Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Lars Drössler

Eignungsuntersuchung von Klassifikations- und Deep-Learning-Verfahren
zur kamerabasierten Erkennung natürlicher Objekte

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Lars Drössler

Eignungsuntersuchung von Klassifikations- und Deep-Learning-Verfahren zur kamerabasierten Erkennung natürlicher Objekte

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Prof. Dr.-Ing. Andreas Meisel

Abgegeben am 15.05.2017

Lars Drössler

Thema der Bachelorarbeit

Eignungsuntersuchung von Klassifikations- und Deep-Learning-Verfahren zur kamerabasierten Erkennung natürlicher Objekte

Stichworte

Convolutional Neural Network, SIFT, SURF, ORB, AKAZE, Deep-Learning, Food, Computer Vision, Haar-like Features

Kurzzusammenfassung

In dieser Arbeit wird eine Auswahl an Verfahren vorgestellt und auf ihre Eignung zur Klassifizierung natürlicher Objekte untersucht. Hierzu werden eine Reihe von Experimenten basierend auf publizierten Verfahren durchgeführt, in denen überprüft wird, wie gut die entsprechenden Verfahren natürliche Objekte in Fotos erkennen können. Als natürliche Objekte wurden für diese Testreihe verschiedene Gemüse- und Obstsorten gewählt. Diese Arbeit ist dazu ausgelegt, als Entscheidungshilfe für die Erstellung von Anwendungen zu dienen, in welchen es notwendig ist natürliche Objekte korrekt zu klassifizieren.

Lars Drössler

Title of the paper

Examination of classification and deep learning procedures for the camera based recognition of natural objects.

Keywords

Convolutional Neural Network, SIFT, SURF, ORB, AKAZE, Deep Learning, Food, Computer Vision, Haar-like features

Abstract

This paper presents a selection of methods and examines them for their suitability to recognize natural objects. For this purpose, a series of experiments based on published procedures are conducted. These experiments test how said procedures perform and if they are used to recognize natural objects in images. The natural objects used in these experiments are different kinds of vegetables and fruits. This paper is intended to serve as a decision-making aid for the implementation of an application which should, amongst other things, recognize natural objects.

Inhaltsverzeichnis

1 Einleitung

In der heutigen Zeit halten Computer immer mehr Einzug in unser alltägliches Leben. Wo vor Jahren noch darüber geschrieben wurde, dass nun die E-Mail allmählich die Post ablöst [Welteroth2005] oder Digitale Musik ihr analoges Gegenstück verdrängt [Linz2015], ist heute davon die Rede, dass selbst fahrende Autos in naher Zukunft die Notwendigkeit der manuellen Steuerung obsolet machen könnten [Levin et al. 2017]. Nicht zuletzt trägt wohl die rasante Entwicklung von Hard- und Software in den letzten Jahren dazu bei. Der Einfluss dieses Fortschrittes auf unseren Alltag lässt sich wohl am besten durch die Millionen an erhältlichen Apps für jede mögliche Alltagssituation verdeutlichen. Selbst in unseren Wohnungen findet immer mehr komplexer werdende Technik Verwendung. Unter dem Schlagwort „Smart Home“ findet man zahlreiche Projekte, welche versuchen den Alltag im Haus durch ausgeklügelte Technik und durchdachte Software zu vereinfachen. Leider bleibt bei der ganzen Entwicklung die Interaktion zwischen Mensch und Maschine ein wenig auf der Strecke. Natürlich gibt es schon einige Anwendungen, wie Apples Siri oder Androids Assistant, bei denen eine natürlichere Interaktion über beispielsweise Spracheingabe stattfindet. Dies ist aber eher eine Ausnahme. Eines der größeren Ziele vieler Entwickler aus unterschiedlichen Bereichen, wie unter anderem Smart Homes, ist es, die Interaktion mit Anwendungen menschlicher zu gestalten. Aus diesem Grund wurde für diese Arbeit ein Anwendungsbeispiel eines Systems ausgesucht, welches den Vorgang des Kochens begleitet und den Nutzer mit möglichst natürlicher Interaktion durch den Zubereitungsprozess führt.

Eine solche Anwendung, die eine annähernd menschliche Interaktion bieten würde, erfordert die Nutzung vieler verschiedener Techniken wie Objekterkennung, Spracherkennung, Gestenerkennung etc. Die Frage, die sich hier stellt, ist: Sind wir in den verschiedenen Gebieten überhaupt schon so weit fortgeschritten, dass es möglich wäre eine solche Anwendung zu erstellen?

Eines der wichtigsten Elemente für eine möglichst menschenähnliche Interaktion ist die optische Wahrnehmung. In den letzten Jahren gab es zahlreiche bahnbrechende Entwicklungen im Bereich der Computer Vision, wie sie beispielsweise in [Delko2015] beschrieben werden. Dieser Fortschritt und die steigende Zugänglichkeit von Software, die die neusten Forschungsergebnisse im Bereich Computer Vision anwendet, lässt den Traum von Computern, die wie Menschen sehen können, scheinbar immer näher rücken. Doch auch ältere Methoden der Computer Vision könnten im Zusammenspiel mit den Neuerungen der heutigen Zeit ihren Beitrag zu einer menschlicheren Interaktion mit Computern leisten. Um eben jene menschenähnliche optische Wahrnehmung umsetzen zu können, ist das Erkennen natürlicher Objekte notwendig.

Das Erkennen natürlicher Objekte stellt eine weitaus größere Herausforderung dar als das Erkennen nicht natürlicher Objekte. Dies liegt daran, dass natürliche Objekte der gleichen Klasse nur selten ein komplett übereinstimmendes Aussehen haben. Während beispielsweise zwei Exemplare desselben Buches praktisch identisch sind, gibt es zwischen zwei Paprika Unterschiede in Form und Farbe. Ziel dieser Arbeit ist es verschiedene Verfahren, mit deren Hilfe sich eben dieses Erkennen von natürlichen Gegenständen bewerkstelligen lässt, auf ihre Eignung zu prüfen. Hierzu werden in Kapitel 2 einige bereits implementierte Verfahren vorgestellt, welche für das Erkennen von natürlichen Objekten verwendet werden können. Kapitel 2.4 behandelt einige Projekte, in welchen das Erkennen natürlicher Objekte eine tragende Rolle spielt. In Kapitel 3 werden dann einige Tests durchgeführt, um festzustellen, wie gut sich die entsprechenden Verfahren für den Zweck der Erkennung natürlicher Objekte eignen. Anschließend werden in Kapitel 4.1 noch einmal die Ergebnisse aller Tests im Zusammenhang ausgewertet. Kapitel 4.2 befasst sich mit der

Frage danach, welche Möglichkeiten der aktuelle technische Fortschritt der in dieser Arbeit behandelten Bereiche bietet.

2 Analyse

Die Aufgabe, ein Objekt in einem Bild erkennen und klassifizieren zu können, hat im Laufe der Jahre zahlreiche, immer besser werdende Methoden zu Tage gefördert, um eben dies zu bewerkstelligen. Von Prozessen, die durch das Analysieren eines Referenzbildes Objekte in einem zweiten Bild wiedererkennen können, bis hin zu komplexen Verfahren, welche anhand tausender Beispielbilder das Aussehen eines Objektes erlernen können. Es gibt eine breite Vielfalt an Möglichkeiten das Problem anzugehen. In dieser Arbeit wurde eine kleine Teilmenge dieser Techniken, welche vielversprechend schienen, herausgesucht. In diesem Kapitel werden eben diese Techniken aus dem Bereich der Computer Vision vorgestellt und erklärt. Für die Nutzung der Algorithmen in diesem Kapitel ist zwar kein tiefes Wissen bezüglich der einzelnen Methoden nötig, ein grundlegendes Verständnis der Abläufe und Besonderheiten ist jedoch von Vorteil. Aus diesem Grund wurde hier versucht, die Erklärung der Prozesse so nachvollziehbar wie möglich zu halten, um schnell ein grobes Verständnis der Vorgänge zu schaffen, ohne dass bereits spezifische Vorkenntnisse im Bereich der Computer Vision vorhanden sein müssen. Ziel hiervon ist es, einen schnellen und einfachen Einstieg in das Thema zu ermöglichen.

2.1 Objekterkennung mit Haar-like Features

Bei diesem Verfahren, welches in dem 2001 erschienenen Paper mit dem Titel „Rapid Object Detection using a Boosted Cascade of Simple Features“ [Viola et al. 2001] von Paul Viola und Michael Jones erstmalig vorgestellt wurde, handelt es sich um ein auf maschinelles Lernen

gestütztes Verfahren zur Objekterkennung, welches sehr schnelle Ergebnisse bei hoher Erkennungsrate verspricht. Das Verfahren lässt sich in drei Bestandteile aufteilen:

- Die Einführung einer auf dem Konzept der „Summed Area Tables“ [Crow1984] basierende Art der Bilddarstellung mit dem Namen „Integral Images“, die die Berechnung der Features, welche der Detektor verwendet, drastisch beschleunigt.

- Ein auf AdaBoost [Freund et al. 1995] basierender Lernalgorithmus, welcher die Features eines größeren Sets auf wenige kritische Features reduziert, was zu sehr effektiven Classifiern führt.

- Eine Methode zum Zusammenführen von steigend komplexer werdenden Classifiern zu einer „Cascade“, die es erleichtert, schneller uninteressante Bereiche des untersuchten Bildes, die mit hoher Wahrscheinlichkeit keine der geforderten Features enthalten – wie zum Beispiel den Hintergrund – auszuschließen.

2.1.1 Integral Images

Diese Art der Bilddarstellung beschleunigt das Suchen nach Features im Bild. Für die Erstellung eines Integral Bildes mit den Dimensionen x und y wird aus dem Originalbild ein neues Integralbild mit denselben Dimensionen erstellt. Die Werte für die Pixel des Integralbildes ergeben sich hierbei durch die Addition aller Werte der Pixel im Originalbild. Hierzu werden alle Pixelwerte des Originalbildes in einem rechteckigen Bereich von der oberen linken Ecke des Originalbildes bis hin zur Position des zu berechnenden Pixels aufaddiert. Der Prozess ist in Abbildung 1 noch einmal veranschaulicht. Diese Art der Darstellung erleichtert es enorm aufsummierte Werte aus rechteckigen Bereichen abzulesen, was bei der Verwendung der von Haar-like Features zwingend notwendig ist.

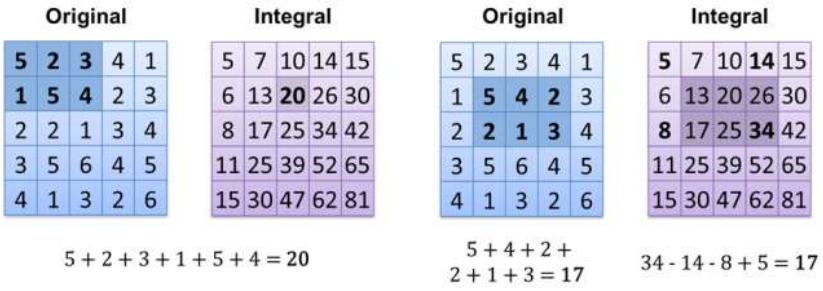

Abbildung 1 Darstellung der einfachen Berechnung des Inhaltes von Bildbereichen unter Nutzung eines Integralbildes. [Dedhia2015]

Bei der Objekterkennung mit Haar-basierenden Classifiern liegen die Punkte, an denen ein Objekt wiedererkannt werden soll, in Form von rechteckigen Features vor. In [Viola et al. 2001] wurden drei verschiedene Arten von Features verwendet:

- Das „Two-Rectangle Feature“ – auch „Edge Feature“ genannt –, dessen Wert sich aus der Summe der Pixel in zwei rechteckigen Bereichen des Bildes berechnet,

- Das „Three-Rectangle Feature“ – auch „Line Feature“ genannt –, dessen Wert sich aus der Summe der beiden äußeren Rechtecke subtrahiert von dem Wert des mittleren Rechtecks ergibt,

- Das „Four-Rectangle Feature“, dessen Wert sich aus dem Unterschied der jeweiligen diagonalen Paare berechnet.

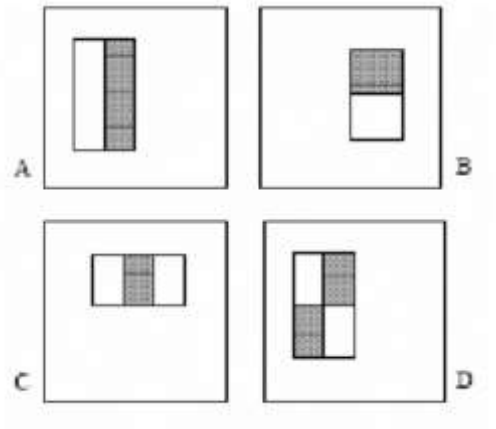

Abbildung 2 Die ursprünglichen in [Viola et al. 2001] vorgestellten Rectangle Features.
A: horizontales Two-Rectangle Feature B: vertikales Two-Rectangle Feature
C: Three-Rectangle Feature D: Four-Rectangle Feature. [Viola et al. 2001] S.512

2.1.2 Lernverfahren

Die Anzahl an Features in einem Bild übersteigt selbst die Zahl der Pixel. Aus diesem Grund wird durch das hier beschriebene Lernverfahren eine Teilmenge an guten Features herausgesucht und der Classifier anhand dieser Teilmenge trainiert. Eigentlich kann jedes maschinelle Lernverfahren verwendet werden. Im Falle von Viola und James wurde aber auf das „Adaptive Boosting“ Verfahren gesetzt. Für dieses Lernverfahren werden positive Bilder, in denen sich das Objekt befindet, das erlernt werden soll, und negative Bilder, in denen sich eben jenes Objekt nicht befindet, überprüft. Da die Qualität des Classifiers stark von der Brauchbarkeit der Bilder abhängt, welche für das Training verwendet werden, sollte auch hier darauf geachtet werden, dass das Objekt auf den Bildern nicht zu stark verfremdet ist. Auch die Anzahl der Positiv- und Negativbilder trägt besonders zur Qualität der Classifier bei. Im Falle von Viola und James wurden 9.832 Positiv- und 10.000 Negativbilder verwendet.
Im Lernvorgang wird zum Überprüfen das zu testende Bild in Subfenster aufgeteilt, die dann mit den Classifiern, welche zu diesem Zeitpunkt aus nur einem Feature bestehen, auf das zu erkennende Objekte untersucht werden. Anschließend wird für jedes Feature die optimale

Schwellenfunktion berechnet, welche benötigt wird, um die positiven von den negativen Bildern zu unterscheiden. Aus allen Features werden nur die mit der geringsten Fehlerrate herausgesucht.

2.1.3 Kombination von Classifiern zu einer Kaskade

Um die Geschwindigkeit des Erkennungsprozesses zu steigern und die Erkennungsrate zu verbessern, werden nun mehrere Classifier zu einer Kaskade zusammengefügt. Eine Kaskade ist hier so aufgebaut, dass mehrere immer komplexer werdende Classifier hintereinander gesetzt werden. Sollte der erste Classifier in der Kaskade den getesteten Bildausschnitt als positiv bewerten, wird derselbe Bildausschnitt noch einmal mit dem darauffolgenden, komplexeren Classifier getestet. Wenn innerhalb der Überprüfung mit der Kaskade auch nur ein Classifier den getesteten Bildausschnitt für negativ befindet, wird der Bildausschnitt vollständig für negativ befunden und verworfen. Da der erste Classifier in der Kaskade sehr simple ist und dennoch eine sehr niedrige Falsch-positiv-Rate hat, ist diese Art des Testens mit einer Kaskade sehr effektiv darin, schnell Bildausschnitte, in denen sich nicht das gesuchte Objekt befindet, auszusortieren. Diese Kaskade wird dann trainiert, indem zu jeder Stufe der Kaskade Features hinzugefügt werden, bis die gewünschte Erkennungsrate und Falsch-positiv-Rate erreicht sind.

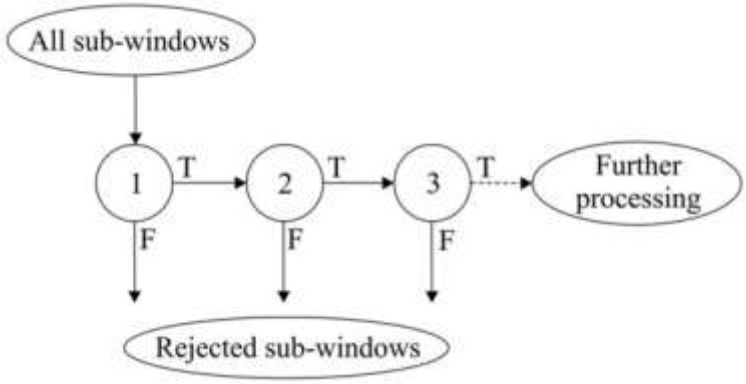

Abbildung 3 Verdeutlichung des Erkennungsprozesses mit Hilfe der Kaskade. Ein False reicht, um den Erkennungsprozess abzubrechen. [Viola et al. 2001] S. 515

2.2 Featurematching

Um Objekte erfolgreich in Bildern erkennen und sogar Orten zu können, nutzen Algorithmen, wie auch wir Menschen, die Eigenschaften der Objekte. Der Mensch hat beispielsweise im Laufe seines Lebens gelernt, dass eine Tomate rot und rundlich ist und immer einen Strunk besitzt. Diese Eigenschaften bezeichnet man im Bereich der Computer Vision als sogenannte Features – auch Keypoints genannt. Wie auch wir, kann ein Computer diese Features erlernen. Die im Folgenden beschriebenen vier Verfahren sind jeweils immer in zwei Phasen aufgeteilt. Jeder dieser Prozesse sucht in Bildern nach diesen Features und beschreibt diese so, dass sie mit anderen gefundenen Features verglichen werden können. Bei dem Finden von Features wird hierbei von Feature Detection und bei dem Beschreiben derselbigen von Feature Description gesprochen.

2.2.1 SIFT

SIFT – kurz für „Scale Invariant Feature Transform“ – ist ein Algorithmus zum Finden von Keypoints und Generieren von Deskriptoren aus eben jenen Keypoints. Vorgestellt wurde SIFT von David Lowe in seinen Papers „Object recognition from local scale-invariant features“ [Lowe1999] und „Distinctive Image Features from Scale-Invariant Keypoints“ [Lowe2004]. Die große Stärke des SIFT-Verfahrens im Gegensatz zu vielen vorhergegangenen Algorithmen zum Finden von Keypoints bzw. Features ist die Tatsache, dass SIFT – wie der Name schon sagt – Keypoints heraussucht, welche invariant gegenüber Skalierung sind. Das bedeutet, dass es für SIFT bei der Erkennung bis zu einem gewissen Grad egal ist, wie weit das Objekt von der Kamera entfernt ist. Das SIFT-Verfahren basiert auf vier grundlegenden Konzepten, die im Folgenden erklärt werden sollen.

2.2.1.1. Scale-space Extrema Detection

Hierbei wird nach Features gesucht, welche unabhängig von der Skalierung des Bildes sind. Das Ursprungsbild wird hierfür in verschiedene Stufen kopiert und um den Faktor 2 verkleinert. Jedes der so entstandenen Bilder I(x, y) wird dann mit einem Gaußfilter G(x, y, σ) mit von Stufe zu Stufe steigenden σ geglättet, sodass für jede Größenstufe des Bildes eine Reihe von steigend geglätteten Bildern entsteht. Anschließend werden alle Bilder mit der jeweiligen Größenstufe, welche adjazent zueinander sind, über die Berechnung des Unterschiedes zu einem Difference of Gaussians-Bild (kurz: DoG-Bild) zusammengeführt. Beim DoG-Bild handelt es sich um eine Darstellungsform, in welcher sich Bereiche stark abzeichnen, in denen eine starke Änderung der Intensität vorzufinden ist, wie zum Beispiel ein schwarzer Punkt auf hellem Grund. Da bei steigendem σ – also steigender Unschärfe der Bilder – immer mehr kleinere Details des Bildes verschwinden und somit nur noch große Bereiche mit starkem Intensitätswechsel vorhanden sind, kann σ hier als Größenfaktor für potentielle Features gesehen werden. Umso größer σ, umso größer wird auch das gefundene Extrem bzw. potentielle Feature sein. Die so entstandenen DoG-Bilder der verschiedenen Größen und Weichzeichnungsstufen bilden zusammen eine DoG-Pyramide. Bei dieser Pyramide wird bei einer Größenstufe von einer „Octave“ gesprochen. Bei den einzelnen Bildern innerhalb einer Octave wird bei steigendem gaußschen Weichzeichner von steigender „Scale“ geredet.

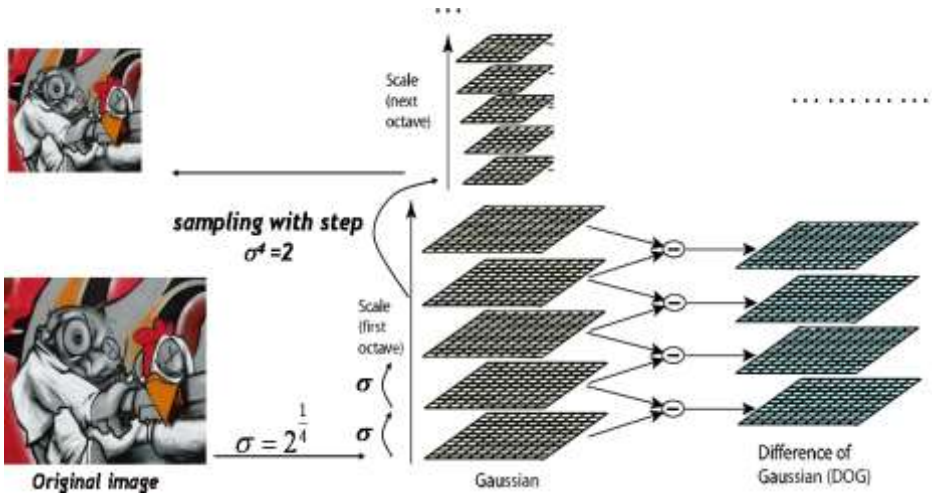

Abbildung 4 Berechnung einer DoG-Pyramide. Jede Größenstufe des Bildes ist hier eine Octave und die verschiedenen Glättungsstufen stellen die Scale dazu. [Tuytelaars et al. 2008] S.248

Die DoG-Pyramide wird nun auf potentielle Keypoints durchsucht. Dies geschieht, indem jedes Pixel mit den fünf Pixeln, die es umgeben, und den jeweils neun Pixeln aus dem über und unter ihm liegenden DoG-Bild verglichen wird. Wenn es sich bei dem überprüften Pixel um ein lokales Extrem handelt, wird es in eine Liste von potentiellen Keypoints eingetragen. Das Ergebnis dieses Suchvorganges ist eine Liste aus Tupeln, welche jeweils die X-Position, die Y-Position und den Scale-Faktor σ des Keypoints angeben (x,y, σ).

