

Bachelor Abschlussarbeit

Visuelle Objekterkennung in Echtzeit

Martin Schwitalla

16. September 2013

Gutachter:

Prof. Dr.-Ing. Gernot A. Fink

M. Sc. Rene Grzeszick

Technische Universität Dortmund

Fakultät für Informatik

Lehrstuhl Informatik XII

<http://ls12-www.cs.tu-dortmund.de>

Inhaltsverzeichnis

1 Einleitung	3
1.1 Motivation und Hintergrund	3
1.2 Aufbau der Arbeit	5
2 Grundlagen	7
2.1 Echtzeit	7
2.2 Objekterkennung	8
2.2.1 Herleitung	9
2.2.2 Merkmale	10
2.2.3 Clustering	13
2.2.4 Quantisierung	14
2.2.5 Klassifikation	14
2.2.6 Lage der Visual Words	16
2.3 Räumliche Informationen	17
2.3.1 Tiefenbilder	17
2.3.2 Microsoft Kinect	18
2.4 Bildverarbeitung	19
2.4.1 Filter	20
2.4.2 Segmentierung	20
2.5 kd-Tree	21
3 Verwandte Arbeiten	23
3.1 Introduction to the Bag of Features Paradigm	23
3.1.1 Bild Klassifikation	23
3.1.2 Bild Retrieval	24
3.1.3 Anwendung	24
3.1.4 Fazit	26
3.2 Spatial Pyramid Matching	26
3.2.1 Idee	27
3.2.2 Auswertung	28

4 Methodik	29
4.1 Aufgabenstellung	29
4.2 Idee der Umsetzung	29
4.2.1 Zusammenstellung einer Trainingsbasis	30
4.2.2 Detektion der Objekte in einer realen Szene	30
4.2.3 Klassifikation der Objekte	31
4.2.4 Interface	32
4.3 Verwendete Bibliotheken	33
4.4 Verarbeitung der Daten	34
5 Auswertung	37
5.1 Metrik	37
5.2 Auswertung mittels Caltech101	37
5.3 Auswertung auf eigenen Bildern	39
6 Fazit	43
Abbildungsverzeichnis	45
Literaturverzeichnis	48
Erklärung	48

Kapitel 1

Einleitung

Die visuelle Objekterkennung beschäftigt sich mit dem Erkennen und Klassifizieren diverser Objekte in Bildquellen. Während ein Mensch mit Leichtigkeit Gesichter oder Objekte, wie zum Beispiel Autos erkennen kann, bereitet dies hingegen einem Computer große Schwierigkeiten. Ein Mensch kennt den Kontext und die Hintergründe eines Bildes bzw. eines Objektes und kann aus diesem Grund nicht-offensichtliche Schlüsse ziehen. Das Problem der Bestimmung von Bildinhalten wird in der visuellen Objekterkennung angegangen. Daraus folgen unterschiedliche Einsatzmöglichkeiten für den Einsatz im Alltag.

1.1 Motivation und Hintergrund

Die Erkennung von Objekten kann auf unterschiedliche Art und Weise nützlich sein. Zum Beispiel ermöglichen autonome Fahrzeuge die Mobilität blinder Menschen. Denkbar wäre auch eine Minienzyklopädie, worin man nicht mehr nach Begriffen sucht, sondern welche die zu sehenden Objekte erkennt und dazu die wichtigsten Informationen liefert. Trotzdem existieren zahlreiche Probleme bei der Objekterkennung. Während ein Mensch dank seiner Erfahrungen viele Verkehrsschilder in unterschiedlichen Ländern (vgl. Abbildung 1.1) interpretieren kann, sind visuelle Objekterkennungssysteme meist nur auf ihr Einsatzgebiet spezialisiert und ordnen aus diesem Grund möglicherweise andere Informationen falsch ein oder erkennen diese nicht. Im Falle eines autonomen Fahrzeugs, würde dies unter Umständen katastrophale Folgen nach sich ziehen.

Wünschenswert wäre somit eine maschinelle Analyse von Bildern, bei der keine Informationen verloren gehen und trotzdem weit genug abstrahiert werden kann, um möglichst auch noch unbekannte Objekte zu erkennen. Solch eine Anwendung hätte nicht nur zahlreiche Vorteile für Einsatzzwecke, wie das Sortieren oder Annotieren von Bildern, sondern auch für viele weitere Fälle im Alltag. Denkbar wäre eine Smartphone-Applikation die zu fotografierten Gebäuden oder Gemälden, in Form einer kleinen Enzyklopädie, Informationen ausgibt. Andere denkbare Einsatzmöglichkeiten finden sich auch in unüblichen Bereichen,



Abbildung 1.1: Links: Radwegsschild aus Deutschland (www.regensburg-digital.de) , Rechts: Schild aus den USA (www.cityofevanston.org)

wie der Medizin, wo solch ein System die Erkennung von Blessuren auf Röntgenbildern erleichtern könnte. Denkbar wären sogar Roboter, die ohne menschliche Hilfe operative Eingriffe durchführen und dafür die entsprechenden Stellen in und am Körper detektieren müssten, um keine lebenswichtigen Organe zu verletzen.

Während hier der Geschwindigkeitsaspekt vernachlässigt werden kann, gibt es auch Anwendungen, die in Echtzeit ausgeführt werden und deren Einhalten von harten Schranken essentiell und zum Teil lebenswichtig sind. Ein gutes Beispiel dafür sind die bereits erwähnten autonomen Fahrzeuge, die von Google aktuell in diversen Staaten der USA eingesetzt werden¹. Sollte hierbei ein Fahrzeug beim Fahren zu lange für das Erkennen von Personen oder Verkehrsschildern benötigen, könnte es zu schlimmen Unfällen kommen. Aber auch das Erkennen von Fahrzeugen im toten Winkel ist auf harte Echtzeit angewiesen, da solche Systeme ansonsten keinen reibungslosen Ablauf ermöglichen, falls das Auto erst erkannt wird, wenn der Fahrer bereits den Fahrstreifen wechselt.

Ein großes Problem der Objekterkennung sind die vielfältigen Formen der Objekte, welche keinem Schema folgen und beliebig aussehen können. Daher ist es entscheidend eine Repräsentation von Objekten zu finden, die möglichst viele Variabilität abdeckt. Für solche Systeme hat sich der Bag of Features Ansatz als vielversprechend herausgestellt. Hierbei werden lokale Bildinformationen extrahiert, welche eine Repräsentation des Bildes darstellen. Diese sogenannten Visual Words können anschließend dazu genutzt werden, um Objekte zu erkennen. Beispielsweise können so auf einem Bild mit einem Haus eine Tür, sowie mehrere Fenster vorkommen, die nun zur Klassifikation herangezogen werden können. Im Vorfeld werden dazu die lokalen Features einer Reihe an Bildern, der Trainingsbasis, extrahiert und mit Verfahren des maschinellen Lernens zu den Visual Words zusammengefasst.

Problematisch bei diesem Ansatz können weitere Objekte im Hintergrund eines Bildes sein. Denn ein Bild eines bestimmten Objekts zeigt selten nur das Objekt selbst, sondern oftmals

¹<http://www.google.com/about/jobs/lifeatgoogle/self-driving-car-test-steve-mahan.html>



Abbildung 1.2: Beispiele aus der Caltech101 Datenbank[12]

auch viele weitere Objekte, wie Bäume oder Menschen im Hintergrund, die die Ergebnisse verfälschen könnten. Somit ist ein Verfahren nötig, dass einzelne Objekte eines Bildes vom Rest trennt, um die Konzentration auf die bestimmenden Objekte eines Bildes zu legen. Zusätzlich kann in einer Echtzeitanwendung die Anzahl der benötigten Visual Words zu Problemen führen. Je nach Umsetzung können solche Vokabularen mehrere tausend Visual Words umfassen, was zu einer sehr detaillierten und auch differenzierten Repräsentation eines Bildes führt. Dabei ist es aber oftmals unwahrscheinlich, dass in einem Bild mit einer Kaffeetasse Visual Words auftauchen, die für ein Fahrzeug oder einen Baum sprechen könnten. Daher ist es auch wichtig Kontextinformationen zu bedenken, indem beispielsweise gewisse Vorkommen von Visual Words einem bestimmten Objekt zugeordnet oder weitere Informationen zur Differenzierung genutzt werden.

Ziel ist es nun eine Anwendung zu erstellen, die mithilfe des Bag of Features Ansatzes eine Objekterkennung in Echtzeit in realen Umfeldern ermöglicht und dabei unter anderem die kurz vorgestellten Probleme und Ansätze berücksichtigt.

1.2 Aufbau der Arbeit

Die vorliegende Abschlussarbeit ist wie folgt aufgebaut. In Kapitel 1 wird die Motivation für die Arbeit erläutert. Kapitel 2 befasst sich mit den Grundlagen der Objekterkennung und des Bag of Features Ansatzes, die für das Verständnis der weiteren Arbeit benötigt werden. In Kapitel 3 werden bereits veröffentlichte Arbeiten aus dem Bereich der Objekterkennung thematisiert. Kapitel 4 erläutert die Aufgabenstellung der Abschlussarbeit, sowie die bei der Lösung dieser entstandenen Probleme und geht dabei auf diverse Lösungsansätze ein. Diese Ergebnisse werden anschließend in Kapitel 5 ausgewertet, indem die Caltech101 Datenbank (vgl. Abbildung 1.2) zur Hilfe genommen und das System einem Test in realer Umgebung unterzogen wird. Schließlich wird in Kapitel 6 ein Fazit der Arbeit gezogen und ein Ausblick auf mögliche Ergänzungen gegeben.

Kapitel 2

Grundlagen

Ein System bzw. Rechner kann ohne entsprechende Hilfe keine Objekte erkennen. Für eine Anwendung ist nicht erkennbar, ob es sich auf einem Bild um ein Auto oder einen Baum handelt. Stattdessen ist es notwendig dem Rechner dieses Wissen anhand von Beispielen beizubringen. Durch das Extrahieren von Merkmalen kann ein System Informationen über die Objekte erhalten, die mit dem Bag of Features Ansatz repräsentiert werden und eine Klassifikation erlauben. Für die gegebene Aufgabenstellung ist es aber auch wichtig, die Ergebnisse in Echtzeit zu berechnen.

Im folgenden werden daher die Definitionen von Echtzeit in Abschnitt 2.1 erläutert, die Grundlagen der Objekterkennung in Abschnitt 2.2 in Bezug auf den Bag of Features Ansatz erklärt und auch Bereiche der räumlichen Informationen in Abschnitt 2.3, Bildverarbeitung in Abschnitt 2.4 und der kd-Tree in Abschnitt 2.5 thematisiert.

2.1 Echtzeit

Der Begriff Echtzeit wird in der DIN 44400 Norm wie folgt definiert:

“Unter Echtzeit versteht man den Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen”[20].

Somit wird von einer Echtzeitanwendung erwartet, dass diese ankommende Daten ständig bearbeiten kann und die daraus resultierenden Ergebnisse in vorher festgelegten Zeitspannen zur Verfügung stehen. Diese Zeitspannen können je nach Anwendungsfall weiter differenziert werden, wobei hauptsächlich zwischen weichen und harten Schranken für Echtzeit unterschieden wird. Eine mögliche Definition für harte Schranken lautet:

“A time-constraint is called hard if not meeting that constraint could result in a catastrophe”[8].



Abbildung 2.1: Ein Zylinder auf dem die Farb- bzw. Lichtverläufe eingezeichnet sind.

Dies bedeutet das dann von harten Schranken gesprochen werden kann, wenn das nicht erreichen dieser zu katastrophalen Ergebnissen führen könnte. Oftmals ist dies in sicherheitskritischen Anwendungen der Fall, wie in nahezu allen Assistenzsystemen in Fahrzeugen. Sollte hier beispielsweise ein Airbag zu spät auslösen, könnte es für den Fahrer zu lebensbedrohlichen Situationen kommen. Bei allen anderen Schranken wird dann von weichen Schranken gesprochen. Hierbei wäre ein Verfehlen einer Zeitschranke zwar nicht wünschenswert für das Verhalten der Anwendung, jedoch allgemein irrelevant, da keine sicherheitsrelevanten Systeme davon abhängen.

Manchmal wird aber auch noch zwischen fester Echtzeit unterschieden. Dabei handelt es sich um Zeitschranken, deren nicht Einhalten keine Schäden zur Ursache hätte, aber die Ergebnisse nach Ablauf der Schranke für die Anwendung nutzlos werden. Solche Zeitschranken finden sich zum Beispiel in Temperaturregelanlagen. Falls hier ein Ergebnis einer Temperaturmessung zu spät eintrifft, kommt es zu keinem Schaden. Jedoch kann die Anwendung für einen kurzen Zeitraum nicht reagieren und arbeitet möglicherweise nicht korrekt.

2.2 Objekterkennung

Mit dem Begriff Objekterkennung werden Verfahren zur optischen Identifizierung von Objekten beschrieben. Damit lassen sich unter anderem das Vorhandensein eines Objektes, das Objekt selbst und auch die Lage dessen in dem untersuchten Bereich bestimmen. Allen Objekterkennungssystemen liegt jedoch zu Grunde, dass sie zu Beginn visuelle Daten sammeln müssen. Beim Menschen geschieht dies über die Augen und visuelle Systeme können hierbei auf eine Vielzahl von Sensoren zurückgreifen. Üblich sind dabei Kameras, welche

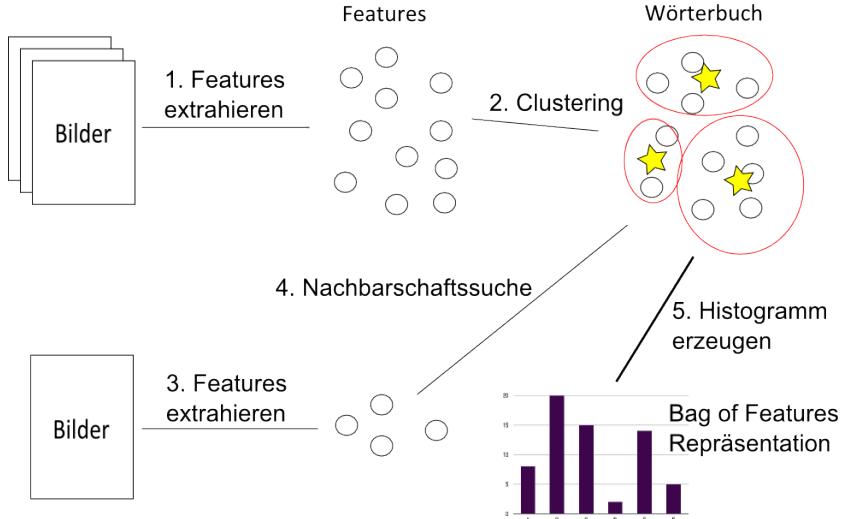


Abbildung 2.2: Die durchzuführenden Schritte, um eine Bag of Features Repräsentation zu erstellen

ein Bild mit einer gewissen Anzahl an Pixeln zurückzugeben. Diese Pixel enthalten jedoch noch keine Informationen über enthaltene Objekte oder ähnliches, sondern repräsentieren nur das Licht an einer bestimmten Stelle in dem Bild, welche durch die Ausprägung der Farben Rot, Grün und Blau dargestellt wird. Das Ziel der Objekterkennung ist es nun mit algorithmischen Grundlagen Informationen aus den gegebenen Pixeln zu extrahieren, um somit Objekte auf dem Bild zu erkennen.

