

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Faruk Uras

Automatische Wolkenanalyse und Wolkendetektion
unter Verwendung einer Fisheye-Kamera

Faruk Uras

Automatische Wolkenanalyse und
Wolkendetektion unter Verwendung einer
Fisheye-Kamera

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Informations- und Elektrotechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Hans-Jürgen Hotop
Zweitgutachter : Prof. Dr.-Ing. Günter Wüsthube

Abgegeben am 03. Juli 2015

Faruk Uras

Thema der Bachelorthesis

Automatische Wolkenanalyse und Wolkendetektion unter Verwendung einer Fisheye-Kamera

Stichworte

Wolken, Wolkenerkennung, Wolken-Bedeckungsgrad, Fisheye-Kamera, Lichtstreuung, Bildverarbeitung, OpenCV, Kamerakalibrierung, Bildentzerrung

Kurzzusammenfassung

In der vorliegenden Arbeit wird eine Software entwickelt, die mithilfe einer Fisheye-Kamera eine Analyse und Detektion der Wolken vornimmt. Die Implementierung erfolgt unter der Nutzung der Programmbibliothek OpenCV. Außerdem wird untersucht, welchen Einfluss eine Bildkorrektur auf die Wolkenerkennung hat.

Faruk Uras

Title of the paper

Automatic cloud analysis and cloud detection by using a fisheye camera

Keywords

clouds, cloud detection, fractional sky cover, fisheye camera, light scattering, image processing, OpenCV, camera calibration, image distortion correction

Abstract

In this thesis a software is developed, which can analyse and detect clouds by using a fisheye camera. The implementation occurs by using the program library OpenCV. In addition, it is investigated which effect an image correction has on clouds detection.

„Das Auge war das Organ, womit ich die Welt faßte.“

- Johann Wolfgang von Goethe -

Danksagung

An dieser Stelle möchte ich mich bei Herrn Prof. Dr. Hans-Jürgen Hotop bedanken für die Bereitstellung des Themas und für die umfassende Betreuung meiner Bachelorarbeit. Ebenso bedanken möchte ich mich beim Herrn Prof. Dr.-Ing. Günter Wüsthube, der sich bereit erklärt hat, bei dieser Arbeit das Zweitgutachten zu übernehmen.

Ein weiterer Dank geht an meinen Vorgesetzten Wolfgang Rittner. Er gab mir die Möglichkeit, dass ich meine Arbeit in der Firma Zodiac Aerospace anfertigen konnte. Zudem gilt mein Dank auch meinem Arbeitskollegen Rüdiger Meckes und Romain Ducos, die mich mit guten Ratschlägen ermutigen konnten.

Ebenfalls geht ein Dank an Fatma Ari und Mustafa Kuzgun, die sich die Mühe gemacht haben, diese Arbeit Korrektur zu lesen und mir somit eine große Hilfe bei der Fertigstellung waren. Außerdem möchte ich mich bei meinen Kommilitonen bedanken, die mich moralisch unterstützt und immerzu motiviert haben.

Abschließend möchte ich mich bei meiner Mutter bedanken, ohne die ich im Leben nicht so weit gekommen wäre.

Inhaltsverzeichnis

Tabellenverzeichnis	7
Abbildungsverzeichnis	8
1 Einleitung.....	10
2 Grundlagen.....	12
2.1 Fisheye-Kamera.....	12
2.1.1 CCD- und CMOS-Bildsensor-Technologien	14
2.1.2 Technische Eigenschaften der eingesetzten Kamera.....	19
2.2 OpenCV	20
2.2.1 Was bietet OpenCV an?.....	21
2.2.2 Wichtige Datentypen und Funktionen.....	22
2.3 Lichtstreuung	24
2.3.1 Rayleigh-Streuung.....	25
2.3.2 Mie-Streuung	26
3 Wolken	28
3.1.1 Helligkeit und Farben der Wolken.....	28
3.1.2 Wolkenklassifikation und ihre Wetterbedeutung	29
4 Wolkenanalyse und Wolkendetektion	34
4.1 Anforderungsbeschreibung	34
4.2 Methoden zur Wolkenanalyse und Wolkendetektion	36
4.3 Fehleranalyse	41
5 Bildspeicherung	45
6 Kamerakalibrierung und Bildverzerrung.....	50
6.1 Das OpenCV Kameramodell.....	50
6.1.1 Kalibrierung mit OpenCV	54

6.1.2	Bildentzerrung mit OpenCV	56
6.2	Das Kameramodell von Scaramuzza	57
6.2.1	Kalibrierung mit Scaramuzza	60
6.2.2	Bildentzerrung mit Scaramuzza	63
7	Analyse und Verarbeitung von Bilddaten	67
7.1	Maskierung	67
7.2	Filterung	68
7.3	Wolkenerkennung und Unterscheidung zwischen hellen und dunklen Wolken.....	70
7.4	Berechnung von Wolken-Bedeckungsgrad	75
7.4.1	Wolken-Bedeckungsgrad für einzelne Zonen	75
7.4.2	Gesamt-Wolken-Bedeckungsgrad	78
7.5	Programmablauf	79
7.6	Ergebnisse	80
8	Schlussfolgerungen.....	85
8.1	Zusammenfassung.....	85
8.2	Ausblick.....	87
	Literaturverzeichnis	89
	Anhang.....	93

Tabellenverzeichnis

Tabelle 2.1 Gegenüberstellung von CCD- und CMOS-Bildsensor [5] [6, p. 36]	18
Tabelle 3.1 Wolkenebenen nach WMO [19] [20, pp. 31-35].....	30
Tabelle 4.1 Wolkenklassen nach Heinle et al. (2010).	39

Abbildungsverzeichnis

Abbildung 2.1 Der Bildwinkel, ausgehend von der Bildmitte	13
Abbildung 2.2 Entstehung tonnenförmiger Verzeichnung [2]	13
Abbildung 2.3 Mosaikfilter [3]	14
Abbildung 2.4 Die Funktion des Mosaikfilters [3].....	15
Abbildung 2.5 RGB-Farbraum [4].....	15
Abbildung 2.6 Schema des CCD-Sensors mit dem Anschluss	17
Abbildung 2.7 Schema des CMOS-Sensors mit.....	17
Abbildung 2.8 ICA-8350 Fisheye-Kamera [7, p. 1].....	20
Abbildung 2.9 Basisstruktur der OpenCV [10, p. 13].....	21
Abbildung 2.10 Schematische Darstellung der Mie- (schwarz)	26
Abbildung 3.1 Schematische Darstellung der Wolkengattungen.....	31
Abbildung 4.1 Die Fisheye-Kamera befindet sich auf dem.....	34
Abbildung 4.2 Beispiel: Klassifizierung nach k-nächster-Nachbarn.....	40
Abbildung 4.3 Fehlinterpretation des Wolken-Bedeckungsgrads	42
Abbildung 4.4 Ursachen für eine Fehlinterpretation der Bildpunkte;	43
Abbildung 4.5 Ein Vogel, der die Sicht der Kamera beeinträchtigt.....	44
Abbildung 5.1 Die Klasse Downloader. Sie ist für Bild importieren	46
Abbildung 5.2 Aktivitätsdiagramm zur Methode download();	47
Abbildung 5.3 Sequenzdiagramm zur Bildspeicherung für.....	48
Abbildung 5.4 Der Testfall.....	49
Abbildung 6.1 Abbildung eines Objektpunkts Q im	50
Abbildung 6.2 Abbildung eines Objektpunkts Q im	52
Abbildung 6.3 Aufnahmen des Schachbrettmusters für die Berechnung	54
Abbildung 6.4 Das Ergebnis der Bildentzerrung mit OpenCV	57
Abbildung 6.5 Das Scaramuzza Kameramodell [32]	58
Abbildung 6.6 Bildkoordinaten, beginnend in der Bildmitte (u, v)	60
Abbildung 6.7 Funktion $f(\rho)$ für die Vorwärtsprojektion in	61
Abbildung 6.8 Bildwinkel, in Abhängigkeit von Distanz ρ	61
Abbildung 6.9 Das Ergebnis der Bildentzerrung mit Scaramuzza, $SF = 7$	63
Abbildung 6.10 Das Ergebnis der Bildentzerrung mit Scaramuzza, $SF = 11$	64

Abbildung 6.11 Aufnahme vom 10.12.2014 14:04 Uhr (teils bewölkerter Himmel) ..	65
Abbildung 6.12 Entzerrung der Aufnahme vom 10.12.2014 14:04 Uhr	65
Abbildung 7.1 Maskierung von Fisheye-Aufnahmen	67
Abbildung 7.2 Das Ergebnis der Filterung; Aufnahme vom 10.12.2014.....	69
Abbildung 7.3 Aufnahmen zur Bestimmung des Schwellenwerts.....	70
Abbildung 7.4 Häufigkeitsverteilung des $V_{n, m}$ bei komplett bewölkten	71
Abbildung 7.5 Das Ergebnis der Wolkenerkennung mit $T = 0,9$	72
Abbildung 7.6 Vergleich der Schwellenwerte, Aufnahme vom 17.11.2014	73
Abbildung 7.7 Detektion der dunklen Wolken, Aufnahme vom	74
Abbildung 7.8 Die Aufteilung in sechs Zonen (Grün); Die analysierte	76
Abbildung 7.9 Bestimmung einer Zone. Hierbei wird die Ringfläche	77
Abbildung 7.10 Ausgabe der Wolken-Bedeckungsgrad für einzelnen Zonen	77
Abbildung 7.11 Endresultat einer Analyse und.....	78
Abbildung 7.12 Die Klasse Cloud. Sie ist für die Wolkenanalyse	79
Abbildung 7.13 Aufnahme vom 30.05.2015 10:22 Uhr.....	80
Abbildung 7.14 Aufnahme vom 11.12.2014 11:22 Uhr.....	81
Abbildung 7.15 Aufnahme vom 15.06.2015 12:56 Uhr.....	82
Abbildung 7.16 Aufnahme vom 11.12.2014 11:22 Uhr.....	82
Abbildung 7.17 Aufnahme vom 10.12.2014 16:18 Uhr.....	83
Abbildung 7.18 Ein äußerst gutes Ergebnis für eine Analyse	84

1 Einleitung

“Fast so gut wie das menschliche Auge: Maschinen lernen sehen.” [1]

Das Fraunhofer-Institut hat eine Software entwickelt, die der Funktion eines menschlichen Auges entspricht. Hierbei arbeitet das System mit Stereofarbkameras. Die Haupttätigkeit übernimmt eine leistungsfähige Software. Mittlerweile können die Routen von Objekten im Bereich der Sicherheitstechnik und Logistik beobachtet werden. Dabei wird der ganze Prozess überwacht, so dass Kollisionen verhindert werden können. Das Programm stellt nicht nur fest, wo sich ein bestimmtes Objekt aufhält, vielmehr kann es auch eine Klassifizierung vornehmen [1]. Eine Funktion dieser Art macht das System intelligent und selbständig. Ein wichtiger Punkt hierbei ist, dass diese Vorgehensweise überall eingesetzt werden kann.

Auch im Bereich der erneuerbaren Energien sind diese intelligenten Systeme einsetzbar. Eine automatische Wolkendetektion kann den Solaranlagen wichtige Informationen über den Wolken-Bedeckungsgrad liefern. Anhand dieser Hinweise können sie sich autonom der Wetterlage anpassen.

Im Rahmen dieser Bachelorarbeit soll eine Software für Solaranlagen entwickelt werden. Das Programm verwirklicht mit Hilfe einer Fisheye-Kamera eine automatische Wolkenanalyse und Wolkendetektion. Zudem soll die Software eine Aussage über den Wolken-Bedeckungsgrad treffen. Der Schwerpunkt dieser Arbeit liegt sowohl in der Analyse der Methoden zur Wolkenerkennung als auch in der Entzerrung der Fisheye-Aufnahmen.

Im folgenden Kapitel werden einige grundlegende Informationen dargelegt, die für die Analyse und Detektion der Wolken notwendig sind. Dabei werden wichtige Auskünfte über die Funktionalität einer Fisheye-Kamera gegeben. Zusätzlich

werden einige Erkenntnisse über die Programmbibliothek OpenCV¹ und Lichtstreuung gewonnen. Im dritten Kapitel wird ein Überblick über Wolken und ihre Helligkeit in Bezug auf ihre Wittereigenschaften gegeben. Kapitel 4 beschreibt die Anforderungen an die Software und stellt Methoden zur Wolkenanalyse und Wolkendetektion vor. Im fünften Kapitel wird das Konzept zur systematischen Bildspeicherung in einem Dateisystem vorgestellt. Das sechste Kapitel beschäftigt sich mit der Kamerakalibrierung und Bildverzerrung. Hierbei wird das Kameramodell von OpenCV und ferner das von Scaramuzza² vorgestellt. Mit Hilfe beider Kameramodelle finden eine Kalibrierung und eine Bildverzerrung statt. Im Kapitel 7 wird die softwaretechnische Analyse und Verarbeitung der Bilddaten wiedergegeben. Außerdem wird die Berechnung vom Wolken-Bedeckungsgrad zur Schau gebracht. Zum Schluss werden eine Zusammenfassung der Resultate sowie ein Ausblick mit Ideen für die Erweiterung dieser Arbeit gegeben.

¹ Ist eine kostenfreie Programmbibliothek. Sie beinhaltet Methoden für Bildverarbeitung und maschinelles Sehen

² Ist ein Professor der Universität Zürich, der im Bereich der Entwicklung von autonomen Maschinen tätig ist

2 Grundlagen

In diesem Kapitel werden die theoretischen Grundlagen dieser Arbeit erörtert. Am Anfang wird die Funktionalität einer Fisheye-Kamera sowie auch die CMOS- und CCD-Bildsensor-Technologien beschrieben. Anschließend folgt ein Einblick in OpenCV. Darauf folgend werden einige Aussagen zum Thema Lichtstreuung getroffen.

2.1 Fisheye-Kamera

Kameras mit Fisheye-Objektiven – mit einer Brennweite von rund 8 mm bei Kleinbild - sind aufgrund ihrer 180°-Öffnungswinkel für photogrammetrische³ Anwendungen sehr gut geeignet. Durch ihre Konstruktion werden die Bildpunkte auf dem Sensor so projiziert, so dass ihr Abstand von der Bildmitte proportional zum Bildwinkel ist, siehe Abbildung 2.1. Aus diesem Grund entstehen am Rand keine Helligkeitsabfälle und die Darstellung des Fisheyes bleibt dadurch ebenfalls exakt kreisförmig. Allerdings werden bei diese Art von Projizierungen gerade Linien, die nicht durch die Bildmitte verlaufen, zum Rand hin sehr stark gekrümmt (tonnenförmige Verzeichnung, siehe Abbildung 2.2).

³ Photogrammetrie (auch Bildmessung genannt): Mit dieser Messmethode werden aus Bilder eines Objektes seine räumliche Lage oder dreidimensionale Form bestimmt

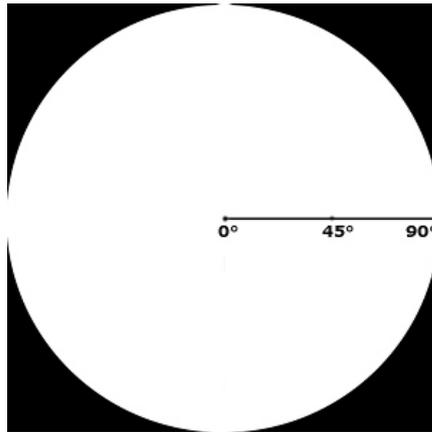


Abbildung 2.1 Der Bildwinkel, ausgehend von der Bildmitte

Bei einer tonnenförmigen Verzeichnung werden die Objekte in der Bildmitte groß und die zum Bildkreisrand hin klein dargestellt. Durch solche Verzeichnungen werden Objekte im Bild fehlinterpretiert. Um derartige Fehlinterpretationen zu vermeiden, müssen solche Abbildungen korrigiert werden. Für die Korrektur wurden spezielle Methoden entwickelt, um die Verzerrung eines Objektivs zu modellieren und zu berechnen. Mit Hilfe der Umkehrung dieser Modelle können Verzeichnungen eines Bildes korrigiert, und somit ein entzerrtes Bild berechnet werden. Zwei solcher Modelle werden im Kapitel 1 näher erläutert.

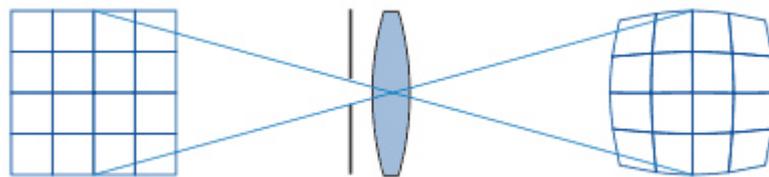


Abbildung 2.2 Entstehung tonnenförmiger Verzeichnung [2]

2.1.1 CCD- und CMOS-Bildsensor-Technologien

Neben dem Objektiv ist der Bildsensor das Herzstück einer Fisheye-Kamera. Zurzeit basieren sie auf zwei wichtige Technologien:

- **C**harged **C**oupled **D**evelopes⁴ (CCD)
- **C**omplementary **M**etal **O**xide **S**emiconductor⁵ (CMOS)

Beide Technologien beruhen auf dem Photoelektrischen Effekt. Dieser zeigt die Freisetzung der Elektronen, die beim Eintreffen des Lichts auf ein Material angeregt werden. Hierfür wird auf die Fotodiode zurückgegriffen.

CCD- und CMOS-Sensoren setzen sich aus vielen Fotodioden zusammen, die nebeneinander angeordnet sind. Die Fotodioden können erst einmal nur Helligkeitsunterschiede bemerken, das bedeutet, sie nehmen Farbdifferenzen zwischen schwarz und weiß wahr. Um eine Farbaufnahme erfassen zu können, wird bei vielen Bildsensoren ein Mosaikfilter auf den Sensor platziert, siehe Abbildung 2.3 und Abbildung 2.4. Dieser Filter hat die Aufgabe das Licht in drei Farbstufen zu filtern, bevor es auf die Fotodiode trifft. Somit reagiert die Diode allein auf den Lichtstrahl mit der gleichen Wellenlänge seines Filters. Sie ist daher nur für eine der drei Grundfarben rot, grün und blau zuständig.

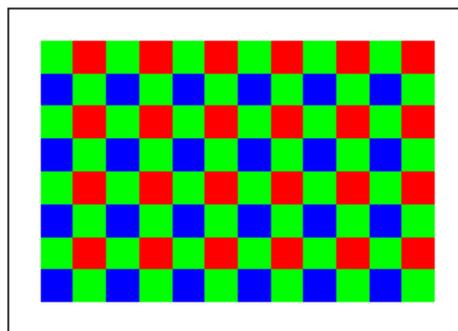


Abbildung 2.3 Mosaikfilter [3]

⁴ Ladungsgekoppeltes Bauelement

⁵ Transistoren, die paarweise komplementär zueinander stehen

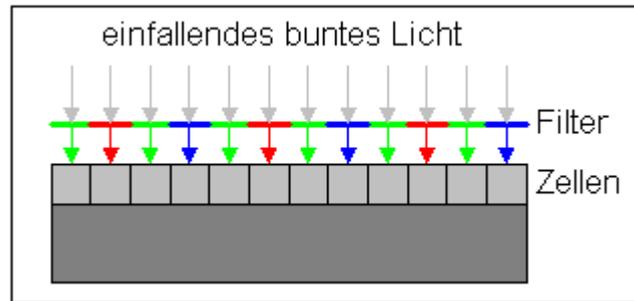


Abbildung 2.4 Die Funktion des Mosaikfilters [3]

Das menschliche Auge kann Helligkeitsabstufungen von grünem Licht besser unterscheiden als von rotem und blauem Licht. Daher ergreifen gewöhnlich etwa die Hälfte der Fotodioden nur das grüne und jeweils ein Viertel rotes und blaues Licht, siehe Abbildung 2.3.

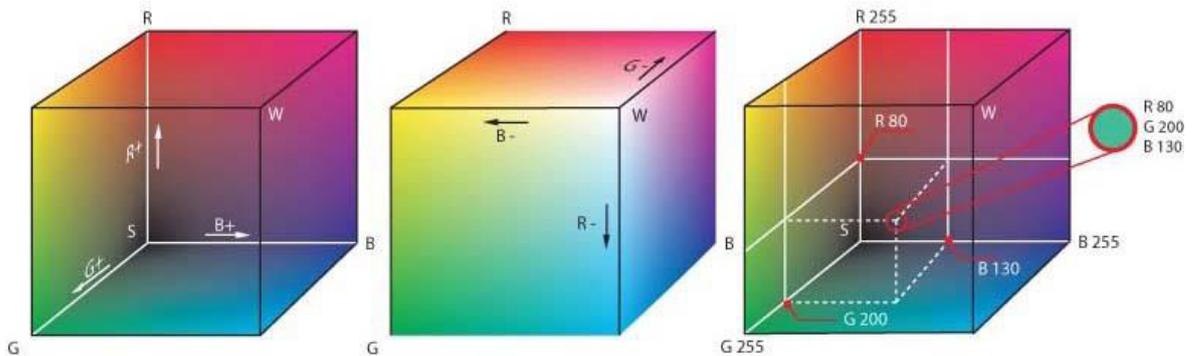


Abbildung 2.5 RGB-Farbraum [4]

Mittels Helligkeitsinformationen der drei Primärfarben können alle Farbtöne nachgebildet werden. Dieser Effekt wird in Abbildung 2.5 sehr gut veranschaulicht. Jedoch sind die Informationen des Sensors ungefüllt, weil jede zweite Fotodiode den Helligkeitswert von Grün und jede vierte den von Rot bzw. Blau misst. Um einen bunten Bildpunkt (Pixel) abbilden zu können, werden die Helligkeitswerte aller drei Farbkanäle gebraucht. Deshalb reichen die vom Sensor gelieferten Werte nicht aus. Aufgrund dessen errechnen Softwarealgorithmen durch Farbinterpolation die fehlenden Farbinformationen der benachbarten Pixel.

Im Allgemeinen heißt es, dass CCD- und CMOS-Bildsensoren nur ein Drittel der Bildinformationen einer Farbaufnahme aufnehmen, während die restlichen zwei Drittel von der Software berechnet werden.

Bei der Farbinterpolation geht es nicht allein darum fehlende Farbinformationen zu berechnen, sondern es müssen auch gleichzeitig mögliche Probleme, wie z.B. Detailverluste, Farbsäume und Moiré-Effekte⁶ umgangen werden. Somit müssen diese speziellen Algorithmen sehr komplexe Berechnungen tätigen, um die Bildqualität auf den besten Stand zu bringen.

Viele Kamerahersteller bringen vor dem Bildsensor ein Anti-Aliasing-Filter an, um solche Probleme zu verringern. Dieser Filter streut das Licht geringfügig, so dass Farbinformationen auch auf die benachbarten Fotodioden verteilt werden. Die Streuung des Lichts macht das Bild leicht unscharf. Die Unschärfe muss im Nachhinein softwaretechnisch behoben werden.

Unterschiede

Die wichtigen Unterschiede der beiden Technologien liegen in dem Ausleseprinzip. Bei CCD-Bildsensoren werden die Bildpunkte zeilenweise ausgelesen und bei CMOS sind sie relativ unabhängig. Deshalb kann es direkt auf den gewünschten Bildpunkt zugegriffen und deren Helligkeitsinformation ermittelt werden. Somit ist die Verarbeitung der aufgenommenen Bilder wesentlich schneller als beim CCD-Sensor.

Auf dem CMOS-Bildsensor ist mehr Elektronik untergebracht, weil jedes Bildelement über einen eigenen Kondensator verfügt, um den direkten Zugriff auf die Helligkeitsinformation gewährleisten zu können.

⁶ (von frz. *moirer*, „moirieren, marmoreiren“) ist ein Muster, das bei der Überlagerung von Rasterstrukturen entstehen können

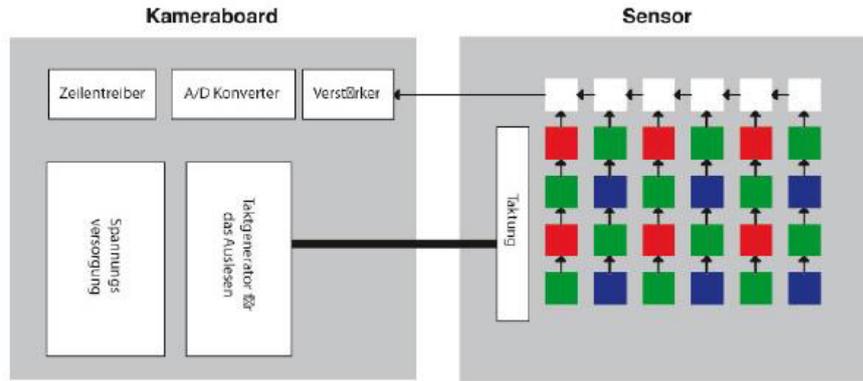


Abbildung 2.6 Schema des CCD-Sensors mit dem Anschluss zum Kameraboard
[5]

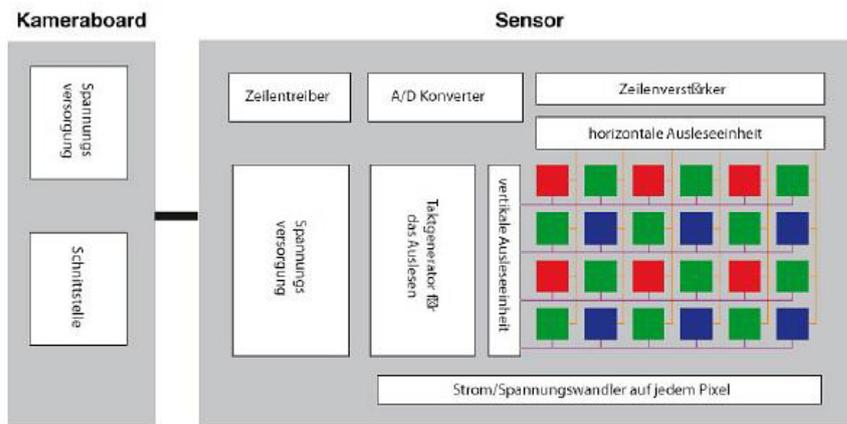


Abbildung 2.7 Schema des CMOS-Sensors mit dem Anschluss zum Kameraboard
[5]

In Abbildung 2.6 ist zu erkennen, dass der CCD-Sensor analoge Signale an das Kameraboard verschickt. Der Verstärker und der A/D-Wandler sind im Kameraboard integriert.

Im Vergleich dazu sind im CMOS-Sensor die wesentlichen Teile der Signal-Kette in den Sensor-Chip eingebaut, siehe Abbildung 2.7. Das heißt, die Kommunikation zwischen dem CMOS-Bildsensor und dem Kameraboard findet auf rein digitaler Ebene statt.

In Tabelle 2.1 werden die oben genannten Merkmale zusammengefasst und zusätzlich die Leistungsfaktoren der beiden Bildsensoren miteinander verglichen.

