Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Mario Glaser

Konzeption eines verteilten Echtzeitsystems zur mehrdimensionalen Bilddatenverarbeitung unter Einsatz von Android und Netzwerkkameras

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Mario Glaser

Konzeption eines verteilten Echtzeitsystems zur mehrdimensionalen Bilddatenverarbeitung unter Einsatz von Android und Netzwerkkameras

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Reinhard Baran
Zweitgutachter: Prof. Dr.-Ing. Andreas Meisel

Eingereicht am: 17. Juni 2014

Mario Glaser

Thema der Arbeit

Konzeption eines verteilten Echtzeitsystems zur mehrdimensionalen Bilddatenverarbeitung unter Einsatz von Android und Netzwerkkameras

Stichworte

Netzwerkkamera, Stereobildauswertung, Android, Routing

Kurzzusammenfassung

Dieses Dokument beschäftigt sich mit der Frage, ob es möglich ist, mit günstiger, massenproduzierter Hardware ein verteiltes Echtzeitsystem zur mehrdimensionalen Bilddatenverarbeitung zu implementieren. Dieses soll der Überwachung eines Luftraums dienen. Dabei kommen Sony DSC-QX10-Netzwerkkameras und Google Nexus 7-Tablets zum Einsatz. Verschiedene Ansätze zum Aufbau eines solchen Systems werden vorgestellt. Der gewählte Ansatz beinhaltet die Integration der Kameras in ein gemeinsames Netzwerk. Es wird eine Stereo-Distanzmessung implementiert, mit deren Hilfe das System evaluiert wird.

Mario Glaser

Title of the paper

Design of a Distributed Real-Time System for Multidimensional Processing of Image Data Using Android and Network Cameras

Keywords

Network Camera, Stereoscopic Image Evaluation, Android, Routing

Abstract

This document deals with the question whether it is possible to implement a distributed real-time system for processing multidimensional image data using low cost, mass produced hardware. The system is meant to be used for air surveillance. For this, Sony DSC-QX10 network cameras and Google Nexus 7 tablets are employed. Different approaches to design a system like that are presented. The chosen approach includes the integration of the camera into a shared network. For evaluating the system, a stereoscopic distance measurement is implemented.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Mittlerweile sind relativ günstige Objektivkameras verfügbar. Diese hochauflösenden Geräte sind für die Verwendung mit einem Smartphone oder Tablet konzipiert und sollen die dort oft schon vorhandene, integrierte Kamera ersetzen. Auf dem Smartphone/Tablet läuft eine App, die dazu dient, das Sucherbild darzustellen und die verschiedenen Funktionen der Kamera zu steuern.

Der Vorteil der Objektivkameras gegenüber den Kameras der Smartphones liegt in der höheren Bildqualität, die mit der einer Kompaktkamera vergleichbar ist. Die bessere Bildqualität ergibt sich unter anderem aus der hohen Auflösung der verwendeten Bildsensoren und einer großen Brennweite.

Für einige dieser Objektivkameras haben die Hersteller Programmierschnittstellen veröffentlicht, die deren Verwendung in eigenen Programmen erlauben. Der relativ günstige Preis und die Verfügbarkeit von Programmierschnittstellen machen die Objektivkameras auch für andere Anwendungsgebiete interessant. Unter Verwendung zweier baugleicher Kameras lassen sich Stereobildauswertungen vornehmen. Ein solches Stereo-System erlaubt zum Beispiel die Überwachung eines Luftraums. Durch die Auswertung der Bilddaten können die Position, Flughöhe, Fluggeschwindigkeit und Flugrichtung eines Objektes am Himmel bestimmt werden.

Dieses Überwachungssystem könnte zum Beispiel in Verbindung mit autonomen oder ferngesteuerten Drohnen, wie Quadrocoptern, zur Anwendung kommen. Diese unbemannten Drohnen sind zur Zeit Gegenstand der Forschung, und es gibt auch schon Pläne, sie zur Auslieferung kleinerer Waren einzusetzen. Dabei können die aus der Stereobildauswertung gewonnenen Daten einerseits dazu genutzt werden, die Navigation der Drohne selbst zu verbessern oder ein bestimmtes Ziel anzufliegen. Andererseits könnte auch eine Art Fail-Safe-Modus implementiert werden. Wenn eine Drohne ein gewisses Gebiet nicht überfliegen oder einen festgelegten Luftraum nicht verlassen darf, können Gegenmaßnahmen eingeleitet werden.

Ein möglicher Aufbau des beschriebenen Systems könnte wie folgt aussehen: Auf zwei benachbarten Hochhäusern wird jeweils eine Kamera in Verbindung mit einem netzwerkfähigen Zusatzgerät aufgestellt und ausgerichtet. Die Positionierung auf einem Hochhaus ermöglicht dabei ein gutes Sichtfeld. Das Zusatzgerät kann ein Handy, Tablet oder sonstiger Computer sein. Eine Basisstation an einem beliebigen Ort fordert die Bilddaten der Kameras an und nimmt die Stereoauswertung vor.

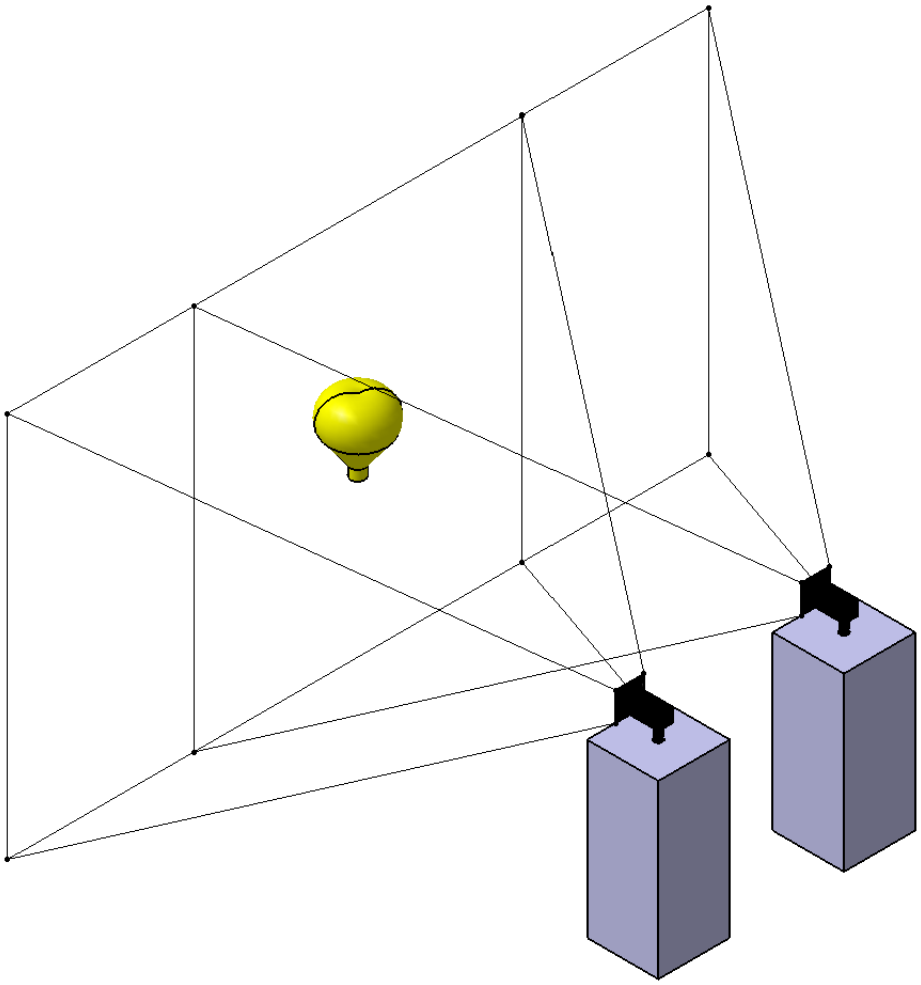

Abbildung 1.1: Zwei Stereokameras auf zwei Hochhäusern

1.2 Zielsetzung

Das Ziel der Arbeit ist es, ein verteiltes Echtzeitsystem zu konzipieren, das als Infrastruktur für das oben vorgestellte oder ein ähnliches Szenario dient. Zum Einsatz kommen dabei zwei Sony-Objektivkameras von Typ DSC-QX10 sowie zwei Tablets Google Nexus 7. Die gegebene Hardware wird auf ihre Tauglichkeit bezüglich ihrer Verwendung in einem Stereo-Kamerasystem untersucht. Dabei wird besonderer Wert auf das Zeitverhalten sowie auf die Netzwerktechnik zur Verbindung der einzelnen Komponenten des Systems gelegt. Zur Demonstration der Funktionsfähigkeit des Systems wird eine Stereo-Distanzmessung implementiert.

1.3 Inhalt der Arbeit

Im Anschluss an die Einleitung werden in Kapitel 2 verschiedene Möglichkeiten verglichen, wie das System zur Bilddatenauswertung aufgebaut werden kann. Dazu wird untersucht, ob die verschiedenen Schnittstellen der verfügbaren Hardware für die Verwendung geeignet sind, und wie man diese am besten zu einem funktionierenden System kombinieren kann. Dabei wird besonders auf die erzielbaren Datenraten und die Reichweiten der Schnittstellen geachtet.

Die API zum Zugriff auf die Objektivkameras wird in Kapitel 3 erklärt. Dabei wird auch auf die benötigten Netzwerkprotokolle eingegangen.

Ein wesentlicher Punkt der Arbeit ist die Integration der Kameras in ein gemeinsames, kabelloses Netzwerk. Der Aufbau eines geeigneten Routing-Systems und die dabei verwendete Hard- und Software werden in Kapitel 4 beschrieben.

Kapitel 5 beschäftigt sich mit der Umsetzung einer einfachen Stereo-Distanzmessung. Es ist aufgeteilt in die Implementierung eines Client-Teils, als Android-App und eines Server-Teils zur Auswertung. Außerdem wird das Verfahren zur Distanzmessung erklärt. Abschließend erfolgt die Analyse des Zeitverhaltens.

Die Ergebnisse der Arbeit werden in Kapitel 6 zusammengefasst.