Möglich wäre hier den Verlauf der Pixel zu untersuchen, da beispielsweise fließende Änderungen in der Belichtung und der Ausrichtung für eine geschlossene Oberfläche sprechen und somit zu einem Objekt gehören können(vgl. Abbildung 2.1). Ähnlich ist die Idee hinter der Extraktion von SIFT-Merkmalen und deren Repräsentation mithilfe des Bag of Features Ansatzes, der jedoch zahlreiche weitere Informationen der Bilder herausarbeitet, um nicht nur Objekte auf einem Bild zu erkennen, sondern diese auch zu klassifizieren. Der Ansatz arbeitet nach dem im folgenden erklärten Prinzip.

2.2.1 Herleitung

Das Bag of Features Verfahren stammt ursprünglich aus der Dokumentenanalyse, in der oftmals von Bag of Words gesprochen wird[21]. Hierbei wird versucht die Themen eines gegebenen Textes herauszufinden. Dazu werden alle Wörter des Textes extrahiert und sehr häufig auftauchende Wörter, wie "das" oder "ein" gestrichen, da sie keine Informationen über den Inhalt des Textes liefern. Die restlichen Wörter werden nun unter ihrem Wortstamm zusammengefasst, sodass zum Beispiel "schnarchen", "schnarche" usw. in einer Gruppe zusammen gehören. Anschließend werden die Vorkommen der einzelnen Wortstämme gezählt und in einem Histogramm repräsentiert, um so eine Aussage über den Inhalt

des Textes treffen zu können.

Ähnlich wie Texte durch Wörter repräsentiert werden, können Bilder über lokale Merkmale und ihre Deskriptoren, welche einen kleinen Ausschnitt eines Bildes beschreiben, ausgedrückt werden. Dabei können solche Deskriptoren sehr ähnlich oder auch völlig unterschiedlich sein, womit sie sich ebenfalls zu größeren Gruppen zusammenfassen lassen. Damit ist es auch möglich die Vorkommen der Deskriptoren zu zählen und das Bild über die Anzahl der Vorkommen zu repräsentieren. Da bei diesem Verfahren nur die Wortstämme durch die Deskriptoren bzw. lokalen Merkmale ersetzt wurden, wird häufig von Bag of Features gesprochen.

Um nun ein Bild zu klassifizieren, sind mehrere Schritte notwendig, welche in Abbildung 2.2 illustriert sind.

1. **Extrahieren von Merkmalen:** Auf einer Reihe von Trainingsbildern werden die lokalen Merkmale, welche in Unterabschnitt 2.2.2 erklärt werden, extrahiert.
2. **Clustering:** Um mit den berechneten Merkmalen nun ein Wörterbuch zu erstellen, müssen diese in größeren Gruppen zusammengefasst werden. Dabei werden ähnliche Deskriptoren in Clustern untergebracht und für jeden Cluster ein Zentroid berechnet, der diesen Cluster repräsentiert. Diese Zentroide bilden nun das Wörterbuch und jeder einzelne Zentroid repräsentiert ein sogenanntes Visual Word. Das Verfahren des Clusterings wird in Unterabschnitt 2.2.3 näher erläutert.
3. **Quantisierung:** Die Merkmale eines einzelnen Bildes werden entnommen und anschließend wird zu jedem Merkmal der nächste Nachbar aus dem Wörterbuch ermittelt, um so das Bild mit den auftauchenden Visual Words zu beschreiben (s. Unterabschnitt 2.2.4).
4. **Klassifikation:** Mit den ermittelten Visual Words des Bildes können nun die Vorkommen dieser gezählt und damit ein Histogramm erstellt werden. Man erhält somit eine Repräsentation des Bildes mit der Anzahl der Visual Words im Bild und kann eine Klassifikation vornehmen (s. Unterabschnitt 2.2.5).

Zu erwähnen ist, dass die Schritte 1 und 2 vor der eigentlichen Bildklassifikation geschehen müssen und oftmals auch als Anlernphase bezeichnet werden. Die letzten beiden Schritte beschreiben dann den eigentlichen Ablauf des Verfahrens, welche auf dem zu analysierenden Bild abläuft.

2.2.2 Merkmale

Im Rahmen der Arbeit werden zur Beschreibung lokaler Bildausschnitte SIFT Features oder Merkmale (Scale Invariante Feature Transformation) besprochen, da sich diese als

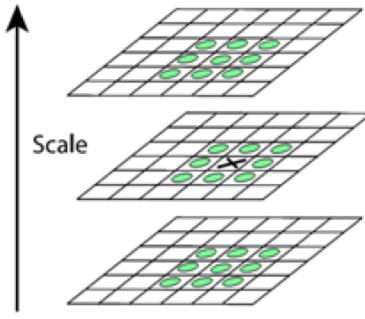


Abbildung 2.3: In der Mitte ist das Differenzbild zu sehen mit dem als X gekennzeichneten Punkt, wessen Nachbarschaft (grüne Punkte) untersucht wird. Entnommen aus [14]

state-of-the-art durchgesetzt haben[14, 21, 22]. Dabei beschreibt ein Merkmal eine bestimmte Region eines Bildes und wird durch einen dazugehörigen Deskriptor repräsentiert. Bestimmt wird ein Merkmal durch insgesamt drei Parameter. Den Keypoint, welcher das Zentrum des Merkmals als x- und y-Koordinate im Bild darstellt, die Skalierung bzw. Radius des Merkmals und die Orientierung. Eine zur Detektion von Merkmalen weit verbreitete Methode ist der SIFT Detektor [14]. Hierbei werden zuerst gegenüber Skalierung invariante Merkmale gesucht, indem auf allen verschiedenen Skalierungen stabile Merkmale gesucht werden. Ein Merkmal wird stabil genannt, wenn es in unterschiedlichen Szenen gefunden wird, in denen das selbe Objekt zu sehen ist. Das Berechnen der Skalierungen geschieht nun über einen *Gaußschen Scale Space*. Dieser ist definiert als:

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

Dabei beschreibt $I(x,y)$ das Eingabebild, $*$ die Faltung der beiden Funktionen und $G(x, y, \sigma)$ ist die Gaußsche Funktion:

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

Für jede Verdopplung von σ , wird das Layer L in jeder Dimension um den Faktor zwei herunterskaliert. Bilder mit den selben Dimensionen werden daher einer sogenannten Oktave zugeschrieben. Um nun Keypoints zu detektieren, wird die Differenz zwischen zwei Gaußschen Ebenen einer Oktave gezogen, indem die adjazenten Ebenen des *Scale Space* pixelweise voneinander subtrahiert werden. Kandidaten für einen Keypoint werden ermittelt, indem für jeden Punkt der Differenzebene die Nachbarpunkte im Differenzbild und den zwei Adjazenzmatrizen untersucht werden (vgl. Abbildung 2.3). Ist dieser Punkt ein lokales Extrema in seiner Nachbarschaft, ist ein Kandidat für einen Keypoint gefunden. Anschließend werden Keypoints, welche am Rand eines Bildes oder an Stellen mit wenig Kontrast liegen, entfernt, um so instabile Merkmale zu eliminieren [14]. Nun wird das

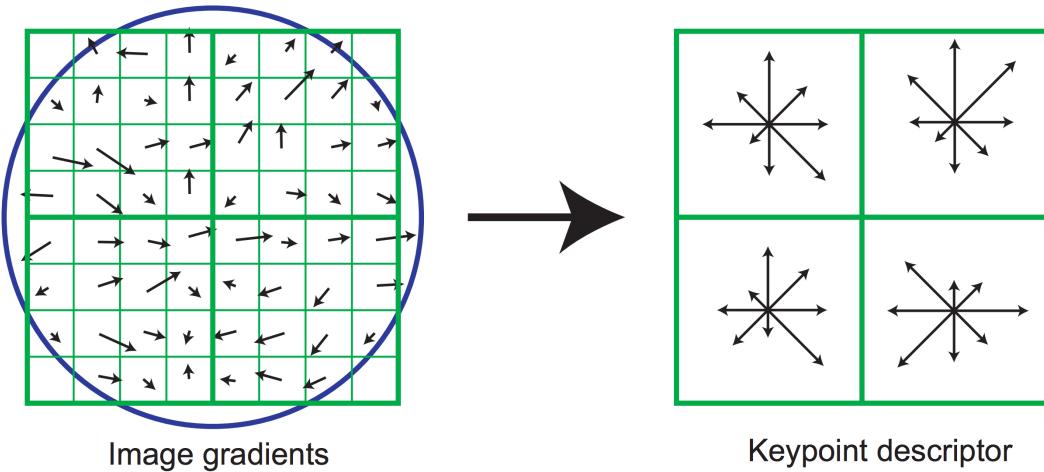


Abbildung 2.4: Links sind die berechneten Gradienten und ihre Orientierungen an den Beispieldpunkten zu sehen. Das Gaußsche Fenster wird durch den blauen Kreis gekennzeichnet. Rechts sieht man den Deskriptor mit den acht Orientierungen und den gewichteten Gradienten. Das gegebene Beispiel zeigt einen 2x2 Deskriptor, welcher aus einem 8x8 großem Set an Gradienten berechnet wird. Im realen Einsatz kommen jedoch größere Deskriptoren vor. Das Bild ist aus [14] entnommen.

Skalierungslevel dem Keypoint zugewiesen und eine Orientierung anhand der Größe der Gradienten des Keypoints bestimmt. Dazu werden die Gradienten in der Nachbarschaft berechnet und in einem Histogramm gesichert. Der höchste Ausschlag im Histogramm ist die Orientierung. Kommen mehrere Höhepunkte im Histogramm in Frage, wird der Keypoint in mehrere Keypoints mit selbem Ort und Skalierung, aber unterschiedlicher Orientierung, getrennt.

Bei dieser Art der Merkmalsdetektion kann es vorkommen, dass wenig Merkmale berechnet werden und diese somit nicht genügend Informationen bereitstellen, um ein Bild oder Objekt ausreichend zu beschreiben, wie es der Bag of Features Ansatz voraussetzt. Darauf wird oft ein Ansatz verwendet, der Merkmale auf einem vorher festgelegten Raster berechnet. Dabei werden die Merkmale in bestimmten Abständen berechnet und können somit auch Regionen, wie Wände oder Böden abdecken, die durch den SIFT Detektor nicht repräsentiert werden würden. Im Vorfeld werden die Abmessungen des Rasters, sowie die Skalierung der Keypoints bestimmt. Dabei muss beachtet werden, dass die Skalierung den Abmessungen des Rasters entspricht. Bei zu kleinen Keypoints können wichtige Informationen verloren gehen und bei zu großen entsteht viel unnötige Redundanz. In [22] wird empfohlen eine Skalierung so zu wählen, dass eine halbe Überdeckung der Deskriptoren entsteht.

Beschrieben werden die Merkmale nun durch einen SIFT Deskriptor. Diese bestehen aus einem Histogramm der Bildgradienten im Umfeld eines Keypoints. Dazu werden in dessen Umfeld alle Gradienten und deren Orientierung in Beispieldpunkten berechnet (vgl. Ab-

bildung 2.4). Danach werden die Gradienten mithilfe eines Gaußschen Fensters gewichtet und in einem Histogramm aufsummiert. (vgl. Abbildung 2.4). Die Länge der Vektoren repräsentiert die Summe der Gradienten in dem Bereich in der jeweiligen Orientierung. Üblicherweise erhält man so 4×4 Regionen mit den sogenannten Bins mit jeweils 8 Gradienten alle 30° und damit einen 128-dimensionalen Vektor, welcher das lokale Merkmal beschreibt.

2.2.3 Clustering

Wie bereits in Unterabschnitt 2.2.1 erwähnt, werden die berechneten Merkmale durch eine festgelegte Anzahl an Repräsentanten beschrieben. Das Finden dieser Repräsentanten wird als Clustering bezeichnet. Dabei muss die Anzahl der Repräsentanten, beim Bag of Features Verfahren, Visual Words genannt, mit Bedacht gewählt werden. Denn je größer das Wörterbuch an Visual Words gewählt wird, umso genauer wird die Darstellung. Jedoch kann es für Anwendungen von Vorteil sein, wenn die Visual Words reduziert und abstrahiert werden, um robust gegenüber leichter Variabilität zu sein.

Zum Clustering wird oftmals der *k-means Algorithmus* verwendet, welcher von MacQueen erstmals in [15] eingeführt wurde. Dabei wird vorher festgelegt, wie viele Cluster k der Algorithmus berechnen soll mit den gegebenen n Merkmalen. Dabei funktioniert der Algorithmus wie folgt:

1. **Initialisierung:** Aus dem Datensatz x_1, \dots, x_n werden die ersten k Vektoren entnommen, um das initiale Wörterbuch mit k Visual Words zu generieren.
2. **Zuweisung:** Für jeden Vektor x_i aus den n Vektoren, wird nun der nächste Nachbar der aktuellen Zentroiden bzw. Visual Words gesucht. Dies kann unter anderem per Euklidischer Distanz berechnet werden und so wird x_i Teil der Partition P .
3. **Update:** Nach dem Hinzufügen des Vektors zur Partition P wird davon ausgehend ein neuer Zentroid für P berechnet. .

Das besondere an dem *k-means* Algorithmus von MacQueen ist, dass er nur mit einer Iteration über die Daten ein Ergebnis liefert. Daher ist der Algorithmus von der Anzahl der Daten abhängig und berechnet nicht immer das optimale Ergebnis für das gegebene Problem. In [19] wurde gezeigt, dass der Algorithmus ein optimales Ergebnis berechnet, wenn die Datenmenge gegen Unendlich strebt. In realen Anwendungen ist dies offensichtlich nicht der Fall, jedoch werden beim Bag of Features große Mengen an Merkmalen berechnet, wodurch eine Approximation an das optimale Ergebnis erreicht wird. Einen anderen Ansatz hat Lloyd in [13] vorgestellt. Hierbei werden die Cluster wie folgt berechnet:

1. **Initialisierung:** Die k Zentroiden werden zufällig im Raum verteilt.

2. **Zuweisung:** Für jeden Vektor x_i aus den n Vektoren wird nun der nächste Nachbar der Zentroiden gesucht. Dies kann durch ein Distanzmaß, z.B. die euklidische Distanz, geschehen.
3. **Update:** Basierend auf der Zuweisung der Vektoren zu den Zentroiden werden nun neue Zentroide berechnet und die Schritte 2 und 3 wiederholt bis sich keine Veränderungen der Zentroiden mehr ergeben.

Dabei wird anders als bei MacQueen ein lokales Optimum der Verteilung der Daten berechnet. Jedoch kann es bei einer großen Anzahl an Daten unter Umständen zu langen Laufzeiten kommen, die bei MacQueen verhindert werden.

2.2.4 Quantisierung

Die Quantisierung beschreibt den Schritt der Zuweisung des ähnlichsten Visual Words zu einem Merkmal. Für die Zuweisung kann daher eine Distanzfunktion, wie die euklidische Distanz, genutzt werden.

$$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Sind zwei Merkmale $\mathbf{x} = (x_1, x_2, \dots, x_n)$ und $\mathbf{y} = (y_1, y_2, \dots, y_n)$ gegeben, kann über die euklidische Distanz so der minimale Abstand zwischen zwei Punkten im \mathbb{R}^n berechnet werden. Damit kann man nun einem gefundenen Merkmal sein entsprechendes Visual Word zuweisen und ein Histogramm erstellen. Da bei unterschiedlichen Bildgrößen auch die Deskriptoren größer oder kleiner werden können, wird die Frequenz gebildet, indem jeder Wert des Deskriptors durch die durchschnittliche Summe eines Vektors geteilt wird. Somit beschreibt das Histogramm dann das Vorkommen der einzelnen Visual Words in einem Bild und stellt damit die Bag of Features Repräsentation dar.

