



FACHHOCHSCHUL -
BACHELORSTUDIENGANG
Sensorik- und Mikrosysteme

**Untersuchung der Anwendbarkeit
des Speeded Up Robust Features
Algorithmus auf Infrarotbilder**

als Bachelorarbeit eingereicht

zur Erlangung des akademischen Grades

Bachelor of Science in Engineering

von

HATHEIER Thomas

August 2009

Betreuung der Bachelorarbeit durch
Dipl. Ing. (FH) Dr. Zauner Gerald



Campus **Wels**

Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen nicht benutzt, die den benutzten Quellen entnommenen Stellen als solche kenntlich gemacht habe und dass die Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.
Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

.....
Thomas, Hatheier

Mühlheim, 20.08.2009

1 Kurzfassung

Diese Arbeit beschäftigt sich mit einem Key-Point basierten Detektions-Algorithmus der Computer Vision - dem so genannten SURF Algorithmus (Speeded Up Robust Features Algorithmus). Es werden die theoretischen Grundlagen zum SURF Algorithmus dargestellt und die Anwendbarkeit auf Infrarotbilder näher untersucht.

Der SURF Algorithmus stellt eine sehr effiziente Methode zur Detektion und Beschreibung von Key-Points in Bildern dar. Die Key-Points die mittels SURF detektiert werden, sind rotationsinvariant, skaleninvariant, robust gegenüber Änderungen des Kamerastandpunktes und weitgehend robust gegenüber Bildstörungen. Des Weiteren lässt sich dieser Algorithmus, im Vergleich zu anderen derartigen Methoden, sehr effizient implementieren. Somit wird auch eine Anwendung ohne spezielle Hardware (GPU basierte Lösungen, Hochleistungsrechner) ermöglicht.

Infrarotbilder weisen sehr homogene Strukturen auf und sind im Allgemeinen sehr arm an Details (die ein wesentliches Kriterium für die Anwendung solcher Algorithmen darstellen). In dieser Arbeit wird untersucht, ob sich Infrarotbilder grundsätzlich für eine Auswertung mittels punktbasierter Feature-Detektoren eignen. Es wird das Verhalten des SURF Algorithmus bezüglich des Auffindens von Key-Points näher untersucht. Des Weiteren wird das Matching von detektierten Punkten in Bildfolgen näher betrachtet. Besonderes Augenmerk wird dabei auf Konfigurationsmöglichkeiten von bestehenden SURF Implementierungen gelegt, wobei auch die Performance von SURF näher untersucht wird.

Inhaltsverzeichnis

1 Kurzfassung	III
Inhaltsverzeichnis	IV
Abbildungsverzeichnis	VI
2 Einleitung	1
2.1 Zielsetzung	1
2.2 Key-Point basierte Algorithmen	1
2.2.1 Algorithmen	1
2.2.1.1 Speeded Up Robust Features - SURF	1
2.2.1.2 Scale Invariant Feature Transform - SIFT	2
2.2.2 Anwendungsbereiche	2
2.2.2.1 Allgemein	3
2.2.2.2 Robotik	3
3 Speeded Up Robust Features - SURF	5
3.1 Allgemeines	5
3.2 Algorithmus	5
3.2.1 Integral Image Darstellung	5
3.2.2 Feature-Detektion	8
3.2.3 Hesse-Matrix basierter Feature Detektor	9
3.2.3.1 Skalenraum	9
3.2.3.2 Fast-Hessian Detektor	11
3.2.3.3 SURF-Filterpyramide	14
3.2.3.4 Lokalisierung der Schlüsselpunkte	16
3.2.4 Interest Point Descriptor	17
3.2.4.1 Bestimmung der Orientierung	17
3.2.4.2 Bestimmung des Descriptors	18
3.3 Verwendete SURF Implementierung	19
3.3.1 Speeded Up Robust Features SURF	19
3.3.1.1 Matlab-Interface	20
3.3.2 OpenSURF	20

4 SURF Key-Points und Matching	21
4.1 Allgemeines	21
4.2 Berechnung der Key-Points	21
4.2.1 Vergleich / Ergebnisse	25
4.3 Matching mit SURF	26
4.3.1 Nearest Neighbour Ratio - Matching Methode	26
4.3.2 Matching	28
4.4 Verhalten von SURF	28
4.4.1 Verhalten des Detektors	28
4.4.1.1 Rotation	30
4.4.1.2 Rauschen	31
4.4.2 Verhalten des Descriptors	32
4.4.2.1 Rotation	33
4.5 Performance von SURF	34
4.5.1 Performance Key-Point Berechnung	35
4.5.2 Performance Matching	36
5 Zusammenfassung	38
Literaturverzeichnis	40

Abbildungsverzeichnis

2.1	Gelbe, Orange und Rote Zone (v.o.) eines RoboCup Rescue Parcours (Quelle: [Homepage, NIST])	4
3.1	Vergleich Originalbild-Daten (links) und Integral Image-Daten (rechts)	6
3.2	Vergleich IR-Bild und Integral Image	6
3.3	Flächensummen in einem Integral Image	7
3.4	Vergleich von Feature-Detektoren (Quelle: [Bay, Ess, Tuytelaars, Van Gool, 2008])	9
3.5	LoG-Funktion L_{xy} , $\sigma = 2$	12
3.6	LoG-Funktionen L_{xx} (links) und L_{yy} (rechts), $\sigma = 2$	12
3.7	D_{yy} und D_{xy} Box-Filterkerne der Größe 9×9 mit $\sigma = 1.2$	13
3.8	Image Pyramide (links) und Filter Pyramide (rechts) (Quelle: [Bay, Ess, Tuytelaars, Van Gool, 2008])	14
3.9	Vergrößerung der Filtermatrix von 9×9 auf 15×15 (Quelle: [Bay, 2006])	15
3.10	$3 \times 3 \times 3$ Nachbarschaft bei der „non-maximum suppression“ (Quelle: [Bay, 2006])	16
3.11	Haar-Wavelet Filter in x Richtung (links) und in y Richtung (rechts) (Quelle: [Bay, Ess, Tuytelaars, Van Gool, 2008])	17
3.12	Diagramm zur Bestimmung der Orientierung (Quelle: [Evans, 2009])	18
3.13	Berechnung des Eigenschaftsvektors (Quelle: [Bay, 2006])	19
4.1	Realbild (links) und Infrarotbild (rechts) mit berechneten Key-Points	22
4.2	Infrarotbild mit angepassten Eigenschaften	23
4.3	Diagramm der detektierten Schlüsselpunkte in IR- sowie Realbildern	25
4.4	Kontrast von Schlüsselpunkten	27
4.5	Matching Beispiel Infrarotbild (320×240 Pixel)	28
4.6	Wiederholbarkeit bei Rotation, $\epsilon = 1,5$	30
4.7	Vergleich Originalbild (links) und verrauschtes Bild (rechts), Amplitude des Rauschsignals $A = \pm 0,071$	31
4.8	Wiederholbarkeit bei Rauschen, $\epsilon = 1,5$	31

4.9 „recall“ vs. „precision“ Verlauf für ein Realbild mit einer Auflösung von 640×480 Pixel und ein Infrarotbild mit einer Auflösung von 320×240 Pixel bei einer Rotation von $\alpha = 50^\circ$	33
4.10 Testbilder für das „recall“ vs. „1-precision“-Diagramm, Infrarotbilder mit einer Auflösung von 320×240 Pixel (links) und Realbilder mit einer Auflösung von 640×480 Pixel (rechts)	33
4.11 Testbild mit 100% positiven Matches	34
4.12 Gegenüberstellung der detektierten Key-Points in den Testbildern	35
4.13 Gegenüberstellung der Berechnungszeiten	35
4.14 CPU Berechnungszeiten für das Matching ohne die Auswertung des Vorzeichens der Hesse-Matrix	36
4.15 CPU Berechnungszeiten für das Matching inklusive der Auswertung des Vorzeichens der Hesse-Matrix	37
4.16 Faktor der Zeitersparnis für verschiedene Einstellungen des Algorithmus bei Verwendung des Vorzeichens der Hesse-Matrix	37

2 Einleitung

2.1 Zielsetzung

Ziel dieser Arbeit ist es, einen Key-Point basierten Algorithmus der Computer Vision näher zu betrachten und dessen Anwendbarkeit auf Infrarotbilder zu untersuchen. Dafür bieten sich mehrere leistungsfähige Methoden an, wobei in dieser Arbeit der Speeded Up Robust Features Algorithmus detaillierter vorgestellt wird.

2.2 Key-Point basierte Algorithmen

In der Bildverarbeitung gewinnen Algorithmen auf der Basis von Key-Points immer mehr an Bedeutung, da sie für gewisse Problemstellungen die einzige Lösungsmöglichkeit darstellen. Auch die in den letzten Jahren gestiegene Rechnerleistung hat es ermöglicht, einige dieser Algorithmen sehr effizient zu nutzen, ohne dafür spezielle Hardware einsetzen zu müssen.

Die meisten dieser Algorithmen arbeiten nach dem folgend kurz zusammengefassten Prinzip: In einem Bild werden „auffällige“ Punkte gesucht, wie zum Beispiel Ecken oder Kanten. Jene Punkte werden nach gewissen Kriterien bewertet und gegebenenfalls als stabile Merkmalspunkte detektiert. Jeder dieser Punkte erhält einen einzigartigen Eigenschaftsvektor fester Länge, welcher die Eigenschaften des Punktes bzw. seiner Umgebung beschreibt. Diese so genannten Key-Points, auch Features oder Schlüsselpunkte genannt, können in späteren Schritten verwendet werden um Aufgaben, wie zum Beispiel das Tracken eines Gegenstandes oder Bewegungsanalysen von Gegenständen, zu lösen.

2.2.1 Algorithmen

2.2.1.1 Speeded Up Robust Features - SURF

Speeded Up Robust Features (kurz SURF) stellt einen Key-Point basierten Bildverarbeitungsalgorithmus dar, welcher 2006 erstmals von Bay H., in [Bay, 2006] vorgestellt wurde. Der SURF Algorithmus bietet die Möglichkeit Key-Points in einem Bild, skalen- und rotationsinvariant und robust gegenüber Bildrauschen zu detektieren. Dabei weist dieser Algorithmus wesentliche Performance-Vorteile gegenüber

anderen vergleichbaren Algorithmen, wie zum Beispiel dem SIFT Algorithmus (vgl. Punkt 2.2.1.2) auf. Für die detailliertere Betrachtung des SURF Algorithmus wird an dieser Stelle auf Kapitel 3 verwiesen.

2.2.1.2 Scale Invariant Feature Transform - SIFT

Der Algorithmus Scale Invariant Feature Transform (kurz SIFT) wurde von D. Lowe 2004 in [Lowe, 2004] vorgestellt und ist ebenfalls ein Key-Point basierter Algorithmus. Wie auch beim SURF Algorithmus werden beim SIFT Schlüsselpunkte in einem Bild detektiert, welche skaleninvariant, rotationsinvariant und in gewissem Maß unempfindlich gegen Änderung der Kameraposition sind. Der SIFT Algorithmus benutzt zur Detektion der Schlüsselpunkte in den verschiedenen Skalen den „Difference of Gaussian“-Operator (kurz DoG-Operator). Dazu wird eine Image-Pyramide¹ verwendet, durch welche sich die Approximation des „Laplacian of Gaussian“² Operators verwirklichen lässt. Die aus den Gauß-gefilterten Bildern erstellten Differenzbilder werden auf Maxima bzw. Minima untersucht, welche anschließend auf Subpixel-Genauigkeit lokalisiert werden. Nach der gegebenenfalls positiven Detektion von Key-Points werden noch zu kontrastarme Punkte ausgefiltert. Für die verbleibenden Schlüsselpunkte wird die Orientierung sowie ein Eigenschaftsvektor mit der Länge von 128 Einträgen, auf der Basis des Gradienten der Umgebung des Punktes, bestimmt.

Der SIFT Algorithmus weist ein sehr gutes Verhalten in Puncto Stabilität und Wiederholbarkeit auf, was für viele Aufgaben sehr interessant ist. Durch den 128 Einträge langen Eigenschaftsvektor und den DoG-Operator treten für den SIFT Algorithmus im Vergleich zu anderen Methoden lange Rechenzeiten auf, was den Algorithmus für Online-Auswertungen oft nicht einsetzbar macht. Abhilfe kann hier eine Implementierung auf einer „schnellen“ Grafikkarte³ schaffen, was einen wesentlichen Performance-Gewinn mit sich bringt. Für nähere Informationen zum SIFT Operator, zu den Anwendungsbereichen und de Implementierung wird an dieser Stelle aber auf die Literatur wie [Lowe, 2004] verwiesen.

2.2.2 Anwendungsbereiche

Im folgenden Abschnitt werden einige mögliche Anwendungsgebiete angeführt, um die vielseitigen Einsatzmöglichkeiten für solche Algorithmen aufzuzeigen.

¹Vgl. die Punkte 3.2.3.1 und 3.2.3.3

²Vgl. Punkt 3.2.3

³GPU - Graphics Processing Unit

2.2.2.1 Allgemein

Ein großer Anwendungsbereich solcher Methoden liegt in der Robotik, auf welche im nächsten Punkt näher eingegangen wird. Im kommerziellen Bereich finden Algorithmen wie SIFT und SURF bereits Anwendung in Bildbearbeitungsprogrammen. Dabei werden die Algorithmen genutzt um beispielsweise Gesichter in Bildern zu erkennen und zuzuordnen oder aus mehreren Bildern eine Panorama-Ansicht zu erstellen. Weiters können Aufgaben wie das Finden eines Gegenstandes in verschiedenen Szenen, Bewegungsanalysen von Objekten oder auch 3D Rekonstruktionen aus Stereobildern gelöst werden. Eine weitere Anwendungsmöglichkeit stellt die automatische Kamerakalibrierung dar, die mit Hilfe solcher Algorithmen verwirklicht werden kann.

2.2.2.2 Robotik

Die Bildverarbeitung stellt für die Robotik eine der wichtigsten Bereiche dar, da oft nur so genügend Informationen für die Berechnungen der weiteren Schritte und Handlungen des Roboters gewonnen werden können. Speziell für autonom agierende Roboter erweisen sich Key-Point basierte Methoden als sehr hilfreich um zum Beispiel Daten für die Navigation zu gewinnen. In Methoden wie Mono-SLAM⁴ und Vision based SLAM werden Key-Points dazu verwendet um daraus die relative Bewegung eines Roboters zu berechnen und somit auf Techniken wie Odometrie⁵, welche sich oft als fehleranfällig erweisen, verzichten zu können.

Folgend wird der RoboCup RescueLeague-Bewerb näher beschrieben, da dort vermehrt auf Techniken der Computer Vision zurück gegriffen wird. Bei diesem Bewerb geht es darum, dass ein voll- oder teilweise autonom agierender Roboter einen Parcours aus Hindernissen durchquert. Bei der Fahrt durch diesen Parcours müssen simulierte Opfer vom Roboter gefunden und in einer vom Roboter erstellten Karte eingezeichnet werden. Der Parcours ist in sechs verschiedene Zonen eingeteilt, welche den Schwierigkeitsgrad des zu absolvierenden Gebiets und die zu erledigenden Aufgaben bestimmen.

⁴Als SLAM-Simultaneous Localization and Mapping wird die Lokalisierung in einem unbekannten Territorium und das Erstellen einer Karte der Umgebung bezeichnet. Mono-SLAM stellt eine dieser Methoden dar, wobei bei dieser Methode eine Kamera als Odometrieersatz verwendet wird.