2 Aufbau des Systems

Das verteilte System kann auf mehrere verschiedene Arten aufgebaut werden. Die möglichen verwendeten Komponenten bieten verschiedene Schnittstellen, die unterschiedlich kombiniert werden können. Im Folgenden wird näher auf die Hardware eingegangen, und mögliche Implementierungen werden vorgestellt. Abschließend werden diese verglichen.

2.1 Hardware

Zur Realisierung des Systems steht folgende Hardware zur Verfügung:

2.1.1 Kameras

Es werden zwei DSC-QX10-Objektivkameras von Sony eingesetzt. Die Kamera kann Bilder mit einer maximalen Auflösung von 4896x3672 Pixeln (ca. 18 Megapixel) bei einem Seitenverhältnis von 4:3 aufnehmen. Beim Seitenverhältnis 16:9 beträgt die Auflösung nur noch 4896x2752 Pixel (ca. 13 Megapixel), da das Bild an der Ober- und Unterseite beschnitten wird. Die Diagonale des Bildsensors beträgt 7.66 mm. Die Bilder werden komprimiert und als JPEG gespeichert. Die Kamera ist mit einem Zehnfach-Zoomobjektiv ausgestattet. Die Brennweite ist von 25 mm bis 250 mm einstellbar. Diese Angaben sind Äquivalenzangaben, die sich auf ein 35mm-Kleinbildformat beziehen. Die Blendenzahl liegt zwischen 3,3 und 5,9. Über ein genormtes UNC-Gewinde, welches im Multimedia-Bereich weit verbreitet ist, kann die Kamera an ein Stativ befestigt werden. Die DSC-QX10 verfügt über eine WLAN-Schnittstelle. Diese erlaubt es, die Kamera von einem Client-Gerät aus zu steuern und aufgenommene Bilder zu übertragen. Es ist auch eine NFC-Schnittstelle integriert. Die Near Field Communication ist ein Funkübertragungsstandard mit geringer Reichweite und Übertragungsrate. Wird ein Client-Gerät in die unmittelbare Nähe der Kamera gehalten, wird automatisch die zugehörige Kamera-App gestartet. Die weitere Datenübertragung erfolgt dann aber wieder per WLAN. Ein USB-Anschluss dient zum Aufladen der Kamera und ermöglicht den Zugriff auf die integrierte Speicherkarte. Eine Steuerung der Kamera ist aber über USB nicht möglich.

2.1.2 Tablets

Als Client-Geräte für die Kameras dienen zwei Google Nexus 7 Tablets. Auf den Tablets läuft das Betriebssystem Android in der Version 4.3. Für Android existiert ein SDK, das die Entwicklung eigener Apps unter Verwendung der Programmiersprache Java erlaubt. Das Tablet verfügt über die Standard-Sensoren und -Schnittstellen, die in den meisten Android-Geräten zu finden sind. Dazu zählen unter anderem GPS, Beschleunigungssensor und Kompass sowie WLAN, Bluetooth, NFC, das Mobilfunknetz und USB. Das Mobilfunknetz erlaubt dabei den Aufbau einer Verbindung mit dem Internet.

2.2 Implementierungsansätze

Es folgt ein Vergleich der verschiedenen Möglichkeiten, das System mit der gegebenen Hardware aufzubauen.

2.2.1 Direkte Verbindung der Kameras mit dem Auswertungscomputer

Abbildung 2.1: Direkte Verbindung mit dem Auswertungscomputer

Zur Steuerung der Kameras kann nur deren WLAN-Schnittstelle verwendet werden. Durch ihre Auslegung für eine Eins-zu-Eins-Verbindung können die Kameras nicht ohne Weiteres in ein bestehendes Netzwerk integriert werden. Dadurch, dass sie einen eingebauten Accesspoint besitzen, stellen sie vielmehr die Netzwerkinfrastruktur bereit. Die Reichweite der Kameras ist auf ca. einen Meter begrenzt, bei höherer Distanz treten bereits Übertragungsfehler auf. Aus der geringen Sendeleistung ergibt sich das Problem, dass die Kameras und der Auswertungscomputer sehr dicht beieinander stehen müssten. Es ist daher besser, ein Client-Gerät, wie zum Beispiel ein Tablet oder Smartphone, zu verwenden. Dieses stellt über WLAN die Verbindung mit der Kamera her. Die Verbindung mit dem Auswertungscomputer wird über eine andere Schnittstelle abgewickelt.

2.2.2 Verbindung der Kameras mit dem Auswertungscomputer unter Verwendung von Client-Geräten

Client-Gerät und NFC

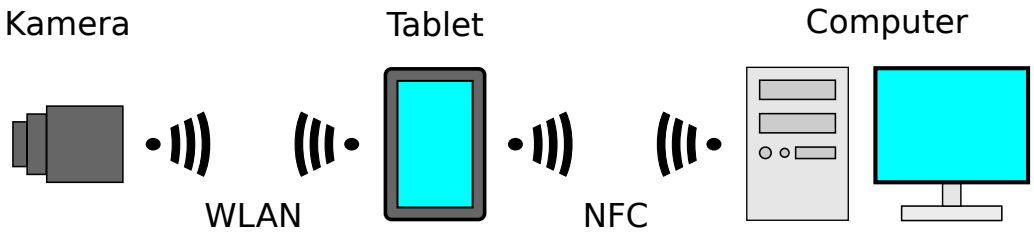

Abbildung 2.2: Verbindung über NFC

NFC ist ein Standard zur bidirektionalen, kabellosen Verbindung zweier Geräte. Der eigentliche Einsatz-Zweck von NFC ist die Übertragung von geringen Datenmengen zwischen mobilen Geräten. Werden diese in die Nähe zueinander gebracht, wird die NFC-Verbindung aktiviert. Diese wird dann verwendet, um die am besten geeignete kabellose Verbindung, wie zum Beispiel WLAN oder Bluetooth, auszuhandeln. Kleinere Datenmengen, wie sie etwa beim Auslesen einer Smartcard anfallen, werden allerdings direkt über NFC übertragen. Die Reichweite von NFC beträgt ca. vier Zentimeter. Die maximale Datenübertragungsrate beträgt $424\,\frac{\text{kbit}}{\text{s}}$. Ein 18 Megapixel-Bild der Kamera hat eine Größe von etwa 6 Megabyte. Betrachtet man die Bildgröße und die Übertragungsrate, so ergibt sich:

$$\frac{6\,\text{MB}}{424\,\frac{\text{kbit}}{\text{s}}} = \frac{48\,000\,\text{kbit}}{424\,\frac{\text{kbit}}{\text{s}}} \approx 113\,\text{s} \tag{2.1}$$

Die Übertragung eines Bildes über NFC dauert im Idealfall also etwa zwei Minuten. Weder die Datenrate, noch die Reichweite sind für den vorliegenden Anwendungsfall ausreichend. Das Problem der geringen Datenrate könnte durch eine Vorverarbeitung der Bilddaten auf dem Client-Gerät entschärft werden. Man verliert dabei aber an Flexibilität, und das Problem der geringen Reichweite bleibt weiterhin bestehen.

Client-Gerät und Bluetooth

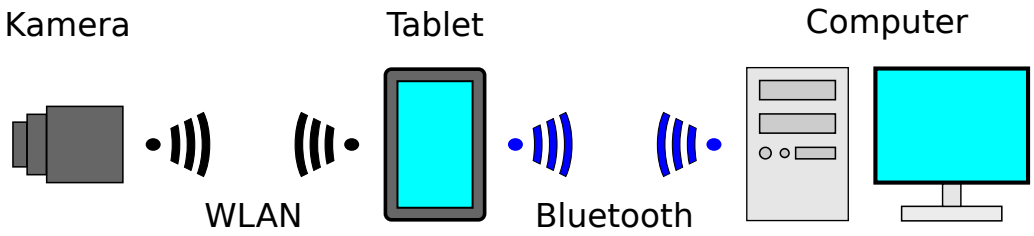

Abbildung 2.3: Verbindung über Bluetooth

Bluetooth ist ein Funkstandard zur Datenübertragung über kurze Entfernungen. Die Reichweite eines Class 1 Bluetooth-Gerätes beträgt unter Idealbedingungen bis zu 100 Meter. Die im Tablet verwendete Bluetooth-4.0-Schnittstelle unterstützt eine maximale Datenrate von $24\,\frac{\text{Mbit}}{\text{s}}$. Betrachtet man wieder die Bildgröße von etwa 6 Megabyte und die Übertragungsrate, so ergibt sich diesmal:

$$\frac{6\,\text{MB}}{24\,\frac{\text{Mbit}}{\text{s}}} = \frac{48\,\text{Mbit}}{24\,\frac{\text{Mbit}}{\text{s}}} = 2\,\text{s} \tag{2.2}$$

Die Datenrate ist für das System ausreichend. Die Reichweite von 100 m unter guten Bedingungen ist aber zu gering, zumal sie innerhalb von Gebäuden noch einmal deutlich niedriger ausfallen wird.

Client-Gerät und USB

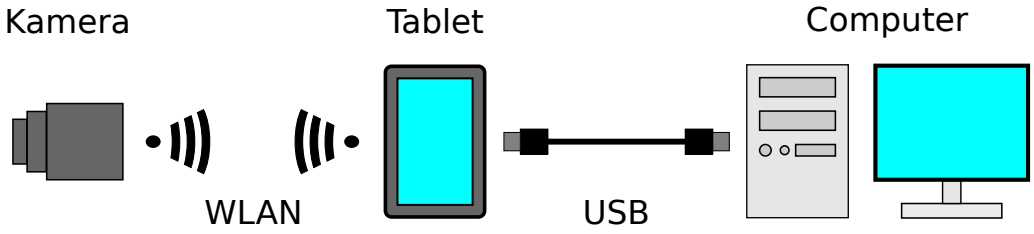

Abbildung 2.4: Verbindung über USB

Android-Geräte erlauben USB-Tethering. Dies ist eine Technik, die eigentlich dazu dient, die mobile Internetverbindung eins Tablets oder Smartphones für einen per USB verbundenen Computer nutzbar zu machen. Auf dem Computer ist das Android-Gerät dann als gewöhnliche Netzwerkschnittstelle ansprechbar. Ist das Tablet oder Smartphone nun nicht mit dem

Internet, sondern mit dem Kamera-WLAN verbunden, so kann der Auswertungscomputer auf die Kamera zugreifen. Jedoch tritt, wenn man mehrere Kameras über die Tablets mit dem Auswertungscomputer verbunden hat, das Problem auf, dass die Kameras die gleiche, feste IP-Adresse besitzen. Dies führt zu Problemen, die nur mit komplexen Routing-Regeln zu lösen sind. USB 2.0 bietet eine sehr hohe Datenrate von $480\ \frac{\text{Mbit}}{\text{s}}$. Die Übertragung des 6 Megabyte-Bildes dauert nur:

$$\frac{6\,\text{MB}}{480\ \frac{\text{Mbit}}{\text{s}}} = \frac{48\,\text{Mbit}}{480\ \frac{\text{Mbit}}{\text{s}}} = 0.1\,\text{s} \tag{2.3}$$

Der begrenzende Faktor ist hierbei schon die Verbindung zwischen der Kamera und dem Client-Gerät, die nach dem WLAN-N-Standard arbeitet und maximal $150\ \frac{\text{Mbit}}{\text{s}}$ übertragen kann. Die Übertragung eines Bildes von der Kamera zum Client-Gerät dauert:

$$\frac{6\,\text{MB}}{150\ \frac{\text{Mbit}}{\text{s}}} = \frac{48\,\text{Mbit}}{150\ \frac{\text{Mbit}}{\text{s}}} = 0.32\,\text{s} \tag{2.4}$$

Die Reichweite der USB-Verbindung ist allerdings für das System nicht ausreichend. USB 2.0 sieht eine maximale Kabellänge von 5 Metern vor.

