Ein vernetztes Android-Framework für Sensordaten

Masterarbeit

von

Christoph Claßen

aus

Heinsberg

vorgelegt am

Lehrstuhl für Technik sozialer Netzwerke
Jun.-Prof. Dr. Kalman Graffi
Heinrich-Heine-Universität Düsseldorf

August 2016

Betreuer:
Raphael Bialon, M. Sc.

Zusammenfassung

Im Rahmen dieser Arbeit wurde ein Android Sensoren Framework erstellt, welches künftig für verschiedene Feldversuche angepasst werden kann. Insbesondere können Daten, die vom Android Betriebssystem nicht als Sensordaten betrachtet werden, wie zum Beispiel die Global Positioning System (GPS) Daten, nun als neue Sensoren definiert und im Framework verwendet werden. Als praktischen Anwendungsfall für das Framework dient eine Nebeldichtemessung. Diese Messungen sollen nach Möglichkeit für weitere hydrogeologische Projekte im Bereich des semiariden Regenwaldes im Raum Salalah im Oman Verwendung finden.

Ein großer Teil der Arbeit war es, geeignete Messmethoden zu finden, mit Hilfe derer man mit Smartphones die Nebeldichte messen kann. Die Herausforderung hierbei war zudem, dass die vielen unterschiedlichen Smartphones, die alle das gleiche Betriebssystem verwenden, alle verschiedene Hardwarekonfigurationen haben. Darüber hinaus können die Gerätehersteller selbst festlegen, welche physikalischen Sensoren die Daten für die vom Betriebssystem bereitgestellten Basissensoren liefern. Als Vorversuche zur Reise in den Oman wurden verschiedene Möglichkeiten ausprobiert, wie Nebel simuliert werden kann, um die Anwendung zu testen. Versuche mit Rauchpatronen sind nicht erfolgreich verlaufen, währenddessen halbtransparente Folien gute Ergebnisse erzielten.

Obwohl vor der Reise intensiv ausprobiert wurde, wie die Messungen am besten durchgeführt werden, stellte sich bei der Auswertung der Daten heraus, dass eine Unterscheidung der Nebeldichten offensichtlich nicht möglich ist. Selbst eine Unterscheidung zwischen klarer Sicht und dichtem Nebel ist nur begrenzt möglich.

Danksagung

Viele Personen haben mich bei meiner Arbeit unterstützt und ich möchte ihnen hiermit danken.

Vor allem möchte ich meiner Familie für die fortwährende Unterstützung während der Ausarbeitung, insbesondere aber auch für die wertvollen Korrekturlesungen danken.

Natürlich gehört auch ein großer Dank an Raphael Bialon für die wöchentlich interessanten Gespräche, die locker, aber auch immer zielgerichtet verliefen. Darüber hinaus ist das überdurchschnittliche Engagement, welches insbesondere während der Feldarbeit im Nebel erwähnenswert. Trotz der feucht-kühlen Bedingungen konnte man sich immer auf seine Mithilfe bei den Messungen verlassen.

Zudem möchte ich Jan Friesen danken. Als erstes für die hilfreichen Denkanstöße während des gesamten Arbeitsverlaufes. Seine andere Blickrichtung als Nicht-Informatiker hat deutlich zum interdisziplinären Charakter dieser Arbeit beigetragen. Insbesondere aber auch für die hervorragende Unterstützung bei der Exkursion in den Oman um die App vor Ort testen zu können. Sei es als Fahrer oder als ortskundiger Führer im Nebel. Ohne ihn wären die zahlreichen guten Messorte nur schwer von uns gefunden worden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

AGYA Arab-German Young Academy of Sciences and Humanities

A-GPS Assisted Global Positioning System

BMBF Bundesministerium für Bildung und Forschung

CUDA Compute Unified Device Architecture

dBm Dezibel Milliwatt

DLSR Sony DLSR-A900

FPS Frames per Second

GPIO General Purpose Input Output

GPS Global Positioning System

GUI Graphical User Interface

HSV Hue Saturation Value

IATI-SGD IATI pilot project: Submarine Groundwater Discharge: Adaption of an autonomous aquatic vehicle for robotic measurements, sampling and monitoring

JNI Java Native Interface

m Meter

N5X LG Nexus 5X

N9 HTC Nexus 9

NFC Near Field Communication

ODK OpenDataKit

OpenCV Open Computer Vision

P2P Peer to Peer

P8 Huawai P8 lite

RGB Red Green Blue

RSSI Received Signal Strength Indicator

S4 Samsung Galaxy S4 mini

SDK Software Developer Kit

slf4j Simple Logging Framework for Java

TRC The Research Council, Oman

UFZ Helmholtz-Zentrums für Umweltforschung

USB Universal Serial Bus

WLAN Wireless Local Area Network

XML Extensible Markup Language

Kapitel 1

Einleitung

Smartphones sind aus dem heutigem Leben nicht mehr wegzudenken. Primär dienen sie der Kommunikation der Menschen untereinander. Jedoch bieten sie auf Grund der verschiedenen verbauten Sensoren auch eine Plattform zur Datenerfassung von physikalischen Messwerten. Dabei ist zu beachten, dass jeder Hersteller sowohl unterschiedliche Sensortypen als auch unterschiedliche Sensorenmodelle verbaut. Das quelloffene Betriebssystem Android, welches maßgeblich von Google entwickelt wird, bietet einem Entwickler vielfältige Möglichkeiten. Die Kombination aus den vorhandenen Sensoren und den Kommunikationsmöglichkeiten, wie zum Beispiel das Mobilfunknetz, Wireless Local Area Network (WLAN) oder Bluetooth, legen die ideale Grundlage für ein verteiltes Sensorenframework. Es gibt zwar schon andere Frameworks für Messungs-Apps, jedoch sind diese wie z.B. das OpenDataKit (ODK), auf ein Smartphone begrenzt und können daher ihre Daten nicht mit anderen Geräten teilen.

In vielen Teilen der Erde sind wir es gewohnt, dass wir kabellos und mobil auf zahllose Dienste zugreifen können. Dazu ist jedoch eine zentrale Infrastruktur notwendig, die diese Angebote auch vor Ort verfügbar macht. Im letzten Schritt vor dem Nutzer geschieht dies überwiegend über das Mobilfunknetz oder WLAN. Ersteres bietet eine größere Flächenabdeckung, wohingegen WLAN derzeit die höhere Übertragungsgeschwindigkeit bietet. Jedoch sind beide Infrastrukturen noch nicht flächendeckend eingerichtet. Auch in Deutschland gibt es immer noch meist dünn besiedelte Gebiete, in denen ein Mobilfunkempfang nicht oder nur mangelhaft vorhanden ist. Aber auch in anderen Teilen der Welt ist ein Mobilfunkempfang nicht immer gewährleistet. Einerseits kann dies permanent sein, zum Beispiel, dass es sich wirtschaftlich nicht lohnt, in bestimmten Gebieten entsprechende Funkmasten aufzustellen oder es ist temporär beispielsweise nach Naturkatastrophen, durch die die lokale Infrastruktur wie Stromversorgung und Sendemasten zerstört wurden.

In beiden Fällen, die zwar unterschiedliche Ursachen haben, aber technisch gesehen sich nicht voneinander unterscheiden, kann ein dezentrales Netzwerk ohne die Notwendigkeit einer zentralen Instanz die Kommunikation ermöglichen. Daher wird folgend ein Framework vorgestellt, welches ermöglicht,

die Daten der Android Sensoren zu verarbeiten und je nach Anwendungsfall entsprechend mit anderen Geräten oder auch direkt mit einem zentralen Server zu teilen.

Die vorliegende Arbeit gliedert sich wie folgt: in Kapitel 2 wird der Kontext und ähnliche Arbeiten dargestellt. Anschließend folgt in Kapitel 3 die Erläuterung der technischen Grundlagen, die dieser Arbeit zugrunde liegen. Darauf folgt eine kurze Dokumentation der Implementierung des Frameworks in Kapitel 4. Erläuterungen, welche Versuche und Tests zur Entwicklung der Messmethoden beigetragen haben werden in Kapitel 5 genannt, welches mit einem Fazit abschließt. In Kapitel 6 werden mögliche weitere Arbeiten, die auf dieser Arbeit aufbauen, genannt.

Kapitel 2

Related Work

In diesem Kapitel werden zur groben Einordnung der vorliegenden Arbeit weitere Arbeiten genannt, von denen ein Teil von dem entwickelten Framework profitieren kann, ein anderer Teil stellt dem Framework die Netzwerkvoraussetzugen zur Verfügung. Ebenfalls wird mit dem ODK ein ähnliches Framework vorgestellt.

Der Zusatz “Android in the Jungle” in der Ausschreibung zu dieser Arbeit machte deutlich, in welche Richtung der hauptsächlich zu betrachtende Anwendungsfall für das Sensorframework geht. In Zusammenarbeit mit Jan Friesen vom Department Catchment Hydrology des Helmholtz-Zentrums für Umweltforschung (UFZ) in Leipzig wurde der Anwendungsfall der Nebeldichtemessung für den semi-ariden Regenwald in der Bergregion nördlich von Salalah in dem Gouvernment Dhofar im Südwesten des Omans entwickelt. Die dortige Gebirgsformation, welche durch ihren Ost-West-Verlauf die Wüstengebiete im Norden von dem arabischen Meer im Süden trennt, staut die von Süden kommende feuchte maritime Luft. Dieser so genannte orographische Nebel [Oro16] ist eine wichtige Wasserquelle der dortigen Flora. Die Bäume und anderen Pflanzen kämmen die kleinen Wassertröpfchen aus der Luft und leiten das Wasser entweder am Stamm entlang zu den Wurzeln oder es tropft direkt von den Blättern auf den Boden. Die Stämme der Bäume sind hierbei nicht nur feucht, sondern das Wasser läuft als sichtbares Rinnsal den Stamm hinunter. Nach Aussage von Jan Friesen kann der Feuchtigkeitsgehalt des Bodens im Bereich unter den einzelnen Bäumen im Vergleich zu baumfreien Flächen somit verdoppelt werden.

2.1 Hydrologische Forschungsprojekte

Im Rahmen verschiedener Forschungsprojekte kann eine Nebeldichtemessung durch Android Smartphones künftig eine weitere Informationsquelle darstellen. Unter anderem kann zum Beispiel das IATI pilot project: Submarine Groundwater Discharge: Adaption of an autonomous aquatic vehicle for ro-

botic measurements, sampling and monitoring (IATI-SGD) von Messungen mit diesem Framework profitieren, indem mit diesem die Nebeldichte bestimmt wird, welche maßgeblich für die Zuflüsse in das Grundwasser durch die Kondensation des Nebels an Bäumen und Sträuchern ist.

Auch das Projekt “P3: People, Pollution and Pathogens- Mountain ecosystems in a human-altered world” welches vom UFZ betreut wird und den Einfluss des Klimawandels auf verschiedene Bergregionen erforscht, kann in Zukunft mit dem entwickelten Framework weitere Daten sammeln.

2.2 Opportunistic Content-Centric Networking

Da gerade in der Anfangszeit einer neuen App noch wenige Benutzer vorhanden sind, ist auch anzunehmen, dass die Benutzer sich seltener über den Weg laufen. Daher kann nicht von durchgehend verlässlichen Verbindungen zwischen den einzelnen Benutzern beziehungsweise Smartphones, welche als Netzwerkknoten agieren, ausgegangen werden. Aus diesem Grund bieten sich opportunistische Peer to Peer (P2P) Netzwerke an, da diese auch nicht durchgehend zusammenhängende Netze in ihrer Routingstrategie berücksichtigen. Es gibt einige Entwicklungen, die ein opportunistisches P2P Netzwerk zum Datenaustausch anbieten, auf denen man aufbauen kann. So nennt Ólafur R. Helgason auch schon den Anwendungsfall der Weiterleitung von Sensordaten “This category relates to applications that require transporting sensor data from devices in the field to a sink node or infrastructure network” [RHYK+10] für das dort vorgestellte Netzwerk.

2.3 Opptain

Aber auch am Lehrstuhl für Technik sozialer Netze wurde ein ähnliches Framework erstellt. Dass Opptain genannte Framework stellt eine opportunistische P2P Netzwerktopologie unter anderem mit der Möglichkeit eines Dateiaustausches für andere Apps oder auch Frameworks zur Verfügung [IG15].

2.4 Open Data Kit Sensors

Ein anderer Ansatz eines Sensor Frameworks für Android ist von W. Brunette in “Open Data Kit Sensors: A Sensor Integration Framework for Android at the Application-Level” [BSC+12] beschrieben. Dort wird ein Framework vorgestellt, um Programmierern von darauf aufbauenden Apps den Sensorzugriff zu vereinheitlichen und somit zu erleichtern. Im Gegensatz zur vorliegenden Arbeit wird jedoch kein Netzwerkansatz erwähnt.

2.5 PRISM

Es gibt nicht nur Entwicklungen für das Android Betriebssystem, sondern zum Beispiel auch für Windows Smartphones. “PRISM: Platm for Remote Sensing Using Smartphones” [DMP+10] stellt ein solches Projekt dar. Ebenso wie das ODK und die vorliegende Arbeit wird mit PRISM der Ansatz der einfachen Sensorzugriffe für den Programmierer der App verfolgt. Zudem wird bei PRISM ein großer Wert auf die Modularisierung der Anwendungen gelegt.

2.6 Onboard Fahrzeugkamera

In [HTLA06] stellen die Autoren eine Sichtweitenmessung vor, die mit in Fahrzeugen verbauten Kameras durchgeführt wird. Dazu wird eine einzelne Kamera verwendet. Es müssen sowohl die Fahrbahn als auch der Himmel teilweise auf den Bildern vorhanden sein. Der Anwendungsfall ist fast identisch, nur die Motivation und die verwendete Hardware unterscheidet sich von dem in dieser Arbeit vorgestellten Framework.

Kapitel 3

Grundlagen

In diesem Kapitel werden die verwendeten Testgeräte, sowie die für den Anwendungsfall der Nebeldichtemessung verwendeten Techniken vorgestellt. Zuerst wird hierzu ein Überblick über die vom Android Betriebssystem zur Verfügung gestellten Sensoren gegeben.

3.1 Sensoreigenschaften

Im Folgenden werden die Gegebenheiten des Android Betriebssystems im Bezug auf die Sensoren dargestellt.

Das Android Betriebssystem unterstützt von Haus aus eine Vielzahl von Sensoren. Diese werden dabei nach ihren Messparametern in drei Hauptgruppen unterteilt: Bewegungs-, Umwelt- und Positionssensoren [And16b].

Weiterhin müssen auch die verschiedenen Abstraktionsschichten berücksichtigt werden. So gibt es auf der Hardwareebene die physischen Sensoren. Diese sind softwareseitig durch verschiedene sogenannte Basissensoren angesprochen. Hierbei ist aber zu beachten, dass es sich nicht um eine 1:1 Relation handelt, sondern ein physikalischer Sensor auch für mehrere Basissensoren Daten liefern kann. Mehrere physikalische Sensoren können zusammengefasst werden, um einen bestimmten Messwert zu erhalten. Diese werden als Kompositsensoren bezeichnet. Für Appentwickler stehen durch das Betriebssystem die Basissensoren und die Kompositsensoren zur Verfügung. Es ist den Hardwareherstellern freigestellt, aus welchen physikalischen Sensoren die Basis- und Kompositsensoren ihre Daten erhalten. Für einen Teil der Sensoren wird gefordert, dass diese nur sehr wenig Energie verbrauchen dürfen, da es sich um Sensoren handelt, die ständig im Hintergrund aktiv sein müssen. Als Beispiel kann der PickUp Sensor genannt werden, der ein Signal sendet, sobald das Smartphone aufgehoben wird. Hier könnten im Hintergrund sowohl Beschleunigungssensoren, Helligkeits-

Tabelle 3.1: Basis und Kompositsensoren der Geräte

	Nexus 5X	P8 lite	Y3	Nexus 9	Galaxy mini
Beschleunigung	x	x	x	x	x
Magnetfeld	x	x		x	x
Ausrichtung	x	x		x	x
Gyroskop	x			x	x
Helligkeit	x	x			x
Luftdruck	x			x	
Abstand	x	x			x
Gravitation				x	
Lineare Beschleunigung	x			x	x
Rotationsvektor	x			x	x
Magnetometer, unkalibriert	x			x	
Gyroskop, unkalibriert	x			x	
signifikante Bewegung	x			x	x
Schritterkennung	x			x	x
Schrittzähler	x			x	x
Temperatur	x				
game rotation vector	x			x	
PickUp Gesture	x				

sensoren oder auch der Hall Sensor entsprechende Werte liefern. Daher ist ein Entwickler abhängig davon, welche Basissensoren der Hersteller implementiert hat und welche physikalischen Sensoren hierfür als Grundlage genommen werden. Aus den verbauten physikalischen Sensoren lassen sich so beim Nexus 5X die Basis- und Kompositsensoren proximity, light, accelerometer, gyroscope, gyroscope_uncalibrated, magnetic_field, magnetic_field_uncalibrated, pressure, orientation, step_detector, step_counter, significant_motion, gravity, linear_acceleration, rotation_vector, geomagnetic_rotation_vector, game_rotation_vector, tilt_detector, pick_up_gesture, sowie sync, double_twist, double_tap, window_orientation und internal_temperature mit Daten versorgen.

Für die vorliegende Arbeit sind primär die Bewegungs- und Umweltsensoren von Interesse. Die Positionssensoren werden jedoch auch genutzt, um Metadaten für die anderen gemessenen Sensoren zu erhalten. So kann es für den Lichtsensor von Interesse sein, in welche Richtung das Smartphone gehalten wird, da dies im freien Feld einen großen Einfluss auf die Helligkeit hat.

Die Tabelle 3.1 zeigt, bei welchen getesteten Smartphones welche Basis- und Kompositsensoren zur Verfügung stehen. Mit sinkendem Alter und steigendem Preis steigt auch die Bandbreite der verfügbaren Sensoren.