⁵Als Odometrie wird die Gewinnung der zurückgelegten Wegstrecke aus der Anzahl der Umdrehungen der Räder bezeichnet, beispielsweise über Inkrementalgeber an der Motorwelle.

Folgende Abbildung stellt schematisch drei dieser Zonen dar:



Abbildung 2.1: Gelbe, Orange und Rote Zone (v.o.) eines RoboCup Rescue Parcours
(Quelle: [Homepage, NIST])

Wie sich aus Abbildung 2.1 zeigt, müssen sich die Roboter über schwieriges Terrain hin fortbewegen, wodurch sich ein auf Odometrie basierender SLAM Algorithmus nur schwer verwirklichen lässt. Wie man ebenfalls erkennen kann stellt die Bildverarbeitung eine wichtige Komponente der Sensorik dar, um die simulierten Opfer zu detektieren. Ein wesentlicher Bestandteil ist dabei die Infrarotkamera, da Opfer teilweise nur aufgrund der Wärmeinformation gefunden werden können.⁶

⁶Für nähere Informationen zur Thermographie für einen Rescue Roboter und zur Rescue League wird auf vorhergehende Arbeiten wie [Hatheier, 2009] verwiesen.

3 Speeded Up Robust Features – SURF

3.1 Allgemeines

Der SURF Algorithmus zählt zu den Key-Point basierten Algorithmen der Bildverarbeitung, genauer der Computer Vision. Dieser Algorithmus ermöglicht es, unter der Verwendung von Integral Images, so genannte Schlüsselpunkte aus einem Bild zu extrahieren. Zur Detektion solcher Key-Points wird ein Hesse-Matrix basierter Feature-Detektor verwendet, welcher eine skalen- und rotationsinvariante Detektion der Features ermöglicht. Die extrahierten Punkte werden durch einen Eigenschaftsvektor mit 64 Einträgen charakterisiert.

In den folgenden Punkten werden die genannten Bestandteile des Speeded Up Robust Features Algorithmus näher erläutert. Dabei wird speziell auf die Verwendung der Methoden in Verbindung mit SURF eingegangen.

3.2 Algorithmus

In diesem Abschnitt werden die einzelnen Komponenten des SURF Algorithmus näher erläutert.

3.2.1 Integral Image Darstellung¹

Die Berechnung von Integral Images stellt eine neue Darstellungsform von Bildern dar, welche in der Bildverarbeitung erstmals von Viola P. und Jones M. in [Viola, Jones, 2001] verwendet wurde. Ursprünglich stammt die Berechnung von Integral Images oder auch „Summed Area Tables“ aus dem Bereich der Computer Graphik wo diese für die Texturberechnung von 3D Objekten verwendet wird.

Die Berechnung eines Integral Images erfolgt durch die Bildung der Summe der Intensitätswerte eines Bildes, welche links bzw. oberhalb des Pixels an Position (x, y)

¹Vgl. [Derpanis, 2007] Vgl. [Viola, Jones, 2001]

liegen. Folgende Formel stellt diesen Zusammenhang dar, wobei $I(x, y)$ das Originalbild und $II(x, y)$ das Integral Image an der Position (x, y) darstellt:

$$II(x, y) = \sum_{i \leq x, j \leq y} I(i, j) \quad (3.1)$$

Folgende Abbildung veranschaulicht die Vorgehensweise zur Berechnung eines Integral Images anhand eines Ausschnitts der Datenmatrix eines Graustufenbildes.

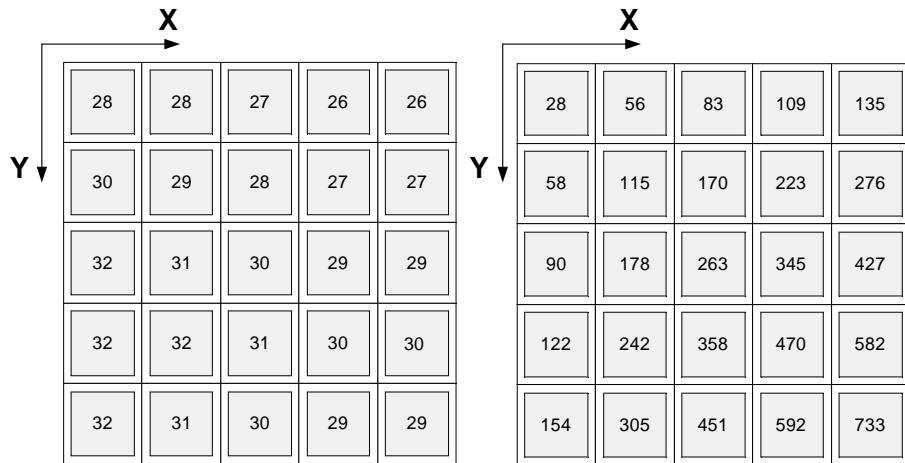


Abbildung 3.1: Vergleich Originalbild-Daten (links) und Integral Image-Daten (rechts)

Aus folgender Abbildung ist ein berechnetes Integral Image im Vergleich zu einem Infrarotbild ersichtlich, woraus sich zeigt, dass diese Darstellungsform nur für Berechnungen und nicht zur Visualisierung der Daten geeignet ist.

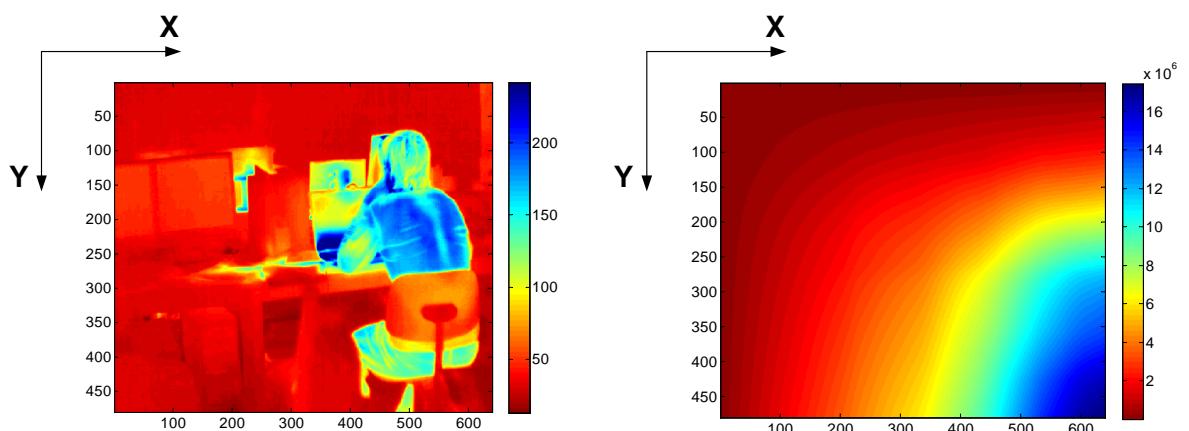


Abbildung 3.2: Vergleich IR-Bild und Integral Image

Für die Berechnung eines Integral Images in einem Programmcode besteht folgender rekursive Zusammenhang, wobei gilt:

$$s(x, -1) = II(-1, y) \equiv 0 \quad (3.2)$$

$$s(x, y) = s(x, y - 1) + I(x, y) \quad (3.3)$$

$$II(x, y) = II(x - 1, y) + s(x, y) \quad (3.4)$$

Die Anwendung von Integral Images bringt einen wesentlichen Vorteil in der Berechnung von Flächensummen, da diese nun, wie in Abbildung 3.3 dargestellt, mit einfachen Additionen und Subtraktionen von Pixelwerten berechnet werden können. Folgende Beispiele stellen die Berechnung verschiedener Flächensummen dar, wobei $A B C D$ die Flächeninhalte und sX die Pixelwerte des Integral Images darstellen.

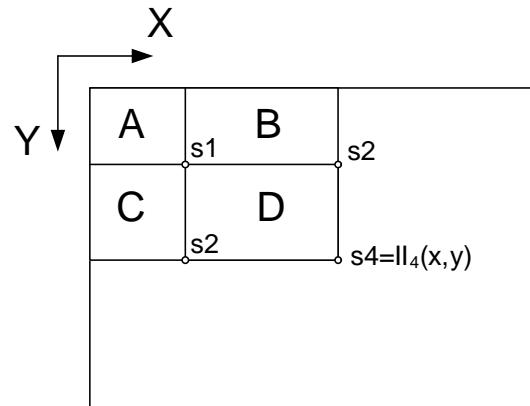


Abbildung 3.3: Flächensummen in einem Integral Image

Durch folgende Ausdrücke können die oben dargestellten Flächen berechnet werden:

$$A = s1 \quad (3.5)$$

$$B = s2 - s1 \quad (3.6)$$

$$C = s3 - s1 \quad (3.7)$$

$$D = s4 + s1 - s2 - s3 \quad (3.8)$$

Für den SURF Algorithmus muss das Integral Image aus dem Eingangsbild einmalig berechnet werden, was in Verbindung mit den in Punkt 3.2.3 beschriebenen

Box-Filtern einen wesentlichen Performance-Gewinn bedeutet. Weiters ergeben sich durch die einfachen Additionen und Subtraktionen konstante Berechnungszeiten für die Flächen, was sich ebenfalls positiv für die Performance des Algorithmus auswirkt.

3.2.2 Feature-Detektion

Zur Detektion von so genannten Key-Points wird beim Speeded Up Robust Features Algorithmus ein Hesse-Matrix basierter Detektor verwendet. Dieser weist im Vergleich zum meist verwendeten Harris-Detektor eine Skaleninvarianz auf. Wie in [Bay, Ess, Tuytelaars, Van Gool, 2008] beschrieben, gibt es neben dem Harris-Detektor auch den Harris-Laplace bzw. Hessian-Laplace Detektor, welche beide ebenfalls eine Skaleninvarianz für die gefundenen Punkte aufweisen. Im Speziellen wird beim Hessian-Laplace Detektor die Determinante der Hesse-Matrix² dazu verwendet, die Schlüsselpunkte zu lokalisieren und die Spur³ der Hesse-Matrix $tr(\mathcal{H}(\vec{x}))$ (welche gleichbedeutend dem Laplace⁴-Wert des Bildpunktes ist) für die Bestimmung der Skale. Aus [Bay, Tuytelaars, Van Gool, 2008] geht hervor, dass der im folgenden Punkt beschriebene Hesse-Matrix basierte Detektor, der nur die Information aus der Determinante der Hesse-Matrix nutzt, gewisse Vorteile gegenüber den anderen Detektoren aufweist.

²Die Hesse Matrix stellt eine Matrix aus den zweiten Ableitungen einer mehrdimensionalen Funktion $f(\vec{x})$ dar.[Richter, 2001] Folgendes Beispiel zeigt die Hesse Matrix für eine zweidimensionale Funktion $f(x, y)$, $\mathcal{H}(x, y) = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}$ wobei die Determinante $det(\mathcal{H}(x, y)) = \frac{\partial^2 f}{\partial x^2} \frac{\partial^2 f}{\partial y^2} - \left(\frac{\partial^2 f}{\partial x \partial y} \right)^2$ der Hesse Matrix dazu benutzt werden kann um Punkte auf lokale Minima bzw. Maxima zu untersuchen.

³Die Spur (engl. trace) einer Matrix ist die Summe der Komponenten entlang der Hauptdiagonale einer quadratischen Matrix. Sei A eine quadratische $n \times n$ Matrix, $A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$, so gilt $tr(A) = \sum_{i=1}^n a_{ii} = a_{11} + a_{22} + \dots + a_{nn}$ [Weltner, 2006]

⁴Der Laplace - Operator ist wie folgt definiert, $\Delta f(\vec{x}) = \nabla^2 f(\vec{x})$ wobei dies die Summe der zweiten partiellen Ableitungen in Richtung \vec{x} bedeutet.[Burger, Burge, 2005/2006] Zwischen Hesse-Matrix und Laplace Operator besteht folgender Zusammenhang: $\Delta f(\vec{x}) = tr(\mathcal{H}(\vec{x}))$

Folgende Abbildung zeigt einen Vergleich mehrerer Feature Detektoren bezüglich Bildrotation, Änderung des Kamerastandpunktes, Skalenänderung und Verringerung der Helligkeit. Es zeigt sich, dass der Fast-Hessian Detektor, welcher auch beim SURF Algorithmus zum Einsatz kommt, den anderen Detektoren überlegen ist.

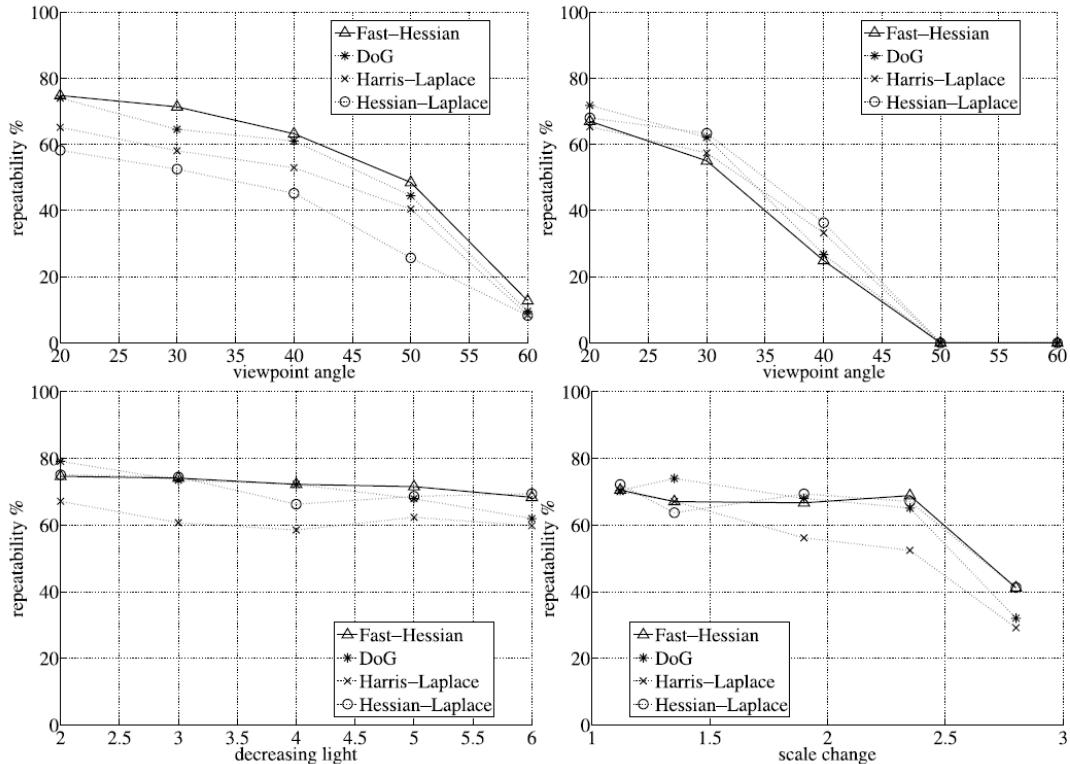


Abbildung 3.4: Vergleich von Feature-Detektoren
 (Quelle: [Bay, Ess, Tuytelaars, Van Gool, 2008])

Wie zum Beispiel aus [Mikolajczyk, Tuytelaars, 2006] und [Lindenberg, 1998] ersichtlich, existieren noch eine Reihe weiterer Feature-Detektoren welche ihre Vor- und Nachteile in gewissen Anwendungsbereichen aufweisen.

