Karlsruhe Institute of Technology

Fraunhofer IOSB

Bildbasierte Objekterkennung mittels mobiler Endgeräte für das Mobile Lernen

Diplomarbeit
von

Stefan Bürger

Institut für Anthropomatik
Fraunhofer IOSB

Gutachter: Prof. Dr.-Ing. J. Beyerer
Betreuer: Dipl.-Inf. Alexander Streicher, Fraunhofer IOSB

Bearbeitungszeit: 09.01.2012 – 08.07.2012

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 3. Juli 2012

Zusammenfassung

Mobile Endgeräte eignen sich als orts- und zeitunabhängige Informationsträger und Wissensvermittler. Durch die Kamera mobiler Endgeräte kann der Kontext erfasst werden, in dem sich der Benutzer befindet, um kontextabhängige Assistenzinformationen zu Lernzwecken anbieten zu können.
Für das Roboterverbundsystem von *AMFIS* (Aufklärung mit mobilen und ortsfesten Sensoren im Verbund) des Fraunhofer-Instituts für Optronik, Systemtechnik und Bildauswertung IOSB wurde eine bildbasierte Objekterkennung realisiert. Diese wurde für mobile Endgeräte mit Android Betriebssystem auf Basis von OpenCV entworfen.
Im Rahmen dieser Diplomarbeit wurde das *Bag of Words* Verfahren im Hinblick auf seine Eignung für die Objekterkennung auf mobilen Endgeräten evaluiert. Dabei wird eine Bildrepräsentation in Form eines Histogramms erzeugt. Dieses enthält die Anzahl gefundener visueller Wörter im Bild, welche durch lokale Features repräsentiert werden. Für die Parametrisierung des *Bag of Words* Verfahrens wurden SIFT und SURF Features, sowie fuzzy und räumliche Histogramme evaluiert. Für die Klassifikation wurde der k-Nearest-Neighbor Klassifikator und die Support Vector Machine betrachtet. Die Evaluation findet bezüglich der Klassifikationsergebnisse und der Laufzeit auf mobilen Endgeräten statt. Auf Basis dieser Ergebnisse wurde eine Parametrisierung gewählt, die in der Lage ist in sehr schneller Zeit (350 ms auf einem ASUS Transformer Prime TF201) eingelernte Objekte auf dem Testdatensatz fast perfekt zu erkennen (Mean Average Precision 98,7). Die Implementierung kann durch das Einlernen neuer Objekte auf andere Anwendungsgebiete übertragen werden.

Abstract

Mobile devices are suitable as location and time independent information carriers and knowledge brokers. A user can detect his context by using the camera of a mobile device. This information can be used in order to provide context-sensitive assistance information for learning purposes.
For the components of the *AMFIS* system (reconnaissance with mobile and stationary sensors in the network), which is being developed at the Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB, an image-based object recognition was realized. It was implemented for mobile Android devices and is based on OpenCV.
This work analyzes the *bag of words* approach regarding its suitability for object recognition on mobile devices. The *bag of words* approach uses locale features and a trained visual vocabulary to generate an image representation in form of a histogram. Each bin of the histogram counts the appearance of one of the visual words. The evaluation compares SIFT and SURF features, as well as fuzzy and spatial histograms. For the classification the k-nearest-neighbor algorithm and support vector machines were analyzed. The evaluation considers classification results and computing time on mobile devices. The final parameterization can recognize learned objects almost perfecty on the test data (Mean Average Precision of 98.7) in a very short time (350 milliseconds on an ASUS Transformer Prime TF201). The implementation can be used for other application areas by training new objects.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Motivation . . . 2
- 1.2 Ziel der Arbeit . . . 3
- 1.3 Stand der Forschung und Technik . . . 3
- 1.4 Aufbau der Arbeit . . . 6

2 Grundlagen der Kontextdetektion für das Mobile Lernen **7**
- 2.1 Bildbasierte Objekterkennung für Assistenzsysteme . . . 7
- 2.2 Begriffserklärung Kontext und Kontextsensitivität . . . 8
- 2.3 Begriffserklärung Mobiles Lernen . . . 8
- 2.4 Herausforderungen . . . 9
- 2.5 Einsatzgebiete für die Kontextdetektion mit Smartphones und Tablets . . . 10

3 Bildverarbeitung auf Smartphones und Tablets mit Android Betriebssystem **15**
- 3.1 Hardware . . . 15
- 3.2 Android Betriebssystem . . . 17
- 3.3 Android Software Developement Kit (SDK) . . . 18
- 3.4 Android Native Developement Kit (NDK) . . . 19
- 3.5 Open Source Computer Vision Library (OpenCV) . . . 19

4 Methoden der Bildverarbeitung zur Objektidentifikation **21**
- 4.1 Lokale Features . . . 22
  - 4.1.1 Scale-invariant feature transform . . . 22
  - 4.1.2 Speeded Up Robust Features . . . 25
- 4.2 K-Means Clustering für das visuelle Vokabular . . . 29
- 4.3 FLANN . . . 30
- 4.4 Bildung des BoW-Histogramms . . . 30
- 4.5 Klassifikatoren . . . 32
  - 4.5.1 k-Nearest-Neighbor . . . 33
  - 4.5.2 Support Vector Machine . . . 33

5 Implementierung **37**
- 5.1 Systemaufbau . . . 37
- 5.2 Bildverarbeitung . . . 38
- 5.3 User Interface der Android App . . . 40
- 5.4 Benchmark App . . . 44

6 Evaluation durch Experimente **47**
- 6.1 Güte der Trainings- und Testdaten . . . 47
- 6.2 Evaluationsmethoden . . . 48
- 6.3 Evaluationsergebnisse . . . 51
- 6.4 Bewertung der Ergebnisse . . . 61

7 Zusammenfassung und Ausblick **63**

Abbildungsverzeichnis **67**

Literaturverzeichnis **69**

1. Einleitung

Bei der Bewältigung von Katastrophen, wie dem Reaktorunfall in Fukushima im Jahr 2011, werden moderne Technologien benötigt, damit Rettungskräfte ihren Einsatz planen können. Im Krisenfall muss schnell reagiert werden, obwohl lediglich unzureichende Informationen über die tatsächliche Situation am Einsatzort vorhanden sind. Um das Leben von beteiligten Rettungskräften nicht zu gefährden, können Roboter eingesetzt werden, die mit einer Vielzahl an Sensoren zur Informationsgewinnung beitragen (Abbildung 1.1). Für diesen Zweck wurde am Fraunhofer-Institut für Optronik, Systemtechnik und Bildauswertung IOSB ein System zur Aufklärung mit mobilen und ortsfesten Sensoren im Verbund (*AMFIS*) entwickelt. Dieses besteht unter anderem aus Robotereinheiten, die auf dem Boden, in der Luft oder im Wasser operieren. Die gesammelten Informationen können unter anderem auf einem digitalen Lagetisch visualisiert werden. Die komplexen Einzelkomponenten dieses Verbundes unterscheiden sich untereinander sehr stark, sodass für Inbetriebnahme, Steuerung und Wartung der Einzelkomponenten individuelles Know-How benötigt wird. Im Katastrophenfall sind jedoch nicht alle beteiligten Einsatzkräfte Spezialisten, die sich mit den Komponenten des *AMFIS*-Systems auskennen. Von ehrenamtlichen Helfern des Roten Kreuzes kann nicht verlangt werden, dass diese sich in die komplexen Details der Roboter einarbeiten. Handbücher und Dokumentationen in Papierform bieten zwar die benötigten Informationen, eignen sich jedoch nicht für den realen Einsatz, da sie aufgrund ihres Volumens und ihres Gewichts die Einsatzkräfte behindern.

Abbildung 1.1: Im Katastrophenfall werden Quadrocopter zur Informationsgewinnung eingesetzt. [1]

Eine Alternative zu gedruckten Handbüchern und Dokumentationen stellen digitale Assistenzsysteme dar. Mit Hilfe von mobilen Endgeräten, wie zum Beispiel Smartphones oder Tablets, können Informationen digital gespeichert und dem Anwender über ein Display angezeigt werden. Für das Roboterverbundsystem von *AMFIS* wurde zu diesem Zweck am Fraunhofer IOSB die mobile Anwendung *Mobile Tutor* [1] entwickelt. Diese verwandelt Android-basierte Smartphones und Tablets in Informationsträger und Wissensvermittler, indem Handbücher und Dokumentationen dem Anwender zur Verfügung gestellt werden. Durch die digitale Form ist die Menge an Informationen lediglich durch den Speicher des Endgeräts beschränkt und Informationen können stets auf dem aktuellen Stand gehalten werden. Aufgrund der geringen Größe und des geringen Gewichts dieser Geräte können sie im Einsatz mitgeführt werden, ohne dass die Einsatzkräfte unnötig belastet werden. Die Suche nach relevanten Informationen kann jedoch sehr umständlich und aufwendig sein. Besitzt der Anwender zum Beispiel kein Vorwissen über die genauen Bezeichnungen der Einzelkomponenten des *AMFIS*-Systems, wird die Suche nach relevanten Informationen in der Anwendung *Mobile Tutor* zeitaufwendig oder sogar unmöglich.

1.1 Motivation

Aktuelle Smartphones sind Multifunktionsgeräte, deren Ausstattung Anwendungsgebiete ermöglichen, die weit über das Telefonieren hinausreichen. In modernen Smartphones kommen Multi-Core-Prozessoren zum Einsatz [2]. Für das User-Interface stehen große, hochauflösende Displays zur Verfügung. Neben dem berührungsempfindlichen Display kann die mannigfaltige sensorische Ausstattung, bestehend aus Beschleunigungssensoren, Gyro-Sensoren, GPS, Licht-Sensoren und Kameras, zur Informationsgewinnung eingesetzt werden [3]. Für die Vernetzung kommen Bluetooth, WLAN, UMTS oder LTE zum Einsatz. Neben Smartphones hat sich eine weitere Geräteklasse am Markt etabliert. Die Einführung des Apple iPads verhalf den Tablet-Computern zum Durchbruch. Tablets bringen viele Eigenschaften von Smartphones mit, können jedoch mit einem größeren Display aufwarten.

Assistenzsysteme, wie die *Mobile Tutor* Applikation (App) des Fraunhofer IOSB verwenden Smartphones und Tablets als ortsunabhängige Informationsträger und Wissensvermittler für das Mobile Lernen. Dabei kann der Nutzer Komponenten des *AMFIS*-Systems über Toucheingaben auf dem Display auswählen, sodass Assistenzinformationen zu dieser Komponente angezeigt werden. Die umfangreiche Sensorik der mobilen Endgeräte ermöglicht weiterführende Konzepte der Benutzerinteraktion, die dem Anwender den Informationszugang bei Assistenzsystemen vereinfachen. Bei der Betrachtung der sensorischen Ausstattung mobiler Endgeräte bietet die Kamera im Vergleich zu anderen Sensoren das größte Potential. Die Kamerabilder können mit Hilfe von Bildverarbeitungsalgorithmen für diverse Aufgaben ausgewertet werden.

Das User Interface der *Mobile Tutor* App kann durch die bildbasierte Erkennung von Systemkomponenten ergänzt werden. Hierdurch wird der Zugang zu relevanten Informationen vereinfacht und beschleunigt. Anstatt die gewünschten Informationen manuell in der App zu suchen, muss der Anwender lediglich die Komponente abfotografieren, für die er sich gerade interessiert. Das mobile Endgerät übernimmt die Identifikation der Systemkomponente, die auf dem Bild zu sehen ist. Nach Auswertung des Bildes können vor Ort kontextbezogene Assistenzfunktionen, wie zum Beispiel Anleitungen zur Inbetriebnahme, Steuerung und Wartung der Komponente, angezeigt werden. Diese Form der kontextsensitiven (*engl.* context-aware) Assistenz ermöglicht Einsatzkräften den schnellen und unkomplizierten Zugriff auf relevante Informationen während des Einsatzes, auch wenn sie keine Vorkenntnisse über die Komponenten des *AMFIS*-Systems besitzen.

1.2 Ziel der Arbeit

Die vorliegende Diplomarbeit hat das Ziel eine Anwendung für ein mobiles, Android-basiertes Endgerät zu entwickeln, welches eine bildbasierte Kontextdetektion für ein mobiles Assistenzsystem vornimmt. Hierzu soll die Anwendung mit Hilfe der Kamerabilder des Endgerätes und Verfahren der Bildverarbeitung Komponenten des *AMFIS*-Systems erkennen. Neben kompletten Komponenten sollen auch Teilbereiche einer Komponente identifiziert werden können, sodass kontextbezogene Assistenzinformationen zu diesen Geräten bereitgestellt werden können.

1.3 Stand der Forschung und Technik

Je nach Einsatzgebiet wird der Begriff Objekterkennung für unterschiedliche Aufgaben verwendet. Bei der *Klassifikation* wird die Frage beantwortet, ob ein Bild eine Instanz einer bestimmten Objektklasse enthält [4]. Die *Detektion* hingegen gibt an, wo sich im Bild eine Instanz einer bestimmten Objektklasse befindet [4]. Im Kontext der *Robotik* ist die Identifikation einer Instanz einer Objektklasse eng verbunden mit der 6D-Lokalisierung des Objekts um dieses greifen zu können [5]. In dieser Arbeit wird der Begriff Objekterkennung für die Aufgabe der Klassifikation definiert. Die Lage der Instanz der Objektklasse im Bild ist für die Anzeige von Assistenzinformationen nicht relevant.

Im Bereich der bildbasierten Objekterkennung stehen eine Vielzahl unterschiedlicher Algorithmen zur Verfügung. Diese lassen sich in modellbasierte und ansichtsbasierte Verfahren gliedern. Bei den modellbasierten Ansätzen wird eine Modelldatenbank aller Objekte verwendet. Zur Bestimmung einer Objektklasse wird die Ähnlichkeit zwischen Modell- und Bilddaten analysiert. Bei der Verwendung eines CAD-Drahtgitters als Modellrepräsentation werden zum Beispiel die 3D Linien auf die Bildebene projiziert und anhand von Kanteninformationen im Bild evaluiert. Modellbasierte Verfahren haben den Vorteil, dass sie gleichzeitig die Lage des Objekts bestimmen, sie funktionieren jedoch nicht, wenn das Objekt nicht aus klar identifizierbaren geometrischen Primitiven besteht oder das Objekt verformbar ist [5].

In dieser Arbeit wird ein ansichtsbasiertes Verfahren verwendet. Diese Algorithmen verzichten auf die Verwendung von Objektmodellen. Anstelle dessen kommt eine Menge an Trainingsbildern zum Einsatz, welche die Eigenschaften der Objekte beschreiben. Globale Verfahren bilden dabei einen Merkmalsvektor für das gesamte Bild. Um Objekte zu erkennen, wird der Merkmalsvektor eines Eingabebildes mit den Vektoren der Trainingsbilder verglichen. Farbmerkmale können zum Beispiel mit Histogrammen [6] oder mit Momenten [7] dargestellt werden. Für die Erfassung von Texturmerkmalen werden die Co-occurence-Matrix [8] oder die Gabor Wavelet Transformation [9] eingesetzt. Histogramme lokaler rezeptiver Felder [10] und die Histograms of Oriented Gradients (HOG) [11] bilden weitere Ansätze um Objekte anhand ihrer Form zu erkennen.

Ein besonders schneller Ansatz zur ansichtsbasierten Objekterkennung wurde von Viola und Jones [12] vorgestellt. Dabei wird mit Hilfe des Lernalgorithmus AdaBoost eine kleine Anzahl an Haar Features ausgewählt, welche für die Erkennung eines Objektes relevant sind. Diese können mit dem Integral Image sehr effizient berechnet werden. Der Algorithmus wird beschleunigt, indem eine Kaskade gebildet wird, welche aus mehreren in ihrer Komplexität steigenden Klassifikatoren besteht.

Für die Erkennung von dreidimensionalen Objekten mit unterschiedlichen Ansichtswinkeln in zweidimensionalen Bildern verwenden Sheng und Qi-cong [13] Texturinformationen, Farbmomente, Hu's Momente und affine Moment-Invarianten sowie ein Backpropagation Neuronales Netzwerk für die Klassifikation. He *et al.* [14] nutzen individuelle Klassifikatoren um einzelne Teile eines Objekts zu detektieren. Ein Objekt als Ganzes wird erkannt, wenn die erkannten Einzelteile strukturell zusammenpassen.

Lokale Features

Eine Alternative zu den globalen ansichtsbasierten Methoden stellen lokale Features dar, welche in den letzten Jahren immer mehr an Bedeutung gewonnen haben. Lokale Features bestehen aus Keypoints, welche auch Interestpoints genannt werden, und Deskriptoren. Keypoints beschreiben interessante Bildpositionen, während Deskriptoren Repräsentationen des lokalen Bildausschnitts um Keypoints herum darstellen. Verfahren, die lediglich den Keypoint bestimmen, sind der Harris Corner Detector [15], die Good Features to Track [16], welche auch als Shi-Tomasi Features bezeichnet werden, und die Maximally Stable Extremal Regions (MSER) [17]. Die bekanntesten Vertreter der lokalen Features sind SIFT Features (Scale-invariant feature transform) von Lowe [18] [19], welche Berechnungsvorschriften für Keypoints und Deskriptoren beinhaltet. PCA SIFT Features [20], bei denen die Dimension der Merkmalsvektoren durch die Principal Component Analysis (PCA) reduziert wird, und GLOH-Features (Gradient location orientation histogram) [21], die anstelle eines Gitternetzrasters ein log-polar Gitter zur Berechnung des Merkmalsvektors verwenden, stellen modifizierte Versionen von SIFT dar. Bei der Berechnung von SURF Features (Speeded-Up Robust Features) [22] werden Rechteckfilter und Integral Images verwendet um die Berechnung von Keypoints und Deskriptoren im Vergleich zu SIFT zu beschleunigen. Für das in dieser Arbeit verwendete Verfahren zur Objekterkennung werden SIFT und SURF Features evaluiert.

Lokale Features können für die Objekterkennung genutzt werden, indem die Deskriptoren eines Bildes mit Hilfe des nächsten Nachbars zu den Deskriptoren gespeicherter Trainingsbilder zugeordnet werden. Die Menge der richtigen und falschen Zuordnungen wird anschließend einer geometrischen Verifikation unterzogen. Hierbei kommen die Verfahren der Houghtransformation, RANSAC (*engl.* random sample consensus) und der Least squares homography estimation zum Einsatz. Dieser Ansatz wird vor allem in der Robotik verwendet, da hiermit eine 6D-Lokalisierung möglich ist [23].

Bag of Words

In dieser Arbeit wird für die Objekterkennung das *Bag of Words* Verfahren verwendet. Dieses stammt ursprünglich aus der Textanalyse, bei der es zur Kategorisierung von Texten eingesetzt wird. Csurka *et al.* [24] haben diesen Ansatz für die Einordnung von Bildern in Objektkategorien übertragen. In der Literatur wird das Verfahren auch als *Bag of visual Words*, *Bag of Keypoints* oder *Bag of Features* bezeichnet. Dabei werden lokale Features genutzt um zunächst Bildausschnitte (*engl.* Patches) zu detektieren und mit Hilfe eines Deskriptors zu beschreiben. Die Deskriptoren der Trainingsbilder werden verwendet um mit Hilfe des k-Means Clustering ein visuelles Vokabular, auch Codebuch genannt, zu lernen. Den Deskriptoren eines Bildes werden die ähnlichsten Codebucheinträge zugeordnet. Mit Hilfe dieser Zuordnungen wird ein Histogramm, der sogenannte *Bag of Keypoints*, berechnet, welches die Anzahl der Zuordnungen zu den einzelnen visuellen Wörtern beschreibt. Das Histogramm wird in der Veröffentlichung von Csurka *et al.* verwendet um den Bildinhalt mit Hilfe einer Support Vector Machine oder dem Naive Bayes Klassifikator zu klassifizieren. In dieser Arbeit wird für die Klassifikation neben der Support Vector Machine auch der k-Nearest-Neighbor Klassifikator evaluiert.

Die Ergebnisse des *Bag of Words* Verfahrens hängen von einer Vielzahl von verwendeten Algorithmen und Parametern für die einzelnen Teilabschnitte ab. In der Literatur finden sich viele Veröffentlichungen, die sich mit der Optimierung des *Bag of Words* Verfahrens beschäftigen. Nowak *et al.* [25] zeigen unter anderem, dass ab einer bestimmten Größe eine Erweiterung des visuellen Vokabulars nicht zu einer Verbesserung der Resultate führt, da eine Sättigung eintritt. Jiang *et al.* [26] betrachten den Einfluss des Keypoint Detektors, der Größe des visuellen Vokabulars und der Wahl des Kernels auf die Ergebnisse. Sie zeigen

weiterhin, dass sich die Ergebnisse verbessern lassen, indem der *Bag of Words* Ansatz mit globalen Features, wie zum Beispiel Farbmomente oder Wavelet Texturen, kombiniert wird. Die Optimierung des *Bag of Words* Verfahrens für die Anwendung auf mobilen Endgeräten ist Teil dieser Arbeit.

Juri und Triggs [27] bemängeln, dass das in dieser Arbeit verwendete k-Means Clustering mangelhafte Codebücher generiert, da sich die Cluster lediglich auf die dichtesten Regionen des Deskriptorenraums beschränken. Sie schlagen anstelle dessen ein Mean-Shift basiertes Clustering mit festem Radius vor. Wu *et al.* [28] kritisieren in ihrer Arbeit den Verlust der Semantik bei der Bildung des visuellen Vokabulars, da beim Clustering lediglich der euklidsche Abstand betrachtet wird. Als Lösung schlagen sie die Verwendung eines *Semantics-Preserving Codebook* (SPC) vor. Dieses wird gebildet, indem der Abstand zwischen semantisch identischen Features minimiert wird.

Bei der Berechnung des Histogramms können die Bins der Codebucheinträge unterschiedlich gewichtet werden. Während beim ursprünglichen Ansatz ein Bildausschnitt ausschließlich das Bin des ähnlichsten Codebucheintrags erhöht, werden bei Bouachir *et al.* [29] und Jiang *et al.* [26] mehrere Bins in Abhängigkeit der Ähnlichkeit der Deskriptoren der Bildausschnitte zu den Codebucheinträgen beeinflusst. Das Softweighting führt zu einer fuzzy Repräsentation, welche einer robusteren Bildsignatur entspricht. In dieser Arbeit werden sowohl normale als auch fuzzy Histogramme evaluiert.

Der ursprüngliche *Bag of Words* Ansatz betrachtet lediglich die Anzahl der gefundenen Codebucheinträge im Bild. Dabei spielt es keine Rolle, wo im Bild oder in welcher geometrischen Anordnung zueinander sich diese befinden. Somit gehen sämtliche räumliche Informationen verloren. Eine Möglichkeit, zumindest einen Teil der räumlichen Informationen zu berücksichtigen, ist die Unterteilung des Bildes in Kacheln, wie von Viitaniemi und Laaksonen [30] beschrieben. Von jeder Kachel wird ein eigenes *Bag of Words* Histogramm berechnet, welche anschließend konkateniert werden. Neben einer festen Zuordnung von Position im Bild zu einzelnen Kacheln wird auch eine weiche Zuordnung vorgestellt, bei der ein Keypoint die Histogramme mehrerer Bildregionen beeinflusst. Bei räumlichen Pyramiden, die von Lazebnik *et al.* [31] vorgestellt und in Yang und Newsam [32] sowie Bosch *et al.* [33] verwendet werden, wird das Bild durch eine Sequenz von immer feineren Gittern geteilt. Auf jeder Pyramidenebene wird für jede Kachel ein eigenes Histogramm erstellt, welche anschließend gewichtet konkateniert werden. Cao *et al.* [34] projezieren für ihre räumliche *Bag of Words* Variante die gefundenen Bildausschnitte in unterschiedliche Richtungen. Dabei kommen sowohl lineare als auch kreisförmige Projektionen zum Einsatz. Diese Arbeit evaluiert räumliche Histogramme in Form einer Aufteilung in vier Bildregionen und einer Pyramidenrepräsentation.

Bildverarbeitung auf mobilen Endgeräten

Für die Architektur von bildverarbeitenden Anwendungen für mobile Endgeräte gibt es zwei unterschiedliche Ansätze. Auf der einen Seite kann eine Client-Server Architektur verwendet werden, bei der der Hauptrechenaufwand vom Server erledigt wird. Bei einer Stand-Alone Architektur werden sämtliche Berechnungen auf dem mobilen Gerät durchgeführt. In [35] wird ein interaktiver Museumsguide als Client-Server System vorgestellt. Der Client dient dabei einzig als Ein- und Ausgabegerät. Das aufgenommene Bild wird an einen Server gesendet, der die Bildverarbeitung vornimmt und das Resultat zurück zum Client sendet. Für die Identifikation von Gebäuden setzen Wang *et al.* in [36] ebenfalls das Client-Server Prinzip ein. Einen hybriden Ansatz wählen Viswanathan *et al.* [37]. Dabei wird zunächst auf dem mobilen Gerät ein Gist Vektor berechnet, der die wichtigsten globalen Eigenschaften des Bildes zusammenfasst. Diese Bildsignatur wird an einen Server gesendet, der mit Hilfe von mehreren Support Vector Machines die Klassifikation

vornimmt. Bay *et al.* [38] zeigen, dass für die bildbasierte Erkennung von Objekten auf einem mobilen Gerät kein Server notwendig ist. Sie verwenden einen Tablet-PC und eine Webcam um Bilder von Museumsobjekten aufzunehmen, die anschließend mit Hilfe von SURF Features ausgewertet werden um das Exponat zu erkennen. Die Implementierung dieser Arbeit basiert ebenfalls auf einer Stand-Alone Architektur.

Echtzeitfähige Objektdetektion ohne Client-Server Prinzip war das Ziel von Jeong und Moon in ihrer Arbeit [39]. Sie verwenden zur Featuredetektion keine rechenaufwendigen SIFT oder SURF Detektoren, sondern an Stelle dessen FAST Features um in Echtzeit ein Logo detektieren zu können. Als Klassifikatoren werden eine Support Vector Machine und ein Backpropagation Neuronales Netz eingesetzt.

Für ihren mobilen Museumsführer *PhoneGuide* setzen Bimber und Bruns auf eine Kombination mehrer Techniken um die Objektdetektion robuster zu gestalten [40]. Die einfache Objektklassifikation mit Hilfe von globalen Farbinformationen wird durch die Verwendung einer Bildsequenz erweitert. Weiterhin wird der Nutzer per Bluetooth lokalisiert und der wahrscheinlichste Weg durch das Museum vorausberechnet. Zusätzlich setzen sie auf die Kommunikation zwischen den mobilen Geräten, damit bei veränderten äußerlichen Erscheinungen der Objekte alle Geräte die aktuellen Klassifikationsinformationen besitzen.

1.4 Aufbau der Arbeit

In Kapitel 2 werden die Grundlagen der Objekterkennung für das Mobile Lernen behandelt. Hierzu gehören neben den Definitionen der Begriffe Kontext, Kontextsensitivität und Mobiles Lernen die Herausforderungen, die bei der bildbasierten Objekterkennung auftreten. Das Kapitel schließt mit einer Vorstellung weiterer Einsatzbereiche für die bildbasierte Objekterkennung mit mobilen Endgeräten.

Kapitel 3 widmet sich den Eigenschaften von Hardware und Software mobiler Endgeräte, die Einfluss auf die Entwicklung mobiler Bildverarbeitungslösungen haben.

Die Bildverarbeitungsalgorithmen, die in dieser Arbeit für die Objekterkennung zum Einsatz kommen, werden in Kapitel 4 vorgestellt. Das *Bag of Words* Verfahren wird mit seinen Einzelschritten für die Berechnungen lokaler Features, das Clustering des visuellen Vokabulars, die Histogrammbildung und die Klassifikation beschrieben.

Die Implementierung wird anhand des Systemaufbaus, der Bildverarbeitung und dem User Interface in Kapitel 5 vorgestellt. Weiterhin wird die Benchmark App vorgestellt, die für die Auswertung der Rechenzeiten der Bildverarbeitungsalgorithmen implementiert wurde.

Kapitel 6 behandelt neben den Evaluationsmethoden die Evaluationsergebnisse, die durch Experimente ermittelt wurden.

Die Arbeit schließt mit Kapitel 7, welches die Arbeit zusammenfasst und einen Ausblick gibt.

2. Grundlagen der Kontextdetektion für das Mobile Lernen

Für die Entwicklung einer mobilen Anwendung zur bildbasierten Kontextdetektion werden zunächst die Grundlagen vorgestellt. Hierzu werden die Begriffe Kontext und Kontextsensitivität erläutert und das Anwendungsgebiet Mobiles Lernen in Form eines Assistenzsystems vorgestellt. Die bildbasierte Erkennung von Objekten auf der Hardware mobiler Endgeräte bringt verschiedene Herausforderungen mit sich, die beachtet werden müssen. Das Kapitel schließt mit der Vorstellung weiterer Einsatzbereiche, in denen die bildbasierte Objekterkennung mit einem mobilen Endgerät eingesetzt werden kann.

2.1 Bildbasierte Objekterkennung für Assistenzsysteme

Assistenzsysteme helfen dem Nutzer bei der Verwendung von Produkten. Sie können vom Anwender gezielt eingesetzt werden, wenn er kein ausreichendes Wissen über die Benutzung eines Produktes besitzt. Bei der Verwendung von Software findet man Assistenzfunktionen zum Beispiel in Form von Tooltips. Diese kann der Benutzer sichtbar machen, indem er die Maus auf einer Schaltfläche positioniert, deren Funktion er nicht kennt. Assistenzsysteme sind jedoch nicht auf Softwareanwendungen beschränkt. Autos der neueren Generation enthalten zum Beispiel Fahrassistenzfunktionen, die beim Einparken oder beim Halten der Fahrspur helfen. Die Notwendigkeit von Assistenzfunktionen ist nicht auf bestimmte Orte beschränkt. Durch den Einsatz von Smartphones und Tablets können Assistenzsysteme entwickelt werden, die der Nutzer mit sich führen kann. Informationen wie zum Beispiel Bedienungsanleitungen können vor Ort und unmittelbar angezeigt werden, sodass offene Fragestellungen direkt beantwortet werden können. Assistenzbedarf ergibt sich im Zusammenhang mit Objekten der realen Welt, die den Anwender umgibt. Informationen und Lerninhalte zu diesen Objekten liegen digital gespeichert vor. Durch die bildbasierte Objekterkennung wird die Lücke zwischen den real existierenden Objekten und den zugehörigen virtuellen Daten geschlossen. Der Nutzer erhält auf natürliche Art und Weise Assistenzfunktionen zu einem Objekt, indem er das reale Objekt abfotografiert.

