

Hochschule Heilbronn

Robotik und Automation



Analyse verschiedener Merkmalsextraktionsverfahren zur Objekt-Klassifizierung

Name	Elias Marks
Matrikelnummer	184949
Anschrift	Max-Planck-Str. 27 74081 Heilbronn

Betreuer	Prof. Dr. Dieter Maier
Email	dieter.maier@hs-heilbronn.de
Telefon	+49 7131 504 399
Anschrift	Hochschule Heilbronn Max-Planck-Str. 39 74081 Heilbronn

Heilbronn, 14.03.2016

..

Elias Marks

Inhaltsverzeichnis

1 Einleitung	1
1.1 Motivation	1
1.2 Untersuchte Algorithmen	2
2 Inhalt	3
2.1 SIFT	3
2.1.1 Merkmalsextraktion	4
2.1.2 Deskriptorberechnung	7
2.2 SURF	8
2.2.1 Merkmalsextraktion	8
2.2.2 Deskriptorberechnung	9
2.3 ORB	11
2.3.1 Merkmalsextraktion	11
2.3.2 Deskriptorberechnung	13
2.4 Bruteforce Matcher	16
2.5 Homographie	17
2.6 RANSAC	18
3 Merkmalsextraktion	19
3.1 Verwendete Software	19
3.2 Hardware	19
3.3 Erstellte Software	20

3.3.1	GPU Beschleunigung	22
3.3.2	Benchmark	23
4	Ergebnisse	25
4.1	Laufzeit	25
4.2	Anzahl der korrekten Matches	26
4.3	Stabilität gegen Rotation	27
4.4	Stabilität gegen Skalierung	27
4.5	Stabilität der Lokalisierung	28
4.6	Testvideos	30
5	Ausblick	33
6	Zusammenfassung	35
	Glossar	39
	Literaturverzeichnis	39

Kapitel 1

Einleitung

1.1 Motivation

Die Bilddatenverarbeitung gewinnt in unserem technologischem Zeitalter immer mehr an Bedeutung. Durch die Verbreitung der Automatisierung in den meisten industriellen und sozialen Umfeldern werden immer neue und effizientere Methoden zur Datenverarbeitung benötigt. Eine der größten Herausforderungen bei Autonomen Systemen ist die Erfassung der Umgebung um Interaktionen mit dieser, oder kollisionsfreie Bewegungen zu ermöglichen. Zu diesem Zweck werden unter anderem Kameras verwendet. Diese haben den Vorteil dass sie für ein sehr breites Spektrum an Aufgaben einsetzbar sind da sie eine enorme Menge an Daten sammeln. Dies hat jedoch zur Folge dass sehr komplexe Algorithmen benötigt werden um die relevanten Informationen, wie die Art oder Position eines Objekts, zu extrahieren. Als die ersten Steine des heutigen Wissens in diesem Bereich gelegt wurden war die Leistungsfähigkeit der Computer das größte Problem. Da die Entwicklung schnellerer und günstigerer Recheneinheiten (Prozessoren und Grafikkarten) seit dem sehr große Schritte gemacht hat und weiter machen wird, sind der Verbreitung dieser Technologie immer weniger Grenzen gesetzt. Außerdem steigt auch die Verbreitung von Handys mit äußerst leistungsstarken Kameras immer mehr und ermöglicht somit neue Anwendungsbereiche dieser Technologie

im Alltag.

1.2 Untersuchte Algorithmen

Es gibt verschiedene Ansätze der Bildverarbeitung, auch abhängig von dem voraussichtlichen Anwendungsgebiet. Neben dem Morphologischen und dem neuesten Ansatz des Deep Learnings wird vor allem die Merkmalsbasierte Bildverarbeitung genutzt. Dieser besteht bestehet in der Bestimmung von Punkten im Bild die bei möglichst vielen Lichtverhältnissen und aus unterschiedlichen Blickwinkeln wiedererkannt werden können. Durch die Anordnung dieser lokalen Merkmale kann ein Objekt anhand einer Datenbank aus solchen Mustern wiedererkannt und lokalisiert werden. Zu dieser Sparte der Bildverarbeitung gehören unter anderen die in dieser Arbeit analysierten Algorithmen. Die Einführung des *Scale-invariant feature transform* (SIFT) war ein sehr großer Schritt in der Entwicklung dieses Bereichs und ermöglichte somit die Verbreitung der Bildverarbeitung in industriellen Umgebungen. Die anderen hier untersuchten Algorithmen, *SURF* und *ORB*, sind mehr oder weniger eine Ableitung von *SIFT* und sind vor allem ein versuch die Kosten der Berechnung zu mindern um diese Technologie echtzeitfähig zu machen und den Einsatz auf weniger performanten Geräten zu ermöglichen.

Kapitel 2

Inhalt

Im Laufe dieser Arbeit wurden verschiedene Algorithmen zur lokalen Merkmalsextraktion und Deskriptor Berechnung getestet und ausgewertet. Das Ziel ist hierbei Merkmale zu finden die so gut wie möglich folgenden Ansprüchen genügen und somit als stabil bezeichnet werden:

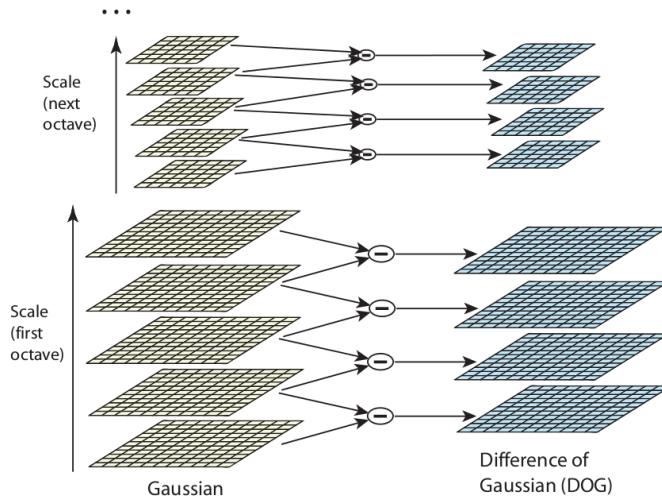
- *Skaleninvarianz*: Unabhängigkeit von der Größe auf dem Bild (Distanz zur Kamera)
- *Rotationsinvarianz*: Merkmale können trotz einer Rotation erkannt werden
- *Beleuchtungsinvarianz*: Ermöglicht Erkennung der Merkmale bei verschiedenen Lichtverhältnissen
- *Blickwinkelunabhängigkeit*: Unabhängigkeit von der Position aus der das Objekt aufgenommen wird

2.1 SIFT

Die SScale Invariant Feature Transform ist bis heute eine der am erfolgreichsten verwendeten Methoden zur lokalen Merkmalsextraktion. Sie wurde 1999 von David Lowe vorgestellt und ist von der Objekterkennung im menschlichen Gehirn inspiriert.

2.1.1 Merkmalsextraktion

Aus dem zu untersuchenden Bild wird als erstes der sogenannte Scale-space berechnet.



Hierzu wird das Bild mit einem Gauss-Kernel wiederholt geglättet um die Scales der ersten Octave zu erstellen. Um die Gaussgeglätteten Bilder $L(x, y, \sigma)$ zu erhalten wird der Gauss-Kernel

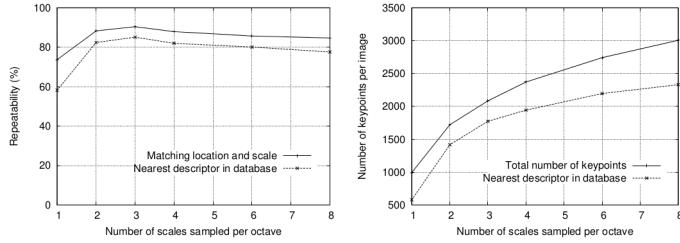
$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \quad (2.1)$$

mit dem Ursprungsbild $I(x, y)$ gefaltet:

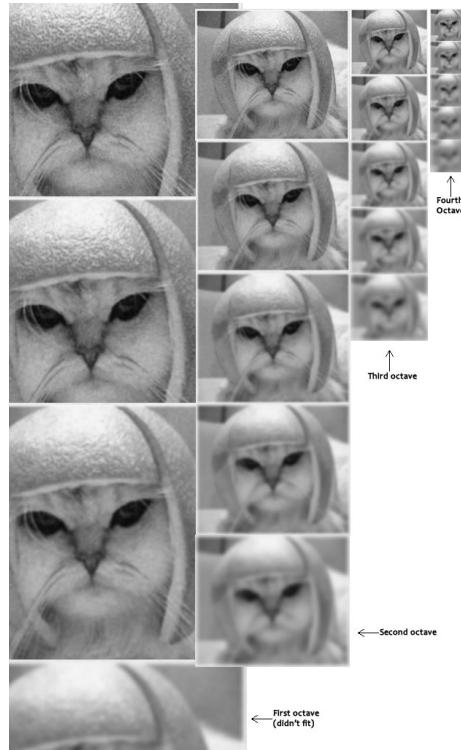
$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \quad (2.2)$$

Hiermit werden schrittweise immer mehr kleine Merkmale entfernt, was dazu führt dass größere Merkmale und Bedeutung gewinnen. Die Breiten der Gaussfunktionen mit der zwei benachbarte Scales berechnet werden, haben das konstante Verhältnis k , hierdurch sind die Scales im Scale-space equidistant in Bezug auf σ .