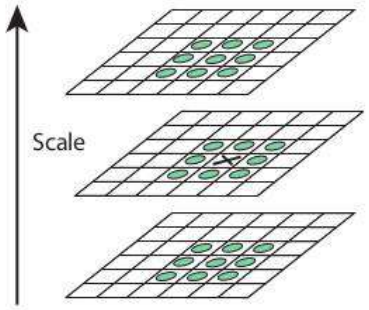

Abbildung 5 Überprüfung eines potentiellen Keypoints. Hier werden sowohl die den Punkt umgebenden Pixel, als auch die 3 mal 3 Pixel in der Scale über und unter dem Punkt überprüft. [Lowe2004] S.95

2.2.1.2. Keypoint Lokalisierung

Durch Nutzung der Taylor Series Expansion of Scale Space wird in diesem Schritt die genauere Position der jeweiligen Extrema herausgefunden. Hierzu wird vereinfacht gesprochen eine quadratische Fläche über die benachbarten Werte gelegt, um die interpolierte Position des Maximums zu finden. So wird die genauere Position im Bereich zwischen den Pixeln festgestellt. An diesem neu bestimmten Punkt wird anschließend durch den Vergleich mit den benachbarten Punkten der Threshold bzw. Schwellenwert gemessen. Der Threshold bezieht sich hier auf den Unterschied zu angrenzenden Pixeln bezüglich der Intensität. Sollte der gemessene Threshold unter einem gewissen Wert liegen, wird der Keypoint ignoriert, da dieser ein sehr schwach ausgeprägter Keypoint wäre und eventuell durch Bildrauschen verursacht worden sein könnte.

Beim Finden der potentiellen Keypoints wurde das „Difference of Gaussian“-Verfahren verwendet. Dies ist zwar schnell und performant, führt aber zu starken Ausschlägen entlang der Kanten im Bild. Keypoints, welche auf diesen Kanten liegen, sind denkbar schlecht wiederzuerkennen, da entlang der Kante viele gleich aussehende Keypoints liegen. Dies führt dazu, dass beliebige Teile der Kante als immer gleicher Keypoint wiedererkannt werden würden. Aus diesem Grund verwendet Lowe einen an das Verfahren von Harris und Stephens [Harris et al. 1988] angelehnten Prozess, um die Kanten herauszufiltern. Hierbei wird basierend auf einer 2x2 Hesse-Matrix das Verhältnis der Eigenwerte zueinander berechnet. Sollte der hierbei errechnete Wert einen bestimmten Schwellwert übersteigen (im Paper von Lowe [Lowe2004] entspricht dieser Wert 10), so handelt es sich um einen Keypoint auf einer Kante, welcher folglich ignoriert wird.

2.2.1.3. Bestimmung der Orientierung

Für jeden der noch verbleibenden Keypoints wird in diesem Schritt eine Ausrichtung bestimmt. Dies schafft eine gewisse Invarianz in Bezug auf die Rotation des Keypoints und

hilft somit dabei, in einer rotierten Version des Objektes weiterhin die entsprechenden Keypoints zu erkennen. Für diese Berechnung der Ausrichtung eines Keypoints wird eine von der Scale des Punktes abhängige Region um den Punkt ausgewählt, in der der Gradient jedes Pixels bestimmt wird. Der Gradient der einzelnen Pixel ist hierbei von der Intensitätsänderung, welche die Gewichtung des Gradienten bestimmt, und der Richtung, in welcher die größte Intensitätsänderung in Relation zum geprüften Pixel stattfindet, abhängig.

Aus den so erhaltenen Gradienten wird schließlich ein Orientierungs-Histogramm erstellt. Dieses Histogramm unterteilt die 360° der möglichen Richtungen in 36 Klassen. Der Wert dieser Klassen wird von der Summe der Gewichtungen der Gradienten, welche in den entsprechenden Gradbereich fallen, bestimmt. Der höchste Wert in diesem Histogramm bestimmt die Ausrichtung des Keypoints. Sollte es andere Klassen im Histogramm geben, die einen Wert von 80% oder mehr des höchsten Wertes haben, wird für diese Werte jeweils ein separater Keypoint mit der gleichen Position und Scale erstellt, welcher jedoch eine andere Ausrichtung hat. Dies steigert die Stabilität des Matchings enorm, trifft jedoch laut Lowe nur für ca. 15% aller Keypoints zu [Lowe2004].

2.2.1.4. Erstellen der Deskriptoren

In diesem Schritt werden aus den durch die vorherigen Schritte erhaltenen Keypoints sogenannte „Keypoint-Deskriptoren“ erstellt. Diese Keypoint-Deskriptoren sollen möglichst einzigartig und unabhängig von Blickwinkel und Beleuchtung sein.

Zu diesem Zweck wird über jeden Keypoint ein 16 mal 16 Pixel großes Quadrat gelegt, in dessen Mitte sich der Keypoint befindet. Für die 16 mal 16 Pixel wird jeweils der Gradient berechnet. Dieses Quadrat wird dann in 4 mal 4 Regionen unterteilt. Für jeden der aus der Aufteilung resultierenden 16 Bereiche wird ein aus 8 Klassen bestehendes Histogramm erstellt, in dem 360° auf die 8 Klassen des Histogramms abgebildet werden. Auch hier wird

der Wert der einzelnen Klassen durch die Gewichtung und die Richtung der einzelnen Gradienten im jeweiligen Bereich bestimmt. Anders als bei der Bestimmung der Ausrichtung der Keypoints, hat hier aber auch die Entfernung des Gradienten zum Keypoint Auswirkungen auf den Einfluss auf die zugehörige Klasse des Histogramms. Umso weiter ein Gradient vom Keypoint in der Mitte entfernt ist, umso geringer wird der Wert, der der entsprechenden Klasse im Histogramm hinzugefügt wird. Dies wird über die Anwendung einer gaußschen Gewichtungsfunktion bewerkstelligt. Nach der Erstellung der Histogramme erhält man also 4 mal 4 Quadranten mit jeweils 8 Werten. Daraus ergibt sich ein aus 128 Werten bestehender „Feature-Vector“. Um den Feature-Vector schließlich noch unempfindlicher gegen Änderungen in der Beleuchtung zu machen, wird er normalisiert. Dies führt laut Lowe [Lowe2004] dazu, dass eine Helligkeitsänderung, welche jedes Pixel im Bild betrifft, nicht die Werte der Gradienten beeinflusst, da diese durch Pixelunterschiede berechnet werden.

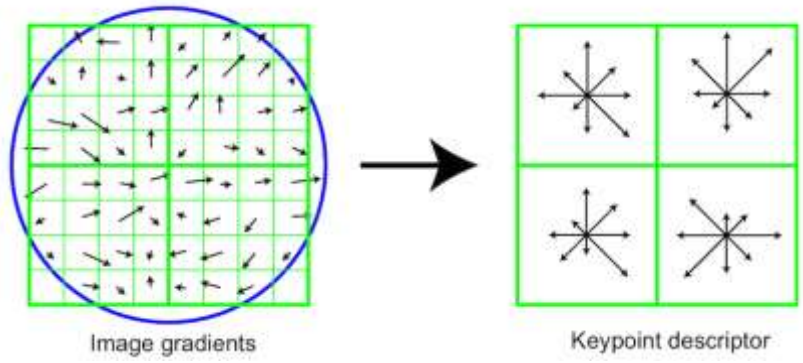

Abbildung 6 Erstellung des Keypoint-Deskriptors anhand von 8 mal 8 Probewerten, welche der Umgebung des Keypoints entnommen werden. Links: Die 8 mal 8 Probewerte, welche abhängig von ihrer Entfernung zum Keypoint gewertet werden. Rechts: Der daraus entstehende Deskriptor. Dieses Beispiel ist vereinfacht. Bei den Experimenten in [Lowe2004] wurden 16 mal 16 Probewerte genommen, was zu einem 4 mal 4 Deskriptor führt. [Lowe2004] S.101

2.2.1.5. Matching

Für das Matching zwischen zwei Bildern werden in beiden Bildern zu allen gefundenen Features Deskriptoren erstellt. So können anschließend alle Deskriptoren aus dem ersten Bild mit allen Deskriptoren aus dem zweiten Bild verglichen werden. Um die Zahl fehlerhafter Matches zu reduzieren, werden nicht eindeutige Matches fallen gelassen. Dies geschieht folgendermaßen: Sollte es für einen Deskriptor aus dem ersten Bild mehrere gute Matches mit Deskriptoren des zweiten Bildes geben, wird die Differenz der Übereinstimmung des erstbesten Matches und die Übereinstimmung des zweitbesten Matches bestimmt. Sollte diese Differenz sehr niedrig sein – die Matches sind also nahezu identisch in ihrer Übereinstimmung – wird für den Deskriptor des ersten Bildes kein Match gewählt. Sollte die Differenz jedoch größer sein, wird das beste Match gewählt. Dieses Verfahren des Aussortierens hilft laut Lowe [Lowe2004] drastisch fehlerhafte Matches zu verringern, während nur ein kleiner Teil der korrekten Matches dadurch verloren geht.

2.2.2 SURF

Bei dem SURF Algorithmus – SURF steht für „Speeded-Up Robust Features“ – handelt es sich um einen weiteren Algorithmus zum Finden von Keypoints in Bildern und dem Erstellen von Deskriptoren. SURF wurde das erste Mal auf der „European Conference on Computer Vision“ in Graz von Herbert Bay, Tinne Tuytelaars und Luc Van Gool vorgestellt. Das Paper [Bay2008], auf welchem der folgende Abschnitt teilweise basiert, wurde dann im Jahre 2008 veröffentlicht. Wie der Name schon verrät, handelt es sich hierbei um eine schnellere Alternative zu SIFT. Dieser Algorithmus stützt sich zu großen Teilen auf den SIFT Algorithmus. Aus diesem Grund wir in diesem Kapitel an einigen Stellen von einer genaueren Erklärung abgesehen und auf das Kapitel zur Erklärung von SIFT verwiesen. Auch der SURF Algorithmus lässt sich in die zwei grundlegenden Schritte des Findens der Interest-Points und dem Erstellen entsprechender Deskriptoren zerlegen.

2.2.2.1. Finden von Interest-Points

Wie schon bei der Objekterkennung mit Haar-like Features, werden auch beim Surf Algorithmus die sogenannten Integralbilder verwendet, um den Prozess zu beschleunigen. Für eine genauere Erläuterung zu Integralbildern wird an dieser Stelle auf Kapitel 2.1.1 verwiesen.

Der SURF Algorithmus verwendet zum Finden von Interest-Points – also potentiellen Keypoints – einen auf der Hesse Matrix basierenden Blob Detector.
Das Bild wird mit Hilfe einer Hesse Matrix in Kombination mit Box-Filtern in verschiedenen Skalierungen untersucht. SURF vergrößert hierzu – im Gegensatz zu SIFT – die Größe der Boxfilter und wendet sie auf das Bild an, statt das Bild wiederholt mit einem Gaußfilter zu bearbeiten und zu verkleinern (siehe Kapitel 2.2.1.1). Ebenfalls wie bei SIFT wird das Bild auch hier in Oktaven und Ebenen unterteilt. Diese Aufteilung wird in [Huber et al. 2013] wie folgt beschrieben:

„Die unterschiedlichen Größen der angewendeten Filter werden Oktaven zugeordnet, wobei eine Oktave einem Skalierungsfaktor von etwa zwei entspricht. Die Filter innerhalb einer Oktave werden als Ebenen bezeichnet […]. Mit drei Oktaven, die aus je vier Ebenen bestehen, werden nahezu alle wichtigen Punkte erkannt.“ [Huber et al. 2013] S.4

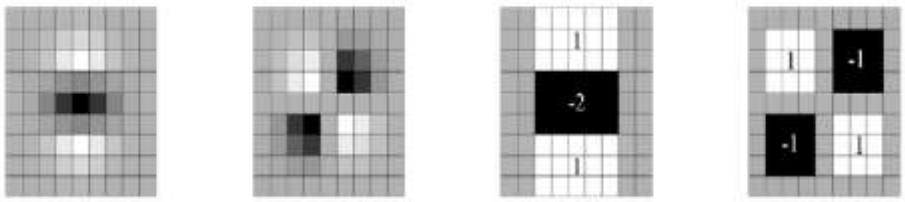

Abbildung 7 Links: Zwei Features wie sie in SIFT durch Verwendung von DoG auftreten. Rechts: Abschätzung durch Verwendung von Box-Filtern. [Bay et al. 2008] S. 348

Dieses Prinzip wird noch einmal in Abbildung 8 veranschaulicht. Das Verwenden von Box-Filtern ist hier durch die Kombination mit dem Integralbild äußerst performant, da die Werte für die Bereiche des Box-Filters sich so mit gleichbleibendem, minimalem Aufwand berechnen lassen. Bei dieser Überprüfung auf verschiedenen Skalierungsebenen entsteht eine Liste von approximierten Determinanten für alle Skalierungen, welche in einer so genannten Blob Response Map gespeichert wird. Diese Blob Response Map, wie sie in [Bay et al. 2008] genannt wird, wird anschließend auf lokale Maxima und Minima durchsucht. Um Skalierungsvarianz gewährleisten zu können, werden wie auch schon beim SIFT Algorithmus (siehe Abbildung 5), die benachbarten Punkte des potentiellen Keypoint in der darüberliegenden und darunterliegenden Skalierungsebene untersucht.

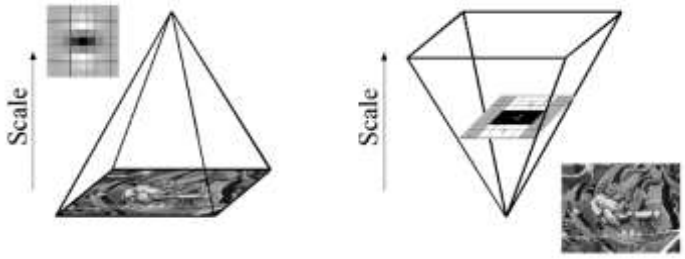

Abbildung 8 Links: Bei SIFT wird für Skalierungsinvarianz die Größe des Bildes verringert. Rechts: Bei SURF muss, um den gleichen Effekt zu erhalten, lediglich der Filter vergrößert werden. [Bay et al. 2008] S.349

2.2.2.2. Erstellen von Deskriptoren

Nachdem die Interest-Points gefunden wurden, wird nun ein Deskriptor erstellt, um jeden Featurepoint einzigartig und somit wiedererkennbar zu machen. Um dies zu bewerkstelligen, wird für jeden Interest-Point die zugehörige Orientierung bestimmt. Hierzu werden im Umkreis jedes Interest-Points Haar Wavelet Responses berechnet. Die Größe der Haar Wavelet Responses ist hierbei abhängig von der Skalierungsstufe, in welcher der Interest-Point sich befindet. Nun werden alle Responses in einem Fenster von 60°, welches

schrittweise um den Interest-Point rotiert wird, aufaddiert. Da es sich bei den Responses sowohl um horizontale, als auch vertikale Responses handelt, entstehen dadurch mehrere Orientierungsvektoren. Aus diesen Vektoren wird schließlich der größte ausgewählt, um die dominante Orientierung des Interest-Points bzw. Features zu beschreiben.

Anschließend wird über den Interest-Point ein an der im vorherigen Schritt errechneten Orientierung ausgerichtetes quadratisches Feld, welches in 4 mal 4 Subregionen unterteilt wird, gelegt. Aus jeder dieser 4 mal 4 Subregionen werden nun 5 mal 5 Haar Wavelet Responses entnommen. Diese Wavelet Responses werden bei steigender Entfernung zum Interest-Point schwächer gewertet. Für jedes der 4 mal 4 Gebiete wird nun ein Vektor erstellt, welcher sich aus der Summe der horizontalen Wavelet Responses, der Summe der vertikalen Wavelet Responses und der Summe der Absolutwerte sowohl der horizontalen als auch der vertikalen Wavelet Responses zusammensetzt ($V = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$). Der Deskriptor ist dann schlussendlich ein Vektor, welcher sich wiederum aus den 16 so erhaltenen Vektoren zusammensetzt. Damit hat der Vektor eine Länge von 64. Das Matching kann bei SURF genau wie das Matching bei SIFT durchgeführt werden.

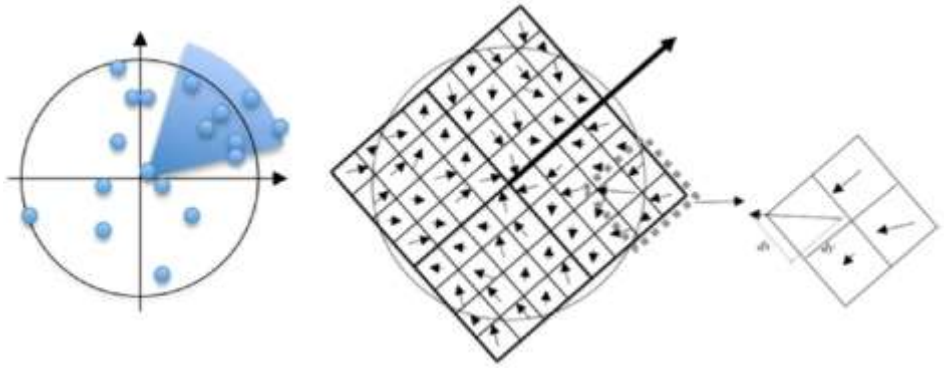

Abbildung 9 Rechts: Bestimmung der globalen Orientierung des Features. Links: Erstellen des Feature-Vectors. Zur Vereinfachung wurde hier jedes der 4x4 Quadrate nur in 2x2 Bereiche für die Wavelet Responses unterteilt. [Krig2014] S.255

2.2.3 ORB

Oriented FAST and rotated BRIEF wurde 2011 von Rublee, Rabaud, Konolige und Bradski in ihrem Paper mit dem Titel „ORB: An efficient alternative to SIFT or SURF“ [Rublee et al. 2011] vorgestellt. Es handelt sich, wie der Titel des Papers vermuten lässt, um eine effiziente Alternative zu SIFT und SURF. ORB ist unter anderem aus dem Grund sehr beliebt, dass SIFT und SURF patentiert sind und bei der kommerziellen Nutzung der beiden im Gegensatz zu ORB Kosten anfallen. ORB basiert auf einer modifizierten Version des FAST-Detektors und nutzt zur Beschreibung eine ebenfalls modifizierte Version des BRIEF-Deskriptors. An dieser Stelle wird von einer ausführlichen Erklärung zu FAST und BRIEF abgesehen. Stattdessen wird hier der Fokus auf die Besonderheiten von ORB gelegt. Falls ein tieferes Verständnis für FAST und BRIEF gewünscht ist, werden an dieser Stelle die Papers „Machine learning for high-speed corner detection“ [Rosten et al. 2006], „Fusing points and lines for high performance tracking.“ [Rosten et al. 2005] und „Brief: Binary robust independent elementary features“ [Calonder et al. 2010] empfohlen.

2.2.3.1. Finden von Featurepoints

Zum Finden von Keypoints wird hier der FAST-Algorithmus verwendet. Da FAST aber leider nur Ecken im Bild findet und einige Schwächen besitzt, welche zur Nutzung in ORB ausgeglichen werden müssen, wurden hier einige Änderungen vorgenommen. FAST findet im Gegensatz zu SIFT und SURF keine Features in verschiedenen Skalierungsebenen. Um dies dennoch zu bewerkstelligen, wird bei ORB eine Pyramide bestehend aus immer kleiner werdenden Versionen des Ausgangsbildes erstellt, in welchen jeweils FAST-Features gesucht werden. Laut [Rublee et al. 2011] zeigte es sich, dass viele der vom FAST-Detektor gefundenen Punkte auf einer Linie lagen. Da dies, wie schon bereits in Kapitel 2.2.1.2 beschrieben, eher hinderlich für ein einzigartiges Feature ist, wurde bei ORB ein „harris corner measure“ genutzt, um die guten Eckpunkte herauszufinden. Für jeden der nach dieser Prozedur verbleibenden Eckpunkte wird anschließend eine Orientierung bestimmt. Da auch

eine Orientierung des Features nicht in FAST vorgesehen ist, musste auch hier eine Anpassung stattfinden. Im Umkreis der gefundenen Eckpunkte wird jeweils anhand der Intensität ein Mittelpunkt (intensity-weighted centroid) bestimmt. Die Orientierung des Featurepunktes richtet sich vom Featurepunkt selbst hin zum intensitätsabhängigen Mittelpunkt.

2.2.3.2. Erstellen von Deskriptoren

Als Deskriptor verwendet ORB eine modifizierte Version des BRIEF-Deskriptors: das auch „rBRIEF“ genannte „Rotation-Aware BRIEF“. Um den folgenden Absatz zu verstehen, reicht es zu wissen, dass ein BRIEF-Deskriptor erstellt wird, indem mehrere Paare von Bereichen um den Interest-Point ausgewählt werden, welche dann miteinander verglichen werden. Diese Paare werden nach [Calonder et al. 2010] Binary Tests genannt. Das Problem bei BRIEF-Deskriptoren ist, dass sie abhängig von Rotation sind. Dadurch würde ein Feature nur in exakt derselben Orientierung wiedererkannt werden. Da aber auch rotierte Objekte beziehungsweise deren Features wiedererkannt werden sollten, wird die dominante Orientierung der individuellen Features genutzt, um ein rotationsunabhängigen Deskriptor zu schaffen. Hierzu wird eine Matrix erstellt, in welcher sich die Lokation der „Binary Tests“ befindet, aus denen sich ein BRIEF-Deskriptor bildet. Anschließend wird mit Hilfe der dominanten Orientierung des Features eine Rotationsmatrix erstellt, welche die Pixelpaare enthaltende Matrix rotiert. So wird unabhängig von der Rotation eines Interest-Points immer der gleiche Deskriptor erstellt. Leider geht laut [Rublee et al. 2011] bei dieser Rotation viel Varianz verloren. Um dem entgegenzuwirken und „Binary Tests“ mit hoher Varianz und niedrigem Zusammenhang zu finden, wird bei ORB ein Greedy-Search-Algorithmus nach den besten Paaren verwendet.