Client-Gerät und Mobilfunknetz

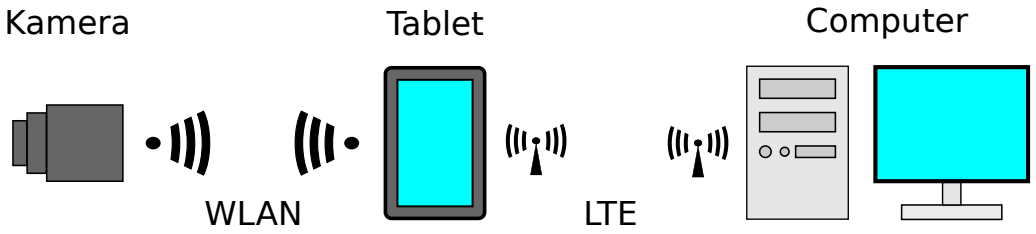

Abbildung 2.5: Verbindung über LTE

Die verwendeten Tablets erlauben die Verwendung des neuen LTE-Funkstandards. LTE steht für Long Term Evolution und erlaubt die drahtlose Verbindung mit dem Internet bei einer maximalen Datenrate von $300\ \frac{\text{Mbit}}{\text{s}}$. Die Übertragung eines Bildes dauert:

$$\frac{6\,\text{MB}}{300\ \frac{\text{Mbit}}{\text{s}}} = \frac{48\,\text{Mbit}}{300\ \frac{\text{Mbit}}{\text{s}}} = 0.16\,\text{s} \tag{2.5}$$

Wie beim USB-Tethering ist auch hier die WLAN-Verbindung der Kamera zum Tablet der Flaschenhals. Hinsichtlich der erzielbaren Datenrate und der beliebigen Reichweite, bedingt

durch eine gute Netzabdeckung, ist die Verwendung des Mobilfunknetzes augenscheinlich optimal.

Jedoch haben Android-Geräte eine softwaretechnische Einschränkung, die eine gleichzeitige Verwendung der WLAN-Schnittstelle und der mobilen Internet-Verbindung ausschließt. Immer wenn eine WLAN-Verbindung verfügbar ist, wird diese für den Datenverkehr bevorzugt. Während dieser Zeit kann nicht über LTE auf das Internet zugegriffen werden. Dieses Problem könnte eventuell gelöst werden, indem man sich Root-Zugriff auf das Tablet verschafft. Allerdings geht hierbei unter Umständen der Garantieanspruch verloren.

Eine weitere Möglichkeit, Mobilfunknetz und WLAN „gemeinsam“ zu benutzen, besteht darin, die Schnittstellen sequentiell zu verwenden. Das Client-Gerät verbindet sich dabei per WLAN mit der Kamera und fordert die Bilddaten an. Nachdem diese empfangen wurden, wird die WLAN-Verbindung getrennt und die Datenverbindung über das Mobilfunknetz hergestellt. Über diese können dann die auszuwertenden Bilddaten übertragen werden. Dabei ist jedoch zu beachten, dass das Trennen und Aufbauen der Verbindungen mehrere Sekunden in Anspruch nehmen kann.

Dieser Vorgang könnte manuell eingeleitet werden, indem das Client-Gerät über die Mobilfunkverbindung auf Anfragen des Auswertungscomputers reagiert oder könnte automatisiert werden. Dabei veranlasst das Client-Gerät die Aufnahme von Bildern zu bestimmten, vorher festgelegten Zeitpunkten und leitet die Bilder selbstständig zur Auswertungsstation weiter.

2.3 Gewählter Aufbau

Im Vergleich der verschiedenen Implementierungsansätze schneidet die Kombination des Tablets mit dem Mobilfunknetz theoretisch am besten ab. Durch die fast unbegrenzte Reichweite ist man sehr flexibel bei der Wahl des Aufstellungsortes der Kameras. Die Datenrate ist ebenfalls ausreichend. Es bleibt allerdings das Problem, dass WLAN und mobile Internetverbindung nicht gleichzeitig genutzt werden können. Da es sich dabei jedoch nur um eine Einschränkung der Software und nicht der Hardware handelt, könnten spätere Versionen von Android eine gleichzeitige Verwendung beider Schnittstellen erlauben.

In Kapitel 4 wird eine Lösung vorgestellt, die es erlaubt, alle beteiligten Geräte kabellos in einem WLAN zu verbinden. Diese dient primär dazu, die Entwicklung einer beispielhaften Stereo-Anwendung zu ermöglichen, welche in Kapitel 5 vorgestellt wird.

Sollte die Einschränkung bei der Verwendung der Schnittstellen in Zukunft aufgehoben werden, bedarf es lediglich einer Änderung der IP-Adresse des Auswertungscomputers in den Apps der Tablets, um die Verwendung des mobilen Datennetzes zu ermöglichen.

3 Die Camera Remote API

Die verwendeten Netzwerkkameras Sony DSC-QX10 können über die Sony Camera Remote API angesprochen werden. Die API liegt derzeit in der Version 1.50 vom 15.04.2014 vor und wird von Sony noch als Beta-Version bezeichnet. Die erste Version (1.0) wurde am 01.09.2013 veröffentlicht. Es handelt sich also um ein recht junges Projekt, sodass davon ausgegangen werden muss, dass noch weitreichende Änderungen einfließen werden.

Die Camera Remote API kann mit verschiedenen WLAN-fähigen Kameras benutzt werden. Der unterstützte Funktionsumfang unterscheidet sich jedoch stark von Modell zu Modell und ist auch von der verwendeten Firmware-Version der Kamera abhängig. Ein Update wird notwendig, wenn die API erweitert wird. So wurden in der Version 1.50 einige wichtige Funktionen, die vorher nur über Sonys PlayMemories App genutzt werden konnten, verfügbar gemacht. Dies erforderte ein Firmware-Update der DSC-QX10-Kameras.

Abbildung 3.1 zeigt, welche API-Aufrufe von welchen Kameras unterstützt werden. Für die Stereobildauswertung sind vor allem die Funktionen wichtig, die eine möglichst genaue Einflussnahme auf das aufgenommene Bild erlauben. Die Bilder der beiden Kameras sollten möglichst identisch sein. Besonders große Auswirkungen hierauf haben die Funktionen Shutter-Speed (Verschlusszeit), F-Number (Blendenzahl), ISO-Speed-Rate (Empfindlichkeit) und White-Balance (Weißabgleich). Wie der Tabelle zu entnehmen ist, können einige Parameter, wie Blendenzahl und Verschlusszeit, bei der DSC-QX10 nicht manuell angepasst werden. Bei der DSC-QX100 ist dies jedoch möglich.

Supported API groups for each compatible cameras

	HDR-AS15*2	HDR-AS30 HDR-AS100	HDR-MV1	A7R*3 A7*3 NEX-6*3 NEX-5R*3 NEX-5T*3 A5000*3 A6000*3 DSC-HX60*3 DSC-HX400*3	DSC-QX100*2 DSC-QX10*2
Shoot mode	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Still capture	No	**Yes***4	No	**Yes**	**Yes**
Movie recording	**Yes**	**Yes**	**Yes**	No	**Yes**
Audio Recording	No	No	**Yes**	No	No
Liveview	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**
Zoom	No	No	No	**Yes**	**Yes**
Half-press shutter	No	No	No	No	**Yes***2
Touch AF position	No	No	No	No	**Yes***2
Self-timer	No	No	No	**Yes**	**Yes**
Exposure mode	No	No	No	No	**Yes***2
Focus mode	No	No	No	No	**Yes***2*5
Exposure compensation	No	No	No	No	**Yes***2
F number	No	No	No	No	**Yes***2*5
Shutter speed	No	No	No	No	**Yes***2*5
ISO speed rate	No	No	No	No	**Yes***2
White balance	No	No	No	No	**Yes***2
Still size	No	No	No	No	**Yes***2
Postview image size	No	No	No	**Yes**	**Yes**
Beep mode	No	No	No	No	**Yes***2
Date/time setting	No	No	No	No	**Yes***2
Event notification	**Yes**	**Yes**	**Yes**	**Yes**	**Yes***6
Camera setup*1	No	No	No	**Yes**	No
Server information	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**

API groups - Compatible cameras

*1: Some camera models need "Camera setup" API call before accessing camera remote shooting functions.
*2: The latest firmware update is needed.
*3: These cameras are compatible with the PlayMemories Camera Apps "Smart Remote Control" application.
The latest version of the application should be installed and started to use the APIs.
*4: These cameras support only "actTakePicture".
*5: Only DSC-QX100 supports the APIs group.
*6: These cameras support "getEvent (version 1.1)" in addition to "getEvent (version 1.0)".