Besonderheiten	CCD	CMOS
Pixelsignal	Elektronenpaket	analoger Spannung
Chipsignal	analoger Spannung	digitale Bits
Füllfaktor	groß	mittel
Systemrauschen	niedrig	mittel bis hoch
Sensorkomplexität	niedrig	hoch
Systemkomplexität	hoch	niedrig
Leistung	CCD	CMOS
Empfindlichkeit	mittel	etwas besser
Dynamik	hoch	mittel
Geschwindigkeit	mittel bis hoch	hoch bis sehr hoch
Pixelgleichheit (Uniformität)	hoch	niedrig bis mittel
Uniforme Belichtungszeit	schnell, zusammen	schwach
Windowing ⁷	begrenzt	erweitert
Antiblooming ⁸	hoch bis gar nicht	hoch
Taktung	variabel	fest

Tabelle 2.1 Gegenüberstellung von CCD- und CMOS-Bildsensor [5] [6, p. 36]

Der CMOS-Bildsensor liegt in den Bereichen Dynamik und Uniformität deutlich hinten. Die Dynamik setzt sich aus dem Verhältnis der Sättigungsgrenze eines Pixels zu seiner Lichtempfindlichkeit. Die CCD-Bildsensoren haben unter gleichen Verhältnisse ungefähr die doppelte Dynamik wie die CMOS-Bildsensoren.

Die Gleichförmigkeit der Pixel ändert sich zwischen den Bedingungen hell und dunkel. Bisher erzielten CMOS-Sensoren unter alle beiden Bedingungen schlechte Resultate, weil aufgrund ihrer Konstruktion, jedes Pixel seinen eigenen Verstärker hat. Aber seit dem die Verstärker ihren Ausgang überwachen, können auch CMOS-

⁷ Windowing ist eine Fähigkeit, die dem Anwender erlaubt nur bestimmte Bildbereiche auszulesen

⁸ Das Überlaufen des Elektronenpakets in die Nachbapixel wird verhindert. Dieser Fall kann auftauchen, wenn die Belichtung an einer Stelle zu stark ist. Die CMOS-Bildsensoren sind aufgrund ihrer Verschaltung nicht davon betroffen. Die CCD-Sensoren hingegen benötigen einen zusätzlichen Schutz, um das *Blooming* zu verhindern

Bildsensoren eine hohe Uniformität erlangen. Aufgrund der Offsetvariationen haben sie jedoch noch Probleme bei Dunkelheit. [6, pp. 32-35]

2.1.2 Technische Eigenschaften der eingesetzten Kamera

Für die Durchführung des Projektes wurde die Fisheye-Kamera „ICA-8350“ von der Firma Planet verwendet. Im Datenblatt [7] sind folgende wichtige Merkmale aufgelistet:

- **Kamera:**
 - ✚ FOV⁹ = $180^\circ \pm 5^\circ$ (Tiefe/Horizontal/Vertikal)
 - ✚ Brennweite $f = 1.25\text{mm}$
 - ✚ 0.5 lux Beleuchtungsstärke bei einer max. Blende von F2.0
 - ✚ Maximale Auflösung von 1536 x 1536 Pixel
 - ✚ Sensor: 1/2.8“, 3.4 Mega-Pixel CMOS-Sensor
- **Video:**
 - ✚ Kompressionsformat: H.264/MPEG-4 AVC (ISO/IEC 14496-10) und M-JPEG¹⁰ (Motion-JPEG)
 - ✚ Bildrate bis zu 22 fps
- **Bildübertragung:**
 - ✚ Netzwerkstandards: IEEE 802.3 10Base-T , IEEE 802.3u 100Base-TX, IEEE 802.3ab 1000Base-T
 - ✚ Netzwerkprotokolle: IPv4, IPv6, TCP/IP, UDP, HTTP, HTTPS, SMTP, FTP, NTP, DNS, DDNS, DHCP, ARP, Bonjour, UPnP, RTSP, RTP, RTCP, IGMP, PPPoE, 3GPP, ICMP, Samba

⁹ Sichtfeld, Bildwinkel (engl. field of view)

¹⁰ bei diesem Videocodec, wird jedes Einzelbild separat als JPEG-Bild komprimiert



Abbildung 2.8 ICA-8350 Fisheye-Kamera [7, p. 1]

Die Bildübertragung kann mittels eines der oben genannten Netzwerkprotokolle geschehen. Unten sind zwei wichtige Beispiele aufgelistet:

- Der Zugriff mit dem HTTP¹¹-Protokoll auf ein Einzelbild erfolgt mit folgender URL: „*http://IP-Adresse:Port/image.jpg*“
- Für die Übertragung über das RTSP¹²-Protokoll muss die URL so lauten: „*rtsp:IP-Adresse:Port/ipcam.sdp*“

Für das Projekt ist die Nutzung vom HTTP-Protokoll, mit der Kombination von M-JPEG vorteilhafter, weil durch den Aufruf der URL-Adresse nur ein Einzelbild erfasst wird. Aus diesem Grund muss keine Zusatzfunktion implementiert werden, die die Schnappschüsse aus einer Videoübertragung realisiert.

2.2 OpenCV

OpenCV ist eine kostenlose Programmbibliothek, die in erster Linie für den Bereich Objekterkennung vorgesehen ist. Sie ist mit ihren recheneffizienten C/C++-

¹¹ **H**yper**t**ext **T**ransfer **P**rotocol (HTTP) ist ein allgemeines, zustandsloses, objektorientiertes Protokoll zur Übertragung von Daten im **World Wide Web** (www). Die wesentlichsten Funktionen sind Dateien vom Webserver zu verlangen und zum Browser zu senden

¹² **R**ea**t**-**T**ime **S**treaming **P**rotocol (RTSP) ist ein Netzwerkprotokoll, welches Kontrolle über die Übertragung von Echtzeit-Mediaströmen (Livestreams) bietet. Bei diesem Protokoll werden keine Nutzdaten übertragen, deshalb wird es ebenfalls als Netzwerk-Fernbedienung bezeichnet

Algorithmen in der Bildverarbeitung sehr beachtenswert und hat ebenso Schnittstellen zu den Programmiersprachen Java und Python, die unter Linux, Windows, Mac OS, IOS und Android kompiliert werden können.

Das Projekt OpenCV wurde im Jahr 1999 von Intel ins Leben gerufen, wird aber zurzeit hauptsächlich von Willow Garage¹³ konzipiert. [8] Heute hat die Bibliothek weltweit eine Usergruppe von 47.000 Mitgliedern und wurde mehr als 9 Millionen Mal heruntergeladen, siehe [9]. Deshalb zählt sie zum Standard für die Bildverarbeitung.

2.2.1 Was bietet OpenCV an?

OpenCV besteht aus fünf Hauptkomponenten, die in Programmen unabhängig voneinander verwendet werden können. Jede Komponente beinhaltet Algorithmen für verschiedene Anwendungsbereiche, wie z.B. für Bildverarbeitung (CV und CXCORE), siehe Abbildung 2.9. [10, p. 13]

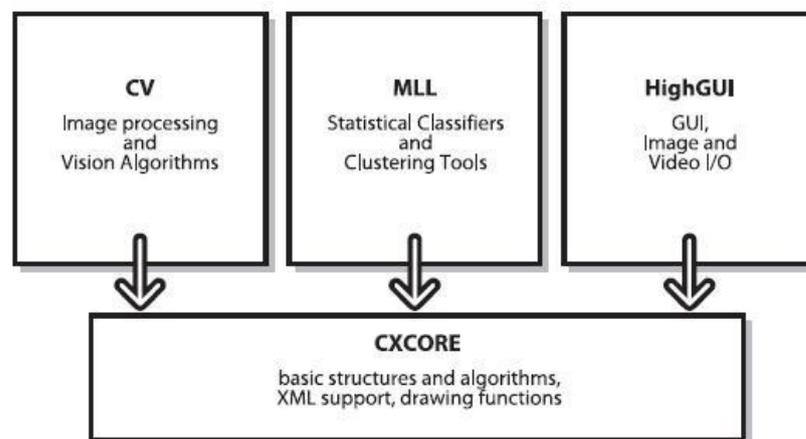


Abbildung 2.9 Basisstruktur der OpenCV [10, p. 13]

¹³ Willow Garage ist ein US-Unternehmen mit Hauptsitz in Kalifornien, das auf Robotertechnologie spezialisiert ist. Es hat einen Personalroboter (PR2) entwickelt, der Billard spielen und Socken zusammenlegen kann

In Abbildung 2.9 sind vier von fünf Bestandteilen der OpenCV aufgelistet. Es ist zu erkennen, dass sie in zwei Ebenen aufgeteilt sind. Die unterste Ebene ist die CXCORE-Bibliothek. Sie beinhaltet die grundlegenden Datenstrukturen, die für die Arbeit mit OpenCV gebraucht werden (Punkte, Skalare, Matrizen und Bilder), um z.B. mathematische Berechnungen und Darstellungen durchführen zu können. In der zweiten Ebene befinden sich die Bibliotheken CV, MLL und HighGUI. Die CV-Bibliothek hat Algorithmen für Bildverarbeitung (Filter, geometrische Transformationen), Bildanalyse (Histogramme) und Objekterkennung. Die Bibliothek MLL (**M**achine **L**earning **L**ibrary) enthält Algorithmen des maschinellen Lernens, welche im Kontext des maschinellen Sehens (Computer Vision) eingesetzt werden. Mit Hilfe von HighGUI-Bibliothek können sehr leicht grafische Oberflächen erstellt werden. Sie bietet außerdem Möglichkeiten Bilder und Videos - ohne viel Aufwand - zu importieren und zu speichern. Die fehlende Komponente heißt *CvAux* und wird scheinbar in Zukunft in das Hauptelement *CV* überliefert. [11] Sie umfasst spezielle Funktionen für dreidimensionale Objektverfolgung oder Gesichtserkennung.

In diesem Projekt wird hauptsächlich mit den Zweigen CXCORE, *CV* und *HighGUI* gearbeitet. Als Programmiersprache wird C++ genutzt, weil dieser im Vergleich zu C objektorientiert und genauso recheneffizient ist.

2.2.2 Wichtige Datentypen und Funktionen

OpenCV bietet nicht nur Funktionen für die Bildverarbeitung an, sondern stellt auch die notwendigen Datenstrukturen zur Verfügung, um Bilder und Ergebnisse der Rechenoperationen zu speichern.

Unter OpenCV stehen folgende wichtige Datentypen (Klassen) mit dazugehörigen Funktionen (Methoden) [10, pp. 31-47] zur Verfügung:

- CvPoint

Sie ist eine einfache Struktur mit zwei Integer-Parametern x und y , die ein Pixel in einem Bild darstellen.

Dieser Datentyp ist auch in anderen Varianten vorhanden, wie *CvPoint2D32f* und *CvPoint3D32f*. Sie geben einen Punkt in einem 2D- oder 3D Raum mit Fließkommagenauigkeit an.

CvPoint3D32f hat in seiner Struktur einen zusätzlichen Parameter z für die dritte Dimension.

- CvScalar

Diese Struktur beinhaltet vier Parameter mit Fließkommagenauigkeit, die vorwiegend zur Darstellung von Farbinformationen genutzt werden und existiert in drei Varianten.

Bei der ersten Variante (*CvScalar*) dürfen alle vier Parameter unterschiedliche Werte haben. Bei der Zweiten hingegen (*CvScalarAll*) werden alle vier Attribute mit dem gleichen Wert initialisiert. Und bei der letzten Variante (*Cv_RGB*) müssen die Parameter den Wert der übergebenen Farbe im RGB-Farbraum entsprechen.

- CvSize

Die Struktur *CvSize* beschreibt eine rechteckige Fläche mit Höhe und Breite. Sie besteht mit Integer- und Fließkommagenauigkeit (*CvSize2D32f*).

- CvRect

CvRect stellt ebenso eine rechteckige Fläche dar. Jedoch beinhaltet sie zwei weitere Parameter (x,y) , die die Position des Rechtecks angeben.

- CvMat

Matrizen werden mit der Klasse *CvMat* beschrieben. Mit der Funktion *cvCreateMat* kann eine beliebige Matrix erzeugt und deren Speicher reserviert werden. Nachdem die Matrix nicht mehr benötigt wird, kann mittels *cvReleaseMat* der reservierte Speicher wieder freigegeben werden.

Zudem können die erstellten Matrizen anhand von Matrizenoperationen der OpenCV modifiziert werden.

- IplImage

IplImage ist die wichtigste Datenstruktur, die von OpenCV genutzt wird, um Bilddaten zu importieren. Mit Hilfe dieser Struktur werden auch Metadaten der Bilder, wie Bildgröße und Farbraum gespeichert.

Anhand der Funktion *cvCreateImage* kann ein Bild erstellt und mit *cvLoadImage* ein beliebiges importiert werden. Um den reservierten Speicherplatz wieder freizugeben muss die Funktion *cvReleaseImage* aufgerufen werden.

In diesem Projekt werden die oben aufgezählten Datentypen und deren Funktionen für Bildanalyse und Datenimport eingesetzt. Außerdem werden andere hilfreiche OpenCV-Funktionen für Bildmodifikationen angewendet. Diese werden hier nicht weiter betrachtet.

2.3 Lichtstreuung

Die Lichtstreuung tritt auf, wenn das Licht, durch Auftreffen auf ein Objekt, in verschiedene Richtungen abgelenkt wird. Dieser Effekt spielt hauptsächlich in der Erdatmosphäre eine wichtige Rolle, denn Lichtstrahlen der Sonne, die ein Medium wie Atmosphäre durchqueren, werden durch unendlich viele Partikeln absorbiert oder gestreut. Der englische Physiker Lord Rayleigh konnte im Jahr 1871 dieses Phänomen nachweisen.

Bei dieser Bilanz spielen Form und Größe (Radius r) des Partikels und die Wellenlänge der Strahlung λ eine wichtige Rolle. Mit Hilfe der beiden Parameter kann der Mie-Größenparameter α [12, p. 18] berechnet werden:

$$\alpha = \frac{2 \cdot \pi \cdot r}{\lambda} \quad (2.1)$$

2.3.1 Rayleigh-Streuung

Die Rayleigh-Streuung entsteht erst dann, wenn $\alpha < 0.1$ [12, p. 18] ist. In diesem Fall ist der Radius des streuenden Partikels viel kleiner als die Wellenlänge des Lichtstrahls.

Die winkelabhängige Streuintensität $I(\varphi)$ wird mit folgender Formel beschrieben [12, p. 21]:

$$I(\varphi) = \frac{3}{4}(1 + \cos^2(\varphi)) \quad (2.2)$$

Der Rayleigh-Streukoeffizient $\sigma_R(\lambda)$ hingegen kann mit der unten dargestellten Formel berechnet werden [13]:

$$\sigma_R(\lambda) = \frac{8\pi^3(n_\lambda^2 - 1)^2}{3\lambda^4 N} = \text{const} \cdot \lambda^{-4} \quad (2.3)$$

Zusammen mit der Wellenlänge des Lichtstrahls ist die Formel (2.3) ebenso von n_λ (wellenlängenabhängiger Brechungsindex der Luft) und N (Anzahl der Moleküle pro cm^3) abhängig. Es ist gut zu erkennen, dass der Rayleigh-Streukoeffizient proportional zu λ^{-4} ist. Dadurch ist der blaue Anteil des sichtbaren Lichtes ($\lambda = 450\text{nm}$) an atmosphärischen Luftmolekülen stärker gestreut als der rote Anteil ($\lambda = 710\text{nm}$). Aus diesem Grund erscheint der Himmel für das menschliche Auge blau.

2.3.2 Mie-Streuung

Die Mie-Streuung tritt bei kugelförmigen Teilchen auf, deren Größenparameter $\alpha > 0.1$ (Aerosole¹⁴, Wolkentröpfchen) sind. Hierbei wächst die Vorwärtsstreuung mit zunehmendem Teilchendurchmesser, während die Lichtstreuung vertikal zur Einfallsebene abfällt.

Bei dieser Art von Streuung ist die Streuintensität unabhängig von der Wellenlänge des Lichtstrahls. Aus diesem Grund sind wassertropfenreiche Wolken nahezu weiß.

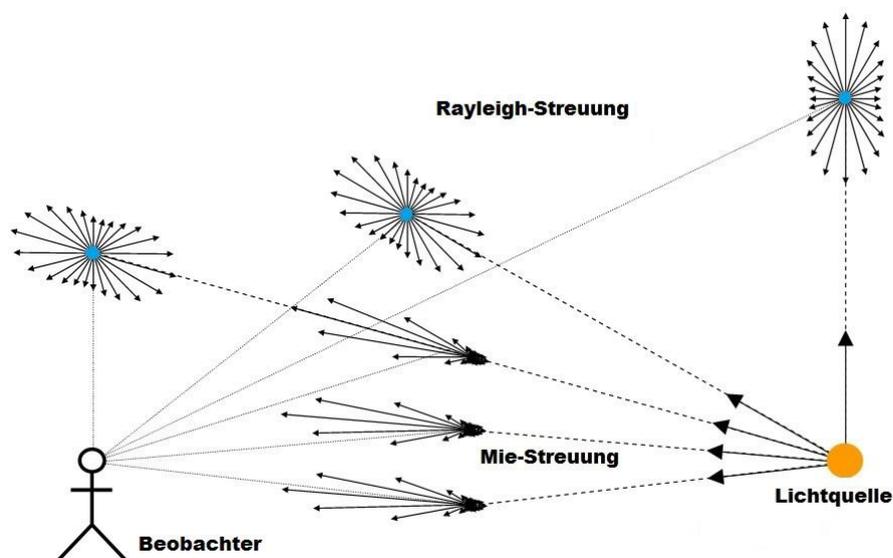


Abbildung 2.10 Schematische Darstellung der Mie- (schwarz) und Rayleigh-Streuung (blau) [14]

In Abbildung 2.10 sind die beiden unterschiedlichen Streuungsarten abgebildet. Es ist gut zu erkennen, dass bei der Rayleigh-Streuung der Lichtstrahl sich in allen Richtungen verteilt. Im Vergleich dazu wird bei der Mie-Streuung das Licht asymmetrisch gestreut.

In der bewölkten Atmosphäre ist sowohl die Mie- als auch die Rayleigh-Streuung vertreten. Die Anzahl der Partikel legen fest, welche der beiden Streuungsarten

¹⁴ Aerosol ist ein Gemisch aus einem Gas und einer fein verteilten Flüssigkeit oder Feststoff wie Staub

überwiegend vorhanden ist. Je mehr Wassertröpfchen in der Atmosphäre sind, desto größer ist der prozentuale Anteil des nach vorne gestreuten Lichtes.

3 Wolken

Wolken sind eine Ansammlung von kondensierten Wasserteilchen und Eiskristallen, die in der Erdatmosphäre schweben. Sie kommen hauptsächlich in der Troposphäre vor, weil sich dort der größte Anteil des atmosphärischen Wassers aufhält. Mit geringer Wahrscheinlichkeit tauchen sie ebenfalls in der Stratos- und Mesosphäre auf.

Außerdem enthalten Wolken zusätzliche Partikeln, die in Abgasen, Rauch und Staub auftauchen. Temperatur und Wasserdampfgehalt der Erdatmosphäre sind wichtige Grundlagen für die Wolkenschöpfung. Neben ihrer Wichtigkeit für die Wetterlage, sind Wolken für den Strahlungshaushalt der Atmosphäre sehr bedeutsam. [15] Dabei wird das kurzwellige Licht von den ihnen zurückgeworfen, doch das langwellige Infrarotlicht bleibt zwischen Wolkendecke und Erdoberfläche eingeschlossen. Die Albedo¹⁵ von Wolken liegt zwischen 60 und 90% [16].

3.1.1 Helligkeit und Farben der Wolken

Die Helligkeit der Wolken hängt vom Licht ab, das von ihren Partikeln reflektiert, gestreut und durchgelassen wurde. Dieses Licht stammt oft aus direkter oder diffuser Sonnenstrahlung, es kann aber auch vom Mond oder der Erdoberfläche sein.

Flächen, die mit Eis und Schnee bedeckt sind, haben eine große Albedo. Deshalb werden die meisten Lichtstrahlen, die auf solche Flächen treffen zurückreflektiert, und dieser Effekt lässt die Wolken heller erscheinen.

Die Wolkenhelligkeit kann sich ebenso durch Einfluss von Dunst oder besondere Lichterscheinungen in der Erdatmosphäre verändern. Sobald sich der Dunst zwischen Beobachter und Wolke befindet, erscheint durch Abhängigkeit der Wolkendichte und Richtung der einfallenden Lichtstrahlen, die Wolke für das

¹⁵ ist ein Maß für das Rückstrahlvermögen von diffus reflektierenden Oberflächen

menschliche Auge heller oder dunkler. Außerdem schwächt Dunst die Kontraste der Wolken ab. [17]

Die Farben der Wolken hängen von Wellenlängen der Lichtstrahlen ab, die sie beleuchten. Sie selbst können ihre Farben nicht ändern, da die Aussagen der Rayleigh-Streuung nicht zutreffen. Sobald sich Dunst oder Staub zwischen Beobachter und Wolke befindet, kann die Färbung der Wolke leicht verändert werden. Dieser Effekt lässt sehr weit liegende Wolken gelb oder orange erscheinen. Wenn der Sonnenstand hoch ist, dann sind Wolken oder ihre beleuchteten Abschnitte weiß oder grau. Diejenigen Abschnitte, die das Licht hauptsächlich vom blauen Himmel erhalten, sehen blaugrau aus. Bei Sonnenauf- und Sonnenuntergang kann sich ihre Farbe von Gelb bis zu Rot verändern, weil dadurch der Weg des Lichtes in der Atmosphäre verlängert und somit ein Überschuss der kurzwelligen Lichtanteile seitlich weggestreut wird, siehe Rayleigh-Streuung. Deshalb ist überwiegend das langwellige Licht vertreten, der die Sonne, den Himmel und die Wolken rötlich darstellt. [17]

Die Wolkenhöhe, sowie auch die jeweilige Stellung zum Beobachter und zur Sonne, sind ebenfalls wichtige Kriterien für die farbliche Darstellung der Wolken. Die hohen Wolken werden weiß dargestellt, wenn die Sonne sich dicht ober- oder unterhalb des Horizonts befindet. Die mittelhohen Wolken hingegen erscheinen für das menschliche Auge orange oder rötlich. Sehr niedrige Wolken sehen grau aus. Die Wolken in gleicher Höhe werden bei Blickrichtung gegen die Sonne weniger rötlich dargestellt als in der entgegengesetzter Richtung.

3.1.2 Wolkenklassifikation und ihre Wetterbedeutung

Der britische Chemiker und Apotheker Luke Howard (1772 – 1864) stellte in seiner Publikation „Modifikation der Wolken“ aus dem Jahr 1803, die erste Klassifizierung der Wolken nach ihrer charakteristischen Form auf.

Die heutige Wolkenklassifikation wird im International Cloud Atlas dokumentiert. Dieser wird von der World Meteorological Organization (WMO) überarbeitet und neu

herausgegeben. Dieses Dokument wird weltweit für die Wolkenbeobachtung verwendet. [18]

Das erste Unterscheidungsmerkmal ist die Höhe der Wolke über der Erdoberfläche, dabei wird eine grobe Unterteilung in drei Ebenen (Tief, Mittelhoch und Hoch) vorgenommen. Die Grenzen der Ebenen sind stark von der geographischen Breite und der Jahreszeit abhängig, siehe Tabelle 3.1.

Ebene	Polargebiete	Gemäßigte Zone	Tropische Zone
Hoch	3 bis 8 km	5 bis 13 km	6 bis 18 km
Mittelhoch	2 bis 4 km	2 bis 7 km	2 bis 8 km
Tief	0 bis 2 km	0 bis 2 km	0 bis 2 km

Tabelle 3.1 Wolkenebenen nach WMO [19] [20, pp. 31-35]

Es gibt noch eine weitere Gruppe von Wolken, die sich im Vergleich zu anderen, in allen drei Ebenen gleichzeitig aufhalten können. Solche Wolken werden vertikale Wolken genannt. Sie umfassen Höhenbereiche von 0 bis 15 km. In den Tropen können sie sogar Höhenbereiche bis 18 km belegen.

Neben der Höhenunterteilung werden sie gleichzeitig nach Form und Gestalt eingeteilt. Dabei werden zehn Wolkengattungen unterschieden, die sehr verschieden aussehen, siehe Abbildung 3.1. Innerhalb der Gattungen werden Wolken nach ihrer Art differenziert. Mit der Angabe der Wolkenart werden Wolkengattungen noch genauer nach ihrer Gestalt und dem inneren Aufbau unterteilt. Es gibt 14 Wolkenarten, die von der WMO klassifiziert sind. Darüber hinaus existieren neun Wolkenunterarten, die die Anordnung der Wolkenteile und ihre Lichtdurchlässigkeit beschreiben. [20, pp. 31-35]

Zahlreiche Wolken haben eine spezielle Erscheinung inne, die gemeinsam mit der Hauptmasse der Wolken oder auch separat davon auftreten können. Solche Erscheinungen werden Sonderformen oder Begleitwolken genannt.

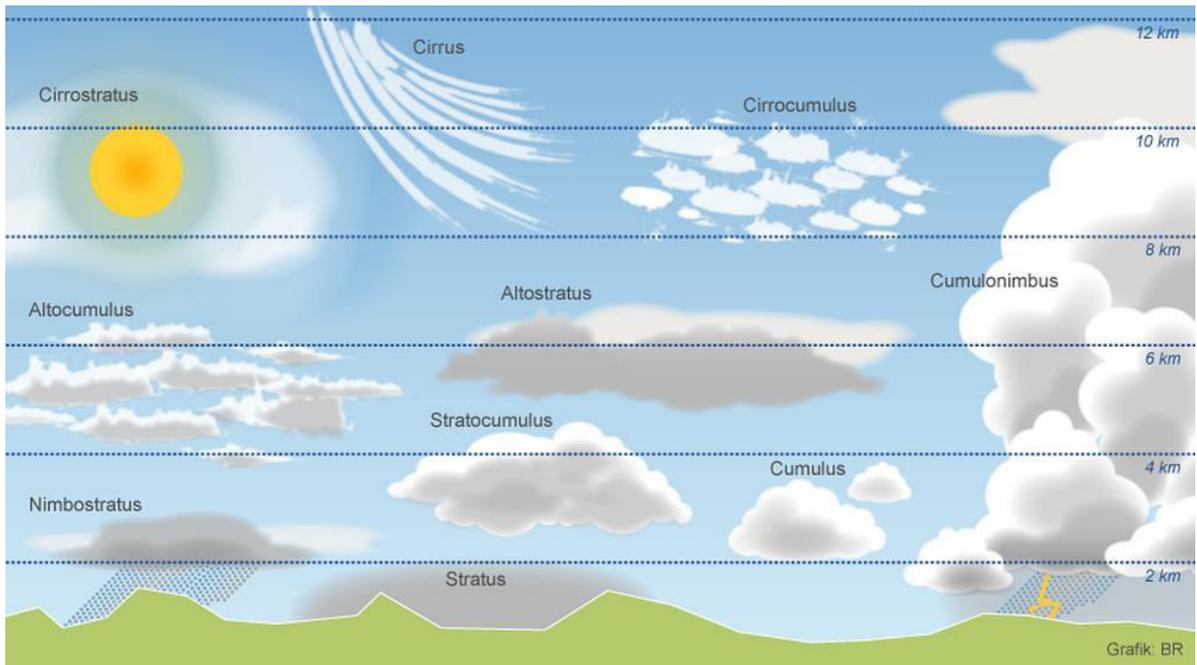


Abbildung 3.1 Schematische Darstellung der Wolkengattungen in Bezug auf ihre Höhe [21]

Bei der internationalen Wolkenklassifikation ist auch von Mutterwolken die Rede. Mutterwolken sind Wolken, die sich nicht aus einem wolkenlosen Zustand entwickeln, sondern aus anderen Gattungen hervorgehen. Genau gesagt, kann z.B. eine Cumulus-Wolke zu einer Cumulo-Nimbus (in Abbildung 3.1 Cumulonimbus) aufwachsen. [20, pp. 31-35]

Im unteren Abschnitt werden Wolkengattungen näher beschrieben und zudem ihre Wetterbedeutung erklärt, die einzelnen Arten sind in der Abbildung 3.1 zu finden [20, pp. 31-35] :

Cirrus (Ci)

Ist eine Wolke, die sich in große Anhöhe befindet und nur aus Eis zusammengesetzt ist. Ihr Erscheinungsbild ist aufgrund vom starken Wind fedrig und ausgefranst. Diese Wolken produzieren keinen Regen.