Das Problem der Objekterkennung lässt sich allgemein wie folgt definieren. Für ein Bild i soll bestimmt werden, ob eine Instanz einer Objektklasse w_j auf dem Bild zu sehen ist. Hierzu wird mit visuellen Merkmalen eine Bildrepräsentation x_i erstellt. Gesucht wird ein Maß, welches die Wahrscheinlichkeit angibt, dass die Bildrepräsentation x_i eine Instanz der Objektklasse w_j beschreibt:

$$p(w_j|x_i)$$

2.2 Begriffserklärung Kontext und Kontextsensitivität

Informationen werden als Kontext bezeichnet, wenn sie für die Charakterisierung einer Situation eines Interaktionsteilnehmers eingesetzt werden können. Eine differenziertere Definition des Begriffs Kontext wurde von Dey und Abowd vorgestellt:

"Kontext ist jede Information, die verwendet werden kann um die Situation einer Entität zu charakterisieren. Eine Entität ist eine Person, ein Ort oder ein Objekt, welche für die Interaktion zwischen einem Anwender und einer Anwendung, einschließlich dem Anwender und Anwendungen, als relevant angesehen wird." [41]

Key und Abowd stellen dazu vier primäre Kategorien von Kontextinformationen vor, die von einer Entität gewonnen werden können: Ort, Identität, Aktivität und Zeit. Weiterhin liefern sie eine Definition für kontextsensitive (engl. context-aware) Systeme:

"Ein System ist kontextsensitiv, wenn es Kontext verwendet um dem Anwender sachdienliche Informationen und/oder Dienste anzubieten. Die Sachdienlichkeit bezieht sich auf die Tätigkeit des Anwenders." [41]

In dieser Arbeit werden Kontextinformationen in Form der Identität von Komponenten des *AMFIS*-Systems mit Hilfe der Kamera mobiler Endgeräte erfasst. Das Anzeigen von relevanten Assistenzinformationen zu diesen Objekten führt zu einem kontextsensitiven System, das den Anwender bei der Benutzung der *AMFIS* Komponenten unterstützt.

2.3 Begriffserklärung Mobiles Lernen

Das mobile Assistenzsystem *Mobile Tutor* des Fraunhofer IOSB gehört zu den Anwendungen des Mobilen Lernens. In den vergangenen Jahren wurde die Forschung und Entwicklung solcher Anwendungen von vielen Organisationen vorangetrieben. Durch die dezentrale Weiterentwicklung entstanden eine Vielzahl von Definitionen für das Mobile Lernen. Winters [42] teilt diese Definitionen in vier Kategorien ein um zum Verständnis von Mobilem Lernen beizutragen.

1. **Technozentristische Perspektive**
   In dieser Perspektive wird das Lernen auf mobilen Endgeräten wie zum Beispiel Handy, Personal Digital Assistant (PDA), Smartphone, iPod oder der PlayStation Portable als Mobiles Lernen bezeichnet.

2. **Beziehung zum E-Learning**
   Hierbei wird Mobiles Lernen als Erweiterung des E-Learning verstanden.

3. **Erweiterung des formalen Lernens**
   Diese Sichtweise stellt Mobiles Lernen als Ergänzung zum formalen Lernen dar.

4. **Lernerzentristische Perspektive**
   Diese Perspektive stellt die Mobilität des Lerners in den Vordergrund. Demnach wird jede Art von Lernvorgang, der nicht an einem festen, vorbestimmten Platz stattfindet oder ein Lernvorgang, bei dem der Benutzer die Lernmöglichkeiten von mobilen Technologien nutzt, als Mobiles Lernen bezeichnet.

Moses fasst diese vier Perspektiven zu einer Defintion zusammen, um eine einheitliche Definition für den Begriff Mobiles Lernen zu schaffen:

"Mobiles lernen ist eine Form des E-Learnings, die jede Art von Lernen unter der Verwendung von mobilen Geräten beinhaltet, um eine an jedem Ort und zu jeder Zeit möglichen Lernerfahrung zu erzeugen, um auf die Bedürfnisse unterschiedlicher Lernender einzugehen und um ihre formalen Lernerfahrungen zu erweitern." [43]

2.4 Herausforderungen

Die Herausforderungen bei der bildbasierten Kontextdetektion mit Hilfe eines mobilen Endgerätes lassen sich im Wesentlichen auf zwei Punkte reduzieren. Zum einen soll die Erkennung der Objekte möglichst fehlerfrei sein und zum anderen darf die Berechnung nicht lange dauern, damit die Anwendung benutzerfreundlich ist.

Der Erfolg der Klassifikation hängt stark von der Qualität der Bilder ab. Variable Lichtverhältnisse sorgen dafür, dass das gleiche Objekt auf unterschiedlichen Bildern, wie in Abbildung 2.1, verschiedenartig aussieht. Das Wetter und die Tatsache, ob ein Bild innerhalb eines Gebäudes unter Kunstlicht oder im Freien aufgenommen wurde, sorgen dafür, dass die Intensitätswerte sich nicht nur additiv oder multiplikativ unterscheiden. Je nach Lage der Lichtquelle können Schatten entstehen. Spiegelnde Oberflächen führen zu störenden Artefakten im Bild. Der Abstand zwischen Fotograf und Objekt sowie die verwendeten Zoomstufen entscheiden, wie groß das Objekt im Bild erscheint. Dies kann dazu führen, dass das Objekt nur einen kleinen Bereich des Bildes ausfüllt oder dass das Objekt über die Bildränder hinausragt. Bei schlecht gewählter Aufnahmeposition können andere Gegenstände das Objekt teilweise verdecken. In den meisten Fällen füllt ein Objekt nicht das gesamte Bild aus. Dies führt dazu, dass das Bild neben dem Objekt zu einem Großteil aus Hintergrund besteht. Die unkorrelierten Hintergrundinformationen stören bei der Konstruktion einer Bildrepräsentation, vor allem wenn der Hintergrund nicht homogen ist. Beim Bildaufnahmeprozess wird das dreidimensionale Objekt auf die zweidimensionale Bildebene projiziert. Je nachdem von welcher Seite dreidimensionale Objekte fotografiert werden, sehen diese auf den Bildern sehr unterschiedlich aus. Ziel bei der Kontextdetektion ist es, die Objekte unabhängig vom Beobachtungspunkt der Kamera zu erkennen.

Die Qualität eines Bildes hängt ebenfalls von der Qualität der Kamerahard- und software der mobilen Endgeräte ab. Diese verbessern sich zwar kontinuierlich, allerdings reicht die Bildqualität nicht an die Bildqualität von Stand-Alone Digitalkameras heran. Vor allem unter schwachen Lichtverhältnissen enthalten diese Bilder unerwünschte Effekte. So führt der automatische Weißabgleich teilweise zu überbelichteten Bereichen im Bild und die mangelhaften Bildstabilisatoren zu verwackelten und verrauschten Bildern.

Abbildung 2.1: Herausforderungen bei der bildbasierten Objektdetektion: Beide Bilder unterscheiden sich bezüglich Hintergrund, Ansichtswinkel, Lichtverhältnissen, Schattierung und Spiegelung, zeigen jedoch zwei Instanzen der gleichen Objektklasse.

Neben den verwendeten Algorithmen hängt die Rechenzeit für die bildbasierte Objekterkennung von der Hardware des verwendeten mobilen Endgeräts ab. Obwohl die Leistungsfähigkeit aktueller Hardware in Smartphones und Tablets ein hohes Niveau erreicht hat und weiter steigt, sind sie längst nicht so leistungsstark wie PCs, bei deren Entwicklung

nicht auf geringe Größe und Akkubetrieb geachtet werden muss. Abschnitt 3.1 behandelt die Gegebenheiten der Hardware aktueller Smartphones und Tablets.

2.5 Einsatzgebiete für die Kontextdetektion mit Smartphones und Tablets

Das Mobile Lernen ist nicht das einzige Anwendungsgebiet für die bildbasierte Objekterkennung. Im Folgenden werden weitere Anwendungsbereiche vorgestellt.

Marketing

Für das Marketing ergeben sich durch die bildbasierte Objekterkennung mit einem mobilen Endgerät ganz neue Vermarktungsstrategien. Die Kamera der Geräte kann eingesetzt werden um statische Werbeformen, wie zum Beispiel Plakate oder Printanzeigen, durch mobile interaktive Werbemittel zu ergänzen oder zu ersetzen. Dies können zum Beispiel Gewinnspiele, Rabattgutscheine, Musikvideos oder Leseproben sein. Weiterhin kann auch direkt eine Kaufmöglichkeit des Produkts auf dem mobilen Endgerät angezeigt werden. Weit verbreitet ist die Verwendung von Quick Response Codes (QR-Codes, Abbildung 2.2a), die, wenn sie fotografiert und decodiert werden, auf Webseiten verlinken. Die Volkswagen AG nutzt in ihrer App *VW seeMore* (Abbildung 2.2b) in Bilder integrierte unsichtbare Codes um den Zugriff auf diverse multimediale Inhalte zu ermöglichen [44]. Sie erweitern damit den statischen Inhalt um interessante, interaktive Inhalte, die auf dem Smartphone genutzt werden können.

a) b)

Abbildung 2.2: a) QR-Codes ermöglichen einen markerbasierten Informationszugang. [45]
b) Die *VW seeMore* App erkennt in Bilder integrierte unsichtbare Codes um Marketinginformationen anzuzeigen. [44]

Zur Verknüpfung von zweidimensionalen Bildern mit Informationen aus dem Internet sind jedoch längst keine sichtbaren oder unsichtbaren Markierungen (Codes) mehr notwendig. Apps mit der entsprechenden Bildverarbeitung können zum Beispiel abfotografierte Filmplakate erkennen um zugehörige Trailer abzuspielen. Ein abfotografiertes Werbeplakat einer Autovermietung kann zu einer Wegbeschreibung oder zu Flotten- und Preisinformationen führen [46]. Eine kommerziell verfügbare App für diese Zwecke ist *U Snap* von WallDecaux (Abbildung 2.3) [47]. Diese erkennt mit Hilfe von Bildverarbeitungsalgorithmen Motive abfotografierter Plakate und verknüpft diese mit Mehrwerten im Internet.

Abbildung 2.3: Die *U Snap* App verknüpft Plakate mit Mehrwerten im Internet. [47]

Aktuelle Algorithmen beschränken sich jedoch nicht auf das Erkennen von zweidimensionalen Objekten. Die Fähigkeit dreidimensionale Objekte erkennen zu können kann auch im Marketingbereich eingesetzt werden. Auf der Automobilmesse IAA nutzen die Aussteller bisher QR-Codes um den Messebesuchern Zusatzinformationen zu ihren Produkten zugänglich zu machen. Anstelle der extra angebrachten Codes könnten Besucher die Autos auch direkt abfotografieren. Durch die Identifikation des Automodells im Bild können dem Nutzer zum Beispiel Ausstattungsmerkmale des Produkts angezeigt werden.

Touristik

Im Bereich Touristik bewegen sich Menschen außerhalb ihres gewohnten Umfeldes. Sie bereisen neue Gebiete und Städte, in denen sie sich nicht auskennen. Die meisten Urlauber haben dabei das Ziel mehr über ihren Urlaubsort zu erfahren. Zu diesem Zweck nutzen sie Reiseführer und Informationsbroschüren, um Informationen zu Gebäuden und anderen Sehenswürdigkeiten nachlesen zu können. Aufgrund der eingeschränkten Anzahl an Seiten beinhalten diese Informationsquellen lediglich einen kleinen Auszug der interessanten Informationen. Weiterhin bestehen diese Inhalte nur aus Texten und statischen Bildern.

Abbildung 2.4: Die bildbasierte Erkennung von Gebäuden erlaubt die Entwicklung von interaktiven Reiseführern. [48]

Die weite Verbreitung von Smartphones ermöglicht ganz neue Konzepte für die Touristikbranche. GPS und Kameras von mobilen Endgeräten können genutzt werden um Touristen einen vereinfachten Zugang zu Informationen, passend zum Kontext, in dem sich der Benutzer gerade befindet, zu bieten. Durch das Abfotografieren eines Gebäudes [48] können Hintergrundinformationen auf dem Display dargestellt werden (Abbildung 2.4). Dabei sind die Informationen nicht auf Texte und Bilder beschränkt, sondern können durch multimediale und interaktive Inhalte ergänzt werden. Neben Gebäuden können zum Beispiel auch Fotos von Bussen und S-Bahnen als Informationszugänge dienen. Dabei können passende Informationen wie der Netzfahrplan des öffentlichen Nahverkehrs angezeigt werden.

Museum

Neben der Anwendung für den Touristikbereich können bildbasierte Objekterkennungsverfahren auch in Museen eingesetzt werden. Die Präsentation von Exponaten geschieht in den meisten Museen auf eine passive Art und Weise. Möchte der Besucher Hintergrundinformationen erhalten, muss er in einer Broschüre aufwendig nach den gewünschten Informationen suchen. Bildbasierte Objekterkennung auf Mobilgeräten kann dem Besucher die Suche nach Zusatzinformationen abnehmen [38], [40], [49]. Dies ermöglicht eine neue Form der Interaktion zwischen Museumsbesucher und den Ausstellungsstücken. Ein Blättern in Broschüren ist nicht mehr erforderlich, denn Erklärungen zu abfotografierten Exponaten oder Informationen zur Biographie des Künstlers werden dem Besucher automatisch auf dem mobilen Endgerät präsentiert.

Kontextdetektion für Sehbehinderte

Die Kontextdetektion mit Hilfe von Bildverarbeitung auf einem Smartphone kann auch als Hilfe für Sehbehinderte eingesetzt werden. Diese haben das Handicap, dass sie entweder gar nicht oder nur teilweise sehen können. Dies führt dazu, dass sie sich zum einen nur erschwert orientieren können und zum anderen Objekte nicht identifizieren können. Bei der Orientierung sind sie auf Unebenheiten am Boden angewiesen, die sie mit Hilfe eines Blindenstocks erfühlen. Hierdurch erhalten sie jedoch keine Informationen, ob sie sich zum Beispiel vor einem bestimmten Gebäude befinden. Eine Smartphone App, die in der Lage ist, den Kontext mit Hilfe der Kamera zu erkennen, ermöglicht es Sehbehinderten selbstständiger zu werden. Der Sehbehinderte kann, wenn er zum Beispiel das Rathaus einer Stadt aufsuchen möchte, ein Foto des Gebäudes machen und wird vom mobilen Endgerät informiert, ob er vor dem gewünschten Gebäude steht. Das Ergebnis der bildbasierten Detektion kann ihm dabei akustisch übermittelt werden.

Ein Smartphone kann ebenfalls beim Identifizieren von Objekten helfen. Ohne technische Hilfe erkennen Blinde Objekte durch Ertasten. Sie können dabei das Objekt jedoch nur erkennen, wenn sie es bereits kennen. Mit Hilfe einer Smartphone App können Blinde neue Objekte kennen lernen, indem diese von der Software erkannt und entsprechende Informationen über das Objekt übermittelt werden. Eine solche App hilft ebenfalls, wenn sich Objekte durch Ertasten nicht eindeutig identifizieren lassen, weil es mehrere Objekte gibt, die sich zwar in Farbe und Textur unterscheiden, jedoch die gleiche Form besitzen. Objekte, die sich zu weit weg befinden oder zu groß sind, sodass sie nicht in der Reichweite eines Menschen liegen, können auf diese Art und Weise Sehbehinderten ebenfalls zugänglich gemacht werden. Mit dem *LookTel Money Reader* [50] (Abbildung 2.5), der Geldscheine erkennt, und dem *LookTel Recognizer* [50], der beim Unterscheiden von alltäglichen Objekten hilft, gibt es bereits Apps für das iPhone, die speziell für die Sehbehindertenhilfe entwickelt wurden.

Abbildung 2.5: Die iPhone App *LookTel Money Reader* hilft sehbehinderten Menschen bei der Identifikation von Geldscheinen. [50]

3. Bildverarbeitung auf Smartphones und Tablets mit Android Betriebssystem

Bei der Entwicklung von Bildverarbeitungsanwendungen für Smartphones und Tablets muss auf die spezifischen Charakteristika geachtet werden, die sowohl Hardware als auch Software mobiler Endgeräte mit sich bringen. Dieses Kapitel gibt einen Überblick über die Hardware aktueller Android-basierter Endgeräte, sowie der Software, die bei der Entwicklung von Apps verwendet wird. Die Basis für die Implementierung von Apps für das Android Betriebssystem bildet das Android Software Developement Kit[1]. Dieses kann mit dem Android Native Developement Kit[2] erweitert werden, um nativen Code für performancekritische Anwendungsbereiche zu integrieren. Das Kapitel schließt mit der Vorstellung der Bildverarbeitungsbibliothek OpenCV[3], welche in dieser Arbeit zum Einsatz kommt.

3.1 Hardware

Bei einem Desktop PC spielt die Größe der Hardware kaum eine Rolle. Somit hat sich ein System durchgesetzt, bei dem die Einzelkomponenten, wie zum Beispiel CPU, Grafikkarte und Speicher, modular mit Hilfe des Motherboards verknüpft werden. Diese Bauweise ermöglicht die Kühlung der Komponenten mit aktiven Lüftern. Die Energieversorgung findet per Kabel statt, sodass der Fokus bei der Entwicklung von Hardware für Desktop PCs weniger auf stromsparenden, sondern vielmehr auf rechenstarken Komponenten liegt.

Smartphones und Tablets unterscheiden sich jedoch fundamental von dieser Bauweise, sodass bei der Entwicklung von Hardware für mobile Endgeräte andere Ziele im Fokus liegen. Der Anwender wünscht sich möglichst flache Geräte. Ein Großteil des Volumens eines mobilen Endgerätes wird zusätzlich von dem Akku belegt, sodass für die restliche Hardware wenig Platz bleibt. Aufgrund der platzsparenden Bauweise muss auf die Verwendung von aktiven Kühlern verzichtet werden. Um dem Anwender eine möglichst lange Akkulaufzeit bieten zu können, ist eine stromsparende Konzipierung der verwendeten Hardware unabkömmlich. Die Lösung für diese Herausforderungen bieten hoch integrierte Systems-on-a-Chip [51]. Diese verbinden alle oder einen Großteil der relevanten Komponenten auf einem

[1] Ein Download der aktuellen Version des Android Software Developement Kit ist auf der Seite `http://developer.android.com/sdk/index.html` möglich.

[2] Unter `http://developer.android.com/sdk/ndk/index.html` kann das Android Native Developement Kit heruntergeladen werden.

[3] Die Webseite von OpenCV ist unter `http://opencv.org` zu finden.

Chip. Hierzu gehören neben mehreren Rechenkernen (CPU), die Grafikeinheit (GPU), Speicher-Controller, der Speicher sowie I/O-Controller für Schnittstellen, Ports für USB-Geräte, SD-Karten, Mikrofone, Lautsprecher, Touch-Sensoren und Mobilfunkmodems. Die Rechenleistung der CPU-Kerne kann dabei nicht mit den Rechenkernen von Desktop PCs mithalten. Damit der Benutzer bei der Bedienung mobiler Geräte trotzdem das Gefühl einer schnellen Hardware bekommt, werden zusätzlich mehrere Beschleuniger für spezielle, rechenaufwendige Aufgaben integriert. So beinhaltet der Apple A4 SoC, der im iPhone 4s und dem iPad 2 zum Einsatz kommt, einen speziellen Audio-Prozessor, der die Störgeräusche für die Sprachsteuerung Siri herausfiltert [51]. Weitere Anwendungsgebiete solcher Beschleuniger sind zum Beispiel die Videowiedergabe oder die Verschlüsselung.

Nvidia, Qualcomm, Samsung und Texas Instrument produzieren SoCs und kaufen hierzu Standardkomponenten wie zum Beispiel die Rechenkerne als geistiges Eigentum (*engl.* Intellectual Property, kurz IP) und kombinieren diese mit Eigenentwicklungen zu einem kompletten SoC-Bauplan. Die Rechenkerne der meisten aktuellen Smartphones und Tablets kommen von der britischen Firma ARM, die sich auf die Entwicklung mobiler CPUs spezialisiert hat. Nvidia, Samsung und Texas Instrument integrieren lediglich den Funktionsblock der ARM-Kerne. Qualcomm hingegen nutzt eine Architekturlizenz um eigene ARM-kompatible Rechenkerne zu entwickeln. Tabelle 3.1 gibt einen Überblick über aktuelle Systems-on-a-Chip, die in modernen Smartphones und Tablets zum Einsatz kommen. SoCs mit Cortex-A8 ARM-Kernen können Befehle nicht umsortieren, da sie mit dem In-Order-Prinzip arbeiten. Dieses Prinzip ist stromsparend, führt jedoch zu einer schlechten Auslastung der CPU. Die neueren Cortex-A9-Kerne setzen aus diesem Grund auf das Out-of-Order Prinzip. SoCs auf Basis von Cortex-A15 stehen noch nicht zur Verfügung. Ein gängiges Konzept zum Stromsparen ist die zusätzliche Integration eines Rechenkerns, der nicht auf hohe Performance, sondern auf niedrige Leistungsaufnahme optimiert ist. Der Tegra 3 Quadcore von Nvidia verwendet zum Beispiel neben seinen vier Hauptkernen einen fünften Kern, der lediglich mit 500 MHz getaktet ist.

SoC	Kern	# Kerne	Taktung in GHz
Samsung Exynos 3	Cortex-A8	1	1
Samsung Exynos 4	Cortex-A9	2 oder 4	1,2 bis 1,5
Samsung Exynos 5	Cortex-A15	2	1,7
Nvidia Tegra 2	Cortex-A9	2	bis 1,2
Nvidia Tegra 3	Cortex-A9	4 + 1	bis 1,4
Nvidia Tegra 4	Cortex-A15		
Texas Instrument OMAP 3	Cortex-A8	1	0,6 bis 1,2
Texas Instrument OMAP 4	Cortex-A9	2	1 bis 1,8
Texas Instrument OMAP 5	Cortex-A15	2	bis 2
Qualcomm Snapdragon S2	Scorpion	1	0,8 bis 1,4
Qualcomm Snapdragon S3	Scorpion	2	1,2 bis 1,7
Qualcomm Snapdragon S4	Krait	2 oder 4	1,5 bis 1,7

Tabelle 3.1: Überblick über die in Smartphones und Tablets verwendeten Systems-on-a-Chips der Firmen Samsung [52], Nvidia [53], Texas Instrument [54] und Qualcomm [55].

Die Version der eingesetzten ARM-Kerne, die Anzahl der Kerne und die Taktfrequenz sind jedoch nicht die einzigen Merkmale, die Einfluss auf die Performance eines mobilen Endgerätes haben. Neben der GPU und dem internen Bus-System unterscheiden sich die SoCs verschiedener Hersteller vor allem durch die Optimierung und die Integration der verschiedenen Beschleuniger. Die Rechenleistung der Beschleuniger hat jedoch nur dann einen Einfluss auf die Performance, wenn diese gezielt von der Software genutzt werden. Die

Performance der Algorithmen für die bildbasierte Objekterkennung dieser Arbeit wird in Abschnitt 6.3 ausführlich auf einem Samsung Galaxy S2 GT-I9100 mit 1,2 GHz Dual-Core ARM Cortex-A9 CPU evaluiert. Weiterhin wird die finale Parametrisierung des verwendeten Objekterkennungsverfahrens auf mehreren mobilen Endgeräten mit unterschiedlichen SoCs getestet.

3.2 Android Betriebssystem

Android ist eine Open-Source-Plattform für Mobilgeräte wie Smartphones und Tablets. Angestoßen wurde das Android Projekt von Google. Eigentümer und Entwickler ist die Open Handset Alliance, ein Konsortium aus 84 Technologie- und Telekommunikationsunternehmen, bei dem neben Google zum Beispiel T-Mobile, Nvidia oder Samsung mitwirken. Das Android Betriebssystem wurde mit dem Ziel entworfen, dass es auf Geräten mit unterschiedlichen Ausprägungen eingesetzt werden kann. Es sind keine Einschränkungen bezüglich Bildschirmgröße, Auflösung oder Chipsatz vorhanden.

Abbildung 3.1 zeigt die Systemarchitektur [56] [57] des Android-Betriebssystems. Sie besteht aus den Schichten Linux Kernel, Libraries, Android Runtime, Application Framework und Applications:

1. **Linux Kernel**
   Die unterste Schicht und damit die Basis des Betriebssystems bildet der Linux-Kernel 2.6, welcher von Google an die Bedürfnisse mobiler Geräte angepasst wurde. Dieser bringt die zentralen Systemdienste wie Sicherheit, Speichermanagement, Prozessmanagement, Netzwerk-Stack und Treibermodell mit sich. Er stellt die Abstraktionsschicht zwischen Hardware und Software-Stack dar.

2. **Libraries**
   Die zweite Schicht enthält eine Sammlung an nativen Bibliotheken. Diese sind in C oder C++ geschrieben und werden speziell für die verwendete Hardware kompiliert um performancekritische Funktionen schnell berechnen zu können. Hierzu gehören zum Beispiel die Verwaltung der Benutzeroberfläche mit Hilfe des Surface Managers oder die Wiedergabe von Audio- oder Videodateien auf Basis des Media Frameworks.

3. **Android Runtime**
   Die Android Laufzeitumgebung besteht aus einer Ansammlung an Kernbibliotheken und der Dalvik Virtual Machine, welche für die Ausführung der in Java geschriebenen Android Anwendungen verantwortlich ist. Da die Standard Java Virtual Machine (JVM) den Gegebenheiten von mobilen Geräten nicht gerecht wird, entschied Google eine eigene Virtual Machine zu entwickeln, die für den Gebrauch auf mobilen Endgeräten optimiert ist [58]. Bei der Entwicklung der Dalvik Virtual Machine fanden die speziellen Eigenschaften mobiler Geräte wie zum Beispiel die Energiezufuhr per Batterie, langsame Prozessoren oder wenig Arbeitsspeicher Beachtung. Während die Standard Java Virtual Machine auf einem Kellerautomaten basiert, wird bei der Dalvik VM eine Registermaschine als virtuelle Prozessorarchitektur verwendet. Jede Anwendung bekommt bei ihrer Ausführung eine eigene Instanz der Dalvik VM. Da die Dalvik VM keinen Java-Bytecode ausführen kann, muss der Java-Bytecode zuvor in Dalvik-Bytecode umgewandelt werden.

4. **Application Framework**
   Der Anwendungsrahmen enthält eine Ansammlung an Java-Bibliotheken, die speziell für Android entwickelt wurden. Diese stehen dem Anwendungsentwickler zur Verfügung. Die von den Bibliotheken angebotenen Dienste ermöglichen einen einfachen Zugriff auf diverse Systemfunktionen, wie zum Beispiel WLAN, GPS und andere Sensoren.

5. **Applications**
   In der obersten Schicht sind die in Java entwickelten Anwendungen zu finden, die vom Benutzer bedient werden. Sie werden in Form einer Android Package Datei (.apk) auf dem mobilen Endgerät installiert.

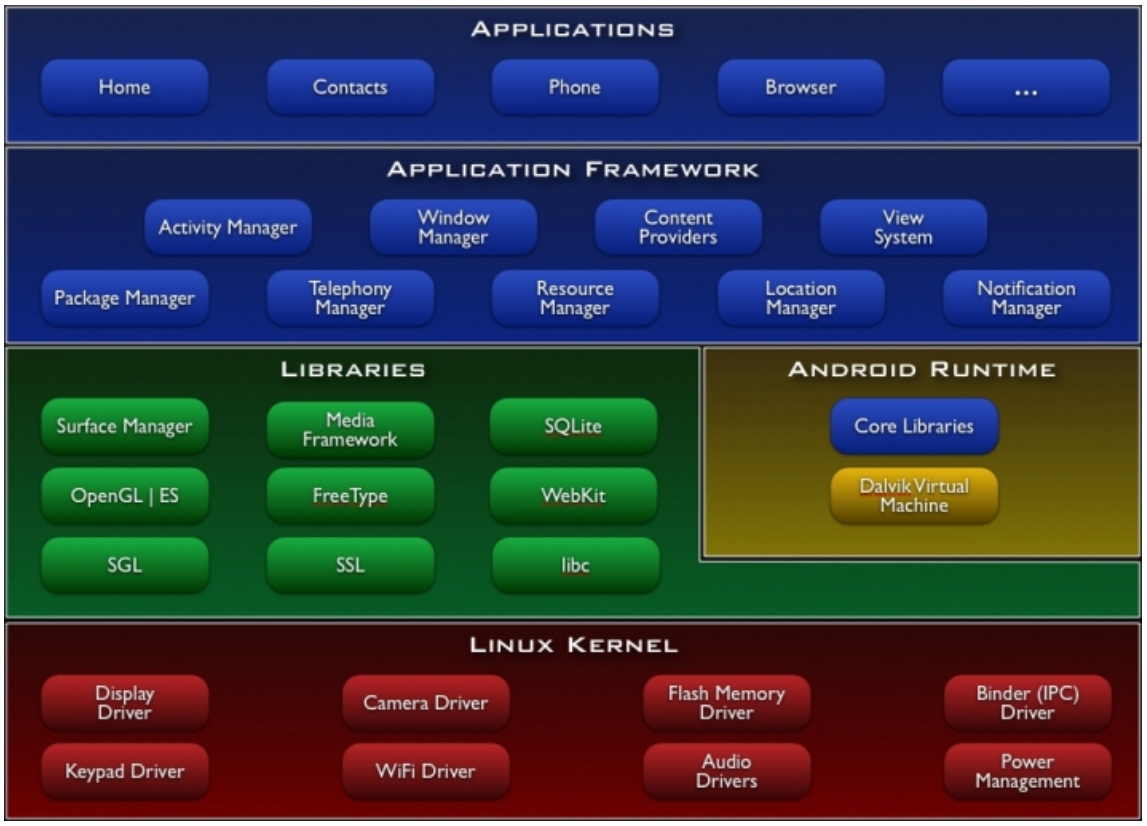


Abbildung 3.1: Die Systemarchitektur des Android Betriebssystems besteht aus den fünf Schichten Linux Kernel, Libraries, Android Runtime, Application Framework und Applications. [56]

3.3 Android Software Developement Kit (SDK)

Für die Entwicklung eigener Android Anwendungen steht das Android SDK zur Verfügung. Dieses bringt neben einer Sammlung an Bibliotheken einige Werkzeuge mit, die den Entwicklungsvorgang vereinfachen. Hierzu gehören zum Beispiel der *Dalvik Debug Monitor Service* zum Debuggen oder ein Emulator, der eine Android Umgebung simuliert um Anwendungen ohne Hardwaregerät testen zu können.

