CAMPUS FRIEDBERG

IEM
Informationstechnik-
Elektrotechnik-Mechatronik

Entwicklung eines Verfahrens zur automatischen Initialisierung für Augmented Reality-Anwendungen

Studiengang Medieninformatik

Masterarbeit

vorgelegt von

Hans Christian Arlt
geb. in Frankfurt am Main

durchgeführt in der
Technischen Hochschule Mittelhessen, Friedberg

Referent der Arbeit:	Prof. Dr. Cornelius Malerczyk
Korreferent der Arbeit:	Prof. Dr. Michael Guckert

Fachbereiche

Informationstechnik-Elektrotechnik-Mechatronik IEM
und
Mathematik, Naturwissenschaften und Datenverarbeitung MND

Friedberg, 2015

Für meinen geliebten Neffen und Patenkind,
Joshua Samuel Fredrik Brage

Danksagung

Bei einigen Personen,die zum Gelingen dieser Arbeit beigetragen habe, möchte ich mich an dieser Stelle bedanken. Ein ganz besonderer Dank gilt meinen Eltern, die mich auch noch nach meinem Bachelorabschluss in der Zeit meines Masterstudiums sowie während der Zeit, in der ich diese Arbeit verfasst habe, bestmöglich unterstützt haben. Weiterhin möchte ich mich bei meiner Freundin Natalie bedanken für ihre Unterstützung und Ermutigung, wenn ich nicht mehr weiterwusste. Meinen Freunden danke ich für das Verständnis, dass ich für eine Zeit nicht mehr zu erreichen war.

Ein großer Dank geht an meinen Referenten, Professor und Mentor Prof. Dr. Cornelius Malerczyk, der mich unterstützt und mir ermöglicht hat, mir neben dem Masterstudium, Erfahrungen in Lehre und Forschung anzueignen. Dank ihm und meinem Korreferenten Prof. Dr. Michael Guckert erhielt ich die Möglichkeit, in dem Forschungsprojekt PIMAR mitzuarbeiten und im Rahmen dieses Projektes meine Abschlussthesis zu schreiben.

Natalie Göpfert, Johanna und Hartmut Arlt sowie Daniela Dormehl danke ich zusätzlich noch ganz herzlich für die Zeit, die sie sich genommen haben, um diese Arbeit Korrektur zu lesen.

Selbstständigkeitserklärung

Hiermit versichere ich, dass ich die eingereichte Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die von mir angegebenen Quellen und Hilfsmittel nicht benutzt und die den benutzten Werken wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Friedberg, April 2015

Hans Christian Arlt

Inhaltsverzeichnis

Danksagung i

Selbstständigkeitserklärung iii

Inhaltsverzeichnis v

Abbildungsverzeichnis vii

Listings ix

Tabellenverzeichnis xi

1 Einleitung 1
- 1.1 Motivation . . . 1
- 1.2 Problemstellung und Zielsetzung . . . 4
- 1.3 Organisation der Arbeit . . . 5
- 1.4 Was nicht behandelt wird . . . 5
- 1.5 Zusammenfassung der wichtigsten Ergebnisse . . . 6

2 Stand der Technik 9
- 2.1 Einleitung . . . 9
- 2.2 Augmented Reality-Anwendungen . . . 10
  - 2.2.1 ProSieben Galileo Zeitreise . . . 11
  - 2.2.2 LEGO Verpackung . . . 12
  - 2.2.3 Ingress . . . 12
  - 2.2.4 Head-up-Displays . . . 12
  - 2.2.5 Arvika . . . 14
  - 2.2.6 Wikitude App . . . 15
- 2.3 Agumented Reality-Entwicklungsumgebungen . . . 15
  - 2.3.1 Qualcomm® Vuforia™ . . . 16
    - Die Vuforia™-Tracking Möglichkeiten . . . 17
  - 2.3.2 Metaio . . . 18
    - Die Metaio Tracking Möglichkeiten . . . 19
- 2.4 Positions- und Orientierungsbestimmung für AR-Tracking . . . 20

2.4.1 Initialisierung durch visuelle Marker . . . 21
2.4.2 Drahtlose Innenraum-Positionierung . . . 21
RFID . . . 22
NFC . . . 23
BLE . . . 24
Wireless-LAN . . . 24
2.4.3 Kombination von GPS und visuellem Tracking . . . 25
2.4.4 Image Matching-Verfahren . . . 26
2.4.5 Vereinfachtes SIFT-Verfahren . . . 27
2.5 Zusammenfassung . . . 27

3 Konzeptentwicklung **29**
3.1 Einleitung . . . 29
3.2 Mögliche Anwendungsszenarien . . . 29
3.2.1 Werkstattbetrieb . . . 30
3.2.2 Tourismus / Stadtführung . . . 31
3.2.3 Museumsbetrieb . . . 32
3.3 Anforderungen . . . 33
3.4 Sensorik . . . 35
3.5 Entwicklung des Verfahrens TBOC . . . 36
3.5.1 Statistische Auswertung . . . 38
3.5.2 Bayes-Klassifikator . . . 39
Bayes-Theorem . . . 40
Naive-Bayes-Klassifikator . . . 41
3.6 Zusammenfassung . . . 42

4 Anwendungsentwicklung **45**
4.1 Einleitung . . . 45
4.2 Android und Eclipse . . . 46
4.3 Metaio SDK . . . 51
4.4 WEKA for Android . . . 54
4.4.1 Dataset . . . 54
Die @relation Deklaration . . . 55
Die @attribute Deklaration . . . 55
Die @data Deklaration . . . 56
4.4.2 Classifier . . . 56
4.4.3 Datenverwaltung . . . 57
Generierung eines Datensatzes . . . 59
4.5 Die Bildanalyse-Verfahren . . . 60
4.5.1 Minimal-Histogramme . . . 60
4.5.2 Regionale RGB-Mittelwerte . . . 62
4.6 Aufbau der Android Anwendung . . . 65
4.6.1 Die Anlern-Phase . . . 67
Programmatische Umsetzung . . . 68

4.6.2 Die Erkennungs-Phase . . . 71
Programmatische Umsetzung . . . 72
4.7 Weitere Systemfunktionen . . . 75
4.7.1 System-Einstellungen . . . 75
4.7.2 Trainingsdatenverwaltung . . . 78
4.8 Zusammenfassung . . . 80

5 Prototypische Implementierung in eine reale Anwendung 83
5.1 Die Mathematikum App . . . 83
5.1.1 Pi mit den Füßen . . . 84
Aufbereitung für markerloses AR-Tracking . . . 84
Animation und Konvertierung des 3D-Modells . . . 85
Integration in die Android Activity . . . 87
5.1.2 Die Deutschlandtour . . . 88
Umsetzung des Szenarios . . . 89
5.1.3 Knack den Code . . . 90
5.2 Zusammenfassung . . . 91

6 Test und Auswertung des TBOC-Verfahrens 93
6.1 Einleitung . . . 93
6.2 Performance des TBOC-Verfahrens . . . 94
6.2.1 Auswirkungen auf die Dateigröße der Trainingsdaten . . . 94
6.2.2 Auswirkungen auf das Modell . . . 95
Auswirkungen auf die Dateigröße und die Erstellungszeit . . . 95
Auswirkungen auf die Erkennungszeit . . . 97
6.2.3 Statistische Hypothesenprüfung in Bezug auf das Datenmodell . . . 98
6.3 Ermittlung der Erkennungsraten . . . 102
6.3.1 Statisches und dynamisches Anlernen im Vergleich . . . 102
6.3.2 Erkennungsraten in Bezug zu den Trainingsdaten . . . 104
6.3.3 Erkennungsraten in Bezug auf die Objektanzahl . . . 105
6.3.4 Statistische Hypothesenprüfung in Bezug auf die Erkennungsrate . . . 107
6.3.5 Auswirkung des NULL-Objektes . . . 109
6.3.6 Optimierung durch Anpassung der Trainingsdatenanzahl . . . 110
6.4 Optimierung bei der Bildverarbeitung . . . 111
6.4.1 onPictureTaken() . . . 111
6.4.2 onPreviewFrame() . . . 111
6.4.3 TextureView.getBitmap() . . . 111
6.5 Anwendertest . . . 112
6.6 Zusammenfassung . . . 112

7 Zusammenfassung, Ergebnisse und Ausblick 115
7.1 Ausblick . . . 118

A TrackingData_MarkerlessFast.xml 119

B TBOC-Klassendiagramm **121**

Glossar **124**

Literaturverzeichnis **129**

Abbildungsverzeichnis

1.1 Beispiel für Augmented Reality-Anwendungen 1
1.2 Markerless 3D-Tracking 3
1.3 3D Punktwolken 4

2.1 Reality-Virtuality Continuum 10
2.2 Augmented Reality Anwendungen Pro7 und LEGO 11
2.3 Augmented Reality Anwendungen Ingress und Head-up-Display 13
2.4 Augmented Reality Anwendungen Arvika und Wikitude - The App 14
2.5 Augmented Reality SDK Liste 16
2.6 Funktionsweise von AREL 18
2.7 Initialisierung durch Marker und NFC Technologie 22
2.8 Ortsbestimmung mit BLE sowie Referenzpunkte für WLAN Fingerprinting . . . 25
2.9 Kombination aus GPS / visuellem Tracking sowie visuelles Tracking im Stadtumfeld 26

3.1 Anwendungsszenario Werkstatt Betrieb 30
3.2 Anwendungsszenario Tourismus / Stadtführung 31
3.3 Anwendungsszenario Museumsbetrieb 33
3.4 Overview vom TBOC Verfahren 35
3.5 Verwendete Sensorik 36
3.6 Detailansicht des TBOC Verfahrens 37

4.1 Android-Projektdaten-Übersicht 46
4.2 Android Activity Lebenszyklus 48
4.3 Android-Layout-Editor in Eclipse 49
4.4 Minimal-Histogramme 61
4.5 Ergebnisse “NaiveSimilarity” Vergleich 63
4.6 RegionaleRGB-Mittelwerte 64
4.7 Packagediagramm der TBOC-Anwendung 65
4.8 Klassendiagramm 65
4.9 Die GUI der MainActivity 66
4.10 Activity zum Objekt anlernen 67
4.11 Activity zum Erkennen von Objekten 72
4.12 Systemeinstellungen der Anwendung 77
4.13 Trainingsdaten Verwaltung in den Systemeinstellungen 78

5.1 Aufzeichnung von 3D-Tracking-Punktwolken 85
5.2 Animation der 3D-Modelle und Konvertierung dieser für das Metaio SDK . . . 86
5.3 Das Exponat "Pi mit den Füßen" 87
5.4 Das Exponat "Die Deutschlandtour" 89
5.5 Das Exponat "Knack den Code" 90

6.1 Abhängigkeit der Dateigröße zur Anzahl der Trainingsdaten 94
6.2 Einfluss der Datensätze auf die benötigten Zeit bei der Modell Erzeugung . . . 96
6.3 Einfluss der Objektanzahl auf die Modellgenerierung 96
6.4 Einfluss der Anzahl der Datensätze auf die Erkennungszeit 97
6.5 Einfluss der Objektanzahl auf die Erkennungszeit 98
6.6 Boxplott und Einzelwerte Erstellungszeiten 99
6.7 Einzelwerte im XY-Plott Erkennungszeit 101
6.8 Statisches und dynamisches Anlernen von Objekten 102
6.9 Erkennungsraten beim statischen und dynamischen Anlernen und Erkennen . . 103
6.10 Erkennungsraten bei Zunahme der Trainingsdaten 104
6.11 Übersicht der Exponate für den Erkennungsraten-Test bei Objekterhöhung . . . 105
6.12 Verlauf der Erkennungsraten im Verhältnis zur Erweiterung der Objektanzahl . 106
6.13 Erkennungsraten der Einzelnen Exponate bei Objekterweiterung 107
6.14 XY-Plott der Einzelwerte der Erkennungsrate bei Erhöhung der Datensätze . . 109
6.15 Auswirkung des NULL-Objektes auf die Erekennung 110

Listings

4.1 AndroidManifest.xml . . . 47
4.2 Android Activity . . . 49
4.3 Layout.xml . . . 50
4.4 Activity-Aufruf über ein Intent . . . 51
4.5 Hinterlegen der metaioSDKSignature . . . 51
4.6 Aufbau einer AR-Activity . . . 52
4.7 setTrackingConfiguration . . . 52
4.8 createGeometry . . . 53
4.9 Anpassungen an dem Model . . . 53
4.10 setRelativeToScreen . . . 53
4.11 WEKA ARFF-Datei . . . 55
4.12 Anlegen eines Klassifikators . . . 57
4.13 Klassifikator-Modell erstellen . . . 57
4.14 Klassifizieren einer Instanz . . . 57
4.15 Laden von Datensätzen . . . 58
4.16 Explizites Laden einer ARFF-Datei . . . 58
4.17 Speichern von Datensätzen . . . 58
4.18 Explizites Speichern einer ARFF-Datei . . . 59
4.19 Erstellen eines Datensatzes im Speicher . . . 59
4.20 Hinzufügen von Daten in einen Datensatzes . . . 60
4.21 Aufruf der getSimpleHist Methode . . . 61
4.22 Sicherstellung einer Korrekten Balkenbreite im Histogramm . . . 62
4.23 Erstellen eines Farbhistogramms mit dynamischer Balkenanzahl . . . 62
4.24 Aufruf der calcSignatureRGB Methode . . . 63
4.25 Erstellen der Farbsignatur über Regionale Mittelwerte . . . 64
4.26 Laden der SharedPreferences . . . 66
4.27 TimerTask als Thread zum sammeln der Sensordaten . . . 68
4.28 takePicture() Callback-Methode . . . 69
4.29 setOneShotPreviewCallback() . . . 69
4.30 Bitmap aus der TexturView auslesne . . . 70
4.31 Das Speichern des Datensatzes . . . 71
4.32 Zusammensetzten zweier Datensatzes . . . 71
4.33 Das Erzeugen eines Modells über die ObjectClassifier-Klasse . . . 73
4.34 Pausieren eines Threads . . . 73

4.35 Objekt Klassifikation 74
4.36 Ausgabe des Benutzerhinweises für Erkanntes Objekt 74
4.37 Deklaration der PreferenceScreen-XML 76
4.38 Änderung der Einstellungsbeschreibung bei Änderung einer Auswahl 77
4.39 Füllen der Datensatz-Auswahllsite 79
4.40 Wechseln der View üb der den ViewSwitcher 79
4.41 Ändern der SharedPreferences über den SharedPreferences.Editor 80
5.1 Start der Simulation “Pi mit den Füßen” 88
5.2 Umwandeln von Bildern zu einem IGeometry Objekt 89

Tabellenverzeichnis

6.1 Auswertung der Messwerte der Erstellungszeit bei Erhöhung der Objekte 99
6.2 Korrelationskoeffizienten Erstellungszeit-Test 100
6.3 Test auf Korrelation bei den Erstellungszeiten 100
6.4 Test auf Korrelation bei den Erkennungszeiten 101
6.5 Zusammengezählte Erkennungsraten bei Datensatzerhöhung 108
6.6 Pearson Chi-Square Testergebnisse 108

Kapitel 1

Einleitung

1.1 Motivation

Die große Menge an digitalen Informationen, die heute verfügbar ist, macht es immer bedeutsamer, dass Inhalte schnell, übersichtlich und effizient zur richtigen Zeit angezeigt werden können. Dies gilt gleichermaßen für viele Bereiche der Industrie, den Handel, das Bildungssystem sowie das öffentliche Leben. Beispielhaft erwähnt sei hier nur die Schulung von Mitarbeitern, die Präsentation neuer Produkte, die räumliche Darstellung von Planungsvarianten oder die umfassende Information der Besucher in einem Museumsbetrieb. In der Regel werden Informationen weitestgehend mit Hilfe klassischer Darstellungsmethoden und Materialien wie Filmen, Plakaten, Texten etc. vermittelt. Eine innovative Möglichkeit bietet hier nun die Erweiterte Realität (engl.: Augmented Reality, kurz: AR). Mit ihrer Hilfe können Informationen auf eine völlig neue Art und Weise genau da angezeigt werden, wo sie benötigt werden: im Blickfeld des Betrachters. [CL08] Durch die große Verbreitung

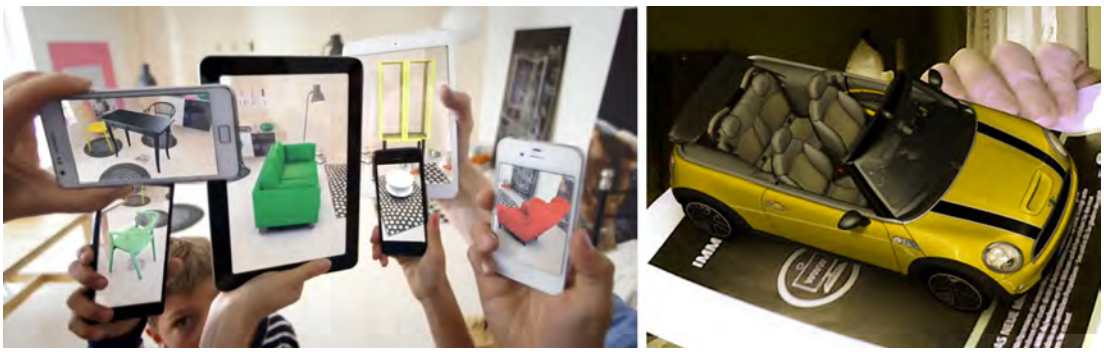


Abbildung 1.1: Zwei AR-Anwendungen von großen Marken als Beispiel: Der interaktive IKEA-Katalog, der es ermöglicht Möbel in die eigene Wohnung zu platzieren (links, http://goo.gl/Nxby2v Stand: 7. Januar 2015) und eine AR-Werbeanzeige von Mini Cooper. Auf einem Werbeflyer lässt sich mittels eines mobilen Telefons oder einer Webcam ein 3D-Modell des Mini Cooper anzeigen (rechts, http://goo.gl/x33rcU Stand: 7. Januar 2015).

von "Smartphones", der neuen Generation von mobilen Telefonen, gewinnt die Erweiterte Realität in letzter Zeit einen immer größeren Anwendungsbereich und Bekanntheitsgrad. Viele große Firmen wie LEGO, IKEA, McDonald's oder Audi greifen immer häufiger auf AR für ihr Produktmarketing zurück.[1] Bei Augmented Reality-Anwendungen werden mittels einer Kamera reale Objekte erfasst, über einen entsprechenden Tracker identifiziert und anschließend mit computergenerierten Zusatzinformationen überlagert [Oeh04], [MBS14]. Diese können als virtuelle Daten in Form von Texten, Bildern, Videos oder auch 3D-Modellen zur Verfügung gestellt werden (siehe Abbildung 1.1). Die virtuellen Objekte werden räumlich korrekt positioniert in das Bild eingefügt und ergänzen damit das reale Bild. So verschmilzt die digitale Information mit der realen Umwelt des Betrachters und ermöglicht dadurch das Anzeigen der Informationen an der Stelle, an der sie gebraucht werden. [CL08] Da die heute verfügbaren mobilen Telefone neben einer qualitativ hochwertigen Kamera auch mehrere Sensorbauteile wie GPS und Richtungsanzeige enthalten, ist die AR-Anwendung auch für den privaten Nutzer realisierbar. Schon heute sind außergewöhnliche und eindrucksvolle Produktpräsentationen verfügbar (siehe Abbildung 1.1). Ein zukünftiges Anwendungsfeld, das in dieser Arbeit auch näher betrachtet wird, liegt in der erweiterten Verfügbarkeit von Informationen beim Museumsbesuch. Hier kann z.B. der Museumsbesucher ohne ein ausgeliehenes spezielles Gerät, wie heute noch üblich, umfangreiche Informationen zu den einzelnen Exponaten auf seinem Mobiltelefon abrufen.

Es gibt eine Vielzahl von unterschiedlichen Methoden, um ein AR-Tracking zu ermöglichen. Zur Zeit am meisten verbreitet sind die, die auf Codemarker (z.B. QR-Codes) oder einfache Bildmarker zurückgreifen, da sie einfach und schnell zu verarbeiten sind. Aufwendiger sind hingegen Methoden, die auf markerloses Tracking setzen. Hierfür stehen entweder markerlose 2D-Marker (Ausschnitte aus Bildern etc.) oder markerlose 3D-Marker zur Verfügung. Die markerlosen 3D-Marker bieten den enorm großen Vorteil, dass reale Objekte oder Räume direkt getrackt werden können und keinerlei weitere Marker notwendig sind, was eine größere Flexibilität und Bewegungsfreiheit beim Trackingvorgang gewährleistet. Hierbei sind die Struktur des Bildmarkers oder die Beschaffenheit der Form des Objektes relevant, die bei der Erkennung mit dem aktuellen Kamerabild abgeglichen werden. Die Anwendung versucht beim Vergleich bestimmte Merkmale zu erkennen.[2] Beim 3D-Markerless-Verfahren werden 3D-Punktwolken mit Tiefeninformationen benutzt, um Objekte oder Räumlichkeiten zu erkennen und zu verfolgen (siehe Abbildung 1.2). Jedoch kommen die zur Zeit aktuellen Modelle der mobilen Geräte durch ihre begrenzte Prozessorleistung und limitierten Arbeitsspeicher an ihre Grenzen, wenn es darum geht, eine Vielzahl von aufwendigen und großen AR-Trackingdaten mit dem aktuellen Kamerabild in Echtzeit zu vergleichen. Gerade bei den markerlosen Tracking-Systemen, die mit sehr anspruchsvollen Verfahren arbeiten, um das Tracking zu ermöglichen, sind diese technischen Grenzen der mobilen Devices schnell erreicht.

[1] http://goo.gl/y8KaVs Stand: 7. Januar 2015
[2] http://goo.gl/fQsl7b Stand: 7. Januar 2015

Diese Arbeit entsteht im Rahmen des Forschungsprojektes PIMAR[3] - "Platform Independent Mobile Augmented Reality", einem Forschungsprojekt, welches von der Hessen Agentur - HA Hessen Agentur GmbH[4] und LOEWE - Exzellente Forschung für Hessens Zukunft[5] gefördert wird. Das Projektkonsortium setzt sich aus dem Fachbereich Mathematik, Naturwissenschaften und Datenverarbeitung (MND) der Technischen Hochschule Mittelhessen (THM)[6], dem Fachbereich Mathematik und Informatik der Philipps-Universität Marburg[7] und dem Industriepartner advenco Consulting GmbH[8] zusammen. Die Projektziele sind unter anderem die Schaffung einer Infrastruktur für die plattformunabhängige Entwicklung von Applikationen für mobile Endgeräte über modellgetriebene Softwareentwicklung sowie die Integration von AR-Technologie. Der Augmented Reality-Schwerpunkt liegt hierbei auf dem markerlosen 3D-Tracking. Aus den Anwendungsszenarien des Industriepartners ergibt sich die Anforderung, eine Vielzahl von Objekten automatisch zu erkennen und passende AR-Inhalte zu laden. Deshalb soll in dieser Arbeit ein Verfahren entwickelt werden, welches die Leistungsbeschränkung der mobilen Geräte umgeht und es ermöglicht, eine große Menge von Objekten markerlos zu tracken.

Abbildung 1.2: Markerless 3D-Tracking: Angelernte Punktwolken mit Tiefeninformationen (links), Tracking und Anzeige eines virtuellen Objektes anhand des angelernten Punktwolkenmarkers (rechts).

[3] http://pimar.mnd.thm.de Stand: 7. Januar 2015
[4] http://www.hessen-agentur.de/ Stand: 7. Januar 2015
[5] http://www.proloewe.de Stand: 7. Januar 2015
[6] http://galileo.mnd.th-mittelhessen.de Stand: 7. Januar 2015
[7] https://www.uni-marburg.de/fb12 Stand: 7. Januar 2015
[8] http://www.advenco.de Stand: 7. Januar 2015

1.2 Problemstellung und Zielsetzung

Um bei den markerlosen Verfahren eine Vielzahl von unterschiedlichen Objekten gleichzeitig zu erkennen, werden in der Regel QR-Codes zur Initialisierung verwendet.[9] In alternativen Verfahren muss der Anwender für jedes einzelne Objekt die entsprechende Anwendung über ein Benutzermenü auswählen. Der Grund hierfür liegt in der Komplexität bei der Erkennung von großen Punktwolken. Für jedes potentielle Objekt das erkannt werden soll, muss permanent eine angelernte Punktwolke im Gerätespeicher gehalten und wiederholt mit den aktuellen Kamerabildern abgeglichen werden. Damit kommen die mobilen Geräte sehr schnell an ihre Leistungsgrenze, da die großen Datenmengen hier nicht nur entsprechend viel Speicher, sondern insbesondere auch eine hohe Prozessorlast erfordern. Erst wenn vom System eine der Punktwolken verifiziert wurde, können die anderen Punktwolken verworfen und die Verfolgung der Szene sowie deren virtuelle Überlagerung gestartet werden (siehe Abbildung 1.3). Aus diesem Grund begrenzen die unterschiedlichen Entwicklungsumgebungen für AR die Anzahl der Objekte, die gleichzeitig markerlos erkannt werden können. In dem SDK von metaio[10], welches in dieser Arbeit als Grundlage für AR-Anwendungen dienen soll, ist die maximale Anzahl nach eigenen Angaben auf 15 limitiert. Jedoch machen sich schon kleinere Anzahlen sehr stark in einer sinkenden Framerate der Anzeige beim Tracking bemerkbar.

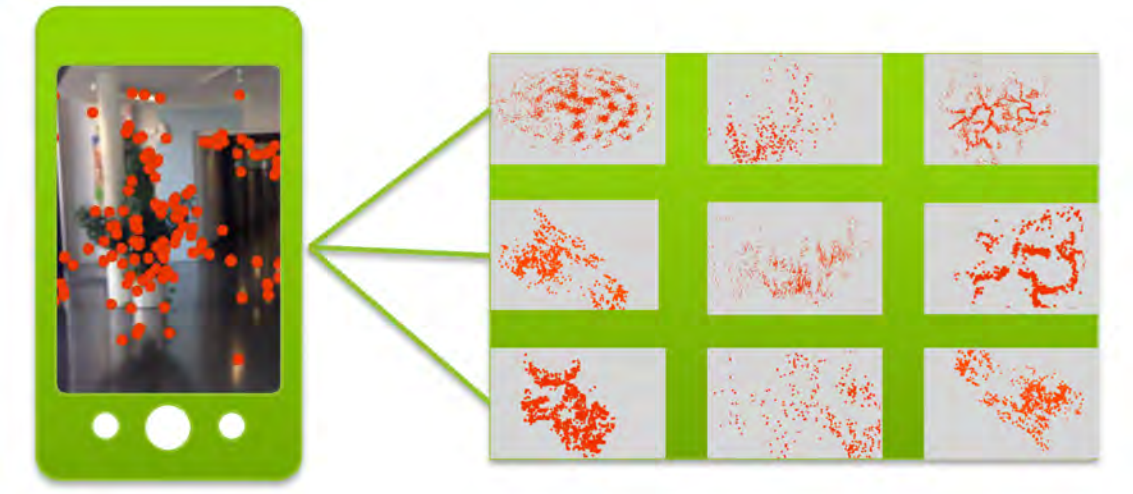


Abbildung 1.3: Gleichzeitiger Abgleich mehrerer 3D-Punktwolken. Das System muss alle Punktwolken im Speicher halten und diese mit dem aktuellen Kamerabild abgleichen.

Um dieses Problem zu lösen, soll in dieser Arbeit ein Verfahren entwickelt werden, dass mit der vorhandenen Technik von mobilen Geräten eine automatische Initialisierung vor das eigentliche AR-Tracking setzt. So soll die Möglichkeit geschaffen werden eine

[9] http://goo.gl/DNqJT8 Stand: 7. Januar 2015
[10] http://www.metaio.com Stand: 7. Januar 2015

Vielzahl von ortsgebundenen, unterschiedlichen, markerlosen Tracking-Szenarien zu laden. Dies soll ohne weitere Benutzereinwirkung oder Initialisierung über angebrachte Marker geschehen. Der Benutzer soll von dieser Vorentscheidung nichts bemerken. Es soll das Gefühl vermittelt werden, dass allein durch das Ausrichten der Kamera auf ein Objekt, dieses erkannt und ein entsprechendes AR-Szenario gestartet wird. Das System soll intuitiv und ohne weitere Einweisung nutzbar sein. Ebenfalls sollen die zu erkennenden Objekte für eine erfolgreiche Erkennung nicht in irgendeiner Form verändert werden müssen. Das Verfahren soll in verschiedenen mobilen prototypischen Applikationen umgesetzt werden und auf Leistungsfähigkeit, Benutzerfreundlichkeit und Effizienz getestet werden. Der Praxistest soll mit einem Prototypen in einer Applikation für das mathematische Mitmach-Museum "Mathematikum" realisiert werden. Hier sollen unterschiedliche Exponate im Museum erkannt und daraufhin die jeweils passenden AR-Szenen geladen werden.

1.3 Organisation der Arbeit

In Kapitel 2 werden zuerst die Technologien behandelt, mit denen Augmented Reality-Anwendungen erstellt werden können und erläutert, wie die aktuellen Produkte auf dem Markt einzuordnen sind. Weiter wird beschrieben, welche Möglichkeiten für eine Positions- und Orientierungsbestimmung für AR-Tracking existieren. In diesem Zusammenhang erfolgt eine Übersicht über vorliegende Forschungsarbeiten. In Kapitel 3 wird ein Konzept entwickelt, welches ein Verfahren zur automatischen Initialisierung für AR- Szenen umsetzt. Hierbei werden die möglichen Anwendungsszenarien beschrieben und die einzelnen Verfahrensbestandteile genauer erklärt. In Kapitel 4 wird die Entwicklung des Prototypen beschrieben. Hier spielt das Data Mining-Paket Weka, welches für die statistische Wahrscheinlichkeitsrechnung herangezogen wird, eine große Rolle. Weiter wird die Umsetzung der beiden Phasen, die das Verfahren bilden, genauer beschrieben. Das Kapitel 5 beschreibt die umgesetzte Anwendung für das mathematische Mitmach-Museum "Mathematikum"[11]. Hierbei werden die einzelnen AR-Szenarien näher beleuchtet. Das Kapitel 6 beschäftigt sich mit Tests des Verfahrens sowie dessen Evaluation und Auswertung. Die Ergebnisse der Arbeit werden anschließend in Kapitel **??** dargestellt und bewertet. Abschließend folgt in Kapitel 7 die Zusammenfassung der gesamten Arbeit sowie der Ausblick auf weitere mögliche Arbeiten / Forschungen, die in Zusammenhang mit dem in dieser Arbeit behandelten Forschungsbereich stehen, auf die aber im Rahmen der hier vorliegenden Arbeit nicht näher eingegangen werden konnte.

1.4 Was nicht behandelt wird

Diese Masterarbeit befasst sich mit der Entwicklung eines Verfahrens zur Initialisierung von AR-Szenarien. Hierbei wird auf den aktuellen Forschungsstand eingegangen und ein Konzept für das Verfahren entwickelt und in Form von mehreren Prototypen exemplarisch umgesetzt. Somit liegt der Schwerpunkt dieser Arbeit auf dem Verfahren und den anschließenden Tests

[11] www.mathematikum.de Stand: 8. Januar 2015

sowie deren Auswertung. Auf die Umsetzung von AR-Szenarien wird nur kurz eingegangen, indem die umgesetzten Szenarien für eine reale Museumsanwendung beschrieben werden. Auch die Integration, welche die modellgetriebene Softwareentwicklung beinhaltet, wird in dieser Arbeit nicht weiter beleuchtet. In einigen Bereichen wird der Java-Quelltext eingebunden. Tiefergehende Erläuterungen des Quelltextes werden in der Masterarbeit nicht getätigt, sondern auch hier wird lediglich auf die Funktionalität und die Zusammenhänge eingegangen.

1.5 Zusammenfassung der wichtigsten Ergebnisse

Durch die Flut an digitalen Daten kommt der Wunsch auf, Informationen schnell, übersichtlich und effizient zur richtigen Zeit am richtigen Ort zu erhalten. Das unter dem Begriff "Time-to-Content" zusammengefasste Prinzip umfasst unter anderem auch das Überblenden der Wirklichkeit mit virtuellen Informationen (AR). Bei Augmented Reality können neben einfachen Überlagerungen eines Kamerabildes auch Informationen räumlich im Blickfeld positioniert werden. Hierbei können einfache Marker wie QR-Codes oder Bilder für das sogenannte AR-Tracking verwendet werden, aber es besteht auch die Möglichkeit, reale Objekte für das Tracking zu verwenden. Hier müssen von den Objekten sogenannte 3D-Punktwolken erstellt werden, die markante Punkte des Objektes im räumlichen Verhältnis erfassen und abspeichern. Anhand dieser Punktwolken können die Objekte erkannt und verfolgt werden, was es ermöglicht, Informationen in Form von Texten, Bildern, Videos und 3D-Objekten an diesen Objekten zu platzieren. Jedoch ist das markerlose Erkennen von Objekten sehr rechenintensiv und schon bei gleichzeitigem Abtasten mehrerer Objekte kommt es zu Leistungseinbrüchen. Um eine Vorerkennung der Objekte zu ermöglichen, wurde in dieser Arbeit ein Verfahren entwickelt, welches in der Lage ist, Objekte in Innenräumen oder in Außenbereichen zu erkennen und gezielt die benötigten AR-Daten zu laden. So muss die Initialisierung nicht über Marker oder Benutzereinwirkung geschehen.
Anhand von Anwendungsszenarien (Werkstattbetrieb, Museumsbetrieb und Tourismus / Stadtführung) des Industriepartners des Projektes PIMAR wurden im Konzept die Anforderungen des Verfahrens festgelegt:

- Keine Verwendung von visuellen Markern
- Keine Anschaffung / Installation von weiterer Hardware (WLAN, iBeacon, etc.)
- Keine Nutzung von Datenaustausch über WLAN / Mobilfunknetz
- Nutzung der vorhandenen Sensorik und der Kamera der mobilen Devices
- Datenverarbeitung lokal direkt auf dem Gerät
- Kein großer Datenspeicherverbrauch

Um die Datenlast auf dem mobilen Gerät gering zu halten, werden die von der Kamera aufgezeichneten Bilder nicht gespeichert. Vielmehr werden die Bilder über zwei Bildanalyse-Algorithmen untersucht und die so gewonnene Informationen über die im Bild vorhandenen Farbwerte und deren regionaler Verteilung mit den restlichen Sensorwerten (Magnetometer,

Gyrometer und eventuell GPS) in einen Parameterraum transformiert und im System verarbeitet. Über eine statistische Wahrscheinlichkeitsverteilung bestimmt das TBOC-Verfahren das zu erkennende Objekt, indem es einen Featurevektor mit einem zuvor angelernten Datenmodell vergleicht. Hierfür verwendet das Verfahren einen Bayes-Klassikator. Dieser setzt voraus, dass das System in zwei Phasen unterteilt ist. In der ersten Phase, der sogenannten `Anlernphase`, werden die gesammelten Daten in Form des Featurevektor erfasst und über eine Trainingsdatendatei (ARFF-Datei) einem Objekt zugeordnet. Für die zweite Phase, die `Erkennungsphase`, wird aus den Trainingsdaten ein Datenmodell generiert. Die für die Erkennung gesammelten Bild- und Sensorwerte werden ebenfalls als Featurevektor übergeben und mit dem Modell verglichen und mit einer bedingten Wahrscheinlichkeit einem der angelernten Objekte zugeordnet.
Im Rahmen dieser Arbeit wurde eine mobile Entwickler-Anwendung implementiert, die das TBOC-Verfahren repräsentiert. Mit Hilfe dieser Anwendung können Objekte mit Trainingsdaten angelernt und erkannt werden. Es stehen zahlreiche Funktionen zur Verwaltung der Trainingsdaten zur Verfügung sowie Funktionen, die das Arbeiten und Testen mit dem Verfahren erleichtern. Des Weiteren wurde eine prototypische Integration des Verfahrens in die mobile Museums-Anwendung des Mathematikums in Gießen beschrieben. In dieser Anwendung wird exemplarisch gezeigt, wie das Verfahren in einer mobilen Anwendung für die Initialisierung von komplexen AR-Szenarien verwendet werden kann. In der Anwendung können drei Exponate des Museums erkannt sowie bei zwei dieser Exponate eine AR-Szene geladen werden. Beim dritten Exponat wird anhand des Ladens eines Mini-Spieles passend zu dem Exponat gezeigt, dass dieses Verfahren nicht nur zur Initialisierung von AR genutzt werde kann. Es ist möglich jeden beliebigen Inhalt einer Anwendung ortsgebunden zu laden und somit dem Anwender Informationen passend zu seiner Position anzuzeigen.
Zum Abschluss dieser Arbeit wurden unterschiedliche Test für die Funktionalität und Leistung des Verfahrens gemacht. Hierbei wurde der Datenspeicherverbrauch sowie die Zeit, die für die Erstellung des Datenmodells benötigt wird, betrachtet sowie die Auswertung der Erkennungsraten und der Zeit, die für eine Erkennung benötigt wird.
Die Ergebnisse dieser Tests zeigen, dass die gesetzten Ziele durch die Entwicklung des TBOC-Verfahrens umgesetzt werden konnten. Die Erkennungsraten von im Schnitt 80% bei 18 Objekten, kann durch gezieltes Anlernen auch noch verbessert werden. Auch der Speicherplatz im Dateisystem des mobilen Gerätes überschreitet die 1mb Grenze nur in Ausnahmefällen.

Kapitel 2

Stand der Technik

In diesem Kapitel wird kurz auf den Begriff Augmented Reality eingegangen und neben den unterschiedlichen Entwicklungsumgebungen werden auch die verschiedenen Tracking-Methoden und deren Anwendungsgebiete näher beleuchtet. Ein besonderes Augenmerk wird auf die Möglichkeit von Ortsbestimmung und Objekterkennung zur Vorentscheidung für AR-Szenarien gelegt. Hierbei werden aktuelle Forschungsarbeiten vorgestellt und die unterschiedlichen Ansätze miteinander verglichen.

2.1 Einleitung

Erweiterte Realität (engl.: Augmented Reality, kurz AR), ist eine innovative Art der Mensch-Computer-Interaktion. Gerade bei der Überflutung an digitalen Informationen in unserem Alltag gewinnt "Time-to-Content" (der schnelle Zugriff auf die gewünschten Informationen zur richtigen Zeit) an Bedeutung. Im Gegensatz zu Virtual Reality (VR), bei der es um die Wahrnehmung der Wirklichkeit und ihrer physikalischen Eigenschaften in einer in Echtzeit computergenerierten Realität geht, und bei der die reale Umwelt ausgeblendet wird, ist das Ziel bei Augmented Reality die Anreicherung der bestehenden, realen Welt mit computergenerierten Zusatzobjekten. Hierbei werden keine gänzlich neuen Welten erschaffen. Die vorhandene Wirklichkeit wird stattdessen mit einer virtuellen Realität erweitert. [KM09] Der Oberbegriff, der die Vermischung von realer und virtueller Welt miteinander beschreibt, lautet Mixed Reality (MR). Die Abgrenzung zwischen Realität und Virtualität kann nach Paul Milgram (1994) auf ein Kontinuum (Reality-Virtuality Continuum) aufgetragen werden (siehe Abbildung 2.1), welches ebenso die Einordnung der Augmented Reality sowie auch der Augmented Virtuality darstellt. [MTUK95]
AR stellt eine innovative Alternative zur Verfügung, Informationen auf eine neue Art und Weise im Blickfeld des Betrachters zu präsentieren. Mit AR ist eine Vielzahl von neuen Anwendungen möglich, bei denen der Nutzen hauptsächlich in der Verschmelzung mit der Realität liegt. Hierbei stehen nicht nur Anwendungen für den Entertainmentbereich im Vordergrund, da es auch in der industriellen Produktion, im Verkehrsbereich oder bei Präsentationen von großem Nutzen ist, wenn Informationen gezielt dort angezeigt werden können, wo sie von Interesse sind. [CL05] Google plant mit der Einführung des Projektes "Google Glass"

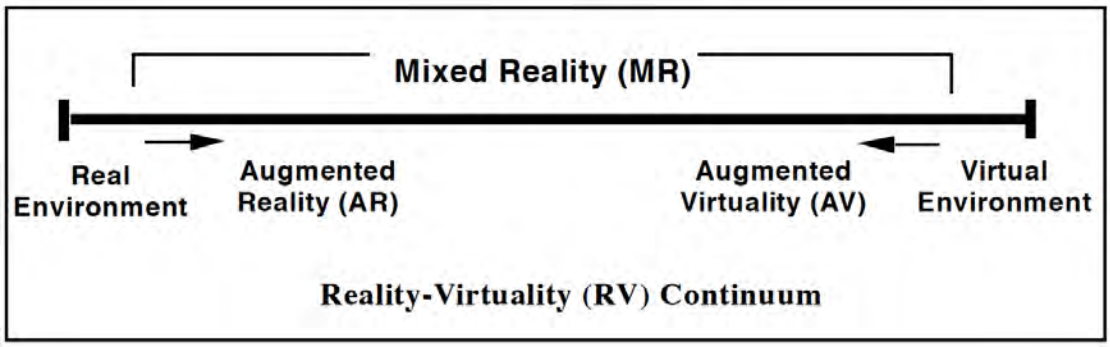


Abbildung 2.1: Die Abgrenzung zwischen Realität und Virtualität wird als Reality-Virtuality Continuum bezeichnet [MTUK95].

sogar die Platzierung von Augemented Reality im alltäglichen Leben. Die Funktionen z.B. eines Smartphones werden über das Mini-Display direkt im Blickfeld des Nutzers angezeigt. Hierbei ist zu erkennen, dass in der Kombination mit mobilen Technologien AR in naher Zukunft immer mehr an Relevanz gewinnen wird.
Die einfachste Art von Erweiterter Realität ist das statische Einblenden von Text-, Bild-, und / oder Videoinformationen auf dem Kamerabild. Die nächste Variante, die auch in der Öffentlichkeit am meisten mit AR in Verbindung gebracht wird, ist die Überlagerung von Text-, Bild-, und oder Videoinformationen oder sogar von 3D-Objekten auf einem Marker, der von der Kamera getrackt wird. Ein Marker kann in diesem Zusammenhang ein Codemarker (z.B. QR-Code) oder ein Bildmarker sein, oder das Tracking erfolgt über das sogenannte markerlose Tracking von Bildern (2D-markerless Tracking) oder von dreidimensionalen Objekten (3D-markerless Tracking). Mit Hilfe der GPS- und Gyrosensoren von mobilen Geräten ist eine Orts- bzw. auf die Orientierung bezogene Einblendung ebenso möglich. In diesem Kapitel soll der aktuelle Markt von AR-Anwendungen sowie die zur Verfügung stehenden Entwicklungsumgebungen vorgestellt und ausgewertet werden. Ein weiterer wichtiger Punkt wird die Möglichkeit der automatischen Initialisierung von AR-Szenarien innerhalb einer AR-Anwendung sein. Hier werden aktuelle Forschungsarbeiten und Methoden vorgestellt, die in der Lage sind, ortsgebundene AR-Inhalte zu laden. Wobei hier meistens zwischen Innenräumen und Außenbereichen unterschieden wird und dementsprechend verschiedene Ansätze vorliegen.

2.2 Augmented Reality-Anwendungen

Um einen Überblick über aktuelle AR-Anwendungen auf dem Markt zu geben, werden hier aus unterschiedlichen Anwendungsbereichen einzelne Anwendungen vorgestellt und deren Nutzen diskutiert. Hierbei steht die Fragestellung im Vordergrund, ob diese Anwendungen eher als Technik-Eye-Catcher dienen oder ob bereits ernstzunehmende Applikationen mit einem nachgewiesenen Mehrwert für den Anwender existieren.

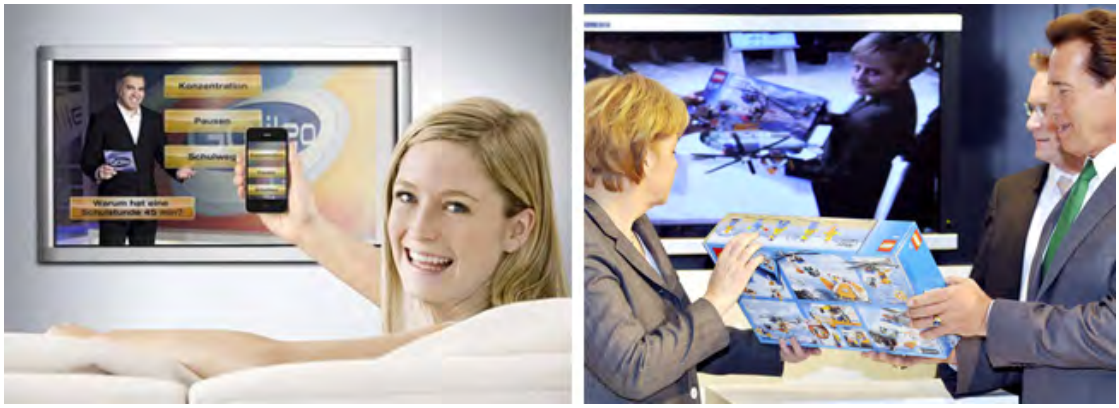


Abbildung 2.2: Anwendungsbeispiel für den Bereich Unterhaltung / Medien. Die App für die Sendung Galileo von ProSieben (links, http://goo.gl/lydg2a Stand: 11. Januar 2015). Ein Beispiel für den Bereich Produktmarketing / Eyecatcher. Die App von LEGO, die es ermöglicht das LEGO-Spielzeug als 3D-Modell auf der Packung anzuzeigen (rechts, http://goo.gl/2yUaAa Stand: 11, Januar 2015).

2.2.1 ProSieben Galileo Zeitreise

Einordnung: Unterhaltung / Medien

Bereits Ende 2011 stellte der Fernsehsender ProSieben in seiner vorabendlichen pseudo-wissenschaftlichen Sendung "Galileo" Augmented Reality als Mittel für ein bidirektionales Fernsehprogramm vor und ermöglichte es dem Zuschauer in einem Anwendungsszenario einer Zeitreise, interaktiv an einem Quiz auf seinem Smartphone teilzunehmen (siehe Abbildung 2.2 links). Während der Sendung wurden Marker eingeblendet und mit einer zuvor geladenen und auf dem Handy installierten Applikation konnte nun aktiv am Quiz teilgenommen werden. Die ProSieben-Zeitreise bedient sich eines texturbasierten Markersystems ausschließlich, um Ereignisse (trigger-events) innerhalb der eigentlichen Quiz-Applikation auszulösen. Das System arbeitet unter den unterschiedlichsten Umgebungsbedingungen, macht sich dabei aber zunutze, dass ein aktives Fernsehgerät eine relativ konstante Helligkeit und Kontrasterscheinung hat, und damit gute Grundvoraussetzungen für die Markererkennung gegeben sind. Nichtsdestoweniger hat diese Anwendung gezeigt, dass AR im Massenbetrieb im häuslichen Umfeld einsetzbar ist. Auswertungen über Erkennungsraten in der App sind zwar nicht veröffentlicht, die Tatsache, dass über 250.000 Zuschauer die App geladen und installiert hatten[1], zeigt aber, dass es auf reges Interesse bei den Zuschauern gestoßen ist und bei einem Großteil der Anwender keine technischen Probleme aufgetreten sind, die ja dann offensichtlich den Spielspaß getrübt hätten.

[1] http://goo.gl/lydg2a Stand: 8. Januar 2015

2.2.2 LEGO Verpackung

Einordnung: Produktmarketing / Eyecatcher
Seit April 2010 findet man in LEGO-Stores die "Digital Box". Dabei handelt es sich um ein Kiosksystem, welches dem Kunden über ein Display das fertig aufgebaute LEGO-Produkt als animiertes 3D-Modell zeigt (siehe Abbildung 2.2 rechts). LEGO nutzt hier Augmented Reality, da eine Kamera im Kiosksystem die Produktschachtel als vordefinierten Marker erkennt und auf dem Videobild dann die 3D-Szene überlagert. So wird dem Kunden das fertige Produkt mit allen Eigenschaften vorgeführt und der Kunde hat die Möglichkeit, sich das Produkt von allen Seiten anzuschauen. Hier werden Informationen und Faszination in einem Eyecatcher zusammengefasst. LEGO bedient sich hierbei der einfachsten aber auch verbreitetsten Art von Erweiterter Realität, bei der eine Überlagerung von einer Information (hier das 3D-Modell des Produktes) auf einem vorher definierten Marker (hier das Deckelbild der Verpackung) im Videobild stattfindet.[2] Dieses Prinzip findet man auch bei vielen AR-Spielen für Smartphones wie zum Beispiel beim "AR Defender", wo auf einem Marker, der mit dem Smartphone getrackt wird, ein Abwehrturm als 3D-Modell angezeigt wird und man mit Berührungen auf dem Display Gegner davon abhalten muss einen Angriff auf den Turm auszuführen.

2.2.3 Ingress

Einordnung: Computerspiel / Games
Das von Google kostenlos bereitgestellte Arcadespiel "Ingress", ist ein Smartphone Onlinespiel mit mehr als acht Millionen Downloads. Dabei sollen Realität und virtuelle Welt miteinander verschmelzen. Das Spielprinzip ist, dass angemeldete Spieler reale Orte (mit GPS-Daten und Karteneinträgen ähnlich Google Maps) als sogenannte "Portale" für das Spiel melden können. Diese werden dann vom Spielebetreiber freigeschaltet. Ab diesem Zeitpunkt sehen alle Mitspieler, an den realen Koordinaten eines Portals, virtuelle Einblendungen (Beispielsweise sogenannte Resonatoren, die "exotische Materie" verstärken) (siehe Abbildung 2.3 links).[3] Das Spiel ist eine neuartige Variante von Rollenspielen, da die Spieler in die Rolle eines Agenten schlüpfen und im Team mit anderen Spielern gegen das Böse kämpfen müssen. Es nutzt soziale Vernetzung, die große Verbreitung von mobilen Geräten wie Smartphones und Tablets sowie die Vermischung von realer und virtueller Welt für ein einzigartiges Spielerlebnis.