Die meisten Sensoren, wie zum Beispiel der Helligkeitssensor liefern ihre Ergebnisse in abgeleiteten SI-Einheiten. Im Falle des Helligkeitssensors ist es die Einheit Lux. In einer optimalen Welt würden

zwei Helligkeitssensoren bei gleichem Versuchsaufbau das gleiche Ergebnis liefern. Jedoch ist die reale Welt keine optimale Welt und es gibt Abweichungen. In Kapitel 5.1 wird anhand eines einfachen Versuchs mit dem Helligkeitssensor ein Beispiel der möglichen Abweichungen dargestellt.

3.1.1 Bewegungssensoren

Bewegungssensoren dienen der Bestimmung, wie schnell und wohin sich das Smartphone bewegt. Vielfach gibt es Schnittmengen mit den Positionssensoren. Aus zwei GPS Positionsbestimmungen hintereinander kann eine Bewegung des Smartphones bestimmt werden. Zur Gruppe der Bewegungssensoren werden Beschleunigungssensoren, Sensoren für signifikante Bewegungen oder der Game Rotation Sensor gezählt.

3.1.2 Umweltsensoren

Unter dem Begriff Umweltsensoren werden die Sensoren zusammengefasst, welche Daten über den Zustand der Umgebung des Smartphones sammeln. Hierunter fallen zum Beispiel der Helligkeitssensor, der Temperatursensor, der Gravitationssensor, der Magnetfeldsensor, der Luftdrucksensor, der Luftfeuchtigkeitssensor, aber auch die Kamera.

3.1.3 Positionssensoren

Mit den Positionssensoren lässt sich die Position und Ausrichtung des Smartphones bestimmen. Das Gyroskops liefert die Lage des Smartphones sehr genau und mittels GPS wird weltweit die geographische Position auf wenige Meter (m) genau bestimmt. Hierbei können die verfügbaren WLANs und das Mobilfunknetz unterstützend wirken. Mit dem Abstandssensor wird erkannt, ob die Vorderseite des Smartphones von irgendeinem Objekt verdeckt wird oder nicht. In Kombination mit dem Gyroskop kann man so bestimmen, ob der Benutzer sich das Smartphone ans Ohr hält oder es umgekehrt auf einem Tisch o.ä. liegt.

Die unterschiedlichen Sensoren führen auf Grund ihrer unterschiedlichen Bauweise zu mehr oder weniger voneinander abweichenden Messergebnissen.

3.1.4 Zusammenfassung

Es wurde in diesem Abschnitt gezeigt, dass es viele verschiedene Android Smartphones gibt, die zwar als große Gemeinsamkeit das gleiche Betriebssystem verwenden, aber hardwaretechnisch deutliche Unterschiede aufweisen. Auch softwareseitig gibt es eine Vielzahl an verschiedenen Betriebssystemversionen. Tabelle 3.2 zeigt deutlich die Fragmentierung der Betriebssystemversionen. Die Zählung basiert auf den Geräten, die sich im Zeitraum zwischen 31.05. und dem 06.06.2016 im Google Play Store eingeloggt haben [And16a].

Version	Codename	API	Anteil (%)
2.2	Froyo	8	0.1
2.3.3 - 2.3.7	Gingerbread	10	2.0
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.9
4.1.x	Jelly Bean	16	6.8
4.2.x		17	9.4
4.3		18	2.7
4.4	KitKat	19	31.6
5.0	Lollipop	21	15.4
5.1		22	20.0
6.0	Marshmallow	23	10.1

Tabelle 3.2: Android Versionen am 6. Juni 2016 [And16a]

3.2 Gerätevorstellung

In diesem Kapitel werden die verwendeten Testgeräte vorgestellt. Es wird gezeigt, wie unterschiedlich die Gerätekonfigurationen ausfallen. Die Auswahl an Smartphones ist derzeit so vielfältig, dass nicht alle Geräte berücksichtigen werden können. Daher ist es notwendig, eine kleine Auswahl an Geräten auszuwählen, die möglichst repräsentativ die Fähigkeiten von Android Smartphones wiedergeben. Zum Testen der Software werden in dieser Arbeit verschiedene Smartphones von verschiedenen Herstellern eingesetzt. Es muss beachtet werden, dass zwischen physikalischen Sensoren und Basissensoren unterschieden wird.

3.2.1 Nexus 5X

Das Nexus 5X (bullhead) ist ein von Google in Kooperation mit LG hergestelltes und 2015 veröffentlichtes Smartphone [Nex16a]. Im Gegensatz zu seinen Vorgängern hat das Nexus 5X auch einen Hall Sensor und eine verbesserte Kamera. Im Nexus 5X sind folgende Sensoren verbaut:

- Abstandssensor
- Lichtsensor
- Beschleunigungsmesser
- Gyroskop (kalibriert und unkalibriert)
- Kompass
- Drucksensor
- GPS, Assisted Global Positioning System (A-GPS), Glonass
- Hall

Die Nexus Serie versprach auch beim Nexus 5X top Hardware und zügige Softwareupdates, auf Grund des nativen Android Systems und das zu einem günstigen Preis. Das Nexus 5X eignet sich sehr gut für Entwickler, da diese direkt die nativen Android Funktionen verwenden können und nicht auf die von den Geräteherstellern angepassten Firmware angewiesen sind. Die herstellerspezifischen Softwareanpassungen liefern oftmals interessante Funktionen, die nicht oder noch nicht im Android System vorhanden sind, jedoch kann nicht vorausgesetzt werden, dass auf jedem Smartphone von jedem Hersteller auch diese Funktionen zur Verfügung stehen. Aus diesem Grund erweist sich das Nexus 5X als ideale Entwicklerplattform. Die Kamera besitzt 12,3 Megapixel.

3.2.2 HTC Nexus 9

Das HTC Nexus 9 ist ein Tablet, welches von Google in Kooperation mit HTC 2014 veröffentlicht wurde. Es gehört wie das Nexus 5X zur Nexus-Serie und besitzt damit ein relativ pures Android Betriebssystem ohne weitere herstellerseitige Software.

Folgende Sensoren sind im Nexus 9 verbaut [Nex16b]:

- GPS, A-GPS, Glonass, BeiDou
- Lichtsensor
- Abstandssensor
- Beschleunigungsmesser
- Kompass

Die Kameraauflösung beträgt 8 Megapixel.

3.2.3 Huawai P8 lite

Das P8 lite (HUAWAI ALE-L21) stammt wie das Y3 (siehe Kapitel 3.2.5) vom chinesischen Hersteller Huawai. Im Gegensatz zum Y3 besitzt das P8 lite eine weitaus größere Anzahl an verbauten Sensoren, reicht aber nicht an die Topausstattung des Nexus 5X heran. Trotzdem schlägt die Kamera des P8 lite mit 13 Megapixeln knapp das Nexus 5X. Das P8 lite stellt eine abgespeckte Version des Flaggschiffs P8 dar. Es ist etwas leistungsschwächer, aber dafür auch günstiger als das P8.

Das P8 lite besitzt folgende Sensoren [P8l16]:

- GPS, A-GPS, Glonass
- Beschleunigungsmesser
- Abstandssensor
- Lichtsensor
- Kompass

3.2.4 Samsung Galaxy S4 mini

Das Galaxy S4 mini (serranoveltexx) wurde von Samsung 2013 veröffentlicht. Die verwendete Modellnummer ist GT-I9195. Es kommt, bezogen auf die verbauten Sensoren, sehr nahe an das Nexus 5X heran. Jedoch ist die Kamera mit 8 Megapixeln deutlich schwächer.

Im S4 mini sind folgende Sensoren verbaut [S4S16]:

- GPS, A-GPS, Glonass
- Gyroskop
- Beschleunigungsmesser
- Abstandssensor
- Lichtsensor
- Kompass

3.2.5 Huawai Y3

Das Huawai Y3 (HUAWAI Y360-U61) ist ein Low Budget Smartphone, welches seit Dezember 2015 verfügbar ist. Es ist mit einem Einführungspreis von unter 100 Euro sehr günstig und besitzt dementsprechend nur eine beschränkte Hardwareausstattung. Als einziger Sensor ist ein Beschleunigungssen-

sor vorhanden. Es besitzt die von Huawai entwickelte Benutzeroberfläche Emui. Dieses Smartphone deckt den unteren Rand der Ausstattungsmöglichkeiten ab.

3.3 Logging

Logging ist ein zentraler Bestandteil des Frameworks. Einerseits müssen die Messdaten mitgeloggt werden, aber andererseits ist es notwendig, dass einzelne Schritte im Programmablauf protokolliert werden. Auch bei der Entwicklung hilft ein genaues Logging der Systemfunktionen bei einer Fehlersuche. Jedoch hat jeder Entwickler sein eigenes bevorzugtes Loggingsystem. Damit es nicht dazu kommt, dass in einem Projekt mehrere verschiedene Logger verwendet werden müssen, weil die einzelnen Komponenten von verschiedenen Autoren stammen, wird zwischen dem Code und dem Logger eine weitere Abstraktionsschicht eingeführt, das Loggingframework. Dieses dient dazu, im Code einheitliche Logging-Aufrufe zu verwenden, ungeachtet dessen, mit welchem Logger letztenendes die Ausgabe erfolgt. In der vorliegenden Arbeit wird das Simple Logging Framework for Java (slf4j) [1] in Verbindung mit der logback-android Bibliothek verwendet. Logback ermöglicht es, mehrere Ziele zu definieren, in die unterschiedliche Logs geschrieben werden [log16]. Je nach Entwicklerwunsch können die Logeinträge so unter anderem in eine Datenbank, Datei oder das Systemlog geschrieben werden.

3.4 Netzwerkansätze

Zentraler Aspekt des Sensorenframeworks ist die Netzwerkanbindung. Hierzu werden exemplarisch mehrere Ansätze dargestellt. Der erste Ansatz ist die direkte Verbindung des messenden Gerätes zum zentralen Datenserver. Dieser Server dient dazu, alle Daten, die über die verschiedenen Geräte gesammelt wurden, zentral zu speichern, um anschließend auf Basis der Daten Auswertungen durchführen zu können. Er ist in jedem Fall das Ziel aller Messdaten. Auf Grund der Verlagerung des Schwerpunktes auf die Messmethoden wurde nur der Dateiexport realisiert. Für die anderen Möglichkeiten stehen entsprechende Schnittstellen zur Verfügung.

3.4.1 Datenbankserver

Der Datenbankserver ist die einzige zentrale Instanz im gesamten Framework. Er hat keinerlei verwaltungstechnische oder kontrollierende Wirkung, sondern dient lediglich dazu, die dezentral gesam-

[1] http://www.slf4j.org/android/

melten Messwerte an einem Ort zu sammeln, um im späteren Verlauf die Daten zur Auswertung zur Verfügung zu stellen. Er besitzt eine Internetverbindung, um die Messdaten empfangen zu können. Die einfachste Version stellt ein Webinterface zur Verfügung, um die von den Smartphones gesammelten Daten zu empfangen und in die zentrale Datenbank zu schreiben. Weitere Ausbaustufen die zum Beispiel Auswertungsmöglichkeiten beinhalten sind möglich, beeinflussen aber die eingehende Kommunikation zwischen den sammelnden Smartphones und dem Server nicht.

3.4.2 Direkte Verbindung

In diesem Szenario hat das Smartphone, welches Messdaten gesammelt hat, direkten Zugriff auf das Internet. Hierüber kann es den Datenbankserver direkt erreichen und die Daten hochladen. Der Vorteil dieser Methode liegt in der unmittelbaren Übermittlung der Daten an den Server. Die Daten sind zeitnah verfügbar und es sind relativ wenig Faktoren vorhanden, die die Übertragung stören oder verzögern.

3.4.3 P2P Verbindung

Der P2P Ansatz ist zum Beispiel für große Gebiete sinnvoll, in denen es großflächig noch keine drahtlose IT Infrastruktur wie zum Beispiel das Mobilfunknetz oder WLAN Accesspoints gibt. Als mögliche Software für die Netzwerkkommunikation kann das am Lehrstuhl entwickelte Opptain Netzwerk verwendet werden. Dies dient dazu einen möglichen P2P Ansatz zu zeigen.

Wenn ein Smartphone keine Verbindung zum Internet herstellen kann, aber ein anderes Smartphone, auf welchem auch die auf dem Framework basierende App läuft, dann können diese beiden Smartphones die Messdaten untereinander direkt austauschen. Wenn eines dieser am Austausch beteiligten Geräte wieder eine Verbindung zum Internet und somit auch zum Datenbankserver herstellen kann, können die Messwerte beider Smartphones an den Server übertragen werden. Durch die Weiterleitung über gegebenenfalls mehrere Geräte hinweg kann es eine gewisse Zeit dauern, bis die Daten auf dem Server ankommen.

3.4.4 Dateiexport

Um auch ohne irgendwelche Netzwerke wie Mobilfunknetz oder WLAN an die gesammelten Daten zur Auswertung heranzukommen, ist eine dritte Variante der Datenübertragung, ein Dateiexport, möglich. Auch wenn dies netzwerkunabhängig ist, wird diese Variante genannt, da auch sie zum Da-

tenexport gehört. Hierbei wird der gesamte Inhalt der Datenbanktabelle für die Messungen in eine kommaseparierte Wertedatei geschrieben. Diese so genannte CSV Datei kann zum Beispiel per Universal Serial Bus (USB) Kabel auf einen PC übertragen werden. Das Dateiformat der CSV lässt viele Möglichkeiten der Weiterverarbeitung zu. So ist ein Import in eine Datenbank möglich, aber auch ein Import in Tabellenkalkulationsprogramme wie Microsoft Excel oder LibreOffic Calc.

3.5 OpenCV Android

OpenCV steht für Open Computer Vision und ist mit über 2500 Algorithmen ein sehr umfangreiches Framework zur digitalen Bildverarbeitung, so wie für maschinelles Lernen. Es enthält sowohl klassische, als auch neue Algorithmen im Bereich der Bildverarbeitung. Neben einem Interface für Java und dem Support für Android stehen auch noch weitere Interfaces für C++, Python oder Matlab zur Verfügung. Zudem werden neben Android auch Windows, Mac und Linux unterstützt. Ursprünglich in C++ geschrieben, wird aktuell eine Anbindung an Compute Unified Device Architecture (CUDA) verwirklicht. Damit die Funktionen auch in Java verwendet werden, wird das Java Native Interface (JNI) verwendet (vgl. [Abo16]).

3.5.1 Einbindung in eigene Projekte

Um OpenCV für eigene Android Projekte verwenden zu können, sind grundsätzlich zwei Schritte notwendig. Zum einen muss das Software Developer Kit (SDK) in das Projekt importiert werden. Zum Zweiten muss beim ersten Start der App der OpenCV Manager aus dem Appstore Android Play heruntergeladen werden.

In der Android Version ist für Live-Bildverarbeitung mit OpenCV notwendig, dass eine Activity mit einem Vorschauelement verwendet wird. Dies erschwert das Modularisieren der App, da die Elternklasse auf jeden Fall von Activity erben muss. Ein Einbinden in Services, um z.B. im Hintergrund Bilder aufnehmen zu können, sind mit Boardmitteln nicht möglich. Daher ergibt sich eine starke Einschränkung bei der flexiblen Gestaltung des Frameworks. Soll die Kamera als ein Sensor verwendet werden, bei dem fortlaufend Funktionen aus OpenCV ausgeführt werden sollen, so muss dies in einer Activity und somit im Vordergrund der App geschehen.

3.6 Anwendungsfall Nebelmessung

In den folgenden Abschnitten wird der Anwendungsfall “Nebeldichtemessung” des verteilten Sensorenframeworks untersucht. Einleitend steht eine Erklärung, was Nebel genau ist. Denn nur wenn genau bekannt ist, was gemessen werden soll, ist es möglich passende Messverfahren zu entwickeln. Darauf folgend sind mehrere verschiedene Ansätze skizziert, die sich jeweils als mehr oder weniger praktikabel erwiesen haben. Hierzu werden alle Möglichkeiten in Betracht gezogen, mit denen ein Smartphone Informationen über die Umgebung erhalten kann. Neben den normalen Sensoren sind dies auch die Kamera und die Kommunikationsmöglichkeiten wie Near Field Communication (NFC), Bluetooth und WLAN.

3.7 Physikalische Grundlagen

Um Verfahren für die Nebeldichtemessung zu entwickeln, ist es notwendig, vorher die physikalischen Eigenarten von Nebel zu kennen. Nebel ist wie eine Wolke in Bodennähe. Ebenso wie Wolken entsteht Nebel durch winzige Wassertröpfchen, welche an sogenannten Kondensationskeimen in der Luft kondensieren. Diese Kondensationskeime können zum Beispiel Staubpartikel oder Pollen sein. Wenn warme, feuchte Luftmassen auf kalte Luftmassen treffen, kühlt sich die warme Luft etwas ab und kann dadurch nicht mehr so viel Wasserdampf wie bisher speichern, als Folge kondensiert der Wasserdampf und wird zu Nebel. Hierbei ist zu beachten, dass Wasserdampf den unsichtbaren, gasförmigen Zustand des Wassers meint und nicht den Nebel, der beispielsweise über einem Kochtopf mit kochendem Wasser beobachtet werden kann. Meteorologen klassifizieren Nebel anhand der Sichtweiten in drei Gruppen. Über 500 m Sichtweite spricht man von leichtem Nebel, bis 200 m Sichtweite von dichtem Nebel und unter 100 m Sichtweite heißt es sehr dichter Nebel [SWR06].

3.8 Nebelmessung

In diesem Kapitel werden die Methoden beleuchtet, die für eine Nebeldichtemessung sinnvoll erscheinen. Zum einen sind die in die Smartphones eingebaute Sensoren hilfreich, aber auch die Kamera, welche nicht direkt zu den Sensoren gehört, weist einige Einsatzmöglichkeiten auf.

