in zeitkritischen Systemen. Einfache Experimente auf realen Bildsequenzen zeigten bei der visuellen Auswertung, dass SIFT und SURF potentziell auch alleine ein robustes 2D-Tracking ermöglichen können.

Verbesserungen

Grundsätzlich besteht an einigen Stellen die Möglichkeit, das System robuster im Hinblick auf die Spurverfolgung zu machen und im Vergleich der Features komplexer zu gestalten. Je nach Detektortyp könnten bildvorverarbeitende Schritte unternommen werden, was die Performance einzelner Feature Detektoren steigert. Die so geänderten Rahmenbedingungen für das Vergleichskriterium über 2D-Tracking müssten dann angepasst werden. Besonders wichtig zur Verbesserung des Systems wären reale Bildsequenzen mit präzisen Referenzinformationen. Auch könnten die in dieser Arbeit genutzten Kameramodelle (Kleinbild Digitalkamera und Webcam) durch hochwertigere Kameras ersetzt werden, die qualitativ hochwertigere Bilder mit weniger Rauschen und schnellere Helligkeitsanpassunge bieten. Die eingesetzte Webcam benötigte für die Einstellung der Belichtungszeit vergleichsweise lange, was sich besonders bei KLT in den Ergebnissen widerspiegelte. Eine Verallgemeinerung auf komplexere Bewegungsmodelle kann erreicht werden, indem zusätzlich die Bildtransformationen Rotation, Skalierung und affine Transformationsmodelle berücksichtigt werden.

A. Geometrische Grundlagen

Im Folgenden werden die Konzepte des zweidimensionalen, euklidischen Vektorraums benutzt.
Ein Vektor hat die Form:

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

A.1. Metrik

Als Metrik wird die euklidische Metrik (euklidische Distanz) zur Abstandsmessung herangezogen. Es ist die Norm der Differenz der Vektoren $\vec{x}$, $\vec{y}$.

$$\begin{aligned} dist(\vec{x}, \vec{y}) &= |\vec{x} - \vec{y}| = \sqrt{(\vec{x} - \vec{y}) \cdot (\vec{x} - \vec{y})} && \text{(A.1)} \\ &= \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2} && \text{(A.2)} \end{aligned}$$

A.2. Winkel zwischen zwei Vektoren

Der Winkel zwischen zwei Vektoren $\vec{x}$, $\vec{y}$ in Bezug zum Origo wird berechnet durch die spezielle Form *atan2* des Arcustangens. Sie projiziert kartesische Koordinaten auf Polarkoordinaten, wobei die Projektion durch die Berücksichtigung des Vorzeichens in den korrekten Quadranten abbildet; der Winkel wird ausgegeben in Radiant.

$$atan2(y, x) = \begin{cases} sgn(y) \cdot \left|\frac{y}{x}\right| & x > 0 \\ sgn(y) \cdot \frac{\pi}{2} & x = 0 \\ sgn(y) \cdot (\pi - 2) & x < 0 \end{cases} \quad \text{(A.3)}$$

Die Funktion $sgn(y)$ gibt das Vorzeichen an: sgn(y)$= \begin{cases} +1 & x \geqq 0 \\ -1 & x < 0 \end{cases}$.

A.3. Mittelwert von zirkulären Werten

Der arithmetische Mittelwert ist nicht definiert für eine Menge von Winkeln. Eine Möglichkeit dennoch einen Mittelwert zu bestimmen ist, die Polarkoordinaten der Winkel in kartesische Koordinaten auf dem Einheitskreis umzuwandeln und darüber zu mitteln.

Für eine beliebige Menge von Winkeln $\alpha_{1,}..., \alpha_n, n \in \mathbb{N}$ gilt:

$$\bar{\alpha} = atan2\left(\frac{1}{n}\sum_{i=1}^{n}\sin\alpha_i, \frac{1}{n}\sum_{i=1}^{n}\cos\alpha_i\right) \tag{A.4}$$

$atan2$ ist der Arcustangens von y/x unter Berücksichtigung der Vorzeichen (siehe A.2) .