2.2.5 Klassifikation

Mithilfe der Bag of Features Repräsentation eines Bildes kann eine Klassifikation des Bildinhaltes vorgenommen werden. Dazu wird die Repräsentation einer der vorher definierten Klassen zugewiesen und dafür stehen einem unterschiedlichen Methoden zur Verfügung. Bekannt sind hierbei zum Beispiel die Suche nach dem nächsten Nachbarn, sowie der Einsatz einer Support Vector Machine. Wenn ein Objekt das zentrale Objekt eines Bildes ist und den Großteil des Inhaltes einnimmt, kann dabei auch von Objektklassifikation gesprochen werden.

Die nächste Nachbar Suche kann, wie in Unterabschnitt 2.2.4 beschrieben, per euklidischer Distanz geschehen. Der nächste Nachbar Ansatz wird oftmals bei der Suche nach ähnlichen Bildern, dem sogenannten Retrieval, eingesetzt, wie in [10] gezeigt. Jedoch eignet er sich weniger für die Klassifikation von Bildinhalten, da er von Ausreißern in den Repräsentation

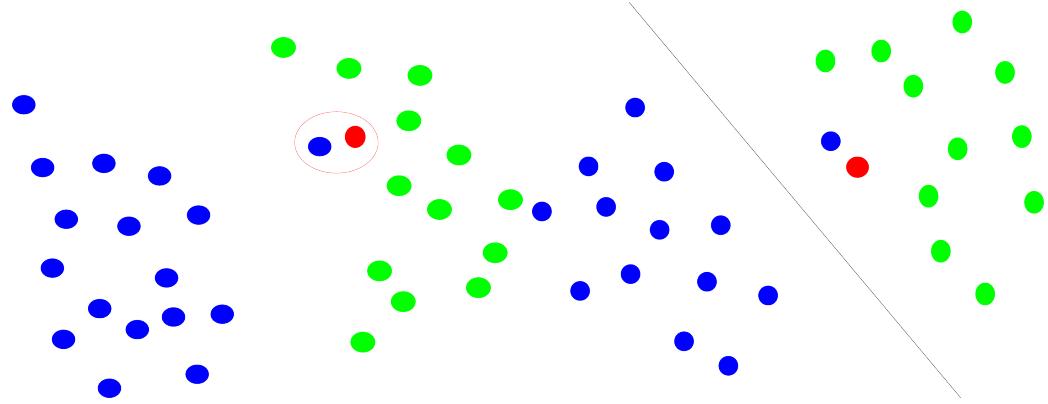


Abbildung 2.5: Auf beiden Bildern ist die Verteilung von Klassen in einem beispielhaften Modell gegeben. Dabei gehören alle blauen bzw. grünen Kreise zu einer Klasse. Links ist der Einsatz eines nächsten Nachbarn Verfahrens zu sehen. Dort ist gut zu erkennen, dass das zu klassifizierende Bild eigentlich zur grünen Klasse gehört. Durch den blauen Ausreißer wird das Ergebnis jedoch verfälscht, da er der nächste Nachbar ist (Roter Kringel). Mit den k-nächsten Nachbarn wäre hier trotzdem noch eine richtige Klassifikation möglich. Rechts ist die selbe Szene mit einer SVM dargestellt. Zwischen beiden Klassen wurde eine Hyperebene aufgestellt, welche die beiden Klassen unterteilt. Hier wird das Ergebnis richtigerweise der grünen Klasse zugeordnet, da der Ausreißer keinen Einfluss auf das Ergebnis hat.

stark beeinflusst werden kann (vgl. Abbildung 2.5). Diesen Umstand könnte man ändern, indem man die k-nächsten Nachbarn berechnet, um so eine Mehrheitsentscheidung zu treffen.

Für die Aufgabe der Klassifikation eignet sich der Einsatz einer Support Vector Machine deutlich besser. Dabei handelt es um ein Verfahren des maschinellen Lernens. Eingeführt wurde das Prinzip der Support Vector Machine von [3]. Dabei basiert die Idee darauf zwischen zwei verschiedenen Klassen eine Hyperebene aufzuspannen, um die Klassen voneinander zu trennen. Im Bereich der Bag of Feature Repräsentation werden alle Repräsentationen einer Klasse von denen einer anderen getrennt. Dafür wird eine Hyperebene mit maximalen Abstand zu den sogenannten Support Vectors, den zur Hyperebene nächsten Vektoren, berechnet. Für die gegebenen Trainingsdaten x und die Hyperebene definierenden Parameter y und b , kann folgende Funktion definiert werden.

$$f(x) = y^T x + b$$

Somit wird ein Bild einer Klasse zugeordnet, wenn $f(x) > 0$ ist oder der anderen, wenn $f(x) < 0$ ist (vgl. Abbildung 2.5). Jedoch sind zwei Klassen nicht immer mit einer linearen Hyperebene trennbar, weswegen die Idee besteht den Raum mit den Vektoren in einen höherdimensionalen Raum zu überführen, in dem die Vektoren dann linear trennbar sind. Dies geschieht mit dem Kerneltrick κ :

$$\kappa(u, v) = \Theta(u) \cdot \Theta(v)$$

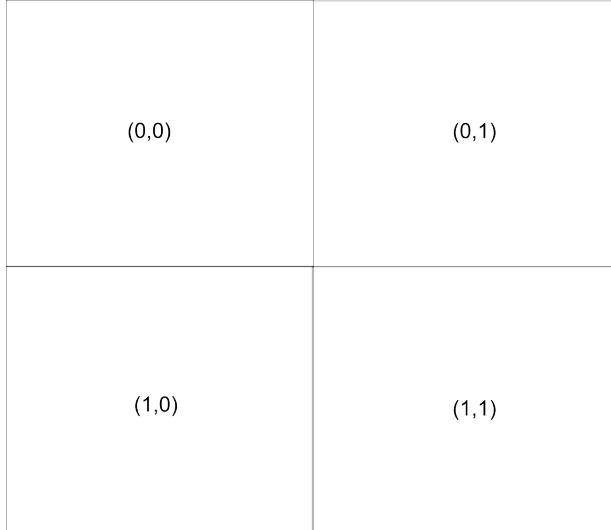


Abbildung 2.6: Ein in Unterabschnitte eingeteiltes Bild, dass nun die entsprechende Zuordnung der Merkmale ermöglicht

Hierbei versucht man eine Kernelfunktion κ zu finden, welche die Überführungsfunktionen in einen höherdimensionalen Raum beschreibt. Üblicherweise sind solche Kernelfunktionen polynomiell oder radialbasierte Funktionen [19].

Um nun gegebene Probleme mit mehreren Klassen statt nur zwei zu entscheiden, gibt es zwei verschiedene Ansätze. Einmal den “one vs all” und den “one vs one” Ansatz. Beim “one vs all” Ansatz wird über alle Klassen K iteriert und entschieden, ob das Ergebnis in der jeweiligen Klasse a oder nicht in Klasse a liegt, wobei $a \in K$ gilt. Beim “one vs one” Ansatz werden alle möglichen Paare von Klassen gebildet und jeweils eine Entscheidung getroffen. Da für jedes Paar an Klassen eine Entscheidung getroffen wird, kann anschließend eine Entscheidung für alle Klassen gemacht werden. Damit eignet sich der “one vs one” Ansatz besonders gut für Klassifikationsaufgaben.

2.2.6 Lage der Visual Words

Statt sich nur auf die Anzahl der Merkmale in einem Bild zu konzentrieren, kann auch deren Lage zur Klassifikation genutzt werden. Dazu kann ein Bild in mehrere Unterabschnitte (vgl. Abbildung 2.6) unterteilt werden. Um nun die Lage eines Merkmals hinzuzunehmen, wird geprüft in welchem Bereich des Bildes sich das Merkmal befindet und diese Information dem Merkmal hinzugefügt. Damit können nun diverse Fälle abgedeckt werden. Unter anderem könnte es vorkommen, dass das gezeigte Bild die Visual Words für Reifen und Fenster enthält. Eine mögliche Klassifikation könnte nun ergeben, dass auf dem Bild ein Auto zu sehen ist, weil ein Auto aus Reifen und Fenstern besteht. Durch die Lage der Visual Words würde aber bestimmt werden, dass die Reifen oberhalb der Fenster liegen, was

im Idealfall keinem Auto entspricht und somit muss sich ein anderes Objekt auf dem Bild befinden und man hat eine mögliche falsche Klassifikation verhindert. Der Erfolg durch diese Methode hängt jedoch stark von der gewählten Anzahl an Unterteilungen ab. Wählt man die Unterteilung zu grob, gewinnt man nahezu keine Informationen, wenn beispielsweise die Fenster und Reifen aus dem vorigen Beispiel in einem Abschnitt vorzufinden sind. Ist die Unterteilung jedoch zu fein, kann es auch zu falschen Klassifikationen kommen, da gewisse Merkmale eines Objektes, wie die Reifen eines Autos, zwar immer unten auftauchen, aber durch unterschiedliche Positionierung des Objektes im Bild nicht immer an einer bestimmten Position (x, y) im Bild.

Um nun das erwartete Ergebnis zu erhalten, werden dem Wertevektor v des Deskriptors zwei weitere Dimensionen für die x - und y -Koordinate hinzugefügt. Wie in [7] vorgestellt, müssen die xy-Koordinaten den Deskriptor dominieren, um das gewünschte Verhalten zu erzielen. Dazu wird die Summe aller 128-Dimensionen des SIFT-Deskriptors auf ungefähr 1 skaliert, indem jeder Wert des Deskriptors durch die durchschnittliche Länge eines Deskriptors geteilt wird. Somit werden die Werte des Deskriptors durch die xy-Koordinaten dominiert. Der neue Deskriptor wird mit dem folgenden Wertevektor v beschrieben: $v = (a_0, a_1, \dots, a_n, q(x), q(y))$ Dabei sind a_0, \dots, a_n die neuen Werte des Vektors und $q(x)$ und $q(y)$ die quantisierten Koordinaten des Deskriptors. Im Falle von Abbildung 2.6 also entweder $(0, 0)$, $(0, 1)$, $(1, 0)$ oder $(1, 1)$.

2.3 Räumliche Informationen

Die bisher vorgestellten Ansätze konzentrierten sich nur auf die in den Bildern auffindbaren Visual Words, um eine Klassifikation der Daten zu ermöglichen. In realen Anwendungen funktioniert dies zwar, jedoch können die Ergebnisse durch die Hinzunahme weiterer Informationen verbessert werden. Eine Möglichkeit ist die Hinzunahme von räumlichen Informationen. Im folgenden soll der Informationsgewinn durch Tiefenbilder in Betracht gezogen werden.

2.3.1 Tiefenbilder

Eine weitere Möglichkeit für zusätzliche Informationen bieten Tiefenbilder. Hierbei wird das Bild nicht als 2D-RGB Bild aufgenommen, sondern mithilfe spezieller Geräte in 3D aufgenommen. Dabei beschreiben die Pixel des Bildes nicht mehr die Lichtintensität in diesem Punkt, sondern den Abstand von dem Aufnahmegerät [23]. Dadurch kann man bestimmen, ob sich Objekte eher im Vordergrund oder im Hintergrund des Bildes befinden oder ein Objekt teilweise von einem anderen verdeckt wird. Für die Aufnahme solcher Bilder gibt es zwei unterschiedliche Verfahren, die aktiven und passiven. Ein bekanntes passives Verfahren ist die Verwendung von zwei Kameras im Verbund, womit sich die Tiefe recht einfach bestimmen lässt, wenn zwei korrespondierende Punkte gefunden wurden. Zwei



Abbildung 2.7: Bild des Kinect Infrarot-Musters

Punkte in den beiden aufgenommenen Bildern werden korrespondierend genannt, wenn sie die Projektion derselben Struktur im Bild repräsentieren [23]. Das Problem hierbei ist aber, dass im Normalfall für einen Punkt in einem Bild mehrere korrespondierende Punkte in dem anderen in Frage kommen. Die richtige Zuweisung dieser Punkte ist daher ein großes Problem und kann nur durch global erhältliche Kontextinformationen gelöst werden. Wird dieses aber gelöst, kann mithilfe einfacher Berechnungen der Abstand errechnet werden und so auch zusammenhängende Strukturen bestimmt werden. Dabei kann es passieren, dass Strukturen welche in Bild 1 sichtbar sind, in Bild 2 von etwas anderem verdeckt werden. Dies kann dann zu lückenhaften Repräsentationen führen. Eine Lösung des Problems wäre die Hinzunahme einer weiteren Kamera, die dann möglicherweise das Objekt aus Bild 1 wieder enthält. In der Praxis gewährleistet aber auch dies nicht die komplette Abdeckung der Szene.

2.3.2 Microsoft Kinect

Die Microsoft Kinect Kamera wurde ursprünglich als weiteres Eingabegerät für Microsofts Videospielkonsole Xbox 360 konzipiert, um damit das Betriebssystem, sowie auch Spiele über Gesten steuern zu können. Dazu besitzt die Kinect über ein Mikrofon-Array aus vier Mikrofonen, einem CMOS-Farbsensor und einer Infrarot-Kamera für die Tiefenbilder [9]. Sowohl Farb- als auch Tiefenbilder werden in einer Auflösung von 640x480 Pixel mit einer Bildwiederholfrequenz von 30Hz erzeugt.