3.2.3 Hesse-Matrix basierter Feature Detektor

Der SURF Algorithmus verwendet allein die Determinante der Hesse-Matrix für die Untersuchung eines Bildes nach so genannten Schlüsselpunkten. Bevor der Detektor im einzelnen erläutert wird, wird im Folgenden noch das dafür notwendige Konzept des „Skalenraums“ vorgestellt.

3.2.3.1 Skalenraum⁵

Der Skalenraum bzw. die Skalenrepräsentation in der Bildverarbeitung stellt eine Möglichkeit dar, Elemente in einem Bild auch unter dem Gesichtspunkt der Skale

⁵Vgl. [Lindenberg, 1991]

bzw. einer „Größenordnung“ zu betrachten und diese Eigenschaft in die Beschreibung und Suche von Schlüsselpunkten mit einzubeziehen. Folgender einfache Vergleich soll das „Skalenproblem“ anhand eines Beispiels verdeutlichen.

Bei der Betrachtung eines Baumes in der Natur ist es nicht sinngemäß diesen in einer Skale bzw. in einem Maßstab von Mikrometern oder Millimetern zu betrachten. Vielmehr macht eine Betrachtung in einem Maßstab von einigen Zentimetern oder Metern Sinn. Im Bereich von Mikrometern wäre eher die Betrachtung von Eigenheiten einzelner Blatt- oder Holzfasern sinnvoll.

Ähnlich ist es auch notwendig, Eigenschaften oder Objekte in einem Bild, unter der Betrachtung einer Skale zu beschreiben. Wird die Skale nicht mit in die Untersuchungen einbezogen, könnten zwei völlig verschiedene Objekte im Vergleich als ident betrachtet werden, weil die Bilder unter verschiedenen Skalierungs-Bedingungen aufgezeichnet wurden.

Die Idee dabei ist, aus einem Bild eine ganze Anordnung von Bildern in verschiedenen Skalen zu generieren. Wie in [Lindenberg, 1991] erläutert, wird dazu das Bild als eine kontinuierliche Funktion aus Signalen dargestellt die einerseits von den Koordinaten (x, y) und vom Skalenparameter t abhängig ist. Im kontinuierlichen Fall sieht der mathematische Zusammenhang dieser Skalendarstellung folgendermaßen aus:

Für die zweidimensionale Funktion $f : R^2 \rightarrow R$ ist der Skalenraum $L : R^2 \times R_+ \rightarrow R$ durch eine Faltung mit einer zweidimensionalen Gauß-Funktion definiert⁶:

$$L(x, y, t) = \int_{\xi=-\infty}^{\infty} \int_{\eta=-\infty}^{\infty} \frac{1}{\sqrt{2\pi t}} e^{-\frac{(\xi^2+\eta^2)}{2t}} f(x-\xi, y-\eta) d\xi d\eta \quad (3.9)$$

$$L(x, y, t) = g(x, y, t) \otimes f(x, y, t) \quad (3.10)$$

Dies ist gleichbedeutend mit der Lösung der zweidimensionalen Diffusionsgleichung mit der Anfangsbedingung $L(x, y, 0) = f(x, y)$:

$$\frac{\partial L}{\partial t} = \frac{1}{2} \left(\frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2} \right) \quad (3.11)$$

Aus [Lindenberg, 1991] geht ebenfalls hervor, dass die Gauß-Funktion die beste Möglichkeit darstellt um die Skalenrepräsentation bzw. den Skalenraum zu erstellen, da es wesentlich ist, keine neuen Objekte bzw. Strukturen in höheren Skalen durch die Filterung einzuführen.

In der Bildverarbeitung wird die Skalenrepräsentation meist als Bildpyramide dargestellt, welche als „Multiresolution“-Repräsentation bezeichnet wird. Hierbei wird das

⁶ t ist gleichbedeutend mit σ^2 was der Standardverteilung der Gauß-Funktion entspricht

Originalbild wiederholt mit einem Gauß-Filter geglättet und in der Auflösung meist um einen Faktor von zwei verringert. Die Anordnung mit den ständig kleiner werdenden Bildern ergibt eine Bildpyramide, welche eine Untersuchung in verschiedenen Skalen erlaubt. In dem in Punkt 2.2.1.2 beschriebenen SIFT Algorithmus werden solche Image-Pyramiden in Verbindung mit einer DoG - „Difference of Gaussian“ Methode verwendet, um Extrema im Skalenraum zu detektieren. Die Skalenrepräsentation wird beim SURF Algorithmus in abgeänderter Weise realisiert, was in Punkt 3.2.3.3 näher dargestellt wird.

Für weitere Informationen zur Skalentheorie und der Skalenrepräsentation wird auf die ausführliche Literatur wie [Lindenberg, 1991] und [Lindenberg, 1998] verwiesen.

3.2.3.2 Fast-Hessian Detektor

Wie bereits erwähnt, benutzt der Hesse-Matrix basierte Detektor die zweite Ableitung zur Detektion von auffälligen Punkten in einem Bild. Wie in [Burger, Burge, 2005/2006] erläutert, stellen „second order derivative“ Methoden zu Extraktion von Ecken und Kanten in einem Bild, ein sehr Effizientes Mittel der Bildverarbeitung dar. Da diese Operatoren eine große Empfindlichkeit gegenüber Bildrauschen zeigen, müssen die Bilddaten vorher mit einem Gauß-Filter geglättet werden. Weiters erlaubt die Verwendung eines Gauß-Filters eine skalenabhängige Untersuchung der Daten. Die Hesse Matrix $\mathcal{H}(\vec{x}, \sigma)$ in Abhängigkeit vom Ortsvektor \vec{x} und der Standardabweichung⁷ σ , die für die Untersuchung des Bildes verwendet wird, ist wie folgt definiert:

$$\mathcal{H}(\vec{x}, \sigma) = \begin{pmatrix} L_{xx}(\vec{x}, \sigma) & L_{xy}(\vec{x}, \sigma) \\ L_{xy}(\vec{x}, \sigma) & L_{yy}(\vec{x}, \sigma) \end{pmatrix} \quad (3.12)$$

Dabei stellt $L_{xx}(\vec{x}, \sigma)$ die Faltung des Bildes $I(x, y)$ mit der „second order Gaussian derivative“ Funktion $\frac{\partial^2}{\partial x^2} g(\vec{x}, \sigma)$ in Punkt \vec{x} dar. Der gleiche Zusammenhang gilt auch für $L_{yy}(\vec{x}, \sigma)$ und $L_{xy}(\vec{x}, \sigma)$, wobei $g(\vec{x}, \sigma)$ durch folgenden Ausdruck definiert ist: $g(\vec{x}, \sigma) = \frac{1}{\sqrt{2\pi}\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$ was der zweidimensionalen Gauß-Funktion entspricht. Die „second order Gaussian derivative“ Funktion wird auch als LoG - „Laplacian of Gaussian“ Funktion bezeichnet. Die kontinuierliche LoG-Funktion L_{xy} ist in folgender Abbildung dargestellt:

⁷ σ entspricht der Standardabweichung der zweidimensionalen Gauß-Funktion

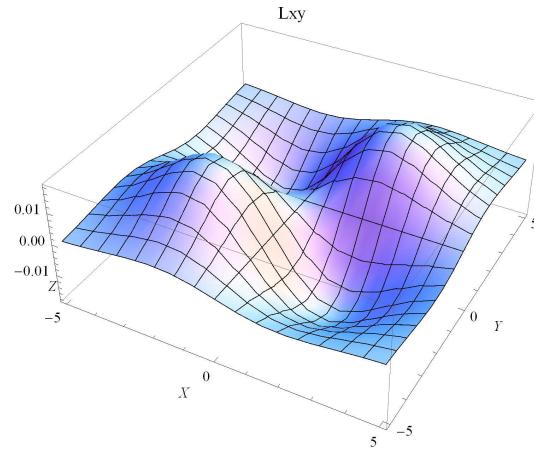


Abbildung 3.5: LoG-Funktion L_{xy} , $\sigma = 2$

Die für die Auswertung der Hesse Matrix ebenfalls benötigten partiellen Ableitung in x und y Richtung sind in folgender Abbildung dargestellt:

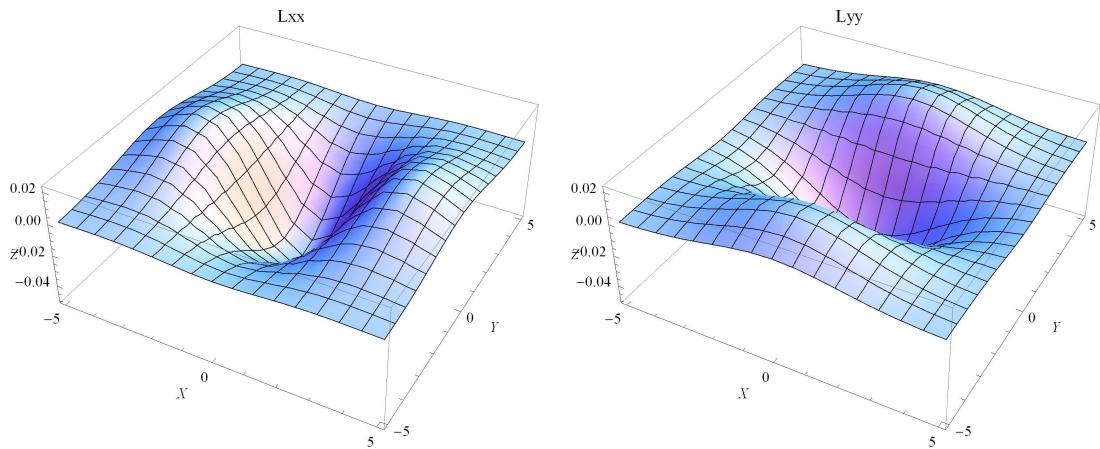


Abbildung 3.6: LoG-Funktionen L_{xx} (links) und L_{yy} (rechts), $\sigma = 2$

Zur Verwendung der „second order Gaussian derivative“-Funktion in der Bildverarbeitung muss diese diskretisiert werden um Filter-Matrizen zu erhalten, welche dann mittels Faltung auf die Bilddaten angewendet werden können. Dieser Vorgang bringt aber ein nachteiliges Verhalten der Operatoren in Puncto Bildrotation mit sich. Der LoG-Operator wurde von D. Lowe in [Lowe, 2004] für den SIFT⁸-Algorithmus durch den DoG - „Difference of Gaussian“ Operator approximiert, um die Extrema unter Verwendung einer Image-Pyramide im Skalenraum zu detektieren. Diese Approximation wird im SURF Algorithmus noch erweitert. Um eine effiziente Berechnung der Filterantworten zu ermöglichen, werden für die Filter-Matrizen so genannte „Box-Filter“ verwendet. In Verbindung mit der Darstellung der Bilddaten als Integral Images lassen sich somit die Filterantworten unabhängig von der Größe des Filters effizient und in konstanter Zeit berechnen. In nachfolgender Abbildung 3.7 sind die

⁸Vgl. Punkt 2.2.1.2

im SURF Algorithmus verwendeten Box-Filter dargestellt:

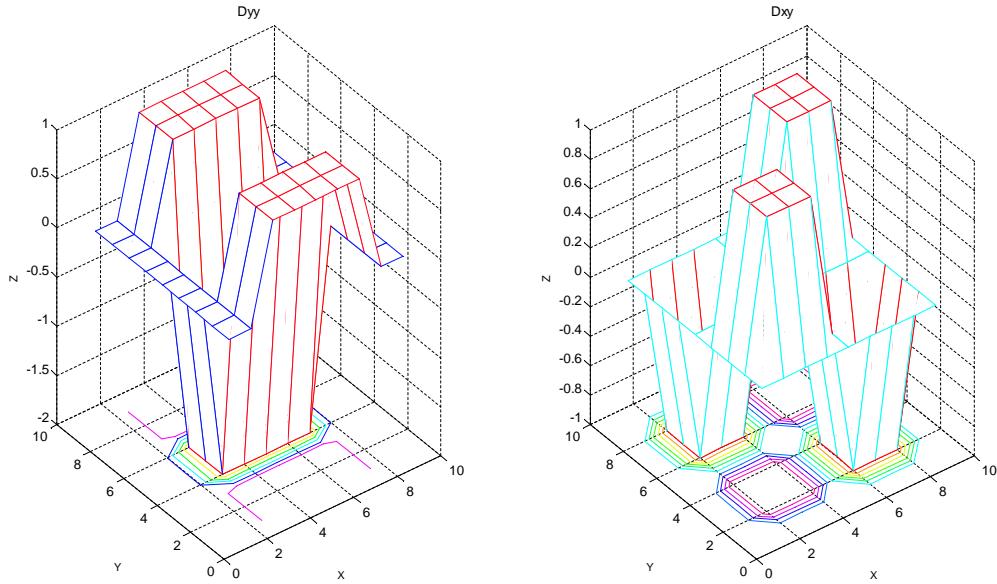


Abbildung 3.7: D_{yy} und D_{xy} Box-Filterkerne der Größe 9×9 mit $\sigma = 1.2$

Diese beiden Filter repräsentieren die Approximation der LoG-Funktion, wobei diese beiden Filter-Matrizen die niedrigste im SURF Algorithmus verwendete Skale mit $\sigma = 1,2$ darstellen. Die Determinante der Hesse-Matrix kann mit Hilfe der Richtungsableitungen durch nachstehende Formel 3.13 berechnet werden. Dabei erfolgt für die einzelnen Filterantworten eine Gewichtung mittels Parameter w .

$$\det(\mathcal{H}_{approx}) = D_{xx}D_{yy} - (w D_{xy})^2 \quad (3.13)$$

Die Gewichtung wird auf Grund der Approximation der kontinuierlichen LoG-Funktion durch die Box-Filter benötigt, wobei die Gewichtung für jede Skale berechnet werden kann. Da sich die Gewichtung, wie aus [Bay, 2006] hervorgeht, nicht maßgeblich auswirkt, wird an dieser Stelle auf die Berechnung nicht mehr näher eingegangen. Im SURF Algorithmus wird für die Gewichtung ein konstanter Wert von $w = 0,9$ verwendet. Bei der Anwendung der Filter auf die Bilddaten, werden diese gemäß ihrer Größe normiert um eine konstante Frobenius⁹-Norm für alle Skalen zu garantieren. Die in den verschiedenen Skalen erhaltenen Werte für die Determinante der Hesse-Matrix werden in einer so genannten Blob-Response Map gespeichert. Wie in folgendem Punkt beschrieben, werden diese Maps auf lokale Minima und Maxima untersucht um invariante Schlüsselpunkte zu erhalten.