Lowe fand experimentell heraus, dass die optimale Anzahl an Scales per Octave, in Bezug auf die Wiederholgenauigkeit der Detektion der Merkmale nach verschiedenen Bildtransformationen, 3 beträgt. Nachdem die Scales der ersten Octave berechnet wurden, wird das Bild mit dem doppelten Initialwert von σ runterskaliert in dem jedes



zweite Pixel jeder Reihe und jeder Spalte beibehalten wird. Das erhaltene Bild ist der erste Scale der nächsten Octave aus dem die darauf folgenden wie zuvor erstellt werden. In der folgenden Abbildung ist das Ergebnis dieses Vorgangs veranschaulicht.



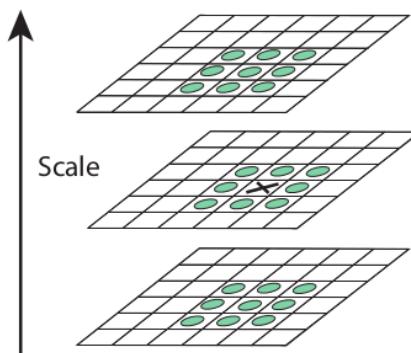
Um die Merkmale zu finden, werden die lokalen Extrema der *Difference of Gaussian* Funktion zwischen den benachbarten Scales berechnet. Mikolajczyk fand in experimentellen Vergleichen mit anderen Bildfunktionen, wie der Gradient, der Hesse- und der Harrisfunktion heraus, dass diese Methode die stabilsten Merkmale extrahiert.

Als erstes werden hierzu die *Difference of Gaussian* zwischen den verschiedenen Ebenen

des Scale-spaces ermittelt:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (2.3)$$

Um die Lokalen Maxima zu finden, wird jedes Pixel mit den 8 benachbarten im selben Scale und den 9 benachbarten in den darüber und darunter liegenden Scales verglichen. Stellt das Pixel ein Maximum oder ein Minimum in der betrachteten Menge von Pixeln dar, stellt es einen potentiellen *Keypoint* da.



Um nur die stabilsten *Keypoints* beizubehalten wurde von Brown vorgeschlagen mit Hilfe der quadratischen Taylorexpansion die Position der Keypoints auf Subpixel-Genauigkeit zu bestimmen. Außerdem werden auch Keypoints mit einem niedrigen Kontrast gefiltert.

Da die Suche nach Keypoints mit der *Difference of Gaussian* Funktion an den Kanten viele unsignifikante Punkte liefert werden anschließend die Gradienten der Punkte in x und y-Richtung und deren Verhältnis r berechnet. Ist dieses größer als ein Schwellwert (Lowe verwendet $r=10$) wird der Keypoint gelöscht.

Um die Keypoints Rotationsinvariant zu machen wird ihnen eine Richtung zugewiesen. Lowe hat herausgefunden dass sich hierzu die Orientierung der Bildpunkte um den Keypoint gut eignet. Er berechnet die Größe und die Orientierung der Gradienten der umliegenden Pixel im Scale des jeweiligen Keypoints:

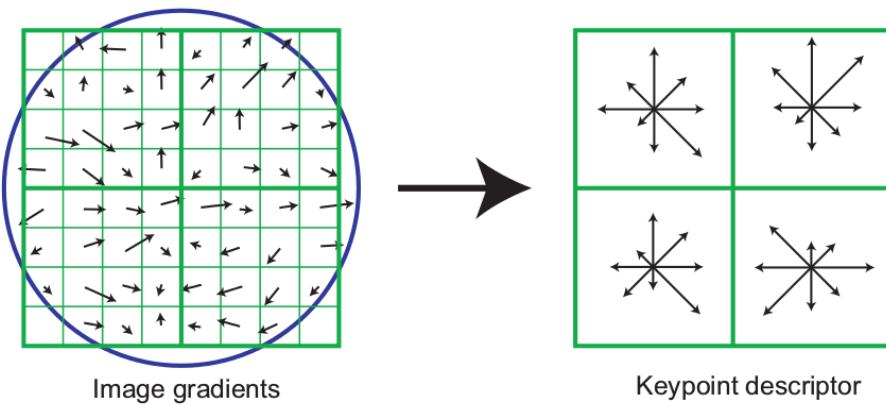
$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \quad (2.4)$$

$$\Theta(x, y) = \tan^{-1}((L(x, y + 1) - L(x, y - 1))/(L(x + 1, y) - L(x - 1, y))) \quad (2.5)$$

Die Orientierungen werden anschließend mit den Größen gewichtet und in 36 bins eines Histogramms unterteilt. Dem Keypoint wird die häufigste Orientierung des Histogramms zugewiesen und zusätzlich wird ein Keypoint mit jeder Orientierung die mindestens 80% der maximalen Häufigkeit besitzt erstellt.

2.1.2 Deskriptorberechnung

Um die Keypoints zu charakterisieren wird ein sogenannter *Deskriptor* berechnet. Hierzu werden die Gradientengrößen und Orientierungen eines 16x16 Pixel Feldes um den Keypoint herum bestimmt und mit der Gaußfunktion gewichtet um den Einfluss weiter entfernter Punkte zu verringern. Die Rotationsinvarianz des *Deskriptors* wird erreicht in dem die Orientierungen der Gradienten relativ zur Orientierung des Keypoints gedreht werden.



Die Pixel werden in 4x4 Pixel große Felder unterteilt und innerhalb dieser werden wie bei der Orientierungsbestimmung der Keypoints die gewichteten Werte der Gradienten in Histogramme akkumuliert. Diese besitzen 8 bins die den Orientierungen entsprechen. Zusätzlich werden die Gradienten trilinear interpoliert indem sie nach einer Gewichtung in Abhängigkeit der Distanz auch auf die benachbarten Histogramme

aufsummiert werden. Dadurch entsteht bei der Wiedererkennung der Merkmale ein bestimmtes Maß an Toleranz, dass die Erkennung von Objekten aus einem anderen Sichtwinkel oder nach einer nicht-starren Deformation ermöglicht.

Die *Deskriptoren* werden als Vektoren gespeichert, die alle $4 \times 4 \times 8 = 128$ Werte der bins der Histogramme enthalten. Die Merkmale sind gegen Helligkeitsveränderungen die gleichermaßen alle Pixel aus denen sie berechnet wurden betreffen stabil, da die Gradienten durch Differenzen gebildet werden. Um Stabilität gegen Kontrastveränderungen zu erreichen wird der Merkmalsvektor normiert. Um den Einfluss von nicht linearen Helligkeitsveränderungen zu verringern, werden alle Werte des Vektors die Größer als 0.2 sind auf diesen Wert gesetzt. Anschließend wird der Vektor nochmals normiert. [1]

2.2 SURF

Nachdem es mit SIFT und anderen Algorithmen möglich war sehr charakterisierende und somit mit sehr hoher Wiederholgenauigkeit erkennbare Merkmale aus Bildern zu extrahieren, kam das Bedürfnis nach einer Lösung die weniger rechenintensiv ist auf um Merkmalserkennung auch in Online-Systemen anzuwenden. Mit diesem Ziel entwickelten H. Bay, T. Tuytelaars und L. Van Gool die Speeded Up Robust Features; kurz SURF, und veröffentlichten diese im Jahr 2006.