Da das Android Betriebssystem kontinuierlich weiterentwickelt wird, entsteht eine Vielzahl an Versionen und damit eine starke Fragmentierung. Bei der Entwicklung einer Android Anwendung ist weniger die Android-Versionsnummer, sondern vielmehr das API-Level von Bedeutung. Die Wahl des API-Levels gibt an, auf welchen Android Geräten die Software später ausgeführt werden kann und auf welchen nicht. Tabelle 3.2 gibt einen Überblick über die verfügbaren Android-Versionen mit den zugehörigen API-Leveln. Umso niedriger das gewählte API-Level liegt, auf umso mehr Geräten kann die Anwendung später verwendet werden. Für die Entwicklung der Android App dieser Arbeit wird API-Level 8 verwendet. Sie ist somit auf 94,6% aller aktiv genutzten Android Geräte lauffähig [59].

Android Version	API-Level	Name	Verteilung
Android 1.5	3	Cupcake	0,2%
Android 1.6	4	Donut	0,5%
Android 2.1	7	Eclair	4,7%
Android 2.2	8	Froyo	17,3%
Android 2.3 - 2.3.2	9	Gingerbread	0,4%
Android 2.3.3 - 2.3.7	10	Gingerbread	63,6 %
Android 3.1	12	Honeycomb	0,5%
Android 3.2	13	Honeycomb	1,9%
Android 4.0 - 4.0.2	14	Ice Cream Sandwich	0,2%
Android 4.0.3 - 4.0.4	15	Ice Cream Sandwich	10,7%

Tabelle 3.2: Alle aktiv genutzten Android Versionen. Die Verteilung gibt an, welche Android Versionen am weitesten verbreitet sind. [59]

3.4 Android Native Developement Kit (NDK)

Android unterstützt nur die Programmiersprache Java. Das Android Native Developement Kit ist eine Erweiterung des Android SDKs, welches die Integration von nativem Code ermöglicht, welcher zum Beispiel in C oder C++ geschrieben ist. Die Auslagerung von Bereichen einer App in nativen Code bietet sich für performancekritische Bereiche einer Android App an. Aus dem nativen Code werden *shared Libraries* erzeugt. Die enthaltenen nativen Funktionen können mit Hilfe des Java Native Interfaces[4] (JNI) in Java aufgerufen werden. In dieser Arbeit wird das Android NDK eingesetzt, um den bildverarbeitenden Teil der Android Anwendung in C++ implementieren zu können.

3.5 Open Source Computer Vision Library (OpenCV)

Für die Implementierung der Bildverarbeitung für die bildbasierte Objekterkennung wird in dieser Arbeit die Open Source Computer Vision Library (OpenCV) [60] eingesetzt. Diese ist eine freie Bibliothek mit einer Vielzahl an Funktionen für Standardprobleme der Bildverarbeitung und des maschinellen Sehens (*engl.* Computer Vision). Initiiert wurde das OpenCV Projekt im Jahr 1999 durch Intel. Mittlerweile wird die Entwicklung von Willow Garage betreut. OpenCV steht unter den Bedingungen der BSD-Lizenz zur Verfügung und ermöglicht akademische als auch kommerzielle Nutzung. Ziel bei der Entwicklung von OpenCV ist die effiziente Berechnung von Bildverarbeitungsalgorithmen für Echtzeitanwendungen. OpenCV kann unter Windows, Linux, Mac und Android eingesetzt werden. Neben den C++, C und Python Interfaces steht extra für Android ein Java Interface zur Verfügung.

Die Algorithmen von OpenCV wurden ursprünglich für die Anwendung auf Intel-Prozessoren optimiert. Da Android erst seit OpenCV Version 2.2 unterstützt wird, besitzen viele Algorithmen noch keine vollständige Optimierung für die in mobilen Endgeräten verbauten ARM-Prozessoren. Welcher Geschwindigkeitsgewinn durch die Optimierung erreicht werden kann, zeigt der Vergleich der OpenCV Version 2.3.1 mit der aktuellen in dieser Arbeit verwendeten Version 2.4.1. Messungen auf einem Samsung Galaxy S2 GT-I9100 mit 1,2 GHz Dual-Core ARM Cortex-A9 CPU haben ergeben, dass die Berechnung von SURF Features bei Version 2.4.1 im Durchschnitt um den Faktor $6,6$ schneller ist.

[4] Die Spezifikation des Java Native Interfaces ist unter `http://docs.oracle.com/javase/6/docs/technotes/guides/jni/spec/jniTOC.html` zu finden.

Die mehr als 2500 Algorithmen für die Bildverarbeitung gliedern sich in mehrere Module. Die folgenden Module wurden in dieser Arbeit verwendet:

- core-Modul: Grundlegende Datenstruktuen und Funktionen
- highgui-Modul: Interface für den Bilderzugriff
- imgproc-Modul: Algorithmen für die lineare und nicht-lineare Bildfilterung, für geometrische Bildtransformationen, Farbraumtransformationen und Histogramme
- features-2d- und nonfree-Modul: Feature Detektoren und Deskriptoren
- ml-Modul: Verfahren des maschinellen Lernes, wie zum Beispiel die Support Vector Machine

4. Methoden der Bildverarbeitung zur Objektidentifikation

Für die bildbasierte Erkennung der *AMFIS* Komponenten wird in dieser Arbeit das *Bag of Words* (BoW) Verfahren verwendet. Hierbei wird für jedes Bild ein globaler Merkmalsvektor berechnet. Die Merkmalsvektoren der Trainingsbilder bilden die Grundlage um einen Klassifikator zu trainieren. Bei Testbildern dient der Merkmalsvektor als Eingabe für den Klassifikator. Die Berechnung des *Bag of Words* Merkmalsvektors gliedert sich in mehrere Teilabschnitte, welche in Abbildung 4.1 skizziert sind.

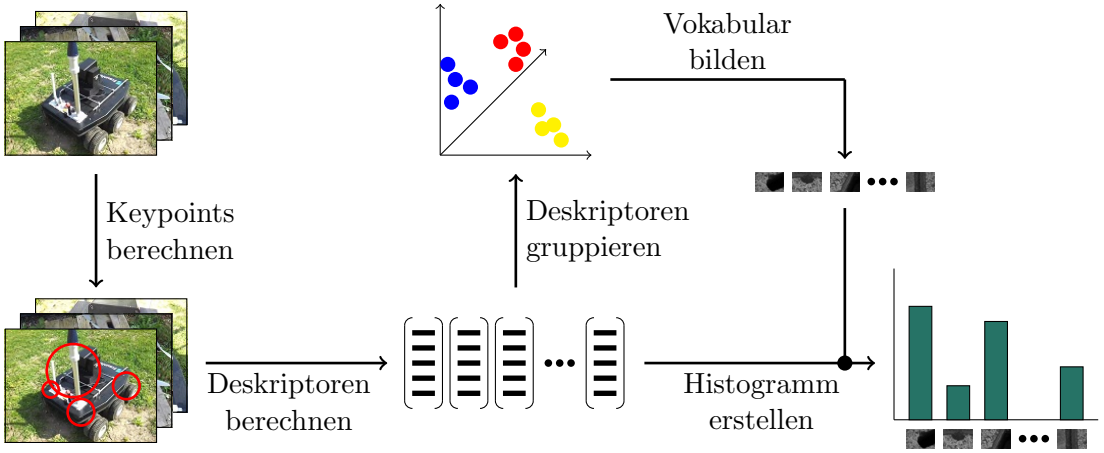


Abbildung 4.1: Der Basisalgorithmus für die Berechnung eines *Bag of Words* Histogramms.

Zunächst werden mit Hilfe von lokalen Features interessante Bildpunkte in Form von Keypoints detektiert. Anschließend werden Deskriptoren berechnet, welche den Inhalt des Bildausschnittes um den Keypoint herum beschreiben. Die Deskriptoren der Trainingsbilder werden in sogenannte Cluster gruppiert um ein visuelles Vokabular zu bilden, welches auch als Codebuch bezeichnet wird. Die Zentren dieser Cluster beschreiben die einzelnen Einträge des visuellen Vokabulars. Die Bildrepräsentation wird erstellt, indem den Deskriptoren eines Bildes die ähnlichsten Einträge des visuellen Vokabulars zugeordnet werden. Die Anzahl gefundener Codebucheinträge im Bild wird in einem Histogramm, dem sogenannten *Bag of Keypoints*, gezählt.

Die einzelnen Teilabschnitte des *Bag of Words* Verfahrens lassen sich jeweils durch verschiedene Algorithmen realisieren, die untereinander austauschbar sind. In den folgenden Abschnitten werden Berechnungsvorschriften vorgestellt, die in dieser Arbeit zum Finden einer optimalen *Bag of Words* Parametrisierung für die Anwendung auf mobilen Endgeräten evaluiert werden. Für die Keypoint- und Deskriptorenberechnung werden SIFT und SURF Features behandelt. Das visuelle Vokabular wird mit Hilfe des k-means Clusterings erzeugt. Für die Berechnung des *Bag of Words* Histogramms kann das Standardverfahren durch eine fuzzy Zuordnung der Deskriptoren zu den Codebucheinträgen oder durch räumliche Histogramme ergänzt werden. Das Kapitel schließt mit der Beschreibung der verwendeten Klassifikatoren. Neben dem k-Nearest-Neighbor Klassifikator werden Support Vector Machines vorgestellt, die anhand des *Bag of Words* Histogramms eines Bildes Hypothesen für die zu sehende Objektklasse aufstellen.

4.1 Lokale Features

Lokale Features bilden die Grundlage zur Berechnung eines *Bag of Words* Histogramms. Sie setzen sich durch einen Keypoint, auch Interestpoint genannt, und einem Deskriptor zusammen. Keypoints beschreiben dabei markante, reproduzierbare Punkte im Bild. Der Deskriptor dient als Merkmalsrepräsentation der Bildumgebung um den Keypoint herum. Die Berechnungsvorschriften für Keypoints und Deskriptoren sind unabhängig voneinander, sodass unterschiedliche Verfahren kombiniert werden können. An dieser Stelle werden die in dieser Arbeit verwendeten SIFT und SURF Features vorgestellt. SIFT und SURF enthalten jeweils Algorithmen für die Berechnung von Keypoints und Deskriptoren.

4.1.1 Scale-invariant feature transform

Der SIFT Algorithmus (Scale-invariant feature transform) wurde erstmals im Jahr 1999 von Lowe [18] vorgestellt und wird seitdem kontinuierlich weiterentwickelt [19]. Der Algorithmus beschreibt zum einen die Detektion von Keypoints und zum anderen die Berechnung des Deskriptors. SIFT Features sind skalierungs- und orientierungsinvariant und verhalten sich robust gegenüber Helligkeitsänderungen, Rauschen und Änderungen des Ansichtswinkels.

Keypoint-Detektion

Zur Detektion der Keypoints wird zunächst eine Skalenraumrepräsentation $L(x, y, \sigma)$ des Bildes erstellt um Skalierungsinvarianz zu erreichen. Hierzu wird das Eingabebild $I(x, y)$ mit einem Gaußfilter

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

mit zunehmend starken Glättungsfaktoren σ gefaltet:

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

Lowe schlägt vor, das Eingabebild wiederholt mit einem Gaußfilter zu glätten, sodass sich der Glättungsfaktor benachbarter Bilder um den konstanten Faktor k unterscheiden. Der Skalenraum wird nun erweitert, indem zusätzliche Oktaven hinzugefügt werden. Benachbarte Oktaven unterscheiden sich durch die Halbierung der Seitenlängen des Bildes. Der Skalenraum besteht nun aus einer Menge an Bildern, die mit zunehmender Skala stärker geglättet und mit zunehmender Oktave in der Auflösung reduziert sind. Anschließend wird der differenzielle Skalenraum erstellt, indem in jeder Oktave benachbarte Bilder voneinander abgezogen werden (Abbildung 4.2):

$$D(x, y, k\sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

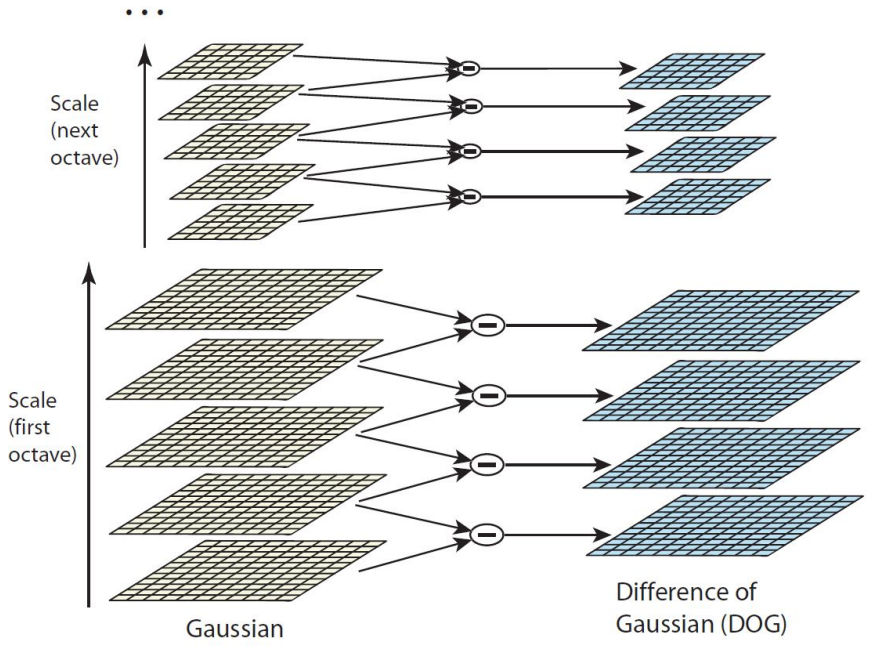


Abbildung 4.2: Berechnung des differenziellen Skalenraums: In jeder Oktave werden benachbarte Bilder, welche mit zunehmend starken Gaußfiltern geglättet wurden, voneinander subtrahiert. [19]

Die Subtraktion dieser Bilder wird als Difference-of-Gaussian (DoG) Funktion bezeichnet. Diese stellt eine Approximation des Laplacian-of-Gaussian (LoG) Verfahrens dar. Der Difference-of-Gaussian Ansatz ist weniger rechenaufwendig, da die Bilder zur Deskriptorberechnung sowieso Gauß-gefiltert werden und die DoG-Bilder durch einfache Bildsubtraktion erzeugt werden.

Im differenziellen Skalenraum werden geeignete Keypoint-Kandidaten in einer 26er Nachbarschaftsumgebung bestimmt, indem nach Extremstellen gesucht wird. Wie in Abbildung 4.3 zu sehen, werden neben den 8 Nachbarn der gleichen Ebene die jeweils neun Werte der darüber- und darunterliegenden Ebene betrachtet. Die lokalen Minima und Maxima des differenziellen Skalenraums beschreiben mögliche Keypointkandidaten.

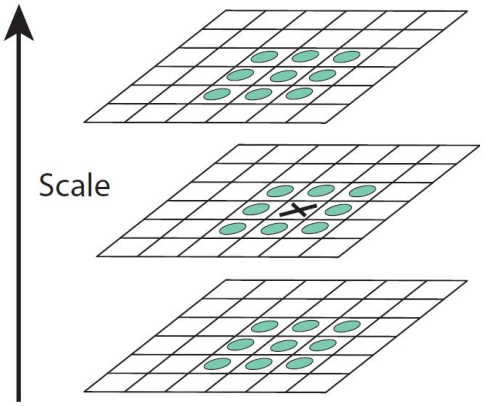


Abbildung 4.3: Bei der Detektion von SIFT Keypointkandidaten werden Extremstellen im differenziellen Skalenraum gesucht. Ein Pixel wird dazu mit seinen acht Nachbarn derselben Ebene und den jeweils neun Nachbarn der darüber- und darunterliegenden Ebene verglichen. [19]

Die subpixelgenaue Position eines Keypoints wird bestimmt, indem eine dreidimensionale quadratische Funktion an die Umgebung des Keypoints angepasst wird. Hierzu wird die Taylorentwicklung

$$D(\mathrm{x}) = D + \frac{\partial D^T}{\partial \mathrm{x}}\mathrm{x} + \frac{1}{2}\mathrm{x}^T\frac{\partial^2 D}{\partial \mathrm{x}^2}\mathrm{x}$$

der Skalenraumfunktion $D(x, y, \sigma)$ verwendet. $\mathrm{x} = (x, y, \sigma)^T$ beschreibt dabei den Offset zwischen diskreter und interpolierter Position des Keypoints. Die Extremstelle $\hat{\mathrm{x}}$ wird durch Ableiten und anschließendem Gleichsetzen mit Null berechnet. Durch Umformungen ergibt sich folgendes lineares Gleichungssystem zur Berechnung des Offsets:

$$\hat{\mathrm{x}} = \frac{\partial^2 D^{-1}}{\partial \mathrm{x}^2}\frac{\partial D}{\partial \mathrm{x}}$$

Die subpixelgenaue interpolierte Position wird durch Addition des Offsets zur ursprünglichen Position berechnet.

Anschließend werden Keypointkandidaten eliminiert, die aufgrund ihrer charakteristischen Eigenschaften später schlecht wiedererkannt werden können. Hierzu zählen Keypoints, die auf einer Kante liegen, oder Keypoints, deren Kontrast zu niedrig ist. Die Reduzierung der Keypointanzahl anhand des Kontrastes ist für die Anwendung auf mobilen Endgeräten von besonderer Bedeutung, da hierdurch die Zahl der Deskriptoren, die berechnet werden müssen, reduziert werden kann. Dies führt zu einer Verringerung der Rechenzeit. Keypoint-Kandidaten, deren Betrag des Funktionswertes $D(\hat{\mathrm{x}})$ an der Extremstelle kleiner als ein Schwellenwert τ ist, werden eleminiert:

$$|D(\hat{\mathrm{x}})| = |D + \frac{1}{2}\frac{\partial D^T}{\partial \mathrm{x}}\hat{\mathrm{x}}| < \tau$$

Orientierungsinvarianz wird erreicht, indem dem Keypoint anhand der lokalen Keypoint-Umgebung eine Hauptorientierung zugewiesen wird. Somit kann der Deskriptor relativ zur Hauptorientierung des Keypoints berechnet werden. Auf dem Gauß-geglätteten Bild, welches der Skalierung des Keypoints am nächsten kommt, werden Gradientenstärke $m(x, y)$ und -orientierung $\theta(x, y)$ berechnet:

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

$$\theta(x, y) = \tan^{-1}\left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}\right)$$

Aus den Gradienten in einem Bereich um den Keypoint herum wird ein Orientierungshistogramm mit 36 Bins gebildet, sodass ein Bin einen Bereich von 10° abdeckt. Die Gradienten werden dabei mit ihrer Stärke und einem Gaußfilter gewichtet. Die Maxima in dem Gradientenhistogramm spiegeln die Hauptorientierungen des Keypoints wieder. Es werden sämtliche Maxima als dominante Orientierungen des Keypoints akzeptiert, deren Histogrammwerte größer als 80% des globalen Maximums sind. Wenn mehrere Hauptorientierungen vorhanden sind, werden mehrere Keypoints erstellt, die zwar die gleiche Lage und Skalierung, jedoch unterschiedliche Orientierungen besitzen. Um die Genauigkeit zu erhöhen wird eine Parabel an die 3 Werte, die sich am nächsten zu einem Histogrammmaximum befinden, angepasst. Der Scheitel der Parabel gibt den interpolierten Orientierungswert an.

Berechnung des Merkmalsdeskriptors

Bei der Keypointberechnung wird jedem Keypoint Lage, Skalierung und Orientierung zugewiesen. Diese Parameter werden nun verwendet um mit Hilfe des lokalen Bildausschnittes um einen Keypoint den zugehörigen Deskriptor zu berechnen. Hierbei ist das Ziel eine

Invarianz gegenüber Helligkeitsunterschieden und unterschiedlichen dreidimensionalen Ansichtswinkeln zu erreichen. Zunächst wird das Bild der enstprechenden Skala des Keypoints relativ zur Hauptorientierung ausgerichtet um Rotationsinvarianz zu erreichen. Der SIFT Deskriptor setzt sich aus mehreren Gradientenhistogrammen zusammen. Die Gradienten werden Gauß-gewichtet, sodass nahe gelegene Gradienten einen größeren Einfluss auf die Bildung des Deskriptors nehmen als weiter entfernte. Die Region um den Keypoint wird in 4×4 Kacheln eingeteilt. Für jede der 16 Regionen wird ein eigenes Gradientenhistogramm mit jeweils 8 Orientierungsbins berechnet (Abbildung 4.4). Der Merkmalsvektor eines SIFT Deskriptors setzt sich aus den konkatenierten Gradientenhistogrammen der 16 Regionen zusammen, sodass sich ein Vektor mit $4 \times 4 \times 8 = 128$ Dimensionen ergibt.

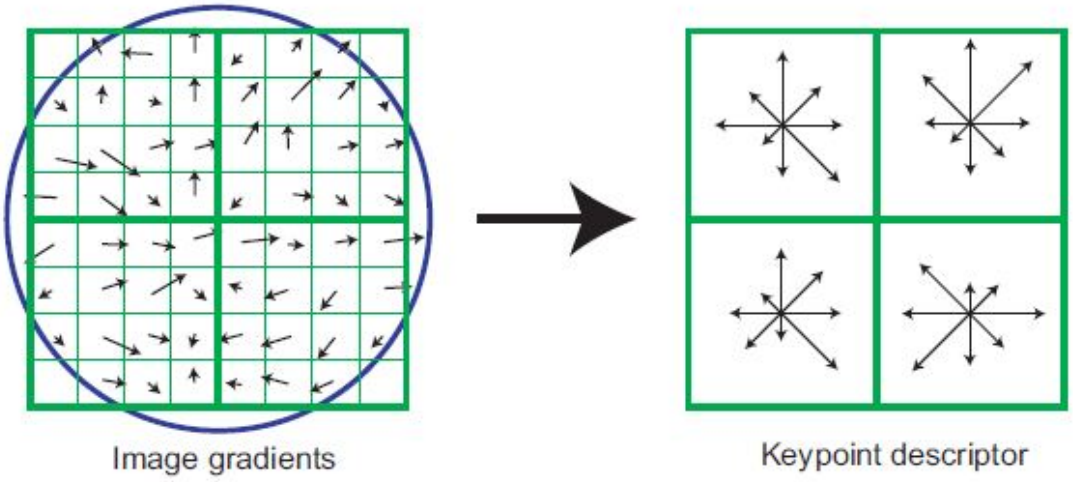


Abbildung 4.4: Berechnung des SIFT Deskriptors. Die Gradienten in einer Region um den Keypoint herum werden Gauß-gewichtet in mehrere Histogramme mit jeweils 8 Orientierungsbins eingeteilt. Der Merkmalsdeskriptor wird gebildet, indem die einzelnen Histogramme konkateniert werden. [19]

Additive Helligkeitsänderungen, bei denen ein konstanter Wert zu jedem Pixel addiert wird, haben keine Auswirkungen auf die Bildung des Deskriptors, da die Gradientenwerte durch Substraktion der Pixelwerte berechnet werden. Um zusätzlich Invarianz gegenüber multiplikativen Helligkeitsänderungen zu erreichen, wird der Vektor auf eine Einheitslänge normiert.

4.1.2 Speeded Up Robust Features

Die Veröffentlichung von Bay [22], welche die Speeded Up Robust Features (SURF) beschreibt, enthält wie bei den SIFT Features einen Algorithmus zur Detektion von Keypoints und eine Berechnungsvorschrift für Deskriptoren. Ziel von Bay war es im Vergleich zu SIFT robustere und schnellere Features zu entwickeln.

Keypoint-Detektion

Die Detektion der SURF Keypoints basiert auf der Determinante der Hesse-Matrix $\mathcal{H}(\mathrm{x}, \sigma)$ an Bildpunkt $\mathrm{x} = (x, y)$:

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

Diese enthält die zweiten partiellen Ableitungen des mit einem Gaußfilter mit der Standardabweichung σ geglätteten Bildes I an der Stelle x.

$$L_{xx}(\mathrm{x}, \sigma) = \frac{\partial^2}{\partial x^2} g(\sigma) * I(\mathrm{x})$$

Die Berechnungen von L_{xx} und L_{xy} erfolgen analog. Die Gauß-gefilterten zweiten partiellen Ableitungen werden wie in Abbildung 4.5 zu sehen mit Rechteckfiltern approximiert um die Berechnung zu beschleunigen.

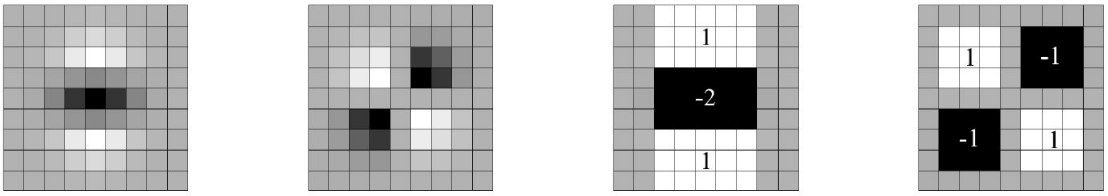


Abbildung 4.5: Links sind die zweiten partiellen Ableitungen der Gaußfunktion in y- (L_{yy}) und xy-Richtung (L_{xy}) zu sehen. Die rechten beiden Bilder zeigen die approximierten Versionen D_{yy} und D_{xy}, die mit Hilfe des Integral Images sehr schnell berechnet werden können. [22]

Rechteckfilter können effizient mit dem Konzept der Integral Images, welche von Viola und Jones [12] für die Bildverarbeitung eingeführt wurden, berechnet werden. Integral Images ermöglichen die Berechnung von Rechtecksummen unabhängig von der Skalierung in konstanter Zeit. Dabei beschreibt das Integral Images $II(x, y)$ an der Position (x, y) die Summe der Intensitäten im linken oberen Rechteck des Eingabebildes I, welches durch den Punkt (x, y) beschrieben wird:

$$II(x, y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j)$$

Für die Berechung mehrerer Rechtecksummen muss das Integral Image lediglich einmal erzeugt werden. Nachfolgend können wie in Abbildung 4.6 zu sehen beliebige Rechtecksummen mit vier Speicherzugriffen und drei Additionen berechnet werden.

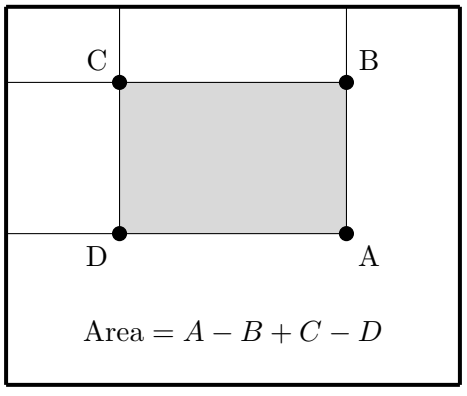


Abbildung 4.6: Mit dem Integral Image können Rechtecksummen unabhängig von der Größe in konstanter Zeit berechnet werden. In Anlehnung an [22]

Mit Hilfe der Rechteckfilter lässt sich die Determinante der approximierten Hesse-Matrix

$$\det(\mathcal{H}_{\text{approx}}) \approx D_{xx} D_{yy} - (0,9 D_{xy})^2$$

effizient berechnen. Um skalierungsinvariante Keypoints zu erhalten muss bei SURF im Gegensatz zu SIFT keine explizite Skalenraumrepräsentation des Bildes aufgebaut werden, indem dieses wiederholt Gauß-gefiltert und verkleinert wird. Anstelle dessen werden die Rechteckfilter mit zunehmender Größe auf das unveränderte Originalbild angewendet. Durch die Verwendung des Integral Image sind die Rechenkosten unabhängig von der Größe der Filter konstant. Positiver Effekt dieser Methode ist, dass beim Verkleinern der Bilder keine Aliasing-Effekte entstehen. Abbildung 4.7 visualisiert den Unterschied der Detektion von Keypoints auf unterschiedlichen Skalen bei SIFT und SURF Features.

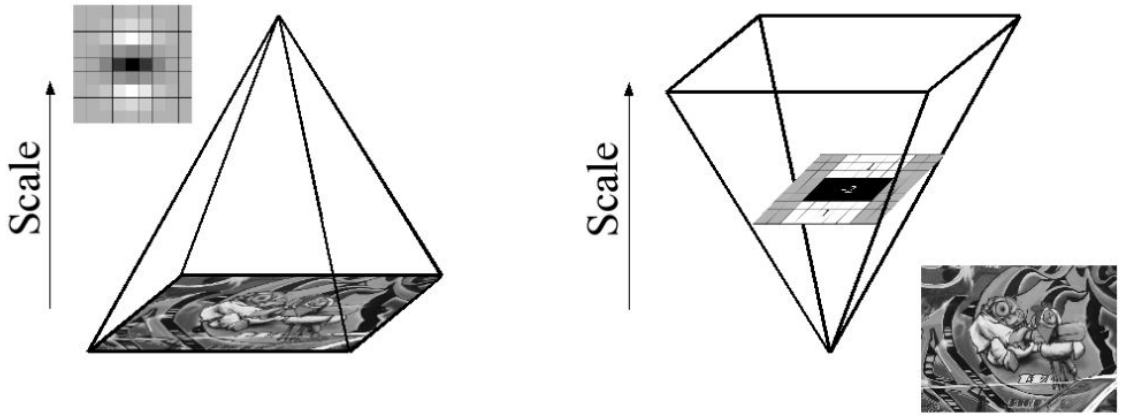


Abbildung 4.7: Bei der SIFT Keypoint-Detektion (links) wird die Skalenraumrepräsentation aufgebaut, indem das Bild mehrfach Gauß-gefiltert und skaliert wird. Die SURF Keypoint-Detektion (rechts) vergrößert anstelle dessen die verwendeten Rechteckfilter, welche auf das Bild in Originalgröße angewendet werden. [22]

Die Menge der entstehenden Keypoints kann mit Hilfe eines Schwellenwertes gesteuert werden, mit dem die Antworten des Skalenraumes verglichen werden. Mit einer Erhöhung des Schwellenwertes kann die Geschwindigkeit der Feature Berechnung verbessert werden, da hierdurch weniger Deskriptoren berechnet werden müssen. Im Skalenraum werden die Keypoints durch eine Non Maximum Supression in einer $3 \times 3 \times 3$ Nachbarschaftsumgebung detektiert. Die subpixelgenaue Position wird wie bei SIFT mit Hilfe der Taylorreihenentwicklung bestimmt.

