Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Andreas Liedtke

Realisierung eines robusten Verfahrens zur Identifikation von Positionsmarken unter Verwendung von Bildverarbeitung in Videobildern

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Andreas Liedtke

Realisierung eines robusten Verfahrens zur Identifikation von Positionsmarken unter Verwendung von Bildverarbeitung in Videobildern

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.Ing. Andreas Meisel
Zweitgutachter : Prof. Dr. rer. nat. Reinhard Baran

Abgegeben am 30. März 2011

Andreas Liedtke

Thema der Bachelorarbeit

Realisierung eines robusten Verfahrens zur Identifikation von Positionsmarken unter Verwendung von Bildverarbeitung in Videobildern.

Stichworte

SURF, SURF-Marken, SIFT, Lokalisierung mobiler Roboter

Kurzbeschreibung

In der vorliegenden Arbeit wird ein Verfahren, zur Lokalisierung autonomer mobiler Roboter, auf Basis des SURF-Merkmalsextraktor und optimaler SURF-Marken, entwickelt. Autonome mobile System, ausgestattet mit einer Kamera, sollen ihre Position anhand der verwendeten Positionsmarken bestimmen können. Das entwickelte Verfahren soll so robust wie möglich gegenüber Störeffekten wie Bildrauschen, Belichtungsänderungen und Perspektivischen Verzerrungen sein.

Andreas Liedtke

Title of the paper

Implementation of a robust procedure for landmark identification in video streams using image processing.

Keywords

SURF, SURF-Marker, SIFT, localization of mobile robots

Abstract

This paper describes a methode for localization of autonomous mobile robots. Based on the SURF feature detector and optimal SURF-Markers. Autonomous mobile robots equipped with a camera, shall be able to estimate their position from the used landmarks. The developed procedure target to be as robust as possible against negative effects like image noise, illumination changes and perspective distortion.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einführung

Als autonome mobile Roboter bezeichnet man Roboter, die sich in Ihrer Umgebung selbständig zurechtfinden, und selbständig Aktionen durchführen können. Damit ein Roboter als autonomes System klassifiziert werden kann, reicht es wenn sich die steuernde Logik (vgl. Abb. 1.1 Ebene: Planung, Koordination und Kooperation), als Hardware oder Software, auf dem Roboter selbst befindet. Das der Roboter möglicherweise Aufgaben, oder Anweisungen wie genau er seine Aufgaben zu erledigen hat, von einer externen Anwendung, die sich nicht auf dem Roboter befindet, bekommt, schränkt seine Autonomie nicht ein.

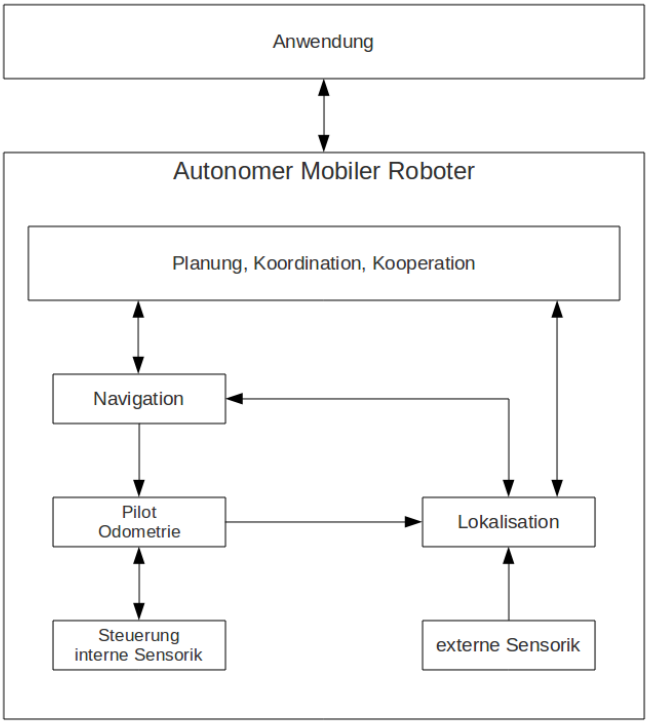

Abbildung 1.1.: Schemenhafter Aufbau eines autonomen mobilen Roboters. Quelle: [Rupp (2001)]

Autonome mobile Roboter sind, in der Industrie, schon seit geraumer Zeit zu finden. Sie befördern z. B. Werkstücke zwischen den einzelnen Stationen einer Fertigungsstraßen, oder

verwalten Hochregallager vollautomatisch. In jüngster Vergangenheit drängen autonome mobile Roboter in immer mehr Einsatzbereiche, sie sind beispielsweise als mobile Informationssystem in Museen oder bei der Überwachung von Gebäuden zu finden.
Unter der Lokalisierung mobiler Systeme, versteht man die Bestimmung der Postion des mobilen Systems. Eine Lokalisierung kann dabei **relativ**, zu einer vorher bekannten Postion, oder **absolut** in einem Koordinatensystem erfolgen. Eine möglichst ressourcenschonendes System zur Lokalisierung, ist dabei für autonome mobile Roboter besonders wichtig, da die Lokalisierung eine Aufgabe ist, die durchgehen bewältigt werden muss. Die relative Lokalisierung stellt in der Regel die einfachere Möglichkeit dar, da sie auf Basis der internen Sensorik, durchgeführt werden kann. Bewegt der Roboter sich zum Beispiel, von einer bekannten Postion aus, fünf Meter in eine bestimmte Richtung, kann die neue Postion mit sehr geringem Rechenaufwand bestimmt werden. Die absolute Lokalisierung ist in der Regel auf externe Sensorik angewiesen. Das in dieser Arbeit beschriebene Lokalisierungssystem, auf Basis von Positionsmarken, die mittels Bildverarbeitung ausgewertet werden, ist eine Möglichkeit zur absoluten Lokalisierung, mittels externer Sensoren (vgl. Abb. 1.1). Ein großer Vorteil der absoluten Lokalisierung, ist die Möglichkeit nach einem Ausfall oder bei der ersten Inbetriebnahme des Systems, ohne Vorkenntnisse über die aktuelle Position, eine Lokalisierung zu erreichen.

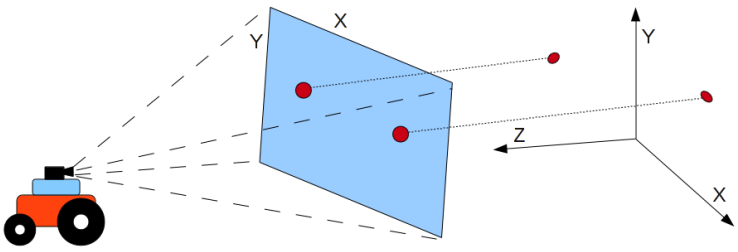

Abbildung 1.2.: Der mobile Roboter lokalisiert Positionsmarken in einem zweidimensionalen Kamerabild, und bestimmt dadurch seine absolute Position im Raum. In dieser Arbeit für nur die Lokalisierung der Positionsmarken behandelt.

1.1. Lösungsansatz

In Ihrer Arbeit *"Maximum Detector Response Markers for SIFT and SURF"* entwickeln [Schweiger u. a. (2009)] optimale Marken für die Merkmalsextraktoren SIFT [Lowe (2004)] und SURF [Bay u. a. (2006)] . Nach der Herleitung des optimalen Designs der Marken wird deren hohe Nachweisbarkeit bestätigt und beispielhaft einige Anwendungsmöglichkeiten aufgezeigt. Es bleiben aber auch Fragen offen. Die Experimente mit den optimalen Marken weisen zum einen auf gravierende Mängel der verwendeten SURF-Implementierung hin. Zum anderen werden die Experimente nur mit synthetischen Bildern und Bildern hochauflösender Kameras durchgeführt. Die Ergebnisse lassen also keine endgültige Aussage über die Praxistauglichkeit der optimalen SIFT- und SURF-Marken zu. In dieser Arbeit soll untersucht werden ob es möglich ist, basierend auf den Optimalen-SURF-Marken, ein praxistaugliches System zur Lokalisierung autonomer mobiler Roboter zu implementieren, und welche Robustheit mit einem solchen System gegenüber typischen Störgrößen, in der Bildverarbeitung, erreicht werden kann. Die Überlegenheit von SURF gegenüber SIFT, bezüglich der Performance, wurde bereits an vielen Stellen aufgezeigt vgl. z. B. (Juan u. Gwun, Seite 156). Da das System auch auf eingebetteten Systemen, mit beschränkter Rechenleistung, Lokalisierung in nahezu Echtzeit ermöglichen soll, wird im Rahmen dieser Arbeit, nur ein auf SURF basierender Ansatz Untersucht.

1.2. Aufbau dieser Arbeit

Dieser Abschnitt beschreibt den Aufbau der Arbeit und nennt die wesentlichen Punkte der einzelnen Kapitel.

Kapitel 2 gibt eine Übersicht über den Themenbereich der Merkmalsextraktion (english: feature detection). Im Detail wird auf die Funktionsweise und Parametrierung des SURF (Speeded Up Robust Features) Algorithmus eingegangen.

In Kapitel 2.2.4 werden die grundlegenden Anforderungen an eine SURF-Implementierung als Basis eines Lokalisierungssystems definiert und mehrere Kandidaten vorgestellt.

In Kapitel 3 werde die Optimalen-SURF-Marken von [Schweiger u. a. (2009)] vorgestellt und in mehreren Testreihen mit synthetischen Bildern Untersucht.

In Kapitel 4 folgt die detaillierte Beschreibung der technischen Umsetzung des gefunden Lösungsansatz.

Kapitel 5 fasst die Ergebnisse dieser Arbeit zusammen, nennt offene Fragen und mögliche Verbesserungsvorschläge.

2. Merkmalsextraktion

Es gibt keine allgemeingültige Definition für den Begriff "Merkmalsextraktion"(english: feature detection), gemein haben alle Algorithmen die unter diesem Begriff zusammengefasst werden, das sie in einem Eingangsbild bestimmte markante Merkmale (english: features) auffinden. Typen von Merkmalen sind z. B. Kanten (edge detection), Ecken (corner detection) oder tropfenförmige Strukturen (blob detection). Über diese Gruppen ist allerdings keine eindeutige Klassifizierung der Algorithmen möglich, da es Algorithmen gibt die sowohl Kanten und Ecken oder Ecken und tropfenförmige Strukturen erkennen. Optional liefern einige Merkmalsextratoren, zusätzlich zu den Koordinaten des gefunden Merkmals, eine Beschreibung des Merkmals oder dessen Umgebung. Diese Beschreibung kann als eine Art Fingerabdruck für das jeweilige Merkmal verstanden werden, mit dessen Hilfe das Merkmal, in einem anderen Bild, wiedererkannt werden kann. Merkmalsextratoren sind in praktischen Anwendungen in der Regel low-level Algorithmen, mit deren Hilfe übergeordnete Algorithmen komplizierte Aufgabenstellungen lösen. Mögliche Aufgabenstellungen und Anwendungsbeispiele sind:

- Objekterkennung
  - Schrifterkennung
  - Gesichtserkennung
  - Gestenerkennung
- 3D-Modellierung
- Bildabgleich (image matching)
  - Augmented Reality.

2.1. SIFT: Scale-Invariant Feature Transform

Mit der Veröffentlichung von SIFT, frei übersetzt: “Größen-unabhängige Merkmalstransformation“, gelang “David Lowe“ 2004 [Lowe (2004)] ein Durchbruch bei Algorithmen zur Extraktion von Bildmerkmalen. Die mit SIFT extrahierten Merkmale sind robust gegenüber:

- Translation (Parallelverschiebung)
- Skalierung (Größenänderungen, english: “scale-invariance“)
- Rotation (english: “rotation-invariance“)
- Belichtungsverhältnis
- und eingeschränkt sogar gegenüber perspektivischer Verzerrungen.

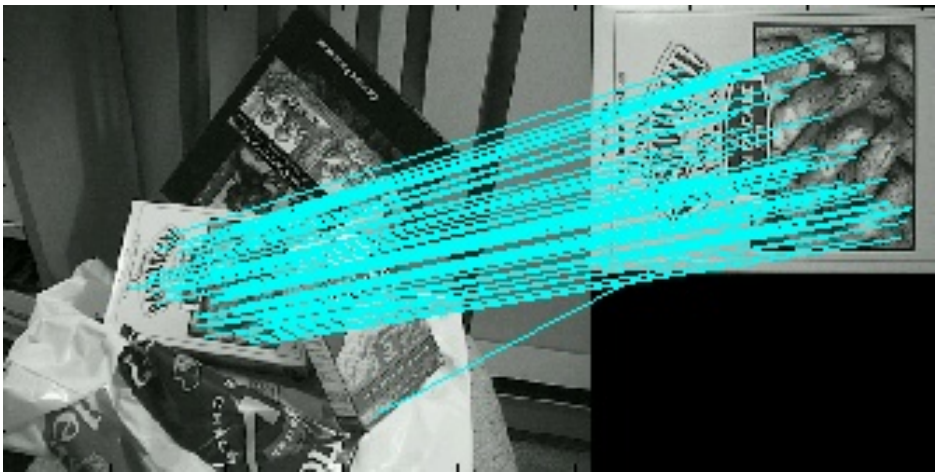

Abbildung 2.1.: SIFT Beispielbild. Quelle: [Lowe (2004)] .

Der Algorithmus lässt sich in vier Hauptschritten unterteilen:

1. **Suche nach potentiellen Merkmalen**
   Als Grundlage dient die sogenannten Bild-Pyramide oder Gauß-Pyramide, die Pyramide ist unterteilt in Oktaven, die Elemente einer Oktave werden durch iteratives glätten des Eingangsbild mit einem Gauß-Filter konstruiert. Für jede neue Oktave wird das Bild in der Größe sowohl in X als auch in Y-Richtung halbiert. Startwert für die erste Oktave ist das Originalbild, für jede weitere Oktave das jeweils letzte Bild der vorherigen Oktave. Die einzelnen, unterschiedlichen stark geglätteten, Ebenen innerhalb einer Oktave werden Intervall genannt. Durch die Subtraktion benachbarter Intervalle wird die sogenannten *“Difference of Gaussian (DoG)“* Pyramide erzeugt (siehe Abb. 2.2). Ist ein Punkt in der DoG-Pyramide ein lokales Minimum oder ein lokales Maximum (siehe Abb. 2.9), so ist dieser Punk ein Kandidat für ein Bildmerkmal. Durch die Analyse mit Hilfe der Pyramide erreicht SIFT die Unabhängigkeit gegenüber Größenänderungen von Bildmerkmalen.
2. **Unterdrücken von schwachen Bildmerkmalen**
   Im zweiten Schritt werden Kandidaten die in Regionen mit einem niedrigen Kontrast oder entlang von Kanten liegen entfernt.

3. **Bestimmung der Hauptorientierung** In diesem Schritt wird jedem Bildmerkmal das den Stabilitätsanforderungen aus Schritt zwei stand gehalten hat, eine Orientierung zugewiesen. Die Orientierung ergibt sich aus dem Helligkeitsänderungen in einer Region um das Bildmerkmal.

4. **Extraktion des Merkmalsvektor**
   Im letzten Schritt wird aus den Bildpunkten einer quadratischen Region um den Merkmalspunkt der Merkmalsvektor erstellt. Die Region wird an der Orientierung ausgerichtet, damit ist der Merkmalsvektor robust gegenüber Drehungen des Bilds.