-
- Cirrostratus (Cs)** Ist ebenfalls eine hohe Wolke, die vorwiegend aus Eis besteht. Die Unterseiten sind hell. Dieser Wolkentyp produziert keinen Niederschlag.
- Cirrocumulus (Cc)** Ist eine hohe Schichtwolke, die hauptsächlich aus Eis besteht. Sie produziert ebenfalls keinen Niederschlag.
- Alto cumulus (Ac)** Ihr Erscheinungsbild ist ausgefranst, streifig oder einheitlich. Sie verdecken den Himmel entweder vollkommen oder partiell. Sie sind an manchen Stellen so dünn, dass die Sonne ganz schwach hindurchscheinen kann. Es sind keine Niederschlagswolken.
- Altostratus (As)** Ist eine mittelhohe Schichtwolke, aus der nur leichter Niederschlag fallen kann. Dabei ist Nieselregen und auch Schnee möglich.
- Stratocumulus (Sc)** Sind tiefhängend und klar begrenzte Wolkenbänke, die schichtförmig den Himmel bedecken. Aus diesen Wolken sind etwas Regen und leichter Schneefall möglich.
- Stratus (St)** Ist eine tiefe konturlose Wolke, die zum Teil Berge einhüllen kann. Sie wird daher Hochnebel genannt. Nieselregen oder leichter Schneefall sind ebenfalls aus diesen Wolken möglich.
- Cumulus (Cu)** Sind weiße Wolken, deren Unterseite dunkler erscheinen. Sie bilden sich bei schönem Wetter und können sich bis zu Cumulo-Nimbus weiterentwickeln. Bei dieser Art von Wolken fällt kein Niederschlag.

**Cumulo-Nimbus
(Cb)**

Ist eine hochreichende Wolke, deren Unterseite sehr dunkel ist. Die obere Struktur, die aus Eis besteht, erscheint hell, sobald sie von der Sonne beleuchtet wird. Sie ist eine Gewitterwolke, die Donner, Blitz, kräftige Regen-, Schnee- und Hagelschauer sowie auch Sturm und Orkanböen mit sich bringen kann.

Nimbostratus (Ns)

Ist eine graue Schichtwolke, die viel mächtiger als Stratus ist. Aus diesem Wolkentyp fällt Niederschlag in Form von Regen, Schnee oder Graupel.

4 Wolkenanalyse und Wolkendetektion

Im diesem Kapitel werden als Erstes die Anforderungen an dem Programm erläutert. Als Nächstes werden einige Methoden zur Wolkenanalyse und Wolkendetektion präsentiert. Zum Schluss wird über mögliche Fehler diskutiert, die bei einer Wolkenerkennung auftreten können.

4.1 Anforderungsbeschreibung

Die Fisheye-Kamera aus dem Kapitel 2.1.2 ist auf dem Hochhaus des Departments Technik und Informatik der HAW Hamburg installiert, siehe Abbildung 4.1. Damit eine Aussage über den aktuellen Wolken-Bedeckungsgrad¹⁶ getroffen werden kann, müssen aus ihren Bildern, die Wolken detektiert und anschließend analysiert werden.



Abbildung 4.1 Die Fisheye-Kamera befindet sich auf dem Dach des Departments Technik und Informatik der HAW Hamburg

¹⁶ Beschreibt die Stärke der Bedeckung durch Wolken im sichtbaren Abschnitt des Himmels. Dieser Wert wird in Achteln angegeben ($\frac{0}{8}$: wolkenlos, ... , $\frac{8}{8}$: bedeckt)

Um das softwaretechnisch richtig realisieren zu können, muss zunächst ein Algorithmus implementiert werden, der die Kamerabilder in einem beliebigen Zeitintervall abspeichert (z.B. alle 60 Minuten). Dafür kann eins der Bildübertragungsprotokolle verwendet werden, die im Kapitel 2.1.2 erwähnt worden sind. Die Abspeicherung soll in einem Dateisystem¹⁷ erfolgen, der von Benutzer beliebig ausgesucht werden kann. Das heißt, der Benutzer ist gezwungen einen Pfad (z.B. unter Windows „/Users/Mustermann/Bilder/“) anzugeben, wo das Kamerabild aufbewahrt werden darf. Es ist außerdem erforderlich, die Laufzeit des Programms (z.B. 20 Tage) vom Benutzer festlegen zu lassen. Diese Anforderungen werden im Kapitel 1 konzipiert.

Während ein Bild im ausgewählten Pfad abgespeichert wird, muss in der Zwischenzeit ihr Inhalt im Programm weiterverarbeitet werden. Zuvor kann das Bild zur Vereinfachung in eine andere Ebene transformiert werden, weil es eine tonnenförmige Verzeichnung beinhaltet, die die Objekte falsch darstellen lässt, siehe Kapitel 2.1. Im fünften Kapitel wird auf diese Problematik näher eingegangen. Anschließend ist es nötig, mittels Wolkenanalyseverfahren die Daten des Bildes zu untersuchen, damit die Wolkenabschnitte, die im Bild zu sehen sind, identifiziert werden können. Für die Identifikation müssen die einzelnen Bildpunkte richtig interpretiert werden. Um das bewerkstelligen zu können, sind sowohl Kenntnisse über das Prinzip der Lichtstreuung, als auch die Theorie zur Helligkeit und Farben der Wolken erforderlich. Nach Erkennung der Wolken im Bild kann der Wolken-Bedeckungsgrad berechnet werden.

Für die Analyse ist es wichtig Störfaktoren sowie Einfluss der Sonne auf das Bild zu reduzieren oder ganz zu eliminieren. Andere Objekte, die die Berechnungen ebenfalls verfälschen können, müssen vorab ausgefiltert werden.

Darauf folgend können die erkannten Wolken nach ihrer Helligkeit (hell oder dunkel) kategorisiert werden. Die Kategorisierung nach WMO und die Erkennung der Bewegungsrichtung werden in dieser Arbeit nicht realisiert, aber die Vorgehensweise wird im nächsten Kapitel kurz erläutert.

¹⁷ Ein Dateisystem ist eine Ablageorganisation auf einem Datenträger eines Rechners

Ferner ist es notwendig die Himmelsrichtung anzugeben, weil ansonsten der Benutzer nicht weiß, von welcher Richtung die abgebildeten Wolken sich annähern. Letztlich muss das analysierte Bild in dem gleichen Pfad wie das Originalbild abgespeichert werden.

4.2 Methoden zur Wolkenanalyse und Wolkendetektion

Im Kapitel 3.1.1 wurde bereits erwähnt, aus welchen Farben sich die Wolken in der Erdatmosphäre zusammensetzen. Diese Farbinformationen können genutzt werden, um programmtechnisch Wolken vom blauen Himmel differenzieren zu können. Dabei ist es relevant, die Aufnahmen der Fisheye-Kamera richtig zu analysieren.

Die erste Überlegung ist, viele Aufnahmen der Fisheye-Kamera pixelweise statistisch zu untersuchen, um anschließend einen Grenzwert für den blauen Himmel oder Wolken definieren zu können. Danach kann mittels Schwellenwertverfahren das Bild binarisiert werden. Dieses Verfahren würde das gesuchte Objekt ein- und den Hintergrund ausblenden. So eine ähnliche Methode ist im Jahr 1998 von Long und DeLuisi [22, pp. 171-174] [23, pp. 633-650] genutzt worden, um den Wolken-Bedeckungsgrad zu ermitteln. Wie bereits im Kapitel 2.1.1 veranschaulicht wurde, besteht ein Bildpixel P in einem 8-bit-RGB-Farbraum aus den Grundfarben rot, grün und blau, die jeweils die Werte 0 bis 255 annehmen können. Für Long und DeLuisi ist - aufgrund der Lichtstreuung - nur die Helligkeitswerte für Rot und Blau interessant gewesen. Dabei stellten sie folgendes Kriterium, um einen blauen Bildpunkt des Himmels von den Weißen der Wolke zu separieren:

Ein Bild in einem RGB-Farbraum kann auch als eine Matrix dargestellt werden:

$$B = \begin{bmatrix} R_{0,0}, G_{0,0}, B_{0,0} & R_{0,1}, G_{0,1}, B_{0,1} & \dots & R_{0,M}, G_{0,M}, B_{0,M} \\ R_{1,0}, G_{1,0}, B_{1,0} & R_{1,1}, G_{1,1}, B_{1,1} & \dots & R_{1,M}, G_{1,M}, B_{1,M} \\ \vdots & \vdots & \ddots & \vdots \\ R_{N,0}, G_{N,0}, B_{N,0} & R_{N,1}, G_{N,1}, B_{N,1} & \dots & R_{N,M}, G_{N,M}, B_{N,M} \end{bmatrix} \quad (4.1)$$

Aus allen Bildpunkten, die nicht zur Maske (z.B. Rahmen, Objekte) oder Filter (z.B. Sonnenstrahlen, Dispersion durch Acrylglaskuppel der Kamera) gehören wird das Verhältnis Rot zu Blau gebildet:

$$V_{Pixel} = V_{n,m} = \frac{R_{n,m}}{B_{n,m}} \quad \text{falls } R_{n,m}, B_{n,m} \notin \text{Maske, Filter} \quad (4.2)$$

Das Ergebnis für V_{Pixel} kann nun mit einem Schwellenwert T verglichen werden. Falls V_{Pixel} größer ist als T , dann handelt es sich um einen Bildpunkt, der zu einer Wolke gehört, im anderen Fall hingegen gehört er zum blauen Himmel. Der Schwellenwert ist sowohl von der Farbwiedergabe der eingesetzten Fisheye-Kamera als auch von der Tageszeit (Sonnenstand) abhängig. Um einen akkuraten Schwellenwert zu bestimmen, müssen V_{Pixel} zahlreicher Kamerabilder nach ihrer Häufigkeitsverteilung untersucht werden. Dabei ist zu beachten, dass für die Untersuchungen Aufnahmen mit komplett bedecktem und unbedecktem Himmel gewählt werden, die in unterschiedlichen Tageszeiten getätigt worden sind.

Nachdem der Schwellenwert ermittelt und mittels Schwellenwertverfahren die Pixel eines Bildes binarisiert (0: unbewölkt, 1: bewölkt) worden sind, kann der Wolken-Bedeckungsgrad berechnet werden.

Für den Wolken-Bedeckungsgrad N_{WBed} müssen als Erstes alle Bildpunkte der bewölkten (n_{cloudy}) und ebenfalls der unbewölkten (n_{no_cloudy}) Himmel summiert werden. Anschließend kann der Bedeckungsgrad mit Hilfe folgender Formel berechnet werden:

$$N_{WBed} = \frac{n_{cloudy}}{n_{cloudy} + n_{no_cloudy}} \quad (4.3)$$

Das Ergebnis der Formel (4.3) gibt den Wolken-Bedeckungsgrad im Wertebereich von $[0, \dots, 1]$ an.

Mit der Methode von Long und DeLuisi kann eine Kategorisierung nach WMO nicht verwirklicht werden, weil dafür die berechneten Bildinformationen zum blauen Himmel und zur Wolke nicht ausreichend sind. Damit so eine Kategorisierung durchgeführt werden kann, müssen Wolken nach ihren Besonderheiten (Höhe, Helligkeit, Zustandsform) differenziert werden.

Anhand der Farbdifferenzen aus dem Kapitel 3.1.1 können Wolken grob nach ihre Höhe und optische Dicke abgeschätzt werden. Für eine präzisere Messung wird jedoch mindestens eine weitere Kamera benötigt, die aus einer anderen Position Aufnahmen von Wolken tätigt. Mit Hilfe von Bildinformationen und Angaben über die innere (Kamerakonstante, Hauptpunkt, Objektivverzeichnung) und äußere (Standort, Winkel) Orientierung beider Kameras, können die 3D-Koordinaten beliebiger Wolkenpunkte, die in beiden Bildern abgebildet sind, berechnet werden. Demzufolge kann durch Tracking der erkennbaren Wolkenpunkte aus den Bildsequenzen¹⁸, die Darstellung der 3D-Wolkenbewegung erfolgen. Der exakte Kalibrierungsvorgang ist in [24] beschrieben.

Die Messung der Wolkenhöhe kann ansonsten mit einen Laser-Ceilograph (auch Laser-Ceilometer genannt) geschehen. Dieser misst die Laufzeit eines Lichtimpulses eines Lasers, der vom Boden aus senkrecht zur Wolkendecke gesendet und zurückreflektiert wurde. Aus der gemessenen Laufzeit und der Lichtgeschwindigkeit kann die Höhe der Wolke berechnet werden.

¹⁸ Die Bildsequenz (auch Bildfolge genannt) steht für mehrere in gleichmäßigen Abständen aufgenommene Bilder

Eine weitere Methode zur automatischen Klassifizierung der Bildaufnahmen ist im Jahr 2010 in der wissenschaftlichen Publikation Heinle et al. [25, pp. 557-567] abgedruckt worden. In dieser Methodik werden sieben Klassen unterschieden (siehe Tabelle 4.1), die sich sowohl nach WMO als auch an optische Darstellung der Wolken auf Bildern orientieren. Dieser Algorithmus beruht auf das k-nächster-Nachbar-Verfahren (kNN) nach [26]. Dafür sind zunächst Wolkenbilder nötig, die jeweils zu einer der Klassen gehören.

Klasse	Wolkentyp
1	Cumulus
2	Cirrus/Cirrostratus
3	Cirrocumulus/Alto cumulus
4	Wolkenfrei (nicht mehr als 10% bedeckt)
5	Stratocumulus
6	Stratus
7	Cumulo-Nimbus/Nimbostratus

Tabelle 4.1 Wolkenklassen nach Heinle et al. (2010). Ihre Merkmale sind im Kapitel 3.1.2 aufgelistet

Mit diesen Wolkenbildern wird das Programm trainiert, deswegen müssen sie unbedingt mit der eingesetzten Kamera gemacht werden. Die Trainingsbilder von anderen Kameras würden zu einer Fehlkonfiguration führen, und somit die Klassifizierung verfälschen.

Folglich werden aus diesen Bildern Trainingsdaten erzeugt, die die Eigenschaften der jeweiligen Klasse beschreiben. Für die Unterscheidung der dunklen und optisch dicken Wolken von den dünnen Wolken werden die Farbeigenschaften der Wolkenpixel nach arithmetischem Mittelwert und Standardabweichung der Kanäle Rot und Blau, Schiefe und Differenz der RGB-Werte untersucht. Die Untersuchung nach Energie, Entropie, Kontrast und Homogenität der Grauwerte gibt Auskunft über die Helligkeitsverteilung des Bildes. Die hierfür verwendeten Formeln sind aus dem [25, pp. 557-567] zu entnehmen. [27, pp. 26-30] Als Nächstes werden die Ergebnisse dieser Berechnungen in einem Raum platziert, deren Dimension von der Anzahl der Eigenschaften der Klassen abhängt. Das heißt, bei n Eigenschaften,

besteht der Raum aus n Dimensionen. Nun können die Testdaten ebenfalls platziert und mittels einer Distanzfunktion (z.B. euklidische Distanz) die k nächsten Nachbarn berechnet werden. Bei einer gleichmäßigen Gewichtung wird ein Testwert der Klasse zugewiesen, welche die größte Anzahl der Trainingsdaten dieser k Nachbarn hat. Eine ungleichmäßige Gewichtung hingegen würde dafür sorgen, dass die Distanz der jeweiligen Merkmal in der Dimension zu- oder abnimmt. Bei einer gleichen Anzahl der Trainingsdaten aus mehreren Klassen, dominiert die Klasse mit dem naheliegenden Trainingswert. In Abbildung 4.2 wird diese Methode anhand eines Beispiels dargestellt. Da in dieser Arbeit die Klassifizierung der Wolken nicht realisiert wurde, wurden die Test- und Trainingsdaten von einem Zufallsgenerator erzeugt.

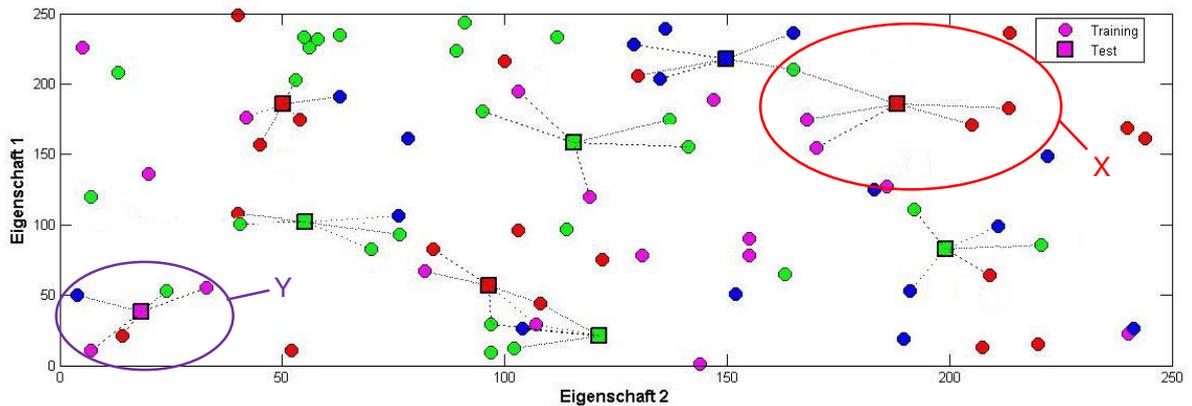


Abbildung 4.2 Beispiel: Klassifizierung nach k -nächster-Nachbarn Verfahren mit euklidischer Distanzmessung. $k = 5$, 80 Trainingsdaten (Kreise), neun Testdaten (Quadrate) und vier Klassen (Grün, Blau, Pink, Rot). Die Test- und Trainingsdaten wurden durch Zufallsprinzip erstellt

Es ist zu erkennen, dass bei zwei Eigenschaften (z.B. Mittelwert von Rot und Mittelwert von Blau) die Test- und Trainingsdaten in einem 2D-Raum abgebildet werden. Die vier unterschiedlichen Farben stellen hier vier verschiedene Klassen (z.B. Cumulus, Cirrus/Cirrostratus, wolkenfrei oder Stratus) dar. Für den Fall X wurde der Testwert zur Klasse Rot zugeteilt, weil die Distanz eines roten Trainingswerts der k nächsten Nachbarn geringer ist als die von einem pinken Wert.

Im Fall Y wurde der Testwert zur Klasse Pink gruppiert, weil dieser die größte Anzahl der zugehörigen Trainingsdaten in der Menge k nächster Nachbarn hat.

In dieser Art der Klassifizierung ist es sehr wichtig den richtigen Wert für k auszuwählen. Wenn der Wert für k klein gewählt ist, besteht die Gefahr, dass Fehler in den Trainingsdaten die Klassifizierung verfälscht. Falls jedoch k zu groß gewählt wird, können Punkte mit großem Abstand zu Testwert in die Klassifikationsentscheidung einbezogen werden. Diese können ebenfalls die Klassifizierung verschlechtern.

In diesem Projekt wurde die Methode von Long und DeLuisi angewendet, weil dieser einfach zu implementieren ist und bei einem guten Schwellenwert T brauchbare Ergebnisse liefert. Im Kapitel 1 wird diese Methode softwaretechnisch umgesetzt.

4.3 Fehleranalyse

Bei der Auswertung der Kamerabilder kann der Wolken-Bedeckungsgrad aufgrund des Kameraobjektivs (Kapitel 2.1) fehlinterpretiert werden, deshalb ist es notwendig die Verzerrung der Bilder zu korrigieren. Falls die Analyse ohne Korrektur durchgeführt wird, müssen theoretisch die Bildpunkte unterschiedlich gewichtet werden, weil in den Aufnahmen einer Fisheye-Kamera die Objekte am Bildkreisrand kleiner dargestellt werden - obwohl sie eine größere Fläche umfassen - als die in der Mitte.

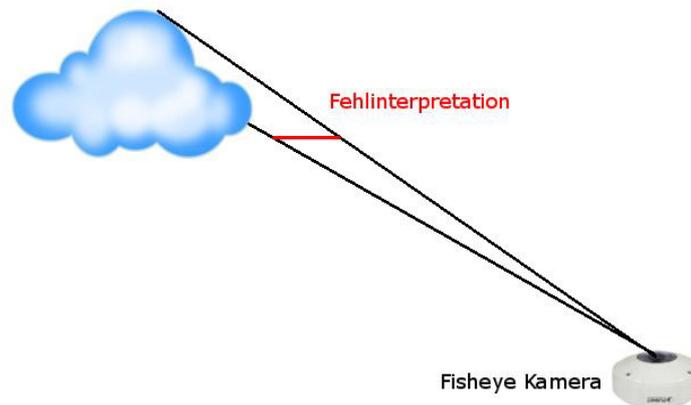
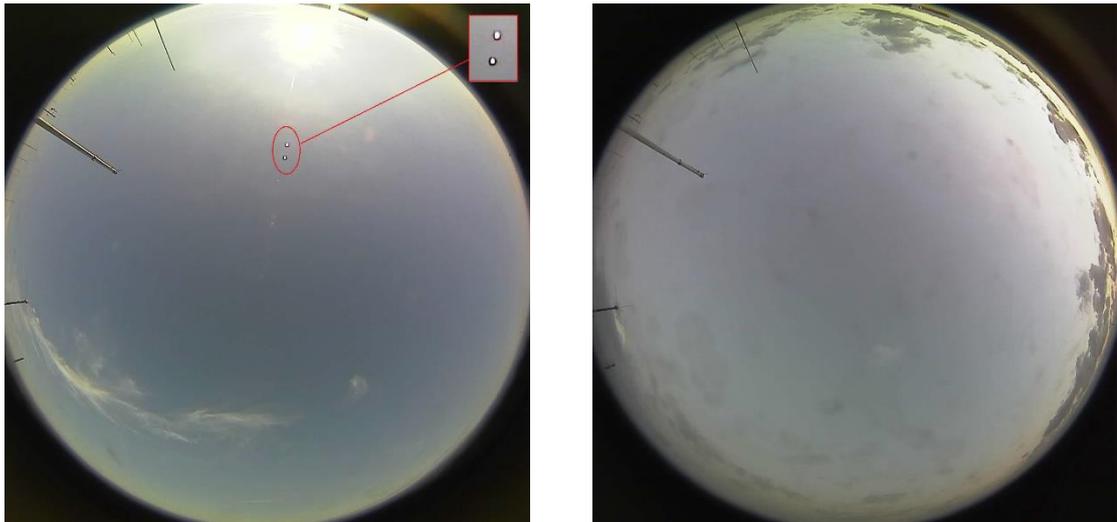


Abbildung 4.3 Fehlinterpertation des Wolken-Bedeckungsgrads bei leicht bedecktem Himmel

Bei leicht bedecktem Himmel werden die Wolkenseiten ebenfalls in die Bildebene projiziert, so dass Wolken größer dargestellt werden, siehe Abbildung 4.3. Die hohe Gewichtung dieser Bildpunkte haben gewaltige Folgen für den Wolken-Bedeckungsgrad. Aufgrund dessen ist es sinnvoll, bei leicht bedecktem Himmel, die Gewichtung zu vernachlässigen. Bei einem komplett bedeckten Himmel spielt eine unterschiedliche Gewichtung ebenfalls keine Rolle.

Weitere Berechnungsfehler können dann auftreten, wenn die Sonne auf dem Bild zu erkennen ist. Bildpunkte, die nah an der sichtbaren Sonne sind, werden nahezu weiß dargestellt und können vom Programm als Wolke angenommen werden, siehe Abbildung 4.4a. Dieser Effekt tritt auf, wenn das Licht an Dunst und Staubpartikel in der Erdatmosphäre stark gestreut wird. Die Reflexion der Sonnenstrahlen an der Acrylglaskuppel (Abbildung 4.4a) kann im Bild weiße Flecken verursachen. Diese gehen in der Berechnung ebenfalls als Fehler ein.



(a)

(b)

Abbildung 4.4 Ursachen für eine Fehlinterpretation der Bildpunkte; (a) wurde am 04.12.2014 um 12:14 Uhr aufgenommen. Sonne und nahliegende Pixel werden nahezu weiß dargestellt. Die Reflexion der Sonnenstrahlen an der Acrylglaskuppel verursacht im Bild weiße Flecken; (b) wurde am 10.12.2014 um 16:19 Uhr aufgenommen. Aufgrund der Helligkeit des Bildes können wolkenfreie Bereiche des Himmels als Wolke interpretiert werden

Ein weiteres Problem liegt an der Helligkeit des Bildes. Bei einer dunklen Bildaufnahme (z.B. Abbildung 4.4b) können Bereiche des Bildes, die nicht zu einer Wolke gehören, als Wolke dargestellt werden. Wolkenerkennungsfehler können auch entstehen, wenn Regen und Schnee die Aufnahme der Kamera behindern oder Flugobjekte und Vögel die Sicht beeinträchtigen, siehe Abbildung 4.5.



*Abbildung 4.5 Ein Vogel, der die Sicht der Kamera beeinträchtigt. Aufnahme vom
17.11.2014 um 10:52 Uhr*

5 Bildspeicherung

In diesem Hauptabschnitt wird die Methode aufgeklärt, wie Aufnahmen einer Fisheye-Kamera strukturiert in einem Dateisystem abgelegt werden. Hierbei werden folgende Anforderungen realisiert:

- Der Benutzer gibt beim Starten des Programms über die Konsole den Zielpfad und die Programmlaufzeit an:

```
> (Programmname) (Pfad) (Programmlaufzeit in Tage)
```

Beispiel:

```
> Clouds /Users/Mustermann/Bilder/ 3
```

- Bereitstellung des Bildes in einem nützlichen Format für die Weiterverarbeitung:

Für die Importierung des Bildes wird der OpenCV-Datenstruktur *IplImage* verwendet. Da das Bild sich nicht auf dem lokalen Rechner befindet, sondern erst über das HTTP-Protokoll eingelesen werden muss, kann die OpenCV-Methode *cvLoadImage()* nicht in Anspruch genommen werden. Stattdessen muss eine eigene Methode implementiert werden, die diese Aufgabe erfüllt. Für dieses Projekt wurde die Methode *getImageFromIPcam()* implementiert, siehe Abbildung 5.1.