Damit orientierungsinvariante Deskriptoren berechnet werden können, wird die reproduzierbare Hauptorientierung des Keypoints bestimmt. Hierzu werden die Antworten der Haar-Wavelet Filter in x- und y-Richtung (Abbildung 4.8) in einer kreisförmigen Nachbarschaft um den Keypoint herum berechnet. Die Skala des Keypoints s bestimmt den Radius der Nachbarschaft $6s$, die Abtastrate s und die Seitenlänge der Haar-Wavelet Filter $4s$. Zur effizienten Berechnung der Haar-Wavelet Filter kommt wiederum das Integral Image zum Einsatz.

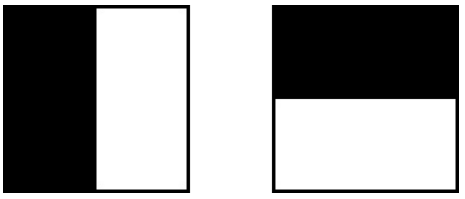


Abbildung 4.8: Haar-Wavelet Filter in x- (links) und y-Richtung (rechts) lassen sich unabhängig von der Größe mit dem Integral Image effizient berechnen. [22]

Die Antworten der Haar-Wavelet Filter werden mit einem Gaußfilter gewichtet, dessen Zentrum sich an der Position des Keypoints befindet. Anschließend werden sie in ein Diagramm übertragen, indem die Abszisse horizontale Antworten und die Ordinate vertikale Antworten repräsentieren. Die Hauptorientierung des Keypoints wird bestimmt, indem, wie in Abbildung 4.9 zu sehen, die Antworten innerhalb eines Suchfensters der Breite $\frac{\pi}{3}$ zu einem lokalen Orientierungsvektor aufsummiert werden. Der längste Vektor definiert die Hauptorientierung des Keypoints.

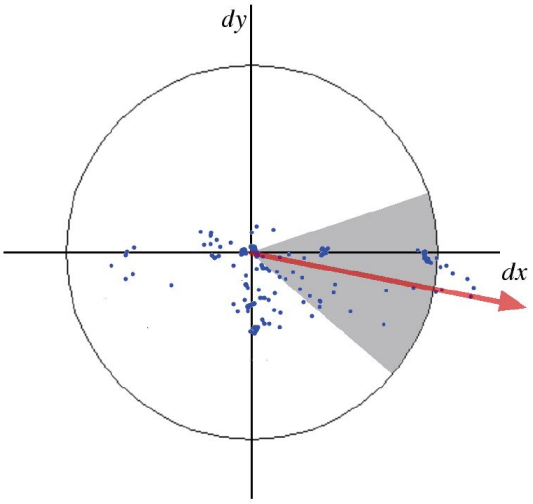


Abbildung 4.9: Die Hauptorientierung eines SURF Keypoints wird mit Hilfe eines Suchfensters in Form eines Kreisausschnittes bestimmt. [22]

Berechnung des Merkmalsdeskriptors

Für die Berechnung des Deskriptors wird eine quadratische Region mit der Größe $20s$ bestimmt, wobei s weiterhin die Skala des Keypoints repräsentiert. Diese Region wird bezüglich der Hauptorientierung des Keypoints ausgerichtet und in 4×4 gleichgroße Unterregionen unterteilt (Abbildung 4.10). In jeder dieser Flächen werden die Haar-Wavelet Filter in x- und y- Richtung mit der Größe $2s$ an 5×5 gleichverteilten Punkten ausgewertet. Die Ergebnisse werden mit Standardabweichung $\sigma = 3,3s$ Gauß-gewichtet um die Robustheit gegenüber geometrischen Verformungen und Lokalisierungsfehlern zu erhöhen. Jede Unterregion wird durch vier Summen

$$v = \left(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right)$$

repräsentiert, wobei d_x der Antwort in x-Richtung und d_y der Antwort in y-Richtung entspricht. Der finale Deskriptor setzt sich aus den konkatenierten Vektoren der einzelnen Unterregionen zusammen, sodass ein $16 \times 4 = 64$ dimensionaler Merkmalsdeskriptor entsteht. Abschließend wird der Deskriptor auf Einheitslänge normiert.

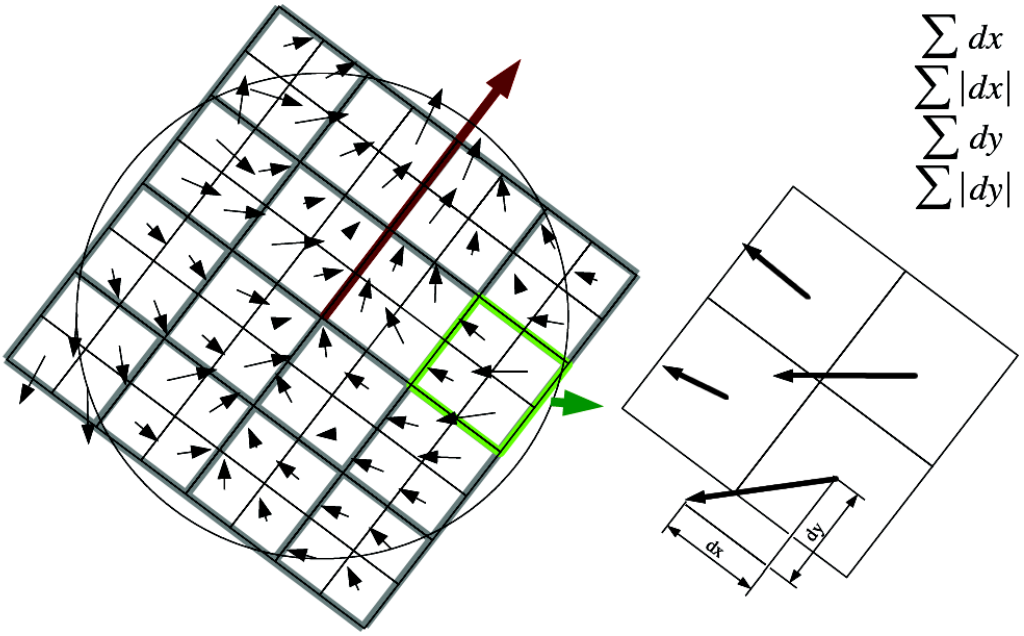


Abbildung 4.10: Für die Berechnung des SURF Deskriptors wird die Region um den Keypoint in 16 Unterregionen eingeteilt, die jeweils durch die vier Summen $\sum d_x$, $\sum |d_x|$, $\sum d_y$ und $\sum |d_y|$ beschrieben werden. [22]

4.2 K-Means Clustering für das visuelle Vokabular

Für die Berechnung des visuellen Vokabulars für das *Bag of Words* Verfahren werden die Deskriptoren der lokalen Features gruppiert. Hierzu wird die Menge der n Deskriptoren $x_1, x_2, \ldots, x_n$ in k Cluster eingeteilt. Beim k-Means basierten Clustering [61] wird die Anzahl der Cluster k a-priori vorgegeben. Im Falle des *Bag of Words* Ansatzes entspricht dies der Größe des visuellen Vokabulars. Gesucht wird die Einteilung der Deskriptoren x_i in die Cluster $S_1, S_2, \ldots, S_k$ mit den Clusterzentren $\mu_1, \mu_2, \ldots, \mu_k$, sodass die Funktion

$$J = \sum_{j=1}^{k} \sum_{x_i \in S_j} ||x_i - \mu_j||^2$$

minimiert wird. Initialisiert wird das Verfahren, indem für jedes der k Cluster ein Mittelpunkt vorgegeben wird. Anschließend wird iterativ vorgegangen, indem zunächst jeder Deskriptor x_i auf Basis des euklidischen Abstands dem nächstgelegenen Cluster S_l zugeordnet wird:

$$l = \arg\min_{j=1}^{k} ||x_i - \mu_j|| \Rightarrow x_i \in S_l$$

Anschließend wird für jedes Cluster anhand des Mittelwerts aller zugeordneten Instanzen das neue Cluster-Zentrum μ_j bestimmt:

$$\mu_j = \frac{\sum_{i=1}^{|S_j|} x_i}{|S_j|}$$

Diese zwei Schritte werden solange wiederholt, bis das Verfahren konvergiert und sich die Lage der Cluster-Zentren nicht mehr ändert.

Das k-means basierte Clustering bringt zwei Nachteile mit sich. Zum einen kann es die Anzahl der Cluster nicht selber bestimmen und zum anderen hängen die Ergebnisse von den initialisierten Mittelpunkten der Cluster ab. Je nach Lage der initialen Cluster-Zentren

endet das Verfahren in einem lokalen Minimum, sodass suboptimale Lösungen gefunden werden. Um dieses Manko zu umgehen kann das Verfahren mehrfach mit unterschiedlich Initialwerten ausgeführt werden. Man entscheidet sich dann für das Ergebnis, dessen aufsummierter Abstand der Instanzen zu ihren Clustern minimal ist.

4.3 FLANN

Die Deskriptoren der lokalen Features eines Bildes werden zur Bildung des *Bag of Words* Histogramms den ähnlichsten Einträgen des visuellen Vokabulars zugeordnet. Anstelle einer linearen Suche, welche in hochdimensionalen Räumen sehr rechenaufwendig ist, können approximative Algorithmen verwendet werden, die wesentlich schneller arbeiten und trotzdem nur einen geringen Verlust der Genauigkeit mit sich bringen. FLANN (*Fast Library for Approximate Nearest Neighbors*) [62] verwendet für die Berechnung des approximativen nächsten Nachbarn multiple zufällige Kd-Bäume und einen hierarchischen K-Means-Baum als Indexierungsmethoden:

- Multiple zufällige Kd-Bäume sind Erweiterungen normaler Kd-Bäume. Bei einem normalen Kd-Baum werden die Daten auf jedem Baumlevel anhand der Dimension, welche die größte Varianz aufweist, halbiert. Bei multiplen zufälligen Kd-Bäumen wird hingegen die Dimension, anhand derer geteilt wird, unter den fünf Dimensionen mit der größten Varianz zufällig gewählt.
- Für die Bildung eines hierarchischen K-Means-Baums werden die Datenpunkte mit dem K-Means Clustering in K Cluster unterteilt. Die Datenpunkte eines Clusters werden rekursiv weiter aufgeteilt.

Bei einer Suchanfrage für den nächsten Nachbarn werden die Datenstrukturen anhand einer Priority-Queue durchlaufen, welche die noch nicht besuchten Blätter anhand der Distanz anordnet. Der Geschwindigkeitsgewinn, der durch die Verwendung von FLANN bei der Deskriptorenzuordnung erreicht werden kann, wird bei der Evaluation der Experimente in Abschnitt 6.3 betrachtet.

4.4 Bildung des BoW-Histogramms

Für die Berechnung des *Bag of Words* Histogramms gibt es neben dem Standard Histogramm weitere Ansätze. Im Folgenden werden Methoden für die Berechnung eines fuzzy Histogramms und eines Histogramms vorgestellt, welches räumliche Informationen beinhaltet.

Fuzzy Histogramme

Bei der von Csurka *et al.* [24] vorgestellten Version des *Bag of Words* Verfahrens wird der Deskriptor eines Keypoints genau einem Bin des Histogramms zugeordnet. Das entsprechende Bin wird mit Hilfe des nächsten Nachbarn unter den Einträgen des visuellen Vokabulars bestimmt. Diese Methode bringt jedoch zwei Nachteile mit sich. Ist der Abstand eines Deskriptors zu zwei visuellen Wörtern fast identisch, erhält lediglich derjenige mit der größten Ähnlichkeit das volle Gewicht mit dem Betrag 1, während der andere leer ausgeht. Mehrdeutige Deskriptoren, die sich in der Nähe der Voronoigrenzen befinden, werden somit mangelhaft repräsentiert. Zum anderen haben alle Deskriptoren den gleichen Einfluss auf ihren nächsten Nachbarn, obwohl sich die Ähnlichkeit zum Codebucheintrag signifikant unterscheiden kann. Um diese Nachteile auszugleichen, wurden fuzzy Histogramme eingeführt. Dabei beeinflusst ein Deskriptor mehrere Bins. Der Zuwachs wird dabei in Abhängigkeit der Ähnlichkeit des Deskriptors zu den visuellen Wörtern auf mehrere Bins verteilt. Das fuzzy Histogramm wird auch als *soft-Histogramm* bezeichnet.

Die Methode von Bouachir *et al.* [29] zur Bildung eines fuzzy Histogramms basiert auf der Zugehörigkeitsfunktion des Fuzzy-C-Means Algorithmus [63]. Dabei wird der Grad der Zugehörigkeit U_{ij} eines Deskriptors $p_j, j \in \{1, \ldots, M\}$ zu einem visuellen Wort $v_i, i \in \{1, \ldots, k\}$ des Codebuchs der Größe k mit

$$U_{ij} = \frac{1}{\sum_{n=1}^{k} \left(\frac{||p_j - v_i||}{||p_j - v_n||} \right)^{\frac{2}{m-1}}}$$

beschrieben. Der Grad der Unschärfe (*engl.* fuzziness) wird durch den Parameter $m \in]1, \infty[$ bestimmt. Bouachir *et al.* schlagen den Wert $m = 1,1$ vor, den sie empirisch bestimmt haben. Nachteil dieser Methode ist, dass jeder Deskriptor Einfluss auf jedes Histogrammbin hat, sodass ein hoher Rechenaufwand entsteht. Unterscheiden sich Deskriptor und Codebucheintrag stark, ist dieser Einfluss jedoch sehr gering.

Das fuzzy Histogramm nach Jiang *et al.* [26] wird gebildet, indem ein Deskriptor die Histogrammbins der N nächsten visuellen Wörter erhöht. Bei einem Codebuch mit K Einträgen berechnet sich der Eintrag t_k des Histogramms $T = [t_1, \ldots, t_k, \ldots t_K]$ folgendermaßen:

$$t_k = \sum_{i=1}^{N} \sum_{j=1}^{M_i} \frac{1}{2^{i-1}} sim(j, k)$$

Dabei beschreibt M_i die Anzahl an Deskriptoren, deren i-ter nächster Nachbar das visuelle Wort k ist. Der Einfluss eines Deskriptors j ist dabei zum einen von der Ähnlichkeit zum visuellen Wort k abhängig, welche durch das Ähnlichkeismaß $sim(j, k)$ beschrieben wird. Weiterhin hängt er davon ab, der wievielte nächste Nachbar er ist. Der i-te Nachbar erhält die Gewichtung $\frac{1}{2^{i-1}}$. Jiang *et al.* schlagen vor, die $N = 4$ nächsten Nachbarn zu verwenden.

Räumliche Histogramme

Bei Csurka *et al.* [24] spielt die räumliche Lage der Keypoints bei der Berechnung des Histogramms keine Rolle. Es wird lediglich das Vorhandensein eines Keypoints bewertet. Dadurch gehen die Informationen verloren, die durch die geometrische Anordnung der Keypoints vorhanden sind. Die räumliche Lage kann bei der Identifikation von Objekten jedoch eine wichtige Rolle spielen.

Um die Information der räumlichen Lage der Keypoints nicht zu verlieren wird das Bild in mehrere Regionen aufgeteilt. Von jeder Region wird anschließend ein eigenes Histogramm berechnet. Die einzelnen Histogramme der unterschiedlichen Bereiche werden zum Schluss konkateniert um einen Merkmalsvektor für das gesamte Bild zu erhalten. Abbildung 4.11 zeigt mögliche Muster zur Aufteilung des Bildes. Neben rechteckigen Blöcken, bei denen das Bild in 2×2, 3×3, 4×4, 5×5 oder 6×6 Bereiche aufgeteilt wird, ist auch eine zentrumsumschließende (*engl.* center-surround) Aufteilung möglich, die entweder aus 5 (cs-5) oder aus 10 (cs-10) Teilen besteht [30].

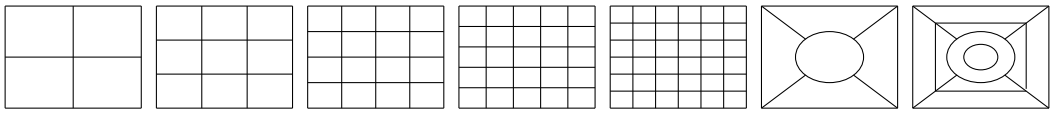


Abbildung 4.11: Muster für die Aufteilung des Bildes zur Einbeziehung von räumlichen Informationen in das *Bag of Words* Verfahren: 2×2, 3×3, 4×4, 5×5, 6×6, cs-5 und cs-6. In Anlehnung an [30]

Es ist ebenfalls möglich eine Bildrepräsentation zu verwenden, welche auf einer pyramidenförmigen Analyse des Bildes beruht [31]. Hierzu wird das Bild wiederholt in eine unterschiedliche Anzahl an Blöcken eingeteilt, sodass eine Pyramidenrepräsentation entsteht. Wie in Abbildung 4.12 zu sehen, besteht dabei Pyramidenlevel i aus 4^i Blöcken. Für jeden einzelnen Block wird ein eigenes Histogramm der auftretenden Codebucheinträge generiert. Bei der Konkatenation der Blöcke können die Histogramme unterschiedlicher Level variabel gewichtet werden.

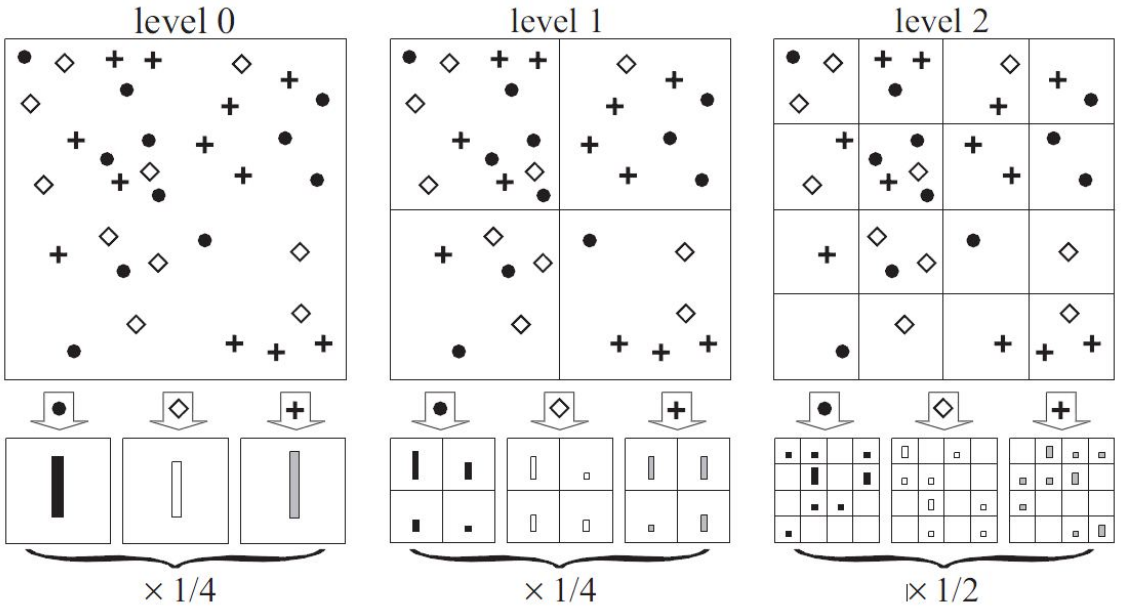


Abbildung 4.12: Räumliche Pyramidenrepräsentation mit drei Leveln. Auf jedem Level wird das Bild in eine unterschiedlich große Anzahl an Blöcken unterteilt. Jeder Block bekommt sein eigenes Histogramm. Die Histogramme werden anschließend gewichtet konkateniert. [31]

Auf Level 0 sind keine räumlichen Informationen enthalten. Blöcke höherer Level beschreiben hingegen Teilkompenenten des zu sehenden Objekts inklusive der räumlichen Beziehung untereinander. Bei der Verwendung einer Pyramidenrepräsentation muss beachtet werden, dass die Größe des Merkmalsvektors, welcher das Bild beschreibt, mit der Zahl der Level sehr schnell wächst. Bei einer Codebuchgröße von 250 visuellen Wörtern und 3 Pyramidenleveln besteht die Pyramidenrepräsentation aus $250 \cdot 4^0 + 250 \cdot 4^1 + 250 \cdot 4^2 = 5250$ Einträgen, während ein einfaches Histogramm lediglich 250 Einträge besitzt. In Abschnitt 6.3 wird sowohl die Aufteilung des Bildes in 2×2 als auch eine Pyramidenrepräsentation mit 3 Leveln evaluiert.

4.5 Klassifikatoren

Der abschließende Schritt bei der Identifikation eines Objektes mit dem *Bag of Words* Ansatz stellt die Klassifikation dar. Hierzu werden Verfahren des maschinellen Lernens verwendet, die anhand des *Bag of Words* Histogramms Hypothesen über die auf Bildern zu sehenden Objektklassen aufstellen. In dieser Arbeit werden für das überwachte Lernen (*engl.* Supervised Learning) der k-Nearest-Neighbor Klassifikator und die Support Vector Machines verwendet.

4.5.1 k-Nearest-Neighbor

Die Klassifikation mit dem k-Nearest-Neighbor Klassifikator [64] gehört zu den instanzbasierten Lernverfahren. Dabei wird beim Lernen jedes Trainingsbeispiel x_i zusammen mit seiner Klassenzugehörigkeit $f(x_i)$ abgespeichert. Für das *Bag of Words* Verfahren werden die Histogramme der Trainingsdaten mit der zugehörigen Objektklasse verwendet. Für die Klassifikation einer neuen Instanz x_q seien $x_1, x_2, \ldots, x_k$ die k nächsten Instanzen der Trainingsdatenmenge und $v_1, v_2, \ldots, v_s$ die Menge der s Objektklassen. Die Funktion

$$\hat{f}(x_q) \leftarrow \arg\max_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$$

bestimmt die Klasse der neuen Instanz $\hat{f}(x_q)$ mit Hilfe der Deltafunktion, für die $\delta(a, b) = 1$ gilt, wenn $a = b$, sonst nimmt $\delta(a, b)$ den Wert 0 an. Für die Berechnung der k nächsten Nachbarn stehen neben der L2-Norm weitere Distanzmaße, wie die Minkowski Distance, die Chi-Square Distance (χ^2) oder die Earth Mover Distance, zur Verfügung [65]. Die Evaluation dieser Arbeit beschränkt sich auf die Verwendung der L2-Norm:

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

Mit dem k-Nearest-Neighbor Klassifikator können beliebige Zielfunktionen modelliert werden. Die Generalisierungsentscheidung wird bis zu einer konkreten Klassifikationsanfrage aufgeschoben. Beim Training gehen keine Informationen der Trainingsdaten beim Trainieren verloren. Während der Rechenaufwand beim Lernen sehr gering ist, da lediglich die Trainingsbeispiele abgespeichert werden müssen, ist der Aufwand bei der Klassifikation dafür höher. Der k-Nearest-Neighbor Klassifikator gewichtet alle Einträge eines Histogramms gleich, auch wenn Einträge für die Klassifikation irrelevant sind. Weiterhin haben alle k nächsten Nachbarn gleichen Einfluss auf das Klassifikationsergebnis, auch wenn sie unterschiedliche Distanzwerte besitzen. Dieser Nachteil lässt sich mit der Verwendung eines Fuzzy-k-Nearest-Neighbor Algorithmus [66] umgehen. Hierzu wird für jede Klasse i ein Zugehörigkeitsmaß $u_i(x_q)$ auf Basis der Abstände der k nächsten Nachbarn berechnet:

$$u_i(x_q) = \frac{\sum_{j=1}^{k} u_{ij} \frac{1}{d(x_q, x_j)}}{\sum_{j=1}^{k} \frac{1}{d(x_q, x_j)}}$$

Dabei gibt u_{ij} die Zugehörigkeit des j-ten Nachbarn zur Objektklasse i an. Die Objektklasse mit dem größten Zugehörigkeitsmaß u_i stellt das Klassifikationsergebnis dar.

4.5.2 Support Vector Machine

Eine Support Vector Machine (SVM) [67] trennt zwei Klassen durch eine Trennhyperebene (Abbildung 4.13). Diese wird in den Merkmalsraum gelegt, sodass der Abstand der Stützvektoren, der Vektoren, welche sich am nächsten zur Trennhyperebene befinden, zu der Trennhyperebene maximal wird. Grundlage für den überwachten Lernvorgang, bei dem die optimale Lage dieser Trennhyperebene bestimmt wird, bildet eine Menge an beschrifteten Trainingsdaten (x_i, y_i), mit dem *Bag of Words* Histogramm $x_i \in R^n$ und der Klassenzugehörigkeit $y_i \in \{-1, +1\}$ für $i = 1, \ldots, l$ mit der Anzahl der Trainingsdaten l.

Linear trennbare Trainingsdaten

Zunächst wird der Fall betrachtet, dass die Menge der Trainingsdaten im Merkmalsraum R^n linear separierbar ist. Somit kann eine Trennhyperebene gefunden werden, die die positiven von den negativen Trainingsbeispielen trennt. Jede Hyperebene im Raum kann durch

$$w \cdot x + b = 0$$

beschrieben werden, wobei w den Normalenvektor und b die Verschiebung angibt. Die Hyperebene hat einen Abstand von $\frac{b}{||w||}$ zum Ursprung.

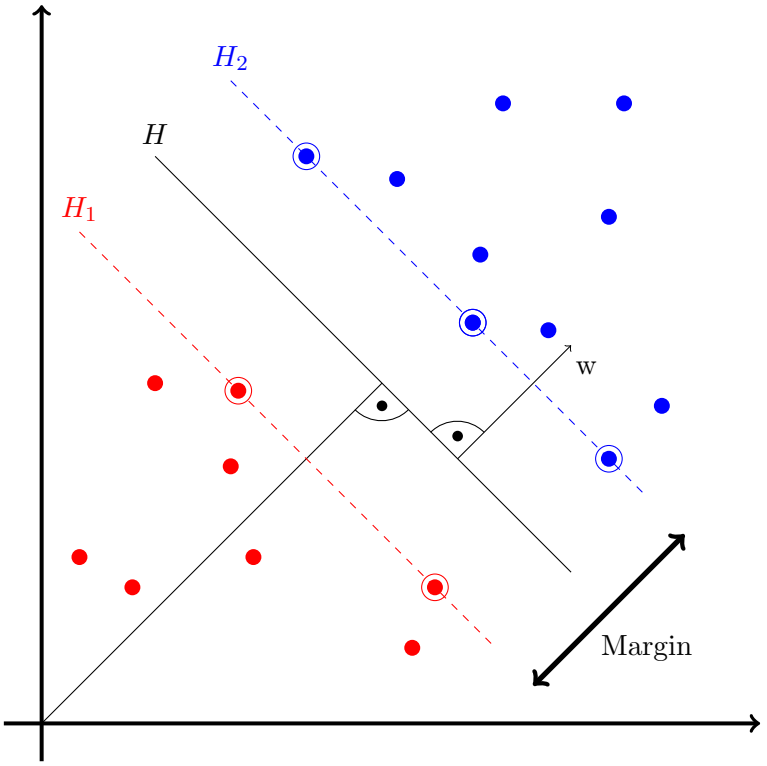


Abbildung 4.13: Die Trennhyperebene einer Support Vector Machine. In Anlehnung an [67]

Zur eindeutigen Beschreibung der Hyperebene werden w und b relativ zu den Trainingsdaten skaliert, sodass die Bedingung

$$\min_{\forall i} |w \cdot x_i + b| = 1$$

erfüllt ist. Diese Form der Hyperebene wird als kanonische Hyperebene bezeichnet. Um sicher zu stellen, dass die Trainingsdaten von der Hyperebene korrekt getrennt werden, wird die Nebenbedingung

$$y_i(x_i \cdot w + b) - 1 \geq 0 \; \forall i$$

eingeführt. Die Vektoren, für die $y_i(x_i \cdot w + b) - 1 = 0$ gilt, besitzen den geringsten Abstand zur Trennhyperebene. Diese werden als Stützvektoren (*engl.* support vectors) bezeichnet. Die Stützvektoren der positiven und negativen Klasse bilden die Hyperebenen H_1 und H_2.

$$H_1 : x_i w + b = +1$$

$$H_2 : x_i w + b = -1$$

Die Ebenen H_1, H_2 und H liegen parallel. Zwischen H_1 und H_1 befinden sich keine Trainingsvektoren. H_1 besitzt einen Abstand von $\frac{|1-b|}{||w||}$ zum Ursprung und H_2 einen Abstand

von $\frac{|-1-b|}{||w||}$. Das Margin weist somit eine Breite von $\frac{2}{||w||}$ auf, welches maximiert werden soll, um eine möglichst gute Generalisierung zu erreichen. Dies wird erreicht, indem $||w||^2$ unter Einhaltung der Nebenbedingung $y_i(w \cdot x_i + b) \geq 1 \forall i$ minimiert wird.

Minimierung mit der Lagrange-Methode

Durch die Einführung der positiven Lagrange-Multiplikatoren $\alpha_i \geq 0$ kann das Minimierungsproblem mit Hilfe der Lagrange-Funktion formuliert werden.

$$L_P = \frac{1}{2}||w||^2 - \sum_{i=1}^{l} \alpha_i y_i (x_i \cdot w + b) + \sum_{i=1}^{l} \alpha_i$$

Diese wird bezüglich w und b minimiert und bezüglich α_i maximiert. Zur Minimierung von w und b werden die Ableitungen von L_P bezüglich w und b gleich 0 gesetzt,

$$\frac{\partial}{\partial b} L_P = 0 \text{ und } \frac{\partial}{\partial w} L_P = 0$$

sodass sich die Gleichungen

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \text{ und } w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

ergeben. Diese werden zur Bildung der dualen Lagrange-Gleichung in die Lagrange-Funktion eingesetzt:

$$L_D = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j x_i \cdot x_j$$

Die duale Lagrange-Gleichung hängt nur noch von α ab und wird diesbezüglich unter Einhaltung der Nebenbedingungen $\sum_{i=1}^{m} \alpha_i y_j = 0$ und $\alpha_i \geq 0 \ \forall i$ maximiert. Für jedes Trainingsbeispiel existiert nun ein Lagrange-Multiplikator α_i. Die Trainingsbeispiele mit $\alpha_i > 0$ bilden die Stützvektoren, welche auf den Hyperebenen H_1 und H_2 und somit am nächsten zur Trennhyperebene. Die Lage der Trennhyperebene wird allein durch die Stützvektoren bestimmt. Andere Trainingsbeispiele haben keinen Einfluss. Der Normalenvektor

$$w = \sum_{i=1}^{N_s} \alpha_i y_i x_i$$

der Trennhyperebene berechnet sich über die Summe der Stützvektoren mit der Anzahl N_s.