Die Tiefenbilder werden mithilfe der Infrarot-Kamera erstellt. Dabei tastet die Kinect die Szene mit einem Infrarot-Laser ab und legt ein ihr bekanntes Infrarot-Muster über die Szene. Anhand der Verformung bzw. Verschiebung des Musters kann nun der Abstand zu einem Punkt berechnet werden [11]. Jedoch beträgt der minimale Abstand zum Sensor ca. 50cm, da bei kleineren Entferungen das Infrarot-Muster überblendet. Im nahen Bereich beträgt der Fehler bei der Messung ca. 1-2mm und steigt anschließend quadratisch zur Entfernung an. Bei Entfernungen über 5m kann es zu unbrauchbaren Ergebnissen kommen, da das Infrarot-Muster zu schwach und nicht mehr erkannt wird. Dies hängt jedoch von

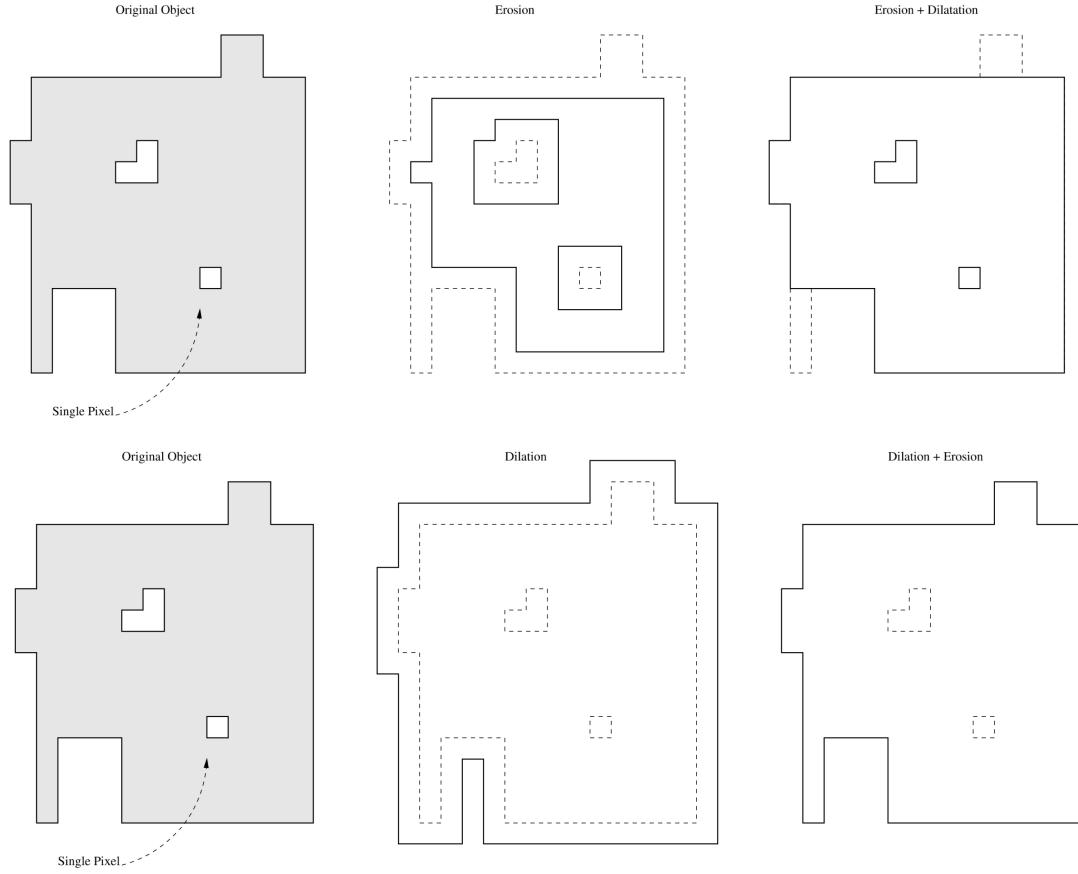


Abbildung 2.8: Oben ist der Effekt einer Erosion, sowie des Openings zu sehen und unten der Dilatation und des Closings. Entnommen aus [4]

diversen Variablen, wie der Belichtung ab. Außerdem kann die Tiefe von Oberflächen nicht bestimmt werden, die das Infrarot-Muster reflektieren oder absorbieren, sodass die Tiefe von Gläsern, Flaschen, Fenstern oder auch Laptops nicht entsprechend berechnet werden kann. Zudem kommt es zu Schattenwürfen im Tiefenbild, in denen keine Tiefendaten erfasst werden, da zwischen dem Infrarot-Laser und der Infrarot-Kamera eine Verschiebung besteht, welche für diesen Schatten zuständig ist.

2.4 Bildverarbeitung

Um Bilder mit den Bag of Features Ansatz klassifizieren zu können, sind unterschiedliche Vorverarbeitungsschritte notwendig. Im folgenden werden Filteroperationen und die Segmentierung erläutert.

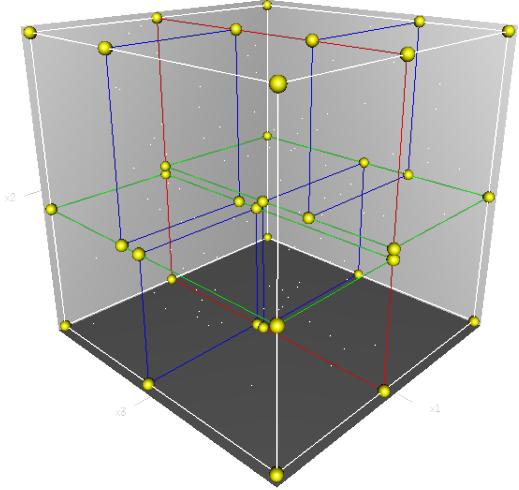


Abbildung 2.9: Zu sehen ist die Visualisierung eines dreidimensionalen kd-Trees. Dabei teilt die rote Fläche den Würfel in zwei Hälften, die grüne in zwei weitere Unterhälften und die blaue noch weiter ein. Somit lässt sich nun anhand der Koordinaten schnell feststellen, in welchem Teilwürfel sich ein Punkt befindet. (http://en.wikipedia.org/wiki/K-d_tree)

2.4.1 Filter

Obwohl Grauwertbilder scheinbar komplett Flächen in schwarz oder anderen Grautönen besitzen, sind viele Flächen nicht komplett geschlossen. Um Flächen nun zu schließen, zu vergrößern oder auch zu verkleinern, kommt das sogenannte Opening und Closing zum Einsatz, ein Verfahren aus der Bildverarbeitung. Dazu müssen die Begriffe Erosion und Dilatation erklärt werden.

Bei der Erosion wird die Umgebung eines jeden Pixels des Bildes untersucht und der Minimalwert der Umgebung dem Pixel zugewiesen, sodass dunkle Bereiche des Bildes vergrößert und hellere verkleinert werden. Die Dilatation arbeitet genau im Gegensatz und sucht den Maximalwert der Umgebung, um hellere Bereiche zu vergrößern.

Das Opening beschreibt nun den Vorgang der Erosion gefolgt von einer Dilatation. Dabei werden schmale helle Bereiche eines Bildes entfernt und dunkle beibehalten. Das Closing beschreibt wiederum den Vorgang der Dilatation gefolgt von einer Erosion, um Löcher in hellen Bereichen zu schließen.

2.4.2 Segmentierung

Der Begriff der Segmentierung befasst sich damit inhaltlich zusammenhängende Regionen auf einem Bild zu erkennen. Dazu stehen unterschiedliche Verfahren zur Verfügung. Ein besonders einfaches und schnelles Verfahren bilden pixelorientierte Verfahren. Hierbei wird für jeden Pixel eines Bildes die Entscheidung getroffen, ob er zu einem Segment gehört oder nicht. Bei einem Grauwertbild kann dies mit einem Schwellenwert erreicht werden.

Dabei wird für jeden Pixel geprüft, ob er über oder unter dem vorher festgelegten Schwellenwert liegt und je nach Fall wird der Pixel abgeändert. So können zum Beispiel ansonsten schwammig erkennbare Formen klar extrahiert werden. Für den Erfolg des Verfahrens ist jedoch die Wahl des Schwellenwertes entscheidend. Bei einem Grauwertbild kann zum Beispiel der Mittelwert aus dem hellsten und dunkelsten Pixel im Bild berechnet werden. Eine weitere Möglichkeit sind Kantenorientierte Verfahren. Hierbei wird nach zusammenhängenden Kanten im Bild gesucht, welche sich mit dem Sobel-Operators finden lassen. Dieser gewichtet den Bereich um einen Pixel $x_{i,j}$ entsprechend folgender Formel [16]:

$$f_{x_{i,j}} = |(x_{i-1,j-1} + 2x_{i,j-1} + x_{i+1,j-1}) - (x_{i-1,j+1} + 2x_{i,j+1} + x_{i+1,j+1})| + |(x_{i-1,j-1} + 2x_{i-1,j} + x_{i-1,j+1}) - (x_{i+1,j-1} + 2x_{i+1,j} + x_{i+1,j+1})|$$

$f_{x_{i,j}}$ weist so dem Pixel $x_{i,j}$ anhand einer 3x3 Matrix um den Pixel einen neuen Grauwert zu.

2.5 kd-Tree

Der Quantisierungsprozess im Bag of Features Ansatz verlangt das zu jedem Deskriptor in einem Bild der nächste Nachbar gefunden wird und diese Werte dann in einem Histogramm abgelegt werden. Das führt zu einer Laufzeit im Worst-Case bei einem naiven Ansatz von $O(n)$, wenn ein Distanzmaß, wie die euklidische Distanz eingesetzt wird. Die Laufzeit kann bei Einsatz eines kd-Trees jedoch verbessert werden. Dieser hat zwar im Worst-Case auch eine Laufzeit von $O(n)$, jedoch ist Durchschnittlich eine Laufzeit von $O(\log n)$ möglich. Dabei stellt der kd-Tree eine Datenstruktur zur Verfügung, welche Punkte im k -dimensionalen Raum ordnet. Grundlegend ist der kd-Tree jedoch ein simpler Binärbaum, wobei hier die Knoten eine Spaltung des Raums mithilfe einer Hyperebene symbolisieren. Dazu wird jedem Knoten eine der k Dimensionen zugewiesen und wenn an einem Knoten A nun der Raum in der X -Dimension gespalten wird, haben alle darauffolgenden Knoten im linken Teilbaum einen kleineren X -Wert als A und im rechten Teilbaum einen größeren X -Wert. Eine bildhafte Veranschaulichung der Datenstruktur ist in Abbildung 2.9 zu sehen.

Kapitel 3

Verwandte Arbeiten

Im folgenden werden nun zwei verwandte Arbeiten aus dem Bereich des Bag of Features Verfahrens vorgestellt, die sich mit unterschiedlichen Themen aus dem Gebiet befasst haben. Dazu werden *Introduction to the Bag of Features Paradigm for Image Classification and Retrieval* von Stephen O'Hara und Bruce A. Draper und *Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories* von Svetlana Lazebnik, Cordelia Schmid und Jean Ponce vorgestellt.

3.1 Introduction to the Bag of Features Paradigm

In *Introduction to the Bag of Features Paradigm for Image Classification and Retrieval* befassen sich Stephen O'Hara und Bruce A. Draper mit dem Bag of Features Ansatz. Dabei geben Sie eine Einführung in die Thematik des Bag of Features Ansatzes, beschreiben entscheidende Designentscheidungen und untersuchen diverse Literatur zu dem Thema. Sie gehen insbesondere auf die Verbesserungsmöglichkeiten der Quantisierung, sowie die verschiedenen Anwendungsgebiete des Bag of Features Ansatzes ein. Es werden im folgenden mithilfe der von O'Hara und Draper vorgestellten Ergebnisse die Anwendungsgebiete näher erläutert.

3.1.1 Bild Klassifikation

Ein Gebiet, in dem der Bag of Features Ansatz eingesetzt wird, ist die Bild Klassifikation. Hierbei werden wie z.B. bei der Objekt oder Szenen Klassifikation auf einem ganzen Bild Merkmale berechnet und darauf das entsprechende Bild klassifiziert. Auf Grund fehlender räumlicher Informationen eignet sich der Bag of Features Ansatz nur bedingt für die Detektion von Objekten in einem Bild. Wenn jedoch, wie in der Caltech101 Datenbank alle Bilder nahezu die gesamte Bildfläche einnehmen, kann auch von einer Objektklassifikation gesprochen werden. Andernfalls ist es erforderlich durch eine Segmentierung Bildausschnitte für die Klassifikation der Objekte zu ermitteln.

Dies steht im Gegensatz zu Teilbasierten Modellen (in der Regel HOG-Deskriptoren) die über ein Bild bewegt werden, um Objekte zu detektieren. Hierbei wird für jedes Objekt ein Modell trainiert, sodass die Detektion die Klassifikation beinhaltet.

Für die Bild Klassifikation kommen oftmals statt der SIFT-Deskriptoren auch Gist Deskriptoren zum Einsatz. Diese berechnen anders als ein SIFT-Deskriptor für das gesamte Bild einen einzigen globalen Deskriptor, der im Falle vom Holistic Spatial Envelope 512 Dimensionen umfasst. Dies hat zur Folge, dass Gist Deskriptoren deutlich schneller und einfacher zu ermitteln sind, aber dafür viele Informationen durch die grobe Beschreibung eines Bildes verloren gehen.

Ein weiteres Gebiet ist die Ortslokalisierung. Statt der Verwendung eines GPS-Sensors wird mithilfe der Bild Klassifikation der Ort der Aufnahme über die visuelle Ähnlichkeit ermittelt. Dazu muss bestimmt werden zu welchem Ort das aufgenommene Bild gehört, wozu lange Zeit SIFT-Deskriptoren miteinander verglichen wurden. Mittlerweile hat sich auch hier der Bag of Features Ansatz durchgesetzt.

3.1.2 Bild Retrieval

Ein weiteres Einsatzgebiet ist das Bild Retrieval. Hierbei ist es die Aufgabe zu einem gegebenen Bild ein möglichst ähnliches Bild aus einer Galerie oder Datenbank zu finden. Dabei wird zwischen *Content-Based Image Retrieval* (CBIR), bei der man Bilder mit ähnlichem Inhalt sucht, und dem allgemeinen Bild Retrieval, bei dem nach einem ähnlichen Gesamtbild gesucht wird, unterschieden. Letzterer wird mithilfe des Bag of Features Ansatzes implementiert und nutzt ein einfaches Distanzmaß zwischen zwei Bildern und benötigt somit keinerlei Trainingsphase. Dies ist ähnlich zum bereits vorgestellten Nächster Nachbar Verfahren (Vgl. Unterabschnitt 2.2.4) bei dem oftmals der kd-Tree eingesetzt wird. Meist handelt es sich beim Retrieval aber um approximative nächste Nachbarn Verfahren.

Dagegen ist beim CBIR von Nöten, dass man für alle Objekte Detektoren entwickelt, um somit in einer Galerie mithilfe der Detektionsergebnisse den gesamten Datensatz zu indexieren. Damit lässt sich eine Anfrage beim Retrieval als Kombination von gewünschten Objekten ausdrücken, welche dann mit den indexierten Bildern verglichen werden. So lässt sich das Retrieval einfach realisieren. Das Problem beim CBIR ist aber, dass für alle Objekte Detektoren entwickelt und validiert werden müssen, sodass im Vorfeld viel Arbeit entsteht und sich das Verfahren somit nur für begrenzte Datensätze eignet.

3.1.3 Anwendung

Eine im Paper vorgestellte Anwendung des Bild Retrieval ist das Video Google Paper. Darin wurde versucht mittels *MSER* und *Harris-Affine* Keypoint Detektoren Merkmale zu detektieren, welche durch SIFT-Deskriptoren repräsentiert wurden. Ein Vokabular wurde mittels k-means Clustering (s. Unterabschnitt 2.2.3) generiert und der nächste Nachbar

zu den berechneten SIFT-Deskriptoren mittels euklidischer Distanz und einer Gewichtung der Merkmale ermittelt. Dabei wurde gezeigt, dass sich der Bag of Features Ansatz für das Bild Retrieval zwar eignet, aber in dieser Form nicht mit großen Datensätzen zuverlässig arbeiten kann. Um beispielsweise das Internet zu durchsuchen, müssen effizientere Vorgehen entwickelt werden. Im Video Google Paper wurden ca. 4000 Bilder zum Retrieval genutzt und das Vokabular umfasste 10.000 Merkmale. Um Anfragen nun effizient zu gestalten, wurde ein invertiertes Dateisystem genutzt, sodass die Anfrage ca. 0,1 Sekunden für eine Antwort benötigte. Die Idee des invertierten Dateisystems ist, dass sich jedes Merkmal aus dem Vokabular merkt in welchen Bildern und wie oft es auftaucht. Um jetzt ein Anfrageergebnis zu erhalten, werden die Merkmale auf einem Bild berechnet und mittels des invertierten Dateisystems für die auftauchenden Merkmale die Liste der Bilder herausgesucht, in denen sie auftauchen. Nun kann zu jedem Bild im invertierten Index die Distanz aufsteigend berechnet werden, um die Anfrage auszuwerten. Eine andere Möglichkeit bietet der Stop-Words-Ansatz, bei dem zu häufig auftauchende Merkmale entfernt werden, um ähnlich wie in Dokumenten Wörter wie z.B. "ein" oder "das" zu streichen, welche keine Relevanz für das Dokument haben.