- Strukturierte Dateiablage:

Die Bilder werden hierarchisch in einem Zielpfad abgelegt. Es existiert ein Hauptordner mit der Bezeichnung „Fisheye_Camera“. In diesem Ordner wird täglich ein neuer Ordner erzeugt, der das aktuelle Datum als Namen trägt. Die Bilder, die am aktuellem Tag getätigt werden, werden den zuletzt erstellten Ordnern zugeordnet. Jedes Bild existiert in zwei Auffassungen. Das erste Bild ist die Originalaufnahme, die den Namen „Datum_Uhrzeit“ beinhaltet. Das zweite Bild hingegen ist das analysierte Bild, deshalb wird es folgendermaßen benannt „Datum_Uhr_Analyzed“.

- Die Abspeicherung der Bilder geschieht in einem Zeitintervall (z.B ein Bild pro Stunde):

Die OpenCV-Methode `cvSaveImage()` speichert ein beliebiges Bild ab. Sie erwartet die Parameter „Dateiname inklusive Zielpfad“ und „Bildstruktur von Typ `IplImage`“ um diesen Befehl auszuführen.

- Ausgabe von Start- und Endzeit und Anzahl der Bilder, die erfasst worden sind.

```
Visual Paradigm Enterprise Edition (University of Applied Sciences)
Downloader
- path : string
- dirName : string
- counter : int
- startDate : string
- endDate : string
- processingDaysCounter : int
- setprocessingDurationDays : int
- oldDay : int
- currentPath : string
- currentImage : Mat
- image : IplImage*

+ Downloader(path : string, setprocessingDurationDays : int)
+ Downloader()
+ ~Downloader()
+ download(flag : bool) : void
+ timeStamp() : string
+ getDate() : string
+ getDay() : int
+ getHour() : int
+ display() : void
+ getprocessingDaysCounter() : int
+ getCurrentPath() : string
+ getCurrentImage() : Mat
+ setImage(src : IplImage *) : void
+ getImageFromIPcam(CameraIP : char *) : IplImage *
```

Abbildung 5.1 Die Klasse *Downloader*. Sie ist für Bild importieren und speichern zuständig

In Abbildung 5.1 ist die Klasse *Downloader* abgebildet. Sie beinhaltet einen Standard-Konstruktor, einen Konstruktor mit zwei Eingangsparametern (Pfad und Dauer der Programmlaufzeit) und einen Destruktor. Die wichtigste Methode dieser Klasse ist die sogenannte *download()*. Sie erwartet einen Parameter von Typ Boolean, der im Algorithmus als einen Merker (hier: *flag*) genutzt wird. Falls der Merker den Wert *false* zugewiesen bekommt, versetzt sich die Methode in den Initialisierungszustand. Im Initialisierungszustand wird der Hauptordner „Fisheye_Camera“ und ein Unterordner mit aktuellem Datum erstellt. Besitzt jedoch

der Merker den Wert *true*, dann erfolgt die Abspeicherung des Bildes, siehe Abbildung 5.2.

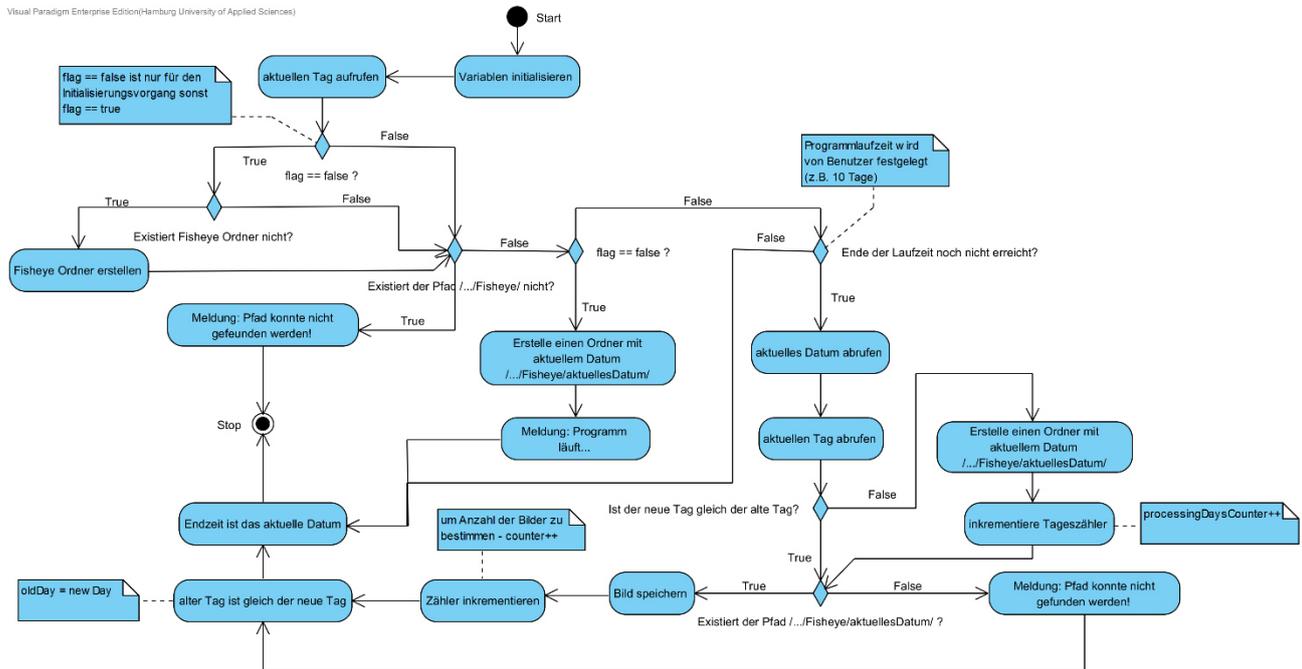


Abbildung 5.2 Aktivitätsdiagramm zur Methode *download()*; Im Anhang 1 wird folgendes Diagramm größer dargestellt

In Abbildung 5.2 ist das Aktivitätsdiagramm zur Methode *download()* zu entnehmen. Dieses beschreibt das Verhalten der Methode und gibt Auskunft über ihre jegliche Zustände. Sobald der Merker den Wert *true* aufweist, dann wird für den nächsten Tag ein neuer Ordner erstellt und die Abspeicherung der Bilder wird dort fortgesetzt.

Sowohl die Interaktion zwischen dem Benutzer und dem Hauptprogramm, als auch zwischen dem Hauptprogramm und der Klasse *Downloader* wird in Abbildung 5.3 mittels eines Sequenzdiagramms zur Schau gestellt. Bei der Kommunikation zwischen *Main* und der Klasse *Downloader* lässt sich der Abschnitt 3 gut bemerkbar machen. Hierbei fordert *Main* durch den Aufruf der Methode *getImageFromIPcam()* das importierte Bild. Die Methode der Klasse *Downloader* antwortet mit dem angeforderten Bild (Abschnitt 3.1). Anschließend übergibt *Main* das erhaltene Bild wieder zurück an die Klasse *Downloader*. Dieser Vorgang scheint merkwürdig zu

klingen, er ist aber notwendig, um die Originalaufnahme und ebenso das analysierte Bild im gleichen Pfad zu speichern. Das analysierte Bild wird in *Main* (siehe Abbildung 5.3, Abschnitt 5.1.1.2) und das Originalbild hingegen wird in der Methode *download()* (siehe Abbildung 5.2) abgespeichert.

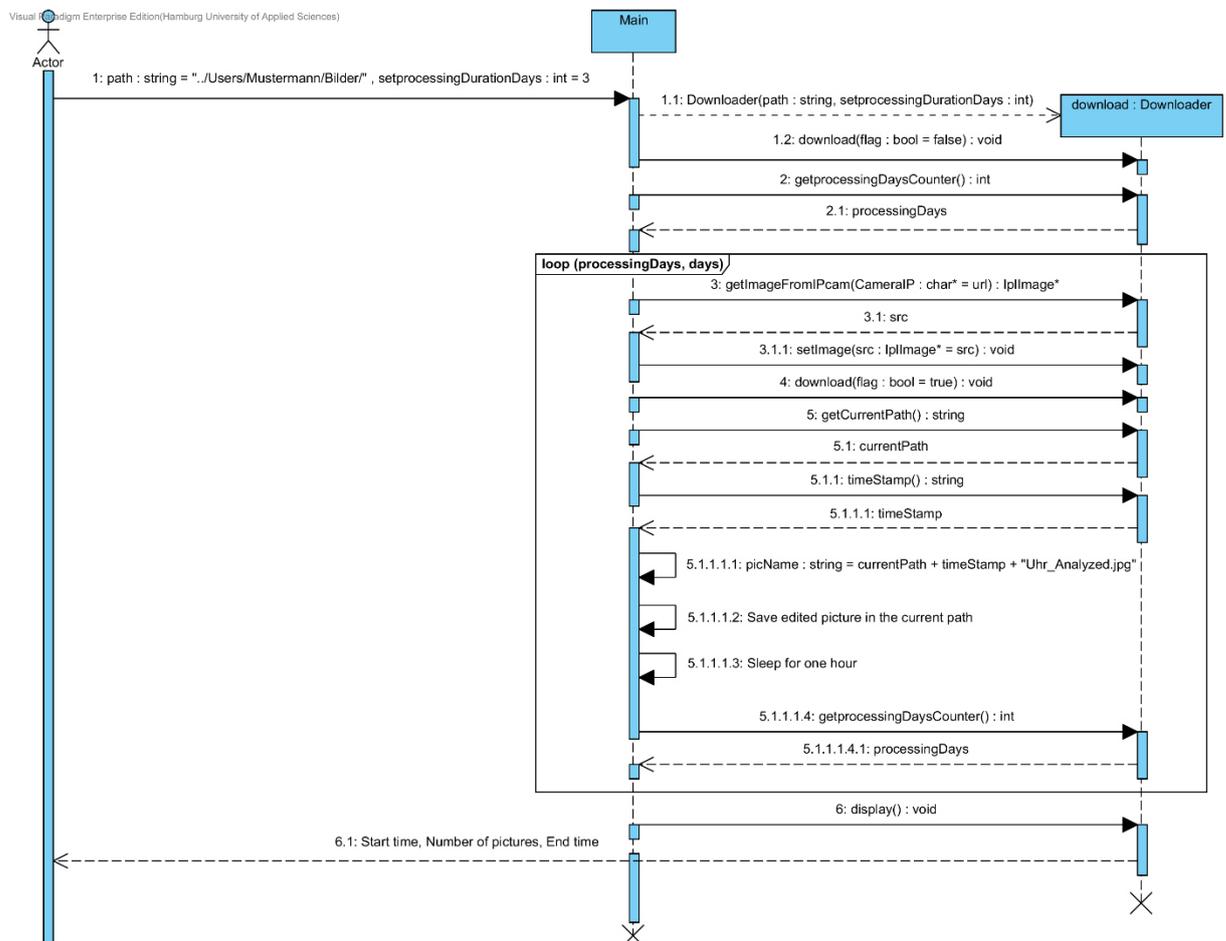
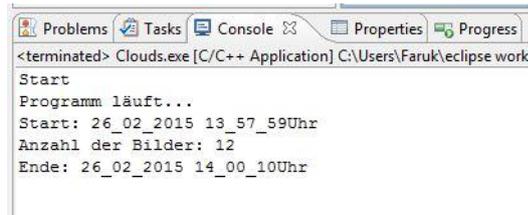


Abbildung 5.3 Sequenzdiagramm zur Bildspeicherung für eine drei tägige Programmlaufzeit; Im Anhang 2 wird folgendes Diagramm größer dargestellt

Es ist ebenfalls zu erkennen, dass *Main* den aktuellen Pfad fordert (Abschnitt 5), um im Nachhinein das analysierte Bild dort abzuspeichern.

Für den Testfall wurde das Programm leicht modifiziert. Das Zeitintervall wurde auf 10 Sekunden gesetzt, so dass in diesem Zeitraum nur ein Bild aufgenommen und weiterverarbeitet wird. Die Programmlaufzeit wurde auf drei Minuten gestellt und es

sollte nicht alle 24 Stunden ein neuer Ordner erstellt werden, sondern jede Minute. Theoretisch müssen insgesamt 18 Bilder aufgenommen werden. Die Abbildung 5.4 zeigt jedoch, dass in diesem Zeitraum nur 12 Bilder erstellt worden sind. Der Grund dafür liegt an der Ausführung der Funktionen, die sowohl für den Input und Output, als auch für die Bildverarbeitung, zuständig ist. Diese beanspruchen einen gewissen Moment und sorgen dafür, dass das Programm nicht zeitpräzise arbeitet.



```
<terminated> Clouds.exe [C/C++ Application] C:\Users\Faruk\workspace
Start
Programm läuft...
Start: 26_02_2015 13_57_59Uhr
Anzahl der Bilder: 12
Ende: 26_02_2015 14_00_10Uhr
```

Abbildung 5.4 Der Testfall

6 Kamerakalibrierung und Bildentzerrung

In diesem Kapitel werden als Erstes das OpenCV Kameramodell und anschließend das Kameramodell von Scaramuzza beschrieben. Zusätzlich wird erklärt, wie beide Modelle kalibriert werden. Ferner wird versucht, mit Hilfe von beiden Modellen Aufnahmen einer Fisheye-Kamera erfolgreich zu entzerren.

6.1 Das OpenCV Kameramodell

Das OpenCV Kameramodell basiert auf einem einfachen Modell einer Lochkamera, welches die Abbildungstheorie eines Objektpunkts im Weltkoordinatensystem auf einer 2D-Bildebene beschreibt [10, pp. 370-403] [28, pp. 1-11]. Dieses Modell beinhaltet zusätzlich eine Verzerrungsfunktion, um die von einer Linse entstandene Verzerrung modellieren zu können. Letztendlich wird als Resultat die Position des Objektpunkts in der Sensor-Ebene der realen Kamera ausgegeben.

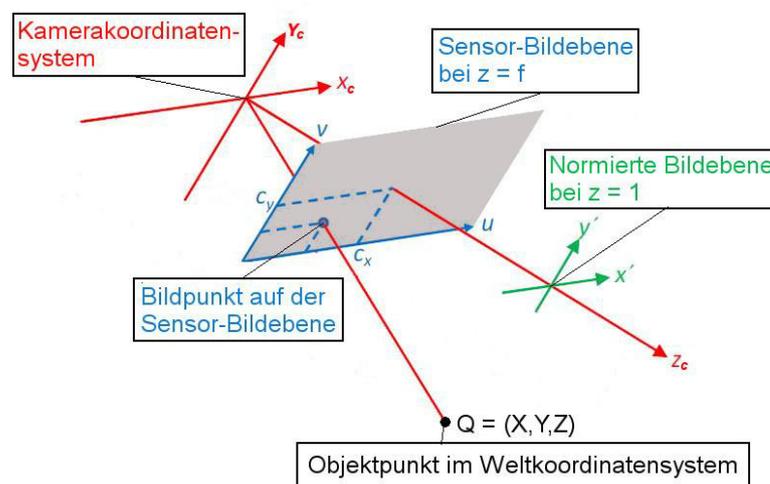


Abbildung 6.1 Abbildung eines Objektpunkts Q im Weltkoordinatensystem auf die Bildebene einer Lochkamera. Die Verzerrung wurde vernachlässigt [28, pp. 1-11]

In Abbildung 6.1 wird die ideale Projektion eines Objektpunkts im Weltkoordinatensystem auf die Bildebene einer Lochkamera grafisch dargestellt. Diese Darstellung enthält zunächst keine Verzerrung. Die Parameter c_x und c_y

beschreiben die Position des Hauptpunkts, an dem die z-Achse des Kamerakoordinatensystems die Sensor-Bildebene schneidet. Des Weiteren ist eine normierte Bildebene abgebildet, die sich bei $z_c = 1$ befindet. Das oben dargestellte Modell wird mathematisch folgendermaßen beschrieben:

Ein Objektpunkt Q mit der Position $(X \ Y \ Z)^T$ im Weltkoordinatensystem wird als Erstes durch extrinsische Transformation in das Kamerakoordinatensystem überführt. Dort erscheint der Punkt Q an der Position $(x_c \ y_c \ z_c)^T$.

$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = R_{3 \times 3} \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \vec{t} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \cdot \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} \quad (6.1)$$

Hierbei ist R eine 3x3 Drehmatrix und \vec{t} ein drei dimensionaler Verschiebevektor. Die Drehmatrix kann aus einem Rodriguez-Vektor \vec{r} ermittelt werden. Dieser Vektor definiert hier die Drehachse im Raum an, um die gedreht werden soll. Die Berechnung der Drehmatrix R aus einem Rodriguez-Vektor \vec{r} kann dermaßen geschehen:

Durch Normieren von \vec{r} errechnet sich der Einheitsvektor \vec{v} , der in Richtung der Drehachse zeigt. Der Drehwinkel θ hingegen lässt sich aus dem Betrag des Rodriguez-Vektors entnehmen, siehe Formel (6.2).

$$\theta = |\vec{r}| = \sqrt{r_x^2 + r_y^2 + r_z^2} \quad (6.2)$$

$$\vec{v} = \frac{\vec{r}}{\theta} \quad (6.3)$$

Als Nächstes wird die Kreuzproduktmatrix $[\vec{v}]_{\times}$ gebildet.

$$[\vec{v}]_{\times} = \sum_{i=1}^3 (\vec{v} \times \vec{e}_i) \otimes \vec{e}_i = \begin{pmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{pmatrix} \quad (6.4)$$

Die Vektoren \vec{e}_i sind drei verschiedene Einheitsvektoren. In (6.4) wird das Kreuzprodukt der Vektoren \vec{v} und \vec{e}_i gebildet. Anschließend wird aus dem Resultat das dyadische¹⁹ Produkt mit \vec{e}_i berechnet.

Aus dem Drehwinkel θ , den Formeln (6.3) und (6.4), und der 3x3-Einheitsmatrix I kann folgende Formel zur Berechnung von Rotationsmatrix aufgestellt werden:

$$R_{3 \times 3} = \cos(\theta) \cdot I_{3 \times 3} + \sin(\theta) \cdot [\vec{v}]_{\times} + (1 - \cos(\theta)) \cdot \vec{v} \cdot \vec{v}^T \quad (6.5)$$

Nachdem der Punkt Q auf dem Kamerakoordinatensystem an die Position $(x_c \ y_c \ z_c)^T$ überführt wurde, erfolgt die ideale Projektion des Punkts an die Stelle $(x' \ y')^T$ in der Bildebene mit der Brennweite $f = 1$, siehe Abbildung 6.2.

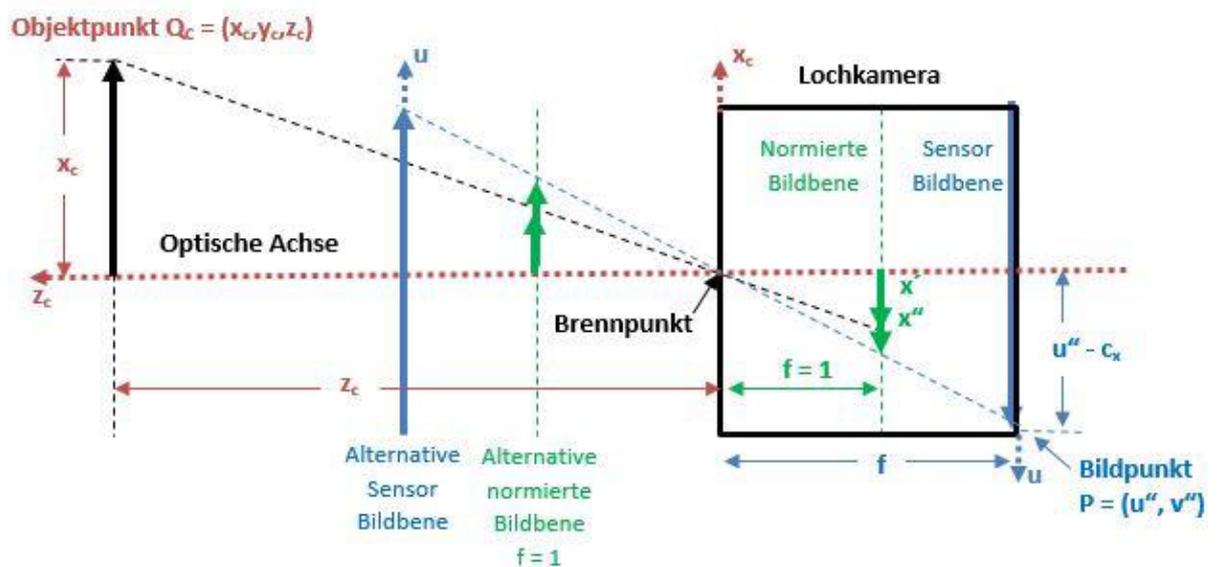


Abbildung 6.2 Abbildung eines Objektpunkts Q im Weltkoordinatensystem auf die Bildebene einer Lochkamera in einer anderen Darstellung [28, pp. 1-11]

¹⁹ Ist ein spezielles Produkt zwischen zwei Vektoren. Das Ergebnis des dyadischen Produkts ist eine

$$\text{Matrix: } \vec{a} \otimes \vec{b} = \vec{a} \cdot \vec{b}^T = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \cdot (b_1 \ b_2 \ b_3) = \begin{pmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 \\ a_2 b_1 & a_2 b_2 & a_2 b_3 \\ a_3 b_1 & a_3 b_2 & a_3 b_3 \end{pmatrix}$$

Dieser Ansatz wird folgenderweise zum Ausdruck gebracht:

$$\frac{x_c}{z_c} = \frac{x'}{f} \quad (6.6)$$

$$x' = f \cdot \frac{x_c}{z_c} \quad (6.7)$$

Für y' gilt die gleiche Bedingung wie bei (6.6). Wenn für die Brennweite $f = 1$ eingesetzt wird, dann folgen aus (6.7):

$$x' = \frac{x_c}{z_c} \quad y' = \frac{y_c}{z_c} \quad (6.8)$$

Es ist zu bemerken, dass $(x' \ y')^T$ keine Verzerrung beinhaltet. Die Verzerrung wird zunächst mit den Gleichungen (6.9) und (6.10) modelliert.

$$x'' = x' \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) + 2 \cdot p_1 \cdot x' \cdot y' + p_2 \cdot (r^2 \cdot x'^2) \quad (6.9)$$

$$y'' = y' \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) + p_1 \cdot (r^2 + 2 \cdot y'^2) + 2 \cdot p_2 \cdot x' \cdot y' \quad (6.10)$$

Hierbei ist $r^2 = x'^2 + y'^2$, die Parameter k_1, k_2, k_3 sind die radialen²⁰ und p_1, p_2 die tangentialen²¹ Verzerrungskoeffizienten.

Nun wird die verzerrte, normierte Bildposition $(x'' \ y'')^T$ mittels intrinsische Transformation in die Sensor-Bildebene überführt. Dort erscheint dieser an der Stelle $(u'' \ v'')^T$, siehe Gleichungen (6.11) und (6.12).

$$u'' = f_x'' \cdot x'' + c_x'' \quad (6.11)$$

$$v'' = f_y'' \cdot y'' + c_y'' \quad (6.12)$$

²⁰ Bei einer radialen Verzerrung werden die Bildpunkte vom Focus hinweg oder in Richtung Focus kreisförmig verzerrt. Diese Art von Verzerrung wird in zwei Kategorien aufgeteilt: Ist der Abstand des Bildpunktes zum Focus kleiner als der ideal projizierte Bildpunkt, dann handelt es sich um einen kissenförmigen Verzerrung. Der andere Fall wird als tonnenförmige Verzeichnung bezeichnet

²¹ Bei einer tangentialen Verzerrung wird der Abstand des Bildpunktes zum Focus entlang einer Tangente skaliert

Die Parameter c_x'' , c_y'' geben die Position des Hauptpunkts und f_x'' , f_y'' die Bildweite als Anzahl von Pixelbreiten²² in x- und y-Richtung an.

6.1.1 Kalibrierung mit OpenCV

Eine Kamerakalibrierung ist notwendig, um die unbekannt Parameter (f_x'' , f_y'' , c_x'' , c_y'' , k_1 , k_2 , k_3 , p_1 , p_2) des obengenannten Kameramodells bestimmen zu können. Diese Parameter können anschließend für nutzbare Zwecke genutzt werden, wie z.B. Bildentzerrung. OpenCV nutzt das Verfahren von Zhang [29, pp. 1330-1334], um diese neun Unbekannten ermitteln zu können.