3.8.1 Android Sensoren

Für die Nebeldichtemessung erscheinen auf den ersten Blick das Hydrometer, das Barometer und das Thermometer geeignet, um relevante Messwerte aufzunehmen. Das Hydrometer misst die Luftfeuchtigkeit. Somit liefert es einen ersten Anhaltspunkt, ob das Entstehen von Nebel überhaupt möglich ist. Denn in trockener Luft kann kein Nebel entstehen (vgl. 3.7). Das Barometer zeigt den Luftdruck, und das Thermometer die Temperatur an. Diese beiden Messwerte liefern einen Hinweis, ob die Voraussetzungen für die Nebelentstehung gegeben sind. Jedoch ist gerade das Hydrometer ein nur sehr selten verbauter Sensor. Eine genaue Angabe lässt sich hier nicht tätigen, da es keine zentrale Datenbank gibt, in der die Spezifikationen der Smartphones aufgelistet sind. Vergleichsportale wie phonearea.com [2] liefern zwar die genauen Spezifikationen, lassen sich jedoch nicht nach der Sensorausstattung durchsuchen. Trotz intensiver Suche konnte nur das Samsung Galaxy S4 gefunden werden, welches ein Hydrometer besitzt.

3.8.2 Beeinträchtigung von Funksignalen

Da Nebel aus vielen kleinen Wassertropfen in der Luft besteht, liegt es nahe, dass Nebel die Funksignale von z.B. WLAN oder Bluetooth analog zum sichtbaren Licht ablenkt und schwächt. Auf definierten Distanzen wäre es möglich, ausgehend von der Stärke der Beeinflussung, Rückschlüsse auf die Nebeldichte zu ziehen. Als Messaufbau werden zwei Smartphones verwendet, von denen eines der beiden ein konstantes WLAN Signal aussendet und das Andere in einer definierten Entfernung die Stärke des noch empfangenen Signals misst. Jedoch musste dieser Ansatz verworfen werden, da Gough Lui im Paper “Differences in RSSI Readings Made by Different Wi-Fi Chipsets: A Limitation of WLAN Localization” [GLLDR11] darlegt, dass Ergebnisse aus Entfernungsmessungen auf Basis der Signalstärke stark zwischen verschiedenen Geräten variieren. Eine Schwankung zwischen verschiedenen Geräteypen wäre zu erwarten gewesen, da es nahe liegt, dass unterschiedliche Hardware auch unterschiedliche Messergebnisse liefert.

Die Versuchsaufbauten sind vergleichbar, da Lui bei gleicher Entfernung gleiche Messwerte bei gleichen Umgebungsvariablen erwartet. Bei der Nebeldichtemessung werden bei gleicher Entfernung mit steigender Nebeldichte schwächere Signale erwartet. Lui zeigt jedoch, dass Abstand und Signalstärke nicht miteinander korrelieren. Es wird begründet, dass Funkchiphersteller an keine Norm gebunden sind, wie die Singnalstärke zu berechnen ist. Zudem werden in gleichen Smartphonemodellen gelegentlich unterschiedliche Funkchips eingebaut, je nachdem welche gerade lieferbar sind.

Die NFC ist auf Grund ihrer geringen Reichweite gemäß Spezifikation von 10 cm [JL10] nicht geeig-

[2] http://www.phonearena.com/

net, da auf dieser kurzen Strecke keine nennenswerte Streuung der Funksignale im Nebel stattfinden kann.

Optische Kommunikation wie zum Beispiel Infrarot wird hier nicht näher betrachtet, da diese Übertragungsmöglichkeit bei Android Smartphones derzeit nur vereinzelt anzutreffen ist. Lediglich bei wenigen neuen Geräten werden Infrarotempfänger und Sender wieder verbaut, um beispielsweise als digitale Fernbedienung dienen zu können.

3.8.3 Entropie

Als nächster Ansatz wurde ein grafischer Ansatz gewählt. Es wurde die Hypothese aufgestellt, dass bei Bildern, die neblige Landschaften zeigen, auf Grund der weichzeichnenden Eigenschaften von Nebel die Entropie der Fotos abnimmt. Für einige wenige Bilder war dies auch der Fall, jedoch traf diese Annahme nur auf Bilder mit starkem Nebel zu. Bei den anderen waren die Elemente im Vordergrund noch so deutlich sichtbar, sodass die Entropie noch hoch war. Im ersten Schritt wurden die Fotos in 256-stufige Graustufenbilder konvertiert und anschließend die Histogramme überprüft. Auffällig war, dass bei der Mehrheit der Nebelbilder der Anteil der Helligkeitsstufen im Bereich von 200 einen Peak aufwies. Jedoch war dieser nicht signifikant genug, um aus dem Auftreten dieses Peaks auf Nebel im Bild schließen zu können.

Auch die Vermutung, dass Nebelbilder ein sehr geglättetes Histogramm aufweisen, konnte nicht verifiziert werden. So wies nur ein Bild aus dem Testset eine glatte, nicht gezackte Kurve des Helligkeitsverlaufes auf.

Die Berechnung der Entropie

Zur Berechnung der Entropie wird zuerst für das als Parameter übergebene Graustufenbild mit Hilfe der OpenCV Funktion calcHist ein Histogramm mit 256 Einträgen erstellt. Die Pixelanzahl wird anschließend aus Höhe und Breite des Bildes berechnet. Der Hauptschritt der Berechnung durchläuft die Histogrammliste und berechnet für jeden Farbwert die Wahrscheinlichkeit des Auftretens und multipliziert diese mit dem Logarithmus der Auftrittswahrscheilichkeit zur Basis 2. Der zugehörige Python Code sieht folgendermaßen aus:

```
1 def entropy(image):
2     hist = cv2.calcHist([image], [0], None, [256], [0, 256], True)
3     histlist = list(hist)
4
5     entr = 0.0
```

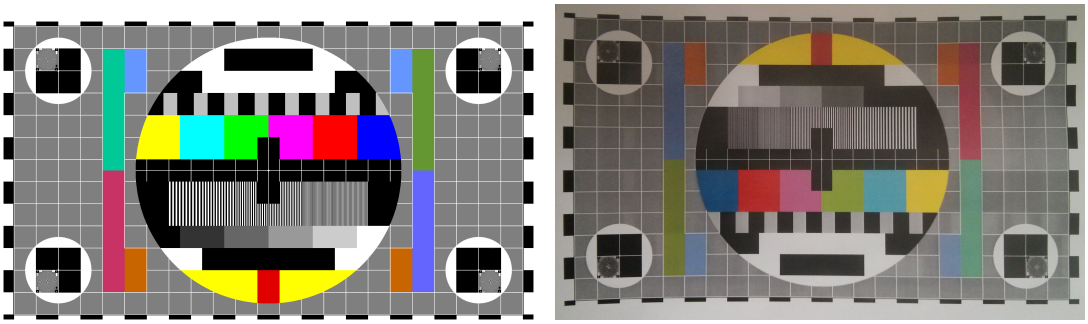
```
6       height, width = image.shape
7       totalsize = height * width
8
9       for i in xrange(0,255):
10          occ = histlist[i][0]
11
12          if (occ > 0):
13              entr = entr + (((occ*1.0)/(totalsize*1.0)) *  log(((occ*1.0)/(
        ↪ totalsize*1.0)), 2))
14
15      entr = entr * -1
16      return entr
```

Fernsehtestbild

Da viele Faktoren die Bildeigenschaften beeinflussen, wurde ein Bildschirmtestbild, wie man es von früher vom Fernseher kennt, verwendet 3.1a. Im ersten Schritt wurde aus der originalen Grafik das Histogramm 3.2a gewonnen. Dieses weist nur wenige einzelne Peaks auf, da die einzelnen Farben diskrete Werte sind und keinerlei Farbverläufe oder Abstufungen enthalten. Nun wurde diese Grafik ausgedruckt und mit verschiedenen Handykameras mehrfach fotografiert. Von diesen Fotos 3.1b wurden nun auch Histogramme 3.2b erstellt. Auf Grund von Umgebungseinflüssen wie Lichteinfall und Qualität der Kamera werden einheitliche Farbflächen der Ursprungsgrafik im Foto in verschiedenen Farbabstufungen dargestellt. Daher sind die Histogramme der Fotos nicht mehr diskret, sondern weisen viele verschiedene Farbstufen auf. Als nächsten Schritt wurden mehrere Fotos im Grafikprogramm Gimp mit einer künstlichen, teiltransparenten Nebelschicht überlagert. Allerdings konnte kein signifikanter Unterschied zwischen den Fotos und den Fotos mit überlagerter Nebelebene festgestellt werden . Auch eine Betrachtung der Entropie brachte keine nennenswerten Unterschiede. Lediglich beim Vergleich der Entropie zwischen der Ursprungsgrafik und der überlagerten Grafik konnte ein signifikanter Unterschied festgestellt werden (vgl. Tabelle 3.3). Jedoch ist genau dieser Fall für die praktische Nebeldichtemessung uninteressant, da es mit Handykameras kaum möglich sein wird, solche Fotos mit diskreten Farbabstufungen zu erstellen.

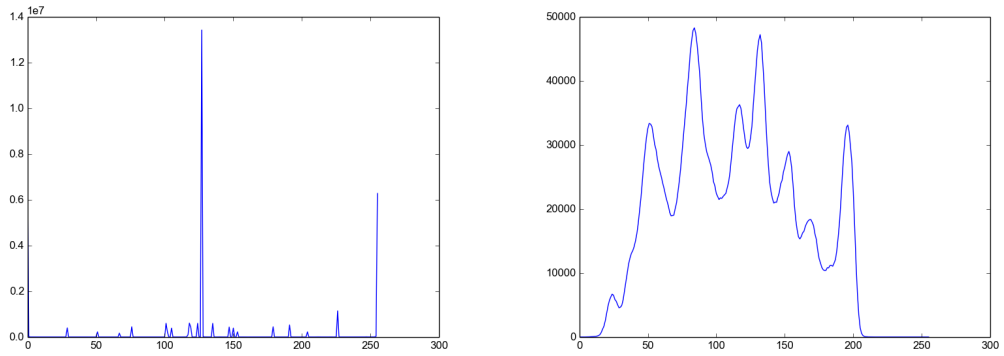
	ohne Nebel	mit Nebel
Originalgrafik	2.5138	6.4190
Foto	7.3878	7.2627

Tabelle 3.3: Entropie Testbild



(a) Originalgrafik (b) Foto

Abbildung 3.1: Testbild



(a) Originalgrafik (b) Foto

Abbildung 3.2: Histogramme

3.8.4 Mehrere Referenzpunkte

Nachdem die Messmethoden über die Entropie und die Gerätesensoren wenig erfolgreich waren, war es notwendig, eine neue Messmethode zu entwickeln. Im Gegensatz zu den bisherigen Methoden, die lediglich die Gerätesensoren verwendet haben, sind nun auch Zieltafeln im Messbereich vor der Kamera notwendig. Diese werden dazu in gleichmäßigen Abständen auf einer geraden Strecke aufgestellt. Nun wird ein Foto von den Zieltafeln gemacht, sodass alle Tafeln nebeneinander sichtbar sind. Hierbei ist zu beachten, dass der sichtbare Abstand ausreichend ist, um die einzelnen Tafeln voneinander trennen zu können. Denn bei Nebel werden die entferntesten Markierungen ab einer gewissen Nebeldichte nicht mehr messbar sein. Aus der Anzahl der noch sichtbaren Tafeln und dem bekannten Abstand zwischen den jeweiligen lässt sich mit einer Ungenauigkeit, die dem Abstand der Markierungen entspricht, die Sichtweite bestimmen. Bei einer angenommenen lokalen, gleich verteilten Nebeldichte kann man auf Grund der Sichtweite auf die Nebeldichte schließen. Im Praxiseinsatz ist eine feststehende Halterung denkbar, um eine einheitliche Perspektive über mehrere Benutzer hinweg zu ermöglichen. Eine erfolgreiche Umsetzung eines solchen festen Fotopunktes ist beispielsweise

Abbildung 3.3: Fire Recovery Dokumentation in den USA

die Dokumentation der Regeneration von Waldbrandgebieten in den USA. In Abbildung 3.3 ist ein solcher Fotopunkt zu sehen [3] .

Bei dieser Messmethode ist zu beachten, dass auf Grund der notwendigen ortsfesten Zieltafeln nur ein sehr begrenzter Messbereich verfügbar ist. Neben den Tafeln ist die Reichweite auch von der Leistungsfähigkeit der Kameras abhängig. So müssen in mehreren dutzend Metern Entfernung die Tafeln relativ groß sein, damit diese durch die Smartphonekameras noch ausreichend groß wahrgenommen werden können. Der Messbereich erstreckt sich jeweils vom Kamerastandort bis zum letzten Markierungspunkt.

[3] Quelle: http://www.modernhiker.com/2014/12/02/help-document-the-springs-fire-recovery/

Kapitel 4

Implementierung

In diesem Kapitel wird die Implementierung des Sensorframeworks dokumentiert und erläutert. Dazu wird jedem Modul ein eigener Abschnitt gewidmet. Bevor auf die einzelnen Module eingegangen wird, wird ein Überblick über das Zusammenspiel der einzelnen Module gegeben.

Das Modul Datenbankanbindung sorgt, wie der Name schon sagt, für die Verwaltung der Datenbank. Hier werden alle Methoden zur Erstellung und Update der Datenbank, sowie zum Daten ein- und auslesen zur Verfügung gestellt. Dieses Modul wird nur über die zentrale Klasse `SaveClass` angesprochen. Da diese als Singleton implementiert ist, wird in der gesamten App nur eine Datenbankinstanz benötigt.

Der Großteil des Loggings wird von der verwendeten Bibliothek übernommen, sodass diese im Framework nur eingebunden werden und die Logaufrufe getätigt werden müssen.

Ebenfalls in ein Package zusammengefasst wurde die Implementierung der CustomSensoren. Dies besteht aus einem `GenericSensor`, von dem weitere Sensoren abgeleitet werden können. Zudem sind hier auch die notwendigen Klassen `CustomSensorEvent` und `CustomSensorEventListener` untergebracht.

Die restlichen Klassen sind die zentralen Komponenten, die Basisklassen `MeasurementActivity`, `SensorMaster` und `Processor` und bilden das Rückgrat des Frameworks.

4.1 Logging

Zur Konfiguration der logback-android Bibliothek wird eine Extensible Markup Language (XML) Datei verwendet. In der vorliegenden Konfiguration werden lediglich die Lognachrichten mit einer

gekürzten Angabe der hervorbringenden Klasse und der entsprechenden Nachricht angezeigt. Die Ausgabe wird in das logcat geschrieben, die standardmäßige Logausgabe von Android.

4.2 Datenbankanbindung

Die Datenbankanbindung ist ein zentraler Aspekt des Frameworks, denn hiermit werden die gesammelten Daten für eine spätere Weiterverarbeitung gespeichert. Zudem gibt es Methoden zum Datenexport. Folgend werden die hierfür benötigten Klassen erläutert.

4.2.1 Measurements

Zu allererst ist die Klasse `Measurements` zu nennen. Sie repräsentiert einen Messwert und entspricht einer Zeile in der Datenbanktabelle. Diese Klasse enthält Attribute für jede Spalte der Datenbanktabelle. Neben dem eigentlichen Messwert des einzelnen Sensors wird auch der Sensortyp, der Zeitpunkt der Messung und eine eindeutige DeviceID gespeichert. Zudem wird eine fortlaufende numerische ID mit in der Datenbank gespeichert. Pro Gerät ist somit jeder Datensatz eindeutig identifizierbar. In Kombination mit der DeviceID wird ein global eindeutiger, zusammengesetzter Primärschlüssel möglich. Diese Klasse dient der Datenhaltung und enthält dementsprechend keine Verwaltungslogik. Somit enthält die Klasse nur einen Konstruktor, der alle Attribute setzt. Zudem existiert für jedes Attribut eine Getter und Setter Methode zum Auslesen und Ändern der Attribute. Abschließend dient die toString() Methode hauptsächlich zu Debugzwecken.

4.2.2 Helper

Die Klasse `Helper` ist für die Erstellung der Datenbanktabelle sowie deren Update zuständig. Zudem stellt sie öffentlich Konstanten für die Spaltennamen der Datenbanktabelle zur Verfügung. Durch dieses Vorgehen müssen die Spaltennamen nur an einer Stelle im Code definiert werden und müssen bei einer Änderung nur an dieser zentralen Stelle geändert werden. Neben einem Konstruktor stehen in dieser Klasse nur zwei Methoden zur Verfügung. Eine onCreate() Methode zur initialen Anlage der Datenbanktabelle und die andere onUpgrade() zum Upgrade der Tabellenstruktur. Da die `Helper` Klasse vom SQLiteOpenHelper erbt, müssen beide Methoden vorhanden sein, auch wenn im Framework die onUpgrade() Methode nicht verwendet wird. Die Elternklasse SQLiteOpenHelper gehört zum Standard Android System für den Datenbankzugriff.

4.2.3 Source

Als dritte und letzte Klasse der Datenbankanbindung gibt es die `Source` Klasse. Diese bietet dem gesamten Framework eine komfortable Schnittstelle zur Datenbankinteraktion. Die beiden Methoden open() und close() dienen dazu die Datenbankverbindung zu öffnen und wieder zu schließen. Zudem wird jeweils ein entsprechender Logeintrag erzeugt. Mittels der load() Methode wird eine per Parameter übergebene Anzahl an Zeilen geladen und anschließend als Datenbankcursor zurückgegeben.

Außerdem wird mit Hilfe der exportCSV() Methode ein Export im CSV-Format erstellt. Diese Datei wird an der als Parameter übergebenen Stelle im Dateisystem abgelegt. Empfohlen ist hierbei der standardmäßig auf Android Smartphones vorhandene Ordner Downloads. Beim Export wird zuerst eine Überschriftzeile mit den Spaltennamen der Datenbank erstellt. Anschließend werden die einzelnen Datensätze zeilenweise als `Measurement` geladen und der Datei hinzugefügt.