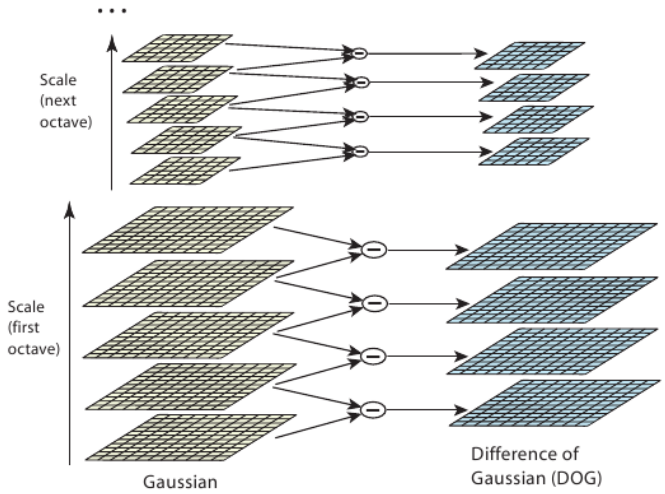

Abbildung 2.2.: SIFT Difference of Gaussians Bildpyramide. Quelle: [Lowe (2004)] .

2.2. SURF: Speeded Up Robust Features

frei übersetzt: „Beschleunigte, robuste Merkmale“, ist ein von “Herbert Bay u. a.“ 2006 [Bay u. a. (2006)] vorgestellter Merkmalsextraktor (english: feature detector). SURF ist in seiner Funktionsweise stark an SIFT angelehnt, durch die, im Vergleich zu SIFT, noch radikalere Approximierung durch einen Hesse-Matrix basierenden Ansatz, der den Einsatz von Box-Filtern und Integralbildern erlaubt, erreicht SURF eine noch bessere Performance (vgl. (Bay u. a., 2006, Seite 8)). Der SURF-Algorithmus kann Grundsätzlich in zwei Aufgaben unterteilt werden. Der SURF-Detektor findet Interessenpunkte in einem Bild, der SURF-Deskriptor extrahiert zu jedem gefunden Merkmal einen Vektor der die Umgebung des SURF-Merkmals beschreibt.

Abbildung 2.3.: SURF Beispielergebnis. In das Bild wurden die 25 stärksten SURF-Merkmale gezeichnet. Rot: helles Merkmal auf dunkleren Untergrund (Vorzeichen des Laplacian negativ), Blau: dunkles Merkmal auf helleren Untergrund (Vorzeichen des Laplacian positiv). Die Quadrate um die einzelnen Merkmale sind die jeweiligen Bildausschnitte aus denen der SURF-Deskriptor extrahiert wird.

2.2.1. Integralbild

Integralbilder sind die Basis für fast alle Funktionen sowohl im SURF-Detektor als auch im SURF-Deskriptor, sie wurden 2001 von “Paul Viola“ und “Michael Jones“ zum ersten mal in der Bildverarbeitung eingesetzt [Viola u. Jones (2001)] . Der Wert für einen Bildpunkt in einem Integralbild ist die Summe aller Bildpunkte in dem aufrechten Rechteck zwischen dem Bildursprung und dem korrespondierendem Bildpunkt im Eingangsbild. Bei einem gegebenen Eingangsbild I und den Bildkoordinaten (x, y) ergibt sich für das Integralbild I_Σ folgende Definition:

$$I_\Sigma(x, y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j). \tag{2.1}$$

Mit Hilfe von Integralbildern lässt sich die Summe aller Bildwerte in beliebig großen, aufrechte Rechtecke, in konstanter Zeit, mit nur drei Additionen, nach folgender Formel berechnen:

$$\Sigma = I_\Sigma(D) - I_\Sigma(A) - I_\Sigma(C) + I_\Sigma(B). \tag{2.2}$$

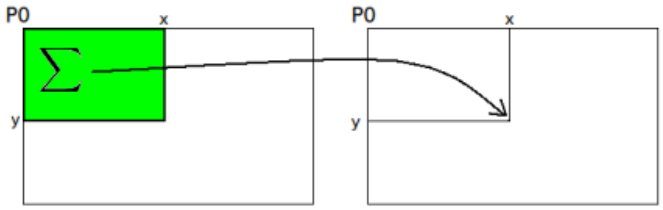

Abbildung 2.4.: Konstruktion eines Integralbilds.

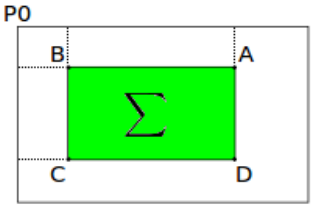

Abbildung 2.5.: Berechnung von Pixelsummen mit Hilfe eines Integralbilds.

Die Im Vergleich zu anderen aktuellen Algorithmen sehr gute Performance von SURF, ist zu einem groß Teil auf die Verwendung von Integralbildern zurückzuführen, da die in SURF verwendeten Box-Filter (siehe z.B. Abbildung 2.6) unabhängig von Ihrer Größe sehr effizient berechnet werden können.

2.2.2. Beschleunigter Hesse-Detektor

Der SURF-Detektor verwendet zum auffinden von Interessenpunkten die Determinante der Hesse-Matrix. Für einen gegebenen Bildpunkt (x, y) und den "scale" σ ist die Hesse-Matix definiert durch:

$$\mathcal{H}_{(x,y,\sigma)} = \begin{bmatrix} L_{xx}(x,y,\sigma) & L_{xy}(x,y,\sigma) \\ L_{xy}(x,y,\sigma) & L_{yy}(x,y,\sigma) \end{bmatrix}. \tag{2.3}$$

Wobei $L_{xx}(x, y, \sigma)$ eine Faltung des Eingangsbild mit der zweiten Ableitung der Gauß-funktion $\frac{\partial^2}{\partial x^2} g(\sigma)$ repräsentiert. $L_{xy}(x, y, \sigma)$ und $L_{yy}(x, y, \sigma)$ sind analog dazu definiert. Für die Faltung eines zweidimensionalen Bildes werden die kontinuierlichen Funktionen $L_{xx}(x, y, \sigma)$, $L_{xy}(x, y, \sigma)$ und $L_{yy}(x, y, \sigma)$ üblicherweise in Filtermasken diskretisiert (siehe: Abb. 2.6 Obere Reihe). Bei SURF wird eine noch radikalere Approximationen durch Box-Filter D_{xx}, D_{yy} und D_{xy} verwendet. Die Determinante der so approximierten Hesse-Matrix lässt sich durch:

$$det(\mathcal{H}_{approximation}) = D_{xx}D_{yy} - (\omega D_{xy})^2 \tag{2.4}$$

berechnen. Um die *"scale-invariance"* sicherzustellen werden die Ergebnisse der drei Box-Filter im Bezug auf ihrer Größe Normiert. Der Gewichtungsfaktor $\omega = 0.912\ldots \simeq 0.9$ gleicht den Unterschied zwischen dem Ergebnis der ursprünglichen Filtermasken und dem Ergebnis der Approximationen an. Genau genommen variiert ω abhängig von der Größe der approximierten Filter, in SURF ist der Wert aber konstant gewählt. Ist der Wert der Determinante positiv so liegt ein lokales Maximum vor.

Die Spur der Matrix:

$$Spur(\mathcal{H}_{approximation}) = D_{xx} + D_{yy} \tag{2.5}$$

approximiert den Laplace-Operator(Laplacian) und liefert zusätzliche Informationen über den Kontrast des gefunden Punkt. Ist das Vorzeichen positiv handelt es sich bei dem Merkmal um einen dunklen Punkt auf einem helleren Hintergrund, ist das Vorzeichen negativ handelt es sich um einen hellen Punkt auf einem dunkleren Untergrund. An Punkten an denen die Determinante der Hesse-Matrix ein lokales Maximum annimmt, ist der Laplacian entweder streng positiv oder streng negativ, deshalb reicht an dieser Stelle eine einfache Untersuchung, mit einem Schwellwert von null, um die Punkte in zwei Gruppen zu separieren. Das Vorzeichen des Laplacian kann dazu verwendet werden den Abgleich von SURF-Merkmalen zu beschleunigen, da Merkmal mit Unterschiedlichem Vorzeichen des Laplacian keine Übereinstimmung liefern können.

2.2.2.1. Skalenraum

Eine Untersuchung im Skalenraum ermöglicht das Auffinden von Interessenpunkten in unterschiedlichen Größen (*"scale-invariance"*). In der Bildverarbeitung wird der Skalenraum typischerweise als Bildpyramide implementiert. Dafür wird das Bild iterativ mit einem Gauss-Filter geglättet und durch sub-sampling in der Größe halbiert (siehe Abb. 2.2). Im SURF-Algorithmus wird zur Konstruktion des Skalenraum ein anderer Ansatz verfolgt. Im Gegensatz zum SIFT-Ansatz bleibt die Größe des Bildes konstant, alle Berechnung beziehen sich direkt auf das Eingangsbild, bzw. dessen Integralbild Darstellung. Die niedrigste Ebene des Skalenraumes wird mit dem kleinst möglichen Filter, mit einer Kantenlänge von 9 Pixel, erzeugt. Die weiteren Ebenen des Skalenraum werden durch Schrittweise Vergrößerung der verwendete Filter konstruiert (siehe Abb. 2.7). Tabelle 2.1 zeigt die Konfiguration des Skalenraum mit vier Oktaven. Die kleinstmögliche Filtergröße sowie die Schritte in denen der Filter vergrößert werden kann, ergeben sich aus dem Layout der Filter. Die Filter bestehen aus drei

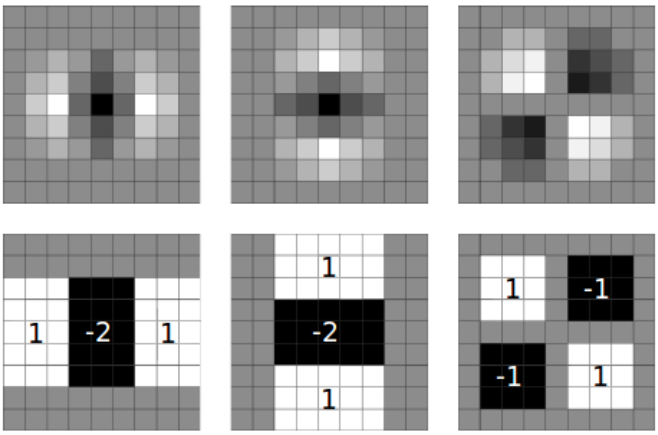

Abbildung 2.6.: Obere Zeile: Diskretisierte Filtermasken für die zweiten partiellen Ableitungen der Gauß-Funktionen in x, y und xy -Richtung L_{xx}, L_{yy} und L_{xy}.
Untere Zeile: Die in SURF verwendete Approximationen D_{xx}, D_{yy} und D_{xy}.
Die grauen Regionen werden mit 0 gewichtet.

Teilen sogenannten *"lobe's"* (deutsch: Lappen, siehe Abb. 2.8). Um das Layout der Filter bei einer Vergrößerung nicht zu verändern, müssen die Teile des Filters gleichmäßig vergrößert werden, die einzelnen Teile des Filters müssen bei jedem Schritt mindestens um zwei Pixel, in jedem Fall aber um eine Gerade Anzahl an Pixel, vergrößert werden, damit Das Vorhandensein eines Zentralpixel sichergestellt ist. Daraus ergibt sich eine minimale Vergrößerung um $2 \times 3 = 6$ Pixel, dies ist gleichzeitig der Vergrößerungsschritt für die erste Oktave, für jede weitere Oktave wird der Vergrößerungsschritt verdoppelt. Für SURF wird ähnlich wie in SIFT, ein *"sampling-step"* eingeführt. Der Standardwert für den *"sampling-step"* der erste Oktave ist zwei. Das bedeutet das, in der ersten Oktave, sowohl in X-Richtung als auch in Y-Richtung nur jeder zweite Pixel ausgewertet wird, Die Anzahl der Ausgewerteten Bildpunkte ist also ein viertel der Anzahl der Pixel im Eingangsbild.

Die Ausgabe des ersten Intervalls der ersten Oktave, mit einer Filtergröße von 9 Pixel, approximiert einen Faltung mit einem Gaussian-Kernel $\sigma = 1.2$. Da der Layout der Filter bei der Vergrößerung konstant bleibt kann der *"scale"* der mit den weiteren Filtern approximiert wird, mit folgender Formel berechnet werden:

$$\sigma_{approximation} = \text{Filtergröße} \times \frac{\text{Initialer-}\textit{"scale"}}{\text{Initiale-Filtergröße}} = \text{Filtergröße} \times \frac{1,2}{9} \tag{2.6}$$

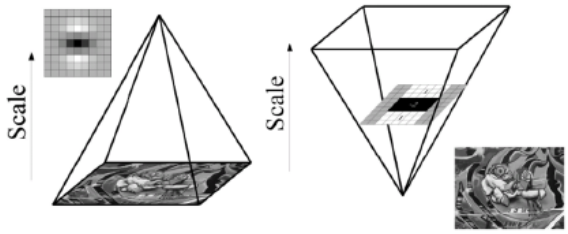

Abbildung 2.7.: Die Skalenraum-Pyramide. Links: Traditioneller Ansatz. Rechts: SURF Ansatz. Quelle: [Bay u. a. (2006)] .

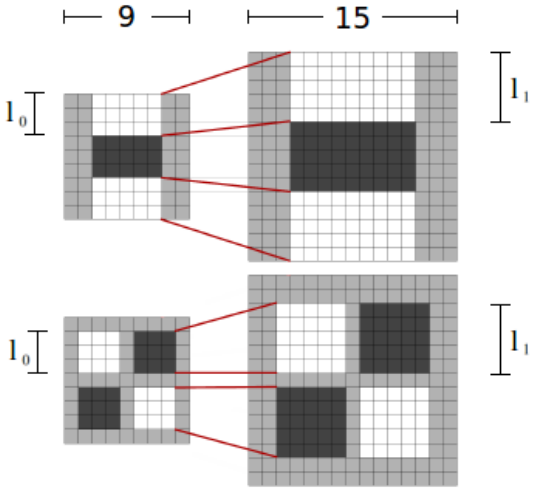

Abbildung 2.8.: SURF-Filter Vergrößerungsschritt. Quelle (modifiziert): [Bay u. a. (2006)] .

2.2.2.2. Auffinden von Interessenpunkten

Damit ein Punkt im Skalenraum als SURF-Merkmal in Betracht gezogen wird, muss zum einen die Determinante der Hesse-Matrix (sieh Formel 2.4) an dieser Stelle größer sein als ein Vorher definiert Schwellwert (SURF-Parameter: hessianThreshold), zum anderen muss der Punkt innerhalb seiner $3 \times 3 \times 3$ Nachbarschaft ein lokales Maximum sein (Abb. 2.9). Dieses Verfahren wird *"3D Non-Maximum Supression"* genannt, die SURF Autoren verwenden hierfür eine besonders effizient Implementierung vgl. [Neubeck u. Gool (2006)] .

Da nicht der gesamte Skalenraum komplett sondern jede Oktave für sich alleine nach Interessenpunkten untersucht wird, können die Punkte in dem jeweils ersten und letzten Intervall einer Oktave keine Interessenpunkte darstellen, da diese Punkte über keine vollständige $3 \times 3 \times 3$ Nachbarschaft verfügen. Diese Punkte dienen nur als Vergleichswerte. Der minimale *"scale"* der mit einer Oktave untersucht werden kann, liegt aufgrund der Interpolation

Oktave Nr.	Filtergrößen	step	scale-min	scale-max	scale-step	realer-scale-min	realer-scale-max
1	9, 15, 21, 27	6	1,2	3,6	0,8	1,6	3,2
2	15, 27, 39, 51	12	2,0	6,8	1,6	2,8	6,0
3	27, 51, 75, 99	24	3,6	13,2	3,2	5,2	11,6
4	51, 99, 147, 195	48	6,8	26,0	6,4	10,0	22,8

Tabelle 2.1.: Beispielkonfiguration des SURF-Skalenraum

(siehe Kapitel 2.2.2.3) der Interessenpunkte genau zwischen dem *"scale"* des ersten und dem des zweiten Intervalls einer Oktave. Der maximale *"scale"* liegt analog dazu zwischen dem vorletzten und dem letzten Intervall einer Oktave. Für den minimalen *"scale"* der ersten Oktave folgt daraus: $\sigma_{approximation} = \frac{9+15}{2} \times \frac{1,2}{9} = 1,6$ (siehe Tabelle 2.1, realer-scale-min und realer-scale-max).