2.2.1 Merkmalsextraktion

Der Detektor von SURF-Merkmalen basiert auf der Hesse-Matrix, diese enthält die zweiten Ableitungen der 2D-Gaussfunktion:

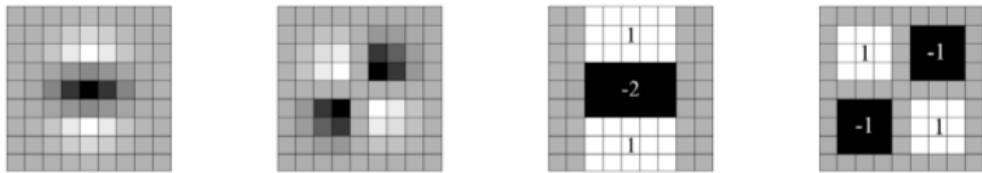
$$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{xy}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix} \quad (2.6)$$

Es wird die Determinante dieser benutzt um die Position und den Scale der Keypoints zu detektieren. Diese wird anschließend noch durch die Varianz der Gaussfunktion

geteilt um das Ergebnis zu normalisieren:

$$DoH(x, y, \sigma) = \frac{G_{xx}(x, y, \sigma) \cdot G_{yy}(x, y, \sigma) - G_{xy}(x, y, \sigma)^2}{\sigma^2} \quad (2.7)$$

Statt wie bei den SIFT-Merkmalen den Laplacian of Gaussian durch die Difference of Gaussian zu approximieren, verwenden die Autoren des SURF Verfahrens den Rechteck Filter.



Dieser ist zwar nicht so genau, aber dafür durch die Benutzung von Integralbildern viel schneller zu berechnen. Jede Position $X = (x, y)$ des Integralbildes beinhaltet die Summe aller Pixel aus dem Originalbild die innerhalb des durch die Position X und den Ursprung aufgespannten Rechtecks liegen:

$$I_\Sigma(X) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(x, y) \quad (2.8)$$

Ein weiterer Geschwindigkeitsvorteil wird dadurch erzielt dass der Scale-Space nicht im Voraus berechnet werden muss. Durch die Integralbilder sind die Kosten der Berechnung der Rechteckfilter nicht von der Größe der Maske abhängig und somit können die verschiedenen Scales durch die Anpassung der Maskengröße simuliert werden und das Originalbild muss für die verschiedenen Octaves nicht mehr verkleinert werden.

2.2.2 Deskriptorberechnung

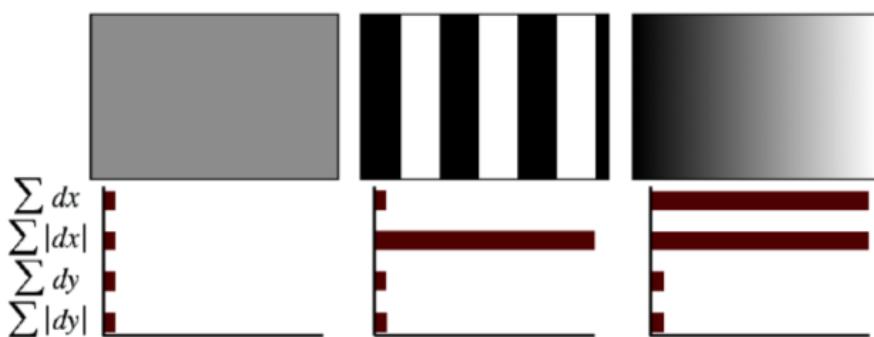
Der SURF Deskriptor ist von der Grundidee dem SIFT-Deskriptor sehr ähnlich. Auch hier wird als erstes die Orientierung extrahiert und dann das Merkmal durch sein Umfeld charakterisiert. Hierzu verwendet SURF allerdings Haar Wavelets da diese sehr schnell berechenbar sind.

Um die Orientierung zu bestimmen werden ein horizontales und ein vertikales Haar Wavelet in einem Umkreis von $6s$ um das Merkmal in dem entsprechenden Scale s des Merkmals berechnet. Auch hier wird auf Integralbilder zurückgegriffen da die Wavelets eine Seitenlänge von $4s$ haben und somit für größere Scales sehr groß werden. Der Umkreis des Merkmals wird in Abschnitte mit einer Breite von $\frac{\pi}{3}$ unterteilt und in jedem werden die Ergebnisse des Horizontalen Wavelets und anschließend die des Vertikalen aufsummiert. Somit erhält jeder Abschnitt ein Orientierungsvektor und der mit dem größten Betrag wird als Orientierung des Merkmals verwendet.

Um den Deskriptor zu extrahieren wird ein Quadrat mit $20s$ Seitenlänge, das nach der vorher bestimmten Orientierung ausgerichtet ist, um das Merkmal gespannt. Dieses wird in 4 gleichgroße Unterbereiche unterteilt und in jedem dieser werden die Horizontalen und Vertikalen (in Bezug auf die Orientierung des Merkmals) Haar-Wavelets d_x und d_y für 5×5 gleichmäßig verteilte Punkte ermittelt. Die Ergebnisse der Haar-Wavelets werden dann aufsummiert und zusätzlich werden noch die Beträge der einzelnen Komponenten aufsummiert. Hiermit erhält man einen vierdimensionalen Vektor für jeden der 16 Unterbereiche:

$$v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \quad (2.9)$$

Wie diese Methode eine Charakterisierende Beschreibung für das Umfeld eines Merkmals ist beschreiben die Autoren im folgenden Bild.



Wie auch der Merkmalsvektor bei SIFT, ist dieser bei SURF invariant gegenüber

globalen Beleuchtungsveränderungen. Auch hier wird eine Kontrastinvarianz durch die Normierung erzielt. [2]

2.3 ORB

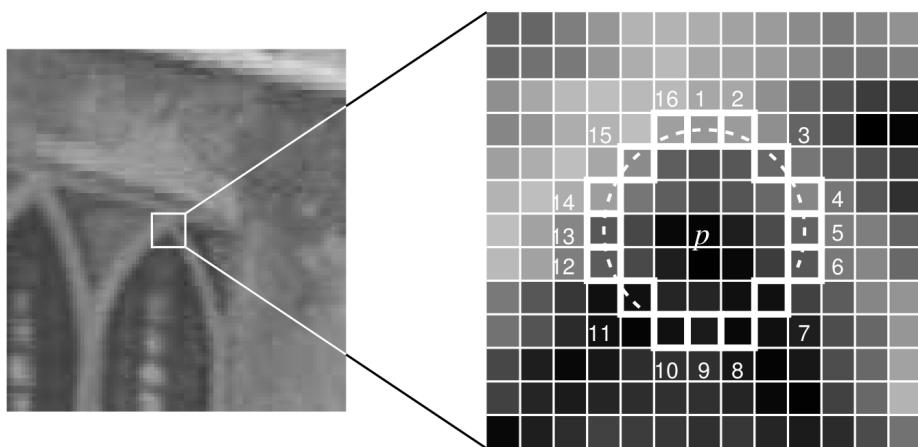
Ein weiterer Algorithmus um effizient Merkmale aus Bildern zu extrahieren wurde von Edward Rosten und Tom Drummond unter dem Namen ORB vorgestellt. ORB steht für *Oriented FAST and Rotated BRIEF* da die Merkmalslokalisierung auf FAST basiert und die Deskriptorbeschreibung eine verbesserte Version von BRIEF ist.

2.3.1 Merkmalsextraktion

Um die Funktionsweise von Oriented FAST besser zu beschreiben wird im folgenden der darunterliegende Algorithmus FAST erläutert.

FAST

Der *Features from Accelerated Segment Test* (FAST) ist vom Segment Test abgeleitet.