2.2.4 Head-up-Displays

Einordnung: Militär / Autoindustrie
Ein Head-up-Display (kurz HUD, wörtlich: Kopf-oben(lassen)-Anzeige) ist ein Anzeigefeld in Blickrichtung des Betrachters (Nutzers). Schon seit den 1940er Jahren werden bei Piloten von Kampfflugzeugen wichtige Informationen in das Hauptsichtfeld projiziert, sodass die Kopfhaltung bzw. die Blickrichtung nicht geändert werden muss, um diese Informationen

[2] http://goo.gl/2yUaAa Stand: 11. Januar 2015
[3] http://goo.gl/1CrNRd Stand: 11. Januar 2015

Abbildung 2.3: Anwendungsbeispiel für ein Computerspiel / Games. In dem Spiel Ingres müssen die Spieler eine geheimnisvolle Energie entweder bekämpfen oder erforschen. Diese Energiefelder werden auf dem Smartphone visualisiert, wenn man bestimmte physische Orte besucht. © Ingress (links, http://goo.gl/83IsGw Stand: 11. Januar 2015). Ein Beispiel für den Bereich Militär / Autoindustrie. Ein HUD als Frontscheibenprojektion, welche wichtige Systeminformationen des Fahrzeuges oder Navigationsinformationen im Blickfeld des Fahrers auf der Frontscheibe anzeigt (rechts, http://goo.gl/rm7afP Stand: 11, Januar 2015).

abzulesen. In den folgenden Jahren ist die Technologie immer weiter ausgereift bis hin zu komplexen Frontscheibenprojektionen (siehe Abbildung 2.3 rechts). In der heutigen Zeit ist solch ein HUD die wichtigste Anzeige im Cockpit. Im Jahr 2003 brachte BMW ein von Siemens VDO Automotive AG entwickeltes HUD im Automobilbereich in Großserie. Die Informationen, die in einem HUD angezeigt werden, können wie folgt unterteilt werden:

- statische Informationen
- kontaktanaloge Informationen

Statische Informationen sind die Fahrzeuggeschwindigkeit und die Motordrehzahl, welche immer an der selben Stelle im Blickfeld des Fahrers angezeigt werden. Der Fahrer sieht diese Information in einer senkrechten Ebene etwas über der Motorhaube. Entsprechend sind dies im Flugzeug Angaben über Höhe und Geschwindigkeit oder auch im militärischen Bereich die Waffenlast.
Kontaktanaloge Informationen sind Anzeigeelemente, die dem Fahrer in seiner aktuellen Sicht so eingeblendet werden, dass er das Gefühl hat, als seien sie fester Bestandteil der Umwelt. Hierzu zählen Navigationsinformationen wie ein Navigationspfeil, der so angezeigt wird, als läge er direkt auf der Straße oder ein Sicherheitsabstandsbalken, der je nach Geschwindigkeit des Autos dem Fahrer anzeigt, welchen Abstand er zum vorausfahrenden Auto halten muss. In Kampfflugzeugen sind es Informationen wie Systeminformationen oder Zielerfassung. Wenn vom Radar- oder IR -System ein Ziel erfasst wird, wird dieses mit einem Leuchtkreis markiert[4].

[4]http://de.wikipedia.org/wiki/Head-up-Display Stand: 11. Januar 2015

Abbildung 2.4: Links ein Anwendungsbeispiel für den Bereich Maschinenbau / Industrie. Anhand von 2D-Barcodes, die an den Einzelteilen der zu reparierenden Maschine angebracht sind, ermittelt das System Kameraposition und -richtung und sendet diese Daten an den Experten. © Fraunhofer FKIE (links, http://goo.gl/gyfhCI Stand: 11. Januar 2015). Rechts ein Beispiel für den Bereich Shopping / Tourismus. Die Wikitude App Zeigt dem Benutzer mit Hilfe von GPS und Orientierungssensor passend zu der Position und Ausrichtung des Gerätes Billbords. Diese können für die Positionen vorhandener Geschäfte und Lokale oder andere Informationen stehen (rechts, http://goo.gl/b3NaLz Stand: 11, Januar 2015).

2.2.5 Arvika

Einordnung: Maschinenbau / Industrie

Schon im Jahr 2001 war die Forschung daran interessiert, AR-Technologie gerade für den Bereich Maschinenbau zu nutzen. Im Leitprojekt ARVIKA vom Fraunhofer-Institut für Graphische Datenverarbeitung (kurz: IGD) werden AR-Technologien zur Unterstützung von Arbeitsprozessen in Entwicklung, Produktion und Service für komplexe technische Produkte und Anlagen benutzerorientiert und anwendungsgetrieben erforscht und realisiert. Durch die visuelle Überlagerung realer Objekte mit rechnergenerierten virtuellen Objekten erlauben Augmented Reality-Techniken im Sinne einer erweiterten Realität das situationsgerechte Agieren in realen Arbeitsumgebungen. [WS03] Der Gedanke bei diesem Projekt war, digitale Handbücher zu erstellen, die einem Monteur/Arbeiter die nächsten Schritte erklären und in seinem Blickfeld gleichzeitig die nötigen Informationen einblenden. Auch durch das Anzeigen von verdeckten oder schwer zugänglichen Bauteilen, könnte mit solchen Industrie-AR-Anwendungen dem Mechaniker die Arbeit erleichtert werden. Wie im Projekt vom Fraunhofer-Institut für Kommunikation, Informationsverarbeitung und Ergonomie (kurz: FKIE) ist auch die Möglichkeit der Fernwartung denkbar (siehe Abbildung 2.4 links). Hierbei wird das Kamerabild einem Experten übermittelt, der dann an bestimmten Punkten Hinweise für den Techniker vor Ort an der Maschine platzieren kann. Diese Hinweise werden dann dem Techniker in seinem Display an der Entsprechenden Position angezeigt. Bewegt er das Display, bewegt sich das Bild mit.[5]

[5] http://goo.gl/gyfhCI Stand: 11. Januar 2015

2.2.6 Wikitude App

Einordnung: Shopping / Tourismus

Anwendungen wie die App von `Wikitude` gibt es zahlreich. Diese Anwendungen nutzen die Sensoren des mobilen Gerätes wie GPS, Kompass und Gyrosensor, um die eigene Position und die Blickrichtung zu erkennen und dem Benutzer im Blickfeld der Kamera sogenannte Billbords anzuzeigen, die für interessante Orte in Bezug auf die eigene Position stehen. Auf den Billbords wird neben dem Namen des "point of interest" (kurz POI), die Entfernung vom Betrachter und vielleicht noch ein Bild dazu angezeigt (siehe Abbildung 2.4 rechts). Durch Drücken auf die Billboards kann sich der Benutzer weitere Informationen anzeigen oder sich zu diesem Ort navigieren lassen.[6] Mit solchen Anwendungen ist es gerade in Städten sehr schnell möglich, den nächsten Bankautomaten, Supermarkt oder das nächste Hotel ausfindig zu machen. In Touristengegenden können damit Sehenswürdigkeiten geortet und Informationen dazu abgefragt werden. Auch wenn die POIs auf einer Kartenansicht genauso gut angezeigt werden könnten, ist die Variante mit der Anzeige im Kamerabild, gerade was die Orientierung angeht, eine hilfreiche Alternative.

2.3 Agumented Reality-Entwicklungsumgebungen

Das Wort Entwicklungsumgebung ist an dieser Stelle fast etwas missverständlich: Hier ist natürlich nicht die typische Entwicklungsumgebung/IDE-Software gemeint, also ein Softwaresystem, das die Werkzeuge zum einfachen Erstellen von Softwarepaketen und -Anwendungen liefert, sondern Bibliotheken/SDKs, die die Hauptfunktionalitäten für Erweiterte Realität und ihre Anwendungen bereitstellen und in eigene Applikationen integriert werden können. Diese Basisfunktionalitäten sind unter anderem:

- Bilderverarbeitende Algorithmen, mit denen die Videoströme analysiert und markante Punkte, Referenzobjekte und Referenzmarkierungen sowie Texturen erkannt und analysiert werden können.
- Bereitstellung von diversen Sensoren und deren Auslesen und Ansteuerung auf den unterschiedlichsten Softwareplattformen und auf einer Vielzahl von Endgeräten.
- Darstellung von virtuellen Erweiterungen sowohl als zweidimensionale Objekte wie Texte, Bilder, Videos als auch dreidimensionale Objekte, die in Echtzeit gerendert werden können.

Auf dem Markt gibt es eine Vielzahl von freien und kostenpflichtigen Entwicklungsumgebungen mit denen Augmented Reality-Anwendungen erstellt werden können. Jedoch sind diese in ihrer Kompatibilität teilweise sehr eingeschränkt. Die bekanntesten und verbreitetsten SDKs sind in Abbildung 2.5 aufgeführt.

[6] http://www.wikitude.com/app/ Stand: 11. Januar 2015

Abbildung 2.5: Auflistung der bekanntesten und verbreitetsten SDKs im Bereich AR.

Die großen Software Bibliotheken unterstützen alle iOS, Android, zum Teil auch Windows Phone und Blackberry. Die Entwicklungsumgebungen, die am meisten Flexibilität und Erweiterbarkeit bieten, sind Vuforia von Qualcom und Metaio, auf die im kommenden Abschnitt weiter eingegangen wird und die anschließend miteinander verglichen und bewertet werden.

2.3.1 Qualcomm® Vuforia™

Vuforia ist ein natives SDK für iOS & Android. Es stellt Bibliotheken zur Verfügung, die das Tracking und das Rendering übernehmen. Zum Entwickeln von Applikationen für Android kann hier auf Eclipse und das ADT von Google und für iOS auf xCode zurückgegriffen werden. Ebenfalls stellt Vuforia auch Bibliotheken für die Game-Engine Unity3D[7] zur Verfügung, die es erlauben, die Trackingverfahren auch in die sehr flexible Game-Engine zu integrieren und so auf die sehr leistungsstarken 3D-Rendering-Möglichkeiten von Unity3D zuzugreifen. Unity3D bietet als Game-Engine das Grundgerüst zur Erstellung von 2D- oder 3D-Spielen sowie für interaktive industrielle Simulationen. Hierbei stellt die Engine eine Vielzahl von Bestandteilen zur Verfügung, die das Erstellen solcher Anwendungen erleichtert. Hierzu gehört unter anderem die Grafik-Engine, Sound-Engine und die Steuerung. Durch die Möglichkeit aus Unity3D native Anwendungen für Android und iOS zu entwickeln, ist es sogar möglich Multiplattform-Entwicklung so gut wie nur mit einem Mausklick zu betreiben.

[7] http://unity3d.com Stand: 8. Januar 2015

Die Vuforia™-Tracking Möglichkeiten

Vuforia ist in der Lage verschiedene Arten von Markern zu erkennen, die in der folgenden Auflistung kurz erklärt werden:[8]

Framemarker sind Bilder, die ein wenig wie QR-Codes aussehen, jedoch die Freiheit bieten, ein eigenes Bild (Logo) ins Innere zu setzen. Sie sind ideal für Spielsteine oder andere Anwendungsszenarien, bei denen mehrere Targets gleichzeitig getrackt werden.

Imagemarker sind ausreichend detaillierte Bilder, die vorher definiert sind und dann als Tracker benutzt werden können. Beispiele sind Bilder in Zeitschriften, Anzeigen oder Produktverpackungen.

Benutzerdefinierte Imagemarker geben dem Benutzer die Möglichkeit grundlegende AR-Elemente zu erstellen, die überall realisiert wer+den können. Die Erstellung ist so einfach wie das Fotografieren von Alltagsgegenständen, beispielsweise von einer Buchseite, einem Plakat, einem Magazin-Cover oder sogar einem Gemälde. Benutzerdefinierte Bilder eignen sich besonders gut für Spiele.

Einfache Boxen mit flachen Seiten und mit ausreichenden visuellen Details können erkannt werden. Dazu gehören Objekte aus flachen Oberflächen, wie Schachteln und Verpackungen.

Zylinder, wie Flaschen, Dosen, Schalen und Becher, können ebenfalls als Marker erkannt werden.

Objekte können ab der neuen Vuforia™ 4.0 Beta Version auch erkannt werden. So können komplizierte 3D-Objekte wie Spielwaren und andere Produkte erkannt und verfolgt werden, sodass Spielzeuge zum Leben erweckt werden und digitale Funktionen an Produkten hinzugefügt werden können.

[8] http://goo.gl/RNbMQM Stand: 8. Januar 2015

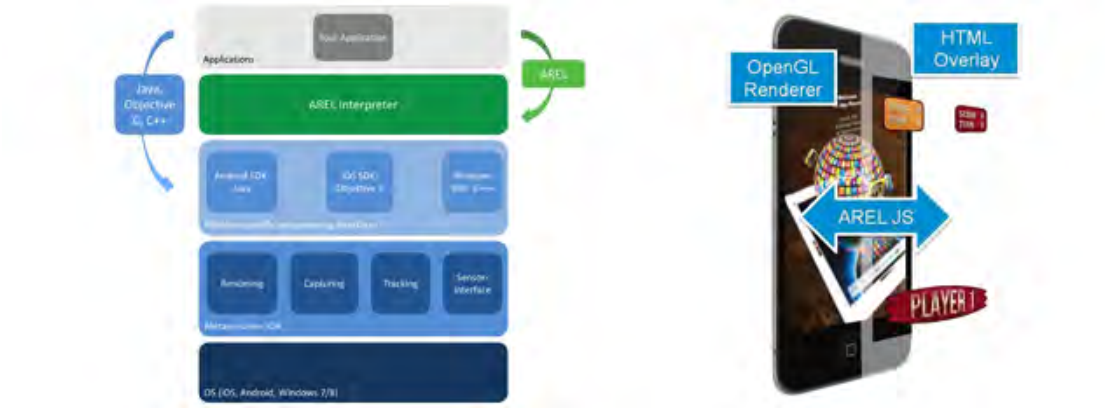


Abbildung 2.6: Die Abbildung links zeigt, wie der AREL Interpreter bei Metaio in das System interferiert wird. Abbildung rechts zeigt das Prinzip des Overlays bei AREL Anwendungen (http://dev.metaio.com/arel/overview/ Stand: 12. Januar 2015).

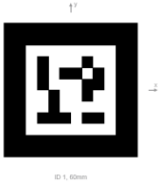
2.3.2 Metaio

Metaio ist wie auch Vuforia ein natives SDK für iOS & Android und stellt auch Bibliotheken zur Verfügung, die mit xCode bzw. Eclipse und dem ADT von Google für die Entwicklung von nativen Anwendungen für mobile Geräte genutzt werden können. Ebenfalls stellt auch Metaio die nötigen Bibliotheken für die Integration in Unity3D bereit. Neben der Multiplattform-Entwicklung durch Unity3D bietet das Metaio SDK auch noch die Möglichkeit mit AREL / HTML5, Multiplattform-Entwicklung zu betreiben. Jedoch handelt es sich dabei dann nicht mehr um native Anwendungen. Bei der Scriptsprache AREL (Augmented Reality Experience Language) wird die plattformunabhängige Anwendungsentwicklung auf der Basis einer Kombination von JavaScript, dem Metaio SDK und einer statischen XML-Content Definition entwickelt. AREL fügt HTML5-Overlays und einfache GUI Elemente hinzu (siehe Abbildung 2.6). Anwendungen, die mit Hilfe von AREL umgesetzt wurden, können bei Metaio in der hauseigenen Junaio-Plattform online anderen Usern zur Verfügung gestellt werden. Junaio ist ein fortschrittlicher mobiler AR-Browser, der es ermöglicht kostenfrei, schnell und einfach Augmented Reality Erfahrungen auf bekannten Web-Technologien wie XML und HTML5 zur Verfügung zu stellen. Neben den von der Community hinzugefügten AR-Anwendungen kann die Junaio-App auch dem Benutzer interessante Orte wie Geschäfte, Lokale etc. ortsgebunden anzeigen.[9]

[9] http://dev.metaio.com/junaio Stand: 12. Januar 2015

Die Metaio Tracking Möglichkeiten

Die große Stärke von Metaio ist die große Vielzahl von Tracking-Möglichkeiten sowie die Integration von Non-optical-Tracking-Technologien, wie GPS-Sensoren. Die folgende Aufzählung beschreibt die optischen Tracking-Technologien von Metaio[10]:



ID Marker sind die einfachsten und am schnellsten zu erkennenden Marker in Metaio. Ein Mobiltelefon kann ca. 10 bis 30 ID Marker ohne Performance-Einbußen handhaben.

LLA Marker LLA steht für Latitude, Longitude, Altitude (deutsch: Längengrad, Breitengrad und Höhe) und ist ein Format zur Definition von Geo-Positionen auf dem Globus. Während die Höhe in Metern über dem Meeresspiegel gemessen wird, sind die Breite und die Länge in Grad gemessen. Mit solchen Markern ist eine annähernde Positionsbestimmung möglich.

Picture Marker liegen irgendwo zwischen den ID Markern und dem markerless Tracking. Es kann jedes beliebige Bild mit genug Bildinhalt benutzt werden. Wichtig ist, dass das Referenzbild genau den gleichen Bildausschnitt wie der ausgedruckte Marker hat.

Markerless-2D Ist ein Trackingverfahren, bei dem über einen Bildausschnitt Referenzpunkte erkannt werden können. Hier ist es nicht notwendig, dass wie beim Image Marker, dass ganze Bild bekannt ist. Es ist ebenfalls möglich mehrere Marker-Punkte in einem Gesamtbild anzulegen und diese auch gleichzeitig zu tracken. So ist es zum Beispiel möglich gleichzeitig verschiedene Informationen auf einem großen Bild anzeigen zu lassen, wenn die Kamera beide im Blickfeld hat.

[10] http://dev.metaio.com/sdk/documentation Stand: 12. Januar 2015

Markerless-3D ist die neuste Trackingmethode von Metaio und ermöglicht es jedes reale Objekt als Tracking-Referenz zu benutzen. Entweder wird aus der eigenen Anwendung mit Hilfe der Kamera die Trackingreferenz ermittelt, die verwendet werden soll, oder es wird über die Metaio Toolbox eine mobile Anwendung geladen, die es ermöglicht, Markerless-3D Trackingdaten zu speichern und in der eigenen Anwendung als Referenz zu benutzen, hierbei werden die gewünschten Objekte gescannt und die entstandenen Trackingdaten werden in einer Tracking-XML gespeichert.

QR und Barcode Reader Metaio ermöglicht das Lesen von QR-, sowie Barcodes. Hierfür muss nur die TrackingData_BarCode.xml eingelesen werden.

Die beiden vorgestellten Entwicklungsumgebungen sind in ihrem Funktionsumfang sehr ähnlich. So ist es auch leicht zu verstehen, dass sie die in der AR-Entwicklung verbreitetsten Systeme sind. Zu dem Zeitpunkt, als die Entscheidung getroffen wurde welches AR-System in dieser Arbeit und auch in dem Forschungsprojekt PIMAR verwendet werden soll, ist das SDK Vuforia noch nicht in der Lage, Objekte zu erkennen. Diese Möglichkeit ist erst jetzt mit der Vuforia™ 4.0 Beta Version gegeben.[11] Jedoch ist für das Projekt PIMAR, die Erkennung von realen Objekten und die Augmentierung auf Basis dieser Objekte zwingend notwendig. Es soll die Möglichkeit geschaffen werden, anhand der erkannten Objekte, Hilfestellungen und Informationen direkt auf den Objekten (Maschinen, Ausstellungsstücke in einem Museum etc.) eingeblendet zu bekommen. Somit reichen die anderen Tracking-Methoden mit primitiven oder Bildmarkern nicht aus. Das SDK von Metaio entspricht daher den Anforderungen des Projektes besser und wird deshalb in dieser Arbeit verwendet.

2.4 Positions- und Orientierungsbestimmung für AR-Tracking

Die am meisten verbreiteten AR-Anwendungen sind auf den stationären Gebrauch ausgelegt, wie die Digital Box von LEGO (siehe Erklärung in Abschnitt 2.2.2) sowie ähnliche Ansätze, wo die Anwendung bekannte ID- oder Image-Marker erkennt und einfache Augmentierungen ausführt. Ortsgebundene Augmentierungen, wie bei der Wikitude App (siehe Erklärung in Abschnitt 2.2.6), werden mit Hilfe von GPS und dem Orientierungssensor umgesetzt. Jedoch funktionieren diese Anwendungen nur, wenn die aktuelle Position des Benutzers bestimmt werden kann, was in vielen Fällen nur sehr ungenau bzw. meistens nur im Außenbereich

[11] http://developer-beta.vuforia.com Stand: 13. Januar 2015

richtig funktioniert. Für die Problemstellung der Positions- und Orientierungsbestimmung für Augmented Reality-Anwendungen bzw. für deren Initialisierung gibt es zur Zeit viele verschiedene Ansätze. Im nächsten Abschnitt werden diese kategorisiert und die Funktionsweisen näher beleuchtet.

2.4.1 Initialisierung durch visuelle Marker

ID-Marker sind die verbreitetsten und auch am einfachsten zu trackenden Marker, wenn es um optische tracking Methoden geht. ID-Marker gibt es in unterschiedlichen Ausführungen. Einer der bekanntesten ist wohl der QR-Code (engl. Quick Response), der nicht nur Anwendung im Bereich AR findet, sondern zum Beispiel auch für die schnelle Verknüpfung von Weblinks genutzt wird. Der QR-Code ermöglicht es Informationen so aufzubereiten, dass sie von Maschinen schnell gefunden und ausgelesen werden können.[12] So können Informationen zur Initialisierung direkt in dem Marker hinterlegt werden. Ähnlich aufgebaut sind die ID-Marker von Metaio, die es ermöglichen bis zu 512 verschiedene Marker fast ohne Leistungseinbuße zu verwenden.[13] Somit ist diese Art von Marker der einfachste und schnellste Weg, um eine Ortsbestimmung gerade in Innenräumen durchzuführen. In dem Forschungsprojekt DIB - Dienstleistungen im industriellen Bauprozess - werden Bauabläufe sowie Planungs- und Realisierungsprozesse mit Hilfe von Augmented Reality optimiert. [CG11] Über eine mobile App können in Echtzeit dreidimensionale Baupläne über die Aufnahmen des Bauobjektes projiziert werden.[14] Für die Initialisierung und die damit verbundene Ortsbestimmung wird in diesem Projekt auf klassische QR-Codes zurückgegriffen (siehe Abbildung 2.7 links). Ein Studienprojekt der Hochschule Reutlingen beschäftigt sich ebenfalls mit der Problematik der automatischen Initialisierung und der damit verbundenen Positions- und Orientierungsbestimmung für AR-Tracking. Hier wird für die Entwicklung einer AR-Schnitzeljagd für interessierte Schüler im Fachbereich Informatik auf die sogenannten LLA-Marker (Latitude, Longitude, Altitude) von Metaio zurückgegriffen, um ortsgebundenen AR-Inhalte in der Anwendung zu laden. [Thi11] Wo in Außenbereichen üblicherweise mittels GPS-Ortung eine ungefähre Position bestimmt werden kann, können mit Hilfe der LLA-Marker auch in Innenräumen die benötigten Positionsdaten über den QR-Code-ähnlichen Marker an die Anwendung übermittelt werden. LLA-Marker können über ein Webinterface bei Metaio im Entwicklerbereich auf der Webseite generiert werden.[15]

2.4.2 Drahtlose Innenraum-Positionierung

In Außenbereichen stehen zur Positionsbestimmung von Anwendern mehrere Methoden und Systeme zur Verfügung. Bei der Bestimmung der Position über Mobilfunknetze hängt die Genauigkeit stark von der Zellengröße ab. Bei der Nutzung von Satellitennavigation (GPS) steht dem Nutzer eine für die meisten Anwendungsfälle ausreichend genaue Methode zur

[12] http://goo.gl/hao89a Stand: 15, Januar 2015
[13] http://goo.gl/7ktln5 Stand: 15. Januar 2015
[14] http://goo.gl/DNqJT8 Stand: 16. Januar 2015
[15] http://goo.gl/JO0Ns6 Stand: 16. Januar 2015

Abbildung 2.7: Abbildung links zeigt die Initialisierung der AR-Überlagerung mit Hilfe eines QR-Codes im Forschungsprojekt DIB (http://goo.gl/DNqJT8 Stand: 16. Januar 2015). Abbildung rechts zeigt das Handyticketsystem Touch&Travel der Deutschen Bahn. Über ein Smartphone kann via NFC, QR-Code oder die Eingabe der Kontaktpunktnummer ein Bahnticket gekauft werden (http://goo.gl/N8yHdO Stand: 19. Januar 2015).

Verfügung. Jedoch beschränkt sich die Nutzung solcher Systeme in den meisten Fällen auf Regionen außerhalb von Gebäuden. Die verwendeten Signale werden durch dickes Mauerwerk blockiert und erreichen so im Innenbereich keine ausreichende Signalstärke für eine genaue Positionsbestimmung. [LDBL07] Um trotzdem eine Position innerhalb eines Gebäudes bestimmen zu können, müssen zusätzliche Installationen von Hardwarekomponenten berücksichtigt werden. Hierfür hat sich noch kein Standard durchgesetzt, was dazu führt, dass sehr unterschiedliche Verfahren in Verwendung sind, die sich hinsichtlich ihrer Genauigkeit und Kosten noch sehr unterscheiden. [Sch09]
Da der Schwerpunkt dieser Arbeit auf der Verwendung von mobilen Geräten wie Smartphones oder Tablets liegt, werden im folgenden Abschnitt nur die Verfahren vorgestellt, die auf der einen Seite mit der heutigen Technik von mobilen Geräten umgesetzt werden können und auf der anderen Seite auch schon den größeren Anteil der auf dem Markt verbreiteten Anwendungen für sich beanspruchen können.

RFID

RFID (engl. Radio-Frequency Identification) ist eine Technologie für Sender-Empfänger-Systeme zum automatischen und berührungslosen Identifizieren und Lokalisieren von Objekten mit Radiowellen. Durch die geringe Größe, die unauffällige Auslesemöglichkeit und den geringen Preis der Transponder werden heutzutage RFID-Transponder in sehr vielen Einsatzgebieten genutzt. [Fin06] [GH07] Seit dem 1. November 2010 befinden sich RFID-Transponder in den neu eingeführten deutschen Personalausweisen.[16] Die RFID-Technologie ermöglicht Waren- und Bestandsmanagement und erleichtert so Logistikstrukturen. Die

[16] http://goo.gl/gtPqBd Stand: 19. Januar 2015

Reichweite liegt für die RFID-Technologie je nach Anwendungsbereich und Frequenz bei 0,5 - 10 Metern.[17] Durch die recht hohe Reichweite findet die Technologie auch Anwendung bei der Zeiterfassung im Sport. Hierbei werden die Transponder am Sportgerät, an der Kleidung oder an der Startnummer befestigt und dienen als Identifikationsmerkmal. Ein großes Einsatzgebiet ist auch die Zugangskontrolle zum Skilift in den meisten Skigebieten. So muss der Skifahrer nur durch das Drehkreuz gehen und das System liest aus der Skikarte die dazugehörige ID aus, die mit einer globalen Datenbank verglichen wird und nach erhaltener Berechtigung das Drehkreuz frei gibt. Durch diese Lokalisierungen des Anwenders, ist es möglich, dem Skifahrer auch eine Statistik über die von ihm genutzten Lifte und somit im Rückschluss auch auf die von ihm gefahrenen Pistenkilometer und Höhenmeter etc. zur Verfügung zu stellen. [PSM13] [Ker07] Wie dies zum Beispiel von einem Zusammenschluss aus mehreren Skigebieten mit dem Internet-Portal "skiline" seinen Kunden angeboten wird.[18] Es gibt auch verschiedene Ansätze mit RFID-Technologie Positionsbestimmungen in Innenräumen umzusetzen. [LDBL07] Eines der Verfahren ist unter anderem das Prototypsystem LANDMARK, ein Ortserfassungssystem für Innenräume welches mit aktiven Transpondern arbeitet. [NLLP03] Ein anderes Verfahren ist SpotON, welches einen Aggregations-Algorithmus für eine 3D-Ortserfassung auf Basis einer Funksignalstärken-Analyse verwendet [HWB00].

NFC

NFC (engl. Near Field Communication) ist ein Übertragungsstandard auf Basis von RFID zum kontaktlosen Austausch von Daten per Funk über kurze Distanzen von wenigen Zentimetern. [LR10] Dieser Standard wird heute zum Beispiel beim bargeldlosen Bezahlen bei kleinen Beträgen bis zu 20 Euro[19] angewendet. Die meisten Hochschulen benutzen NFC-Chips in ihren Studentenausweisen, damit die Studenten in der Mensa damit bezahlen können. Auch für Zugangskontrollen kann die NFC Technik benutzt werden. Aktuelle Smartphones besitzen in der Regel ebenfalls NFC-Sensoren, sodass diese Technologie auch für das schnelle Verbinden zweier Geräte zum Datenaustausch genutzt werden kann. Mit Hilfe der NFC-Technologie wäre eine Positionsbestimmung insofern möglich, wenn der Benutzer sein Gerät an vordefinierte Stationen hält, die dem Gerät über eine ID mitteilen, wo sich der Benutzer gerade befindet. Diese Funktion wurde im Jahr 2006 vom Rhein-Main-Verkehrsverbund (RMV) in den Regelbetrieb als "NFC Handy Ticketing" übernommen. [STT09] Durch halten eines NFC fähigen Handys an entsprechende Terminals können Fahrkarten elektronisch bezahlt, gespeichert und entwertet werden. Dieses Verfahren hat sich die Deutsche Bahn mit dem Verkehrsmittel übergreifenden Handyticketsystem Touch&Travel [EJ10] [JZB14] ab 2007 zu eigen gemacht (siehe Abbildung 2.7 rechts). Deutschlandweit und auf ausgewählten Strecken ins europäische Ausland können Kunden der Deutschen Bahn das System nutzen.[20] Vor der Fahrt muss der Kunde sich über einen

[17] http://goo.gl/gtPqBd Stand: 19. Januar 2015
[18] http://www.skiline.cc Stand: 19. Januar 2015
[19] http://goo.gl/K1R1CI Stand: 19. Januar 2015
[20] http://goo.gl/caAG58 Stand: 19. Januar 2015

Kontaktpunkt am Bahnsteig einchecken und am Zielort wieder auschecken. Die Bestimmung der gefahrenen Route und des richtigen Tickets übernimmt das Touch&Travel-System und berechnet automatisch den günstigsten Tarif.[21]

BLE

BLE steht für Bluetooth Low Energy und ist eine Funktechnik aus dem Bluetooth Standard 4.0, mit der sich Geräte in einer Umgebung von bis zu 50 Metern mit einer kabellosen bidirektionalen Datenübertragung verbinden können.[22,23] Zudem können Hersteller die Reichweite von Bluetooth-Geräten mit der neuen Spezifikation anpassen und dank eines erhöhten Modulationsindex auf über 100 Meter steigern. Eine 24-Bit-Fehlerkorrektur soll die Verbindungen robust machen. Für die Verschlüsselung wird der Blockchiffre "Advanced Encryption Standard" AES-128 verwendet. Mit BLE ist eine besonders energiesparende Betriebsweise möglich, sodass für Geräte mit Knopfzellenbatterien eine längerfristige Betriebszeit gewährleistet wird. Die Technik wird unter dem Namen "Bluetooth Smart" vermarktet und findet in letzter Zeit immer häufiger Anwendung bei Drahtloser Nahfeldkommunikation und somit verbundenen Innenraum Ortserfassungen. Es gibt verschiedenen Produkte auf dem Markt (Bluetooth Low Energy Beacons), die eine sehr genaue Lokalisierung in Innenräumen ermöglichen und somit die Möglichkeit für standortbezogene Dienste (engl. Location-based Services kurz LBS) auf mobilen Geräten bereitstellen. Hier ist der sogenannte "iBeacon"[24], ein proprietärer Standard für die Navigation in geschlossenen Räumen, von `Apple`[25] einer der bekanntesten Standards. Ein solcher Beacon sendet dauerhaft seine ID und einen Indikator für die Sendeleistung. Weitere Funktionen hat so ein Beacon nicht. Es werden auch keine Daten empfangen oder eine Kopplung mit anderen bluetooth-fähigen Geräten ermöglicht. Über die Sendeleistung kann eine Entfernung zu einem Beacon ermittelt werden. Wenn mehrere Beacons mit festen und bekannten Positionen im Gebrauch sind, kann eine Benutzerposition z.B. in einem Geschäft ermittelt werden. Eine genaue Positionsbestimmung ist jedoch erst möglich, wenn sich bei mindestens drei Beacons die Sendeleistungen überschneiden. [VC14] [Kud14] [Gas14] So ist es also möglich dem Kunden beim Betreten eines Ladengeschäftes oder wenn er sich einer Position in dem Geschäft nähert, aktuelle Informationen über Produkte oder Angebote direkt auf seinem Mobiltelefon anzuzeigen (siehe Abbildung 2.8 links). Ein weiteres Anwendungsfeld bietet hier ein Museum, wo ein Besucher entweder durch dieses navigiert werden kann und oder ihm, wenn er sich einem Ausstellungsstück nähert, passende Informationen auf dem Gerät anzeigt werden können.

Wireless-LAN

Viele Indoornavigationssysteme arbeiten zur Zeit mit der Wireless-LAN (kurz: WLAN) Technologie. Es liegt auf der einen Seite daran, dass diese Technologie inzwischen ausgereift ist

[21] http://goo.gl/XmQtwn Stand: 19. Januar 2015
[22] http://goo.gl/aZWSTt Stans: 20. Januar 2015
[23] http://goo.gl/L84dPz Stand: 20. Januar 2015
[24] https://developer.apple.com/ibeacon/ Stand: 31. Januar 2015
[25] https://www.apple.com/de/

Abbildung 2.8: Abbildung links zeigt die Ortsbestimmung über das iBeacon. Aktuelle Angebote in einem Laden werden dem Endkunden direkt auf das Handy geschickt, sobald er den Laden betritt (http://goo.gl/u42w7K Stand: 16. Januar 2015). Abbildung rechts zeigt die Lokalisierung der Referenzpunkte einer WLAN Fingerprinting-Umgebung [Kup05].

und auf der anderen Seite besteht der Vorteil darin, dass die meisten mobilen Geräte die Technik unterstützen. Wie bei der iBeacon Technologie, die mit BLE Technik arbeitet, wird auch bei der Positionsbestimmung über WLAN mithilfe der Reichweite und der Schnittpunkte verschiedener WLAN-Hotspots die aktuelle Position sehr präzise berechnet. Dabei greifen die meisten Systeme auf eine Messung der Signalstärken (RSS), des Signal-Rausch-Abstandes (SNR) oder auf das sogenannte Proximity-Sensing zurück. Wichtig für dieses Verfahren ist, das die Positionen der einzelnen Hotspots dem System bekannt sind und im System ein "Location Server" installiert ist, der die Auswertung übernimmt. Eine weitere noch genauerer Methode ist die Positionsbestimmung über "WLAN Fingerprinting". Dieses Verfahren unterteilt sich in zwei Phasen im Positionierungsprozess. In der Off-Line-Phase, die auch als Trainings- oder Kalibrierungsphase bezeichnet wird, wird eine sogenannte "Radio-Map" erzeugt [Kup05]. In dieser Map werden die gemessenen Signalstärken der verfügbaren Hotspots an zuvor festgelegten Referenzpunkten gespeichert (siehe Abbildung 2.8 rechts). Die zweite Phase ist die On-Line-Phase, in der die eigentliche Positionsbestimmung vorgenommen wird. Hier werden die von einem Endgerät gesammelten Signalstärken der Hotspots an einem bestimmten Punkt mit der vorher erstellten RadioMap verglichen, um den besten Referenzpunkt zu finden, der zu den gemessenen Signalstärken passt. Als Position wird nun der ermittelte Referenzpunkt zurückgegeben. [LDBL07] [STK05] [CCBA07] [Sch09]

2.4.3 Kombination von GPS und visuellem Tracking

Nach Gerhard Reitmayr und Tom W. Drummond verlassen sich Augmented Reality Systeme für den Außenbereich häufig allein auf GPS-Technologie, um große Flächen abzudecken. Auch wenn visuelle Tracking-Anwendungen genauere Positionsabschätzungen liefern können, benötigen diese aber typischerweise eine manuelle Initialisierung. In dem Artikel "Initialisation for Visual Tracking in Urban Environments" beschreiben sie die Kombination von

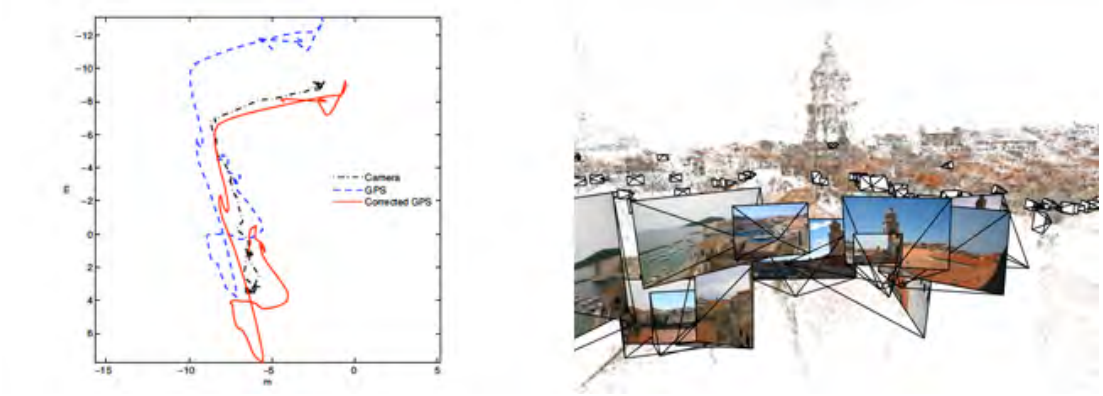


Abbildung 2.9: Abbildung links zeigt eine 2D-Projektion von einem Beispiel-Kamerapfad (schwarz) mit nicht korrigierten GPS-Track (blau) und korrigierten GPS-Track (rot). Die korrigierte Schätzung ermöglicht einen schnelleren Neuinitialisierung [RD07]. Abbildung rechts zeigt, eine Rekonstruktion eines Stadtteiles an Hand von einer Punktwolke aus Feature-Punkten die aus einer großen Bilddatenbank extrahiert wurden. (http://www.cs.cornell.edu/projects/p2f/ Stand: 16. Januar 2015)

GPS-Tracking und visuellem Tracking. Dabei wird ein akkurates Lokalisierungssystem kreiert, welches für eine (Re)Initialisierung keine zusätzliche Eingabe benötigt. Die 2D-GPS-Position im Zusammenspiel mit der durchschnittlichen Größe des Benutzers wird als initialer Anhaltspunkt für das visuelle Tracking benutzt. Das Problem, dass das derzeitig verfügbare GPS noch hohe Ungenauigkeiten aufweist, demgegenüber aber eine hohe Genauigkeit für die Initialisierung benötigt wird, wurde überwunden, indem ein Suchverfahren genutzt wird, welches versucht die Zeit, die für die Suche benötigt wird, zu reduzieren, indem es die Wahrscheinlichkeit optimiert, dass früh die korrekte Schätzung gefunden wird. Nach vielen Abbrüchen des GPS wurde die (Re)Initialisierung vom visuellen Tracking-System verbessert, indem versucht wurde die GPS-Fehler mit einem Gauß'schen Prozessfilter zu differenzieren und damit eine bessere Genauigkeit für die Lokalisierung zu erzielen und gleichzeitig die Such-Zeit zu minimieren. Bei diesem Verfahren spielt die Sensorfusion mit dem Orientierungssensor die entscheidende Rolle zur Bestimmung der Szenerie. Der Schwerpunkt der Arbeit liegt bei der (Re)Initialisierung des Trackings [RD07].

2.4.4 Image Matching-Verfahren

Yunpeng Li et al. schreiben in ihrer Arbeit "Location Recognition using Prioritized Feature Matching" über eine schnelle und einfache Ortbestimmungs- und Bildlokalisierungsmethode [LSH10]. Aus einer großen Sammlung von Bildern werden mit *Image Matching*-Verfahren und *Structure from Motion*-Techniken 3D-Modelle erzeugt. Mit Hilfe eines SIFT (*Scale Invariant Feature Transform*)-Verfahrens werden Übereinstimmungen zwischen den Bildern des verwendeten SfM und der rekonstruierten 3D-Geometrie gesucht. Ähnliche Ansätze für Ortsbestimmung für große Gebiete, wie zum Beispiel einer Stadt, mit Hilfe von großen

Bilddatenbanken werden in der Arbeit von Grant Schindler et al. "City-Scale Location Recognition" [SBS07] sowie in der Arbeit von Duncan Robertson und Roberto Cipolla "An Image-Based System for Urban Navigation" aufgezeigt [RC04]. Die meisten Arbeiten, die sich mit Bildvergleichen beschäftigen, nutzen das SIFT-Verfahren zum Vergleich von Bildmerkmalen. SIFT ist robust gegenüber Skalierung, Lichtwechsel, teilweise Verdeckung, und perspektivischen Veränderungen. Einen Überblick über lokale Merkmalbeschreibung findet man in der Arbeit von Mikolajczyk et al. [MS05]. Der generelle Ablauf des SIFT-Algorithmus erfolgt dabei über vier Schritte:

- die Extremumdetektion im Skalenraum (mit Hilfe einer Gaußdifferenzfunktion)
- die Exakte Schlüsselpunkt-Lokalisierung (mit Hilfe eines Orts- und Skalenmodells)
- das Zuweisen von Orientierungen pro Schlüsselpunkt
- die Generierung der Schlüsselpunktbeschreibung

SIFT ist zwar in der Objekterkennung weit verbreitet, jedoch ist es sehr rechenleistungsintensiv. [WRM+08]

2.4.5 Vereinfachtes SIFT-Verfahren

Um das Problem der großen Rechenlast zu lösen, entwickelten Werner et al. in der Arbeit "Pose tracking from natural features on mobile phones" [WRM+08] ein vereinfachtes SIFT-Verfahren, welches die 3D-Position eines 2D-Objektes schätzt. Auf Grundlage diese Verfahrens entwickelten Niels Henze et al. in der Arbeit "What is That? Object Recognition from Natural Features on a Mobile Phone" das Verfahren weiter. In diese Arbeit wird ein markerloses Objekterkennungsverfahren beschrieben, welches mit mehreren Kamerabildern von einem mobilen Telefon Bilder und sogar Objekte erkennt [HSB09]. Das Verfahren kombiniert dabei ein vereinfachtes SIFT-Verfahren, einen skalierbaren *Vocabulary-Tree* und einen einfachen *Feature Matching*-Algorithmus.

2.5 Zusammenfassung

Das Gebiet der Erweiterten Realität (engl. Augmented Reality) beschreibt eine innovative Form der Mensch-Computer Interaktion, bei der die reale Welt mit Zusatzinformationen angereichert wird. Auf diese Art und Weise bieten sich neue Möglichkeiten Informationen im Blickfeld des Betrachters zu präsentieren. Dies wird zunehmend attraktiver, da in der heutigen Zeit der Begriff "Time-to-Content" immer mehr an Bedeutung gewinnt. Das resultiert daraus, dass bei der digitalen Überflutung von Informationen durch eine Filterung über Ort und Zeit einem genau dann die Informationen angezeigt werden, wenn sie wirklich von Relevanz sind. Der Oberbegriff für die Vermischung der Realität mit einer virtuellen Welt lautet "mixed Reality". Dieser Begriff umfasst sowohl Augmented Reality als auch virtual Reality.

Die flexibelsten und mächtigsten SDK's unter den größten momentan auf dem Markt vertretenen AR-Entwicklungsumgebungen sind von der Firma Qualcomm Vuforia™ und das Metaio SDK, die in erster Linie ähnliche Funktionen bieten, jedoch teilweise andere Ansätze verfolgen. Als wichtigster Nachteil von Vuforia™ ist zu erwähnen, dass die Entwicklungsumgebung zur Zeit der Entscheidung, welches SDK im Projekt PIMAR und damit auch in der hier vorliegenden Arbeit verwendet werden soll, noch nicht in der Lage war, ganze Objekte zu tracken. Dies ist zwar laut eigenen Angaben von Qualcomm seit der neuen 4.0 Beta Version möglich, jedoch fiel bei Projektstart deshalb die Entscheidung für das System von Metaio.

In Bezug auf Positions- und Orientierungsbestimmung zur Initialisierung von AR-Anwendungen ist auffällig, dass die meisten, besonders die kleineren AR-Applikationen eher stationäre Anwendungen sind, bei denen in erster Linie eine Vielzahl von Bild-ID-Markern erkannt werden müssen. Oft handelt es sich auch um ortsgebundene Augmentierungen, die über die GPS-Position und den Orientierungssensor dem Anwender auf seinen Standort bezogene Informationen im Display anzeigen. Jedoch liefert die GPS-Ortung in der Regel nur außerhalb von Gebäuden verlässliche Ergebnisse. Um eine Ortung in Innenräumen zu gewährleisten bzw. eine Positionsbestimmung vorzunehmen, erfolgt die Initialisierung zum Beispiel durch visuelle Marker oder die Drahtlose-Innenraum-Positionierung. Beispielhafte Verfahren hierfür sind RFID bzw. NFC aber auch die neu entwickelte Bluetooth Low Energy (kurz: BLE) Technik, die mit Hilfe sogenannter iBeacons eine Lokalisierung in Innenräumen ermöglicht. Eine weitere Methode ist die Ortung über sogenannte WLAN-Fingerprints. Hier werden Referenzpunkte in einem WLAN-Netz bestimmt und die Signalstärken vermessen, um eine sogenannte "RadioMap" zu erzeugen. Bei der Ortung eines Gerätes, wird die aktuelle Signalstärke zu den unterschiedlichen Hotspots mit der RadioMAp verglichen und die entsprechenden Referenzpunkte bestimmt.

Es existieren auch optische Verfahren, die zum Beispiel durch die Kombination von GPS-Tracking und visuellem Tracking ein akkurates Lokalisierungssystem generiert, welches bei kritischen GPS-Abbrüchen durch optisches Tracking und Sensorfusion eine (Re)Initialisierung ermöglicht. Es gibt auch Verfahren, die mittels einer inhaltsbasierten Bildanalyse dem sogenannten SIFT-Verfahren und großen Bilddatenbanken markante Feature-Punkte in den Bildern ermitteln und vergleichen und somit eine Lokalisierung bzw. eine Blickrichtungsbestimmung ermöglichen. Da die bekannten Ansätze zur Objekterkennung mittels SIFT sehr hohe Leistungsanforderungen haben, gibt es ein vereinfachtes SIFT-Verfahren bei dem die 3D-Position eines 2D-Objektes geschätzt wird. Mit diesem Ansatz sind Verfahren zur Objekterkennung auf mobilen Geräten ohne große Prozessorleistung möglich.

Allgemein ist zu bemerken, dass bereits viele verschiedene Ansätze zur Lokalisierung bzw. Objekterkennung existieren. Viele davon sind nur im Außenbereich anwendbar, da sie ausschließlich mit GPS-Technologie arbeiten. Für die Innenraumnutzung ist meist zusätzliche Hardware notwendig, wie bei dem Beispiel der iBeacon. Die vorgestellten optischen Verfahren benötigen für die Bildanalyse mittels SIFT- oder auch dem vereinfachten SIFT-Verfahren große Bilddatenbanken und somit eine Internetverbindung. Im nachfolgenden Kapitel wird mit Hilfe dieser Erkenntnisse das Konzept für das zu entwickelnde Verfahren erarbeitet.

Kapitel 3

Konzeptentwicklung

Auf Grundlage der Analyse der Arbeiten, die sich mit ähnlichen Fragestellungen auseinandersetzen, wird nun das Konzept für die Entwicklung des Verfahrens erstellt. Auf der Basis dieser Erkenntnisse werden die Verfahrensanforderungen ausgearbeitet und der Aufbau des Verfahrens entwickelt. Hierbei spielt die Sensorik der heutigen mobilen Geräte sowie auch die Verwendung der Kamera und die Verarbeitung der aufgenommenen Bilder eine große Rolle. Es wird dargestellt, wie aus den gesammelten Daten der Sensorwerte und den Daten aus einer Bildanalyse, eine Entscheidung zur Initialisierung getroffen werden kann. Hierbei wird die Möglichkeit einer Klassifizierung mit Hilfe eines statistischen Modells beschrieben.

3.1 Einleitung

Für das Konzept des Verfahrens wird anhand möglicher Anwendungsszenarien eine Anforderungsanalyse an das in dieser Arbeit zu entwickelnde System erstellt. Die Anwendungsszenarien in dieser Arbeit werden primär aus den Anwendungsbereichen der Firma Advenco des Industriepartners des Projektes PIMAR erarbeitet. Advenco entwickelt mobile Anwendungen für verschiedene Industriefirmen zur Wartung und Überwachung von Produktionsmaschinen und deren Abläufe. Ein weiterer großer Anwendungsbereich bei Advenco betrifft die Tourismus-Sparte. Hier werden Anwendungen für Stadtführungen oder für den Besuch in einem Museum entwickelt.

3.2 Mögliche Anwendungsszenarien

In allen Anwendungsbereichen stellt die Integration von Augmented Reality einen großen Nutzen dar. Neben der Erweiterung bzw. Einblendung von wichtigen Informationen genau an der Stelle, wo sie gerade benötigt werden, ist auch die Möglichkeit, Objekte automatisch zu erkennen oder das Anzeigen von sogenannten POI (engl.: Point of Interest) von Bedeutung. Hierbei werden mit Hilfe von GPS-Daten Orte, die von Interesse sein könnten, mittels

Abbildung 3.1: Abbildung links zeigt eine Illustration für eine Werkstattanwendung. Verschiedene Bauteile einer Maschine werden dem Nutzer in der Anwendung überlagernd angezeigt oder wie in der rechten Abbildung können Arbeitsschritte visualisiert und das dafür zu verwendende Werkzeug mit eingeblendet werden.

sogenannter Billboards im Kamerabild angezeigt [Kal11]. So ergeben sich drei Szenarien, die in den nachfolgenden Abschnitten weiter erläutert werden.

- Werkstattbetrieb
- Tourismus / Stadtführung
- Museumsbetrieb

Die Unterteilung in Tourismus / Stadtführung und Museumsbetrieb hängt mit den unterschiedlichen Voraussetzungen zusammen. Da eine Stadtführung in erster Linie in großen Stadtbereichen Anwendung findet, liegt hier der Schwerpunkt auf GPS-Nutzung. Bei einem Museumsbetrieb befinden sich die verschiedenen Kunstwerke oder Exponate meist in Innenräumen in einem sehr nahem Umfeld zueinander.