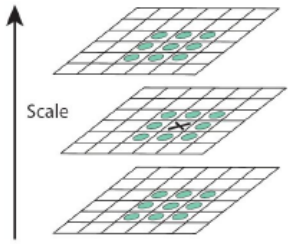

Abbildung 2.9.: 3D-Non-Maximum Suppression, suche nach lokalen Maximums im Skalenraum. Quelle: [Evans (2009)] .

Deshalb ist es notwendig das sich die Oktaven überschneiden, da sonst Lücken im Skalenraum auftreten würden. In den Bereichen des Skalenraumes in denen sich Oktaven überschneiden kann es zu einer doppelten Erkennung von Interessenpunkten kommen.

2.2.2.3. Subpixel genaue Bestimmung der Interessenpunkte

Abschließend wird für die gefunden Interessenpunkte eine Interpolation sowohl für die Bildkoordinaten (x, y) als auch für den *"scale"* (σ) durchgeführt. Dies ist besonders in die höheren Ebenen des Skalenraumes wichtig, da dort der Abstand zwischen den einzelnen Intervallen einer Oktave relativ groß ist. Die Interpolation wird mit der von [Brown u. Lowe (2002)] entwickelten Methode durchgeführt, dabei wird eine Skalenraumkoordinate $F_{interpoliert} = (x, y, \sigma)$ durch eine Taylor-Reihe mit dem detektierten SURF-Merkmal als Entwicklungspunkt angenähert.

Der Beschleunigte Hesse-Detektor liefert als Ergebnis eine Menge folgendes Tupel:

Name	Obligatorisch / Optional	Beschreibung
X	Obligatorisch	X-Koordinate des gefunden SURF-Merkmals
Y	Obligatorisch	Y-Koordinate des gefunden SURF-Merkmals
Scale	Obligatorisch	Größe des gefunden SURF-Merkmals
Strength	Optional	Stärke des gefunden SURF-Merkmals (Determinante der Hess-Matrix)
Laplacian	Optional	Vorzeichen des Laplacian (Spur der Hess-Matrix)

Tabelle 2.2.: Ergebnistupel des beschleunigten Hesse-Detekor

Zusätzlich zu den X und Y- Koordinaten des Merkmals ist der *"scale"*, die Ebene des SURF-Merkmals in der Skalenraum Pyramide, von der auf die Größe des Merkmals geschlossen werden kann, zwingend. Da alle Berechnungen im SURF-Deskriptor abhängig von der Größe des Merkmals durchgeführt werden, dadurch wird die Unabhängigkeit gegenüber Größenänderungen der Interessenpunkte (*"scale-invariance"*) des SURF-Algorithmus erreicht. *"Strength"* die stärke eines SURF-Merkmals ermöglicht eine qualitative Beurteilung des gefunden Merkmals. Umso größer dieser Wert ist, umso besser ist dieser Punkt als SURF-Merkmal geeignet. Es wird in der offiziellen Beschreibung des SURF-Algorithmus nicht explizit definiert, im Allgemeinen, wie auch in allen im Rahmen dieser Arbeit untersuchten quelloffen SURF-Implementierungen, wird davon ausgegangen das es sich bei der Stärke um die Determinante der Hesse-Matrix handelt.

2.2.3. SURF-Deskriptor

Der SURF-Deskriptor beschreibt, ähnlich wie SIFT, die Verteilung der Grauwert-Intensität in einem, von der größe *"scale"* des gefunden Merkmals abhängigen, Ausschnitt um das gefundene Merkmal. Zur Beschreibung der Merkmale greif SURF, auf die schon aus dem Hesse-Detektor bekannten Integralbilder in Kombination mit Haar-Filtern zurück.

Haar-Filter sind einfache Filter mit denen sich Gradienten in X und Y Richtung bestimmen lassen. Unter Gradienten versteht man im Bereich der Bildverarbeitung Vektoren die Helligkeitsänderungen beschreiben. Haar-Filter lassen sich ähnlich wie die Box-Filter des Hesse-Detektors sehr effizient mit Hilfe von Integralbilder berechnen.

Die Erstellung des Deskriptor gliedert sich in zwei Aufgaben. Zuerst wird jedem Merkmal eine Eindeutige Orientierung zugewiesen, danach werden aus einem rechteckigen Bereich, der an der zuvor bestimmten Orientierung ausgerichtet wird, die einzelnen Komponenten

Abbildung 2.10.: Haar-Filter. Links: Filter für X-Richtung, Rechts: Filter für Y-Richtung.

des Deskriptor-Vektors extrahiert. Für Anwendung die nicht auf *"rotation-invariance"* angewiesen sind, wird in SURF der Parameter "upright SURF (U-SURF)" eingeführt. Wird dieser Parameter aktiviert, wird der erste Schritt im SURF-Deskriptor übersprungen, und dem Merkmal eine Orientierung von null zugewiesen, was einer Ausrichtung entlang der X-Achse entspricht. "Upright SURF" verringert den Berechnungsaufwand und bietet immer noch eine Invarianz gegenüber Rotationen von bis zum $+/-15°$.

2.2.3.1. Bestimmung der Orientierung

Um Invarianz gegenüber Bilddrehungen zu erreichen wird jedem SURF-Merkmal eine reproduzierbare Orientierung zugewiesen. Dazu wird für eine Reihe von Bildpunkten, in einem Kreis dessen Mittelpunkt auf dem Merkmal liegt, und der einen Radius von 6σ hat, die Ergebnisse für Haar-Filter der Größe 4σ in X und Y-Richtung berechnet, die Bildpunkte werden mit einem Abtastschritt von σ bestimmt. Die Ergebnisse werden mit einem Gauß-Filter mit der Standardabweichung = $2{,}5\sigma$, zentriert über dem Merkmalspunkt, gewichtet, σ steht an dieser Stelle für den *"scale"* des jeweiligen Merkmals. Die Ergebnisse werden dann als Punkte in einem zweidimensionalen Vektorraum betrachtet, der X-Wert des Punktes ist das gewichtet Ergebnis des Haar-Filters in X-Richtung, der Y-Wert analog dazu das Ergebnis mit dem Haar-Filter in Y-Richtung. Anschließen wird ein Fenster der Größe $\frac{\pi}{3}$ über den Kreis gleiten gelassen[1], die X und Y-Werte werden aufsummiert und bilden einen Vektor. Die Orientierung des Merkmals ist die Richtung des vom Betrag her größten Vektors (siehe Abb. 2.11).

2.2.3.2. Zusammensetzung des Deskriptor

Zur Extraktion des Deskriptor wir eine quadratische Region, zentriert auf dem Merkmalspunkt platziert und an der im vorherigen Schritt ermittelten Orientierung ausgerichtet. Die Kantenlänge dieser Region beträgt 20σ (20 mal dem *"scale"* des Interessenpunktes). Diese Region wird in 4×4 Unterregionen aufgeteilt, in jeder Unterregion werden für $5 \times 5 = 25$ gleichmäßig Verteilte Bildpunkte die Summen und die Summe der Beträge für die beiden,

[1] Die SURF-Autoren machen keine Angabe darüber in welcher Schrittgröße sie das Fenster über den Kreis gleiten lassen

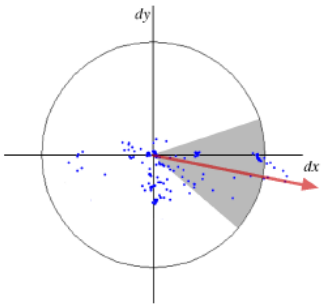

Abbildung 2.11.: Bestimmung der Merkmals-Orientierung. Der graue Bereich kennzeichnet ein mögliches Orientierungsfenster der Größe $\frac{\pi}{3}$. Die blauen Punkte sind die Ergebnisse der Haar-Filter. Der Rote Pfeil zeigt die gefundene Orientierung. Quelle: [Bay u. a. (2006)] .

aus dem vorherigen Schritt bekannten, Haar-Filter mit einer Kantenlänge von 2σ aufsummiert. Dabei erfolgt, wie bei der Bestimmung der Orientierung, eine Gewichtung mit einem auf den Merkmalspunkt zentriertem Gauss-Kernel, mit der Standardabweichung = 3.3. Für den Vektor einer Unterregion ergibt sich daraus folgender Aufbau:

$$\mathcal{D}_{Unterregion} = \left[\sum dx, \sum |dx|, \sum dy, \sum |dy|\right] . \tag{2.7}$$

Für den gesamten Deskriptor-Vektor ergibt sich, bei 16 Unterregionen, eine Länge von $16 \times 4 = 64$. Abb. 2.12 veranschaulicht die Konstruktion des SURF-Deskriptor. Die SURF-Autoren definieren zusätzlich ein erweiterte Version dieses Deskriptors (SURF-Parameter: “extended-Descriptor“), dabei werden die Summen für dx und dy weiter aufgeteilt, jeweils Abhängig vom Vorzeichen des Anderen. Dies resultiert in einem doppelt so langen Vektor, der unverwechselbarer ist als der Standard Deskriptor, allerdings erhöht sich der Rechenaufwand für den Vergleich der Vektoren. Ein vergleich der

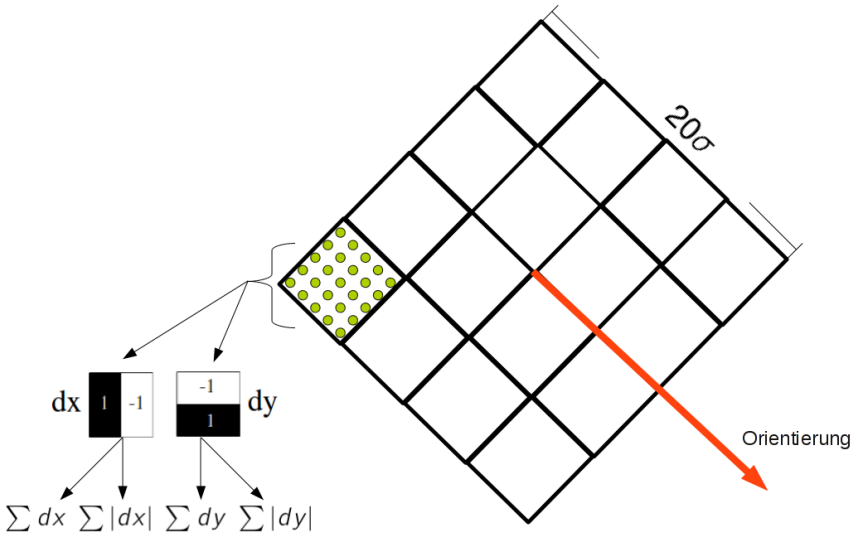

Abbildung 2.12.: Komponenten des SURF-Deskriptor.

2.2.4. SURF-Implementierungen

In diesem Kapitel werden die im weiteren Verlauf dieser Arbeit untersuchten SURF-Implementierungen kurz Vorgestellt und die grundlegenden Anforderungen an eine Implementierung als Basis des Lokalisierungssystem definiert.

Kriterien für die Auswahl der SURF-Implementierung:

Betriebssystem
Die Referenzimplementierung des Lokalisierungssystem wird zwar auf einem Linux-System erfolgen, für zukünftige Anwendungen ist es dennoch wünschenswerte das ein Einsatz auch auf einem Windows Betriebssystem erfolgen könnte.

Quelloffen
Eine “open source“ Implementierung ermöglicht die Portierung auf andere Betriebssystem und Architekturen, und den bei rechenintensiven Bildverarbeitungsalgorithmen, oft großen Vorteil, der Optimierung auf eine bestimmte Prozessorarchitektur zur Kompilierzeit. Im Fall des Lokalisierungssystems gilt dies im besonderen Maße, da es sich dabei nicht um eine typische SURF-Anwendungen handelt, und deshalb davon auszugehen ist das Detailanpassungen zwingend notwendig seien werden, oder zumindest zu erheblichen Vorteilen führen könnten.

Performance

Die Anforderungen die das Lokalisierungssystem an die Rechenleistung stellt, sollten gerade im Hinblick auf den Einsatz auf eingebetteten Systemen, so gering wie möglich gehalten werden. Sollten sich im Verlauf der Untersuchungen große Unterschiede im Bezug au die Performance der Implementierungen offenbaren, so wäre dies ein starkes Argument für oder gegen eine Implementierung.

2.2.4.1. LibSURF

LibSURF ist die Referenzimplementierung der SURF Autoren. Die Library steht für den nicht kommerziellen Einsatz frei zur Verfügung. Entwickelt wurde die SURF Referenzimplementierung in C++, sie liegt allerdings nur als vorkompiliertes Binärpaket für Linux 32-bit und Windows-32 vor. Alle Untersuchungen und Angaben in dieser Arbeit beziehen sich, falls nicht abweichend angegeben, auf die Version 1.0.9 der libSURF.

URL: http://www.vision.ee.ethz.ch/~surf/download.html

2.2.4.2. OpenSURF

OpenSURF ist eine quelloffen Nachbau des SURF Algorithums, veröffentlicht unter der GNU Public License. Grundlage der Implementierungen ist eine tief gehende Analyse des SURF Algorithmus *"Notes on the OpenSURF Library"* [Evans (2009)] . Die OpenSURF Library ist wie Ihr Vorbild in C++ entwickelt. Der Autor orientiert sich bei dem Design an der Referenzimplementierung und versucht den beschleunigten Hesse-Detektor auf Basis der SURF Dokumentation so exakt wie möglich zu implementieren, für das Beschreiben der gefunden Bildmerkmale verwendet OpenSURF allerdings nicht den original SURF-Deskriptor sondern implementiert den *"modified upright SURF (MU-SURF)"* [Agrawal u. a. (2008)] Deskriptor. Für das laden von Bilder greift der Autor auf die OpenCV Library zurück. OpenSURF kann sowohl unter Linux als auch unter Windows kompiliert werden. Für das .NET Framework besteht zusätzlich eine offizielle Portierung nach C#.

URL: http://www.chrisevansdev.com

2.2.4.3. OpenCV

Open Source Computer Vision ist eine unter der BSD lizenzierte Softwarebibliothek zur Bildverarbeitung in Echtzeit. Durch die Lizenzierung unter der BSD ist die OpenCV Library sowohl für akademische als auch kommerzielle Nutzung kostenfrei. OpenCV wurde 1999 als Forschungsprojekt von Intel gestartet. OpenCV ist sowohl für Unix-artige Betriebssystem

als auch für Microsoft Windows verfügbar. Mit dem Ziel einer größtmöglichen Verbreitung, wurden die ersten Versionen in C entwickelt und Wrapper zu vielen anderen Programmiersprachen gepflegt, darunter C++, C#, Python, Ruby und Java. Seit Version 2, veröffentlicht im Oktober 2009, werden die meisten Neuerungen in C++ entwickelt. Ziel ist es, die Menge an Quelltext zu reduzieren und das Auftreten häufiger Programmierfehler wie Speicherlecks zu verringern. Nachteil dieser Umstellung ist, das es deutlich schwerer ist Wrapper in anderen Programmiersprachen für C++ zu erstellen, als es im Vergleich zu C ist. Deshalb fehlen in den Wrappern gerade neuere OpenCV Funktionen. Als Stärken von OpenCV sind, die hohe Performance, die große Anzahl an verfügbaren Bildverarbeitungsalgorithmen und die große Entwickler- und Benutzergemeinde zu nennen. OpenCV bietet die Möglichkeit durch den Einsatz der *IPP: Intel Integrated Performance Primitives* [IPP (2011)] und der *TBB: Intel Threading Building Blocks* [TBB (2011)] die ohnehin schon gute Performance noch deutlich zu steigern. Die TBB Library wird von Intel unter einem dualen Lizenzierungsmodell veröffentlicht, zum einen in einer kommerzielle Version die Support beinhaltet, aber im Gegensatz zur IPP, zum anderen auch unter einer GPL kompatiblen Lizenz, die eine kostenfreie kommerzielle Nutzung erlaubt.