Abbildung 3.1: Übersicht über die unterstützten Kameras und deren Funktionsumfang
Quelle: (Sony Corporation, 2014a)[S. 8, Abb. 1]

3.1 Camera Remote API beta SDK

Um die API nutzbar zu machen, bietet Sony das „Camera Remote API beta SDK“ zum Download an. Dieses beinhaltet zwei PDF Dokumente, den Development Guide und die API Reference. Im Development Guide wird beschrieben, wie man die Verbindung mit der Kamera aufnimmt und die API-Funktionen nutzt. Die API Reference ist im Wesentlichen eine Beschreibung der verschiedenen Funktionen und Datenformate. Sie enthält aber auch einige Beispielsequenzen, die zeigen, wie man ein Bild aufnimmt oder auf Ereignisse reagiert. Im SDK ist außerdem Beispielquellcode für die Betriebssysteme Android und iOS enthalten. Es handelt sich um ein einfaches Programm, das das Sucherbild darstellen und Fotos aufnehmen kann.

3.2 Verwendete Protokolle

Die Camera Remote API ist plattformunabhängig (Sony Corporation, 2014b, S. 6). Sie kann von jedem Betriebssytem genutzt werden, das über eine HTTP-Implementierung verfügt. Die einzelnen API-Funktionen sind entfernte Methodenaufrufe. Die Methoden werden von der Kamera bereitgestellt. Das verwendete Protokoll heißt JSON-RPC. Im Folgenden werden die wichtigsten verwendeten Protokolle erklärt.

3.2.1 HTTP

Das Hypertext Transfer Protocol ist ein Protokoll zur Nachrichtenübertragung in einem Netzwerk. Es kennt nur zwei Nachrichtentypen, Request und Response. Ein Client sendet eine Anfrage (Request) an einen Server, und dieser sendet eine Antwort (Response). Man nennt ein solches Protokoll auch Request/Response-Protokoll. Die zur Zeit am weitesten verbreitete Version HTTP/1.1 wurde von der Internet Engineering Taskforce in der RFC2616 spezifiziert (vgl. Internet Engineering Taskforce (1999a)). Im OSI-Schichtenmodell ist HTTP ein Protokoll der Anwendungsschicht.

Ein Server stellt bestimmte Ressourcen zur Verfügung. Diese müssen eindeutig identifiziert werden. Dazu dient die URI (Uniform Resource Identifier). Eine URI kann ein Ort (URL: Uniform Resource Location) oder ein Name (URN: Uniform Resource Name) sein. Im Kontext von HTTP ist eine URI ein formatierter String, der eine Ressource identifiziert.

HTTP URL Aufbau:

```
"http:" "//" host [ ":" port ] [ abs_path [ "?" query ]]
```

Listing 3.1: host: Adresse des Zielrechners
port: Port des Services auf dem Zielrechner (optional, standardmäßig 80)
abs_path: Pfad zu Ressource auf dem Zielrechner (optional)
query: optionale Anfrage

Die DSC-QX10 stellt die Ressource Endpoint URL bereit. Über diese werden die API-Funktionen aufgerufen.

Endpoint URL:

```
http://10.0.0.1:10000/sony/camera
```

Da HTTP ein zustandsloses Protokoll ist, sollte es über eine verlässliche Verbindung übertragen werden. Normalerweise kommt hier TCP/IP zum Einsatz. Es ist aber auch möglich, HTTP über UDP zu versenden. Dieses Verfahren wird vom Simple Service Discovery Protocol benutzt, das hier dazu dient, eine Kamera im Netzwerk zu identifizieren. Eine genauere Beschreibung folgt in Kapitel 3.2.4.

HTTP unterstützt verschiedene Zugriffsmethoden auf Ressourcen. Für die Camera Remote API werden nur zwei davon benötigt: GET und POST. GET empfängt Daten von einer Ressource und POST sendet Daten an eine Ressource. Mit Hilfe von POST werden in der Camera Remote API die entfernten Methodenaufrufe realisiert. Der Client ruft zum Beispiel per POST die Methode actTakePicture. ActTakePicture veranlasst die Kamera, ein Bild aufzunehmen. Die Antwort der Kamera ist eine URL auf das aufgenommene Bild. Auf Diese URL wird dann GET angewendet. Die Antwort besteht aus den Bilddaten.

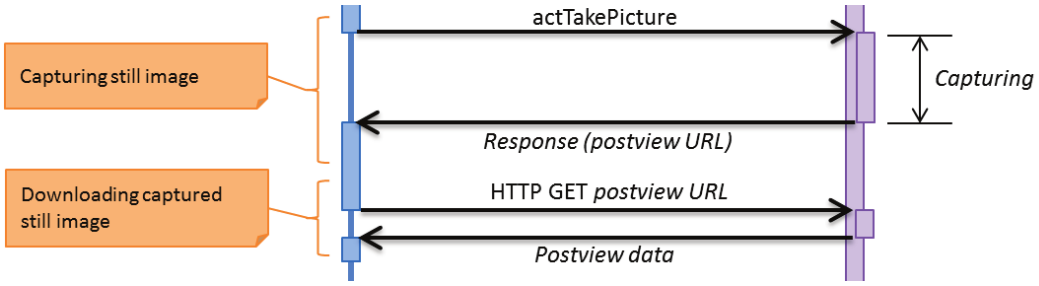

Abbildung 3.2: Beispielsequenz für die Aufnahme eines Bildes
Quelle: (Sony Corporation, 2014a)[S. 117, Abb. 1]

Der Server antwortet auf die Anfrage eines Clients unter anderem mit einer Statuszeile. Diese enthält Status-Codes, die den Erfolg oder Misserfolg der Anfrage anzeigen. Die Status-Codes sind in 5 Kategorien eingeteilt:

Informational 1xx: Der Server hat die Anfrage des Clients erhalten, die Bearbeitung dauert aber noch an.

Succesfull 2xx: Die Anfrage des Clients wurde erfolgreich empfangen, verstanden und akzeptiert.

Redirection 3xx: Der Client muss weitere Schritte unternehmen, um die Anfrage zu erfüllen. Es kann sich um z.B. um eine Umleitung oder eine geänderte URL handeln.

Client Error 4xx: Es ist ein Fehler auf der Client-Seite aufgetreten.

Server Error 5xx: Es ist ein Fehler auf der Server-Seite aufgetreten, oder der Server ist nicht in der Lage, die Anfrage zu bearbeiten.

3.2.2 JSON

Die Javascript Object Notation ist ein sprachunabhängiges Format für den Datenaustausch. Es ist ein Textformat, das der Serialisierung strukturierter Daten dient. Die Syntax ist an Javascript angelehnt. JSON wird in der RFC4627 spezifiziert (vgl. Internet Engineering Taskforce (2006)). Die Designziele von JSON sind Minimalität, Portabilität, Textbasierung und Kompatibilität zu Javascript ("minimal, portable, textual, and a subset of JavaScript", (Internet Engineering Taskforce, 2006, Z. 53f)).

Es gibt vier primitive Datentypen:

string: Folge von null oder mehr Unicode-Zeichen.

number: Beliebige Zahl.

boolean: Wahrheitswert.

null: Leerer Wert.

Und zwei zusammengesetzte Datentypen:

object: Ungeordnete Ansammlung von null oder mehr Name/Werte-Paaren. Ein Name ist dabei ein String, und ein Wert kann ein beliebiger primitver Datentyp sein.

array: Geordnete Folge von null oder mehr Werten.

Beispiel für ein JSON-Objekt:

```
{
    "Image": {
        "Width":  800,
        "Height": 600,
        "Title":  "View from 15th Floor",
        "Thumbnail": {
            "Url":    "http://www.example.com/image/481989943",
            "Height": 125,
            "Width":  "100"
        },
        "IDs": [116, 943, 234, 38793]
    }
}
```

Beispiel für ein JSON-Array:

```
{
    "precision": "zip",
    "Latitude":  37.7668,
    "Longitude": -122.3959,
    "Address":   "",
    "City":      "SAN FRANCISCO",
    "State":     "CA",
    "Zip":       "94107",
    "Country":   "US"
},
{
    "precision": "zip",
    "Latitude":  37.371991,
    "Longitude": -122.026020,
    "Address":   "",
    "City":      "SUNNYVALE",
    "State":     "CA",
    "Zip":       "94085",
    "Country":   "US"
}
```

3.2.3 JSON-RPC

JSON-RPC ist ein einfaches Remote-Procedure-Call-Protokoll, das im Hinblick auf Einfachheit spezifiziert wurde (vgl. (JSON-RPC.ORG, 2005)).

Zwei Teilnehmer unterhalten eine Datenverbindung. Dabei können sie jeweils Methoden des anderen Teilnehmers aufrufen. Es gibt drei Nachrichtentypen:

Request: Ein Request ist ein einzelnes Objekt, das durch JSON serialisiert wird. Es hat drei Bestandteile.

method: Ein String, der den Namen der aufzurufenden Methode enthält.

params: Ein Array von Objekten, die als Argumente an die aufzurufende Methode übergeben werden.

id: Kann von beliebigem Typ sein und wird benutzt, um Request und Response einander zuzuordnen.

Response: Ein Response ist ein einzelnes Objekt, das durch JSON serialisiert wird. Es hat drei Bestandteile.

result: Ein Objekt, das von der aufgerufenen Methode zurückgegeben wird. Im Fehlerfall muss es null sein.

error: Ein Objekt, das zurückgegeben wird, wenn es einen Fehler beim Aufruf der Methode gab. Wenn kein Fehler aufgetreten ist, muss es null sein.

id: Die ID muss vom selben Typ und Wert sein, wie die ID des Requests, auf den geantwortet wird.

Notification: Die Notification ist ein Sonderfall des Requests, auf die es keine Response gibt. Sie hat die gleichen Bestandteile wie das Request-Objekt, mit dem Unterschied, dass die ID null sein muss.

Um eine entfernte Methode aufzurufen, muss ein Teilnehmer einen Request an den anderen Teilnehmer senden. Sofern der Request keine Notification ist, muss er mit einer Response beantwortet werden.

JSON-RPC wird normalerweise direkt über eine TCP/IP-Verbindung benutzt. Es kann aber auch über HTTP, unter Verwendung der Zugriffsmethode POST, verwendet werden. In der Camera API ist dies der Fall.

Die Camera Remote API hält sich nicht in allen Punkten an die JSON-RPC-Spezifikation. Sie fügt einige Erweiterungen hinzu, hat aber auch einige Einschränkungen.

Erweiterungen:

- Im Falle eines erfolgreichen Requests wird kein Error-Element zurückgegeben. Im Fehlerfall wird kein Result-Element zurückgegeben.