3.2.1 Werkstattbetrieb

In großen Industrieanlagen, aber auch in kleinen Werkstätten, gibt es viele hoch automatisierte Abläufe, die von Maschinen erledigt werden. Jedoch müssen diese Maschinen gewartet, kontrolliert oder repariert werden. Diese manuellen Tätigkeiten setzen enormes Fachwissen voraus und erfordern meist eine umfassende Dokumentation, die vor Ort bisher nicht verfügbar ist. Hier wäre es denkbar, dass der zuständige Mechaniker mit seinem mobilen Gerät vor einer Maschine steht und diese in der Anwendung automatisch erkannt wird, um anschließend entsprechende Informationen, wie in einem Benutzerhandbuch in Form von Arbeitsanweisungen etc. bereitzustellen. Wenn der Mechaniker Reparaturarbeiten erledigen muss, können für ihn relevante Bauteile oder Arbeitsschritte im Display genau an der Stelle markiert oder angezeigt werden, wo sie sich befinden oder ausgeführt werden müssen (siehe Abbildung 3.1). Dem Mechaniker können so komplexe Arbeitsschritte mit

einfachen Animationen und weiteren Hilfestellungen Schritt für Schritt vorgegeben werden. So ist der Mechaniker in der Lage schnell und effektiv seine Arbeit mit Software- und AR-Unterstützung zu erledigen. Die Dokumentation von Wartungsarbeiten kann direkt in der Anwendung hinterlegt werden.

Technischer Aspekt

In diesem Szenario bzw. bei großen Industrieanlagen kann nicht davon ausgegangen werden, dass im gesamten Areal eine durchgängige WLAN-Vernetzung besteht. Also kann bei einer Anwendung ein Internetzugriff nicht vorausgesetzt werden, da neben dem fehlenden WLAN-Netz auch der Datenzugriff über das Mobilfunknetz in den Maschinenhallen nicht gewährleistet sein kann. So ergibt es sich, dass bei fehlendem Internetzugang die Anwendung lokal im Offline Modus arbeiten können muss. Für die automatische Erkennung könnten in diesem Szenario gegebenenfalls Marker angebracht werden, da diese im betrieblichen Ablauf keine Störung darstellen.

3.2.2 Tourismus / Stadtführung

Abbildung 3.2: Bei den beiden Abbildungen werden verschiedene Möglichkeiten für eine AR-Anwendung im Bereich Tourismus / Stadtführung dargestellt. Die linke Abbildung zeigt eine Illustration, die darstellt, wie eine Anwendung Sehenswürdigkeiten erkennt, kurze Infos anzeigt und weitere Sehenswürdigkeiten in der Umgebung in einem Radarfeld bzw. über Billboards visualisiert. Abbildung rechts zeigt eine Illustration, die visualisiert, wie eine AR-Anwendung zum Beispiel Sehenswürdigkeiten mit virtuellem Content überlagern könnte. (Illustration zusammengesetzt aus Bildmaterial http://goo.gl/zFUMZo, http://goo.gl/5r1xvl, http://goo.gl/G08g0L, http://goo.gl/lmhKst, http://goo.gl/Ufyqq7 Stand: 2. Februar 2015.)

Bei einer Stadtführung sollte die Anwendung dem Nutzer Informationen zu einzelnen Sehenswürdigkeiten anzeigen. Da sich diese über ein großes Stadtgebiet verteilen, bietet es sich an, neben der Erkennung von Sehenswürdigkeiten auch weitere Informationen als POI in der Anwendung in Form von Billboards anzuzeigen. Bei einer Stadtführung ist immer davon auszugehen, dass ein ausreichendes GPS-Signal vorhanden ist, sodass die

Ortsbestimmung einfach zu realisieren ist. Über die Kompassfunktion (Orientierungssensor wird in Abschnitt 3.4 näher betrachtet) können naheliegende, aber auch in der Ferne befindliche Objekte im Kamerabild angezeigt werden (siehe Abbildung 3.2 links). Über eine radarähnliche Anzeige im oberen Bildbereich werden die POI in der Nähe angezeigt. Wenn sich der Tourist um die eigene Achse dreht, werden über Billboards die Namen der Sehenswürdigkeiten in der zugehörigen Himmelsrichtung eingeblendet.

Neben den schon häufig umgesetzten Funktionen der POI-Anzeige bei diesen Anwendungen[1], soll weiter die Möglichkeit gegeben sein, einzelne Sehenswürdigkeiten automatisch zu erkennen und diese mit Informationen zu überlagern. Hier ist neben der einfachen Einblendung von Texten, Bildern oder Videos auch die Möglichkeit gegeben, diese mit 3D-Modellen zu erweitern oder mit interaktiven Überlagerungen, wie in der Illustration in Abbildung 3.2, rechts. Hier wird das Brandenburger Tor mit einer Darstellung des Tores aus vergangener Zeit überlagert und der Anwender kann über eine Berührung des Displays die Überlagerung auf diesem verändern. So könnte durch die Einbindung von Interaktionen des Benutzers eine Stadtführung mit Hilfe einer mobilen Anwendung erlebnisreich gestaltet werden.

Technischer Aspekt
Die Positionsbestimmung kann in erster Linie über GPS-Ortung umgesetzt werden. Marker oder andere technische Verfahren können hier nicht in Betracht gezogen werden, da zum Beispiel ein Anbringen von visuellen Markern an Sehenswürdigkeiten eine besondere Genehmigung voraussetzt. Eine Nutzung von mobilen Daten über das Mobilfunknetz könnte in Betracht gezogen werden, da in Stadtgebieten das Netz gut ausgebaut ist. Jedoch ist hier zu berücksichtigen, dass die Anwendung auch für Touristen aus dem Ausland nutzbar sein sollte, die unter anderem aufgrund von hohen Roaming-Gebühren nicht unbedingt über einen Zugang zum mobilen Internet verfügen. Daher ist hier eine Offline-Verarbeitung zu bevorzugen.

3.2.3 Museumsbetrieb

Eine Anwendung für ein Museum sollte dem Benutzer neben allgemeinen Informationen zu dem Museum, insbesondere Erklärungen und Hinweise zu den einzelnen Kunstwerken oder Exponaten zur Verfügung stellen. Eine solche Anwendung ist vergleichbar mit den Audioguide - Touren, die eine Vielzahl von Museen anbieten.[2] Jedoch bieten sich bei einer mobilen Anwendung mehrere Vorteile an. Der Museumsbesucher muss sich nicht noch ein meist kostenpflichtiges Extragerät ausleihen und hat nebenbei noch die Möglichkeit, einen visuellen Mehrwert aus einer solcher Anwendung zu erhalten. So ist es möglich, dem Besucher neben ergänzenden Informationen wie Texten, Bildern oder Videos auch

[1] http://goo.gl/LgFfsC Stand: 3. Februar 2015
[2] http://goo.gl/V1GA5S Stand: 3. Februar 2015

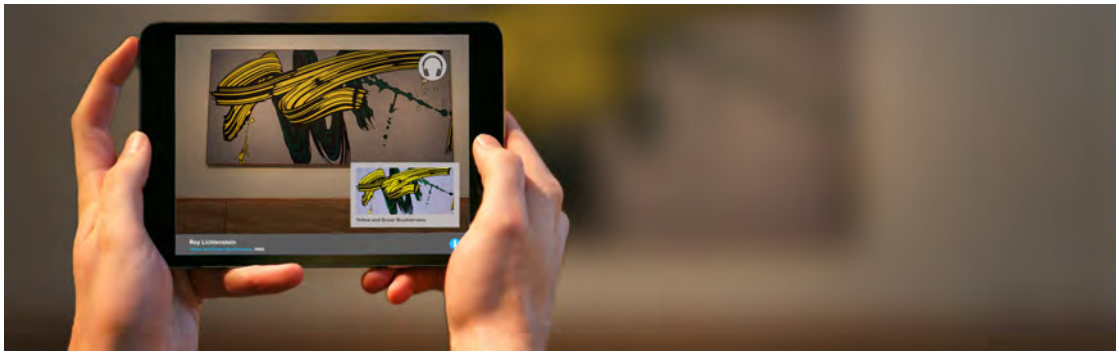


Abbildung 3.3: Illustration einer Museumsanwendung. Das mobile Gerät erkennt ein Kunstwerk in einem Museum und blendet zugehörige Informationen ein. (Illustration zusammengesetzt aus Bildmaterial http://goo.gl/5r1xvl, http://goo.gl/lyCwRW Stand: 2. Februar 2015.)

ein visuelles Erlebnis durch Überlagerung der Exponate mit 3-dimensionalem Content zu bieten. Der Besucher soll in jeder Situation ohne zusätzliche Nutzung einer Menüstruktur in der Anwendung das mobile Gerät auf ein Ausstellungsstück richten können und das System erkennt automatisch das Exponat und stellt dem Besucher den zusätzlichen Content sofort zur Verfügung (siehe Abbildung 3.3).

Technischer Aspekt
In einem Museum müssen die Gegebenheiten vor Ort berücksichtigt werden. Ein Anbringen von Markern etc. für die Ortsbestimmung ist keine Alternative. Das Erscheinungsbild von Exponaten kann in keiner Weise verändert werden. Eine Positionsbestimmung mittels GPS ist hier ebenfalls nicht immer gewährleistet, da in Innenräumen eine Ortung, wenn überhaupt, nur sehr ungenau möglich ist. Auch ist in einem Museum nicht vorauszusetzen, dass ein WLAN-Netz für einen Datenaustausch zur Verfügung steht. Ebenfalls kann eine Datenverbindung über das Mobilfunknetz nicht gewährleistet werden, was dazu führt, dass auch diese Anwendung nur mit lokalen Daten arbeiten kann.

3.3 Anforderungen

Aus den möglichen Anwendungsszenarien, die im vorigen Abschnitt beschrieben wurden und den dort erwähnten technischen Aspekten erfolgt nun eine Aufstellung der Anforderungen an das Verfahren. Hierbei wird darauf geachtet, dass das zu entwickelnde System allen Szenarien gerecht wird, um so die Möglichkeit der Verwendung des Verfahrens für alle ausgewählten Anwendungsgebiete zu gewährleisten.

Eine der wichtigsten Anforderungen ist, dass die Initialisierung der AR-Szene bzw. die Lokalisierung des Anwenders nicht über visuelle Marker geschehen soll, die zum Beispiel

an den zu erkennenden Objekten angebracht werden müssten. Gerade in Bezug auf das Szenario im Museum ist die Veränderung der vor Ort gegebenen Situation in keiner Weise denkbar, da so ein schwerwiegender Eingriff in die Museumsstruktur vom Betreiber des Museums, aber auch vom Besucher nicht gewünscht ist. Weiter soll der Besucher auch nicht in den Ablauf der Initialisierung miteinbezogen werden. Das heißt er soll nicht aktiv daran beteiligt sein, was durch das Scannen eines Markers der Fall ist. Das Verfahren sollte möglichst intuitiv und selbsterklärend funktionieren und dabei dem Nutzer das Gefühl geben, dass er das Gerät nur auf ein Objekt richten muss und so seine Position bzw. das Objekt automatisch erkannt wird. Ebenfalls soll das Verfahren ohne zusätzliche Technik auskommen. Eine zusätzliche Installation von WLAN-Netzwerken mit Location Servern oder Systemen, die mittels iBeacon die Position eines Anwenders bestimmen können, soll für die Anwendung des Verfahrens nicht benötigt werden. Um bei allen Szenarien eine uneingeschränkte Nutzung zu gewährleisten, darf das System auch nicht auf die Nutzung von Netzwerkdiensten zum Datenaustausch oder für die Datenverarbeitung zurückgreifen. Hier ist eine lokale Verarbeitung direkt auf dem Gerät des Anwenders zwingend erforderlich. Somit bleibt zur Ortsbestimmung bzw. Objekterkennung nur der Abgleich mit Sensorwerten und einer Bildanalyse. Da auch hier keine großen Bilddatenbanken abgeglichen werden können, da wie erwähnt der Netzzugriff nicht vorausgesetzt werden kann, muss ein Verfahren entwickelt werden, welches den Abgleich und die Zuordnung lokal auf dem Gerät des Anwenders umsetzt. Dabei sollte vermieden werden, große Datenmengen im lokalen Speicher des Gerätes abzulegen. Unter diesen Voraussetzungen kann man wie folgt die aus den Szenarien resultierenden Anforderungen an das Verfahren auflisten:

- Keine Verwendung von visuellen Markern
- Keine Anschaffung / Installation von weiterer Hardware (WLAN, iBeacon, etc.)
- Keine Nutzung von Datenaustausch über WLAN / Mobilfunknetz
- Nutzung der vorhandenen Sensorik und der Kamera der mobilen Devices
- Datenverarbeitung lokal direkt auf dem Gerät
- Kein großer Datenspeicherverbrauch

Aus den zusammengestellten Anforderungen ergibt sich folgendes Konzept für den Aufbau des Verfahrens. Da das System die Daten direkt auf dem Gerät verarbeiten soll und keine Daten über eine Netzwerkverbindung mit Servern ausgetauscht werden, muss das System die zu erkennenden Objekte kennen und in der Lage sein, die Auswertung mit den Live-Daten zu vollziehen. Daraus ergibt sich, dass das Verfahren aus zwei Phasen besteht. In der Anlernphase müssen die zu erkennenden Objekte in dem System hinterlegt werden und in der sogenannten Erkennungsphase muss das System die Daten, die live erfasst werden, mit den im System hinterlegten Daten abgleichen. In den beiden Phasen werden jeweils unterschiedliche Werte von den Sensoren gesammelt und gegebenenfalls für die Verwendung aufbereitet. Ebenfalls wird ein Bild mit der Kamera aufgenommen und für die Verarbeitung angepasst, indem es auf eine Vorschaugröße (Thumbnail) skaliert wird. So wird auch

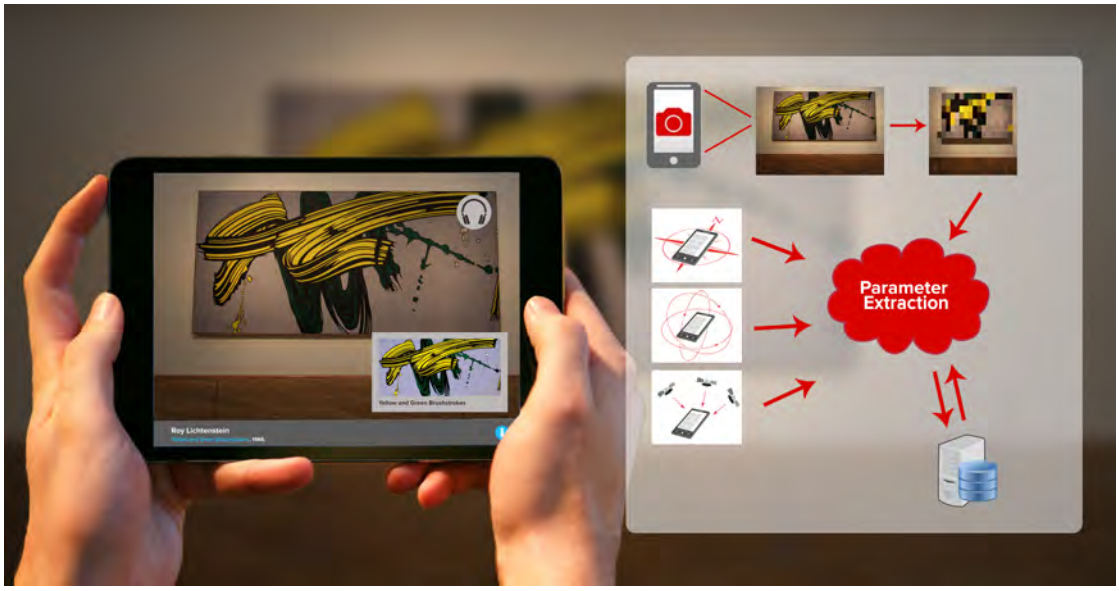


Abbildung 3.4: Überblick über die Funktionsweise des Verfahrens der AR-Initialisierung in einer Museumsanwendung auf einem Tablet. Die Illustration zeigt, dass mehrere Sensorwerte und Informationen aus einer Bilddatenanalyse verarbeitet werden, und entweder abgespeichert oder mit den abgesicherten Werten verglichen werden. (Illustration zusammengesetzt aus Bildmaterial http://goo.gl/5r1xvl, http://goo.gl/lyCwRW, http://goo.gl/E0dvHQ, http://goo.gl/Q7YnZV Stand: 2. Februar 2015.)

sichergestellt, dass das Bild, was später verarbeitet wird, immer die gleiche Größe bzw. das entsprechende Seitenverhältnis aufweist. Nach der Parameterextraction, welche in erster Linie die Aufbereitung der Sensorwerte und die Bildanalyse beinhaltet, werden die Daten entweder gespeichert oder in der Erkennungsphase mit den gespeicherten Werten verglichen. Die Funktionsweise wird in Abbildung 3.4 an dem Beispiel eines Museumsszenarios visualisiert. Wenn der Besucher das Gerät auf ein Kunstwerk richtet (hier: Roy Lichtenstein -Yellow and Green Brushstrokes, 1966), werden die aktuellen Sensorwerte mit den Werten der Bildanalyse verarbeitet und mit den im System hinterlegten Trainingsdaten verglichen. Wenn die Daten im System erkannt werden, wird in der Anwendung die entsprechende Oberfläche geladen und dem Besucher werden erweiterte Informationen im Display überlagert angezeigt.

3.4 Sensorik

Die heutigen mobilen Telefone beinhalten eine Vielzahl von Sensoren, die die unterschiedlichsten Möglichkeiten für Anwendungen bieten. Jedoch sind nicht alle Sensoren für die Ortsbestimmung bzw. Objekterkennung geeignet. Es bietet sich an, für die Klassifizierung nur die Sensorik zu verwenden, die auch eine Relevanz bei dem späteren AR-Tracking hat. Diese Sensoren sind in Abbildung 3.5 symbolisch abgebildet. Die wichtigste Information

Abbildung 3.5: Die für das Verfahren interessante Sensorik von mobilen Telefonen: Kompasssensor (3-axis magnetometer), GPS Sensor und der Gyroscope Sensor, der zur Lagebestimmung genutzt wird. (Bildmaterial auf Basis von http://goo.gl/E0dvHQ, http://goo.gl/Q7YnZV Stand: 2. Februar 2015)

ist zunächst die Blickrichtung des Gerätes. Hierzu wird mit Hilfe des Orientierungssensors, dem sogenannten Magnetometer, die Himmelsrichtung bestimmt. Neben der Ausrichtung anhand des Kompasses wird über den Drehbewegungssensor (Gyrometer) die Neigung des Gerätes in allen drei Achsen bestimmt. Hier liefert der Sensor Roll-Pitch-Yaw-Winkel zurück, über die die Neigung in allen Achsen ermittelt werden kann. Auch die Werte, die der GPS-Sensor liefert, dienen dem Verfahren zur Bestimmung. Unter der Berücksichtigung, dass GPS in Innenräumen meist keine verlässlichen Ergebnisse liefert, sollen, wenn verlässliche GPS-Daten vorliegen, diese auch in die Klassifikation integriert werden.

Die Sensordaten alleine liefern nicht genug Informationen für die Zuordnung des Objektes. So ist zusätzlich die visuelle Auswertung mit dem Kamerasystem des mobilen Telefons notwendig. Die Kamera ist in der Lage, verlässliche Informationen über eine Bildauswertung für die Objekterkennung beizusteuern. Jedoch sollen bei dem Verfahren keine Bilder gespeichert oder mit großen Bilddatenbanken verglichen werden, um die Verifizierung des Objektes zu ermöglichen. Jedoch kann das Kamerabild analysiert werden und damit können Werte gewonnen werden, zur Auswertung hinzugefügt werden. So können die Werte, die aus den Bildern gewonnen und wie Sensordaten behandelt werden. Daher werden die über Bildanalyseverfahren aus dem Kamerabild extrahierten Werte in dieser Arbeit auch als Sensorwerte eingeordnet. Im nachfolgenden Kapitel 4 werden die Methoden zur Bildanalyse näher vorgestellt und beschrieben.

3.5 Entwicklung des Verfahrens TBOC

Wie in den Anforderungen im Abschnitt 3.3 schon verdeutlicht, ist das Verfahren in zwei Phasen unterteilt. In beiden Phasen werden zum Anlernen von Objekten oder zum Erkennen von Objekten Sensordaten und Daten, die über eine Bildanalyse aus dem Kamerabild des mobilen Gerätes gewonnen werden, erfasst. Zu den Sensorwerten zählt der Kompasswert,

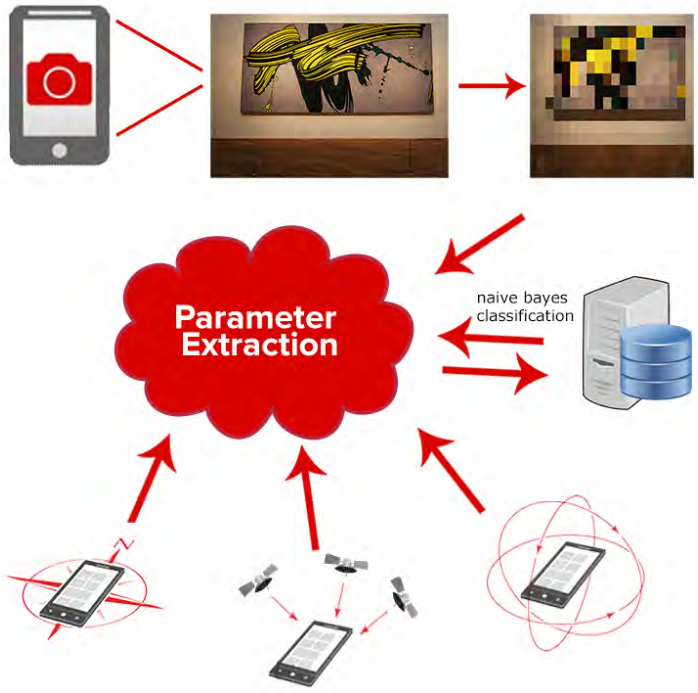


Abbildung 3.6: Eine detaillierte Abbildung des TBOC-Verfahrens. Bein Anlernen oder Erkennen von Objekten werden Sensorwerte vom Magnetometer, Groyskop und vom GPS gesammelt. Ebenfalls wird mit der Kamera des Gerätes ein Bild erzeugt und für die Verarbeitung auf Thumbnail Größe skaliert. Über eine Bildanalyse werden Informationen aus dem Bild ausgelesen. Diese Werte werden mit den Sensordaten in einen Parameterraum transformiert, bevor sie dann im System abgespeichert werden oder mit den gespeicherten Werten verglichen werden, um ein Ergebnis für die Erkennung zu erhalten. (Illustration zusammengesetzt aus Bildmaterial http://goo.gl/E0dvHQ, http://goo.gl/Q7YnZV Stand: 2. Februar 2015)

der vom Magnetometer bereitgestellt wird und die Geräteausrichtung in den drei Achsen des Gyrometers, um den Roll & Pitch des Gerätes sowie den Längen- und Breitengrad zu bestimmen, den der GPS-Sensor liefert. Wie im vorigen Abschnitt beschrieben, werden auch die aus dem Kamerabild extrahierten Daten als Sensorwerte behandelt. Damit die Erkennung von einer Vielzahl unterschiedlicher Objekte zur Echtzeit gewährleistet ist, müssen diese Sensorwerte aufgrund des beschränkten Speicherraumes weitestgehend minimiert werden, aber trotzdem einen Detaillierungsgrad aufweisen, so dass eine Verarbeitung für die Erkennung gewährleistet ist. Dafür werden die Daten zur Laufzeit in einen Parameterraum übertragen. Diese Transformation in den Parameterraum ermöglicht eine Klassifikation der Daten, wenn diese einem bestimmten Objekt zugeordnet werden können. Bei der Transformation werden sogenannte Featurevektoren gebildet, die die Datenwerte als Parameter beinhalten. Diese Featurevektoren werden in die Trainigsdatendatei geschrieben oder in der Erkennungsphase zur Klassierung verwendet (siehe Abbildung 3.6). Durch diese Vorgehensweise erhält das

Verfahren auch seinen Namen "Transformation Based Object Classification" (kurz: TBOC).

3.5.1 Statistische Auswertung

Unter der Annahme, dass die unterschiedlichen, gespeicherten Parameter im Zusammenspiel miteinander ein Objekt beschreiben, indem sie in der Anlernphase klassifiziert und somit einem Objekt zugeordnet werden, ist der Rückschluss die Hypothese, dass über einen Featurevektor mit entsprechenden Parameterdaten über eine statistische Auswertung eine Wahrscheinlichkeit der Zugehörigkeit zu einem Objekt berechnet werden kann. Hierfür müssen die Trainingsdaten analysiert und Muster und Abhängigkeiten ermittelt werden. Diese Analyse nennt man auch Data-Mining oder Maschinelles Lernen. [HK00] Helge Petersohn beschreibt den Begriff Data-Mining in seinem gleichnamigen Buch wie folgt:

> „Data Mining beschreibt [somit] einen Analyseprozeß, der aus Daten entscheidungsrelevante Zusammenhänge herleitet. Die Prozeßabschnitte werden nicht sequentiell durchlaufen, sondern weisen Rücksprünge auf. Es handelt sich um einen interaktiven Prozeß, der weitestgehend automatisiert sein sollte." [Pet05]

Also ist das Ziel von Data-Mining aus der Analyse bereits bekannter Beispiele eine Beurteilung und Zuordnung unbekannter Daten zu erreichen. Nach [CFZ09] kann Maschinelles Lernen in folgende Kategorien von Lernzielen unterteilt werden:

- Klassifikation
- Clustering
- Assoziation
- Numerische Vorhersage (Regression)

Das im Rahmen dieser Arbeit zu entwickelnde Verfahren bezieht sich auf die Klassifikation von Objekten. Also in erster Linie die Zuordnung von Attributen in Form eines Featurevektors zu vordefinierten Objekten. Die Charakteristik für die Klassifikation bildet ein Datensatz mit klassifizierten Objekten, der sogenannte Trainingsdatensatz. Die Klassifikation bezieht sich meist auf ein Attribut, jedoch können die Abhängigkeiten auch so unterschiedlich sein, dass ein Zusammenspiel mehrerer Attribute für die Zuordnung erforderlich ist. Die Ausprägung dieser Attribute muss in dem Trainingsdatensatz enthalten sein. Im nächsten Schritt versucht ein Algorithmus Muster in den Trainingsdaten zu finden, um ein Klassifikationsmodell zu erstellen. Diese Muster oder Regeln werden schließlich verwendet, um neue, unklassifizierte Featurevektoren zuzuordnen. [Mit97]

Der "Naive Bayes", einer der Bayes-Klassifikatoren, repräsentiert einen der einfachsten und ältesten Algorithmen des Maschinellen Lernens. Trotz seiner Einfachheit liefert er vernünftige Ergebnisse und wird daher auch heute noch in zahlreichen Data-Mining-Projekten

Anwendung verwendet. Sein einfacher Aufbau ohne zusätzliche Optimierungsparameter und seine leicht interpretierbaren Ergebnisse sind weitere Gründe, dass diese Methode trotz stark vereinfachter Grundannahmen heute noch Anwendung findet [WKRQ+07]. Mit dieser Methode wird auf Basis des durch den englischen Mathematiker Thomas Bayes aufgestellten Bayestheorems [Bay63] ein Klassifikator erstellt, mit dessen Hilfe neue Datensätzen, bezogen auf die Trainingsdaten, in die "wahrscheinlichste" Klasse zugewiesen werden.

3.5.2 Bayes-Klassifikator

Mit Hilfe der sogenannten Bayes-Klassifikator-Verfahren ist eine Entscheidungsfindung für oder gegen ein Objekt möglich, wenn die Voraussetzung erfüllt wird, dass die Klassifikationen über eine gemeinsame Wahrscheinlichkeitsverteilungsfunktion zusammenhängen. Beim Bayes'schen Lernen wird davon ausgegangen, dass über eine gemeinsame Wahrscheinlichkeitsverteilungsfunktion ein Zusammenhang der unterschiedlichen Attribute einer Klasse besteht und sich somit auch die verschiedenen Klassen durch ihre jeweiligen Wahrscheinlichkeitsverteilungsfunktionen voneinander unterscheiden. Aus dieser Annahme lässt sich schließen, dass Klassifizierungsentscheidungen durch die Interpretation dieser Wahrscheinlichkeiten unter Berücksichtigung von beobachteten Daten zu fällen sind [Mit97]. Um eine kompakte und übersichtliche Repräsentation der Wahrscheinlichkeitsverteilung aller einbezogenen Variablen unter Berücksichtigung bekannter bedingter Unabhängigkeiten wiederzugeben, wird hierbei ein Bayes'sches Netz verwendet.

Beweise für diverse unterschiedliche Hypothesen können mit Hilfe des Bayes'schen Anlernverfahrens verschieden gewichtet werden. Lernalgorithmen, die auf Wahrscheinlichkeiten einwirken, basieren auf Bayes'schem Lernen und selbst Algorithmen, die nicht in irgendeiner Form mit Wahrscheinlichkeiten agieren, können mit dessen Hilfe analysiert werden.

- Für bestimmte Anwendungen, wie zum Beispiel den Naive-Bayes-Klassifikator als Textklassifikator, stellen die mit Wahrscheinlichkeiten operierenden Bayes'schen Lernalgorithmen eine anerkannte wirkungsvolle Methode dar.
- Mit Hilfe der Bayes'schen Analyse kann überprüft werden, unter welchen Gegebenheiten Algorithmen, die nicht mit Wahrscheinlichkeiten operieren, die mutmaßlich wahrscheinlichste Hypothese für Trainingsdaten liefern.

Die Methoden der Bayes'schen Anlernverfahren besitzen folgende Eigenschaften:

- Lernmethoden, die auf Bayes basieren, sind anpassungsfähiger als Verfahren, die nach und nach unstimmige Hypothesen aussortieren und verwerfen, weil sich durch jedes

weitere angelernte Trainingsdatum schrittweise die Wahrscheinlichkeit für die Richtigkeit der Hypothese erhöht.

- Für das Definieren der Wahrscheinlichkeit einer Hypothese können beobachtete Daten mit Hintergrundwissen kombiniert werden.
- Die A-Priori-Wahrscheinlichkeit für jede Hypothese ist im Bayes'schen Anlernverfahren das Hintergrundwissen. Des Weiteren bildet es eine Wahrscheinlichkeitsverteilungsfunktion für die beobachteten Daten für jede mögliche Hypothese.
- Hypothesen, die Wahrscheinlichkeitsvorhersagen treffen, können mit den Bayes'schen Verfahren angeglichen werden.
- Durch die zusammengefügte Voraussage mehrerer Hypothesen, welche nach der Wahrscheinlichkeit ihres Zutreffens gewichtet werden, wird eine Neuklassifizierung durchgeführt.
- In Fällen, in denen sich Bayes'sche Verfahren nicht für ihre praktische Anwendung eignen, können die Algorithmen dennoch herangezogen werden, um einen Standard für die bestmögliche Entscheidungsfindung zu bilden, an dem sich die Algorithmen, welche dann für die praktische Anwendung genutzt werden, messen müssen.

Nachteile der Verwendung Bayes'scher Algorithmen ergeben sich daraus, dass vor der Verwendung dieser Methoden bereits eine Vielzahl an bekannten Wahrscheinlichkeiten gegeben oder durch Hintergrundwissen geschätzt werden müssen. Des Weiteren kann eine hohe Rechenleistung gefordert sein, die auf Grund der hohen Anzahl an zu berechnenden Wahrscheinlichkeiten entstehen. [Mit97]

Bayes-Theorem

Das Bayes-Theorem wurde nach Thomas Bayes (1702-1761) benannt, weil dieser es erstmals in seinem Werk 'An Essay towards solving a Problem in the doctrine of Chances', 1763 [Bay63] erörtert hat. Da im Allgemeinen die Wahrscheinlichkeit für das Eintreten eines Ereignisses davon abhängt, ob ein weiteres Ereignis bereits eingetreten ist, arbeitet das Theorem ausschließlich mit sogenannten bedingten Wahrscheinlichkeiten. Wenn A und B die einzelnen Ereignisse beschreiben, wird mit der Schreibweise $A|B$ dargestellt, dass das Ereignis A nur dann wahr sein kann, wenn B bereits eingetreten ist, also kurz A unter der Bedingung von B. Damit ist die Wahrscheinlichkeit von $A|B$ eine sogenannte bedingte Wahrscheinlichkeit. Mit dieser bedingten Wahrscheinlichkeit kann dann eine Abschätzung getroffen werden, wie glaubhaft die Aussage $A|B$ ist. Bedingte Wahrscheinlichkeiten sind ein geeignetes Mittel, eine Wahrscheinlichkeit für ein Ereignis anzugeben, für das bereits vorhandenes Wissen eine große Relevanz hat. Die bedingte Wahrscheinlichkeit, also die Wahrscheinlichkeit von A unter der Bedingung von B, wird als Formel wie folgt dargestellt:

$$p(A|B) \tag{3.1}$$

Aus dem Produktgesetz und dem Summengesetz lassen sich alle Gesetze der Wahrscheinlichkeitstheorie ableiten, die für das Bayes-Theorem benötigt werden. Nach Cox [Cox46] und Jaynes [Jay95] wird das Produktgesetz der Wahrscheinlichkeit in folgender Weise beschrieben:

$$P(AB|C) = P(A|C)P(B|AC) = P(B|C)P(A|BC) \tag{3.2}$$

mit

$$P(S|C) = 1 \tag{3.3}$$

und das Summengesetz:

$$P(A|C) + P(\bar{A}|C) = 1 \tag{3.4}$$

In dieser Formel stehen die Variablen A,B und C für allgemeine Aussagen, S ist die sichere Aussage, was bedeutet, dass sichergestellt ist, dass S definitv wahr ist, wenn die Bedingung C wahr ist und $\bar{A}$ steht für die sogenannte Negation, also die Verneinung der Aussage A. Aus dieser Formel erhält man die Definition der bedingten Wahrscheinlichkeit, wie sie in der traditionellen Statistik dargestellt wird, indem die Bedingung C im Produktgesetz vernachlässigt und nach der bedingten Wahrscheinlichkeit (Formelschreibweise: $p(A|B)$) aufgelöst wird. Wird das Produktgesetz (3.2) nach $p(A|B)$ aufgelöst, entsteht das Bayes-Theorem.

$$P(A|BC) = \frac{P(A|C)P(B|AC)}{P(B|C)} \tag{3.5}$$

In der Formel steht A für die Aussage über ein unbekanntes Ereignis, B steht für ein Statement, welches Informationen über das Ereignis A beinhaltet und C steht für zusätzliches Wissen über das Ereignis. $P(A|C)$ wird A-Priori-Wahrscheinlichkeit genannt, diese wird auch als Anfangs- oder Ursprungswahrscheinlichkeit bezeichnet, hierbei geht es um einen Wahrscheinlichkeitswert, der auf Basis von Vorwissen ermittelt wird. $P(A|BC)$ wird als A-Posteriori-Wahrscheinlichkeit bezeichnet, dieser Wert beschreibt den aktuellen Stand des Wissens bezüglich des unbekannten Ereignisses, nach der Beobachtung von Stichproben. Mit $P(B|AC)$ wird die Wahrscheinlichkeit bezeichnet. Die Anfangswahrscheinlichkeit (A-Priori-Wahrscheinlichkeit) wird also durch die Wahrscheinlichkeit verändert, die Informationen bezüglich des unbekannten Ereignisses enthält, daraus ergibt sich die A-Posteriori-Wahrscheinlichkeit. [Koc13] [Sac84]

Naive-Bayes-Klassifikator

Um nun die Komplexität bei der Berechnung der Wahrscheinlichkeiten zu reduzieren, wird der Naive-Bayes-Klassifikator herangezogen. Seinen Namen erhält der Naive-Bayes-Klassifikator daher, dass er voraussetzt, dass die Unabhängigkeitsannahme wahr ist. Bei der Unabhängigkeitsannahme wird davon ausgegangen, dass bei der Klassifizierung die Auswirkungen, die ein Merkmal hat, nicht abhängig von der Charakteristik anderer Eigenschaften ist. Da diese Annahme in der Realität selten zutrifft ist sie naiv. Die Berechnung der Wahrscheinlichkeit lässt sich auf diese Weise stark vereinfachen. [HK00] Bei der Verwendung des Naive-Bayes-Klassifikators wird eine Vielzahl von Klassen gebildet. Es werden Trainingsdaten angelernt, welche in sogenannten Featurevektoren gespeichert

werden, diese Featurevektoren wiederum werden den Klassen zugeordnet. Die Zuweisung der Featurevektoren zu den Klassen kann entweder per Hand oder automatisch durch einen sogenannten Klassifikator durchgeführt werden. Bei der Klassierung wird ein Objekt aufgrund einer Übereinstimmung der Featurevektoren einer entsprechenden Klasse zugeordnet, dabei soll die Quote einer falschen Zuweisung möglichst gering gehalten werden. In Bezug auf die Benutzung des Bayes-Theorems bedeutet das, dass das unbekannte Objekt genau der Klasse zugewiesen werden soll, die am dichtesten an der Wahrscheinlichkeit liegt, welche den aktuellen Wissensstand bezüglich des Ereignisses beinhaltet, also die sogenannte A-Posteriori-Wahrscheinlichkeit "maximiert". Daher erhält diese Methode den Namen Maximum-A-posteriori-Lösung oder kurz MAP-Lösung.

Wie in den Anforderungen in Abschnitt 3.3 schon dargelegt, wird das TBOC-Verfahren in zwei Phasen unterteilt, die Anlern- und Erkennungsphase. Daher liegt es nahe, auch das schnelle und effiziente Klassifizierungverfahren mittels eines Naive-Bayes-Klassifikators zu wählen, da damit die Voraussetzungen für diese Klassifikation schon gegeben sind.

Die gesammelten Sensordatenwerte ergeben die Instanzen bzw. die Featurevektoren, die in den Trainingsdaten gespeichert werden. Aus dem Trainingsdatenset lässt sich dann mittels des Naive-Bayes-Klassifikators das statistische Datenmodell erstellen. In der Erkennungsphase wird dann die aktuelle Instanz als Featurevektor mit dem Modell verglichen und das zu erkennende Objekt klassiert.
Jedoch ist zu beachten, dass der Naive-Bayes-Klassifikator immer nur die mögliche Wahrscheinlichkeit für ein angelerntes Objekt errechnet. Wenn die Werte in einem Featurevektor nicht mit einem Objekt übereinstimmen, gibt das System allerdings trotzdem das Objekt, welches mit der größten Wahrscheinlichkeit passen könnte, zurück. Der Ansatz einer Lösung dieses Problems wird in Kapitel 4 beschrieben.

3.6 Zusammenfassung

Auf der Grundlage der Anwendungsbereiche des Industriepartners Advenco ergeben sich die folgenden Einsatzbereiche für mögliche Anwendungsszenarien des TBOC-Verfahrens:

- Werkstattbetrieb
- Tourismus / Stadtführung
- Museumsbetrieb

In jedem dieser Szenarien ergeben sich unterschiedliche Anforderungen an das System. Die wichtigsten Aspekte hierbei sind die Offline-Verarbeitung und der Verzicht auf die Nutzung von visuellen Markern (wie z.B: QR-Codes etc) zur Initialisierung. Da in vielen Bereichen eine Nutzung von Internettechnologie über Mobilfunk oder WLAN nicht sichergestellt werden kann, wird die Möglichkeit einer Offline.Anwendung des Systems vorausgesetzt.

Ebenso ist es in vielen der Einsatzbereiche nicht möglich Marker direkt an den Objekten oder in der Nähe dieser anzubringen, warum auf eine markerlose Initialisierung zurückgegriffen werden muss. Des Weiteren wird die Anschaffung / Installation weiterer Hardware zur Anwenderlokalisierung für das Verfahren ausgeschlossen. Die Initialisierung soll mittels der vorhandenen Sensorik in mobilen Geräten geschehen, wobei großer Datenspeicherverbrauch vermieden werden muss, um den Datenspeicher des Anwendergerätes nicht zu stark zu belasten. Die verwendete Sensorik setzt sich aus dem Magnetometer, dem Gyroskop und GPS zusammen. Um auch optische Werte mit hinzuzuziehen, werden Werte, die über Bildanalyseverfahren aus dem Kamerabild extrahiert werden, ebenfalls als Sensorwerte betrachtet und weiterverwertet. Der Name des Verfahrens "Transformation Based Object Classification" (kurz: TBOC) bezieht sich auf die Transformation der Sensordaten-Werte in einen Parameterraum. Das Verfahren besteht aus zwei Phasen. In der "offline"-Phase, der Erkennungsphase, werden die zu erkennenden Objekte sensorisch vermessen und die gewonnenen Sensorparameter in Form eines Featurevektors in der Trainingsdaten-Datei abgespeichert. Aus den Trainingsdaten wird über einen Bayes-Klassifikator ein statistisches Modell erzeugt. Als Klassifikation wurde der Naive-Bayes-Klassifikator gewählt, da er einfach zu handhaben ist und solide Ergebnisse liefert. In der zweiten Phase, der Erkennungsphase, werden wieder Sensor-Daten ermittelt und in den Parameterraum transformiert. Diesmal wird der Featurevektor mit dem Modell verglichen und das gesuchte Objekt klassiert.

Im nächsten Kapitel wird anhand des hier beschriebenen Konzeptes, die Entwicklung der prototypischen Applikation für das TBOC-Verfahren dargelegt. Hierbei wird eine Anwendung in Form einer Android-App entwickelt, die die Funktionsweisen des Verfahrens repräsentiert. Sie bietet die Möglichkeit zum Anlernen und auch zum Erkennen von Objekten.

Kapitel 4

Anwendungsentwicklung

Auf Basis des aus der vorangegangenen Anforderungsanalyse resultierenden Konzeptes wird nun die prototypische Implementierung des TBOC-Verfahrens erörtert. Als erstes wird die zu Grunde liegende Infrastruktur dargestellt. Hierbei wird auf die benutze AR-Entwicklungsumgebung Metaio und die Data-Mining Bibliothek WEKA eingegangen und das Zusammenspiel dieser innerhalb der Anwendung erläutert. Anschließend werden die beiden Phasen des TBOC-Verfahrens, die sogenannte Anlernphase und die Erkennungsphase, wie sie aus dem Konzept hervorgehen und auch durch die Zuhilfenahme eines Bayes-Klassifikators, wie in 3.5.1 erläutert wurde, vorausgesetzt werden, detailliert beschrieben und die funktionelle Umsetzung geschildert. Außerdem wird auf die Notwendigkeit eines sogenannten Dummy-Objektes bzw. NULL-Objektes eingegangen. Das Problem der Fehlerkennungen wird beschrieben und der Lösungsansatz hierfür geschildert. Abschließend wird in diesem Kapitel noch ein Überblick über die restlichen Systemfunktionen der TBOC-Anwendung gegeben.

4.1 Einleitung

Die TBOC-Applikation ist eine prototypische Implementierung des gleichnamigen Verfahrens, welche dazu dient zu überprüfen, ob das entwickelte Verfahren den Zielsetzungen dieser Arbeit bezüglich der Objekterkennung für die Initialisierung von markerlosem 3D-Tracking gerecht werden kann. Des Weiteren kann die Anwendung durch die modularisierte Umsetzung später für das reale Anwendungsszenario in die Mathematkum-Applikation integriert werden. Die App wurde in dem Entwicklungstool Eclipse[1] mit den "Android Developer Tools"[2] (kurz: ADT) entwickelt. In der Anwendung kann die Erkennung ebenfalls vollzogen werden, jedoch in erster Linie nur zur Kontrolle mit einer Ausgabe eines Benutzerhinweises, welcher ausgibt, welches Objekt bzw. Exponat erkannt wurde. Wenn das Verfahren in weitere Anwendungen integriert wird, wird hier jeweils nur die Android-Klasse für die Objekterkennung benötigt. Diese bietet eine Schnittstelle, in der für jedes erkannte Objekt gewünschte, beliebige Android-Klassen aufgerufen werden können. Weiter bietet die TBOC-Anwendung Möglichkeiten zur Objektverwaltung. Hier können Trainingsdaten-Dateien angelegt sowie die

[1] https://eclipse.org/ Stand: 14. Februar 2015
[2] http://developer.android.com/tools/help/adt.html Stand: 14. Februar 2015

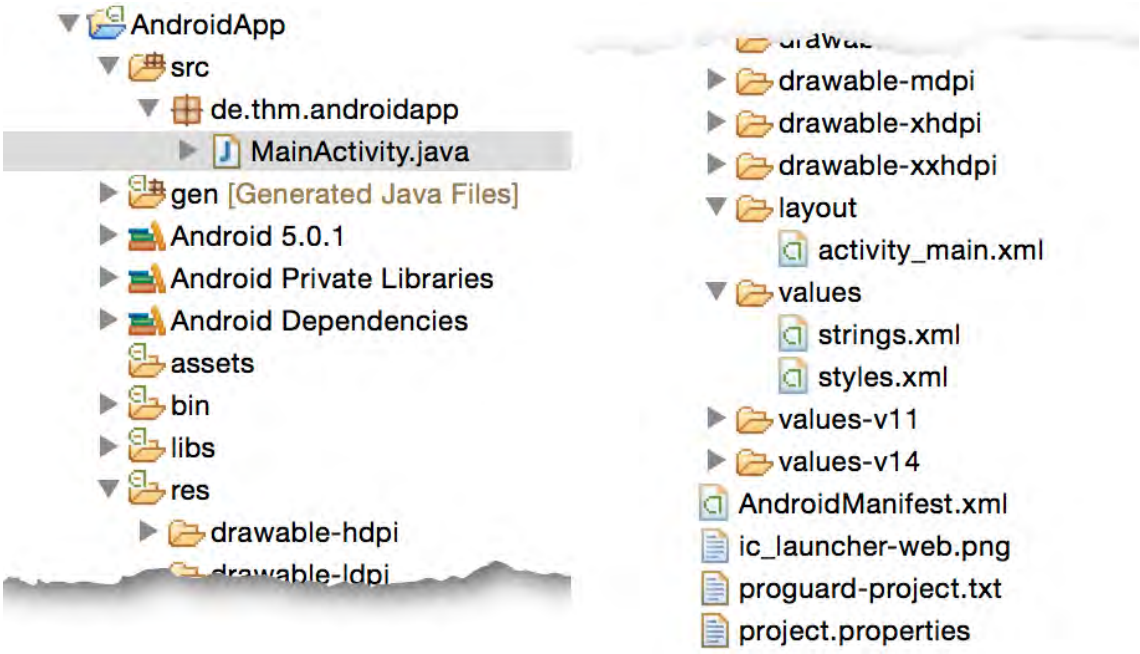


Abbildung 4.1: Übersicht über die Datenstruktur eines Android-Projektes in Eclipse (Screenshot aus dem Entwicklertool Eclipse).

Anzahl und die Benennung der zu erkennenden Objekte definiert werden. Ebenfalls können die Trainigsdaten-Dateien gelöscht oder zur Verwendung ausgewählt werden.
In den nachfolgenden Abschnitten wird auf die Erstellung der Anwendung und ihrer einzelnen Klassen und Funktionen eingegangen. Um zunächst einen Überblick über die Entwicklung von Android in Eclipse zu erhalten, werden die für das Verständnis dieser Arbeit notwendigen Grundlagen kurz erklärt, damit in den folgenden Abschnitten die Funktionsweise ohne weitere Erläuterungen verständlich ist.

4.2 Android und Eclipse

Ein "Android-Application-Projekt" besteht aus einer Vielzahl von Dateien, die zusammen das Android-Projekt beschreiben (siehe Abbildung 4.1). Vergleichbar mit einem Java-Projekt in Eclipse, befinden sich die Klassen in dem `src`-Ordner. Hier werden sie weiter in Packages unterteilt. Die androidspezifischen Klassen werden als `Activity` bezeichnet und stellen die Funktionalitäten der App bereit. Die Benutzeroberfläche wird in einer XML-Datei beschrieben, die unter dem Resourcen-Ordner `res` unter `layout` abgelegt wird. Grafiken und Icons werden in den `drawable`-Ordnern hinterlegt. In Android soll so viel wie möglich mehrfach verwendet werden können. So sollen zum Beispiel Texte, wie das Label eines Buttons, nicht direkt in den Quellcode geschrieben werden, diese können in der `strings.xml`, die im Ordner `values` abgespeichert ist, über eine String-Definition hinterlegt werden. Dieser Ansatz macht es leichter, die Anwendung in mehrere Sprachen zu übersetzen. Mit dem gleichen

Ansatz könnten auch Farben definiert werden, um diese an unterschiedlichen Stellen in der Anwendung zu verwenden. Hierfür muss eine entsprechende `Resources`-Datei mit dem Namen `colors.xml` unter dem Ordner `values` abgelegt werden. Die `AndroidManifest.xml` ist das Herzstück einer Android-Anwendung. Diese Datei muss in jeder Anwendung mit genau diesem Namen im Stammverzeichnis liegen. Hier werden wichtige Informationen und Einstellungen hinterlegt. Beispielsweise wird in dieser Datei die Versionsnummer, der Name der Anwendung und das Icon angegeben. Ebenfalls werden die Anwendungsberechtigungen definiert. Alle anwendungsrelevanten Activitys müssen im Bereich `Application` aufgeführt werden.
Der Aufbau einer Manifestdatei ist im folgenden Code-Beispiel 4.1 zu sehen. Neben dem `versionCode` und dem `versionName` wird noch die `minSdkVersion` sowie die `targetSdkVersion` angegeben, die über den möglichen Funktionsumfang der Anwendung entscheidet, je nachdem für welche Android-Version die Applikation ausgelegt ist. Das App-Icon und der App-Name werden gesetzt und der Style des `Theme` wird definiert. Themes sind Android-Mechanismen zum Anlegen eines einheitlichen Stils für die Anwendung oder Activity. Den in einer Anwendung verwendeten Activitys können im Manifest noch Optionen oder einzelne Berechtigungen (engl.: Permissions) übergeben werden.