URL: http://opencv.willowgarage.com/wiki/

2.2.4.4. LTI-Lib-2

Die LTI-Lib wurde ursprünglich am Lehrstuhl für Technische Informatik (LTI) der RWTH-Aachen University entwickelt. Ziel war eine objektorientierte C++ Softwarebibliothek für Aufgaben aus dem Bereich Bildverarbeitung und maschinellem Sehen. Die Version 2 der LTI-Lib ist wie OpenCV unter der BSD Lizenz veröffentlicht und ist damit auch für kommerzielle Zwecke frei nutzbar. Kompiliert werden kann die LTI-Lib unter Linux mit GCC oder auf Windowssystemen mit Visual C++. Besonders geachtet wurde bei der Entwicklung der LTI-Lib auf die konsequente Durchsetzung des objektorientierten Ansatz und eine außergewöhnliche hohe Qualitätssicherung. Der gravierendste Nachteil im Vergleich zu z. B. der OpenCV Bibliothek ist die nur sehr kleine Gruppe an Entwicklern und Benutzern der LTI-Lib. Es werden zwar in regelmäßigen Abständen, neue Versionen mit Korrekturen oder Fehlerbehebungen veröffentlicht, um neue Funktionen oder neue Algorithmen ist die Library aber seit geraumer Zeit nicht erweitert worden. Im Rahmen dieser Arbeit wurde die LTI-LIB-2 Build: 110228 verwendet.

URL: http://www.ie.itcr.ac.cr/palvarado/ltilib-2

2.2.4.5. Zusammenfassung

Die LibSURF library der SURF-Autoren wird sehr wahrscheinlich nicht für die Realisierung des Lokalisierungssystems in frage kommen da sie nur in Binärform vorliegt. Eine endgültige Aussage, welche der anderen Implementierungen am besten als Grundlage für das Lokalisierungssystems geeignet ist, findet auf Basis der Untersuchungsergebnisse aus Kapitel 3.2, in Kapitel 4.1 statt.

Implementierung	Quelloffen / Binär	Lizenz	Betriebs-system	Version
LibSURF	Binär	as/is	Linux 32bit, Windows 32bit	1.0.9
OpenSURF	Quelloffen	GPL	Linux, Windows	Build 27/05/2010
OpenCV	Quelloffen	BSD	Linux, Windows	2.1
LTI-Lib-2	Quelloffen	BSD	Linux, Windows	Build 110228

Tabelle 2.3.: Übersicht SURF-Implementierungen.

3. Optimale SURF-Marken

Die von [Schweiger u. a. (2009)] entwickelten *"Maximum Detector Response Markers for SIFT and SURF"* sollen in einem Bild das mit dem jeweiligen Merkmalsextraktor analysiert wird, einen maximalen, oder zumindest deutlich höheren, Ausschlag erzeugen, als es natürliche Merkmale tun. Im Bezug auf SURF bedeutet das, einen möglichst großen Wert der approximierten Hesse-Determinante zu erzielen. Da sowohl SIFT als auch SURF nur auf Grauwertbildern operieren, gibt es von den Marken jeweils zwei Versionen, eine positive und eine negative (siehe Abb. 3.1). Im weiteren Verlauf dieser Arbeit werden die Marken als: "Optimale-SURF-Marken" bezeichnet.

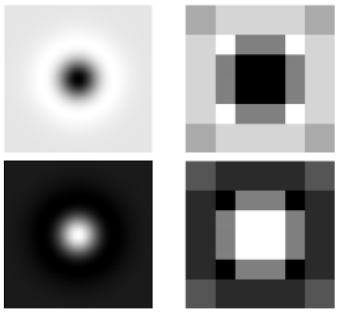

Abbildung 3.1.: Optimale SIFT und SURF Marken. Links: SIFT, Rechts: SURF

Eine Unterscheidung der Marken ist bei beiden Algorithmen über den jeweiligen Merkmals-Deskriptoren möglich. Bei SURF besteht außerdem die Möglichkeit, die Marken anhand des Vorzeichen des Laplacian zu unterscheiden (siehe Kapitel 2.2.2, Formel 2.5). Die Entwickler der Marken führen dabei folgende Terminologie ein:

Name	Übersetzung	Beschreibung	Vorzeichen des Laplacian	Abbildung 3.1
LIGHT MARKER	Helle-Marke	Dunkler Punkt auf hellem Untergrund	positiv	Rechts oben
DARK MARKER	Dunkle-Marke	Heller Punkt auf dunklem Untergrund	negativ	Rechts unten

Tabelle 3.1.: Optimale-SURF-Marken

Für die Herleitung des Design der Marken, sei an dieser Stelle auf die Originaldokumentation der Entwickler verwiesen.
URL: http://www.lmt.ei.tum.de/team/florian/markers

3.1. Signatur

In Ihrer Arbeit führen Schweiger u. a. (2009) den Begriff Signatur ein. Die Signatur ist eine Untermenge des SURF-Deskriptor. Wie in Kapitel 2.2.3.2 beschrieben, ist die Kantenlänge des Fensters aus dem der SURF-Deskriptor extrahiert wird $20 \times \sigma$, wobei σ für den *"scale"* des gefunden Merkmals steht. Der *"scale"* in dem die Optimalen-SURF-Marken gefunden werden ist, wie die folgenden Untersuchungen bestätigen werden, ca. $\frac{1}{8}$ der Kantenlänge der Marken im Bild. Ein Optimalen-SURF-Marke mit einer Kantenlänge von 16 Pixel, wird vom SURF-Detektor also ca. in einem *"scale"* von zwei gefunden.

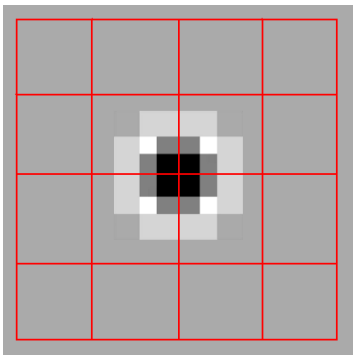

Abbildung 3.2.: Signatur der Optimalen-SURF-Marken

Nummeriert man die Unterregionen des SURF-Deskriptors, von links oben nach rechts unten, bei eins startend durch, beeinflussen die Optimalen-SURF-Marken nur die Unterregionen 6, 7, 10 und 11 (siehe Abb. 3.2). Da jede Unterregionen ein Vektor der Länge vier ($\mathcal{V}_{Unterregionen} = \left[\sum dx, \sum |dx|, \sum dy, \sum |dy|\right]$) zum Deskriptor beiträgt, hat die Signatur eine Länge von 16. Zusätzlich wird eine *"Version independent signatur (vis)"* definiert, für diese Signatur werden nur die Summe der Beträge von dx und dy berücksichtigt ($\mathcal{V}_{Unterregionen} = \left[\sum |dx|, \sum |dy|\right]$), die *"vis"* hat somit eine Länge von acht. Da die *"vis"* für die helle und für die dunkle Optimale-SURF-Marke identisch ist kann sie zur Identifikation beider SURF-Marken benutzt werden.

3.2. Versuchsreihe SURF-Marken

In diesem Kapitel werden die zuvor vorgestellten optimalen SURF-Marken in mehreren Testreihen mit Hilfe synthetischer Testbildern untersucht. Jedes Testszenario wird mit jeder der in Kapitel 2.2.4 vorgestellten SURF-Implementierung analysiert. Zu diesem Zweck wurde eine Testumgebung auf Basis der LTI-Lib entwickelt, die es ermöglicht den Optimalen-SURF-Marken folgenden Störeffekten auszusetzen:

- Größenänderung
- Rotation
- Perspektivisches Kippen
- Additives weißes gaußsches Rauschen
- Kontrast (Belichtung)

Dabei werden folgende Messgröße erfasst:

- SURF-Detektor Resonanz
  Wie groß ist der Ausschlag im SURF-Detektor ?
  vgl. Kapitel 2.2.2
- Subpixel genaue Lokalisierung
  Wie beeinflussen die Störeffekte die Genauigkeit mit der die Bildkoordinate der SURF-Marke bestimmt wird ?
  vgl. Kapitel 2.2.2.3
- Scale
  In welchem *“scale“* (in welcher Größe) wird die Marke gefunden ?
  vgl. Kapitel 2.2.2.1
- Orientierung
  Welche Orientierung wird der Optimalen-SURF-Marke zugeordnet ?
  vgl. Kapitel 2.2.3.1
- Signatur der Optimalen-SURF-Marke
  Wie groß ist die euklidische Distanz zu der vorher ermittelten Referenzsignatur der Optimalen-SURF-Marke ?
  vgl. Kapitel 3.1

Die erzeugten Testbilder habe eine Größe von 513x513 pixel. Die SURF-Marken werden mittig auf dem Testbild platziert. Der Mittelpunkt der Marken befindet sich konstant bei $P_M(256,5;256,5)$.

Alle Untersuchungen wurden sowohl mit der hellen SURF-Marke, als auch mit der dunklen SURF-Marke durchgeführt (siehe Abb. 3.1). An dieser Stelle werden nur die Ergebnisse der hellen SURF-Marke vorgestellt, sollte nicht explizit darauf hingewiesen werden, lieferte der Test mit der dunklen SURF-Marke ein äquivalentes Ergebnis.

Falls nicht abweichend angegeben wurden alle Versuchsreihen mit folgenden SURF-Parametern durchgeführt:

Name	Wert
Octaves	3
Intervals	4
Initial-Sample-Step	2
Initial-Filter-Size	9
Initial-Kernel-Increment-Step	6
Extended-Descriptor	false
Descriptor-Window-Size	20
Upright-SURF	false

Tabelle 3.2.: Standard SURF-Parameter für die Versuchsreihen

Ein Überblick über die entwickelte Testumgebung liefert Anhang B.

Ziel der Untersuchungen ist es, zum einen Erfahrungswerte mit dem Optimalen-SURF-Marken zu sammeln, zum anderen sollen Unterschiede, Schwächen und Stärken der verwendeten SURF-Implementierungen offen gelegt werden. Die Ergebnisse der verschiedenen Implementierungen können aufgrund implementierungs-spezifischer Details wie z. B. unterschiedlicher interner Darstellung von Bilder oder unterschiedlicher Normierung, nur Qualitativ jedoch nicht Quantitativ miteinander verglichen werden. Um die Ergebnisse der untersuchten Implementierungen im Bezug auf den Ausschlag, den die Optimalen-SURF-Marken im Hesse-Detektor erzeugen, beurteilen zu können, wurde auf jede Implementierung das aus Kapitel 2.2 bekannte Sonnenblumenfeld Bild Abb. 2.3 angewendet, und der Maximale sowie der Durchschnitt über die zehn stärksten SURF-Merkmale ermittelt (siehe Tabelle 3.3).

Implementierung	Max.	Durchschnitt
libSurf	127,20	105,30
OpenSURF	0,01657	0,01452
OpenCV	33.053,4	23.412,2
LTI-Lib-2	569,18	470,94

Tabelle 3.3.: Vergleichswerte für den Hesse-Detektor-Ausschlag

3.2.1. Größenänderung

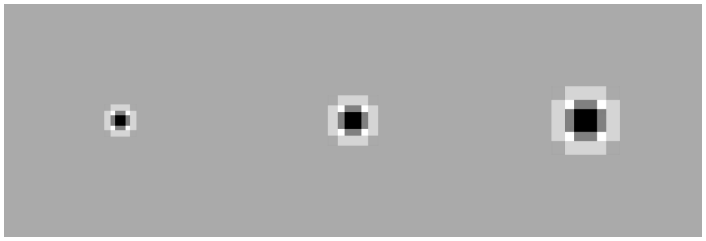

Abbildung 3.3.: Versuchsreihe: Größenänderung Beispiele. Von links nach Rechts: Helle-Optimale-SURF-Marke der Kantenlänge 71pixel, 111pixel und 151pixel.

Die Kantenlänge der Optimale-SURF-Marke wird in Schritten von zwei Pixel vergrößert. Als Größe für den ersten Durchlauf, wurde die minimale Größe der SURF-Filter von neun Pixel Kantenlänge gewählt. Für den ersten Test wurde Abweichen von der Standardkonfiguration, wenn es die Implementierung erlaubt, der Parameter *"Upright-SURF"* aktiviert, um mögliche Störungen durch unterschiedlich erkannte Orientierungen zu vermeiden.

Besonderes Interesse gilt bei dieser Testreihe, neben der Resonanz die im SURF-Detektor hervorgerufen wird, der minimalen bzw. maximalen Größe der Marken im Bild. Bei den Untersuchungen die [Schweiger u. a. (2009)] mit OpenCV durchgeführt haben, traten an zwei Stellen im Skalenraum, zum einen bei einer Größe von ca. 50 Pixel und zum anderen bei einer Größe von ca. 100 Pixel ein kompletter Einbruch des Detektor-Ausschlags ein vgl. (Schweiger u. a., 2009, Seite 6).

Für die minimale Marker-Größe wird ein Wert von $1,6 \times \frac{9}{1,2} \simeq 12$ erwartet, für die maximale Größe wird, bei einer Konfiguration mit 3 Oktaven, ein Wert von $11,6 \times \frac{9}{1,2} \simeq 90$ erwartet.