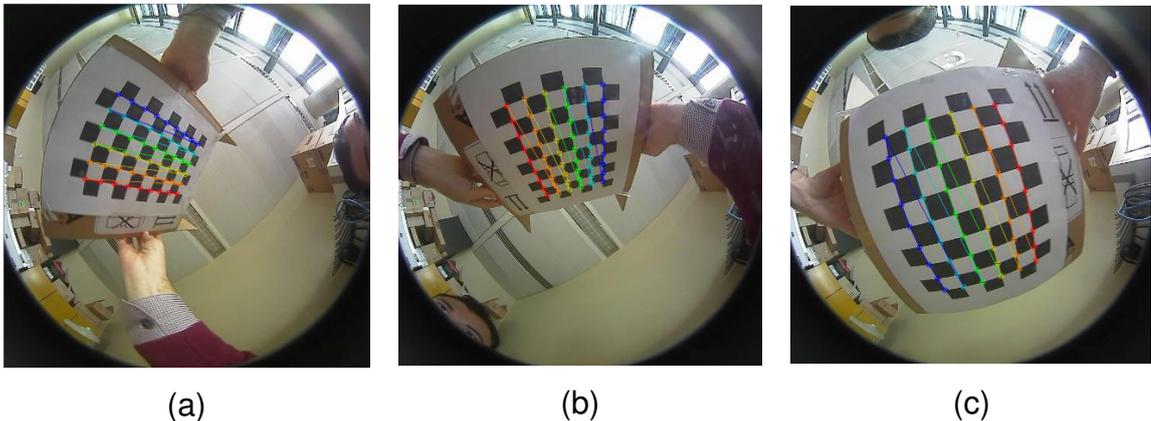


Abbildung 6.3 Aufnahmen des Schachbrettmusters für die Berechnung der Kameraparameter mittels OpenCV nach Zhang [29, pp. 1330-1334]

In diesem Verfahren wird das Schachbrettmuster in verschiedenen Ansichten vor der Kamera platziert, siehe Abbildung 6.3. Um gute Ergebnisse zu erzielen, werden Aufnahmen von mindestens zehn Ansichten benötigt. Aus diesen Aufnahmen werden anhand der Methoden der Bildverarbeitung die Koordinaten der Eckpunkte des Musters ermittelt. Die gefundenen Koordinaten sind markante Positionen im Weltkoordinatensystem. Sie werden danach an dem Verfahren von Zhang

²² Der Abstand von zwei benachbarten Pixeln

übergeben werden, welches sowohl für jedes Bild die extrinsischen Parameter, als auch die Verzerrungsparameter der Linse und die intrinsischen Kameraparameter mittels Schätzverfahren errechnet. [28, pp. 1-11] Dieses erfolgt mit Hilfe der Methode `cvCalibrateCamera2()`. Für die drei Aufnahmen, die in Abbildung 6.3 zu sehen sind, wurden folgende Parameter berechnet:

Intrinsische Parameter:

$$f_x'' = 658.58837607737166$$

$$f_y'' = 652.91953340070279$$

$$c_x'' = 767.5$$

$$c_y'' = 767.5$$

Radiale Verzerrungsparameter:

$$k_1 = -0.24798314030533794$$

$$k_2 = 0.0148618822591075 \quad k_3 = 0$$

Tangentiale Verzerrungsparameter:

$$p_1 = 0$$

$$p_2 = 9.5677912018302339 \cdot 10^{-3}$$

Extrinsische Parameter:

Abbildung 6.3a

$$\vec{r} = \begin{pmatrix} 1.2378488449951475 \\ 0.39150344200890191 \\ -2.5186305509373819 \end{pmatrix}$$

$$\vec{t} = \begin{pmatrix} -24.335155601810179 \\ 19.997879146064538 \\ 346.48663637203094 \end{pmatrix}$$

Abbildung 6.3b

$$\vec{r} = \begin{pmatrix} 0.85715270886677208 \\ 0.56501894531835894 \\ -1.6586301710784448 \end{pmatrix}$$

$$\vec{t} = \begin{pmatrix} -70.547774821476096 \\ 22.680303175252458 \\ 319.40627089249131 \end{pmatrix}$$

Abbildung 6.3c

$$\vec{r} = \begin{pmatrix} 0.01.338334321761881 \\ -0.14243911990209673 \\ 1.2643765830457652 \end{pmatrix}$$

$$\vec{t} = \begin{pmatrix} 27.570703638269066 \\ -88.614135136262490 \\ 148.20183963762204 \end{pmatrix}$$

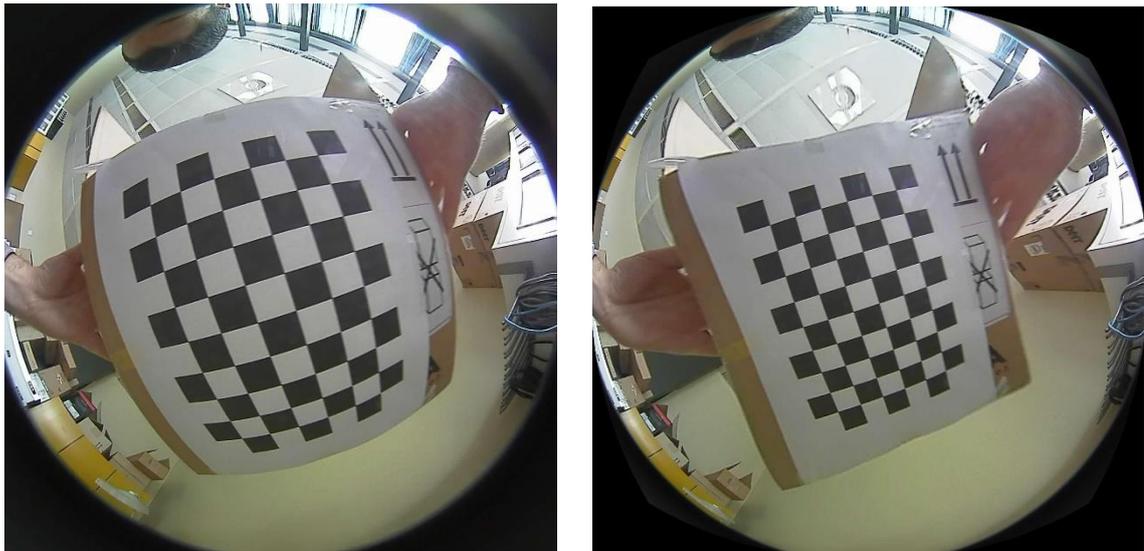
6.1.2 Bildentzerrung mit OpenCV

Nachdem der Kalibrierungsprozess beendet ist, kann mittels extrinsischen, intrinsischen und Verzerrungsparametern ein verzerrtes Bild korrigiert werden. OpenCV bietet Methoden (z.B. `cvUndistort2()`) an, die anhand der Kamera-Parameter die Entzerrung vornimmt. Diese Methoden berechnen die entzerrten Positionen der Pixel im verzerrten Bild. Ein mathematisches Beispiel für dieses Vorgehen ist in [28, pp. 1-11] konzipiert. Der wichtigste Schritt ist dabei, dass aus der verzerrten Koordinate $(x'' \ y'')^T$ die entzerrte $(x' \ y')^T$ berechnet wird. Um das zu bewerkstelligen, müssen die inversen Funktionen der Gleichungen (6.9) und (6.10) bestimmt werden. Sie lassen sich jedoch nicht berechnen, deshalb wird die entzerrte Koordinate iterativ mit dem zweidimensionalen Newton-Verfahren ermittelt, siehe [28, pp. 1-11]. Als Nächstes wird sie in die normierte Bildebene $(x'_s \ y'_s)^T$ transformiert, die sich im Sichtbereich einer, senkrecht zum Weltkoordinatensystem, ausgerichteten Kamera befindet:

$$\begin{pmatrix} x'_h \\ y'_h \\ z'_h \end{pmatrix} = R^T_{3 \times 3} \cdot \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \quad (6.13)$$

$$\begin{pmatrix} x'_s \\ y'_s \end{pmatrix} = \begin{pmatrix} \frac{x'_h}{z'_h} \\ \frac{y'_h}{z'_h} \end{pmatrix} \quad (6.14)$$

Abschließend wird die entzerrte Bildkoordinate $(x'_s \ y'_s)^T$ in der normierten Bildebene in die Sensor-Bildebene $(u' \ v')^T$ überführt.



(a) Original

(b) Entzerrt

Abbildung 6.4 Das Ergebnis der Bildentzerrung mit OpenCV

In Abbildung 6.4 ist das Resultat der Bildentzerrung zu veranschaulichen. Es ist zu erkennen, dass das Originalbild (Abbildung 6.4a) nicht ordnungsgemäß entzerrt wurde, siehe Abbildung 6.4b. Bei kleinen Bildwinkel hat der Algorithmus geringe Probleme das Bild zu entzerren, bei größeren jedoch ist die Entzerrung erfolglos. Daher ist dieses Modell für Fisheye-Kameras mit großem Bildwinkel ungeeignet.

6.2 Das Kameramodell von Scaramuzza

Im Kapitel 6.1 wurde bereits erwähnt, dass das einfache Lochkameramodell nicht in der Lage ist, Blickwinkel, die größer und gleich 180° sind, mathematisch richtig darzustellen. Davide Scaramuzza ist diesem Problem nachgegangen. Er veröffentlichte im Jahr 2006 in einem Dokument [30] und im Jahr 2008 in seiner Dissertation [31] ein neues Kameramodell, welches auch für größere Blickwinkel tauglich ist. In diesem Modell wird jeder Bildpunkt durch einen 3D-Vektor dargestellt, der ausgehend von der effektiven Sicht der Kamera, auf den entsprechenden Objektpunkt im Weltkoordinatensystem zeigt. Dieser 3D-Vektor kann normiert

werden, so dass die dreidimensionale Darstellung eines Bildpunktes auf der Oberfläche einer imaginären Einheitskugel liegt, siehe Abbildung 6.5.

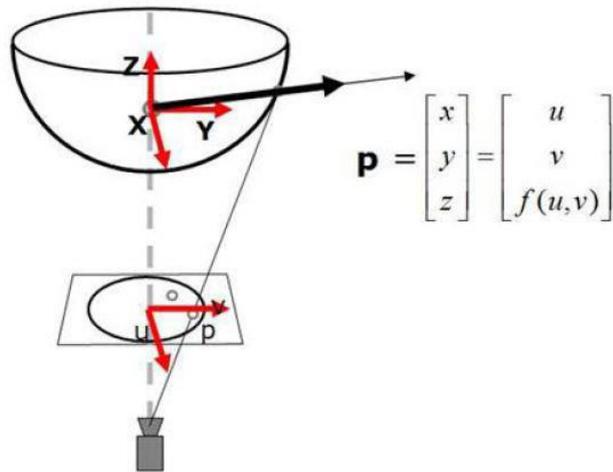


Abbildung 6.5 Das Scaramuzza Kameramodell [32]

Der 3D-Vektor $(x \ y \ z)^T$ eines Bildpunktes (u, v) wird durch folgende Gleichung beschrieben [30]:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} u \\ v \\ f(u, v) \end{pmatrix} \quad (6.15)$$

Es ist zu beachten, dass (u, v) Pixelkoordinaten sind, die von der Bildmitte ausgehen. Im nächsten Schritt kann der 3D-Vektor normiert werden, damit die Projektion auf einer imaginären Einheitskugel erfolgt [33, pp. 5-8]:

$$\begin{pmatrix} x_s \\ y_s \\ z_s \end{pmatrix} = \begin{pmatrix} \frac{x}{\sqrt{u^2 + v^2 + f(u, v)^2}} \\ \frac{y}{\sqrt{u^2 + v^2 + f(u, v)^2}} \\ \frac{z}{\sqrt{u^2 + v^2 + f(u, v)^2}} \end{pmatrix} \quad (6.16)$$

Die Funktion $f(u, v)$ - anders formuliert $f(\rho)$, mit $\rho = \sqrt{u^2 + v^2}$ als euklidischer Distanz der Pixelposition (u, v) , beginnend in der Bildmitte – ist ein Taylorpolynom mit den Koeffizienten a_i , die die intrinsischen Parameter der Kamera beinhaltet. Sie modelliert gleichzeitig die radiale Verzerrung der Linse. Sie wird für die Vorwärtsprojektion benötigt. [32]

$$f(\rho) = a_0 + a_1 \cdot \rho + a_2 \cdot \rho^2 + a_3 \cdot \rho^3 + a_4 \cdot \rho^4 + \dots + a_n \cdot \rho^n \quad (6.17)$$

Um einen dreidimensionalen Punkt, der auf der imaginären Einheitskugel (x_s, y_s, z_s) liegt, auf die Bildebene reproduzieren zu können (z.B. um ein Fisheye-Bild zu entzerren), wird die inverse Funktion der Gleichung (6.17) benötigt, die ebenfalls als ein Taylorpolynom $\rho(\theta)$ dargestellt wird:

$$\rho(\theta) = b_0 + b_1 \cdot \theta + b_2 \cdot \theta^2 + b_3 \cdot \theta^3 + b_4 \cdot \theta^4 + \dots + b_n \cdot \theta^n \quad (6.18)$$

In diesem Fall ist θ der Winkel des Lichtstrahls zu der z-Achse.

$$\theta = \arctan\left(\frac{z_s}{\sqrt{x_s^2 + y_s^2}}\right) \quad (6.19)$$

Aus der Gleichung (6.16) ergibt sich folgende Gleichung für die Reproduktion eines Punkts in einem dreidimensionalen Raum:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} x_s \cdot \frac{\rho(\theta)}{\sqrt{x_s^2 + y_s^2}} \\ y_s \cdot \frac{\rho(\theta)}{\sqrt{x_s^2 + y_s^2}} \end{pmatrix} \quad (6.20)$$

Wie bereits oben mehrmals erwähnt worden ist, werden die Koordinaten (u, v) ausgehend von der Bildmitte gezählt. Für die praktische Anwendung muss jedoch die Zählung von links oben des Bildes beginnen. Dafür gibt Scaramuzza folgende affine Transformation an, um die echten Koordinaten des Bildes zu bestimmen [32]:

$$\begin{pmatrix} u' \\ v' \end{pmatrix} = \begin{pmatrix} c & d \\ e & 1 \end{pmatrix} \cdot \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} x'_c \\ y'_c \end{pmatrix} \quad (6.21)$$

Diese Transformation kann auch invertiert werden [33, pp. 5-8]:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{c - d \cdot e} \begin{pmatrix} 0 & -d \\ -e & c \end{pmatrix} \cdot \begin{pmatrix} u' - x'_c \\ v' - y'_c \end{pmatrix} \quad (6.22)$$

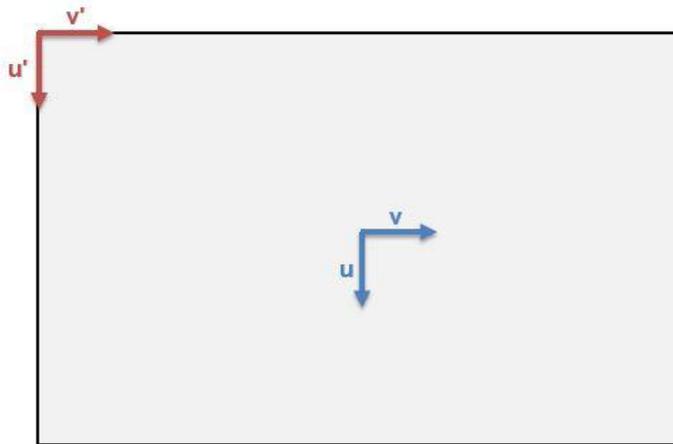


Abbildung 6.6 Bildkoordinaten, beginnend in der Bildmitte (u, v) und oben links (u', v')

6.2.1 Kalibrierung mit Scaramuzza

In dem Kalibrierungsprozess werden die Koeffizienten der Polynome $f(\rho)$ und $\rho(\theta)$, die Koordinaten der Bildmitte (x'_c, y'_c) und die Parameter der affinen Transformation (c, d, e) berechnet. Dieser wird mit Scaramuzza's Omnidirectional Camera Calibration Toolbox [32] für Matlab²³ gefertigt. Um die Kalibrierung mit diesem Tool durchzuführen, wird - sowie bei einer Kalibrierung mit OpenCV - eine

²³ Matlab ist eine Software zur Lösung mathematischer Probleme und zur Wiedergabe der Ergebnisse. Sie ist hauptsächlich für numerische Berechnungen ausgelegt

Reihe von Aufnahmen von einem Schachbrettmuster benötigt. Bezogen auf die Lage der Kanten zwischen den schwarzen und weißen Quadranten, berechnet das Programm die gewünschten Parameter der Kamera aus.

Um einen Vergleich zwischen dem OpenCV und Scaramuzza Kameramodell herzustellen, werden für die Kalibrierung zuvor aufgenommene Bilder, die für OpenCV verwendet wurden, ebenfalls hier angewendet. Das Tool kalkuliert anschließend die gesuchten Parameter. In Abbildung 6.7 wird die Funktion der Vorwärtsprojektion grafisch dargestellt.

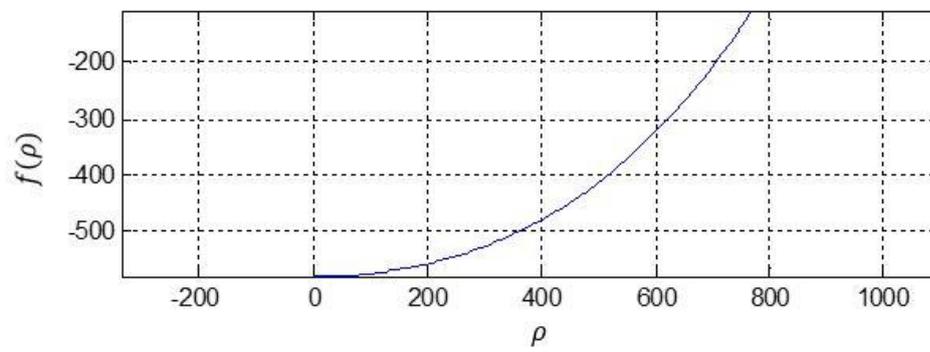


Abbildung 6.7 Funktion $f(\rho)$ für die Vorwärtsprojektion in Abhängigkeit von Distanz ρ in Pixeln zum Bildmittepunkt

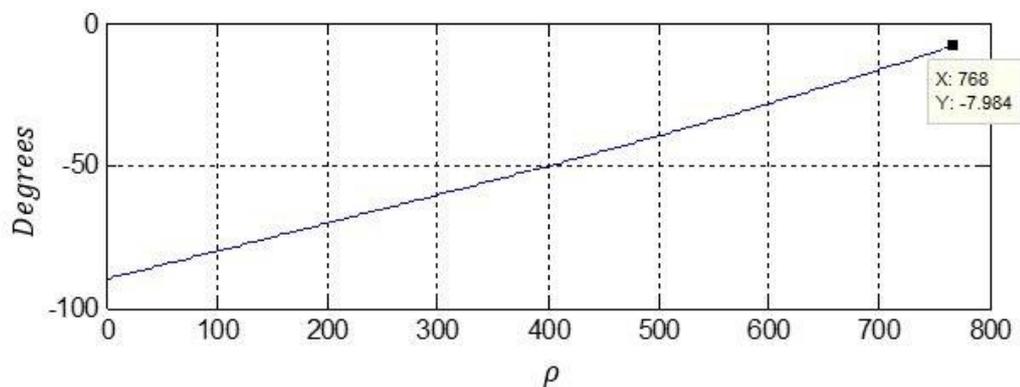


Abbildung 6.8 Bildwinkel, in Abhängigkeit von Distanz ρ in Pixeln zum Bildmittelpunkt

In Abbildung 6.8 wird der Bildwinkel in Bezug auf die Distanz ρ zum Bildmittelpunkt abgebildet. Es ist zu erkennen, dass der Winkel von der Bildmitte nach außen hin nicht 90° , sondern $82,016^\circ$ beträgt. In diesem Fall würde das Sichtfeld $180^\circ - 15,968^\circ$ betragen. Laut Datenblatt hat die Kamera jedoch ein Sichtfeld von $180^\circ \pm 5^\circ$.

Das Ergebnis der Kalibrierung kann hier veranschaulicht werden:

Koeffizienten der Funktion $f(\rho)$:

- $a_0 = 5$
- $a_1 = -581.7866$
- $a_3 = 0$
- $a_4 = 5.292157 \cdot e^{-4}$
- $a_5 = 1.283292 \cdot e^{-7}$
- $a_6 = 2.983718 \cdot e^{-10}$

Koeffizienten der Funktion $\rho(\theta)$:

- $b_0 = 10$
- $b_1 = 831.221285$
- $b_2 = 453.954688$
- $b_3 = 13.065970$
- $b_4 = 86.658009$
- $b_5 = 27.973631$
- $b_6 = 19.412660$
- $b_7 = 20.494012$
- $b_8 = -8.976375$
- $b_9 = -14.289296$
- $b_{10} = -3.620456$

Koordinaten der Bildmitte:

- $x'_c = 762.666079$
- $y'_c = 761.535359$

Parameter der affinen Transformation:

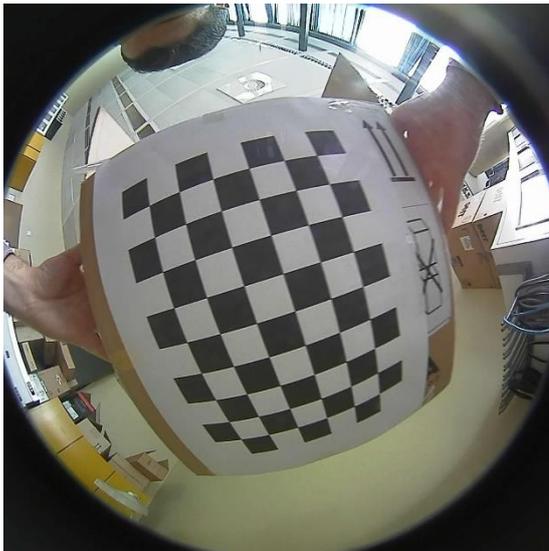
- $c = 0.998291$
- $d = -0.000825$
- $e = -0.00064$

Diese Parameter können nun für die Bildverzerrung genutzt werden.

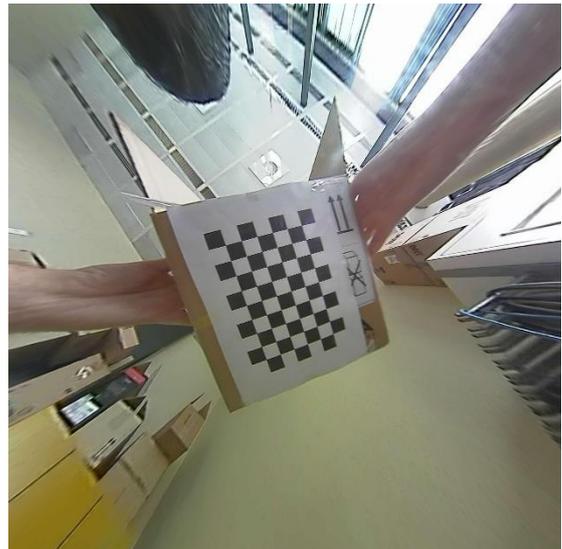
6.2.2 Bildentzerrung mit Scaramuzza

Die Bildentzerrung erfolgt mit Eclipse unter Verwendung von OpenCV Programmbibliothek. Scaramuzza stellt dafür Methoden zur Verfügung, die die Entzerrung ohne viel Aufwand einleiten. Die wichtigste Methode hierbei ist `world2cam()`. Sie transformiert mittels Kalibrierungsparameter einen 3D-Punkt in eine zweidimensionale Koordinate. Diese Funktion wird in der Methode `create_perspective_undistortion_LUT()` aufgerufen, wofür einzelnen Koordinaten des Speichers reserviert werden.

Da dieser Vorgang reversibel ist, bietet Scaramuzza die Methode `cam2world()` an, um die Transformation rückgängig zu machen. Das bedeutet, dadurch kann eine zweidimensionale Koordinate in einen 3D-Vektor überführt werden.



(a) Original



(b) Entzerrt

Abbildung 6.9 Das Ergebnis der Bildentzerrung mit Scaramuzza, $SF = 7$

In Abbildung 6.9 ist das Ergebnis der Entzerrung mit $SF = 7$ (Skalier Faktor) zu sehen. Mit dem Skalierfaktor wird die Entfernung der Kamera zu der zweidimensionalen Bildebene festgelegt. Je größer SF ist, desto größer ist die Entfernung. Bei $SF = 7$ ist die Entzerrung äußerst ordentlich. Problematisch ist jedoch, dass nicht alle Objekte aus dem Originalbild auf dem entzerrten Bild

dargestellt werden können. Des Weiteren ist es zu erkennen, dass die Objekte, die sich im äußeren Bereich des Bildes und nahe an der Kamera aufhalten, unscharf abgebildet werden. Dieser Effekt tritt auf, weil bei dem Entzerr-Vorgang die Pixel neu geordnet und anschließend durch die bilineare Interpolation ihre Nachbarn bestimmt werden.



Abbildung 6.10 Das Ergebnis der Bildentzerrung mit Scaramuzza, $SF = 11$

Bei $SF = 11$ kommt es zu keinerlei Informationsverlusten, siehe Abbildung 6.10. Jedoch wird das Bild dadurch unschärfer und Objekte im äußeren Bereich werden sehr verzerrt abgebildet. Diese Art von Verzerrung war im Grunde genommen zu erwarten. Zwar hat sich die Kamera von der Bildebene weiter entfernt, jedoch hat die Distanzierung nicht direkt im Raum stattgefunden. Vielmehr hat die Software aus den bestehenden Bildinformationen das Bild neu berechnet.

Beide Konfigurationen haben gewisse Nachteile. Für die Wolkenanalyse und Wolkendetektion ist es sehr wichtig, dass es zu keinerlei Informationsverlusten

kommt. Anbei ist es ebenfalls relevant, dass das Bild nicht unscharf und verzerrt ist, weil dadurch Pixel fehlinterpretiert werden können oder Objekte viel größer dargestellt werden als sie es sind. Da in dieser Anwendung die verlustfreie Abbildung eine höhere Priorität besitzt, ist es sinnvoll mit der Einstellung $SF = 11$ weiterzuarbeiten. Allerdings machen Unschärfe und Verzerrung das Bild unbrauchbar, siehe Abbildung 6.12b.

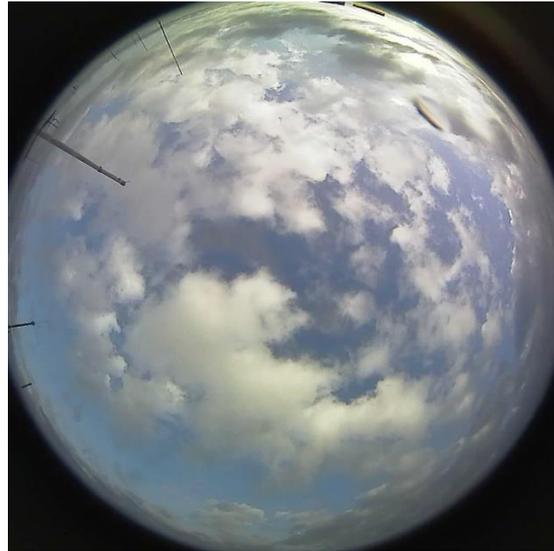
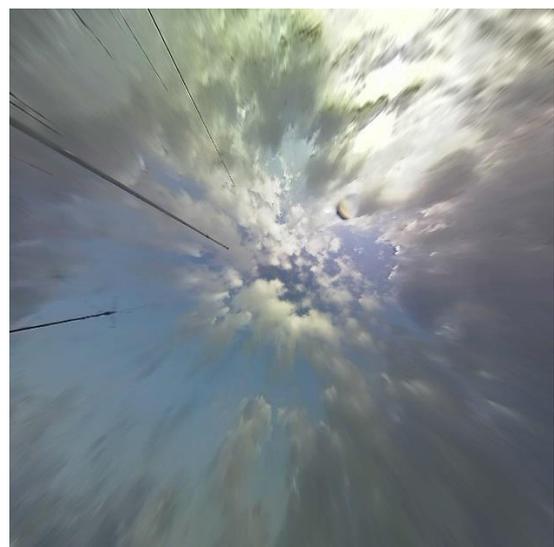


Abbildung 6.11 Aufnahme vom 10.12.2014 14:04 Uhr (teils bewölkter Himmel)



(a) $SF = 7$



(b) $SF = 11$

Abbildung 6.12 Entzerrung der Aufnahme vom 10.12.2014 14:04 Uhr

In Abbildung 6.12a lässt sich der Informationsverlust wieder bestätigen. Das Bild ist aber auch leicht unscharf und verzerrt.

Durch die Entzerrung verliert das Bild an Schärfe und signifikante Bildinformationen gehen verloren. Weil dies einen großen Einfluss auf die Analyse und Detektion hat, wird die Bildentzerrung in der vorliegenden Arbeit nicht berücksichtigt.

7 Analyse und Verarbeitung von Bilddaten

In diesem Kapitel wird ein Überblick über die implementierte Software zur Bildanalyse und Detektion gegeben. Darunter werden die wichtigen Abschnitte Maskierung, Filterung, Wolkendetektion und Differenzierung zwischen hellen und dunklen Wolken zur Schau gestellt. Außerdem wird die Berechnung von Wolken-Bedeckungsgrad wiedergegeben.

7.1 Maskierung

Bevor eine Analyse und Detektion der Bildpunkte beginnt, müssen als Erstes alle Bildpunkte, die zum Rahmen und zu umliegenden Objekten gehören auf den Wert RGB-Wert (0,0,0) gebracht, d.h. schwarz bemalt werden. Dieser Vorgang ist für später sehr wichtig, denn andernfalls können diese Pixel nicht aussortiert werden und sie würden zur Fehlkalkulation führen. In Abbildung 7.1 ist eine grafische Darstellung dieses Ablaufs zu erblicken. Jedoch wird diese Aufgabe softwaretechnisch ganz anders bewältigt.

Zuvor ist darauf hinzuweisen, dass durch das Anbringen von Maske 2 die Lage der Kamera nicht verändert werden darf. Im Falle einer Verschiebung muss die Maske 2 an die neue Sicht angepasst werden.