⁹Die Frobenius-Norm für eine $n \times n$ Matrix ist wie folgt definiert: $\| A \|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^n a_{ij}^2}$
[Herrmann, 2007]

3.2.3.3 SURF-Filterpyramide

Wie schon in Punkt 3.2.3.1 beschrieben ist es notwendig, die Extrema in verschiedenen Skalen zu untersuchen. Oftmals wird dazu eine Image-Pyramide verwendet. Diese Vorgehensweise wird beim SURF Algorithmus nicht benötigt, da die Box-Filter in Verbindung mit der Darstellung des Bildes als Integral Image eine effiziente Berechnung selbst von großen Filtermatrizen erlauben. Beim SURF Algorithmus wird nicht das Bild wiederholt mit einem Gauß-Filter geglättet und dann um einen konstanten Faktor verkleinert, sondern die Filtermatrizen werden vergrößert und immer wieder auf das Originalbild angewandt. Dieses Vorgehen hat des weiteren den Vorteil, dass keine Aliasing-Effekte auftreten, wie das bei der Verkleinerung der Bilder Fall sein kann. Folgende Abbildung veranschaulicht den Vergleich der Image-Pyramiden und der Filter-Pyramiden die im SURF Algorithmus Anwendung finden:

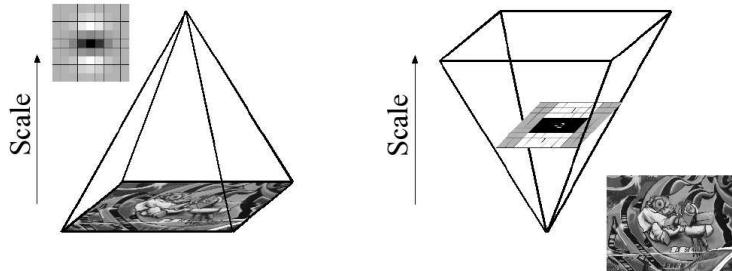


Abbildung 3.8: Image Pyramide (links) und Filter Pyramide (rechts)
 (Quelle: [Bay, Ess, Tuytelaars, Van Gool, 2008])

Die in Abbildung 3.7 dargestellten Filterkerne werden für die Berechnung einer neuen Skale um einen konstanten Faktor vergrößert, wobei der kleinste verwendete Filter ein 9×9 Filter ist mit einer Skale $s = 1,2$, was der Standardverteilung einer Gauß-Funktion von $\sigma = 1,2$ entspricht. Der Skalenraum wird bei der Untersuchung in Oktaven unterteilt, wobei jede Oktave mehrere Bilder mit unterschiedlichen Skalen enthält. Die Skale steigt mit der Größe der Filtermatrix welche auf das Bild angewendet wird. Die erste Oktave startet wie bereits erwähnt mit einer Filtergröße von 9×9 Pixel, entsprechend einer Skale von $s = 1,2$.

Bei der Vergrößerung des Filters wird die Breite der werte-behafteten Bereiche erhöht. Bei diesem Vorgang muss darauf geachtet werden, dass ein zentrales Pixel vorhanden sein muss. Aus diesem Grund kann sich der Filter nur um eine gerade Anzahl an Pixel und um minimal zwei Pixel pro Bereich erhöhen. Wie aus Abbildung 3.7 ersichtlich ist, enthalten die D_{xx} und D_{yy} Filtermatrizen drei solche Bereiche, wodurch sich eine Vergrößerung der Matrix um sechs Pixel ergibt. Folgende Abbildung stellt die Vergrößerung der Filtermatrix dar:

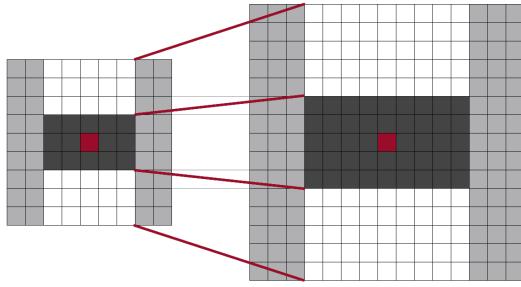


Abbildung 3.9: Vergrößerung der Filtermatrix von 9×9 auf 15×15 (Quelle: [Bay, 2006])

Dabei ergeben sich für die erste Oktave folgende Filtergrößen: 9×9 , 15×15 , 21×21 und 27×27 . Für die weiteren Oktaven wird der Faktor der Vergrößerung jeweils verdoppelt, wodurch sich für die zweite Oktave folgende Filtergrößen ergeben: 15×15 , 27×27 , 39×39 , 51×51 und für die dritte Oktave ergeben sich die Größen: 27×27 , 51×51 , 75×75 , 99×99 . Ist das Bild groß genug, kann auch noch eine vierte Oktave berechnet werden welche dann folgende Filtergrößen verwendet: 51×51 , 99×99 , 147×147 , 195×195 . Wie in [Bay, 2006] erläutert könnten noch weitere Oktaven berechnet werden. Wie sich jedoch in Untersuchungen gezeigt hat, ist dies nicht notwendig, da die Anzahl der gefundenen Punkte in weiteren Oktaven stark abnimmt.

Die Schlüsselpunkte werden mittels einer „non-maximum suppression“ (NMS) in einer $3 \times 3 \times 3$ Nachbarschaft detektiert, wobei die Lokalisierung dabei auf Subpixel-Genauigkeit durchgeführt wird. Aus diesem Grund können in den niedrigsten und höchsten Skalen jeder Oktave keine Extrema gefunden werden, da diese nur zu Vergleichszwecken verwendet werden. Für die Skale bedeutet dies, dass die niedrigste Skale die betrachtet wird bei $s = 1,6$ liegt. Dies ergibt sich aus der späteren Interpolation zwischen den Skalen, wobei die Skale $s = 1,6$ einer Filtergröße von 12×12 entsprechen würde, was einer Interpolation zwischen den Filtergrößen 9×9 und 15×15 gleichkommt. Die interpolierte Skale berechnet sich wie folgt:

$$s_{neu} = s_{Anfang} \frac{Filtergroesse_{Neu}}{Filtergroesse_{Anfang}} = 1,2 \frac{12}{9} = 1,6 \quad (3.14)$$

Ähnliche Überlegungen gelten auch für die maximal erreichte Skale in der ersten Oktave. Für höhere Oktaven erfolgt die Berechnung auf die selbe Weise, wobei für eine genauere Beschreibung auf [Bay, 2006] verwiesen wird.

Um in der ersten Oktave einen zu großen Skalensprung, zwischen dem 9×9 und dem 15×15 Filter von von $\frac{15}{9} = 1,7$ zu vermeiden, wird beim SURF Algorithmus zusätzlich noch eine feinere Skalarendarstellung verwendet. Dabei wird das Image durch lineare Interpolation um einen Faktor von 2 vergrößert. Weiters wird für die

erste Oktave ein Filter der Größe 15×15 verwendet, wobei die restlichen Filter dieser Oktave folgende Größen aufweisen: 21×21 , 27×27 , 33×33 . Für die höheren Oktaven erfolgt die Vergrößerung der Matrizen wie im oberen Absatz beschrieben. Durch diese Vorgehensweise wird der Skalensprung von $\frac{15}{9} = 1,7$ auf von $\frac{21}{15} = 1,4$ verringert.

3.2.3.4 Lokalisierung der Schlüsselpunkte¹⁰

Bei der Lokalisierung der Schlüsselpunkte wird in einem ersten Schritt die Determinante der Hesse-Matrix für jeden Datenpunkt im Bild berechnet. Dabei können durch einen Threshold mögliche Schlüsselpunkte detektiert werden. Im nächsten Schritt wird einen „non-maximum suppression“ durchgeführt, wobei für jeden möglichen Schlüsselpunkt überprüft wird ob dieser in einer 27er Nachbarschaft, wie in folgender Abbildung dargestellt wird, ein lokales Maxima/Minima darstellt.

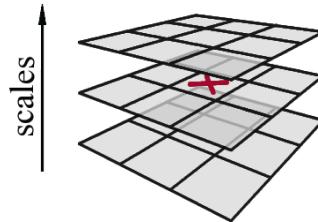


Abbildung 3.10: $3 \times 3 \times 3$ Nachbarschaft bei der „non-maximum suppression“ (Quelle: [Bay, 2006])

Wie aus Abbildung 3.10 hervorgeht, wird bei der NMS der mögliche Schlüsselpunkt mit seinem Nachbarn in der selbe Skale und den Nachbarn in der Skale darunter bzw. darüber verglichen.

Für die somit lokalisierten Schlüsselpunkte muss noch die Position bestimmt werden. Dies geschieht durch die Interpolation einer quadratischen Funktion, an der Position des Schlüsselpunktes in der Blob Response Map (Determinante der Hesse-Matrix).[Bay, 2006] Dazu wird die Determinante der Hesse-Matrix $H(x, y, s)$, die eine Funktion vom Ort sowie der Skale darstellt, in eine Taylor-Reihe entwickelt. Der Vektor \vec{x} stellt dabei die Verschiebung von der ursprünglichen Position des lokalisierten Schlüsselpunktes dar:

$$H(\vec{x}) = H + \frac{\partial H^T}{\partial \vec{x}} \vec{x} + \frac{1}{2} \vec{x}^T \frac{\partial^2 H}{\partial \vec{x}^2} \vec{x} \quad (3.15)$$

Die Position des Schlüsselpunktes kann dabei durch Ableiten und Null setzen dieser Funktion gefunden werden, wodurch sich folgender Ausdruck für $\vec{x} = (x, y, s)$ ergibt:

¹⁰Vgl. [Bay, 2006]
Vgl. [Lowe, 2004]

$$\vec{x} = \left(\frac{\partial^2 H}{\partial \vec{x}^2} \right)^{-1} \frac{\partial H}{\partial \vec{x}} \quad (3.16)$$

Die Ableitungen werden durch Differenzen aus benachbarten Pixeln realisiert. Ergibt sich für eine der Komponenten des Vektors \vec{x} dabei ein Wert $> 0,5$, wird die Position angepasst und die Interpolation erneut ausgeführt. Diese Schritte werden solange wiederholt bis für jede Komponente $< 0,5$ gilt, oder eine maximale Anzahl an Iterationen erreicht ist. Jene Punkte, bei welchen diese Berechnung nicht konvergiert, werden aussortiert und nicht mehr weiter betrachtet. Die Interpolation auf Subpixel-Pixel Genauigkeit bei Position und Skale bewirkt beim SURF Algorithmus eine besondere Verbesserung, da im Allgemeinen die einzelnen Skalen bzw. Oktaven weit auseinander liegen.

3.2.4 Interest Point Descriptor

Der Interest Point Descriptor des SURF Algorithmus beschreibt die Intensitätsverteilung der Pixel um den Schlüsselpunkt, ähnlich dem Descriptor der beim SIFT Algorithmus zum Einsatz kommt. Mit Hilfe von Haar-Wavelets wird die erste Ableitung in x und y Richtung in der Umgebung des Schlüsselpunktes berechnet und daraus ein Eigenschaftsvektor mit 64 Einträgen erstellt.

3.2.4.1 Bestimmung der Orientierung

Um eine Rotationsinvarianz zu garantieren, wird in einem ersten Schritt die Ausrichtung des Schlüsselpunktes berechnet. Die folgenden Berechnungen zu den Eigenschaften der Umgebung des Schlüsselpunktes werden alle relativ zu dieser Ausrichtung berechnet. In einer Umgebung vom Radius $6s$, wobei s den Skalenparameter darstellt, werden die Filterantworten von Haar-Wavelets berechnet. Die Haar-Wavelets stellen eine Ableitung in x und y Richtung dar. Diese werden der jeweiligen Skale in der sich der Schlüsselpunkt befindet, durch eine skalen-abhängige Filtergröße von $4s$, angepasst. Die verwendeten Haar-Wavelets sind beispielhaft in folgender Abbildung dargestellt:



Abbildung 3.11: Haar-Wavelet Filter in x Richtung (links) und in y Richtung (rechts) (Quelle: [Bay, Ess, Tuytelaars, Van Gool, 2008])

Die in Abbildung dargestellten Filter weisen eine Gewichtung von +1 Weiß und -1 Schwarz auf. Die mit Hilfe der Wavelets berechneten Filterantworten werden mit

einem Gauß-Filter $\sigma = 2$ gewichtet, wobei das Zentrum des Filters an der Position des Schlüsselpunktes liegt. Die gewichteten Punkte werden abhängig vom Wert der Ableitung in x und y Richtung in einem Diagramm eingezeichnet, wie aus folgender Abbildung hervorgeht:

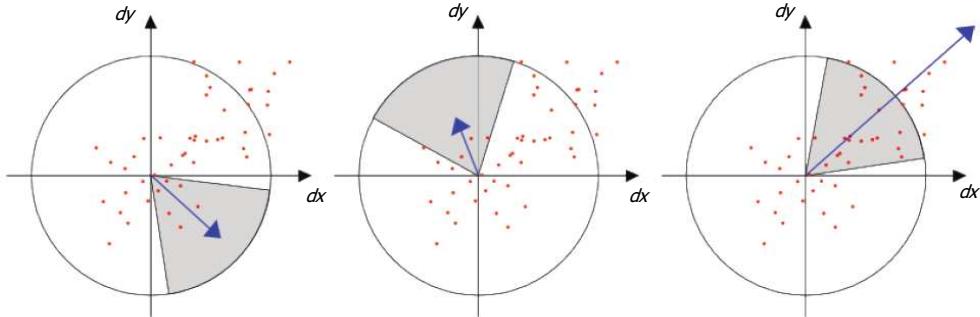


Abbildung 3.12: Diagramm zur Bestimmung der Orientierung
(Quelle: [Evans, 2009])

Die dominierende Orientierung wird durch die Verschiebung eines $\frac{\pi}{3}$ großen Kreisfensters bestimmt. Für jedes Fenster wird ein Richtungsvektor, welcher die vektorielle Summe aller im Fenster enthaltenen Punkte darstellt, berechnet. Der längste so berechnete Richtungsvektor stellt die dominierende Ausrichtung des Schlüsselpunktes dar und dessen Orientierung wird für die weiteren Berechnungen verwendet.

Wird bei der Anwendung von SURF keine Rotationsinvarianz benötigt, besteht die Möglichkeit den Schritt der Richtungsbestimmung nicht durchzuführen. Dieser Algorithmus ist als Upright-SURF (kurz U-SURF) bekannt. Bei der rotationsvarianten Version des SURF Algorithmus ist eine Robustheit der Schlüsselpunkte bis zu einer Bildrotation von $\pm 15^\circ$ zu erwarten, wie aus [Bay, Tuytelaars, Van Gool, 2008] hervorgeht.

3.2.4.2 Bestimmung des Descriptors

Für die Berechnung eines Eigenschaftsvektors wird ein skalens abhängiges Quadrat der Größe $2s$ über den Schlüsselpunkt gelegt. Dieses Quadrat wird nach der Orientierung des Schlüsselpunktes ausgerichtet und in 4×4 Subregionen eingeteilt. Diese Einteilung liefert laut [Bay, 2006] die besten Ergebnisse um noch einen schnell zu verarbeitenden und gleichzeitig einen verlässlichen Descriptor zu erhalten. Für jede der 16 Subregionen werden 5×5 also 25 Punkte bewertet, welche über die Region gleichmäßig verteilt werden. Für diese Konstellation werden anschließend die Antworten für die Haar-Wavelet Filter, welche in Abbildung 3.11 dargestellt sind, der Größe $2s$ berechnet. Folgende Abbildung verdeutlicht diese Schritte:

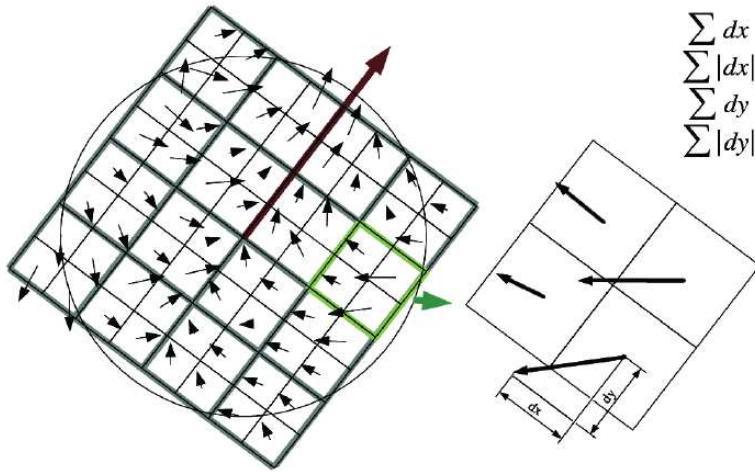


Abbildung 3.13: Berechnung des Eigenschaftsvektors (Quelle: [Bay, 2006])

Die sich daraus für jeden Sample Punkt ergebenden Werte, werden folgend durch dx und dy abgekürzt. Um eine bessere Wiederholbarkeit zu erhalten und auch unempfindlicher gegen Bildrauschen zu sein, werden die Werte wie schon bei der Bestimmung der Orientierung mit einem Gauß-Filter mit $\sigma = 3,3$ geglättet. Für den Eigenschaftsvektor wird die Summe der Filterantworten für jeden Samplepunkt in der Subregion berechnet, wobei die Beträge dieser Werte summiert werden:

$$\vec{v}_{Subregion} = (\sum dx, \sum |dx|, \sum dy, \sum |dy|) \quad (3.17)$$

Aus den Vektoren für die einzelnen Subregionen setzt sich der Vektor für jeden Schlüsselpunkt zusammen, was eine gesamte Vektorlänge von 64 Einträgen ergibt.