B. Ergebnisse Details

	Features	Matches		Detektor Zeit (sec)	Abweichung (Distanz in Pixel)
Keine Aktion					
KLT	100			0,08	0
SIFT	1752,5	1731,02	98,77%	1,46	0
SIFT Lowe	1594,17	1578,23	99,00%	1,25	0
SURF	438,5	433,79	98,93%	0,36	0
SURF-128	438,5	433,79	98,93%	0,4	0
U-SURF	438,5	433,79	98,93%	0,19	0
Rauschen					
KLT	(Abbruch)				-
SIFT	1476,89	760,79	51,51%	1,29	0,24
SIFT Lowe	1469,18	776,55	52,86%	1,04	0,18
SURF	312,96	224,56	71,75%	0,27	1,15
SURF-128	312,96	213,2	68,12%	0,3	1,16
U-SURF	312,96	242,59	77,51%	0,16	1,01
Helligkeitsänderung					
KLT	(Abbruch)				-
SIFT	2037,77	1622	79,60%	1,64	0,13
SIFT Lowe	1570,06	1294,08	82,42%	1,24	0,13
SURF	437,32	371,29	84,90%	0,36	0,54
SURF-128	437,32	357,71	81,80%	0,42	0,50
U-SURF	437,32	397,89	90,98%	0,18	0,48
Rauschen und Helligkeitsänderung					
KLT	(Abbruch)				-
SIFT	1842,24	915,39	49,69%	1,51	0,29
SIFT Lowe	1602,63	853,08	53,23%	1,25	0,71
SURF	354,97	249,36	70,25%	0,31	1,27
SURF-128	354,97	233,78	65,86%	0,35	1,22
U-SURF	354,97	277,31	78,12%	0,16	1,12

Tabelle B.1.: Übersicht Ergebnisse Bildoperationen für Bildsequenzen mit Stillstand In den Spalten stehen die Anzahl der Features, die Anzahl zugeordneter Features, die Zeit, die für die Detektion benötigt wird in Sekunden, und die Abweichung (Fehler) bezüglich Distanz.

	Fehler-bilder	Features	Matches		Detektor Zeit (sec)	Distanz-fehler
Translation						
KLT	4	100	55,41		0,16	32,95
SIFT	0	1664,44	786,63	47,26%	1,44	2,77
SIFT Lowe	2	1752,09	1008,31	57,55%	1,37	1,74
SURF	0	399,68	100,39	25,12%	0,33	0,79
SURF-128	0	399,68	107,08	26,79%	0,37	0,66
U-SURF	0	399,68	108,37	27,11%	0,18	0,7
Mittelwert:		785,93	361,03	36,77%	0,64	6,6
Translation und Rauschen						
KLT	4	100	70,85		0,14	17,34
SIFT	0	1505,11	547,31	36,36%	1,31	2,08
SIFT Lowe	1	1440,93	555,3	38,54%	1,19	2,69
SURF	0	325,8	119,87	36,79%	0,28	0,69
SURF-128	0	325,8	116,68	35,81%	0,32	0,64
U-SURF	0	325,8	133,42	40,95%	0,16	0,57
Mittelwert:		670,58	257,24	37,69%	0,57	4
Translation und Helligkeitsänderung						
KLT	6	(Abbruch)				
SIFT	1	2017,94	1216,47	60,28%	1,64	1,86
SIFT Lowe	1	1867,91	1223,69	65,51%	1,41	2,57
SURF	0	463,76	244,7	52,76%	0,38	0,69
SURF-128	0	463,76	237,96	51,31%	0,43	0,68
U-SURF	0	463,76	269,75	58,17%	0,2	0,57
Mittelwert:		1055,43	638,51	57,61%	0,81	1,27
Translation, Rauschen und Helligkeitsänderung						
KLT	6	(Abbruch)				
SIFT	1	1848,2	742,04	40,15%	1,52	1,72
SIFT Lowe	1	1605,11	708,72	44,15%	1,26	1,83
SURF	0	367,45	180,52	49,13%	0,32	0,9
SURF-128	0	367,45	170,83	46,49%	0,35	0,85
U-SURF	0	367,45	205,1	55,82%	0,17	0,77
Mittelwert:		911,13	401,44	47,15%	0,72	1,22