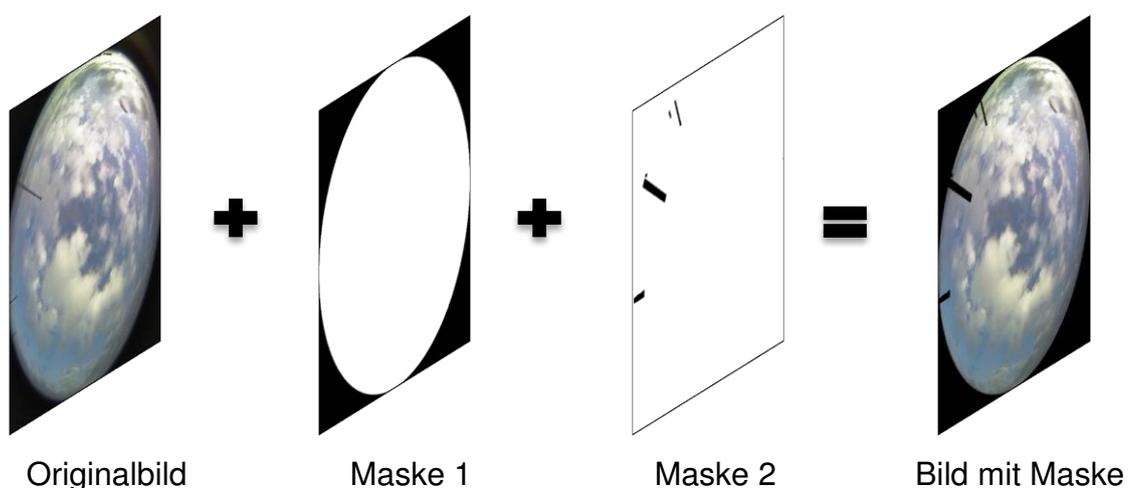


Abbildung 7.1 Maskierung von Fisheye-Aufnahmen

Programmtechnisch bringt die Methode *putMask()* die Maske in gewünschter Konfiguration in dem Originalbild. Im ersten Schritt wird die Maske 1 mittels der OpenCV-Methode *circle()* erzeugt. Hierbei wird nur ein Kreis mit demselben Durchmesser wie das Fisheye-Bild erstellt. Der Kreis ist mit den RGB-Wert (255,255,255), d.h. mit der Farbe Weiß, gefüllt. Die anderen Pixel beinhalten den RGB-Wert (0,0,0). Als Nächstes wird eine Matrix derselben Größe wie das des Originalbildes erstellt. Sie wird als Puffer²⁴ genutzt. Danach wird mit der OpenCV-Methode *copyto()* die Pixel vom Originalbild in den Puffer kopiert, in der die Maske 1 den RGB-Wert (255,255,255) aufweist. In diesem Kontext heißt es, dass nur der Inhalt des Kreisausschnitts im Originalbild in den Puffer kopiert wird.

Die Maske 2 wurde zuvor mit dem Bildbearbeitungssoftware GIMP 2 [34] erstellt und der Methode *putMask()* übergeben. Ferner wird die Methode *copyto()* ein letztes Mal genutzt, um nur die Pixel des Puffers an die Zielmatrix zu kopieren, in der die Maske 2 den RGB-Wert ungleich (0,0,0) entspricht.

Zum Schluss wird als Resultat die Zielmatrix zurückgegeben.

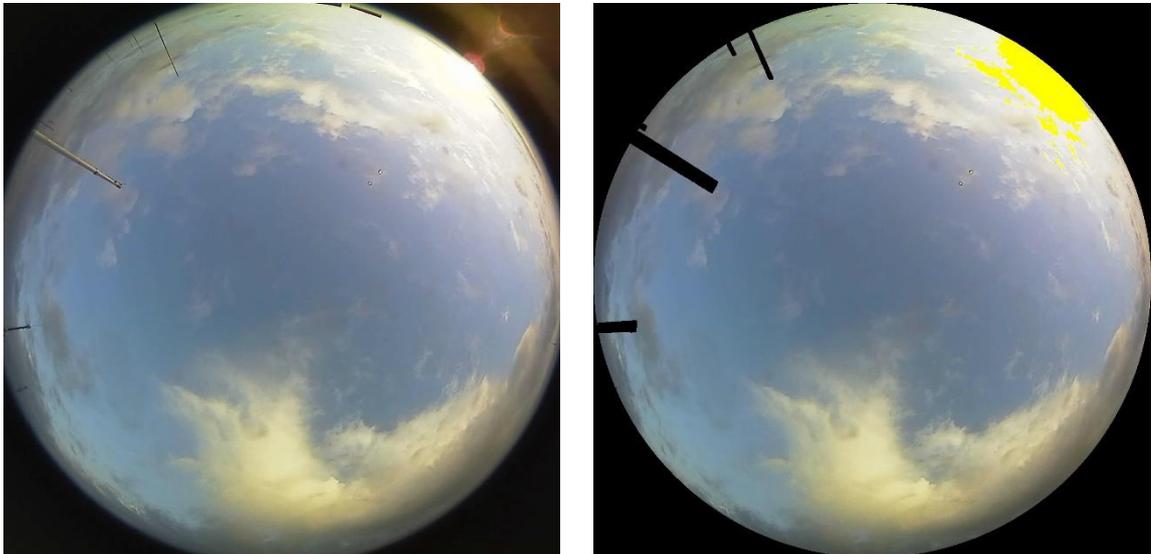
7.2 Filterung

Eine Filterung ist notwendig, um Sonne und das Streulicht herum, sowohl auch Dispersion durch die Acrylglaskuppel der Kamera, welches als Wolke erkannt werden können, von der Berechnung auszuschließen. Dieser Filter ist in der Methode *findClouds()* implementiert. Dabei werden die RGB-Werte (R,G und B jeweils getrennt) der einzelnen Pixel $P_{n,m}$ mittels Schwellenwertverfahren geprüft. Falls einer davon den Wert 145 überschreitet, dann wird dieser Bildpunkt gefiltert und gelb gefärbt, siehe Abbildung 7.2.

$$P_{n,m} = \{R_{n,m}, G_{n,m}, B_{n,m}\} \quad (7.1)$$

²⁴ (engl.: Buffer) sind in der Softwareentwicklung Zwischenspeicher

$$\forall P_{n,m} \in Filter \Leftrightarrow (R_{n,m} \cup G_{n,m} \cup B_{n,m}) > 145 \quad (7.2)$$



(a) Original

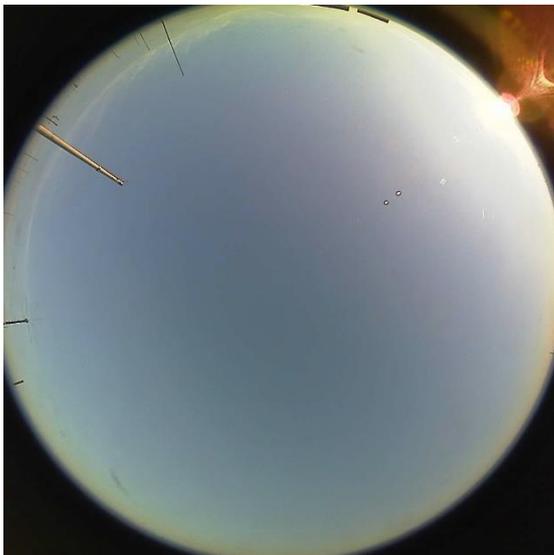
(b) Maskiert und gefiltert

Abbildung 7.2 Das Ergebnis der Filterung; Aufnahme vom 10.12.2014 15:11 Uhr
(teils bewölkter Himmel), Gelb: Filter, Schwarz: Maske

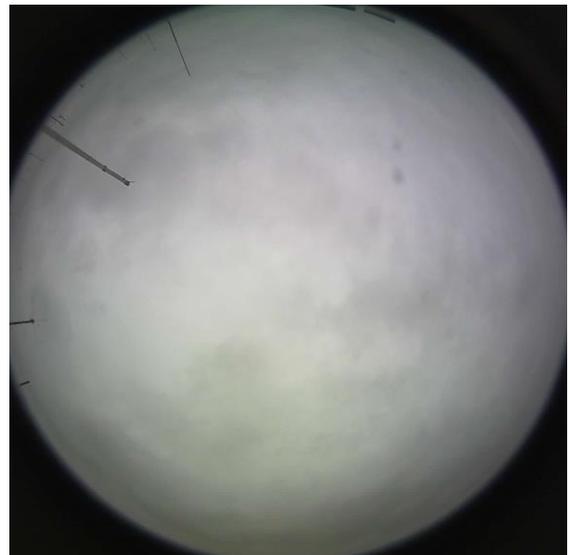
Der Wert 145 wurde durch Ausprobieren festgelegt. Im Falle eines zu niedrig angesetzten Schwellenwerts können Bildpunkte, die zum Himmel oder Wolke gehören, aussortiert werden. Im anderen Fall hingegen würden Pixel, die ausgefiltert werden müssen, als Himmel oder Wolke interpretiert, und diese würden im Nachhinein als Fehler in die Berechnungen eingehen. Der Schwellenwert 145 ist nicht ideal, aber im Vergleich zu den anderen Werten ist er weniger fehleranfällig. Um die Fehleranfälligkeit noch geringer zu halten, kann ein Algorithmus implementiert werden, der als Erstes zu dem aufgenommenen Bild einen Schwellenwert berechnet. Somit würde für jede Aufnahme ein anderer Vergleichswert vorhanden sein. Diese Methode wurde leider aufgrund geringer Zeit nicht implementiert.

7.3 Wolkenerkennung und Unterscheidung zwischen hellen und dunklen Wolken

Die Wolkenerkennung wird mit der Methode von Long und De Luisi durchgeführt. Diese wurde bereits im Kapitel 4.2 anschaulich gemacht. In der Methode *findClouds()* erfüllt die Gleichung (4.2) die Hauptaufgabe. Mittels dieser Normierung lässt sich auslegen, dass die Pixel im Falle eines bewölkten oder unbewölkten Himmels stetig verteilen, siehe Abbildung 7.3, und Abbildung 7.4. Zudem kann nun ein Schwellenwert T bestimmt werden, damit die Identifizierung der Wolkenpixel erfolgen kann. In Abbildung 7.4 wurde der Schwellenwert auf 0,9 festgelegt, weil dieser die besten Ergebnisse geliefert hat.



(a) Unbewölkter Himmel,
Aufnahme vom 17.11.2014
14:37 Uhr



(b) Kompletter bewölkter Himmel,
Aufnahme vom 13.11.2014
13:44 Uhr

Abbildung 7.3 Aufnahmen zur Bestimmung des Schwellenwerts

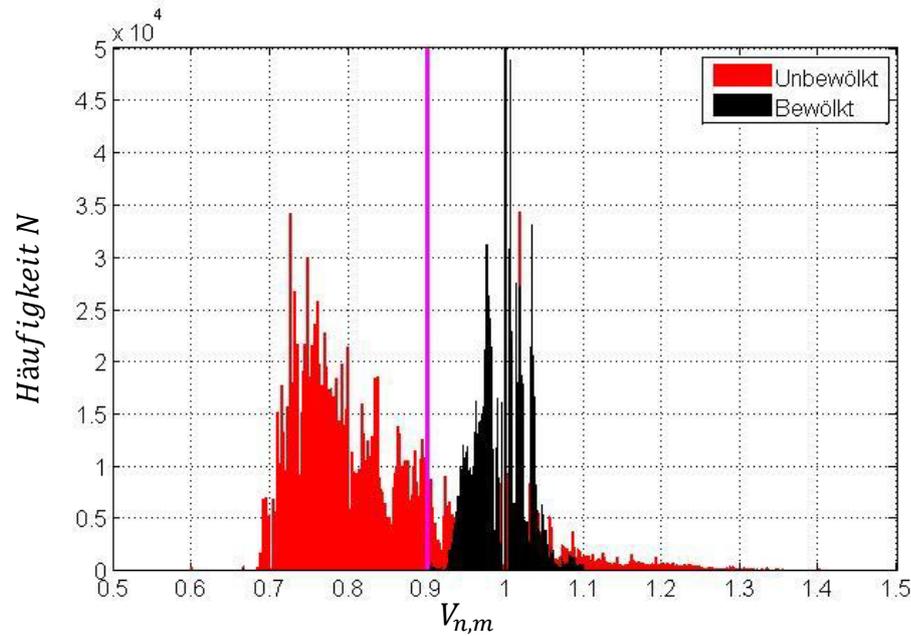


Abbildung 7.4 Häufigkeitsverteilung des $V_{n,m}$ bei komplett bewölkten (Rot) und unbewölkten Himmel (Schwarz). Schwellenwert $T = 0,9$ (Magenta)

Die Implementierung des Schwellenwertverfahrens wurde unter folgenden Bedingungen durchgeführt:

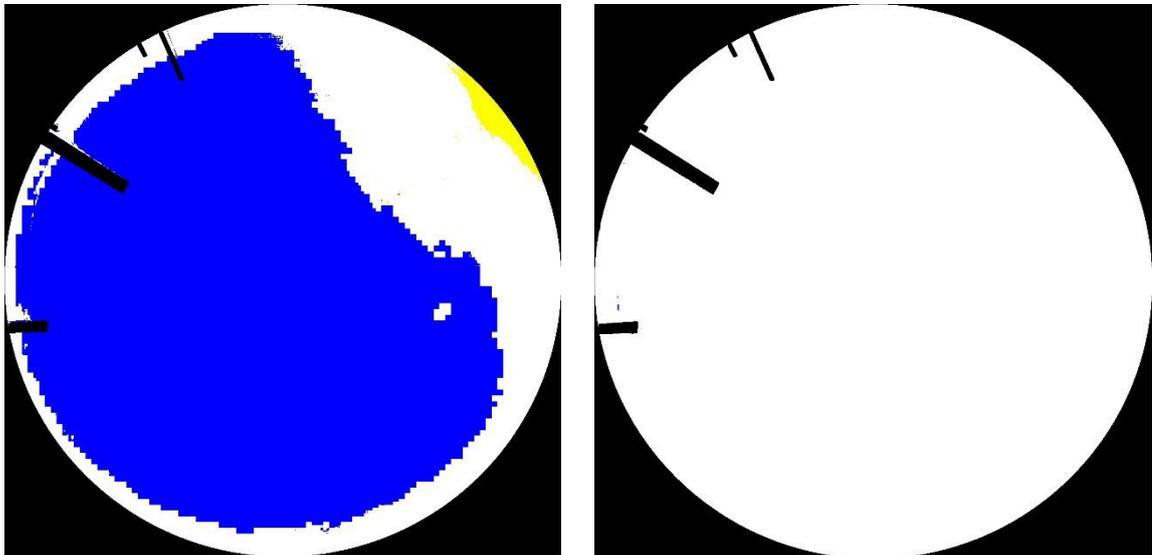
$$\forall P_{n,m} \in \text{Wolke} \Leftrightarrow V_{n,m} > T \quad (7.3)$$

$$\forall P_{n,m} \in \text{blauer Himmel} \Leftrightarrow V_{n,m} \leq T \quad (7.4)$$

$$\text{mit } V_{n,m} = \frac{R_{n,m}}{B_{n,m}}$$

In Abbildung 7.5 ist das Ergebnis der Wolkenerkennung für $T = 0,9$ zu sehen. Es ist zu erkennen, dass bei einer unbewölkten Aufnahme die Pixel, die sich am Kreisrand befinden, als Wolke interpretiert werden. Zusätzlich wird das Streulicht, welches Pixel, die nahezu Sonne weiß darstellen lässt, ebenfalls als Wolke aufgefasst. Diese

Tatsachen sind nicht zu vermeiden und gehen deshalb als Fehler in die Berechnung ein.



(a) Unbewölkter Himmel,
Aufnahme vom 17.11.2014
14:37 Uhr

(b) Komplette bewölkter Himmel,
Aufnahme vom 13.11.2014
13:44 Uhr

Abbildung 7.5 Das Ergebnis der Wolkenerkennung mit $T = 0,9$. Für den Test wurden die Aufnahmen der Abbildung 7.3 verwendet. Blau: Blauer Himmel, Weiß: Wolke, Gelb: Filter

Ein niedriger Schwellenwert führt dazu, dass zu viele Bildpunkte des blauen Himmels als Wolke interpretiert werden. Im anderen Fall hingegen führt ein zu hoher Schwellenwert dazu, dass ein großer Anteil von Wolkenpixel als blauer Himmel betrachtet wird. Da schließlich - wie bei der Filterung - kein idealer Schwellenwert für jede Aufnahme berechnet werden kann, wird in dieser Arbeit mit dem Wert $T = 0,9$ gearbeitet.

Wie ein schlecht gewählter Schwellenwert die Wolkendetektion beeinträchtigen kann, kann in der Abbildung 7.6 veranschaulicht werden.



(a) Original

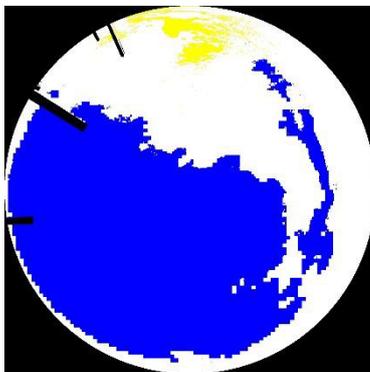
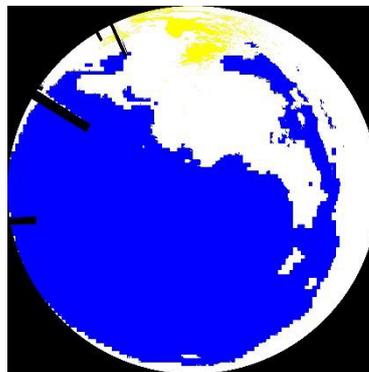
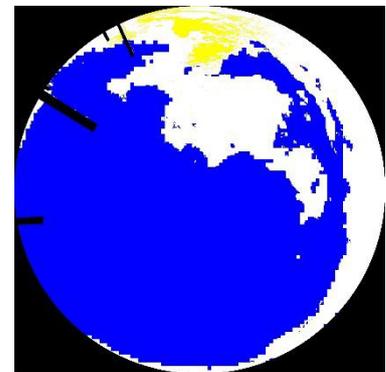
(b) $T = 0,85$ (c) $T = 0,9$ (d) $T = 0,95$

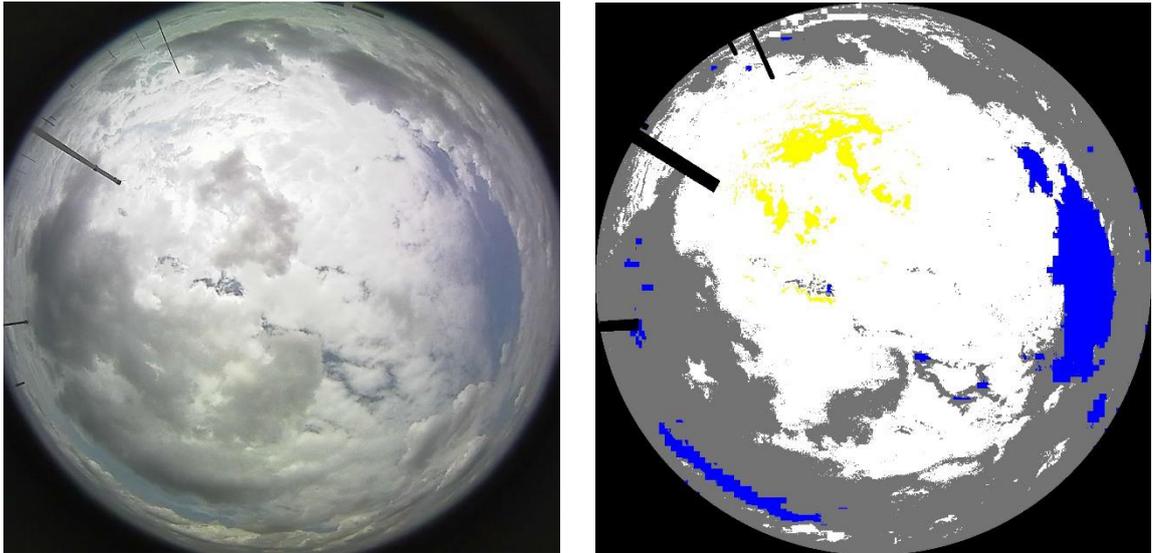
Abbildung 7.6 Vergleich der Schwellenwerte, Aufnahme vom 17.11.2014 15:03 Uhr (teils bewölkter Himmel). Blau: blauer Himmel, Weiß: Wolke, Gelb: Filter

Als Nächstes wird eine eigene Methode angewendet, um die hellen Wolken von den dunkleren differenzieren zu können. Mittels der Gleichungen (7.5) und (7.6) werden zuerst die Mengen A und B gebildet. Anschließend werden Bildpunkte, die zur Menge A und Menge B gehören als Pixel der dunklen Wolke identifiziert, siehe Gleichung (7.7).

$$A = (R_{n,m} \cap G_{n,m} \cap B_{n,m}) < 160 \quad (7.5)$$

$$B = |B_{n,m} - G_{n,m}| + |G_{n,m} - R_{n,m}| < 30 \quad (7.6)$$

$$\forall P_{n,m} \in \text{dunkle Wolke} \Leftrightarrow A \cap B \quad (7.7)$$



(a) Original

(b) Analysiert

Abbildung 7.7 Detektion der dunklen Wolken, Aufnahme vom 06.01.2015
13:27 Uhr (stark bewölckter Himmel). Blau: blauer Himmel, Weiß: helle Wolke,
Grau: dunkle Wolke, Gelb: Filter

Das Ergebnis der Gleichung (7.7) ist auf der Abbildung 7.7b abgebildet. Es ist gut zu erkennen, dass die meisten dunklen Wolken erkannt worden sind. Wolken, die sich am Bildkreisrand befinden, werden aufgrund der Lichtverhältnisse auf dem Originalbild immer dunkler dargestellt. Deshalb werden diese Wolken meistens als dunkle Wolken interpretiert. Dieses Problem ist nicht zu vermeiden und geht deshalb ebenfalls als Fehler in die Berechnung ein.

7.4 Berechnung von Wolken-Bedeckungsgrad

Nachdem die Wolkenpixel erfolgreich detektiert wurden, kann der Wolken-Bedeckungsgrad berechnet werden. Um das bewerkstelligen zu können, müssen als Erstes Anzahl der Pixel, die zu einer Wolke gehören (n_{cloudy}) bestimmt werden. Das wird softwaretechnisch in der Methode *findClouds()* realisiert. Die Variable n_{cloudy} wird jedes Mal inkrementiert, sobald ein Wolkenpixel erkannt wurde. Nebenbei läuft der Zähler n_{Gesamt} , um Anzahl der Pixel, die analysiert wurden zu bestimmen. Die Variable n_{Gesamt} setzt sich aus der Anzahl der bewölkten (n_{cloudy}) und unbewölkten (n_{no_cloudy}) Pixeln zusammen. Ferner wird mit der Methode *calcCloudliness()* der Wolken-Bedeckungsgrad berechnet. Die Berechnung erfolgt mit der Gleichung (4.3). Das Ergebnis dieser Methode ist ein Wert zwischen Null und Eins.

Das Ziel ist nun mit Hilfe dieser Methode einmal den Gesamt-Wolkenbedeckungsgrad und den Bedeckungsgrad für bestimmte Zonen im Bild zu berechnen. Die Berechnung und Darstellung des Wolken-Bedeckungsgrads für einzelne Zonen wird im Kapitel 7.4.1 näher behandelt. Im Kapitel 7.4.2 hingegen wird das Resultat des Gesamt-Wolken-Bedeckungsgrads zur Schau gestellt.

7.4.1 Wolken-Bedeckungsgrad für einzelne Zonen

Die Fisheye-Kamera fängt durch seinen Blickwinkel (laut Datenblatt [7]: $180^\circ \pm 5^\circ$, gemessen nach Scaramuzza [32]: $180^\circ - 15,968^\circ$) alles ein, was vor ihr passiert. Da es sich in diesem Fall um eine große Fläche handelt, in dem sich die aufgenommenen Wolken befinden, ist es von Vorteil das Bild in mehrere Zonen aufzuteilen, weil die Entfernung der Wolken mittels einer Kamera nicht zu kalkulieren ist. Im Nachhinein kann dann der Wolken-Bedeckungsgrad für einzelnen Zonen in Abhängigkeit der Bildwinkel bestimmt werden.

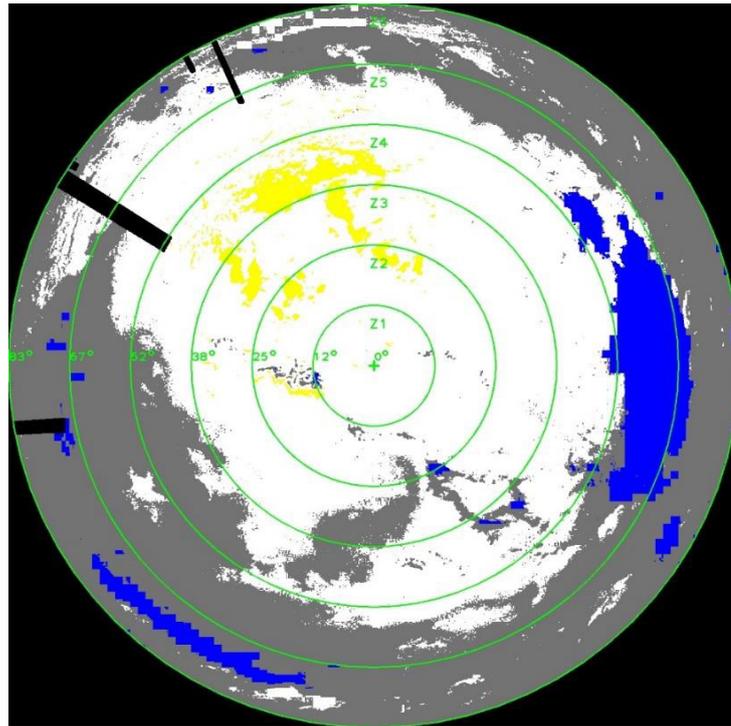


Abbildung 7.8 Die Aufteilung in sechs Zonen (Grün); Die analysierte Aufnahme ist vom 06.01.2015 13:27 Uhr (stark bewölkter Himmel). Blau: blauer Himmel, Weiß: helle Wolke, Grau: dunkle Wolke, Gelb: Filter

In Abbildung 7.8 wird eine Aufteilung in sechs Zonen demonstriert. Dabei ist die mittlere Kreisfläche die Zone 1 und die oberste Ringfläche die Zone 6. Die Grenze zwischen Zone 1 und Zone 2 hat einen Bildwinkel von 12° . Der Abschnitt zwischen Zone 5 und Zone 6 hingegen hat einen Winkel von 67° . Diese Aufteilung wurde mit der Methode `drawCircles()` verwirklicht. Der Bildwinkel wurde ausgehend von der Bildmitte mit der Methode `getTheta()` errechnet. Dafür wurden die Kameraparameter des Scaramuzza's Kameramodells verwendet.