Eine weitere Verbesserung der Performance wurde von Nister und Stewenius ermöglicht, indem das Wörterbuch der Visual Words als Baumstruktur gesichert wurde. Damit lassen sich bis zu 16 Millionen Blätter im Baum erreichen. Zur Generierung der Histogramme wurde an jedem Knoten im Baum ein invertiertes *Index Listig* genutzt. Dagegen stellt Philbin einen approximativen k-means Clustering Algorithmus zur Erstellung des Wörterbuchs vor, welcher zufällig generierte kd-Trees nutzt. Der kd-Tree wird anschließend auch zur Quantisierung verwendet.

Weitere Probleme sind Anfrageergebnisse und die Erstellung des Wörterbuches. Für die Verbesserung der Ergebnisse wurden diverse Methoden vorgestellt, welche die gelieferten Ergebnisse analysieren und verbessern, indem beispielsweise versucht wird, wie von Sivic vorgestellt, die Merkmale zwischen dem Anfragebild und dem Ergebnisbild aufeinander abzubilden. Zwar verbessert dies die Ergebnisse, aber erhöht auch die Rechenzeit deutlich. Jegou hat 2010 die *Rank Aggregation* vorgestellt, bei der die Anfrage mehrmals mittels verschiedener Wörterbücher ausgewertet wird und die finale Anfrage dann beispielsweise den durchschnittlichen Rank der Bilder in den Ergebnissen ergibt. Dies erhöht aber die Rechenzeit, welche durch Parallelisierung der Anfragen umgangen werden kann und sorgt auch für höheren Speicherbedarf durch die unterschiedliche Indexierung der Ergebnisse. Eine dritte Möglichkeit wird in Form der *Query Expansion* von Chum vorgestellt, welche die besten Ergebnisse einer Anfrage als neue Anfrage stellen, wodurch sich die Bilder immer weiter annähern. Um jedoch falsche ausgegebene Bilder zu umgehen, wird eine räumliche Konsistenz der Bilder vorausgesetzt. Dies ist ein wichtiger Schritt, da die Anfragen sonst schlechtere Ergebnisse liefern, als ohne *Query Expansion*.

Das Problem der Erstellung des Wörterbuchs ist das so erstellte Wörterbücher oftmals

nur für einen geringen Datensatz einsetzbar sind und sich so für eine Internetsuche nur geringfügig eignen. Eine abschließende Antwort auf diese Frage konnte noch nicht gefunden werden, jedoch stellte Nowak vor, dass ein aus zufällig generierten SIFT-Deskriptoren erstelltes Wörterbuch für die Bild Klassifikation gute Ergebnisse erzielt, die vergleichbar sind mit den für den jeweiligen Datensatz erstellten Wörterbüchern. Möglicherweise könnte dieses Vorgehen für Internetsuchen eingesetzt werden.

Aktuell bietet Google bereits eine *Reverse Image Search* an, womit Bilder hochgeladen werden können und die Google Bildersuche findet dazu passende ähnliche Bilder und findet auch eine Beschreibung des Bildes.

3.1.4 Fazit

Obwohl der Bag of Features Ansatz ein einfaches und schnelles Vorgehen für die oben vorgestellten Anwendungen darstellt, konnten O'Hara und Draper Probleme erkennen, die den Einsatz in anderen Gebieten, insbesondere der Objekterkennung, erschweren.

Ein Problem sei das Fehlen von räumlichen Informationen in herkömmlichen Bag of Features Repräsentationen. Somit lassen sich Objekte nur schwer in Szenen detektieren und auch die Ausgabe von relationalen Konzepten, wie "Person steht neben einem Auto", können kaum realisiert werden. Es gäbe zwar Ansätze dazu, jedoch entfernen diese sich vom Bag of Features Ansatz und verlieren dadurch oftmals die Einfachheit des Konzeptes.

Ebenfalls problematisch ist das Fehlen einer semantischen Bedeutung der Visual Words. Das führt zu Problemen in der Evaluation des Bag of Features Ansatzes. Denn durch die fehlende semantische Bedeutung ist es nicht möglich zu bestimmen, was der Bag of Features Ansatz in Wirklichkeit erkennt. Zwar mag der Ansatz auf einer gegebenen Datenbank sehr gute Ergebnisse liefern, aber diese Ergebnisse in einem leicht anderen Set nicht bestätigen können. Beispielsweise können statt der eigentlichen Objekte, wie ein Gesicht, Dinge im Hintergrund erkannt werden, die über das Set an Daten ähnlich bleiben, aber sich in einem anderen kaum wiederfinden. Somit kommen O'Hara und Draper zu dem Schluss, dass der Bag of Features Ansatz ein außergewöhnlich einfaches und schnelles Verfahren für visuelle Aufgaben ist, dass gute Ergebnisse auf diversen Datensätzen liefert, aber insbesondere bei komplexen Szenen noch problematisch ist, da keine Detektion oder Segmentierung erfolgt.

3.2 Spatial Pyramid Matching

In *Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories* befassen sich Svetlana Lazebnik, Cordelia Schmid und Jean Ponce mit dem Verfahren des Spatial Pyramid Matching, um natürliche Szenen zu klassifizieren. Im folgenden wird deren Idee, Umsetzung und Ergebnisse vorgestellt.

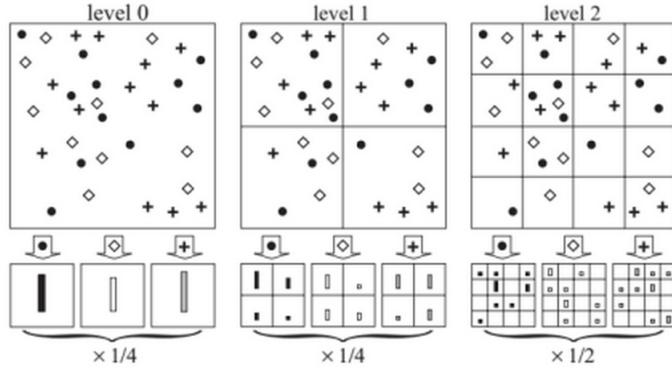


Abbildung 3.1: Visualisierung der Level des Spatial Pyramid Matching einer 3-Level Pyramide. Das dargestellte Bild besteht aus drei unterschiedlichen Merkmalstypen (visualisiert durch Kreise, Rauten und Kreuze). Oben sind die drei unterschiedlichen Level mit den jeweils feineren Aufteilungen des Bildes zu sehen. Darunter wird die Häufigkeit des Auftauchens der Visual Words in einem Teil des Bildes dargestellt. Entnommen aus [22].

3.2.1 Idee

Die in der Arbeit vorgestellte Idee basiert darauf statt eines Histogramms für das gesamte Bild mehrere Histogramme für Teile des Bildes zu berechnen. Beim Spatial Pyramid Matching wird deshalb die Auflösung des Bildes, sowie der Merkmale nicht geändert, sondern nur die Auflösung der Ausschnitte. Dazu wird das Bild in mehrere Stufen jeweils immer feiner unterteilt und auf jedem Teilbild ein Histogramm der vorzufindenden Visual Words berechnet (s. Abbildung 3.1). Um die nächsten Nachbarn zwischen zwei Visual Word Histogrammen zu berechnen, wird das Verfahren des Pyramid Matching angepasst. Dort kann die Anzahl an Übereinstimmungen zwischen zwei Merkmalssätzen X und Y mit folgender Ähnlichkeitsfunktion berechnet werden:

$$I(H_X^l, H_Y^l) = \sum_{i=1}^D \min(H_X^l(i), H_Y^l(i))$$

Dabei gibt es $0, \dots, L$ Level mit jeweils feiner aufgelösten Rastern über dem Bild, die auf Level l 2^l Zellen entlang einer Bildseite haben. Somit gibt $H_X^l(i)$ das Histogramm von X in der Auflösung l und den Punkten in der i -ten Zelle eines Rasters wieder. Im folgenden wird $I(H_X^l, H_Y^l)$ nun I^l genannt. Anschließend werden die Übereinstimmungen gewichtet, da Übereinstimmungen in größeren Zellen ungenauere Bildbeschreibungen enthalten, als feinere. Das Gewicht in einem Level l ist $\frac{1}{2^{L-l}}$. Somit ergibt sich zusammengefasst für die Übereinstimmungen:

$$\kappa^L(X, Y) = \frac{1}{2^L} I^0 + \sum_{l=1}^L \frac{1}{2^{L-l+1}} I^l$$

Das Matching von zwei Bildern X und Y mit L Leveln lässt sich dann als Summe der Übereinstimmungen einzelner Merkmale definieren:

$$K^L(X, Y) = \sum_{m=1}^M \kappa^L(X_m, Y_m)$$

3.2.2 Auswertung

Für die Auswertung wurden drei Datensätze herangezogen auf denen die Klassifikationsraten mit jeweils 16, 200 und 400 Visual Words, sowie 0 bis 3 Leveln geprüft wurde. Dabei stellte sich heraus, dass die vorigen Ergebnisse mittels Bag of Features Ansatz immer übertroffen werden konnten. Beispielsweise konnten auf einem Datensatz mit 15 natürlichen Szenen die Klassifikationsraten um mehrere Prozent verbessert werden. Dies konnte auch auf der Caltech101 Datenbank erzielt werden, wobei die Performance von der jeweiligen Klasse abhing. Probleme machten dort Klassen mit Tieren, die entweder sehr dünn waren, wie Ameisen oder nur schwer vom Hintergrund zu separieren sind, wie Krokodile. Auf der Graz-Datenbank wurden die vorher besten Ergebnisse jedoch nicht ganz erreicht. Diese besteht aus drei Klassen (Fahrräder, Menschen und Hintergrund) und in der Arbeit wurde ein Ansatz verfolgt, der zwischen zwei Klassen (Objekte, Hintergrund) unterscheiden sollte. Auf Grund der hohen Variabilität in den Bildern ist es für das Verfahren des Spatial Matching schwer geeignete globale Visual Words zu finden, sodass ein leichter Rückgang in den Raten zu verzeichnen ist. Dafür ließ sich feststellen, dass eine Erhöhung der Visual Words von 200 auf 400 und von 2 auf 3 Leveln nur geringfügig auf die Ergebnisse auswirkte. Das Optimum aus Klassifikationsraten und Laufzeit lag somit bei den getesteten Datensätzen bei 200 Visual Words im Wörterbuch mit 2 Leveln in der Pyramide.

Kapitel 4

Methodik

In Kapitel 4 wird zuerst die Aufgabenstellung der Arbeit und dann deren Umsetzung präsentiert.

4.1 Aufgabenstellung

Im Rahmen der Arbeit soll eine Anwendung entstehen, welche mithilfe der im Grundlagen-Kapitel vorgestellten Konzepte eine Objektklassifikation vornehmen soll. Dazu soll mit einer Kamera ein Video einer realen Umgebung aufgenommen und auf diesem dann eine Klassifikation der dort zu sehenden Objekte mittels Bag of Features Ansatz ermittelt werden. Dabei soll die Klassifikation im Idealfall ähnlich schnell arbeiten, wie ein Mensch zur Erkennung des Objektes brauchen würde. Wie in Abschnitt 2.2 vorgestellt, wird für die Klassifikation mittels Bag of Features Ansatz eine Trainingsbasis benötigt. Diese soll ebenfalls für die Anwendung selbst zusammengestellt werden, wobei das Training der Klassen bereits durch ein vorhandenes Framework zur Verfügung gestellt wird.

4.2 Idee der Umsetzung

Wie in Abschnitt 4.1 dargestellt, müssen für die Anwendung folgende Schritte echtzeitfähig realisiert werden.

1. Zusammenstellung einer Trainingsbasis
2. Detektion von Objekten in einer realen Szene ermöglichen
3. Klassifikation der Objekte realisieren
4. Interface für Nutzerinteraktionen

Im folgenden werden die einzelnen Punkte näher erläutert und die Ideen zur Umsetzung der Probleme vorgestellt.

4.2.1 Zusammenstellung einer Trainingsbasis

Für die Zusammenstellung einer Trainingsbasis müssen mehrere Entscheidungen getroffen werden. Entscheidend ist dabei unter anderem die Anzahl der zu erkennenden Objekte. Es gibt zwar schon diverse Datenbanken, wie Caltech101 [12] mit 101 unterschiedlichen Klassen oder VOC2012 [6] mit 20 Klassen. Es gibt jedoch mehrere Probleme mit diesen, da zum Beispiel die Caltech101 Datenbank zwar 101 verschiedene Klassen enthält, aber davon die Mehrzahl aus verschiedenen Tieren, Pflanzen oder auch Symbolen besteht, die sich für eine reale Anwendung eher weniger eignen. Zudem bietet die große Anzahl an Klassen auch viel Raum für Fehler bei der Klassifikation, da bei einer kleineren Anzahl an Klassen schon bei zufälliger Klassifikation die Erfolgsrate höher ist. Beispielsweise beträgt bei 101 Klassen die Wahrscheinlichkeit die richtige Klasse per Zufall auszuwählen gerade mal 0,9%. Bei 20 Klassen, wie sie in der VOC2012 Datenbank eingesetzt werden, beträgt die Wahrscheinlichkeit bereits 5% ohne jegliche Informationen über das Bild zu haben. Trotzdem eignet sich auch diese Datenbank nicht für den Einsatz in der geforderten Anwendung, da sie wiederum auf Fahrzeuge, Züge und andere ähnliche Gegenstände spezialisiert ist, welche mit einer Anwendung innerhalb einer intelligenten Umgebung, wie der FINCA¹ (A Flexible, Intelligent Environment with Computational Augmentation) nur sehr schwer zu Filmen sind. Daher wurde eine eigene kleine Basis an Trainingsbildern zusammengestellt, die aus 15 Klassen besteht, welche in einem normalen Haushalt vorzufinden sind. Die Klassen umfassen folgende Objekte:

Bücher	Kameras	Handys
Stühle	Tassen	Gesichter
Flaschen	Notebooks	PC-Mäuse
Scheren	Bälle	Tacker
Tastaturen	Regenschirme	Uhren

4.2.2 Detektion der Objekte in einer realen Szene

Ein weiteres Problem besteht in der Detektion der einzelnen Objekte in einer Szene. Eine Kamera nimmt ein Bild der Szene auf und im Normalfall sind viele Objekte darauf zu sehen. Daher ist es nötig im Vorfeld diese Objekte grob zu erkennen und dann einzeln zu klassifizieren. Ansonsten würde die Klassifikation des Bildes nicht funktionieren, da von allen Objekten die Visual Words, sowie die Hintergründe im Histogramm auftauchen würden. Dies führt dazu, dass die einzelnen Objekte nicht mehr genau klassifiziert werden können. Ermöglicht werden soll dieser Schritt durch den Einsatz einer 3D-Kamera, der Microsoft Kinect for Xbox Kamera (vgl. Unterabschnitt 2.3.2) [11]. Mit der Hinzunahme

¹Details zur FINCA sind zu finden unter: patrec.cs.tu-dortmund.de/cms/en/home/Research/SE

des Tiefenbildes der Szene, können Änderungen innerhalb dieser visualisiert werden. Dies wird folgendermaßen realisiert.