Dieser untersucht das Umfeld der einzelnen Punkte im Bild in dem er den umliegenden Kreis mit 16 Pixeln Länge mit dem Zentrum vergleicht. Um möglichst schnell viele Bildpunkte raus zu filtern werden vorerst nur die Pixel 1, 5, 9 und 13 berücksichtigt

und nur Punkte bei den 3 von diesen Pixeln heller oder dunkler als der berücksichtigte Bildpunkt sind als Kandidaten genommen. Die Bildpunkte die den ersten Test bestehen werden anschließend genauer analysiert in dem jedes Pixel des umliegenden Kreises mit dem Zentrum verglichen wird. Wenn der Kreis eine zusammenhängende Kette von 12 Pixeln die heller oder dunkler als der zugehörige Bildpunkt sind beinhaltet, wird dieser als Merkmal hinterlegt. Um diesen Vorgang zu optimieren wird bei FAST von maschinellem lernen gebraucht gemacht: es wird ein *Decision Tree* wie von Quinlan in *Induction of Decision Trees* beschrieben aufgebaut. Um das System einzulernen wird ein Set aus Bildern genutzt, die bestenfalls dem Typ von Umgebung entsprechen in dem das Bildverarbeitungssystem eingesetzt werden soll. Jedes Pixel dieser Bilder wird mit jedem Punkt $x \in 1..16$ des umliegenden Kreises verglichen um dann jedem Kreispunkt einen Zustand zuzuweisen:

$$S_{p \rightarrow x} = \begin{cases} d, & I_{p \rightarrow x} \leq I_p - t \\ s, & I_p - t < I_{p \rightarrow x} < I_p + t \\ b, & I_p + t \leq I_{p \rightarrow x} \end{cases} \quad (2.10)$$

wobei I_p die Intensität des Bildpunktes und $I_{p \rightarrow x}$ die Intensität des Pixels an der Stelle x auf dem Kreis um I_p ist. Anhand dieses Zustands werden alle Pixel p der Menge P aller Bildpunkte der Trainings-Bilder in die 3 Untermengen P_d , P_s und P_b unterteilt. Um das x zu bestimmen welches am besten aussagt ob ein Pixel eine Ecke ist, wird die Entropie der Ecken-Eigenschaft K (bei Pixeln p die als Ecke erkannt wurden hat K_p den Wert 1) in den Mengen P, P_d, P_s und P_b berechnet. Hieraus wird der Informationsgewinn berechnet als:

$$H(P) - H(P_d) - H(P_s) - H(P_b) \quad (2.11)$$

Das x mit dem höchsten Informationsgewinn wird zur Unterteilung von P in seine Teilmengen verwendet. Dieser Vorgang wird so lange wiederholt bis die Entropie einer der Untermengen zu 0 wird und somit entweder nur oder keine Ecken enthält.

Oriented FAST

Da FAST keine Orientierungskomponente hat ist der Algorithmus nicht rotationsinvariant. Um dieses Problem zu beheben wird von dem *Intensitäts-Schwerpunkt* Gebrauch gemacht da dieser bei einer Ecke nie mit dem Mittelpunkt übereinstimmt. Dieser Schwerpunkt wird wie folgt berechnet:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \quad (2.12)$$

mit den Momenten:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \quad (2.13)$$

Dem Merkmal wird dann die Orientierung des Vektors der vom Mittelpunkt des Winkels $O = (0, 0)$ zum *Intensitäts-Schwerpunkt* zeigt zugewiesen:

$$\Theta = \text{atan2}(m_{01}, m_{10}) \quad (2.14)$$

Um die Rotationsinvarianz zu optimieren, werden die Momente nur aus Pixeln berechnet deren x und y Werte innerhalb eines durch den Radius r um den Mittelpunkt aufgespannten Kreises liegen.

Bild Intensitätsschwerpunkt?

2.3.2 Deskriptorberechnung

BRIEF

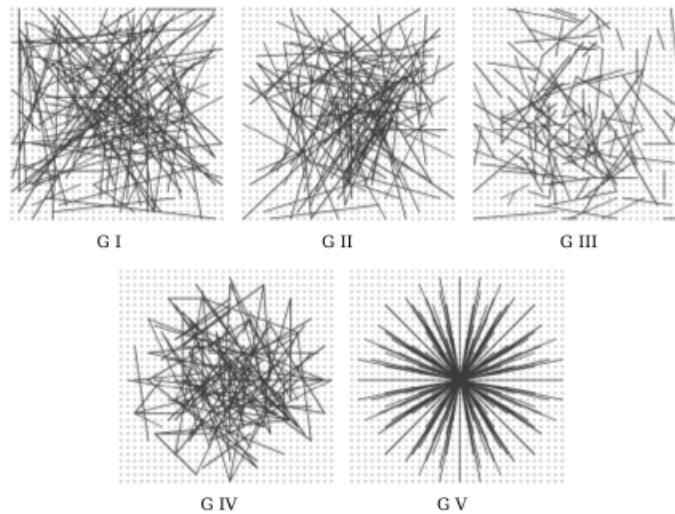
BRIEF ist eine Methode zur Merkmalsextraktion die gegenüber SURF noch einen Schritt weiter in der Rechenzeitoptimierung geht. Hierzu verwenden die Autoren die Erkenntnisse der Arbeiten *Fast Keypoint Recognition Using Random Ferns* und *Keypoint Recognition Using Randomized Trees*. Diese haben gezeigt das Teile eines Bildes effektiv durch wenige Vergleiche der Intensität in Pixelpaaren beschrieben werden können. Um einen BRIEF Deskriptor zu berechnen wird der folgende Test $\tau(p; x, y)$ auf

die zu untersuchenden Pixelpaare angewandt und die Binären Ergebnisse in einem Vektor gespeichert:

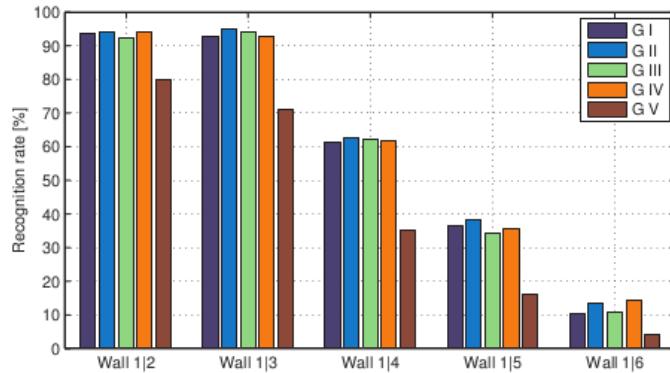
$$\tau(p; x, y) = \begin{cases} 1, & \text{if } p(x) < p(y) \\ 0, & \text{otherwise} \end{cases} \quad (2.15)$$

Um die Störungsanfälligkeit zu reduzieren wird das Bild davor mit einem Gaussfilter geglättet. Um den Filter effektiv zu nutzen haben die Autoren die drei Einflussgrößen experimentell bestimmt:

- *Kernel des Gaussfilters*: Nachdem verschiedene Gausskurvenbreiten σ zwischen 0 und 3 getestet wurden, hat sich $\sigma = 2$ als die beste Wahl herausgestellt. Als Größe des Kernels wurde 9x9 Pixel genommen.
- *Räumliche Anordnung der Tests*: Wie die Versuche mit verschiedenen Strategien zur bestimmung der Anordnung der Tests gezeigt haben, spielt diese eine wichtige Rolle für die Güte (Wiedererkennungsrate) der Deskriptoren. Folgende Anordnungen wurden verglichen:



Die ergebnisse des Vergleichs sind im folgenden Graphen zu sehen: Die symmetrische Anordnung ist mit Abstand die schlechteste und die Zweite ist in den



meisten Fällen ein bisschen im Vorsprung gegenüber dem Rest, daher wird diese verwendet. Es handelt sich hierbei um eine isotropische Gaussverteilung mit $\sigma^2 = \frac{1}{25}S^2$ wobei S die Seitenlänge des zu betrachtenden viereckigen Bildausschnitts ist.