- Das Error-Element ist ein Array, definiert als [error_code, error_message]. Error_code ist vom Typ Integer und error_message ist ein String.
- Der Client muss im Request das Element „version“ angeben. „Version“ ist vom Typ String und wird durch zwei, durch einen Punkt getrennte Nummern angegeben (z.B. „1.0“).

Einschränkungen:

- Die Elemente param und result sind Arrays fester Länge.
- Einige API-Aufrufe geben „results“ statt „result“ zurück.
- Die ID ist vom Typ Integer und darf nicht 0 sein.

3.2.4 SSDP

Das Simple Service Discovery Protocol dient zur Auffindung von HTTP-Ressourcen durch einen HTTP-Client in einem LAN. Das heißt, ein HTTP-Client benötigt einen speziellen Service im LAN und muss feststellen, welche HTTP-Ressource diesen bereitstellt. Im einfachsten Fall kommt SSDP ohne Konfiguration, Management und Administration aus (vgl. Internet Engineering Taskforce (1999b)).

Ablauf: Clients können einen Service auffinden, indem sie einen „Discovery Request“ an die spezielle Multicast-Adresse 239.255.255.250 Port 1900 senden. Die Übertragung erfolgt dabei per HTTP über UDP. Die von SSDP spezifizierte Zugriffsmethode heißt M-SEARCH. Der gewünschte Service wird über das Search-Target (ST) angegeben.

```
1 M-SEARCH * HTTP/1.1
2 HOST: 239.255.255.250:1900
3 MAN: "ssdp:discover"
4 MX: seconds to delay response (ex. MX: 1)
5 ST: urn:schemas-sony-com:service:ScalarWebAPI:1
6 USER-AGENT: OS/version product/version
```

Listing 3.2: Beispiel aus dem Development Guide S.10

Services empfangen die Multicast-Nachrichten und schicken ihre Antworten, bei Übereinstimmung des Search-Targets, per Unicast an den anfragenden Client. Die Übertragung erfolgt wieder per HTTP über UDP.

```
1 HTTP/1.1 200 OK
2 LOCATION: http://10.0.0.1:64321/dd.xml
3 CACHE-CONTROL: max-age=1800
```

```
EXT:
SERVER: OS/version UPnP/1.0 product/version
ST: urn:schemas-sony-com:service:ScalarWebAPI:1
```

Listing 3.3: Beispiel aus dem Development Guide S.11

Ein Service kann auch Announcements an die Multicast-Adresse schicken, um auf sich aufmerksam zu machen.

3.3 Zugriff auf Kamerafunktionen

Die Kamera stellt einen WLAN-Accesspoint zur Verfügung, mit dem sich das Client-Gerät (Tablet, Handy, Computer etc.) zunächst verbinden muss.

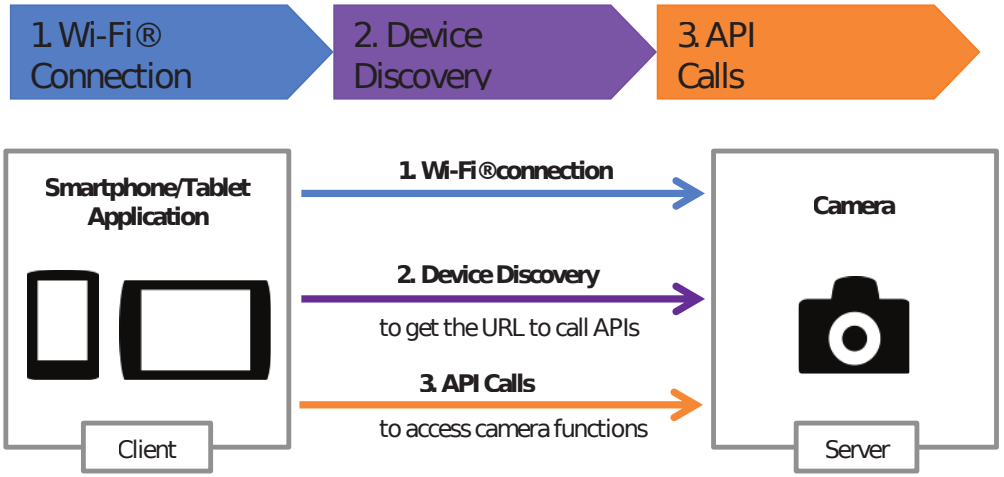

Abbildung 3.3: Zugriff auf Kamera
Quelle: (Sony Corporation, 2014b)[S. 9, Abb. 2]

Um API-Aufrufe ausführen zu können, wird die sogenannte Endpoint-Url benötigt. Diese setzt sich aus der ActionList_URL und dem API_Service zusammen. Die ActionList_URL beinhaltet die IP-Adresse der Kamera und einen Port, gefolgt von „/sony“.

ActionList_URL Beispiel:

```
http://10.0.0.1:10000/sony
```

Der API_Service unterteilt die API in drei Funktionsbereiche:

guide: Wird benutzt, um unterstützte API Services zu ermitteln

camera: Setzen/Lesen von Kamereinstellungen und Zugriff auf Kamerafunktionen

system: Setzen/Lesen von Systemeinstellungen

Möchte man zum Beispiel ein Bild aufnehmen, würde die Endpoint-URL wie folgt lauten:
Endpoint-URL:

```
1 http://10.0.0.1:10000/sony/camera
```

Da die Endpoint-Url nicht bei allen Kameramodellen gleich ist, ist vorgesehen, dass sie automatisch ermittelt wird. Dazu dient die „Device Discovery“. Sie wird mit Hilfe von SSDP durchgeführt.

Das Client-Gerät muss einen Discovery-Request mit dem Search-Target „urn:schemas-sony-com:service:ScalarWebAPI:1“ an die Multicast-Adresse 239.255.255.250:1900 senden. Kameras, die die Camera Remote API unterstützen, antworten mit einem Discovery-Response. Dieser enthält einen LOCATION-Header, der auf die Device Description verweist.

```
1 LOCATION http://<xmlhost>:<xml_port>/dd.xml
```

Die Device Description ist eine XML-Datei, die vom Client per HTTP-GET heruntergeladen werden muss. Sie enthält Informationen über das Kameramodell. Eine Kamera unterstützt die Remote API, wenn der Abschnitt „X_ScalarWebAPI_Deviceinfo“ vorhanden ist. In diesem Abschnitt finden sich auch die ActionList_URL und die API_Services. Mit diesen Informationen kann die Endpoint-URL zusammengesetzt werden.

API-Aufrufe können nun ausgeführt werden, indem die serialisierten JSON-Objekte per HTTP-POST an die Endpoint-URL geschickt werden.

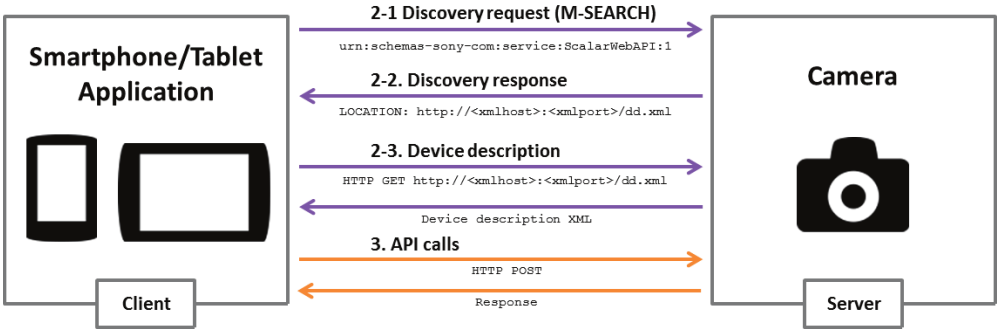

Abbildung 3.4: Ablauf der Device Discovery und API-Aufrufe
Quelle: (Sony Corporation, 2014b)[S. 10, Abb. 1]

4 Integration der Kameras in ein gemeinsames kabelloses Netzwerk

Die verwendeten Netzwerkkameras sind von Sony nur für eine direkte Eins-zu-eins-Verbindung zwischen der Kamera und dem Endgerät vorgesehen ((Sony Corporation, 2014b, S.9)). Hierfür stellt die Kamera einen WLAN-Accesspoint bereit, mit dem sich genau ein Client zur Zeit verbinden kann.

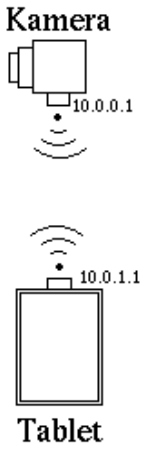

Abbildung 4.1: Netzwerktopologie ohne Router

Da die verwendeten Tablets nur über eine WLAN-Schnittstelle verfügen, ist es nicht möglich, gleichzeitig eine Verbindung zu einem weiteren Gerät, wie zum Beispiel einer zweiten Kamera oder einem auswertenden PC, zu unterhalten. Später soll die Verbindung zum auswertenden Computer zwar über das Handy-Netz ablaufen, aufgrund der in Kapitel 2 beschriebenen Einschränkungen bei der gleichzeitigen Verwendung der WLAN- und Mobilfunkschnittstelle ist dies jedoch nicht ohne weiteres umsetzbar.

Es muss daher eine Möglichkeit gefunden werden, die es erlaubt, beide Kameras, Tablets und weitere Geräte in einem gemeinsamen, hinreichend schnellen Netzwerk zu betreiben. Innerhalb dieses Netzwerks müssen die Kameras eindeutig adressierbar sein.

Im Folgenden wird die gewählte Lösung für dieses Problem beschrieben.

4.1 Lösungsansatz

Die gewählte Lösung sieht vor, dass ein zusätzlicher Computer als Router fungiert und die Adressierbarkeit der Kameras über vorher festgelegte IP-Adressen erlaubt. Der Computer stellt zusätzlich einen WLAN-Accesspoint bereit, mit dem sich alle weiteren beteiligten Geräte verbinden können. Um dies zu erreichen, ist der Router mit drei WLAN-Schnittstellen ausgestattet. Eine davon stellt den Accesspoint bereit, die anderen zwei verbinden sich mit jeweils einer Kamera.