Soft Margin Hyperebene

Für den Fall, dass die Trainingsdaten nicht linear separierbar sind, gibt es die Möglichkeit bei der Berechnung der Trennhyperebene eine geringe Zahl an Missklassifikationen zuzulassen. Hierzu wird die Randbedingung mit Hilfe der Schlupfvariablen $\xi_i \ \forall i$ aufgeweicht.

$$y_i(x_i w + b - 1) \geq 0 + \xi_i \forall i, \xi_i \geq 0$$

Die Summe der Schlupfvariablen $\sum_i \xi_i$ gibt eine obere Schranke für die Anzahl der Trainingsfehler an. Für die Berechnung einer Soft Margin Hyperebene wird

$$\frac{1}{2}||w||^2 + C \sum_{i=1}^{m} \xi_i$$

minimiert, wobei über den Parameter C gesteuert werden kann, wieviele Missklassifikationen erlaubt sind. Ein hoher Wert führt zu wenig Missklassifikationen und einem schmalen Margin, während ein kleiner Wert das Gegenteil bewirkt.

Der Kernel Trick

Im Allgemeinen ist nicht garantiert, dass die Trainingsdaten linear separierbar sind, auch nicht mit der Soft Margin Hyperebene. Somit wird eine generalisierte Methode für den nicht-linearen Fall benötigt. Die Lösung bietet eine Transformation der Daten in einen höherdimensionalen Raum $\mathcal{H}$, in dem die Daten linear separierbar sind. Diese Transformation ist durch die Abbildung ϕ realisiert.

$$\phi : R^n \rightarrow \mathcal{H}$$

Die explizite Transformation der Daten ist sehr rechenaufwendig. Die Daten treten beim Trainieren der SVM und beim Klassifizieren jedoch nur in der Form eines Skalarproduktes auf. Dieses kann durch die Kernelfunktion $K(x_i, x_j)$ ersetzt werden.

$$K(x_i, x_j) = \phi(x_i) \cdot \phi(x_j)$$

Durch die Verwendung der Kernelfunktion ist Wissen über ϕ und eine explizite Transformation der Daten nicht mehr nötig. Folgende Kernelfunktionen sind weit verbreitet:

- Skalarprodukt $K(x, y) = x \cdot y$
- Polinomial $K(x, y) = (\gamma x * y + r)^d$
- Radiale Basisfunktion (RBF) $K(x, y) = e^{-\gamma||x-y||^2}, \gamma > 0$
- Sigmoid $K(x, y) = \tanh(\gamma x * y + r)$

Die Parameter r, d und γ müssen je nach verwendetem Kernel anhand der Verteilung der Trainingsdaten angepasst werden. Alle Support Vector Machines, die in dieser Arbeit eingesetzt werden, arbeiten mit dem RBF Kernel.

Multi-Klassen Problem

Für die Trennung von mehreren Klassen reicht eine SVM nicht aus, da immer nur zwei Klassen voneinander getrennt werden können. Um mehrere Klassen voneinander zu trennen wird in dieser Arbeit das one-against-all Schema verwendet. Dabei werden für die Klassifikation von $n > 2$ Klassen n SVMs verwendet, wobei jeweils eine Klasse von allen anderen Klassen getrennt wird.

5. Implementierung

Für diese Arbeit wurden drei Anwendungen implementiert. Die erste Anwendung in Form eines Windows-Programms dient dem Training des visuellen Vokabulars und der Klassifikatoren. Es wird weiterhin für die Evaluation der Algorithmen anhand der Trainingsdaten verwendet. Das zweite Programm ist die Android App, welche die bildbasierte Objekterkennung auf mobilen Endgeräten vornimmt. Als drittes wurde eine Benchmark App für die Evaluation der Geschwindigkeiten der verwendeten Bildverarbeitungsalgorithmen implementiert. Die Android Apps wurden für Android API-Level 8 entwickelt. Der bildverarbeitende Teil wird mit dem Android NDK Revision 7 eingebunden. Grundlage für die Bildverarbeitungsalgorithmen bildet die OpenCV Bibliothek in der Version 2.4.1.

5.1 Systemaufbau

Wie in Abbildung 5.1 skizziert, gliedert sich die Implementierung in einen offline und einen online Teil. Der offline Bereich in Form des Windowsprogramms, welches in C++ implementiert wurde, verwendet zunächst die Trainingsdaten um das visuelle Vokabular zu erstellen, indem die Deskriptoren der Trainingsbilder in Cluster gruppiert werden. Mit Hilfe des visuellen Vokabulars und der Deskriptoren wird für jedes Trainingsbild eine Bildrepräsentation in Form eines *Bag of Words* Histogramms erstellt. Zusammen mit der Annotation der Trainingsdaten bilden die Histogramme die Grundlage für das überwachte Training des Klassifikators.

Die Android App repräsentiert den Online-Teil. Die Basisfunktionen der App wurden in Java geschrieben. Hierzu gehört die Ein- und Ausgabe sowie die Anbindung des Videostreams der Kamera. Die Bildverarbeitung der einzelnen Frames wurde in C++ implementiert und mit dem Android NKD an das Java Programm angebunden. Die App lädt das offline gelernte Vokabular und den Klassifikator um auf dem mobilen Endgerät Objekte erkennen zu können. Hierzu wird wiederum für die Frames des Videostreams das *Bag of Words* Histogramm berechnet. Dieses wird dem Klassifikator übergeben, sodass dieser Hypothesen über die zu sehende Objektklasse aufstellt. Die Objekterkennung wird dabei vollständig auf dem mobilen Endgerät vorgenommen. Es wird keine Client-Server Architektur verwendet. Dies ermöglicht kürzere Antwortzeiten, da kein Delay der Datenverbindung entsteht und die Benutzung der App funktioniert, auch wenn keine Datenverbindung zur Verfügung steht.

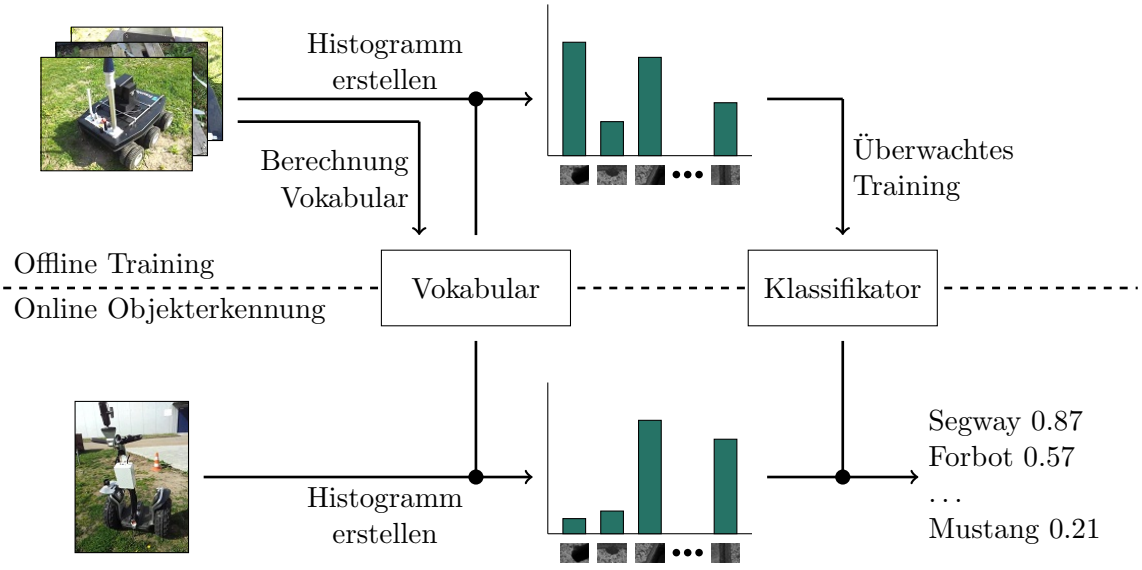


Abbildung 5.1: Der Systemaufbau der Implementierung: Der Trainingsvorgang findet offline statt. Die online Objekterkennung auf dem mobilen Endgerät nutzt das trainierte visuelle Vokabular und den trainierten Klassifikator.

5.2 Bildverarbeitung

Aufgrund der geringen Rechenleistung der Prozessoren der mobilen Engeräte ist es nicht möglich jedes Frame des Videostreams auszuwerten. Aus diesem Grund wird die Verarbeitung des aktuellen Frames nur dann angestoßen, wenn die letzte Auswertung beendet wurde. Alle zwischenliegenden Frames werden verworfen. Eine genaue Betrachtung der Rechenzeiten findet in Abschnitt 6.3 statt. Damit der Benutzer während der Auswertung eines Frames auf mobilen Endgeräten mit mehr als einem Prozessorkern keinen statischen Bildschirm sieht, wird für die Bildverarbeitung ein seperater Thread gestartet, sodass der Videostream weiterhin flüssig angezeigt werden kann. Die Verarbeitungs-Pipeline eines einzelnen Frames wird in Algorithmus 1 beschrieben. Zunächst wird das Bild mit Hilfe von bilinearer Interpolation verkleinert um die Berechnung der SIFT und SURF Features zu beschleunigen. Weiterhin wird ein Geschwindigkeitsvorteil erzielt, indem die Zahl der Keypoints mit Hilfe eines Schwellenwertes reduziert wird (Vergleich Abschnitte 4.1.1 und 4.1.2). Die Auswirkungen der verschiedenen Bildauflösungen (640×480, 480×360, 320×240 und 240×180) und der unterschiedlichen Schwellenwerte auf die Klassifikationsergebnisse und Rechenzeiten werden in Abschnitt 6.3 betrachtet.

Algorithm 1 GETBAGOFKEYPOINTS(img)

1: $img \leftarrow$ RESIZEIMAGE(img)
2: $keypoints \leftarrow$ GETKEYPOINTS(img)
3: $descriptors \leftarrow$ GETDESCRIPTORS($img, keypoints$)
4: $histogram \leftarrow$ CREATEHISTOGRAM($descriptors, vocabulary$)
5: **return** $histogram$

Für die Berechnung des *Bag of Words* Histogramms wurden unterschiedliche Verfahren implementiert. Neben dem Standard Histogramm aus der Veröffentlichung von Csurka *et al.* [24] (Algorithmus 2) wurden ein fuzzy Histogramm und räumliche Histogramme für die Evaluation umgesetzt. Die Implementierung des fuzzy Histogramms (Algorithmus 3) orientiert sich an der Methode von Jiang *et al.* [26]. Für die räumlichen Histogramme wurde neben der Aufteilung des Bildes in 2×2 Regionen, welche in Algorithmus 4

beschrieben wird, auch eine Pyramidenrepräsentation mit 3 Leveln evaluiert.

Algorithm 2 CREATESTANDARDHISTOGRAM($descriptors, vocabulary$)

```
matches ← NEARESTNEIGHBOR(descriptors, vocabulary)
for i = 1 → matches.size do
    bin ← matches[i]
    histogram[bin] ← histogram[bin] + 1
end for
histogram ← NORMALIZE(histogram)
return histogram
```

Algorithm 3 CREATEFUZZYHISTOGRAM($descriptors, vocabulary$)

```
matchesknn ← KNEARESTNEIGHBOR(descriptors, vocabulary, kneighbors)
for i = 1 → matchesknn.size do
    for rank = 1 → kneighbors do
        score ← (1 − matchesknn[i][rank].distance)/POW(2, rank)
        bin ← matchesknn[i][rank]
        histogram[bin] ← histogram[bin] + score
    end for
end for
histogram ← NORMALIZE(histogram)
return histogram
```

Algorithm 4 CREATESPATIALHISTOGRAM($keypoints, descriptors, vocabulary$)

```
matches ← NEARESTNEIGHBOR(descriptors, vocabulary)
for i = 1 → matches.size do
    region ← 0
    if keypoints[i].x > img.width/2 then
        region ← region + 1;
    end if
    if keypoints[i].y > img.height/2 then
        region ← region + 2;
    end if
    bin ← matches[i] + region ∗ vocabulary.size
    histogram[bin] ← histogram[bin] + 1
end for
histogram ← NORMALIZE(histogram)
return histogram
```

Die *Bag of Words* Histogramme der Trainingsdaten bilden die Grundlage für das überwachte Training der Klassifikatoren. Für den k-Nearest-Neighbor Klassifikator werden die Vektoren zusammen mit ihrer Klassenzugehörigkeit gespeichert. Für die Klassifikation mit Support Vector Machines wird für jede Objektklasse eine eigene SVM mit RBF-Kernel trainiert (one-against-all Schema). Für das Training der SVMs werden die annotierten *Bag of Words* Histogramme der Trainingsdaten und der automatische Trainingsmodus der SVM Implementierung von OpenCV verwendet. Dieser bestimmt die Lage der Trennhyperebene, den RBF-Kernel-Parameter γ und für die Soft Margin Hyperebene den Parameter C (Vergleich Abschnitt 4.5.2). Dabei wird für jeden Parameter über ein logarithmisches Werteraster iteriert und anhand des minimalen Cross-Validation Testmengenfehlers die

optimale Parametrisierung gewählt.

Bei der Klassifikation auf dem mobilen Endgerät wird zunächst für jede Objektklasse ein Score berechnet, der angibt, wie wahrscheinlich es ist, dass eine Instanz der Objektklasse auf dem Bild zu sehen ist. Die Anwendung zeigt dem Nutzer lediglich die Objektklassen an, deren Scores oberhalb eines Schwellenwertes liegen. Mit Hilfe dieses Schwellenwertes soll erreicht werden, dass für Bilder, auf denen kein eingelerntes Objekt zu sehen ist, keine Objektklasse erkannt wird. Ein Schwellenwert von 0 beschreibt bei der Verwendung von Support Vector Machines die Trennung der Daten anhand der Hyperebene. Den Schwellenwert kann der Nutzer in der Android App an seine Bedürfnisse anpassen. Eine Verringerung des Schwellenwertes führt dazu, dass der Algorithmus zusätzlich weniger wahrscheinliche Objektklassen zurückgibt. Die Beschränkung auf besonders wahrscheinliche Ojektklassen wird durch eine Erhöhung des Schwellenwertes erreicht.

5.3 User Interface der Android App

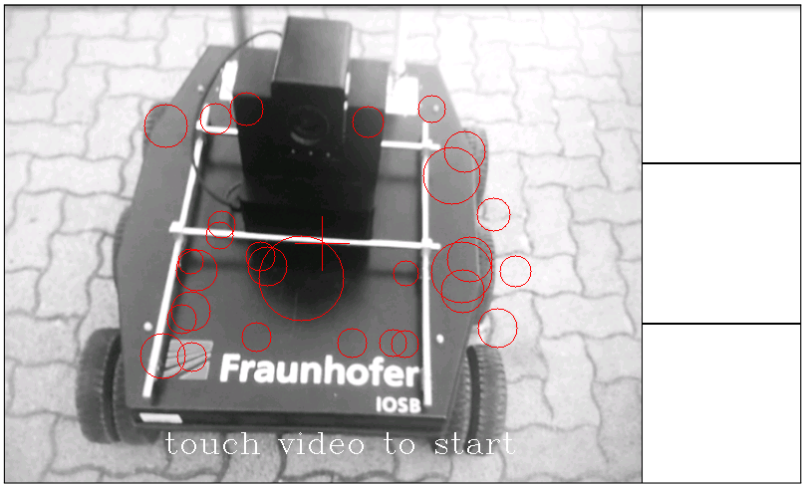
Ziel bei der Gestaltung des User Interfaces war es, die Bedienung möglichst intuitiv zu gestalten. Dies ist bei einem Assistenzsystem wie diesem von besonderer Bedeutung. Personen, die kein Vorwissen besitzen, sollen das Programm ohne Hindernisse nutzen können. Abbildung 5.2 zeigt einen Anwendungsfall, der verdeutlicht, wie die App als Assistenzsystem verwendet werden kann:

1. Zunächst wird dem Benutzer der Videostream der Kamera angezeigt. Damit der Nutzer weiß, wo er das Objekt im Bild positionieren soll, enthält das Bild ein Fadenkreuz in zentraler Position. Zusätzlich wird in einer rechteckigen Region um das Fadenkreuz herum eine kleine Menge an Keypoints angezeigt, damit der Nutzer Feedback für die korrekte Platzierung des Objektes erhält. Diese werden, wie in Abbildung 5.2a durch rote Kreise visualisiert. Die Berechnung dieser Keypoints findet auf einem stark verkleinerten Bild statt, damit für die Berechnung möglichst wenig Zeit benötigt wird.
2. Der Anwender positioniert die Kamera so, dass das Fadenkreuz auf das Objekt zeigt und die Keypoints auf dem Objekt liegen. In Abbildung 5.2b richtet der Benutzer die Kamera auf den Forbot aus.
3. Durch Berühren des Videostreams startet die Auswertung der Frames des Videostreams für die bildbasierte Objekterkennung. Dieser Zustand wird dem Nutzer durch einen roten Rand des Videostreams angezeigt (Abbildung 5.2c). Für die Bildverarbeitung wird ein separater Thread gestartet, damit der Videostream auf Geräten mit mehr als einem Prozessorkern weiter ohne Delay angezeigt werden kann.
4. Die Bildverarbeitung stellt Hypothesen über die auf dem Bild zu sehenden Instanzen der eingelernten Objektklassen auf. Die wahrscheinlichsten Objektklassen werden durch ihre Namen und Bilder am rechten Rand des User Interfaces angezeigt. In Abbildung 5.2d wird der Forbot erkannt und durch ein Icon am rechten Rand dargestellt. Es ist möglich, dass bis zu drei Objektklassen visualisiert werden, wenn ihr Score oberhalb eines Schwellenwertes liegt (Vergleich Abschnitt 5.2). Die Darstellung der Ergebnisse der bildbasierten Objekterkennung durch Bilder hat im Gegensatz zu Text den Vorteil, dass ein Benutzer, der nicht weiß, wie eine Komponente heißt, anhand des Bildes erkennen kann, ob das Klassifikationsergebnis dem real existierenden Objekt entspricht. Um die Ergebnisse robuster zu gestalten werden zur Auswertung die Scores der jeweils letzten drei Frames ausgewertet.
5. Eine Berührung des Videos stoppt die Auswertung (5.2e). Der rote Kasten um den Videostream verschwindet. Die Auswertung kann direkt nach dem ersten Frame gestoppt werden, wenn bereits die richtige Objektklasse erkannt wurde. Alternativ kann

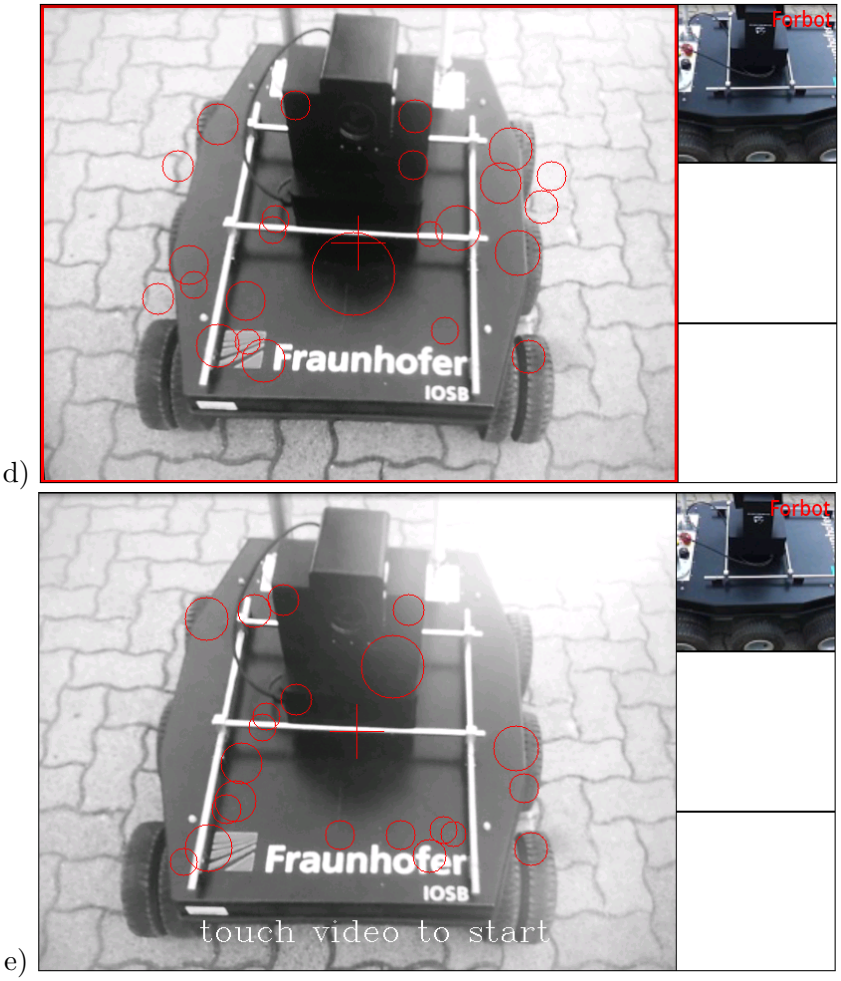
sie auch zu einem späteren Zeitpunkt beendet werden, sodass der Anwender den Ansichtswinkel der Kamera ändern kann, wenn die Auswertung der ersten Frames nicht die korrekte Objektklasse als Ergebnis liefert. Der Nutzer kann Assistenzinformationen zu einer erkannten Objektklasse anzeigen lassen, indem er auf das entsprechende Icon tippt.

a)

b)



c)



d)

e)

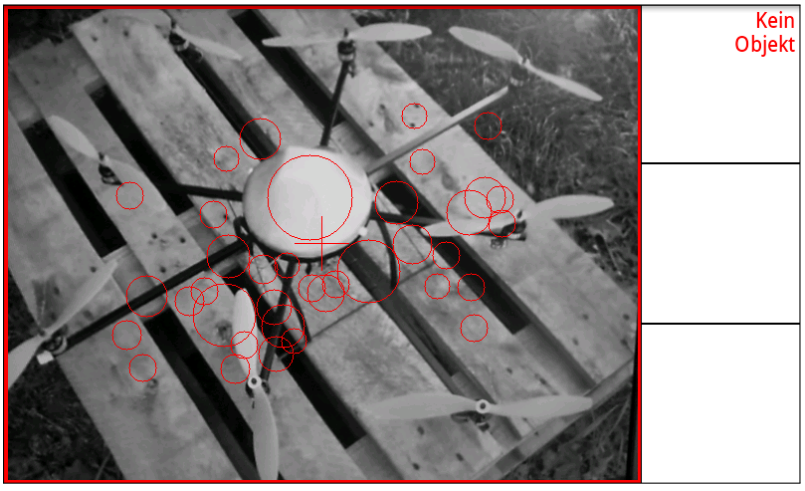
Abbildung 5.2: Anwendungsfall 1: Das User Interface bei der Erkennung des Forbots. a) Die Kamera ist auf kein Objekt ausgerichtet. b) Die Kamera wird auf den Forbot ausgerichtet. c) Die Objekterkennung wird gestartet. d) Die App hat einen Forbot im Bild erkannt und stellt das Ergebnis durch ein Bild auf der rechten Seite dar. e) Der Benutzer beendet die Objekterkennung.

Abbildung 5.3 zeigt den Fall, dass die bildbasierte Objekterkennung zunächst kein Objekt erkennt, obwohl die Kamera auf eine Instanz einer eingelernten Objektklasse ausgerichtet ist. Damit das Objekt trotzdem erkannt wird, kann der Benutzer weniger wahrscheinliche Objekte anzeigen lassen:

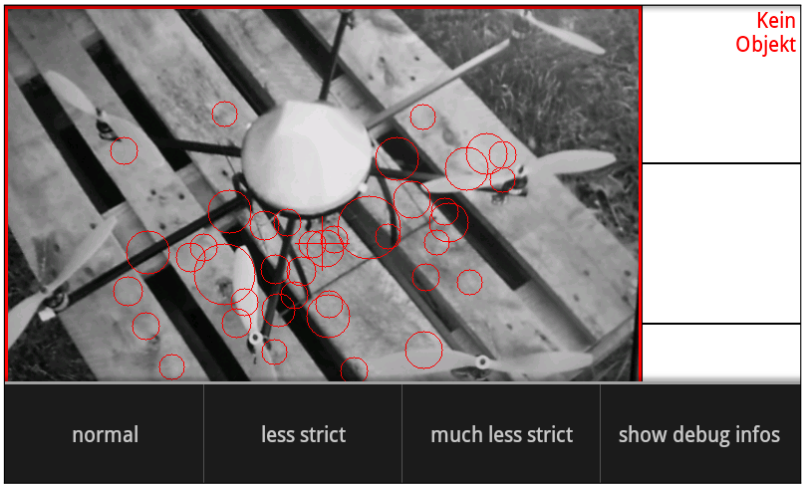
1. In Abbildung 5.3a wird der Quadrocopter nicht erkannt, obwohl er im Sichtfeld der Kamera platziert wurde.
2. Abbildung 5.3b zeigt die Menüoptionen *normal*, *less strict* und *much less strict*, mit denen der Schwellenwert, mit dem die Scores der Hypothesen verglichen werden, angepasst werden können (Vergleich Abschnitt 5.2). Die Option *normal* entspricht einem Schwellenwert von 0, *less strict* einem Wert von -3 und *much less strict* einem Wert von -6. Umso kleiner der Wert ist umso weniger wahrscheinliche Objektklassen werden als Ergebnis angezeigt. Die Menüoption *show debug infos* führt zu einer Anzeige von detaillierten Informationen, welche für die Verwendung der App jedoch nicht relevant sind.
3. Der Benutzer hat die Option *less strict* gewählt, woraufhin der Quadrocopter erkannt wird. Die Wahl der Option *less strict* wird durch ein Plus im rechten oberen Eck des Videostreams visualisiert (Abbildung 5.3c).

4. Die Auswahl der Option *much less strict* führt dazu, dass weitere Objektklassen erkannt werden. In Abbildung 5.3d wird neben dem Quadrocopter noch die Leiste des Forbots und die CPU des Quadrocopters erkannt. Durch die Auswahl der Option *much less strict* erscheinen zwei Plus im rechten oberen Eck des Videostreams.

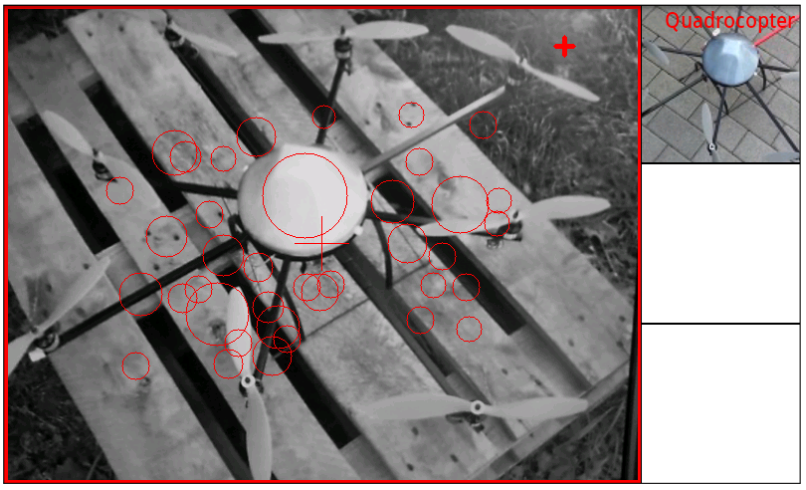
a)



b)



c)



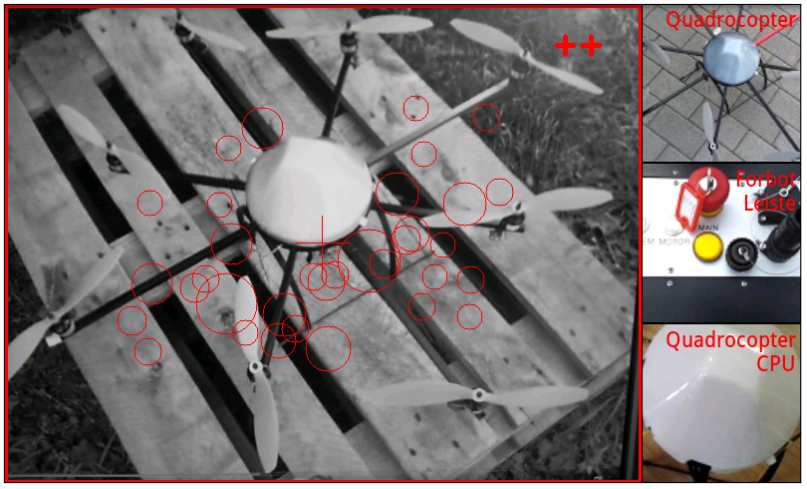
d) 

Abbildung 5.3: Anwendungsfall 2: Das User Interface für die Anzeige von weniger wahrscheinlichen Objekten. a) Der Quadrocopter wird mit der Standard Einstellung nicht erkannt. b) Menüauswahl für die Einstellungen *normal*, *less strict* und *much less strict*. c) Der Quadrocopter wird mit der Einstellung *less strict* erkannt. d) Mit der Einstellung *much less strict* werden zusätzlich weitere Objekte erkannt.

5.4 Benchmark App

Für die Bewertung der Rechenzeiten, die für die einzelnen Schritte des *Bag of Words* Verfahrens benötigt werden, wurde eine Benchmark App implementiert. Anstelle von Bildern des Videostreams der Kamera eines mobilen Endgeräts werden die Algorithmen anhand von fünf Beispielbildern (Abbildung 5.4) ausgewertet, die unterschiedliche Komponenten von *AMFIS* zeigen. Diese Bilder werden den Bildverarbeitungsalgorithmen als Eingabe zugeführt. Der Einfluss folgender Parameter für das *Bag of Words* Verfahren werden von der Benchmark App zeitlich erfasst:

- Bildauflösung mit 640×480, 480×360, 320×240 und 240×180 Pixeln
- SIFT mit Keypoint-Schwellenwert 0.01, 0.03, 0.05, 0.07 und 0.09
- SURF mit Keypoint-Schwellenwert 400, 600, 800, 1000 und 1200
- Histogramme mit 50, 100, 200, 300, 400, 500, 600, 700, 800, 900 und 1000 Bins
- Fuzzy Histogramme
- Räumliche Histogramme mit 2×2 Regionen und einer Pyramidenrepräsentation mit 3 Leveln
- Klassifikation mit Support Vector Machines und dem k-Nearest-Neighbor Klassifikator mit 1000, 2000 und 4000 gespeicherten BoW-Histogrammen

Für die Ergebnisse in Abschnitt 6.3 werden die durchschnittlichen Rechenzeiten verwendet, die für die Bearbeitung eines Bildes benötigt werden.