Es ist nicht leicht den Wolken-Bedeckungsgrad für einzelnen Zonen direkt aus einem Bild zu berechnen, weil dafür exakt die Pixel mit in die Berechnung eingehen müssen, die sich wirklich innerhalb dieser Fläche befinden. Um dies zu ermöglichen, wird ein Trick angewendet, um dieses zu ermöglichen. Die Berechnung kann leicht durchgeführt werden, wenn die einzelnen Zonen sich in separaten Bildern befinden. So können nur die Pixel analysiert werden, die den RGB-Wert ungleich $(0,0,0)$

aufweisen. Ein Beispiel für diese Vorgehensweise wird in Abbildung 7.9 dargestellt. Für dieses Handeln ist die Methode `getZone()` verantwortlich.

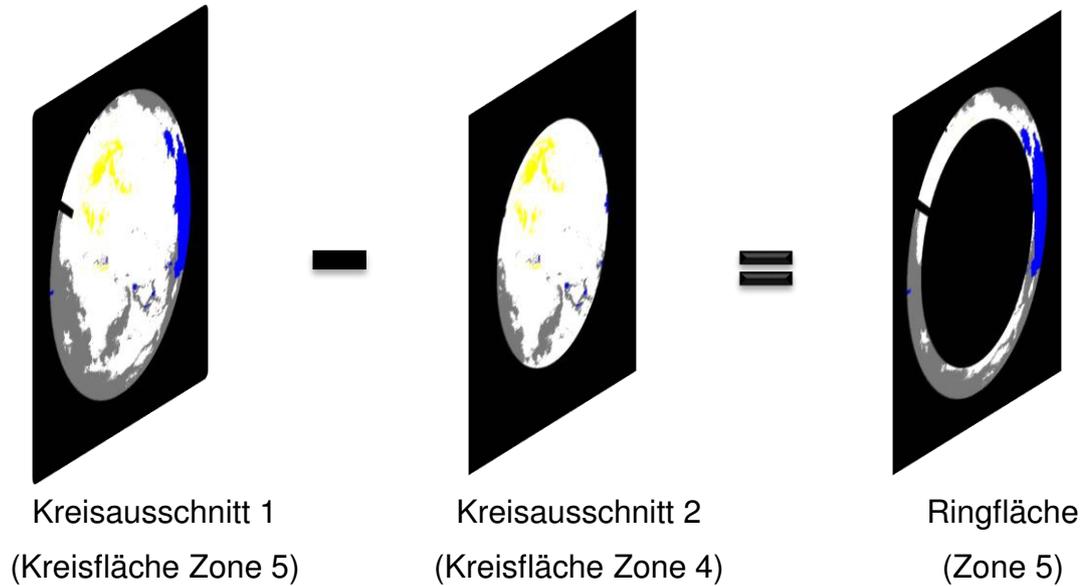


Abbildung 7.9 Bestimmung einer Zone. Hierbei wird die Ringfläche der Zone 5 zurechtgeschnitten

Ferner kann für jede Zone der Wolken-Bedeckungsgrad ermittelt und angesichts von der Methode `putResults()` auf dem Bild ausgegeben werden.

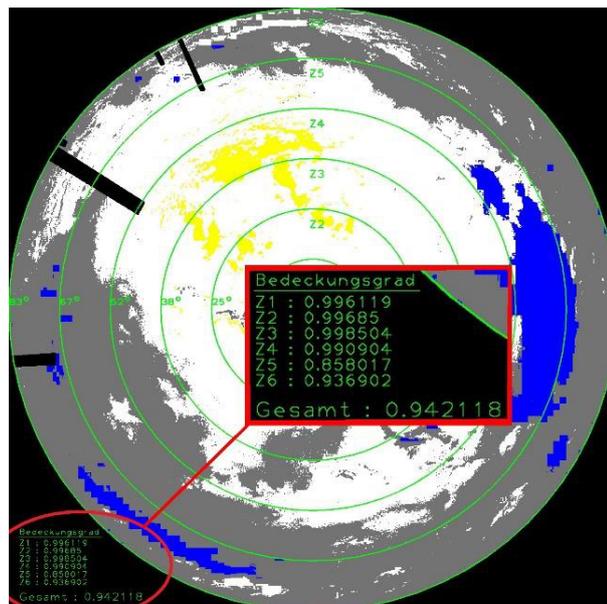


Abbildung 7.10 Ausgabe der Wolken-Bedeckungsgrad für einzelnen Zonen

7.4.2 Gesamt-Wolken-Bedeckungsgrad

Nachdem der Bedeckungsgrad für die einzelnen Zonen berechnet worden ist, kann daraus der Gesamt-Wolken-Bedeckungsgrad ermittelt werden. Das kann mithilfe einer Mittelwertberechnung erfolgen, siehe Gleichung (7.8).

$$N_{WBedGesamt} = \frac{N_{WBedZ1} + N_{WBedZ2} + \dots + N_{WBedZN}}{N} \quad (7.8)$$

Ein Ergebnis dieser Berechnung ist in Abbildung 7.10 zu sehen. In diesem Fall ist dieser Wert eine Zahl zwischen Null und Eins. Da im Normalfall der Wolken-Bedeckungsgrad immer in Achteln angegeben wird, wird eine zusätzliche Methode (*putDivide()*) implementiert, der diesen Wert in Achteln auf dem Bild darstellt. Außerdem wird zur Orientierung ein Kompass (*putCompass()*) und eine Legende (*putLegend()*) erstellt, siehe Abbildung 7.11.

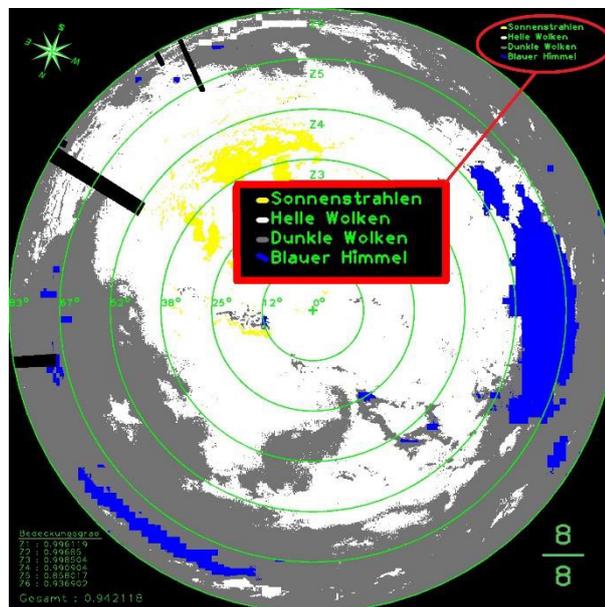


Abbildung 7.11 Endresultat einer Analyse und Detektion.
Aufnahme vom 06.01.2015 13:27 Uhr (stark bewölkter Himmel)

7.5 Programmablauf

Das Programm besteht aus zwei Klassen. Die erste Klasse ist in Abbildung 5.1 zu sehen. Sie ist für die Bildspeicherung zuständig. Die zweite Klasse hingegen ist für die Wolkenanalyse und Wolkenerkennung verantwortlich. Sie ist in Abbildung 7.12 abgebildet. Klassen, die für die Bildverzerrung berechtigt sind, wurden vernachlässigt, weil die Analyse und Detektion ohne Bildverzerrung stattfindet.

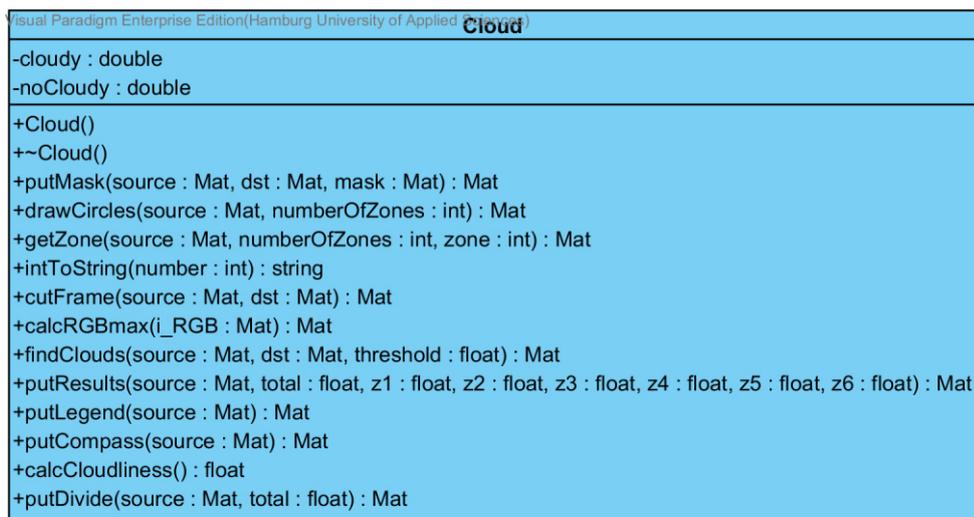


Abbildung 7.12 Die Klasse *Cloud*. Sie ist für die Wolkenanalyse und Wolkendetektion zuständig

Der Anhang 3 enthält ein Sequenzdiagramm zur Hauptklasse. Das ist eine Erweiterung des Diagramms, welches in Abbildung 5.3 dargestellt ist. Ergänzt wurden die Zugriffe auf die Klasse *Cloud*. Die Programmlaufzeit wurde bei drei Tagen belassen. Es ist zu erkennen, dass bei der Ausführung das Programm drei tagelang stündlich ein Bild im Zielpfad ablegt. Währenddessen wird ihr Inhalt analysiert und wird abschließend das analysierte Bild ebenfalls im gleichen Pfad abgespeichert.

Es wurde bereits erwähnt, dass in der Methode *findCloud()* die Funktion *getTheta()* aufgerufen wird, um den pixelabhängigen Bildwinkel zu bestimmen. Diese Funktion wurde im Source-File *ocam_functions.cpp* implementiert, denn dort befinden sich die Parameter des Kameramodells von Scaramuzza.

7.6 Ergebnisse

Wie bereits erwähnt, treten bei der Wolkendetektion wetterbedingte Fehler auf, die leider nicht zu vermeiden sind. Dadurch wird der Wolken-Bedeckungsgrad falsch eingeschätzt. Die größten Fehler treten im wolkenfreien Himmel auf. Zudem wird unterschieden, wo genau die Sonne sich auf dem Bild befindet. Falls die Sonne sich direkt über der Kamera aufhält, dann kann der Wolkenbedeckungsgrad um 50% fehlerkalkuliert werden, siehe Abbildung 7.13. Der Grund hierfür ist, dass der Filter das Streulicht um die Sonne herum nicht komplett aussondert. Diese werden von Software als Wolke interpretiert. Einige Bereiche werden sogar als dunkle Wolke selektiert.

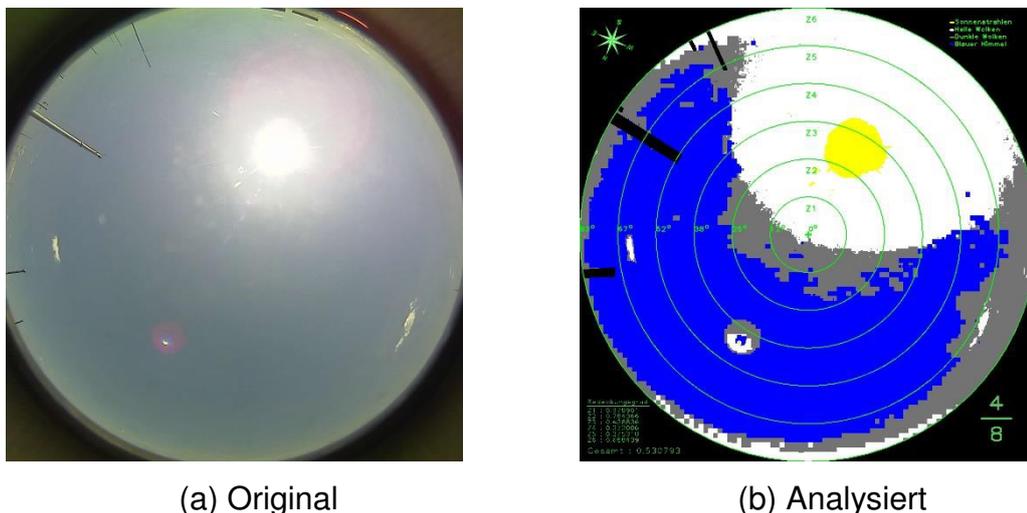
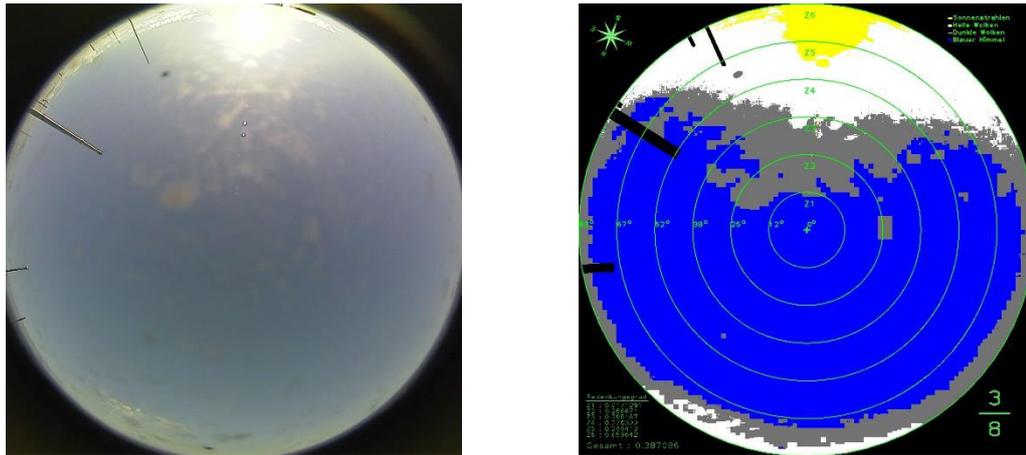


Abbildung 7.13 Aufnahme vom 30.05.2015 10:22 Uhr (unbewölckter Himmel);
 Ergebnis: $Z1 = 0.978901$, $Z2 = 0.784366$, $Z3 = 0.438836$, $Z4 = 0.372006$,
 $Z5 = 0.375318$, $Z6 = 0.688439$, Gesamt = 0.530793; Blau: blauer Himmel, Weiß:
 helle Wolke, Grau: dunkle Wolke, Gelb: Filter

In Abbildung 7.14 tritt ebenfalls dieser Fehler auf. Hier wird der Wolken-Bedeckungsgrad auf $\frac{3}{8}$ ermessen. Im Originalbild ist es jedoch zu beobachten, dass der Wolken-Bedeckungsgrad geringer als $\frac{1}{8}$ ist. Daraus folgt eine Fehleinschätzung von größer als 25%.



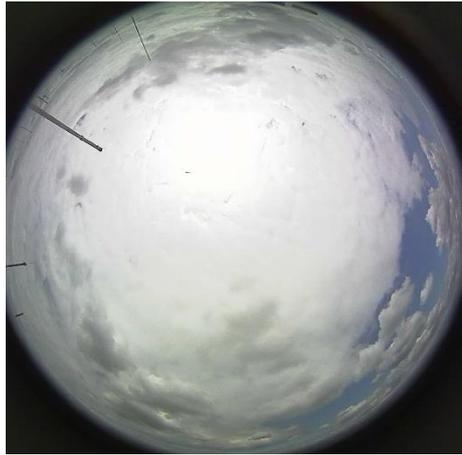
(a) Original

(b) Analysiert

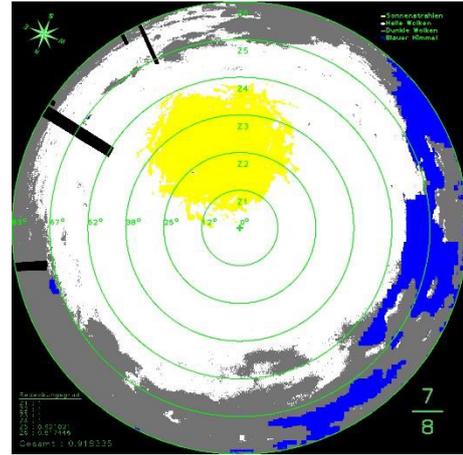
Abbildung 7.14 Aufnahme vom 11.12.2014 11:22 Uhr (sehr gering bewölkter Himmel); Ergebnis: $Z1 = 0.017129$, $Z2 = 0.286671$, $Z3 = 0.295157$, $Z4 = 0.270322$, $Z5 = 0.299412$, $Z6 = 0.659642$, Gesamt = 0.387086; Blau: blauer Himmel, Weiß: helle Wolke, Grau: dunkle Wolke, Gelb: Filter

Bei stark bewölktem und komplett bedecktem Himmel lassen sich die Wolken ganz gut detektieren, siehe Abbildung 7.15 und Abbildung 7.16. Des Weiteren ist es zu erkennen, dass die dunklen Wolken von den Hellen gut differenziert werden. Das Ergebnis des Wolken-Bedeckungsgrads für die Abbildung 7.15 ist akzeptabel. In Abbildung 7.16 wurde ebenfalls ein vertretbares Ergebnis kalkuliert. Aber hier wurden die Regentropfen, die sich auf dem Acrylglaskoppel der Kamera befinden als blauer Himmel interpretiert. In diesem Fall sind die Anzahl der Regentropfen gering. Bei starkem Regen können größere Bereiche der Aufnahme als blauer Himmel aufgefasst werden und diese würden als Fehler in die Berechnung eingehen.

Eine Analyse für den Schneefall konnte nicht realisiert werden, weil in diesem Zeitraum die Wetterverhältnisse nicht passend waren.

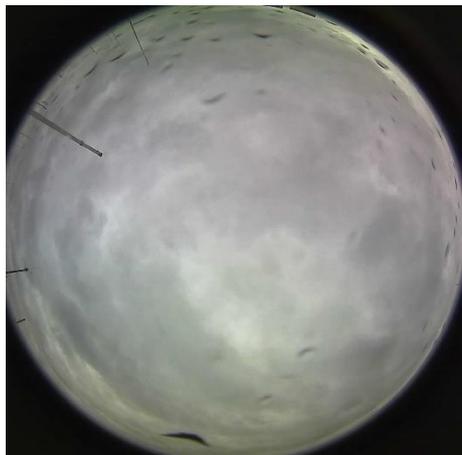


(c) Original

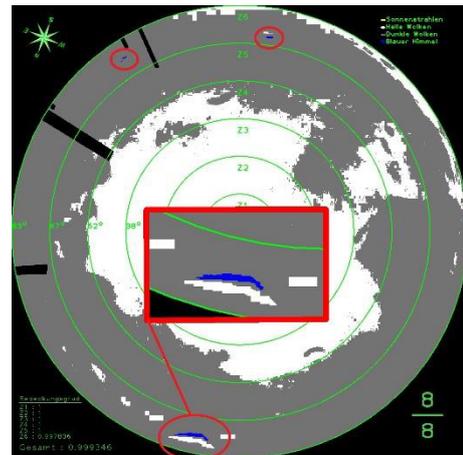


(d) Analysiert

Abbildung 7.15 Aufnahme vom 15.06.2015 12:56 Uhr (stark bewölkter Himmel);
 Ergebnis: $Z1 = 1$, $Z2 = 1$, $Z3 = 1$, $Z4 = 1$, $Z5 = 0.921021$, $Z6 = 0.817446$,
 Gesamt = 0.918335; Blau: blauer Himmel, Weiß: helle Wolke, Grau: dunkle Wolke,
 Gelb: Filter



(a) Original



(b) Analysiert

Abbildung 7.16 Aufnahme vom 11.12.2014 11:22 Uhr (komplett bedeckter Himmel
 inkl. Regen); Ergebnis: $Z1 = 1$, $Z2 = 1$, $Z3 = 1$, $Z4 = 1$, $Z5 = 1$,
 $Z6 = 0.997836$, Gesamt = 0.999346; Blau: blauer Himmel, Weiß: helle Wolke,
 Grau: dunkle Wolke, Gelb: Filter

Eine weitere Schwäche dieses Verfahrens lässt sich beim Sonnenuntergang feststellen. Die Abbildung 7.17 zeigt wie fehlerhaft die Berechnung ist, wenn beim Sonnenuntergang eine Analyse der leicht bewölkten Himmel stattfindet. Aufgrund der schwachen Helligkeit wird sowohl das Streulicht als auch der unbewölkter Himmel als Wolke gesichtet. Daraus resultiert ein Wolken-Bedeckungsgrad von $\frac{8}{8}$. Im Originalbild lässt sich jedoch eine Bedeckung von $\frac{1}{8}$ beobachten. Ein komplett bedeckter Himmel würde in diesem Fall zu keine Fehler führen.

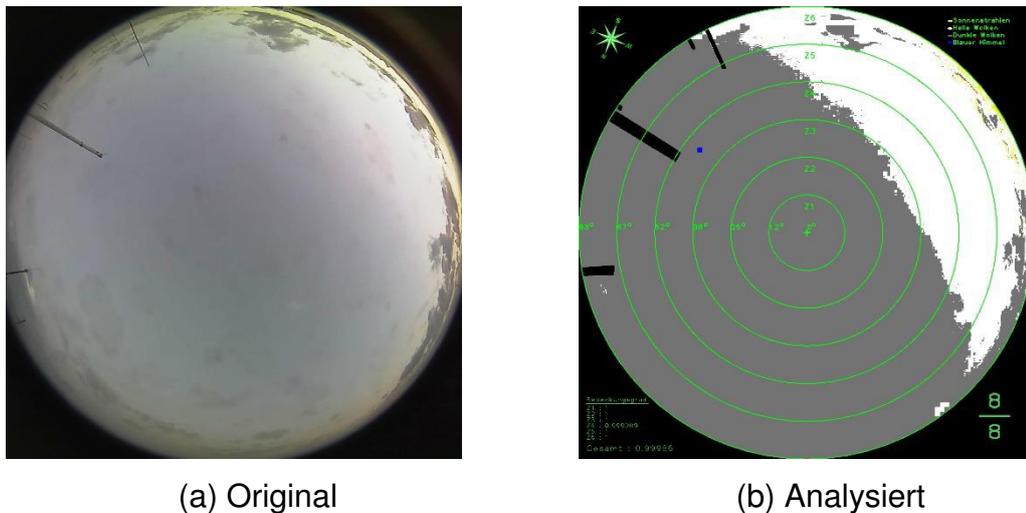


Abbildung 7.17 Aufnahme vom 10.12.2014 16:18 Uhr (leicht bewölkter Himmel bei Sonnenuntergang); Ergebnis: $Z1 = 1$, $Z2 = 1$, $Z3 = 1$, $Z4 = 0.999289$, $Z5 = 1$, $Z6 = 1$, Gesamt = 0.999986; Blau: blauer Himmel, Weiß: helle Wolke, Grau: dunkle Wolke, Gelb: Filter

Die Abbildung 7.18 ist ein Beweis dafür, dass die Software bei guten Verhältnissen erfreuliche Ergebnisse liefern kann. Über die Selektion der dunklen Wolken kann noch diskutiert werden, aber das Ergebnis $\frac{3}{8}$ lässt sich bei einer Beobachtung bestätigen.

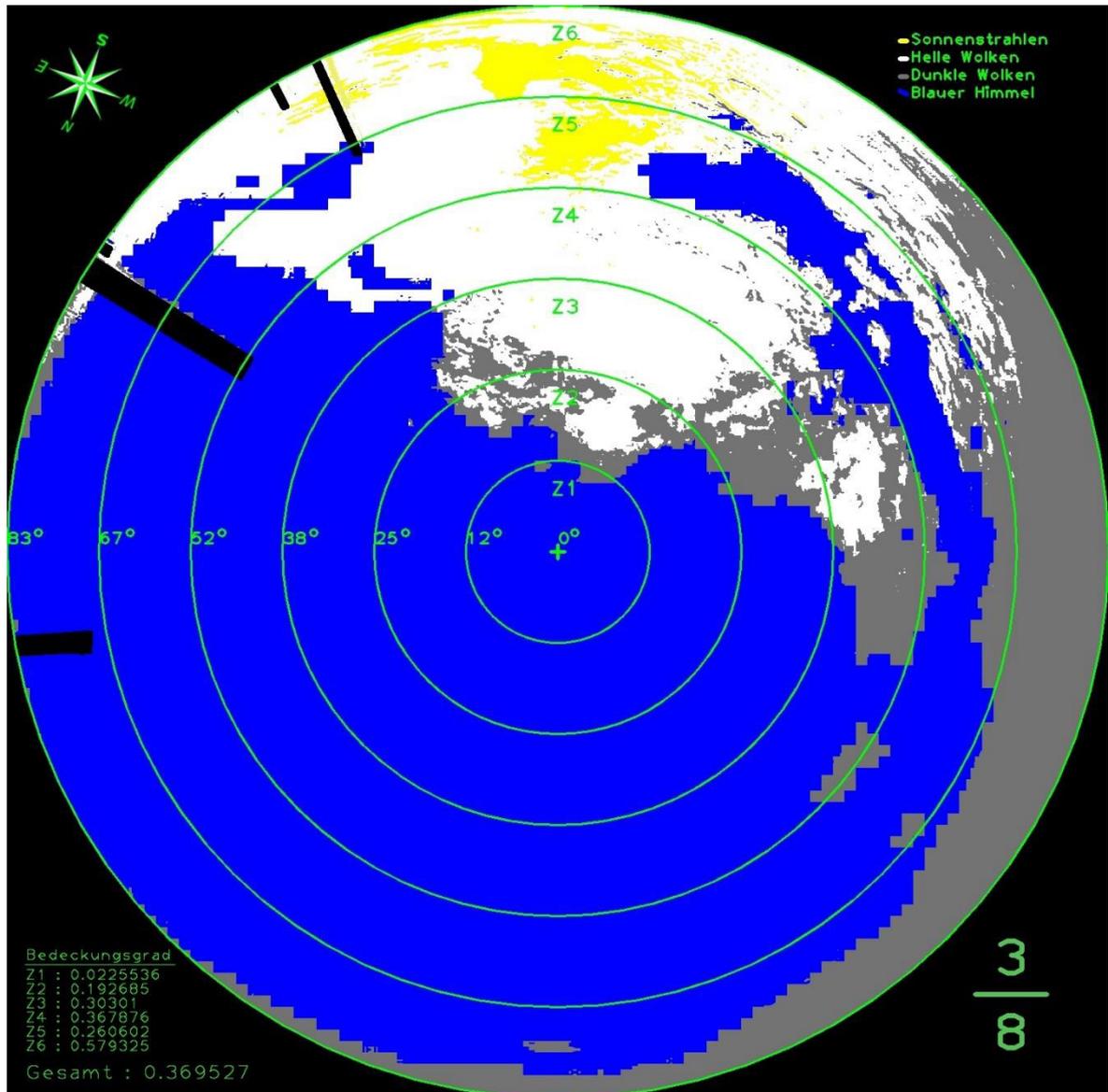


Abbildung 7.18 Ein äußerst gutes Ergebnis für eine Analyse und Detektion der Wolkenpixel. Das Originalbild ist aus der Abbildung 7.6a zu entnehmen. Blau: blauer Himmel, Weiß: helle Wolke, Grau: dunkle Wolke, Gelb: Filter

8 Schlussfolgerungen

Im Folgenden soll der Aufbau dieser Bachelorthesis zusammengefasst werden. Außerdem soll beschrieben werden, wie das Resultat dieser Arbeit mit Blick in die Zukunft noch optimiert werden kann.