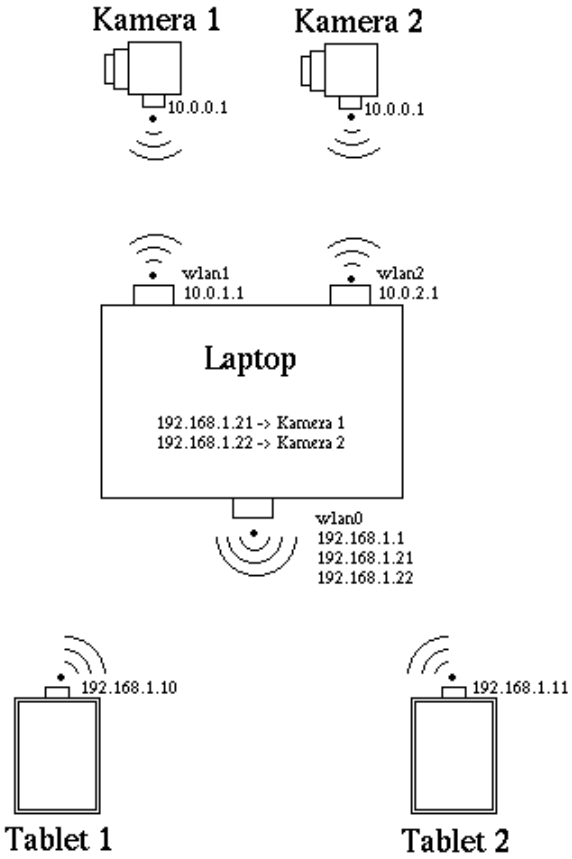

Abbildung 4.2: Netzwerktopologie mit Router

Im weiteren Verlauf wird auf die für die Implementierung genutzte Hard- und Software eingegangen.

4.1.1 Verwendete Hardware

Als Router kommt ein IBM Thinkpad T43 Laptop zum Einsatz. Es handelt sich schon um ein recht altes Gerät. Für die benötigte Netzwerkfunktionalität ist keine hohe Rechenleistung erforderlich, es werden jedoch spezielle Anforderungen an die integrierte WLAN-Karte gestellt. Um mit dem Programm hostapd, auf das im Kapitel 4.1.2 näher eingegangen wird, einen Accesspoint betreiben zu können, muss die WLAN-Karte den sogenannten Host-Modus unterstützen. Im Thinkpad ist eine kompatible Atheros AR5212 Karte verbaut.

Für die Verbindung zu den Kameras kommen zwei Sitecom USB-WLAN-Sticks mit Realtek-Chipsätzen zum Einsatz. An diese Geräte werden keine speziellen Anforderungen gestellt.

4.1.2 Verwendete Software

Auf dem Laptop kommt das Betriebssystem OpenSuse-Linux in der Version 12.3 zum Einsatz. Linux wurde gewählt, weil es schon von Haus aus alle benötigten Netzwerkfunktionalitäten und -Tools bereitstellt und darüberhinaus große Flexibilität in der Konfiguration bietet.

Zur Bereitstellung des Accesspoints dient das Programm hostapd. Hostapd ist ein softwarebasierter IEEE 802.11 Accesspoint sowie ein IEEE 802.1X/WPA/WPA2/EAP/RADIUS Authenticator (Linux Wireless).

Um den Geräten, die sich mit dem Accesspoint verbinden, automatisch eine IP-Adresse zuweisen zu können, wird das Programm dnsmasq verwendet. Dnsmasq ist sowohl ein DNS- als auch ein DHCP-Server. Im Rahmen dieser Arbeit wird aber nur die DHCP-Funktionalität verwendet.

Das Routing der Pakete wird mit Hilfe der Linux-Firewall Netfilter und des Programmpakets Iproute2 realisiert. Iproute2 erlaubt es die Routing-Tabellen des Linux-Kernels zu verändern. Netfilter wird benötigt, um Pakete zu markieren und Source- und Destination-NAT zu ermöglichen.

4.2 Aufbau

Die Einrichtung des Routers erfolgt mit Hilfe eines Shell-Skripts. Die benutzen Befehle zur Netzwerkkonfiguration benötigen Root-Rechte.

4.2.1 Accesspoint

Die drei WLAN-Schnittstellen werden vom Linux-Kernel, bei null beginnend, durchnummeriert und mit dem Präfix "wlan"versehen. Wlan0 ist zuständig für den Accesspoint, wlan1 stellt die Verbindung zur ersten Kamera her, und wlan2 stellt die Verbindung zur zweiten Kamera her.

Im nächsten Schritt wird der Accesspoint eingerichtet. Dieser soll unter der IP-Adresse 192.168.1.1 erreichbar sein. Die beiden Kameras sollen unter den festen Adressen 192.168.1.21 und 192.168.1.22 erreichbar sein. Der Schnittstelle wlan0 werden daher drei Adressen zugewiesen:

```
ifconfig wlan0 192.168.1.1 netmask 255.255.255.0 up
ip addr add 192.168.1.21/24 dev wlan0
ip addr add 192.168.1.22/24 dev wlan0
```

Dann wird hostapd gestartet:

```
hostapd hostapd.conf
```

Hostapd wurde so konfiguriert, dass es einen Accesspoint mit der SSID CAMERA_AP anlegt. Als Verschlüsselungsverfahren wurde WPA2 gewählt, da es auch schon bei den Kameras zum Einsatz kommt. Das Passwort lautet "1cam2era".

Für die automatische Adressvergabe an Clients wird der DHCP-Server dnsmasq gestartet.

```
dnsmasq -C dnsmasq.conf
```

In „dnsmasq.conf“ wurde festgelegt, dass IP-Adressen in dem Bereich von 192.168.1.10 bis 192.168.1.20 vergeben werden. Dies ist eine willkürliche Einschränkung. Es ist nur wichtig, dass die für die Kameras reservierten Adressen .21 und .22 nicht vergeben werden.

4.2.2 Verbindung zu den Kameras

Die beiden Kameras haben die gleiche IP-Adresse 10.0.0.1 mit der Netzwerkmaske 255.255.0.0. Den Schnittstellen wlan1 und wlan2 wird nun jeweils eine eigene Adresse aus dem Adressbereich der Kameras zugewiesen:

```
ifconfig wlan1 10.0.1.1 netmask 255.255.0.0 up
ifconfig wlan2 10.0.2.1 netmask 255.255.0.0 up
```

Mit Hilfe des Programmes wpa_supplicant werden jetzt die WLAN-Verbindungen zu den Kameras hergestellt:

```
wpa_supplicant -i wlan1 -c wpa_supplicant_camera1.conf
wpa_supplicant -i wlan2 -c wpa_supplicant_camera2.conf
```

In den Konfigurationsdateien werden die SSID der jeweiligen Kamera sowie das zugehörige Passwort (PSK: Pre-Shared Key) und die Art der Verschlüsselung (hier WPA2) festgelegt.

4.2.3 Routing

Normalerweise wird die Route für ein Paket in einem TCP/IP-Netzwerk anhand seiner Zieladresse mit Hilfe einer Routingtabelle ermittelt. Dieses Verfahren ist hier allerdings nicht anwendbar, da die verwendeten Netzwerkkameras nicht zum Betrieb in einem gemeinsamen Netzwerk bestimmt sind und sich eine IP-Adresse teilen. Zur Lösung des Problems kommt das sogenannte Policy Routing zum Einsatz. Das Policy Routing erlaubt es, eine Route für ein Paket nicht nur anhand seiner Zieladresse zu ermitteln, sondern auch anhand anderer Eigenschaften, wie zum Beispiel der Quelladresse, Ports oder sogar des Paketinhalts. Die zielbasierte Routingtabelle wird durch die Routing Policy Database (RPDB) ersetzt. Die RPDB ist eine Liste von Regeln (rules), denen eine Priorität zugeordnet wird. Es soll eine Regel geben, die dafür sorgt, dass Pakete, welche an 192.168.1.21 adressiert sind, über die Schnittstelle wlan1 geleitet werden. Eine weitere Regel soll dafür sorgen, dass Pakete für 192.168.1.22 über wlan2 geleitet werden. Damit dies funktioniert, werden auf wlan0 eingehende Pakete mit einer Nummer markiert. Dafür kommt die Linux-Firewall zum Einsatz:

```
iptables -t mangle -I PREROUTING -d 192.168.1.21
    -j MARK --set-mark 1
iptables -t mangle -I PREROUTING -d 192.168.1.22
    -j MARK --set-mark 2
```

Die Firewall-Regeln besagen, dass ein eingehendes Paket, das an 192.168.1.21 (die erste Kamera) adressiert ist, mit einer Eins markiert wird. Analog dazu wird ein an die zweite Kamera adressiertes Paket mit einer Zwei markiert. Die Kette „PREROUTING“ wird verwendet, da die Markierung der Pakete vor dem eigentlichen Routing erfolgen muss.

Jetzt wird für jede WLAN-Schnittstelle eine Routing-Tabelle angelegt:

```
ip route add default dev wlan1 table 1
ip route add default dev wlan2 table 2
```

Die mit einer eins markierten Pakete sollen die Routing-Tabelle 1 benutzen. Die mit einer zwei markierten Pakete sollen die Routing-Tabelle 2 benutzen. Dafür werden zwei Policy Routing-Regeln hinzugefügt:

```
ip rule add fwmark 1 pref 10 lookup 1
ip rule add fwmark 2 pref 11 lookup 2
```

Damit das Routing der Pakete funktioniert, muss noch das IP-Forwarding des Kernels eingeschaltet werden:

```
sysctl -w net.ipv4.ip_forward=1
```

Ohne diese Option können keine Netzwerkpakete weitergeleitet werden. Ein Routing wäre nicht möglich.

4.2.4 NAT

Ein Paket, das an die erste Kamera adressiert ist, hat die Zieladresse 192.168.1.21. Mit Hilfe des Routings wird es an die Schnittstelle wlan1 weitergeleitet. Allerdings würde es die Kamera nicht erreichen, denn diese hat die Adresse 10.0.0.1. Die Zieladresse des Pakets muss also ausgetauscht werden. Hierfür kommt wieder die Firewall zum Einsatz, diesmal unter Verwendung der Tabelle "nat". Zur Identifizierung der zu modifizierenden Pakete dient die in 4.2.3 gesetze Markierung:

```
iptables -t nat -A PREROUTING -m mark --mark 1
    -j DNAT --to-destination 10.0.0.1
iptables -t nat -A PREROUTING -m mark --mark 2
    -j DNAT --to-destination 10.0.0.1
```

Diese Technik, bei der die Zieladresse eines Paketes ersetzt wird, heißt Destination Network Address Translation.