Abbildung 5.4: Die fünf Bilder, die von der Benchmark App verwendet werden, um die Rechenzeiten der Algorithmen zu messen.

6. Evaluation durch Experimente

Die in Kapitel 4 vorgestellten Algorithmen für die bildbasierte Objekterkennung wurden einer Reihe von Experimenten unterzogen. Ziel der Evaluation war es, Algorithmen und Parameter zu bestimmen, die zum einen möglichst gute Klassifikationsergebnisse und zum anderen geringe Rechenzeiten auf mobilen Endgeräten ermöglichen. Zur Auswertung der Klassifikationsgenauigkeit wurde die Trainingssoftware um eine Evaluationsmöglichkeit ergänzt. Zur Betrachtung der Geschwindigkeit der Algorithmen wurden die Rechenzeiten der Benchmark App ausgewertet, welche für diese Zwecke implementiert wurde. Im Folgenden werden die verwendeten Trainingsdaten, Evaluationsmethoden und die Ergebnisse vorgestellt.

6.1 Güte der Trainings- und Testdaten

Alle Trainingsdaten wurden mit Kameras mobiler Endgeräte aufgenommen, damit die Bilder die charakteristischen Eigenschaften enthalten, die diese Sensoren mit sich bringen. Digitalkameras, die nicht in mobilen Endgeräten verbaut sind, erzeugen zwar qualitativ hochwertigere Bilder, führen jedoch zu Unterschieden zwischen den Daten, die für das Training und die Evaluation verwendet werden und den Daten, die bei der Verwendung der App dem Objekterkennungsalgorithmus zugeführt werden. Von den einzelnen Objekten wurden Videos aufgenommen, die anschließend in Einzelframes zerlegt wurden. Aufgrund der Tatsache, dass sich benachbarte Frames eines Videos nur minimal unterscheiden, wurde für die Evaluation lediglich jedes dreißigste Frame in den Datensatz aufgenommen. Um möglichst realitätsnahe Trainingsbilder zu erhalten wurden die Objekte bei unterschiedlichen Lichtverhältnissen aufgenommen. Hierzu gehörte unter anderem Kunstlicht, Sonneneinstrahlung und Schatten. Weiterhin wurden die mobilen Geräte vor unterschiedlichen Hintergründen abgelichtet. Dabei wurde Wert darauf gelegt, dass die Objekte sowohl vor homogenen als auch komplexen Hintergründen positioniert wurden. Abschließend wurde der Datensatz durch eine Menge an Bildern ergänzt, die keine Komponenten des *AMFIS*-Systems enthalten. Für die Evaluation werden insgesamt 2140 Bilder verwendet, die sich auf 18 Objektklassen verteilen. Eine Objektklasse beschreibt dabei entweder eine komplette Komponente des *AMFIS*-Systems oder einen Teilbereich einer Komponente. Abbildung 6.1 zeigt die Verteilung der Bilder des Trainings- und Testdatensatzes auf die einzelnen Objektklassen.

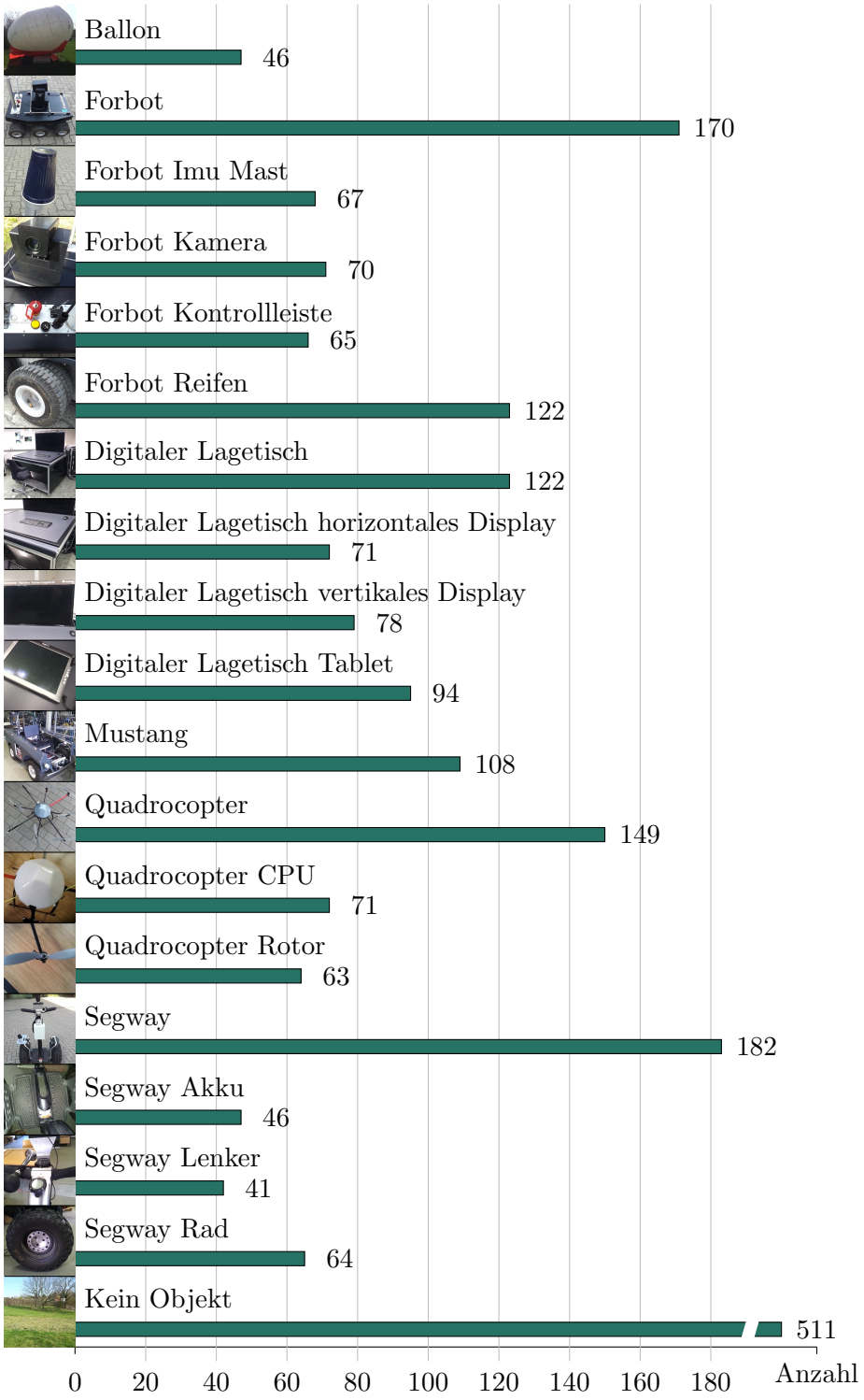


Abbildung 6.1: Verteilung der Bilder des Datensatzes auf die einzelnen Objektklassen.

6.2 Evaluationsmethoden

Damit die verschiedenen Algorithmen und Parameter verglichen werden können, um eine optimale Parametrisierung wählen zu können, wird ein Performancemaß benötig. In dieser Arbeit wird die Evaluation der Klassifikation mehrerer Klassen realisiert, indem jeder binäre Klassifikatior, bei dem jeweils eine Objektklasse vom Rest getrennt wird, zunächst einzeln evaluiert wird. Dies hat den Vorteil, dass jede Objektklasse gleichen Einfluss auf die

Gesamtperformance hat, unabhängig von der Anzahl der Testbilder der einzelnen Klassen. Die Ergebnisse eines binären Klassifikators werden mit der Annotation der Trainingsdaten (*engl.* Groundtruth) verglichen, sodass sich einer der vier Fälle aus Tabelle 6.1 ergibt:

	Ergebnis positiv	Ergebnis negativ
Positives Beispiel	*True positive* (TP)	*False negative* (FN)
Negatives Beispiel	*False positive* (FP)	*True negative* (FN)

Tabelle 6.1: Einordnung der Ergebnisse eines binären Klassifikators. *True* und *False* sagen aus, ob das Klassifikationsergebnis korrekt ist, während *Positive* und *Negative* das Ergebnis des Klassifikators beschreiben.

Die einfachste Methode, einen Klassifikator zu bewerten, ist die Klassifikationsgenauigkeit, welche den Anteil an korrekt klassifizierten Beispielen angibt. Diese Methode wird mittlerweile jedoch kaum mehr verwendet. Eine differenziertere Bewertung der Performance ist durch die True Positive und die False Positive Rate möglich:

$$\text{True Positive Rate (TPR)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{False Positive Rate (FPR)} = \frac{\text{FN}}{\text{TN} + \text{FP}}$$

Jeder binäre Klassifikator liefert für ein Bild einen Score, welcher die Sicherheit angibt, dass auf dem Bild eine Instanz der entsprechenden Objektklasse zu sehen ist. Diese können mit unterschiedlichen Schwellenwerten verglichen werden um eine Receiver Operating Characteristics (ROC) Kurve zu erzeugen [68]. Hierbei handelt es sich um ein Schaubild, bei dem die True Positive Rate auf die False Positive Rate aufgetragen wird (Abbildung 6.2 a). Die ROC-Kurve eines idealen Klassifikators führt vom Punkt (FPR = 0,TPR = 0) über den linken oberen Punkt (FPR = 0,TPR = 1) zum Punkt (FPR = 1,TPR = 1).

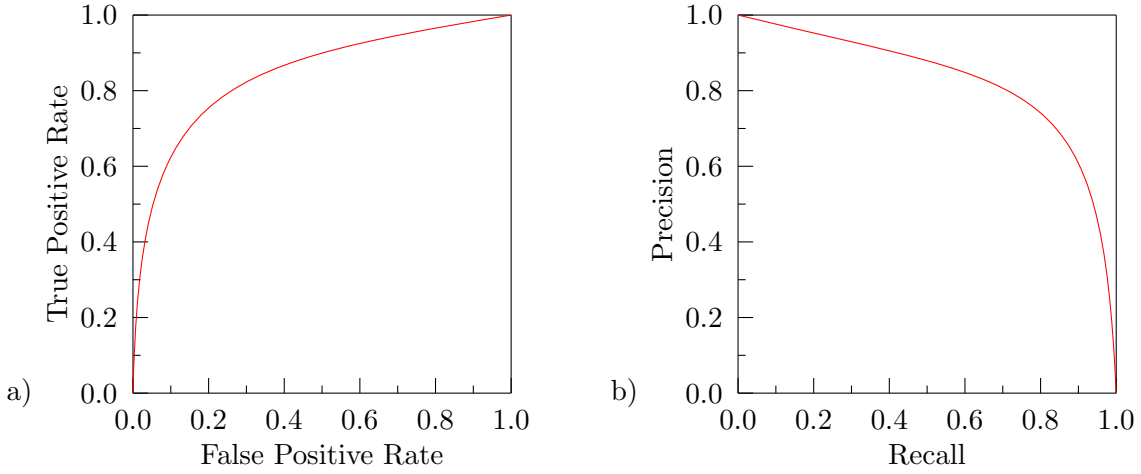


Abbildung 6.2: Beispiele für a) eine Receiver Operating Characteristics und b) eine Precision-Recall Kurve.

Precision-Recall Kurve

Eine Alternative zur Receiver Operating Characteristics stellt die Precision-Recall Kurve dar. Der Recall, welcher auch als Sensitivity bezeichnet wird und der True Positive Rate entspricht, gibt den Anteil der True Positives an allen positiven Instanzen des Datensatzes

an. Die Precision hingegen gibt den Anteil der True Positives an allen positiven Klassifikationsergebnissen an. Zur Bildung der Precision-Recall Kurve wird die Precision $p(r)$ als Funktion des Recalls r gezeichnet (Abbildung 6.2 b).

Ein idealer Klassifikator liefert sowohl für den Recall als auch für die Precision den Maximalwert 1. Dies spiegelt sich in der Precision-Recall Kurve durch eine horizontale Linie auf Höhe des Precision Wertes 1 wieder. Im Allgemeinen sind Precision und Recall jedoch zueinander gegensätzlich, sodass dieser Fall nicht eintritt. Wenn ein System darauf trainiert wird, möglichst alle positiven Beispiele korrekt als positiv zu klassifizieren, sodass ein hoher Recall ensteht, werden meist auch negative Beispiele ebenfalls positiv klassifiziert, was zu einer Verringerung des Precision Wertes führt. Auf der anderen Seite verringert sich der Recall, wenn das System darauf trainiert wird negative Beispiele auf keinen Fall als positiv zu klassifizieren. Die Form der Kurve wird durch das Integral von $r = 0$ bis $r = 1$ zur Average Precision (AP)

$$AP = \int_0^1 p(r)dr$$

zu einem numerischen Wert zusammengefasst [69]. In der Praxis wird das Integral durch eine finite Summe ersetzt, bei der $\Delta r(i)$ den Abstand zwischen den Recall Leveln $i-1$ und i beschreibt.

$$AP = \sum_i^n p_{\text{interp}}(i)\Delta r(i)$$

Der interpolierte Precision Wert $p_{\text{interp}}(i)$ ergibt sich aus dem Maximum der Precision Werte, deren Recall Level größer als i ist [4].

$$p_{\text{interp}}(i) = \max_{\bar{i}:\bar{i}\geq i} p(\bar{i})$$

Eine hohe Average Precision erhält man nur dann, wenn ein Verfahren auf allen Recall Leveln hohe Precision Werte besitzt. Für die Gesamtperformance eines Systems mit mehreren binären Klassifikatoren wird über alle m Klassifikatoren gemittelt. Hierdurch entsteht die Mean Average Precision (MAP)

$$\text{MAP} = \frac{\sum_{i=1}^m AP(i)}{m}$$

welche in dieser Arbeit zum Vergleich der verschiedenen Algorithmen und Parameter eingesetzt wird.

Confusion Matrix

Eine Confusion Matrix visualisiert die Ergebnisse eines überwachten Lernvorgangs. Jede Zeile der Matrix enthält die Instanzen der tatsächlichen Klasse, während jede Spalte die vorhergesagten Ergebnisse enthält. Somit ist es möglich abzulesen, welche Klassen vom Klassifikator häufig verwechselt werden. Die Confusion Matrix eines perfekten Klassifikationsergebnisses enthält lediglich Einträge auf der Diagonalen.

Cross Validation

Bei der Evaluation muss die verwendete Datenmenge sinnvoll in Trainings- und Testdaten unterteilt werden. Zum einen werden Trainingsdaten benötigt, die das Modell trainieren und zum anderen werden Testdaten benötigt um das trainierte Modell zu evaluieren. Um das Evaluationsergebnis möglichst unabhängig von den ausgewählten Trainings- und Testdaten zu gestalten eignet sich das statistische Modell der Cross-Validation. Die am weitesten verbreitete und in dieser Arbeit eingesetzte Variante der Cross-Validation ist die k-fold

Cross-Validation. Hierbei wird die Datenmenge in k möglichst gleich große Teilmengen unterteilt. In jedem Durchgang sind $k-1$ Teilmengen die Trainingsdatenmenge, während die verbleibende Teilmenge die Testmenge darstellt. Es werden k Durchgänge ausgeführt, in denen jeweils der Klassifikator trainiert und anschließend getestet wird. Dies garantiert, dass jede der k Teilmengen einmal die Testmenge ist. Jeder Durchgang wird einzeln evaluiert. Die Gesamtperformance ergibt sich aus dem Mittelwert aller k Einzelwertungen. Für die Evaluation der Algorithmen wird in Abschnitt 6.3 eine 5-fold Cross-Validation mit der *Mean Average Precision* als Performancemaß verwendet.

6.3 Evaluationsergebnisse

Die Wahl der einzelnen Algorithmen und ihrer Parametrisierung hat Einfluss sowohl auf die Ergebnisse der Klassifikation als auch auf die Zeit, die für die Verarbeitung eines Frames benötigt wird. Im Folgenden werden Algorithmen und Parameter evaluiert und ausgewählt um ein *Bag of Words* Verfahren mit kurzer Rechenzeit und einer hohen Mean Average Precision auf dem Trainings- und Testdatensatz zu erhalten. Die Rechenzeiten basieren auf Messungen der Benchmark App auf einem Samsung Galaxy S2 GT-I9100 Smartphone mit 1,2 Ghz Dual-Core CPU und Android 2.3.6.

Lokale Features

Die Berechnung von lokalen Features auf mobilen Endgeräten benötigt sehr viel Zeit. Diese hängt neben der Wahl des Features von der Größe des Bildes und des verwendeten Schwellenwertes für die Keypoints ab. Auf einem Bild mit höherer Auflösung werden mehr Keypoints detektiert als auf dem gleichen Bild mit geringerer Auflösung. Wenn durch die Keypoint-Detektion mehr Keypoints gefunden werden, bedeutet dies eine Erhöhung der Rechenzeit, da für jeden Keypoint ein zugehöriger Deskriptor erzeugt werden muss. Neben der Reduktion der Bildauflösung kann die Zahl der Keypoints durch die Anpassung eines Schwellenwertes (Vergleich Abschnitte 4.1.1 und 4.1.2) verringert werden. Dadurch werden weniger, dafür aber eindeutigere Keypoints detektiert. Bei kontrastarmen Lichtverhältnissen kann dies jedoch dazu führen, dass nur noch wenige Keypoints im Bild gefunden werden.

Abbildung 6.3 (SIFT) und Abbildung 6.4 (SURF) zeigen die Auswirkungen der unterschiedlichen Bildauflösungen (640 × 480, 480 × 360, 320 × 240 und 240 × 180) und der Veränderung des Schwellenwertes auf die Anzahl der Keypoints. Für die SIFT Keypoint-Detektion werden die Schwellenwerte 0.01, 0.03, 0.05, 0.07 und 0.09 und für die SURF Keypoint-Detektion die Schwellenwerte 400, 600, 800, 1000 und 1200 betrachtet. Die Anzahl der Keypoints bezieht sich auf den durchschnittlichen Wert der fünf Bilder der Benchmark App.

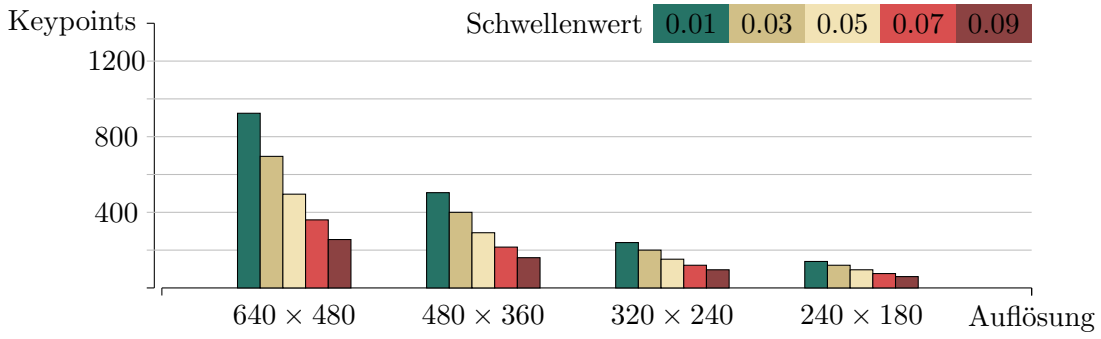


Abbildung 6.3: Durchschnittliche Anzahl an SIFT Keypoints auf den fünf Bildern der Benchmark App für unterschiedliche Bildauflösungen und Schwellenwerte.

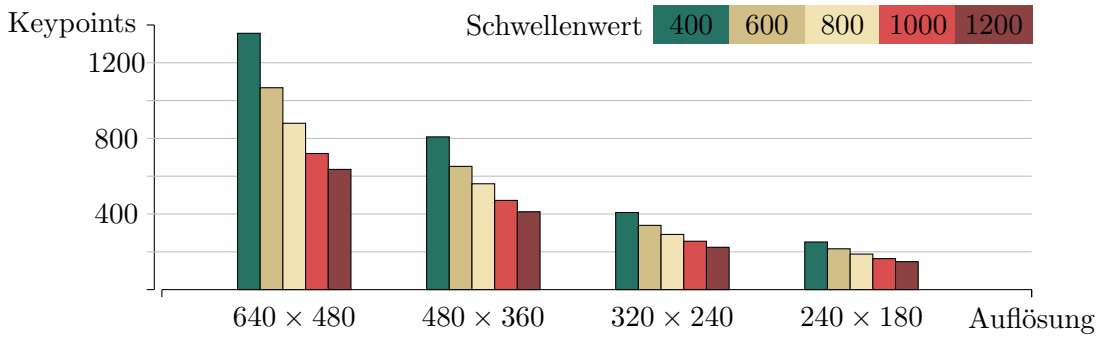


Abbildung 6.4: Durchschnittliche Anzahl an SURF Keypoints auf den fünf Bildern der Benchmark App für unterschiedliche Bildauflösungen und Schwellenwerte.

Die Auswirkungen der unterschiedlichen Bildauflösungen und Schwellenwerte für die Keypoint-Detektion auf die Zeit, die zum Berechnen der lokalen Features benötigt wird, sind in den Diagrammen in Abbildung 6.5 (SIFT) und Abbildung 6.6 (SURF) zu sehen. Als Grundlage für die zeitliche Messung wurde wiederum die Benchmark App mit ihren fünf Bildern verwendet. Der Rechenaufwand setzt sich aus der Zeit für die Skalierung des Bildes sowie den Zeiten für die Berechnung der Keypoints und der Deskriptoren zusammen. Die Zeit für die Skalierung des Bildes ist verhältnismäßig klein und somit bei dem Maßstab der Diagramme kaum zu sehen. Sowohl bei SIFT als auch bei SURF Features wird der größte Geschwindigkeitsgewinn durch die Reduzierung der Auflösung erreicht. Der Geschwindigkeitsgewinn durch eine Erhöhung des Schwellenwertes fällt weniger stark aus. Vor allem bei Bildern mit geringer Auflösung ist der Geschwindigkeitsgewinn, der durch den Schwellenwert erreicht werden kann, kaum noch sichtbar. Im Vergleich zwischen SIFT und SURF Features schneiden SURF Features, wie zu erwarten, deutlich besser ab. Dies wird deutlich, wenn die durchschnittliche Rechenzeit für ein Feature betrachtet wird. Gemittelt über alle unterschiedlichen Bildauflösungen und Schwellenwerte wird für die Berechnung eines SURF Features im Schnitt $0,67$ ms benötigt, wohingegen die Berechnung eines SIFT Features $2,39$ ms in Anspruch nimmt.

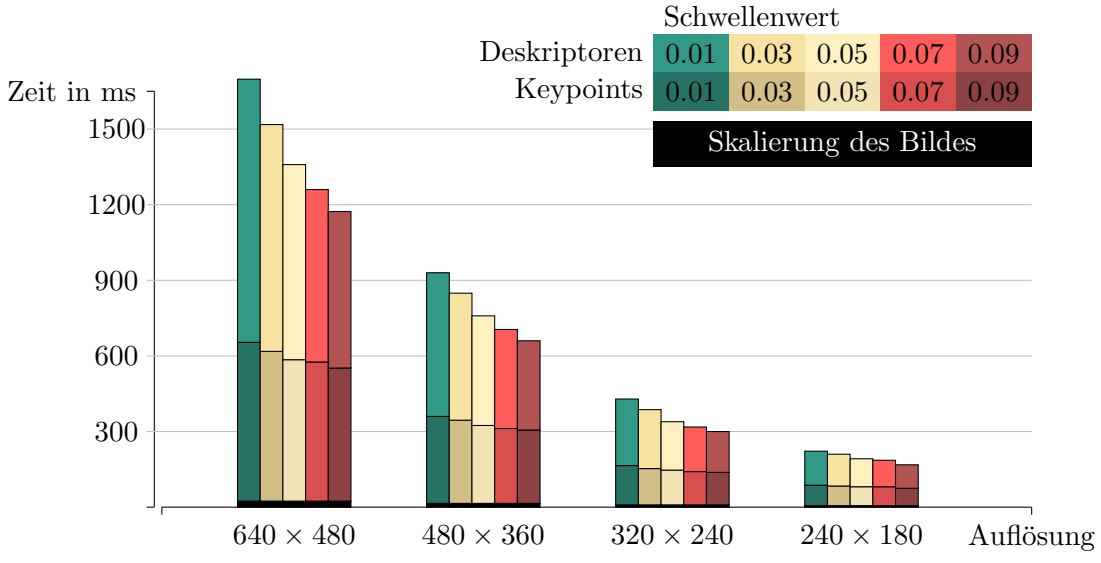


Abbildung 6.5: Durchschnittliche Zeit, die für die Berechnung von SIFT Features auf den Bildern der Benchmark App benötigt wird, für unterschiedliche Bildauflösungen und Schwellenwerte.

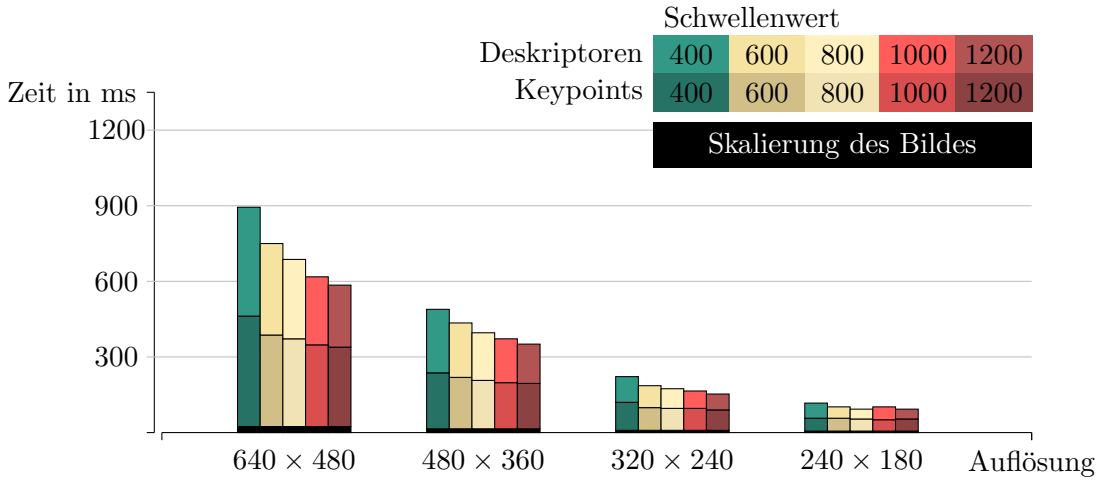


Abbildung 6.6: Durchschnittliche Zeit, die für die Berechnung von SURF Features auf den Bildern der Benchmark App benötigt wird, für unterschiedliche Bildauflösungen und Schwellenwerte.

Eine geringe Rechenzeit allein macht keinen guten Objekterkennungsalgorithmus aus. Als zweites Kriterium werden deswegen die Klassifikationsergebnisse betrachtet, die mit Hilfe der Mean Average Precision (MAP) verglichen werden. Die Diagramme in Abbildung 6.7 (SIFT) und 6.8 (SURF) zeigen die unterschiedlichen MAP-Werte für die verschiedenen Bildauflösungen und Schwellenwerte für die Keypoint-Detektion.

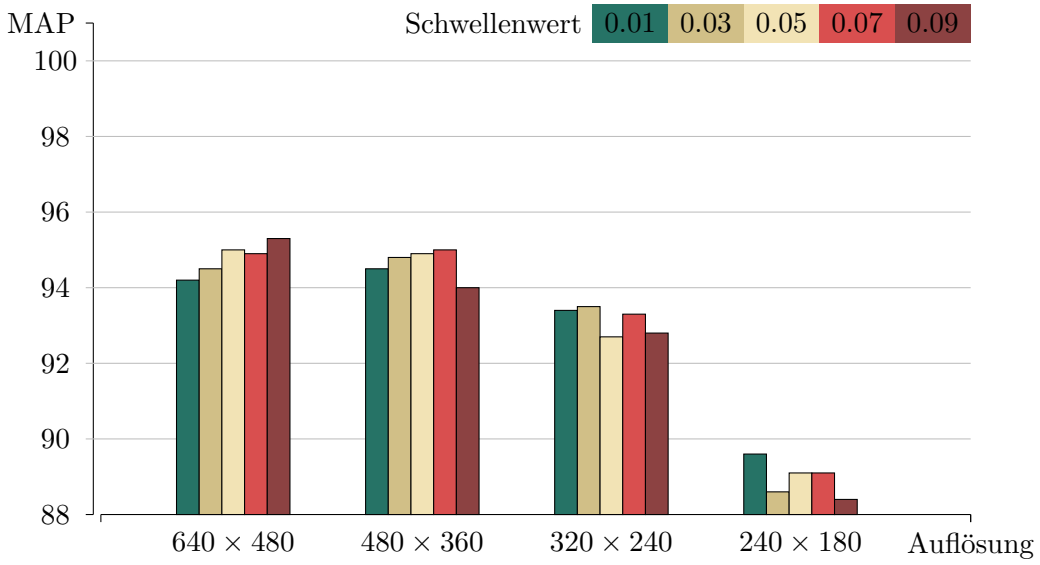


Abbildung 6.7: Mean Average Precision bei der Verwendung von SIFT Features für unterschiedliche Bildauflösungen und Schwellenwerte. *Bag of Words* Parameter: Codebuchgröße 200, Standard Histogramm, SVM mit RBF Kernel.

Die MAP-Werte bei der Verwendung von SIFT Features steigen für die Bildauflösung 640×480 an, wenn der Schwellenwert erhöht wird und damit weniger, aber dafür eindeutigere Keypoints detektiert werden. Die Ergebnisse verhalten sich bei der Bildauflösung 480×360 ähnlich, wobei für den Schwellenwert 0.09 ein Abfall des MAP-Wertes zu beobachten ist. Weiterhin ist sichtbar, dass die Verringerung der Bildauflösung auf 320×240 eine leichte und die Verringerung auf 240×180 eine starke Reduzierung der MAP-Werte zur Folge hat.

Bei der Verwendung von SURF Features zeigt sich, dass die Bildauflösungen mit 640×480 und 480×360 Pixeln ähnlich gute Ergebnisse bringen. Die höhere von den beiden Auflösungen hat nur einen minimalen Vorteil. Ähnlich verhält es sich mit den Auflösungen 320×240 und 240×180, die jedoch mit den beiden größeren Auflösungen nicht mithalten können. Im Gegensatz zu SIFT Features verringern sich die MAP-Werte durch eine Erhöhung der Schwellenwerte bei allen Bildauflösungen.