Listing 4.1: AndroidManifest.xml

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.thm.androidapp"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8" android:targetSdkVersion="21" />
     <application
         android:allowBackup="true"
         android:icon="@drawable/ic_launcher"
         android:label="@string/app_name"
         android:theme="@style/AppTheme" >
         <activity android:name="MainActivity" android:screenOrientation="portrait"></activity>
    </application>
</manifest>
```

Um eine Klasse als Activity zu definieren, muss sie von der Android-Basisklasse `Activity` abgeleitet werden. Neben der einfachen Activity gibt es auch modifizierte Versionen, wie zum Beispiel die `ActionBarActivity`, die es ermöglicht eine Toolbar in der Activity zu platzieren oder die `PreferenceActivity`, die für das Anzeigen von Benutzereinstellungen genutzt wird. Eine Android-Activity-Klasse steht so immer für eine einzelne Benutzeraktion, die der Anwender ausführen kann. Aus diesem Grund übernimmt eine Activity auch das Bereitstellen eines Fensters in dem die Benutzeroberfläche platziert werden kann. Benutzeroberflächen werden über Layout-XML-Dateien beschrieben und können mittels `setContentView (View)` an eine Activity übergeben werden. Eine Activity kann mehrere Zustände annehmen. Wenn eine Activity den Zustand wechselt, können mit Hilfe von `Callback-Methoden` Operationen durchgeführt werden (siehe Abbildung 4.2).

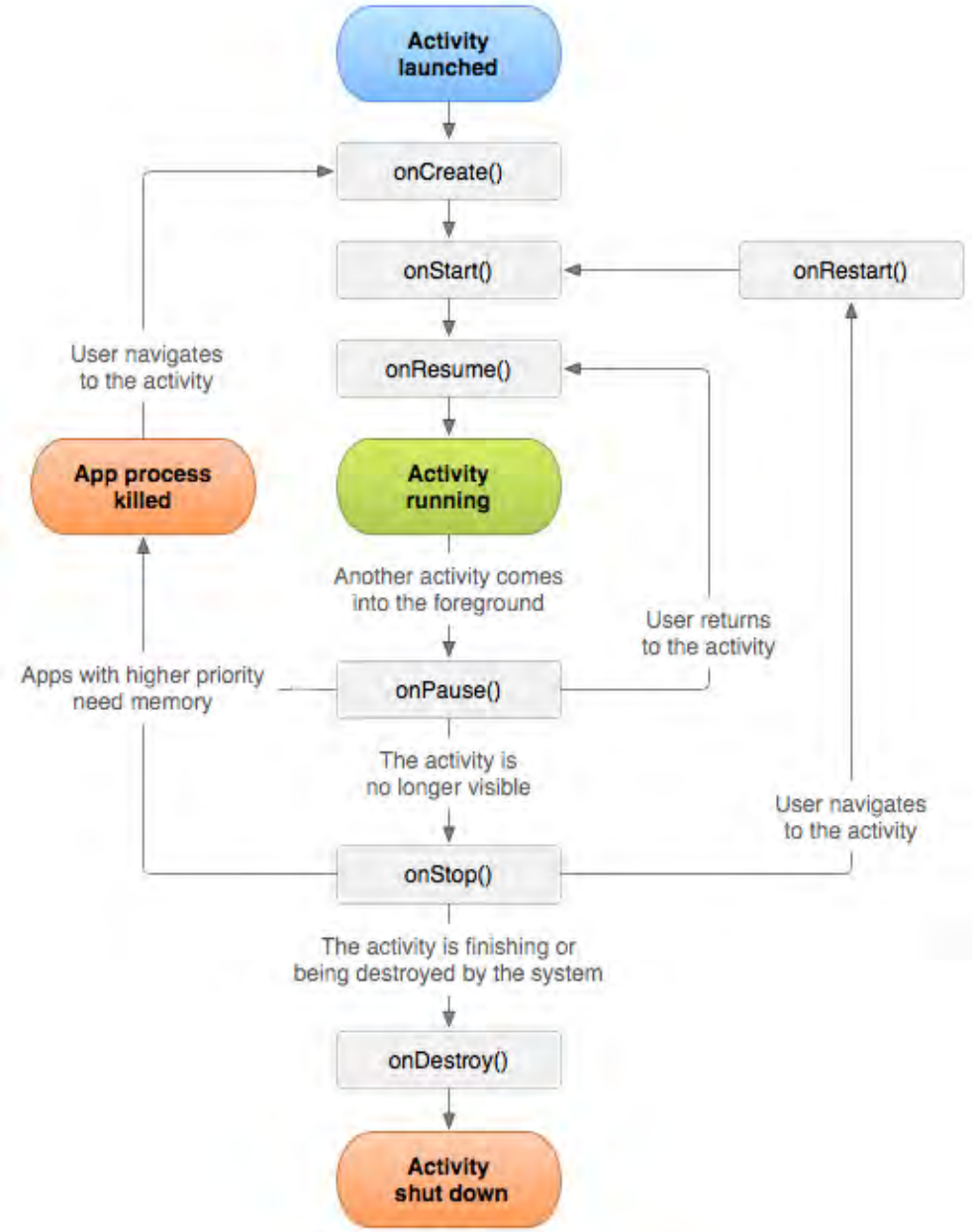


Abbildung 4.2: Die wichtigsten Zustände einer Android-Activity. Die Rechtecke stellen Callbackmethoden dar, die implementiert werden können, um Operationen durchzuführen, wenn der Zustand der Activity sich ändert. Die farbigen Ovale stehen für die Zustände der Activity (http://developer.android.com/reference/android/app/Activity.html Stand: 14. Februar 2015).

Der Grundaufbau einer Android-Activity ist im folgenden Code-Beispiel 4.2 dargestellt.

Listing 4.2: Android Activity

```
package de.thm.androidapp;

import android.app.Activity;
import android.os.Bundle;

public class MainActivity extends Activity {
        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);
        }

        @Override
        protected void onPause() {

        }

        @Override
        protected void onResume() {
                super.onResume();
        }
}
```

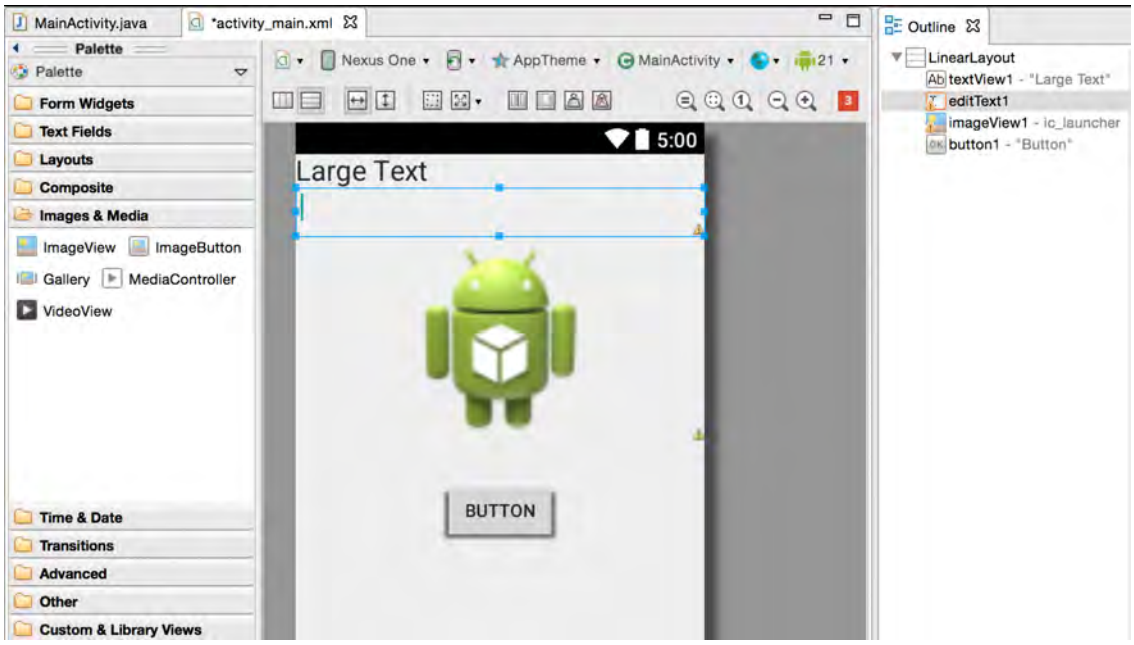


Abbildung 4.3: Im Layout-Editor können die Benutzeroberflächen angeordnet werden. Hierfür steht die Kombination verschiedener Layouts wie Grid, Linear, Relative etc. zur Verfügung. Weitere Inhalte wie Texte, Eingabefelder, Bilder oder Buttons können nach Belieben auf der Fläche platziert werden (Screenshot aus dem Entwicklertool Eclipse).

In der onCreate-Methode wird üblicherweise die Benutzeroberfläche bzw. das Layout mit der Methode setContentView(View) übergeben. Die Layoutdatei lässt sich in Eclipse in einem graphischen Editor bearbeiten oder man hat die Möglichkeit, die Beschreibung direkt im Quellcode der XML-Datei zu vollziehen (siehe Quellcode-Beispiel 4.3). Zuerst wird in einer Layoutdatei ein Layout definiert, hier stehen GridLayout, LinearLayout, RelativeLayout etc. zur Verfügung. Diese Layouts können beim Gestalten der Benutzeroberfläche auch verschachtelt eingesetzt werden. Weitere Inhalte wie Texte, Eingabefelder, Bilder, Listen und Buttons etc. können nach Belieben im Zusammenspiel der einzelnen Layouts auf der Fläche platziert werden (siehe Abbildung 4.3). Die einzelnen Komponenten können über eine Vielzahl von Parametern definiert werden, wie in dem Quellcodebeispiel 4.3 zu sehen ist. Die Höhe oder Breite einer Komponente kann über eine Angabe in dp bzw. dip (engl.: Device independent pixel; dt.: Geräteunabhängige Pixel) oder mit den Konstanten FILL_PARENT, MATCH_PARENT, WRAP_CONTENT definiert werden, die dafür sorgt, dass sich die Größe an den Inhalt oder die Größe des Eltern-Objektes bzw. der Fenstergröße anpasst. Wie im oberen Abschnitt schon erwähnt, sollten Texte immer über die String.xml eingefügt werden. Dies wird mit der Deklaration @string/string_name erreicht. Hierbei ist der string_name der Name, der in der XML-Datei zuvor festgelegt wurde. [Gar11] [MS14]

Listing 4.3: Layout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/default_text"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="text" >
        <requestFocus />
    </EditText>

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="match_parent"
        android:layout_height="160dp"
        android:layout_gravity="center"
        android:layout_marginBottom="30dp"
        android:layout_weight="0"
        android:contentDescription="@string/image_descripton"
        android:scaleType="fitCenter"
        android:src="@drawable/ic_launcher" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/button_text" />
</LinearLayout>
```

4.3 Metaio SDK

Das SDK von Metaio wird als Bibliothek in das Android-Projekt eingebunden und stellt so Klassen zur Verfügung, die es ermöglichen, AR-Activitys anzulegen und somit Augmented Reality-Funktionen in die App einzubinden. Das TBOC-Verfahren nutzt selber nicht die Funktionalität des Metaio SDK's, da es lediglich die Vorentscheidung zur Initialisierung von Augmented Reality liefert. Das bedeutet, dass, wenn in dem Verfahren ein Objekt erkannt wird, eine gewünschte Activity mit einem sogenannten "Intent" geladen wird (siehe Quellcode-Beispiel 4.4). Wenn an dieser Stelle eine AR-Funktion erwünscht ist, kann diese über eine AR-Activity geladen werden.

Listing 4.4: Activity-Aufruf über ein Intent

```
// Start the activity connect to the
// specified class

Intent i = new Intent(this, ActivityName.class);
startActivity(i);
```

Da die Umsetzung einer AR-Activity auf diese Weise auch mit der Entwicklung und Nutzung der TBOC-App in Verbindung steht, wird im folgenden Abschnitt auf die Besonderheiten und die Funktionalitäten von AR-Activitys eingegangen. Metaio setzt bei der Nutzung seines SDK's eine Registrierung der Android-App im Entwicklerportal[3] von Metaio voraus. Hierfür wird über den "App Namen" und dem "App Identifier" ein "SDK Signature Schlüssel" generiert. Dieser muss in der `String.xml`-Datei unter dem Namen `metaioSDKSignature` hinterlegt werden (siehe Quellcode-Beispiel 4.5).

Listing 4.5: Hinterlegen der metaioSDKSignature

```
<resources>
        <string name="metaioSDKSignature">
                ROc5VM4O0P350/9gBOoYfl8Agm2KMcXWAkoXd82y8QA=
        </string>
        ...
</resources>
```

Um in einer Android-Activity die AR-Funktionailtäten anzuwenden, wird die Klasse von der Metaio-Klasse `ARViewActivity` abgeleitet. Eine `ARViewActivity` benötigt eine Vielzahl von implementierten Methoden (siehe Quellcode-Beispiel 4.6). Mit der Methode `getGUILayout()` wird eine interne Methode zur Verfügung gestellt, um das Layout der Activity zu übergeben. Die Methode `loadContents()` ist dafür da den für das AR-Tracking benötigten Content, wie beispielsweise die Trackingdaten, Label oder 3D-Modelle zu laden. Weiter gibt es noch eine Callback-Methode, welche Callbacks vom Metaio SDK abfängt und die Möglichkeit gibt, entsprechende Operationen auszuführen. Hier kann zum Beispiel abgefangen werden, dass eine Animation beendet wird. Die `onGeometryTouched`-Methode ermöglicht das Abfangen eines Touch-Ereignisses (wörtlich: das Drücken auf etwas) auf ein AR-Objekt und das Ausführen einer Operation.

[3] https://my.metaio.com Stand: 15. Februar 2015

Listing 4.6: Aufbau einer AR-Activity

```
public class ARActivity extends ARViewActivity {

        @Override
        protected int getGUILayout() {
                // TODO Auto-generated method stub
                return return R.layout.ar_activityLayout; //Layout XML
        }

        @Override
        protected IMetaioSDKCallback getMetaioSDKCallbackHandler() {
                // TODO Auto-generated method stub
                return null;
        }

        @Override
        protected void loadContents() {
                // TODO Auto-generated method stub

        }

        @Override
        protected void onGeometryTouched(IGeometry arg0) {
                // TODO Auto-generated method stub

        }

}
```

Das Herzstück einer AR-Klasse ist die Methode `loadContents()`. Hier gibt es die Möglichkeit in wenigen Schritten eine einfache AR-Activity zu erzeugen:

- die Tracking-Referenz-Datei an das SDK zu übergeben
- das 3D-Modell oder andere Objekte zur Überlagerung zu laden
- wenn nötig die Skalierung, Rotation und Position anzupassen

Das Laden einer Tracking-Referenz-Datei geschieht über die Methode `setTrackingConfiguration()`. Hier wird die Konfigurations-XML für das Tracking als `trackingDataFile` übergeben (siehe Quellcode-Beispiel 4.7). Mit dieser XML-Datei wird die Tracking-Komponente des SDKs konfiguriert. Sie beinhaltet Informationen über den Marker und die Trackinggenauigkeit (Vergleich Anhang A).

Listing 4.7: setTrackingConfiguration

```
// Load desired tracking data
metaioSDK.setTrackingConfiguration(trackingDataFile);
```

Anstelle der XML kann des Weiteren eine `3dmap`-Datei übergeben werden. Dabei handelt es sich um eine Tracking-Datei, welche für das markerlose 3D-Tracking eingesetzt wird. Inhalt dieser Datei ist die Beschreibung der aufgezeichneten 3D-Punktwolken. Um reale Objekte für das markerlose 3D-Tracking zu nutzen, werden diese mit der Metaio-Anwendung `Toolbox` gescannt und markante Featurepunkte an den Objekten in einer 3D-Punktwolke erfasst. Eine genaue Beschreibung, wie solche Daten aufgenommen und verarbeitet werden, folgt in Kapitel 5.

Die Objekte, die in einer AR-Activity überlagert werden können, sind vom Datentyp `IGeometry`. Hierbei können diese Objekte Bilder, Videos oder 3D-Modelle sein. Um diese Objekte zu laden, erstellt das Metaio-SDK daraus 3D-Geometrien. Der folgende Quellcode 4.8 zeigt den entsprechenden Methodenaufruf. Hier wird der Pfad zur Datei als `metaioModel` mit übergeben.

Listing 4.8: createGeometry

```
// Loading 3D geometry
IGeometry mModel createGeometry(metaioModel);
```

In vielen Fällen sind noch Anpassungen an den sich überlagernden Objekten notwendig. Der Marker, ob Bild, ID oder auch beim markerlosen Tracking die Punktwolken-Datei, stellt bei der AR-Activity die Referenzgröße dar. Das Objekt wird im Verhältnis vom Marker in der Szene dargestellt. Hierbei ist der Mittelpunkt des Markers auch der Mittelpunkt der AR-Szene. Gerade beim markerlosen Tracking kommt es hier zu Schwierigkeiten, da der Ursprung sowie die Ausrichtung bzw. die Lage der Punktwolke im 3D-Raum nicht mit der realen Welt übereinstimmt. So ist eine Korrektur der Größe, Ausrichtung und Position der Objekte notwendig. Im folgenden Beispiel-Quellcode 4.9 wird eine exemplarische Anpassung aufgezeigt.

Listing 4.9: Anpassungen an dem Model

```
// Set geometry properties
mModel.setScale(new Vector3d(0.8f, 0.8f, 0.8f));
mModel.setRotation(new Rotation(new Vector3d(0f,0f,-0.28f)));
mModel.setTranslation(new Vector3d(-2.0f, -0.5f, 0.0f));
```

Die Rotation wird über den Radiant angegeben. Die Einheit der Translation wird in Millimetern beziffert. Bei "relative-to-screen"-Objekten, kann die Einheit ohne weiteres als Pixel interpretiert werden. Objekte können wie im Beispiel 4.10 "relative-to-screen" gesetzt werden, um sie aus dem AR-Koordinatenraum in den Bildschirmkoordinatenraum zu transferieren. Die übergebene Konstante gibt den "Anker" an, wo das Objekt an den Bildschirm gesetzt werden soll. Im folgenden Beispiel ist es zentriert in der Mitte des Bildschirmes.

Listing 4.10: setRelativeToScreen

```
// Set geometry relative to screen
mModel.setRelativeToScreen(IGeometry.ANCHOR_CC);
```

Für die Verwendung von AR-Tracking muss keine Anpassung an einer Layoutdatei vorgenommen werden. Lediglich wenn Buttons zur Interaktion gewünscht sind, müssen diese in der entsprechenden Layout XML-Datei positioniert werden. Die weiteren Anpassungen, wie das Anlegen der View für das Kamerabild, werden von dem Metaio-SDK übernommen. Hier ist zu beachten, dass über den Quellcode die selbst gesetzten Elemente nicht mehr angesprochen werden können, da durch die Verarbeitung des Metaio-SDKs die Verknüpfung aufgehoben wird. Lediglich ein Button-Event kann in der Activity abgefangen und verarbeitet werden.

4.4 WEKA for Android

Für die Klassifizierung und Klassierung der Parametervektoren wird bei dem TBOC-Verfahren das von der *University of Waikato*, Neuseeland, entwickelte `Data Mining-Paket` "Waikato Environment for Knowledge Analysis" (kurz: WEKA) verwendet [WF05] [HFH+09]. Die Open-Source-Software liefert alle notwendigen statistischen Implementierungen, sie enthält WEKA-Funktionen für die Datenaufbereitung, die Regressionsanalyse, die Klassifikationsverfahren, die Clusteranalyse und die Visualisierung. Daneben können auch neue Methoden für maschinelles Lernen erstellt werden. Es gibt kaum ein Data-Mining-Verfahren, das nicht in WEKA implementiert ist. WEKA stellt auch ein Benutzerinterface zur Verfügung, in dem sich verschiedene Algorithmen als Workflows zusammenfügen lassen. Weiter kann WEKA auch als Bibliothek in ein Java-Projekt eingebunden werden und so in eigene Anwendungen integriert werden. Es gibt ein auf Java basierendes Projekt "WEKA-for-Android"[4], entwickelt um eine Portierung von *WEKA 3* zu der Android-Plattform zu gewährleisten. Hierfür wurden die GUI-Elemente aus WEKA entfernt, damit es unter Android verwendet werden kann. Eine hundertprozentige Funktionalität wird vom Entwickler nicht sichergestellt, jedoch ist das Projekt im Stande, in Android-Projekte eingebunden zu werden und die Data-Mining-Funktionen können genutzt werden [LCTY12].

4.4.1 Dataset

Ein grundlegendes Konzept beim maschinellen Lernen ist die Verwendung von Sätzen aus Datenelementen. Eine Datenmenge entspricht in etwa einer zweidimensionalen Tabellenkalkulations- oder Datenbanktabelle. In WEKA wird es über die `weka.core.Instances`-Klasse realisiert. Ein Datensatz entspricht hier einer Sammlung von Trainingsdaten, die jeweils die Klasse `weka.core.Instance` repräsentieren. Jede Instanz besteht aus einer Anzahl von Attributen, diese können unter anderem nominal, numerisch oder eine Zeichenfolge sein.

- `weka.core.Instances` - hält einen kompletten Datensatz. Diese Datenstruktur ist zeilenbasiert. Auf einzelne Zeilen wird über einen 0-basierten Index über die `Instanz(int)` Methode zugegriffen. Informationen zu den Spalten erhält man über die `Attribut(int)`-Methode. Diese Methode liefert Objekte vom Typ `weka.core.Attribute`.

- `weka.core.Instance` - repräsentiert eine einzelne Zeile. Es ist im Grunde ein Behälter für Feature-Daten. Die Klasse selber enthält keine Informationen über den Typ der Spalten, hier wird immer ein Zugriff zu einem `weka.core.Instances`-Objekt benötigt.

- `weka.core.Attribute` - hält die Datentyp-Informationen über eine einzelne Spalte in der Datenmenge. Es speichert den Typ des Attributs sowie die Beschriftungen für nominale Attribute, die möglichen Werte für String-Attribute oder Datensätzen für relationale Attribute (diese sind Objekte vom Typ `weka.core.Instances`).

[4] https://github.com/rjmarsan/Weka-for-Android Stand: 17. Februar 2015

Die Instanzen-Klasse wird extern als ARFF-Datei repräsentiert. Eine ARFF (Attribut-Relation File Format)-Datei ist eine ASCII-Textdatei. Wie im Beispiel 4.11 exemplarisch dargestellt, besteht die ARFF-Datei aus einem "Header", der den Namen der Relation, eine Liste der Attribute (die Spalten in den Daten) und ihre Datentypen enthält. Die einzelnen Daten werden im "@data"-Bereich als Kommas getrennte Liste angehängt.

Listing 4.11: WEKA ARFF-Datei

```
@relation Location

@attribute Object {'Exponat 1',Exponat 2','Exponat 3'}
@attribute OrientationDegree numeric
@attribute OrientationPitch numeric
@attribute OrientationRoll numeric
@attribute GPSLatitude numeric
@attribute GPSLongitude numeric
...

@data
'Exponat 1',102.2,-66,2,0,0,...
'Exponat 2',269.6,-66.6,0.2,50.581234,8.666348,...
'Exponat 2',265,-65.2,-0.2,50.581327,8.666533,...
...
```

Die @relation Deklaration

Der Relationsname ist der interne Name des Datensatzes. Dieser wird in der ersten Zeile der ARFF-Datei definiert.

```
@relation <relation-name>
```

Hierbei ist <relation-name> ein String, der in Hochkommas geschrieben wird, wenn der Name aus mehreren Wörtern besteht.

Die @attribute Deklaration

Die Attribut-Deklaration nimmt die Form einer geordneten Folge von @attribute Statements an, diese Statements definieren die ihnen zugehörigen Attribute jeweils eindeutig über ihren Attribut-Namen und ihren Datentyp. Die Reihenfolge der Attribute bestimmt die Spaltenposition in dem Datenabschnitt der Datei.

```
@attribute <attribute-name> <datatype>
```

Der <attribute-name> darf nur mit einem Buchstaben beginnen. Wenn Leerzeichen im Namen enthalten sind, muss der Name in Hochkommas angegeben werden. Der <datatype> kann einer der von WEKA unterstützten vier Datentypen sein:

- *numeric*
- *integer* is treated as *numeric*
- *real* is treated as *numeric*
- <nominal-specification>
- *string*
- *date* <date-format>
- *relational* for multi-instance data

Die @data Deklaration

Die ARFF-Datensektion der Datei beinhaltet die Datendeklarationszeile, sowie die Zeilen der aktuellen Instanz. Die `@data`-Deklaration ist eine einzelne Zeile, die den Start des Datensegmentes der Datei angibt.

```
@data
```

Jede Instanz wird in einer einzigen Zeile dargestellt. Mit Zeilenumbrüchen wird das Ende der Instanz angegeben. Die Attributwerte für jede Instanz werden durch Kommas getrennt. Die Reihenfolge der Attributwerte muss die gleiche sein, wie die der Attribut-Deklaration im Kopfbereich der Datei.

```
@data
'Exponat 1',102.2,-66,2,0,0,...
```

4.4.2 Classifier

Jeder Lernalgorithmus wird in WEKA von der abstrakten Klasse `weka.classifiers.AbstractClassifier` abgeleitet. Diese wiederum implementiert die Klasse `weka.classifiers.Classifier`. Für einen einfachen Klassifikator werden folgende Methoden benötigt:

- eine Methode, die ein Klassifikator-Modell aus einer Trainingsdaten-Datei erzeugt (`buildClassifier()`)
- eine Methode, die eine Wahrscheinlichkeitsverteilung für alle Klassen erzeugt (`distributionForInstance()`)

Beim Anlegen eines Klassifikators können je nach Klassifikationstyp verschiedene Optionen übergeben werden, von denen die meisten die Auswertung beeinflussen. Im folgenden Quellcode Beispiel 4.12 ist die Deklaration für eine `NaiveBayes`- und eine `BayesNet` Klassifikation aufgezeigt.

Listing 4.12: Anlegen eines Klassifikators

```
// Set an NaiveBayes Classifier
private NaiveBayes m_ClassifierNB = new NaiveBayes();

// Set an BayesNet Classifier with options
private BayesNet m_ClassifierBN = new BayesNet();
String[] options = {"-D", "-Q", "weka.classifiers.bayes.net.search.local.K2", "--", "-P", "1"
    , "-E", "weka.classifiers.bayes.net.estimate.SimpleEstimator", "--", "-A", "0.5"};
m_ClassifierBN.setOptions(options);
```

Um einen Klassifikator zu trainieren, wird ihm die Trainingsdaten-Datei beim Aufruf der `buildClassifier`-Methode übergeben (siehe Quellcode-Beispiel 4.13). Die Methode erstellt aus den Trainingsdaten dann das Klassifikationsmodell.

Listing 4.13: Klassifikator-Modell erstellen

```
import weka.classifiers.bayes.NaiveBayes;
import weka.core.Instances;
...
Instances data = ... // Trainingsdata set from somewhere
m_ClassifierNB.buildClassifier(data);
```

Ein Featurevektor kann über den Methodenaufruf `classifyInstance` mit dem Modell verglichen und so klassifiziert werden. Der folgende Quellcode 4.14 zeigt wie eine Instanz (Featurevektor) über den Klassifikator mit dem Modell verglichen wird. Die `classifyInstance(Instance)`-Methode gibt für numerische Klassen den Regressionswert und bei nominalen Klassen den 0-basierten Index der Liste der verfügbaren Klassennamen zurück. Mit Hilfe der Methode `distributionForInstance` können die Wahrscheinlichkeiten der einzelnen Klassen zurückgegeben werden.

Listing 4.14: Klassifizieren einer Instanz

```
// Make features into test instance.
Instance instance = ... // from somewhere

double predicted = m_ClassifierNB.classifyInstance(instance);
double[] distribution = m_ClassifierNB.distributionForInstance(instance);

System.out.println(m_Data.classAttribute().value((int)predicted));
System.out.println(Utils.arrayToString(distribution));
```

4.4.3 Datenverwaltung

WEKA stellt zur Datenverarbeitung mehrere Klassen zur Verfügung. In dem Package `weka.core.converters` befinden sich unter anderem Klassen zum Laden und

Speichern von Datensätzen. Hier gibt es die Möglichkeiten, WEKA über die Dateiendung den entsprechenden Ladevorgang wählen zu lassen oder die explizite Lade-Methode direkt anzugeben. Die `DataSource`-Klasse (eine innere Klasse der `weka.core.converters.ConverterUtils`-Klasse) kann zum Laden von Dateien benutzt werden, die das geeignete Datenformat besitzen (siehe Quellcode 4.15).

Listing 4.15: Laden von Datensätzen

```
import weka.core.converters.ConverterUtils.DataSource;
import weka.core.Instances;
...
Instances data1 = DataSource.read("/some/where/dataset.arff");
Instances data2 = DataSource.read("/some/where/dataset.csv");
Instances data3 = DataSource.read("/some/where/dataset.xrff");
```

Wenn die Datei eine andere Dateiendung hat oder ein expliziter Ladevorgang gewünscht ist, kann dieser direkt aufgerufen werden. Der Beispiel-Quellcode 4.16 zeigt einen Ladevorgang einer explizit angegeben ARFF-Datei.

Listing 4.16: Explizites Laden einer ARFF-Datei

```
import weka.core.converters.ArffLoader;
import weka.core.Instances;
import java.io.File;
...
ArffLoader arffLoader = new ArffLoader();
arffLoader.setSource(file);
Instances newDataSet = arffLoader.getDataSet();
```

Das Speichern von `weka.core.Instances`-Objekten ist so einfach wie das Laden. Auch hier stehen wieder zwei Möglichkeiten zur Verfügung. Mit der `DataSink`-Klasse (innere Klasse der `weka.core.converters.ConverterUtils`-Klasse), ist der Datentyp der Datei erst mal nicht vordefiniert (siehe Beispiel 4.17). Er muss nur einem geeigneten Datenformat entsprechen und WEKA übernimmt die Entscheidung, welcher Converter zum Speichern verwendet werden soll.

Listing 4.17: Speichern von Datensätzen

```
import weka.core.Instances;
import weka.core.converters.ConverterUtils.DataSink;
...
// data structure to save
Instances data = ...
// save as ARFF
DataSink.write("/some/where/data.arff", data);
// save as CSV
DataSink.write("/some/where/data.csv", data);
```

Das Beispiel 4.18 zeigt den Speichervorgang über eine explizite Methode. Hier wird der Datensatz als ARFF-Datei gespeichert.

Listing 4.18: Explizites Speichern einer ARFF-Datei

```
import weka.core.Instances;
import weka.core.converters.ArffSaver;
import java.io.File;
...
// data structure to save
Instances data = ...
// save as ARFF
ArffSaver saver = new ArffSaver();
saver.setInstances(data);
saver.setFile(new File("/some/where/data.arff"));
saver.writeBatch();
```

Generierung eines Datensatzes

Neben dem Laden und Speichern von Datensätzen, können diese auch "on-the-fly" im Speicher erzeugt werden. Das Erstellen eines Datensatzes (d.h. eines `weka.core.Instances`-Objektes) ist in zwei Stufen unterteilt:

1. definieren des Formates der Daten, durch Einrichten der Attribute
2. hinzufügen der aktuellen Daten, Zeile für Zeile

Für alle unterschiedlichen Typen von Attributen hat WEKA die selbe Klasse `weka.core.Attribute`, jeweils mit unterschiedlichen Konstruktoren. Wenn die benötigten Attribute angelegt sind, werden diese in einem Array-List-Objekt (`java.util.ArrayList<Attribute>`) zusammengeführt und an ein Datensatz-Objekt der Klasse `weka.core.Instances` übergeben. Der folgende Quellcode zeigt das Definieren von zwei `numeric`-Attributen und einem `nominal`-Attribut mit zwei Werten "yes" und "no".

Listing 4.19: Erstellen eines Datensatzes im Speicher

```
Attribute num1 = new Attribute("num1");
Attribute num2 = new Attribute("num2");
ArrayList<String> labels = new ArrayList<String>();
labels.addElement("no");
labels.addElement("yes");
Attribute cls = new Attribute("class", labels);
ArrayList<Attribute> attributes = new ArrayList<Attribute>();
attributes.add(num1);
attributes.add(num2);
attributes.add(cls);
Instances dataset = new Instances("Test-dataset", attributes, 0);
```

Nachdem die Struktur des Datensatzes festgelegt wurde, können Zeile für Zeile Daten hinzugefügt werden. Die Attributwerte werden in einem Array vom Typ "double" gehalten und an die abstrakte Instanz-Klasse `weka.core.DenseInstance` übergeben. Beim Initialisieren der Instanz wird sogleich ihre Gewichtung definiert. Zuletzt wird die erzeugte Instanz aus dem "double"-Array dem Datensatz hinzugefügt (siehe Quellcode-Beispiel 4.20).[5]

[5] http://www.cs.waikato.ac.nz/ml/weka/documentation.html

Listing 4.20: Hinzufügen von Daten in einen Datensatzes

```
values = new double[dataset.numAttributes()];
values[0] = 2.34;
values[1] = 4.28;
values[2] = data.attribute(2).indexOf("yes");

Instance inst = new DenseInstance(1.0, values);
dataset.add(inst);
```

4.5 Die Bildanalyse-Verfahren

Wie im Kapitel 3 im Absatz 3.4 bereits dargelegt wurde, sollen im TBOC-Verfahren für die Klassifikation neben den Werten, die mittels der Handysensorik ermittelt werden, auch Werte einbezogen werden, deren Bestimmung anhand verschiedener Bildverarbeitungsalgorithmen erfolgt. Diese Werte sollen auf die gleiche Weise wie auch die Sensorwerte verarbeitet werden, was bedeutet, dass sie in Form eines numerischen Wertes in einem Featurevektor abgespeichert werden. Weiter wurde in den Anforderungen im Konzept festgelegt, dass für die Bildanalyse die Bilder nicht dauerhaft gespeichert werden sollen, um die Datenspeicherlast des Anwenders gering zu halten und auch der Vergleich mit großen Bilddatenbanken soll für die Gewährleistung einer Offlineanwendung vermieden werden. Aus diesem Grund werden die Kamerabilder über ihre RGB-Werte analysiert und die Erkenntnisse in die Klassifikation miteinbezogen. Zur Bildauswertung wurden in dieser Arbeit zwei Klassen entwickelt. In der ersten wird ein reduziertes RGB-Histogramm erzeugt und in der zweiten werden regionale RGB-Mittelwerte aus dem Bild ermittelt. Beide Klassen werden in ihrer Funktionsweise und Umsetzung in den folgenden Abschnitten erläutert.

4.5.1 Minimal-Histogramme

Ein RGB-Bild besteht aus drei Farbkomponenten: jeweils ein Farbkanal beinhaltet je eine der drei Grundfarben rot, grün und blau. Die Farbtiefe, auch Datentiefe genannt, gibt an, wie viele unterschiedliche Farben für ein Bild zur Verfügung stehen. Zu beachten ist, dass dabei nicht die Menge der Farben genannt wird, sondern die Anzahl der für die Farbinformationen verwendeten Bits. Bei einem RGB-Bild bedeutet eine Farbtiefe von 24 Bit, dass pro Farbkanal 8 Bit (3 x 8 = 24) für die Farbinformationen verwendet werden. Das heißt, dass pro Farbkanal bei 8 Bit Farbtiefe, $2^8 = 256$ Farbabstufungen zur Verfügung stehen. Daraus ergeben sich bei drei Farbkanälen insgesamt $256^3 = 16,7$ Millionen Farben, die jedes Pixel annehmen kann. Diese Farbabstufungen werden auch Helligkeitswert oder Tonwert genannt. Bei einem Graustufenbild bedeutet das, dass jedes Pixel einen Wert zwischen 0 und 255 annehmen kann. Je höher der Tonwert, desto heller das Pixel. Das bedeutet ein Wert von 0 steht für Schwarz ein Wert von 255 für Weiß. Entsprechend des Graustufenbildes lässt sich ein RGB-Bild in drei Halbtonbilder in den Grundfarben aufteilen. Die RGB-Kanäle können als "Farbebenen" betrachtet werden, die als Halbtonbilder die Werte der jeweiligen Farbe beinhalten. Die Mischung dieser drei Farbkanäle ergibt die Farben der einzelnen Pixel. Hat ein Pixel beispielsweise die RGB-Werte (243, 176, 60), was einem orangenen Farbton

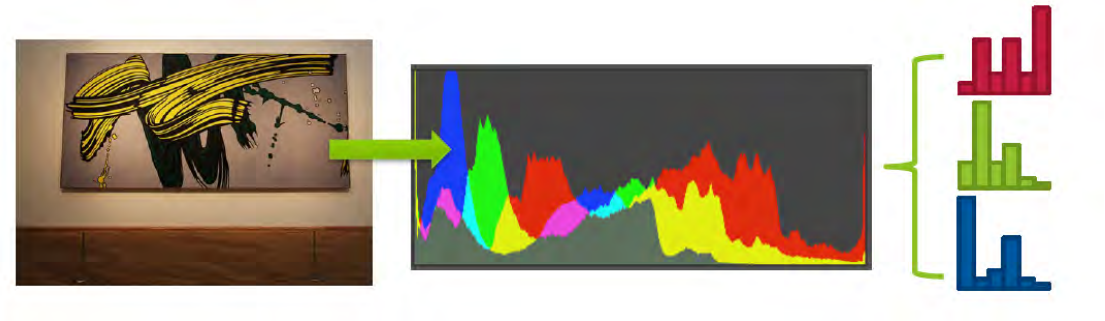


Abbildung 4.4: Exemplarische Darstellung der Erstellung reduzierter RGB-Histogramme. Aus dem aufgenommen Bild werden die Farbinformationen ausgelesen und jeweils ein reduziertes Histogramm für die einzelnen Kanäle R-G-B berechnet. Durch die Reduktion der Balken minimiert sich auch der Rechenaufwand für die Klassifikation und die geringe Anzahl der verwendeten Werte lässt ausreichend Spielraum für eventuelle Farbabweichungen in den Kamerabildern zu, die zum Beispiel durch wechselnde Lichtverhältnisse entstehen.

entspricht, so ist der Rotkanal-Tonwert 243, der Grünkanal-Tonwert 176 und der Tonwert für den blauen Kanal 60. Am dunkelsten ist somit der blaue Wert während der rote Wert am hellsten ist. Mit Hilfe eines Histogramms lässt sich die Verteilung der Farben und der Graustufen eines Bildes visualisieren. Dafür wird für jedes Pixel einzeln der Farbwert bestimmt und in einem Koordinatensystem mit zwei Achsen abgebildet: Auf der x-Achse befinden sich die Tonwerte von schwarz (0, ganz links) bis weiß (255, ganz rechts) und auf der y-Achse ist abzulesen, wie häufig die jeweiligen Tonwerte im Bild vorhanden sind. Bei einem RGB-Bild erhält man so drei Histogramme, die als Bildstatistik gelesen werden können, welche Aufschluss über die im jeweiligen Bild gegebene Farbverteilung liefert. [Erm12]

Die `tboc.vision.MinimalHistogramm`-Klasse ermöglicht über die `getSimpleHist()`-Methode das Ermitteln der Farbverteilung eines ihr übergebenen Bildes (siehe Quellcode 4.21). Hierbei wird für alle Pixel der jeweilige Tonwert für jeden RGB-Kanal ermittelt und in einem Integer-Array, welches das Histogramm widerspiegelt, der entsprechende Zähler für diesen Tonwert um eins erhöht. Um für die Klassifikation nicht 256 Werte übergeben zu müssen, kann beim Methodenaufruf die Histogrammbalken-Dimension angegeben werden. Die Histogrammbalken repräsentieren die im Histogramm verwendeten Unterräume, auch Bins oder Container genannt. Die Dimension beschreibt die Größe dieser Unterräume und somit die Breite der Balken. Wenn die Balken breit sind, fallen mehrere Tonwerte in einen Unterraum und die gesamte Anzahl der Balken wird so reduziert. So kann über die Dimension das Histogramm auf einen beliebig kleinen Wert reduziert werden (siehe Abbildung 4.4).

Listing 4.21: Aufruf der getSimpleHist Methode

```
int[] mImageHist = MinimalHistogramm.getSimpleHist(bitmap, barDim);
```

Die Reduktion der Histogrammwerte, auf einen minimalen Wert von bis zu lediglich drei Werten pro Farbkanal, hat mehrere Vorteile. Zum einen reduziert sich mit nur wenigen Parametern der Rechenaufwand während der Klassifikation und ermöglicht eine hohe Bildwiederholungsrate der Kamera, zum anderen lässt die geringe Anzahl von Werten ausreichend Farbabweichungen in den Kamerabildern zu, wie sie typischerweise unter realen Bedingungen beispielsweise durch sich ändernde Lichtverhältnisse vorkommen. Der nachfolgende Quellcode 4.22 zeigt, wie bei der Bestimmung der Balkenanzahl durch die Balkendimension sichergestellt wird, dass die Dimension die Grenzen des Histogramms nicht überschreitet.

Listing 4.22: Sicherstellung einer Korrekten Balkenbreite im Histogramm

```
int tempC = Math.round(255.0f/(float)barDim);
barDim = (int) (255.0f / (float) tempC);
int histBarCount = (int)(255/barDim);
```

Die einzelnen Histogrammwerte werden zusammen in ein Array vom Typ Integer gespeichert. Hierfür wird die Größe des Arrays auf die dreifache Balkenanzahl gesetzt. Über eine verschachtelte For-Schleife wird über die Pixel des Bildes iteriert, wobei die Zahl der Schleifendurchläufe durch die Breite und Höhe des Bildes bestimmt wird. Über die `getPixel()`-Methode der `android.graphics.Bitmap`-Klasse wird die Farbe des Pixels ausgelesen. Diese kann über die `android.graphics.Color`-Klasse in die einzelnen Farbräume zerlegt werden und anschließend im Histogramm zugeordnet werden (Vergleich Quellcode 4.23).

Listing 4.23: Erstellen eines Farbhistogramms mit dynamischer Balkenanzahl

```
import android.graphics.Bitmap;
import android.graphics.Color;
...
int[] hist = new int[((histBarCount)*3)];

for (int i =0;i<width;++i){
   for(int j=0;j<height;++j){
      int pixel = bitmap.getPixel(i,j);
      r = Color.red(pixel);
      g = Color.green(pixel);
      b = Color.blue(pixel);

      hist[(int)(r/(barDim+1))]++;
      hist[(int)(g/(barDim+1))+histBarCount]++;
      hist[(int)(b/(barDim+1))+((histBarCount)*2)]++;
   }
}
```

4.5.2 Regionale RGB-Mittelwerte

Neben der zuvor beschriebenen grundsätzlichen Farbverteilung mittels des reduzierten Histogramms, welche nur einen allgemeinen Eindruck über die Farbgebung des gesamtes Bildes widerspiegelt, soll ein weiteres Verfahren die Farbwerte in einzelnen lokalen Bereichen im Bild ermitteln und vergleichbar machen. Dies soll auf Basis des Algorithmus von

Abbildung 4.5: Ergebnisse von Bildvergleichen über die "NaiveSimilarityFinder"-Methode von Rafael Santos. In der linken Spalte ist das Originalbild, welches mit mehreren Bildern verglichen wurde. In den folgenden Spalten sind die besten sieben Ergebnisse, in absteigender Reihenfolge sortiert, abgebildet und in den beiden letzten Spalten werden die zwei schlechtesten Übereinstimmungen gezeigt. In der letzten Zeile ist deutlich zu erkennen, dass, wenn ein Bild von der Farbgebung her zwar ähnlich ist, jedoch die Farbverteilung an sich in unterschiedlichen Bereichen des Bildes liegt, der Abstand der ermittelten Farbwerte natürlich dennoch sehr gering zu denen des Originalbildes ist (http://www.lac.inpe.br/JIPCookbook/6050-howto-compareimages.jsp Stand: 20. Februar 2015).

Rafael Santos erfolgen, der in seinem *Java Image Processing Cookbook* [6] ein Verfahren beschreibt, welches auf dem Vergleich zweier Bilder beruht. Hierbei werden die Bilder in Regionen unterteilt und für deren Bereiche jeweils ein individueller Farbmittelwert ermittelt. Mit diesem Verfahren wird über die einzelnen Mittelwerte der Regionen von zwei Bildern eine Distanz bestimmt, die als Aussage über die Ähnlichkeit der beiden Bilder betrachtet werden kann. Wenn zwei identische Bilder miteinander verglichen werden, sollte die Distanz null ergeben. Im Umkehrschluss bedeutet das, dass je größer die ausgelesene Distanz zwischen den Bildern ist, desto unterschiedlicher sind sie (siehe Abbildung 4.5).

Das beschriebene Verfahren wird für die hier vorliegende Arbeit adaptiert und eine Migration auf das Androidsystem durchgeführt. Dabei werden die wesentlichen Aspekte des Verfahrens übernommen. Die `tboc.vision.RegionalAverage`-Klasse bekommt zunächst beim Aufruf der Methode `calcSignatureRGB()` ein Bild übergeben (siehe Quellcode 4.24).

Listing 4.24: Aufruf der calcSignatureRGB Methode

```
int[]  mImageSignature = RegionalAverage.calcSignatureRGB(bitmap);
```

In dieser Methode wird zunächst eine 5x5-Matrix, die über das Bild gelegt wird, definiert, welche das Bild in 25 einzelne Regionen einteilt. In jedem der 25 Felder wird über die einzel-

[6] http://goo.gl/X8im6q Stand: 20. Februar 2015

Abbildung 4.6: Darstellung der regionalen RGB-Durchschnittsberechnung in 25 Feldern. Im Bild zu erkennen, die definierte 5 x 5 Matrix zur Einteilung der Regionen. In jedem Feld werden über die Pixel die Farbwerte ermittelt und der Mittelwert bestimmt. Die 25 Mittelwerte beschreiben hierbei die Bildsignatur.

nen Pixel ein Farbwert bestimmt und ein RGB-Farbmittelwert berechnet (siehe Abbildung 4.6). Die Methode `averageAround()` ermittelt die verschiedenen Farbwerte der einzelnen Pixel über den Bereich der jeweiligen Region und speichert diese für jeden Farbkanal ab. Über die Pixelanzahl in dem Bereich wird aus den Farbkanal-Werten der Farbmittelwert für die Region ermittelt, anschließend zu einem gesamten Farbwert zusammengesetzt und der übergeordneten Methode wieder übergeben. Diese speichert die Farbmittelwerte sortiert nach Farbkanälen in ein Array ab. Die so resultierenden 75 Farbinformationen beschreiben die Signatur des Bildes und können in der Klassifikation verwendet werden. Der Aufbau der Methode wird im folgenden Beispiel 4.25 aufgezeigt.

Listing 4.25: Erstellen der Farbsignatur über Regionale Mittelwerte

```
//Signaturwert  5 * 5 * 3 fuer RGB
int[] sig = new int[25*3];

// For each of the 25 signature values average the pixels around it.
// Note that the coordinate of the central pixel is in proportions.
float[] prop = new float[] {1f / 10f, 3f / 10f, 5f / 10f, 7f / 10f, 9f / 10f};
int count = 0;
for (int x = 0; x < 5; x++){
   for (int y = 0; y < 5; y++){
      int averageColor = averageAround(i, (double)prop[x], (double)prop[y]);
      sig[count] = Color.red(averageColor);
      count++;
      sig[count] = Color.green(averageColor);
      count++;
      sig[count] = Color.blue(averageColor);
      count++;
   }
}
```

4.6 Aufbau der Android Anwendung

Der Programmcode der TBOC-Anwendung umfasst 116 Methoden in 12 Klassen, die in drei "Packages" unterteilt sind. Das `tboc.vision`-Package beinhaltet die Bildverarbeitungsklassen `MinimalHistogramm` und `RegionalAverage`, die im vorherigen Abschnitt beschrieben wurden. Alle Activity-Klassen und somit die Klassen, die mit der Benutzer-Interaktivität und gleichzeitig mit der GUI-Ausgabe in Verbindung stehen, befinden sich im `tboc.gui`-Package. Die Klassen, die Methoden und Funktionalitäten für die Activity-Klassen bereitstellen, befinden sich im `tboc.util`-Package. Die Activity-Klassen importieren jeweils die benötigten Klassen aus den beiden anderen Packages (siehe Abbildung 4.7). Das Zusammenspiel und die Abhängigkeiten der einzelnen Klassen sind im Schaubild B in Form eines Klassendiagramms abgebildet. Darin befinden sich auch schon die verwendeten Klassen der WEKA Bibliothek.

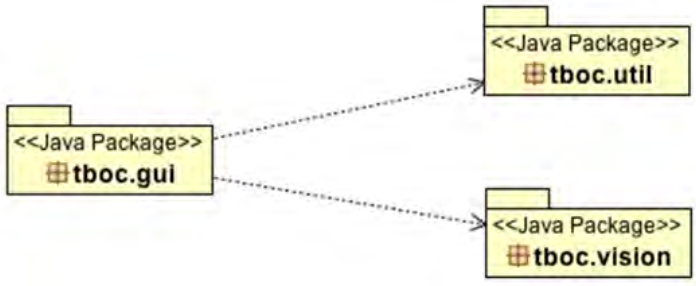


Abbildung 4.7: Die TBOC-Anwendung ist in drei Packages unterteilt. In den GUI-Packages werden alle Activity-Klassen gehalten. Die Klassen importieren jeweils Klassen aus den anderen Packages. Alle Klassen, die Methoden für die GUI Klassen bereitstellen, befinden sich im Utility-Package sowie die beiden Bildanalyse-Klassen im Vision-Package.

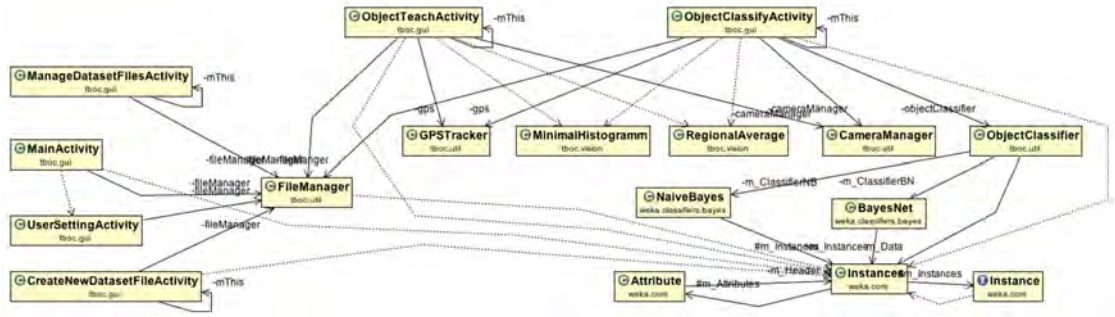


Abbildung 4.8: Übersicht über die Klassenstruktur der TBOC-Anwendung. Das Klassendiagramm zeigt alle im System entwickelten Klassen und deren Verbindungen. Um die komplette Programmstruktur abzubilden, sind auch die abhängigen Klassen der WEKA-Bibliothek mit dargestellt (siehe auch Anhang B für eine größere Darstellung).