2.2.3.3. Matching

Der Vorgang des Matchings kann – wie auch schon bei SIFT und SURF – via Brute-Force (d.h. alle Features aus Bild 1 werden mit allen Features aus Bild 2 verglichen) geschehen. Der einzige Unterschied zum Matching mit SURF und SIFT Deskriptoren liegt in der Art und Weise, wie die Deskriptoren verglichen werden. Dadurch, dass der rBRIEF-Deskriptor ein binärer Deskriptor ist, der aus einem kurzen, binären String besteht, kann das LSH zum Vergleich zweier Deskriptoren verwendet werden. LSH steht dabei für „Locality Sensitive Hashing“. Hierbei handelt es sich kurz gesagt um eine Hashfunktion bei der ähnliche Strings denselben Hash bekommen [Gionis et al. 1999]. Im Falle von ORB wurde eine erweiterte Version des LSH, das sogenannte „Multi-probe LSH“, wie es in [Lv et al. 2007] vorgestellt wird, verwendet, da diese laut [Rublee et al. 2011] eine geringere Anzahl an Hashtabellen benötigt und somit weniger fordernd ist, was den Arbeitsspeicher angeht.

2.2.4 AKAZE

Das AKAZE-Verfahren [Alcantarilla2013], kurz für „Accelerated KAZE“, ist eine weiterentwickelte Form des KAZE-Verfahren [Alcantarilla2012]. Der Hauptunterschied zwischen KAZE und bisherigen Algorithmen ist, dass bei der Erstellung des Scale Spaces (siehe Abbildung 4) ein gaußscher Weichzeichner verwendet wird. Der gaußsche Weichzeichner sorgt aber nicht nur für eine Verminderung von Bildrauschen im Bild, sondern auch für den Verlust eigentlich wichtiger Details wie die Objektgrenzen bzw. Konturen. Dies wird noch einmal in Abbildung 10 veranschaulicht. KAZE verzichtet aus diesem Grund auf die Nutzung von einem gaußschen Weichzeichner und nutzt für die Erstellung des Scale Spaces sogenanntes „Nonlinear Diffusion Filtering“, welches durch „Adaptive Operator Splitting Schemes“ [Weickert et al. 1998] bewerkstelligt wird. Kurz gesagt ist es damit möglich, die kleinen bzw. unwichtigen Details von Bildern wie zum Beispiel Bildrauschen zu entfernen, ohne die Umrisse der im Bild befindlichen Objekte zu beschädigen.

Abbildung 10 Oben: Gaußscher Scale Space (linear). Unten: Nonlinear Diffusion Scale Space. [Alcantarilla et al. 2012] S.220

Leider sind Adaptive Operator Splitting Schemes recht komplex zu berechnen und somit auch sehr rechenaufwändig was CPU oder GPU angeht. Aus diesem Grund nutzt AKAZE zum Erstellen des „Nonlinear Diffusion Scale Spaces“ das 2010 vorgestellte Verfahren der „Fast Explicit Diffusion“ [Grewenig et al. 2010].

2.2.4.1. Finden von Features

Um das Finden von Features zu ermöglichen, wird von jedem der in der Scale Space Pyramide erstellten Bilder eine Hesse-Matrix errechnet. Anschließend wird für jedes Bild überprüft, ob die Detector Response höher ist als ein vordefinierter Threshold. Sollte dem so sein, wird noch getestet, ob die direkt angrenzenden Pixel nicht eine höhere Response liefern. Anschließend wird noch überprüft, ob der gefundene potentielle Featurepunkt auch ein Extremum in Bezug auf die Scale Level über und unter ihm ist. Anschließend wird, wie auch schon bei SIFT mit Genauigkeit im Subpixelraum berechnet, wo sich das Extremum befindet. Dies geschieht hier ebenfalls dadurch, dass eine zweidimensionale quadratische Funktion über die 3 mal 3 Pixel der Hesse-Matrix gelegt wird, die das Extremum umgeben.

2.2.4.2. Erstellen von Feature Deskription

Für die Beschreibung der Features wird bei AKAZE eine modifizierte Version von „Local Difference Binary“ [Yang et al. 2012] mit dem Namen „Modified-Local Difference Binary“, kurz M-LDB, genutzt. Diese bezieht Informationen über Intensität und Gradienten aus dem zuvor erstellten, nicht linearen Scale Space. LDB funktioniert sehr ähnlich wie BRIEF, nur dass die Binary Tests hier nicht zwischen einzelnen Pixeln stattfinden, sondern zwischen Durchschnittswerten ganzer Bereiche. Ein weiterer Unterschied ist, dass hier nicht nur die Intensität, sondern auch der Mittelwert der horizontalen und vertikalen Ableitung verglichen werden. Es werden also pro Binary Test drei Bits erstellt. Um im Gegensatz zu LDB invariant gegenüber Rotation zu sein, wird bei M-LDB die Orientierung des Featurepoints bestimmt und das Raster, an welchem sich die Binary Tests ausrichten, entsprechend dieser Orientierung rotiert. Die Bestimmung der Orientierung geschieht ähnlich wie bei SURF. Der Unterschied liegt hier darin, dass an Stelle von Wavelet Responses die zweite Ableitung in X und Y-Richtung genutzt wird, um die Orientierung des Featurepunktes zu berechnen. Die Größe der Bereiche, welche für die Binary Tests genutzt werden, ist abhängig von der Scale. Dies macht den Deskriptor laut [Alcantarilla et al. 2013] robuster gegen Änderungen in der Größe. Das Matching kann hierbei wie das Matching bei ORB durchgeführt werden.

2.3 Convolutional Neural Networks

Bei „Convolutional Neural Networks“ handelt es sich um eine Unterart der neuralen Netzwerke, welche sich in den letzten Jahren besonders im Bereich der Objekterkennung bewährt hat.

Wie auch schon bei den meisten der bereits in dieser Arbeit vorgestellten Vorgängen, arbeiten CNN ebenfalls mit Features. Im Gegensatz zu SIFT, SURF etc. werden hier aber

ähnlich wie bei Haar-like Features die Features durch einen großen Satz an Beispielbildern erlernt. Für den Lernvorgang muss eine entsprechende Menge an Positivbildern vorhanden sein. Basierend auf diesen Positivbildern erlernt das System passende Features für das zu lernende Objekt. Das Lernen von Features erfolgt hierbei quasi rückwärts. Zuerst weiß das System lediglich, dass sich auf dem Foto aus der Testgruppe das zu lernende Objekt, wie zum Beispiel ein Mensch, befindet. Nach und nach werden dann unterliegende Features des Menschen gelernt. So lernt das System beispielsweise, dass ein Mensch immer einen Kopf und dieser in der Regel auch zwei Augen hat. Ein großer Vorteil gegenüber den anderen in diesem Paper vorgestellten Vorgängen ist, dass bei der Objekterkennung mit Convolutional Neural Networks neben Graustufenbildern auch Farbbilder untersucht werden können. Dies ist beim Unterscheiden zweier Objekte, deren einziger Unterschied in ihrer Färbung liegt, äußerst nützlich. Ein weiterer Vorteil ist, dass ein trainiertes CNN einen ganzen Pool an Objekten unterscheiden kann. Hierbei sorgt das Vorhandensein bestimmter, einem Objekt zugeordneten Features dafür, dass eben jenes Objekt als Antwort ausgegeben wird. Sollten beispielsweise Augen gefunden werden, würde dies schon einmal für alles was Augen besitzt und gegen alles, was Selbige eben nicht besitzt, wie zum Beispiel eine Pflanze, sprechen. Die meisten Convolutional Neural Networks lassen sich in vier Bausteine zerlegen: Convolutional Layer, Activation Layer, Pooling Layer und fully-connected Layer.

2.3.1 Convolutional Layer

Um den Prozess der Convolution verständlicher zu gestalten, sollte man sich sowohl das Bild, welches als Input dient, als auch die daraus resultierenden „Feature-Maps“ als einen Würfel bzw. Quader vorstellen. Das Input Bild besteht im Falle eines Farbbildes aus drei zweidimensionalen Matrizen. Jede Matrix steht hierbei für einen der drei Farbkanäle. So ergibt sich beispielsweise aus einem 100 mal 100 Pixel großen Farbbild ein Quader mit insgesamt 30.000 Werten. Ein Filter oder auch Kernel deckt immer die volle Tiefe seines Inputs ab. Bei einem Farbbild muss der Filter also ebenfalls immer eine Tiefe von 3 haben. Bei diesem Schritt wird das Bild systematisch nach den Features durchsucht.

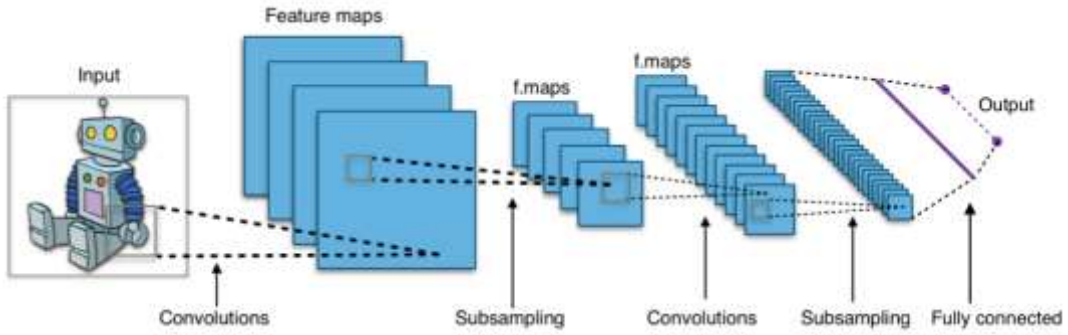

Abbildung 11 Beispielhafter Aufbau eines Convolutional Neural Networks. Der Input ist hier ein Farbbild und lediglich zur Vereinfachung als ein Layer dargestellt. Normalerweise bestünde er aus drei Layern für Rot, Grün und Blau. Subsampling ist hierbei das Pooling. Bei vielen der aktuellen Netze wird hinter jedem Convolution Layer ein ReLU Layer angewandt. Dies ist in dieser Abbildung nicht zu sehen. [Wikipedia2017]

Hierzu werden die Filter Schritt für Schritt über jeden Teil des Bildes gelegt und die Übereinstimmung mit dem Teil des Bildes und dem Filter geprüft. Für jeden vom Filter überprüften Bereich erhält man einen Wert, welcher bei größerer Übereinstimmung zwischen Filter und Bildbereich auch entsprechend hoch ausfällt. Wichtig ist hierbei die Größe des Filters und der sogenannte „Stride“. Beim Stride handelt es sich um den Abstand, um welchen der Filter jeweils in jedem Schritt verschoben wird. Durch jeden auf den Input angewandten Filter erhält man eine zweidimensionale Matrix, welche Feature-Map genannt

wird. So entsteht aus einem 15 mal 15 Pixel großem Farbbild und einem 3 mal 3 mal 3 großen Filter mit einem Stride von 1 beispielsweise eine Feature-Map bestehend aus 13 x 13 Werten. Da abhängig von der Größe des Kernels die Höhe und Breite der entstehenden Feature-Maps recht schnell schwindet, wird öfters ein sogenanntes „Padding“ verwendet. Das heißt das Bild wird mit Nullen aufgefüllt, um eine Feature-Map mit der gleichen Höhe und Breite entsprechend des Inputs zu erhalten. Dies geschieht, damit auf die erhaltenen Feature-Maps noch weitere Durchgänge der Convolution, Refined Layer Unit oder Pooling angewandt werden können. Die Feature-Maps aller im Schritt der Convolution auf das Bild angewandten Filter werden schließlich zusammengefügt und bilden so wieder einen Würfel bzw. Quader, welcher auch Convolutional Layer genannt wird. Auf diesen können anschließend weitere Stufen der Convolution, Refined Layer Unit oder Pooling angewandt werden.

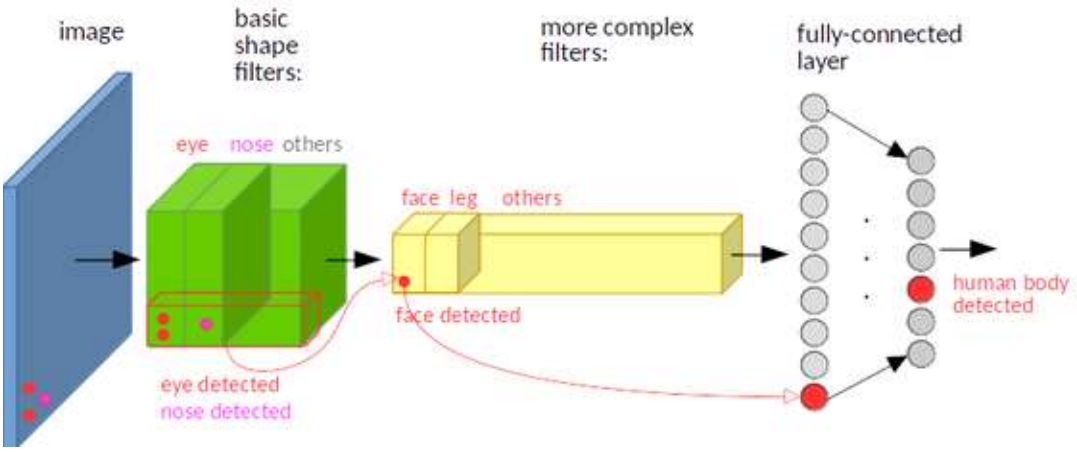

Abbildung 12 Bei dem Vorgang der Convolution werden Neuronen von bestimmten Features angeregt. Diese Neuronen regen wiederum andere Neuronen an. So können immer größere Features wahrgenommen werden. [Rolt2016]

2.3.2 Activation Layer

In diesem Layer wird auf jedes Pixel der einzelnen Feature-Maps eine nicht lineare Aktivierungsfunktion angewandt. Der Output hat hierbei exakt die Maße des Inputs. Eine oft

vorkommende Aktivierungsfunktion ist zum Beispiel die ReLU (Rectified Linear Unit) Funktion. Bei dieser wird einfach jeder Wert, der kleiner als 0 ist, auf 0 gesetzt. Abbildung 13 zeigt wie sich dieser Schritt auf eine Feature-Map auswirkt.

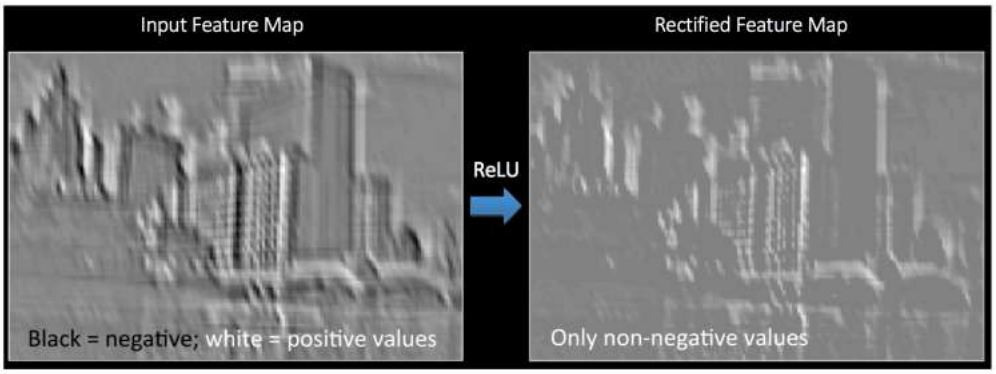

Abbildung 13 Visuelle Veranschaulichung der Anwendung der ReLU Funktion auf eine Feature-Map. [Fergus2015] S.37

2.3.3 Pooling Layer

Der Pooling Layer wird verwendet, um die Informationen zu vereinfachen ohne wichtige Details zu verlieren. Beim Pooling wird jede Matrix des Inputs einzeln behandelt. Es wird ein Quadrat mit den Maßen F x F , einer Tiefe von eins und einem Stride (S) über jede einzelne Matrix des Inputs bewegt. Aus den Werten innerhalb des Quadranten wird je nach Art des Poolings ein Wert gewählt und als Wert für den Output gespeichert. So entsteht aus jeder Matrix des Inputs eine verkleinerte Matrix, welche trotz Verkleinerung noch die wichtigen Informationen (das Vorkommen von Features) enthält. Hierbei wird auf Genauigkeit bezüglich des Vorkommens der Features verzichtet, um im Gegenzug für eine schnellere Berechnung zu sorgen. Dadurch, dass beim Pooling jede Matrix des Inputs einzeln bearbeitet wird und eine neue Matrix erzeugt, ändert sich nur die Höhe und Breite des Outputs im Gegensatz zum Input. Die Tiefe bleibt erhalten. Aus einem 40 x 40 x 15 großen Input (wobei 3 in diesem Fall die Tiefe ist) entsteht beispielsweise durch ein Max-Pooling mit einem F von 2 und einem Stride von ebenfalls 2 ein Output mit den Maßen 10 x 10 x 15. Hauptsächlich

wird beim Pooling das Max-Pooling, bei welchem jeweils der maximale im Quadranten vorkommende Wert gewählt wird, verwendet. [Li et al. 2016]

2.3.4 Fully-connected Layer

Im fully-connected Layer, welcher immer am Ende eines Convolutional Neural Networks angesiedelt ist, nehmen alle Werte des vorigen Layers Einfluss darauf, was das CNN meint im Bild zu erkennen. Hierbei gibt es einen Pool an Neuronen, welche für die jeweiligen Antworten stehen. Alle Neuronen aus dem vorigen Layer sind hierbei mit allen Antwortneuronen verbunden. Jedes Antwortneuron hat einige mit ihm verbundene Neuronen, welche stärker gewichtet werden. So sollte beispielsweise ein Neuron mit einem hohen Wert, welcher durch einen Filter zustande kam, der auf grüne Bälle anspricht, für ein Antwortneuron, welches nach Menschen sucht, sehr gering gewichtet werden. Ein fully-connected Layer kann auch aus mehreren Zwischenabstimmungen bestehen, deren Antworten wieder genutzt werden, um eine Abstimmung zu starten. Bei diesen Neuronen, welche für Zwischenabstimmungen genutzt werden, wird von einem „Hidden Layer“ gesprochen (siehe Abbildung 14).

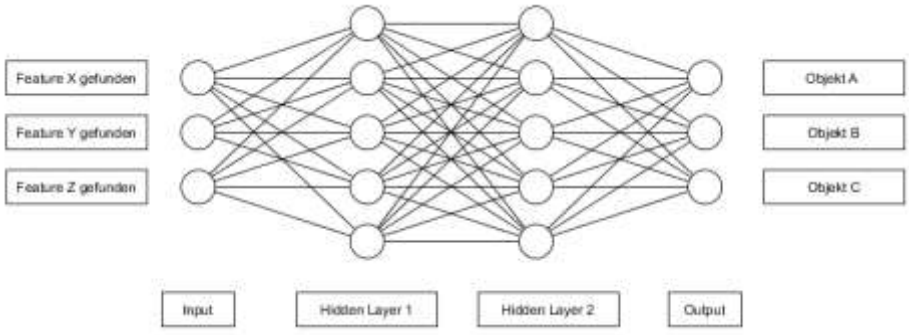

Abbildung 14 Vereinfachte Darstellung eines fully-connected Layer. Der Input besteht hierbei aus drei Neuronen, die auf die Anwesenheit bestimmter Features im Bild reagieren. Der Output besteht aus drei Neuronen, welche jeweils für ein zu klassifizierendes Objekt stehen.

2.3.5 Lernprozess

Sowohl die Features als auch die Gewichtungen im fully-connected Layer werden im Lernprozess gebildet. Dadurch, dass dem System ein Pool an Bildern zum Lernen gegeben wird, von denen das System weiß, was sich auf ihnen befindet, kann es anhand der Fehlerrate die Gewichtungen und die einzelnen Pixel der Features bzw. der Filter einstellen. Um dies zu erreichen, werden die einzelnen Werte entsprechend der Fehlerrate erhöht oder verringert, bis die Fehlerrate niedrig genug ist. Dieser Vorgang wird „Backpropagation“ genannt.
Neben den Faktoren, welche selbst erlernt werden, existieren viele andere Werte, welche manuell konfiguriert werden können, um die Performance des Netzes zu steigern. Hierzu gehören beispielsweise die Anzahl und Größe der Features bzw. Filter im Convolutional Layer, die Fenstergröße und der Stride im Pooling Layer und die Anzahl der Hidden Layer im Fully-connected Layer. Eines der wichtigsten Elemente, welches bei einem CNN manuell geregelt werden kann, ist der Aufbau des Netzes an sich. Die Anzahl und Reihenfolge der oben genannte Layer spielt eine sehr große Rolle, was die Fehlerquote und Geschwindigkeit des CNN angeht.

2.4 Ähnliche Projekte

In diesem Kapitel werden einige Projekte vorstellen, welche sich hauptsächlich mit der Erkennung von unter anderem natürlichen Objekten beschäftigen. Hierbei werden auch Techniken verwendet, welche in Kapitel 2 vorgestellt wurden. Wie auch in dieser Arbeit wurden für die Tests in den folgenden Projekten natürliche Objekte in Form von Lebensmitteln genutzt. Die Auswahl dieser Projekte fiel mit dem Hintergedanken der späteren Nutzung des in dieser Arbeit vermittelten Wissens zur Erstellung einer Anwendung zur Unterstützung beim Kochen.