3.3 Verwendete SURF Implementierung

Auf die Implementierung des SURF Algorithmus wird in dieser Arbeit nicht genauer eingegangen, da bereits einige Algorithmen für die nicht kommerzielle Verwendung frei zur Verfügung stehen.

3.3.1 Speeded Up Robust Features SURF

Dieses Projekt wird vom Computer Vision Laboratory der ETH Zürich für die nicht kommerzielle Verwendung auf der Homepage [Homepage, SURF] zum Download angeboten. Dieser Code stammt von Bay H., Van Gool L. und Tuytelaars T. und stellt somit die ursprüngliche Implementierung des SURF Algorithmus dar. Auf der Website wird auch auf eine GPU basierte Implementierung verwiesen welche von Van Gool L. und Cornelis N. stammt und unter folgendem Link, [Homepage, GPU-SURF], zum Download zur Verfügung gestellt wird. Die GPU basierte Variante des SURF

Algorithmus wird aber im Rahmen dieser Arbeit nicht verwendet und darum auch nicht näher betrachtet.

3.3.1.1 Matlab-Interface

Auf der Homepage der Lund Universität wird unter [Homepage, Matlab SURF] eine auf dem ursprünglichen SURF Code basierte Matlab Version des SURF Algorithmus angeboten. Dieser Code nutzt die Funktionalität von Mex-Files um den C-Code des SURF Algorithmus von [Homepage, SURF] auch im Interface von Matlab verwenden zu können. Durch die Verwendung des Matlab Interfaces eignet sich diese Implementierung nur mehr bedingt für Online-Anwendungen, da im Vergleich zu reinem C/C++ Code die Performance doch geringer ist.

3.3.2 OpenSURF

Das OpenSURF Projekt ist ein Programmcode von Evans C. in welchem der SURF Algorithmus mit Hilfe der OpenCV Bibliothek implementiert wurde. Dieses Projekt ist unter [Homepage, OpenSURF] ebenfalls frei erhältlich und relativ einfach in ein bestehendes C++ Projekt einzubinden. In dem Dokument [Evans, 2009], welches ebenfalls zum Download angeboten wird, wird der SURF Algorithmus noch einmal näher vorgestellt und einige Bestandteile der Implementierung näher betrachtet.

Für die in den folgenden Kapiteln erläuterten Versuche sowie Untersuchungen, wird der Algorithmus aus Punkt 3.3.2 verwendet.

4 SURF Key-Points und Matching

4.1 Allgemeines

In diesem Kapitel wird die Anwendung des SURF Algorithmus bzw. das Berechnen von SURF-Features näher betrachtet. Dazu werden zuerst einige Beispielbilder mit berechneten Key-Points dargestellt. Im nächsten Schritt wird das Matchen mit SURF näher betrachtet, wobei hier auch einige Performance-Kriterien des Algorithmus untersucht werden. Der Algorithmus wird in diesem Kapitel auf Realbilder und Infrarotbilder angewendet. Der Schwerpunkt der Betrachtung zielt jedoch auf die Infrarotinformationen ab.

Alle Aufnahmen die in den folgenden Punkten dargestellt werden, sind mit folgender Hardware aufgenommen worden:

- Realbilder: Logitech WebCam QuickCam E 2500
- Infrarotbilder: FLIR ThermoVision A320

Folgende Tabelle 4.1 zeigt die Kenndaten des Rechners, welcher für die Berechnung der präsentierten Testergebnisse verwendet wurde:

Komponente	Kenndaten
CPU - Intel Core 2 Quad Q9400	2,67 GHz
Arbeitsspeicher	4 GB
Motherboard - Asus P5Q Pro	
Betriebssystem - Microsoft Windows Vista	32 Bit

Tabelle 4.1: Kenndaten des Testrechners

4.2 Berechnung der Key-Points

In diesem Kapitel wird die Berechnung von Key-Points näher betrachtet, wobei alle Berechnungen mit dem Implementierung aus Punkt 3.3.2 durchgeführt werden. Da die Infrarotkamera, Bilder mit einer Auflösung von 320×240 Pixel liefert, werden auch die Realbilder größtenteils in dieser Auflösung betrachtet, um einen besseren Vergleich zu ermöglichen.

Wie schon aus der theoretischen Betrachtung des SURF Algorithmus in Kapitel 3 hervorgeht, kann der Algorithmus noch auf die jeweilige Szene angepasst werden. Dies kann mit dem so genannten Threshold für die Determinante der Hesse-Matrix erfolgen, wobei aber zu beachten ist, dass bei einer Verringerung des Thresholds auch vermehrt nicht maximal stabile Punkte in die Reihe der Schlüsselpunkte aufgenommen werden. Weiters ist es möglich, die Anzahl der Oktaven, die initiale Abtastrate bei der Detektion und die Anzahl der Filterintervalle in den einzelnen Oktaven auf die Anwendung anzupassen. Diese Einstellungen wirken sich bei Verringerung natürlich ebenfalls negativ auf die Stabilität der detektierten Schlüsselpunkte aus, so wie sich eine Vergrößerung der verwendeten Werte negativ auf die Rechenzeit auswirkt.

Beim verwendeten CodeProjekt „OpenSURF“ gibt es ebenfalls die genannten Einstellungsmöglichkeiten, die folgend noch einmal kurz angeführt sind:

- Anzahl der Oktaven (folgend okt abgekürzt, Standardwert: $okt = 4$)
- Anzahl der Filterintervalle (folgend f_i abgekürzt, Standardwert: $f_i = 4$)
- Initiale Abtastrate für die Detektion (folgend s_r abgekürzt, Standardwert: $s_r = 2$)
- Threshold (folgend t_h abgekürzt, Standardwert: $t_h = 0,0006$)

Für die angeführten Einstellungen des Algorithmus werden im Folgenden die obig angeführten Abkürzungen verwendet.

Folgende Abbildung zeigt ein Realbild und ein Infrarotbild mit den berechneten Key-Points, unter Verwendung der Standardeinstellungen des Detektors:

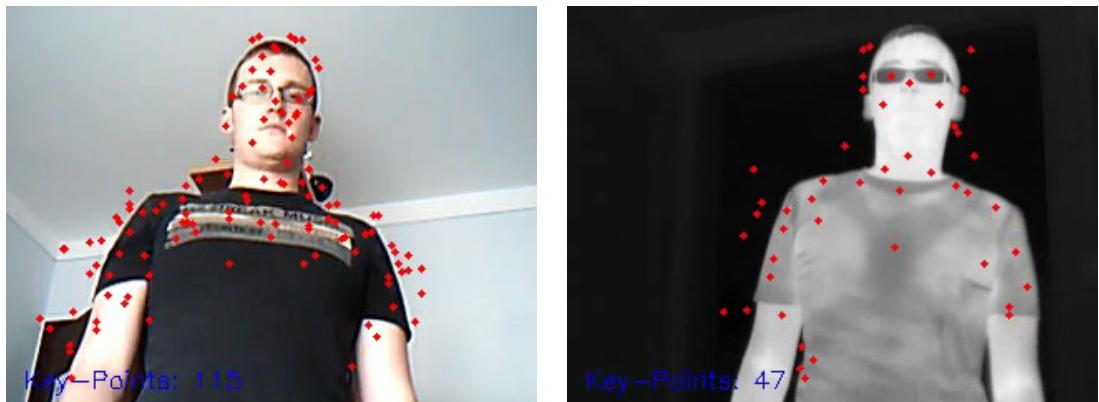


Abbildung 4.1: Realbild (links) und Infrarotbild (rechts) mit berechneten Key-Points

Wie man bereits aus Abbildung 4.1 erkennen kann, werden im Infrarotbild mit identen Detektoreinstellungen, weniger Key-Points detektiert. Um auch bei Infrarotbil-

dern eine vergleichbare Anzahl an Key-Points zu detektieren, bedarf es einer Anpassung der Einstellungen, wobei vor allem der Threshold und die Start-Abtastrate ausschlaggebend sind. Folgende Abbildung 4.2 zeigt das selbe Infrarotbild mit einem Threshold von $t_h = 0,00006$ und einer initialen Abtastrate von $s_r = 2$, was eine wesentliche Verbesserung gegenüber der Standardeinstellung bewirkt:

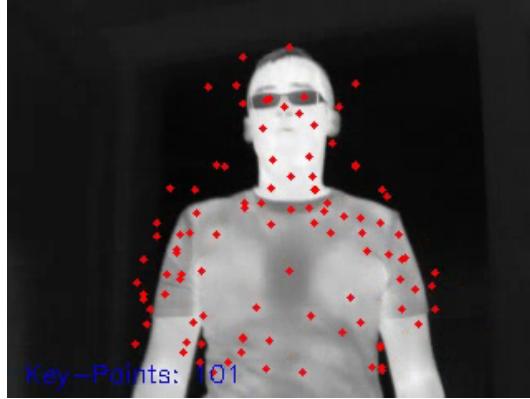


Abbildung 4.2: Infrarotbild mit angepassten Eigenschaften

Wie Abbildung 4.2 zeigt, werden mit diesen Detektoreinstellungen ähnlich viele Key-Points im Infrarotbild detektiert wie im Realbild. Folgende Tabellen 4.2 und 4.3 stellen Testergebnisse sowie detektierte Punkte und die Framerate des Algorithmus für verschiedene Einstellungen gegenüber. Als Datenquelle für die Untersuchung dienen ein Real-Video und ein Infrarot-Video mit jeweils 200 Bildern und einer Auflösung von 320×240 Pixel.

Einstellungen	Punkte Mittelwert	Punkte Gesamt	FPS Mittelwert
[okt;f _i ;s _r ;t _h]	[Anzahl]	[Anzahl]	[fps]
4;4;2;0,0006	6	1376	24,3
4;6;2;0,0006	6	1376	24,3
4;4;3;0,0001	23	4748	22,2
4;4;2;0,0001	27	5476	18,1
4;6;2;0,0001	27	5476	17,5
4;4;1;0,0001	37	7567	8,2
4;4;3;0,00001	80	16062	13,8
4;4;2;0,00001	97	19563	10,7
4;6;2;0,00001	97	19563	10,6
4;4;1;0,00001	153	30774	5,2
4;4;3;0,000001	169	33867	7,5
4;4;2;0,000001	204	40860	6,1
4;6;2;0,000001	204	40860	6,1
4;4;1;0,000001	337	67408	3,3

Tabelle 4.2: Testergebnisse Infrarot-Video (320×240 Pixel)

Einstellungen	Punkte - Mittelwert	Punkte Gesamt	FPS Mittelwert
[okt;f _i ;s _r ;t _h]	[Anzahl]	[Anzahl]	[fps]
4;4;2;0,0006	72	14421	12,7
4;6;2;0,0006	72	14421	12,5
4;4;3;0,0001	90	18067	12,3
4;4;2;0,0001	109	21880	9,6
4;6;2;0,0001	109	21880	9,6
4;4;1;0,0001	134	26828	8,9
4;4;3;0,00001	171	34283	6,9
4;4;2;0,00001	171	34283	6,8
4;6;2;0,00001	205	41041	6
4;4;1;0,00001	212	42566	4,4
4;4;3;0,000001	258	51717	4,7
4;4;2;0,000001	258	51717	4,8
4;6;2;0,000001	353	70714	3,3
4;4;1;0,000001	514	102828	2,3

Tabelle 4.3: Testergebnisse Real-Video (320 × 240 Pixel)

4.2.1 Vergleich / Ergebnisse

Wie aus den Tabellen 4.2 und 4.3 erkennbar ist, werden bei den Infrarotbildern mit identen Einstellungen für den Algorithmus weniger Punkte detektiert als bei Realbildern. Folgendes Diagramm stellt diesen Zusammenhang graphisch dar, wobei im Diagramm die gesamt detektierten Punkte aus 200 Bildern dargestellt werden.

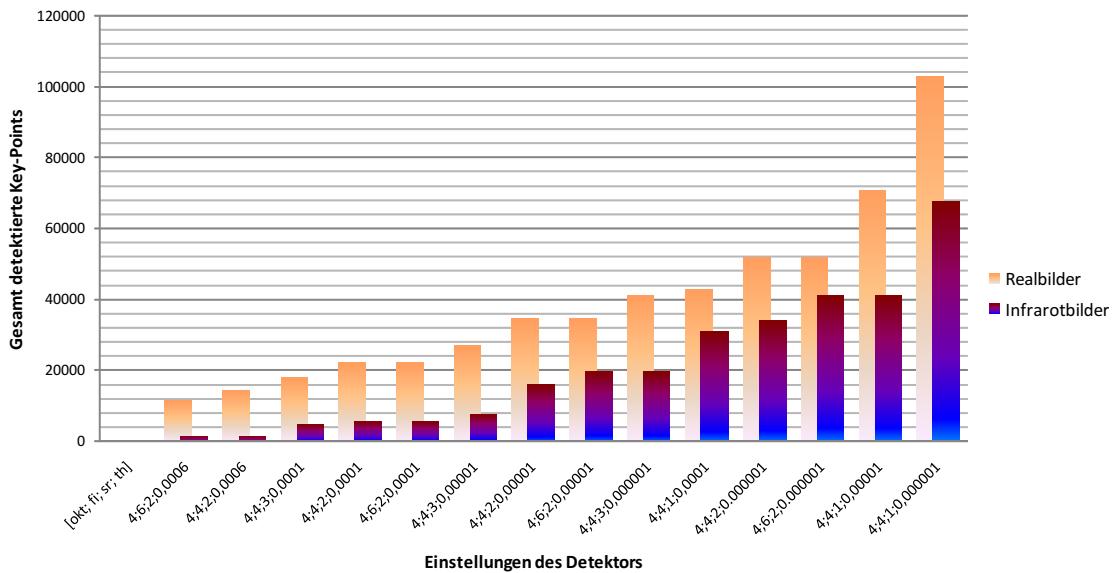


Abbildung 4.3: Diagramm der detektierten Schlüsselpunkte in IR- sowie Realbildern

Dieses Verhalten liegt an den Eigenschaften der Infrarotbilder, da diese im Vergleich zu Realbildern wesentlich weniger Details und feine Strukturen enthalten. Viele Objekte in den Szenen weisen ähnliche Temperatur auf, wodurch sich ein sehr homogenes Bild ergibt. Mit einer Anpassung der SURF-Detektoreinstellungen kann dies jedoch verbessert werden, wobei aber gerade bei Infrarotbildern darauf geachtet werden muss, nicht zu viel instabile Punkte zu detektieren. Im folgenden wird das Matching mit SURF näher betrachtet wo die Stabilität der Schlüsselpunkte eine sehr wichtige Rolle spielt.