1. Aufnahme einer leeren Szene I_0
2. Für ein gegebenes Bild I , wird ein Differenzbild D berechnet durch

$$D(x, y) = |I_j(x, y) - I_0(x, y)| \quad \forall x, y \text{ mit Bildpixeln } x, y$$

3. Um einen Teil des Rauschens auf dem Differenzbild durch die Kinect Kamera im weit entfernten Hintergrund zu eliminieren, wird ein Schwellwertverfahren eingesetzt.

$$D(x, y) = \begin{cases} 0 & D(x, y) < s \\ 1 & \text{sonst} \end{cases}$$

In der Anwendung wird ein Wert von $s=5$ genutzt, welcher für den Einsatz in der FINCA ermittelt wurde. An anderen Einsatzorten muss möglicherweise ein anderer Schwellwert genutzt werden.

4. Um weiteres Rauschen zu entfernen, wird ein Opening auf dem Differenzbild durchgeführt.

Dabei muss jedoch gewährleistet werden, dass die Kamera ständig den selben Bildausschnitt aufnimmt und nur einzelne Objekte in der Szene sich ändern. Für einen Videofeed mit vielen beweglichen Gegenständen, wie im Straßenverkehr, oder einer sich bewegenden Kamera eignet sich dieses Vorgehen nicht. Wenn sich aber nun die Szene durch Ablegen eines Objektes ändert, wird dieses im Differenzbild sichtbar. Anschließend kann mit einer Segmentierung (vgl. Unterabschnitt 2.4.2) auf dem Differenzbild der Ort des Objektes im Bild bestimmt werden. Dies wird bewerkstelligt, indem zusammenhängende Konturen extrahiert werden. Jedoch werden Konturen, die kleiner als 200 Pixel groß sind bzw. 0,07% des Bildes ausmachen, ignoriert, da hier davon auszugehen ist, dass es sich nur um Sensorsrauschen handelt. Dies wurde in einem Test ermittelt und kann ebenfalls an anderen Einsatzgebieten unterschiedlich gewählt werden.

4.2.3 Klassifikation der Objekte

Nachdem mittels Segmentierung die Objekte erfasst wurden, kann auf dem entsprechenden Bildbereich eine Klassifikation mittels des Bag of Features Ansatzes vorgenommen werden. Dazu werden auf dem Differenzbild die minimalen und maximalen x - und y -Koordinaten der zusammenhängenden Kontur ermittelt, um damit auf dem RGB-Bild den entsprechenden Bildbereich auszuschneiden. Es muss jedoch beachtet werden, dass auf dem RGB-Bild

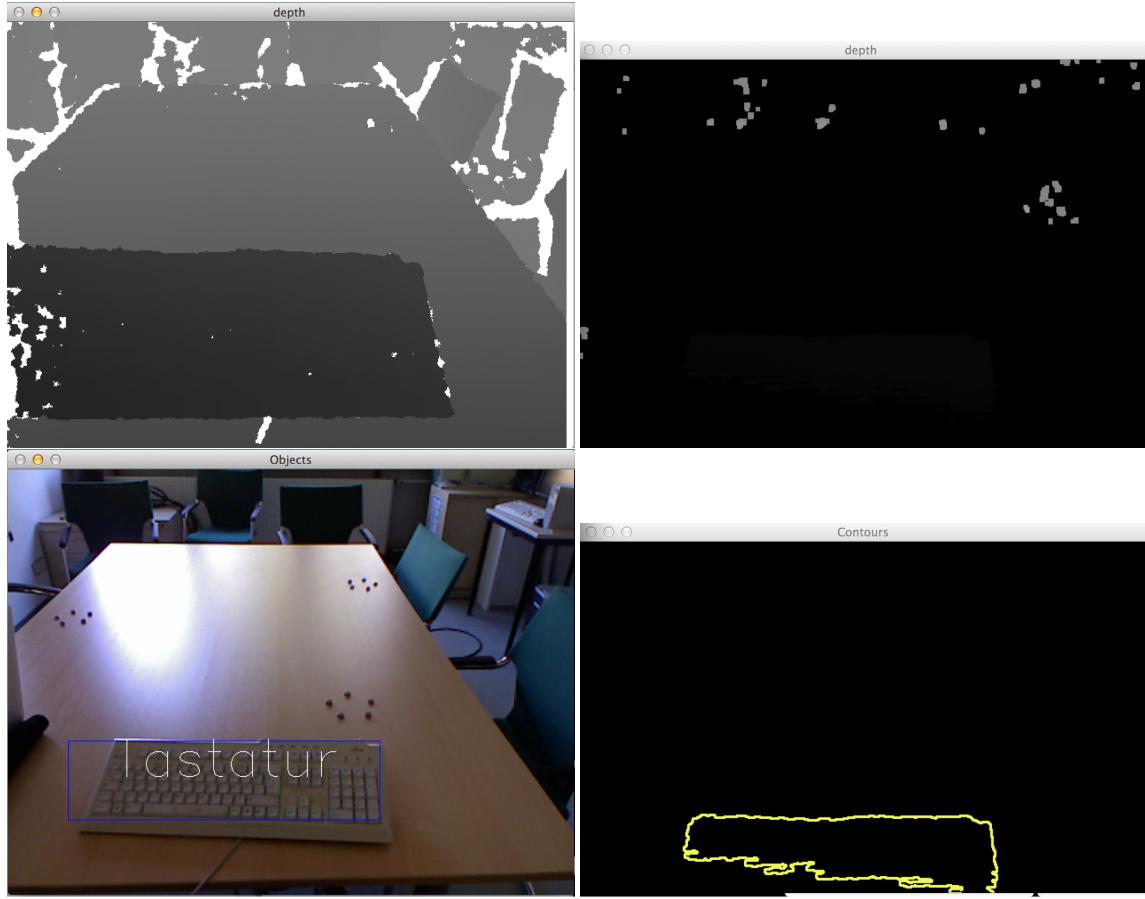


Abbildung 4.1: Oben links ist ein uninitialisiertes Tiefenbild mit einer vor die Kamera gehaltenen Tastatur zu sehen, die auf Grund der Nähe dunkler ist, als der Hintergrund. Rechts oben ist dann ein initialisiertes Tiefenbild mit etwas Rauschen zu sehen. Unten links ist das RGB-Bild mit Klassifikationsergebnis verdeutlicht und rechts daneben ist der segmentierte Teil dargestellt.

der Bildausschnitt um 50 Pixel auf der x -Koordinate nach links verschoben werden muss, da zwischen dem Infrarot- und dem CMOS-Sensor ein Versatz besteht. Anschließend werden auf dem ausgeschnittenen Bildbereich SIFT-Merkmale auf einem Raster berechnet. Dieses Raster wird alle 8 Pixel berechnet und besteht aus einem 16 Pixel großen Bin. Danach wird eine Bag of Features Repräsentation mit räumlichen Informationen (Vgl. Unterabschnitt 2.2.6) des ausgeschnittenen Bildbereichs ermittelt und die Klassifikation mittels einer Support Vector Machine auf dem vorher erstellten Trainingsdatensatz (Vgl. Unterabschnitt 4.2.1) durchgeführt.

4.2.4 Interface

Das Interface der Anwendung muss bestimmte Aufgaben erfüllen. Einerseits soll es möglichst simpel gehalten werden, um den Nutzer mit unwichtigen Informationen nicht zu überfordern und andererseits soll es auch genügend Interaktionsmöglichkeiten bieten. Des-

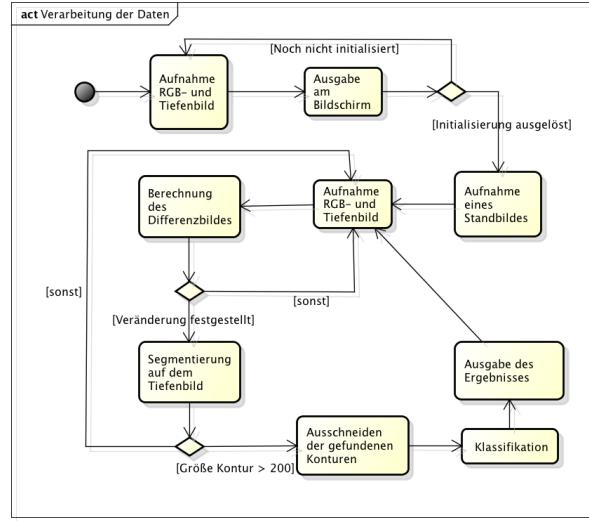


Abbildung 4.2: Die Verarbeitungsreihenfolge, welche durchlaufen wird, um die ankommenden Daten entsprechend der Aufgabenstellung zu bearbeiten.

halb wird auf Buttons in dem Interface verzichtet und die Anwendung nur per Tastatur gesteuert. Das Interface selbst besteht aus vier Fenstern. Einem Fenster für das normale RGB-Bild der Szene, eins für das aktuelle Tiefenbild bzw. nach der Initialisierung für das Differenzbild und eins auf dem die Segmentierung visualisiert wird. Auf dem vierten Fenster ist wieder ein RGB-Bild der Szene zu sehen, jedoch wird hier das Klassifikationsergebnis ausgegeben und das klassifizierte Objekt mit einem Rechteck eingerahmt (vgl. Abbildung 4.1).

Die Tastatureingaben beschränken sich auf drei Eingaben. Mittels der *i*-Taste kann die Anwendung initialisiert werden, was dazu führt, dass das aktuelle Tiefenbild gesichert wird und durch das Differenzbild in der Ausgabe auf dem Bildschirm ersetzt wird. Durch erneutes Betätigen der *i*-Taste kann nach versehentlicher Verstellung der Kamera ein neues Tiefenbild aufgenommen werden. Mit der *s*-Taste kann das aktuelle RGB-Bild auf der Festplatte gesichert und mit der ESC-Taste die Anwendung beendet werden.

4.3 Verwendete Bibliotheken

Zur Umsetzung der gegebenen Aufgabenstellung in Echtzeit sollen unterschiedliche C++ Bibliotheken zum Einsatz kommen. Hauptsächlich kommen die Bibliotheken OpenCV [5] und Vlfeat [1] zum Einsatz. OpenCV wird verwendet, weil dort eine Vielzahl an Datenstrukturen und Methoden bereitgestellt werden, um mit Bildquellen auf unterschiedliche Art und Weise zu interagieren und auf diesen dann auch Operationen auszuführen. Beispielsweise kann die Berechnung des Differenzbildes für die Segmentierung mit einer einfachen Subtraktion zweier Matrizen realisiert werden, da Bilder in OpenCV als Matrix

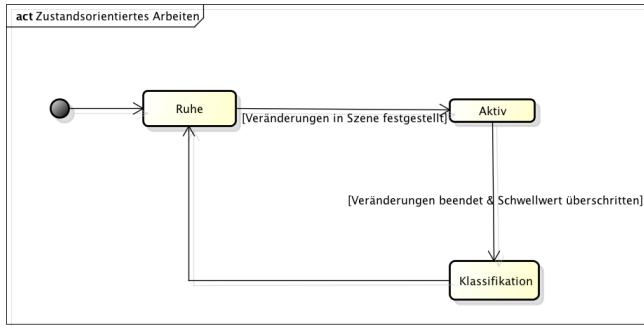


Abbildung 4.3: Zustandsdiagramm der Applikation, um Laufzeit niedrig zu halten

gespeichert werden können. In der VLfeat Bibliothek hingegen finden sich zahlreiche Implementierung von Standardalgorithmen aus dem Bereich Visual Computing und unter anderem auch die Berechnung der SIFT-Merkmale auf einem Raster oder das Arbeiten mit einem kd-Tree, was für die Geschwindigkeit der Anwendung von Vorteil ist. Weitere verwendete Bibliotheken umfassen die libSVM [2] und Open Kinect [17] Bibliothek. Erstere stellt alle essentiellen Methoden zur Verwendung einer SVM für die Klassifikation zur Verfügung und mittels Open Kinect Bibliothek lässt sich Microsofts Kinect Kamera in die Anwendung integrieren und damit dann unter anderem auch Tiefenbilder extrahieren.

4.4 Verarbeitung der Daten

Insgesamt gesehen werden während der Ausführung der Anwendung viele Schleifen für die Quantisierung oder auch Segmentierung durchlaufen. Um trotz der vielen Berechnungen ein flüssiges Bild auszugeben, wird nicht ständig eine Segmentierung des Bildes vorgenommen, sondern ein Zustandsbasiertes Vorgehen (vgl. Abbildung 4.3) eingesetzt. Hierbei wird nach der Aufnahme des Standbildes die Anwendung in einen Ruhezustand versetzt. Nun werden die nicht schwarzen Pixel, also die Veränderungen, gezählt und hat sich das Bild um mehr als ca. 6% der Pixel² verändert, wechselt man in den aktiven Zustand. Hier wird registriert, dass gerade vielleicht ein Objekt in die Szene gelegt wird oder möglicherweise jemand durch das Bild geht. Deshalb wird solange sich das Bild noch ändert keine Segmentierung gestartet. Erst wenn das Bild für 15 Bilder wieder still steht, wird die Segmentierung und anschließende Klassifikation gestartet. Erkannt wird der Stillstand dadurch das erkannt wird, dass in dem Bild weiterhin mehr als 6% der Pixel verändert wurden und sich die Pixel von Bild zu Bild maximal um 1500 Pixel unterschieden haben. Jedes aufgenommen Bild durchläuft dann eine Pipeline (vgl. Abbildung 4.2), um die Daten entsprechend zu verarbeiten. Am Anfang der Verarbeitungspipeline steht die Aufnahme

²Dieser Wert wurde in der FINCA ermittelt, indem eine ruhige Szene für zehn Minuten gefilmt und das durchschnittliche Rauschen des Bildes ermittelt wurde.

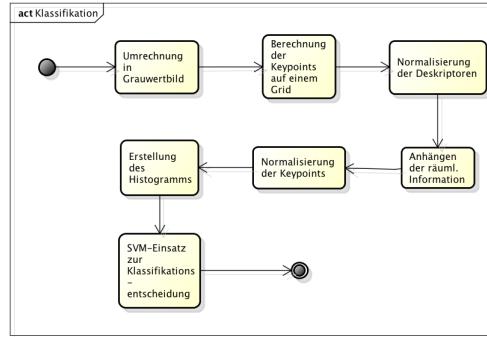


Abbildung 4.4: Die Klassifikation wird, wie in dem obigen Bild beschrieben, ausgeführt.

eines RGB- und eines Tiefenbildes. Diese werden direkt auf dem Bildschirm ausgegeben, wenn der Nutzer noch nicht die Initialisierung per Tastendruck auslöst. Wird diese ausgelöst wird ein Standbild als Tiefenbild aufgenommen, wovon anschließend das aktuelle Tiefenbild abgezogen wird, um so die Veränderungen im Bild sichtbar zu machen und eventuelle Objekte darzustellen. Veränderungen werden dann als weiße bzw. graue Pixel (je nach Abstand von der Kamera) dargestellt und können dementsprechend gezählt werden. Sollte hierbei der vorher vorgestellte Schwellenwert von 6% überschritten werden, startet eine Segmentierung auf dem Bild, um zusammenhängende Konturen in dem Differenzbild festzustellen. Diese werden dann dazu benutzt, um die entsprechenden Bildteile auszuschneiden und darauf eine Klassifikation durchzuführen. Es werden aber nur Segmente berücksichtigt, die eine bestimmte Größe (200 Pixel) überschreiten, um Sensorrauschen nicht zu klassifizieren. Auf diesen ausgeschnittenen Bildern wird nun eine Klassifikation vorgenommen, die dann auf dem RGB-Bild ausgegeben wird. Dazu wird das klassifizierte Objekt mit einem Rechteck eingegrenzt und darin dann auch das Ergebnis der Klassifikation ausgegeben.