2.4.1 FoodBoard

Eines der Projekte, welches thematisch recht nah an den in dieser Arbeit durchgeführten Tests liegt, ist das Projekt “FoodBoard: Surface Contact Imaging for Food Recognition” [Pham2013]. Der Kern dieses Projektes ist ebenfalls das Erkennen von natürlichen Objekten in Form von Lebensmitteln. Bei diesem Projekt wird gleichermaßen eine Kamera mit optischen Erkennungsverfahren verwendet. Der große Unterschied liegt hierbei aber darin, dass im Projekt FoodBoard die Kamera nicht direkt auf das Schneidebrett bzw. die Arbeitsfläche, auf welcher sich das Brett befindet, gerichtet ist. In diesem Ansatz befindet sich die Kamera quasi im Schneidebrett und filmt eine Matrix aus Fieberglasleitungen. Dies sorgt für ein recht grobes Bild. Der große Vorteil an dieser Herangehensweise ist der Ausschluss unwichtiger Informationen. Dadurch, dass die Kamera nur den von Fieberglasleitungen abgedeckten Raum auf dem Schneidebrett überwacht, muss keine Filterung von Objekten stattfinden, welche sich außerhalb des Bretts oder über dem Brett befinden. Für die Erkennung des Lebensmittels wird hier eine Kombination aus Farbprüfung und Feature Matching mit Surf Features verwendet, welche jeweils eine bestimmte Gewichtung haben. In diesem Fall wurde die Farbe des Objektes etwas stärker gewertet als die SURF Features. Wie auch bei dem für diese Bachelorarbeit durchgeführten Versuch in Kapitel 3.3.2 wurde in diesem Paper der K-Means Algorithmus verwendet, um die durchschnittliche Farbe des Objektes zu erfassen. Die Wahl des Verfahrens zum Finden und Beschreiben von Features fiel hierbei laut [Pham et al. 2013] auf SURF, da SURF Features invariant gegenüber Rotation und Skalierung sind. Die kombinierte Nutzung verschiedener Features macht diese Arbeit zu einem guten Beispiel dafür, dass ein Zusammenspiel von Erkennungsmerkmalen zu einer guten Erkennungsrate führen kann, wo die Verwendung von nur einem Merkmal nicht eindeutig genug wäre. Die Resultate bei diesem Ansatz waren von Objekt zu Objekt unterschiedlich, so hatten Karotten und Bacon eine sehr hohe Erkennungsrate von über 90%, geschälte Zwiebeln zum Beispiel lieferten aber eine Falsch-positiv-Rate von über 20% [Pham2013].

2.4.2 Recognition and volume estimation of food intake using a mobile device

Eines der Kernthemen des Papers mit dem Titel "Recognition and volume estimation of food intake using a mobile device" [Puri et al. 2009] ist das Erkennen von fertig zubereiteten Mahlzeiten. In diesem Projekt wird die Erkennung der Mahlzeiten jedoch eingesetzt, um in Zusammenarbeit mit der Berechnung des Volumens der jeweiligen Gerichte die Anzahl an Kalorien der Zutaten zu errechnen. Hierbei wird die Interaktion mit Nutzern über ein Smartphone bewerkstelligt, während die schwereren Berechnungen wie Sprach- und Bilderkennung über einen externen Server geregelt werden, welcher über das Internet angesprochen wird. Des Weiteren wird bei diesem Projekt nur ein Bild verarbeitet. Um den Pool der möglichen Zutaten einzugrenzen und somit die Erkennung der Zutaten zu erleichtern, wird hier der Nutzer beim Senden des Bildes aufgefordert den Inhalt des Bildes verbal wiederzugeben. Über Spracherkennung wird dann daraus der Pool der zu erkennenden Zutaten generiert. Für die eigentliche Erkennung der Zutaten wird hier neben Textur Features auch auf die Farbe der Zutat geachtet. Um farbliche Abweichungen, welche durch Beleuchtung von unterschiedlicher Intensität und Farbton verursacht werden, zu korrigieren, wird in diesem Ansatz ein Marker mit vierzehn verschiedenfarbigen Vierecken mitfotografiert. Basierend auf den dem System bekannten Werten der farbigen Vierecke, werden die Farbwerte des Bildes korrigiert. Auch das Problem der Verzerrung des Bildes durch einen schrägen Blickwinkel wurde hier beachtet. Durch ein Schachbrettmuster, welches sich ebenfalls auf dem mitfotografierten Marker befindet, kann die Verzerrung korrigiert werden. Als Features für die Erkennung der Texturen verwenden Puri, Zhu und Yu in ihrem Projekt sogenannte Texton Histogramme. Die Classifier werden mit dem AdaBoost Verfahren erstellt. Diese Arbeit ist ein gutes Beispiel dafür, wie ein Marker mit bekannten Maßen und Farben zur Kalibrierung eines Bildes verwendet werden kann, um Features wie Farbe und Größe mit einer höheren Genauigkeit zu bestimmen.

2.4.3 Image Recognition of 85 Food Categories by Feature Fusion

In diesem Paper [Hoashi et al. 2010] wurde ein Objekterkennungssystem für 85 verschiedene Essenskategorien erstellt, welches ein Foto eines unbekannten Gerichts nimmt und es in eine dieser 85 Kategorien einordnet. Hierbei wird beim Erkennen eine Kombination verschiedener Features verwendet, namentlich: bag-of-features (BoF), Farbhistogramme, Gabor Textur Features und Gradient Histogramme. Diese Erkennungsmerkmale werden durch sogenanntes Multiple Kernel Learning für jedes erlernte Gericht unterschiedlich gewichtet, da sich laut [Hoashi et al. 2010] beispielsweise ein Hamburger eher durch seine Textur und Suppe eher durch ihre Farbe erkennen lässt. Je nach Kategorie reichten die Erkennungsraten in diesem Ansatz von 17% für Ganmodoki bis hin zu 95% für Soba Nudeln. Wie auch schon beim vorangegangenen Ansatz liegt der Hauptunterschied zur Herangehensweise bei den im Rahmen dieser Arbeit durchgeführten Tests darin, dass in der Arbeit von Hoashi, Joutou und Yanai Bilder von bereits zubereiteten Gerichten an Stelle von unbearbeiteten natürlichen Objekten untersucht wurden. Der Unterschied zu den in dieser Arbeit vorgenommen Tests ist hierbei jedoch etwas geringer als bei [Puri et al. 2009], da hier das Gericht nicht in seine Bestandteile aufgeteilt wird, sondern an sich als ein organisches Objekt gesehen wird. Die Tatsache, dass ein Gericht bei jeder Zubereitung anders aussieht, stellt eine starke Parallele zu den Unterschieden im Auftreten der in dieser Arbeit getesteten Objekte dar. Ein weiterer Unterschied ist, dass die überprüften Fotos hier alle aus einem zufälligen Winkel fotografiert wurden. Bei dem Großteil der im Rahmen dieser Arbeit getesteten Bilder wurde eine gleichbleibende Konfiguration von Kamera und Objekt genutzt. Eine Ausnahme bildet hierbei der Test der Convolutional Neural Networks im Kapitel 3.4. Die dort verwendeten Bilder bilden ebenfalls die zu überprüfenden Objekte aus unterschiedlichen Winkeln und mit leicht variierender Beleuchtung ab. Diese Arbeit zeigt sehr gut, wie die Gewichtung mehrerer Features objektabhängig erlernt werden kann, um eine bessere Klassifikation zu ermöglichen. Eine ähnliche Vorgehensweise findet sich bei den Convolutional Neural Networks, in denen auch erlernt wird, wie stark die jeweiligen Features für ein bestimmtes Objekt zu gewichten sind.

2.4.4 Food Detection and Recognition Using Convolutional Neural Network

Das aktuellste Paper, welches an dieser Stelle vorgestellt werden soll, trägt den Namen "Food Detection and Recognition Using Convolutional Neural Network" [Kagaya et al. 2014]. Diese Arbeit handelt wie auch schon [Hoashi et al. 2010] vom Erkennen von fertig zubereiteten Gerichten. Allerdings wird hier auf die Verwendung von Convolutional Neural Networks gesetzt, da dies eine aktuelle und sehr effiziente Methode zur Objekterkennung ist. Als Informationsquelle wurde [Aizawa et al. 2014] genutzt. Bei der in [Aizawa et al. 2014] erstellten App für Smartphones können Nutzer für ein Essenstagebuch Fotos von ihren täglichen Mahlzeiten machen und die verschiedenen Gerichte in diesem Foto markieren. Aus den beschrifteten Bildern dieser App wurde in [Kagaya et al. 2014] eine Datenbank von ca. 170.000 Bildern erstellt. Aus dieser Datenbank wurden wiederum die zehn häufigsten Gerichte entnommen, um ein Convolutional Neural Network zu trainieren. In diesem Paper wurden hauptsächlich zwei Aufgaben getestet: Die Unterscheidung von Essen und Non-Food Gegenständen und das Klassifizieren von Gerichten in die 10 erlernten Kategorien. Es zeigte sich, dass CNN in beiden Aufgaben bessere Ergebnisse lieferten als bisher genutzte Techniken zur Objekterkennung. Dieses Paper zeigt sehr gut, dass durch die Nutzung von Convolutional Neural Networks ein breites Einsatzgebiet in der Computer Vision abgedeckt werden kann. Wo zuvor noch viele verschiedene Verfahren für die Klassifikation von Objekten kombiniert werden mussten, lassen sich heute schon durch die Nutzung von CNN ohne weitere Hilfen sehr gute Ergebnisse verzeichnen.

3 Design und Realisierung

Im folgenden Kapitel werden einige Tests zu den im vorangegangenen Kapitel vorgestellten Methoden zur Erkennung von natürlichen Objekten durchgeführt. Zusätzlich werden noch zwei kleinere Verfahren, welche lediglich als Hilfsmittel bei der Erkennung von Objekten dienen, getestet: ein Verfahren zum Bestimmen der Größe des Objektes und ein Verfahren zum Berechnen der dominanten Farbe des Objektes. Für die Umsetzung der Tests fiel die Wahl auf Python, da Python mit den wichtigsten Bibliotheken wie zum Beispiel OpenCV [Bradski2000] und Keras [Chollet et al. 2015] arbeiten kann, ohne dabei den Code übermäßig komplex erscheinen zu lassen. Da die OpenCV Library für Python lediglich ein Wrapper für den unterliegenden C bzw. C++ Code ist, dürfte es auch keine starken Geschwindigkeitsunterschiede geben. Bezüglich der Geschwindigkeit der einzelnen Verfahren wurden in dieser Arbeit jedoch keine Tests durchgeführt. Bei allen Tests wurde auf gleichbleibende Bedingungen geachtet. Bis auf eine Ausnahme wurden alle Bilder immer mit einer gleichen Entfernung von Kamera zum Objekt, dem gleichen Aufnahmewinkel und der gleichen Beleuchtung erstellt. Die Ausnahme bildet hierbei die Überprüfung der Convolutional Neural Networks, da dort zum Testen eine Sammlung von lizenzfreien Bildern aus dem Internet verwendet wurde. In einer späteren Anwendung im realen Umfeld ist es natürlich nicht immer möglich diese gleichbleibenden Bedingungen einzuhalten. Die Ergebnisse der Tests spiegeln dementsprechend nur das Verhalten in der vorhandenen Laborumgebung wieder. Es ist nicht auszuschließen, dass durch andere Verhältnisse auch andere Ergebnisse erzielt werden. Die Testergebnisse sind hierbei auch weniger als Bewertung der einzelnen Verfahren im Gesamten zu verstehen. Sie sollen eher als

Hilfestellung bei der Wahl eines Verfahrens zur Erkennung natürlicher Objekte im Kontext der Implementierung einer Anwendung dienen.

3.1 Testen von verschiedenen Verfahren zur Erkennung von und Beschreibung von Keypoints

3.1.1 Aufbau

Für diesen Test wurde eine Kamera fest über einem Schneidebrett installiert. Das Schneidebrett wurde mit zwei direkten Beleuchtungsquellen und durch das natürliche Licht eines Fensters, welches sich an der Arbeitsfläche befindet, ausgeleuchtet. Mit diesem Versuchsaufbau wurde eine Videoaufnahme erstellt, in der verschiedene, für den Prozess der Zubereitung notwendige Objekte nacheinander auf das Schneidebrett gelegt wurden. Da für diese Arbeit nur die Algorithmen an sich auf Eignung zur Wiedererkennung getestet werden und es für Tests einfacher ist auf Bilder zurückzugreifen, wurden für diese Tests einzelne Frames mit den entsprechenden Objekten aus dem Video extrahiert. Bei den Objekten handelt es sich um organische Objekte (hier Orangen, Tomaten und Paprika) und um Werkzeuge zur Verarbeitung der Zutaten (hier zwei verschiedene Messer und ein Gewürzstreuer). Von jedem der organischen Objekte werden zwei Exemplare verwendet, um sicherzustellen, dass auch verschiedene Objekte der gleichen Art wiedererkannt werden können. Als Referenzbild der zu erkennenden Objekte dienen zugeschnittene Versionen des jeweiligen Objektes, welche ebenfalls aus den Frames des Videos extrahiert wurden.

Für jedes Referenzbild werden zuerst Positivtests und anschließend Negativtests vorgenommen. In den Positivtests werden Bilder durchsucht, welche das zu findende Objekt enthalten. In den Negativtests ist das Gegenteil der Fall. Die Auswertung der Tests muss manuell vorgenommen werden, da es sich bei den Testbilden nicht um computergenerierte

Bilder handelt, bei denen Rotation, Position etc. des zu findenden Objektes bekannt sind. Für die grobe Auswertung wird betrachtet, wie viele der Keypoints des Referenzbildes auf Punkte innerhalb des Objektes im Testbild abgebildet werden. Eine genauere Auswertung würde nur die Treffer werten, welche sich auch tatsächlich auf dasselbe Feature des Objektes beziehen. Diese Art der Auswertung wird hier verwendet, da für die reine Klassifizierung eines Objektes nicht relevant ist, wo ein für dieses Objekt typisches Feature gefunden wird, solange es innerhalb des Objektes ist.

Alle in diesem Test zu untersuchenden Bilder sind, bis auf die Konvertierung hin zum Graustufenbild, unbearbeitet. Es wurde im Gegensatz zu späteren Tests keine Eingrenzung der Region of Interest vorgenommen. Bei der Region of Interests handelt es sich um einen Bildbereich, welcher von besonderer Relevanz ist. Ebenfalls konträr zu späteren Tests wurde auf ein Filtern von zu kleinen Features durch gaußsches Weichzeichnen verzichtet.

3.1.2 Finden von Keypoints

In diesem Unterkapitel werden die von den jeweiligen Algorithmen gefundenen Keypoints ausgewertet und verglichen. Die Qualitätsmaßstäbe hierbei sind Anzahl und Einzigartigkeit der im Objekt gefundenen Keypoints. Alle Algorithmen wurden im ersten Durchlauf mit deren Standartparametern ausgeführt.

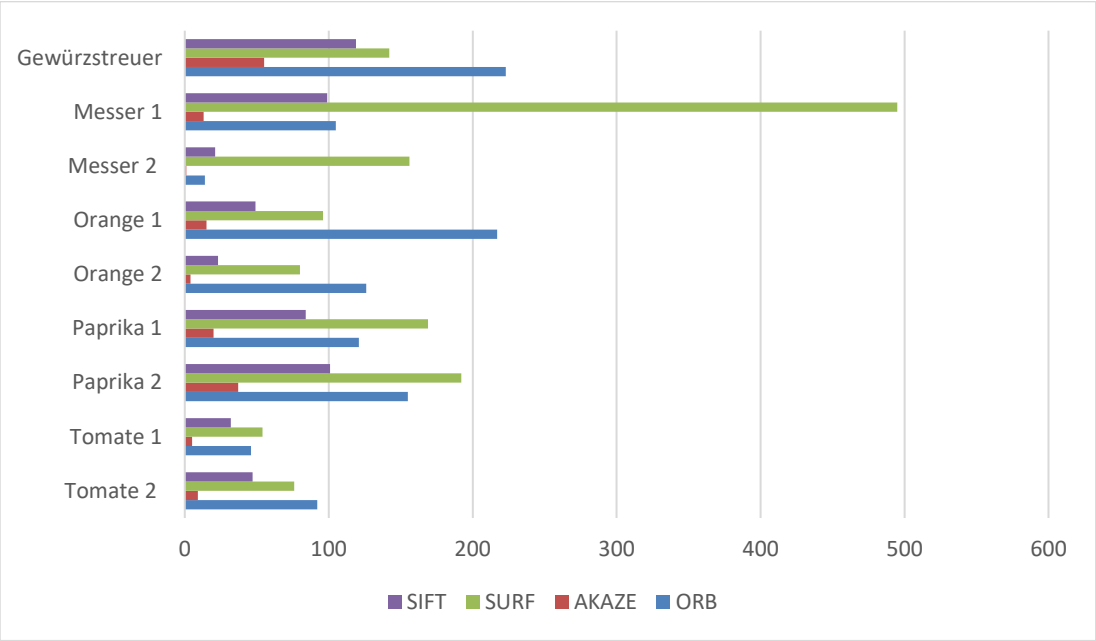

Abbildung 15 Anzahl der gefundenen Features pro Objekt

3.1.2.1. SIFT

Der SIFT Algorithmus liegt, was die Anzahl der gefundenen Keypoints betrifft, mit durchschnittlich 80 Keypoints für Werkzeuge und 56 für die getesteten Lebensmittel auf dem vorletzten Platz. Leider sind die wenigsten der gefundenen Punkte einzigartig und damit für das spätere Matching mehr hinderlich als hilfreich. Ein sehr großer Teil der von SIFT gefundenen Keypoints sind sehr klein und somit sehr anfällig für Bildrauschen. Des Weiteren liegen viele Keypoints entlang der Kanten des Objekts, obwohl im SIFT Algorithmus eigentlich dafür gesorgt wird, dass Keypoints auf Kanten ausgeschlossen werden. Die starke Beleuchtung von oberhalb des Schneidebretts sorgt zwar für weniger Schattenbildung, aber auch für Reflektionen auf den glänzenden Oberflächen der Objekte, welche von SIFT ebenfalls als Keypoint erkannt werden. Da diese Reflektionen stark abhängig von der Position des Objektes sind, sind sie nicht beständig und sollten somit ebenfalls nicht beachtet werden.

3.1.2.2. SURF

Der SURF Algorithmus liegt, gemessen an der Anzahl der gefundenen Keypoints auf dem ersten Platz. Es wurden durchschnittlich 264 Keypoints in den Werkzeugen und 111 Keypoints in den Lebensmitteln gefunden. Der SURF Algorithmus fand, wie in Abbildung 16 zu sehen ist, weitaus mehr große Keypoints als SIFT. Diese dürften beständiger gegenüber Noise bzw. Bildrauschen sein. Leider hat auch SURF sehr viele Punkte entlang der Kanten gefunden. Diese Punkte sind für ein späteres Matching sehr hinderlich, da sie überall auf der Kante wiedererkannt werden können und somit verrutschen. Wie schon bei SIFT wurden von SURF auch mehrere Spieglungen auf den glatten Oberflächen der Objekte als Keypoints gewertet.

3.1.2.3. ORB

Der ORB Algorithmus liegt mit durchschnittlich 117 gefundenen Keypoints für die Werkzeuge und 126 für die Lebensmittel auf dem zweiten Platz, was die Anzahl der gefundenen Punkte betrifft. Von allen getesteten Algorithmen hat ORB die höchste Anzahl an größeren Keypoints hervorgebracht. Dies sollte – gesetzt dem Fall, dass diese größeren Keypoints auch einzigartig genug sind – für ein stabiles Matching sorgen. Im Gegensatz zu SIFT und SURF wurden hier auf der Kontur des Objektes liegende Keypoints zum Großteil herausgefiltert. Bei allen getesteten Objekten fand mit ORB eine starke Überlagerung der gefundenen Punkte vor, wie in Abbildung 16 sehr gut zu sehen ist. Leider wurden auch bei ORB einige Spieglungen als Keypoints verwendet.

3.1.2.4. AKAZE

Mit durchschnittlich 23 gefundenen Keypoints für die Werkzeuge und 15 für die Lebensmittel liegt der AKAZE Algorithmus auf dem letzten Platz bezüglich der Anzahl der gefundenen

Keypoints. AKAZE schaffte es besser als alle Algorithmen schlechte Keypoints auf der Kontur der Objekte herauszufiltern. Leider gestaltet sich durch die geringe Anzahl der gefundenen Punkte ein Matching schwer. Des Weiteren sind alle der wenigen gefundenen Keypoints recht klein.

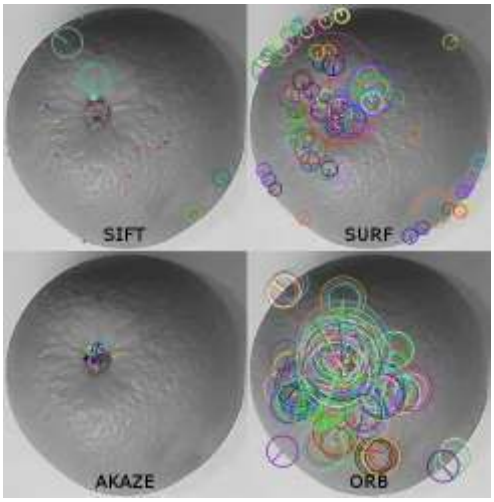

Abbildung 16 Vergleich der gefunden Keypoints am Beispiel einer Orange.

3.1.3 Matching

Es wird versucht alle Keypunkte der jeweiligen Objekte mit Hilfe eines Brute-Force-Matchers in den Testbildern zu lokalisieren. Für SIFT und SURF wird der Brute-Force-Matcher mit Standartwerten initialisiert. Da ORB und AKAZE binäre Deskriptoren verwendet, muss im Falle dieser beiden Algorithmen die Entfernungsmessung auf Hamming-Abstand gesetzt werden. Mit Hilfe des Hamming-Abstandes kann hier die die Abweichung der Strings, aus denen die binären Deskriptoren bestehen, gemessen werden. Als Matchingalgorithmus wird der K-Nearest-Neighbors Algorithmus mit K = 2 verwendet, damit später ein Ratio Test durchgeführt werden kann. Bei allen Algorithmen wird nach dem Matching noch ein Ratio Test, wie er in [Lowe2004] erwähnt wird, durchgeführt.