In Puncto Rechen-Performance zeigt sich, dass die Framerate sehr schnell mit steigender Anzahl an detektierten Punkten sinkt. Für das Matchen von Bildern oder das Tracken von Objekten in Live-Aufnahmen ist eine kurze Berechnungszeit für die Key-Points ein sehr wichtiger Faktor, was wiederum dafür spricht den Detektor nicht zu empfindlich einzustellen.

4.3 Matching mit SURF

Für das Matchen von Schlüsselpunkten in zwei verschiedenen Bildern gibt es mehrere Methoden, wobei in dieser Arbeit nur auf die „Nearest-Neighbour Ratio“-Methode näher eingegangen wird. Zuvor werden allerdings die beiden Begriffe Matching und Tracking noch näher betrachtet.

Matching Unter Matching versteht man die Suche nach gleichen Schlüsselpunkten in zwei oder mehreren Bildern. Bei der Detektion in verschiedenen Bildern treten im Allgemeinen nicht genau die selben Eigenschaften für einen Schlüsselpunkt auf, wodurch dieser Schritt meist innerhalb einer gewissen Toleranz erfolgt.

Tracking Tracking beschreibt die Suche eines Objektes in einer Bildfolge, wobei die Suche nach dem Objekt wiederum über das Matchen von Schlüsselpunkten erfolgen kann. Es werden zum Beispiel 50 Schlüsselpunkte in einem Beispielbild, einem so genannten „Pattern“ oder „Template“, berechnet und diese in der Bildfolge gesucht und verfolgt. Ein typisches Anwendungsbeispiel eines für Tracking ist die Gesichtsdetektion in Live Aufnahmen.

4.3.1 Nearest Neighbour Ratio - Matching Methode

Die „Nearest Neighbour Ratio“ Matching Methode basiert auf der Euklidischen Distanz zwischen zwei Eigenschaftsvektoren. Wobei hier nicht nur die zwei nächsten „Nachbarn“ innerhalb eines bestimmten Thresholds als positives Match gewertet werden (wie bei der „Nearest Neighbour“ Methode), sondern zusätzlich auch der zweit nächste „Nachbar“ berücksichtigt wird. Dies geschieht in der Form, dass zuerst der nächste und zweit nächste „Nachbar“ für den eben untersuchten Schlüsselpunkt gesucht werden. Daraufhin werden die Distanzen zwischen den Punkten berechnet. Liegt der Wert des Quotienten aus diesen Distanzen unterhalb eines Thresholds, wird Punkt zwei als positives Match gewertet. Mit Hilfe dieser Untersuchung wird sichergestellt, dass der zweit nächste „Nachbar“ zum untersuchten Punkt weit genug entfernt ist und somit der nächste Nachbar ein eindeutiges Match-Ergebnis darstellt. Folgende Formel stellt diesen Zusammenhang dar, wobei D_A den untersuchten Schlüsselpunkt, D_B den nächsten „Nachbarn“ und D_C den zweit nächsten „Nachbarn“ darstellen.

$$\frac{|D_A - D_B|}{|D_A - D_C|} < R_{th} \quad (4.1)$$

Das Formelzeichen R_{th} steht für den Ratio-Threshold, welcher üblicherweise im Bereich von $R_{th} = 0,6 - 0,7$ liegt.

Auf diese Weise werden alle Eigenschaftsvektoren der Schlüsselpunkte des ersten Bildes mit den Eigenschaftsvektoren der Schlüsselpunkte aus dem zweiten Bild verglichen. Daraus ergibt sich gegebenenfalls eine Summe von positiven Matches, welche weiter ausgewertet werden können. Nähere Informationen zu den Matching Methoden bzw. Analysen bezüglich der Performance und Wiederholbarkeit können aus der Literatur wie [Mikolajczyk , Schmid, 2005] entnommen werden.

Beim SURF Algorithmus gibt es noch eine zusätzliche Möglichkeit zur Auswertung der Matches. Neben der Euklidischen Distanz zwischen den Eigenschaftsvektoren kann auch noch das Vorzeichen der Spur der Hesse-Matrix ausgewertet werden. Dieser Wert steht durch die Berechnung der Blob-Response Map ohnehin zur Verfügung und benötigt somit keine extra Rechenzeit. Die Spur der Hesse-Matrix weist auf den Kontrast der Umgebung im Vergleich zum Schlüsselpunkt hin. (d.h.: Ob der Punkt in einer helleren Umgebung oder einer dunkleren Umgebung detektiert wurde.) Folgende Abbildung 4.4 verdeutlicht diesen Zusammenhang:

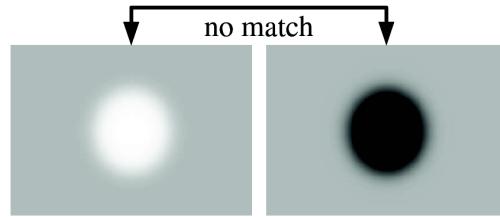


Abbildung 4.4: Kontrast von Schlüsselpunkten

Wie auch in [Bay, Ess, Tuytelaars, Van Gool, 2008] beschrieben, können durch die Verwendung der Spur der Hesse-Matrix die Schlüsselpunkte beim Matching-Prozess bereits vor den eigentlichen Berechnungen ausgeschlossen werden, wodurch sich die Rechenzeit beim Matching verringert und die Anzahl der positiven Matches erhöht wird.

4.3.2 Matching

Beim Matching werden korrespondierende Schlüsselpunkte in zwei oder mehreren Bildern gesucht. Folgende Abbildung zeigt zwei Infrarotbilder mit den positiv detektierten Matches, wobei das rechte Bild die Szene von einem anderen Kamera-standpunkt aus zeigt und zusätzlich noch um 10° gedreht ist:

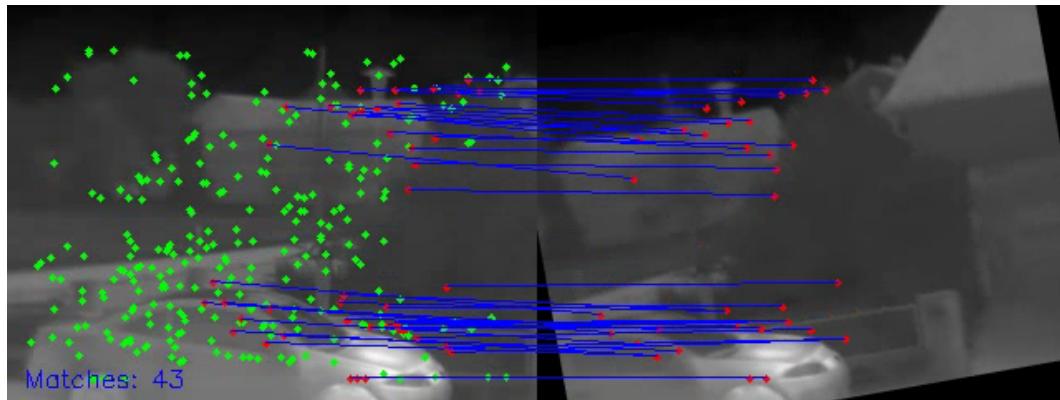


Abbildung 4.5: Matching Beispiel Infrarotbild (320×240 Pixel)

Wie aus Abbildung 4.5 ersichtlich, werden von den 309 detektierten Key-Points im linken Bild, 43 im rechten Bild richtig wiedergefunden. Wie schon in Punkt 4.2 beschrieben, wirken sich die Einstellungen des Detektors wiederum auch auf das Matchen von Key-Points aus. Im Allgemeinen werden nicht alle Schlüsselpunkte, die im Originalbild detektiert werden, auch in einer anderen Szene wieder gefunden. Aus diesem Grund werden in den folgenden Punkten einige Untersuchungsergebnisse zu verschiedenen Detektoreinstellungen sowie zur Wiederholbarkeit des Detektors bzw. des Matching-Prozesses näher betrachtet.

4.4 Verhalten von SURF

In diesem Abschnitt wird das Verhalten des Algorithmus bezüglich Bildrotation und Bildrauschen getestet. Dafür gibt es verschiedene Bewertungskriterien, welche in den folgenden beiden Punkten näher erläutert werden.

4.4.1 Verhalten des Detektors

Das meist verwendete Kriterium zur Bewertung des Detektors stellt die Wiederholbarkeit dar. Die Wiederholbarkeit gibt an, wie viele der detektierten Schlüsselpunkte eines Ausgangsbildes in einem veränderten Bild korrekt wieder detektiert werden, relativ zur (kleineren) Anzahl der detektierten Punkte in den beiden Bildern. Für die Anzahl der Punkte werden nur jene verwendet, die auch szenenbedingt in beiden Bildern detektiert werden können.[Schmid, Mohr , Bauckhage, 2000] Für die Auswer-

tung über die Richtigkeit der wieder detektierten Punkte gibt es mehrere Methoden. Die wohl einfachste Methode ist die Punkte visuell auf Korrespondenz zwischen den beiden Bildern zu prüfen. Da dies aber eher langwierig und fehleranfällig ist, werden die Auswertungen für diese Arbeit über die so genannte Homographie-Matrix durchgeführt. Die Homographie-Matrix stellt eine Matrix dar, mit deren Hilfe die Schlüsselpunkte vom Ausgangsbild in das geänderte Bild transformiert werden können. Für ein um den Winkel α rotiertes Bild kann die Homographie-Matrix zur Umrechnung der Ausgangspunkte in das rotierte Bild als zweidimensionale Drehmatrix dargestellt werden. Sei \vec{P}_1 der Schlüsselpunkt im Ausgangsbild und \vec{P}_2 der Schlüsselpunkt im rotierten Bild, so gilt für die Transformation des Ausgangspunktes in das rotierte Bild folgender Zusammenhang:

$$\vec{P}_2 = H \vec{P}_1 \quad (4.2)$$

wobei H die Homographie-Matrix darstellt, welche für ein rotiertes Bild wie folgt aufgestellt werden kann:

$$H = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \quad (4.3)$$

Da die Schlüsselpunkte meist nicht exakt an der richtigen Position wieder detektiert werden, muss die Überprüfung auf Korrespondenz mittels der Distanz zwischen den transformierten Ausgangspunkten und den neu detektierten Punkten durchgeführt werden. Liegt diese Distanz unter einer gewissen Maximaldistanz ϵ , wurde dieser Punkt korrekt wieder detektiert und der Punkt kann als korrespondierender Punkt gewertet werden. Die Bestimmung des Abstandes zwischen den Punkten erfolgt über die Euklidische Distanz d_E ($d_E = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$). In den folgenden Abbildungen ist die Wiederholbarkeit auch unter Berücksichtigung der Skale dargestellt, wobei dafür die Skale als dritte Vektorkomponente des Schlüsselpunktes verwendet wird.

Mit Hilfe der Anzahl der berechneten Korrespondenzen $R_i(\epsilon)$ kann die Wiederholbarkeit r_i des Detektors in Abhängigkeit von ϵ durch folgende Formel berechnet werden:

$$r_i(\epsilon) = \frac{|R_i(\epsilon)|}{\min(n_1, n_2)} \quad (4.4)$$

Die Formelzeichen n_1 und n_2 , stellen die detektierten Schlüsselpunkte in den Bildern dar.

4.4.1.1 Rotation

Folgende Abbildung zeigt das Verhalten des Detektors bei der Rotation des Bildes von $0^\circ - 180^\circ$ für $\epsilon = 1,5$:

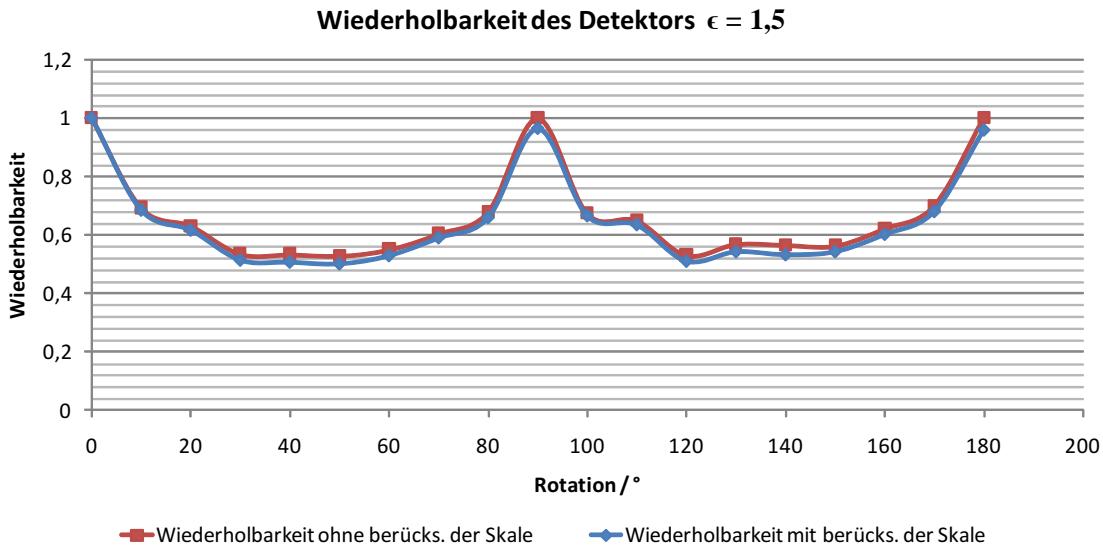


Abbildung 4.6: Wiederholbarkeit bei Rotation, $\epsilon = 1,5$

Wie obiges Diagramm zeigt, sinkt die Wiederholbarkeit bis zu einem Winkel von $\alpha = 45^\circ$ und beginnt dann wieder zu steigen. Dieses Verhalten ist auf die Box-Filter zurückzuführen, da zum Beispiel die Filtermatrix D_{xx} (vergleiche Abbildung 3.7) bei einer Drehung des Bildes von $\alpha = 90^\circ$, $\alpha = 180^\circ$ und $\alpha = 270^\circ$ genau der Filtermatrix D_{yy} entspricht und umgekehrt. Aus diesem Grund werden viele Schlüsselpunkte an der selben Position wie im Ausgangsbild detektiert.