Rotated BRIEF

BRIEF besitzt eine sehr schlechte Rotationsinvarianz, daher wurde für ORB die Variante Rotated BRIEF entworfen. Für das Set aus Tests zur Erstellung des Deskriptorvektors an den Stellen (x_i, y_i) wird diese Matrix definiert:

$$S = \begin{pmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \end{pmatrix} \quad (2.16)$$

Anschließend wird die der Orientierung Θ des Merkmals entsprechende Rotationsmatrix R_Θ verwendet um S_Θ wie folgt zu berechnen:

$$S_\Theta = R_\Theta \cdot S \quad (2.17)$$

Θ wird auf Abschnitte mit einer Breite von $\frac{2\pi}{30}$ diskretisiert und es wird eine Lookup-Tabelle mit den Positionen der zu testenden Punkte für jeden Diskretisierungsschritt im fertigen Programm hinterlegt. Somit lassen sich die rotationsunabhängigen Deskriptoren praktisch genauso schnell wie die rotationsabhängigen berechnen. [3]

2.4 Bruteforce Matcher

Der *Bruteforce Matcher* ist ein von OpenCV bereitgestelltes Werkzeug um Übereinstimmungen zwischen extrahierten Merkmalen zu finden. Hierzu wird die sogenannte Distanz zwischen den jeweiligen Deskriptoren der Merkmale berechnet. Dieser Wert ist ein Maß für die Ähnlichkeit zweier verglichener Deskriptoren und somit die Wahrscheinlichkeit dass die zugehörigen Merkmale Übereinstimmen. Diese Distanz wird durch die Differenz verschiedener Vektornormen bestimmt. Die Wahl des richtigen Algorithmus hängt unter anderem von dem Typ des Deskriptors ab (Fließkomma- oder Binärwerte) und kann durch den Parameter *normType* festgelegt werden. Es stehen folgende Vektornormen zu Verfügung:

- *NORM_L1*
- *NORM_L2*
- *NORM_HAMMING1*
- *NORM_HAMMING2*

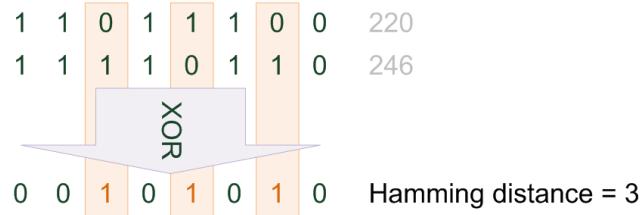
Für das Matching der SIFT und SURF Deskriptoren wird in dieser Arbeit die *NORM_L2* verwendet. Diese auch unter dem Namen *Euclidische Norm* bekannte Größe wird nach folgender Formel berechnet:

$$\ell^2 = \|x\| = \sqrt{\sum_{k=1}^n |x_k|^2} \quad (2.18)$$

wobei $|x_k|$ der Betrag der komplexen Zahl x_k ist und somit für Deskriptoren mit rein reellen Zahlen (wie SIFT und SURF) $|x_k|^2 = x_k^2$ ist.

Um das Matching der ORB Deskriptoren durchzuführen wird von *NORM_HAMMING1* gebraucht gemacht. Es handelt sich nicht um eine Vektor-Norm im engen Sinn wird aber manchmal als eine solche bezeichnet. Diese Größe ist auch als *Hamming Distanz* bekannt und ist die Anzahl unterschiedlicher Bits

in zwei gleich langen Bit-Ketten. Hierzu werden alle Positionen in den zwei Ketten stellenweise verglichen und falls die Werte nicht übereinstimmen die *Hamming Distanz* um 1 erhöht.



Diese Operation kann mit minimalen Kosten berechnet werden da es genügt die beiden Bit-Ketten mit einem *XOR* Operator zu vergleichen und die positiven Bits zu zählen.
[4]

2.5 Homographie

Homographie ist der mathematische Zusammenhang von zwei 2-Dimensionalen Abbildungen eines 3-Dimensionalen Objekts. Die Berechnung um diese zu finden wird in P^2 , der homogenen Darstellung, durchgeführt. Ein Punkt $P' \in \mathbb{R}$ in kartesischer Darstellung kann aus dem Punkt $P \in \mathbb{P}$ in homogener Darstellung durch die folgenden Beziehungen abgeleitet werden:

$$x' = \frac{x}{z} \quad \text{und} \quad y' = \frac{y}{z} \quad (2.19)$$

Umgekehrt wird P aus P' wie folgt bestimmt:

$$P' = \begin{pmatrix} x' \\ y' \end{pmatrix} \quad \rightarrow \quad P = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad (2.20)$$

Hartley und Zisserman bezeichnen die Homographie wie folgt

Ein Zusammenhang von $\mathbb{P}^2 \rightarrow \mathbb{P}^2$ ist eine Homographie wenn eine nicht singuläre 3x3 Matrix H existiert sodass aus jedem Punkt x der zusammenhängende als Hx berechnet werden kann. [6]

2.6 RANSAC

Random Sample Consensus ist ein Algorithmus um die Übereinstimmung von einem mathematischen Modell mit Testdaten zu überprüfen und somit das sogenannte Matching durchzuführen. Das zuvor verbreitete Matchingverfahren *Least Median of Squares Regression* berücksichtigte für die Auswertung die Gesamtheit der Testdaten und konnte somit nur auf Testdaten mit einem Ausreißeranteil der kleiner als 50% ist angewandt werden. Hierfür wurde RANSAC entwickelt. Es wird zunächst eine Teilmenge S1, mit der minimalen Anzahl an Punkten um die nötigen Parameter der Modellfunktion zu berechnen, zufällig aus der Menge der Testdaten P gewählt. Anschließend wird das *Consensus Set* S1* aus Punkten die innerhalb einer Tolleranzgrenze dem gefundenen Modell entsprechen bestimmt. Wenn diese Menge eine gewisse Mindestanzahl an Punkten enthält wird es als Kandidat für die Lösung genommen. Diese Schritte werden mehrmals wiederholt um die Modellhypothese mit dem Größten Consensus Set zu finden. Hierbei ergeben sich drei Parameter für RANSAC:

- Zugelassene Fehlertoleranz um ein Punkt in ein Consesus Set aufzunehmen
- Anzahl der Iterationen
- Mindestanzahl der Punkte in einem Consesus Set

[7]

Kapitel 3

Merkalsextraktion

3.1 Verwendete Software

Um die verschiedenen Verfahren zur Merkalsextraktion zu vergleichen wurde ein Programm in C++ erstellt. Also Entwicklungsumgebung wurde die freie und quellenoffene Software Eclipse verwendet. Die genutzten Implementierungen der Bildverarbeitungsalgorithmen stammen aus der Bibliothek OpenCV. Hierbei handelt es sich um eine unter der BSD Lizenz veröffentlichten Software die sehr viele Methoden zur Bildverarbeitung und Maschinellem sehen enthält.

3.2 Hardware

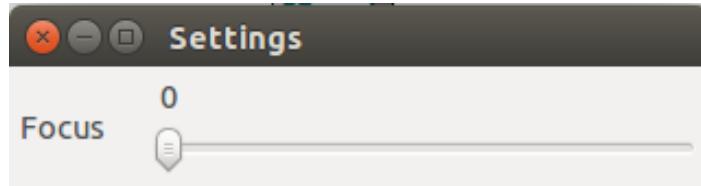
Das Programm wurde auf einem Laptop mit folgender Hardware erstellt und getestet:

- Intel Core i7-4720HQ mit bis zu 3.6 GHz
- 8 GB DDR3L-SDRAM mit 1600 MHz
- Nvidia Geforce GTX 960M

Als Kamera wurde eine Logitech HD Pro C920 Webcam verwendet, da diese sich durch ein sehr gutes Preis-Leistungs Verhältnis auszeichnet.

3.3 Erstellte Software

Zum testen und vergleichen der drei behandelten Merkmalsverfahren wurde ein Programm erstellt, das mit Hilfe dieser durch Bilder vorgegebene Objekte in dem Videostream einer Webcam erkennt. Der verwendete Algorithmus lässt sich durch das setzen einer Variable bestimmen. Dies ist zwar nicht so benutzerfreundlich wie die Übergabe eines Arguments beim Aufruf des Programms, da bei jeder Änderung neu kompiliert werden muss, aber für Testzwecke ist es ausreichend. Nachdem die Header-Dateien der benötigten Bibliotheken eingebunden wurden, werden die Bilder der zu suchenden Objekte aus einem definierbaren Ordner geladen. Es können theoretisch beliebig viele Bilder in diesen Ordner eingefügt werden, es wurden aber maximal 6 gleichzeitig getestet. Da der Autofocus der Webcam nicht sehr zuverlässig funktioniert, wurde eine Manuelle Focus Funktion in Form einer OpenCV Trackbar Implementiert. Um die Da-