4.3 Zentralkomponenten

Die Zentralkomponenten bestehen aus einer Activtiy und einem Service, die beide als Grundlage für die anwendungsspezifischen Spezialisierungen dienen. Die `Processor` Klasse lagert die Bildverarbeitungsmethoden der Activity aus, um diese übersichtlicher zu halten. Zudem stellt die `SaveClass` als Singleton die zentrale Instanz zum Speichern und Loggen sowohl für die Activities als auch für die Services dar.

4.3.1 SaveClass

Die `SaveClass` ist als Singleton implementiert und kann daher nur einmal pro Lebenszyklus der App instanziiert werden. Beim ersten Aufruf wird eine Instanz der Klasse erstellt, bei weiteren Aufrufen stattdessen die bereits erstellte Instanz der Klassen zurückgegeben. Die `SaveClass` bietet als zentrale Instanz den Zugang zur Datenbankanbindung für die anderen Klassen. Das Abspeichern der Messwerte geschieht durch die saveValue() Methode, welche als Parameter ein `Measurements` Objekt übergeben bekommt. Die Daten hieraus werden nun in ein ContentValues Objekt überführt, da die Android Datenbankschnittstelle auf der die `Source` (vgl. 4.2.3) diese Aufbereitung der einzufügenden Daten erwartet. Die Zeitangabe der Messung wird zur Kompatibilität mit der SQLite Datenbank vorher in das Datumsformat der Datenbank formatiert. Zum Exportieren aller in der Datenbank gespeicherten Datensätze leitet die `SaveClass` den Aufruf lediglich an die `Source` Klasse der Datenbankanbindung weiter.

4.3.2 MeasurementActivity

Die `MeasurementActivity` stellt die Basisklasse für eine Benutzeroberfläche mit einer Cameraview für die Bildverarbeitung mittels Open Computer Vision (OpenCV) zur Verfügung. Es werden hier lediglich die grundlegenden Einstellungen vorgenommen.

In der onCreate() Methode wird eine rudimentäre Benutzeroberfläche erstellt. Es wird explizit auf die XML basierte Möglichkeit verzichtet, da mit der CameraView nur ein einzelnes Objekt in der View angezeigt werden soll. Diese Vorgehensweise erspart die Erstellung einer eigenen Layoutdatei. Es wird lediglich definiert, dass die rückseitige Kamera gewählt werden soll und dass die Frames per Second (FPS) eingeblendet werden sollen. Zudem wird eine Instanz eines `Processors` erzeugt und als Attribut gespeichert. Auch die `SaveClass` und der Logger werden als Attribute referenziert.

In den onPause(), onDestroy() und onResume() Methoden wird die Cameraview, die unter anderem auch für den Kamerazugriff zuständig ist, ordnungsgemäß beendet, damit auch andere Apps anschließend wieder auf die Gerätekamera zugreifen können. Im Gegensatz zu den Sensoren kann die Kamera immer nur von einer App zeitgleich angesprochen werden.

Die onTouch() Methode erfasst die Koordinaten, auf die ein Benutzer auf der CameraView getippt hat und leitet diese zur Verarbeitung weiter. Die Methode onCameraFrame() leitet die übergebene Bildmatrix zur weiteren Verarbeitung an den `Processor` weiter.

Zur einfachen Kontrolle der Hintergrundservices stehen die beiden Methoden startService() und stopService() zur Verfügung. Sollte der Hintergrundservice auch Daten an das Graphical User Interface (GUI) senden, so steht die callback() Mehode zur Verfügung, um die Daten annehmen zu können.

4.3.3 PostProdActivity

Die `PostProdActivity` ist sehr ähnlich zur `MeasurementActivity`. Der größte Unterschied besteht darin, dass diese Klasse keine aktuellen Kamerabilder, sondern bereits abgespeicherte Bilder verwendet, die manuell vom Benutzer ausgewählt werden können. Ansonsten basiert die Bildverarbeitung auf den gleichen Funktionen.

4.3.4 Processor

In dem `Processor` werden große Teile der Bildverarbeitung ausgelagert, damit die `Measurement-Activity` nicht überfrachtet wird und soweit wie möglich eine Trennung zwischen Benutzeroberfläche und Anwendungslogik erfolgt. Da es sich hierbei um eine möglichst einfache Demo handelt, liefert die processMatrix() Methode die übergebene Matrix unverändert zurück. In anwendungsspezifischen Erweiterungen wird diese Methode in abgeleiteten Klassen überschrieben, um die dort notwendige Logik unterzubringen. Die weiteren Methoden wie Konstruktor und die durch `Measurement-Activity` aufgerufene deliverTouchEvent() bestehen nur aus leeren Rümpfen.

4.3.5 SensorMaster

Der `SensorMaster` stellt die Grundlage des Hintergrundservices dar. In der onStartCommand() Methode wird der Logger und die `SaveClass` referenziert. Anschließend besteht die Möglichkeit, die gewünschten Sensoren abzufragen. Da das zentrale Element, das `SensorEvent` von Android nicht über Java Methoden aufrufbar ist, musste für eigene Sensoren eine Alternative, die Custom-Sensoren(siehe 4.4) verwendet werden. Hieraus ergibt sich die Notwendigkeit, dass native Android Sensoren und selbst implementierte Sensoren über zwei verschiedene SensorEventListner abgefragt werden müssen. Das Verhalten beider ist jedoch identisch.

4.4 Eigene Sensoren

Um weitere Sensoren neben den üblicherweise von Android bereitgestellten Sensoren ebenfalls einbinden zu können, mussten einige Klassen selbst implementiert werden.

4.4.1 CustomSensorEvent

Das zentrale Problem, warum das `SensorEvent` von Android nicht auch für eigene Sensoren verwendet werden kann, liegt an dem Umstand, dass es keinen Javacode zum Setzen der Attribute sowie keinen Konstruktor enthält. Dies erfolgt beides in tieferen Schichten des Betriebssystems, worauf man als Appdeveloper im allgemeinen keinen Zugriff hat. Die `CustomSensorEvent` Klasse stellt die grundlegenden Getter zur Verfügung, die auch die `SensorEvent` Klasse bietet. Zusätzliche Setter ermöglichen es, auch die entsprechenden Attribute zu füllen. Die vier Attribute sind: der Sensortyp

als Integer, die Values als Floatarray, ein Float Attribut für die Messgenauigkeit, sowie ein als Long abgespeicherter timestamp des Messzeitpunktes.

4.4.2 CustomSensorEventListener

Der `CustomSensorEventListener` ist ein Interface, welches beispielsweise vom `SensorMaster` implementiert werden kann. Ebenso wie der androideigene `SensorEventListener` beinhaltet der `CustomSensorEventListener` zwei Methoden. Die onSensorChanged() wird aufgerufen, sobald neue Sensordaten für den betrachteten Sensor verfügbar sind. Als zweite Methode gibt es die onSensorStatusChanged(), welche aufgerufen wird, sobald sich etwas beim Sensor ändert. Zum Beispiel kann man hiermit erfassen, wann der Übergang vom Mobilfunknetz in ein WLAN erfolgt.

4.4.3 GenericSensor

Der `GenericSensor` ist eine abstrakte Klasse, die die notwendigen Attribute eines jeden selbst erstellten Sensors bereitstellt. Dies ist die SensorID zur eindeutigen Indentifikation, ein menschenlesbarer Name, ein CustomEventListener als Callback, der bei Veränderungen der Daten aufgerufen wird, sowie eine samplingPeriod um die Aktualisierungshäufigkeit zu beeinflussen. Zudem wird ein Logger und ein Logtag festgelegt. Als abstrakte Methoden müssen zunächst ein Konstruktor und ein Getter für die SensorID durch abgeleitete Klassen implementiert werden. Darüber hinaus muss die Methode registerListener() implementiert werden, um den CustomEventListener und die samplingPeriod zu setzen.

4.4.4 LocationSensor

Als Beispielimplementation für den `GenericSensor` wird der `LocationSensor` aufgeführt. Die üblicherweise durch Android nicht als Sensor zugänglichen Positionsdaten werden mit Hilfe dieser Klasse abgefragt und entsprechend umgeformt, sodass auch die Postionsdaten als Sensorwerte aufgefasst und anschließend weiter verarbeitet werden können.

4.5 Custom Klassen

Als Anwendungsfall und Beispiel, wie das Framework verwendet werden kann, gibt es die Custom Klassen, also die Klassen, die ein Entwickler schreiben muss, um das Framework auf seine Bedürfnisse einzustellen. Diese Klassen tragen alle den Präfix `My` um zu verdeutlichen, dass diese nicht zum eigentlichen Framework gehören, sondern einen beispielhaften Anwendungsfall darstellen.

4.5.1 MyMeasurementActivity

Die `MyMeasurementActivity` ist eine anwendungsspezifische Erweiterung der `Measurement-Activity`. Auf Grund der komplexeren GUI wird diese durch eine XML Layoutdatei abgebildet. Neben der CameraView werden auch die benötigten Buttons und Textfelder in der onCreate() Methode angebunden. Zuletzt wird der `SensorMaster` als Hintergrundservice gestartet. Zum Kamerastart werden mit der Methode onCameraViewStarted() grundlegende Einstellungen für die CameraView getätigt. So wird dort neben der Registrierung eines `Processors` die Größe des Farbspektrums für die Bloberkennung und die Farbe für die Umrandung der Farbblobs gesetzt. Sobald die onTouch() Methode aufgerufen wird, wird aus einer 9x9 Pixel großen Umgebung um die berührte Position herum den durchschnittlichen Farbwert und das zugehöriges Spektrum berechnet. Dieser wird anschließend als Zielwert für die Farbblob-Erkennung verwendet. Zuletzt wird ein Flag gesetzt, dass eine Farbe zur Bloberkennung gewählt wurde.

Die Methode onCameraFrame() überschreibt die Standardimplementierung aus `Measurement-Activity`. Zuerst wird eine Helligkeitsanpassung basierend auf den Benutzereingaben vorgenommen. Anschließend wird geprüft, ob das Flag für die Bloberkennung gesetzt wurde. Falls dies gesetzt ist, wird die aktuelle Bildmatrix an den `MyProcessor` übergeben. Anschließend werden die dort berechneten Umrisse, sowie die Zielfarbe und das Zielspektrum als Legende auf die Bildmatrix gezeichnet und letztenendes an die CameraView zur Anzeige weitergeleitet.

Die saveMatToFile() Methode speichert die aktuelle Bildmatrix als Bilddatei in einem automatisch für das Framework erstellten Ordner im Downloadverzeichnis des Smartphones. Dieses Verzeichnis bietet sich an, da hierhin auch die Messdaten, wie in 4.2.3 genannt, gespeichert werden. Die Methode saveBlobCount() wird durch einen Buttonklick aufgerufen und speichert die aktuelle Anzahl der erkannten Farbblobs als Messwert mit Hilfe der `SaveClass` in die Datenbank.

4.5.2 MyProcessor

Die Klasse `MyProcessor` ist hauptsächlich für die Farbbloberkennung zuständig. Mit der Methode setHsvColor() wird das Farbspektrum im Hue Saturation Value (HSV) Format berechnet. Das häufig verwendete Red Green Blue (RGB) Format ist nicht optimal, da im HSV Format unterschiedliche Helligkeitsstufen eines Farbtons besser abgebildet werden können. Im Gegensatz zur Änderung nur eines Parameter müsste im RGB Farbraum jeder der drei Farbkanäle verändert werden.

In der zentralen Methode process() wird die Bildmatrix auf 1/16 ihrer ursprünglichen Größe verkleinert. Durch diese Verkleinerung sind die Berechnungen schneller möglich, da weniger Bildpunkte betrachtet werden müssen. Im nächsten Schritt wird der Farbraum von ursprünglich RGB in HSV umgewandelt. Die Blobs, die in das vorher definierte Spektrum passen, werden herausgesucht. Anschließend werden für diese Blobs die Umrisse bestimmt, wieder für die ursprüngliche Matrixgröße skaliert und als Attribut gespeichert.

Kapitel 5

Feldtests

In diesem Kapitel werden die einzelnen Versuche und Tests vorgestellt, mit denen das Framework getestet und entwickelt wurde. Einen Großteil macht die Forschungsreise in den Oman aus. Jedoch wurden auch im Vorfeld mehrere Versuche durchgeführt, um eine Basis für effektive Messreihen im Oman zu erhalten. Daher sind die Grundlagen nicht weniger wichtig.

5.1 Helligkeitsmessung

Als einer der ersten Tests wurden die Helligkeitssensoren verschiedener Smartphones miteinander verglichen. Dieser Sensor bot sich an, da er Werte liefert, die auch mit den menschlichen Sinnen nachvollziehbar sind, nämlich die Intensität der Lichteinstrahlung. Zu diesem Zeitpunkt befand sich das Framework noch in einem frühen Anfangsstadium, die Sensordaten wurden unmittelbar nach Erfassung einfach in eine Liste geschrieben. Der Versuchsaufbau bestand aus einer fest positionierten Lichtquelle. Jedes Smartphone wurde so zur Lichtquelle ausgerichtet, dass die Lichtstrahlen senkrecht auf den Helligkeitssensor treffen. Nun wird der Abstand zwischen Lichtquelle und Sensor Stück für Stück verkürzt. Für jede Position wird anschließend zehn Sekunden lang mit der höchstmöglichen Sensorfrequenz die Intensität des einfallenden Lichtes gemessen und aus diesen Werten der Mittelwert berechnet. Eine genaue Angabe, wie oft die Daten vom Sensor bereitgestellt werden, kann nicht exakt angegeben werden, da auch andere Systemprozesse, die auf den Sensor zugreifen, die Geschwindigkeit ändern können. An dieser Stelle sei nochmals erwähnt, dass das Android Betriebsystem mehreren Apps oder Prozessen den Zugriff auf die einzelne Sensoren simultan erlaubt und dabei die Aktualisierungsfrequenz vom höchsten erwarteten Wert abhängig ist. Als grobe Einordnung lässt sich sagen, dass mehrere Messungen pro Sekunde erfolgten.

In Abbildung 5.1 sind die Messergebnisse der gleichen Lichtquelle in den verschiedenen Abständen zum jeweiligen Sensor, gemessen mit zwei verschiedenen Android Smartphones, dargestellt. Auffäl-

lig ist, je heller eine Lichtquelle ist, desto größer werden die Messunterschiede. Anfängliche Unterschiede von wenigen Lux steigern sich bis auf über 2000 Lux. Das ist in Anbetracht dessen, dass der Messbereich lediglich von 0 bis 10.000 Lux reicht, eine Abweichung von 20% bezogen auf den Messbereich. Dieses Experiment ist das Einzige, bei dem das Nexus 4 zum Einsatz kam, da das Nexus 5 zu diesem Zeitpunkt noch nicht zur Verfügung stand.

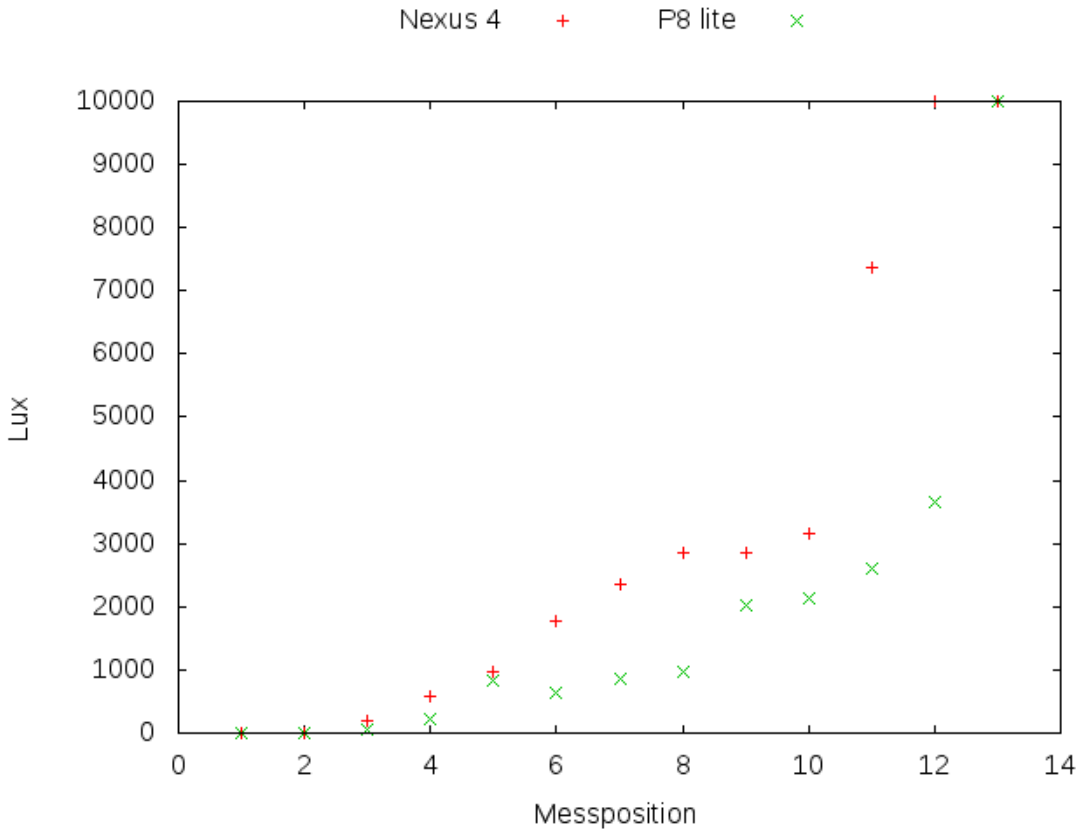


Abbildung 5.1: Abweichende Testergebnisse des Lichtsensors bei verschiedenen Smartphones

5.2 Simulierte Nebeldichte

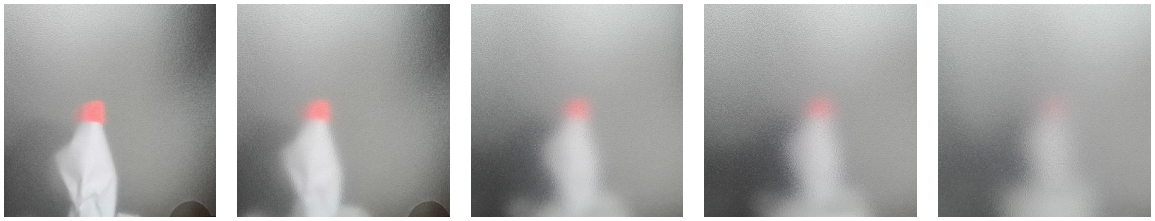
In unmittelbarer Umgebung ist das natürliche Auftreten von Nebel eher selten. Daher muss zu Testzwecken auf eine künstliche Nebelquelle zurückgegriffen werden. Ein erster Versuch wurde mit Rauchpatronen durchgeführt. Obwohl Rauch im Gegensatz zu Nebel aus festen statt flüssigen Kleinstpartikeln besteht, sind die optischen Beeinträchtigungen ähnlich. Für den Versuchsaufbau wurde ein eindeutig erkennbares Zielobjekt, eine Rauchpatrone als Nebelersatz und ein Smartphone in einer Reihe angeordnet. Auf Grund von Sicherheitsvorgaben des Herstellers der Rauchpatronen musste dieser Versuch im Freien stattfinden. Obwohl ein möglichst windstiller Zeitpunkt abgewartet wurde, reichte eine Windstärke von etwa 1-2 Beaufort, um den Rauch der Patrone so schnell zu verteilen, dass

eine sinnvolle Messung nicht möglich war. In Abbildung 5.2 erkennt man, dass der von der Patrone ausgehende Rauch anfangs sehr dicht ist, sich anschließend jedoch schnell verteilt.