4.4.1.2 Rauschen

In diesem Punkt wird die Wiederholbarkeit des Detektors für unterschiedlich verrauschte Bilder getestet. Dazu wird zum Ausgangsbild ein künstlich generiertes Rauschen aus Zufallszahlen addiert. Die Wiederholbarkeit wird für verschiedene Amplituden des Rauschsignals bestimmt. Folgende Abbildung zeigt einen Vergleich des Originalbildes mit dem künstlich verrauschten Bild:



Abbildung 4.7: Vergleich Originalbild (links) und verrauschtes Bild (rechts), Amplitude des Rauschsignals $A = \pm 0,071$

Die Bestimmung der korrespondierenden Punkte kann bei diesem Test ohne Homographie-Matrix erfolgen, da sich die Position der Punkte aus dem Ausgangsbild nicht verändern sollte. Folgende Abbildung zeigt die Wiederholbarkeit des Detektors für eine Maximaldistanz von $\epsilon = 1,5$:

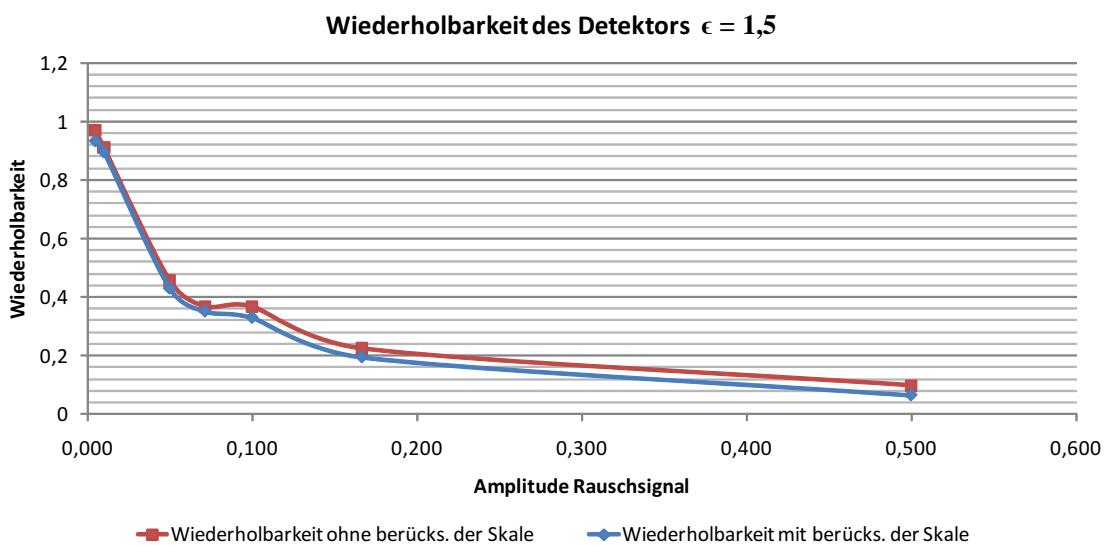


Abbildung 4.8: Wiederholbarkeit bei Rauschen, $\epsilon = 1,5$

Wie sich aus Abbildung 4.8 zeigt, fällt die Wiederholbarkeit mit zunehmender Rauschsignalamplitude sehr schnell ab. Dieser Verlauf entspricht dem erwarteten Verhalten,

da sich das Bild durch das zunehmende Rauschen stark verändert und somit auch die Richtungsableitungen. Dadurch wird eine genaue Detektion erschwert.

Für alle präsentierten Testergebnisse wurden folgende Einstellungen des Algorithmus verwendet:

- $okt = 4$
- $f_i = 4$
- $s_r = 2$
- $t_h = 0,00001$

4.4.2 Verhalten des Descriptors

Das Verhalten des Descriptors bzw. des Algorithmus bezüglich des Matchings wird meist durch die beiden Werte „recall“ und „1-precision“ bewertet. Der Wert „recall“ beschreibt die Anzahl der positiven Matches bezogen auf die Anzahl der korrespondierenden Schlüsselpunkte die in beiden Bilder detektiert werden.[Mikolajczyk , Schmid, 2005] Folgende Formel stellt diesen Zusammenhang dar:

$$recall = \frac{\text{positive Matches}}{\text{korrespondierende Schlüsselpunkte}} \quad (4.5)$$

Die korrespondierenden Schlüsselpunkte werden mit Hilfe der in Punkt 4.4.1 beschriebenen Vorgehensweise berechnet.

Der Wert „1-precision“ wird durch die Anzahl der falschen Matches bezüglich der gesamten Matches berechnet, wie folgende Formel darstellt:

$$1 - precision = \frac{\text{falsche Matches}}{\text{gesamte Matches}} \quad (4.6)$$

Für das Matchen der Schlüsselpunkte wird die in Punkt 4.3.1 erläuterte Methode verwendet mit der Erweiterung, dass das Vorzeichen der Spur der Hesse-Matrix ausgewertet wird. In den folgenden Punkten wird das Verhalten für verschiedene Einstellungen des Algorithmus untersucht. Für die Erstellung einer „recall“ über „1-precision“ Kurve wird der Ratio-Threshold R_{th} der „Nearest Neighbour Ratio“ Matching Methode (welcher das Verhältnis zwischen dem ersten und dem zweiten Nachbar des positiven Matches darstellt) zwischen $0,15 < R_{th} < 0,95$ variiert.

4.4.2.1 Rotation

Die Untersuchungen werden für ein um den Winkel $\alpha = 50^\circ$ rotiertes Bild berechnet. Folgende Abbildung zeigt einen Kurvenverlauf einer „recall“ vs. „1-precision“ Kurve für ein Realbild mit einer Auflösung von 640×480 Pixeln und ein Infrarotbild mit der Auflösung von 320×240 Pixel:

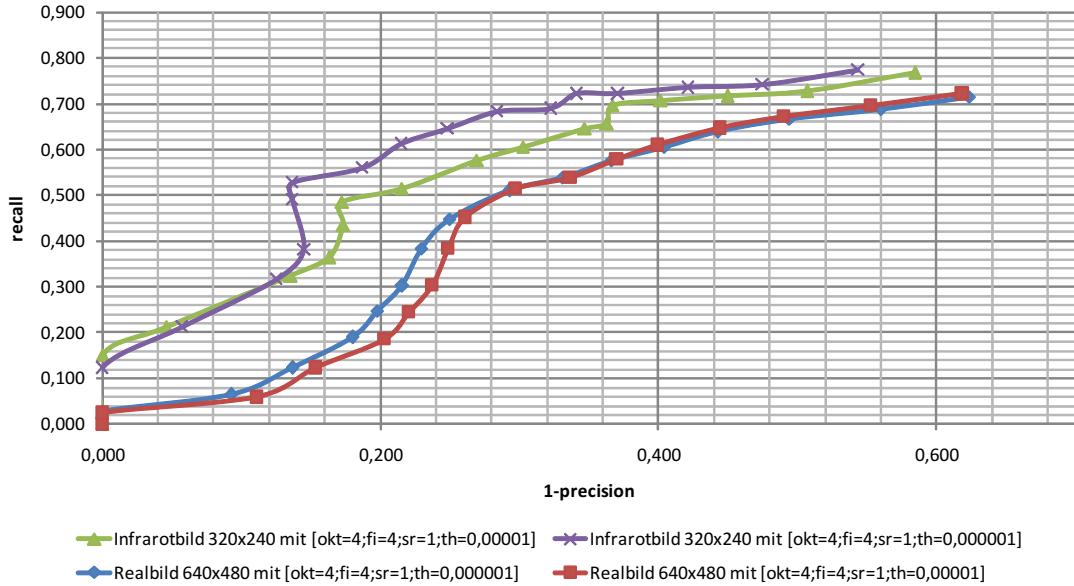


Abbildung 4.9: „recall“ vs. „precision“ Verlauf für ein Realbild mit einer Auflösung von 640×480 Pixel und ein Infrarotbild mit einer Auflösung von 320×240 Pixel bei einer Rotation von $\alpha = 50^\circ$

Aus Abbildung 4.9 zeigt sich ein erwarteter Verlauf. Für einen kleinen Ratio-Threshold ergibt sich die maximale Genauigkeit, $1 - precision = 0$. Dabei weist der Wert „recall“ (sprich die Anzahl der positiven Matches zur Anzahl der möglichen Matches) einen sehr geringen Wert auf, da keine bis sehr wenige Schlüsselpunkte als Match erkannt werden. Steigt der Ratio-Threshold R_{th} an, steigt auch der „recall“, da mehr Punkte detektiert werden, wobei aber auch die Zahl an falschen Matches ansteigt. Dadurch ergibt sich ein absinken der Genauigkeit, $1 - precision \rightarrow 1$.

Folgende Abbildung zeigt die verwendeten Testbilder, die für die Bestimmung der „recall“ vs. „1-precision“ Kurven verwendet wurden:



Abbildung 4.10: Testbilder für das „recall“ vs. „1-precision“-Diagramm, Infrarotbilder mit einer Auflösung von 320×240 Pixel (links) und Realbilder mit einer Auflösung von 640×480 Pixel (rechts)

4.5 Performance von SURF

In diesem Punkt werden einige Untersuchungen bezüglich der Performance des SURF Algorithmus näher betrachtet, wobei für die Untersuchungen der Programmcode beschrieben in Punkt 3.3.2 verwendet wird. Die Tests werden mit drei verschiedenen Bildern (zwei Realbildern und einem Infrarotbild) durchgeführt, um einen Vergleich der Ergebnisse zu ermöglichen. Für jedes Bild werden die Schlüsselpunkte berechnet, welches den ersten Teil der Analyse darstellt. Die Untersuchungen werden für verschiedene Detektoreinstellungen durchgeführt um deren Auswirkungen auf die Berechnungszeit der Schlüsselpunkte zu zeigen.

Die Matching-Performance wird auf folgend Weise getestet: Die im ersten Schritt berechneten Schlüsselpunkte werden im Matching-Algorithmus mit sich selbst verglichen. Somit ergibt sich eine Anzahl an möglichen Matches von 100%. Durch dieses Vorgehen ergibt sich eine Anzahl an positiven Matches von ebenfalls 100%, wodurch sich auch die Rechenzeit für den Matching-Vorgang maximiert. Folgende Abbildung zeigt eine Testberechnung bei der zwei völlig identische Bilder verglichen werden:

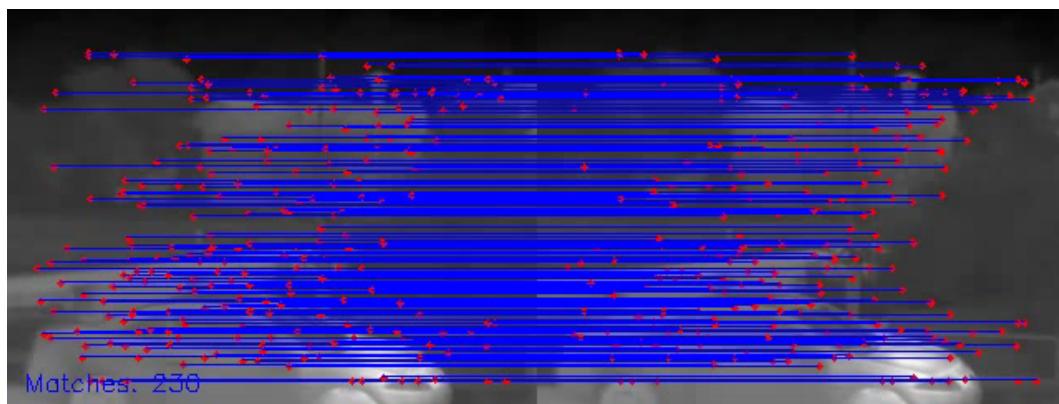


Abbildung 4.11: Testbild mit 100% positiven Matches

Für die Untersuchungen wird der Testrechner mit den in Tabelle 4.1 dargestellten Kenndaten verwendet. Die in den folgenden Punkten angegebenen Zeiten für die Berechnungsdauer entsprechen der reinen CPU Zeit für den getesteten Programmschnitt. Dadurch können sich in Verbindung mit dem Code zur Darstellung der Bilder, sowie sonstigen Auswertungen, andere Rechenzeiten ergeben.

4.5.1 Performance Key-Point Berechnung

Abbildung 4.12 zeigt ein Diagramm zur Gegenüberstellung der detektierten Key-Points für verschiedene Einstellungen¹ des Algorithmus.

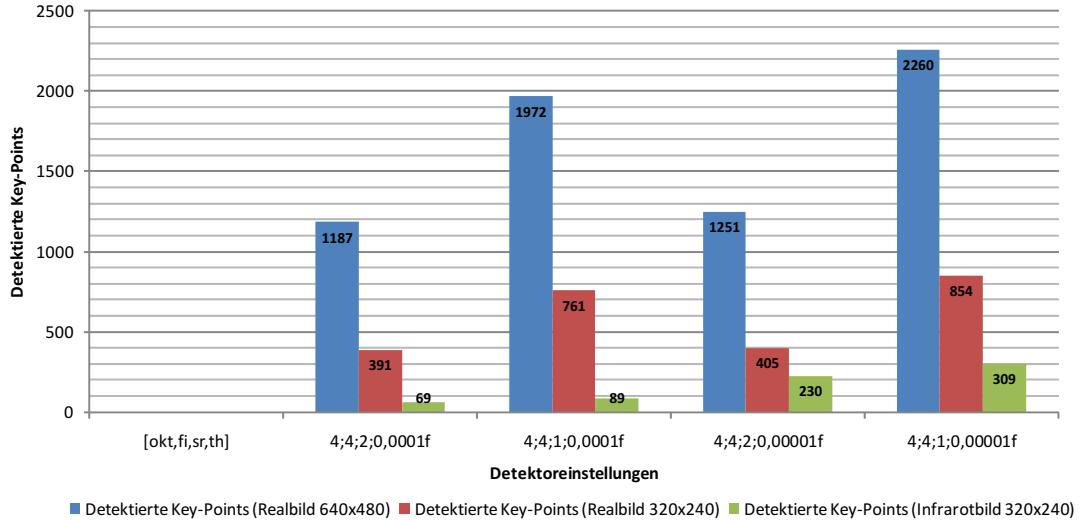


Abbildung 4.12: Gegenüberstellung der detektierten Key-Points in den Testbildern

Wie aus Abbildung 4.12 ersichtlich, werden mit sinkendem Threshold t_h sowie mit der Änderung der Anfangs-Abtastrate auf $s_r = 1$ mehr Key-Points detektiert. Die für die Berechnung der Schlüsselpunkte benötigten Zeiten sind in folgender Abbildung dargestellt:

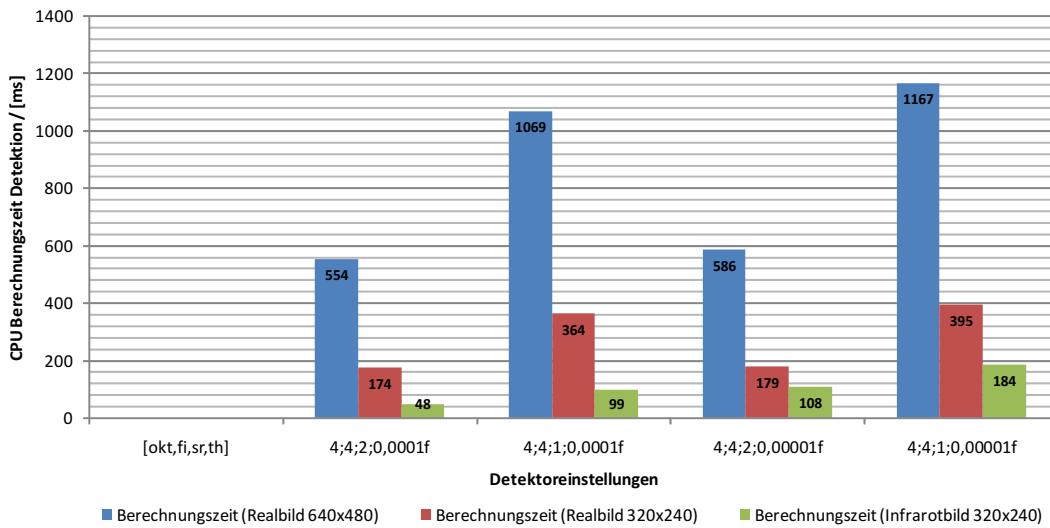


Abbildung 4.13: Gegenüberstellung der Berechnungszeiten

Wie sich aus Abbildung 4.13 deutlich zeigt, korreliert die Zeit zur Berechnung mit der Anzahl der detektierten Schlüsselpunkte. Des Weiteren ergeben die Messungen,

¹ $[okt, f_i, s_r, t_h]$ Vgl. Punkt 4.2

dass sich die Änderung der Anfangs-Abtastrate stärker auswirkt als eine Verringerung des Thresholds, was sich auf die Menge der detektierten Schlüsselpunkte zurückführen lässt. Wie schon in Punkt 4.2.1 erwähnt, werden im Infrarotbild mit gleichen Detektoreinstellungen weniger Schlüsselpunkte detektiert, was eine Anpassung des Algorithmus an Infrarotbilder erforderlich macht.