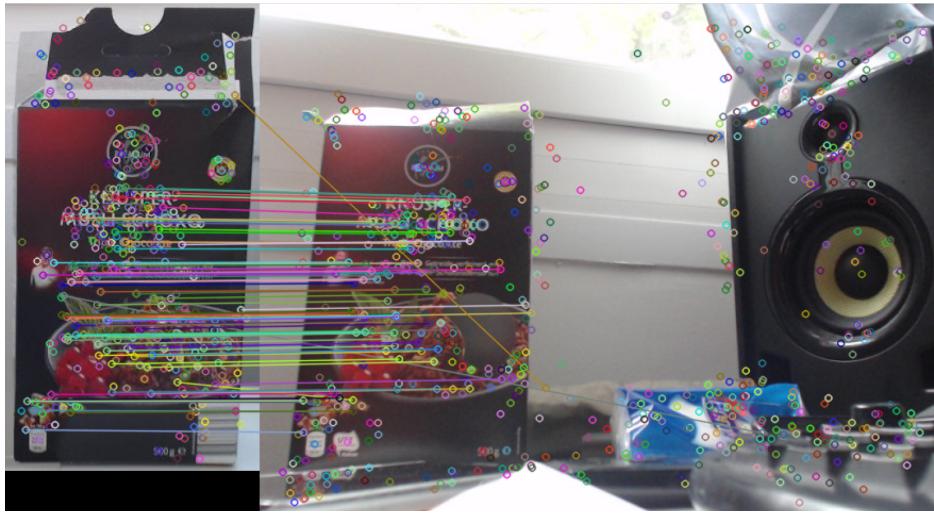


ten der Kamera verarbeiten zu können wird ein VideoCapture Objekt angelegt über das man die Mat-Container der einzelnen Frames beziehen kann. Um die erkannten Objekte am besten Farblich darzustellen wird der Farbkreis in die Anzahl der zu erkennenden Objekte unterteilt. Das lässt sich am einfachsten im HSI Modell realisieren in dem jedes Objekt einen Hue wert:

$$H = i \cdot \frac{180}{\text{Anzahl der Objekte}} \quad \text{mit } 0 < i < \text{Anzahl der Objekte} \quad (3.1)$$

Die so erhaltenen Farben werden anschließend in das RGB-Format umgerechnet und in einen Vektor für die spätere Verwendung gespeichert. Da die Algorithmen auf Graustufenbildern arbeiten werden die Bilder der zu erkennenden Objekte konvertiert und in ein separates Array gespeichert. Anschließend werden nach dem gewünschten Verfahren die Keypoints aus den Bildern extrahiert und die Deskriptoren berechnet. Dies

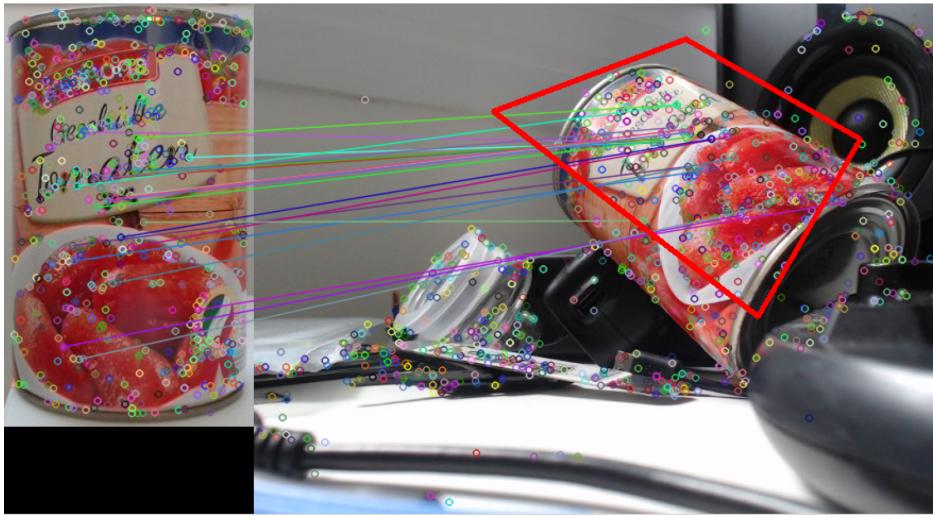
wird anhand der Methoden *detect* und *compute* der Klassen *xfeatures2d::SIFT*, *xfeatures2d::SURF* und *ORB* bewerkstelligt. Falls bei diesem Schritt ein Fehler auftritt muss das betroffene Objekt in Form des zugehörigen Bildes, der Keypoints und der Deskriptoren aus den jeweiligen Vektoren gelöscht werden. Dies ist insbesondere wichtig da die zu einem Objekt gehörenden Daten nur durch die Stelle in den Vektoren zusammenhängen. Nachdem die Merkmale aus den Objektbildern extrahiert wurden, springt das Programm in eine nur durch die Beendung des Programms unterbrechbare Endlosschleife in der die aktuellen Frames der Webcam bearbeitet werden. Das neueste Frame wird in Form eines Mat-Objekts erfasst und, wie die Bilder der darzustellenden Objekte, wird auch dieses in ein Graustufenbild verwandelt um anschließend die Merkmale in Form von Keypoints und Deskriptoren zu extrahieren. Nachdem nun alle Merkmale aus den Objekt-Bildern und aus dem aktuellen Frame vorhanden sind, gilt es das Set aus Merkmalen jedes Objekts mit jenen des neuesten Frames zu vergleichen um mögliche Übereinstimmungen zu finden. Hierzu wird als erstes von dem BruteForceMatcher (BFMatcher) aus OpenCV gebrauch gemacht. Dieser berechnet die Übereinstimmung der Merkmale anhand der Deskriptoren. Da dieser aber noch viele falsche Matches ausgibt, werden diese anschließend noch anhand der von Lowe vorgestellten Methode gefiltert. Es werden nur Matches bei denen das Verhältnis der Deskriptordistanzen (wie in 2.4 beschrieben, sagt diese Distanz aus wie unterschiedlich zwei Deskriptoren sind) des zweitbesten zum erstbesten Match kleiner als ein bestimmter Wert (in unserem Fall 0.9) sind beibehalten. Dies bedeutet in der Praxis dass es nur einen wahrscheinlichen Kandidaten für das Matching gibt und somit die Wahrscheinlichkeit einer Fehlzuweisung gering ist. Die so erhaltenen Matches werden anschließend mit der OpenCV Funktion *drawMatches* in das Bild eingezeichnet. Anschließend wird aus den gefundenen Matches die Homographie, welche die Ausrichtung des gefundenen Objekts zu dessen hinterlegtem Bild beschreibt, berechnet. Hierzu wird die OpenCV-Funktion *findHomography* verwendet. Beim Aufruf dieser Funktion wird der Parameter *method* übergeben, durch welchen man festlegen kann ob die Homographie herkömmlich berechnet wird oder mit Hilfe von RANSAC oder Least-Median die



Ausreißer rausgefiltert werden. Da die Punktemenge der wiedererkannten Keypoints aus der Homographie ermittelt wird typischerweise Falschzuordnungen enthält, kommt die herkömmliche Methode nicht in Frage. Bei Versuchen mit den beiden letzteren Methoden ergab sich RANSAC als die stabilere Variante und wird somit hier verwendet. Mit der gefundenen Homographie-Matrix kann anschließend die Position des Objekts im aktuellen Frame ermittelt werden. Hierzu werden die Eckpunkte der hinterlegten Abbildung in einen Vektor gespeichert und samt der Homographie an die Funktion *perspectiveTransform* übergeben. Diese liefert die Position der Eckpunkte im Frame, hierdurch ist die aktuelle Lage des Objekts bestimmt. Mit Hilfe der Funktion *line* wird eine Box um das gefundene Objekt gezeichnet.

3.3.1 GPU Beschleunigung

Da die hier verwendeten Algorithmen zum Teil sehr rechenintensiv sind und auf dem Test-Laptop nicht alle Frames der Webcam verarbeitet werden konnten, wurde die Berechnung eines Teils der Operationen auf die GPU verlagert. Dank der CUDA Bibliothek von NVIDIA und den Implementierungen der Bildverarbeitungsalgorithmen aus OpenCV die auf dieser basieren ist das relativ leicht zu realisieren. Um die Bibliotheken nutzen zu können muss OpenCV samt den Plugins des *contrib* Repository aus dem Quellcode kompiliert werden. Zusätzlich zu den der üblichen *Mat*, müssen



hier `cuda::GpuMat` Klassen verwendet werden. Diese werden mit dem `Mat` Bild initialisiert, somit wird dieses auf die Grafikkarte kopiert und es wird in der `GpuMat` ein Pointer auf dieses gespeichert. Anschließend können die Klassen `cuda SURF_CUDA` und `cuda ORB` verwendet werden um die Berechnungen durchzuführen. Für SIFT gibt es keine CUDA Implementierung, vermutlich aus patentrechtlichen Gründen. Daher wurde versucht die SIFT Implementierung *CudaSift* des Github Benutzers *Celebrandil* in das Programm einzubauen. Leider ist es nicht gelungen die Ausgangsschnittstellen dieser Bibliothek mit dem restlichen Programm zu verknüpfen.