8.1 Zusammenfassung

In der vorliegenden Arbeit wurde eine Software entwickelt, die die Funktionalität besitzt, mithilfe der bereitgestellten Fisheye-Kamera eine automatische Wolkenanalyse und Wolkendetektion zu bewerkstelligen. Für die Realisierung erfolgte zunächst eine Einarbeitung im Bereich der digitalen Bildverarbeitung. Dabei wurde ganz gezielt die Nutzung von OpenCV erlernt, um die Anforderungen verwirklichen zu können.

Als Nächstes wurden Methoden zur Analyse und Detektion der Wolken aufgelistet. Es konnte festgestellt werden, dass mittels der Methode von Long und De Luisi [22, pp. 171-174] Wolkenpixel ganz leicht detektiert werden können, siehe Gleichungen (7.3), (7.4) und Abbildung 7.6. Jedoch ist diese Methode sehr fehleranfällig (siehe Abbildung 7.13 und Abbildung 7.17) und muss optimiert werden, damit sie auch in verschiedenen Lichtverhältnissen beim gering bewölktem Himmel gute Resultate liefern kann.

Für eine Wolkenklassifizierung ist allerdings die Methode von Heinle et al. [25, pp. 557-567] sehr günstig. Damit kann die Klassifizierung nur mit einer Kamera durchgeführt werden. Sie nutzt das k-nächster-Nachbar-Verfahren aus, um Testwerte an einer der sieben Wolkenklassen (Tabelle 4.1) zuzuordnen. Ein Beispiel hierfür wird im Kapitel 4.2 genannt (Abbildung 4.2).

Für die Bildspeicherung wurde die Klasse *Downloader* entwickelt. Ihre Aufgabe ist es stündlich, sowohl originale Kamerabilder als auch die analysierten Bilder chronologisch in einem Dateisystem abzuspeichern. Der Hauptordner hat den

Namen „Fisheye_Camera“. Die Unterordner hingegen werden mit aktuellem Datum betitelt. Die Programmlaufzeit kann durch den Benutzer auf beliebige Tage festgelegt werden. Bei der Ausführung des Programms wurde erkannt, dass es nicht zeitgemäß arbeitet. Grund hierfür ist, dass sowohl die Methoden für den Input und Output als auch die Funktionen für die Bildverarbeitung die Prozedur ins Stocken bringt, siehe Abbildung 5.4.

Folglich wurden zwei Kameramodelle präsentiert. Mit Hilfe der beiden Kameramodelle wurde versucht, Aufnahmen der Fisheye-Kamera verlustfrei zu entzerren. Es hat sich herausgestellt, dass mittels OpenCV Kameramodell die Entzerrung der größeren Bildwinkel nicht möglich ist (Abbildung 6.4). Das Kameramodell von Scaramuzza hingegen konnte die Entzerrung meistern (Abbildung 6.9). Leider war das Ergebnis der Entzerrung für die Bildanalyse unbrauchbar, weil dabei Ereignisse wie Informationsverlust (z.B bei $SF = 7$) und Unschärfe (aufgrund von bilineare Interpolation) aufgetreten sind. Aufgrund dessen wurde beschlossen dem Programm diese Funktionalität zu entziehen und die Detektion ohne Entzerrung durchzuführen.

Aus den bisher gesammelten Erkenntnissen wurde anschließend die Klasse *Clouds* entwickelt. Diese Klasse ist für die Wolkenanalyse und Wolkendetektion verantwortlich. Als Erstes wurde eine Maske erstellt, um Bildpunkte die zum Bildrahmen und zu den umliegenden Objekten gehören von der Analyse und Detektion leichter auszuschließen, siehe Abbildung 7.1. Danach wurde versucht alle Pixel, die zur Sonne und das Streulicht herum gehören zu filtern (Abbildung 7.2). Die wichtigste Methode der Klasse *Clouds* ist die *findClouds()*. Sie arbeitet nach dem Prinzip von Long und De Luisi, um Wolkenpixel von blauem Himmel zu separieren. Der hierfür benötigte Schwellenwert wurde auf den Wert 0.9 festgelegt, weil damit die besten Ergebnisse erzielt worden sind (Abbildung 7.6). Nachdem die Differenzierung gelungen war, wurde der Wolken-Bedeckungsgrad für sechs verschiedenen Zonen ermittelt. Anschließend wurde der Gesamt-Wolkenbedeckungsgrad kalkuliert. Ferner wurden die Ergebnisse auf dem

analysierten Bild übertragen. Demnach wurde die Methode *findClouds()* erweitert, um die dunklen Wolken von den hellen zu unterscheiden.

Es wurde festgestellt, dass bei den Berechnungen wetterbedingte Fehler auftreten. Beim wolkenfreien Himmel liegt die Fehlerrate bei 50% (siehe Abbildung 7.13), weil das Streulicht um die Sonne herum als Wolke aufgefasst wurde. Bei Sonnenuntergang im wolkenfreien Himmel ist die Fehlerrate am größten. Hierbei wird ein Wolken-Bedeckungsgrad von $\frac{8}{8}$ eingeschätzt (Abbildung 7.17). Es lässt sich jedoch eine Bedeckung von $\frac{1}{8}$ beobachten.

Weitere Fehlkalkulationen entstehen, wenn sich auf der Acrylglaskuppel Regentropfen befinden. Diese werden vom Programm als blauer Himmel interpretiert (Abbildung 7.16).

8.2 Ausblick

Zur Erweiterung und Optimierung des Programms sind im Laufe der Arbeit vielfältige Grundgedanken entstanden. Die wichtigsten werden hier für die nachfolgenden Arbeiten kurz erläutert.

Eine Optimierung kann in Richtung des maschinellen Lernens gehen. Im Kapitel 4.2 wurde bereits erklärt, wie eine Klassifizierung der Wolken nach Heinle et al. [25, pp. 557-567] funktioniert. Dieser kann an der bereits existierenden Software integriert werden.

Eine weitere wichtige Erweiterung des Programms wäre die Bestimmung der Bewegungsrichtung, Entfernung und Höhe der Wolken. Im Kapitel 4.2 wurde erwähnt, dass dafür mindestens eine weitere Kamera benötigt wird, die von einem anderen Standort die Wolken aufnimmt. Aus beiden Bildern können danach die 3D-Koordinaten beliebiger Wolkenpunkte bestimmt werden. Durch Verfolgung der Bildpunkte, aus einer Vielzahl an Bildern kann die Bewegungsrichtung ermittelt

werden. Sobald erkannt wurde, dass die Wolken sich in Richtung der Kamera bewegen, kann durch die Beobachtung eines beliebigen Bildpunkts der Wolke ihre zeitliche Ankunft im Zentrum der Kamera errechnet werden.

Für die Umrandung der Wolken können Kantendetektions-Filter wie Canny- und Sobel-Algorithmus verwendet werden. Diese Funktionen sind bereits in OpenCV zu finden.

Um die Fehlerrate beim unbewölkten Himmel möglichst gering zu halten, kann als Erstes die Position der Sonne ermittelt werden. Demnach können alle Bildpunkte, die sich im bestimmten Umkreis der Sonne aufhalten, von der Detektion ausgeschlossen werden. So würde das Streulicht um die Sonne herum, nicht mehr als Wolke detektiert.

Die Analyse und Detektion der Wolken durch die Software, können den Solaranlagen in Zukunft wichtige Informationen über die aktuelle Wetterlage liefern. Nach Sonnenuntergang sind für sie diese Informationen irrelevant, deshalb kann eine weitere Funktion eingebaut werden, die das Programm kurz nach dem Sonnenuntergang schlafen legt. Erst beim Sonnenaufgang soll die Analyse und Detektion wieder fortgeführt werden. Mit der Zeitgleichung [35] kann für den aktuellen Tag, die Uhrzeit für den Sonnenaufgang und Sonnenuntergang berechnet werden.

Literaturverzeichnis

- [1] **H. Schneider**, „*Fraunhofer IWU*,“ 24 02 2015. [Online]. Available: http://www.iwu.fraunhofer.de/de/presse_und_medien/Intec_2015.html. [Zugriff am 23 06 2015].
- [2] **OLYPEDIA**, „*OLYPEDIA*,“ 2015. [Online]. Available: http://olypedia.de/Bild:Verzeichnung_Blende_Wikimedia.png. [Zugriff am 03 04 2015].
- [3] **sprut**, „*sprut*,“ 10 12 2002. [Online]. Available: <http://www.sprut.de/misc/digifoto/digifoto.htm>. [Zugriff am 03 04 2015].
- [4] **Wikipedia**, „*RGB-Farbraum*,“ 14 11 2014. [Online]. Available: https://de.wikipedia.org/wiki/RGB-Farbraum#/media/File:RGB_farbwuerfel.jpg. [Zugriff am 27 03 2015].
- [5] **W. Oertel**, „*Fotos Docoer*,“ 2006. [Online]. Available: <http://www.fotos.docoer-dig.de/Bildsensoren.htm>. [Zugriff am 04 04 2015].
- [6] **D. Göhring**, „*Digitalkameratechnologien - Eine vergleichende Betrachtung*,“ Humboldt Universität zu Berlin, Berlin, 2002.
- [7] **P. T. Corporation**, „*3 Mega-Pixel Vandal Proof Fish-Eye IP Camera ICA-8350 Datasheet*,“ PLANET Technology Corporation, Taiwan, 2015.
- [8] **M. Smith**, „*Introduction to OpenCV*,“ 2015.
- [9] **OpenCV**, „*OpenCV*,“ 2015. [Online]. Available: <http://opencv.org/>. [Zugriff am 01 03 2015].
- [10] **G. Bradski und A. Kaehler**, „*Learning OpenCV - Computer Vision with OpenCV Library*,“ Köln, O'REILLY, 2008.
- [11] **M. Politze**, „*OpenCV in a Nutshell*,“ Fachhochschule Aachen, Aachen, 2010.
- [12] **C. von Savigny**, „*Einführung in Atmosphäre und Klima*,“ Universität Greifswald, Greifswald, 2012.
- [13] **Wikiversity**, „*Streumechanismen*,“ 2009. [Online]. Available: https://de.wikiversity.org/wiki/Projekt:FE_Beobachtung_1/Lidar/Streumechanismen. [Zugriff am 04 04 2015].

- [14] **C. Baddiley**, „*Britastro*,“ 2015. [Online]. Available: <http://www.britastro.org/dark-skies/images/scattering.jpg>. [Zugriff am 04 04 2015].
- [15] **T. Kroker**, „*Wolken - Ihre Entstehung und ihr Einfluss auf den Strahlungshaushalt in der Atmosphäre*,“ Technische Universität zu Braunschweig, Braunschweig, 2014.
- [16] **Alfred-Wegener-Institut**, „*Meereisportal*,“ 2014. [Online]. Available: <http://www.meereisportal.de/meereiswissen/die-globale-bedeutung-von-meereis/wechselwirkungen-von-meereis-mit-anderen-komponenten-des-klimasystems/meereis-und-strahlungsbilanz.html>. [Zugriff am 05 04 2015].
- [17] **World Meteorological Organization**, „*Internationaler Wolkenatlas*,“ Deutscher Wetterdienst, Offenbach am Main, 1990.
- [18] **Meteorologisches Observatorium Lindenberg - Richard-Aßmann-Observatorium**, „*Bundesministerium für Verkehr und digitale Infrastruktur*,“ 2014. [Online]. Available: http://www.dwd.de/bvbw/appmanager/bvbw/dwdwwwDesktop?_nfpb=true&_pageLabel=_dwdwww_aufgabenspektrum_ueberwachung&_state=maximized&_windowLabel=T14607449251144912106311&T14607449251144912106311gsbDocumentPath=Content%252FForschung%252FFELG%252FMO L4%252Fmol. [Zugriff am 10 04 2015].
- [19] **B. Mühr**, „*Der Karlsruher Wolkenatlar - Wolkenklassifikation*,“ 01 07 2012. [Online]. Available: <http://www.wolkenatlas.de/wolken/class.htm>. [Zugriff am 05 04 2015].
- [20] **R. Vogel**, „*Entwicklung eines automatisierten Wolkendetektions- und Wolkenklassifizierungsverfahrens mit Hilfe von Support Vector Machines angewendet auf METEOSAT-SEVIRI-Daten für den Raum Deutschland*,“ Rheinische Friedrich-Wilhelms-Universität Bonn, Bonn, 2011.
- [21] **BR Wissen**, „*BR Wissen*,“ 2014. [Online]. Available: http://www.br.de/themen/wissen/meteorologie-wetter-wolkenarten-100~_v-img__16__9__xl_-

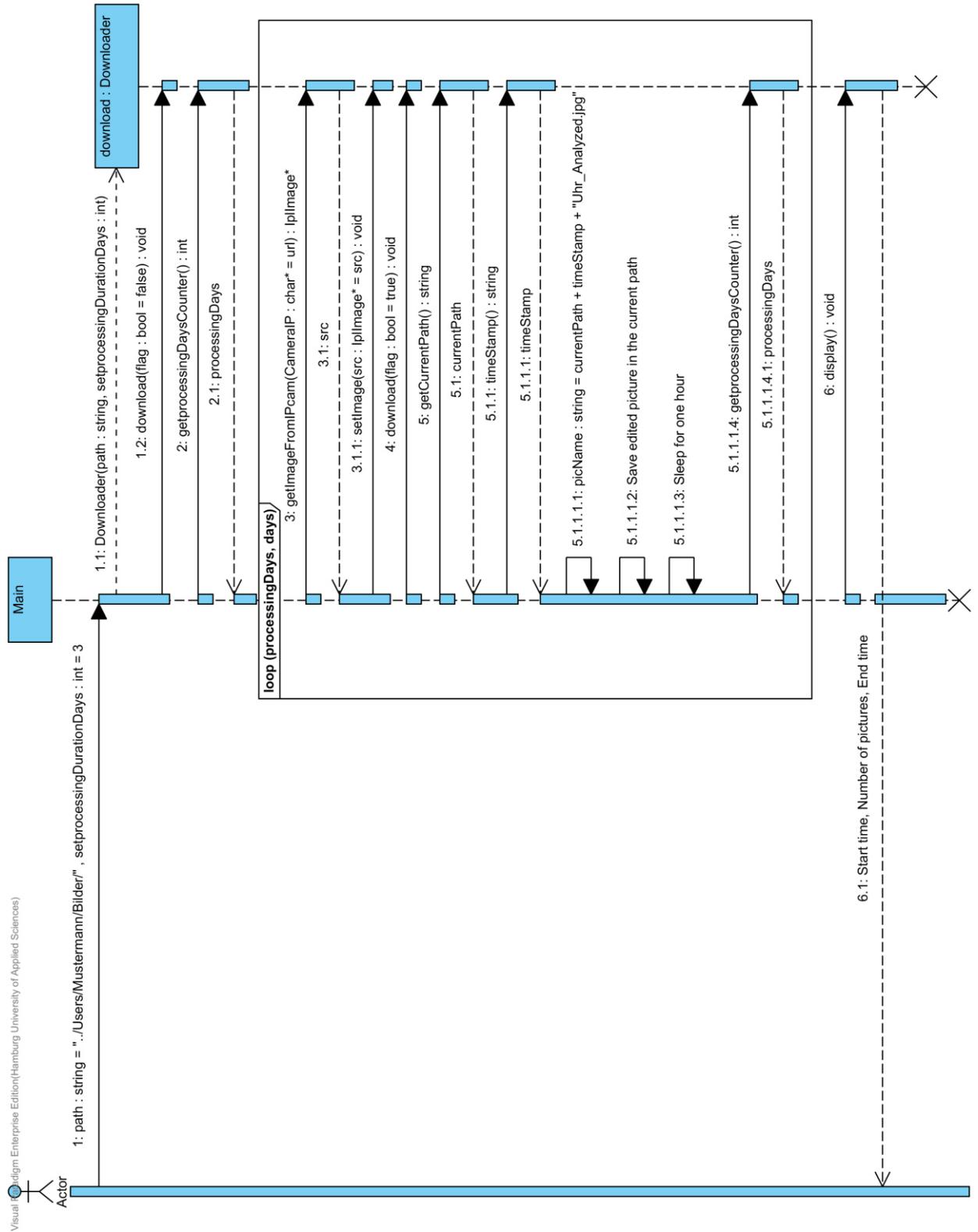
- d31c35f8186eb80b0cd843a7c267a0e0c81647.jpg%3Fversion%3D81515.
[Zugriff am 10 04 2015].
- [22] **C. Long und J. DeLuisi**, „*Development of an Automated Hemispheric Sky Imager for Cloud Fraction Retrievals*,“ Amer. Meteor. Soc., Phoenix Arizona, 2008.
- [23] **C. Long, J. Sabburg, J. Calbo und D. Pages**, „*Retrieving Cloud Characteristics from Ground-Based Daytime Color All-Sky Images*,“ Journal of Atmospheric and Oceanic Technology, Girona, 2006.
- [24] **G. Seiz und E. Baltasvias**, „*Cloud mapping using ground-based imagers*,“ Meteorological Observations and Instrumentation, Zürich, 2001.
- [25] **A. Heinle, A. Macke und A. Srivastav**, „*Automatic cloud classification of whole sky images*,“ Atmospheric Measurement Techniques, Kiel, 2009.
- [26] **R. O. Duda, P. E. Hart und D. G. Stork**, „*Pattern Classification*“, John Wiley & Sons, 2001.
- [27] **J. Kalisch**, „*Der Einfluss von Wolken auf den Strahlungsantrieb der Erde*,“ Kiel, 2011.
- [28] **B. Lang**, „*Vorgehen zur Kalibrierung von Kamerabildern*,“ HS-Osnabrück, Osnabrück, 2013.
- [29] **Z. Zhang**, „*A flexible new technique for camera calibration*,“ IEEE Transactions on Pattern Analysis and Machine Intelligence. 22(11), Redmond, WA, 2000.
- [30] **D. Scaramuzza, A. Martinelli und R. Siegwart**, „*A Flexible Technique for Accurate Omnidirectional Camera Calibration and Structure from Motion*,“ ICVS 2006, Lausanne, 2006.
- [31] **D. Scaramuzza**, „*Omnidirectional Vision: From calibration to robot motion estimation*,“ Citeseer, Zürich, 2008.
- [32] **D. Scaramuzza**, „*OCamCalib: Omnidirectional Camera Calibration Toolbox for Matlab*,“ 2013. [Online]. Available: <https://sites.google.com/site/scarobotix/ocamcalib-toolbox>. [Zugriff am 20 03 2015].

-
- [33] **P. Krüsi**, „*Visual Detection of an Object of Known Geometry in a Fisheye Camera Image*,“ ETH Zürich, Zürich, 2009.
- [34] **gimp**, „*GNU Image Manipulation Program*,“ 2001. [Online]. Available: <http://www.gimp.org/>. [Zugriff am 25 03 2015].
- [35] **A. Barmettler**, „*Zeitgleichung*,“ 2014. [Online]. Available: <http://lexikon.astronomie.info/zeitgleichung/>. [Zugriff am 15 06 2015].

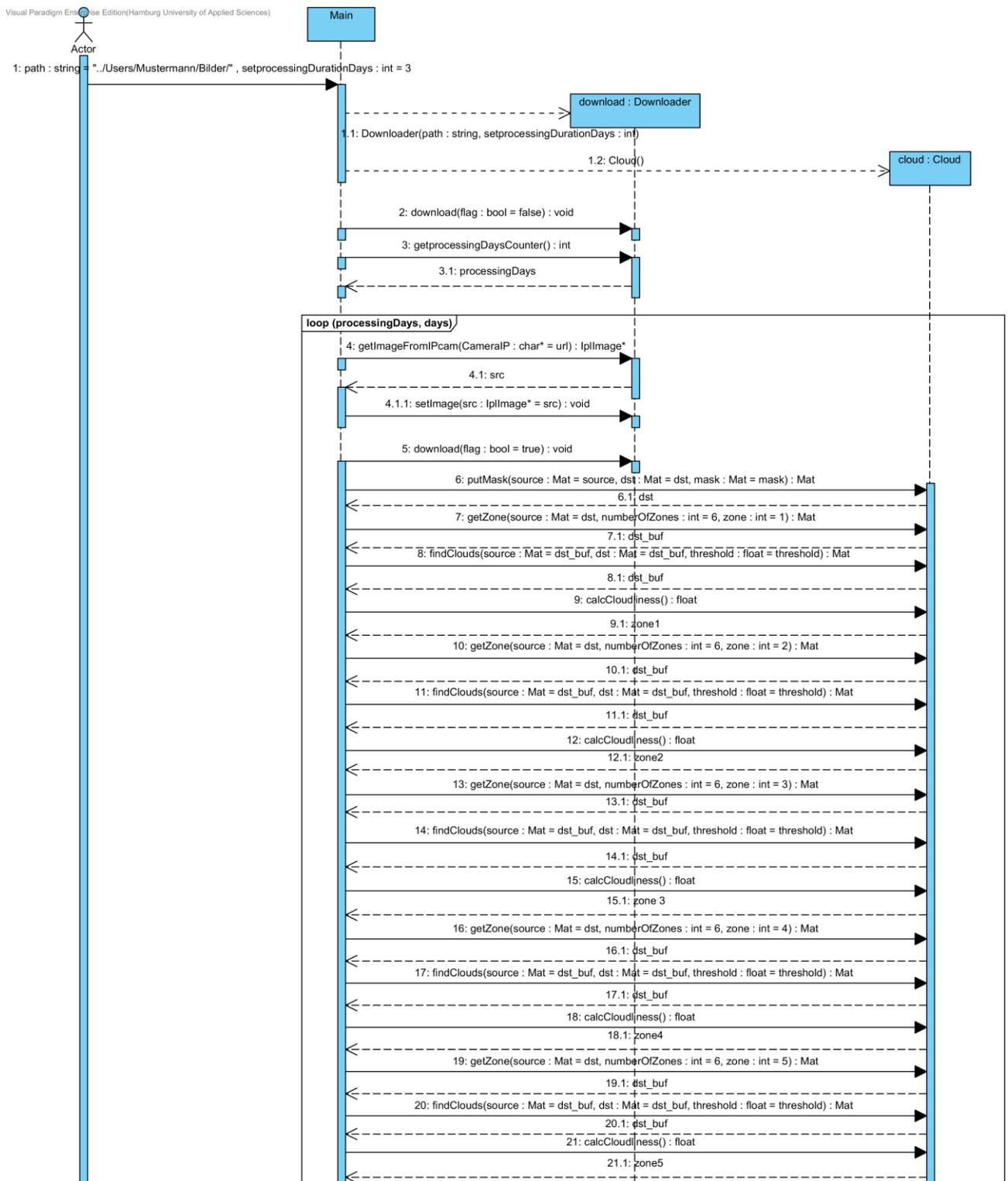
Anhang

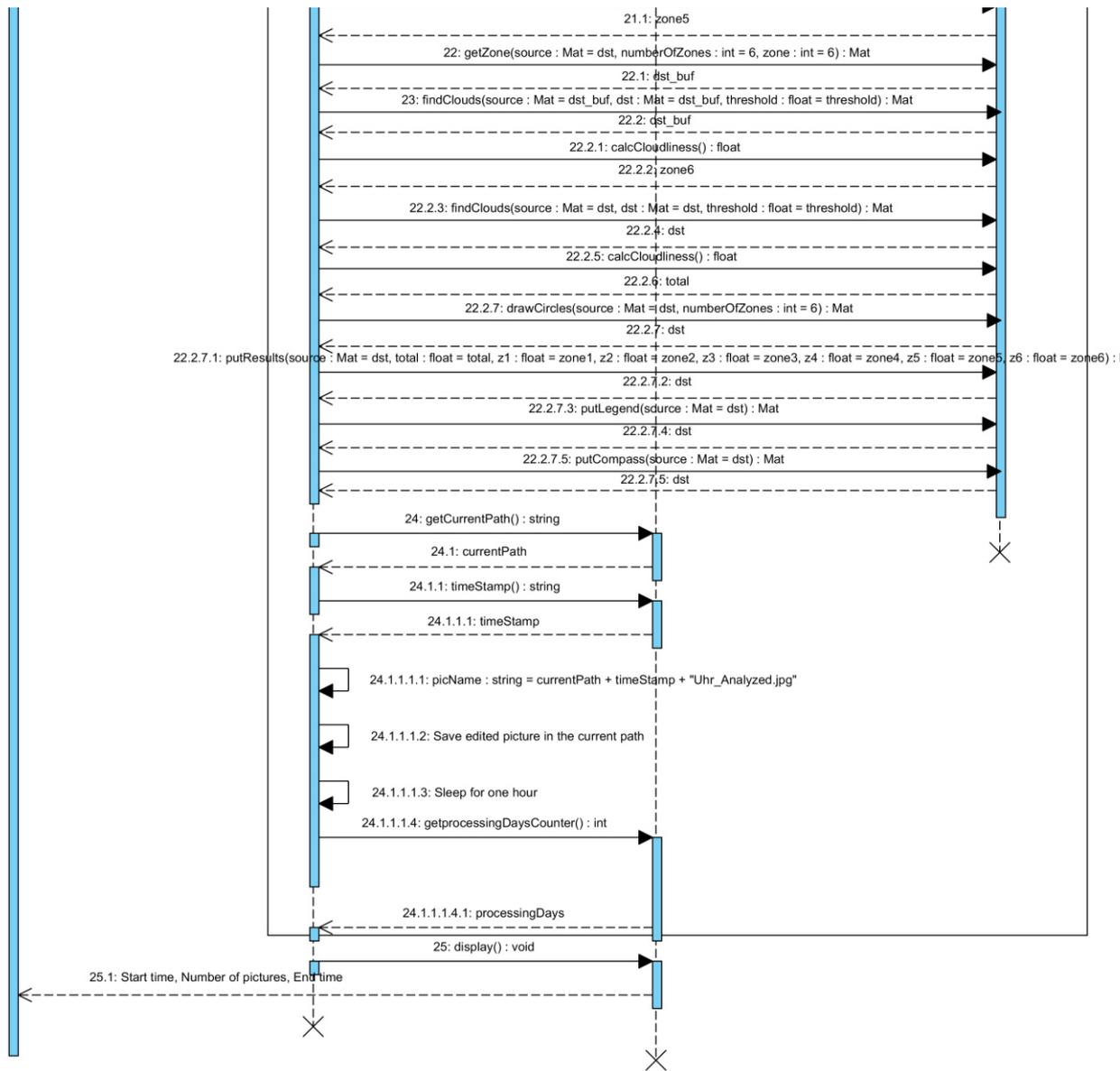
1. Anhang: Aktivitätsdiagramm zur Methode <i>download()</i>	1
2. Anhang: Sequenzdiagramm zur Bildspeicherung	2
3. Anhang: Sequenzdiagramm der Hauptklasse (Laufzeit: 3 Tage).....	3
4. Anhang: DVD.....	5

2. Anhang: Sequenzdiagramm zur Bildspeicherung



3. Anhang: Sequenzdiagramm der Hauptklasse (Laufzeit: 3 Tage)





4. Anhang: DVD

Weiterer Anhang dieser Bachelorarbeit befindet sich auf DVD, einzusehen bei den Prüfern Prof. Dr. Hans-Jürgen Hotop oder Prof. Dr.-Ing. Günter Wüsthube.

Versicherung über die Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 03. Juli 2015
Ort, Datum

Unterschrift