Abbildung 5.2: Rauchpatrone mit geringer Rauchausbeute die vom Wind weggeweht wird.

5.3 Klarsichthüllen

Eine weitere Möglichkeit, die Auswirkungen des Nebels zu simulieren, besteht in der Verwendung halbtransparenter Folien. Ein Prototyp zum Testen wurde aus einem großen Karton als Halterung und einer handelsüblichen Klarsichthülle gebaut. Als Zielobjekt dient ein Stück einer orangen Warnweste auf einer weißen Papierhalterung. Durch die auffällige Farbgebung hebt die Warnweste sich gut vom Hintergrund ab. Die ursprüngliche Idee, mehrere dieser Hüllen hintereinander parallel in den Karton zu hängen, erwies sich als zu viel. Es reichte lediglich eine einzige Hülle aus, die auf Grund ihrer Oberflächenstruktur für eine ausreichende Streuung sorgte. Mit wachsendem Abstand wurde dieser Effekt stärker, entsprechend der Wirkung natürlichen Nebels. In Abbildung 5.3 sind eine Auswahl von Fotos mit verschiedenen Abständen zwischen Folie und Zielobjekt dargestellt.



(a) 2 cm (b) 4 cm (c) 6 cm (d) 8 cm (e) 12 cm

Abbildung 5.3: Streuungseffekt durch Klarsichthülle bei verschiedenen Abständen

5.4 Oman

In diesem Kapitel werden die Messungen und Erfahrungen während der Forschungsreise in den Oman präsentiert. Wie schon in Kapitel 2 dargestellt, liegt die Hauptmotivation zur Entwicklung dieses Frameworks in der Erforschung der Wasservorkommen im semiariden Regenwald in den südlichen Küstenregionen des Omans.

Während der Feldarbeit wurden mehrere Messreihen durchgeführt, diese werden durch einzelne Buchstaben von A bis G gekennzeichnet, damit eine schnelle Identifikation ermöglicht wird. Als zweite Beschreibung gibt es noch einen Klarnamen, der die Umgebung charakterisiert. Diese Kennzeichnung ist für Menschen besser nachvollziehbar als eine willkürliche Nummerierung. In Tabelle 5.1 wird die Zuordnung zwischen Buchstabe und Klarname zusammengefasst. Sechs der Messorte liegen in den feuchten Regenwaldregionen. Eine weitere Messreihe wurde in der Wüste durchgeführt. Diese dient als Referenzmessung, da dort im Gegensatz zu den nebligen Bergregionen eine klare Sicht über mehrere Kilometer herrscht.

Tabelle 5.1: Messreihen Bezeichnungen

Kennbuchstabe	Klarname
A	Berghang
B	altes Haus
C	Pfeiler
D	Wüste
E	Hochebene Taiq
F	Felswiese
G	Parkplatz

Neben den verschiedenen Android Geräten standen während der Feldarbeit weitere Geräte zur Aufnahme von Messdaten zur Verfügung. Als Android Geräte standen ein N5X, ein HTC Nexus 9 (N9), ein Huawai P8 lite (P8) und Samsung Galaxy S4 mini (S4) zur Verfügung. Zudem diente die mobile Wetterstation Kestrel 4500 zur Erfassung von Windgeschwindigkeit, Lufttemperatur und Luftfeuch-

tigkeit an den jeweiligen Messorten. Des weiteren wurden mit einem GARMIN GPSMAP 76CS die Koordinaten des Messortes aufgezeichnet. Andere Ortsbestimmungen sind nicht möglich, da die Messungen teilweise abseits der Straßen, mitten in der Natur stattgefunden haben. Teilweise wurden mit der Digitalkamera Sony DLSR-A900 (DLSR) neben den Smartphones ebenfalls Fotos von der Zieltafel in den verschiedenen Abständen gefertigt. Außerdem wurde zum Abmessen von kurzen Distanzen der Laserdistanzmesser Leica Disto D5 eingesetzt.

Zur Bestimmung der Abstände zwischen Smartphone und Zielobjekt wurde nach Möglichkeit eine Laserdistanzmessung verwendet. Jedoch störte der dichte Nebel die Funktionsfähigkeit des Gerätes enorm. Bei Messungen von Distanzen größer als 15 m versagte diese Messmethode regelmäßig. Als Alternative wurde die App “Acoustic Rouler” für iPhones verwendet. Diese App basiert auf der Laufzeit von Schallwellen, die zwischen zwei Geräten ausgetauscht werden [Aco16]. Zur Abstandsbestimmung wurden zwei iPhones verwendet. Es wurde absichtlich kein Maßband verwendet, da alle Messungen digital durchgeführt werden sollten. In der weiteren Entwicklung wäre es somit möglich, auch diese Messmethode in das Framework einzubauen, um lediglich eine einzelne App für alle Messungen zu erhalten.

Der überraschend dichte Nebel erschwerte die Messungen, da sich durch Kondensation auf allen Oberflächen sehr schnell Wassertröpfchen bildeten und somit die Bedienung der Geräte erschwerte. Zudem mussten die Linsen der Kameras sowie die Wetterstation ständig getrocknet werden, um die Messungen möglichst unverfälscht durchführen zu können. Die Tabelle 5.2 bietet einen Überblick über die Wetterbedingungen und die eingesetzten Geräte für den jeweiligen Messort. Die Sichtweite wurde auf ganze Meter gerundet, für Luftfeuchtigkeit und Temperatur wird der Mittelwert der minütlichen Messungen und für die Windgeschwindigkeit das Minimum und Maximum angegeben.

Tabelle 5.2: Übersicht der Rahmenbedingungen der einzelnen Messorte.

Messort	A	B	C	D	E	F	G
Datum	01.08.	02.08.	02.08.	02.08.	03.08.	04.04.	04.08.
Uhrzeit	16:55	10:38	11:35	13:22	13:33	08:11	10:46
Sichtweite (m)	33	29	25	>2000	58	42	30
Luftfeuchtigkeit (%)	n.v.	100	100	78	82	87	83
Windgeschwindigkeit (m/s)	n.v.	0,3-2,9	0-4,5	0,8-8,9	0,5-6,1	0-5	0-2,2
Temperatur (°C)	n.v.	22	22	26	22	25	26
DLSR	x	x	x	x	x	x	x
N5X	x	x	x	x	x	x	
N9				x	x	x	x
P8				x	x	x	x
S4				x	x	x	x

5.4.1 A - Berghang

Am Messort A wurde die erste Messreihe durchgeführt. Dieser Standort war ein leicht abfallender Berghang, der auf einer ausreichenden Fläche eben und frei von Bäumen war. Primär diente dies zur Erfahrungssammlung im Umgang mit den Messgeräten und dem Versuchsprotokoll zu sammeln. Als Ausgangspunkt wurde die Kestrell Wetterstation auf einem Stativ aufgestellt. Ebenfalls wurden die GPS Koordinaten von diesem Punkt aus ermittelt. Von diesem Startpunkt ausgehend wurden anschließend in Abständen von etwa je 5, 10, 15, 20 und 33 m Fotos mit dem Nexus 5X sowie mit der Digitalkamera von der Zieltafel gefertigt. Als Zieltafel dient das in Abbildung 3.1a gezeigte Testbild. Im Nachgang wurden diese Fotos so beschnitten, dass der verbleibende Ausschnitt nur noch die Zieltafel zeigt. Von diesen Ausschitten wurde nun die Entropie berechnet. Die Ergebnisse für Messpunkt A sind in Abbildung 5.4 dargestellt. Die Beschriftung der Punkte ist wie folgt aufgebaut. An erster Stelle wird der Kennbuchstabe des Messortes genannt, hier A. Anschließend folgt getrennt durch einen Doppelpunkt das Aufnahmegerät.

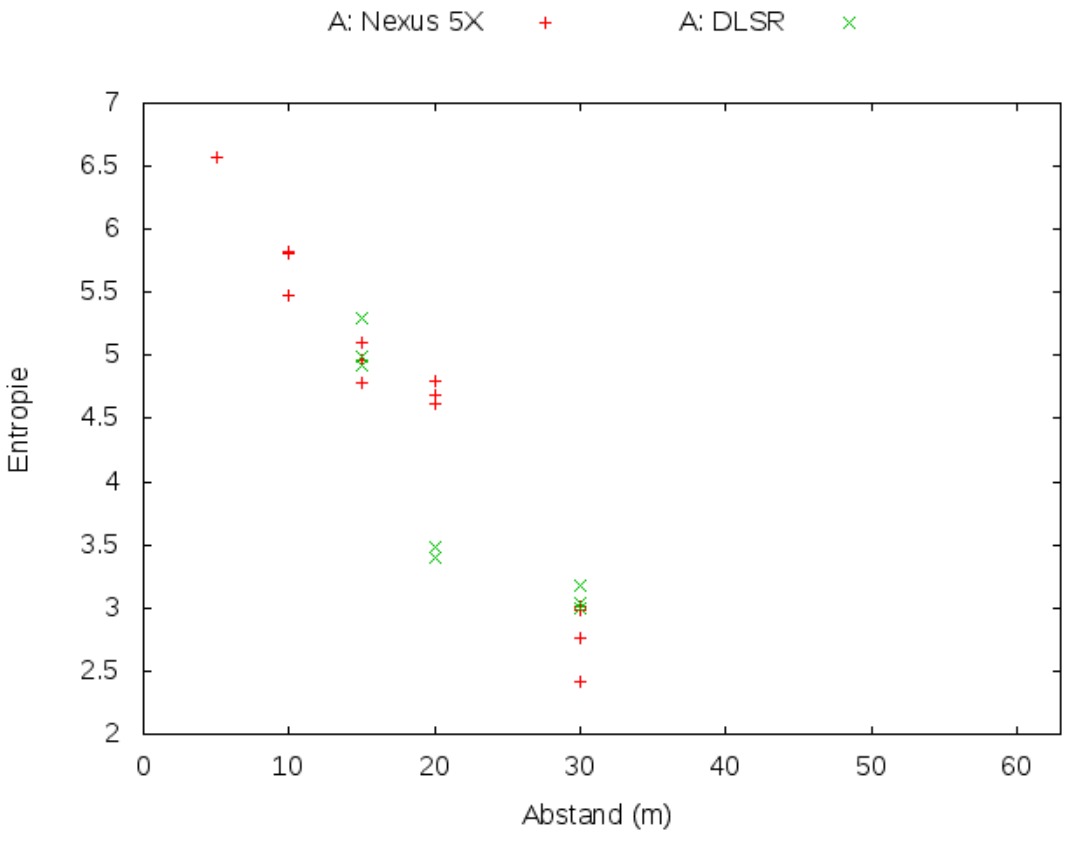


Abbildung 5.4: Entropiewerte der Fotos von Messpunkt A

Die Abnahme der Entropie mit steigender Entfernung ist deutlich erkennbar. Auffällig ist, dass dies beim N5X zwischen 20 und 30 m ein sprunghafter Abfall vorliegt, bei der Digitalkamera jedoch schon

zwischen 15 und 20 m. Pro Distanz und Gerät ist die Streuung der verschiedenen Entropien mit einem Wert von bis zu ca. 0.6 nur gering.

Als zweite Messung wurde zeitgleich auch der Received Signal Strength Indicator (RSSI) mit dem N5X aufgezeichnet. Analog zu den Fotos wurden auch hier immer drei Messungen pro Distanz getätigt. In Abbildung 5.5 sind die Messwerte grafisch dargestellt. Der RSSI wird in Dezibel Milliwatt (dBm) angegeben. Der Leistungspegel liegt jeweils unter einem Milliwatt, wodurch sich negative Werte ergeben. Je näher der dBm an Null ist, desto stärker erreicht das Signal den Empfänger. Als Signalquelle dient in diesem Versuch ein iPhone, mit welchem ein WLAN Hotspot erzeugt wurde, mit dem das N5X verbunden wurde.

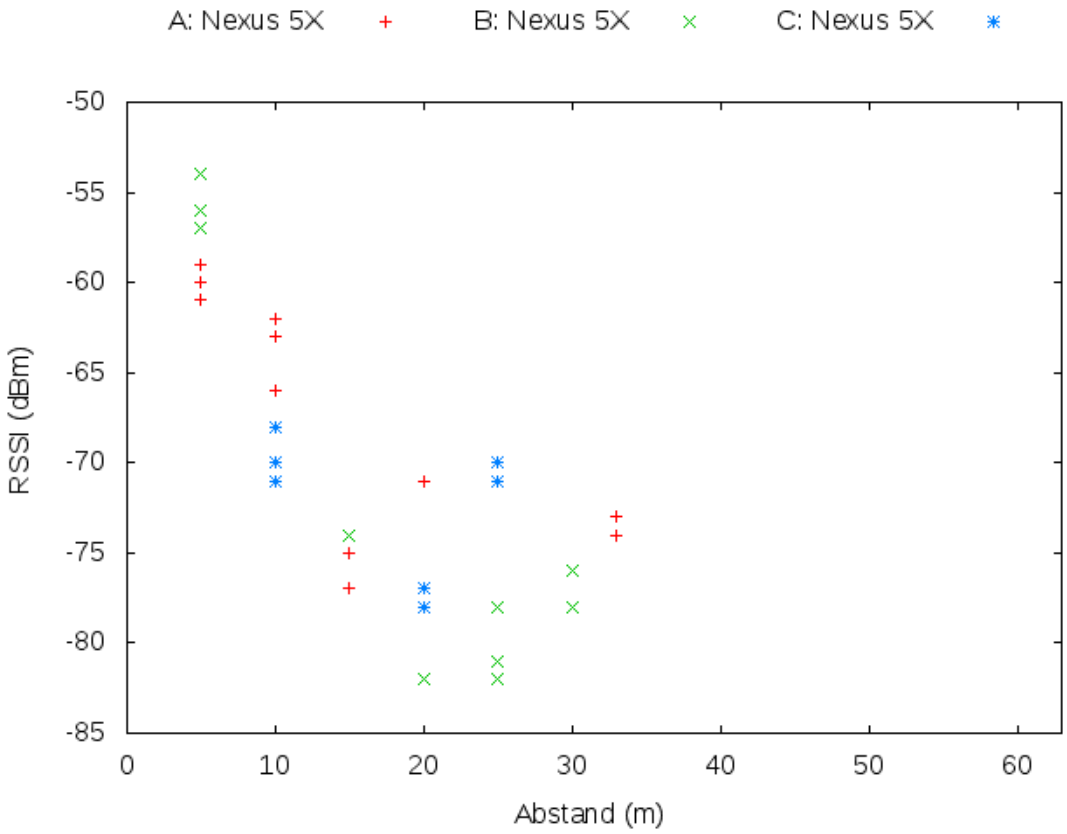


Abbildung 5.5: RSSI des N5X an den Messpunkten A, B und C

5.4.2 B - Altes Haus

Der Messort B liegt in unmittelbarer Umgebung von einem alten, verfallenen Haus. Hier wurde die Messung ebenfalls mit dem N5X und der Digitalkamera durchgeführt. Während mit dem N5X jeweils drei Fotos pro Entfernung gemacht wurden, gibt es hier von der Digitalkamera lediglich ein Foto. Die entsprechenden Entropiewerte der Ausschnitte sind in Abbildung 5.6 dargestellt. Entsprechend zum

Messort A wurde auch hier der RSSI mit dem N5X ermittelt. Dieser ist in Abbildung 5.5 dargestellt. Bei mehreren Entfernungen wurde dreimal der gleiche Wert gemessen, sodass dort nur ein einzelner Punkt in der Grafik angezeigt wird. Während für den Messort A die Messwerte grob gesehen linear abnehmen, die exponentielle Abnahme wird durch die logarithmische Skala ausgeglichen, steigen die Werte für Messort A mit steigender Distanz wieder an. Eine mögliche Erklärung hierfür kann eine zum Messzeitpunkt abnehmende Nebeldichte sein. Subjektiv konnte während der Messung keine Veränderung der Sichtweite festgestellt werden. Dem steht jedoch die sinkende Entropie der Fotoausschnitte entgegen, was für eine gleichbleibende Nebeldichte spricht.