Die Klassifikation (vgl. Abbildung 4.4) selbst wird folgendermaßen ausgeführt. Zuerst wird das ankommende Bild in ein Grauwertbild umgerechnet und auf diesem werden dann die Merkmale auf einem Raster berechnet (vgl. Unterabschnitt 2.2.2). Ausgehend davon werden die Deskriptoren erst normalisiert, um unterschiedliche Bildgrößen aufzufangen (vgl. Unterabschnitt 2.2.4). Dazu wird jeder Wert des Deskriptors mit 512 multipliziert und dann auf 255 begrenzt. Danach werden wie in Unterabschnitt 2.2.6 erklärt, die räumliche Information in Form der x - und y -Koordinate des Keypoints hinzugefügt. Anschließend werden Deskriptoren auf uniformen Regionen erkannt, indem die Grauwert-Differenz gemessen wird. Unterschreitet diese den Wert 0.005, wird der Wertvektor auf 0 gesetzt. Danach werden die Deskriptoren mit einem kd-Tree dem nächsten Visual Word zugeordnet, das Histogramm erstellt und mit einer SVM eine Klassifikationsentscheidung getroffen.

Kapitel 5

Auswertung

Im folgenden wird die Auswertung der während der Abschlussarbeit erarbeiteten Ergebnisse vorgestellt.

5.1 Metrik

Zur Ermittlung der Klassifikationsrate K einer Trainingsbasis T wurde folgende Metrik genutzt:

$$K(T) = \frac{\#(\text{korrekte Klassifikationen})}{\#(\text{Testbilder})}$$

5.2 Auswertung mittels Caltech101

Zur Verifizierung der Arbeitsweise bei der Klassifikation wurde die Caltech101 Datenbank hinzugezogen. Diese besteht aus 101 Kategorien mit jeweils 40 bis 800 Bildern, auf denen das entsprechende Objekt meist zentriert und im Vordergrund aufgenommen wurde. Zudem verfügen die Bilder über wenig Rauschen und meist sind die Objekte nicht durch Hände oder andere Gegenstände teilweise verdeckt, was für eine Klassifikation von Vorteil ist.

Für die Auswertung wurden alle 8677 Bilder aus der Datenbank verwendet und 3030 davon zufällig ausgewählt und für das Training genutzt. Die restlichen 5647 Bilder kamen zur Verifizierung der Ergebnisse zum Einsatz. Dabei wurden mehrere Verfahren zur Klassifikationsentscheidung eingesetzt, um die Eignung der verschiedenen Verfahren zu verdeutlichen (vgl. Abbildung 5.1). Für alle Klassifikationsmethoden wurden jedoch folgende Parameter verwendet. Jeder SIFT Deskriptor bestand aus einem Bin von 16 Pixeln und wurde jeweils um 8 Pixel verschoben, um die Merkmale auf einem Raster zu berechnen. Das Codebuch wurde mit 200 Visual Words antrainiert und wie für die Anwendung wurden die Deskriptoren normalisiert, indem die Werte mit 512 multipliziert und auf 255 begrenzt wurden. Unterschieden wird im folgenden zwischen den Verfahren mit der Berechnung der nächsten

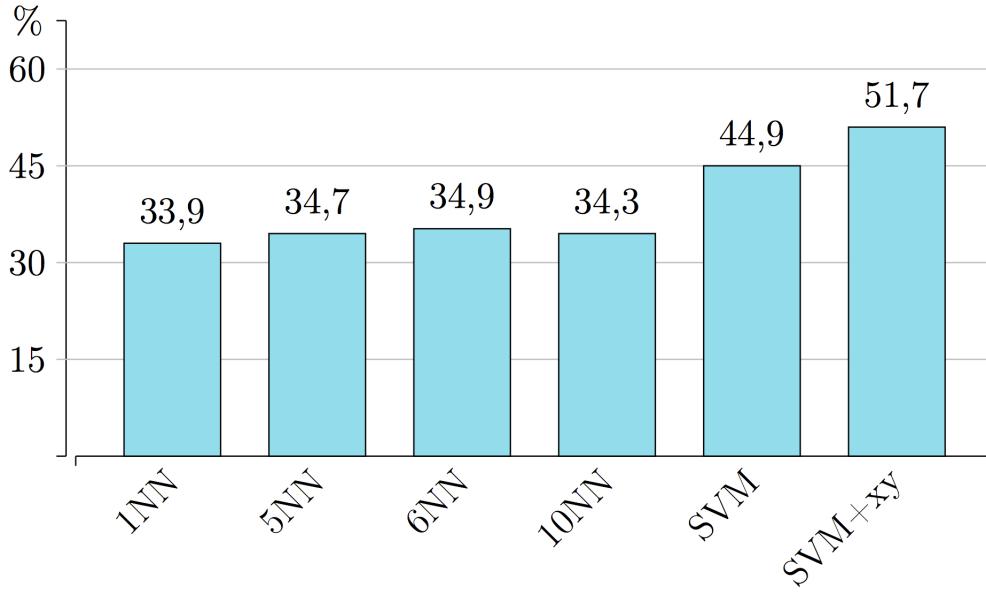


Abbildung 5.1: Diagramm, welches die Klassifikationsraten der jeweiligen Klassifikationsarten in Vergleich zueinander stellt. Zu sehen ist, dass bei Berechnung eines nächsten Nachbars ca. 35% erreicht werden können. Dies ändert sich auch mit Berechnung mehrerer nächster Nachbarn nur geringfügig. Erst der Einsatz einer SVM steigert die Ergebnisse auf über 50%.

Nachbarn und dem Einsatz einer SVM. Dabei ergab sich für die Berechnung eines nächsten Nachbars eine Klassifikationsrate von 33,9%. Um wie in Abbildung 2.5 vorgestellt auch gegenüber Ausreißern in der Verteilung unabhängig zu sein, wurde ebenfalls versucht mehrere nächste Nachbarn zu berechnen und eine Mehrheitsentscheidung zu treffen. Dabei ergab sich bei der Berechnung von sechs nächsten Nachbarn mit 34,9% ein lokales Maximum für das gegebene Trainingsset. Zum Vergleich wurden ebenfalls die Klassifikationsraten für fünf und zehn nächste Nachbarn dokumentiert, welche mit 34,7% und 34,3% nur etwas schlechter sind. Das die Berechnung noch mehr nächster Nachbarn als sechs nicht mehr von Vorteil ist, sondern eher negative Ergebnisse liefert, liegt daran, dass je nach Verteilung der Werte zu viele Werte einer falschen Klasse zugeordnet werden und somit die eigentlich richtige Klasse überdecken würden (vgl. Abbildung 5.2). Mittels Support Vector Machine konnten 44,9% der Bilder richtig zuordnen. Bei Hinzunahme der xy-Koordinaten zu den Deskriptoren konnte die Klassifikationsrate auf 51,7% gesteigert werden, wodurch mehr als die Hälfte aller Bilder richtig zugeordnet wurden. Diese Werte decken sich auch mit denen in [7] und verifizieren die richtige Implementierung des Bag of Features Ansatzes.

Um den Echtzeitaspekt der Klassifikation auszuwerten, wurde die Zeit gemessen, die benötigt wurde, um die 5647 Objekte bzw. Bilder zu klassifizieren (vgl. Abbildung 5.3). Dabei wurde zwischen einem Iterativen Vorgehen, dem kd-Tree und einer SVM unterschieden, jedoch wurde für die Quantisierung der Merkmale stets ein kd-Tree eingesetzt. Beim ite-

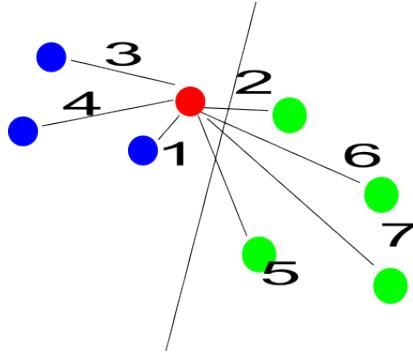


Abbildung 5.2: In rot ist das zu klassifizierende Objekt zwischen zwei Klassen (blau, grün) eingezeichnet. Bei der Berechnung von einem nächsten Nachbarn wird das Objekt richtigerweise der blauen Klasse zugeordnet. Dies ist auch bis zu sechs nächsten Nachbarn der Fall. Werden sieben nächste Nachbarn berechnet, überwiegt die Anzahl der grünen Nachbarn und die Klassifikation wird falsch vorgenommen.

rativen Vorgehen wurde zu jedem Histogramm aus der Trainingsbasis mittels euklidischer Distanz der nächste Nachbar gesucht und ergab somit eine Laufzeit von 8:29m. Dies ergibt eine durchschnittliche Klassifikationszeit von 0,09 Sekunden pro Bild. Der Einsatz des kd-Trees für die Berechnung des nächsten Nachbarn senkt die Laufzeit auf ca. 3:30m, was etwa zwei Drittel der Zeit einspart und somit pro Bild 0,037 Sekunden benötigt. Dies lässt sich darauf zurückführen, dass der kd-Tree nicht wie beim iterativen-Vorgehen für jedes Bild die Distanzen zu jedem Bild berechnen muss. Der kd-Tree wird für die gesamte Klassifikation einmal erstellt und kann dann für jedes Bild in einer durchschnittlichen Laufzeit von $O(\log n)$ durchlaufen werden. Die Support Vector Machine kann die Laufzeit noch weiter auf ca. 2:30m senken und ergibt somit eine durchschnittliche Klassifikationszeit von 0,026 Sekunden. Wie in [18] beschrieben, braucht der Mensch für die selbe Aufgabe je nach Sicht auf das Objekt zwischen 0,7 und 0,825 Sekunden. Somit würde das iterative-Vorgehen bereits für den Echtzeitanspruch genügen, wobei beachtet werden muss, dass in der Caltech101 Datenbank meist kleine Bilder (Längste Seite meist nur 300px groß) vorhanden sind. Werden größere Bilder klassifiziert, steigt die Anzahl der berechneten Deskriptoren und somit auch die benötigte Zeit, wobei dann mit den weiteren Vorgehen, wie der SVM die Zeit weiter gesenkt werden kann.

5.3 Auswertung auf eigenen Bildern

Nachdem der Klassifikationsansatz verifiziert wurde, wurde anschließend die eigene Bilddatenbank und die Klassifikationsrate in der Anwendung (vgl. Abbildung 5.3) ausgewertet. Dafür wurde jeweils ein eigener Datensatz an Bildern erstellt und dieser innerhalb der Anwendung mit 18 Gegenständen, jeweils mindestens ein Gegenstand pro Kategorie, und auf

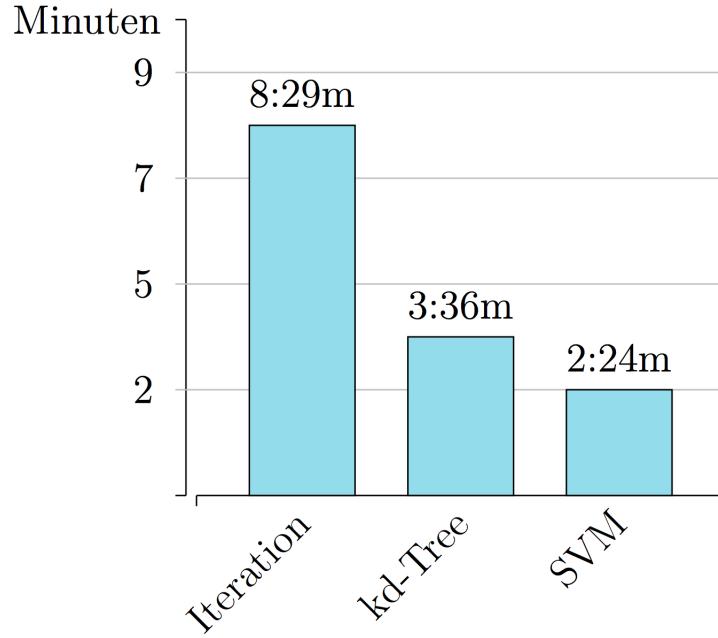


Abbildung 5.3: Dieses Diagramm visualisiert die benötigte Zeit zur Klassifikation der 5647 Objekte aus der Caltech101 Datenbank per Iteration (nächster Nachbar), kd-Tree (nächster Nachbar) und einer SVM.

sich selbst getestet. Für den Test des Datensatzes wurden daher immer bis zu fünf Bilder aus einer Kategorie entnommen und zum Test herangezogen. Die Anwendung wurde mit einem Modell getestet, welches sowohl die Trainings-, als auch die Testdaten enthielt. Zum Test der Anwendung wurden 18 Gegenstände genommen und in einem möglichst neutralen Umfeld, der FINCA, vor die Kinect-Kamera gelegt.

Eine eigene Trainingsbasis wurde zusammengestellt, wie in Unterabschnitt 4.2.1 vorgestellt. Dazu wurden anfangs pro Kategorie 30 Bilder zufällig aus der Caltech101 Datenbank ausgewählt und diese mit zehn eigenen Bildern ergänzt oder falls keine Bilder in der Caltech101 Datenbank vorhanden waren, 40 eigene Bilder erstellt. Wieder wurden die in Abschnitt 5.2 vorgestellten Parameter genutzt. Dabei ergab sich eine Klassifikationsrate von 69,3%, womit die Ergebnisse auf der ganzen Caltech101 Datenbank übertroffen werden konnten. Dies dürfte in erster Linie an den deutlich weniger Klassen liegen. Die Klassifikationsrate konnte jedoch nicht in der realen Anwendung nachgestellt werden. Hier lag die Klassifikationsrate nahe 0% und dürfte unter anderem darauf zurückzuführen sein, dass die Caltech101 Datenbank viele Bilder enthält die gar keine echten Aufnahmen der Objekte darstellen, sondern teilweise auch Skizzen oder WordArt-Grafiken. Hinzu kommt, dass die von der Caltech101 Datenbank aufgenommenen Bilder oftmals aus einer Zeit stammen, in denen Objekte völlig anders aussahen. Während ein Handy damals noch relativ groß war, viele Tasten und teilweise sogar Antennen enthielt, besteht ein Handy heutzutage oftmals

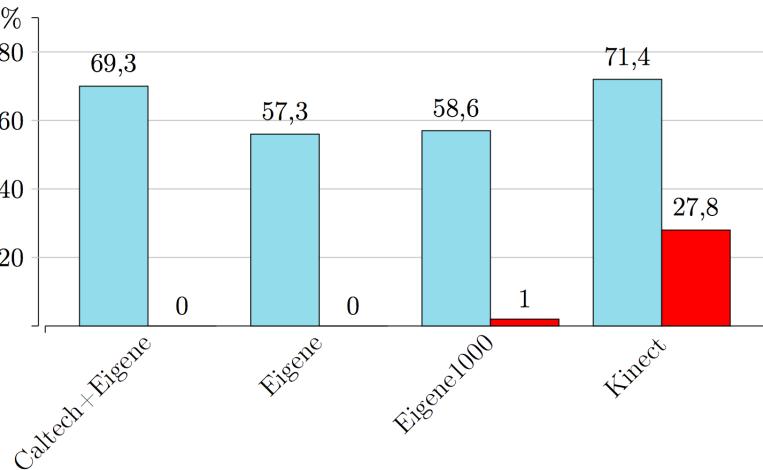


Abbildung 5.4: In türkis sind die Klassifikationsraten der eigenen Bilddatenbanken im Test zu sehen. Rot sind hingegen die Klassifikationsraten im Einsatz in der Anwendung. Unterschieden wurde dabei zwischen einer Mischung aus Caltech101 und eigenen Bildern, nur selbst aufgenommenen Bildern mit 200 bzw. 1000 Visual Words und nur mit der Kinect aufgenommenen Bildern mit 1000 Visual Words. Wie zu sehen ist, arbeiten die ersten drei Datensätze im Test zwar gut, aber fallen im echten Einsatz dann deutlich ab bzw. führen zu unbrauchbaren Klassifikationsergebnissen. Die Gründe dafür werden in Abschnitt 5.3 erläutert. Erst der Einsatz von Bildern, die mit der Kinect aufgenommen wurden, konnte auch den Einsatz in der Anwendung verbessern.

nur aus einem großen Bildschirm.