Der Grundstein der Anwendung wird durch die `tboc.gui.MainActivity`-Klasse gebildet. Sie ist die erste, die beim Start der Anwendung aufgerufen wird und bildet über ihr Layout den Startbildschirm ab (siehe Abbildung 4.9). Hier wird mit Hilfe einer `ListView` das Menü angezeigt, welches zu den Teilen der Anwendung führt, die für die beiden Phasen des Verfahrens verantwortlich sind. Der Aufruf der zugehörigen Actvitys wird über den Intent-Befehl, der im Abschnitt 4.3 im Quellcode-Beispiel 4.4 vorgestellt wurde, realisiert. In der `ActionBar` der Activity wird ebenfalls ein Menü bereitgestellt, über das die Systemeinstellungen der App geladen werden. Für den schnellen Überblick werden die momentanen Einstellungen der Anwendung auf dem Startbildschirm unter dem Hauptmenü in einem `TextLabel` ausgegeben. Hierfür werden die Einstellungen über den `android.preference.PreferenceManager` als `SharedPreferences` abgespeichert. So können sie in jeder Activity ausgelesen werden (siehe Quellcode-Beispiel 4.26).

Listing 4.26: Laden der SharedPreferences

```
SharedPreferences prefs = PreferenceManager
        .getDefaultSharedPreferences(this);
```

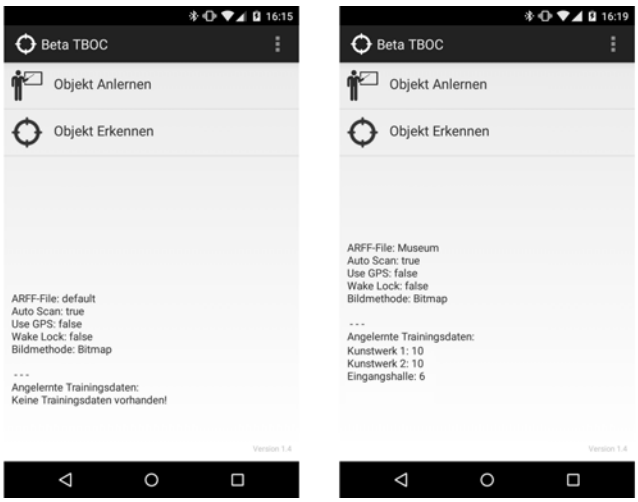


Abbildung 4.9: Die GUI der MainActivity ist der Grundstein der TBOC-Anwendung. Über eine "ListView" werden die Menüeinträge angezeigt, die für die beiden Phasen des Verfahrens stehen. Weiter gibt es ein Menü-Icon oben in der "ActionBar"-Komponente das die Anwendungseinstellungen aufruft. Im Unteren Teil des Displays werden Systemeinstellungen ausgelesen, so wie Informationen zu den geladenen Trainingsdaten.

In einem weiteren `TextView` wird, wenn Daten vorhanden sind, die Anzahl der jeweiligen Trainingsdaten für ein Objekt angezeigt (vergleiche Abbildung 4.9). Hierfür wird die momentan geladenen Trainingsdaten-Datei eingelesen und, wie im Beispiel-Quellcode 4.16 beschrieben, als Typ der Klasse `weka.core.Instances` abgelegt. Über die `attributeStats(int index)`-Methode kann die Anzahl der einzelnen Trainingsdaten für ein Objekt ermittelt werden. Damit in der Anwendung sichergestellt wird, dass

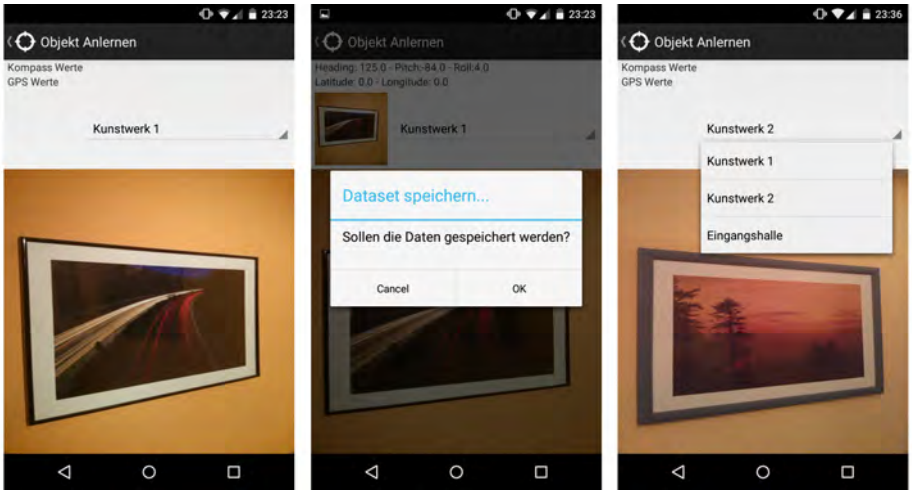


Abbildung 4.10: Die ObjectTeachActivity hat ein schlankes, einfaches Layout. Über eine "TextView" werden die gesammelten Sensordatenwerte ausgegeben. Unterhalb der Textanzeige befindet sich eine "ImageView" über die das Bild, welches verarbeitet wurde, angezeigt wird. Das anzulernende Objekt kann über eine "drop down"-Liste ausgewählt werden und das aktuelle Kamerabild wird in der entsprechenden "View" darunter angezeigt. Durch ein "Touch"-Event wird das Sammeln der Daten und die Bildverarbeitung gestartet. Das Speichern und Ablegen in den Trainingsdaten als Featurevektor kann über einen Speicher-Dialog geschehen.

immer eine Trainingsdatei vorhanden ist, wird in der MainActivity mit der Methode `createDefaultFile()` eine "default"-Datei erzeugt und leer (ohne Datensätze) im System als ARFF-Datei abgelegt. Da beim erstmaligen Start der Anwendung noch keine Trainingsdaten vorhanden sind, ist die Objekt-Erkennung noch nicht möglich und der erste Schritt sollte daher das Anlernen von Objekten sein. Funktionen wie das neue Anlegen von Trainingsdaten mit zugehörigen Objekten und das Verwalten wird später im Abschnitt 4.7 näher beschrieben.

4.6.1 Die Anlern-Phase

Über den Menüpunkt "Objekt Anlernen" wird die `tboc.gui.ObjectTeachActivity` aufgerufen. Die Hauptaufgabe der Klasse ist die Realisierung der ersten Phase des Verfahrens. Sie ist somit eine reine Administrator-Klasse und nur im Verwaltungstool derTBOC-Anwendung vorgesehen. Wie in Abbildung 4.10 zu sehen ist, zeigt die grafische Benutzeroberfläche in einer `TextView` die vom System erfassten Sensordatenwerte zur Kontrolle an. Unterhalb dieser `TextView` befindet sich eine kleine `ImageView`, die das aktuelle Bild, bei der Bildverarbeitung anzeigt. Über einen sogenannten `Spinner` (ein Android Layout-Objekt in anderen IT-Bereiche unter "drop down"-Liste bekannt) kann eines der Objekte ausgewählt werden, zu dem Daten angelernt werden sollen. Die `TextureView` im unteren Bereich des Displays

gibt die aktuelle Live-Vorschau der Kamera wider. Durch Drücken (Touch-Event) auf dieses View-Element wird der Anlernprozess gestartet. Wenn dieser einen Durchlauf beendet hat, wird über einen Benutzerdialog abgefragt, ob die aktuell gesammelten Werte gespeichert oder verworfen werden sollen. Dieser Vorgang kann je nach Belieben für jedes Kunstwerk wiederholt werden.

Programmatische Umsetzung

Das Füllen der `Spinner`-Komponente mit den Namen der Objekte, die angelernt werden sollen, wird über die Methode `addListenerOnSpinnerItemSelection()` realisiert. Wenn die geladene Trainingsdaten-Datei noch keine Objekte beinhaltet, wie es zum Beispiel der Fall bei der Default-Datei bei erstmaligem Öffnen ist, wird die Spinner-Liste über festdefinierte Werte gefüllt, die in der `String.xml` hinterlegt werden können. Wenn in der ARFF-Datei Objekte hinterlegt sind, werden die Namen der Objekte nach dem Einladen der Trainingsdaten ausgelesen und über einen `ArrayAdapter` an den Spinner übergeben. Damit die Live-Vorschau der Kamera auf der `TextureView` angezeigt werden kann, wird der View-Komponente mit der Methode `setSurfaceTextureListener(this)` ein Listener registriert, der das Kamerabild an die View übergibt. Durch ein Drücken auf die `TextureView` wird ein Button-Event ausgelöst, welches von der Methode `calculateAverageBtn()` repräsentiert wird. Diese Methode startet einen `TimerTask`, der als Thread das Sammeln der Sensordaten ausführt (siehe nachfolgenden Quellcode 4.27).

Listing 4.27: TimerTask als Thread zum sammeln der Sensordaten

```
import java.util.Timer;
import java.util.TimerTask;

...
//Declare the timer
Timer t = new Timer();

//TimerTask wird gesetzt, der als Thread die Aufgabe hat, die Sensordaten zu sammeln.
t.scheduleAtFixedRate(new TimerTask() {
    @Override
    public void run() {
    ...
    }
}
```

In der `run`-Methode werden Methoden zum Sammeln der Daten aufgerufen. Je nachdem, ob in den Einstellungen der Anwendung die GPS-Nutzung aktiviert ist, werden die GPS-Koordinaten über die `setGPS()`-Methode und die Orientierungswerte über die `setOrientation(SensorEvent orientationEvent)`-Methode ausgelesen und in Listenobjekte vom Typ `java.util.List<Double>` abgelegt. Über die Konstante `MAX_STACK_SIZE` wird die Größe der Liste definiert und somit auch die Anzahl der Wiederholungen der `run`-Methode bestimmt. Wenn der Zählwert der Konstante erreicht ist, wird der Timer über die Methode `cancel()` gestoppt und alle laufenden Aufgaben angehalten. Aus den in den Listen gesammelten Werten werden die jeweiligen Mittelwerte errechnet, um Ausreißer in den Werten zu eliminieren. Die aktuellen Sensordaten werden in

der zugehörigen TextView im oberen Bereich des Displays ausgegeben.
Bei der Verarbeitung der Kamerabilder ist der Vorgang ähnlich wie bei der Erfassung der Sensordaten. Für das Abgreifen des Kamerabildes stehen mehre mögliche Vorgehensweisen zur Verfügung. Um aus dem Timer-Thread ein Bild von der Kamera zuzulassen, werden Callback-Methoden verwendet. Die naheliegendste Methode scheint hier die takePicture()-CallbackMethode der Kamera zu sein. Hier wird ein Foto mit der Kamera erstellt, welches in dem PictureCallback abgefangen und verarbeitet werden kann. Der Beispiel-Quellcode 4.28 zeigt, wie aus dem byte[]-Array mit den Bilddaten über die android.graphics.BitmapFactory-Klasse eine Bitmap erzeugt werden kann. Da für die weitere Verarbeitung ein Bild mit fest definierter quadratischen Größe benötigt wird, wird die erzeugte Bitmap mit der Methode createScaledBitmap() und der Konstanten IMAGE_BASE_SIZE auf eine einheitliche Größe reduziert. Weil das Auslösen eines Fotos unter Android aber viel Zeit benötigt, ist diese Variante nicht die effektivste.

Listing 4.28: takePicture() Callback-Methode

```
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.hardware.Camera;
import android.hardware.Camera.PictureCallback;

...
PictureCallback jpegCallback = new PictureCallback() {
   public void onPictureTaken(byte[] data, Camera camera) {

      // Convertierung zu einem Bitmap
     Bitmap mBitmap Bitmap.createScaledBitmap(BitmapFactory.decodeByteArray(data, 0, data.
         length),IMAGE_BASE_SIZE,IMAGE_BASE_SIZE, false );
      ...
      }
   };
```

Eine weitere Möglichkeit ist das Auslösen der setOneShotPreviewCallback()-Methode bei der Kamera. Hier wird kein Foto ausgelöst, sondern ein Frame vom aktuellen Vorschaubild abgegriffen. Der unten stehende Quellcode 4.30 zeigt, dass die gewonnenen Bilddaten jedoch erst im "YUV-Farbmodell" vorliegen und dass sie, bevor sie zu einer Bitmap umgewandelt werden können, als android.graphics.YuvImage eingelesen werden müssen. Die Klasse des YuvImage stellt über die Methode compressToJpeg() eine Möglichkeit zur Verfügung, eine Umwandlung in ein JPEG-Format zu vollziehen. Das nun neugewonnene byte[] kann wie bei dem vorher vorgestellten Verfahren in eine Bitmap gewandelt werden. Auch hier wird die Größe des Bildes über den Wert der Konstante auf die einheitliche Größe reduziert. Diese Möglichkeit ist im Bezug auf die erste zwar schneller in der Verarbeitung, jedoch zeigte sich bei den ersten Tests mit unterschiedlichen Geräten, dass sie zu Helligkeitsproblemen des Kamerabildes führt, wenn diese Methode auf den neuen *Google Nexus 5*-Geräten ausgeführt wird. Hier wird, sobald das Vorschaubild ausgelesen wird, die automatische Belichtung der Kamera ausgeschaltet, und somit ist das Kamerabild von da an unterbelichtet.

Listing 4.29: setOneShotPreviewCallback()

```
import java.io.ByteArrayOutputStream;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.hardware.Camera;
import android.graphics.YuvImage;
...
private Camera.PreviewCallback mPrevCallback = new Camera.PreviewCallback(){
   public void onPreviewFrame( byte[] data, Camera Cam ) {
      // Convert to JPG
      Size previewSize = mCamera.getParameters().getPreviewSize();
      YuvImage yuvimage=new YuvImage(data, ImageFormat.NV21, previewSize.width, previewSize.
          height, null);
      ByteArrayOutputStream baos = new ByteArrayOutputStream();
      yuvimage.compressToJpeg(new Rect(0, 0, previewSize.width, previewSize.height), 80, baos
          );
      byte[] jdata = baos.toByteArray();

      // Convertierung zu einem Bitmap
      Bitmap mBitmap Bitmap.createScaledBitmap(BitmapFactory.decodeByteArray(jdata, 0, jdata.
          length),IMAGE_BASE_SIZE,IMAGE_BASE_SIZE, false );
      }
   };
```

Die letzte Möglichkeit, die in dieser Arbeit vorgestellt wird, ist vielleicht auch die einfachste. Durch die Verwendung der `TextureView` und nicht, wie es auch möglich wäre, einer `SurfaceView` für das Anzeigen des Kamera Vorschaubildes in der Activity, besteht die Möglichkeit, das aktuelle Bild, was auf der `TextureView` angezeigt wird, über die Methode `getBitmap()` übergeben zu bekommen. Beim Aufruf dieser Methode, kann gleich die gewünschte Bildgröße mit angegeben werden.

Listing 4.30: Bitmap aus der TexturView auslesne

```
import android.graphics.Bitmap;
...
Callback callback = new Callback() {
   @Override
   public boolean handleMessage(Message msg) {

       Bitmap mBitmap = mTextureView.getBitmap(IMAGE_BASE_SIZE,IMAGE_BASE_SIZE));
   }
};
```

In der TBOC-Anwendung wurden für Testzwecke alle drei Möglichkeiten realisiert. In den Systemeinstellungen der Anwendung kann die jeweilige gewünschte Methode ausgewählt werden. Die erstellte Bitmap wird dann in der Methode `catchingImageData()` an die beiden in dieser Arbeit entwickelten Bildanalyse-Klassen `tboc.vision.MinimalHistogramm` und `Tbc.vision.RegionalAverage` übergeben. Die von diesen beiden Klassen zurückgegebenen Integer-Arrays mit Farbwerten (vergleiche Abschnitt 4.5.1 und 4.5.2) werden in einem Listen-Objekt vom Typ `java.util.List<int[]>` gespeichert. Zur besseren Kontrolle für den Anwender wird das aktuell verarbeitete Bild auch noch in der `ImageView` unterhalb der Sensordaten angezeigt. Nach Erreichen der festgelegten Durchlaufanzahl werden die Listen wie auch die der Sensordatenwerte gemittelt.

Über einen Dialog-Callback kann der Anwender entscheiden, ob er die Daten in die Trainingsdaten übernehmen oder sie verwerfen will. Wenn die Daten gespeichert werden

sollen, wird die Methode `generateDataSet()` aufgerufen. Hier wird, wie im Abschnitt 4.4.3 beschrieben, ein Datensatz aus den gesammelten Daten generiert und der Methode `saveDataSet()` übergeben, die das Speichern der Daten in die zugehörige Trainingsdaten-Datei vornimmt. Die Besonderheit an dieser Methode ist, dass zusätzlich zu dem im Abschnitt 4.7.2 beschriebenen Vorgehen beim Speichern von ARFF-Dateien noch eine Überprüfung stattfindet, ob die Datei schon Trainingsdaten beinhaltet. Wenn die Datei noch leer ist, wird einfach der gerade neu generierte Datensatz in die Datei geschrieben. Sollten aber schon Daten vorhanden sein, wird mit der Methode `appendDataSet()` der neue Datensatz an den Datensatz in der Datei angehängt (siehe Quellcode 4.31). Hierzu wird die Datei als Datensatz eingelesen und der neue Datensatz wird Zeile für Zeile dem alten Datensatz aus der Datei angefügt. Der zusammengeführte Datensatz wird zurückgegeben und in die ARFF-Datei geschrieben. Der Quellcode 4.32 zeigt das Zusammenführen der beiden Datensätze in der `appendDataSet()`-Methode.

Listing 4.31: Das Speichern des Datensatzes

```
import java.io.File;
import java.io.IOException;
import weka.core.Instances;
import weka.core.converters.ArffSaver;
...
private void saveDataSet(Instances dataSet) throws IOException{
   File theFile = new File(filePath);
   ArffSaver saver = new ArffSaver();
   if(fileManager.checkFileExists(theFile)){
      if(theFile.length()>0){
         dataSet = appendDataSet(theFile, dataSet);
       }

      saver.setInstances(dataSet);
      saver.setFile(theFile);
      saver.writeBatch();
    }
}
```

Listing 4.32: Zusammensetzten zweier Datensatzes

```
import java.io.File;
import java.io.IOException;
import weka.core.Instances;
import weka.core.converters.ArffLoader;
...
private Instances appendDataSet(File file, Instances dataSet)throws IOException{
   ArffLoader arffLoader = new ArffLoader();
   arffLoader.setFile(file);
   Instances newDataSet = arffLoader.getDataSet();

   for(int i = 0; i< newDataSet.numInstances(); i++){
      dataSet.add(newDataSet.get(i));
   }
   return dataSet;
}
```

4.6.2 Die Erkennungs-Phase

Die `tboc.gui.ObjectClassifyActivity`-Klasse repräsentiert die Erkennungsphase des Verfahrens. Diese Activity-Klasse kann so auch direkt in andere Android-Projekte integriert

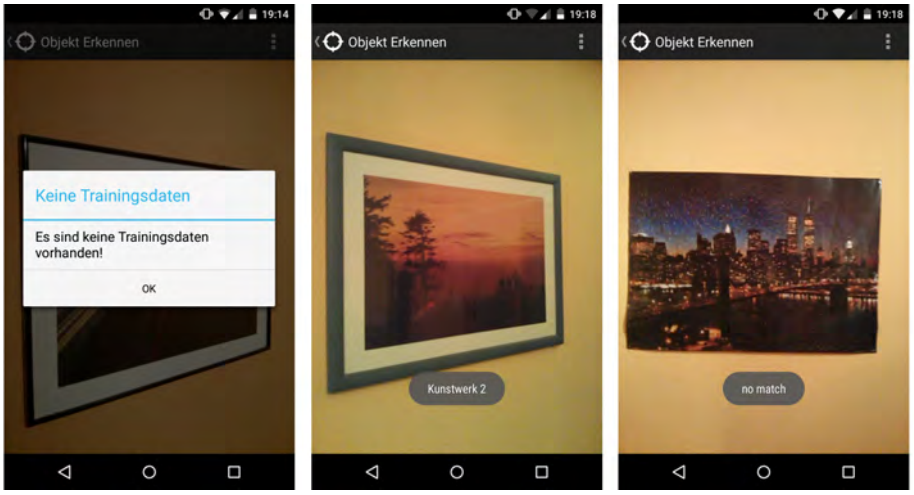


Abbildung 4.11: Die ObjectClassifyActivity hat außer einer "TextureView", auf der das Vorschaubild der Kamera angezeigt wird, keine weiteren GUI-Elemente. Über das Menü in der "ActionBar" kann das statistische Datenmodell neu geladen werden. Die Klassifikation kann nur durchgeführt werden, wenn auch Trainingsdaten vorhanden sind. Wenn dies nicht der Fall ist, wird ein Benutzerhinweis ausgegeben und die Activity beendet. Ansonsten wird durch Drücken auf die View die Klassifikation gestartet. Wenn ein Objekt erkannt wird, wird der Name des Objektes ausgegeben. Bei Nichterkennen kommt ein Benutzerhinweis.

werden, um die Funktionalität der Objekterkennung zu benutzen. Abbildung 4.11 zeigt die Oberfläche der Activity. Sie besteht nur aus eine `TextureView`, die sich über den ganzen Bildschirm zieht. Auf dieser View wird, wie auch bei der Anlern-Activity, das Vorschaubild der Kamera angezeigt. Jedoch kann eine Klassifikation nur dann erfolgen, wenn auch Trainingsdaten in der geladenen ARFF-Datei vorhanden sind. Wenn keine Daten in der Datei hinterlegt sind, wird über einen Benutzerdialog eine Meldung ausgegeben, die Activity beendet und die Anwendung kehrt zur Hauptseite der Anwendung zurück. Wenn Trainingsdaten vorhanden sind, kann durch Drücken auf die View die Klassifikation gestartet werden. Wenn das System ein Objekt erkennt, wird der Name des Objektes mittels eines Benutzerhinweises ausgegeben. Wenn kein Objekt erkannt wird, wird eine "no match" Meldung ausgegeben und der Klassifikationsvorgang so lange wiederholt, bis ein Objekt erkannt wurde, oder der Benutzer die Activity beendet und zur Hauptseite zurückkehrt.

Programmatische Umsetzung

Wenn die Activity geladen wird, wird direkt in der `onCreate()`-Methode überprüft, ob für die geladene Trainingsdaten-Datei schon ein Bayes-Klassifikator-Modell im Speicher angelegt wurde oder ob in den Systemeinstellungen das automatische Neuladen des Modells aktiviert ist (sie Quellcode 4.33). Die Methode `loadModel()` lädt die im System momentan ausgewählte ARFF-Datei und überprüft, ob diese Trainingsdaten beinhaltet. Über einen `DialogHandler` wird ein Benutzerhinweis ausgegeben, wenn keine Daten vor-

handen sind und die Activity navigiert die Anwendung zurück zum Hauptfenster der `MainActivity`. Wenn die geladene ARFF-Datei Datensätze beinhaltet, wird über die Methode `createModel()` ein Objekt vom Typ `tbbc.util.ObjectClassifier` erzeugt. Über dieses Classifier-Objekt kann anhand des Featurevektor, wie im Abschnitt 4.4.2 beschrieben, später die Klassifikation erfolgen. Der folgende Quellcode zeigt, wie über das `ObjectClassifier`-Objekt das Modell im Speicher des Gerätes abgespeichert wird.

Listing 4.33: Das Erzeugen eines Modells über die ObjectClassifier-Klasse

```
import java.io.ObjectOutputStream;
import tboc.util.ObjectClassifier;
...
public void createModel(String modelPath, String arffPath) throws Exception{
    ObjectClassifier = objectClassifier = new ObjectClassifier(classifierName);
    objectClassifier.createModel(arffPath);

    // Save objectClassifier object
    ObjectOutputStream modelOutObjectFile = new ObjectOutputStream(new FileOutputStream(
        modelPath));
    modelOutObjectFile.writeObject(objectClassifier);
    modelOutObjectFile.close();
}
```

Wenn einmal ein Modell im Speicher für die jeweilige ARFF-Datei abgelegt ist, wird das Modell beim Öffnen der ObjectClassifyActivity nicht geladen. Wenn die Trainingsdaten sich jedoch seit dem Erstellen des Modells geändert haben, kann über das Menü in der `ActionBar` der Actvity das Modell von Hand neu geladen werden. Um gerade für Testzwecke das ständige Neuladen per Hand zu vermeiden, kann in den Systemeinstellungen ein automatisches Laden des Modells festgelegt werden.

Der Klassifikations-Prozess wird durch Drücken des Anwenders auf die `TextureView` oder automatisch beim Aufrufen der Activity gestartet. Hierfür muss die "Auto Scan" Funktion in den Einstellungen aktiviert sein. Das Sammeln von Sensordaten und die Analyse des Kamerabildes über die beiden Verfahren werden wie auch bei der `ObjectTeachActivity` durch einen Timer und einen separaten Thread realisiert (vergleiche Absatz 4.6.1). Da jedoch beim automatischen Starten der Erkennungsphase die "run"-Methode schneller startet als der `SensorEventListener` initialisiert wird, ist ein Sicherheitsmechanismus in den Thread eingebaut, der beim ersten Durchlauf für eine bestimmte Zeit "schlafen" gelegt wird (siehe nachfolgendes Quellcode-Beispiel 4.34).

Listing 4.34: Pausieren eines Threads

```
int tempTime = 2000;
...
try {
    // Thread wartet beim erstmaligen Durchlaufen der Schleife im Thread für "tempTime" bis
        der "SensorEventListener" initialisiert ist.
    Thread.sleep(tempTime);
} catch (InterruptedException e) {
    // Da, der "try"-Block nur für den Aufruf der "sleep"-Methode benötigt wird, ist hier
        keine weitere Aktion von Nöten.
    e.printStackTrace();
}
```

Die gesammelten Daten werden nicht in Listen-Objekten gespeichert und am Ende der Durchläufe gemittelt, wie es beim Anlernen der Fall war, sondern nach jedem Durchlauf werden die aktuellen Daten genommen und mit dem Datenmodell verglichen und ein Objekt klassiert. Die Klassierten Objekte werden in einem Listenobjekt gespeichert und nach Beendigung der Thread-Durchläufe, wird das am häufigsten erkannte Objekt mit der `getMaxRecognizedPosition()`-Methode ermittelt. Damit die gesammelten Daten mit dem Modell verglichen werden können, wird mit Hilfe der `generateFeatureVector()`-Methode ein Featurevektor aus den Daten erstellt. Hierfür werden alle Werte in der entsprechenden Reihenfolge in einen kommagetrennten String geschrieben. Dieser String und der Pfad zum Datenmodell wird an die Methode `classifieLocation()` übergeben. Der Quellcode 4.35 zeigt, wie das Modell aus dem Speicher geladen wird und als Objekt vom Typ `tboc.util.ObjectClassifier` angelegt wird.

Listing 4.35: Objekt Klassifikation

```
import java.io.FileInputStream;
import java.io.ObjectInputStream;
import tboc.util.ObjectClassifier;
...
public String classifieLocation(String modelFilePath, String featureVector) throws Exception{
    ObjectInputStream modelInObjectFile =  new ObjectInputStream(new FileInputStream(
        modelFilePath));
    ObjectClassifier objectClassifier = (ObjectClassifier) modelInObjectFile.readObject();
    modelInObjectFile.close();

    //Determination of the object
    String location = objectClassifier.classifyObject(featureVector);
    return location;
}
```

Die Klasse von diesem Objekt stellt die Methode `classifyObject()` zur Verfügung, der der Featurevektor übergeben wird. In dieser Methode wird das Objekt mit dem Modell verglichen und klassiert, wie es im Abschnitt4.4.2 im Quellcode-Beispiel 4.14 schon aufgezeigt wurde. Wenn am Ende der Durchläufe das am häufigsten klassierte Objekt in der `getMaxRecognizedPosition()` ermittelt wird, wird dazu über ein `android.widget.Toast` ein View-Element, was wie eine kleine Notiz am unteren mittleren Bildrand des Display angezeigt wird, das erkannte Objekt ausgegeben. Wie im folgenden Quellcode zu sehen ist, wird hier überprüft, ob das erkannte Objekt den Namen `NULL` hat.

Listing 4.36: Ausgabe des Benutzerhinweises für Erkanntes Objekt

```
Context context = getApplicationContext();
int duration = Toast.LENGTH_SHORT;
CharSequence text;
if(!popular.equals("NULL")){
    text = popular;
}else{
    text = getString(R.string.nomatch);
}

Toast toast = Toast.makeText(context, text, duration);
toast.show();

if(popular.equals("NULL") && autoScan){
    startSearchForLocation();
}
```

Dieses sogenannte NULL-Objekt oder auch Dummy-Objekt dient der Lösung des Problemes, welches im Absatz 3.5.2 in Kapitel 3 angesprochen wurde. Da der Naive-Bayes-Klassifikator immer nur die größte Wahrscheinlichkeit unter den angelernten Objekten berechnet, , wird auch immer ein angelerntes Objekt als wahrscheinlicher als die anderen bestimmt und somit zurückgegeben, auch wenn ein nicht angelerntes Objekt fokussiert wurde. Hier kommt das NULL-Objekt zum Einsatz, das in diesem Fall die Fehlanzeige kenntlich machen soll. Das wird aber nur erreicht, wenn das NULL-Objekt eine ausreichend große Wahrscheinlichkeit aufweisen kann. Daher müssen für das NULL-Objekt besondere Trainingsdaten hinterlegt werden, die sich deutlich von den Daten der angelernten Objekte abheben müssen, aber auch das Umfeld noch ausreichend wiedergeben. Wenn das NULL-Objekt erkannt wird, wird ein Hinweis ausgegeben, dass kein Objekt gefunden wurde. Wenn in diesem Fall das automatische Suchen über die "Auto-Scan"-Einstellung in den Einstellungen aktiv ist, wird der Suchvorgang nach einem passenden Objekt erneut gestartet.

4.7 Weitere Systemfunktionen

Neben der Hauptactivity und den beiden Activity-Klassen, die die Phasen des Verfahrens repräsentieren, gibt es noch die Activitys, die für die Verwaltung des Systems verantwortlich sind. Die Systemeinstellungen wurden in den vorhergehenden Abschnitten immer wieder erwähnt, jedoch nicht genauer vorgestellt. Die Aufgaben der Anwendungseinstellungen unterteilen sich in zwei Bereiche:

- die allgemeinen Systemeinstellungen
- die Verwaltung der Trainingsdaten

Die allgemeinen Systemeinstellungen stellen Möglichkeiten zur Modifikation der Anwendung zur Verfügung, die das Arbeiten und Testen erleichtern. In dem Datenverwaltungsbereich können Trainingsdaten ausgewählt, neu angelegt oder gelöscht werden.

4.7.1 System-Einstellungen

Die `tboc.gui.UserSettingActivity` ist eine Activity-Klasse, die von `android.preference.PreferenceActivity` abgeleitet wird. Das Layout und die Struktur der Einstellungen wird über die Methode `addPreferencesFromResource()` gesetzt, in dem eine `PreferenceScreen`-XML übergeben wird. Diese XML-Datei beschreibt über einen besonderen Satz von Steuerelementen die Benutzeroberfläche. Diese XML wird anders als die normalen Layout XML-Dateien im Ressourcenornder `res/xml` abgelegt. Wie der folgende Beispiel-Quellcode 4.37 zeigt, wird jedes Einstellungslayout in einer Hierarchie beschrieben und beginnt mit dem `PreferenceScreen`-Element. Einzelne Einstellungsblöcke können über `PreferenceCategory` gruppiert werden. Neben einer einfachen `Preference`, die mit einem Titel und einer Beschreibung versehen werden können. Über einen `Intent` können hier weitere Activitys angegeben werden, die geladen werden sollen. Weitere

Einstellungselemente sind die `ListPreference`, die eine Auswahlliste zur Verfügung stellt oder die `CheckBoxPreference` für Einstellungen über ein Kontrollkästchen. [MS14] [Gar11] Die einzelnen Listeneinträge für die `ListPreference` werden in der `string.xml`-datei als `string-array` definiert.

Listing 4.37: Deklaration der PreferenceScreen-XML

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android" >
    <PreferenceCategory android:title="System Einstellungen" >
        <Preference
            android:title="Trainingsdaten verwalten"
            android:summary="Verwalte deine Trainingsdatendaten">
             <intent
                 android:action="android.intent.action.VIEW"
                 android:targetPackage="de.thm.tboc"
                 android:targetClass="tboc.gui.ManageDatasetFilesActivity"
             />
        </Preference>
        <ListPreference
            android:key="imagepref"
            android:title="Bildmethode"
            android:entries="@array/imageMethod"
            android:entryValues="@array/imageMethodValue"
            android:defaultValue="Bitmap"/>
        <CheckBoxPreference
            android:defaultValue="true"
            android:key="prefAutoScan"
            android:summary="Automatische Scannen beim Öffnen"
            android:title="Auto Scan" >
        </CheckBoxPreference>
    </PreferenceCategory>
</PreferenceScreen>
```

Wie in Abbildung 4.12 zu sehen ist, werden in den Einstellungen hauptsächlich Funktionen über Checkboxen verwaltet. Hierzu zählt die Funktion *Auto Scan*, die dafür sorgt, dass beim Starten der Erkennungs-Activity, die Objekterkennung selbstständig ohne Benutzereinwirkung startet. Wenn dieser Haken nicht gesetzt ist, muss der Anwender die Erkennung per Hand, über eine Berührung des Displays, starten. *Auto Load* bezieht sich auf das automatische Laden des Modells. Im Normalfall muss das Modell nur einmal generiert werden. Wenn sich die Trainingsdaten ändern, zum Beispiel neue hinzugekommen sind, muss das Modell nachgeladen werden. Dies kann in der Erkennungsphase über das Menü der `ActionBar` gemacht werden. Da das aber leicht in der Entwicklungsphase, also beim Anlernen und gleichzeitigem Kontrollieren mit der Erkennung, vergessen werden könnte, kann das über den *Auto Load* automatisiert werden. Da in vielen Bereichen das Hinzunehmen von GPS-Daten nur bedingt hilfreich ist, kann in den Einstellungen entschieden werden, ob die GPS-Daten in der Anlern- und auch in der Erkennungsphase miteinbezogen werden sollen. Mit der Einstellung *Wake Lock* kann das Ausschalten des Displays verhindert werden, wenn die Anwendung offen ist. Weiter gibt es für das eben vorgestellte NULL-Objekt die Möglichkeit beim Neuerzeugen "DUMMY"-Trainingsdaten zu verwenden. Dies sind vordefinierte Werte, die keiner speziellen Zuordnung unterliegen. Beim Anlernen dieser allgemeinen Daten wurden möglichst farbneutrale Flächen berücksichtigt und um die räumliche Verteilung zu erfassen alle Richtungen und Neigungen mit eingeschlossen.

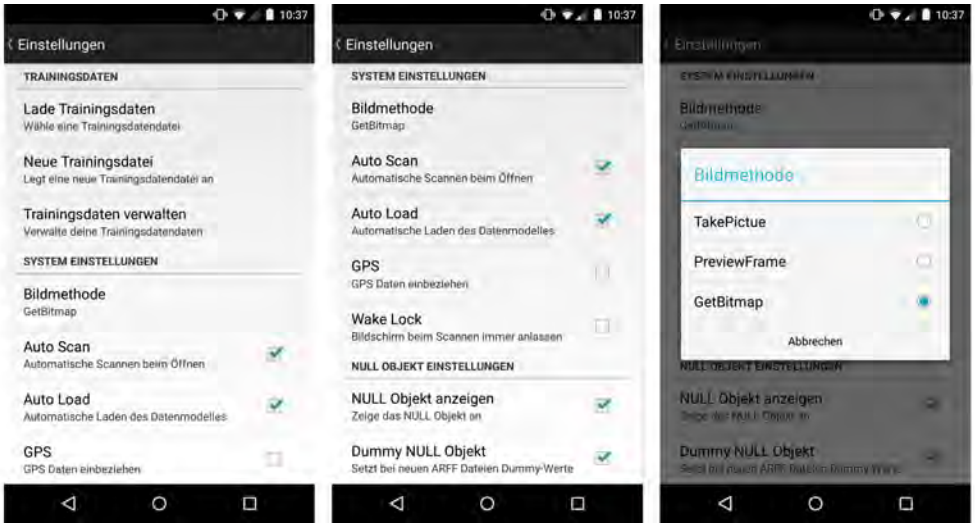


Abbildung 4.12: Die Systemeinstellungen der Anwendung werden über die `UserSettingActivity` angezeigt. Das Layout und die Einstellungselemente werden über eine `PreferenceScreen`-XML definiert. Neben Einstellungselementen, die nur aus einem Button bestehen und weiter Activitys laden können, stehen auch Checkboxen oder Auswahllisten zur Verfügung. In der TBOC-Anwendung werden in den Einstellungen Trainingsdaten verwaltet und Systemfunktionen angepasst.

Mit diesem "Test NULL-Objekt" können in der Anfangsphase schon gute Ergebnisse erzielt werden. Erst wenn den anderen angelernten Objekte eine größere Anzahl von Datensätzen zugeordnet wurden und sie damit eine höhere Wahrscheinlichkeit erreichen, kommt es bezüglich des Null-Objekt zu Fehlerkennungen. Zu diesem Zeitpunkt müssen die Träningsdaten des NULL-Objektes erweitert werden und damit den gegebenen räumlichen Randbedingungen angepasst werden. Die Einstellung "NULL-Objekt anzeigen" ermöglicht, das Objekt beim Anlernen und auch in der Hauptansicht auszublenden. Was nicht bedeutet, dass die Daten bei der Klassierung nicht beachtet werden. Um eine der drei unterschiedlichen Möglichkeiten zu nutzen, Bilder aus der Kamera auszulesen, kann über den Punkt *Bildmethode* ein Verfahren ausgewählt werden. Hierbei wird die Liste der möglichen Methoden in einer Auswahlliste angezeigt. Die Besonderheit hier ist, dass die Beschreibung des Einstellungspunktes sich bei der Änderung der Auswahl an die Änderung anpasst. Hierfür wird in der `tboc.gui.UserSettingActivity` in der `onSharedPreferenceChanged()`-Methode eine Änderung des Einstellungseintrages abgefangen (siehe nachfolgendes Quellcode-Beispiel 4.38).

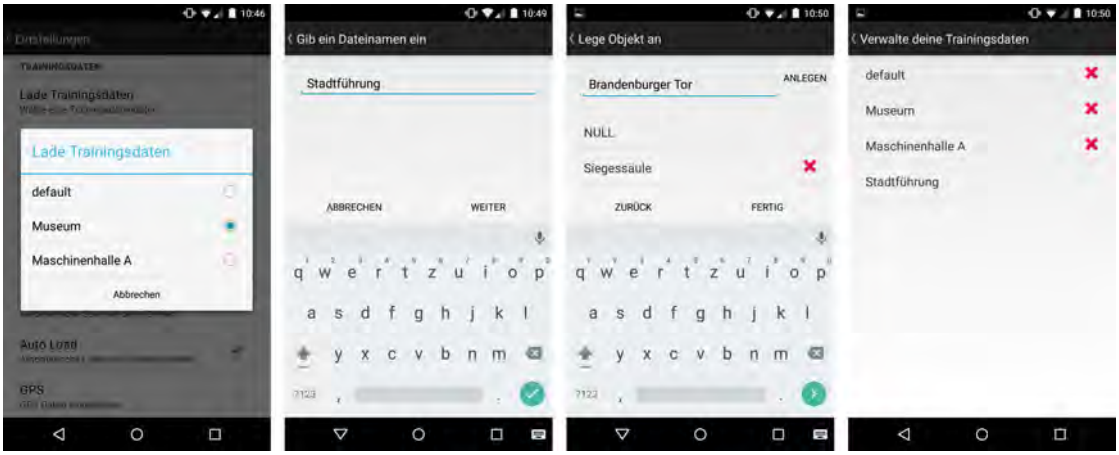


Abbildung 4.13: Im Bereich der Trainigsdatenverwaltung in den Systemeinstellungen kann aus einer Liste der vorhandenen Trainingsdaten eine ausgewählt und im System geladen werden. Weiter gibt es die Möglichkeit, neue Dateien anzulegen. Hierfür wird zuerst der Name der Datensätze eingegeben, danach besteht die Möglichkeit Objekte anzulegen, die später angelernt und erkannt werden sollen. Weiter ist es möglich angelegte Datensätze wieder vom Gerätespeicher zu löschen.

Listing 4.38: Änderung der Einstellungsbeschreibung bei Änderung einer Auswahl

```
import android.content.SharedPreferences;
import android.content.SharedPreferences.OnSharedPreferenceChangeListener;
import android.preference.ListPreference;
import android.preference.Preference;
...
public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {
    Preference pref = findPreference(key);
    if(key.equals("imagepref")){
        if (pref instanceof ListPreference) {
            ListPreference listPref = (ListPreference) pref;
            pref.setSummary(listPref.getEntry());
        }
    }
}
```

4.7.2 Trainingsdatenverwaltung

Die Trainingsdatenverwaltung besteht aus der Auswahl und dem damit verbundenen Laden einer Datensatz-Datei ins System, dem neuen Erstellen einer Trainingsdaten-Datei und dem Löschen von Datensätzen aus dem lokalen Speicher des Gerätes (siehe Abbildung 4.13). Hierbei wird das Laden der Datensätze noch in der `tboc.gui.UserSettingActivity` über eine `ListPreference` gelöst. Anders als bei der Auswahl der Bildmethode im letzten Abschnitt, wird diese Liste nicht über ein `String-Array` gefüllt, sondern in der `setSharedPreferences()`-Methode. Hier wird der Speicherort der Trainingsdaten ausgelesen und eine Liste der beinhalteten Dateien erstellt. Über diese Liste wird iteriert und

der Dateiname in ein Array gespeichert. Dieses Array wird anschließend mit der Methode `setEntries()` an die `ListPreference` übergeben (siehe Quellcode 4.39).

Listing 4.39: Füllen der Datensatz-Auswahllsite

```
ListPreference lp = (ListPreference) findPreference("listpref");
File PfadObj = getExternalFilesDir("arff");
File[] files = PfadObj.listFiles();
String[] value = new String[files.length];

for (int i = 0; i < files.length; i++) {
   value[i] = fileManager.parseFileName(files[i].getName());
}

lp.setEntries(value); // set displayed text
lp.setEntryValues(value); // set associated values
```

Für das Anlegen und Löschen von Datensätzen wird jeweils nur ein `Intent` zu weiteren Activitys aus den Einstellungen aufgerufen. Für das Anlegen von Datensätzen und Objekten ist die `tboc.gui.CreateNewDatasetFileActivity` zuständig. Für die bessere Übersichtlichkeit und Struktur beim Anlegen von Dateien wurde hier das Konzept eines `ViewSwitcher` verwendet. Dies ermöglicht es in einer Activity das Layout der Anzeige zu ändern [DC11]. Hier kann dabei zwischen zwei Layouts gewechselt werden und damit der Inhalt von Eingabemasken übersichtlich verteilt und die Verarbeitungsstruktur in der selben Klasse verarbeitet werden. Das Ändern des Layouts kann wie im Quellcode-Beispiel 4.40 über das `ViewSwitcher`-Objekt mit den Methoden `showNext()` und `showPrevious()` ausgelöst werden.

Listing 4.40: Wechseln der View üb der den ViewSwitcher

```
ViewSwitcher switcher;
switcher = (ViewSwitcher) findViewById(R.id.profileSwitcher);
...
switcher.showNext();  // Switches to the next view
```

Beim Anlegen einer neuen Datei wird überprüft, ob ein Name eingeben wurde oder ob eine Datei mit dem Namen schon vorhanden ist. Wenn einer dieser Fälle eintritt, wird über einen Benutzerdialog jeweils eine passende Fehlermeldung ausgegeben. Ansonsten wird die nächste View geladen und es besteht die Möglichkeit, Objekte zur Erkennung anzulegen. Wenn das nicht gewünscht ist, kann dieser Punkt auch einfach übersprungen werden. Dann stehen, wie bei der *default*-Datei, zum Anlernen von Objekten nur die drei Standard-Objektnamen, die im System hinterlegt sind, zur Verfügung. Beim Anlegen von Objekten werden die Objektnamen in einer `ListView` abgelegt. Um Duplikaten vorzubeugen, wird beim Eintrag, der neue Name mit den Einträgen der Liste verglichen und gegebenenfalls eine Fehlermeldung ausgegeben. Die `ListView` besteht aus einer Text- und einer Bildkomponente. Die Bildkomponente wird mit einem Icon gefüllt, welches für die Möglichkeit des Löschens des Listeneintrages steht. Das NULL-Objekt wird automatisch mitangelegt und angezeigt, wenn dies in den Einstellungen so ausgewählt ist. Jedoch wird durch ein transparentes Icon vom System verhindert, dass der Eintrag für das NULL-Objekt aus der Liste entfernt werden kann. Mit einem `setOnItemClickListener()` wird die Auswahl eines Listeneintrages abgefangen und dieser aus der Liste entfernt, wenn er nicht das NULL-Objekt ist. Durch

Bestätigen des *FERTIG*-Buttons wird anhand der Listeneinträge und der `createARFF()`-Methode ein leere Datensatz-Instanz erstellt, in der die Listeneinträge dem nominalen Attribut zugeordnet werden und der somit die zu erkennenden Objekte repräsentiert. Der zurückgegebene Datensatz vom Typ `weka.core.instances` wird anschließend mit Hilfe der `weka.core.converters.ArffSaver`-Klasse abgespeichert. Damit die neu angelegte Datei auch gleich im System geladen ist, wird sie noch in den `SharedPreferences` als *fileName* eingetragen. Hierfür wird über den `android.content.SharedPreferences.Editor` eine Änderung in dem entsprechenden Eintrag vollzogen (siehe Quellcode 4.41).

Listing 4.41: Ändern der SharedPreferences über den SharedPreferences.Editor