3.1.3.1. SIFT

In den Positivtests befanden sich viele der vom SIFT Algorithmus gefundenen Matches auf dem korrekten Objekt im Testbild. Leider wurden viele von den auf Kanten liegenden Keypoints falsch zugeordnet. Am besten funktionierte SIFT zum Finden des Gewürzstreuers, da sich auf diesem eine Aufschrift und ein Logo befanden, welche einen großen Wiedererkennungswert haben. Am schlechtesten schnitt ein Gemüsemesser an, bei welchen aber von vornherein wenige Keypoints gefunden wurden. Die Lebensmittel besaßen leider nur wenige Keypoints, welche halbwegs konstant wiedererkannt werden konnten. Das beste Feature in diesem Testdurchlauf war der Strunk der Orange, welcher aber leider auch fälschlich mit dem Strunk der Tomate gematcht wurde.

In den Negativtests wurden für fast jedes Objekt Matches gefunden. Jedoch weniger als bei den Positivtests. Bei den Matches handelte es sich fast ausschließlich um auf der Kante liegende, zu kleine oder von Spiegelungen verursachte Keypoints. Bei dem Objekt mit den wenigsten Matches in Negativtests handelt es sich um die Orange. Die meisten Matches im Negativtest – was wohl aus den vielen gefundenen Keypoints entlang der Kontur der Klinge resultierte – produzierte ein Messer.

3.1.3.2. SURF

In den Positivtests befanden sich wie bei SIFT der Großteil der gefundenen Matches auf dem korrekten Objekt im zu überprüfenden Bild. Analog zu SIFT wurden viele Keypoints auf der Kontur der Objekte oder Keypoints, die nicht eindeutig genug waren – wie zum Beispiel einzelne Poren der Orange – falsch zugeordnet. Das Feature mit dem größten Wiedererkennungswert war auch hier wieder der Strunk der Orange.

Die Negativtests verhielten sich, was die Beschaffenheit der falschen Matches angeht, ähnlich wie die Negativtests beim SIFT Algorithmus. Auch hier wurden in den Negativtests

weniger Matches als in den Positivtests gefunden. Die Anzahl der Matches ist hier bei SURF jedoch weitaus höher. Dies ist wahrscheinlich auf die größere Anzahl an gefundenen Keypoints im Referenzbild zurückzuführen.

3.1.3.3. ORB

In den Positivtests verhielt sich ORB – vermutlich bedingt durch die große Menge an gefundenen Keypoints – eher negativ. Abgesehen von wenigen Matches, welche sich auf stabile und einzigartige Features bezogen, befanden sich ein Großteil der Matches außerhalb des gesuchten Objektes. Der Gewürzstreuer, die Orangen und die Tomaten stellten hierbei jedoch eine Ausnahme dar. Der Gewürzstreuer wurde aufgrund des Schriftzuges auf dem Deckel, welcher viele differenzierbare Features bot, gut wiedererkannt. Sowohl Orangen als auch Tomaten boten mit ihrem Strunk ein sehr gutes Feature, welches wiederum zu mehreren Keypoints mit passenden Matches sorgte. Der Strunk wurde sogar unabhängig davon, welche Orange als Referenzbild genutzt wurde, durch mehrere Matchings gefunden. Hierbei musste aber der Strunk im Testbild immer sichtbar sein. Wurden Orange oder Tomate so gedreht, dass der Strunk nicht mehr sichtbar war, wurde ein besonders schlechtes Ergebnis erzielt, in dem alle Keypoints des Strunks auf Keypoints außerhalb des Objektes gematcht wurden.

Aufgrund der hohen Zahl an gefundenen Keypoints wurden beim ORB Algorithmus eine größere Menge an Matches in den Negativtests gefunden. Ein Großteil dieser falschen Matches befanden sich außerhalb des Schneidebretts und sind somit nicht so schwerwiegend wie falsche Matchings innerhalb des Bretts, da sie durch Verwenden einer Region of Interests eliminiert werden würden. Besonders viele fälschliche Matchings entstanden im Negativtest zwischen Orange und Paprika. Dies lässt sich auf die ähnliche Beschaffenheit der beiden Objekte – wie beispielsweise die glatte, leicht spiegelnde Oberfläche – zurückführen.

3.1.3.4. AKAZE

Aufgrund der sehr geringen Anzahl an gefundenen Keypoints, ist der AKAZE Algorithmus der einzige Algorithmus, bei dem es vorkam, dass in einigen Positivtests kein einziges Matching entstand. Trotz der niedrigen Anzahl an Keypoints waren diese doch qualitativ hochwertig, was in den Positivtests zu wenigen, jedoch guten Matchings führte. In allen Positivtests kam es insgesamt nur zwei Mal dazu, dass ein Keypoint des Referenzbildes einem Feature außerhalb des gesuchten Objektes im Testbild zugeordnet wurde.

Die Nutzung des AKAZE Algorithmus führte in den Negativtests zu der geringsten Anzahl an fehlerhafter Matchings. Abgesehen von den Fällen, in denen eine Orange oder der Gewürzstreuer als Referenz verwendet wurde, befanden sich fast alle fehlerhaften Matchings außerhalb des Schneidebretts und könnten somit wie schon zuvor beschrieben durch die Verwendung einer Region of Interests aussortiert werden.

3.1.4 Testfazit

Das Fazit aus dieser Testreihe ist, dass in dieser Konfiguration keiner der hier verwendeten Algorithmen und wahrscheinlich auch kein anderer Algorithmus, in dem sich auf einzelne Features konzentriert wird, zum Wiedererkennen bzw. Unterscheiden verschiedener Lebensmittel und Werkzeuge geeignet ist. Es könnte wahrscheinlich eine Verbesserung stattfinden, indem sich auf gröbere Features konzentriert wird. So sollten beispielsweise nicht mehrere Punkte im Strunk einer Orange als Features gesehen werden. Stattdessen sollte der Strunk als Ganzes als Feature gesehen werden. Aber selbst mit dieser Veränderung wäre es immer noch schwer eine eindeutige Identifikation des Objektes zu bewerkstelligen, da natürlich mehrere Lebensmittel ein Merkmal wie einen Strunk besitzen. Ein großer Nachteil aller getesteten Algorithmen zur Feature-Erkennung und Deskriptorerstellung ist, dass durch die Verwendung von Graustufenbildern eines der wichtigsten Merkmale der Lebensmittel außer Acht gelassen wird: die Farbe. Anhand der Farbe könnte, bevor

überhaupt auf Gemeinsamkeiten getestet wird, eine Aussortierung der jeweiligen Objekte stattfinden. Wofür sich die Algorithmen jedoch sehr gut geeignet haben, ist die Wiedererkennung von Objekten mit Beschriftung und Mustern. So könnten SIFT, SURF, ORB und AKAZE verwendet werden, um jene Zutaten zu identifizieren, welche in einer beschrifteten Verpackung gelagert werden. So könnte zum Beispiel im Falle von verpacktem Speck der Nutzer gebeten werden, kurz den Speck samt Verpackung auf das Schneidebrett zu legen, damit dieser erkannt werden kann. Auch für die Wiedererkennung von Gewürzen in einem beschrifteten Streuer eigenen sich die getesteten Verfahren. Es würde sich also anbieten, abhängig von dem Objekt, nach welchem das Bild durchsucht werden soll, den jeweils am besten geeigneten Algorithmus zu wählen.

3.2 Testen von Haar-based Classifiern

3.2.1 Aufbau

Für dieses Testszenario wird ein auf Haar-like Features basierender Classifier erstellt, der die Aufgabe hat Gurken zu erkennen. Zusätzlich zu diesem selbst trainierten Classifier zur Erkennung von Gurken wird noch ein weiterer Classifier zur Erkennung eines Lebensmittels – in diesem Fall einer Banane – und ein hoch spezialisierter Classifier zur Erkennung von Gesichtern getestet. Bei dem Classifier zur Erkennung von Bananen handelt es sich um einen im Laufe eines Tutorials erstellten Classifier von Robin Mehner [Mehner2013]. Bei dem Classifier zum Erkennen von Gesichtern handelt es sich um einen von Rainer Lienhart erstellten Classifier, welcher standartmäßig in OpenCV enthalten ist. Bei diesem Classifier werden verbesserte Features genutzt, um die Fehlerrate zu reduzieren. Für weitere Informationen über die Erstellung bzw. das Training dieses Classifiers wird an dieser Stelle auf das von Rainer Lienhart verfasste Paper mit dem Titel "Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection" [Lienhart et al. 2003] verwiesen.

Das Testen der letzteren beiden dient dem Zweck zu zeigen, dass die Erkennungsrate unter anderem stark abhängig ist von den Bildsammlungen, welche zum Trainieren verwendet wurden. Sollte die Erkennungsrate des für diese Arbeit erstellten Classifiers zur Erkennung von Gurken sehr negativ sein, kann anhand der Vergleichswerte gezeigt werden, dass mit einem besseren Trainingsset von Bildern das gewünschte Ergebnis dennoch erreichbar ist. Um die Erkennungsrate zu testen, werden für jeden der drei Classifier jeweils ein Positivset und ein Negativset erstellt. Sowohl das Positivset als auch das Negativset bestehen aus jeweils 100 Bildern. Das Positivset enthält in jedem Bild das gesuchte Objekt in verschiedenen Ausführungen, Orientierungen und Verdeckungsgraden, während das Negativset eine Sammlung an anderen Objekten in verschiedenen Orientierungen etc. enthält. Diese Sets werden dann anhand der zugehörigen Classifier nach dem zu findenden Objekt durchsucht.

3.2.2 Erstellen des Classifiers

Bei der Erstellung des Classifiers diente eine von R. Mehner verfasste Anleitung [Mehner2013] als Vorbild. Um einen Classifier zu erstellen bzw. zu trainieren, wird ein großes Set an Positivbildern und ein großes Set an Negativbildern benötigt. Im Falle eines hoch qualifizierten Classifiers wie er in [Lienhart et al. 2003] beschrieben wird, wurden 5.000 Positivbilder und 1.000 Negativbilder für das Training verwendet. Die 5.000 Positivbilder wurden aus 1.000 Originalbildern erstellt. Dafür wurden die Originalbilder zufällig rotiert, skaliert und gespiegelt. Bei dem im Rahmen dieses Tests erstellten Classifiers wurden lediglich 1.000 Positivbilder und 600 Negativbilder verwendet. Auch hier basiert das Positivset auf einem Set von 50 Originalbildern aus denen durch zufällige Rotation, Skalierung und Spiegelung 1.000 Positivbilder erstellt wurden. Für jedes Positivbild ist somit die Position, die Rotation und die Skalierung des gesuchten Objektes bekannt. Diese Informationen können zum Trainieren des Classifiers genutzt werden. Die 50 Originalbilder sind Bilder von Gurken, in denen der Hintergrund so gut wie möglich durch Ausschneiden der Gurke entfernt wurde. In diesem Versuch wurde für jedes der Originalbilder dieselbe Gurke verwendet. Um

eine bessere Erkennungsrate zu erhalten, sollten möglichst viele verschiedene Gurken und ein weitaus größerer Satz an Originalbildern verwendet werden. Die Gurke eignete sich gut als Testobjekt zur Erstellung eines Classifiers, da die Anzahl der Posen, in denen eine Gurke platziert werden kann, sehr begrenzt ist und sie von allen Seiten ähnlich aussieht. Im Gegensatz dazu müssten für beispielsweise eine Paprika mehrere Classifier angefertigt werden, da diese von abhängig von ihrer Orientierung stark unterschiedliche Features bietet. Die Anforderungen an das Negativset sind weitaus geringer. Es wurde hauptsächlich darauf geachtet, dass sich in den Bildern nicht das gesuchte Objekt befindet. Die Bilder des Negativsets wurden über ein selbstgeschriebenes Python Skript aus dem Internet heruntergeladen. Als Quelle wurde dazu wie bei [Russakovsky et al. 2014] http://image-net.org/ verwendet.

Mit diesen beiden Trainingssets wurde der Classifier nun trainiert. Bei diesem Training kann unter anderem eingestellt werden wie viele Stages für den Classifier erstellt werden. Sollte hier ein zu geringer Wert eingesetzt werden, könnte der Classifier zu grob werden und zu viele fälschlich positive Ergebnisse liefern. Wird aber ein zu hoher Wert eingesetzt könnte der Classifier zu sensibel werden und somit viele fälschlich negative Ergebnisse liefern. In diesem Test wurde ein Wert von 20 Stages gewählt. Pro Stage wurden 1.000 Positivbilder und 600 Negativbilder für das Training genutzt. Für die Auswahl der Features gibt es zwei Möglichkeiten: Haar-like und LBP. In diesem Test wurden Haar-like Features verwendet. Als Boost wurde hier ein „Gentle AdaBoost“, welcher eine Variation des in [Viola et al. 2001] genutzten Verfahrens ist, verwendet. Die minimale Hit-Rate wurde auf 0.999 und die maximale Falschalarmrate auf 0.5 gesetzt.

3.2.3 Testergebnisse

Der erstellte Classifier zur Erkennung von Gurken war in den Tests sehr anfällig für Rotationen um die Z- Achse. Die getestete Gurke wurde nur in bestimmten Rotationsstufen erkannt. Dies

könnte durch schlecht generierte Trainingsbilder zu Stande gekommen sein. In einigen der extrahierten Frames wurden ein kleiner Bereich außerhalb des Schneidebretts ebenfalls als Gurke erkannt. Die fälschlich positiven Ergebnisse ließen sich durch das Nutzen einer Region of Interests, welche sich auf das Schneidebrett bezieht, eliminieren. Eine weitere Möglichkeit zur Eliminierung wäre, die durchschnittliche Größe des zu findenden Objektes in einer Datenbank zu speichern und anschließend alle Funde, welche außerhalb des Größenbereiches liegen, nicht zu beachten. In keinen der Testbilder des Negativtests wurde eine Gurke entdeckt. Der Classifier könnte durch eine größere Menge an einzigartigen Positivbildern mit unterschiedlichen Gurken und eine bessere Einstellung bezüglich der Rotation bei der Generierung weiterer Bilder ausgebessert werden.

Der Classifier zur Erkennung von Gurken lieferte das im Gegensatz zu den anderen beiden Classifiern schlechteste Ergebnis, was die Positivtests betrifft. Mit nur 44 von 100 erkannten Bildern erkannte dieser Classifier die geringste Anzahl an Objekten. Trotz des negativen Ergebnisses im Positivtest wurden keine Bilder im Negativtest fälschlicherweise erkannt. Diese wenigen Treffer im Positivtest und wenige Treffer im Negativtest könnten bedeuten, dass dieser Classifier übertrainiert ist und somit nur noch Gurken mit geringen Abweichungen von seinen Trainingsbildern erkennt.

Der Classifier zur Erkennung von Bananen hatte mit 77 von 100 erkannten Bildern eine weitaus höhere Erkennungsrate als der im Rahmen dieser Arbeit erstellte Classifier zur Erkennung von Gurken. Diese unterschiedlichen Ergebnisse könnten daher rühren, dass bei der Erstellung dieses Classifiers eine größere Zahl an verschiedenen Bananen verwendet wurden. Leider ist dieser Classifier etwas untertrainiert, was zu einer hohen Fehlerrate in den Negativtests führte. In den Negativtests wurden alle Objekte, die eine abgerundete Kontur hatten, fälschlicherweise als Banane erkannt. Auch dieser Classifier war erhöht abhängig von der Rotation der Banane. Sobald die Banane über einen bestimmten Winkel hinaus gedreht wurde, wurde sie nicht mehr als solche erkannt.

Der für das Erkennen von Gesichtern erstellte Classifier lieferte wie erwartet – mit 70 von 100 erkannten Objekten – sehr gute Ergebnisse. Dieser Classifier ist jedoch stark auf die Erkennung von Gesichtern aus der Frontalansicht spezialisiert. Aus diesem Grund wurden im Positivtest einige Gesichter, welche leicht geneigt waren, nicht erkannt. In den Negativtests gab es keine falsch positiven Befunde. Das Testen dieses Classifiers diente der Veranschaulichung der Tatsache, dass mit einem sorgfältig designten Set an Trainingsbildern und den richtigen Einstellungen ein sehr performanter Classifier erstellt werden kann.

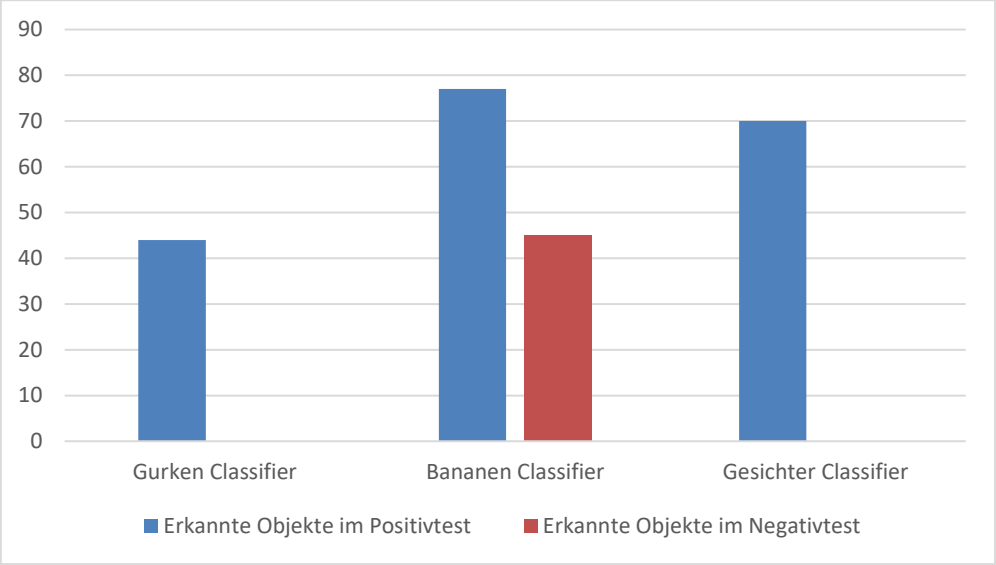

Abbildung 17 Testergebnisse der Classifierkaskade.

3.2.4 Testfazit

Wie schon den Testergebnissen entnommen werden kann, waren die beiden Classifier zur Erkennung von Lebensmitteln im Vergleich zum Classifier zur Erkennung von Gesichtern nicht optimal. Während Gurken nur in bestimmten Positionen erkannt wurden, wurden Bananen fälschlicherweise in vielen Objekten wiedererkannt. Dies lag jedoch an nicht ausreichendem oder falschem Training (z.B. zu stark verfälschte Objekte). Die guten Ergebnisse des Classifiers zur Erkennung von Gesichtern zeigt, dass die Nutzung von Haar-based Classifiern

generell geeignet ist, um natürliche Objekte zu erkennen. Beim Trainieren der Classifier sollte darauf geachtet werden, dass die Erkennungsrate in den Negativtests sehr gering ist. Der Fall, dass ein Objekt einmal nicht erkannt wird, ist weitaus weniger problematisch als der Fall, dass ein Objekt fälschlicherweise erkannt wird. Falls ein Objekt nicht erkannt wird, kann der Nutzer gebeten werden, das Objekt ein wenig zu rotieren oder ein verdeckendes Objekt zu entfernen. Sollte im Rahmen eines Prozesses, in welchem mehrere Objekte in einer klaren Reihenfolge erkannt werden müssen, ein Objekt fälschlicherweise erkannt werden, würde der Prozess fortgeführt werden und der Nutzer müsste sich selbst über den Fehler bewusst sein und diesen korrigieren.

3.3 Eingrenzen durch vorheriges Filtern der Objekteigenschaften

Sowohl bei der Objekterkennung via SIFT, SURF, ORB und AKAZE als auch bei der Objekterkennung durch Haar-like Features werden zwei für die Erkennung hilfreiche Merkmale nicht beachtet: die Größe und die Farbe des Objektes. Skalierungsinvarianz ist bei diesen Verfahren zwar gewollt, ist aber im Falle einer fest installierten Kamera und einer gleichbleibenden Distanz des Objektes zur Kamera nicht zwingend notwendig. Die folgenden beiden Verfahren sind also in ihrem Einsatz auf eben jenes Szenario mit einer festen Kamera und einer gleichbleibenden Distanz des Objektes zur Kamera angewiesen. Die in diesem Kapitel beschriebenen beiden Verfahren speichern jeweils die Größe und die Farbe mehrerer Testobjekte in einer Datenbank. Diese Datenbank kann dann bei der Klassifizierung von Objekten zu Rate gezogen werden. Dies kann auf zwei Arten geschehen: Falls beispielsweise eine Liste von Objekten existiert, welche nacheinander getestet werden sollen, müssen lediglich die Eigenschaften des zu testenden Objektes mit denen des Referenzobjektes aus der Datenbank verglichen werden. Dies wäre zum Beispiel beim Abarbeiten von Zutaten in einem Rezept der Fall. Sollte aber eine Klassifikation eines Objektes gefragt sein, ohne dass

eine Vorgabe besteht, um welches Objekt es sich handeln sollte, würde einfach in der Datenbank nach dem einen Objekt gesucht, dessen Eigenschaften am nächsten an denen des zu testenden Objektes liegen. Um die beiden Vorgänge zu testen, wurde hier das erste Szenario verwendet. Es wurde ein Referenzbild eines Objektes gemacht und ein entsprechender Eintrag mit den zugehörigen Eigenschaften in die Datenbank eingespeist. Anschließend wurden weitere Fotos von einer anderen Version des gleichen Objektes erstellt und die Eigenschaften mit denen der Datenbank abgeglichen.