Tabelle B.2.: Übersicht Ergebnisse Bildoperationen für Bildsequenzen mit Bewegungen. In den Spalten stehen die Anzahl der Fehlerbilder (Abbrüche), Anzahl der Features, die Anzahl zugeordneter Features, die Zeit, die für die Detektion benötigt wird in Sekunden, und die Abweichung (Fehler) bezüglich Distanz.

Abbildungsverzeichnis

Literaturverzeichnis

[1] ANGELI, A., D. FILLIAT, S. DONCIEUX und J.-A. MEYER: *2D Simultaneous Localization And Mapping for Micro Aerial Vehicles*. In: *European Micro Aerial Vehicles (EMAV 2006)*, 2006.

[2] BAY, H., T. TUYTELAARS und L. J. V. GOOL: *SURF: Speeded Up Robust Features*. 3951:404–417, 2006.

[3] BLANCO, J., W. BURGARD, R. SANZ und J. FERNANDEZ: *Fast Face Detection for Mobile Robots by Integrating Laser Range Data with Vision*. In: *Proc. of the International Conference on Advanced Robotics (ICAR)*, 2003.

[4] BOWYER, K., C. KRANENBURG und S. DOUGHERTY: *Edge detector evaluation using empirical ROC curves*. Comput. Vis. Image Underst., 84(1):77–103, 2001.

[5] BRAND, P. und R. MOHR: *Accuracy in Image Measure*. In: EL-HAKIM, S. (Hrsg.): *Proceedings of the* SPIE *Conference on Videometrics III, Boston, Massachusetts, USA*, Bd. 2350, S. 218–228, November 1994.

[6] BROWN, MATTHEW, LOWE und DAVID: *Automatic Panoramic Image Stitching using Invariant Features*. International Journal of Computer Vision, 74(1):59–73, August 2007.

[7] BROWN, M. und D. LOWE: *Invariant Features from Interest Point Groups*. In: *British Machine Vision Conference*, 2002. Poster Session.

[8] DAHLKAMP, H., A. KAEHLER, D. STAVENS, S. THRUN und G. R. BRADSKI: *Self-supervised Monocular Road Detection in Desert Terrain.*. In: SUKHATME, G. S., S. SCHAAL, W. BURGARD und D. FOX (Hrsg.): *Robotics: Science and Systems*. The MIT Press, 2006.

[9] FISCHLER, M. A. und R. C. BOLLES: *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*. Commun. ACM, 24(6):381–395, June 1981.

[10] GRABNER, M., H. GRABNER und H. BISCHOF: *Fast Approximated SIFT*. In: *Asian Conference on Computer Vision*, S. I:918–927, 2006.

[11] HARALICK, R. M., H. JOO, C. LEE, X. ZHUANG, V. G. VAIDYA und M. B. KIM: *Pose estimation from corresponding point data*. Systems, Man and Cybernetics, IEEE Transactions on, 19(6):1426–1446, 1989.

[12] HARRIS, C. und M. STEPHENS: *A combined corner and edge detector*. ?, S. 147–151, 1988.

[13] HEATH, M. D., S. SARKAR, T. SANOCKI und K. W. BOWYER: *Robust Visual Method for Assessing the Relative Performance of Edge-Detection Algorithms*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(12):1338–1359, 1997.

[14] KE, Y. und R. SUKTHANKAR: *PCA-SIFT: A More Distinctive Representation for Local Image Descriptors*. S. 506–513, 2004.

[15] LINDEBERG, T.: *Scale-space theory: A basic tool for analysing structures at different scales*. J. of Applied Statistics, 21(2):224–270, 1994. (Supplement on Advances in Applied Statistics: Statistics and Images: 2).