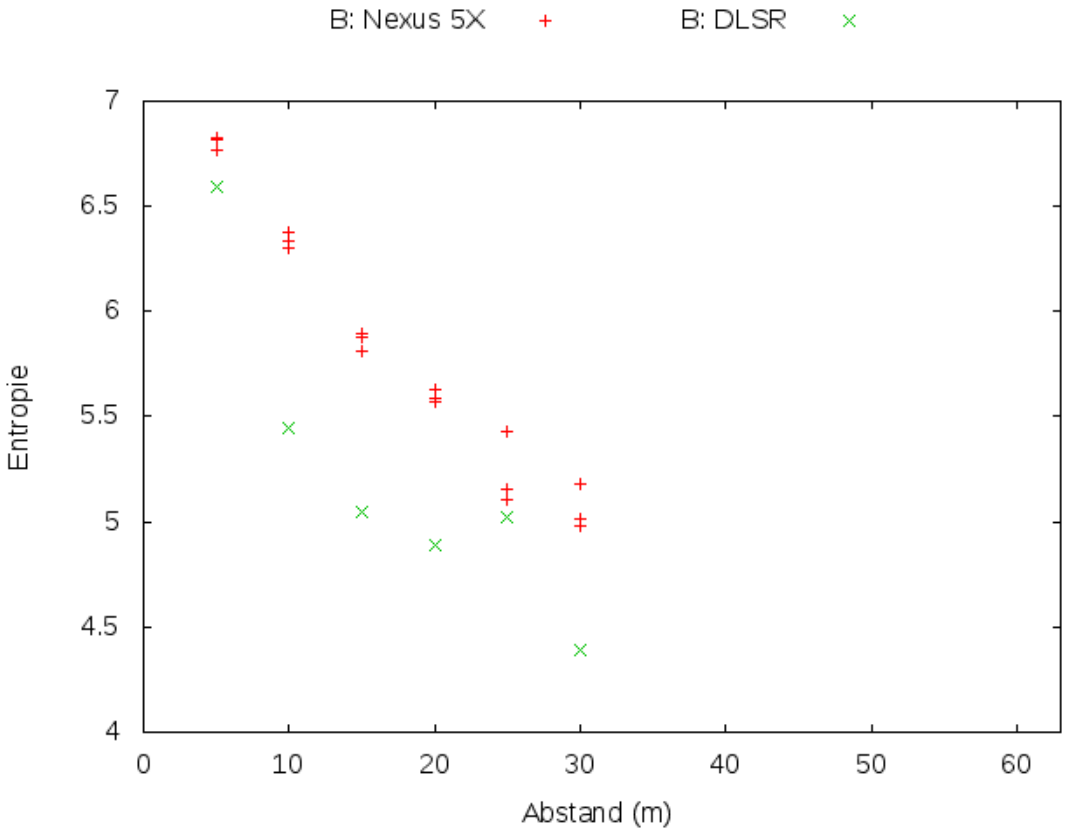


Abbildung 5.6: Entropiewerte der Fotos von Messpunkt B

5.4.3 C - Pfeiler

Der Messort C war eine Nebenstraße, welche entlang ihres Verlaufes Begrenzungspfeiler hatte, die etwa 1,4 m auseinander standen. Ein erster Messversuch, anhand der gelb lackierten Pfeiler eine Blob-Count Messung, wie in Kapitel 5.4.7 dargestellt, durchzuführen scheiterte an dem zu geringen Abstand zwischen den einzelne Pfeilern. Hier konnten die Smartphones schon weit vor Erreichen der Sichtweite die einzelnen Pfeiler nicht mehr unterscheiden.

Stattdessen wurde eine kleine Messung wie schon bei den Messorten A und B durchgeführt. An diesem Messort war die Nebeldichte windbedingt sehr wechselhaft, was die stark abweichenden Werte für die DLSR und das N5X in Abbildung 5.7 zeigen. Die in Tabelle 5.2 angegebene Sichtweite von 25 m bezieht sich auf den Zeitpunkt kurz vor den Fotos der DLSR. Die unmittelbar im Anschluss getätigten Fotos mit dem N5X deuten auf eine Erhöhung der Sichtweite hin.

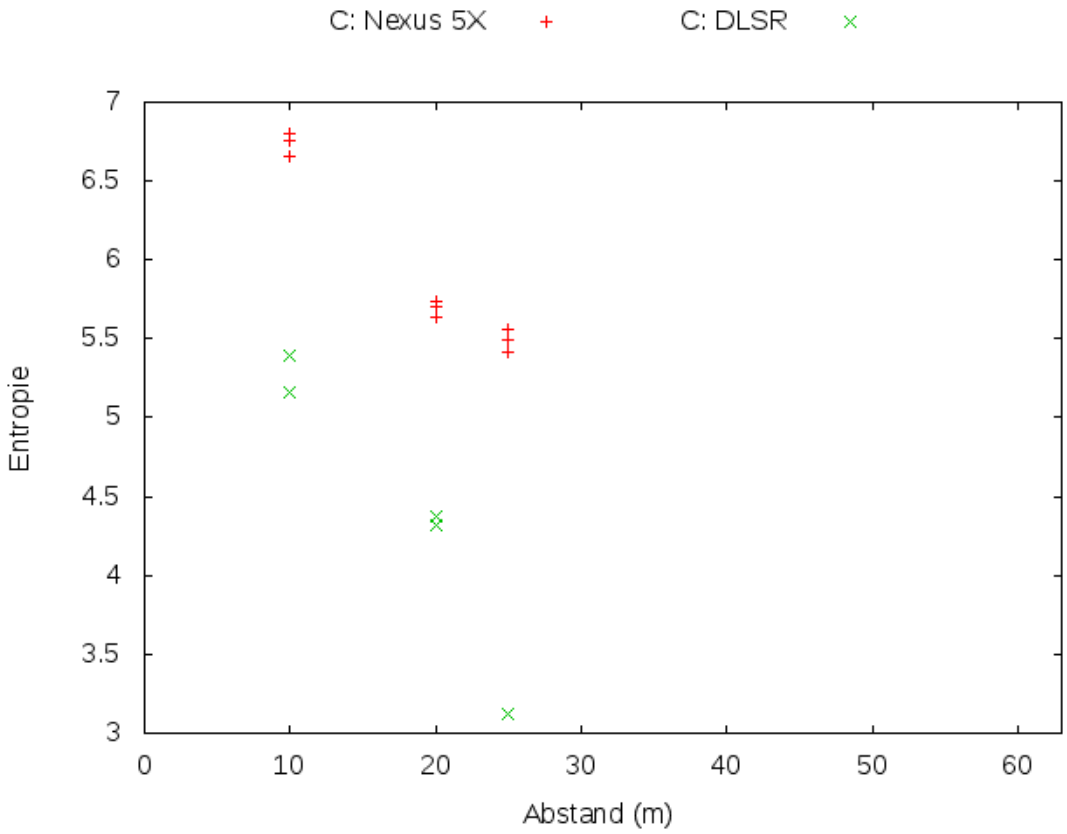


Abbildung 5.7: Entropiewerte der Fotos von Messpunkt C

5.4.4 D - Wüste

Als Vergleichsmessung wurde der Messort D einige Kilometer hinter der Bergkette in der Wüste durchgeführt. Die kahle, trockene Umgebung bildet einen starken Kontrast zu dem nur wenige Kilometer entfernten feuchten Waldgebiet. Mit mehreren Kilometern Sichtweite und trockener Luft boten sich dort gänzlich andere Bedingungen. Im Gegensatz zu den bisherigen und auch folgenden Messorten konnte hier das iPhone nicht als WLAN Hotspot verwendet werden, da diese Funktion nur möglich ist, wenn sich das Smartphone in einem Mobilfunknetz befindet. Als Ausweichgerät wurde ein Thinkpad T440p mit einer Intel Corporation Wireless 7260 (rev 83) WLAN Karte verwendet. Die Umgebungsbedingungen blieben während des Messzeitraumes unverändert. Trotz des deutlich spür-

baren Windes wurde kein sichtbarer Staub aufgewirbelt. Für die einzelnen Abstände wurden jeweils direkt hintereinander die Fotos mit allen Geräten gefertigt, bevor zum nächsten Abstand übergegangen wurde. Dieses Vorgehen ist bei konstanten Bedingungen nicht unbedingt notwendig, sorgt aber bei den folgenden Messorten für möglichst gleiche Bedingungen pro Abstand für alle Geräte. Die berechneten Entropiewerte für alle vier Smartphones und die DLSR sind in Abbildung 5.8 zusammengefasst. Auffällig ist hierbei, dass das N9 deutlich geringere Entropiewerte liefert als die anderen Geräte. Die anderen drei Geräte bilden eine recht kompakte Anhäufung mit relativ hohen Werten. Dies war zu erwarten, da die Testtafel mit ihren verschiedenen Farben eine gewisse Unordnung in den Ausschnitt bringt. Auch auf wachsende Entfernung hin bleiben die hohen Werte erhalten, da die Farben nicht wie bei den anderen Messungen durch den Nebel verblassen. Eine leichte Abnahme der Entropie bei steigender Entfernung lässt sich auch auf die kleinen Kameras zurückführen.

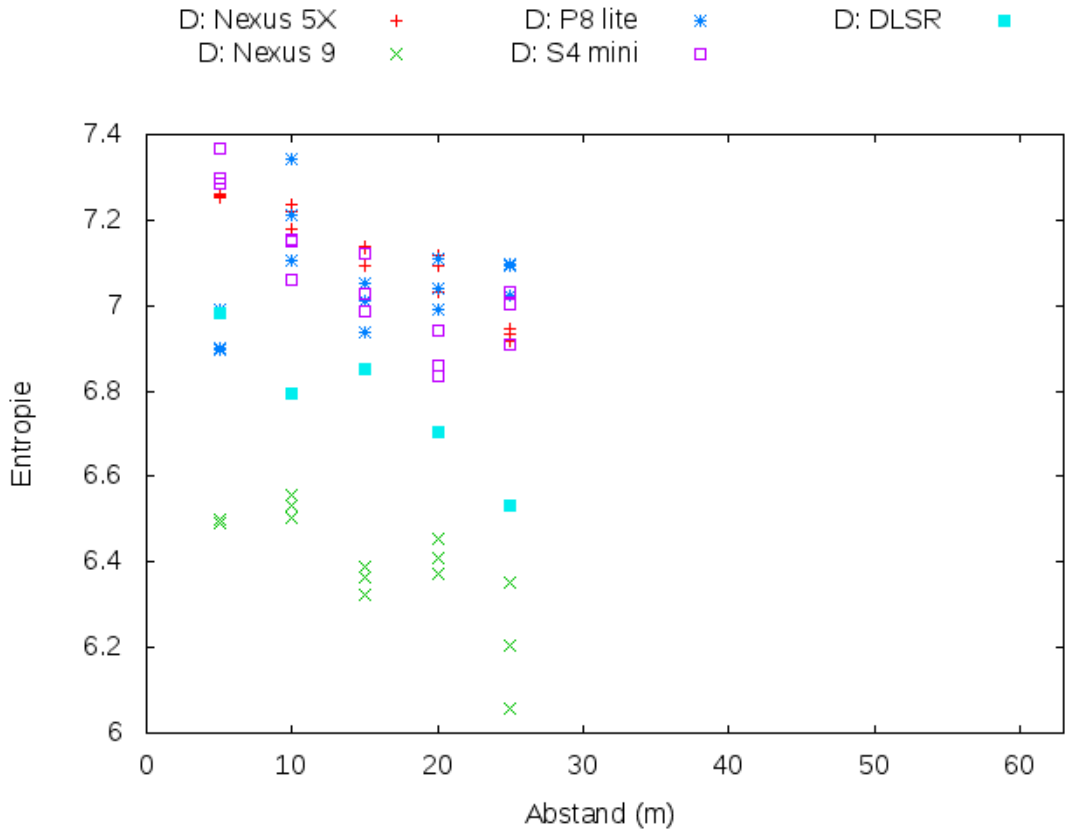


Abbildung 5.8: Entropiewerte der Fotos von Messpunkt D

Im Gegensatz zur optischen Entropie ist in Abbildung 5.9 keine ausgeprägte Struktur zu erkennen. Tendenziell sinken die Werte bei steigender Distanz, jedoch ist die Streuung der einzelnen Messungen pro Gerät und Abstand so groß, dass ein Umkehrschluss von RSSI auf einen gewissen Abstand nicht möglich ist.

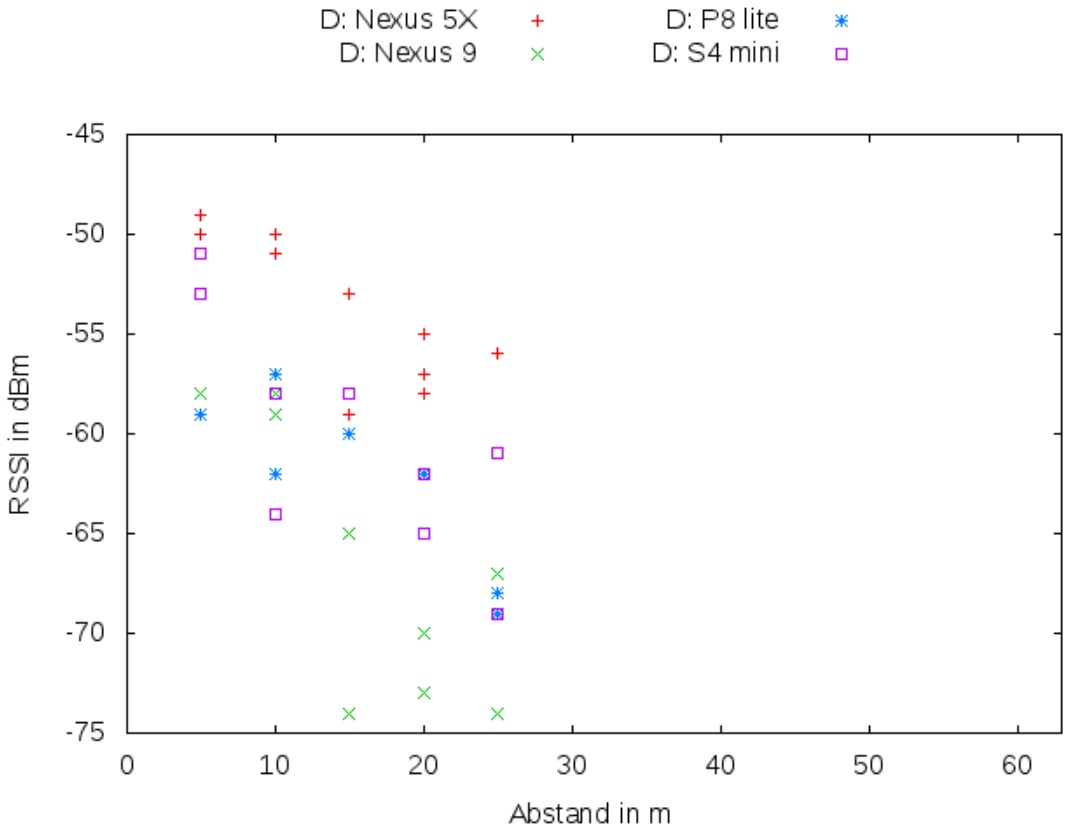


Abbildung 5.9: RSSI am Messpunkt D

5.4.5 E - Hochebene Taiq

Auf einer Hochebene in der Nähe des Taiq Sinkholes wurde die Messung E durchgeführt. Dieses Sinkhole ist als eines der größten bekannten Sinkholes eine Touristenattraktion. Hier wurde analog zu Messort D mit allen zur Verfügung stehenden Geräte sowohl die Entropie als auch der RSSI gemessen. Mit einer Sichtweite von etwa 58 m bot sich hier die größte Sichtweite aller Messungen auf der nebligen Bergseite. Der Messablauf war identisch zu dem an Messort D. Wie schon für Messort D genannt, kann auch für diesen Messort festgehalten werden, dass das N9 relativ niedrige Entropiewerte liefert. In Abbildung 5.10 liegen die Messwerte der anderen Geräte noch dichter zusammen. Lediglich bei der Messung der maximalen Sichtweite vergrößert sich die Streuung und das N9 liegt überraschender Weise nicht am unteren Rand. Es gibt keine Hinweise darauf, warum beim N9 die Entropie bei einem Abstand von 15 auf 20 m ansteigt, bei allen anderen Geräten aber abfällt. Die Fotos pro Entfernungsschritt wurden mit allen Geräten in rascher Abfolge angefertigt, sodass der Einfluss von Sichtweitenänderungen möglichst minimal war.