Da die Kamera nur Pakete entgegennimmt, die aus ihrem Adressbereich stammen, muss auch die Quelladresse eines an die Kamera gerichteten Pakets angepasst werden. Hierfür kommt SNAT (Source Network Address Translation) zum Einsatz:

```
iptables -t nat -A POSTROUTING -o wlan1 -j SNAT --to 10.0.1.1
iptables -t nat -A POSTROUTING -o wlan2 -j SNAT --to 10.0.2.1
```

Ein Paket, das den Router über die Schnittstelle wlan1 verlässt, erhält die Quelladresse 10.0.1.1. Ein Paket, das den Router über wlan2 verlässt, erhält die Quelladresse 10.0.2.1.

4.2.5 ARP

Damit die WLAN-Verbindung funktionieren kann, muss einer Kamera die hardware-abhängige MAC-Adresse des verbundenen Endgeräts bekannt sein. Zur Ermittlung dieser Adresse dient das Adress Resolution Protocol. Die Kamera verschickt einen ARP-Request, einen Broadcast, der eine IP-Adresse enthält. Das Gerät, dass diese IP-Adresse besitzt, beantwortet den Request mit seiner MAC-Adresse. Im hier verwendeten Netzwerkkonzept gibt es aber einen Sonderfall. Die Schnittstellen wlan1 und wlan2 befinden sich im selben Subnetz 255.255.0.0 und sind daher

aus Sicht des Linux-Kernels gleichwertig. Dies führt dazu, dass die ARP-Requests der beiden Kameras immer nur von einer WLAN-Schnittstelle beantwortet werden. Infolgedessen werden alle Pakete an die gleiche Kamera weitergeleitet. Diese Verhalten ist natürlich nicht gewünscht. Der Linux-Kernel kann durch folgende Optionen dazu veranlasst werden, ARP-Requests von beiden Schnittstellen zu beantworten:

```
1 echo 1 > /proc/sys/net/ipv4/conf/wlan1/arp_ignore
2 echo 2 > /proc/sys/net/ipv4/conf/wlan1/rp_filter
3 echo 2 > /proc/sys/net/ipv4/conf/wlan1/arp_announce
4
5 echo 1 > /proc/sys/net/ipv4/conf/wlan2/arp_ignore
6 echo 2 > /proc/sys/net/ipv4/conf/wlan2/rp_filter
7 echo 2 > /proc/sys/net/ipv4/conf/wlan2/arp_announce
```

5 Stereo-Distanzmessung

In diesem Kapitel wird ein Verfahren und dessen Implementierung zur stereoskopischen Distanzmessung vorgestellt. Mit Hilfe der Stereoskopie können, ähnlich wie beim menschlichen Auge, durch die Auswertung von Bilddaten aus unterschiedlichen Blickwinkeln, Tiefeninformationen gewonnen werden. Mrovlje und Vrančić stellen in ihrem Paper „Distance measuring based on stereoscopic pictures“ (Jernej Mrovlje, Damir Vrančić, 2008) ein Verfahren vor, mit dem aus diesen Tiefeninformationen die Distanz eines Objektes zu den zwei aufnehmenden Kameras ermittelt werden kann.

5.1 Grundlagen des Verfahrens zur Distanzmessung

Das Verfahren beruht auf zwei baugleichen Kameras, die in einem definierten Abstand zueinander, parallel ausgerichtet, auf gleicher Höhe stehen. In dem Bereich, in dem sich die Sichtfelder der beiden Kameras überschneiden, kann eine Stereobildauswertung vorgenommen werden. Der horizontale Bildwinkel α_0, die Breite x_0 eines aufgenommenen Bildes in Pixeln und der Abstand B der Kameras zueinander sind bekannt. Wird jetzt noch der horizontale Abstand des zu untersuchenden Objektes auf dem linken und rechten Bild ermittelt (in Pixeln), sind alle Informationen vorhanden, um die Distanz zum Objekt zu ermitteln.

Die Distanz der B der beiden Kameras zueinander ist die Summe von B_1 und B_2 (siehe Abbildung 5.2). Unter Anwendung des Tangens ergibt sich:

$$B = B_1 + B_2 = D * \tan\alpha_1 + D * \tan\alpha_2 \tag{5.1}$$

$$B = D * (\tan\alpha_1 + \tan\alpha_2) \tag{5.2}$$

Die Formel wird nach D umgestellt. So erhält man die Distanz in Abhängigkeit vom Kamera-Abstand und den Winkeln α_1 und α_2:

$$D = \frac{B}{\tan\alpha_1 + \tan\alpha_2} \tag{5.3}$$

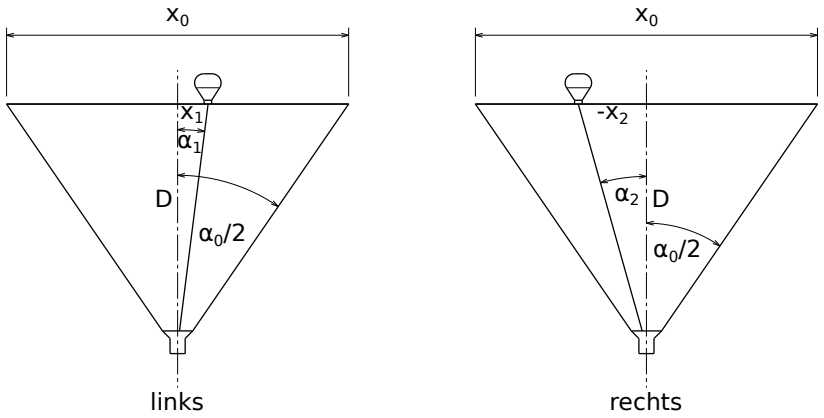

Abbildung 5.1: Winkel α_1 und α_2 sowie Strecken x_1 und $-x_2$

Die Werte von α_1 und α_2 sind nicht bekannt. Sie sollen in Abhängigkeit von der Bildbreite x_0 und des Bildwinkels α_0 angegeben werden:

$$\tan\frac{\alpha_0}{2} = \frac{\frac{x_0}{2}}{D} = \frac{x_0}{2D} \tag{5.4}$$

$$\tan\alpha_1 = \frac{x_1}{D} \tag{5.5}$$

Analog gilt für das rechte Bild:

$$\tan\alpha_2 = \frac{-x_2}{D} \tag{5.6}$$

Für die Winkel und Strecken ergeben sich folgende Verhältnisse:

$$\frac{\tan\alpha_1}{\tan\frac{\alpha_0}{2}} = \frac{\frac{x_1}{D}}{\frac{x_0}{2D}} = \frac{x_1}{\frac{x_0}{2}} \tag{5.7}$$

$$\frac{\tan\alpha_2}{\tan\frac{\alpha_0}{2}} = \frac{\frac{-x_2}{D}}{\frac{x_0}{2D}} = \frac{-x_2}{\frac{x_0}{2}} \tag{5.8}$$

Die Formeln werden umgestellt:

$$\tan\alpha_1 = \frac{x_1}{\frac{x_0}{2}} * \tan\frac{\alpha_0}{2} \tag{5.9}$$

$$\tan\alpha_2 = \frac{-x_2}{\frac{x_0}{2}} * \tan\frac{\alpha_0}{2} \tag{5.10}$$

Jetzt kann der Ausdruck $\tan\alpha_1 + \tan\alpha_2$ ersetzt werden:

$$\begin{aligned}\tan\alpha_1 + \tan\alpha_2 &= \frac{x_1}{\frac{x_0}{2}} * \tan\frac{\alpha_0}{2} + \frac{-x_2}{\frac{x_0}{2}} * \tan\frac{\alpha_0}{2} \\ &= \tan\frac{\alpha_0}{2}(\frac{x_1}{\frac{x_0}{2}} + \frac{-x_2}{\frac{x_0}{2}}) = 2\tan\frac{\alpha_0}{2}(\frac{x_1 - x_2}{x_0})\end{aligned} \tag{5.11}$$

Es wird in Gleichung 5.3 eingesetzt:

$$\begin{aligned}D = \frac{B}{\tan\alpha_1 + \tan\alpha_2} &= \frac{B}{2\tan\frac{\alpha_0}{2}(\frac{x_1 - x_2}{x_0})} \\ = \frac{B}{2\tan\frac{\alpha_0}{2}\frac{1}{x_0}(x_1 - x_2)} &= \frac{Bx_0}{2\tan\frac{\alpha_0}{2}(x_1 - x_2)}\end{aligned} \tag{5.12}$$

Jetzt kann die Distanz D zu einem Objekt in Abhängigkeit von dem Abstand B der Kameras, der Bildbreite x_0, dem horizontalen Bildwinkel α_0 und dem Abstand $(x_1 - x_2)$ des Objektes auf den beiden Bildern ermittelt werden.

Die Berechnung des Abstandes $(x_1 - x_2)$ wird in Abschnitt 5.2 gezeigt.