```
SharedPreferences preferences = PreferenceManager.getDefaultSharedPreferences(this);
Editor editor = preferences.edit();
editor.putString("listpref", fileName);
editor.commit();
```

Das Löschen von Trainingsdaten in der `tboc.gui.ManageDatasetFilesActivity` verhält sich im Prinzip wie das Löschen von einzelnen Objekten in der Liste beim Anlegen der Trainingsdaten-Datei. Zunächst wird der Ordner der Trainingsdaten ausgelesen und für jede Datei im Ordner ein Listeneintrag mit dem Dateinamen in einer `ListView` angelegt. Wie im vorangegangenen Abschnitt besteht auch hier die Liste aus einer Text- und Bildkomponente. Auch hier wird der Liste ein *delete*-Icon angehängt. Wenn die im System verwendete Datei geladen wird, wird stattdessen ein transparentes Icon geladen (siehe Abbildung 4.7.2). Durch die Selektion eines Eintrages wird nach einem Benutzerdialog, der fragt, ob die Datei wirklich gelöscht werden soll, diese in diesem Fall mit der Methode `deleteListItemFile()` aus dem Datenspeicher des Gerätes gelöscht und der entsprechende Eintrag in der Liste entfernt.

4.8 Zusammenfassung

Die entwickelte Anwendung zur prototypischen Implementierung realisiert sowohl die Anlernphase als auch die Erkennungsphase des TBOC-Verfahrens. Für die zugrunde liegende Infrastruktur der Applikation muss die Vorgehensweise bei der Entwicklung von Android-Anwendungen unter der Entwicklungsumgebung Eclipse berücksichtigt werden. Da das Verfahren für die vorgeschaltete Initialisierung von Augmented Reality-Anwendungen gedacht ist, muss außerdem das hierfür ausgewählte SDK von Metaio eingebunden werden, um somit die Grundlage für das Einbinden von AR-Szenarien innerhalb der App zu ermöglichen. Das TBOC-Verfahren an sich ist nicht von Metaio abhängig, da es durch die Erkennung der Objekte lediglich die Vorentscheidung trifft welche entsprechende AR-Activity geladen werden soll.
Durch die Einbindung der WEKA-Bibliothek in die TBOC-Anwendung ist es möglich statistische Klassifikationsmethoden zu nutzen. Mit Hilfe von WEKA können sogenannte *Classifier*-Objekte angelegt und mit diesen Daten Objekte klassiert werden. Des Weiteren schaffen die WEKA eigenen Klassen die Möglichkeit die Datenverwaltung der *Trainingsdaten* zu realisieren.

Für das System wurden zwei Bildanalyseverfahren entwickelt.

- Minimal-Histogramme
- regionale RGB-Mittelwerte

Um nicht ganze Bilder abspeichern zu müssen und dadurch unerwünschte große Datenmengen zu erzeugen, werden mit dem Prinzip der *Minimal-Histogramme* aus einem Bild Pixel für Pixel die unterschiedlichen Farbabstufungen für jeden einzelnen Farbkanal ermittelt. Jedoch werden die 256 Werte auf eine beliebige kleinere Anzahl reduziert, um weniger Werte zu Verarbeiten und Farbabweichungen genug Spielraum zu lassen. Die Reduzierung wird über die Breitenangabe eines Histogrammbalken (auch als Container oder Bin bezeichnet) definiert. Dieses Verfahren ermöglicht es eine generelle Farbverteilung eines Bildes zu ermitteln und diese Werte in der Klassifikation zu verwenden.
Um die fehlenden Daten zu erhalten, die beschreiben an welcher Position im Bild die verschiedenen Farbwerte liegen, wurde mit einem zweiten Bildanalyseverfahren die Möglichkeit geschaffen, die Farbmittelwerte in einem Bild regional zu bestimmen. Der freie Java-Code von Rafael Santos aus dem Projekt *Java Image Processing Cookbook* konnte für das Verfahren adaptiert und für die Zwecke von TBOC angepasst werden. Für diese Bildanalyse wird eine 5x5 Matrix über das Bild gelegt und in jeder der 25 Regionen über die jeweiligen Pixel ein Farbmittelwert erfasst. Für das Verfahren wurden die 25 Durchschnittswerte jeweils noch in die einzelnen Farbkanäle zerlegt und zur weiteren Verarbeitung übergeben.
Durch die Nutzung der Bayes-Klassifikation für die Erkennung entsteht das Problem, dass mit dem Bayes-Theorem immer nur die größte Wahrscheinlichkeit für das Zutreffen eines Objektes in Bezug auf alle ihm bekannten Objekte errechnet wird, daraus ergibt sich, dass immer ein Objekt mit der höchsten Wahrscheinlichtkeit ausgegeben wird. Die Einführung eines Null-Objektes, welches nicht zuordnungsfähige Trainingsdaten enthält, kann zur Lösung dieses Problems beitragen. Das System erkennt dann im Falle, dass keines der realen angelernten Objekte das richtige ist, das Null-Objekt mit einer höheren Wahrscheinlichkeit als die anderen angelernten Objekte.

Im folgenden Kapitel wird die prototypische Implementierung des Verfahrens in eine reale Museums-Anwendung beschrieben. Hierbei wird der Schwerpunkt auf die Szenariobeschreibung und die Implementierung der AR-Funktionalität gelegt, welches die Aufzeichnung der AR-Punktwolken, die Animation von 3D-Modellen und die Zusammenführung in der Anwendung umfasst.

Kapitel 5

Prototypische Implementierung in eine reale Anwendung

Nachdem der Aufbau und die Funktionsweise des Verfahrens dargelegt wurde, soll nun die Integration des Verfahrens in ein reales Anwendungsszenario folgen. Die TBOC-Anwendung ist ein Entwicklungstool für das in dieser Arbeit entwickelte Verfahren. Diese Anwendung bietet die Möglichkeit, Objekte für eine Klassifikation anzulernen und rein für Testzwecke auch zu erkennen. Das Ergebnis, welches Objekt erkannt wurde, erfolgt in diesem Fall jedoch nur durch einen Benutzerhinweis in Textform. Um das Verfahren unter realen Bedingungen testen zu können und auch eine sinnvolle Kombination mit AR-Inhalten zu implementieren, wurde das entwickelte TBOC-Verfahren in eine Applikation für das mathematische Mitmach-Museum "Mathematikum" integriert.

5.1 Die Mathematikum App

Im Rahmen des Projektes PIMAR wird für das mathematische Museum Mathematikum in Gießen eine App generiert. Mit dieser Applikation soll der Funktionsumfang sowie die Flexibilität des Generators an einem praktischen Testszenario erprobt werden. Dabei spielt natürlich auch die Umsetzung verschiedener AR-Szenarien innerhalb des Museumsumfeldes sowie die vorgeschaltete Initialisierung der zu erkennenden Objekte oder Räume mittels des TBOC-Verfahrens eine große Rolle. Um den Nutzen von AR im allgemeinen und auch von dem Verfahren an sich im Umfeld einer solchen Anwendung aufzuzeigen, werden für zwei ausgewählte Exponate des Museums exemplarische AR-Szenarien entwickelt und an einem weiteren Exponat der Nutzen des Verfahrens auch ohne die Einbindung von AR untersucht. Die jeweiligen Umsetzungen dienen lediglich als Muster und werden aus diesem Grund nicht bis ins letzte Detail ausgearbeitet. Dennoch sind sie für sich alleine gestellt schon eine gute Erweiterung und Hilfestellung für den Museumsbesuch. Mit dieser Applikation soll der Funktionsumfang sowie die Flexibilität des Generators an einem praktischen Testszenario erprobt werden. Dabei spielt natürlich auch die Umsetzung verschiedener AR-Szenarien innerhalb des Museumsumfeldes sowie die vorgeschaltete Initialisierung der zu erkennenden Objekte oder Räume mittels des TBOC-Verfahrens eine große Rolle. Um den Nutzen des TBOC-

Verfahrens und auch von AR im Umfeld einer solchen Anwendung aufzuzeigen, werden für zwei ausgewählte Exponate des Museums exemplarische AR-Szenarien entwickelt und an einem weiteren Exponat der Nutzen des Verfahrens auch ohne die Einbindung von AR untersucht. Die jeweiligen Umsetzungen sind lediglich als Muster zu verstehen und werden aus diesem Grund nicht bis ins letzte Detail ausgearbeitet. Dennoch sind sie für sich alleine gestellt schon eine gute Erweiterung und Hilfestellung für den Museumsbesuch. Das erste und auch am aufwendigsten umgesetzte Muster für die Erkennung und Erweiterung eines Ausstellungsstücks durch AR wird für das Exponat "Pi mit den Füßen" erstellt. Die anderen beiden ausgewählten Exponate sind "Die Deutschlandtour" und "Knack den Code", wobei das zuletzt genannte Exponat als Beispiel für eine Implementierung ohne AR-Einbindung dient. Die einzelnen Umsetzungen haben jeweils einen unterschiedlichen Umfang. Dabei erstreckt sich die Spannweite von einfachen Einblendungen von erstellten 2D-Labels über die Anfertigung und Integration aufwendiger 3D-Animationen bis hin zu einem Aufruf eines in der App implementierten Spiels. In den nachfolgenden Abschnitten werden die einzelnen Umsetzungen der jeweiligen Programmabschnitte beschrieben.

5.1.1 Pi mit den Füßen

Das Exponat *Pi mit den Füßen* besteht aus einem auf dunklen Bodenfliesen aufgemaltem weißen Kreismuster. Aufgabe des Museumsbesuchers ist es, sich durch die Messung des Kreisdurchmessers und Umfanges an die Zahl Pi anzunähern. Dies soll über das Zählen der Schritte, die beim Ablaufen des aufgemalten Musters benötigt werden, erfolgen. Die Zahl Pi ergibt sich durch Division der Anzahl der Schritte für den Kreisumfang durch die Schrittzahl für den Durchmesser. Je kleiner, enger die Schritte sind und damit um so größer die Schrittanzahl die benötigt wird, desto genauer wird die errechnete Annäherung an Pi. Durch die Verwendung eines AR-Szenarios in Form einer eingeblendeten Animation soll dem Besucher genau diese Tatsche verdeutlicht werden. So kann der Benutzer mit Hilfe der Anwendung einen menschlichen 3D-Charakter den Kreis und die Durchmesserlinie ablaufen lassen. Hierbei zählt die Anwendung die Schritte des virtuellen Charakters und gibt die Anzahl auf dem Display aus. Sobald der Charakter nach dem Kreisumfang mit dem Durchmesser beginnt, wird dem Anwender im Display die Annäherung an die Zahl Pi im Quotienten angezeigt. Wenn der digitale Charakter seinen letzten Schritt getan hat, sollte das Ergebnis sehr nah an der Zahl Pi sein. Alternativ kann der Museumsbesucher auch eine kleine Maus über den Kreis huschen lassen. Anhand der Tatsache, dass sie viel mehr Schritte benötigt, ist das Ergebnis der Maus noch genauer und damit näher am tatsächlichen Wert der Zahl Pi.

Aufbereitung für markerloses AR-Tracking

Um diese AR-Simulation für das Exponat durchzuführen, muss zunächst das Exponat für das markerlose AR-Tracking vermessen und erfasst werden. Hierfür stellt Metaio mit dem *metaio Toolkit*[1] eine mobile Anwendung für iOS und Android zur Verfügung. Das Programm ermöglicht es unter anderem, *3D-Maps* von physikalischen Objekten oder Umgebungen zu

[1] https://play.google.com/store/apps/details?id=com.metaio.creatorMobile Stand: 26. Februar 2015

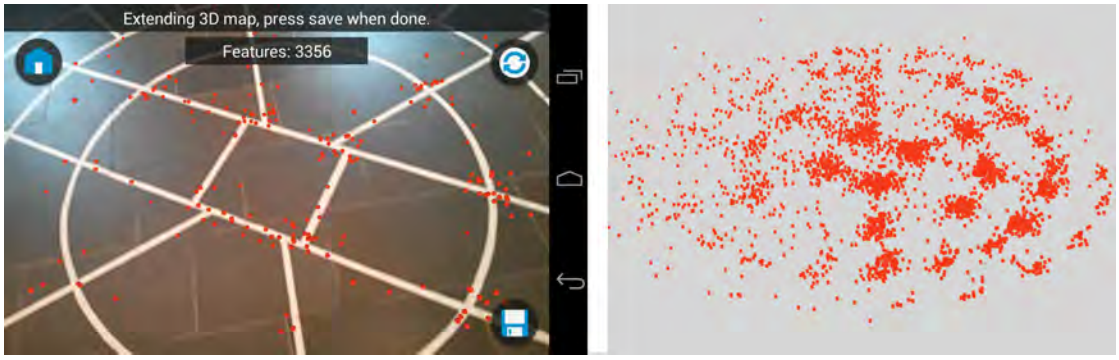


Abbildung 5.1: Für markerloses 3D-Tracking werden bei der Verwendung des Metaio SDKs sogenannte 3D-Maps benötigt, die mit der hauseigenen mobilen Anwendung *metaio Toolkit* erstellt werden können. Hier werden markante Featurepunkte im Bild ermittelt und in einer 3D-Punktwolke zusammengeführt. Jeder einzelne Featurepunkt wird mittels eines roten Punktes symbolisiert.

erstellen. Hierbei werden vom System markante Bildpunkte gesucht und als Featurepunkte in Form roter Punkte in einer 3D-Punktwolke festgehalten (siehe Abbildung 5.1). Beim Erstellen dieser Punktwolken muss sichergestellt werden, dass genügend Featurepunkte gesammelt wurden, so dass eine schnelle und zuverlässige Erkennung des Objektes gewährleistet wird. Dennoch muss auch darauf geachtet werden, nicht unnötigerweise eine zu hohe Anzahl von Feature-Punkten zu speichern, da sonst mit der Anzahl auch die Zeit für die Verarbeitung und Zuordnung dieser ansteigt. Das Toolkit hilft mit der Augmentierung eines Testobjektes dabei schon während des Erstellungsvorganges die Stabilität des Trackings und die Lage des Objektes in der Szene überprüfen zu können. Wenn das Ergebnis dem gewünschten Umfang entspricht, kann die 3D-Punktwolke als `3dmap`-Datei abgespeichert werden. Die ausgegebenen "3D-Maps" können anschließend auf einen Computer exportiert und zusammen mit dem Metaio SDK in einer beliebigen AR-Anwendung verwendet werden. Meistens ist es nützlich, die Map vorher in einer weiteren Software von Metaio zu laden, dem *Creator*[2] von Metaio. Dies ist eine Software, verfügbar für Mac oder PC, die in der Lage ist, die 3D-Punktwolken zu importieren und eventuelle Fehler in den gespeicherten Punkten aus der Datei zu entfernen.

Animation und Konvertierung des 3D-Modells

Da für die Veranschaulichung dieses Exponates das Ablaufen des Pi-Kreises durch einen menschlichen 3D-Charakter und eine 3D-Maus animiert werden sollen, wird zunächst nach passenden 3D-Modellen für die Animation gesucht. Für die Animationen selbst muss eine spezielle Software wie Motion Builder[3], Blender[4] oder Maya[5] verwendet werden. In den einzelnen Anwendungen besteht die Möglichkeit, 3D-Objekte über sogenannte Schlüssel-

[2] http://www.metaio.com/products/creator Stand: 26. Februar 2015
[3] http://www.autodesk.com/products/motionbuilder/overview Stand: 26. Februar 2015
[4] http://www.blender.org/ Stand: 26. Februar 2015
[5] http://www.autodesk.com/products/maya/overview Stand: 26. Februar 2015

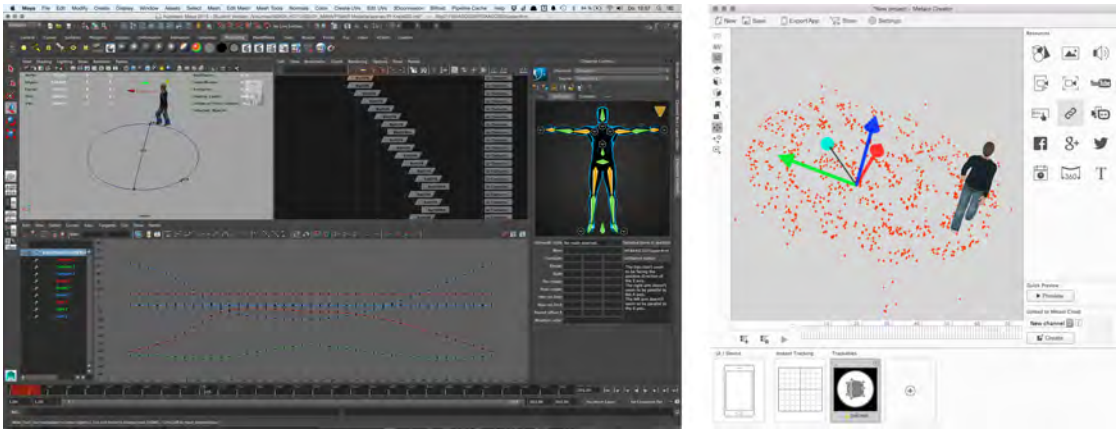


Abbildung 5.2: Die benötigten 3D-Modelle für das AR-Szenario "Pi mit den Füßen" werden in der 3D-Modellierungssoftware Maya von Autodesk animiert. Danach im FBX-Format exportiert und in ein Metaio eigenes Format konvertiert. Anschließend können die Modelle in den Metaio Creator geladen und auf der 3D-Punktwolke ausgerichtet werden.

bilder per Hand zu animieren (Key Frame-Animation) oder eingelesene Daten realer Bewegungsabläufe, sogenannte *Motion-Capturing*-Daten zu verwenden. Die Animation der verwendeten 3D-Objekte für diese Anwendung wird in der 3D-Software Maya von Autodesk realisiert (siehe Abbildung 5.2 links). In der genannten Software wurde ein "Walk Cycle" für den menschlichen Charakter animiert. Um nur einmal die Schrittabfolge eines Laufvorganges animieren zu müssen und diesen später beliebig oft wiederholen zu können erfolgt die Animation des sogenannten "Walk Cycles" des Charakters auf der Stelle, das bedeutet ohne Translation des Charakters im 3D-Raum. So ist es möglich, den Charakter mit seiner zuvor angefertigten Laufanimation über einen "Motion Path" entlang einer vorgezeichneten Kurve im 3D-Raum zu bewegen. Da bei der Maus die Bewegung der Füße und somit auch die Schritte nicht zu erkennen sind, wird hier auf die Laufanimation verzichtet und lediglich der gesamte 3D-Körper ebenfalls entlang eines vorgegebenen Motion Paths transliert. Um die animierten 3D-Modelle in der AR-Anwendung nutzen zu können, müssen sie zunächst in ein Format gebracht werden, mit dem das Metaio SDK arbeiten kann. Hierfür werden zunächst die Modelle als FBX-Datei (Standard-Austauschformat für 3D-Modelle) exportiert. Anschließend lassen sich die Modelle im FBX-Format mit dem *FBXMeshConverter* von Metaio in das *mfbx* Format wandeln. Dieses Format sowie *MD2* und *(*OBJ) sind die 3D-Formate, die von Metaio unterstützt werden. Autodesk Maya bietet zwar ebenfalls die Möglichkeit, 3D-Modelle direkt als das, wie beschrieben, von Metaio ebenfalls unterstütze OBJ-Format zu exportieren, jedoch gehen bei der Verwendung dieses Formates die Animationen auf den Modellen verloren. Aus diesem Grund konnte das Format für diese Umsetzung keine Verwendung finden. Wenn das 3D-Modell eine Textur besitzt, muss dieses mit dem gleichen Namen als PNG in demselben Ordner abgelegt werden, in der auch das Modell abgespeichert ist.

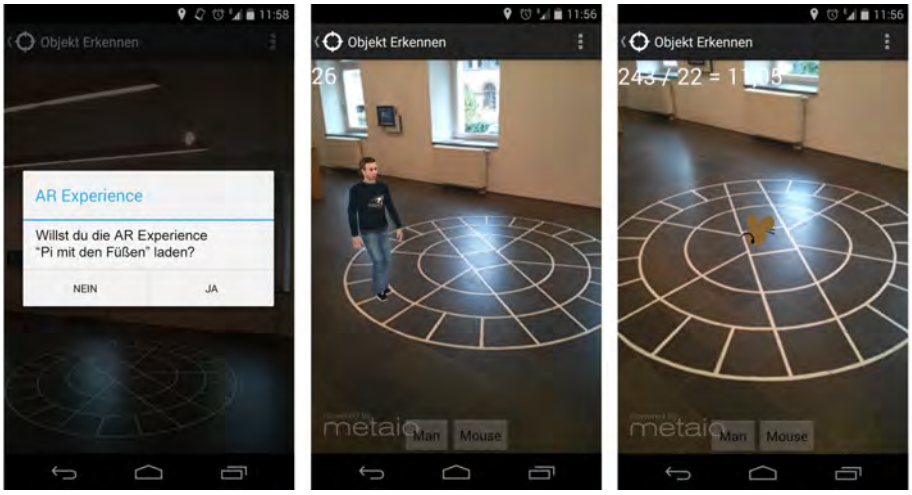


Abbildung 5.3: Nachdem das TBOC-Verfahren das Exponat "Pi mit den Füßen" erkannt hat, wird über einen Benutzerhinweis abgefragt, ob die entsprechende AR-Experience geladen werden soll. In dem AR-Szenario kann über zwei Buttons (mit den Beschriftungen Man und Mouse) die Simulation der Annäherung an die Zahl Pi über das Zählen der Schritte beim Ablaufen des Umfanges und des Durchmessers gestartet werden. Da die Maus mehr Schritte benötigt, wird die Schätzung genauer.

Weil die aufgezeichneten Punktwolken nicht immer perfekt in der 3D-Welt ausgerichtet sind, ist es hilfreich das 3D-Modell in der Creator Software zu überprüfen und Anpassungen an der Position, der Rotation und der Skalierung vorzunehmen. So lässt sich das Objekt auf der Punktwolke platzieren und die Animation kann direkt in der Anwendung getestet und dabei überprüft werden, ob diese auch das gewünschte Ergebnis erzielt (siehe Abbildung 5.2 rechts). Aus dem Creator lassen sich direkt Anwendungen erstellen. Hier werden aber lediglich AREL-Apps herausgeschrieben. Für einen schnellen Test kann dies jedoch nützlich sein. Ein weiteres brauchbares Feature bietet die Möglichkeit, die erstellte AREL-Anwendung direkt über die Junaio-Plattform in einen Webspace und diesen über einen QR-Code dann auf dem Smartphone in der Junaio-Anwendung zu laden. Dieses Feature ist zwar nicht für die eigentliche Anwendung von Nutzen, jedoch erleichtert und beschleunigt diese Funktion den Entwicklungsprozess, da schnell erste Ergebnisse betrachtet werden können. Um die im Creator angepassten Translationswerte der 3D-Modelle in das eigentliche Eclipse-Projekt zu übernehmen, können diese in den Objekteinstellungen ausgelesen werden. Hierbei sollte beachtet werden, dass die Rotationswerte im Creator als Grad angeben sind, und das SDK in Eclipse die umgerechneten Radiant-Werte erwartet.

Integration in die Android Activity

Das Integrieren der Modelle in den Java-Code verhält sich wie im Kapitel 4 in Abschnitt 4.3 beschrieben. In der `loadContents()`-Methode werden die beiden 3D-Modelle über `createGeometry()` als `IGeometry`-Objekte angelegt, die Transformationseigenschaften ein-

gestellt und die Sichtbarkeit der Modelle über die `setVisible`-Methode auf unsichtbar gesetzt. Der Auslöser für die Animation und das Sichtbarwerden der einzelnen 3D-Objekte wird über zwei Buttons geregelt. Durch Betätigung des Buttons für den menschlichen Charakter wird dieser als 3D-Modell eingeblendet und die Maus ausgeblendet (siehe Abbildung 5.3 Mitte). Ebenfalls wird die Animation des Charakters gestartet und falls die Animation der Maus ausgeführt wurde, diese gestoppt (siehe Beispiel-Quellcode 5.1). Über je einen Thread für Umfang und Durchmesser, werden die Schritte der 3D-Modelle hochgezählt und über `TextView`-Elemente im Display ausgegeben. Sobald das Modell den Durchmesser des Kreises abläuft, werden die Schritte dividiert und die Annäherung an Pi im Ergebnis eingeblendet (siehe Abbildung 5.3 rechts).

Listing 5.1: Start der Simulation "Pi mit den Füßen"

```
...
mModel.setVisible(true);
mMouse.setVisible(false);

mModel.startAnimation("Take_001", false);
mMouse.stopAnimation();
...
```

5.1.2 Die Deutschlandtour

Im Anwendungsszenario *Die Deutschlandtour* soll der Museumsbesucher die kürzeste Strecke zwischen elf Städten finden. Der Startpunkt ist Gießen und es sollen mit einer Schnur elf Städte verbunden werden. Dieses "NP-vollständige Problem" ist auch als *Problem des Handlungsreisenden* (engl.: Traveling Salesman Problem) bekannt. Der Begriff "NP-vollständiges Problem" stammt aus der theoretischen Informatik und beschreibt komplexe Entscheidungsfindungen, bei denen davon ausgegangen wird, dass die Problematik nicht mit Hilfe eines effizienten Algorithmus gelöst werden kann.[6] Die am Startpunkt befestigte Schnur reicht bei der Verbindung aller Städte nur dann bis zur letzten Stadt, wenn dieses kombinatorische Problem richtig gelöst wird. Das sogenannte "Problem des Handlungsreisenden" kann den Museumsbesucher schnell frustrieren und enttäuschen. Auf Grund dessen soll mittels einer AR-Unterstützung dem Besucher eine Hilfestellung angeboten werden, die aber nicht gleich zu viel verrät. Beim Erkennen des Exponates unter Verwendung des TBOC-Verfahrens wird der Besucher über einen Dialog gefragt, ob er die AR-Experience für die Deutschlandtour laden möchte. Bei Bejahung wird anschließend die entsprechende 3D-Punktwolke in der AR-Activity geladen und mit der Umgebung verglichen. Wenn beim Tracking eine Übereinstimmung gefunden wurde, wird auf der Deutschlandkarte ein Label eingeblendet mit der Frage ob Hilfe benötigt wird und bei positivem Feedback des Benutzers wird entweder die Lösung für die erste Zielstadt oder das Ende der Tour mit einer Punktmarkierung und einem Pfeil auf der Karte eingeblendet (siehe Abbildung 5.4). Der Pfeil gibt die Richtung an, aus der die Schnur kommen muss, um die Stadt zu erreichen. Mit den jeweiligen Lösungsansätzen sind die weiteren Schritte etwas einfacher und mit etwas Überlegung kommt der Besucher schnell zum richtigen Ergebnis.

[6] http://goo.gl/9Q5f4e Stand: 14.März 2015

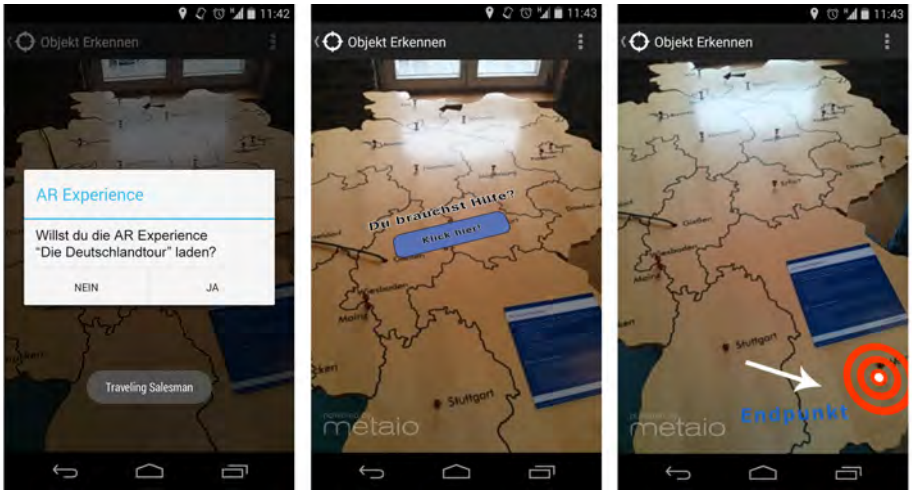


Abbildung 5.4: Bei dem Exponat "Die Deutschlandtour" soll mittels einer Schnur die kürzeste Strecke zwischen elf Städten ermittelt werden. Als Hilfe kann über eine AR-Experience wahlweise das erste oder das letzte Ziel eingeblendet werden.

Umsetzung des Szenarios

Die Vorgehensweise bei der Erstellung des Szenarios verhält sich in den Grundzügen wie die Realisierung des Szenario "Pi mit den Füßen" im Abschnitt 5.1.1. Zunächst wird das Exponat mit dem *metaio Toolkit* für das markerlose Tracking vorbereitet, in dem wie bei dem Pi-Kreis eine 3D-Punktwolke erstellt wird. Da bei diesem Szenario keine 3D-Modelle oder Animationen angepasst werden müssen, wird die 3D-Punktwolke auch nicht im Metaio *Creator* geöffnet, sondern direkt in die Android-Activity eingebunden. Die einzelnen Labels werden über die Methode `createGeometryFromImage()` als `IGeometry` umgewandelt (siehe nachfolgenden Quellcode 5.2).

Listing 5.2: Umwandeln von Bildern zu einem IGeometry Objekt