Die Abbildungen 3.4 und 3.5 Zeigen den Detektor-Ausschlag für die vier untersuchten Implementierungen. Bei allen Implementierungen erzeugt die Optimale-SURF-Marke eine extrem große Resonanz, der Wert liegt bei dem drei bis vierfachen des maximalen Ausschlag, für natürliche SURF-Merkmale, aus dem Referenzbild (vgl. Tabelle 3.3). Bei der minimalen und maximalen Größe der SURF-Marken zeigt nur OpenSURF das vorhergesagte Verhalten. Die

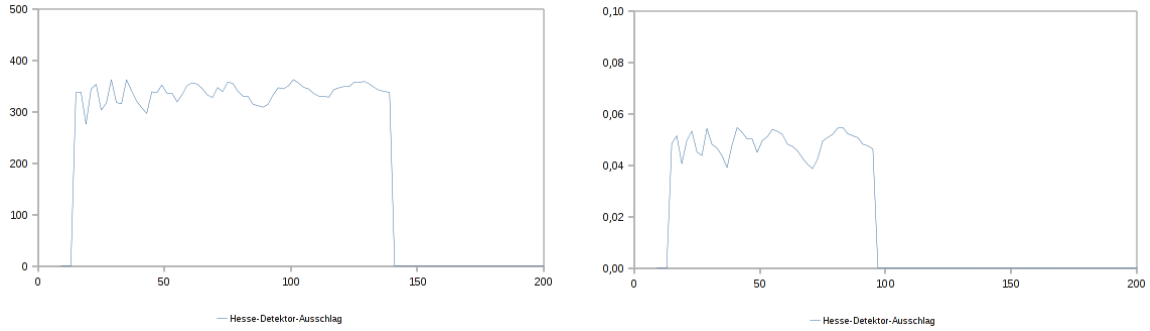

Abbildung 3.4.: Versuchsreihe: Größenänderung. Links: libSurf, Rechts: OpenSURF

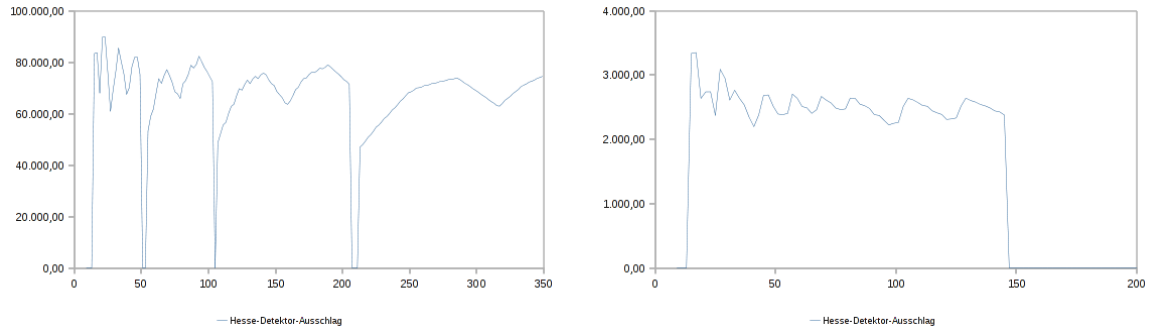

Abbildung 3.5.: Versuchsreihe: Größenänderung. Links: OpenCV, Rechts: LTI-Lib-2

Untersuchung mit OpenCV bestätigt das Ergebnis von Schweiger u. a., und weißt auf eine Fehlerhaft Implementierung des Skalenraum hin. Ein weiteres Indiz dafür, sind die Standardparameter der OpenCV Library SURF-Implementierung für den Skalenraum, standardmäßig arbeitet OpenCV mit fünf Oktaven und zwei Intervallen pro Oktave (siehe Anhang A). Nach Bay u. a. (2006) ist in SURF eine Oktave mit weniger als drei Intervallen nicht zulässig, da die jeweils erst und letzte Intervalle einer Oktave kein Ergebnis tragen kann, sondern nur als Vergleichswerte herangezogen werden (vgl. Kapitel 2.2.2.2). LibSurf und die LTI-Lib-2 zeigen betreffend der maximalen und minimalen Marker-Größe, mit der verwendeten SURF-Konfiguration, ein ähnliches Ergebnis. Dabei handelt es sich allerdings um Zufall, bei Anderne Konfigurationen mit z. B. 4 Oktaven entstehen Unterschiedliche Ergebnisse, eine Analyse des Quelltext der LTI-Lib-2 zeigt, das auch hier Mängel bei der Implementierung des Skalenraum vorliegen. Zum einen findet keine Überlappung der Oktaven statt zum anderen werden die Oktaven nicht jeweils für sich sondern alle Oktaven zusammen als ein Ergebnisraum auf möglich Interessenpunkte hin untersucht [1]. Warum das Ergebnis der libSurf von dem erwar-

[1] Die Fehlerhafte Implementierung des Skalenraum sowohl in OpenCV als auch in der LTI-Lib-2 resultieren wahrscheinlich aus einer unvollständigen Beschreibung des Algorithmus. Von der original Beschreibung der

teten Ergebnis abweicht, kann an dieser Stelle nicht geklärt werden, da keine Analyse des Quelltext möglich ist.

Der für SURF charakteristische plötzlichen Anstieg des Detektor-Ausschlag, auf annähernd den Maximalen Wert, und der genauso plötzlichen Abfall, ist auf die in SURF verwendete Umsetzung des Skalenraum zurückzuführen. Das letzte und vorletzte Intervalle der höchsten Oktave, wird für die OpenSURF Library, z. B. mit Filter-Kernel der Kantenlänge 75 Pixel und 99 Pixel erstellt. Die Determinante der Hesse-Matrix ist in einem Intervall, für einen Bereich um den Punkt, an dem die Kantenlänge der Box-Filter und die Kantenlänge der SURF-Marken identisch sind, maximal. Für das vorletzte Intervall, mit einer Filtergröße von 75 Pixel, ist das z. B. der Bereich von 70 Pixel bis 90 Pixel, in diesem Bereich trägt dieses Intervall das Gesamtergebnis. Mit steigender Größe nimmt der Ausschlag in diesem Intervall ab, wird die Markengröße erreicht, bei der das letzte Intervall einen höheren Ausschlag erzeugt, als das vorletzte, fällt das Gesamtergebnis auf null, da der immer noch sehr starke Ausschlag im vorletzten Intervall kein lokales Maximum mehr darstellt und der noch stärkere Ausschlag im letzten Intervall nicht in einem Interessenpunkt resultieren kann, da für diesen Punkt innerhalb des Skalenraum keine komplette $3 \times 3 \times 3$ Nachbarschaft vorhanden ist.

Bei einer wiederholten Durchführung der Testreihe, mit deaktiviertem *“Upright-SURF“* Parameter, zeigen alle Implementierung ein, im Bezug auf die gefundene Orientierung, nahezu identisches Verhalten. Abb. 3.6 zeigt Beispielhaft die mit der libSurf gefundenen Orientierungen, abhängig von der Größe der SURF-Marke.

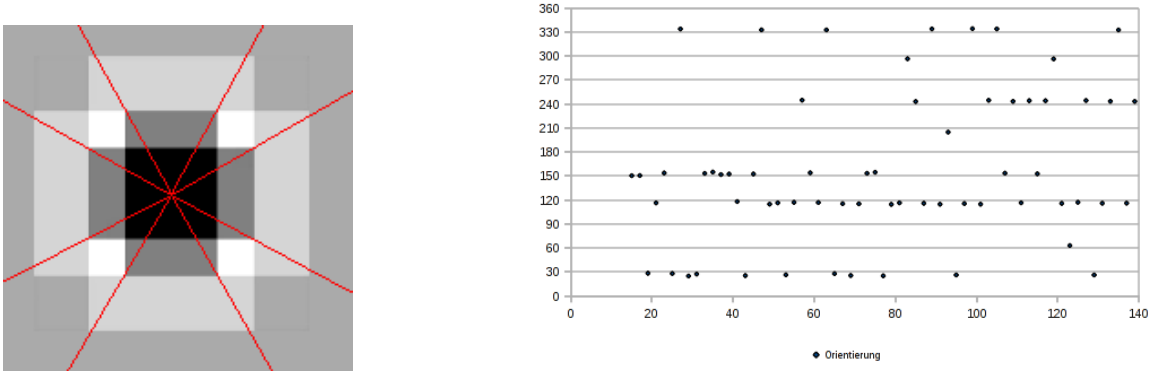

Abbildung 3.6.: Versuchsreihe: Größenänderung, mit libSurf gefunden Orientierungen.

Optimal für ein Vergleich mit der Referenzsignatur sind die Orientierungen: 0°, 90°, 180° und 270°. In Abb. 3.6[Rechts] sind die insgesamt acht gefunden Orientierungen gut zu er-

SURF-Autoren, die von deren Internetseite bezogen werden kann, existieren mindestens zwei Versionen. Die während des Bearbeitungszeitraums dieser Arbeit ausgetauscht wurden. Gerade die oft Fehlerhaft implementierten Details des SURF-Algorithmus, besonders im Bezug auf die Konstruktion des Skalenraum, werden in der ersten Version nur rudimentär beschrieben.

kennen. Abb. 3.6[Links] zeigt die SURF-Marke mit den eingezeichneten Orientierungen. Da das Fenster zur Extraktion des SURF-Deskriptor an der Orientierung ausgerichtet wird, zeigt sich im Vergleich zu der ersten Testreihe eine deutlich größere euklidische Distanz zu der Referenzsignatur der Optimalen-SURF-Marke. Mit eingeschalteter Orientierungserkennung ist die euklidische Distanz zur Referenzsignatur, im Vergleich zu den Untersuchungen mit “upright-SURF“, um den Faktor zehn größer (siehe Abb. 3.7).

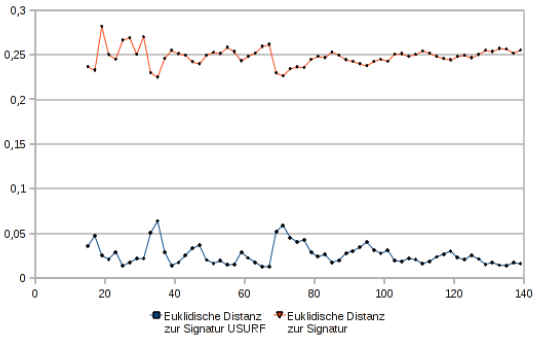

Abbildung 3.7.: Veruchsreihe: Größenänderung, Distanz zur Signatur. Blaue Kurve: euklidische Distanz zur Referenzsignatur mit aktiviertem “upright-SURF“, Rote Kurve: Distanz zu Signatur mit deaktiviertem “upright-SURF“

Die Bildkoordinate der Marke wurde in beiden Testreihen, von allen Implementierungen sehr genau bestimmt. Die maximal festgestellte Abweichung liegt bei 0,7 Pixel. Das gleiche gilt für den detektierten *“scale“*, auch hier liefern alle Implementierungen nahezu identische Ergebnisse, die nur minimal von dem tatsächlichen *“scale“* abweichen.

3.2.2. Rotation

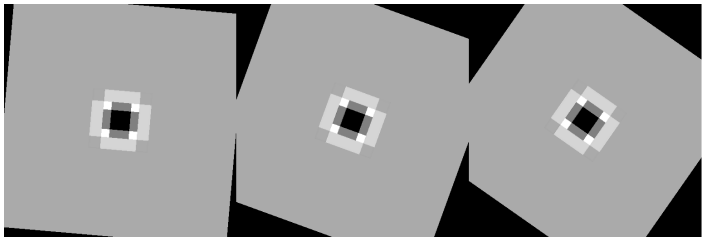

Abbildung 3.8.: Versuchsreihe: Rotation Beispiele. Von links nach Rechts: Helle-Optimale-SURF-Marke im Uhrzeigersinn gedreht um 5°, 20° und 35°.

Bei diesem Versuch wird die Optimale-SURF-Marke in Schritten von 1° um ihrem Mittelpunkt rotiert. Aufgrund des achsensymmetrischen Aufbaus der Marken sollte sich alle 90° eine Wiederholung der Ergebnisse einstellen, es wurde trotzdem eine komplette Drehung der Marke um 360° durchgeführt.

Abbildung 3.9 zeigt den Detektor-Ausschlag für die original Implementierung libSurf und die OpenCV SURF-Implementierung. Abweichend zu den Ergebnissen von Schweiger u. a. (2009), die einen kontinuierlich hohen Detektor-Ausschlag festgestellt haben, wurde in dieser Testreihe für den Detektor-Ausschlag lokale Maxima bei Drehungen von 0° bzw. 360°, 90°, 180° und 270° festgestellt, dazwischen fällt der Detektor-Ausschlag auf ca. 75% des Maximalausschlags ab. Dieses Verhalten konnte mit allen untersuchten SURF-Implementierung nachgewiesen werden. Allerdings liegt auch der minimal festgestellte Detektor-Ausschlag noch deutlich über den Werten für natürliche SURF-Merkmale (vgl. Tabelle 3.3).

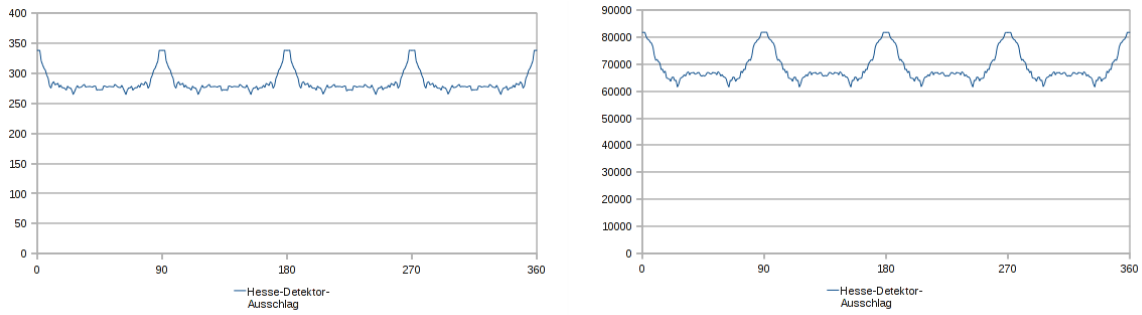

Abbildung 3.9.: Versuchsreihe: Rotation Detektor-Ausschlag Links: libSurf, Rechts: OpenCV

Der *“scale“* verhält sich analog dazu (siehe Abb. 3.10).

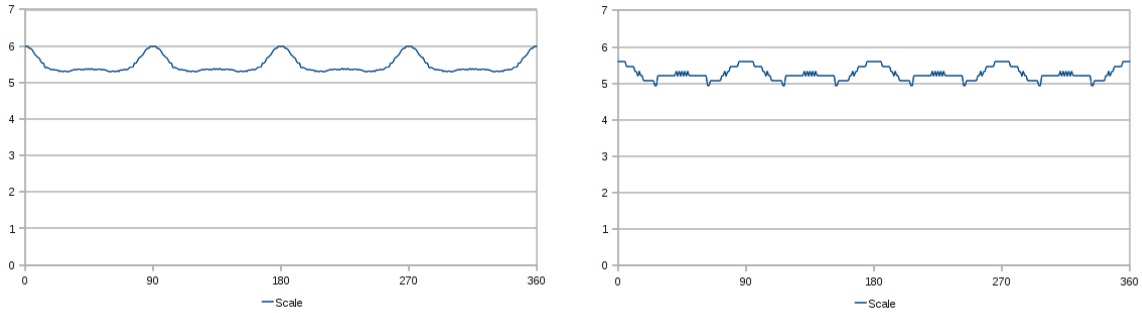

Abbildung 3.10.: Versuchsreihe: Rotation *“Scale“*. Links: libSurf, Rechts: OpenCV

Auf die Sub-pixel genaue Bestimmung der Bildkoordinate zeigt die Rotation keinen nennenswerter störenden Einfluss. Wie schon in der ersten Testreihe lag die gemessene Abweichen konstant unter 1 Pixel.

3.2.3. Perspektivisches Kippen

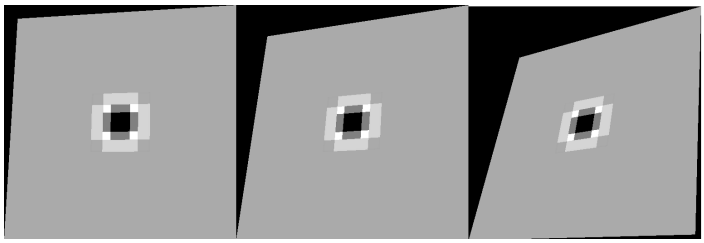

Abbildung 3.11.: Versuchsreihe: Perspektivisches Kippen Beispiele. Von links nach Rechts: Helle-Optimale-SURF-Marke im perspektivisch gekippt um 15°, 30°, und 45°.

In dieser Versuchsreihe wurde eine Perspektivisches Verzerrung simuliert. Die SURF-Marke wird in Schritten von 1° um die Bild-Diagonale gekippt.

Die Ergebnisse bestätigen die gute Robustheit des SURF-Algorithmus und der SURF-Marken gegenüber Perspektivischen-Störungen. Erwartungsgemäß fällt der Detektor-Ausschlag mit steigender Verzerrung (siehe Abb. 3.12 Links). Bemerkenswert ist das mit aktiviertem “upright-SURF“ Parameter, die euklidische Distanz zur Referenzsignatur erst ab einer Kippung von 45° deutlich ansteigt (siehe Abb. 3.12 Rechts). Abb. 3.13 zeigt Beispielhaft das Ergebnis der LTI-Lib-2 für eine Kippung um 55°.