4.5.2 Performance Matching

Bei der Untersuchung des Algorithmus bezüglich der Rechen-Performance beim Matchen von Key-Points wird ein Bild mit sich selbst verglichen. Somit entspricht die Anzahl der positiven Matches den detektierten Punkten im Bild, die aus Abbildung 4.12 ersichtlich sind. Um den Performance-Gewinn mit der Berücksichtigung des Vorzeichens der Spur der Hesse-Matrix (vergleiche Punkt 4.3.1) als zusätzliches Kriterium für ein positives Match zu zeigen, sind in folgender Abbildung 4.14 zunächst die benötigten Zeiten für das Matchen der Schlüsselpunkte ohne dieser zusätzlichen Auswertung dargestellt:

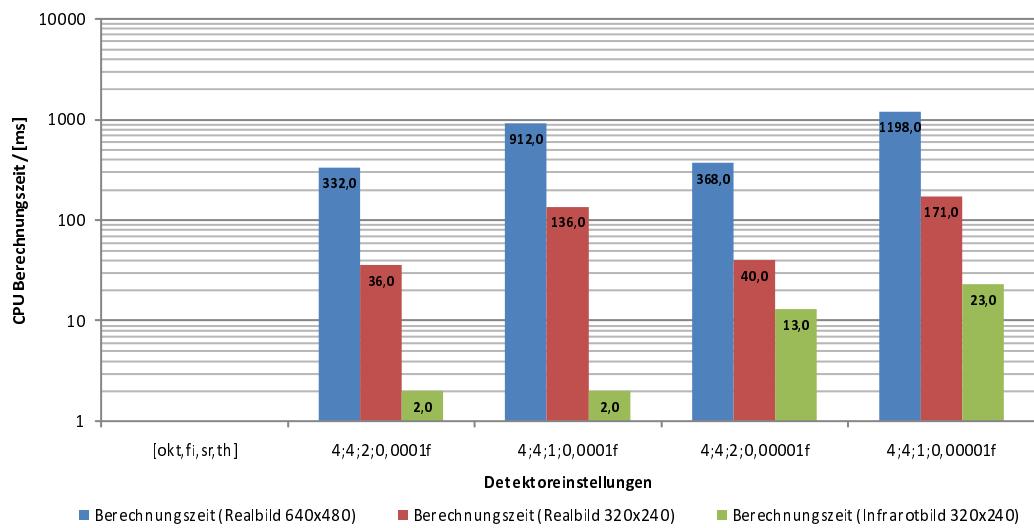


Abbildung 4.14: CPU Berechnungszeiten für das Matching ohne die Auswertung des Vorzeichens der Hesse-Matrix

In nachfolgender Abbildung sind nun die Zeiten für die Berechnungen für das Matchen mit dem oben genannten Kriterium dargestellt:

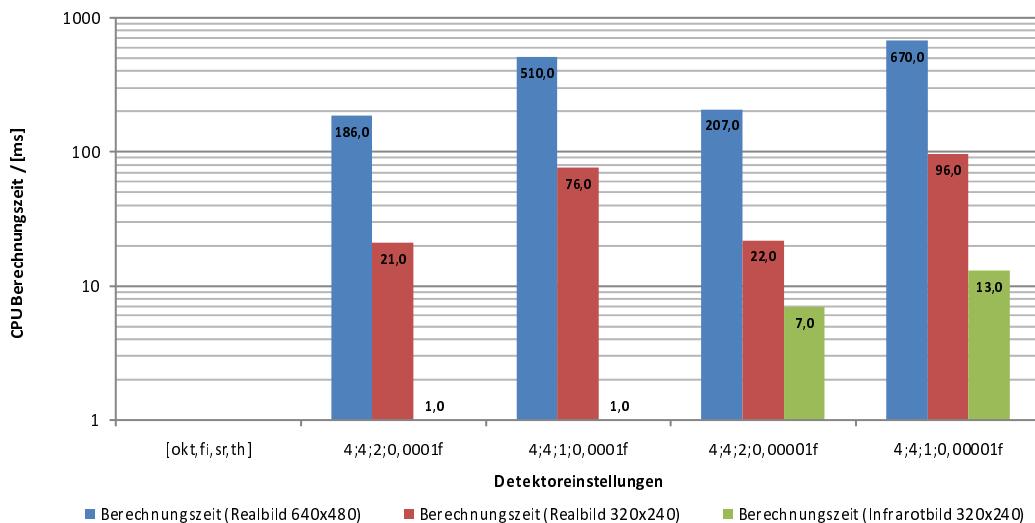


Abbildung 4.15: CPU Berechnungszeiten für das Matching inklusive der Auswertung des Vorzeichens der Hesse-Matrix

Wie aus dem Vergleich der beiden Diagramme ersichtlich ist, verbessert sich die Performance des Matching-Vorgangs durch dieses zusätzliche Kriterium wesentlich. Diese Zeitsparnis lässt sich wie folgt erklären: Jene Schlüsselpunkte die nicht das selbe Vorzeichen für die Spur der Hesse-Matrix bzw. den Laplace-Wert aufweisen, können im Vorhinein aussortiert werden. Folglich muss für diese Punkte keine Euklidische Distanz zwischen den Eigenschaftsvektoren berechnet werden. Wie aus [Bay, Ess, Tuytelaars, Van Gool, 2008] hervorgeht liegt die maximale Zeitsparnis für die Berechnung der Matches bei einem Faktor von 2, wenn das Vorzeichen des Laplace-Wertes berücksichtigt wird. Durch die beschriebenen Untersuchungen bestätigt sich dieser Faktor für die Zeitsparnis, wie folgende Abbildung zeigt:

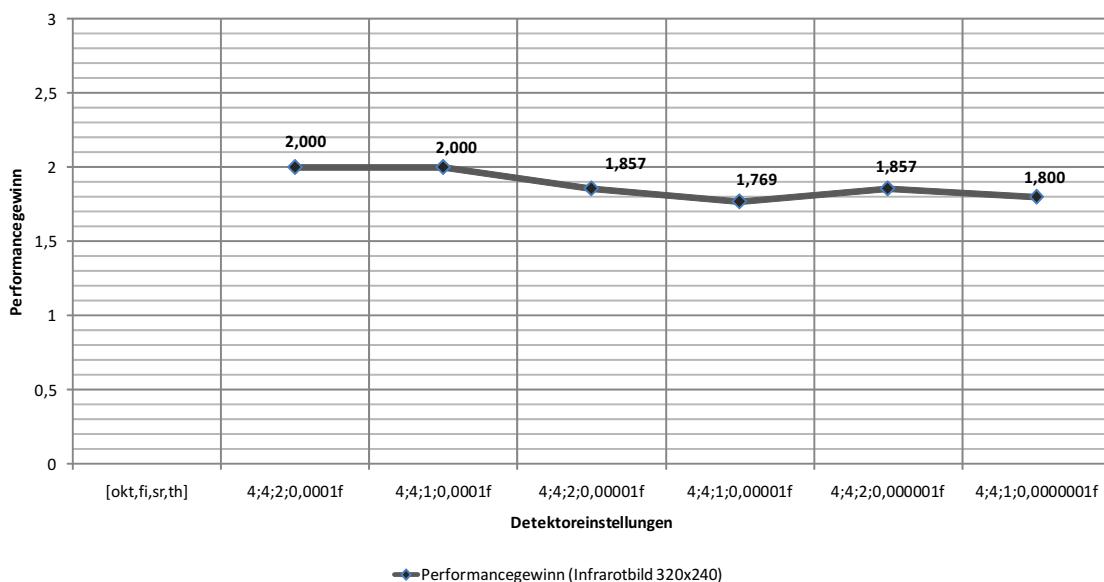


Abbildung 4.16: Faktor der Zeitsparnis für verschiedene Einstellungen des Algorithmus bei Verwendung des Vorzeichens der Hesse-Matrix

5 Zusammenfassung

In dieser Arbeit wurde die Anwendbarkeit eines Key-Point basierten Algorithmus auf Infrarotbilder untersucht. Dazu wurde im ersten Teil der Arbeit der Speeded Up Robust Features Algorithmus näher vorgestellt. Der so genannte SURF Algorithmus nutzt die Integral Image Repräsentation und eine effiziente Approximation der LoG-Funktion durch Box-Filter, was eine sehr effiziente Implementierung erlaubt. Für die Suche nach Features nutzt der SURF Algorithmus einen Hesse-Matrix basierten Feature-Detektor, den so genannten Fast-Hessian Detektor. Jeder Schlüsselpunkt erhält einen Feature-Vektor mit einer Länge von 64 Einträgen, welcher sich aus der Gradienten-Information der Umgebung des Schlüsselpunktes zusammensetzt.

Der zweite Teil dieser Arbeit behandelte die Anwendung und Untersuchung des SURF Algorithmus. Dabei wurde die Berechnung von Schlüsselpunkten und das Matchen näher betrachtet, wobei im Speziellen Infrarotbilder untersucht wurden. Der letzte Abschnitt dieser Arbeit beschäftigte sich mit der Performance des SURF Algorithmus.

Aus den Untersuchungen hat sich gezeigt, dass Infrarotbilder sehr wenig Details und feine Strukturen aufweisen, wodurch eine Anwendung von Key-Point basierten Algorithmen erschwert wird. Wie aber gezeigt werden konnte, ist es durchaus möglich, Schlüsselpunkte in einem Infrarotbild zu detektieren. Des weiteren geht aus den Versuchen hervor, dass in einem Infrarotbild im Allgemeinen weniger Schlüsselpunkte detektiert werden, was eine Anpassung der Einstellungen des Algorithmus erfordert. Die Untersuchung bezüglich Bildrotation hat gezeigt, dass der Algorithmus selbst für Infrarotbilder ein sehr gutes Verhalten aufweist (Wiederholbarkeit $r_i(\epsilon) > 0,5$). Der SURF Algorithmus erlaubt eine sehr effiziente Berechnung der Schlüsselpunkte, ohne der Notwendigkeit zur Verwendung von spezieller Hardware (z.B.: GPU-Implementierung). Aus den Versuchen geht hervor, dass gerade bei Infrarotbildern durch die geringe Berechnungszeit ($T < 200ms$) sowie die geringe Matching-Zeit ($T < 30ms$) eine Online-Anwendung des Algorithmus möglich ist.

Die Anwendbarkeit des SURF Algorithmus auf Infrarotbilder konnte erfolgreich gezeigt werden. Für die Anwendung eines Algorithmus wie SURF auf Infrarotbilder in einer konkreten Aufgabenstellung, müssen sicherlich noch einige Untersuchungen

angestellt werden, um bestmögliche Ergebnisse zu erzielen. Dabei ist es vor allem notwendig, die Einstellungen des Detektors für die konkrete Anwendung näher zu betrachten und anzupassen.

Literaturverzeichnis

- [Burger, Burge, 2005/2006] Burger W., Burge M.J., Digitale Bildverarbeitung, Springer Verlag, Berlin Heidelberg, 2005/2006
- [Bradski, Kaehler, 2008] Bradski G., Kaehler A., Learning OpenCV, O'Reilly Media Inc., Sebastopol CA., 2008
- [Derpanis, 2007] Derpanis K. G., Integral image-based representations, Department of Computer Science and Engineering, York University, Paper, 2007
- [Bay, 2006] Bay H., From Wide-baseline Point and Line Correspondences to 3D, ETH Zurich, Dissertation, 2006
- [Bay, Tuytelaars, Van Gool, 2008] Bay H., Tuytelaars T., Van Gool L., SURF: Speeded Up Robust Features, ETH Zurich, Katholieke Universiteit Leuven, Paper, 2008
- [Bay, Ess, Tuytelaars, Van Gool, 2008] Bay H., Ess A., Tuytelaars T., Van Gool L., Speeded Up Robust Features (SURF), ETH Zurich, Katholieke Universiteit Leuven, Paper, 2008
- [Evans, 2009] Evans C., Notes on the OpenSURF Library, University of Bristol, Paper, 2009
- [Viola, Jones, 2001] Viola P., Jones M., Rapid Object Detection using a Boosted Cascade of Simple Features, Mitsubishi Electric Research Labs und Compaq CRL, Paper, 2001
- [Lindenberg, 1998] Lindenberg T., Feature Detection with Automatic Scale Selection, CVAP KT (Royal Institute of Technology) Schweden, Technical Report, 1998
- [Lindenberg, 1991] Lindenberg T., Discrete Scale-Space Theory and the Scale-Space Primal Sketch, CVAP KT (Royal Institute of Technology) Schweden, 1991
- [Lowe, 2004] Lowe D., Distinctive Image Features from Scale-Invariant Keypoints, Computer Science Department UBC Canada, 2004

- [Mikolajczyk, Tuytelaars, 2006] Mikolajczyk K., Tuytelaars T., Schmid C., Zisserman A., Matas J., Schaffalitzky F., Kadir T., Van Gool L., A Comparison of Affine Region Detectors, Springer Science + Business Media, 2006
- [Mikolajczyk , Schmid, 2005] Mikolajczyk K., Schmid C., A performance evaluation of local descriptors, Dept. of Engineering Science, University of Oxford, INRIA, Paper, 2005
- [Schmid, Mohr , Bauckhage, 2000] Schmid C., Mohr R., Bauckhage C., Evaluation of Interest Point Detectors, INRIA Rhone-Alpes, Paper, 2000
- [Richter, 2001] Richter M., Grundwissen - Mathematik für Ingenieure, Vieweg + Teubner Verlag, 2001
- [Weltner, 2006] Weltner K., Wiesner H., Heinrich P.-B., Engelhardt P., Schmidt H., Mathematik fur Physiker 2 Ausgabe 13, Springer Verlag, 2006
- [Herrmann, 2007] Herrmann N., Höhere Mathematik: Für Ingenieure, Physiker und Mathematiker, Oldenbourg Wissenschaftsverlag, 2007
- [Forster, 2006] Forster O., Analysis Ausgabe 7, Vieweg+Teubner Verlag, 2006
- [Jähne, 2005] Jähne B., Digitale Bildverarbeitung 6 Ausgabe, Springer Verlag, 2005
- [Hattheier, 2009] Hattheier T., Einsatz der Thermographie zur Opferlokalisierung für einen RoboCup Rescue Roboter, Fachhochschule Wels, Bachelor Arbeit, 2009
- [Homepage, SURF] <http://www.vision.ee.ethz.ch/~surf/index.html>, Computer Vision Laboratory, ETH Zurich, Stand 08.2009
- [Homepage, GPU-SURF] <http://homes.esat.kuleuven.be/~ncorneli/gpusurf/>, Stand 08.2009
- [Homepage, Matlab SURF] <http://www.maths.lth.se/matematiklth/personal.../petter/surfmex.php>, Lund Universität, Stand 08.2009
- [Homepage, OpenSURF] <http://code.google.com/p/opensurf1/>, Evans C., Stand 08.2009
- [Homepage, NIST] <http://www.isd.mel.nist.gov/projects/USAR/2009/index.htm>, National Institute of Technology (NIST) - RoboCup Rescue Arena 2009, Stand 02.2009