In Abbildung 5.11 erkennt man, dass die Werte für das N9 wieder deutlich unterhalb den Werten der anderen Geräte liegen. Bei der maximalen Sichtweite von 58 m konnte schon keine Verbindung

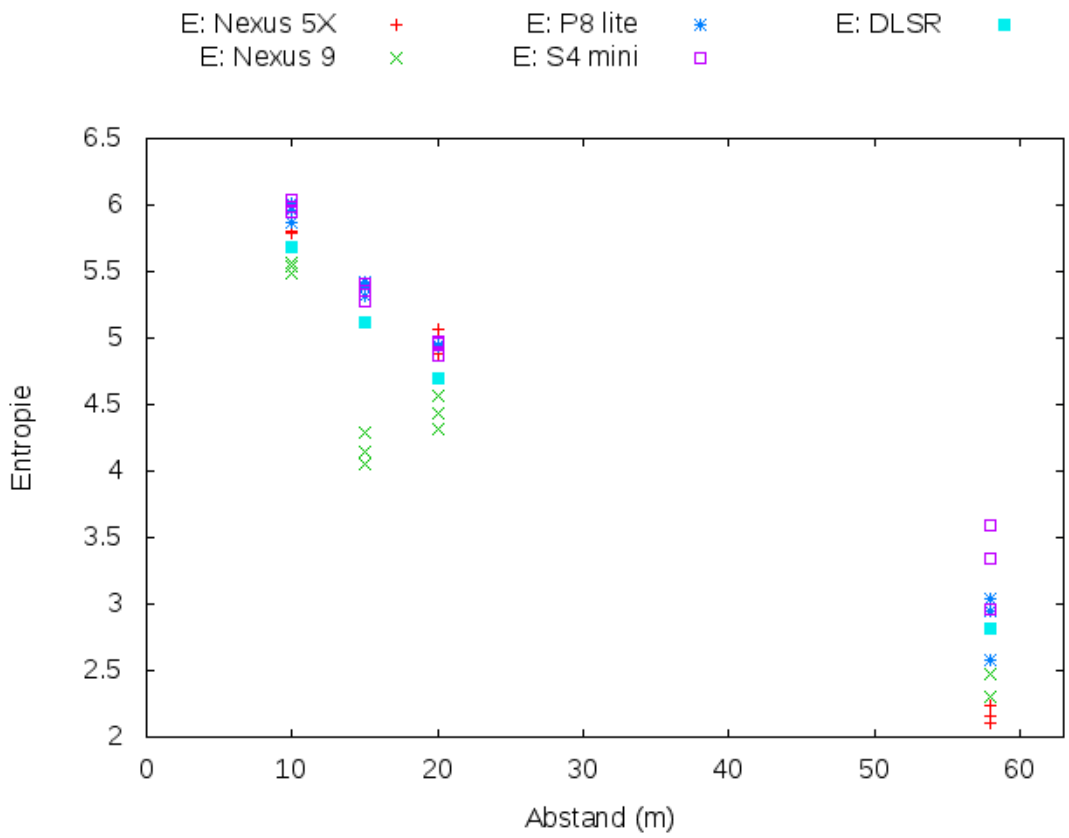


Abbildung 5.10: Entropiewerte der Fotos von Messpunkt E

mehr aufrecht erhalten werden, was zu einem Ergebnis von -127 dBm führte. Bei allen anderen Geräten blieb die Verbindung bestehen und die Signalstärke nahm nur geringfügig ab. Im Vergleich zur Referenzmessung D sind jedoch auch die Signalstärken für die kürzeren Entfernungen schwach.

5.4.6 F - Felswiese

Am Messort F, einer mit niedriger Vegetation überwucherten Felslandschaft, wurde die dritte identische Messung wie schon an den Messorten D und E durchgeführt. Abbildung 5.12 zeigt einen deutlichen Abfall der Entropie mit steigendem Abstand. Auch hier liefert das N9 anfangs wieder die niedrigsten Werte, während es bei der maximalen Sichtweite überraschend die höchste Entropie liefert. Die übrigen Geräte liefern, wie schon bei den vorangegangenen Messungen, sehr ähnliche Werte. Ebenso ist auch hier die Streuung der Werte beim größten Abstand auch am höchsten.

Bei der Messung des RSSI ist wie in Abbildung 5.13 ersichtlich die Streuung der einzelnen Geräte höher und ähnelt damit den Ergebnissen von Messort D. Überraschend ist jedoch, dass sich hier das N9 in den Wertebereich der anderen Geräte einfügt und nicht wie bei den vorangegangenen Messungen

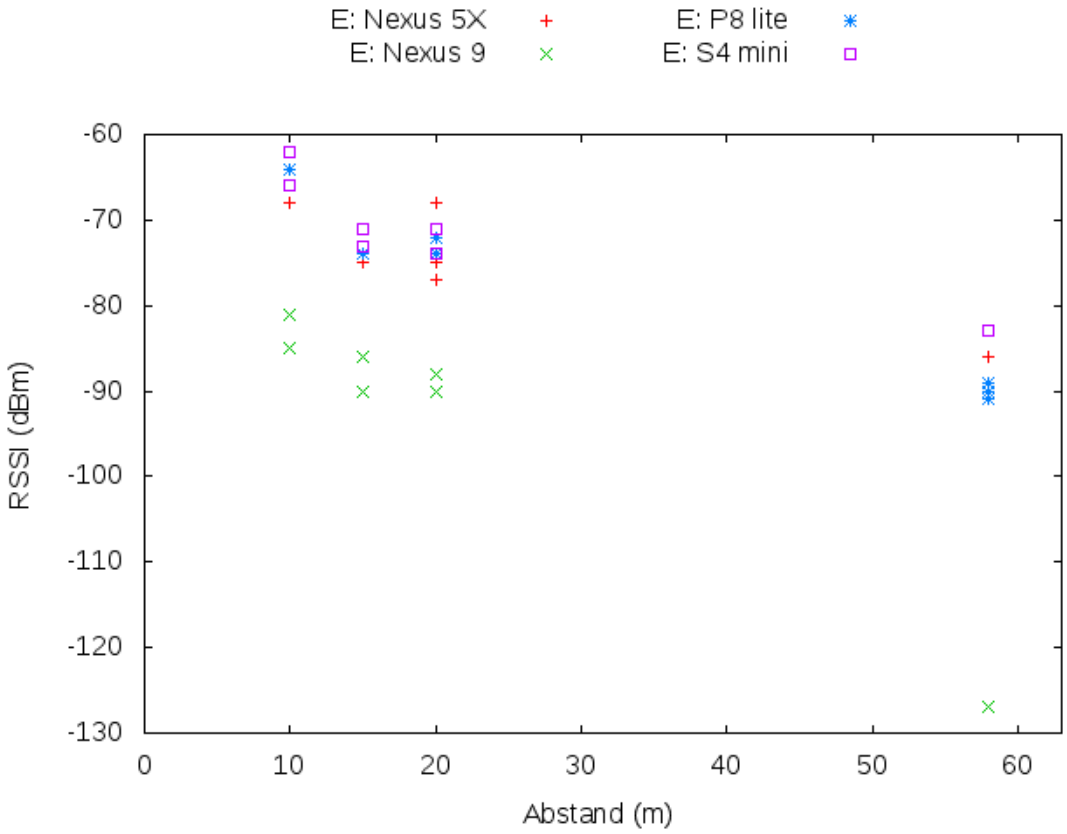


Abbildung 5.11: RSSI am Messpunkt E

einen Ausreißer darstellt.

5.4.7 G - Parkplatz

An diesem Ort wurde eine andere Messmethode ausprobiert. Es wurden mehrere Farbtafeln in einer Reihe aufgestellt. Der Abstand zum Ausgangspunkt betrug hierbei 10, 15, 20 und 25 m. Es wurden nacheinander die Farben rot, gelb, orange ausgewertet. Um zu verhindern, dass die Pappen zu schnell durchweichen, wurden diese mit Klarsichthüllen geschützt. Der kondensierte Nebel bildete sehr schnell viele kleine Wassertröpfchen auf den Folien, was zu einer teilweisen Streuung und somit zu einer optischen Verblassung der Farbtafeln führte.

Da vor Ort Probleme bei der Helligkeit der mit dem Framework gemachten Fotos auftauchten, wurden stattdessen Fotos mit der Standardkamera der jeweiligen Smartphones durchgeführt. Im Nachgang wurden diese Fotos mit dem Framework eingelesen und ausgewertet. Nachdem das Foto eingelesen wurde, wird manuell die nächstgelegene Pappe durch tippen ausgewählt. Dadurch wird der Farbwert ausgewählt, auf dessen Basis die Farbblobs gesucht werden. Diese zusammenhängenden Bereiche mit

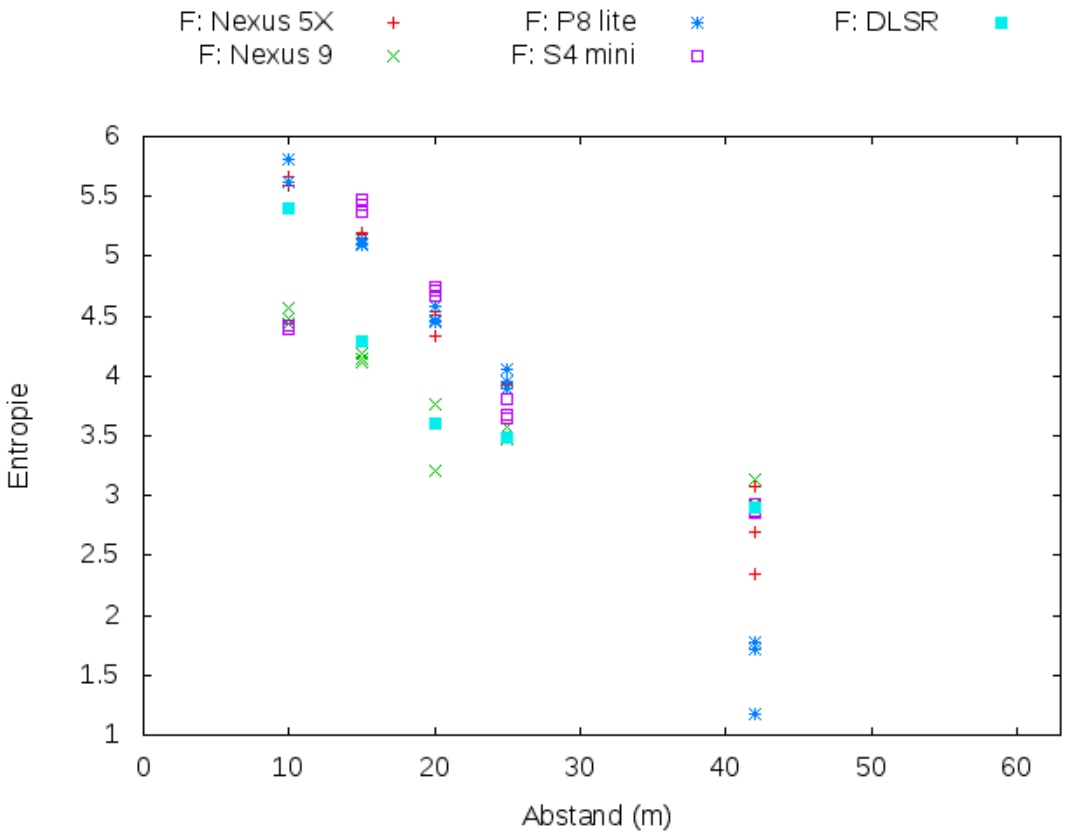


Abbildung 5.12: Entropiewerte der Fotos von Messpunkt F

ähnlichen Farbtönen werden jeweils mit einem Umriss versehen. Auf Grund des manuellen Faktors ist es möglich, dass nicht die ideale Farbe gewählt wird. Für der drei Fotos pro Smartphone wurde jeweils mehrfach die Farbe ausgewählt. Die Tabelle 5.3 zeigt für jedes Foto, wie viele der vier möglichen Farbtafeln erkannt wurden. Hierbei wurde jeweils da Optimum gewählt, bei dem die meisten Tafeln erkannt, aber auch keine sogenannten false positives erfasst wurden. False positives wären Bildbereiche, die fälschlicher Weise auf Grund ihrer ähnlichen Farbe als Zieltafel erkannt werden. Bei den gelben Tafeln ist die Erkennungsrate am geringsten, da hier sehr schnell false positives auftreten. Der helle Himmel und die gelben Tafeln wiesen sehr ähnliche Farbtöne auf. Die roten Tafeln wurden insbesondere vom N5X und dem S4 gut erkannt. Durchgehend am besten erkannt wurden die orangen Tafeln. Von ihnen wurden mindestens zwei, oftmals sogar drei Tafeln korrekt erkannt.

5.4.8 Zusammenfassung

In diesem Kapitel wurden die verschiedenen Tests, die mit dem Framework durchgeführt worden, vorgestellt. Hierbei lassen sich diese in zwei Gruppen aufteilen. Die erste Gruppe diente der Entwicklung des Frameworks und gleichzeitig zum Finden einer geeigneten Messmethode für die Nebeldichte. Der

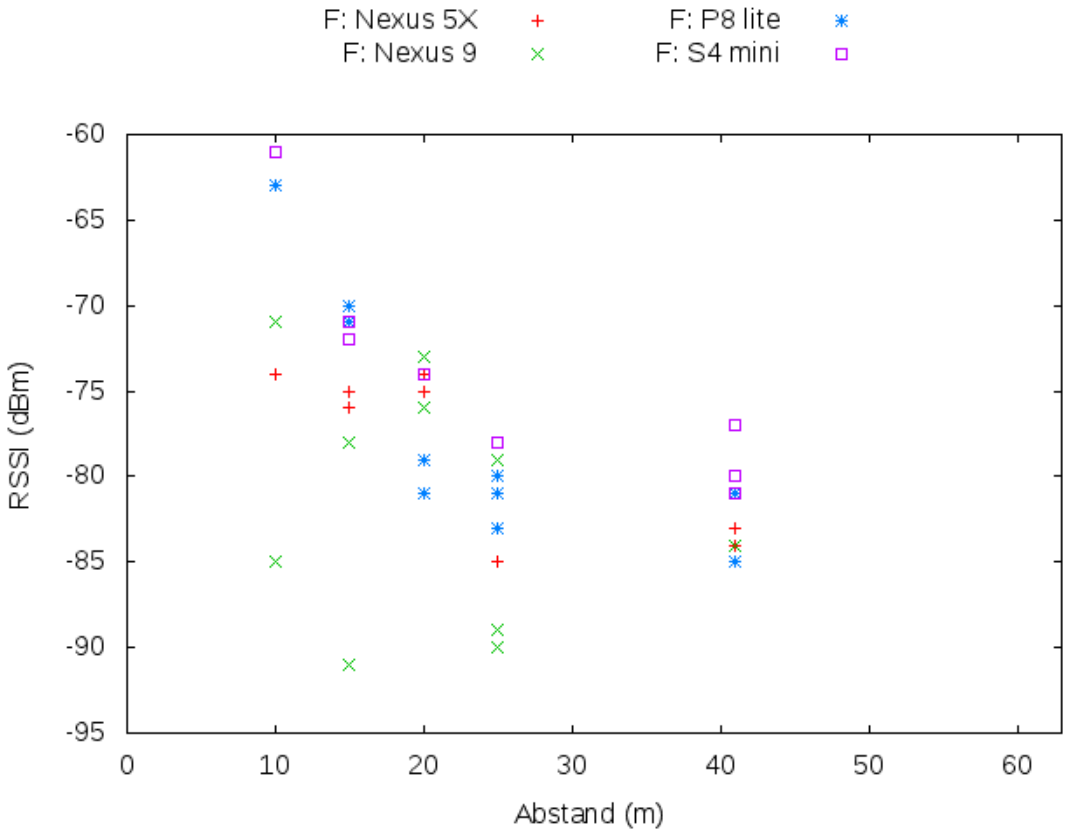


Abbildung 5.13: RSSI am Messpunkt F

Test mit den Rauchpatronen war nicht erfolgreich, währenddessen der auf Klarsichthüllen basierende Test brauchbare Ergebnisse lieferte. Die zweite Gruppe sind die im Oman durchgeführten Tests. Diese sollten die Leistungsfähigkeit des entwickelten Frameworks testen. Insgesamt gab es drei Tests im Oman, zum ersten die RSSI-Messung, zum zweiten die Entropiemessung und zuletzt die Farbblobzählung.

Tabelle 5.3: Anzahl erkannter Blobs pro Farbe und Gerät

Farbe	gelb	rot	orange
Nexus 5X	0,0,0	1,3,3	2,2,3
Nexus 9	0,0,0	1,1,1	2,2,3
P8 lite	0,0,0	1,1,1	3,3,3
S4 mini	0,0,1	2,2,2	2,2,2

5.5 Fazit

Folgend werden die Ergebnisse sowohl aus den Feldtests, als auch die vorangegangene Implementierungsphase zusammengefasst und bewertet.

5.5.1 Vorversuche

Die Versuche, die im Vorfeld der Reise durchgeführt wurden, sind teilweise nahe an die Gegebenheiten im Oman heran gekommen. So vermittelte die Klarsichthülle trotz des einfachen Versuchaufbaus eine dem dichten Nebel im Oman recht ähnliche Wirkung der optischen Beeinflussung. Mit den verschiedenen Abständen hinter der Folie konnte für die Blob-Anzahl Messung eine realitätsnahe Annäherung bewirkt werden. Für die Entropiebestimmung muss der Abstand zwischen Folie und Testtafel aber genauestens eingehalten werden, da ansonsten die Streuung zu stark wird.

Der Versuch mit den Rauchpatronen ist schon im Ansatz an der mangelnden Rauchkapazität gescheitert. Es wurde zu wenig Nebel durch die Patrone produziert. Als nächster Ansatz könnte hier eine Nebelmaschine in einem geschlossenem Raum eingesetzt werden.

5.5.2 Entropie und RSSI

Die meisten Feldtests im Oman bezogen sich auf die Entropie einer Testtafel, welche aus verschiedenen Abständen fotografiert wurde und den in gleichen Abständen gemessenen RSSI. In Abbildung 5.14 sind alle berechneten Entropiewerte der einzelnen Geräte aufgeführt. Am oberen Rand mit den höchsten Werten sind wenig überraschend die Messungen aus der Wüste zu finden. Interessant ist, dass selbst Fotos, die nur wenige Sekunden nacheinander vom gleichen Objekt gemacht wurden, deutliche Unterschiede in der Entropie aufweisen. Auf Grund der Streuungen ist es noch nicht einmal möglich, auf Grund der Messdaten Rückschlüsse auf den Messort zu erhalten. Sogar eine Unterscheidung zwischen Nebelwald und Wüste ist nicht ohne weiteres möglich, da die Werte sehr nahe zusammen liegen. Insbesondere die durchgängig niedrigen Werte des N9 tragen zu einer Verschleierung der Grenze bei. Da es kaum möglich ist, den signifikanten Unterschied zwischen keinem Nebel in der Wüste und dichtem Nebel mit wenigen Metern Sichtweite auf dem Berg auseinander zu halten, ist es für detailliertere Unterscheidungen nicht mehr möglich. Zum Einen ergeben sich durch die verschiedenen Geräte schon bauartbedingt Unterschiede, zum Anderen können zwei Fotos vom gleichen Objekt sehr unterschiedliche Entropiewerte hervorbringen. An dieser Stelle konnte kein deterministisches Verhalten festgestellt werden. Dies bedeutet, dass dieser Ansatz der entropiebasierten Messung keine relevanten Ergebnisse liefern kann. Auffällig ist auch, dass die DLSR sich gut in die Messer-

gebnisse einreiht und keine hervorstechende abweichende Resultate im Vergleich zu den Smartphones liefert.