3.3.1 Erfassen der durchschnittlichen Größe eines Objektes

3.3.1.1. Aufbau

Zuerst erfolgt eine Kalibrierung. Bei dieser wird ein Schneidebrett auf der Arbeitsfläche erkannt. Damit dies geschehen kann, muss das Bild durch Weichzeichnen, Entsättigen und Thresholding vorbearbeitet werden. Thresholding ist hier eine Technik, bei der alle Pixel des Bildes in Schwarz oder Weiß aufgeteilt werden. Hierzu werden die Pixel eines Graustufenbildes abhängig von ihrer Intensität in schwarz oder weiß aufgeteilt. Alle Pixel mit einem Wert zwischen Null (für Schwarz) und einem variablen Schwellenwert werden schwarz eingefärbt. Alle Pixel, deren Wert zwischen dem Schwellenwert und Eins liegt, werden weiß eingefärbt. Das Thresholding des Bildes findet statt, damit das Bild im folgenden Schritt nach

Abbildung 18 Korrigieren der Perspektive des Bildes anhand einer Unterlage mit bekannten Maßen. 1: Das Ursprungsbild. 2: Das Bild nach dem Thresholding. 3: Die Kontur des Bretts wurde erkannt (hier rot umrandet mit Beschriftung). 4: Das korrigierte Bild.

Konturen durchsucht werden kann. Da es sich um ein weißes Schneidebrett auf schwarzem Grund handelt, ist es in diesem Fall nicht nötig die Konturen nachzubearbeiten.
Um die Kontur des Bretts zu finden, wird einfach die größte Kontur herausgesucht, welche vier Ecken hat. Da in diesem Aufbau die Kamera nicht vollkommen akkurat ausgerichtet werden konnte, ist das Schneidebrett im Bild durch perspektivische Verzerrung leider nicht vollkommen rechteckig.

Dies wird durch Transformation des Bildes anhand der vier Eckpunkte des Schneidebretts korrigiert. Für diese Transformation werden die Längen beider waagerechten und beider senkrechten Kanten des Bretts berechnet und jeweils die längere der beiden als Höhe und Breite für das neue, korrigierte Bild verwendet. So entsteht ein Bild, welches nur das Schneidebrett abbildet. Um das anschließende Bestimmen der Größe der auf dem Brett befindlichen Gegenstände zu ermöglichen, werden Höhe und Breite des Bildes verwendet. Da das Bild durch das Kalibrieren nur noch das Schneidebrett enthält und die Maße des Schneidebretts bekannt sind, kann somit anhand der beiden Informationen eine Pixel-pro-Zentimeter-Rate errechnet werden. In Abbildung 18 wird dieser Prozess schrittweise mit Objekten auf dem Schneidebrett gezeigt. Dies dient jedoch nur zur Veranschaulichung. Um Fehlern in der Kalibrierungsphase vorzubeugen, wird dieser Prozess normalerweise mit dem leeren Schneidebrett durchgeführt.

Nach der Kalibrierung wird nun das Bild geladen, in welchem es Objekte zu erkennen gilt. In diesem Versuch war das Schneidebrett auf der Arbeitsfläche befestigt und somit unbeweglich. Dies ermöglichte es, die bei der Kalibrierung erhaltene Transformationsmatrix auf das zweite Bild anzuwenden, um eine Korrektur des Blickwinkels und gleichzeitig eine Eingrenzung der Region of Interest auf das Brett zu erreichen. Der Hintergrund des zu untersuchenden Bildes wird durch Erstellen eines Differenzbildes entfernt. Hierbei wird das im Kalibrierungsprozess erhaltenen Bild des leeren Schneidebretts von dem zu untersuchenden Bild abgezogen. In diesem Szenario wäre es nicht nötig gewesen den

Hintergrund zu entfernen, da mit einem sauberen, weißen Schneidebrett gearbeitet wurde, welches es sehr einfach macht Objekte im Bild zu finden. In späteren Anwendungsfällen könnte sich jedoch das Schneidebrett beispielsweise durch Kratzer oder dauerhafte Flecken verändert haben. Diese würden dann über die Hintergrundsubtraktion entfernt werden. Um nun die Größe der auf dem Brett befindlichen Gegenstände zu berechnen, werden über Thresholding des Differenzbildes die Umrisse der Objekte berechnet. Anschließend wird über die Umrisse iteriert. Für jeden Umriss, der einen gewissen Größenschwellwert überschreitet, wird ein ihn umfassendes Rechteck mit dem kleinstmöglichen Flächeninhalt berechnet. In diesem Test wurde der Größenschwellwert noch manuell eingestellt. Später sollte jedoch die Minimalgröße des kleinsten Objektes in der Objektdatenbank verwendet werden. Der Zweck des Größenschwellwertes ist es, dass zu kleine Objekte, wie Verschmutzungen auf der Schneidefläche oder Bildrauschen, aussortiert werden. Aus der Höhe und Breite des zugehörigen Rechtecks wird nun mit der vorher errechneten Pixel-zu-Zentimeter-Rate die Maße des Objektes errechnet.

Es wurde eine Lernfiktion zum Speichern der Eigenschaften von Objekten geschrieben. Dieser wird lediglich der Name des Objektes übergeben.

Dann wird das auf dem Schneidebrett befindliche Objekt ausgemessen und in der Objektdatenbank gespeichert, falls es noch keinen Eintrag gibt. Sollte schon ein Eintrag existieren, wird lediglich geprüft, ob das Objekt im aktuellen Bild größer als Maximalgröße oder kleiner als die Minimalgröße aller bisherig erlernten Versionen des Objektes ist.

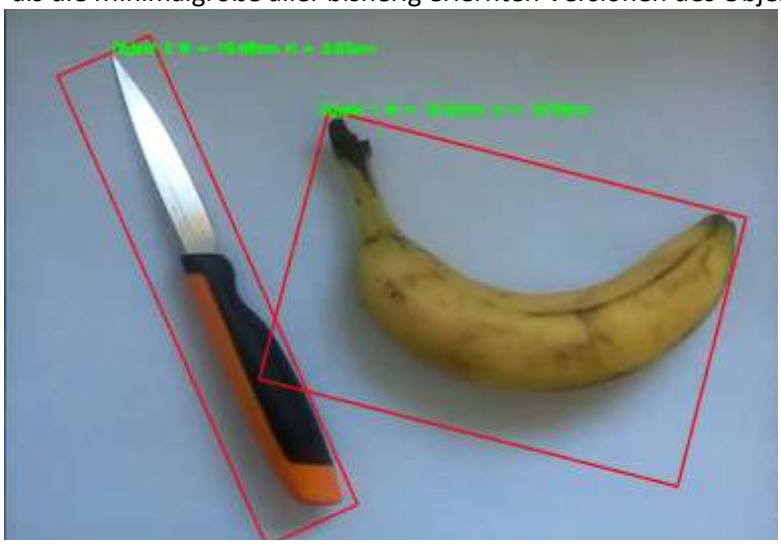

Abbildung 19 Debugausgabe nach dem Erfassen der Größe des Objektes. Hier ist gut zu sehen, dass bei unzureichender Ausleuchtung die Größe durch Wurfschatten beeinflusst werden kann.

Sollte dieser Fall eintreten, wird die Maximal- oder Minimalgröße entsprechend angepasst. Bei der Objektdatenbank handelt es sich aus Testgründen um eine JSON Datei. In späteren Tests könnte aber problemlos eine beliebige andere Datenbank eingesetzt werden.

Eine weitere Möglichkeit zur Ermittlung der Größe der Objekte ist das Verwenden einer Tiefenkamera wie zum Beispiel die Microsoft Kinect. Durch die Nutzung einer Tiefenkamera werden neben Informationen über Farbe auch ein Tiefenbild, welches Informationen über die Entfernung von Kamera zum Objekt enthält, gewonnen. Um die Größe des Objektes zu ermitteln, wird die Tiefenkamera – wie schon die normale Kamera – aus der Draufsicht auf das Schneidebrett gerichtet. Für die Erkennung der Umrisse würden alle Punkte bzw. Pixel, deren Abstand zur Kamera geringer als der Abstand zwischen Kamera und Schneidebrett ist,

verwendet. Durch diese Herangehensweise würden Objektschatten und Fehler durch Verunreinigungen auf dem Schneidebrett vernachlässigt. Dadurch müsste des Weiteren keine Entfernung des Hintergrundes stattfinden.

3.3.1.2. Testergebnisse

Alle der mit diesem Verfahren getesteten Objekte lieferten in mehreren Messungen konstante Größendaten mit minimaler Abweichung von ca. 5 Millimetern. Dies lag daran, dass, wie beispielsweise in Abbildung 19 zu sehen ist, Teile des Objektschattens in die Berechnung einbezogen wurden. Dies ließe sich durch bessere Ausleuchtung oder die Verwendung eines höheren Schwellenwertes beim Thresholding bewerkstelligen. Des Weiteren trat es in den Tests auf, dass Objekte, welche sich überschneiden, als ein größeres Objekt gesehen werden. Es wurde also nach dem Auftreten dieses Problems darauf geachtet, dass sich nur einzelne Objekte im Bild befanden.

3.3.2 Erfassen der durchschnittlichen Farbe eines Objektes

3.3.2.1. Aufbau

Die Vorgehensweise zum Erfassen der durchschnittlichen Farbe eines Objektes gleicht bis zum Tresholding des Differenzbildes der Vorgehensweise zum Erfassen der durchschnittlichen Größe eines Objektes. Im Kalibrierungsprozess wird ein perspektivisch korrigiertes Bild des leeren Schneidebretts erstellt. Mit diesem und einem zweiten Bild, in welchem sich das zu prüfende Objekt auf dem Brett befindet, wird ein Differenzbild erstellt. Aus diesem Differenzbild werden schließlich durch Thresholding die Umrisse des Objektes erstellt. Da das so erhaltene Bild keinerlei Farbinformationen enthält – es handelt sich um eine Maske, welche nur aus Schwarz und Weiß besteht – kann aus diesem natürlich keinerlei Informationen über die Farbe des Objektes gezogen werden. Das Bild wird lediglich als Maske für das perspektivisch korrigierte Bild des Objektes auf dem Schneidebrett verwendet. Somit werden alle Pixel, welche nicht zum Umriss des Objektes gehören, herausgefiltert. Was bleibt ist eine Liste mit Farbinformationen aller Pixel, welche sich innerhalb des Umrisses des Objektes befinden. Auch hier muss auf eine ausreichende Beleuchtung geachtet werden, da sonst Teile des weißen Schneidebretts, auf welche ein Schatten fällt, mit zum Umriss des Objektes gezählt werden. Das würde das Ergebnis der durchschnittlichen Farbe verfälschen. In diesem Versuch wurde mit immer gleichbleibenden Lichtverhältnissen gearbeitet. Da diese sich aber in der späteren Anwendung durchaus verändern können, sollte in der Kalibrierungsphase eine globale Anpassung der RGB-Werte stattfinden. Dies könnte beispielsweise wie in [Puri et al. 2009] mit Hilfe eines Markers, auf welchem sich mehrere farbige Quadrate befinden, die zur Kalibrierung der Farbwerte dienen, stattfinden. In der Objektdatenbank wird jeweils über ein Referenzfoto des Objektes der erste dominante Farbwert und ein Platzhalter für den maximalen zugelassenen euklidischen Abstand gespeichert. Der euklidische Abstand wird hierbei verwendet, um die Abweichung der dominanten Farbe des Referenzobjektes von der dominierenden Farbe des getesteten Objekts zu berechnen. Wenn ein Objekt getestet wird, kann somit über den euklidischen

Abstand festgestellt werden, ob die dominante Farbe des getesteten Objektes eine ähnliche wie die des Referenzobjektes ist. Über eine Lernfunktion kann der maximale euklidische Abstand an ein weiteres getestetes Objekt angepasst werden. Sollte beispielsweise eine Orange nicht entsprechend klassifiziert werden, weil sie eine zu dunkle Färbung besitzt, kann die maximale Farbabweichung über die Lernfunktion automatisch entsprechend dieser Orange angepasst werden.

Um aus der Liste von Farbinformation die durchschnittliche Farbe zu gewinnen, wurden in diesem Versuch zwei verschiedene Ansätze verfolgt. Beim ersten Ansatz handelt es sich um einfaches Berechnen des Durchschnittes der Rot-, Grün- und der Blauwerte. Beim zweiten Ansatz wird ein KNN-Clustering über die verschiedenen Farbwerte der einzelnen Pixel vorgenommen.

3.3.2.2. Einfacher Durchschnitt

In dieser einfachen Methode wird lediglich über alle Pixel des Objektes iteriert und jeweils alle Rot-, Blau- und Grünwerte aufaddiert und anschließend durch die Gesamtzahl der Pixel im Objekt geteilt. So entsteht ein Farbdurchschnittswert aller Pixel im Objekt.

3.3.2.3. Verwenden von KNN-Clustering

Diese Methode orientiert sich einer Anleitung aus [Rosebrock2014]. In diesem Ansatz werden alle Pixel des Objektes in eine beliebige Anzahl an Clustern eingeteilt. Dieses Prinzip lässt sich am besten durch die Einordnung der Pixel in ein dreidimensionales Koordinatensystem, in dem die drei Achsen jeweils für den Rot-, Grün- und Blauwert des Pixels stehen, erläutern. Pixel mit einer ähnlichen Farbe befinden sich in einem bestimmten Bereich des Diagramms und können zu einem Cluster zusammengefasst werden. Für jedes Cluster wird die Farbe ausgegeben, welche den geringsten Abstand zu den Pixeln des Clusters

hat. Diese Herangehensweise hat den Vorteil, dass nicht nur eine durchschnittliche Farbe berechnet wird. So können Objekte mit mehreren stark unterschiedlichen Farben, wie zum Beispiel eine Paprika, besser bearbeitet werden. Ein weiterer Vorteil dieser Methode ist, dass sie es möglich macht einen Teil des Schattens herauszufiltern. Hierzu muss einfach der Cluster, dessen Pixel alle eine ähnliche Farbe wie die der Unterlage besitzen, aussortiert werden.

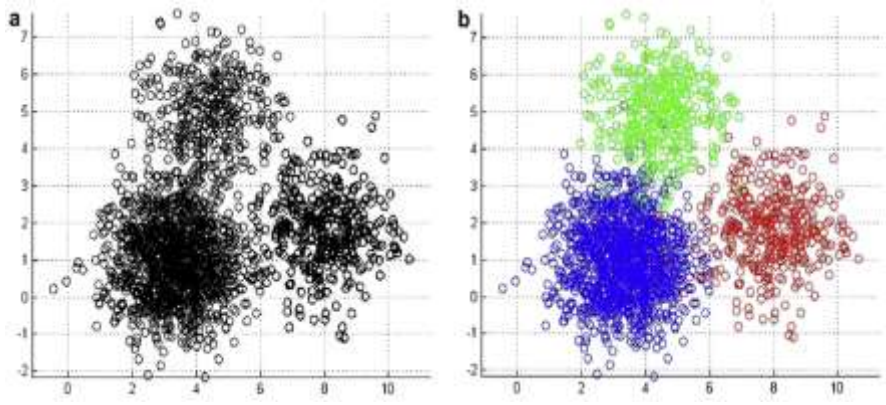

Abbildung 20 Veranschaulichung des K-Means Algorithmus im zweidimensionalen Raum. A: Unsortierte Werte im zweidimensionalen Raum. B: Die Werte wurden entsprechend ihrer Lage zueinander mit Hilfe des K-Means Algorithmus in Cluster eingeteilt. Dies ist hier durch die Einfärbung deutlich gemacht worden. [Piazza et al. 2011] S.1065

3.3.3 Testergebnisse

3.3.3.1. Normaler Durchschnitt

Beim einfachen Berechnen des Durchschnittes war es sehr wichtig auf eine gute Beleuchtung der Szenerie und des Objektes zu achten, da sich starke Schatten auf dem Objekt oder ein starker Schattenwurf auf das Schneidebrett direkt auf die berechnete Farbe auswirken. Wenn beispielsweise durch schlechte Beleuchtung ein starker Wurfschatten auf das

Schneidebrett fällt, wird dieser eventuell mit zum Objekt gezählt. Dies verändert die durchschnittliche Farbe je nach Größe des Schattens ins weiß-gräuliche. Abbildung 21 zeigt vier Ergebnisse der Berechnung der Durchschnittsfarbe. Bei den jeweils beiden Quadranten, welche waagrecht nebeneinanderliegen, handelt es sich um zwei verschiedene Versionen des gleichen Objektes. Die zwei Orangen lieferten in diesem Versuch ein sehr gutes Ergebnis, welches sich dadurch auszeichnet, dass die Rot-, Grün- und Blauwerte jeweils sehr gering voneinander entfernt sind. Im Testergebnis der zwei getesteten Tomaten ist der Unterschied weitaus präsenter. Dies lässt sich darauf zurückführen, dass die erste Tomate näher am Rande des Schneidebretts lag und somit weniger beleuchtet wurde. Trotz der stärkeren Farbunterschiede der beiden Tomaten ist dennoch ersichtlich, dass es sich um zwei verschiedene Rottöne handelt.

Abbildung 21 Vier Farbwerte von zwei Objektpaaren, welche durch das Berechnen des einfachen Durchschnitts erhalten wurden.

3.3.3.2. Unter Nutzung von K-Means Clustering

Die Ergebnisse der Berechnung der durchschnittlichen Farben mit Hilfe des K-Means-Algorithmus zeigen einige Vorteile gegenüber der Berechnung des einfachen

Farbdurchschnitts. In allen der vier in Abbildung 22 abgebildeten Messungen wurden jeweils drei Cluster verwendet, da dies sich im Rahmen der Tests als die am besten funktionierende Zahl an Clustern für Objekte, die größtenteils einfarbig sind, erwiesen hat. Bei Objekten mit eindeutig getrennten Farben, wie zum Beispiel eine Paprika mit grünen Strunk, wäre es natürlich möglich mehr als drei Cluster zu verwenden, um zu vermeiden, dass der grüne Strunk und das Rot der Paprika zu einer Farbe zusammengefasst werden. Die Anzahl der Cluster muss hierbei manuell ausprobiert werden. Ist aber eine Anzahl gefunden, welche zu einem zufriedenstellenden Ergebnis führt, kann diese in der Objektdatenbank für künftige Lern- und Erkennungsprozesse gespeichert werden. An den vier Balken in Abbildung 22 lässt sich gut erkennen, dass die weiß-gräulichen Pixel, welche durch den Schatten des Objektes entstanden sind, einem eigenen Cluster zugeordnet werden. Dies macht es im Gegensatz zur normalen Farbdurchschnittsberechnung möglich, den Schatten fast vollkommen herauszufiltern. Die zwei weiteren Farben eines jeden Balken in der Abbildung beziehen sich jeweils auf die gut ausgeleuchteten und die weniger gut ausgeleuchteten Bereiche des Objektes. Die beiden getesteten Orangen unterscheiden sich sowohl in der Größe des Schattens als auch in den durchschnittlich berechneten Farben kaum. Dies zeigte sich schon bei der Berechnung der durchschnittlichen Farbe mit Hilfe des einfachen Durchschnitts (siehe Abbildung 21). Bei den beiden getesteten Tomaten jedoch besteht ein größerer Unterschied, welcher sich in Form eines etwas größeren Schattens und mehr dunklen bzw. schlecht beleuchteten Bereichen bei Tomate 1 äußert. Diese Unterschiede sorgten bei der vorhergegangenen Methode zur Messung noch für ein weitaus unterschiedliches Ergebnis. Es ist aber ersichtlich, dass die beiden Durchschnittsfarben der jeweils größten Cluster dennoch sehr ähnlich sind. Aus diesem Grund wurde in diesem Versuch jeweils die Durchschnittsfarbe des größten Clusters gewertet. Dieses Vorgehen macht es sehr wichtig auch hier auf eine sehr gute Beleuchtung des zu testenden Objektes zu achten, da sonst eventuell der Objektschatten den größten Cluster ausmachen könnte. Dies würde zu stark verfälschten Ergebnissen führen. Eine weitere Methode, um den Schatten herauszufiltern, ist es, einen Cluster nicht zu werten, wenn seine Durchschnittsfarbe zu nah an die des

Schneidebretts kommt. Dies würde es aber wiederum notwendig machen, dass kein Objekt mit ähnlicher Farbe des Schneidebretts verarbeitet wird.

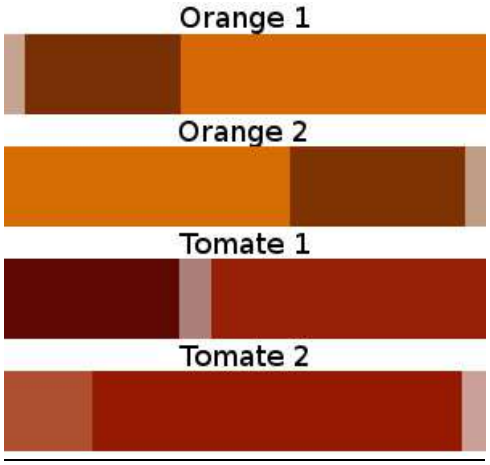

Abbildung 22 Abbildung der jeweils drei dominantesten Farben von jeweils zwei Versionen von zwei Objekten. Hier ist gut zu sehen, dass der Wurfschatten ebenfalls einen kleinen Custer hat.

3.3.4 Testfazit

Sowohl die Farbe als auch die Größe der Objekte sind im Zusammenhang mit einer festen Kameraposition und gleichbleibender Beleuchtung sehr gute Merkmale für die Erkennung natürlicher Objekte. Die beiden Vorgänge zur Berechnung der durchschnittlichen Farbe eignen sich demnach gut zur Eingrenzung des Pools an möglichen Objekten. Natürlich kann nicht immer genau ein Objekt durch die Farbe ermittelt werden, da schließlich mehrere Objekte im Pool der möglichen Objekte vorkommen können, welche eine sehr ähnliche Farbe besitzen. Um die Gruppe der möglichen Objekte einzugrenzen, ist es jedoch völlig ausreichend. Im Falle eines Objektpools, in welchem sich nur Objekte mit hinreichend unterschiedlicher Farbe befinden, wäre dieses Vorgehen sogar für eine Klassifizierung der Objekte geeignet. In Kombination mit der Objektgröße als zu prüfende Eigenschaft könnte sogar ein Objektpool mit teilweise gleichfarbigen Objekten unterschieden werden, solange

die gleichfarbigen Objekte in ihrer Größe stark genug abweichen. Jedoch ist dieses Verfahren, wie auch das Verfahren zur Berechnung der Größe eines Objektes, fehleranfällig in Bezug auf Schatten. Aus diesem Grund sollte immer auf eine gute, gleichbleibende Beleuchtung ohne starke Wurfschattenbildung geachtet werden.