Um nun in der Anwendung die Ergebnisse zu verbessern, wurde ein Datensatz erstellt, der nur aus eigenen Aufnahmen besteht. Diese wurden in mehreren Haushalten auf unterschiedlichen Hintergründen und mit unterschiedlichen Kameras aufgenommen. Es wurde versucht eine einheitliche Belichtung zu erzielen, was durch die unterschiedlichen Räumlichkeiten jedoch nur schwer zu erreichen war. Es ergaben sich 318 Bilder für das Training mit jeweils zwischen 8 und 49 Bildern pro Kategorie. Die verwendeten Parameter wurden aus dem vorigen Test übernommen. Somit ergab sich eine Klassifikationsrate von 57,3% im Test, was etwas unter den Raten mit Caltech101 Bildern liegt, aber weiterhin über denen aus Abschnitt 5.2. In der Anwendung gab es gegenüber dem Datensatz mit Caltech101 Bilder keine Verbesserung. Grund dafür dürften die unterschiedlichen Kamerasensoren sein. Denn Teile der eigenen Bilder wurden mit unterschiedlichen Kameras (unter anderem Smartphone-Kameras) aufgenommen deren Sensoren sich stark unterscheiden und die Kinect-Kamera besitzt wiederum einen völlig anderen Sensor. Somit dürfte auf den eigenen Bildern Rauschen auftauchen, dass möglicherweise statt den Objekten antrainiert wurde und nicht auf den Kinect-Bildern auftaucht. Um außerdem der höheren Variabilität der Gegenstände als in der Caltech101 Datenbank entgegenzuwirken, wurde die Anzahl der Visual Words von 200 auf 1000 erhöht, was die Klassifikationsrate auf 58,7% steigen ließ. In der Anwendung gab es aber keinerlei Veränderung.

Klasse	Ball	Buch	Flaschen	Gesicht	Handy
Anzahl Bilder	9	36	27	5	16
Klasse	Kamera	Laptop	PC-Maus	Regenschirm	Schere
Anzahl Bilder	19	9	8	22	12
Klasse	Stuhl	Tacker	Tasse	Tastatur	Uhr
Anzahl Bilder	12	18	23	12	13

Tabelle 5.1: Eine Auflistung der finalen Trainingsbasis mit der Anzahl der jeweils für das Training genutzten Bilder.

Daher wurden anschließend alle eigenen Bilder mit der Kinect-Kamera neu aufgenommen, um Probleme mit Kamerarauschen und unterschiedlichen Sensoren zu verringern. Somit ergab sich eine Trainingsbasis mit 241 Bildern und jeweils ca. zwischen 5 und 25 Bildern pro Kategorie (genaue Auflistung s. Tabelle 5.1) für die Anwendung. Dadurch ließ sich die Klassifikationsrate im Test auf ca. 71,4% bei 1000 Visual Words steigern. In der Anwendung ergab sich dann eine Klassifikationsrate von 27,8%, was eine signifikante Steigerung gegenüber den anderen Trainingsdaten darstellt, aber gegenüber dem Test weiterhin schlechter arbeitet. Dabei wurden fünf Gegenstände richtig, elf falsch klassifiziert und bei zwei kam es zu extremen Schwankungen in der Klassifikation, die ständig hin- und herschwankte. Dies lag in erster Linie am Sensorrauschen, wodurch der Ausschnitt einer ruhigen Szene zwei unterschiedliche Ergebnisse lieferte, die richtig und falsch waren. Bei den Fehlklassifikationen war zudem zu beobachten, dass von den elf Gegenständen drei nicht detektiert wurden. Grund hierfür war meist die Größe der Gegenstände. Entweder wurden diese als Element des Tisches erkannt, was bei Scheren oder neuen Smartphones der Fall ist, oder aber die Veränderung in der Szene war zu geringfügig, um von der Anwendung nicht als Sensorrauschen erkannt zu werden. Dies war unter anderem bei Tackern und auch kleineren Tassen zu beobachten. Besonders gute Ergebnisse ließen sich jedoch bei Tastaturen und einigen Büchern feststellen. Vor allem Tastaturen wurden aus den meisten Blickwinkeln richtig erkannt, was sicherlich dem sehr einheitlichen Aussehen der Tastaturen zu verdanken ist. Trotzdem konnten die Ergebnisse aus dem Test in der Anwendung auch hier nicht erreicht werden.

Kapitel 6

Fazit

Die Erkennung von Objekten ist ein schwieriges Gebiet des Visual Computing. Man benötigt dazu eine große Anzahl an Bildern, um das jeweilige Objekt anzutrainieren und jedes Bild muss eindeutig repräsentiert werden können, um diese voneinander zu unterscheiden. Ein Verfahren dafür ist der Bag of Features Ansatz, welcher lokale Merkmale auf einem Bild berechnet und gruppiert, um ein Wörterbuch an Visual Words zu erstellen. Dadurch kann ein Bild durch ein Histogramm der Anzahl der auftauchenden Visual Words beschrieben werden. Dieser Ansatz wurde auch für Klassifikation in der entwickelten Anwendung genutzt.

Dabei konnten die Ergebnisse aus [7] bestätigt werden, indem die Klassifikation mittels nächstem Nachbarn eine Klassifikationsrate von 33,9% auf der Caltech101 Datenbank er gab. Mittels einer SVM und den zusätzlichen Informationen durch die xy-Koordinaten der Deskriptoren konnte die Erfolgsrate ebenfalls auf über 50% gesteigert werden.

Durch die Beschränkung auf 15 Kategorien in der realen Anwendung wurde erwartet, dass trotz zahlreicher Variabilität in der Szene mindestens die Ergebnisse aus dem Test bestätigt werden können. Leider erwies sich der Einsatz der Caltech101 Bilder als unvorteilhaft, da nahezu keine Objekte richtig klassifiziert werden konnten. Selbst der Einsatz von selbst aufgenommenen Bildern und der Erhöhung der Visual Words im Training von 200 auf 1000 brachten keine Verbesserung in der Anwendung.

Erst durch die Aufnahme aller Bilder mithilfe der Kinect Kamera konnten die Ergebnisse auf 27,8% gesteigert werden. Dabei blieb das Ergebnis weiterhin unter den Erwartungen. Dies hat ebenfalls mehrere Gründe. Allen voran sind die Ungenauigkeiten der Kinect und die Art der Aufnahme der Tiefenbilder zu nennen. Durch die Generierung des Tiefenbildes über ein Infrarot-Muster werden gewisse Oberflächen, wie das Display eines Notebooks kaum erkannt, was zu kaum sichtbaren Veränderungen im Differenzbild führte. Aber auch die Ungenauigkeiten der Kamera sind ein großes Problem, da Objekte, wie Tacker oder Scheren oftmals viel zu niedrig sind und je nach Aufstellung der Kinect-Kamera nicht als eigenes Objekt, sondern als Teil des Tisches oder der Oberfläche, auf der sie liegen, gekenn-

zeichnet werden. Dies war teilweise selbst bei höheren Tassen noch der Fall und machte mit dem vorgestellten Ansatz über ein Differenzbild die Segmentierung vorzunehmen, diese nahezu unmöglich. Als ebenfalls problematisch erwies sich die geringe Auflösung des Tiefenbildes, wodurch Objekte ab einem gewissen Abstand im Bild so klein ausfielen, dass nicht genügend Deskriptoren für eine erfolgreiche Klassifikation ausgegeben wurden und die Ergebnisse oftmals stark variierten.

Besonders gut funktioniert jedoch der Echtzeitaspekt der Anwendung. Hier ist die Anwendung schneller als die menschliche Wahrnehmung. Während ein Mensch für die Erkennung eines Objektes durchschnittlich 0,75 Sekunden benötigt, braucht die Anwendung mittels SVM ca. 0,026 Sekunden und ist selbst im Worst-Case eines sehr großen Bildes schneller als der Mensch.

Trotzdem lässt sich feststellen, dass dieser Ansatz durchaus Potential hat, wenn mehr Bilder mit einer Kinect-Kamera aufgenommen werden würden, um weitere Variabilitäten abzufangen. Trotzdem bliebe das Problem, das kleine Objekte, wie Tacker, Scheren oder Uhren für eine Anwendung, wie diese kaum zu erkennen. Erst größere Objekte, wie Flaschen oder auch Tastaturen lassen sich mit besser klassifizieren. Daher hat sich gezeigt, dass der Einsatz der Kinect-Kamera viele Probleme mit sich bringt und die Segmentierung über diese sich aus technischen Gründen nur schwer realisieren lässt und teilweise unlösbare Probleme darstellt. Sie verfügt selbst bei ruhigen Szenen über viel Rauschen und kleine Objekte verschwinden gänzlich im Hintergrund. Um die Ergebnisse weiter zu verbessern, wäre es daher vorteilhaft die Segmentierung nicht über das Tiefenbild zu bestimmen, sondern möglicherweise über ein Farbbild oder auch mittels der Suche nach bestimmten Merkmalsverteilungen in einem Bild mit einem Teilbasierten Modell.

Insgesamt lässt sich also feststellen, dass der Bag of Features Ansatz eine extrem einfache und schnelle Klassifikation von Objekten ermöglicht, die sogar schneller eine Entscheidung trifft, als ein Mensch. Falls auf einem Bild das Objekt den zentralen Bildinhalt darstellt, funktioniert der Bag of Features Ansatz auch sehr zuverlässig. Muss jedoch eine Segmentierung des Bildinhaltes vorgenommen werden, kommt es zu Problemen, die im Falle der vorgestellten Anwendung den technischen Beschränkungen der Kinect zuzuschreiben sind. Eine andere Bestimmung des Tiefenbildes oder ein anderer Ansatz der Segmentierung würden für Verbesserungen sorgen.

Abbildungsverzeichnis

1.1	Radwegsschilder	4
1.2	Bsp. Caltech101	5
2.1	Zylinder	8
2.2	BoF-Verfahren	9
2.3	In der Mitte ist das Differenzbild zu sehen mit dem als X gekennzeichneten Punkt, wessen Nachbarschaft (grüne Punkte) untersucht wird. Entnommen aus [14]	11
2.4	LOWE	12
2.5	SVM-Verteilung	15
2.6	xy-Bild	16
2.7	Bild des Kinect Infrarot-Musters	18
2.8	Opening/Closing	19
2.9	kd-Tree	20
3.1	Spatial-Pyramid	27
4.1	Interface	32
4.2	Pipeline	33
4.3	Zustandsdiagramm	34
4.4	Klassifikationspipeline	35
5.1	Klassifikationsraten	38
5.2	k-nächste Nachbarn	39
5.3	Zeitverhalten	40
5.4	Klassifikation Kinect	41

Literaturverzeichnis

- [1] ANDREA VEDALDI, BRIAN FULKERSON: *Vlfeat: an open and portable library of computer vision algorithms*. MM'10 Proceedings of the international conference on Multimedea, 2010.
- [2] CHIH-CHUNG CHANG, CHIH-JEN LIN: *LibSVM: A library for support vector machines*. ACM Transactions on Intelligent Systems and Technology, 2011.
- [3] CORINNA CORTES, VLADIMIR VAPNIK: *Support-Vector Networks*. Machine Learning 20, 1995.
- [4] FINK, PROF. DR.-ING. GERNOT A.: *Computer Vision; Vorlesungsskript*, 2013.
- [5] GARY BRADSKI, ADRIAN KAEHLER: *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, 2008.
- [6] GOOL; CHRIS WILLIAMS, JOHN WINN, ANDREW ZISSERMANN MARK EVERINGHAM; LUC VAN: *Pascal Visual Object Classes Homepage*. <http://pascallin.ecs.soton.ac.uk/challenges/VOC/>, 2013. [Online; accessed 17-August-2013].
- [7] GRZESZICK, RENE, LEONARD ROTHACKER und GERNOT A. FINK: *Bag-of-Features Representations using Spatial Visual Vocabularies for Object Classification*. In: *IEEE Intl. Conf. on Image Processing*, Melbourne, Australia, 2013.
- [8] HERMANN, KOPETZ: *Real-time systems: Design principles for distributed embedded applications*. Kluwer Academic Publishers, 1997.
- [9] iSUPPLI: *The Teardown*. Engineering and Technology Magazine 6, 2011.
- [10] JOSEF SIVIC, ANDREW ZIMMERMAN: *Video Google: A Text Retrieval Approach to Object Matching in Videos*. IEEE International Conference on Computer Vision, 2003.
- [11] KOUROSH KHOSHELHAM, SANDER OUDE ELBERINK: *Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications*. Sensors 2012, 2012.
- [12] LI FEI-FEI, ROB FERGUS, PIETRO PERONA: *Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories*. CVPR 2004, Workshop on Generative-Model Based Vision, 2004.

- [13] LLYOD, STUART P.: *Least Squares Quantization in PCM*. IEEE Transactions on Information Theory, vol. 28, no. 2, pp. 129-137, 1982.
- [14] LOWE, DAVID G.: *Distinctive Image Features from Scale-Invariant Keypoints*. International Journal of Computer Vision, 2004.
- [15] MACQUEEN, J.: *Some Methods for Classification and Analysis of Multivariate Observations*. Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, 1967.
- [16] MÜLLER, HEINRICH: *Mensch-Maschine-Interaktion; Vorlesungsskript*.
- [17] Open Kinect Documentation. http://openkinect.org/wiki/Main_Page, 2013. [Online; accessed 17-August-2013].
- [18] PALMER, STEPHEN E: *Vision science: Photons to phenomenology*, Band 1. MIT press Cambridge, MA, 1999.
- [19] RICHARD O. DUDA, PETER E. HART: *Pattern Classification and Scene Analysis*. John Wiley Sons Inc, 2001.
- [20] SCHOLZ, PETER: *Softwareentwicklung Eingebetteter Systeme*. Springer-Verlag Berlin Heidelberg, 2005.
- [21] STEPHEN O'HARA, BRUCE A. DRAPER: *Introduction to the Bag of Features Paradigm for Image Classification and Retrieval*. Arxiv preprint arXiv:1101.3354, 2011.
- [22] SVETLANA LAZEBNIK, CORDELIA SCHMID, JEAN PONCE: *Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories*. IEEE Conference on Computer Vision and Pattern Recognition, 2006.
- [23] XIAOYI JIANG, HORST BUNKE: *Dreidimensionales Computersehen: Gewinnung und Analyse von Tiefenbildern*. Springer-Verlag, 1997.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den 13. September 2013

Martin Schwitalla