```
...
IGeometry startPointImage = metaioSDK.createGeometryFromImage(imagePath);
...
```

Die Labels können dann wie die 3D-Objekte über die Objekt-Attribute positioniert und angepasst werden. Die beiden Labels, die für die Hilfe eingeblendet werden sollen, werden bei der Initialisierung zunächst unsichtbar gemacht und erst dann angezeigt, wenn der Benutzer durch Toucheingabe ein positives Feedback auf das "Du benötigst Hilfe?"-Label übergibt. Ob dem Nutzer die Startposition oder aber das Ziel der Reise eingeblendet wird, entscheidet sich über eine Zufallsfunktion.

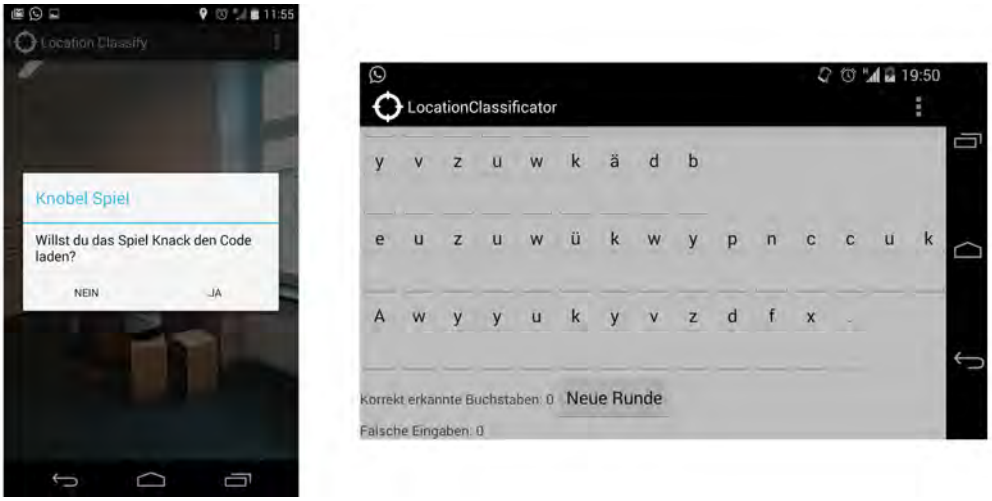


Abbildung 5.5: "Knack den Code" ist ein Exponat, bei dem an einem Rechner ein verschlüsselter Satz entschlüsselt werden soll. Wenn das Exponat gerade in Benutzung ist, kann über die mobile Anwendung eine App-Variante von dem Spiel geladen werden. Um zu diesem Unterpunkt in der Anwendung zu kommen, kann das Exponat einfach mit dem TBOC-Verfahren erkannt werden und das etsprechende in die Applikation implementierte Spiel öffnet sich dann automatisch. (Screenshot rechts ©PIMAR)

5.1.3 Knack den Code

Das letzte Exponat, welche exemplarisch in der Mathematikum-Anwendung umgesetzt wird, ist das Spiel "Knack den Code". Hier soll ein verschlüsselter Satz auf einem Computerbildschirm mit möglichst wenigen Fehlversuchen entschlüsselt werden. Bei den Sätzen sind die Buchstaben zufällig im Alphabet vertauscht. Durch betrachten der Auftrittswahrscheinlichkeit der einzelnen Buchstaben, kann der Schlüssel gefunden werden. Die Fehlversuche des Spielers werden gezählt und ausgegeben, was den Anreiz mit sich bringt, die Verschlüsselung mit so wenig Fehlern wie möglich zu lösen.

Für dieses Exponat wird kein AR-Sezneario entwickelt, sondern im Rahmen des Projektes PIMAR eine mobile Version des Spieles generiert (siehe Abbildung 5.5 rechts), welche über die Menüstruktur der Anwendung geladen wird. Um die Flexibilität und den Nutzen des TBOC-Verfahrens auch ohne die Einbindung eines AR-Szenarios zu demonstrieren, wurde das Exponat über das Verfahren angelernt, so dass es automatisch erkannt und die Spiele-Applikation automatisch geladen wird, wenn der Besucher sein mobiles Gerät auf das Exponat richtet (siehe Abbildung 5.5 links). So ist es möglich, jeden beliebigen Inhalt in einer Anwendung passend zu dem erkannten Objekt zu laden, ohne dass der Anwender die entsprechende Stelle in einer Menüstruktur finden muss.

5.2 Zusammenfassung

Eine prototypische Implementierung des TBOC-Verfahrens ist in einer realen Applikation für das mathematische Mitmach-Museum Mathematikum in Gießen integriert worden. Dafür wurde im Rahmen des Forschungsprojektes PIMAR eine App erstellt, die mit Hilfe des im Projekt entwickelten Generators generiert wurde und die auch als Demonstrator des TBOC-Verfahrens dient. Auf diese Weise wird innerhalb eines Museumsumfeldes die Bereitstellung von AR-Content zur Erweiterung von Informationen für die Exponate oder zur Hilfestellung des Betrachters dargelegt. Wobei das TBOC-Verfahren für die vorgeschaltete Initialisierung der AR-Szenarien genutzt wird. Damit wird ermöglicht, dass der entsprechende Bereich der mobilen Anwendung ohne zusätzliche Benutzereinwirkung geladen wird. Um ein möglichst breites Spektrum abzudecken, wurden drei sich stark voneinander unterscheidende Exponate ausgewählt, die exemplarisch in der Anwendung umgesetzt wurden. Zum einen das Exponat "Pi mit den Füßen", wo mit aufwendigen 3D-Animationen, dem Besucher des Museums das Exponat erklärt wird. Das zweite Exponat "Die Deutschlandtour" hingegen zeigt auf, wie mit einfachen Mitteln eine nützliche Hilfestellung erreicht werden kann. Hier werden lediglich über Labels (2D-Bilder) positionsabhängige Hinweise eingeblendet, die dem Besucher Hilfestellungen für die Lösung des Problems liefern.

Das nächste Kapitel beschäftigt sich mit der Auswertung der Ergebnisse des Verfahrens. Hier sollen die Performance sowie die Erkennungsraten des Verfahrens getestet und ausgewertet werden. Im Mittelpunkt stehen dabei die Fragestellungen wie sich das System bei Zunahme der Datenmenge verhält und ob die Anzahl der zu ermittelnden Objekte eine Auswirkung auf die Erkennungsrate hat.

Kapitel 6

Test und Auswertung des TBOC-Verfahrens

Dieses Kapitel beschäftigt sich mit dem Testen und Auswerten des in dieser Arbeit entwickelten Verfahrens. Nach der prototypischen Implementierung des Verfahrens in Form der TBOC-Applikation und dessen Integration in eine reale Anwendung für das Mathematikum in Gießen soll nun mit Hilfe eines Leistungstests die Performance des Verfahrens untersucht werden.

6.1 Einleitung

Um das in dieser Arbeit entwickelte Verfahren beurteilen zu können, soll es bezüglich seiner Funktionalität und Leistungsfähigkeit (Performance) untersucht werden. Die Durchführung der Tests wird, entsprechend der Struktur des Verfahrens, in zwei Bereiche unterteilt, den Anlernprozess und die Anwendungsphase. Zunächst erfolgt die Untersuchung des Anlernprozesses. Hierbei steht die Fragestellung im Mittelpunkt, wie sich das System bei unterschiedlichen Randbedingungen verhält. Diese Fragestellung lässt sich wie folgt untergliedern:

- Wie wirken sich große Anzahlen von Trainingsdaten oder eine Vielzahl von zu erkennenden Objekten auf die Größe der Trainingsdatendatei aus?
- Wie verändert sich die Zeitdauer, die das System zum erzeugen des Modells braucht, bei einer Erhöhung der Anzahl der Datensätze und der Objekte?

Die zweite Aussage, die aus diesem Kapitel gewonnen werden soll, ist die Bestimmung der Erkennungsrate in der Erkennungsphase. Die Erkennungsrate gibt hierbei Aufschluss über die Funktionalität und die Tauglichkeit des Verfahrens für die Initialisierung von Inhalten und über die zuverlässige Erkennung von Objekten. Auch hier gibt es verschiedene Abhängigkeiten und Fragestellungen:

Abbildung 6.1: Die Dateigröße verhält sich nahezu linear zur Anzahl der Trainingsdaten. Die Grafik zeigt, dass sich bei der Erhöhung der Datensätze in Zehnerschritten die Dateigröße mit einer Steigung von $m = 0,34$ und einem Korrelationskoeffizienten von $r = 0,9998$ linear ansteigt.

- Welche Störfaktoren beeinflussen das Ergebnis?
- Hat der Radius beim Anlernen eines Objektes Auswirkungen auf die gesamte Erkennungsrate?
- Verändert sich die Erkennungsrate bei Zunahme von Trainingsdaten und/oder zu erkennenden Objekten?
- Hat die Anzahl der Trainingsdaten oder der Objekte Auswirkungen auf die Zeit, die für die Erkennung benötigt wird?

6.2 Performance des TBOC-Verfahrens

In diesem Abschnitt soll dargelegt werden, wie sich die Anzahl der Trainingsdaten auf die Größe der Datensatz-Datei aber auch auf das Daten-Modell auswirkt. Hierfür werden Trainingsdaten-Dateien mit einer unterschiedlichen Anzahl von Objekten und/oder Trainingsdaten erstellt.

6.2.1 Auswirkungen auf die Dateigröße der Trainingsdaten

Zunächst erfolgt die Darstellung der Abhängigkeit der Anzahl von Trainingsdaten auf die Dateigröße der Trainingsdaten-Datei (ARFF-Datei). Hierfür werden die Datensätze in Zehner-Schritten erhöht. Abbildung 6.1 zeigt, dass sich die Dateigröße bei einer Steigung von $m = 0,34$ und einem Korrelationskoeffizienten von $r = 0,9998$, nahezu linear verhält. Bei einer Datei mit 100 Trainings-Datensätzen ergibt sich eine Dateigröße von *38kb*.

Wie im folgenden Abschnitt 6.3.1 dargelegt, werden in der Regel zwischen zehn bis 25 Trainingsdatensätze für jedes zu erkennende Objekt benötigt, um eine robuste Erkennung zu gewährleisten. So deckt man bei 100 Datensätzen und im Schnitt 17 Trainingsdaten sechs Objekte ab. Auch bei einer größeren Anzahl von Objekten bzw. Trainingsdaten bleibt die Dateigröße sehr klein, so wird die ARFF-Datei bei 1000 Datensätzen etwa *0,34mb* groß. Die Erhöhung der zu erkennenden Objekte wirkt sich daher nur minimal auf die Dateigröße aus. Dies ist nachvollziehbar, da die Erhöhung der Anzahl von Objekten nur mit einer Erweiterung der nominalen Attributliste verbunden ist.

6.2.2 Auswirkungen auf das Modell

Für die weitergehenden Untersuchung der Modelldatei und des Vorganges zur Erzeugung des Modells wird zunächst der Quellcode der TBOC-Anwendung um einen Zeitnehmer erweitert, der die benötigte Zeit für das Generieren und Abspeichern des Modells misst. So können neben der Dateigröße des Modells auch Informationen zur Erzeugung des Modells Aufschluss geben, wie sich die Anzahl der Datensätze und die Anzahl der zu erkennenden Objekte auf das Modell auswirken. Die Zeitmessungen erfolgen mit den von Java zur Verfügung gestellten Tools. Mit `System.currentTimeMillis()` wird die Zeit vom 1. Januar 1970 in Millisekunden als `long` ausgegeben. `System.nanoTime()` liefert einen Zeitstempel des genauesten System-Zeitgebers des Gerätes zurück. Dieser hat keinen Bezugspunkt zu einem Datum. Durch die Differenz zweier Zeitwerte kann für die Ausführungszeit von Programmen oder Programmabschnitten ermittelt werden. Eine Laufzeitmessung in Java fällt wegen der *Java Virtual Maschine* relativ ungenau aus. Wie genau der so erhaltene Zeitwert ist, hängt hierbei unter anderem von der allgemeinen Systemlast (Garbage Collector, Hintergrundprozesse oder andere Threads) sowie von der Timer-Auflösung der jeweiligen Plattform ab. Um Aussagen bezüglich der Performance des Verfahrens hinsichtlich der Erstellung des Modells bzw. für die Erkennungszeit eines Objektes zu erhalten, reicht die Genauigkeit des somit erhaltenen Zeitwertes aber aus.

Auswirkungen auf die Dateigröße und die Erstellungszeit

Für den Test werden aus Trainingsdaten-Dateien Modelle erzeugt, die Dateigröße ermittelt und die benötigte Zeit zur Erstellung der Modelle gemessen. Im ersten Schritt werden die Datensätze in einer Datei in Zehnerschritten erhöht bei gleichbleibender Objektzahl. Da einzelne Messungen Abweichungen in der Erstellungszeit aufweisen, werden je Datensatz zehn Messungen vollzogen und daraus der Mittelwert genommen. Es zeigt sich, dass die Anzahl der Datensätze keine Auswirkung auf die Dateigröße des Modells hat. Bei zehn sowie auch bei 100 Datensätzen bei jeweils drei Objekten blieb die Dateigröße konstant und lag bei *28kb*. Bei der Zeitmessung jedoch ist ein leichter Anstieg der Zeit, die für die Erstellung des Modells benötigt wird, zu erkennen. Die Abbildung 6.2 links zeigt den Verlauf der Messwerte. Mit einer Steigung von $m = 3,114$ ist eine lineare Abhängigkeit der Erstellungszeit ersichtlich. Hierbei wird für zehn Datensätze eine Zeit von 1,497

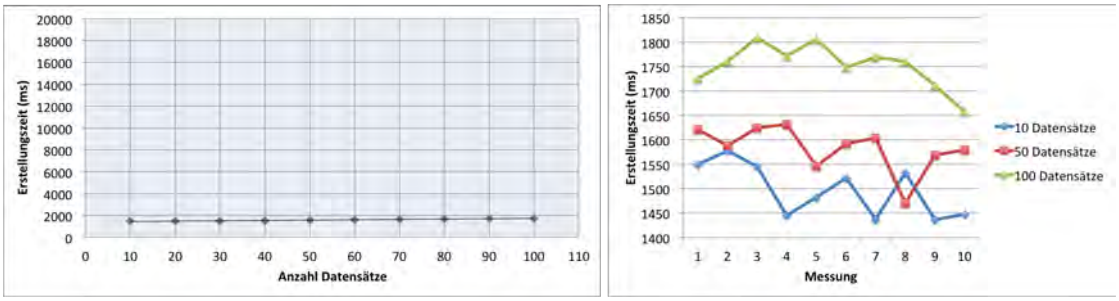


Abbildung 6.2: Die Erhöhung der Datensätze hat für die Erstellungszeit des Modells nur geringe Auswirkungen. Sie nimmt mit einer Steigung von m=3,114 zu. Die Werte in der Abbildung links sind die gemittelten Zeitwerte über zehn Messungen. Die rechte Abbildung zeigt, dass innerhalb der Messreihe größere Abweichungen auftreten. Die Standardabweichung der einzelnen Messreihen liegt zwischen 38,737 und 61,607.

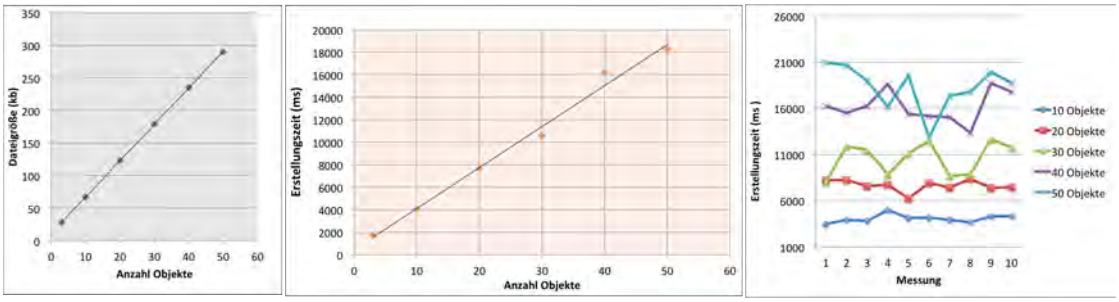


Abbildung 6.3: Die Erhöhung der zu erkennenden Objekte hat im Gegensatz zu der Erhöhung der Trainingsdaten eine erhebliche Auswirkung auf die Dateigröße (Abbildung links). Bei der Erhöhung der Objektanzahl steigt die Zeit, die für die Erstellung des Modells benötigt wird, stärker an (Abbildung Mitte). Ebenso steigt die Differenz der einzelnen Messungen untereinander (Abbildung rechts).

Sekunden für die Modellerstellung benötigt. Bei 100 Datensätzen sind es dann ca. 1,752 Sekunden. Abbildung 6.2 rechts zeigt exemplarisch für drei ausgewählte Datensätzen die Abweichungen innerhalb der Messreihen auf. Hierbei liegt die Standardabweichung der einzelnen Durchläufe zwischen 38,737 und 61,607. Diese Schwankung der Einzelmessungen ist auf die Systemungenauigkeiten der Zeitmessung zurückzuführen.

Im zweiten betrachteten Fall werden die Datensätze konstant mit 100 angesetzt und die Anzahl der Objekte wird erhöht. Zunächst von drei auf zehn und anschließend in Zehnerschritten bis zu 50 Objekten. Bei der Erhöhung der Objektanzahl ist ein deutlicher Anstieg der Dateigröße des Modells erkennbar (siehe Abbildung 6.3 links). Mit einer Steigung von $m = 5,58$ und einem Korrelationskoeffizienten der von $r = 0,999$ steigt die Größe der Datei linear an.

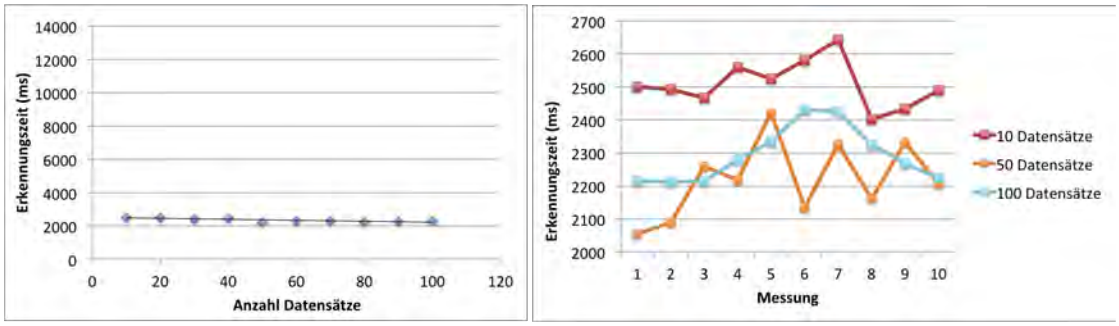


Abbildung 6.4: Durch die Erhöhung der Datensätze bei gleichbleibender Anzahl der Objekte ist eine leichte Verbesserung bei der Erkennungszeit erkennbar. Bei einer Spanne von einem bis 100 Datensätzen werden Zeiten zwischen 2,496 und 2,292 Sekunden gemessen. Die einzelnen Abweichungen der Messreihen haben eine Standardabweichung von 39,696 bis 116,095.

Für die Messung der Zeit, die das System benötigt, um das Modell zu generieren, wird wieder der Mittelwert von zehn Messungen genommen und in Abbildung 6.3 Mitte als Graph abgebildet. Auch hier weisen die Abweichungen der einzelnen gemessenen Werte in den Messreihen größere Schwankungen auf und haben als Standardabweichung einen Wert zwischen 424,948 und 2415,826 (siehe Abbildung 6.3 rechts).

Auswirkungen auf die Erkennungszeit

Ähnlich wie bei den vorangestellten Betrachtungen zur Erstellungszeit erfolgt die Darstellung der Auswirkungen auf die Erkennungszeit. Die Veränderungen der Dateigröße und der Erkennungszeit wird durch die Erhöhung von Datensätzen und der Objektanzahl ermittelt. Zunächst wird bei einer Anzahl von drei Objekten eine Trainingsdaten-Datei für die Messung verwendet und nach und nach die Anzahl der Trainingsdaten erhöht. Gestartet wird mit einem Datensatz, dann wird kontinuierlich bis auf zehn Datensätze erweitert. Anschließend werden die Datensätze in Zehnerschritten erhöht, bis die letzte Trainingsdaten-Datei 100 Datensätze beinhaltet. Um die Schwankungen der Zeitmessung auszugleichen, werden wieder für jede Trainingsdaten-Anzahl zehn Messungen durchgeführt und der Mittelwert für die Auswertung herangezogen. Wie in Abbildung 6.4 links zu sehen ist, nimmt mit Erhöhung der Datensätze die Zeit, die für die Erkennung benötigt wird mit einer Steigung von $m = -0,2467$ linear leicht ab. Die Standardabweichung der einzelnen Messungen liegt zwischen 39,696 und 116,095. Die Schwankungen in den einzelnen Messungen werden in Abbildung 6.4 exemplarisch für 10, 50 und 100 Datensätze aufgezeigt.

Gegensatz zur Erhöhung der Datensätze hat die Erweiterung um Objekte wieder eine größere Auswirkung auf die Erkennungszeit. Für den Test werden 100 Trainingsdaten verwendet. Die Anzahl der Objekte wird von drei auf zehn und anschließend in Zehner-Schritten bis auf 50 erhöht. Bei den Messungen ist ein linearer Anstieg von $m = 239,948$

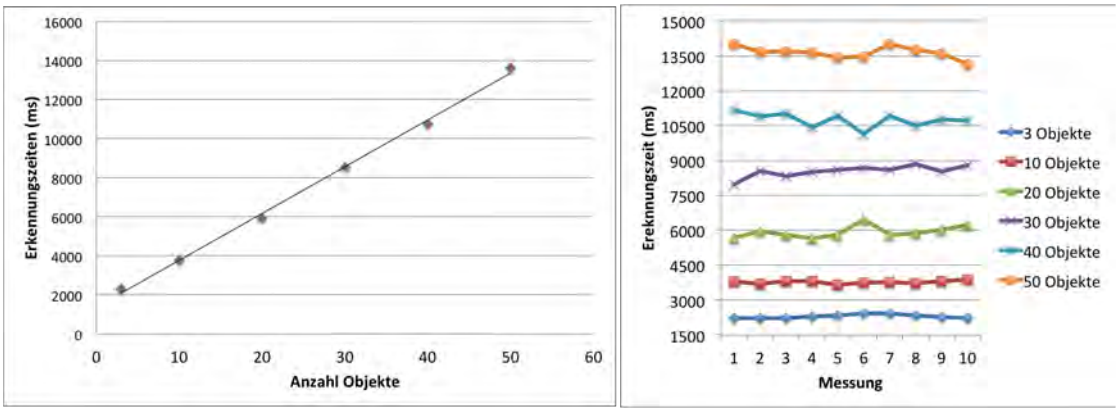


Abbildung 6.5: Bei der Erhöhung der Objekte bei gleichbleibender Anzahl der Datensätze steigt die Zeit für die Erkennung mit einer Steigung von $m = 239,948$ linear an.

abzulesen (siehe Abbildung 6.5 links). Die Abweichungen der einzelnen Messungen, die in Abbildung 6.5 rechts dargestellt werden, weisen eine Standardabweichung zwischen 66,570 und 309,025 auf.

6.2.3 Statistische Hypothesenprüfung in Bezug auf das Datenmodell

Um die einzelnen Messwerte aus den vorangegangenen Tests statistisch zu verifizieren, wird die Abhängigkeit der einzelnen Messwerte über eine Teststatistik ausgewertet. Hierbei sollen die einzelnen Hypothesen der jeweiligen Tests bestätigt werden.

Im ersten Abschnitt sollen zwei Fragestellungen bezüglich der Dauer der Modellgenerierung untersucht werden. Dabei soll statistisch nachgewiesen werden, ob mit zunehmender Datensatzanzahl oder mit Erhöhung der Anzahl der Objekte ein Einfluss auf die Dauer der Erzeugung des Modells besteht. Wie die Darstellung der Testdaten im vorangegangenen Abschnitt deutlich zeigt, ist ein offensichtlicher Trend erkennbar. Aus der reinen Beobachtung lässt sich die Vermutung schließen, dass die Erstellungszeit einen linearen Zusammenhang mit der Steigerung der Datensätze sowie mit der Erhöhung der Objekte hat. Um diese Vermutung zu bestätigen, werden die gewonnen Messwerte über `MYSTAT`[1], der freien Version der Statistik-Analyse-Software `SYSTAT` [WBG96] über eine Teststatistik ausgewertet.

Hierfür werden die einzelnen Ergebniswerte der jeweiligen Messungen in die Software geladen. `MYSTAT` erstellt für die Darstellung der Messwerte ein Box-Plot-Diagramm (siehe Abbildung 6.6 links und Mitte). Das mittlere Diagramm in der Abbildung zeigt die insgesamt 50 einzelnen Messungen in 5 Messreihen bei 10 bis 50 Objekten. Die Striche in den Boxen stehen für den Mittelwert der einzelnen Messreihen. Die dazugehörigen Daten werden für

[1] http://www.systat.com/MystatProducts.aspx Stand: 26. März 2015

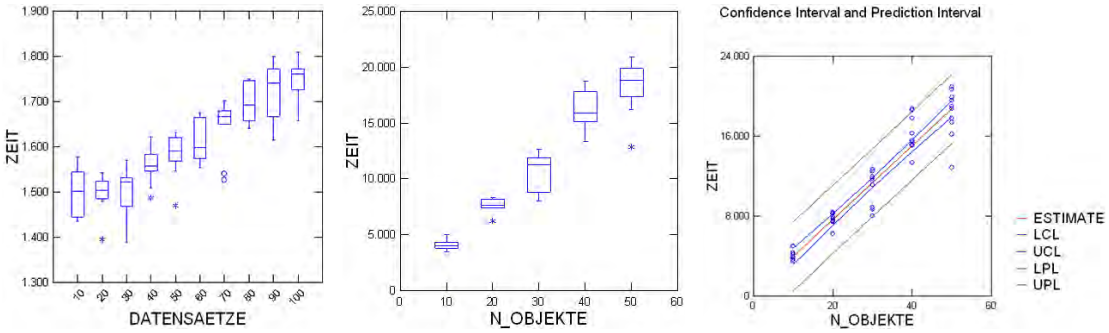


Abbildung 6.6: Box-Plot-Darstellung der Daten aus den Messungen bei Erhöhung der Anzahl der Datensätze (links) und bei Erhöhung der Anzahl der Objekte (Mitte). Die Darstellung der Einzelwerte bei Erhöhung der Objektanzahl mit Regressionsgerade (rechts).

den Test mit der Erhöhung der Anzahl der Objekte in der Tabelle 6.1 gezeigt. Abbildung 6.6 rechts zeigt exemplarisch einen x,y-Plot aller 50 einzelnen Messwerte des Tests mit einer Erhöhung der Objektanzahl. Die Abbildung zeigt neben den Einzelwerten auch die Regressionsgerade als rote Linie sowie die Vertrauensbereiche der Daten.

OBJEKTE	n=10	n=20	n=30	n=40	n=50
N of Cases	10	10	10	10	10
Minimum	3471	6229	7991	13337	12841
Maximum	4989	8362	12665	18717	20915
Range	1518	2133	4674	5380	8074
Median	4029	7639,5	11298,5	15875,5	18851
Arithmetic Mean	4070,5	7650	10566,7	16199,6	18285
Standard Deviation	424,948	616,91	1782.373	1714,90	2415,826

Tabelle 6.1: Auswertung der Messwerte pro Messreihe bei der Ermittlung der Erstellungszeit mit einer Erhöhung der Objekte.

Ob die Daten linear an- oder absteigen, kann über den Korrelationskoeffizienten r angeben werden. Bei einem Wert von ± 1 besteht eine starke Korrelation. Wenn der Wert gegen 0 geht, nennt man das unkorreliert. Das bedeutet, dass keine lineare Abhängigkeit der Werte besteht. [Sac13] Je dichter sich die einzelnen Werte an der Regressionsgeraden befinden, um so höher ist der Korrelationskoeffizient. Die Tabelle 6.2 zeigt die berechneten Korrelationskoeffizienten für die beiden Tests der Erstellungszeiten. Beide Werte zeigen einen deutlichen Trend, der bedeutet, dass die Werte voneinander abhängen. Bei der Erhöhung der Objektanzahl ist die Abhängigkeit der beiden Werte mit einem Wert von $0,954$ sogar noch etwas deutlicher. Anhand der Beobachtungen und der errechneten Korrelationskoeffizienten wird die Hypothese aufgestellt, dass die Erhöhung der Datensätze sowie auch der Anzahl der Objekte einen Einfluss auf die Zeit haben, die für die Modellgenerierung benötigt wird. Um

Dependent Variable	Datensätze	Objekte
N	100	50
Multiple R	0,875	0,954
Squared Multiple R	0,765	0,911
Adjusted Squared Multiple R	0,763	0,909
Standard Error of Estimate	50,060	1672,656

Tabelle 6.2: Der Wert bei Multiple R (farblich markiert) steht für den Korrelationskoeffizienten bei den beiden Tests zur Erstellungszeit des Modells.

diese Hypothese statistisch zu belegen, wird mittels des "Tests auf Korrelation" überprüft, wie hoch die Wahrscheinlichkeit ist, dass die Werte trotzdem durch Zufall entstanden sind. Der Test auf Korrelation liefert einen Wahrscheinlichkeitswert p (engl.: probability). Dieser `p-Wert` steht für die "Irrtumswahrscheinlichkeit", dass die aufgestellte Hypothese des Zusammenhanges der Dauer der Modellgenerierung mit der Anzahl der Datensätze nicht zufällig ist. In Tabelle 6.3 kann der Wert in der Spalte "p-value" abgelesen werden. Da MYSTAT nur die ersten drei Nachkommastellen anzeigt, wird davon ausgegangen, dass die Werte kleiner als $0,001$ sind, also kleiner als $0,1\%$. Bei einem angesetzten Signifikanzniveau von 5% liegen die Irrtumswahrscheinlichkeiten weit unter diesem Wert.
Somit können die beiden Hypothesen bestätigt werden. Es besteht eine statistisch signifikante Abhängigkeit mit einer Irrtumswahrscheinlichkeit von unter $0,1\%$, dass die Dauer der Modellbildung von der Anzahl der Datensätzen sowie auch von der Anzahl der Objekte abhängt.

Effect	Coefficient	Stdrd. Error	Std. Coeffi.	Tolerance	t	p-value
N_OBJEKTE	369,786	16,727	0,954	1,000	22,108	0,000
DATENSAETZE	3,114	0,174	0,875	1,000	17,869	0,000

Tabelle 6.3: Der Wahrscheinlichkeitswert p gibt die Irrtumswahrscheinlichkeit an. MYSTAT zeigt nur drei Nachkommastellen an, somit ist der Wert kleiner als $0,001$, also kleiner als $0,1\%$.

Die beiden folgenden Fragestellungen beziehen sich auf die Erkennungszeit. Dabei soll statistisch nachgewiesen werden, dass die Zeit für eine Erkennung durch die Erhöhung der Datensatzanzahl oder der Erhöhung der Anzahl der Objekte beeinflusst wird. Auch hier kann durch Beobachtung der Messwerte diese Vermutung, wie im vorangegangenen Abschnitt beschrieben, abgeleitet werden. Für den statistischen Beweis werden die einzelnen Messwerte ebenfalls in `MYSTAT` eingelesen. Auf den x,y-Plots ist bei der Erhöhung der Objektanzahl der lineare Anstieg der Messwerte zu beobachten (siehe Abbildung 6.7 rechts). Auch dass die einzelnen Messwerte sehr nah an der Regressionsgeraden liegen, ist aus der Abbildung gut zu erkennen. Im linken Teil der Abbildung hingegen kann durch den großen Abstand der einzelnen Messwerte zu der Regressionsgeraden schon erahnt werden, dass eine geringerer Korrelationskoeffizient zu erwarten ist.

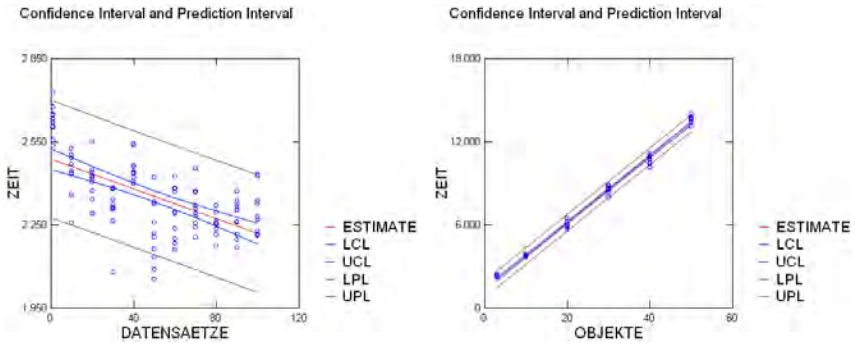


Abbildung 6.7: Die Darstellung der einzelnen Messwerte der Erkennungszeiten bei einer Erhöhung der Anzahl der Datensätze (links) und bei der Steigerung der Objektanzahl (rechts) mit der Regressionsgeraden und den Vertrauensbereichen.

Die errechneten Korrelationskoeffizienten, die von `MYSTAT` geliefert werden, bestätigen die Beobachtungen. Bei der Erhöhung der Datensätze beträgt der Korrelationskoeffizient $-0,630$. Hier kann gerade noch von einem erkennbaren statistischen Zusammenhang gesprochen werden, da der Wert innerhalb der in der Regel angesetzten Grenze von unter $-0,6$ und über $0,6$ liegt.[2] Bei der Erhöhung der Objektanzahl steht der Wert von $0,997$ für einen starken linearen Zusammenhang. Um diese Abhängigkeiten statistisch zu belegen, werden auch hier zwei Hypothesen aufgestellt, dass die Anzahl der Datensätze und die Anzahl der Objekte einen Einfluss auf die Erkennungszeit haben. Auch hier werden die Irrtumswahrscheinlichkeiten über den `Test auf Korrelation` ermittelt (siehe Tabelle 6.4). Die Irrtumswahrscheinlichkeiten betragen in beiden Fällen weniger als $0,001$. Somit liegen diese Werte unter dem angesetzten Signifikanzniveau von 5% und die Hypothesen können so als statistisch belegt angesehen werden.

Effect	Coefficient	Stdrd. Error	Std. Coeffi.	Tolerance	t	p-value
OBJEKTE	239,948	2,309	0,997	1,000	103,920	0,000
DATENSAETZE	-2,693	0,320	-0,630	1,000	-8,424	0,000

Tabelle 6.4: Der Wahrscheinlichkeitswert p gibt die Irrtumswahrscheinlichkeit an. MYSTAT zeigt nur drei Nachkommastellen an, somit ist der Wert kleiner als $0,001$, also kleiner als $0,1\%$.

Es besteht mit einer Irrtumswahrscheinlichkeit von unter $0,1\%$ eine Abhängigkeit der Erkennungszeit zur Erhöhung der Datensätze sowie zu einer Steigerung der Objektanzahl.

[2] http://goo.gl/rb6m3k Stand: 29. März 2015

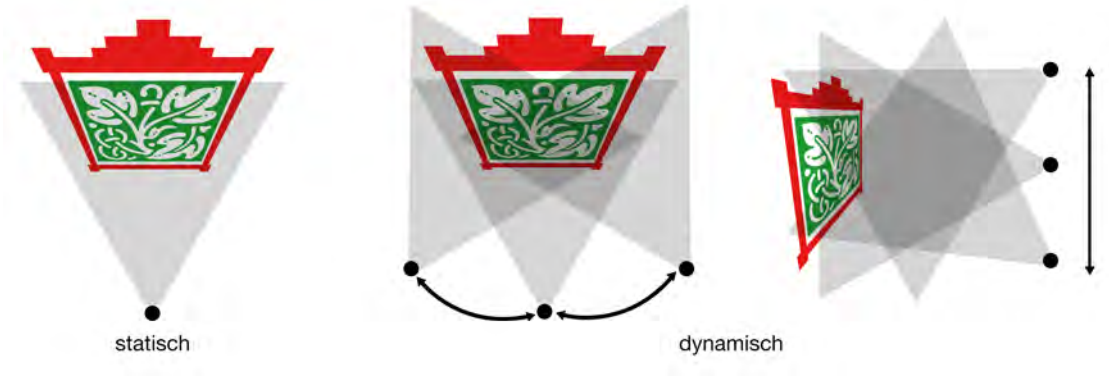


Abbildung 6.8: Das Anlernen und Erkennen erfolgt durch statische und dynamische Positionierung, wobei mit statisch eine feste Position mit optimaler Ausrichtung auf das Exponat / Objekt gemeint ist. Bei dynamischem Anlernen oder Erkennen wird das Objekt aus mehreren sinnvollen Perspektiven angelernt und erkannt.

6.3 Ermittlung der Erkennungsraten

Bei der Ermittlung der Erkennungsraten müssen zunächst die möglichen Positionen der späteren Anwender der App berücksichtigt werden. Im besten Fall richtet ein Museumsbesucher sein Gerät exakt senkrecht auf das Exponat aus. Dies kann jedoch nicht immer vorausgesetzt werden und ist aufgrund der Anwesenheit anderer Museumsbesucher oft auch in einigen Situationen vielleicht gar nicht möglich. Also sollte das Verfahren in der Lage sein, auch aus anderen Positionen und Blickwinkeln zu einem richtigen Ergebnis zu gelangen. Wenn ein Objekt über die optimale Position angelernt wird, wird das als statisches Anlernen bezeichnet. Wenn der Benutzer nur aus dieser Position das Exponat mit der Applikation erkennt, wird es auch statisches Erkennen genannt. Der Ansatz, ein Exponat aus allen möglichen Positionen anzulernen, wird dynamisches Anlernen genannt. Hierbei wird das Objekt aus einer Vielzahl von Blickrichtungen und unterschiedlichen Neigungen des Gerätes angelernt, so dass das Objekt später auch aus diesen Blickrichtungen richtig zu erkennen ist (siehe Abbildung 6.8). Es ist jedoch zu beachten, dass durch das dynamische Anlernen auch Erkennungsfehler auftreten können, wenn die Daten der einzelnen Sensoren in bestimmten Positionen keine eindeutigen Ergebnisse mehr liefern können. In einem Laborversuch soll die Notwendigkeit des dynamischen Anlernens aufgezeigt werden.

6.3.1 Statisches und dynamisches Anlernen im Vergleich

Um das Verhalten der Erkennungsraten beim dynamischen Anlernen, verbunden mit der Erhöhung der Datensätze zu untersuchen, wird unter Laborbedingungen folgender Test durchgeführt. Es werden drei optisch sehr unterschiedliche Exponate simuliert, die auch in klar zuordnungsfähigen Himmelsrichtungen ausgerichtet sind. Der Versuch wird in zwei Phasen unterteilt. In der ersten Phase werden die drei Objekte statisch angelernt und dabei

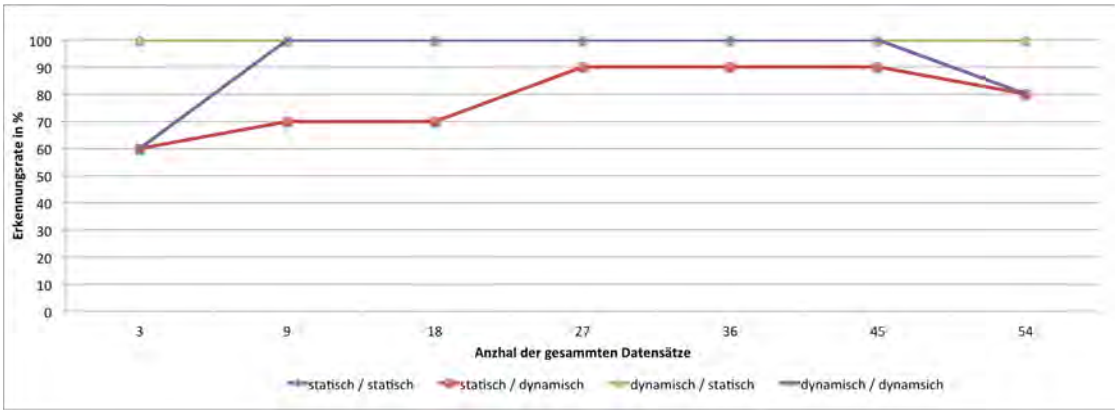


Abbildung 6.9: Die Abbildung zeigt die Messwerte der Erkennungsrate bei statischem und dynamischem Anlernen und Erkennen. Abzulesen ist, dass das statische Erkennen eine 100%-ige Erkennungsrate liefert, wenn statisch oder dynamisch angelernt wird. Wenn nur statisch angelernt wird und aus unterschiedlichen Positionen eine Erkennung erfolgen soll, spiegelt sich das in einer niedrigeren Erkennungsrate wieder.

Schritt für Schritt die Datensätze erhöht. Nach jedem Erhöhen der Datensätze werden die Erkennungsraten der Objekte ermittelt. Dabei wird jeweils aus der statischen Position und aus einer beliebigen Position (dynamisches Erkennen) gemessen. Gestartet wird mit jeweils einem Trainingsdatensatz pro Objekt. Danach werden pro Objekt zwei weitere hinzugefügt, so dass insgesamt neun Trainingsdatensätze benutzt werden. In den nachfolgenden Messreihen werden die Datensätze insgesamt in Neuner-Schritten erhöht. In den einzelnen Messdurchläufen werden über die drei Objekte zehn Messungen gemacht und somit die Erkennungsraten ermittelt. Abbildung 6.9 zeigt, dass bei statischem Anlernen und Erkennen die Erkennungsrate durchgängig bei 100% lag. Beim statischen Anlernen und dynamischen Erkennen liegt die Erkennungsrate zwischen 60% bis 90% mit ansteigender Anzahl der Datensätze. Die noch recht hohe Erkennungszahl von 60% bei nur drei Datensätzen und dynamischem Erkennen ist auf die wenigen Messdurchläufe und die dabei doch nah an der optimalen (statisch angelernten) Position liegenden Perspektive zurückzuführen. Jedoch ist aus dem Graphen in Abbildung 6.9 gut abzulesen, dass ein nur statisches Anlernen zu schlechteren Erkennungsraten führt, die durch die Erhöhung der einzelnen Anlerndatensätze jedoch etwas verbessert werden können. In der zweiten Phase des Testes werden die Objekte dynamisch angelernt und dann jeweils statisch und dynamisch erkannt. Beim dynamischen Anlernen ist die erste Position jedoch auch die für das Exponat optimale statische, wodurch sich das gleiche Ergebnis bei der Erkennungsrate zeigt. Sobald aber bei den weiteren Datensätzen die unterschiedlichen Blickrichtungen miteinbezogen werden, steigt die Erkennungsrate auch beim dynamischen Erkennen bei neun Datensätzen auf 100%. Dass natürlich nicht alle Positionen immer abgedeckt werden können, zeigt die letzte Messung mit 54 Datensätzen, bei denen unter anderem das mobile Gerät sehr dicht an die einzelnen Exponate gehalten wird, so dass der größte Teil des Exponates sich nicht im Blickfeld der Kamera befindet. Dies führt bei zwei Exponaten zu Fehlerkennungen (siehe Abbildung 6.9).

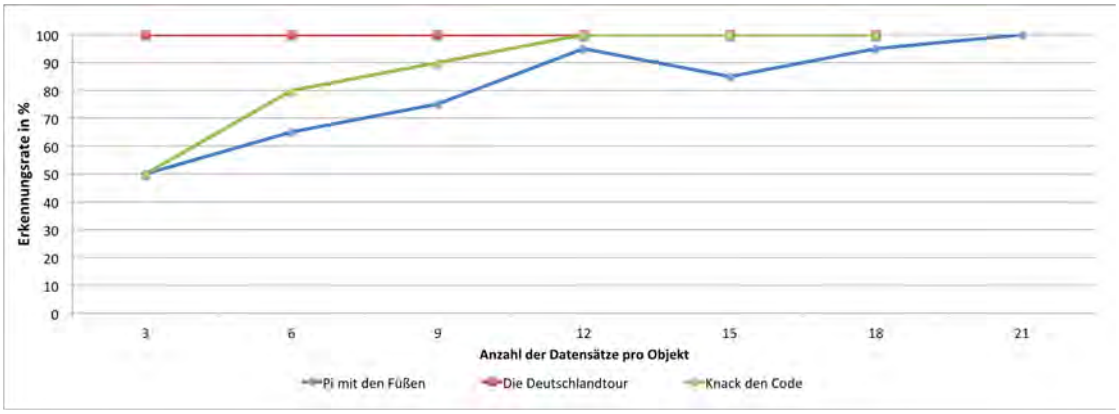


Abbildung 6.10: Bei den drei Exponaten, die in der Mathematikum-Anwendung exemplarisch implementiert wurden, werden die Erkennungsraten ermittelt. Hierfür wird die Anzahl der Trainingsdatensätze so lange erhöht, bis alle drei Exponate eine Erkennungsrate von 100% aufweisen.

6.3.2 Erkennungsraten in Bezug zu den Trainingsdaten

Mit der Erkenntnis, dass dynamisches Anlernen im allgemeinen die Erkennungsrate verbessert und somit auch gerade im Museumsbetrieb eine bessere Erkennung gewährleistet, da man nicht sicher sein kann, von wo ein Besucher versucht, das Exponat zu erkennen, soll in einer Auswertung festgehalten werden, wie sich die Erkennungsrate im Museumbetrieb in Bezug auf die Anzahl der Datensätze verhält. Hierfür werden die drei Exponate, die in der Mathematikum-Anwendung exemplarisch implementiert werden, dynamisch angelernt und die Erkennungsraten jedes einzelnen Objektes ermittelt. In jedem Durchlauf werden für die Exponate jeweils drei neue Trainingsdatensätze angelernt. Es stellt sich heraus, dass das Exponat "Die Deutschlandtour" schon im ersten Durchlauf mit jeweils drei Trainingsdaten sehr dominant die Erkennungsrate beeinflusst. Zum einen wird das Exponat durchweg mit 100% erkannt, zum anderen hat es auch Einfluss auf die anderen beiden Exponate, da es zunächst noch häufig an Stelle der anderen erkannt wird. Diese Fehlerkennungen nehmen mit Zunahme weiterer Trainingsdaten ab. Beim Exponat "Knack den Code" wird ab zwölf Trainingsdatensätzen eine Erkennungsrate von 100% erreicht. Das Exponat "Pi mit den Füßen" hingegen wird durch Lichtreflexe am Boden öfter falsch erkannt. Um das "schwächer" scheinende Exponat zu stärken, werden im letzten Durchlauf nur dessen Trainingsdaten um drei erweitert, um dann bei allen drei Exponaten eine Erkennungsrate von 100% zu erreichen (siehe Abbildung 6.10).

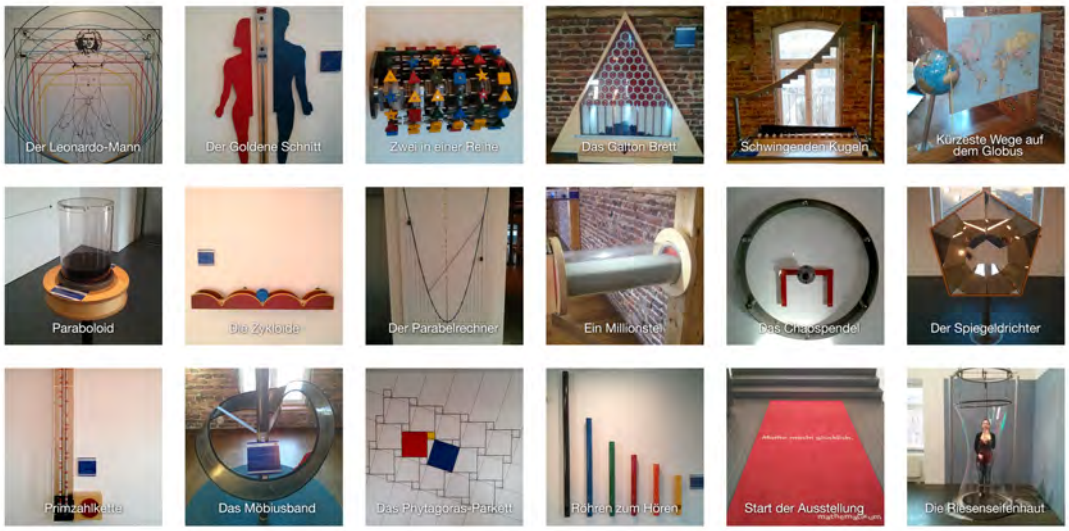


Abbildung 6.11: Übersicht der 18 Exponate, die im Mathematikum für den Test der Erkennungsraten bei Erhöhung der Objektanzahl herangezogen werden. Die Anzahl der Exponate wird in jedem Durchlauf schrittweise immer um drei Objekte erweitert und anschließend die Erkennungsrate jedes einzelnen Objektes ermittelt.

6.3.3 Erkennungsraten in Bezug auf die Objektanzahl

Eine wichtige Erkenntnis, die aus diesem Kapitel noch gezogen werden soll, ist das Verhalten der Erkennungsrate in Bezug auf die Objektanzahl. Hierfür werden im Mathematikum bis zu 18 Exponate mit dem System nacheinander angelernt und anschließend der Verlauf der einzelnen Erkennungsraten ermittelt. Einen Überblick über die einzelnen Exponate, die für diesen Test verwendet werden, zeigt Abbildung 6.11. Bei der Auswahl wird ein breites Spektrum von Exponaten des Mathematikums abgedeckt. Eine Voraussetzung bei der Wahl der Exponate war, dass sie jederzeit mehr oder weniger in gleicher Form und Position zur Verfügung stehen, um die Erkennungsraten nicht durch zusätzliche Parameter zu beeinflussen. Exponate die aus vielen Einzelteilen bestehen (z.B. Puzzleteile) und es somit nicht gewährleistet ist, dass die aktuelle Situation bei der Erkennung der Situation bei der Anlernung entspricht, werden für diesen Test nicht in Erwägung gezogen.

Im ersten Testlauf werden zunächst die ersten drei Exponate mit jeweils 15 Trainingsdaten angelernt und die Erkennungsraten über zehn Messungen ermittelt. Geprüft wird, ob das richtige Objekt erkannt wird oder das System ein falsches Objekt zurückgibt. Anhand der Häufigkeit der richtigen Erkennung über die zehn Messungen wird die prozentuale Erkennungsrate errechnet. In jedem Messdurchlauf wird die Anzahl der Objekte um drei Objekte erweitert und für die neuen Objekte ebenfalls 15 Trainingsdaten angelernt. Abbildung 6.12 zeigt den Verlauf der Erkennungsrate aller Objekte anhand des Mittelwertes und zum Vergleich die Erkennungsraten zweier Exponate, die schon beim ersten Durchlauf angelernt

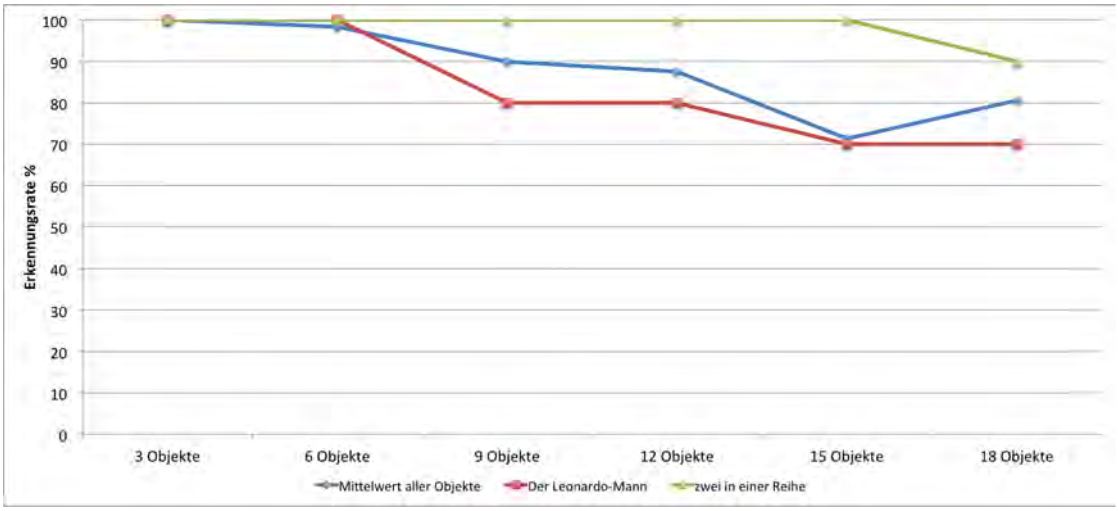


Abbildung 6.12: Verlauf der gesamten Erkennungsrate bei Erhöhung der Objekte. Die blaue Linie zeigt den Mittelwert der Erkennungsrate aller Objekte im Verlauf des Testes an. Die rote und grüne Linie zeigt zum Vergleich die Erkennungsraten von zwei Objekten, die seit dem ersten Testdurchlauf berücksichtigt werden. Es ist zu erkennen, dass mit Zunahme der Objekte die gesamte Erkennungsrate sinkt, es jedoch Exponate gibt, die durchgängig eine Erkennungsrate von über 90% aufweisen.

wurden. Hierbei lässt sich ablesen, dass Exponate über den gesamten Messdurchgang gut erkannt werden und andere durch das Hinzufügen weiterer Objekte in ihrer Erkennungsrate sinken. Wenn Objekte mit ähnlichen Werten hinzukommen, werden diese häufig untereinander falsch erkannt. Hingegen können sich "dominante" Exponate, also Objekte mit sehr eindeutigen Werten in der Vielzahl von Objekten durchsetzen. Die Abbildung 6.13 zeigt den Verlauf jedes einzelnen Exponates mit Hilfe eines Balkendiagramms. Im Vergleich mit den Bildern der Exponate in Abbildung 6.11 kann nachvollzogen werden, dass einzelne Objekte Ähnlichkeiten in den Farbwerten des gesamten Bildes, als auch einzelner Bildregionen aufweisen. Das führt bei der Auswertung der optischen Werte zu ähnlichen Ergebnissen. Da alle Exponate mit einer gleichen Anzahl von Trainingsdaten angelernt werden, können "schwache" Exponate, Objekte, die durch ihre optischen Werte nicht eindeutig zugeordnet werden können, sich anderen Exponaten gegenüber nicht durchsetzen. Eine teilweise Erhöhung der Trainingsdaten kann diese Objekte stärken.
Eine weitere Ursache, die die Erkennungsrate in einzelnen Messdurchläufen zum Schwanken bringt, ist der äußere Einfluss auf das Objekt. Hier kann zum Beispiel eine Änderung der Lichtverhältnisse (starker Sonnenschein) die Gegebenheiten so stark verändern, dass eine falsche Erkennung erfolgt. Um dies zu verhindern, muss in der Anlernphase versucht werden, äußere Einflüsse auf das Objekt mit zu berücksichtigen und damit auch diese Gegebenheiten im System zu hinterlegen. Der starke Einbruch der Erkennungsraten, der beiden Exponate "Die Zykloide" und "Der Parabelrechner" im vorletzten Messdurchlauf ist auf ein solches Problem zurückzuführen.

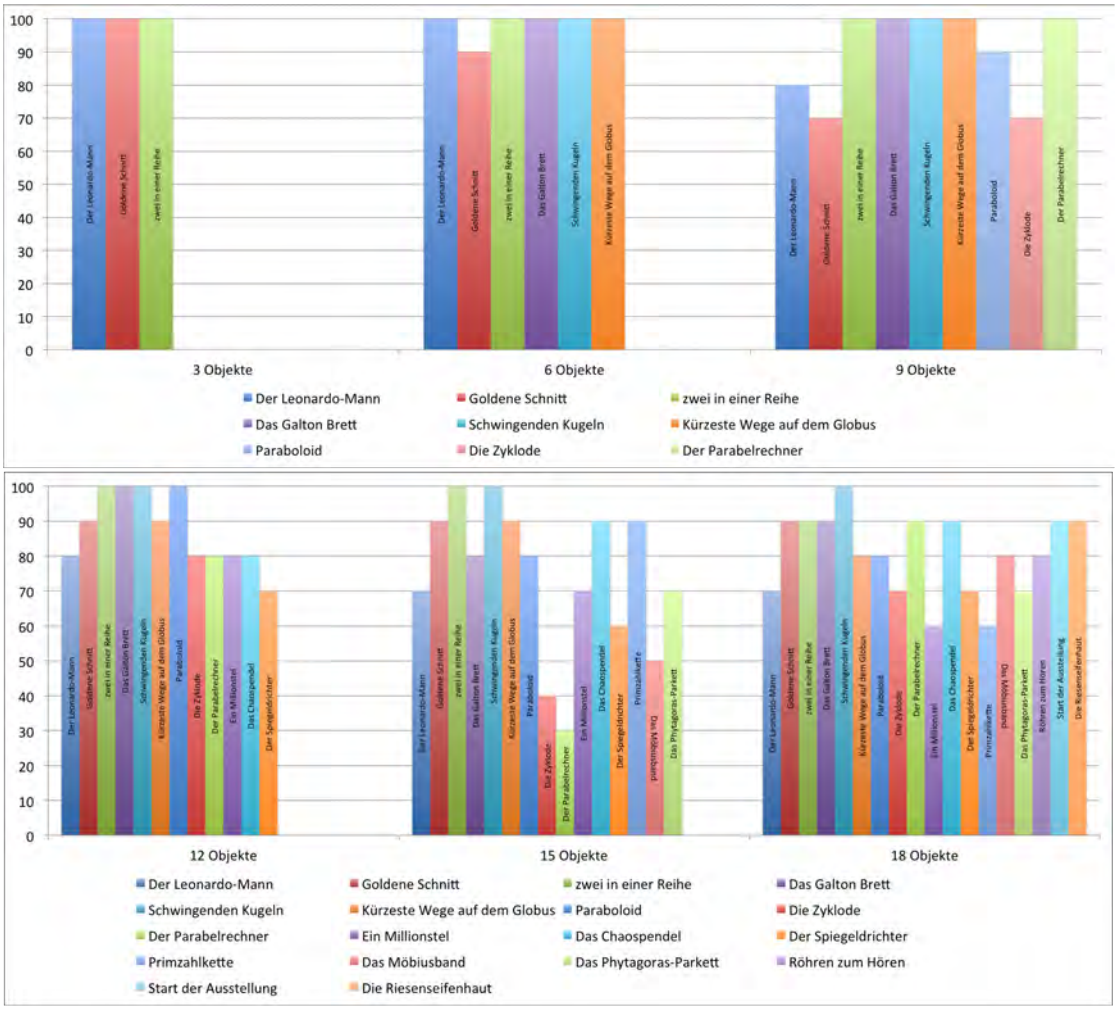


Abbildung 6.13: Die einzelnen Erkennungsraten aller Objekte pro Testdurchlauf. In jedem Testdurchlauf werden drei Objekte mit jeweils 15 Trainingsdaten mit in den Test einbezogen und jeweils die Erkennungsrate über zehn Messungen ermittelt. Es ist erkennbar, dass sich mit Erhöhung der Objektanzahl teilweise die Erkennungsrate einzelner Objekte reduziert. Aber auch äußere Einflüsse, wie starke Änderungen der Lichtverhältnisse während eines Testdurchlauf, können zu einer Verschlechterung der Erkennungsrate führen.

6.3.4 Statistische Hypothesenprüfung in Bezug auf die Erkennungsrate

Auch bei den Tests der Erkennungsraten soll über einen statistischen Test ermittelt werden, ob die Daten nur zufällig sind oder ob eine statistische Signifikanz bestimmt werden kann. Bei den Tests bezüglich der Erkennungsraten liegen keine messbaren Zahlen

vor, sondern eine Klasse mit JA (erkannt) und NEIN (nicht erkannt). Die Rate der Erkennungen wird mit Häufigkeitstabellen getestet. Das bekannteste Testverfahren dafür ist der Chi-Quadrat-Vierfeldertest. Bei diesem Test werden zwei Attribute mit je zwei Ausprägungen gegeneinander getestet. Bei den vorliegenden Messwerten hat das eine Attribut "Erkennung" zwei Ausprägungen, JA und NEIN. Bei dem anderen Attribut, der Erhöhung der Datensätze bzw. der Objektanzahl, bestehen jedoch mehr Ausprägungen. Unter diesen Voraussetzungen wird der Test "Mehrfeldertest" genannt. Die Daten werden zusammengefasst exemplarisch für die Erhöhung der Datensätze in Tabelle 6.5 dargestellt. In MYSTAT heißt der Mehrfelder-Chi-Quadrat-Test Pearson Chi^2. Benannt nach Karl Pearson, der den Test und seine Teststatistik erstmals in einem Artikel im Philosophical Magazine im Jahre 1990 beschrieb. [Pea00] [LB82] Die Ergebnisse, die MYSTAT liefert, werden in Tabelle 6.6 gezeigt. Auch hier werden durch MYSTAT nur drei Nachkommastellen angezeigt. So kann auch an dieser Stelle davon ausgegangen werden, dass die Werte kleiner als $0,001$ sind, also kleiner als $0,1\%$. Damit kann die Hypothese bestätigt werden, dass mit einer Irrtumswahrscheinlichkeit von kleiner als $0,1\%$ die Erkennungsrate von einer Erhöhung der Datensätze und der Steigerung der Objektanzahl abhängt.

DATENSAETZE	3	6	9	12	15	18	21	Total
JA	40	49	53	59	57	89	60	377
NEIN	20	11	7	1	3	1	0	43
Total	60	60	60	60	60	60	60	420

Tabelle 6.5: Die erhaltenen Erkennungsraten zusammengefasst für den Test der Erkennungsrate mit einer Erhöhung der Datensätze mit insgesamt 20 Messungen für je 3 Objekte. Also insgesamt 60 Werte für die Erkennungsrate pro Messung.

Pearson Chi-Square	Value	df	p-value
DATENSAETZE	57,465	6,000	0,000
OBJEKTE	37,175	5,000	0,000

Tabelle 6.6: Pearson Chi-Square Testergebnisse. Der Wahrscheinlichkeitswert p dient als Irrtumswahrscheinlichkeit für die getesteten Fragestellungen.

Der lineare Zusammenhang ist auch in Abbildung 6.14 am Beispiel der Erhöhung der Datensätze offensichtlich, jedoch soll mit einem Test auf Korrelation noch die Frage geklärt werden, ob der Anstieg der richtigen Erkennung und der Abfall der falsch Erkennung statistisch nachweisbar ist. MYSTAT liefert für die richtige Erkennung einen Korrelationskoeffizienten von $r = 0,892$. Dieser recht hohe Wert zeigt einen Anstieg der Erkennungsrate mit steigender Datensatzanzahl. Um die Zufälligkeit der Daten auszuschließen, wird der p-Wert des Tests betrachtet. Dieser liegt bei $0,007$, was einer Irrtumswahrscheinlichkeit von $0,7\%$ entspricht. Analog beim Test für die Nichterkennung, ergab sich ein p-Wert von $0,048$, also eine Irrtumswahrscheinlichkeit von $4,8\%$. Dieser Wert liegt sehr knapp unter

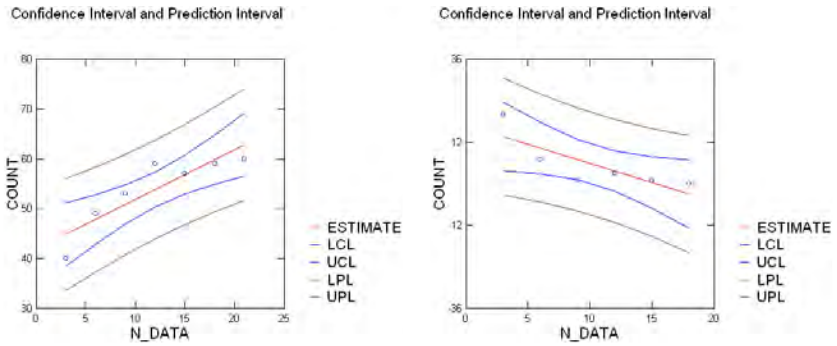


Abbildung 6.14: Die Abbildungen zeigen den Anstieg der Erkennungen (links) und den Abfall der Nichterkennungen (rechts) bei dem Test der Erkennungsrate mit einer Erhöhung der Datensätze.

dem angesetzten Signifikanzniveau, hier kann noch gesagt werden, dass statistisch signifikant nachgewiesen ist, dass eine Erhöhung der Datensätze die Anzahl der Nichterkennung prozentual sinken lässt. Jedoch ist in diesem Fall nur der Test für die richtige Erkennung maßgeblich. So kann mit einer Irrtumswahrscheinlichkeit von $0,7\%$ belegt werden, dass eine Abhängigkeit der zunehmenden richtigen Erkennung mit steigender Anzahl der Datensätze besteht. Dass die Irrtumswahrscheinlichkeit bei dem Test für die Nichterkennung schlechter ist als bei dem Test für die richtige Erkennung, kann mit der insgesamt niedrigen Anzahl der Nichterkennungen im Test zusammenhängen.

6.3.5 Auswirkung des NULL-Objektes

Das NULL-Objekt, wie schon in einem früheren Abschnitt der Arbeit beschrieben, dient der Verhinderung einer Zuordnung eines Objektes an einer falschen Stelle. Wie beschrieben, gibt der Naive-Bayes-Klassifikator immer nur die mögliche Wahrscheinlichkeit für ein angelerntes Objekt zurück. Somit wird immer das Objekt mit der höchsten Wahrscheinlichkeit aller angelernten Objekte als erkannt zurückgegeben. Unter dieser Voraussetzung muss in dem System dem NULL-Objekt bei dem nicht passenden Gegebenheiten eine höhere Wahrscheinlichkeit zugeordnet werden als den restlichen angelernten Objekte, um eine falsche Erkennung abzufangen. Hierfür werden dem NULL-Objekt eine Vielzahl unterschiedlicher Werte zugeordnet und versucht, verschiedene Helligkeitsstufen und Farbwerte mit anzulernen. Es wird darauf geachtet, dass dabei auch unterschiedliche räumliche Ausrichtungen berücksichtigt werden. Weiter werden Teile des Museums mit berücksichtigt, die keinem Exponat zugeordnet werden können. Hierzu gehören leere Wände, der Boden an unterschiedlichen Stellen, Treppenhäuser und Flure sowie Ausschnitte mit für das Museum typischen Materialien. Im Mathematikum sind es die Holzoberfläche von Tischen und Stühlen, die zum Teil auch in den Exponaten Verwendung finden.

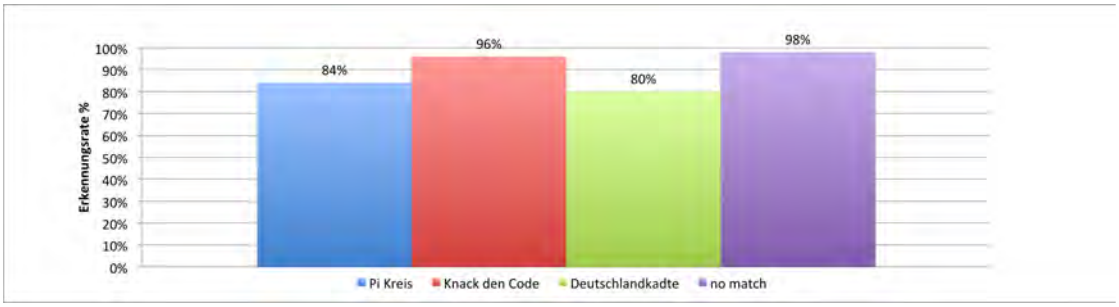


Abbildung 6.15: Auswirkung des NULL-Objektes auf die Erkennungsrate der einzelnen Objekte. Durch das Anlernen des sogenannten NULL-Objektes sinkt im allgemeinen die Erkennungsrate der einzelnen Objekte. Damit das NULL-Objekt immer dann erkannt wird, wenn nicht explizit ein Objekt erkannt werden soll, muss dem NULL-Objekt eine höhere Wahrscheinlichkeit gegeben werden. Dies geschieht durch Anlernen mit Situationen im Mathematikum, die nicht einem Objekt zu geordnet werden.

Für die Anwendung im Mathematikum wird an die angelernten Objekte (vergleiche 6.3.2) ein Trainingsdatensatz für das NULL-Objekt angehängt. Die Trainingsdaten aller drei Objekte ergeben zusammen eine Anzahl von 57 Datensätzen. Für das NULL-Objekt werden insgesamt 125 Trainingsdaten hinterlegt. Anschließend wird mit dem Modell aus den Daten eine neue Messreihe gestartet, in dem die Erkennungsrate der Objekte über 50 Messungen ermittelt wird. Ebenso werden 50 Messungen an einer beliebigen Position im Mathematikum ausgeführt, die nicht den ausgewählten Exponaten zuzuordnen ist, um die Erkennungsrate des NULL-Objektes zu ermitteln. Abbildung 6.15 zeigt, dass alle Erkennungsraten über 80% liegen. Jedoch ist im Vergleich mit Abbildung 6.10 zu erkennen, dass die Erkennungsrate der einzelnen Objekte leicht gesunken ist.

6.3.6 Optimierung durch Anpassung der Trainingsdatenanzahl

Wie in diesem Kapitel schon angesprochen, wird die Hypothese aufgestellt, dass die Erkennungsrate einzelner Objekte durch gezielte Gewichtung der Trainingsdaten noch optimiert werden kann. Natürlich muss dabei beachtet werden, dass dadurch nicht die Erkennungsrate der anderen Objekte beeinflusst wird. Jedoch können die Positionen, bei denen die Erkennung nicht optimal ist, gezielt angelernt werden. Somit werden die einzelnen Werte für die Wahrscheinlichkeitsermittlung gestärkt.

Es ergibt sich somit folgende Schlussfolgerung: Da Exponate von ihren Werten her unterschiedlich dominant sind, ist es nicht ratsam, eine gleichmäßige Verteilung der Trainingsdaten zu erzwingen. Vielmehr sollte darauf geachtet werden, dass die möglichen Positionen, die für eine Erkennung signifikant sind, abgedeckt werden.

6.4 Optimierung bei der Bildverarbeitung

Im Abschnitt 4.6.1 im Kapitel 4 wird auf die unterschiedlichen Möglichkeiten der Bilderfassung eingegangen. Um diese für Testzwecke in der TBOC-Anwendung implementierten Methoden auszuwerten, werden diese in den nachfolgenden Abschnitten in Ihrer Funktionsweise erläutert und über eine Zeitauswertung beurteilt, um die Unterschiede sowie Vor- und Nachteile aufzuzeigen.

6.4.1 onPictureTaken()

Die Methode `onPictureTaken()` wird aufgerufen, wenn über die Kamera die Methode `mCamera.takePicture()` den `PictureCallback` aufruft und wenn die Bilddaten nach dem Auslösen eines Fotos zur Verfügung stehen. Ein Foto mit einem mobilen Gerät aufzunehmen, kostet in der Regel etwas Zeit, da die Kamera natürlich erst fokussieren und das Bild auslösen muss. Des Weiteren ist ein Foto in der Regel sehr hochauflösend, so dass die großen Daten zunächst auch noch reduziert werden müssen. Bei einem *Google Nexus 4* ergaben sich im Durchschnitt 0,19796 Sekunden, die benötigt werden, um ein Bild zu erfassen. Bei 50 Messwerten betrug die minimale Zeit 0,147 Sekunden und die maximale Zeit 0,282 Sekunden. Bei älteren Geräten kann der Wert natürlich höher liegen.

6.4.2 onPreviewFrame()

Über die Kamera Callback-Methode `mCamera.setOneShotPreviewCallback` wird die Methode `onPreviewFrame()` aufgerufen, wenn das Vorschaubild angezeigt wird. In der Methode kann das Bild der Vorschau des Kamera-Streams in eine Bitmap umgewandelt werden. Hierfür vergeht keine Zeit für die Auslösung eines Bildes. Die Bilddaten werden direkt aus dem Video-Stream übertragen. Diese Möglichkeit liefert schnelle Ergebnisse, jedoch kommt es bei *Google Nexus 5* Geräten zu einer Fehlfunktion. Sobald die Methode aufgerufen wird und das Bild ausliest, verdunkelt sich das Vorschaubild und auch das zurückgegebene Bild ist zu dunkel. Dieses Problem scheint ein Fehler im Kamera-Treiber zu sein. In unterschiedlichen Foren wird das Problem erwähnt, jedoch ist zur Zeit noch keine offizielle Stellungnahme von Google aufzufinden. Bei dieser Methode liegt der zeitliche Durchschnitt der 50 Messungen bei 0,10383 Sekunden mit der schnellsten Erzeugung in 0,011 Sekunden und der längsten mit 0,178 Sekunden.

6.4.3 TextureView.getBitmap()

Mit der Verwendung einer `TextureView` als Wiedergabefläche des Kameravorschaubildes kann das Bild der Kamera auch ohne Callback-Methode der Kamera abgefangen werden. Die `TextureView` bietet hierfür die Möglichkeit über die Methode `getBitmap()` das Bild, das auf der `TextureView` angezeigt wird, als Bitmap zurückzugeben. Hierbei hängt die Größe des Bildes von der Skalierung der `TextureView` ab. Bei der Zeitmessung war diese

Methode die schnellste mit einem durchschnittlichen Messewert von 0,01316 Sekunden. Die schnellste Messung betrug sogar nur 0,003 Sekunden. Da im Fall des TBOC-Verfahrens kein großer Wert auf die Bildqualität bzw. auf die Größe der Bilddatei gelegt wird, da diese im System vor der Auswertung sowieso verkleinert wird, ist diese Methode der Bilderfassung optimal für die Anwendung im hier betrachteten Verfahren. Sie liefert schnell Ergebnisse und es ergeben sich auch keine weiteren gerätespezifischen Probleme.

6.5 Anwendertest

Für diese Arbeit wurde auf einen großangelegten Anwendertest verzichtet, da für den Bereich der Objekterkennung keine problematischen Benutzereinwirkungen stattfinden. Auch in den implementierten Anwendungen besteht die Benutzeroberfläche lediglich aus einer Anzeige des aktuellen Kamerabildes. Die Erkennung startet entweder automatisch oder der Benutzer muss den Startvorgang durch Drücken auf das Kamerabild auslösen. Danach muss der Anwender nur noch die Kamera auf das gewünschte Objekt richten, welches er erkennen möchte.

Im Rahmen des Projektes PIMAR wurde die Mathematikum-Anwendung in einem Anwendertest analysiert. Hierbei standen die inhaltliche Umsetzung sowie die Stabilität der Anwendung im Vordergrund. Die Thematik des Anwendertests ist jedoch nicht ausschlaggebend für diese Arbeit. Die Ergebnisse werden zu einem späteren Zeitpunkt in anderen Projekten und studentischen Arbeiten veröffentlicht. Bei der Beobachtung der Probanden kann aber festgehalten werden, dass sie ganz intuitiv bei der Erkennung das Gerät auf das gewünschte Objekt richten, jedoch in den meisten Fällen aus einer beliebigen Position. Wobei der Abstand zu dem Exponat dem angelernten Abstand entsprach. Somit wird bestätigt, dass die Erkennung der einzelnen Exponate nicht nur von der optimalen Position versucht wird. Für die Anwender scheint es selbstverständlich, dass die Anwendung das Objekt aus jeder Position rund um das Exponat erkennen müsste. Dies bestätigt, dass die dynamisch Anpassung unbedingte Voraussetzung für eine optimale Umsetzung des Verfahrens ist.

6.6 Zusammenfassung

Das TBOC-Verfahren wurde unter unterschiedlichen Gesichtspunkten auf Performance untersucht. Ziel der unterschiedlichen Tests war, herauszufinden, wie sich die Trainingsdaten auf das Dateisystem und auf das zu erzeugende Modell auswirken. Weiter sollte ermittelt werden, wie lange ein Erkennungsvorgang unter unterschiedlichen Voraussetzungen dauert und ob Zusammenhänge zu Anzahl der Datensätze oder Anzahl der Objekte bestehen. Eine der wichtigsten Erkenntnisse war hierbei unter anderem auch das Verhalten der Erkennungsrate bei der Objekterkennung.

Die Trainingsdaten-Datei, die für die Erstellung des Modells benötigt wird und die die einzelnen Datensätze für die Objekte beinhaltet, erzeugt keine große Speicherbelastung für das mobile Gerät. Bei 1000 Trainingsdaten ist die Dateigröße mit *0,34mb* relativ überschaubar. Die Anzahl der Objekte hingegen hat keine Auswirkung auf die Dateigröße. Anders sieht es bei dem generierten Modell aus. Hier hat die Anzahl der Datensätze keine Auswirkung auf die Dateigröße. Bei der Erhöhung der Objektanzahl hingegen ist ein linearer Anstieg der Dateigröße mit einer Steigung von $m = 5,58$ zu verzeichnen. So erhält man bei 20 Objekten eine Dateigröße des Modells von *0,123mb*. Somit bleibt die Speicherbelastung des Verfahrens sehr gering. Bei der Untersuchung zum Zeitverhalten für das Anlegen des Modells wurde festgestellt, dass eine Erhöhung der Datensätze eine leichte Erhöhung bei der Erstellungszeit verursacht. Aber der Anstieg von lediglich 0,2545 Sekunden bei einer Erhöhung um 90 Datensätze ist nicht so ausschlaggebend wie der Einfluss bei der Erhöhung der Objektanzahl. Hier benötigt das System 14,21 Sekunden mehr, wenn statt der drei Objekte 50 Objekte in den Trainingsdaten berücksichtigt werden.
Vergleichbar mit der Modellerstellung sind auch die Messzeiten bei der Objekterkennung. Hier stellte sich heraus, dass die Erhöhung der Datensätze sogar eine leichte Verbesserung in der Erkennungszeit verursacht. Es ergibt sich eine lineare negative Steigung von $m = -0,2467$. Wenn die Anzahl der Objekte bei der Erkennung erhöht werden, wirkt sich das ebenfalls wie bei der Erstellungszeit auch auf die Erkennungszeit aus. Hier steigt die Zeit für die Erkennung mit $m = 239,948$. Wobei für drei Objekten mit 100 Datensätzen nur 2,29 Sekunden benötigt werden, erreicht die Messung bei 50 Objekten und 100 Datensätzen eine Zeitspanne von 13,64 Sekunden.

Das Anlernen und das Erkennen von Objekten kann in zwei Varianten unterteilt werden. Hierbei handelt es sich um das statische Anlernen oder Erkennen, wenn das Objekt aus der optimalen Betrachtungsposition, also dem direkten Blick von vorne auf das Objekt angelernt und erkannt wird. Beim dynamischen Anlernen und Erkennen versucht man möglichst alle möglichen Positionen und Blickwinkel auf das Objekt beim Anlernen abzudecken. Hierbei ist zu beachten, dass bei einer 360 Grad Abdeckung der Kompass-Wert keine nennenswerte Rolle mehr in der Erkennung spielt und es so zu Erkennungsfehlern kommen kann. Trotzdem ist es möglich, eine hohe Erkennungsrate zu erzielen. In einem Labortest wurde die Notwendigkeit des dynamisches Anlernens zur Verbesserung der Erkennungsraten bestätigt. Gerade in einem Museumsbetrieb muss damit gerechnet werden, dass die Besucher aus jeder denkbaren Position versuchen, ein Objekt zu erkennen.
Bei der Betrachtung der Erkennungsrate in Bezug auf die Anzahl der Trainingsdaten wurde festgestellt, dass die Erkennungsrate sich durch die Erhöhung der Trainingsdaten grundsätzlich verbessert. Jedoch gibt es immer Objekte die von ihren Daten her dominanter sind als andere. Im Sinne der Optimierung ist es nicht sinnvoll, die Datensatz-Anzahl am schwächsten Objekt auszurichten, sondern mit einer flexiblen Rate zu arbeiten. Es sollten nur so viele Trainingsdaten pro Objekt angelernt werden, wie für eine homogene Erkennung aller Objekte benötigt wird. So können für schwächere Objekte, bei denen es noch zu falschen Erkennungen kommt, zur Stabilisierung zusätzliche Trainingsdaten hinterlegt werden.

Mit dem Wissen, dass dominierende Objekte oder auch ähnliche Objekte die Erkennung anderer Objekte beeinflussen können, ist es nachvollziehbar, dass die Erhöhung der Objektanzahl meist zur Verschlechterung der gesamten Erkennungsrate führt. Wobei es immer Objekte gibt, die mit ihren starken, gut zuordnungsfähigen Werten durchweg eine gute Erkennungsrate aufweisen. Bei 18 Objekten mit jeweils 15 Trainingsdaten hatte kein Objekt eine schlechtere Erkennungsrate als 60%. Aber auch hier würde sich eine Verbesserung einstellen, wenn bei den Objekten mit einer geringeren Erkennungsrate die Anzahl der Datensätze erhöht werden.

Das Prinzip der Einführung eines NULL-Objektes, das zum Abfangen einer falschen Erkennung dient, wurde anhand der angelernten Trainingsdaten für die Mathematikum-Anwendung umgesetzt. Hierbei wurden dem Trainingsdatensatz die Trainingsdaten des NULL-Objektes hinzugefügt. Es wurde darauf geachtet, dass die Trainingsdaten des NULL-Objektes so ausgewählt wurden, dass diese Werte keinem Exponat zugesprochen werden können. Mit diesen ausgewählten Werten muss das NULL-Objekt jedoch immer dann eine höhere Wahrscheinlichkeit erreichen, wenn sonst kein anderes angelerntes Objekt erkannt wird. Das Hinzufügen der Trainingsdaten des NULL-Objektes hat jedoch Auswirkung auf die Erkennungsrate der einzelnen Objekte, da für eine robuste Erkennung des NULL-Objektes eine Vielzahl von Trainingsdaten benötigt wird. Diese Daten können auch Charakteristika enthalten, die denen eines angelernten Objekt ähneln können. Dann wird statt des Objektes das NULL-Objekt erkannt. In einem Testverlauf mit 50 Messungen konnte ein Abfall der einzelnen Erkennungsraten der Objekte von 4% bis 20% verzeichnet werden.

Über einen Ausführungszeiten-Test wurden die unterschiedlichen Methoden zur Bilderfassung untersucht. Es wurde festgestellt, dass die Variante, in dem das Bild von der `TextureView` ausgelesen wird und nicht über eine Kamera-Callback Methode ermittelt wird, einen deutlichen Zeitvorteil bietet. Da im vorliegenden Fall die Bildqualität keine große Rolle spielt, bietet diese Bilderfassung die effektivste Lösung.

Kapitel 7

Zusammenfassung, Ergebnisse und Ausblick

Im Rahmen dieser Arbeit wurde das Verfahren TBOC entwickelt und seine Funktionsweise beschrieben. Das Verfahren ermöglicht eine markerlose Initialisierung zum Laden von komplexen AR-Szenarien ohne Benutzereinwirkung oder weitere Hardware. Wenn reale Objekte als AR-Referenz dienen, wird auf das sogenannte 3D Markerless-Verfahren zurückgegriffen. Bei diesem Verfahren werden Punktwolken mit Tiefeninformationen verwendet. Diese Punktwolken beschreiben markante Punkte auf einem Objekt und deren Position im Raum. Anhand dieser können Objekte oder Räumlichkeiten erkannt und verfolgt werden. Das in dieser Arbeit verwendete AR-SDK bietet zwar die Möglichkeit mehrere Punktwolken zur selben Zeit zu vergleichen und zu erkennen, jedoch führt dies schon bei wenigen Punktwolken zu Leistungseinbrüchen und somit zu niedrigen Frameraten, was zu einem Ruckeln des angezeigten Kamerabildes führt. Um dies zu verhindern, ist es hilfreich, eine Erkennung vor das eigentliche AR-Tracking zu schalten und je nach erkanntem Objekt die entsprechend passende AR-Szene zu laden. Um den Benutzer der Anwendung nicht in den Prozess der Initialisierung einzubinden, in dem er entweder über einen sogenannten QR-Code oder eine Menüstruktur immer das gewünschte AR-Senario starten muss, wurde mit dem TBOC-Verfahren eine Möglichkeit geschaffen, eine markerlose automatische Objekterkennung unabhängig von dem AR-Tracking zu vollziehen. Hierbei kann nach Erkennung eines Objektes neben einer passenden AR-Szene auch jeder beliebige Abschnitt in einer Anwendung geladen werden. Somit können auch andere ortsgebundene Informationen einem Betrachter nach der Erkennung zur Verfügung gestellt werden.
Für die Erkennung und die sogenannte Klassifizierung der Objekte wird mittels der Sensorik der Mobiltelefone und einer Analyse des Kamerabildes mit zwei verschiedenen Bildverarbeitungsalgorithmen ein Datenvektor erstellt, der mittels der "NaiveBayes"-Methode eine Wahrscheinlichkeitsverteilung vornimmt. Für die Klassifikation mittels eines Bayes-Klassifikators ist es notwendig, dass für die zu erkennenden Objekte bekannte Klassifikatoren durch Trainingsdaten angelernt werden. Um ausreißende Werte zu minimieren, werden die Werte in einem Intervall gesammelt und anschließend gemittelt. Aus den mit der Kamera aufgezeichneten Bildern werden Bildmerkmale in Form von Farbwerten extrahiert. Hierfür

wird mit Hilfe einer Methode ein reduziertes Histogramm für jeden Farbkanal erstellt. In dem zweiten Algorithmus wird aus 25 Regionen im Bild der Farbmittelwert bestimmt und in die einzelnen Farbwerte (RGB) zerlegt. In dieser Arbeit wurde ein Entwickler-Tool implementiert, welches das Erstellen von unterschiedlichen Trainingsdatensätzen ermöglicht. Neben dem Anlernen der Objekte gibt es auch die Möglichkeit, diese zu Testzwecken zu erkennen. In der Anwendung wurden Hilfestellungen eingebaut, die das Verwalten von Trainingsdaten und auch das Testverfahren der Erkennung erleichtern. Die Klassen der Erkennung wurden exemplarisch im Rahmen des Forschungsprojektes *PIMAR* in einem Prototypen für das *Mathematikum* in Gießen integriert und dort vor Ort auf seine Praktikabilität getestet. Weiter wurden in dieser Arbeit unterschiedliche Tests für die Funktionalität und Leistung des Verfahrens gemacht. Dabei wurde der Datenspeicherverbrauch sowie die Zeit, die für die Erstellung des Datenmodells benötigt wird, untersucht. Die wichtigsten Erkenntnisse brachte jedoch die Auswertung der Erkennungsraten und der Zeit, die für eine Erkennung benötigt wird.

Das aus dieser Arbeit resultierende Verfahren ermöglicht es, Objekte oder Örtlichkeiten in Innenräumen und Außenbereichen zu erkennen, die dem System vorher bekannt sind. Dies schafft die Voraussetzungen für eine gezielte Initialisierung von AR-Inhalten für das markerlose 3D-Tracking sowie für das Laden jeglichen positionsbezogenen Inhaltes in einer mobilen Anwendung. Die Anforderungen, die durch Anwendungsszenarien des Industriepartners des Projektes PIMAR definiert sind, wurden in der Konzeptentwicklung berücksichtigt. Das TBOC-Verfahren verwendet daher keine visuellen Marker, es ist keine Anschaffung oder Installation von weiterer Hardware wie einer WLAN-Infrastruktur oder die Nutzung von iBeacons etc. von Nöten. Während der Nutzung des Verfahrens in mobilen Anwendungen ist kein Datenaustausch über Mobilfunknetz oder WLAN vorgesehen, da die benötigten Daten direkt in der Anwendung auf dem Gerät des Anwenders hinterlegt sind. Um großen Speicherverbrauch zu vermeiden, werden keine Bilder abgespeichert, sondern neben den Sensorwerten des Magnetometers, des Gyrometers und eventuell des GPS werden die Bilder, die die Gerätekamera aufnimmt, über zwei Bildanalyse-Algorithmen untersucht und lediglich die aus der Analyse gewonnen Werte verwendet. Die gesammelten Daten werden in einen Parameterraum transformiert und im System verarbeitet. Das entwickelte markerlose Verfahren bestimmt die Objekte über eine statistische Wahrscheinlichkeitsverteilung anhand eines Vergleiches der Sensordaten und einem zuvor angelernten Datenmodell. Hierfür wird auf einen Bayes-Klassifikator zurückgegriffen. Somit unterteilt sich das TBOC-Verfahren in zwei Phasen. In der Anlernphase werden die Daten zu einem Objekt erfasst und über eine Trainingsdatendatei (ARFF-Datei) einem Objekt zugeordnet. Aus dieser Trainingsdatendatei wird vor der Erkennung das Datenmodell generiert. Die gesammelten Sensor- und Bilddaten werden in der Erkennungsphase als sogenannter Featurevektor mit diesem Modell verglichen und mit einer bedingte Wahrscheinlichkeit einem der angelernten Objekte zugeordnet.
Das Verfahren wurde mit verschieden Prototypen sowohl unter Laborbedingungen, als auch unter realen Bedingungen in einer Museumsanwendung (Mathematikum in Gießen) getestet. Daraus entstand eine finale Prototyp-Applikation, die die Grundfunktionen des Verfahren in einer Verwaltungsanwendung zusammenführt. Hier ist es möglich, Objekte in Innenräumen und Außenbereichen anzulernen und zu erkennen. Die Programmabschnitte

sind so modular aufgebaut, dass sie leicht in weitere Projekte integriert werden können.
Die Dateigröße der Daten, die auf dem mobilen Gerät gespeichert werden, hängt mit der Anzahl der einzelnen Trainingsdaten und der Anzahl der zu erkennenden Objekte zusammen. Wobei ein Trainingsdatensatz mit 1000 Trainingsdaten gerade mal eine Größe von *0,34mb* einnimmt. Das generierte Modell, mit dem der Vergleich bei der Erkennung vollzogen wird, hat bei einer Anzahl von 20 Objekten und 100 Trainingsdaten eine Größe von *0,123mb*. Somit sind die Daten, die im Regelfall benötigt werden, nur wenige Megabyte groß.
Die Zeit, die für eine Erkennung benötigt wird, liegt bei drei bis 15 Objekten zwischen zwei bis fünf Sekunden. Durch eine Auslagerung der Verarbeitung der Erkennung in einen separaten Thread, ist die Anzeige des Kamerabildes in der Zeit ruckelfrei. Über eine Warte- / Ladeanimation wird dem Benutzer angezeigt, dass die Erkennung läuft. Ab ca. 35 Objekten steigt die Zeit, die für die Erkennung benötigt wird, auf über 10 Sekunden. Die Erkennungsrate bei drei Objekten und ausgeglichenen Trainingsdaten liegt bei 100%. Eine Erhöhung der Objekte, die für die Erkennung einbezogen werden sollen, lässt die allgemeine Erkennungsrate sinken. Jedoch ergab sich bei bis zu zwölf Objekten ein Mittelwert in der Erkennungsrate von 90%. Insgesamt wurde nachgewiesen, dass eine Erkennung von bis zu 18 Objekten mit einer Erkennungsrate von 80% möglich ist. Kleinere Störungen, wie die teilweise Verdeckung der Objekte durch Personen etc. haben keine Auswirkungen auf das System und die Erkennungsrate. Die Störfaktoren, die bei den Erkennungstests herausgefunden wurden, sind große Veränderungen an Form und Position der Exponate sowie starke Veränderungen der Lichtverhältnisse (einfallendes Sonnenlicht, starke Schattenkanten). Diese Störfaktoren können zu Einbrüchen in der Erkennungsrate führen. Durch gezieltes Anlernen können diese Störungen jedoch minimiert werden. Durch Einführung eines NULL-Objektes, welches mit Trainingsdaten so angelernt wird, dass es mit höchster Wahrscheinlichkeit zugeordnet wird, wenn keins der angelernten Objekte mit dem zu erkennenden Objekt übereinstimmt, wird die Möglichkeit gegeben, eine falsche Erkennung abzufangen. Aus den Ergebnissen der unterschiedlichen Tests konnte festgestellt werden, dass eine Verwendung dieses Verfahrens zur Erkennung von Exponaten in einem Museum im alltäglichen Museumsbetrieb funktioniert.

Anhand der Erkenntnisse aus der Integration in die Anwendung des Mitmach-Museums *Mathematikum* und aus den einzelnen Testverfahren kann geschlossen werden, dass das TBOC-Verfahren sowohl in Innenräumen, als auch im Außenbereich genutzt werden kann, um Objekte zu erkennen und in einer mobilen Anwendung virtuelle Inhalte entsprechend zu laden und mittels Augmented Reality direkt auf dem Exponat anzuzeigen. In der Museums-Anwendung konnte anhand des sogenannte Minispieles *Knack den Code*, welches beim Erkennen des Exponates in der Anwendung geladen wird, gezeigt werden, dass sich das Verfahren auch für das Laden interner Programmabschnitte eignet. So kann ein beliebiger Inhalt (Texte, Bild, Videos, Spiele etc) in der eigentlichen Anwendung ortsgebunden angezeigt werden.

7.1 Ausblick

In dieser Arbeit konnte gezeigt werden, dass das entwickelte Verfahren zur markerlosen Erkennung von angelernten Objekten funktioniert. Es wurden Schwächen in der Erkennungszeit und bei den Erkennungsraten erkannt, die entstehen, wenn die Anzahl der angelernten Objekte erhöht wird. Hier könnte die Erkennung gerade bei einer sehr hohen Anzahl von Objekten verbessert werden, indem getestet wird, ob eine Gewichtung der unterschiedlichen Featurevektor-Werte hilfreich ist. Beim derzeitigen Stand der Implementierung stehen sehr viele Farbwerte, die aus den Bildverarbeitungsanalyse ermittelt werden, einer Handvoll Sensorwerte gegenüber, was zur Folge hat, dass in erster Linie die Farbinformationen zur Klassifikation genutzt werden. In Zukunft könnte eine höhere Gewichtung der Werte aus Gyrosensor und Magnetometer sowie in Außenbereichen der GPS-Werte getestet werden, um so auch bei zu starken Ähnlichkeiten der verschiedenen Objekte in der Farbgebung eine bessere Erkennungsrate zu erzielen. Bei der Verwendung einer *XRFF*-Datei[1] (eXtensible attribute-Relation File Format) für die Trainingsdaten kann ausschließlich bei der Verwendung des *Naive Bayes* eine Gewichtung einzelner Attributwerte vorgenommen werden. Das auf XML basierende ARFF-Format ermöglicht es über Metadaten, Attributen Eigenschaften zuzuteilen. So kann im Kopfbereich der Datei den Attributen eine Gewichtung von 0 - 1 zugewiesen werden. Eine weitere Möglichkeit, die Trainingsdaten zu beeinflussen, ist die Gewichtung einzelner Featurevektoren beim Anlernen in der *XRFF*-Datei. So können gezielten Datensätzen eine höhere Bedeutung im Modell zugeordnet werden. Dies ermöglicht es, initial Positionen stärker zu werten oder die Gewichtung einzelner NULL-Objekt-Instanzen schwächer zu behandeln. Die Idee und die funktionelle Umsetzung des NULL-Objektes konnte bestätigt werden, jedoch gibt es hier noch Verbesserungspotential, was das Anlernen und die Anzahl der Daten betrifft. Das Entwicklertool in Form der TBOC-Anwendung ist modular gehalten, jedoch wäre hier eine verbesserte Schnittstelle zu anderen Anwendungsprojekten möglich. Somit bietet der Ansatz dieser Arbeit noch ein großes Forschungsfeld, in dem zu klären ist wie das Verfahren ausgebaut und verbessert werden kann.

[1] http://weka.wikispaces.com/XRFF Stand: 23. März 2015

Anhang A

TrackingData_MarkerlessFast.xml

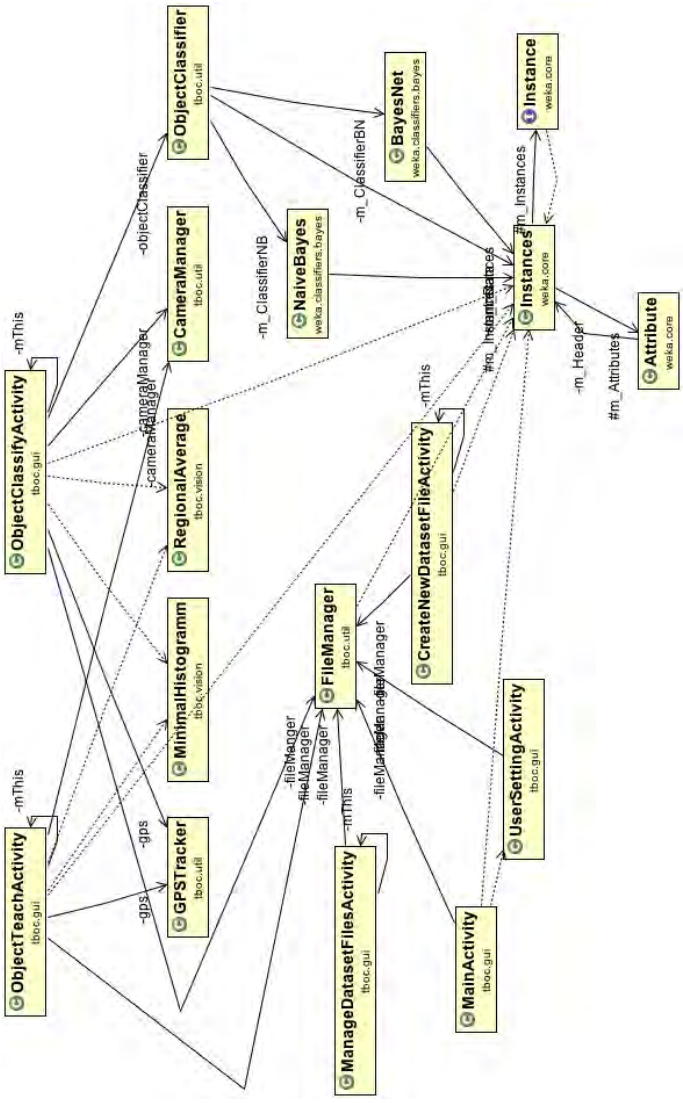
Beispiel für die Konfiguration der Tracking-Komponente des SDKs (siehe Abschnitt 4.3).