3.4 Test von Convolutional Neural Networks

In diesem Versuch werden drei verschiedene Convolutional Neural Networks auf ihre Eignung zur Erkennung von natürlichen Gegenständen, in diesem Fall Obst und Gemüse, getestet. Hierbei wird Python 3.5 mit Keras [Chollet et al. 2015] und dem Tensorflow Backend [Abadi et al. 2015] verwendet, um die folgenden, vortrainierten Convolutional Neural Networks zu testen.

3.4.1 VGG16

Bei diesem Convolutional Neural Network handelt es sich um ein aus 16 „Weight Layern“ bestehendes Netzwerk, welches im Rahmen der „ImageNet Large Scale Visual Recognition Challenge“ [Russakovsky et al. 2015] erstellt und trainiert wurde. Dieses CNN ist ein tiefes Convolutional Neural Network bestehend aus 13 Convolutional Layern und drei fully-connected Layern. In der ILSVRC 2014 [Russakovsky et al. 2014] erreichte die Visual Geometry Group des Department of Engineering Science der University of Oxford unter anderem mit diesem Netzwerk mehrere, hohe Platzierungen. Für weitere Informationen bezüglich des Aufbaus und Trainings dieses Netzes wird an dieser Stelle auf das 2014 erschienene Paper mit dem Titel „Very deep convolutional networks for large-scale image recognition“ [Simonyan et al. 2014] verwiesen.

3.4.2 VGG19

Beim VGG19 handelt es sich um ein weiteres Convolutional Neural Network, welches von der Visual Geometry Group des Departments of Engineering Science der University of Oxford erstellt wurde. Wie schon der Name vermuten lässt, beinhaltet dieses Netz 19 Weight Layer bestehend aus 16 Convolutional Layern und drei fully-connected Layern und ist damit ebenfalls ein sehr tiefes Netz. Auch das VGG19 Netz schnitt in Kombination mit anderen Convolutional Neural Networks der Visual Geometry Group bei der ILSVRC 2014 sehr gut ab. Für weitere Informationen bezüglich des Aufbaus und Trainings dieses Netzes soll auch hier auf das 2014 erschienene Paper mit dem Titel „Very deep convolutional networks for large-scale image recognition“ verwiesen werden.

3.4.3 ResNet50

Das ResNet 50 ist ein vom Microsoft Research Asia Team erstelltes „Deep Residual Network“ [He2015]. Bei einem Deep Residual Network handelt es sich um eine Sonderform des Convolutional Neural Networks, die es ermöglicht Netzwerke erfolgreich zu trainieren, welche laut [He et al. 2015] wesentlich tiefer sind, als die bisherigen Convolutional Neural Networks. Trotz ihrer extremen Tiefe verhalten sich Deep Residual Networks laut dem 2016 erschienenen Paper mit dem Titel „Residual Networks Behave Like Ensambles of Relatively Shallow Networks“ [Veit et al. 2016] aber eher wie ein Zusammenschluss mehrerer flacher Netzwerke. Mit unter anderem dem ResNet 50 erreichte das MSRA-Team im ILSVRC 2015 [Russakovsky et al. 2015] den ersten Platz in der Kategorie „Classification“.

Alle drei Netze wurden basierend auf dem Imagenet-Set trainiert. Dieses Set beinhaltet, wie in [Russakovsky et al. 2015] zu lesen ist, 1,2 Millionen Bilder, welche in 1.000 Kategorien eingeteilt sind. Durch dieses Training ist jedes der drei Netze fähig 1.000 verschiedene Objekte zu unterscheiden. Diese Objekte sind sowohl natürlich, wie zum Beispiel die Objekte

aus diesem Test, als auch nicht natürlich wie beispielsweise ein U-Boot oder ein Einkaufswagen.

3.4.4 Aufbau

In diesem Testaufbau wird jedes der drei Convolutional Neural Networks in der Klassifikation von jeweils vier verschiedenen Objekten (Paprika, Brokkoli, Banane und Gurke) getestet. Hierzu werden jedem Netzwerk pro zu klassifizierendem Objekt 50 Bilder übergeben. Bei den Bildern handelt es sich um handverlesene Bilder mit unterschiedlichen Orientierungen, Positionen und Verdeckungsgraden. Es wurde darauf geachtet, dass die Objekte auf den Bildern in ihrer Ursprungsform abgebildet sind. Beispielsweise wurde vom Testen gewürfelter Paprika abgesehen, da diese andere Features als das Ursprungsobjekt aufweisen würde. Jedes Netz gibt für jedes überprüfte Bild eine Liste mit Prognosen über den Inhalt des Bildes ab. Von diesen Prognosen werden lediglich die ersten fünf gewertet. Sollte sich ein Objekt außerhalb der ersten fünf Plätze befinden, wird die Erkennung als fehlgeschlagen gesehen. Jede Prognose beinhaltet eine Prozentzahl, welche die Wahrscheinlichkeit, mit der es sich um das jeweilige Objekt handelt, widerspiegelt. Die Positionsbestimmung der Objekte wird in diesem Test außer Acht gelassen.

3.4.5 Testergebnisse

Alle drei Netzwerke erbrachten sehr gute Resultate, was sich auch in den Ergebnissen der „Imagenet Large Scalle Visual Recognition Challenge 2014“ [Russakovsky et al. 2014] und „Imagenet Large Scalle Visual Recognition Challenge 2015“ [Russakovsky et al. 2015] widerspiegelt.

3.4.5.1. VGG16

Das VGG16 Netz erkannte 185 der insgesamt 200 Bilder aus vier Kategorien mit der ersten Vermutung. Weitere 12 Bilder wurden ebenfalls erkannt, jedoch nicht auf Platz eins der Vermutungen. Lediglich drei der 200 überprüften Bilder befanden sich außerhalb der ersten fünf Vermutungen und wurden somit als „fehlerhaft klassifiziert“ notiert. Die Bilder, welche beim ersten Versuch erkannt wurden, wurden mit einer durchschnittlichen Wahrscheinlichkeit von 85,6% erkannt. Die Wahrscheinlichkeiten schwanken hierbei jedoch abhängig von der getesteten Kategorie von durchschnittlich 77% für den ersten Platz bei Gurken bis hin zu durchschnittlich 95% bei der Klassifikation von Bananen. Weitaus niedriger sind hier die durchschnittlichen Wahrscheinlichkeiten bei Klassifikationen innerhalb der ersten fünf Vermutungen. Die durchschnittliche Trefferwahrscheinlichkeit der Objekte innerhalb der Top5 (d.h. Objekte, die erst mit dem 2. - 5. Vorschlag korrekt klassifiziert wurden) war beim VGG16 mit 9,8% sehr gering.

3.4.5.2. VGG19

Das VGG19 CNN konnte in diesem Test 182 von 200 Testbildern beim ersten Versuch klassifizieren. Die durchschnittliche Wahrscheinlichkeit belief sich hierbei auf 88%. Damit liegt es, was die Zahl der Bilder angeht, welche beim ersten Versuch korrekt klassifiziert wurden, mit drei Bildern hinter dem Resultat des VGG16-Netzes. Im Gegensatz zum VGG16 konnte hier aber die durchschnittliche Wahrscheinlichkeit, mit der die Testbilder beim ersten Versuch klassifiziert wurden, um 2,4% gesteigert werden. 14 der 200 Bilder wurden innerhalb der ersten fünf Prognosen korrekt klassifiziert. Auch bei diesen Top5-Treffern lagen die Wahrscheinlichkeiten mit durchschnittlich 16,8% höher als beim VGG16. Die Anzahl an korrekten Klassifikationen war beim VGG19, was die einzelnen Kategorien angeht, recht gleichmäßig verteilt. Nur bei der Kategorie der Gurke wurden mehr Bilder erst in den Top5-Vorschlägen korrekt klassifiziert.

3.4.5.3. ResNet50

In diesem Test erzielte das ResNet50 die besten Resultate. Von 200 Testbildern wurden 191 bereits beim ersten Versuch erkannt. Die durchschnittliche Wahrscheinlichkeit lag hierbei bei 90,2%. Zusätzlich zu den 191 erkannten Bildern wurden sieben innerhalb der ersten fünf Versuche mit einer durchschnittlichen Wahrscheinlichkeit von 18,8% klassifiziert. Damit dieses Netzwerk sowohl das VGG16 als auch das VGG19 übertroffen. Hierbei ist jedoch zu beachten, dass dieses Netz neuer als die beiden vorherigen Netze ist und somit beim Aufbau und Training von vorhergehenden Erkenntnissen profitieren konnte. Dieses Ergebnis zeigt also weniger den Vorteil von einem Netz gegenüber einem anderen, sondern dass im Laufe der Zeit immer größere Fortschritte im Bereich der CNN gemacht werden. Dies zeigt sich auch durch die immer geringer werdende Fehlerrate bei der ILSVRC im Laufe der Jahre.

	VGG19	VGG16	ResNet50
banana23.jpg	0	2	1
banana39.jpg	4	0	1
cucumber22.jpg	2	2	1
cucumber32.jpg	1	1	2
cucumber36.jpg	4	5	3
cucumber34.jpg	2	1	1
cucumber41.jpg	4	1	1
cucumber46.jpg	3	3	1
cucumber14.jpg	1	1	2
broccoli17.jpg	3	3	1
broccoli14.jpg	0	4	1
broccoli26.jpg	1	2	1
broccoli36.jpg	2	2	1
broccoli44.jpg	4	0	0
broccoli16.jpg	3	3	2
broccoli5.jpg	2	1	1
bell_pepper30.jpg	1	1	2
bell_pepper10.jpg	0	1	1
bell_pepper15.jpg	2	3	1

Abbildung 23 Unterschiede bei der Klassifizierung zwischen den drei Netzen basierend auf der Platzierung der korrekten Klassifizierung. 0 steht hierbei für eine fehlgeschlagene Klassifikation.

3.4.6 Fazit

Die Testergebnisse zeigen, was der aktuelle Trend bestätigt. Jedes der drei getesteten Convolutional Neural Networks ist sehr gut zur Erkennung von natürlichen Objekten geeignet. Die Bilder, welche in diesem Test verwendet wurden, zeigten zum Teil komplexe Abbildungen des zu klassifizierenden Objektes. In einigen Anwendungen im realen Umfeld ist eine Klassifikation unter solchen komplexen Umständen gar nicht notwendig, da zum Beispiel beim Überwachen von Arbeitsvorgängen per Kamera, wie es im Kapitel 4.2 beschrieben ist, die Kamera meist stationär und die Orientierung des Objektes gleichbleibend ist. Diese vereinfachten Voraussetzungen dürften für noch bessere Ergebnisse sorgen. Dies zeichnete sich auch ansatzweise in den Ergebnissen dieses Tests ab. Die Bilder, auf welchen das Objekt mit einer einfachen Draufsicht und einem statischen Hintergrund zu sehen war, wurden sehr gut erkannt. Bei den wenigen Bildern, welche nicht klassifiziert werden konnten, handelte es sich fast immer um Bilder mit einem komplexeren Hintergrund oder einer Anordnung von mehreren Objekten. Ein großer Vorteil der Convolutional Neural Networks, der sich auch in diesem Test zeigte, ist der, dass nicht für jedes zu klassifizierende Objekt ein Netzwerk erstellt und trainiert werden muss, auch wenn dies möglich wäre. Die Netze decken eine breite Menge an Objekten ab und können bei Bedarf sogar noch für weitere Objekte trainiert werden. Bei dem Trainieren von zusätzlichen Objekten profitieren CNN überdies von den bereits vorher erlernten Features und Gewichten. Dies macht sie sehr gut erweiterbar. Ein weiterer Vorteil ist, dass CNN nicht nur basierend auf zweidimensionalen Bildern, sondern, wie in [Qi et al. 2016], auch mit dreidimensionalen Darstellungen trainiert werden können. Somit könnte beispielsweise mit einer Tiefenkamera ein Objekt durch seine Oberfläche erkannt werden. Im Rahmen der Recherche konnte auch eine rapide Verbesserung der Convolutional Neural Networks in den letzten Jahren festgestellt werden. Dies lässt sich beispielsweise an den Ergebnissen der ILSVRC erkennen. Wo im Jahre 2014 der Sieger der Kategorie „Object detection with provided training data“ 106 von 200 Objektkategorien gewann [ILSVRC2014], gewann der Sieger der ILSVRC 2015 [Russakovsky et al. 2015] schon 194 der 200 Objektkategorien. Dieser rapide Fortschritt lässt auf weitere

Entwicklungssprünge im Bereich der CNN hoffen, welche sie zu einer noch besseren Lösung für die Klassifizierung von natürlichen Objekten machen würde.

4 Fazit

4.1 Gesamtfazit

In diesem Abschnitt wird basierend auf den vorhergegangenen Testergebnissen ausgewertet, welche der getesteten Methoden sich am besten für das Erkennen natürlicher Objekte in Bildern eignet. Beim Erkennen bzw. Klassifizieren von natürlichen Objekten führte die Nutzung von SIFT, SURF, ORB und AKAZE in Verbindung mit Brute-Force Matching eher zu einem unbefriedigenden Ergebnis. Durch das teilweise stark unterschiedliche Auftreten verschiedener Versionen des gleichen Objektes konnten alle vier Algorithmen ihre Stärken nicht gut ausspielen. Bei den Tests zeigte sich aber ebenfalls, dass die vier Algorithmen dennoch sehr geeignet für das Erkennen und Tracken von nicht natürlichen Objekten sind. Sobald das Objekt genügend Features durch zum Beispiel einen Aufdruck oder ähnliches bot, ließ es sich von allen vier Algorithmen gut erkennen. Aus diesem Grund sind die vier Algorithmen weniger zur Erkennung natürlicher Objekte geeignet, dennoch sollte nicht ausgeschlossen werden, dass sie in einer Endanwendung zum Einsatz kommen. Sie könnten beispielsweise in Bezug auf ein Kochszenario zum Erkennen von nicht natürlichen Objekten wie Salzstreuer oder zur Bestimmung der Kameraorientierung genutzt werden. Die beiden in diesem Paper implementierten Verfahren zur Erfassung der durchschnittlichen Farbe und Größe des Objektes sind leider in ihrer Nutzung sehr stark auf ein Szenario beschränkt. Dadurch, dass diese Verfahren an eine feste Kameraposition und eine gleichbleibenden Beleuchtung gebunden sind, ist das Einsatzgebiet eingeschränkt. Des Weiteren ist es nur mit einem sehr kleinen Pool an Objekten möglich die Unterscheidung zwischen zwei Gegenständen lediglich über Größe und Farbe zu fällen. Beide Methoden könnten genutzt werden, um den Objektpool zu verkleinern, sind aber – wie sich durch die Tests im Rahmen

dieser Arbeit zeigte – eher ungeeignet, um zwischen einem größeren Pool an natürlichen Objekten zu unterscheiden. Die Erkennung von natürlichen Objekten durch die Nutzung von Haar-Kaskaden war das erste Verfahren, welches im Rahmen der Tests Ergebnisse lieferte, welche es zu einem geeigneten Kandidaten zur Implementierung einer Anwendung zur Erkennung natürlicher Objekte machen. Hierbei war das Ergebnis jedoch stark vom Training der Kaskade abhängig. Wie schon in Kapitel 3.2.4 beschrieben, lieferte eine professionell trainierte Haar Kaskade ein weitaus besseres Ergebnis als eine in kurzer Zeit und mit einem kleinen Trainingsset erstellte Haar Kaskade. Dennoch hat sich in den Tests herausgestellt, dass die Erkennung von natürlichen Objekten mit Hilfe von Haar Kaskaden nicht die beste Herangehensweise ist. Ein weiterer Prozess hat sich durch seine zahlreichen Vorteile, wie unter anderem eine sehr hohe Erkennungs- und niedrige Fehlerrate, als die vielversprechendste Lösung im Rahmen der Testszenarien herausgestellt: Convolutional Neural Networks. Die Nutzung von professionell trainierten CNN hat sich in den hier angewandten Testszenarien als äußerst effektiv gezeigt. Die getesteten CNN haben es geschafft trotz einem Pool von 1.000 Objekten mit einer sehr hohen Trefferwahrscheinlichkeit den Großteil der Objekte korrekt zu klassifizieren.

Alle der überprüften Verfahren sind definitiv auch heute noch in einigen Variationen nützlich. Die Nutzung von Convolutional Neural Networks zeigte sich zwar in den in diesem Paper durchgeführten Tests am geeignetsten für das Erkennen natürlicher Gegenstände, deckt dabei aber nur eines der vielen Themengebiete der Computer Vision ab, welche für die Umsetzung einer menschenähnlichen, visuellen Wahrnehmung nötig sind. Wie schon in Kapitel 2.4 beschreiben, finden viele der Verfahren auch heute noch Anwendung, da sie für bestimmte Aufgaben, wie beispielsweise das Verfolgen von Objekten, vollkommen ausreichend sind. Der starke Erfolg der CNN, welcher sich dieser Tage verzeichnen lässt, wird nicht das Ende für SURF, ORB und Co sein, selbst wenn CNN einen immer größeren Platz in dem Bereich der Computer Vision einnehmen. Für die Umsetzung des menschlichen Sehens

und Verstehens wird jedoch wahrscheinlich nie nur ein Prozess reichen. Für eine so komplexe Aufgabe wird vermutlich immer eine Kombination vieler verschiedener Verfahren nötig sein.

4.2 Ausblick

In diesem Kapitel soll kurz ein beispielhaftes Projekt vorgestellt werden, welches die im Laufe dieser Arbeit gewonnenen Erkenntnisse nutzt. Wie schon in der Einleitung beschrieben, ist die Idee eine Anwendung zu erstellen, welche den Prozess des Kochens unterstützt und dabei eine möglichst menschenähnliche Interaktion bietet. Das heißt, die Interaktion soll weniger durch Tasteneingaben oder ähnliches stattfinden, sondern durch das Verarbeiten audiovisueller Reize. Die Anforderungen an eine solche Anwendungen wären wie folgt:

Das System sollte über eine Kamera verfügen, welche direkt auf die Arbeitsfläche der Küche gerichtet ist. Diese Anforderung ist notwendig, um die korrekte Nutzungsreihenfolge der Zutaten bezüglich des Rezeptes zu überprüfen. Zusätzlich sollte noch ein Raummikrofon installiert sein, um die Spracheingabe des Nutzers unabhängig von seiner Position im Raum aufnehmen zu können. Um eventuell noch Verarbeitungsprozesse wie beispielsweise das Schneiden von Zutaten zu überwachen, sollte ebenfalls Küchenwerkzeug, welches mit entsprechenden Sensoren zur Erfassung der Orientierung und Bewegung des Werkzeuges ausgestattet ist, vorhanden sein. Die Software sollte den Nutzer mit Hilfe der Eingaben durch die vorhandenen Sensoren durch den Zubereitungsvorgang eines Rezeptes leiten können und dabei eine möglichst menschenähnliche Interaktion bieten. Anweisungen seitens des Nutzers, wie zum Beispiel die Auswahl eines Rezeptes, würden hierbei als Sprachbefehl entgegengenommen. Für eine Spracherkennung könnte einer der zahlreichen bereits existierenden Implementierungen genutzt werden. Anweisungen seitens der Software würden über eine Sprachausgabe ausgegeben. Auch hier existiert bereits eine große Bibliothek an Software, welche dies ermöglicht. Für jeden Schritt sollte das Programm dem

Nutzer via Sprachausgabe die entsprechenden Anweisungen geben und die Durchführung dieser Anweisungen via Kamera und wahlweise des mit Sensoren ausgestattete Werkzeug überwachen. Im Falle eines Fehlers bei der Durchführung des jeweiligen Arbeitsschrittes würde dem Nutzer via Sprachausgabe Rückmeldung gegeben. Die Überwachung der Durchführung ist der Teil des Projektes, bei dem das Wissen aus dieser Arbeit Anwendung findet. Hierfür bietet sich ein Convolutional Neural Network mit einer großen Zahl an erkennbaren Objekten, vergleichbar den Netzwerken, wie sie schon im Rahmen der ILSVRC [Russakovsky et al. 2015] erstellt wurden, an. Die Nutzung eines CNN macht es einfach die Datenbank der bekannten Zutaten zu erweitern, indem das bestehende Netzwerk einfach die neue Zutat anhand von Referenzfotos lernt. Hierbei würde das CNN lediglich dafür genutzt werden, um zu überprüfen, ob gerade die dem Verarbeitungsschritt entsprechende Zutat verarbeitet wird. Der rapide Fortschritt im Bereich der Convolutional Neural Networks lässt darauf schließen, dass es in Zukunft möglich sein wird, dass das CNN den Übergang vom unverarbeiteten Objekt hin zum verarbeiteten Objekt erkennt und somit beispielsweise bestätigen könnte, ob das Objekt korrekt verarbeitet wurde. Bis dies ohne größere Fehler möglich ist, sollte die Überwachung der Verarbeitung des Objektes mit dem mit Sensoren bestücken Küchenwerkzeug durchgeführt werden. Hierbei würde durch Auswertung der Messdaten der Sensoren geprüft, ob die richtige Technik für beispielsweise das Zerkleinern des Objektes verwendet wird.