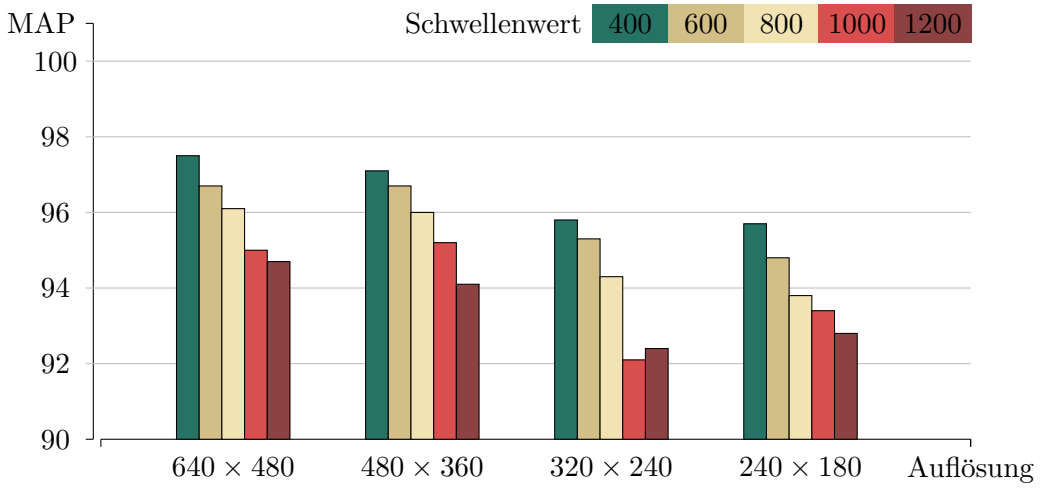


Abbildung 6.8: Mean Average Precision bei der Verwendung von SURF Features für unterschiedliche Bildauflösungen und Schwellenwerte. *Bag of Words* Parameter: Codebuchgröße 200, Standard Histogramm, SVM mit RBF Kernel.

Der beste MAP-Wert bei der Verwendung von SIFT Features hat einen Wert von $95,3$ (Bildauflösung 640×480 und Schwellenwert $0,09$). Bei der Verwendung von SURF Features gibt es gleich sieben Werte, die höhere MAP-Werte erreichen als die beste SIFT Parametrisierung. Das beste Ergebnis bei der Verwendung von SURF Features hat den Wert $97,5$ (Bildauflösung 640×480 und Schwellenwert 400), dicht gefolgt vom Wert $97,1$ (Bildauflösung 480×360 und Schwellenwert 400). Bei der Betrachtung der Rechenzeiten und den Mean Average Precision Werten zeigt sich, dass die Verwendung von SURF Features sowohl bei dem Rechenaufwand als auch bei den Klassifikationsergebnissen Vorteile im Gegensatz zu SIFT Features besitzt.

Bei der Wahl der Bildauflösung und des Schwellenwertes für die SURF Features kann nicht einfach die Parametrisierung mit dem höchsten MAP-Wert (Bildauflösung 640×480 und Schwellenwert 400) ausgewählt werden, da für die Berechnung eine relativ hohe Zeit von 871 ms benötigt wird. Einen Kompromiss stellt die Verwendung von SURF Features mit der Bildauflösung 480×360 und dem Schwellenwert 400 dar. Mit dieser Parametrisierung ist die Berechnung der lokalen Features auf den Bildern der Benchmark App in 475 ms möglich und somit 396 ms schneller als die Parametrisierung mit dem besten MAP-Wert. Auf den Trainings- und Testdatensatz erreicht sie trotzdem den zweitbesten MAP-Wert von $97,1$.

Als Alternative zu SIFT und SURF Features wurden auch GLOH Features in Betracht gezogen. Messungen haben jedoch ergeben, dass ihre Berechnung langsamer ist als SIFT und dass sie bei den Klassifikationsergebnissen keine Verbesserung darstellen. Aus diesem Grund wird auf eine detaillierte Evaluation von GLOH Features verzichtet.

Berechnung des BoW-Histogramms

Neben der Wahl des lokalen Features, der Codebuchgröße und des Histogramm Typs hängt der Rechenaufwand ebenfalls von der Zahl der Features ab, die in einem Bild gefunden

werden. Folgende Zeitmessungen für die Bildung eines *Bag of Words* Histogramms basieren auf der Annahme, dass ein Bild 400 Features enthält. Das Diagramm in Abbildung 6.9 visualisiert die Zeit, die zum Berechnen eines Standard *Bag of Words* Histogramms für unterschiedlich große Codebücher mit SURF und SIFT Features benötigt wird.

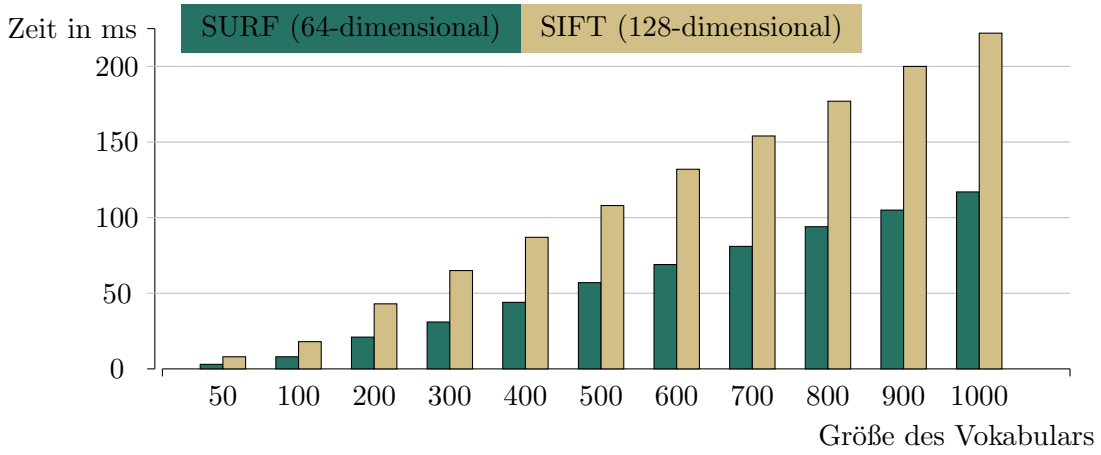


Abbildung 6.9: Rechenaufwand für die Erzeugung des Standard *Bag of Words* Histogramms bei der Verwendung von 64- und 128-dimensionalen Deskriptoren für unterschiedliche Codebuchgrößen. Anzahl der Deskriptoren: 400.

Der Rechenaufwand steigt dabei mit der Größe des visuellen Vokabulars annähernd linear an. Wie bereits bei der Berechnung der lokalen Features bringen SURF Features einen Geschwindigkeitsvorteil im Vergleich zu SIFT Features, der sich auf die unterschiedlich großen Deskriptoren zurückführen lässt. SURF Deskriptor enthalten 64 Einträge, während SIFT Deskriptoren 128 Einträge enthalten. Dies bestätigt die Wahl von SURF Features für die Anwendung auf mobilen Endgeräten. Auf eine weitere Betrachtung von SIFT Features wird daher verzichtet. Ein größeres Histogramm wirkt sich nicht nur auf die Zeit aus, die für die Histogrammbildung benötigt wird, sondern erhöht zusätzlich den Rechenaufwand für die Klassifikation. Die Zeiten für die Klassifikation eines *Bag of Words* Histogramms werden am Ende der Evaluation betrachet.

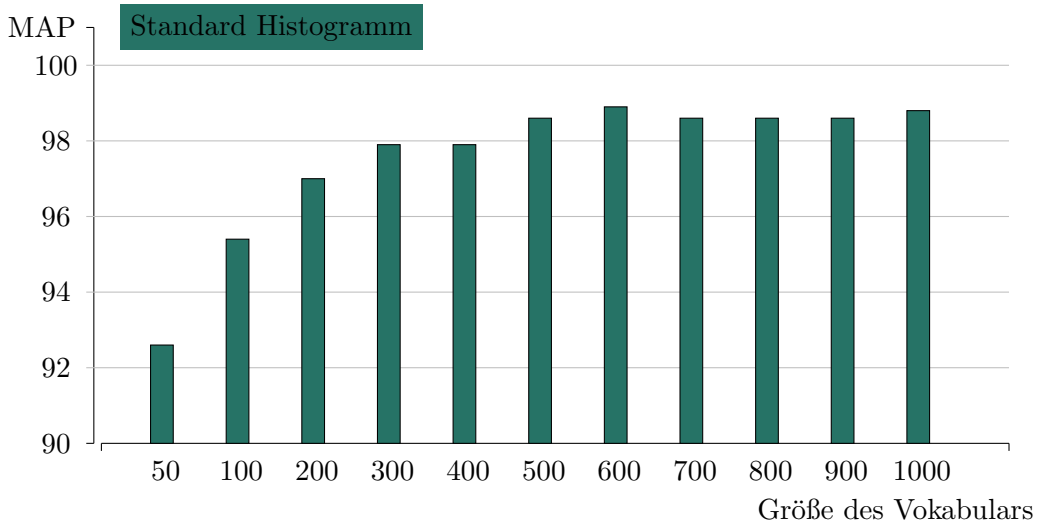


Abbildung 6.10: Mean Average Precision Werte für unterschiedliche Codebuchgrößen. *Bag of Words* Parameter: Bildauflösung 480×360, SURF Features mit Schwellenwert 400, Standard Histogramm, SVM mit RBF-Kernel.

Das Diagramm in Abbildung 6.10 zeigt den Einfluss der Größe des visuellen Vokabulars auf die Mean Average Precision Werte. Dabei ist zu beobachen, dass die MAP-Werte mit der Erhöhung der Codebuchgröße zunächst ansteigen, bis bei einer Codebuchgröße von ungefähr 500 eine Sättigung eintritt. Ein visuelles Vokabular mit 500 Einträgen erreicht einen Mean Average Precision Wert von 98, 6. Eine weitere Erhöhung der Codebuchgröße bringt keine wesentliche Verbesserung der Ergebnisse. Die Werte schwanken zwischen 98, 6 und 98, 9. Die Berechnung des Histogramms mit 500 Bins benötigt 57 ms.

Für die Auswertung räumlicher Histogramme wurde das Bild in 2×2 Regionen sowie in eine Pyramidenrepräsentation mit 3 Leveln unterteilt. Dabei wurden Codebücher mit zwei verschiedenen Größen (100 und 500 Einträge) betrachtet. Die Zeit (Abbildung 6.11a), die für die Berechnung räumlicher *Bag of Words* Histogramme benötigt wird, ist mit denen eines Standard Histogramms vergleichbar. Der Aufwand für die Klassifikation erhöht sich jedoch, da die Anzahl an Bins von räumlichen Histogrammen einem Vielfachen der Codebuchgröße entspricht (Vergleich Abschnitt 4.4).

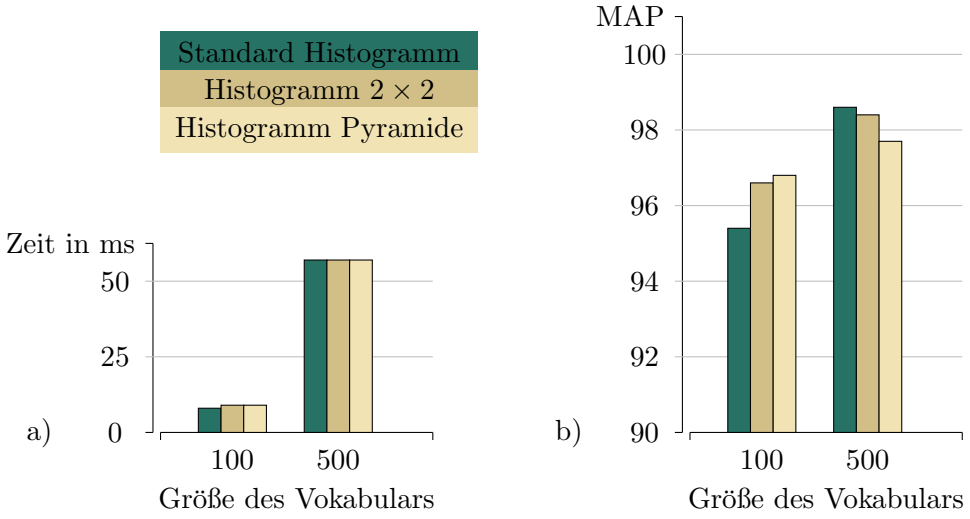


Abbildung 6.11: Vergleich des Standard Histogramms mit den räumlichen Histogrammen mit 2×2 Regionen und einer Pyramidenrepräsentation mit 3 Leveln. a) Rechenaufwand b) Mean Average Precision Werte. *Bag of Words* Parameter: Bildauflösung 480×360, SURF Features mit Schwellenwert 400, SVM mit RBF-Kernel.

Auf dem Trainings- und Testdatensatz verbessern sich die Mean Average Precision Werte (Abbildung 6.11b) im Vergleich zum Standardhistogramm sowohl bei einer Einteilung des Bildes in 2×2 Regionen als auch durch die Verwendung einer Pyramidenrepräsentation, wenn ein Codebuch mit 100 visuellen Wörtern eingesetzt wird. Bei einem Codebuch mit 500 Einträgen verschlechtern sich die MAP-Werte. Darüber hinaus haben Praxistests ergeben, dass räumliche Histogramme wesentlich schlechter abschneiden. Die guten Evaluationsergebnisse für räumliche Histogramme bei einer Codebuchgröße von 100 sind auf eine Überanpassung der Klassifikatoren an den Trainings- und Testdatensatz zurückzuführen (Overfitting).

Als weitere Alternative zu einem Standard Histogramm oder zu räumlichen Histogrammen steht das fuzzy Histogramm zur Verfügung. Hier wird die Methode von Jiang *et al.* [26] evaluiert, bei der ein Deskriptor die Bins der 4 ähnlichsten Codebucheinträge beeinflusst. Für die Berechnung des Fuzzy Histogramms wird, wie in Abbildung 6.12 zu sehen, nur wenig mehr Zeit benötigt als für ein Standard Histogramm. Die Evaluation auf dem Trainings- und Testdatensatz, welche in Abbildung 6.13 dargestellt ist, zeigt, dass die Ver-

wendung eines fuzzy Histogramms für fast alle Codebuchgrößen zu einer Verbesserung des Mean Average Precision Wertes führt.

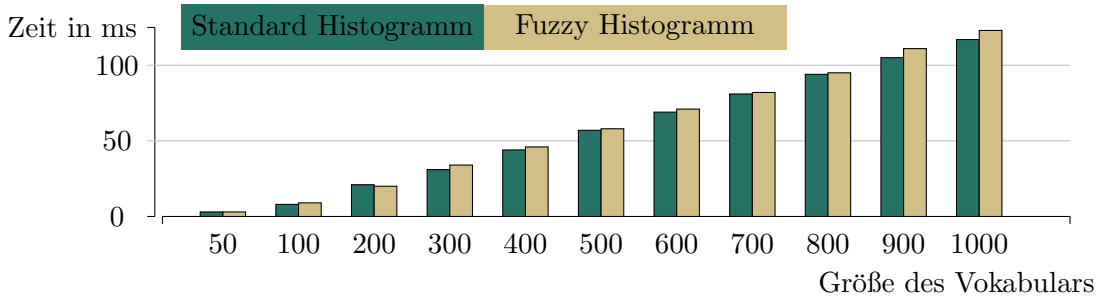


Abbildung 6.12: Vergleich des Rechenaufwandes für die Erzeugung eines Standard Histogramms und eines fuzzy Histogramms für unterschiedlich große Codebücher. Es wurden jeweils 400 Deskriptoren verwendet.

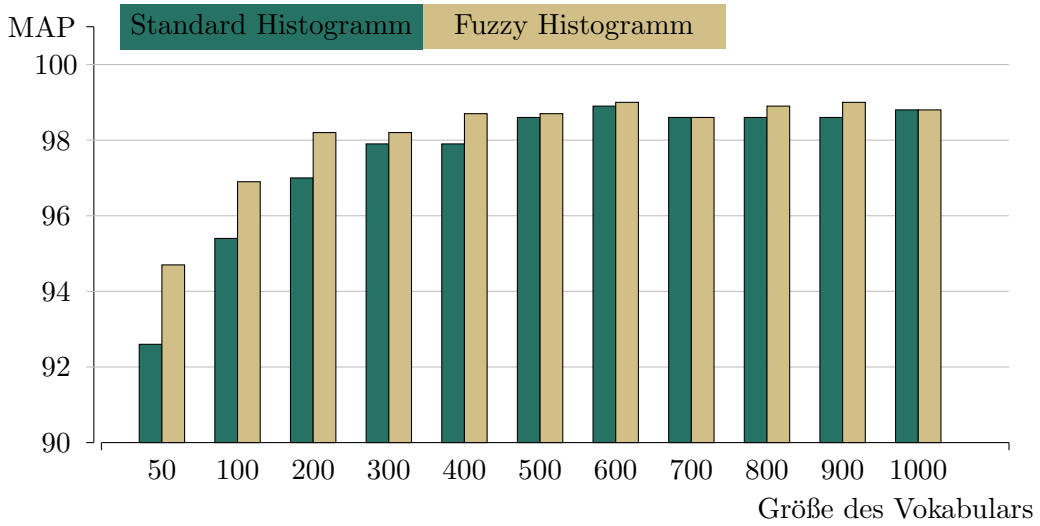


Abbildung 6.13: Vergleich der Mean Average Precision Werte für die Verwendung eines Standard Histogramms und eines fuzzy Histogramms für unterschiedlich große Codebücher. *Bag of Words* Parameter: Bildauflösung 480×360, SURF Features mit Schwellenwert 400, SVM mit RBF-Kernel.

Unter bestimmten Voraussetzungen kann die Zeit für die Berechnung des Histogramms durch die Verwendung von FLANN (Vergleich Abschnitt 4.3) beschleunigt werden. Abbildung 6.14 zeigt ein Diagramm, in dem die Zeiten für die Berechnung des Histogramms mit und ohne FLANN auf einem Samsung Galaxy S2 visualisiert sind. Für kleine Codebuchgrößen verlangsamt FLANN die Berechnung des Histogramms. Erst ab einem Histogramm mit mindestens 500 Einträgen ist eine Beschleunigung durch die Verwendung von FLANN möglich. Der Geschwindigkeitsgewinn wächst mit der Vergrößerung des Vokabulars. Ob sich die Verwendung von FLANN wirklich lohnt, hängt vom verwendeten Endgerät ab. Zum Beispiel haben Zeitmessungen auf einem ASUS Transformer Prime TF201 Tablet, welches mit vier Rechenkernen arbeitet, ergeben, dass die Verwendung von FLANN zu einer Verlangsamung der Histogrammberechnung bei einer Codebuchgröße von 500 führt. Eine Evaluation auf den Trainings- und Testdaten hat bestätigt, dass der Mean Average Precision Wert nicht abfällt, obwohl FLANN die nächsten Nachbarn lediglich approximative berechnet.

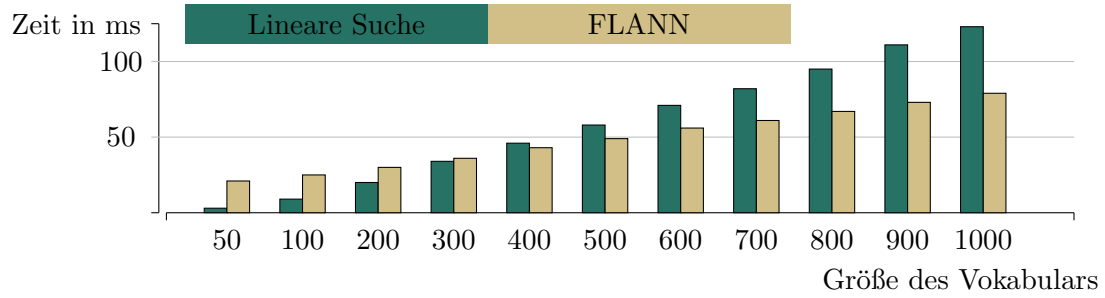


Abbildung 6.14: Auswirkungen auf die Zeit für die Berechnung eines fuzzy Histogramms bei der Verwendung von FLANN

Die bisherige Evaluation für die Berechnung des *Bag of Words* Histogramms lässt folgende Schlüsse zu. Die Verwendung eines fuzzy Histogramms verbessert die Klassifikationsergebnisse, wohingegen räumliche Histogramme aufgrund von Overfitting in der Praxis nicht zu gebrauchen sind. Aus diesem Grund wird für die Parametrisierung des *Bag of Words* Histogramms ein fuzzy Histogramm mit 500 Einträgen auf Basis von SURF Features mit Schwellenwert 400 vorgeschlagen, welches mit Hilfe von FLANN berechnet wird.

Klassifikation

Für die Klassifikation anhand des *Bag of Words* Histogramms werden der fuzzy k-Nearest-Neighbor Algorithmus (kNN) mit der L2-Norm und Support Vector Machines mit RBF-Kernel ausgewertet. Die Rechenzeit, die für die Klassifikation benötigt wird, hängt neben der Wahl des Klassifikators von der Histogrammgröße und der Anzahl der Objektklassen ab, die eingelernt wurden. Die Zeiten des k-Nearest-Neighbor Klassifikators werden zusätzlich von der Zahl der gespeicherten BoW-Histogramme beeinflusst. Für die Klassifikation der 18 eingelernten Objektklassen des in dieser Arbeit verwendeten Datensatzes ergeben sich für die Histogrammgrößen 50, 200, 500 und 800 die Zeiten in Abbildung 6.16.

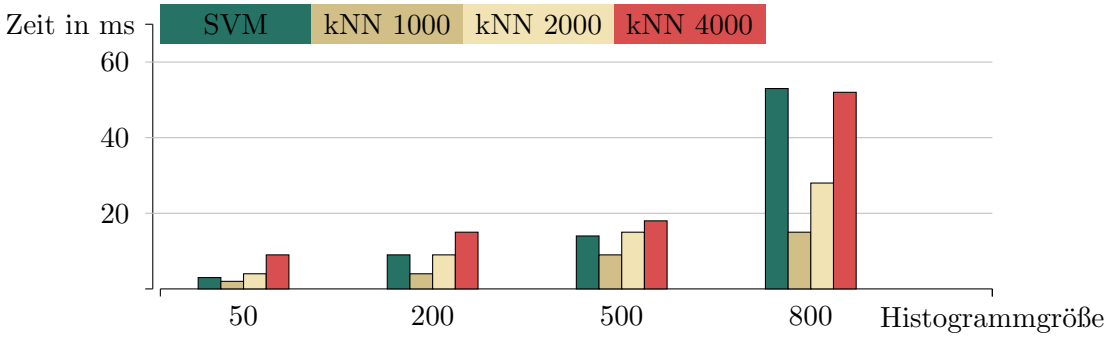


Abbildung 6.15: Rechenzeiten für die Klassifikation mit Support Vector Machines und dem Nearst Neighbor Klassifikator. Letzterer wurde mit 1000, 2000 und 4000 gespeicherten BoW-Histogrammen evaluiert.

Die Ergebnisse zeigen, dass der k-Nearest-Neighbor Klassifikator im Vergleich zu Support Vector Machines schneller arbeitet, solange die Zahl der gespeicherten BoW-Histogramme unterhalb eines Schwellenwertes liegt, der abhängig von der Größe des Histogramms ist. Da die Klassifikationszeiten bei der Verwendung eines Histogramms mit 500 Bins im Vergleich zu anderen Teilabschnitten des *Bag of Words* Verfahrens gering sind, kann der Klassifikator anhand der Mean Average Precision ausgewählt werden. Abbildung 6.16 zeigt, dass dabei die Verwendung von Support Vector Machines (MAP: 98,7) einen deutlichen Vorteil im Vergleich zum fuzzy k-Nearest-Neighbor Klassifikator (MAP: 93,7) hat.

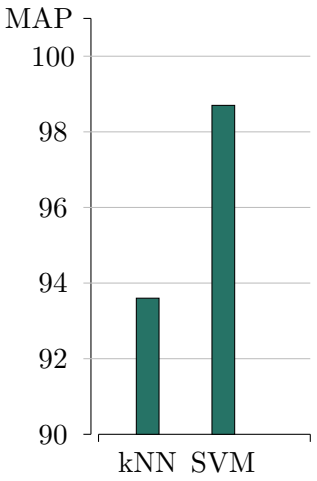


Abbildung 6.16: Mean Average Precision Werte für den fuzzy k-Nearest-Neighbor Klassifikator und die Support Vector Machines mit RBF-Kernel. *Bag of Words* Parameter: Bildauflösung 480 × 360, SURF Features mit Schwellenwert 400, Codebuchgröße 500, fuzzy Histogramm.

Laufzeit auf unterschiedlichen mobilen Endgeräten

Im Folgenden wird die Geschwindigkeit der finalen Parametrisierung des *Bag of Words* Verfahrens auf unterschiedlichen mobilen Endgeräten betrachtet. Hierzu werden die Ergebnisse der Benchmark APP auf 7 Smartphones und 2 Tablets ausgewertet. Tabelle 6.2 zählt die verwendeten Modelle mit den verbauten Prozessoren auf.

Die Rechenzeit für die Auswertung eines Frames setzt sich dabei wie folgt zusammen:

1. Skalierung des Bildes auf 480 × 360 Pixel
2. Berechnung der SURF Keypoints mit Schwellenwert 400
3. Berechnung der SURF Deskriptoren
4. Bildung des fuzzy Histogramms mit 500 Bins
5. Klassifikation mit den Support Vector Machines

ASUS Transformer Prime TF201	mit Android 4.0.3
Prozessor	NVIDIA Tegra 3 1,4 Ghz Quad-Core CPU (Cortex-A9)
Samsung Galaxy S2 GT-I9100	mit Android 2.3.6
Prozessor	Samsung Exynos 4 (4210) 1,2 Ghz Dual-Core CPU (Cortex-A9)
Google Galaxy Nexus	mit Android 4.04
Prozessor	Texas Instruments OMAP 4 (4460) 1,2 Ghz Dual-Core CPU (Cortex-A9)
Samsung Galaxy Tab 10.1 GT-P7500	mit Android 3.1
Prozessor	Tegra 2 1 Ghz Dual-Core CPU (Cortex-A9)
Samsung Galaxy S Plus GT-I9001	mit Android 2.3.3
Prozessor	Qualcomm Snapdragon S2 (MSM8255T) 1,4 Ghz Single-Core CPU (Scorpion)

HTC Desire HD	mit Android 2.3.5
Prozessor	Qualcomm Snapdragon S2 (MSM8255)
	1 Ghz Single-Core CPU (Scorpion)
HTC Desire	mit Android 2.3.7
Prozessor	Qualcomm Snapdragon S1 (QSD 8250)
	1 Ghz Single-Core CPU (Scorpion)
Samsung Galaxy S GT-I9000	mit Android 2.3.3
Prozessor	Samsung Hummingbird (S5PC110)
	1 Ghz Single-Core CPU (Cortex-A8)
Google Nexus S	mit Android 4.04
Prozessor	Samsung Exynos 3 (3110)
	1 Ghz Single-Core CPU (Cortex-A8)

Tabelle 6.2: Überblick über die mobilen Endgeräte mit ihren Prozessoren, auf denen die Rechenzeit der finalen Parametrisierung des *Bag of Words* Verfahrens gemessen wird.

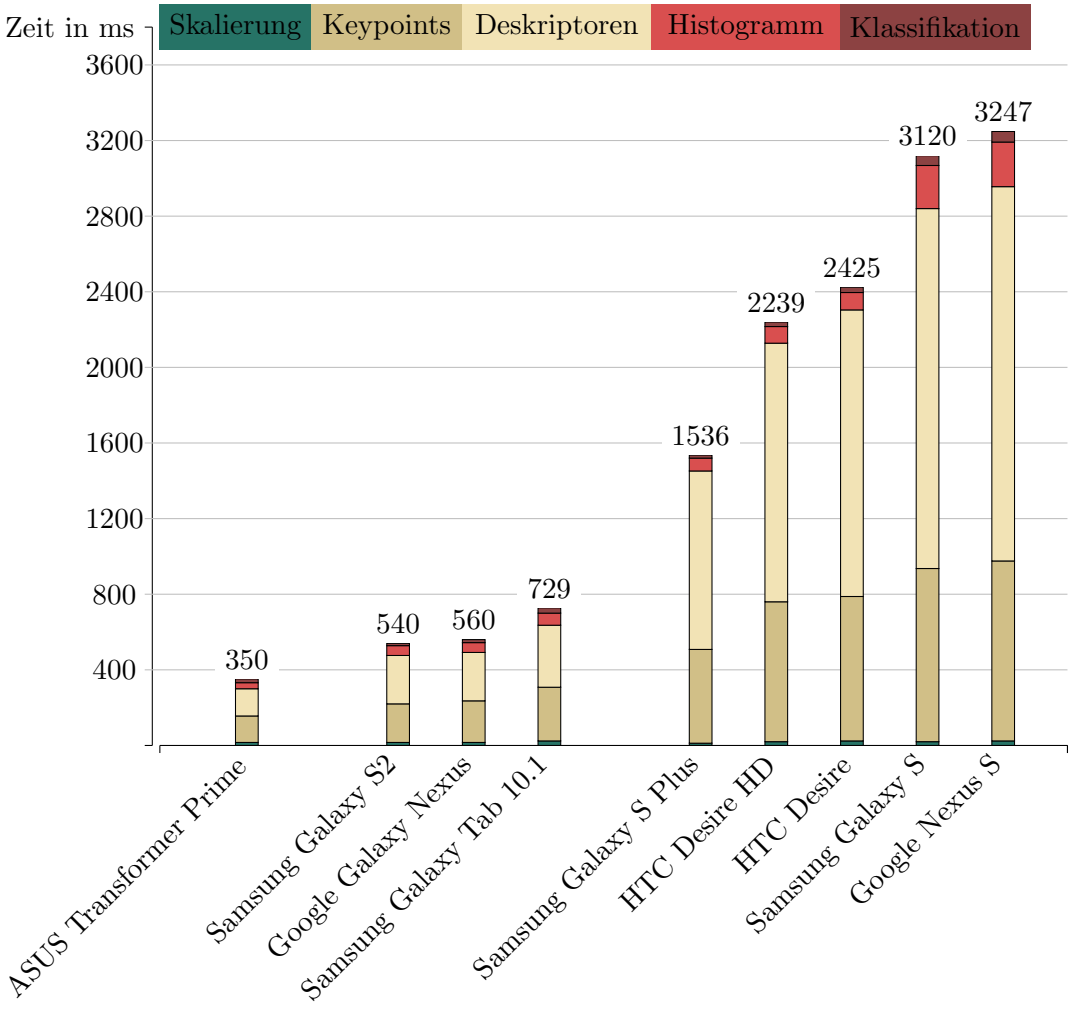


Abbildung 6.17: Vergleich der Verarbeitungszeit eines Frames auf unterschiedlichen mobilen Endgeräten mit Android Betriebssystem.