```
<?xml version="1.0"?>
<TrackingData>
   <Sensors>
      <Sensor Type="FeatureBasedSensorSource" Subtype="Fast">
         <SensorID>FeatureTracking1</SensorID>
            <Parameters>
                <FeatureDescriptorAlignment>regular</FeatureDescriptorAlignment>
                 <MaxObjectsToDetectPerFrame>5</MaxObjectsToDetectPerFrame>
                 <MaxObjectsToTrackInParallel>1</MaxObjectsToTrackInParallel>
                 <SimilarityThreshold>0.7</SimilarityThreshold>
            </Parameters>
            <SensorCOS>
               <SensorCosID>Patch1</SensorCosID>
               <Parameters>
                   <ReferenceImage>metaioman_target.png</ReferenceImage>
                   <SimilarityThreshold>0.7</SimilarityThreshold>
               </Parameters>
            </SensorCOS>
         </Sensor>
      </Sensors>
      <Connections>
         <COS>
            <Name>MarkerlessCOS1</Name>
            <Fuser Type="SmoothingFuser">
               <Parameters>
                   <KeepPoseForNumberOfFrames>2</KeepPoseForNumberOfFrames>
                   <GravityAssistance></GravityAssistance>
                   <AlphaTranslation>0.8</AlphaTranslation
                   <GammaTranslation>0.8</GammaTranslation>
                  <AlphaRotation>0.5</AlphaRotation>
                  <GammaRotation>0.5</GammaRotation>
                  <ContinueLostTrackingWithOrientationSensor>false</
                       ContinueLostTrackingWithOrientationSensor>
                </Parameters>
            </Fuser>
            <SensorSource>
                <SensorID>FeatureTracking1</SensorID>
                <SensorCosID>Patch1</SensorCosID>
                <HandEyeCalibration>
                   <TranslationOffset><X>0</X><Y>0</Y><Z>0</Z></TranslationOffset>
                    <RotationOffset><X>0</X><Y>0</Y><Z>0</Z><W>1</W></RotationOffset>
                </HandEyeCalibration>
                <COSOffset>
                   <TranslationOffset><X>0</X><Y>0</Y><Z>0</Z></TranslationOffset>
                   <RotationOffset><X>0</X><Y>0</Y><Z>0</Z><W>1</W></RotationOffset>
                </COSOffset>
             </SensorSource>
          </COS>
      </Connections>
</TrackingData>
```

Anhang B

TBOC-Klassendiagramm

Übersicht über die Klassenstruktur der TBOC-Anwendung.



Glossar

ActionBar	steht bei Android für eine Aktionsleiste, die es ermöglicht für den Benutzer Aktions- und Navigationsmöglichkeiten zur Verfügung zu stellen.
Activity	repräsentiert bei Android genau ein sichtbares Benutzerinterface. Beispielsweise kann eine Activity einen Bildschirm mit ein Auswahlmenü anzeigen und die evtl. damit verbundenen Aktionen der Applikation verwalten.
Array	Ein Feld (engl.: Array) steht in der Informatik für eine Datenstruktur, die eine Sammlung von vielen gleichartig strukturierten Daten beinhalten kann.
Augmented Reality	(kurz: AR) im Deutschen als Erweiterte Realität bezeichnet, beschreibt die computergestützte Erweiterung der Realitätswahrnehmung.
Bayes-Klassifikator	ist ein aus dem Satz von Bayes hergeleiteter Klassifikator. Dieser ordnet jedes Objekt der Klasse zu, zu der es mit der größten Wahrscheinlichkeit gehört, bzw. bei der durch die Einordnung die wenigsten Kosten entstehen.
Beacon	(Der Markenname iBeacon) ist ein 2013 von Apple Inc. eingeführter, proprietärer Standard für Navigation in geschlossenen Räumen, basierend auf Bluetooth Low Energy (BLE). iBeacon basiert auf einem Sender-Empfänger-Prinzip. Dazu werden im Raum kleine Sender (Beacons) als Signalgeber platziert
BLE	(engl.: Bluetooth Low Energy) ist eine Funktechnik aus dem Bluetooth Standard 4.0 mit der sich Geräte in einer Umgebung von bis zu 50 Metern mit einer wireless bidirektionale Datenübertragung verbinden können
Callback	im Deutschen Rückruffunktion genannt, bezeichnet in der Informatik eine Funktion, die einer anderen Funktion als Parameter übergeben und von dieser unter gewissen Bedingungen aufgerufen wird.

Data-Mining	Unter dem Begriff versteht man die systematische Anwendung statistischer Methoden auf einen Datenbestand. Hierbei geht es um die Gewinnung von Wissen (in der Form von Mustern).
FBX	(engl.: Filmbox) ist ein proprietäres Dateiformat (.fbx) was heute, als allgemeines Austauschformat, in den meisten 3D-Anwendungen benutzt wird.
Feature-Vektor	auch Merkmalsvektor genannt, fasst die (numerisch) parametrisierbaren Eigenschaften eines Musters in vektorieller Weise zusammen. Feature-Vektoren erleichtern eine automatische Klassifikation, da sie die zu klassifizierenden Eigenschaften stark reduzieren.
Frame	(auf Deutsch: Einzelbild, im technischen Sinne Laufbild) ist in der Filmproduktion, Videoproduktion, Animation und verwandten Bereichen eines der vielen Standbilder, die zusammen das vollständig bewegte Bild ergeben.
GPS	(engl.: Global Positioning System) ist ein globales Navigationssatellitensystem zur Positionsbestimmung
GUI	ist die Abkürzung für Grafische Benutzeroberfläche (engl.: graphical user interface) und ist eine Bezeichnung für eine Form einer Benutzerschnittstelle eines Computers. Sie hat die Aufgabe, Anwendungssoftware auf einem Rechner mittels grafischer Symbole oder Steuerelementen, bedienbar zu machen.
Gyrometer	Hier die Bezeichnung für einen Sensor in mobilen Telefonen. Gyrometer wird auch als Kreiselinstrument bezeichnet und dient der genauen Lagebestimmung.
ImageView	ist eine Android View-Element das für das Anzeigen von Bildern benutzt wird.
Key-Frame Animation	Schlüsselbild und Keyframe sind Begriffe aus der Animationstechnik. Die Schlüsselbildanimation, auch Keyframe Animation, ist eine Animationstechnik, die ursprünglich aus der Produktion von Zeichentrickfilmen stammt. Die Schlüsselbilder geben grob den Bewegungsablauf vor, der in der Folge durch Zwischenbilder verfeinert wird.
Klasse	unter Java, ist es das wichtigste Sprachelement und beschreibt einen komplexen Datentypen. Klassen beinhalten Attribute und Methoden.

Konstante	steht für einen Behälter für eine Größe in einem Computerprogramm, die nach der Zuweisung nicht verändert werden kann.
LBS	(engl.: Location-based Services) steht für Standortbezogene Dienste. Damit werden mobile Dienste beschrieben, die unter Zuhilfenahme von positionsabhängigen Daten dem Endbenutzer selektive ortsgebundene Informationen bereitstellen
List	die verkettete Liste ist in der Informatik eine dynamische Datenstruktur, die eine Speicherung von miteinander in Beziehung stehenden Objekten erlaubt. Die Anzahl der Objekte ist im Vorhinein nicht bestimmt.
LLA-Marker	ist die Abkürzung für Lattitude, Longitude, Altitude, also in deutsch Längengrad, Breitengrad und Höhe über dem Meeresspiegel.
Magnetometer	ist ein Sensor in mobilen Telefonen zur Messung magnetischer Flussdichten. Mit Hilfe des Magnetometer kann die Ausrichtung des Gerätes nach Himmelsrichtungen (Kompass) ermittelt werden.
Marker	hier AR-Marker. Ist in der Regel ein zweidimensionales Symbole oder ein Bild welches einer Kamera erlaubt die Position und Rotation relativ zu einer Oberfläche zu bestimmen.
Methode	ist ein Bestandteil einer Klasse und kann auch nur in einer Klasse deklariert und implementiert werden. Methoden sind Interaktionsmittel von Objekten.
Motion Path	(auf Deutsch: Bewegungspfad) wird in der 3D-Animation eine Kurve genannt, die als Grundlage der Animation verwendet wird. Hier können 3D-Objekte entlang dieser Kurve über eine bestimmte Zeit bewegt werden.
Motion-Capturing	ist der Name einer Technik, die es ermöglicht, Bewegungen so aufzuzeichnen und in ein von Computern lesbares Format umzuwandeln, dass diese Bewegungsdaten zum einen analysiert und zum anderen auf im Computer generierte 3D-Modelle übertragen werden können.
NFC	(engl.: Near Field Communication) ist ein internationaler Übertragungsstandard zum kontaktlosen Austausch von Daten per Funktechnik über kurze Strecken von wenigen Zentimetern.

Pitch	(auf Deutsch: Nicken) Roll-Pitch-Yaw Winkel sind eine Möglichkeit zur Beschreibung der Orientierung eines Objektes im dreidimensionalen Raum. Wobei mit Pitch die Drehung um die y-Achse des Referenzsystems bezeichnet wird.
POI	(engl.: Point of Interest) Ist ein Ort von Interesse. Bei AR-Anwendungen können POI's mit Hilfe von GPS-Nutzung in Form von Billboards im Display angezeigt werden.
QR-Code	(engl.: Quick Response, "schnelle Antwort") Ist eine Methode um Informationen so aufzuschreiben, dass sie von Maschinen schnell gefunden und eingegeben werden können.
Quellcode	Als Quelltext, auch Quellcode (engl.: source code), wird in der Informatik der für Menschen lesbare, in einer Programmiersprache geschriebene Text eines Computerprogrammes bezeichnet .
RFID	(engl.: radio-frequency identification) ist eine Technologie für Sender-Empfänger-Systeme zum automatischen und berührungslosen Identifizieren und Lokalisieren von Objekten mit Radiowellen.
Roll	(auf Deutsch: Rollen) Roll-Pitch-Yaw Winkel sind eine Möglichkeit zur Beschreibung der Orientierung eines Objektes im dreidimensionalen Raum. Wobei mit Roll die Drehung um die x-Achse des Referenzsystems bezeichnet wird.
Spinner	ist ein Android View-Element, das einen schnellen Weg bietet, um einen Wert aus einer Menge auszuwählen. Vergleichbar mit einer Drop-down Liste.
TextView	ist ein Android View-Element, das für das Anzeigen von einfachem Text gedacht ist.
Thread	steht in der Informatik für einen Ausführungsstrang oder eine Ausführungsreihenfolge in der Abarbeitung eines Programms, dieser läuft parallel neben der Hauptanwendung.
Toast	Ein Toast ist unter Android ein passives Popup, welches eine einfache Rückmeldung dem Benutzer geben kann, ohne dass er darauf reagieren muss. Die Einblendung des Toast verschwindet nach einer definierten Zeit automatisch wieder.

Tracking	beschriebt die bildbasierte und sensorbasierte Positions- und Rotation-Bestimmung des Anwenders. Erst durch Tracking wird es ermöglicht, passgenaue Überlagerungen über das Videomaterial in Echtzeit einzublenden.
Walk-Cycle	(auf Deutsch: Lauf-Zyklus) ist in der Animation eine Reihe von Frames oder Abbildungen die in einer wiederholten Abfolge, eine Animation eines laufenden Charakter ergeben. Der Walk-Cycle wird immer und immer wieder wiederholt, so dass nicht jeder Schritt einzeln animiert werden muss.
XML	Ist die Abkürzung für Extensible Markup Language (dt.: erweiterbare Auszeichnungssprache) und ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien.
Yaw	(auf Deutsch: Gieren) Roll-Pitch-Yaw Winkel sind eine Möglichkeit zur Beschreibung der Orientierung eines Objektes im dreidimensionalen Raum. Wobei mit Yaw die Drehung um die z-Achse des Referenzsystems bezeichnet wird.
YUV-Farbmodell	Ist ein Farbmodell der Videotechnik. Es verwendet zur Darstellung der Farbinformationen zwei Komponenten, die Luminanz (Lichtstärke pro Fläche, luma) Y und die Chrominanz (Farbanteil, chroma), wobei diese wiederum aus den zwei Unterkomponenten U und V besteht.

Literaturverzeichnis

[Bay63] Bayes, T.: An essay towards solving a problem in the doctrine of chances. In: *Phil. Trans. of the Royal Soc. of London* 53 (1763), S. 370–418

[CCBA07] Ciurana, Marc ; Cugno, Sebastiano ; Barceló-Arroyo, Francisco: WLAN indoor positioning based on TOA with two reference points. In: *WPNC*, IEEE, 2007. – ISBN 1–4244–0870–9, 23-28

[CFZ09] Clarke, Bertrand ; Fokoue, Ernest ; Zhang, Hao H.: *Principles and Theory for Data Mining and Machine Learning*. 1st. Springer Publishing Company, Incorporated, 2009. – ISBN 0387981349, 9780387981345

[CG11] Christian Grefrath, Hagen Schmidt-Bleker Ralf F. Ralf Frombach: DIB: Dienstleistungen im industriellen Bauprozess - Mit "Augmented Reality" in die Zukunft. In: *UdZ - Unternehmen der Zukunft FIR-Zeitschrift für Betriebsorganisation und Unternehmensentwicklung* 3/2011 (2011), S. 33 – 35. – ISSN 1439-2585

[CL05] Christine Ludwig, Christian R.: *Augmented Reality: Information at Focus*. The Icfai University Press, India: Virtual Reality - Concepts and Applications, 2005. – ISSN 1619-7879

[CL08] Christine Ludwig, Christian R.: *Augmented Reality: Information at Focus*. The Icfai University Press, India: Virtual Reality - Concepts and Applications, 2008

[Cox46] Cox, R. T.: Probability, Frequency and Reasonable Expectation. In: *American Journal of Physics* 14 (1946), Nr. 1, 1-13. http://dx.doi.org/http://dx.doi.org/10.1119/1.1990764. – DOI http://dx.doi.org/10.1119/1.1990764

[DC11] Darcey, L. ; Conder, S.: *Sams Teach Yourself Android Application Development in 24 Hours*. Pearson Education, 2011 (Sams Teach Yourself). http://books.google.de/books?id=ZD4uzpB-Fl0C. – ISBN 9780132786881

[EJ10] Egger, Roman ; Jooss, Mario: Die Zukunft im mTourism ? Ausblick auf Technologie- und Dienstentwicklung. Version: 2010. http://dx.doi.org/10.1007/978-3-8349-8694-8_1. In: Egger, Roman (Hrsg.) ; Jooss, Mario (Hrsg.): *mTourism*. Gabler, 2010. – ISBN 978–3–8349–2362–2, 11-25

[Erm12] Ermolin, P.: *Grundlagen zu Farbräumen*. GRIN Verlag, 2012 http://books.google.de/books?id=zpbJM6BEPf8C. – ISBN 9783656232667

[Fin06] Finkenzeller, Klaus ; Finkenzeller, Klaus (Hrsg.): *RFID-Handbuch, Grundlagen und praktische Anwendungen induktiver Funkanlagen, Transponder und kontaktloser Chipkarten*. 4. Hanser, 2006 (3-446-40398-1)

[Gar11] Gargenta, M.: *Einführung in die Android-Entwicklung*. O'Reilly, 2011 https://books.google.de/books?id=34S3Jt1ONTkC. – ISBN 9783868991147

[Gas14] Gast, Matthew S.: *Building Applications with IBeacon: Proximity and Location Services with Bluetooth Low Energy*. O'Reilly Media, Inc., 2014

[GH07] Gillert, F. ; Hansen, W.R.: *RFID für die Optimierung von Geschäftsprozessen: Prozess-Strukturen, IT-Architekturen, RFID-Infrastruktur*. Hanser, 2007 http://books.google.de/books?id=ntWEuQ5dxPEC. – ISBN 9783446405073

[HFH+09] Hall, Mark ; Frank, Eibe ; Holmes, Geoffrey ; Pfahringer, Bernhard ; Reutemann, Peter ; Witten, Ian H.: The WEKA Data Mining Software: An Update. In: *SIGKDD Explor. Newsl.* 11 (2009), November, Nr. 1, 10–18. http://dx.doi.org/10.1145/1656274.1656278. – DOI 10.1145/1656274.1656278. – ISSN 1931–0145

[HK00] Han, Jiawei ; Kamber, Micheline: *Data Mining: Concepts and Techniques*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 2000. – ISBN 1–55860–489–8

[HSB09] Henze, Niels ; Schinke, Torben ; Boll, Susanne: What is That? Object Recognition from Natural Features on a Mobile Phone. In: *Proceedings of the Workshop on Mobile Interaction with the Real World*, 2009

[HWB00] Hightower, Jeffrey ; Want, Roy ; Borriello, Gaetano: SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength / University of Washington, Department of Computer Science and Engineering. Seattle, WA, February 2000 (00-02-02). – UW CSE

[Jay95] Jaynes, E. T.: *Probability Theory: The Logic of Science*. 1995 http://www.cs.toronto.edu/~nernst/papers/jaynes-book.pdf

[JZB14] Jonuschat, Helga ; Zinke, Michaela ; Bock, Benno: Die Nutzersicht: Akzeptanzfaktoren und Integration ins Post-Processing. Version: 2014. http://dx.doi.org/10.1007/978-3-658-01848-1_5. In: Schelewsky, Marc (Hrsg.) ; Jonuschat, Helga (Hrsg.) ; Bock, Benno (Hrsg.) ; Stephan, Korinna (Hrsg.): *Smartphones unterstützen die Mobilitätsforschung*. Springer Fachmedien Wiesbaden, 2014. – ISBN 978–3–658–01847–4, 83-101

[Kal11] Kallenberg, Philipp: Programmierschnittstellen von Augmented-Reality-Browsern im Rahmen des Projekts ?Augmenture ? In: *Informatics Inside: Grenzen überwinden–Virtualität erweitert Realität* (2011), S. 6

[Ker07] Kern, Christian: *Anwendung von RFID-Systemen*. 2., verb. Aufl. Berlin [u.a.] : Springer, 2007 (VDI-[Buch]). http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+520679199&sourceid=fbw_bibsonomy. – ISBN 978–3–540–44477–0

[KM09] Klein, Georg ; Murray, David: Parallel Tracking and Mapping on a Camera Phone. In: *Proc. Eigth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'09)*. Orlando, October 2009

[Koc13] Koch, K.R.: *Einführung in die Bayes-Statistik*. Springer Berlin Heidelberg, 2013 https://books.google.de/books?id=2Jf3BQAAQBAJ. – ISBN 9783642569708

[Kud14] Kudak, Andreas: *Evaluation und Anwendung aktueller Entwicklungen im Bereich Bluetooth Low Energy am Beispiel von iBeacon*, Diplomarbeit, 2014

[Kup05] Kupper, Axel: *Location-based Services: Fundamentals and Operation*. John Wiley & Sons, 2005. – ISBN 0470092319

[LB82] Linder, Arthur ; Berchtold, Willi: *Statistische Methoden*. Birkhäuser, 1982

[LCTY12] Liu, Pengfei ; Chen, Yanhua ; Tang, Wulei ; Yue, Qiang: Mobile WEKA as Data Mining Tool on Android. Version: 2012. http://dx.doi.org/10.1007/978-3-642-27951-5_11. In: Xie, Anne (Hrsg.) ; Huang, Xiong (Hrsg.): *Advances in Electrical Engineering and Automation* Bd. 139. Springer Berlin Heidelberg, 2012. – ISBN 978–3–642–27950–8, 75-80

[LDBL07] Liu, Hui ; Darabi, H. ; Banerjee, P. ; Liu, Jing: Survey of Wireless Indoor Positioning Techniques and Systems. In: *Trans. Sys. Man Cyber Part C* 37 (2007), November, Nr. 6, 1067–1080. http://dx.doi.org/10.1109/TSMCC.2007.905750. – DOI 10.1109/TSMCC.2007.905750. – ISSN 1094–6977

[LR10] Langer, Josef ; Roland, Michael: NFC-Technologie. Version: 2010. http://dx.doi.org/10.1007/978-3-642-05497-6_5. In: *Anwendungen und Technik von Near Field Communication (NFC)*. Springer Berlin Heidelberg, 2010. – ISBN 978–3–642–05496–9, 87-108

[LSH10] Li, Yunpeng ; Snavely, Noah ; Huttenlocher, Daniel P.: Location Recognition Using Prioritized Feature Matching. In: *ECCV (2)* Bd. 6312, Springer, 2010 (Lecture Notes in Computer Science). – ISBN 978–3–642–15551–2, S. 791–804

[MBS14] Mehler-Bicher, A. ; Steiger, L.: *Augmented Reality: Theorie und Praxis*. De Gruyter, 2014 http://books.google.se/books?id=BrjoBQAAQBAJ. – ISBN 9783110375015

[Mit97] Mitchell, Thomas M.: *Machine Learning*. 1. New York, NY, USA : McGraw-Hill, Inc., 1997. – ISBN 0070428077, 9780070428072

[MS05] Mikolajczyk, K. ; Schmid, C.: A performance evaluation of local descriptors. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 27 (2005), Oct, Nr. 10, S. 1615–1630. http://dx.doi.org/10.1109/TPAMI.2005.188. – DOI 10.1109/TPAMI.2005.188. – ISSN 0162–8828

[MS14] Meier, R. ; Schmidt, J.: *Android App-Entwicklung: Die Gebrauchsanleitung für Programmierer*. Wiley, 2014 https://books.google.de/books?id=kZNPAwAAQBAJ. – ISBN 9783527681402

[MTUK95] Milgram, Paul ; Takemura, Haruo ; Utsumi, Akira ; Kishino, Fumio: *Augmented reality: a class of displays on the reality-virtuality continuum*. http://dx.doi.org/10.1117/12.197321. Version: 1995

[NLLP03] Ni, L.M. ; Liu, Yunhao ; Lau, Yiu C. ; Patil, A.P.: LANDMARC: indoor location sensing using active RFID. In: *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on*, 2003, S. 407–415

[Oeh04] Oehme, Olaf: *Ergonomische Untersuchung von kopfbasierten Displays für Anwendungen der erweiterten Realität in Produktion und Service.*, RWTH Aachen University, Diss., 2004. – 1–176 S.

[Pea00] Pearson, Karl: On the Criterion that a given System of Deviations from the Probable in the Case of a Correlated System of Variables is such that can be reasonably supposed to have arisen from Random Sampling. In: *Philosophical Magazine* 50 (1900), 157–175. http://www.economics.soton.ac.uk/staff/aldrich/New%20Folder/kpreader1.htm

[Pet05] Petersohn, Helge: *Data Mining*. Oldenbourg Verlag, 2005

[PSM13] Pfleging, Bastian ; Schmidt, Albrecht ; Michahelles, Florian: Ubiquitous Connectivity in the Mountains: Enhancing the Ski Experience. In: *Pervasive Computing, IEEE* 12 (2013), April, Nr. 2, S. 5–9. – ISSN 1536–1268

[RC04] Robertson, Duncan ; Cipolla, Roberto: An Image-Based System for Urban Navigation. In: *Proceedings of the British Machine Vision Conference*, BMVA Press, 2004, S. 819–828

[RD07] Reitmayr, Gerhard ; Drummond, T.W.: Initialisation for Visual Tracking in Urban Environments. In: *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, 2007, S. 161–172

[Sac84] Sachs, L.: *Angewandte Statistik: Anwendung statistischer Methoden*. Springer, 1984 http://books.google.de/books?id=h5dPQAAACAAJ. – ISBN 9783540128007

[Sac13] Sachs, L.: *Statistische Methoden: Ein Soforthelfer Für Interessierte in Naturwissenschaft, Medizin, Technik, Wirtschaft, Psychologie und Soziologie*. Springer Berlin Heidelberg, 2013 https://books.google.de/books?id=xHWrBgAAQBAJ. – ISBN 9783642963537

[SBS07] Schindler, G. ; Brown, M. ; Szeliski, R.: City-Scale Location Recognition. In: *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, 2007. – ISSN 1063–6919, S. 1–7

[Sch09] Scholze, Jan: *Entwicklung eines integrierten Ansatzes zur Indoor- und Outdoor-Positionsbestimmung auf Basis einer Föderation von Gebäudedatenservern*, Technische Universität Dresden, Diplomarbeit, 2009

[STK05] Sayed, A. H. ; Tarighat, A. ; Khajehnouri, N.: Network-based wireless location: challenges faced in developing techniques for accurate wireless location information. In: *IEEE Signal Processing Magazine* 22 (2005), Juli, Nr. 4, 24–40. http://dx.doi.org/10.1109/msp.2005.1458275. – DOI 10.1109/msp.2005.1458275. – ISSN 1053–5888

[STT09] Siira, E. ; Tuikka, T. ; Tormanen, V.: Location-Based Mobile Wiki Using NFC Tag Infrastructure. In: *Near Field Communication, 2009. NFC '09. First International Workshop on*, 2009, S. 56–60

[Thi11] Thiersch, Bernhard: Entwurfsentscheidungen zur AugmentedReality-Schnitzeljagd Augmenture. In: *Informatics Inside: Grenzen überwinden - Virtualität erweitert Realität*, Hochschule Reutlingen, 2011, S. 17–21

[VC14] Venzke-Caprarese, Sven: Standortlokalisierung und personalisierte Nutzeransprache mittels Bluetooth Low Energy Beacons. In: *Datenschutz und Datensicherheit - DuD* 38 (2014), Nr. 12, 839-844. http://dx.doi.org/10.1007/s11623-014-0329-9. – DOI 10.1007/s11623–014–0329–9. – ISSN 1614–0702

[WBG96] Wilkinson, Leland ; Blank, Grant ; Gruber, Christian: *Desktop Data Analysis SYSTAT*. 1st. Upper Saddle River, NJ, USA : Prentice Hall PTR, 1996. – ISBN 0135693101

[WF05] Witten, I.H. ; Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Elsevier Science, 2005 (The Morgan Kaufmann Series in Data Management Systems). – ISBN 9780080477022

[WKRQ+07] Wu, Xindong ; Kumar, Vipin ; Ross Quinlan, J. ; Ghosh, Joydeep ; Yang, Qiang ; Motoda, Hiroshi ; McLachlan, Geoffrey J. ; Ng, Angus ;

Liu, Bing ; Yu, Philip S. ; Zhou, Zhi-Hua ; Steinbach, Michael ; Hand, David J. ; Steinberg, Dan: Top 10 Algorithms in Data Mining. In: *Knowl. Inf. Syst.* 14 (2007), Dezember, Nr. 1, S. 1–37. http://dx.doi.org/10.1007/s10115-007-0114-2. – DOI 10.1007/s10115–007–0114–2. – ISSN 0219–1377

[WRM+08] Wagner, Daniel ; Reitmayr, Gerhard ; Mulloni, Alessandro ; Drummond, Tom ; Schmalstieg, Dieter: Pose Tracking from Natural Features on Mobile Phones. In: *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. Washington, DC, USA : IEEE Computer Society, 2008 (ISMAR '08). – ISBN 978–1–4244–2840–3, 125–134

[WS03] Weidenhausen, J. ; Stricker, D.: *ARVIKA - Augmented Reality in Entwicklung, Produktion und Service: Entwicklung einer Augmented Reality Software Plattform ; Schlussbericht zum Projekt ; [Laufzeit: 01.07.1999 bis 30.06.2003]*. Fraunhofer-Inst. Graphische Datenverarbeitung, 2003 http://books.google.se/books?id=sTnZPgAACAAJ