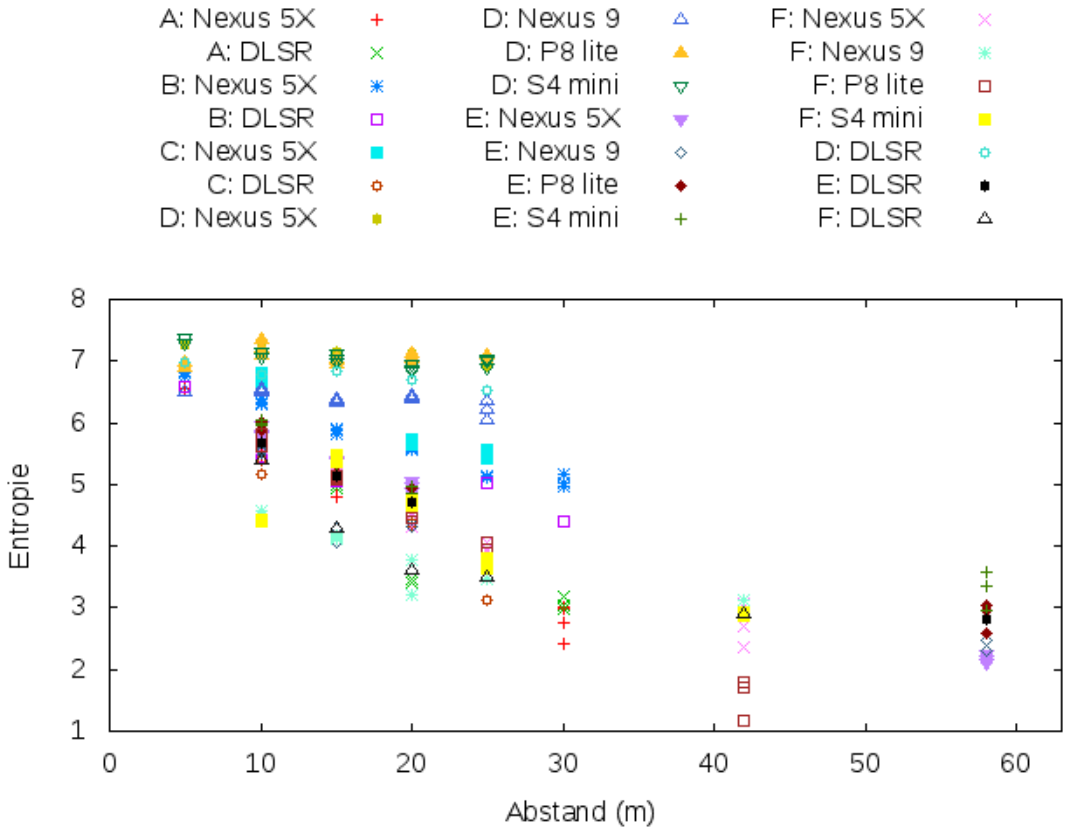


Abbildung 5.14: Entropiewerte aller Messungen

Die Auswertung aller RSSI wie sie in Abbildung 5.15 dargestellt sind, zeigt ähnlich zur Entropiebestimmung keine signifikanten Unterschiede zwischen den Messungen in der Wüste und denen im Nebel. Lediglich lässt sich festhalten, dass der Empfang ohne Nebel tendenziell besser ist, als ohne. Jedoch überlappen sich die gemessenen Werte, sodass keine eindeutige Grenze gezogen werden kann. Auch wenn man die Unterschiede zwischen den einzelnen Abständen anguckt, lässt sich nur jeweils ein sehr geringer Unterschied feststellen. Dieser wird jedoch durch die Streuung der mehrfachen Messungen so stark aufgeweicht, dass eine Unterscheidung kaum möglich ist. Lediglich eine interessante Eigenschaft konnte festgestellt werden. Über alle Messorte und Geräte hinweg ist der RSSI bei einem Abstand von 15 m zwischen Sender und Empfänger überwiegend niedriger als die beiden benachbarten Abständen von 10 und 20 m.

Zusammenfassend lässt sich somit sagen, dass beide Ansätze in ihrer derzeitigen Form nicht geeignet sind, die Nebeldichte zu bestimmen. Es konnte gezeigt werden, dass der Nebel einen gewissen Einfluss auf die Messungen der Entropie hat. Dies ist aber nur für Regionen möglich, in denen die Sichtweiten so gering sind. Auch bei diesen Entfernungen war die Größe des Testbildes auf den Fotos

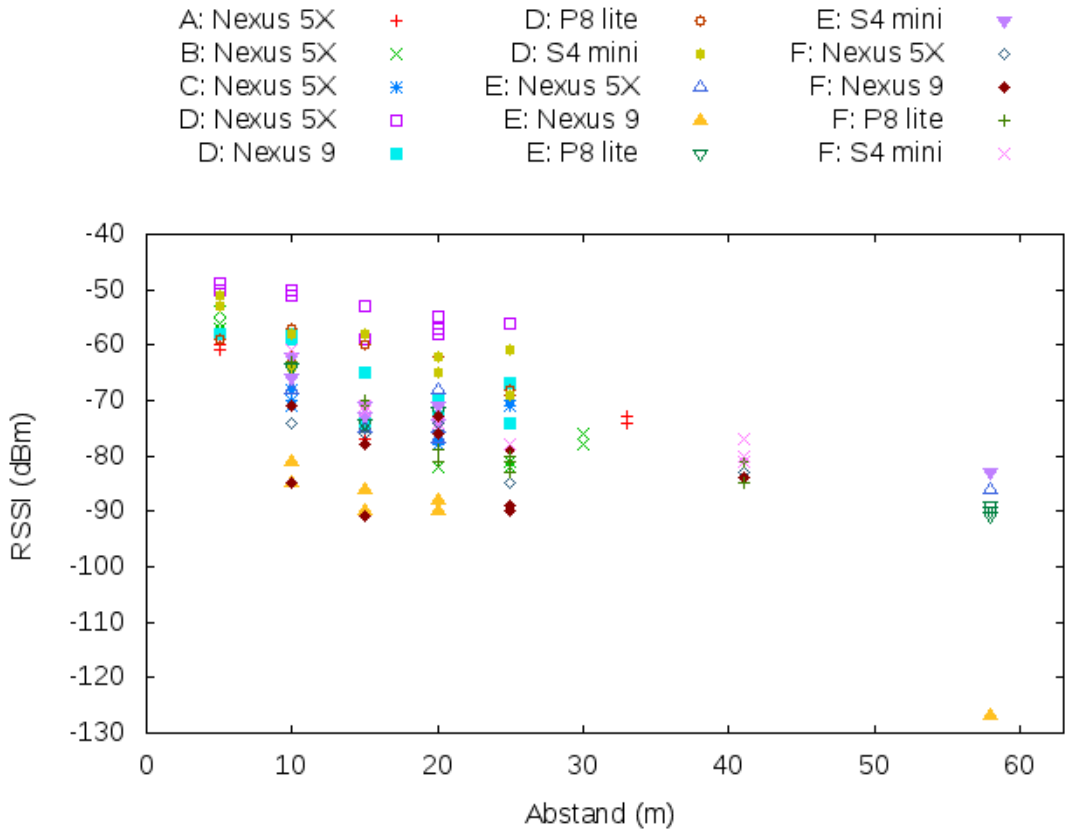


Abbildung 5.15: RSSI aller Messungen

mit teilweise nur noch 18x25 Pixeln sehr klein geworden. Da bei größeren Entfernungen die Größe der Abbildung noch weiter abnehmen wird, kommt relativ bald die Grenze des per Smartphone beobachtbaren Bereiches.

5.5.3 Farbtafeln

Die Messmethode mit den Farbtafeln ist in der verwendeten Konfiguration mit vier Farbtafeln diejenige mit der gröbsten Einordnung. Obwohl die maximale Sichtweite mit 30 m nur knapp hinter der entferntesten Farbtafel (25 m Abstand) liegt, kann noch die dritte Tafel bei oranger Bestückung erkannt werden. Bei klarer Sicht ist es kein Problem, die vierte Tafel zu erkennen. Das bedeutet, dass trotz eines starken Nebels, nur eine Tafel nicht erkannt werden konnte. Dieser Unterschied ist nur gering. Im Vergleich zu den beiden anderen Farben, rot und gelb, wurden mit orange die besten Ergebnisse erzielt.

Der Abstand zwischen den einzelnen Tafeln und der Abstand zur Kamera wurde anscheinend zu gering gewählt. Jedoch führt ein größerer Abstand zu ungenaueren Messungen, da durch diese Messung

lediglich die Aussage getroffen werden “Die Sichtweite liegt zwischen x und y Meter”, wobei x der Abstand der letzten sichtbare und y der Abstand der ersten nicht mehr sichtbaren Tafel ist. Trotzdem ist dies die einzige Messung, bei der eine eindeutige Aussage getroffen werden kann, dass bei dieser Messung starker Nebel vorhanden war.

5.5.4 Zusammenfassung

In dieser Arbeit wurde ein Sensorframework erstellt, mit dem neben den Standardsensoren weitere Basis- oder Kompositsensoren selbst erstellt werden können. Die Anbindung aller Sensoren erfolgt sehr ähnlich, wenn auch nicht vollkommen gleich, da an dieser Stelle ohne tiefgreifende Eingriffe in das Betriebssystem keine Möglichkeiten bestehen. Neben der Entwicklung und Implementierung des Frameworks wurden auch verschiedene Messmethoden für die Nebeldichtemessung entwickelt und getestet. Bei den Entropie- und RSSI-basierten Ansätzen konnte keine relevante Aussage über die Nebeldichte getroffen werden. Der dritte Ansatz, basierend auf mehreren Farbtafeln, konnte zumindest eine grobe Einordnung der Nebeldichte liefern. Insbesondere dieser Ansatz sollte weiter verfolgt werden.

Kapitel 6

Künftige Arbeiten

Auf Grund der begrenzten Bearbeitungszeit sind noch Fragen offen geblieben. Einige der Möglichkeiten diese Thema weiterhin zu untersuchen sind folgend genannt. Diese Auflistung ist nicht abschließend.

6.1 Nebelsimulation

Da der Rauchpatronenansatz mangels ausreichendem Rauch gescheitert ist, sollten künftige Versuche größer aufgezogen werden. So bietet sich ein geschlossener Raum an, in dem sich der Rauch nicht so schnell verzieht und auch die Luftbewegungen auf ein Minimum reduziert werden. Da der Rauch der Rauchpatrone jedoch nicht gesundheitsförderlich ist, sollte stattdessen auf eine Nebelmaschine zurückgegriffen werden, wie sie auch im Eventbereich verwendet werden. Basierend auf den Fotos, die von den Messorten gemacht wurden, besteht somit die Möglichkeit, die künstliche Nebeldichte sehr realitätsnah für weitere Experimente einzustellen.

6.2 Farbkorrekturen mit OpenCV

OpenCV bietet eine sehr große Bandbreite an Bildverarbeitungsfunktionen. Da die in den Smartphones verbauten Kameras auf Grund ihrer kompakten Bauform oftmals einige Abweichungen im Farbraum haben, kann es sinnvoll sein, dass die Kameras kalibriert werden. Dadurch würden auch Auswertungen, die auf die Farben der Fotos oder Filme eingehen, genauer durchgeführt werden können.

6.3 Feinabstimmung Farbtafelmessung

Bei der Messmethode basierend auf den Farbtafeln kann eine genauere Abstimmung der Abstände zwischen den Tafeln auf die aktuelle Sichtweite vorgenommen werden. Eventuell können zur Erhöhung der Auflösung auch mehr Tafeln eingesetzt werden. Jedoch muss hierbei darauf geachtet werden, dass die Tafeln auf dem Foto gut auseinander zu halten sein müssen. Des Weiteren kann noch untersucht werden, wie sich verschiedene Orangetöne auf die Erkennbarkeit auswirken. Bei vorher bekannter Farbe kann zudem der bisherige semiautomatische Ansatz in einen vollautomatischen überführt werden, sodass die Auswahl der Zielfarbe durch den Benutzer nicht mehr notwendig ist.

6.4 Hardware ausweiten

Da die in Android Smartphones verbauten Sensoren nicht auf die Datensammlung für wissenschaftliche Studien ausgelegt sind, ist es vorstellbar, dass zum Beispiel Raspberry Pis als Sensorposten verwendet werden. Entsprechende Portierungen, die eine Ausführung von Android auf einem Raspberry Pi ermöglichen sind schon einige Zeit verfügbar. Mit Sensoren, die über die USB-Ports oder über die General Purpose Input Output (GPIO) Pins angeschlossen werden, können genauere Sensoren verbaut werden. Wenn die einzelnen Geräte dann nahe genug zusammen platziert werden, dann kann mit dem Framework ein Sensorennetzwerk erzeugt werden.

Literaturverzeichnis

[Abo16] *About OpenCV*. `http://opencv.org/about.html`. Version: 2016. Auf CD als AboutOpenCV.pdf

[Aco16] *AcousticRouler (Deutsch)*. `https://iqtainment.wordpress.com/acoustic-ruler-de/`. Version: 2016. Auf CD als acousticRouler.pdf

[And16a] *Android Dashboard Platform Versions*. `https://developer.android.com/about/dashboards/index.html#Platform`. Version: July 2016. Auf CD als AndroidDeveloper-OS-Versionen.pdf

[And16b] *Android Dashboard Sensors Overview*. `https://developer.android.com/guide/topics/sensors/sensors_overview.html`. Version: July 2016. Auf CD als AndroidDeveloper-SensorenOverview.pdf

[BSC+12] BRUNETTE, Waylon; SODT, Rita; CHAUDHR, Rohit; GOEL, Mayank; FALCONE, Michael; VANORDEN, Jaylen; BORRIELLO, Gaetano: Open Data Kit Sensors: A Sensor Integration Framework for Android at the Application-Level. In: *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, 2012 (MobiSys '12), S. 351–364.

[DMP+10] DAS, Tathagata; MOHAN, Prashanth; PADMANABHAN, Venkata N.; RAMJEE, Ramachandran; SHARMA, Asankhaya: PRISM: Platform for Remote Sensing Using Smartphones. In: *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*. New York, NY, USA : ACM, 2010 (MobiSys '10). ISBN 978–1–60558–985–5, 63–76.

[GLLDR11] GOUGH LUI, Thomas G.; LI, Binghao; DEMPSTER, Andrew G.; RIZOS, Chris: Differences in RSSI readings made by different Wi-Fi chipsets: A limitation of WLAN localization. In: *International Conference on Localization and GNSS (ICL-GNSS)*. Tampere, June 2011, S. 53–57.

[HTLA06] HAUTIÉRE, Nicolas; TAREL, Jean-Philippe; LAVENANT, Jean; AUBERT, Didier: Automatic fog detection and estimation of visibility distance through use of an onboard camera. In: *Machine Vision and Applications* 17 (2006), Nr. 1, 8–20. http://dx.doi.org/10.1007/s00138-005-0011-1. DOI 10.1007/s00138–005–0011–1. ISSN 1432–1769.

[IG15] IPPISCH, Andre; GRAFFI, Kalman: An Android Framework for Opportunistic Wireless Mesh Networking. In: *NetSys'15: Proceedings of the Conference on Networked Systems*, 2015, S. 1–2.

[JL10] J. LANGER, M.Roland: *Anwendungen und Technik von Near Field Communication (NFC)*. 2nd. Springer, 2010

[log16] *Logback-android*. https://github.com/tony19/logback-android. Version: 2016. Auf CD als logbackGitHub.pdf

[Nex16a] *Google Nexus 5X*. http://www.phonearena.com/phones/Google-Nexus-5X_id9593. Version: 2016. Auf CD als Nexus5xPhonearea.pdf

[Nex16b] *Google Nexus 9*. http://www.phonearena.com/phones/Google-Nexus-9_id8926. Version: 2016. Auf CD als Nexus9Phonearea.pdf

[Oro16] *Orographischer Nebel*. https://www.dwd.de/DE/service/lexikon/Functions/glossar.html?nn=103346&lv2=101946&lv3=101980. Version: 2016. Auf CD als dwdOrographischerNebel.pdf

[P8l16] *Huawei P8 lite*. http://www.phonearena.com/phones/Huawei-P8-lite_id9358. Version: 2016. Auf CD als P8litePhonearea.pdf

[RHYK+10] R. HELGASON Ólafur; YAVUZ, Emre A.; KOUYOUMDJIEVA, Sylvia T.; PAJEVIC, Ljubica; KARLSSON, Gunnar: A Mobile Peer-to-Peer System for Opportunistic Content-Centric Networking. In: *ACM SIGCOMM*, 2010, S. 21–26.

[S4S16] *Samsung Galaxy S4 mini*. http://www.phonearena.com/phones/Samsung-Galaxy-S4-mini_id7788. Version: 2016. Auf CD als S4miniPhonearea.pdf

[SWR06] *Wie entsteht Nebel?* http://www.swr.de/odysso/alltagswissen-wie-entsteht-nebel/-/id=1046894/did=

2257708/nid=1046894/1mwk8r9/index.html. Version: Oktober 2006.
Auf CD als SWR-WieEntstehtNebel.pdf

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Düsseldorf, 22.August 2016 Christoph Claßen

Diese CD enthält:

- eine *pdf*-Version der vorliegenden Masterarbeit
- die LaTeX- und Grafik-Quelldateien der vorliegenden Arbeit samt aller verwendeten Skripte
- die Quelldateien des im Rahmen der Masterarbeit erstellten Sensoren Frameworks
- die zur Auswertung verwendeten Fotos und Datensätze
- die Websites der verwendeten Internetquellen