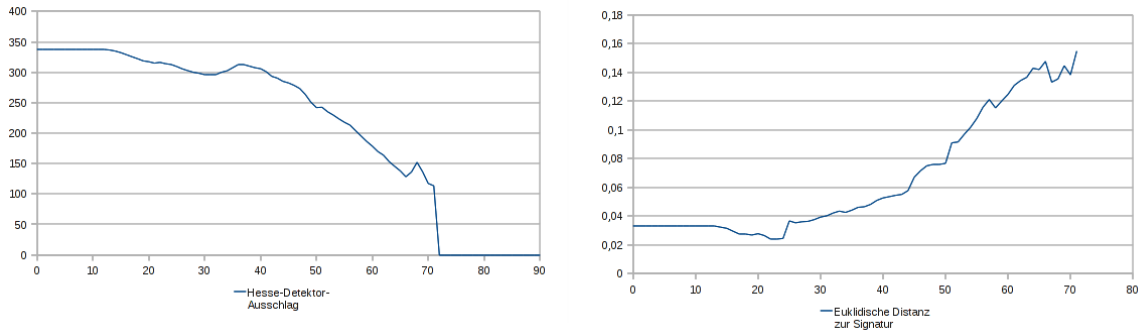

Abbildung 3.12.: Versuchsreihe: Perspektivisches Kippen libSurf. Links: Detektor-Ausschlag, Rechts: Euklidische Distanz zur Referenzsignatur.

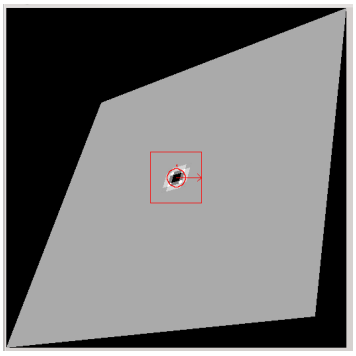

Abbildung 3.13.: Versuchsreihe: Perspektivisches Kippen Beispiel LTI-Lib-2.

3.2.4. Additives weißes gaußsches Rauschen

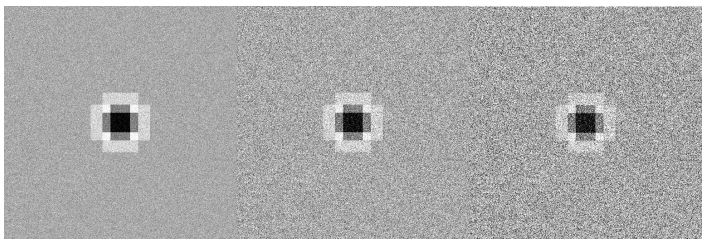

Abbildung 3.14.: Versuchsreihe: Additives weißes gaußsches Rauschen Beispiele. Von links nach Rechts: Helle-Optimale-SURF-Marke mit additivem weißen gauschem Rauschen mit einer Standardabweichung σ von 0.1, 0.2 und 0.3.

Gegenüber Bildrauschen sind die Marken so robust, das hier auf eine weitergehende Präsentation der Ergebnisse verzichtet wird, da selbst für in der Praxis nicht zu erwarten starkes Rauschen, keinerlei nennenswerter Einfluss auf die Messgrößen beobachtet werden konnte.

3.2.5. Kontrast

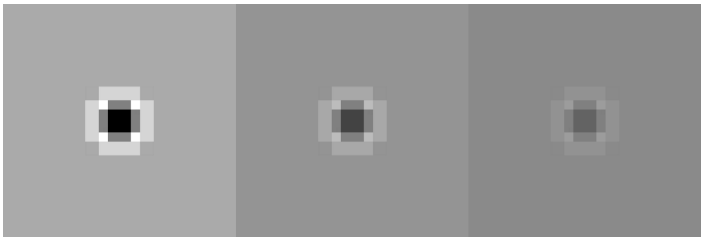

Abbildung 3.15.: Versuchsreihe: Kontrast Beispiele

Im Gegensatz zu Rauschen hat der Kontrast einen sehr großen Einfluss auf den Detektor-Ausschlag. Als vereinfachte Einheit für den Kontrast wird an dieser Stelle die Differenz des maximalen und minimalen Grauwert der Marken genommen. Für die Ursprungsmarke ergibt das einen Kontrast von 255. In Abb. 3.16 zeigt die Kurve des stark sinkende Detektor-Ausschlag exemplarisch für die OpenCV und LTI-Lib-2. Um einen besseren Eindruck für die Empfindlichkeit der Marken gegenüber einem sinkende Kontrast zu erhalten, zeigt Abb. 3.17 das vierte Bild dieser Testreihe. In diesem Bild hat der dunkle Mittelpunkt der Marke einen Grauwert von 40, die hellsten Punkte der Marke haben einen Grauwert von 215. Obwohl subjektiv zwischen diesem Bild und dem Ursprungsbild (siehe z. B. Abb. 3.15 links) noch kein deutlicher Unterschied auszumachen ist. Fällt der Detektor-Ausschlag, der LTI-Lib-2, bei diesem Bild schon auf 1187,2 ab, also um mehr als 50% im Vergleich zum Ursprungsbild ab.

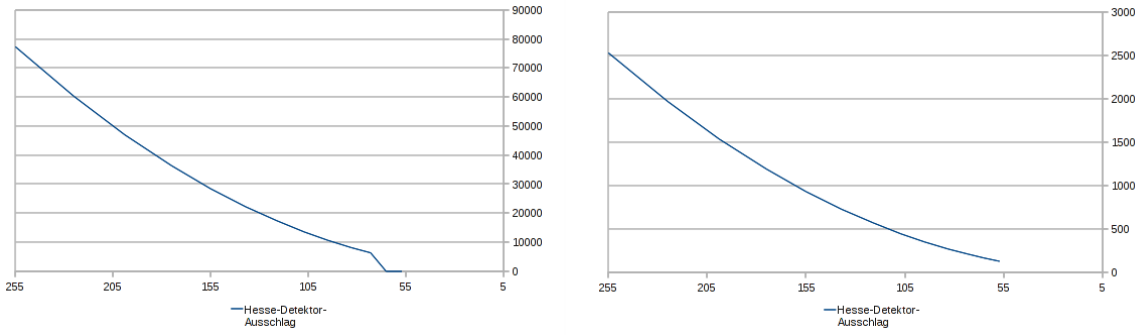

Abbildung 3.16.: Versuchsreihe: Kontrast. Links: Detektor-Ausschlag OpenCV, Rechts: Detektor-Ausschlag LTI-Lib-2.

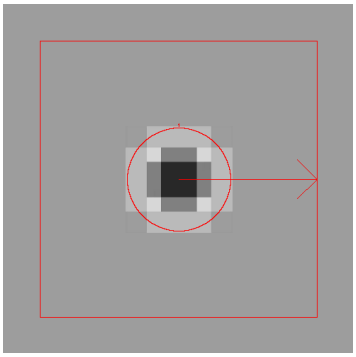

Abbildung 3.17.: Versuchsreihe: Kontrast Beispiel LTI-Lib-2.

3.2.6. Auswertung

Der wichtigst Aspekt der Optimalen-SURF-Marken konnte in den Versuchsreihen bestätigt werden. Der Ausschlag im SURF-Detektor lag bei allen Implementierungen und unter jedem untersuchten Störeffekt deutlich über dem Ausschlag den natürliche SURF-Merkmale erzeugen, dies gilt sogar für starke Perspektivische Verzerrungen, wie das Kippen um 60°. Im Bezug auf den Detektor-Ausschlag ist nur die hohe Empfindlichkeit gegenüber Kontraständerungen, und dementsprechend, eine große Empfindlichkeit gegenüber Belichtungsänderungen, die in realen Anwendungen zu erwarten ist, als problematisch einzustufen. Als einziger weiter negativer Punkt, ist die nicht eindeutig bestimmbare Orientierung der SURF-Marken zu nennen. Daraus resultiert eine große Differenz zur Referenzsignatur, wenn Die SURF-Marke Rotation ausgesetzt wird. Unter Umständen könnte das Verhalten der Marken durch Anpassungen am SURF-Algorithmus verbessert werden. Es ist z. B. vorstellbar das bei einer Verkleinerung des Radius, des Kreises, der zur Bestimmung der Orientierung ausgewertet wird, ein stabileres Ergebnis zustande kommt.

4. Umsetzung

4.1. SURF-Implementierung

Als Grundlage für das Lokalisierungssystem wurde die LTI-Lib-2 SURF Implementierung gewählt. Die libSurf Library der SURF-Autoren stellt, wie schon in Kapitel 2.2.4 erwähnt, aufgrund des nicht verfügbaren Quelltext keine Option da. Im Vergleich zur OpenCV Library, setzt sich die LTI-Lib-2, vor allem durch das komplett Objektorientierte Softwaredesign und die sehr gute Dokumentation des Quelltext durch. Das gleiche gilt, wenn auch in abgeschwächter Form, für die OpenSURF Library. Ein weiterer großer Vorteil der LTI-Lib-2 im Vergleich zu den andern Implementierungen, ist die komplette Parametrierbarkeit zur Laufzeit (siehe Anhang A).

Allerdings haben die Untersuchungen in Kapitel 3.2 auch Schwächen, und von der Originalbeschreibung des Algorithmus abweichendes Verhalten der LTI-Lib-2 Implementierung aufgezeigt. In diesem Kapitel werden alle Fehlerkorrekturen und Änderungen an der LTI-Lib-2 beschrieben. Die daraus resultierende abweichende Implementierung wird als *“modified-LTI-Lib-2“* bezeichnet.

Die LTI-Lib-2 gliedert die Implementierung des SURF-Algorithmus, entsprechend der Dokumentation, in zwei Klassen. Die Klasse *“ltiFastHessianDetection“* implementiert den Beschleunigten-Hesse-Detektor, die Klasse *“ltiSurfLocalDescriptor“* implementiert den SURF-Deskriptor.

Maximum im Skalenraum Die LTI-Lib-2 bietet dem Benutzer der Library, inkorrekterweise über den Parameter: “extrema“, die Möglichkeit einzustellen ob, nur lokale maxima (Maxima), lokale minima (Minima), oder lokale maxima und minima (Both), als Interessenpunkt gewertet werden sollen (Standardeinstellung: Both). Dies ist wahrscheinlich ein Programmierfehler eines Entwicklers mit SIFT Vorkenntnissen, da dort sowohl lokale maxima und lokale minima, als mögliche Interessenpunkt untersucht werden. Bei SURF hingegen befinden sich Interessenpunkt nur an den Stellen, an denen die Determinante der Hesse-Matrix, ein lokales Maximum aufweist. In einem ersten Schritt wurde der Standartwert für diesen Parameter auf Maxima geändert, konsequenterweise sollte der Parameter in Zukunft komplett entfernt werden da lokale minima in SURF keine Bedeutung haben.

Berechnung des Laplacian Die LTI-Lib-2 ist die einzige der untersuchten Implementierungen, die den Laplacian (Spur der Hesse-Matrix) nicht berechnet, für jeden Punkt abspeichert und als Attribut der gefundenen SURF-Merkmale an den aufrufenden Code zurückliefert. Der Laplacian liefert Information über den Kontrast des Interessenpunkt (vgl. Kapitel 2.2.2), und kann für eine sehr zuverlässige Unterscheidung der beiden Optimalen-SURF-Marken benutzt werden (vgl. Tabelle 3.1).

```
1  ...
   fmatrix& det
3  imatrix& lap;

5  for (y = startPos; y < lastMainLoopRow; y += sampleStep) {
     for (x = startPos; x < lastMainLoopCol; x += sampleStep) {
7      ...
       frobRatio = ... ;
9
       dxx = ... ;
11     dyy = ... ;
       dxy = ... ;
13
       det.at(y,x) = (dxx * dyy - lti::pow(frobRatio * dxy, 2));
15     lap.at(y,x) = dxx + dyy > 0 ? 1 : -1;
     }
17 }
   ...
```

Zur Implementierung muss, in jedem Intervall, zusätzlich zu der Matrix die den Detektor-Ausschlag abspeichert (Zeile 2), eine weitere Matrix (Zeile 3) zum Abspeichern des Vorzeichen des Laplacian, angelegt werden. Die Berechnung des Vorzeichen des Laplacian (Zeile 15) stellt nur einen sehr geringen Rechenaufwand da. Dieser Rechenaufwand lohnt sich, da dadurch enorme Einsparungen bei dem Vergleich von SURF-Deskriptoren, durch eine vorhergehende Gruppierung erreicht werden.

Skalenraum Wie schon in Kapitel 3.2.1 erwähnt, weißt die Implementierung des Skalenraum, in der LTI-Lib-2, Abweichungen zu der Dokumentation der SURF-Autoren auf. Abb. 4.1 zeigt die Konfiguration des Skalenraum, mit drei Oktaven, nach der Beschreibung der SURF-Autoren (oben), und die Konfiguration des Skalenraum in der LTI-Lib-2 mit den gleichen Parametern (unten).

In der LTI-Lib-2 werden die Oktaven nicht überlagert, und bei der Auswertung wird der Komplette Skalenraum, quasi als eine große Oktave betrachtet, und nicht wie empfohlen, jede Oktave separat nach Interessenpunkten durchsucht. Die LTI-Lib-2 deckt mit Ihrer Konfiguration zwar immer einen größeren Bereich im Skalenraum ab, aber sowohl bei typischen

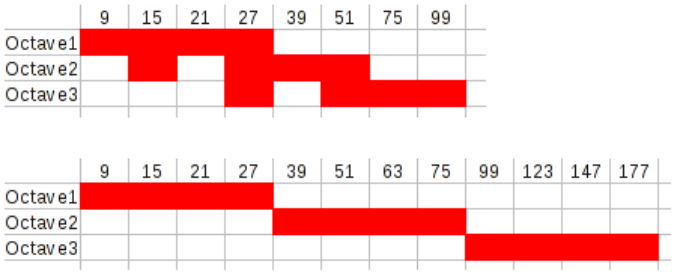

Abbildung 4.1.: Fehlerhafte Konfiguration des Skalenraum

SURF-Anwendungen, wie dem Bildabgleich, als auch bei dem Lokalisierungssystem basierend auf SURF-Marken, ist einen Skalenraum mit maximal vier, sich überschneidenden, Oktaven als ausreichend anzunehmen.
Die mehrfache Verwendung von Intervallen in verschiedenen Oktaven bietet zudem eine weitere Möglichkeit die Performance zu steigern. Für einen Skalenraum von drei Oktaven mit je vier Intervallen pro Oktave, müssen nur acht Intervalle berechnet werden, da das erste Intervall einer Oktave gleich dem zweiten Intervalle der vorherigen Oktave ist, und das zweite Intervall einer Oktave gleich dem letztem Intervall der vorherigen Oktave ist (vgl. Abb. 4.1).

Für die *"modified-LTI-Lib-2"* wurde eine neue Option für den Parameter *"levelSelectionMethod"* eingeführt. Setzt man den Parameter auf: *"IndependentBlocks"* erhält man das von den SURF-Autoren beschrieben Verhalten, setzt man den Parameter auf: *"Blocks"* erhält man das ursprünglichen Verhalten der LTI-Lib-2.