[16] LÓPEZ, A. M., F. LUMBRERAS, J. SERRAT und J. J. VILLANUEVA: *Evaluation of Methods for Ridge and Valley Detection*. IEEE Trans. Pattern Anal. Mach. Intell., 21(4):327–335, 1999.

[17] LOWE, D.: *Object Recognition from Local Scale-Invariant Features*. In: *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 1999.

[18] LOWE, D. G.: *Object Recognition from Local Scale-Invariant Features*. In: *Proc. of the International Conference on Computer Vision ICCV, Corfu*, S. 1150–1157, 1999.

[19] LOWE, D. G.: *Distinctive Image Features from Scale-Invariant Keypoints*. International Journal of Computer Vision, 60(2):91–110, 2004.

[20] LUCAS, B. und T. KANADE: *An Iterative Image Registration Technique with an Application to Stereo Vision*. In: *IJCAI81*, S. 674–679, 1981.

[21] MIKOLAJCZYK, K. und C. SCHMID: *A performance evaluation of local descriptors*, 2003.

[22] MIKOLAJCZYK, K. und C. SCHMID: *A Performance Evaluation of Local Descriptors*. IEEE Trans. Pattern Anal. Mach. Intell, 27(10):1615–1630, 2005.

[23] MIRISOLA, L. G. B., J. LOBO und J. DIAS: *Stereo Vision 3D Map Registration for Airships using Vision-Inertial Sensing*. In: *The 12th IASTED Int. Conf. on Robotics and Applications*, 2006.

[24] MURPHY, K. P., A. B. TORRALBA, D. EATON und W. FREEMAN: *Object Detection and Localization Using Local and Global Features*. In: *Toward Category-Level Object Recognition*, S. 382–400, 2006.

[25] ROTTMANN, A., C. PLAGEMANN, P. HILGERS und W. BURGARD: *Autonomous Blimp Control using Model-free Reinforcement Learning in a Continuous State and Action Space*. In: *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, October 2007. To appear.

[26] ROTTMANN, A., M. SIPPEL, T. ZITTERELL, W. BURGARD, L. REINDL und C. SCHOLL: *Towards an Experimental Autonomous Blimp Platform*. In: *Proc. of the European Conference on Mobile Robots (ECMR)*, Freiburg, Germany, 2007.

[27] SCHMID, C., R. MOHR und C. BAUCKHAGE: *Evaluation of Interest Point Detectors*. International Journal of Computer Vision, 37(2):151–172, 2000.

[28] SE, S., D. LOWE und J. LITTLE: *Local and Global Localization for Mobile Robots using Visual Landmarks*, October 2001.

[29] SE, S., D. LOWE und J. LITTLE: *Vision-based Mobile robot localization and mapping using scale-invariant features*. In: *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, S. 2051–2058, Seoul, Korea, May 2001.

[30] SHI, J. und C. TOMASI: *Good Features to Track*. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, Seattle, Juni 1994.

[31] STEDER, B.: *Techniken für bildbasiertes SLAM unter Verwendung von Lagesensoren*. Diplomarbeit, Albert-Ludwigs-Universität, Freiburg, 2007.

[32] TOMASI, C. und T. KANADE: *Detection and Tracking of Point Features*. Techn. Ber. CMU-CS-91-132, Carnegie Mellon University, April 1991.

[33] TORRALBA, A., K. MURPHY, W. FREEMAN und M. RUBIN: *Context-based Vision System for Place and Object Recognition*. In: *Proc. of the Int. Conf. on Computer Vision (ICCV)*, 2003.

[34] VALGREN, C. und A. J. LILIENTHAL: *SIFT, SURF and Seasons: Long-term Outdoor Localization Using Local Features*. S. 253–258, September 19–21 2007.

[35] VAUGHAN, R., G. SUKHATME, F. MESA-MARTINEZ und J. MONTGOMERY: *Fly spy: Lightweight localization and target tracking for cooperating air and ground robots*, 2000.