Die Erkenntnisse, welche im Rahmen dieser Arbeit gesammelt wurden, lassen vermuten, dass die Umsetzung einer solchen Anwendung mit samt des dazugehörigen Systems mit dem heutigen Wissensstand erreichbar ist. Es ist durchaus möglich eine menschenähnliche Interaktion umzusetzen. Das Stichwort ist hierbei jedoch "ähnlich". Bis der Stand der Technik so weit ist, dass das menschliche Sehen und das damit verbundene Verstehen eins zu eins nachempfunden werden kann, werden zweifelsohne noch einige Jahre Entwicklung nötig sein. Einige der Techniken, welche in dieser Arbeit vorgestellt und untersucht wurden, finden jedoch schon heute ihre Anwendung im Bereich Smart Home. Im Frühjahr 2017 bringt Google

beispielsweise sein Google Home auf den deutschen Markt. Hierbei handelt es sich um einen intelligenten Lautsprecher mit eingebautem Google Assistant. Dieser nutzt unter anderem die in dieser Arbeit vorgestellten Neuronalen Netze, um verschiedene Nutzer an ihrer Stimme zu unterscheiden [Welch2017]. Der Google Assistant, auf welchem Google Home basiert, wird in der momentanen Version nur über Spracheingabe gesteuert, ist aber – was diese angeht – schon recht intelligent und kann kontextbasierte Unterhaltungen führen. Ein weiteres Projekt von Google, welches teilweise Techniken aus dieser Arbeit verwendet, ist die Google Vision API. Mit Hilfe dieser API können Bilder an die Server von Google gesendet, dort in Sekunden untersucht und die dabei erhaltenen Informationen bezüglich des Inhaltes zurückgesendet werden. Die Informationen reichen von Objekten, die sich im Bild befinden, über Text, der aus dem Bild gelesen werden konnte, bis hin zu Emotionen, die über den Gesichtsausdruck von Personen im Bild abgelesen werden. Um diese schnellen und genauen Ergebnisse liefern zu können, setzt auch Google auf Machine Learning [Grief2016]. Es sind also schon definitiv Techniken vorhanden, welche stark in eine Richtung gehen, die eine annähernd menschliche Interaktion (sowohl verbal, als auch visuell) in naher Zukunft möglich erscheinen lässt. Gerade bei Konzernen wie Google scheint es nur eine Frage der Zeit zu sein, bis die schon bestehenden Services wie der Google Assistant und die Google Vision API miteinander verbunden werden, um eine bessere und einfachere Interaktion auf audiovisueller Basis zu schaffen. Die in dieser Arbeit behandelten Themen decken nur einen kleinen Teil von dem ab, was nötig wäre, um eine menschliche Interaktion zu implementieren. Um ein Programm mit einer Künstlichen Intelligenz zu schaffen, müssten noch zahlreiche Themen wie das menschliche Lernen oder das automatische Verknüpfen von Zusammenhängen gelöst werden. Eine solche Anwendung wäre zweifelsohne sehr nützlich für alle Bereiche, in denen Menschen mit Maschinen interagieren, wie zum Beispiel dem Smart Home. In diesem Fall gäbe es jedoch neben den Fragen zur Umsetzung noch eine Reihe an Problemen zu klären, welche den Umgang zwischen Mensch und Maschine betreffen. Es bliebe beispielsweise noch offen, inwiefern sich die Menschen auf eine solche Interaktion

mit einer Maschine einlassen würden. Dies sind jedoch Punkte, welche erst geklärt werden können, wenn eine solche Anwendung bereits erstellt wurde.

Literaturverzeichnis

[Aizawa et al. 2014] **AIZAWA, Kiyoharu; MAEDA, Kazuki; OGAWA, Makoto; SATO, Yohei; KASAMATSU, Mayumi; WAKI, Kayo; TAKIMOTO, Hidemi:** Comparative study of the routine daily usability of foodlog a smartphone-based food recording tool assisted by image retrieval. In: *Journal of diabetes science and technology 8* (2014), Nr. 2, S. 203–208 – URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4025617/ – Zugriffsdatum: 2017-05-10

[Alcantarilla et al. 2012] **ALCANTARILLA, Pablo Fernández ; BARTOLI, Adrien ; DAVISON, Andrew J.:** KAZE Features. In: *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*. Berlin, Heidelberg : Springer-Verlag, 2012 (ECCV'12), S. 214–227. – URL http://dx.doi.org/10.1007/978-3-642-33783-3_16. – ISBN

978-3-642-33782-6
– Zugriffsdatum: 2017-05-10

[Alcantarilla et al. 2013] **ALCANTARILLA, P. F. ; NUEVO, J. ; BARTOLI, A.:** Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces. In: *Proceedings of the British Machine Vision Conference (BMVC)*, BMVA Press, 2013 – URL http://www.robesafe.com/personal/pablo.alcantarilla/papers/Alcantarilla13bmvc.pdf – Zugriffsdatum: 2017-05-10

[Bay et al. 2008] **BAY, Herbert; ESS, Andreas; TUYTELAARS, Tinne; VAN GOOL, Luc :** Speeded-Up Robust Features (SURF). In: *Comput. Vis. Image Underst.* 110 (2008), Juni, Nr. 3, S. 346–359. – URL http://-dx.doi.org/10.1016/j.cviu.2007.09.014. – ISSN 1077-3142 – Zugriffsdatum: 2017-01-05

[Bradski2000] **BRADSKI, G. :** The OpenCV Library. In: *Dr. Dobb's Journal of Software Tools* (2000) – URL http://opencv.org/ – Zugriffsdatum: 2017-01-05

[Calonder et al. 2010] **CALONDER, Michael; LEPETIT, Vincent; STRECHA, Christoph; FUA, Pascal:** BRIEF: Binary Robust Independent Elementary Features. In: *Proceedings of the 11th European Conference on Computer Vision: Part IV*. Berlin, Heidelberg: Springer-Verlag,

2010 (ECCV'10), S. 778–792. – URL http://dl.acm.org/-citation.cfm?id=1888089.1888148. – ISBN 3-642-15560-X, 978-3-642-15560-4

[Chollet et al. 2015] **CHOLLET, François et al.:** Keras. https://github.com/fchollet/-keras, 2015.

[Crow1984] **CROW, Franklin C. :**Summed-area Tables for Texture Mapping. In: *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA : ACM, 1984 (SIGGRAPH '84), S. 207–212. – URL http://doi.acm.org/-10.1145/800031.808600. – ISBN 0-89791-138-5 – Zugriffsdatum: 2017-01-05

[Dedhia2015] **DEDHIA, Vaibhav :** Viola-Jones in Nut-shell. Aug 2015. – URL http://computervisionwithvaibhav.blogspot.de/2015/08/-viola-jones-in-nut-shell.html. – Zugriffsdatum: 2017-01-10

[Delko2015] **DELKO, Krim**: *Fortschritte in der Robotik: Silicon Valley erwartet nächste kambrische Explosion*. September 2015. – URL https://-www.nzz.ch/finanzen/silicon-valley-erwartet-naechste-kambrische-explosion-1.18607264 – Zugriffsdatum: 2017-05-10

[Fergus2015] **FERGUS, Rob.:** Neural Networks MLSS 2015 Summer School. 2015. – URL http://mlss.tuebingen.mpg.de/2015/slides/-fergus/Fergus_1.pdf – Zugriffsdatum: 2017-05-10

[Freund et al. 1995] **FREUND, Yoav ; Schapire, ROBERT E.:** A decision-theoretic generalization of on-line learning and an application to boosting. In: European Conference on Computational Learning Theory, S. 23–37. – URL http://citeseer.ist.psu.edu/-freund95decisiontheoretic.html – Zugriffsdatum: 2017-05-10

[Gionis et al. 1999] **GIONIS, Aristides ; INDYK, Piotr ; MOTWANI, Rajeev:** Similarity Search in High Dimensions via Hashing. In: *The VLDB Journal*, URL http://citeseerx.ist.psu.edu/viewdoc/-summary?doi=10.1.1.43.9482, 1999, S. 518–529 – Zugriffsdatum: 2017-01-05

[Grewenig et al. 2010] **GREWENIG, Sven ; WEICKERT, Joachim ; BRUHN, Andrés:** From box filtering to fast explicit diffusion. In: *Proceedings of the 32nd DAGM conference on Pattern recognition*. Berlin, Heidelberg : Springer-Verlag, 2010, S. 533–542. – URL http://-portal.acm.org/citation.cfm?id=1926320. – ISBN 3-642-15985-0, 978-3-642-15985-5 – Zugriffsdatum: 2017-05-10

[Grief2016] **GREIF, Björn**: *Google macht Öffentliche Beta seiner Cloud Vision API verfügbar*. Februar 2016. – URL http://www.zdnet.de/-88260586/google-macht-oeffentliche-beta-seiner-cloud-vision-api-verfuegbar/
– Zugriffsdatum: 2017-05-10

[Harris et al. 1988] **HARRIS, C. ; STEPHENS, M.:** A Combined Corner and Edge Detector. In: Procedings of the Alvey Vision Conference *1988*, Alvey Vision Club, 1988, S. 23.1–23.6. – URL http://dx.doi.org/-10.5244/c.2.23
– Zugriffsdatum: 2017-05-10

[He et al. 2015] **HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian:** Deep Residual Learning for Image Recognition. In: *Computing Research Repository* abs/1512.03385 (2015). – URL http://-arxiv.org/abs/1512.03385
– Zugriffsdatum: 2017-05-10

[Hoashi et al. 2010] **HOASHI, H. ; JOUTOU, T. ; YANAI, K.:** Image Recognition of 85 Food Categories by Feature Fusion. In: *2010 IEEE International Symposium on Multimedia (ISM),* Dezember 2010, S. 296–301 – URL http://ieeexplore.ieee.org/abstract/document/5693856/
– Zugriffsdatum: 2017-05-10

[Huber et al. 2013] **HUBER, Jakob ; KOPF, Stephan ; SCHABER, Philipp:** *Analyse von Bildmerkmalen zur Identifikation wichtiger Bildregionen*. 2013. – URL http://ub-madoc.bib.uni-mannheim.de/33097/
– Zugriffsdatum: 2017-05-10

[Russakovsky et al. 2014] **RUSSAKOVSKY, Olga; DENG, Jia; SU, Hao; KRAUSE, Jonathan; SATHEESH, Sanjeev; MA, Sean; HUANG, Zhiheng; KARPATHY, Andrej; KHOSLA, Aditya; BERNSTEIN, Michael S.; BERG, Alexander C.; LI, Fei-Fei :** ImageNet Large Scale Visual Recognition Challenge. In: *Computer Research Repository* abs/1409.0575 (2014). – URL http://arxiv.org/abs/1409.0575 – Zugriffsdatum: 2017-05-10

[Russakovsky et al. 2015] **RUSSAKOVSKY, Olga; DENG, Jia; SU, Hao; KRAUSE, Jonathan; SATHEESH, Sanjeev; MA, Sean; HUANG, Zhiheng; KARPATHY, Andrej; KHOSLA, Aditya; BERNSTEIN, Michael S.; BERG, Alexander C.; LI, Fei-Fei :** ImageNet Large Scale Visual Recognition Challenge. In: *International Journal of Computer Vision (IJCV)* 115 (2015), Nr. 3, S. 211–252. – URL https://link.springer.com/article/10.1007/s11263-015-0816-y – Zugriffsdatum: 2017-05-10

[Kagaya et al. 2014] **KAGAYA, Hokuto ; AIZAWA, Kiyoharu ; OGAWA, Makoto:** Food Detection and Recognition Using Convolutional Neural Network. In: *Proceedings of the 22Nd ACM International Conference on Multimedia*. New York, NY, USA : ACM, 2014 (MM '14), S. 1085–1088. – URL http://doi.acm.org/10.1145/-2647868.2654970. – ISBN 978-1-4503-3063-3 – Zugriffsdatum: 2017-05-10

[Krig2014] **KRIG, Scott.:** *Computer Vision Metrics: Survey, Taxonomy, and Analysis*. 1st. Berkely, CA, USA : Apress, 2014. – ISBN 1430259299, 9781430259299 – URL https://link.springer.com/book/10.1007%2F978-1-4302-5930-5 – Zugriffsdatum: 2017-05-10

[Levin et al. 2017] **LEVIN, Mark; HARRIS, Mark**: *The road ahead: self-driving cars on the brink of a revolution in California*. März 2017. – URL https://www.theguardian.com/technology/2017/mar/17/self-driving-cars-california-regulation-google-uber-tesla – Zugriffsdatum: 2017-05-10

[Li et al. 2016] **LI, Fei-Fei ; KARPATHY, Andrej ; JOHNSON, Justin:** *CS231n Convolutional Neural Networks for Visual Recognition.* – URL http://cs231n.github.io/convolutional-networks/. – Zugriffsdatum: 2017-03-1

[Lienhart et al. 2003] **LIENHART, Rainer ; KURANOV, Alexander ; PISAREVSKY, Vadim:** Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection. In: MICHAELIS, Bernd (Hrsg.) ; KRELL, Gerald (Hrsg.): *Pattern Recognition: 25th DAGM Symposium, Magdeburg, Germany, September 10-12, 2003. Proceedings*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2003. – URL http://dx.doi.org/10.1007/978-3-540-45243-0_39. – ISBN 978-3-540-45243-0 – Zugriffsdatum: 2017-05-10

[Linz2015] **LINZ, Carl**: *Musikstreaming löst die CD ab*. Februar 2015. – URL http://www.moz.de/artikel-ansicht/dg/0/1/1364961 – Zugriffsdatum: 2017-05-10

[Lowe1999] **LOWE, D. G.:** Object recognition from local scale-invariant features. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision* Bd. 2, 1999, S. 1150–1157 – URL http://ieeexplore.ieee.org/abstract/document/790410/ – Zugriffsdatum: 2017-05-10

[Lowe2004] **LOWE, David G.:**Distinctive Image Features from Scale-Invariant Keypoints. In: *International Journal of Computer Vision* 60 (2004), Nr. 2, S. 91–110. – URL http://dx.doi.org/10.1023/-B:VISI.0000029664.99615.94. – ISSN 1573-1405 – Zugriffsdatum: 2017-05-10

[Lv et al. 2007] **LV, Qin; JOSEPHSON, William; WANG, Zhe; CHARIKAR, Moses; LI, Kai:** Multi-probe LSH: Efficient Indexing for High-dimensional Similarity Search. In: *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB Endowment, 2007 (VLDB '07), S. 950–961. – URL http://dl.acm.org/-citation.cfm?id=1325851.1325958. – ISBN 978-1-59593-649-3 – Zugriffsdatum: 2017-05-10

[Mehner2013] **MEHNER, Robin:** *Train Your Own OpenCV Haar Classifier*. Jul 2013. – URL http://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html – Zugriffsdatum: 2017-05-10

[Pham et al. 2013] **PHAM, Cuong; JACKSON, Daniel; SCHOENING, Johannes; BARTINDALE, Tom; PLOETZ, Thomas; OLIVIER, Patrick:** FoodBoard: Surface Contact Imaging for Food Recognition. In*: Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. New York, NY, USA : ACM, 2013 (UbiComp '13), S. 749–752. – URL http://doi.acm.org/-10.1145/2493432.2493522. – ISBN 978-1-4503-1770-2 – Zugriffsdatum: 2017-05-10

[Piazza et al. 2011] **PIAZZA, Annalisa Di; PIAZZA, Maria Carmela Di; RAGUSA, Antonella; VITALE, Gianpaolo :** Environmental data processing by clustering methods for energy forecast and planning. In: *Renewable Energy* 36 (2011), Nr. 3, S. 1063 – 1074. – URL http://www.sciencedirect.com/science/article/pii/-S0960148110004234. – ISSN 0960-1481 – Zugriffsdatum: 2017-05-10

[Puri et al. 2009] **PURI, M.; ZHU, ZHIWEI; YU, Q.; DIVAKARAN, A.; SAWHNEY, H.:** Recognition and volume estimation of food intake using a mobile device. In: *2009 Workshop on Applications of Computer Vision (WACV)*, Dezember 2009, S. 1–8. – ISSN 1550-5790 – URL http://ieeexplore.ieee.org/abstract/document/5403087/ – Zugriffsdatum: 2017-05-10

[Qi et al. 2016] **QI, C. R.; SU, H.; NIEßNER, M.; DAI, A.; YAN, M.; GUIBAS, L. J. :** Volumetric and Multi-view CNNs for Object Classification on 3D Data. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Juni 2016, S. 5648–5656 – URL https://arxiv.org/abs/1604.03265

[Rolt2016] **ROLT, Jean Da :** *How is a convolutional neural network able to learn invariant features?* . – URL https://www.quora.com/How-is-a-convolutional-neural-network-able-to-learn-invariant-features – Zugriffsdatum: 2017-05-10

[Rosebrock2014] **ROSEBROCK, Adrian:** *OpenCV and Python K-Means Color Clustering*. Mai 2014. – URL http://www.pyimagesearch.com/-2014/05/26/opencv-python-k-means-color-clustering/. – Zugriffsdatum: 2017-01-05

[Rosten et al. 2005] **ROSTEN, E. ; DRUMMOND, T.:** Fusing points and lines for high performance tracking. In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1* Bd. 2, Oktober 2005, S. 1508–1515 Vol. 2.– URL http://ieeexplore.ieee.org/document/1544896/ – ISSN 1550-5499 – Zugriffsdatum: 2017-05-10

[Rosten et al. 2006] **ROSTEN, Edward ; DRUMMOND, Tom:** Machine Learning for High-speed Corner Detection. In: *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*. Berlin,

Heidelberg : Springer-Verlag, 2006 (ECCV'06), S. 430–443. – URL http://dx.doi.org/10.1007/11744023_34. – ISBN 3-540-33832-2, 978-3-540-33832-1 – Zugriffsdatum: 2017-05-10

[Rublee et al. 2011] **RUBLEE, E.; RABAUD, V.; KONOLIGE, K.; BRADSKI, G. :** ORB: An efficient alternative to SIFT or SURF. In: *2011 International Conference on Computer Vision*, November 2011, S. 2564–2571. – ISSN 1550-5499 – URL http://www.vision.cs.chubu.ac.jp/CV-R/pdf/Rublee_iccv2011.pdf – Zugriffsdatum: 2017-05-10

[Simonyan et al. 2014] **SIMONYAN, Karen ; ZISSERMAN, Andrew:** Very Deep Convolutional Networks for Large-Scale Image Recognition. In: *Computer Research Repository* abs/1409.1556 (2014). – URL http://arxiv.org/abs/1409.1556 – Zugriffsdatum: 2017-01-05

[Abadi et al. 2015] **ABADI , Martín ; AGARWAL , Ashish ; BARHAM , Paul ; BREVDO , Eugene ; CHEN , Zhifeng ; Citro , Craig ; S. , Greg ; Davis , Andy ; Dean , Jeffrey ; Devin , Matthieu ; Ghemawat , Sanjay ; GOODFELLOW , Ian ; HARP , Andrew ; IRVING , Geoffrey ; ISARD , Michael ; YANGQING , Jia ; JOZEFOWICZ , Rafal ; KAISER , Lukasz ; KUDLUR , Manjunath ; LEVENBERG , Josh ; MANÉ , Dan ; MONGA , Rajat ; MOORE , Sherry ; MURRAY , Derek ; OLAH , Chris ; SCHUSTER , Mike ; SHLENS , Jonathon ; STEINER , Benoit ; SUTSKEVER , Ilya ; TALWAR , Kunal ; TUCKER , Paul ; VANHOUCKE , Vincent ;**

VASUDEVAN , Vijay ; VIÉGAS , Fernanda ; VINYALS , Oriol ; WARDEN , Pete ; WATTENBERG , Martin ; WICKE , MARTIN ; Yu , YUAN ; Zheng , XIAOQIANG : *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015. – URL http://-tensorflow.org/. – Zugriffsdatum: 2017-05-10

[Tuytelaars et al. 2008] **TUYTELAARS, Tinne ; MIKOLAJCZYK, Krystian:** Local Invariant Feature Detectors: A Survey. In: *Foundations and trends© in computer graphics and vision* 3 (2008), Juli 2008, Nr. 3, S. 177–280. – URL http://dx.doi.org/10.1561/0600000017. – ISSN 1572-2740 – Zugriffsdatum: 2017-05-10

[Veit et al. 2016] **VEIT, Andreas ; WILBER, Michael J. ; BELONGIE, Serge J.:** Residual Networks are Exponential Ensembles of Relatively Shallow Networks. In: *Computer Research Repository* abs/1605.06431 (2016). – URL http://arxiv.org/abs/1605.06431 – Zugriffsdatum: 2017-05-10

[Viola et al. 2001] **Viola, P. ; Jones, M.:** Rapid object detection using a boosted cascade of simple features. In: 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Bd. 1. Los Alamitos, CA, USA : IEEE Comput. Soc, April 2001, S. 511–518. – URL http://dx.doi.org/10.1109/cvpr.2001.990517. – ISBN 0-7695-1272-0 – Zugriffsdatum: 2017-05-10

[Weickert et al. 1998] **WEICKERT, J. ; Romeny, B. M. T. H. ; VIERGEVER, M. A.:** Efficient and reliable schemes for nonlinear diffusion filtering. In: *IEEE Transactions on Image Processing* 7 (1998), März, Nr. 3, S. 398–410. – ISSN 1057-7149 – URL http://ieeexplore.ieee.org/document/661190/ – Zugriffsdatum: 2017-05-10

[Welch2017] **WELCH, Chris**: *Google Home now supports multiple users, but still can‘t separate work and personal accounts*. April 2017. – URL https://www.theverge.com/circuitbreaker/2017/4/20/-15364960/google-home-speaker-multi-user-new-feature – Zugriffsdatum: 2017-05-10

[Welteroth2005] **WELTEROTH, Silke**: *E-Mail löst Fax und Brief ab*. April 2005. – URL https://www.pressetext.com/news/20050426048 – Zugriffsdatum: 2017-05-10

[Wikipedia2017] **WIKIPEDIA. 2017.** *Convolutional Neural Network* --- Wikipedia, Die freie Enzyklopädie. 2017. –URL https://de.wikipedia.org/-w/-index.php?title=Convolutional_Neural_Network&oldid=161729433. – Zugriffsdatum: 2017-01-10

[Yang et al. 2012] **YANG, Xin ; CHENG, Kwang-Ting:** LDB: An Ultra-fast Feature for Scalable Augmented Reality on Mobile Devices. In: *Proceedings of the 2012 IEEE International Symposium on Mixed and*

Augmented Reality (ISMAR). Washington, DC, USA : IEEE Computer Society, 2012 (ISMAR ’12), S. 49–57. – URL http://-dx.doi.org/10.1109/ISMAR.2012.6402537. – ISBN 978-1-4673-4660-3 – Zugriffsdatum: 2017-05-10

Abbildungsverzeichnis

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den _____ _____