Bestimmung der Orientierung Der Teil des Quelltext, der die Orientierung eines gefunden SURF-Merkmals bestimmt, ist in der ursprünglichen Version der LTI-Lib-2, in der Klasse *"ltiFastHessianDetection"* implementiert. Dies ist zum einen unschön da es nicht die Beschreibung des Algorithmus widerspiegelt, zum anderen hat es auch praktische Nachteile. Wird die Orientierung während des Detektor-Durchlaufs bestimmt, kann dieser Schritt entweder nur, für alle oder für keines der Merkmale durchgeführt werden. Die Bestimmung der Orientierung in der Klasse *"ltiSurfLocalDescriptor"* und damit zu einem späterem Zeitpunkt, bietet den Vorteil das die Orientierung für eine Untermenge, an besonders interessanten Punkten, durchgeführt werden kann, abhängig zum Beispiel, von der Stärke des gefunden Merkmals.

Region of Interest (ROI) Eine *"Region of Interest"*, beschriebt eine Untermenge des Eingangsbilds. Bei dem Lokalisierungssystem sind grundsätzlich zwei Einsatzszenarien vorstellbar. Zum einen, kann für den Fall das bei einem Suchdurchlauf, eine SURF-Marke im

Bild gefunden wurde, für den nächsten Suchdurchlauf der Bereich in dem die Marke zu erwarten ist, eingegrenzt Werden, eventuell auch durch Berücksichtigung von Odometriedaten (siehe Abb. 4.2 links). Zum anderen ist es denkbar das aufgrund des Systemaufbaus, Höhe der Marken an der Wand, Höhe der Kamera auf dem mobilem Roboter sowie der maximalen und minimalen Entfernung des Roboters zu den Marken, ein Auftreten der Marken in bestimmte Bildregionen generell ausgeschlossen werden kann (siehe Abb. 4.2 rechts).

Abbildung 4.2.: Beispiel: Region of Interest

Die einzelnen ROI's sind Matrizen von Integerwerten, die als Binärbild interpretiert werden können. Die Matrizen müssen dabei die gleiche Größe wie die Auflösung des Bildes haben. Ist ein Element in der Matrix null (in Abb. 4.2 schwarz dargestellt), wird der entsprechende Bildpunkt im Eingangsbild nicht auf einen Interessenpunkt hin untersucht. Damit kann die Anzahl der zu untersuchenden Bildpunkten in vielen Fällen stark reduziert werden und ein großer Performance Vorteil erreicht werden.

```
bool apply(const channel& src,
           const imatrix& mask,
           list<location>& locs,
           list<float>& strength,
           & numLocs,
           list<int>& laplacian) const;
```

Der Quelltextausschnitt zeigt die Definition einer der "apply" Methoden der *'modified-LTI-Lib-2"*. In der zweiten Zeile erfolgt die Definition des Parameter *"mask"*. Die Größe der Maske muss mit der Größe des Eingangsbilds *"src"* übereinstimmen, ansonsten wird die Maske ignoriert.

Auf SURF-Marken zugeschnittener Deskriptor Die Untersuchungen in Kapitel 3.2 haben gezeigt das die Kantenlänge der Optimalen-SURF-Marken im Bild, in etwa dem achtfachen des *"scale"* entspricht. Abbildung 3.2 zeigt das der Standard SURF-Deskriptor nicht optimal

für die Beschreibung der SURF-Marken ist. Ein weitere Nachteil bei der Verwendung des Standard Deskriptor ist, das ein Großteil der Berechneten Informationen wieder verworfen wird, und somit nicht zur Identifikation der SURF-Marken beiträgt. An dieser Stelle zahlt sich die überaus gut Parametrierbarkeit der LTI-Lib-2 aus, durch anpassen der beiden Parameter *"numberOfSubregions"* und *"subregionSamples"* lassen sich unterschiedliche Variationen des Deskriptor erzeugen. Der Parameter *"numberOfSubregions"* gibt an, in wie viele Unterregionen, pro Richtung, der Deskriptor unterteilt wird. Die tatsächliche Anzahl an Unterregionen ist also das Quadrat von *"numberOfSubregions"*. Der Parameter *"subregionSamples"* bestimmt wie viele Abtastwerte, pro Richtung, innerhalb einer Unterregionen extrahiert werden. Abb. 4.3 zeigt den, als optimal ermittelten, Aufbau des Deskriptor resultierend aus den Einstellungen *"numberOfSubregions"* = 4 und *"subregionSamples"* = 2. Diese Anpassung reduziert den Rechenaufwand, für das Erstellen eines Deskriptors, um 75%.

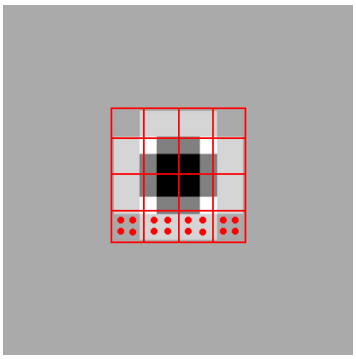

Abbildung 4.3.: Angepasster SURF-Deskriptor

4.2. Verwendete Kamera

Alle Testbilder in dieser Arbeit wurden mit folgender Kamera erstellt: Imaging Source DMK 41BF02. Die Kamera arbeitet mit einer Auflösung von 1280x960 Pixel. Für den effizienten Zugriff auf die Kamera, über den FireWire Bus, wurde ein rudimentäres LTI-Lib Plugin, für die Libray unicap entwickelte. Die unicap Library ermöglicht den Betrieb der Kamera im DMA Modus unter Linux.

URL: http://unicap-imaging.org

4.3. Informationskodierung mit SURF-Marken

Da es nur zwei verschiedene Versionen der SURF-Marke gibt (hell und dunkel), kann mit einer Marke nur eine Information von einem Bit kodierte werden. Ein Möglichkeit wäre es, die SURF-Marken “nur“ als “Eye-Catcher“ zu verwenden und die Eigentlich Information, zum Beispiel, in einem Kreisförmigen-Barcode um die Marke herum zu hinterlegen. Als eine bessere Alternative, wird aber die Umsetzung der von [Schweiger u. a. (2009)] entwickelte Idee, der SURF-Marken-Arrays, betrachtet. Dabei werden mehrere SURF-Marken zu einer Gruppe zusammengefasst und über die beiden, durch den Laplacian unterscheidbaren Marken, eine Bitkombination kodiert. In Ihrer Arbeit beweisen, Schweiger u.a., die Grundsätzliche Realisierbarkeit solcher SURF-Marken-Arrays, anhand der kleinstmöglichen Anordnung in einem 2×2 Array, welches eine Kodierung von drei Bit erlaubt. Ein weitere Vorteil dieses Ansatz, ist das damit ein Zusätzliches Kriterium, für die Unterscheidung von natürlichen SURF-Merkmalen und den Optimalen-SURF-Marken geschaffen wird. Die Untersuchungsergebnisse aus Kapitel 3.2 lassen darauf schließen, das in einer Konkreten Anwendung, in der mit gleichzeitiger Rotation, Perspektivischer-Verzerrung und suboptimaler Belichtung zu rechnen ist, der Detektor-Ausschlag und ein Abgleich mit der Referenzsignatur der Marken nicht zur eindeutigen Identifikation der Marken ausreicht. Bei der Verwendung von Marken-Arrays müssen immer Gruppen, einer bestimmten Anzahl von Marken, die in einer bestimmten räumlichen Nähe und Anordnung zueinander stehen und in etwa in dem gleichen *“scale“* erkannt wurden, existieren.

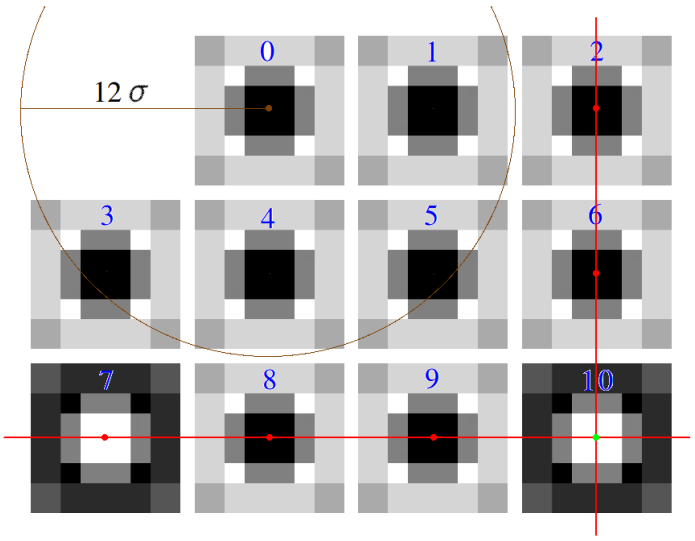

Abbildung 4.4.: Beispiel: SURF-Marken-Array, kodierter Wert: 896

Abb. 4.3 zeig eines der Möglichen elf Bit, 4×3, Marken-Arrays die in dieser Arbeit verwendet werden. Das Marken-Array wird von links nach rechts und von oben nach unten gelesen.

Durch die Lücke an der ersten Stelle im Array, kann dem Array eine Orientierung zugewiesen werden. Die blauen Zahlen bezeichnen den Index der jeweiligen Marke im Array. Die kodierte Bitkombination ergibt sich dabei aus den Vorzeichen des Laplacian der SURF-Marken (Helle-Marke = positiv = 1; Dunkle-Marke = negativ = 0). Der Marke mit dem höchsten Index, die also das *"most significant bit"* darstellt, kommt noch eine weiter Bedeutung zu, die Bildkoordinate dieser Marke wird als Position des Marken-Arrays definiert. Um die Position des Marken-Arrays genauer bestimmen zu können, kann die Position des Arrays auch als Schnittpunkt der Ausgleichsgerade, der Bildkoordinaten, aller Marken der letzten Zeile, mit der Ausgleichsgerade, der Bildkoordinaten, aller Marken der letzten Spalte, definiert werden.

4.4. Lokalisierung der SURF-Marken-Arrays

Um ein Array von SURF-Marken erfolgreich lokalisieren zu können, müssen alle Marken des Arrays erkannt werden, der Verlust einer Marke hat den Ausfall des kompletten Arrays zur Folge. Es gibt drei Kriterien die zur Lokalisierung der einzelnen Marken, eines Arrays, herangezogen werden können:

1. **Hesse-Detektor-Ausschlag**
   Die Beurteilung des Ausschlag erfolgt durch eine Schwellwert Analyse (english: tresholding), liegt der Wert über einem definierten Schwellwert ist das Ergebnis positiv, andernfalls ist das Ergebnis negativ.

2. **Euklidische Distanz zur Referenzsignatur**
   Auch die Beurteilung der euklidischen Distanz zur Referenzsignatur kann mit einer einfachen Schwellwert Analyse durchgeführt werden. Ist die Distanz kleiner als der Schwellwert ist das Ergebnis positiv, andernfalls ist das Ergebnis negativ.

3. **Räumliche Nähe, Anordnung und erkannter *"scale"***
   Alle Marken eines Arrays müssen ungefähr in der gleichen Ebene der Bild-Pyramide gefunden werden. Aus dem *"scale"* lässt sich auch relativ einfach die minimale und maximale Räumliche Distanz, zweier benachbarter Marken, in Pixeln errechnen (siehe Abb. 4.4).

Der Hesse-Detektor-Ausschlag ist sehr empfindlich gegenüber Belichtungsänderungen, andererseits aber sehr robust gegenüber Rotation und Perspektivischen Verzerrungen. Für die euklidische Distanz zur Referenzsignatur, gilt genau das umgekehrte, eine hohe Robustheit gegenüber Belichtungsänderungen, aber eine niedrige Robustheit gegenüber Rotation und Perspektivischen Verzerrungen. Durch die Kombination der drei Kriterien, mit jeweils sehr

großen Toleranzbereichen, lässt sich ein sehr gutes Ergebnis, mit einer maximalen Robustheit gegenüber allen Störeffekten, erzielen.

Im ersten Schritt erfolgt eine Analyse des Eingangsbilds mit dem SURF-Detektor. Mit einem sehr niedrige angesetzten Schwellwert, bleiben nach diesem Schritt noch mehr als 200 mögliche Marken im Bild (siehe Abb. 4.5).

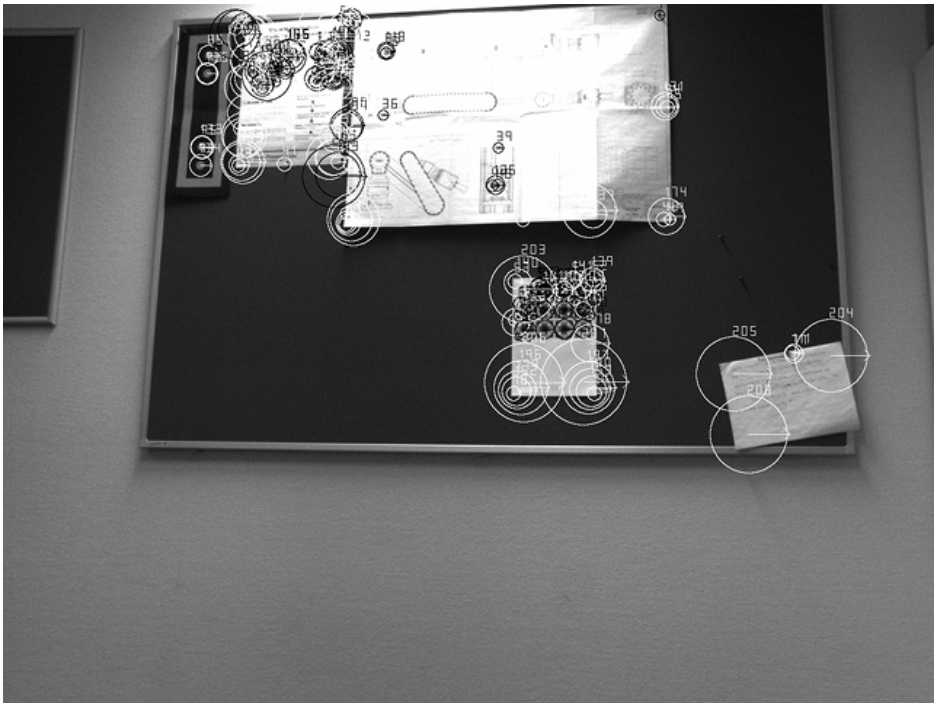

Abbildung 4.5.: Marken-Array-Lokalisierung Schritt 1

Im zweiten Schritt erfolgt der Abgleich mit der Referenzsignatur der jeweiligen SURF-Marke. Da eine Unterscheidung der beiden Marken mit Hilfe des Vorzeichen des Laplacian möglich ist, wurden getrennte Referenzsignatur für die helle und die dunkle SURF-Marke berechnet. Für den Abgleich wird dann, abhängig vom Laplacian des Interessenpunkt, die jeweilige Referenzsignatur ausgewählt. Die entwickelte Testumgebung bietet einen Modus zum Einlernen der Referenzsignaturen (siehe Anhang B). Auch nach dem zweiten Schritt, befinden sind noch falsch positiv erkannte Marken, in dem Testbild (sieh Abb. 4.6).

Diese verbliebenen falsch positiv erkannten Marken, werden im dritten Schritt, bei der Auffindung der Marken-Arrays eliminiert. Der dazu verwendete Algorithmus, startet bei einem zufällig gewählten Marken-Kandidat und sucht für diesen Kandidat, alle noch nicht untersuchten andern Kandidaten, die sich sowohl in dessen Räumlicher Nachbarschaft als auch in dessen *“scale“*-Nachbarschaft befinden. Die gefunden Nachbarn werden der gleiche Gruppe

Abbildung 4.6.: Marken-Array-Lokalisierung Schritt 2

zugeordnet und der Schritt für diese Kandidaten wiederholt bis keine neuen Nachbarn gefunden werden. Dieser Schritt wird wiederholt bis alle Kandidaten untersucht wurden. Besteht eine Gruppe von Kandidat aus elf Elementen, so wurde ein Array gefunden. Abb. 4.7 zeigt das gefundene Marken-Array (Roter Punkt und die dekodierte Bitfolge, interpretiert als unsigned integer), sowie die übrigen Marken die keinem Array zugeordnet werden konnten.