Die Ergebnisse in Abbildung 6.17 zeigen, dass das Quad-Core Tablet ASUS Transformer Prime mit 350 Millisekunden die schnellste Verarbeitungszeit für ein Frame besitzt. Die Dual-Core Geräte Samsung Galaxy S2, Google Galaxy Nexus und Samsung Galaxy Tab 10.1 benötigen zur Auswertung eines Frames bereits mehr als $0,5$ Sekunden. Das beste Single-Core Smartphone (Samsung Galaxy S Plus) benötigt zur Auswertung eines Frames mehr als $1,5$ Sekunden. Die langsamsten Geräte im Test sind mit mehr als 3 Sekunden das Samsung Galaxy S und das Google Nexus S, die beide noch Prozessoren mit veralteten Cortex-A8 Kernen enthalten. Auf allen Geräten benötigt die Berechnung der Keypoints und Deskriptoren den größten Anteil der Verarbeitungszeit. Je nach mobilem Endgerät liegt dieser zwischen $81,14\%$ und $94,02\%$. Es ist zu beachten, dass die reale Rechenzeit für ein Frame in der Praxis von der Anzahl der Keypoints im Bild und der Auslastung des Prozessors des mobilen Endgerätes durch andere Anwendungen, die auf dem Gerät laufen, abhängt.

6.4 Bewertung der Ergebnisse

Mit der finalen Parametrisierung (Bildauflösung 480×360, SURF Features mit Schwellenwert 400, fuzzy Histogramm mit 500 Einträgen und SVM mit RBF Kernel) wurde eine Parametrisierung gefunden, die geringe Rechenzeiten (350 ms auf einem ASUS Transformer Prime TF201) und auf dem Trainings- und Testdatensatz eine sehr hohe Mean Average Precision ($98,7$) erreicht. Abbildung 6.18 zeigt die Confusion Matrix der Klassifikationsergebnisse, die mit der finalen Parametrisierung auf dem Trainings- und Testdatensatz mit einer 5-fold Cross-Validation gewonnen wurde. Ein Bild wurde dabei als *Kein Objekt* klassifiziert, wenn alle Scores der Support Vector Machines unterhalb des Schwellenwertes 0 lagen (Vergleich Abschnitt 5.2). Dies entspricht der Trennung der Daten anhand der Trennhyperebene. Die Confusion Matrix zeigt, dass die Komponenten von *AMFIS* bis auf wenige Ausnahmen gar nicht verwechselt werden. Je nach Komponente kommt es jedoch vor, dass eine Komponente nicht erkannt wird, obwohl sie auf dem Bild zu sehen ist. Dies lässt sich auf die visuellen Eigenschaften der Komponenten zurückführen.

Der Segway zum Beispiel gehört zu den schmalen Objekten und deckt dadurch nur einen kleinen Bereich eines Bildes ab. Dies führt dazu, dass ein Großteil des Bildes aus unkorreliertem Hintergrund besteht. Wenn der Hintergrund nicht homogen ist, werden viele Keypoints auf dem Hintergrund detektiert. Somit wird das *Bag of Words* Histogramm stärker vom Hintergrund als vom Objekt selbst beeinflusst. Objekte, die einen Großteil des Bildes ausfüllen, wie zum Beispiel die Leiste des Forbots, werden besser erkannt.

Darüber hinaus gibt es Objekte, deren Erscheinung wenig Struktur bieten, um Keypoints detektieren zu können. Hierzu gehören zum Beispiel die Kamera des Forbots, welche fast nur aus schwarzen Quadern besteht, der Imu Mast des Forbots mit seiner blauen Oberfläche und der Ballon, welcher eine texturarme weiße Ballonhülle besitzt. Auf diesen Objekten werden sehr wenige Keypoints gefunden, sodass wiederum die lokalen Features auf dem Hintergrund an Einfluss auf das *Bag of Words* Histogramm gewinnen. Dieser Effekt wird durch die Verkleinerung der Bilder verstärkt, auf denen die lokalen Features detektiert werden.

Das *Bag of Words* Verfahren wurde von Csurka *et al.* [24] für die Kategorisierung von Objekten vorgestellt. Dies hat den Vorteil, dass es bei der Erkennung von Objekten eine große Varianz innerhalb einer Objektklasse zulässt. Somit werden zum Beispiel Quadro- und Octocopter unabhängig von der Anzahl ihrer Rotoren erkannt. Sollen jedoch zum Beispiel die Räder von einer großen Zahl unterschiedlicher Roboter unterschieden werden können, führt dies zu sehr ähnlichen *Bag of Words* Histogrammen, was die Unterscheidung sehr erschwert.

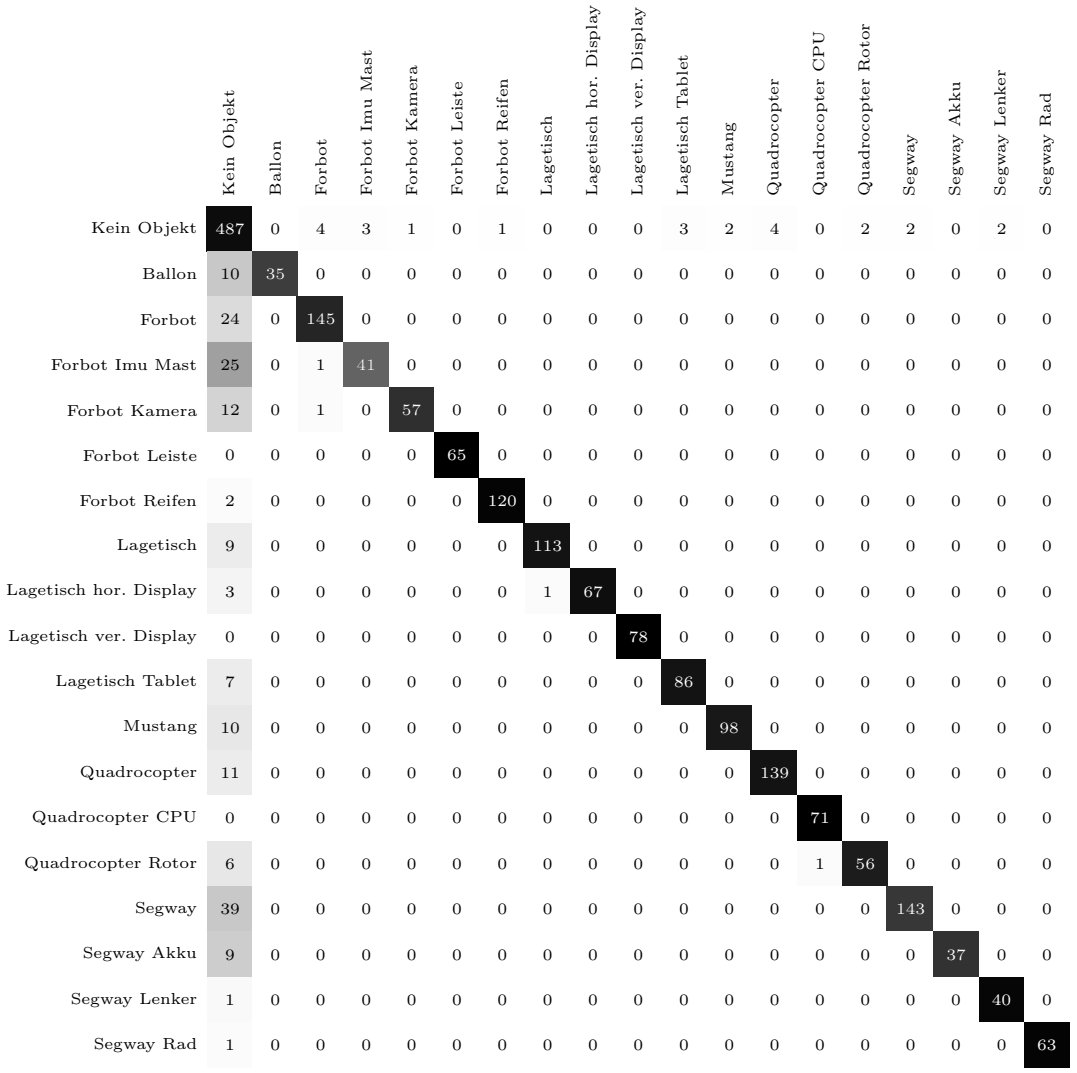


Abbildung 6.18: Die Confusion Matrix für die Klassifikation mit der Support Vector Machine mit RBF Kernel. *Bag of Words* Parameter: Bildauflösung 480×360, SURF Features mit Schwellenwert 400, fuzzy Histogramm mit 500 Einträgen, FLANN

7. Zusammenfassung und Ausblick

Im Rahmen dieser Diplomarbeit wurde eine Android App entwickelt, welche eine bildbasierte Objekterkennung für die Kontextdetektion für das Mobile Lernen vornimmt.

Hierfür wurden zunächst die Grundlagen der Kontextdetektion für den Anwendungsbereich des Mobilen Lernens vorgestellt. Dabei wurden die Begriffe Kontext, Kontextsensitivität und Mobiles Lernen definiert. Es wurden Herausforderungen und weitere Anwendungsgebiete für die bildbasierte Objekterkennung auf mobilen Endgeräten aufgezeigt. Die in dieser Arbeit implementierte App für die bildbasierte Objekterkennung wurde für Smartphones und Tablets mit Android Betriebssystem entwickelt. Dabei wurde das Android Software Developement Kit verwendet. Die Bildverarbeitung der App wurde in C++ implementiert und mit Hilfe des Android Native Developement Kits an die in Java geschriebene App angebunden. Für die Entwicklung der Bildverarbeitungsalgorithmen wurde die OpenCV Bibliothek genutzt.

Für die bildbasierte Objekterkennung wurde der *Bag of Words* Ansatz verwendet. Dabei wird zunächst ein visuelles Vokabular gebildet, indem die Deskriptoren der lokalen Features der Trainingsbilder in Cluster gruppiert werden. Dieses bildet die Grundlage um eine Bildrepräsentation in Form eines Histogramms aufzubauen, welches das Auftreten der einzelnen visuellen Wörter im Bild zählt. Die *Bag of Words* Histogramme der Trainingsbilder werden verwendet um mit Hilfe eines überwachten Trainingsvorgangs einen Klassifikator zu trainieren.

Für die Berechnung des visuellen Vokabulars und das Training der Klassifikatoren wurde ein Windows Programm implementiert. Offline erzeugtes Vokabular und der trainierte Klassifikator werden von der Android App geladen. Das User Interface der Android App wurde anwenderfreundlich konzipiert, sodass es als Assistenzsystem einfach zu bedienen ist. Für die Evaluation der einzelnen Algorithmen und Parameter des *Bag of Words* Verfahrens wurde das Windows Programm erweitert, welches bereits für das Training des Verfahrens verwendet wurde. Die Rechenzeiten auf mobilen Endgeräten wurden mit Hilfe einer für diese Zwecke entwickelten Benchmark App analysiert.

Die Algorithmen für die Teilabschnitte des *Bag of Words* Verfahrens und die Parameter wurden mit einer 5-fold Cross-Validation und der Mean Average Precision als Performancemaß evaluiert. Dabei wurde ein Datensatz mit 2140 Bildern verwendet, die mit mobilen Endgeräten aufgenommen wurden. Für die Berechnung der lokalen Features wurden SIFT und SURF Features mit unterschiedlichen Schwellenwerten für die Keypointdetektion und mehrere Bildauflösungen evaluiert. Für die Berechnung des Histogramms wurden

unterschiedliche Histogrammgrößen, fuzzy Histogramme und räumliche Histogramme betrachtet. Als Klassifikatoren wurden der k-Nearest-Neighbor Algorithmus und die Support Vector Machines evaluiert. Die Parameter wurden bezüglich der Klassifikationsergebnisse und der Rechenzeiten auf mobilen Endgeräten ausgewählt. Die finale Parametrisierung erreicht auf dem verwendeten Trainings- und Testdatensatz eine Mean Average Precision von $98,7$ und ist in der Lage eingelernte Komponenten des *AMFIS*-Systems in 350ms auf einem ASUS Transformer Prime TF201 Tablet zu erkennen.

Durch das Einlernen neuer Objekte kann die entwickelte App für andere Anwendungszwecke eingesetzt werden.

Ausblick

Die Geschwindigkeit, mit der Bildverarbeitungsalgorithmen auf mobilen Geräten ausgeführt werden können, wird sich aufgrund der Entwicklung der Hardware mobiler Endgeräte weiter verbessern. Geräte, die in der Zukunft erscheinen, werden mehr Rechenkerne und eine höhere Taktung besitzen. Neben der Beschleunigung der Bildverarbeitung durch verbesserte Hardware kann eine Beschleunigung erreicht werden, indem die Bildverarbeitungsalgorithmen weiter für die Anwendung auf mobilen Geräten mit ARM Prozessoren optimiert werden.

Die Algorithmen von OpenCV wurden für die Intel Prozessorarchitektur entwickelt. Diese enthält eine *Single Instruction, Multiple Data* (SIMD) Technologie mit dem Namen *Streaming SIMD Extensions* (SSE), mit der eine Instruktion auf mehreren Pixeln eines Bildes simultan ausgeführt werden kann. ARM Prozessoren besitzen mit dem NEON Koprozessor [70] eine ähnliche Technologie, für die die OpenCV Bibliothek in Version 2.4.1 jedoch keine Optimierung besitzt. Die Optimierung der Low Level Routinen von OpenCV für den NEON Koprozessor kann einen erheblichen Geschwindigkeitsvorteil mit sich bringen [71].

Weiterhin können die Bildverarbeitungsalgorithmen mit Hilfe des Grafikprozessors (GPU) der mobilen Endgeräte parallelisiert werden. OpenCV bringt GPU-beschleunigte Algorithmen auf Basis der *Compute Unified Device Architecture* (CUDA) von NVIDIA mit sich. Diese können aktuell auf Android-basierten Endgeräten jedoch noch nicht genutzt werden.

Neben der Optimierung der Geschwindigkeit der Bildverarbeitungsalgorithmen kann die bildbasierte Erkennung von Objekten verbessert werden, indem der *Bag of Words* Ansatz mit globalen Features, wie zum Beispiel Farbmomente, kombiniert wird [26].

Smartphones und Tablets bringen neben der Kamera weitere Sensoren mit, die eingesetzt werden können um die bildbasierte Objekterkennung zu vereinfachen. GPS kann zum Beispiel verwendet werden um sowohl die Komponenten von *AMIFS* als auch den Benutzer des Smartphones zu lokalisieren. Auf Basis dieser Informationen kann eine Vorauswahl getroffen werden, welche Komponenten von *AMFIS* sich in der Nähe des Benutzers befinden. Hierdurch muss sich der Klassifikator lediglich zwischen einer reduzierten Menge an Objekten entscheiden. Innerhalb von Gebäuden kann Bluetooth oder WLAN verwendet werden um Geräte und Anwender zu lokalisieren.

Funktional kann das in dieser Arbeit implementierte Assistenzsystem erweitert werden, indem nicht nur das Objekt im Bild klassifiziert wird, sondern eine Detektion genutzt wird um die Lage einer Komponenten im Bild zu bestimmen. Hierdurch können die Assistenzinformationen direkt ins Bild eingezeichnet und Komponenten direkt beschriftet werden. Wenn die Bildverarbeitungsalgorithmen durch Optimierungen echtzeitfähig werden, kann das Assistenzsystem zu einer Augmented Reality Anwendung erweitert werden, bei der die Assistenzinformationen den auf dem mobilen Endgerät angezeigten Videostream überlagern. Dies führt zu einer weiteren Vereinfachung der Bedienung des Assistenzsystems.

Assistenzsysteme, die durch die Verwendung von Kameras mobiler Endgeräte einen einfachen, schnellen und intuitiven Zugang zu kontextsensitiven Informationen bieten, werden in Zukunft immer mehr an Bedeutung gewinnen. Die Verknüpfung von digital gespeicherten Informationen und real existierenden Objekten auf Basis von bildbasierter Objekterkennung ermöglicht die Entwicklung von neuen kontextsensitiven Anwendungen für verschiedene Anwendungsgebiete.

Abbildungsverzeichnis

1.1 Einsatz eines Quadrocopters im Katastrophenfall 1

2.1 Herausforderungen bei der bildbasierten Objektdetektion 9
2.2 a) QR-Code b) Die *VW SeeMore* App 10
2.3 Die U Snap App 11
2.4 Bildbasierte Gebäudeerkennung 11
2.5 Die *LookTel Money Reader* App 13

3.1 Die Android Systemarchitektur 18

4.1 Der Basisalgorithmus für die Berechnung eines *Bag of Words* Histogramms 21
4.2 Berechnung des differenziellen Skalenraums für SIFT Keypoints 23
4.3 SIFT Keypoint Detektion im differenziellen Skalenraum 23
4.4 Berechnung des SIFT Deskriptors 25
4.5 Approximierte Rechteckfilter der SURF Features 26
4.6 Das Integral Image 26
4.7 Skalenraumrepräsentation bei SIFT und SURF 27
4.8 Haar-Wavelet Filter in x- und y-Richtung 27
4.9 Bestimmung der Hauptorientierung eines SURF Keypoints 28
4.10 Berechnung des SURF Deskriptors 29
4.11 Räumliches *Bag of Words* mit unterschiedlichen Bildaufteilungen 31
4.12 Räumliches *Bag of Words* mit einer Pyramidenrepräsentation 32
4.13 Die Trennhyperebene einer Support Vector Machine 34

5.1 Systemaufbau der Implementierung 38
5.2 Anwendungsfall 1 42
5.3 Anwendungsfall 2 44
5.4 Bilder der Benchmark App 45

6.1 Verteilung der Bilder des Datensatzes auf die einzelnen Objektklassen . . . 48
6.2 Beispiele für eine ROC und eine PR Kurve 49
6.3 Anzahl SIFT Keypoints 51
6.4 Anzahl SURF Keypoints 52
6.5 Zeit für die Berechnung von SIFT Features 52
6.6 Zeit für die Berechnung von SURF Features 53
6.7 Mean Average Precision SIFT Features 53
6.8 Mean Average Precision SURF Features 54
6.9 Zeit für die Berechnung des *Bag of Words* Histogramms 55
6.10 Mean Average Precision Codebuchgrößen 55
6.11 Zeit und Mean Average Precision räumliche Histogramme 56
6.12 Zeit für die Berechnung von Standard und fuzzy Histogramm 57
6.13 Mean Average Precision Standard und fuzzy Histogramm 57
6.14 Zeit für die Berechnung des Histogramms mit FLANN 58

6.15 Zeit für die Klassifikation mit SVM und NN 58
6.16 Mean Average Precision kNN und SVM 59
6.17 Vergleich der Verarbeitungszeit auf unterschiedlichen Android-Geräten . . . 60
6.18 Confusion Matrix für die Klassifikation mit Support Vector Machines 62

Literaturverzeichnis

[1] A. Streicher, D. Szentes, B.-A. Bargel, and W. Roller, "Context-aware mobile learning for reconnaissance and surveillance," in *Online Educa Berlin 2011: Nov 30- Dec 2, 2011; 17th Internatioanl conference on Technology supported Learning and Training; book of abstracts*, pp. 283–284, 2011.

[2] NVIDIA, "The benefits of quad core cpus in mobile devices." White Paper, 2011.

[3] I. Chen, "Sensors in smartphones: Beyond landscape and portrait screen orientation." Electronic Design, April 2012.

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, June 2010.

[5] P. Azad, T. Asfour, and R. Dillmann, "Combining apperance-based and model-based methods for real-time object recognition and 6d localization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5339–5344, 2006.

[6] M. J. Swain and D. H. Ballard, "Color indexing," *International Journal of Computer Vision*, vol. 7, pp. 11–32, Nov. 1991.

[7] M. Stricker and M. Orengo, "Similarity of color images," in *Storage and Retrieval for Image and Video Databases (SPIE)*, pp. 381–392, 1995.

[8] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 3, pp. 610 –621, nov. 1973.

[9] J. G. Daugman, "Complete discrete 2D gabor transform by neural networks for image analysis and compression," *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 36, no. 7, pp. 1169–1179, 1988.

[10] B. Schiele and J. L. Crowley, "Object recognition using multidimensional receptive field histograms," *Energy*, vol. 1064, no. section 6, pp. 610–619, 1996.

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 886–893, 2005.

[12] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (Hawaii), 2001.

[13] X. Sheng and P. Qi-cong, "3d object recognition using multiple features and neural network," in *IEEE Conference on Cybernetics and Intelligent Systems*, pp. 434 –439, sept. 2008.

[14] L. He, H. Wang, and H. Zhang, "Object detection by parts using appearance, structural and shape features," in *International Conference on Mechatronics and Automation (ICMA)*, pp. 489 –494, aug. 2011.

[15] C. Harris and M. Stephens, "A combined corner and edge detection," in *Proceedings of The Fourth Alvey Vision Conference*, pp. 147–151, 1988.

[16] J. Shi and C. Tomasi, "Good features to track," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 593–600, June 1994.

[17] J. Matas, O. Chum, U. Martin, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," in *Proceedings of British Machine Vision Conference*, vol. 1, (London), pp. 384–393, 2002.

[18] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the International Conference on Computer Vision - Volume 2*, ICCV '99, (Washington, DC, USA), pp. 1150–, IEEE Computer Society, 1999.

[19] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–110, November 2004.

[20] Y. Ke and R. Sukthankar, "PCA-SIFT: a more distinctive representation for local image descriptors," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 506–513, 2004.

[21] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 27, pp. 1615–1630, October 2005.

[22] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "Speeded-up robust features (surf)," *Computer Vision and Image Understanding (CVIU)*, vol. 110, pp. 346–359, 2008.

[23] P. Azad, T. Asfour, and R. Dillmann, "Combining harris interest points and the sift descriptor for fast scale-invariant object recognition," in *Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, IROS'09, (Piscataway, NJ, USA), pp. 4275–4280, IEEE Press, 2009.

[24] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on Statistical Learning in Computer Vision, ECCV*, pp. 1–22, 2004.

[25] E. Nowak, F. Jurie, and B. Triggs, "Sampling strategies for bag-of-features image classification," in *Computer Vision - ECCV 2006*, vol. 3954 of *Lecture Notes in Computer Science*, pp. 490–503, Springer Berlin / Heidelberg, 2006.

[26] Y. G. Jiang, C. W. Ngo, and J. Yang, "Towards optimal bag-of-features for object categorization and semantic video retrieval," in *CIVR '07: Proceedings of the 6th ACM international conference on Image and video retrieval*, (New York, NY, USA), pp. 494–501, ACM, 2007.

[27] F. Jurie and B. Triggs, "Creating efficient codebooks for visual recognition," in *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 01*, (Washington, DC, USA), pp. 604–610, IEEE Computer Society, 2005.

[28] L. Wu, S. C. H. Hoi, and N. Yu, "Semantics-preserving bag-of-words models and applications," *IEEE Transactions on Image Processing*, vol. 19, pp. 1908–1920, July 2010.

[29] W. Bouachir, M. Kardouchi, and N. Belacel, "Improving bag of visual words image retrieval: A fuzzy weighting scheme for efficient indexation," in *Fifth International*

Conference on Signal-Image Technology Internet-Based Systems (SITIS), pp. 215 – 220, 29 2009-dec. 4 2009.

[30] V. Viitaniemi and J. Laaksonen, "Spatial extensions to bag of visual words," in *Proceedings of the ACM International Conference on Image and Video Retrieval*, CIVR '09, (New York, NY, USA), pp. 37:1–37:8, ACM, 2009.

[31] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, (Washington, DC, USA), pp. 2169–2178, IEEE Computer Society, 2006.

[32] Y. Yang and S. Newsam, "Bag-of-visual-words and spatial extensions for land-use classification," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, (New York, NY, USA), pp. 270–279, ACM, 2010.

[33] A. Bosch, A. Zisserman, and X. Munoz, "Representing shape with a spatial pyramid kernel," in *Proceedings of the 6th ACM international conference on Image and video retrieval*, CIVR '07, (New York, NY, USA), pp. 401–408, ACM, 2007.

[34] Y. Cao, C. Wang, Z. Li, L. Zhang, and L. Zhang, "Spatial-bag-of-features," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010*, pp. 3352 –3359, june 2010.

[35] B. Ruf, E. Kokiopoulou, and M. Detyniecki, "Mobile museum guide based on fast sift recognition.," in *Adaptive Multimedia Retrieval* (M. Detyniecki, U. Leiner, and A. Nürnberger, eds.), vol. 5811 of *Lecture Notes in Computer Science*, pp. 170–183, Springer, 2008.

[36] J. Wang, Y. He, Y. Zhou, and Y. Qiao, "igapsearch: Using phone cameras to search around the world," in *IEEE International Conference on Information and Automation (ICIA), 2011*, pp. 823 –828, june 2011.

[37] M. Viswanathan, C.-K. Chang, J. Moon, and A. Patlolla, "Goggle (or gist on the google phone): A content-based image retrieval system for the gphone," *Spring*, pp. 1–10, 2009.

[38] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision - ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), vol. 3951 of *Lecture Notes in Computer Science*, pp. 404–417, Springer Berlin / Heidelberg, 2006.

[39] K. Jeong and H. Moon, "Object detection using fast corner detector based on smartphone platforms," in *International Conference on Computers, Networks, Systems and Industrial Engineering (CNSI), 2011*, pp. 111 –115, may 2011.

[40] O. Bimber and E. Bruns, "Phoneguide: Adaptive image classification for mobile museum guidance," in *International Symposium on Ubiquitous Virtual Reality*, pp. 1–4, 2011.

[41] A. K. Dey and G. D. Abowd, "Towards a better understanding of context and context-awareness," in *Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, HUC '99, (London, UK, UK), pp. 304–307, Springer-Verlag, 1999.

[42] N. Winters, "What is Mobile Learning," *Big Issues in Mobile Learning*, pp. 7–11, 2007.

[43] O. O. Moses, "Improving mobile learning with enhanced shih's model of mobile learning," *Education*, vol. 5, no. 11, pp. 22–28, 2008.

[44] Volkswagen, "seeMore." http://www.volkswagen-seemore.com. zuletzt abgerufen am 21.06.2012.

[45] A. Alexandridis, P. Charonyktakis, A. Makrogiannakis, A. Papakonstantinou, and M. Papadopouli, "Forthroid on android: A qr-code based information access system for smart phones," in *18th IEEE Workshop on Local Metropolitan Area Networks (LANMAN), 2011*, pp. 1 –6, oct. 2011.

[46] E. Rukzio, A. Schmidt, and H. Hussmann, "Physical posters as gateways to context-aware services for mobile devices," in *Proceedings of the Sixth IEEE Workshop on Mobile Computing Systems and Applications*, (Washington, DC, USA), pp. 10–19, IEEE Computer Society, 2004.

[47] S. Fösken, "Jetzt wird verkauft," *absatzwirtschaft - Zeitschrift für Marketing*, vol. 3, pp. 52–55, 2012.

[48] G. Fritz, C. Seifert, and L. Paletta, "A mobile vision system for urban detection with informative local descriptors," *IEEE International Conference on Computer Vision Systems, 2006 (ICVS)*, p. 30, 2006.

[49] B. Fasel and L. Van Gool, "Interactive museum guide: accurate retrieval of object descriptions," in *Proceedings of the 4th international conference on Adaptive multimedia retrieval: user, context, and feedback*, AMR'06, (Berlin, Heidelberg), pp. 179–191, Springer-Verlag, 2007.

[50] NantWorks, "LookTel." http://www.looktel.com. zuletzt abgerufen am 21.06.2012.

[51] B. Benz and C. Windeck, "Wohngemeinschaft: Die technik der tablet- und smartphone-prozessoren," *c't - magazin für computer technik*, vol. 6, pp. 110 – 113, 2012.

[52] Samsung, "Exynos Processor." http://www.samsung.com/exynos/. zuletzt abgerufen am 21.06.2012.

[53] NVIDIA, "Tegra." http://www.nvidia.de/object/tegra-de.html. zuletzt abgerufen am 21.06.2012.

[54] Texas Instruments, "Wireless Handset Solutions: OMAP Mobile Processors." http://www.ti.com/general/docs/wtbu/wtbugencontent.tsp?templateId=6123&navigationId=11988&contentId=4638. zuletzt abgerufen am 21.06.2012.

[55] Qualcomm, "Snapdragon Processors." http://www.qualcomm.eu/products/snapdragon. zuletzt abgerufen am 21.06.2012.

[56] Android Developers, "What is Android?." http://developer.android.com/guide/basics/what-is-android.html, 2012. zuletzt abgerufen am 21.06.2012.

[57] M. Gargenta, *Learning Android*. O'Reilly Media, 2011.

[58] D. Bornstein, "Dalvik vm internals, presentation, google inc." http://sites.google.com/site/io/dalvik-vm-internals, May 2008. zuletzt abgerufen am 21.06.2012.

[59] Android Developers, "Platform Versions." http://developer.android.com/resources/dashboard/platform-versions.html, 2012. zuletzt abgerufen am 03.07.2012.

[60] Willow Garage, "Opencv." http://opencv.org/, 2012. zuletzt abgerufen am 21.06.2012.

[61] I. Kononenko and M. Kukar, *Machine Learning and Data Mining*. Horwood Publishing, 2007.

[62] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pp. 331–340, INSTICC Press, 2009.

[63] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1981.

[64] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[65] P. Cunningham and S. J. Delany, "k-nearest neighbour classifiers," tech. rep., 2007.

[66] J. M. Keller, M. R. Gray, and Jr, "A fuzzy k-nearest neighbor algorithm," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, pp. 580–585, 1985.

[67] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, June 1998.

[68] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, pp. 861–874, June 2006.

[69] M. Zhu, "Recall , precision and average precision," working paper, University of Waterloo, 2004.

[70] ARM, "Neon." `http://www.arm.com/products/processors/technologies/neon.php`, 2012. zuletzt abgerufen am 21.06.2012.

[71] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov, "Real-time computer vision with opencv.," *Commun. ACM*, vol. 55, no. 6, pp. 61–69, 2012.