3.3.2 Benchmark

Um die verschiedenen Methoden zu vergleichen wurden verschiedene Tests durchgeführt.

Kapitel 4

Ergebnisse

4.1 Laufzeit

Um die verschiedenen Methoden zu vergleichen wurden verschiedene Tests durchgeführt. Als erstes wurde die Laufzeit der einzelnen Verfahren verglichen. Hierzu wird von der *clock_gettime* Funktion gebraucht gemacht. Diese liefert einen *struct* aus dem die Zeit bis auf Nanosekunden-Genauigkeit ausgelesen werden kann. Es wurden folgende Laufzeiten für jeden Algorithmus ermittelt:

- Extraktion der Merkmale
- Matching der Deskriptoren
- Berechnung der Homographie
- Gesamte Bearbeitung eines Frames

Die für den ersten Punkt benötigte Zeit ist stark von der gewählten Methode abhängig. Die Matching-Zeit hängt zum einem von dem Deskriptortyp ab und zum anderen ist diese sehr von der Umgebung bestimmt da sie von der Anzahl der Merkmale abhängt. Die Laufzeit der Homographie-Bestimmung hängt auch von der Merkmalsanzahl ab. In Volgender Tabelle sind die Durchschnittlichen Laufzeiten aufgelistet. Diese wurden bei der Bearbeitung eines Testvideos ermittelt.

	Extraktion	Matching	Homographie	Gesamt
SIFT	370	90	2	495
SURF	111	47	2	184
ORB	17	25	6	64

4.2 Anzahl der korrekten Matches

Als weiteres Testkriterium wurde die Anzahl der richtig zugewiesenen Merkmale bestimmt. Hierzu wurde von der Bitmaske die von *findHomography* zurückgegeben wird Gebrauch gemacht. Aus dieser können die tatsächlich von der Funktion verwendeten Matches bestimmt werden. Die gefundenen Matches passen somit in die Gesamtkonstellation und werden als richtig eingestuft. Um möglichst aussagekräftige Ergebnisse zu erhalten wurden 40 sekündige Videos der Testobjekte aufgenommen und anschließend die durchschnittlichen Korrekten Matches für jeden Algorithmus berechnet. Um unvorhersehbare Effekte beim Startvorgang des Programms auszuschließen wurden für die Mittelwerte nur Matches ab der 10. Sekunde berücksichtigt.

	Kork	Müsli	Tomatendose	Kirschglas	Spardose
SIFT	80	298	181	161	46
SURF	11	149	80	78	17
ORB	26	156	110	170	59

Wie erwartet schneiden die Merkmale die von einem SIFT-Deskriptor beschrieben werden bei der Wiedererkennung am besten ab. Bei 3 von den 5 Testobjekten werden deutlich mehr Merkmale korrekt wiedererkannt als bei den anderen beiden Methoden. Allerdings werden bei den schwerer zu erkennenden Objekten anhand des ORB-Deskriptors mehr Merkmale richtig zugeordnet. SURF liefert in diesem Test für alle 5 Objekte das schlechteste Ergebnis.

4.3 Stabilität gegen Rotation

Beim zweiten Vergleich wurde getestet wie rotationsinvariant die Algorithmen sind. Hierzu wurde jedes Objekt soweit gedreht bis keine eindeutige Lokalisierung mehr möglich war. Hierbei wurde darauf geachtet dass bei den Vergleichen die Position der Kamera konstant blieb und die Objekte nur um ihre senkrechte Achse gedreht wurden. Nachdem die maximale Rotation erreicht wurde, wurde der Winkel gemessen und auf 5 Grad gerundet, da eine genauere Messung mit den vorhandenen Mitteln nicht möglich war.

	Kork	Müsli	Tomatendose	Kirschglas	Spardose
SIFT	55	60	55	65	35
SURF	45	55	50	45	35
ORB	45	45	30	55	25

Auch hier liefert SIFT die besten Ergebnisse, selbst bei den schwerer zu erkennenden Objekten, obwohl bei diesen weniger Merkmale als bei ORB korrekt zugeordnet wurden. Auch SURF ist trotz der wesentlich geringeren Menge an Merkmalen im schnitt besser als ORB.

4.4 Stabilität gegen Skalierung

Eine weitere wichtige Eigenschaft der Merkmale ist die Skalierungsinvarianz. Diese führt dazu dass Objekte auch erkannt werden wenn sie kleiner oder größer (weiter oder weniger weit entfernt) als auf dem Modellbild sind. Da die SURF und ORB Implementierungen von OpenCV nicht mit kleinen Bildern funktionieren, und die Modellbilder somit nicht runterskaliert werden konnten, war es nicht möglich die Erkennung von Objekten die größer als ihr Modellbild sind zu testen. Da in der Praxis die zu erkennenden Objekte aber kleiner als deren Modellbilder sind, stellt dies aber kein wirkliches Problem da. Um die Algorithmen auf die Fähigkeit kleinere Exemplare der Objekte wiederzuerkennen zu testen, wurde die maximale Distanz von der Kamera zum

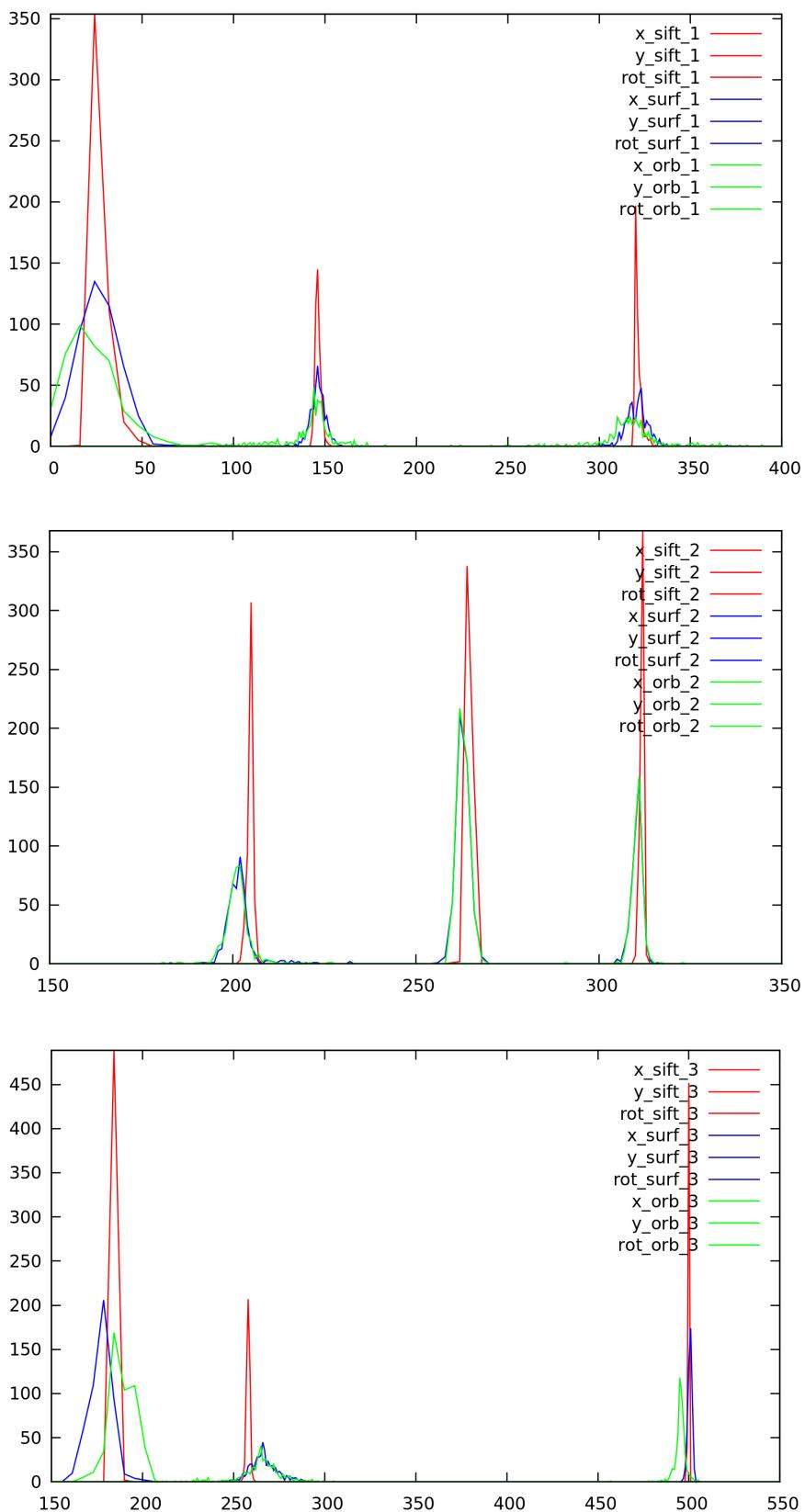
Objekt gemessen bei der dieses noch korrekt erkannt werden konnte.