Abbildung 4.7.: Marken-Array-Lokalisierung Schritt 3

4.5. Dekodierung der SURF-Marken-Arrays

Die Dekodierung der Bitkombination die durch das SURF-Marken-Array repräsentiert wird, erfolgt nach folgendem Algorithmus:

1. Werden die Marken gesucht die im Array an den Ecken liegen. Das sind alle Marken die genau drei Nachbarn haben, dies setzt eine minimale Größe des Marken-Arrays von 3×3 voraus. An welcher Ecke die Marken genau liegen ist zu diesem Zeitpunkt nicht bekannt. Da an der oberen linken Ecke keine Marke platziert wird, müssen immer genau drei Ecken gefunden werden.
2. Für jede gefundene Ecke werden die drei Vektoren zu den jeweiligen Nachbarn addiert. Der Quadrant in den der Vektor zeigt ist die Orientierung der Marke. Der Quadrant in den keiner der drei Vektoren zeigt, ist die Orientierung des gesamten Arrays (blauer Pfeil in Abb. 4.8).
3. Anhand der Orientierung der Marken und der Gesamtorientierung des Arrays, lässt sich eine Zuweisung der Ecken durchführen, relativ zur Ausrichtung des Arrays, die durch die Lücke an der ersten Stelle definiert ist, kann jetzt bestimmt werden welche Ecke links unten, rechts unten und rechts oben ist.

4. Sind alle Ecken bekannt, können daraus die Zwischenschritte zu den einzelnen Elementen des Arrays abgeleitet werden. Und abschließend die Bitkombination ausgelesen werden.

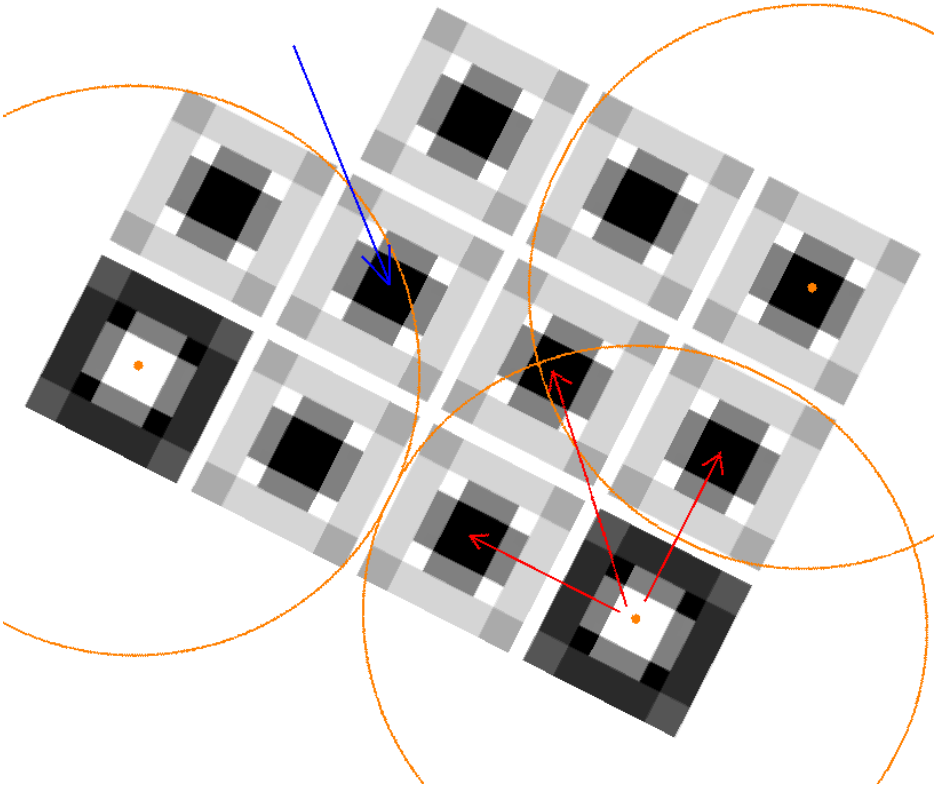

Abbildung 4.8.: Marken-Array-Dekodierung

5. Zusammenfassung

Die in dieser Arbeit entwickelte Lösung, zur absoluten Lokalisierung mobiler System, erreicht eine sehr hohe Robustheit gegenüber allen untersuchten Störeinflüssen, mit Ausnahme der Rotation. Gegenüber Perspektivischer Verzerrung, konnte in Abschließende Test, selbst bei einer Verzerrung von bis zu 45° (siehe Abb. 5.1), noch ein sichere Erkennung und Auswertung der Positionsmarken festgestellt werden. Die schlechte Robustheit gegenüber Rotation, ist auf die nicht eindeutig bestimmbare SURF-Orientierung der Marke zurückzuführen. Für mobile System die sich nur parallel zum Fußboden bewegen können, ist aber nicht mit starken Rotationen zu rechnen, deshalb empfiehlt sich bei solchen Systemen generell der Einsatz von *"upright-SURF"*. Die verwendeten elf-Bit Marken-Arrays, bieten mit 2048 möglichen Bitkombinationen, einen ausreichend großen Raum zur Kodierung kompletter Areale. Die Performance des Lokalisierungssystems, lässt den Einsatz auch auf eingebetteten System zu. Auf einem Testsystem mit einem AMD Dual-Core CPU Prozessor mit 2.3 GHz, wurde bei voller CPU-Last bis zu 8 Bilder pro Sekunde verarbeitet. In der Praxis ist aber, durch die Kombination mit relativer Lokalisierung, oder der Verfolgung von Positionsmarken, und daraus resultierenden suche in einem stark eingeschenktem Bildbereichen, mit einer deutlich geringeren CPU-Last zu rechnen.

Abbildung 5.1.: Lokalisiertes SURF-Marken-Array unter Perspektivischer Verzerrung

Literaturverzeichnis

[Agrawal u. a. 2008] AGRAWAL, Motilal ; KONOLIGE, Kurt ; BLAS, Morten: CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching. In: FORSYTH, David (Hrsg.) ; TORR, Philip (Hrsg.) ; ZISSERMAN, Andrew (Hrsg.): *Computer Vision – ECCV 2008* Bd. 5305. Springer Berlin / Heidelberg, 2008, S. 102–115

[Bay u. a. 2006] BAY, Herbert ; ESS, Andreas ; TUYTELAARS, Tinne ; GOOL, Luc V.: *Speeded-Up Robust Features*. http://www.vision.ee.ethz.ch/~surf/papers.html. Version: 2006

[Brown u. Lowe 2002] BROWN, Matthew ; LOWE, David G.: Invariant Features from Interest Point Groups. In: *BMVC*, 2002

[Evans 2009] EVANS, Christopher: *Notes on the OpenSURF Library*. http://www.chrisevansdev.com. Version: January 2009

[IPP 2011] *Intel Integrated Performance Primitives*. http://software.intel.com/en-us/articles/intel-ipp. Version: 2011

[Juan u. Gwun] JUAN, Luo ; GWUN, Oubong: *A Comparison of SIFT, PCA-SIFT and SURF*

[Lowe 2004] LOWE, David G.: *Distinctive Image Features from Scale-Invariant Keypoints*. http://www.vision.ee.ethz.ch/~surf/papers.html. Version: 2004

[LTI-Lib 2011] *LTI-Lib*. http://ltilib.sourceforge.net/doc/homepage/. Version: 2011

[LTI-Lib-2 2011] *LTI-Lib-2*. http://www.ie.itcr.ac.cr/palvarado/ltilib-2/homepage/. Version: 2011

[Neubeck u. Gool 2006] NEUBECK, Alexander ; GOOL, Luc V.: Efficient Non-Maximum Suppression. In: *ICPR 2006*, 2006. – in press

[OpenCV 2011] *OpenCV*. http://opencv.willowgarage.com. Version: 2011

[Rupp 2001] RUPP, Torsten: *Absolute Lokalisation mobiler Roboter durch Codierungen mit Landmarken*. 2001

[Schweiger u. a. 2009] SCHWEIGER, F. ; ZEISL, B. ; GEORGEL, P. ; SCHROTH, G. ; STEINBACH, E. ; NAVAB, N.: *Maximum Detector Response Markers for SIFT and SURF*. vmv09.tu-bs.de/downloads/papers/sch09a.pdf. Version: Nov 2009

[TBB 2011] *Intel Threading Building Blocks*. http://www.threadingbuildingblocks.org. Version: 2011

[Viola u. Jones 2001] VIOLA, Paul ; JONES, Michael: Rapid Object Detection using a Boosted Cascade of Simple Features. In: *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on* 1 (2001), S. 511. http://dx.doi.org/http://doi.ieeecomputersociety.org/10.1109/CVPR.2001.990517. – DOI http://doi.ieeecomputersociety.org/10.1109/CVPR.2001.990517. – ISSN 1063–6919

Anhang

A. SURF-Parameter

Nr.	Typ	original Paper	libSurf	OpenSURF	OpenCV	LTI-Lib-2
1	integer	Ocatves Def:3	octaves Def:4	Octaves Def:5	nOctaves Def:4	numberOfLevels Def:12
2	integer	Intervals Def:4	Value:4	Intervals Def:4	nOctaveLayers Def:2	AND levelGroupSize Def:4
3	integer/float	hessian treshold	thres Def:0.2f	THRES Def:0.0004f	hessianTreshold	threshold AND locationSelectionMode Def:0.1 AND Absoulute
4	integer	initial sample step Def:2	samplingStep Def:2	init_sample Def:2	SAMPLE_STEP0 Def:1	initialSamplingStep Def:2
5	integer	initial filter size Def:9	initMasksize Def:9	Fasthessian.cpp:155 Value:9	HAAR_SIZE0 Def:9	initialKernelSize Def:9
6	integer	filter increment step Def:6	n/a	Fasthessian.cpp:155-158 Value:6	HAAR_SIZE_INC Def:6	initialKernelStep Def:6
7	float	n/a	n/a	n/a	n/a	normPower Def:4
8	boolean	calculate extended descriptor Def:false	Ext Def:false	n/a	extended Def:false	signSplit Def:false
9	integer	orientation radius Def:6s	n/a	n/a	ORI_RADIUS Def:6s	orientationNeighborhoodFactor Def:6s
10	integer	orientation window Def:60°	n/a	n/a	ORI_WIN Def:60°	orientationWindowWidth Def:60°
11	float	orientation gaussian weight Def:2.5s	n/a	n/a	ORI_SIGMA Def:2.5s	orientationGaussianFactor Def:2.5s
12	integer	side length of haar-wavelets Def:4s	n/a	n/a	Line:463 Value:4s	orientationWaveletSizeFactor Def:4s
13	integer	descriptor windows Def:20s	indexSize Def:4	n/a	PATCH_SZ Def:20s	NumberOfSubregions Def:4
14	integer		n/a	n/a		subregionSamples Def:5
15	float	descriptor windows gaussian weight Def:3.3s	n/a	n/a	DESC_SIGMA Def:3.3s	gaussianWeight Def:3.5
16	integer	descriptor haar wavelet size Def:2s	n/a	n/a		waveletSize Def:2s
17	boolean	U-SURF Def:false	usurf Def:false	upright Def:false	n/a	ComputeOrientation Def:true

1-7: Beschleunigter Hesse-Detektor
8-17: SURF-Deskriptor

Konfiguration zur Laufzeit
Konstante im Quelltext
Fest codiert im Quellcode

B. Synopsis SURFImplTest

```
USAGE:

   ./SURFImplTest  [--drawDescriptorWindow] [-D <LIGHT|DARK>] [-T <int>]
                   [-E] [--printSignature] [--printDescriptor] [-M <LIGHT
                   |DARK>] [-p <string>] [-u] [-e] [-t <float>] [-c <int
                       >]
                   [-k <int>] [-s <int>] [-n <1|2|3|4|5>] [-o
                       <1|2|3|4|5>]
                   [-l <DEBUG|TEST>] -m <TestStaticImage|TestVideo
                   |TestVideoImage|TestVideoDiscontinuity|TestMarkerArray
                   |TestMarkerArrayTracking|TestMarkerScale
                   |TestMarkerInPlaneRotation|
                       TestMarkerOutOfPlaneRotation
                   |TestMarkerNoise|TestMarkerBackground|
                       TestMarkerContrast
                   |MarkerDescriptorTraining|TestDescriptor
                   |CreateExampleTestImages> -i <libSurf|OpenSURF|OpenCV
                   |LTI-Lib-2|MY-LTI-Lib-2> [--] [--version] [-h]


Where:

   --drawDescriptorWindow
     Draw descriptor window

   -D <LIGHT|DARK>, --detect <LIGHT|DARK>
     Which marker/blob to detect

   -T <int>, --topFeatureCount <int>
     Top features count

   -E, --extractMarkers
     Extract Optimal SURF-Markers from image

   --printSignature
     Print signature
```

```
34      --printDescriptor
          Print descriptor
36
        -M <LIGHT|DARK>, --marker <LIGHT|DARK>
38        Which marker to use

40      -p <string>, --path <string>
          Path to the test image
42
        -u, --uprightSURF
44        Use the USURF-Descriptor (not rotation invariant)

46      -e, --extendedDescriptor
          Calculate the Extended (128 Dimesions) SURF-Descriptor
48
        -t <float>, --treshold <float>
50        Blob Treshold

52      -c <int>, --initialKernelIncrementStep <int>
          Initial kernel increment step
54
        -k <int>, --initialKernelSize <int>
56        Initial kernel size

58      -s <int>, --initialSampleStep <int>
          Initial sample step
60
        -n <1|2|3|4|5>, --intervals <1|2|3|4|5>
62        Intervals in Scalespace

64      -o <1|2|3|4|5>, --octaves <1|2|3|4|5>
          Octaves in Scalespace
66
        -I <DEBUG|TEST>, --interactiveMode <DEBUG|TEST>
68        run in interactive DEBUG or in full automatic non-interactive TEST
          mode
70
        -m <TestStaticImage|TestVideo|TestVideoImage|TestVideoDiscontinuity
72          |TestMarkerArray|TestMarkerArrayTracking|TestMarkerScale
            |TestMarkerInPlaneRotation|TestMarkerOutOfPlaneRotation
74          |TestMarkerNoise|TestMarkerBackground|TestMarkerContrast
            |MarkerDescriptorTraining|TestDescriptor|CreateExampleTestImages>,
76          --mode <TestStaticImage|TestVideo|TestVideoImage
```

```
    |TestVideoDiscontinuity|TestMarkerArray|TestMarkerArrayTracking
78      |TestMarkerScale|TestMarkerInPlaneRotation
    |TestMarkerOutOfPlaneRotation|TestMarkerNoise|TestMarkerBackground
80      |TestMarkerContrast|MarkerDescriptorTraining|TestDescriptor
    |CreateExampleTestImages>
82    (required)  Test mode

84  -i <libSurf|OpenSURF|OpenCV|LTI-Lib-2|MY-LTI-Lib-2>, --implementation
    <libSurf|OpenSURF|OpenCV|LTI-Lib-2|MY-LTI-Lib-2>
86    (required)  SURF-Implementation to use

88  --, --ignore_rest
    Ignores the rest of the labeled arguments following this flag.
90
    --version
92    Displays version information and exits.

94  -h, --help
    Displays usage information and exits.
96

98  Speed up robust feature implementation test util
```

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 30. März 2011	
Ort, Datum	Unterschrift