	Kork	Müsli	Tomatendose	Kirschglas	Spardose
SIFT	50	265	120	110	70
SURF	25	140	65	70	50
ORB	30	125	45	50	40

Wie bei den vorherigen Tests schnitt SIFT mit Abstand am besten ab und schaffte es, für die Verhältnisse (Consumer Grade Webcam mit VGA Auflösung), Objekte aus einer sehr großen Entfernung zu erkennen. SURF schnitt bei allen Objekten außer dem Korkuntersetzer besser als ORB ab.

4.5 Stabilität der Lokalisierung

Ein weiteres Maß für die Güte der verschiedenen Extraktionsverfahren ist die Abweichung der Lokalisierung von der Istposition der Objekte. Genauer ist hier die quadratische mittlere Abweichung, die Varianz, von Bedeutung. Um diese zu ermitteln wurde auch hier ein Testvideo erstellt um gleiche Bedingungen zu gewährleisten. Es wurde ein Histogramm von der Verteilung der X-Koordinate, der Y-Koordinate und dem Ausrichtungswinkel in Form eines Vectors erstellt. Die als •csv-Datei aus dem Programm exportierten Daten wurden mittels Gnuplot als Graphen dargestellt. Die Verteilungen sind in den folgenden drei Graphen zu sehen, hierbei ist links die Rotation zu sehen und rechts davon die Translation in x und y Richtung. Die Translation ist jeweils in Pixeln angegeben, dies ist auch die Einheit der Zahlen auf den Achsen. Die Rotation ist hingegen in Grad bemessen und wurde im Vergleich der angeschriebenen Zahlen gestreckt.



Wie zu erwarten da dies auch anhand der Videoausgabe des Programms deutlich sichtbar ist, zeigen die Graphiken deutlich dass die Lokalisierung durch den SIFT Algorithmus wesentlich stabiler als bei den anderen beiden Varianten ist. SURF lokalisiert die Objekte mit größerer Genauigkeit als ORB, hat hierbei aber nur einen mäßigen Vorteil. Man sieht dass die Verteilungen in Einigen Fällen einer summe mehrerer Normalverteilungen entsprechen. Dies kommt daher dass mehr als eine Konstellation an Merkmalen dem Modell ähnlich sind und somit durch die Störeinflüsse manchmal eine vollkommen falsche Position ausgegeben wird.

4.6 Testvideos

Der letzte Vergleich wurde anhand von zwei Testvideos gemacht. Diese beinhalten die 5 Beispielgegenstände in zwei verschiedenen Anordnungen. Die Leistung der 3 Algorithmen wurde die Anzahl der Frames in welchen die Objekte erkannt wurden bestimmt. Da es bei der gleichzeitigen Verwendung von mehreren Objekten leichter zu Fehlerkennungen kommt wurden nur Frames in denen ein Objekt mit mindestens 5 Matches erkannt wurde gezählt. Dies führt dazu dass es zu so gut wie keiner Fehlzuweisung kommt (weniger als 5 pro Video beobachtbar). In folgender Tabelle sind die Anzahlen der Frames des ersten Testvideos in denen die jeweiligen Objekte gefunden wurden aufgelistet:

	Kork	Müsli	Tomatendose	Kirschglas	Spardose
SIFT	268	415	392	212	203
SURF	x	400	366	162	77
ORB	x	370	342	150	56

Es folgen die Frames des zweiten Videos:

	Kork	Müsli	Tomatendose	Kirschglas	Spardose
SIFT	300	793	517	669	696
SURF	x	695	285	548	443
ORB	2	531	123	341	114

Zusätzlich wurden die durchschnittlichen Zuweisungen die zu der Erkennung der Objekte geführt haben berechnet. Erstes Video:

	Kork	Müsli	Tomatendose	Kirschglas	Spardose
SIFT	17	124	40	31	11
SURF	x	48	10	10	5
ORB	x	66	11	11	9

Zweites Video:

	Kork	Müsli	Tomatendose	Kirschglas	Spardose
SIFT	9	65	22	44	26
SURF	x	20	7	13	11
ORB	6	23	9	14	18

Wie nach den vorherigen Tests zu erwarten, stellte sich auch hier SIFT als der Beste Merkmalsextrahierungsalgorithmus heraus. SURF extrahierte wie in einem der vorangegangenen Tests weniger Merkmale als ORB, schaffte es aber trotzdem die Objekte besser zu detektieren als ORB. Das ist eine Folge der eindeutigeren Beschreibung der Keypoints durch den SURF-Deskriptor. Der große Vorteil von SIFT gegenüber den anderen beiden Verfahren ist hierbei nicht nur dass die Objekte öfter erkannt werden sondern auch die Stabilität der Detektion. Hierdurch lässt sich in einer wie in den Testvideos dargestellte Umgebung auch die Orientierung der Objekte mit sehr guter Annäherung bestimmen, während SURF und ORB hier eher nur zur Positionsbestimmung des Mittelpunktes der Objekte geeignet sind.

Kapitel 5

Ausblick

Die Ergebnisse zeigen dass sich die untersuchten Algorithmen zufriedenstellend für die Klassifizierung von bekannten Objekten eignen. Um diese aber mit genügender Genauigkeit zu lokalisieren sind zumindest SURF und ORB nicht ausreichend. Auch mit SIFT wird in manchen Situationen die Orientierung der Objekte nicht genau erkannt. Um die Detektion und die Lokalisierung der Objekte zu stabilisieren könnte ein Kalman-Filter implementiert werden. Dieser verwendet ein Bewegungsmodell welches von der Natur der Objekte und/oder der Bewegung der Kamera abhängt. Im Zusammenspiel mit den Messungen der Position durch den in dieser Arbeit beschriebenen Prozess und der zu erwartenden Ungenauigkeit schätzt der Filter die aktuelle Position der Objekte. Dies hat den Vorteil dass das Ergebnis nicht nur von der aktuellen Messung sondern auch von den vorherigen abhängt und somit das Rauschen der Messungen kompensiert werden kann.

Kapitel 6

Zusammenfassung

In dieser Arbeit wurden drei der meist verbreiteten Merkmalsextraktionsverfahren in einem Programm implementiert um sie untereinander zu vergleichen. Die drei Algorithmen weisen jeweils Stärken und Schwächen auf. Die Güte der Merkmale ist bei SIFT unter jedem Aspekt mit Abstand am besten. Der Schwachpunkt dieses Verfahrens ist allerdings die Performance

Abbildungsverzeichnis

Glossar

octave . 4, 9

Online-System . 8

scale . 4, 5, 6, 9

scale-space . 3, 4, 5

Literaturverzeichnis

- [1] David G. Lowe, *Object Recognition from Local Scale-Invariant Features*, International Conference on Computer Vision, 1999.
 - [2] H Bay, A Ess, T Tuytelaars, L Van Gool, *Speeded-up robust features (SURF)*, Computer vision and image understanding, 2008.
 - [3] E Rublee, V Rabaud, K Konolige, G Bradski, *ORB: An efficient alternative to SIFT or SURF*, Computer Vision (ICCV), 2011.
 - [4] *The OpenCV Reference Manual*, Itseez, 2017.
 - [5] *Open Source Computer Vision Library*, Itseez, 2017.
 - [6] R. Hartley, A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, Cambridge, 2000.
 - [7] M. Fischler, R. Bolles, *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*, SRI International, 1980.
-