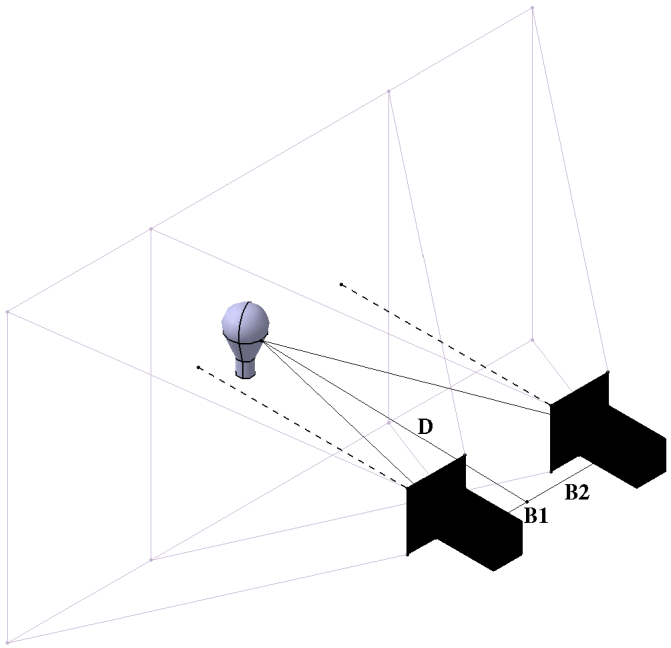

Abbildung 5.2: Kamera-Abstand B und Distanz D zum Objekt

5.1.1 Bildwinkel

Der horizontale Bildwinkel einer Kamera lässt sich nach folgender Formel berechnen:

$$\alpha_0 = 2\arctan(\frac{b}{2f}) \tag{5.13}$$

Dabei ist b die Breite des Aufnahmemediums und f die Brennweite. Das Aufnahmemedium ist im Falle der Digitalkamera der Bildsensor. Der Bildsensor der DSC-QX10 misst 6,17mm in der Breite, die Brennweite des Zoomobjektivs ist von 25mm bis 250mm angegeben. Hierbei handelt es sich allerdings um eine Äquivalenzangabe, die sich auf das 35mm-Kleinbildformat bezieht. Deshalb muss für b 35mm eingesetzt werden. Für die Zoom-Stufe null beträgt der Bildwinkel also:

$$\alpha_0 = 2\arctan(\frac{35mm}{2 * 25mm}) = 2\arctan(\frac{35}{50}) = 2\arctan(0.7) \approx 70° \tag{5.14}$$

Auf der höchsten Zoom-Stufe beträgt der Winkel nur noch:

$$\alpha_0 = 2\arctan(\frac{35mm}{2 * 250}) \approx 8° \tag{5.15}$$

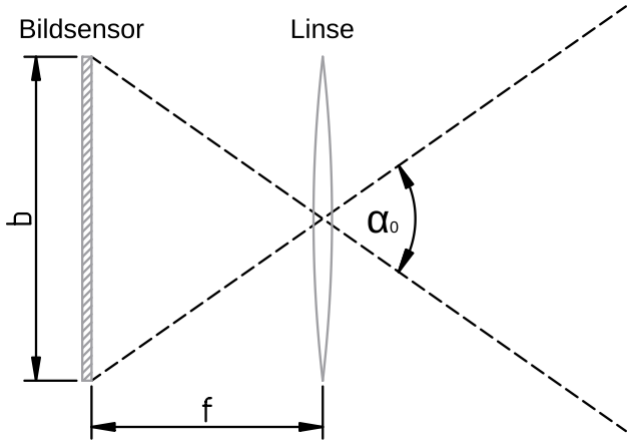

Abbildung 5.3: Zusammenhang zwischen Sensorbreite, Brennweite und Bildwinkel

5.2 Implementierung

Die Implementierung des Systems besteht aus zwei Teilen, dem CameraClient und dem Stereo-Server.

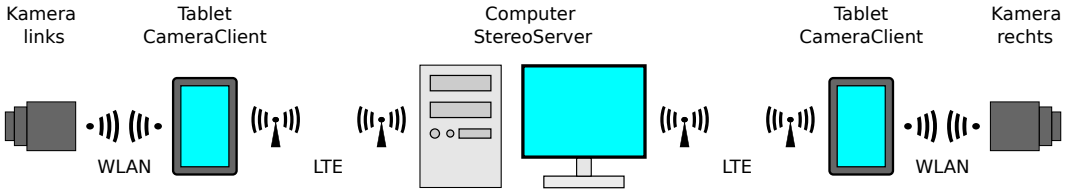

Abbildung 5.4: Aufbau des Systems

Der CameraClient läuft auf dem Tablet und stellt per WLAN die Verbindung mit der Kamera her. Er implementiert die Sony Camera Remote API und erlaubt dadurch die Steuerung der Kamerafunktionen, die Anzeige des Sucherbildes und das Ändern der Kamera-Einstellungen. Der CameraClient verbindet sich mit dem StereoServer und stellt ferngesteuert die auszuwertenden Bilder bereit.

Der StereoServer läuft auf dem Auswertungscomputer und implementiert den Algorithmus zur stereoskopischen Distanzmessung. Er stellt einen Socket-Server bereit, mit dem sich die CameraClients verbinden können. Dieser Aufbau wurde gewählt, da in Zukunft LTE für die Verbindung zwischen CameraClient und StereoServer zum Einsatz kommen soll. Im LTE-Netz werden zurzeit üblicherweise keine öffentlichen, sondern privtate IP-Adressen, unter Einsatz von NAT, vergeben. Diese privaten Adressen sind aus dem Internet nicht erreichbar. Da der Auswertungscomputer über einen kabelgebundenen Internetanschluss mit bekannter öffentlicher IP-Adresse verfügen kann und somit von den CameraClients direkt adressierbar ist, stellt er die Server-Dienste bereit.

Die CameraClients verbinden sich aktiv mit dem StereoServer und können dann von diesem ferngesteuert werden. Die Kommunikation erfolgt über ein einfaches Textprotokoll, das im nächsten Abschnitt erklärt wird. Der CameraClient wartet, bis sich die linke und die rechte Kamera verbunden haben und ist dann bereit, Bilder anzufordern und die Distanzmessung auszuführen.

Im Folgenden werden CameraClient und SteroServer näher beschrieben.

5.2.1 CameraClient

Der CameraClient ist eine Android-App. Android ist ein quelloffenes Betriebssystem für mobile Geräte wie zum Beispiel Tablets oder Smartphones. Es wird von der Firma Google entwickelt

und basiert auf dem Linux-Kernel. Apps können mit der objektorientierten Programmiersprache Java geschrieben werden. Dafür wird das Android SDK benötigt. Diese beinhaltet die API-Bibliotheken und die Entwicklungstools, die nötig sind, um Apps zu entwickeln und zu debuggen. Das SDK beinhaltet das ADT-Plugin (Adroid Developer Tools) für die integrierte Entwicklungsumgebung Eclipse. Das Plugin erweitert Eclipse dahingehend, dass Android-Projekte erstellt werden können und bietet einige Komfort-Funktionen.

User Interface

Das User Interface der App besteht aus mehreren Ansichten. In der Hauptansicht (MainActivity) wird auf dem Bildschirm das Sucherbild der Kamera (Liveview) dargestellt. Dieses wird so skaliert, dass es, je nachdem ob das Tablet hochkant oder quer gehalten wird, die volle Breite beziehungsweise Höhe des Bildschirms ausfüllt. Das Seitenverhältnis kann dabei entweder 4:3 oder 16:9 betragen und ist eine Einstellung der Kamera.

Am linken Bildschirmrand befindet sich eine halbtransparente Leiste, die drei Knöpfe und ein Textfeld beinhaltet:

Take Picture: Veranlasst die Kamera, ein Bild aufzunehmen und legt es im internen Speicher des Tablets ab.

Picture Metadata: Startet die ExifActivity, die dazu dient die Metainformationen des zuletzt aufgenommenen Bildes anzuzeigen.

Connect to Server / Disconnect from Server: Stellt die Verbindung zum StereoServer her oder trennt sie. Der aktuell dargestellte Text dient auch zur Anzeige des Verbindungsstatus. Beim Verbindungsauf- oder Abbau wird der Knopf zunächst grau. Dies dient als Aktivitätsanzeige. Bei erfolgreicher Verbindung wird der Text in „Disconnect from Server“ geändert. Der Knopf ändert dann sein Verhalten und dient dazu, die Verbindung zu trennen.

Textfeld: Das Textfeld dient zur Anzeige von Statusinformationen:

- **Camera Status:** Zeigt den aktuellen Status der Kamera, wie zum Beispiel „IDLE“ oder „StillCapturing“ an.
- **Focus Status:** Zeigt an, ob die Kamera gerade fokussiert ist, oder ob die Fokussierung fehlgeschlagen ist.
- **Zoom Position:** Zeigt die aktuelle Zoom-Position in Prozent an.

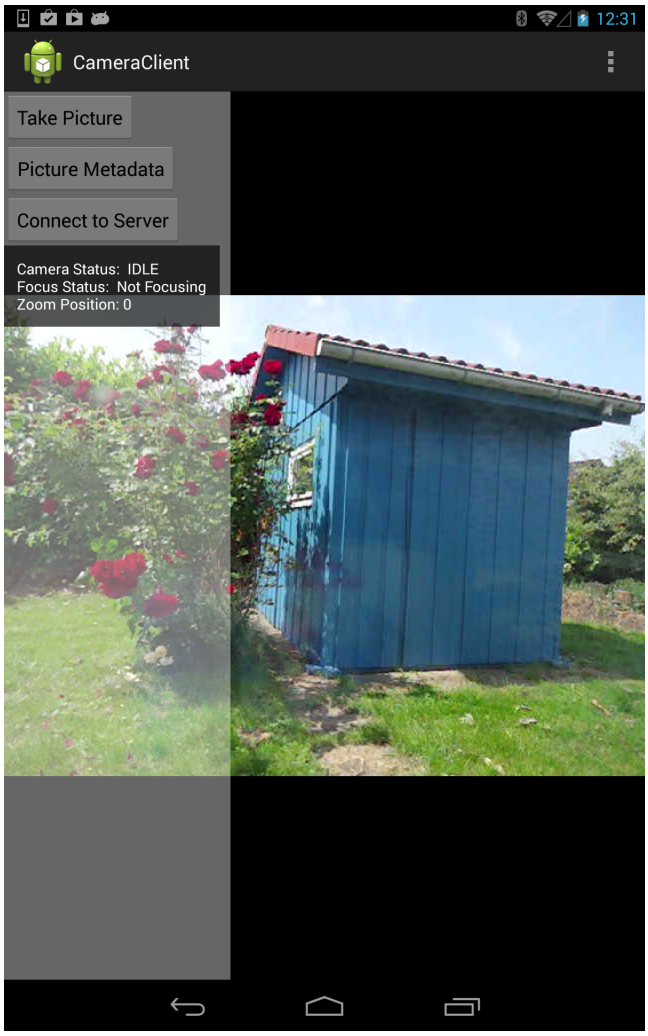

Abbildung 5.5: Hauptansicht MainActivity

Die Metadatenansicht ExifActivity stellt die Bildgröße in Bytes, die Bildauflösung sowie Zeitstempel, Blendenzahl und Belichtungszeit des zuletzt aufgenommenen Bildes in einem Textfeld dar. Ein Druck auf den „Zurück-Pfeil“ führt wieder zur Hauptansicht.

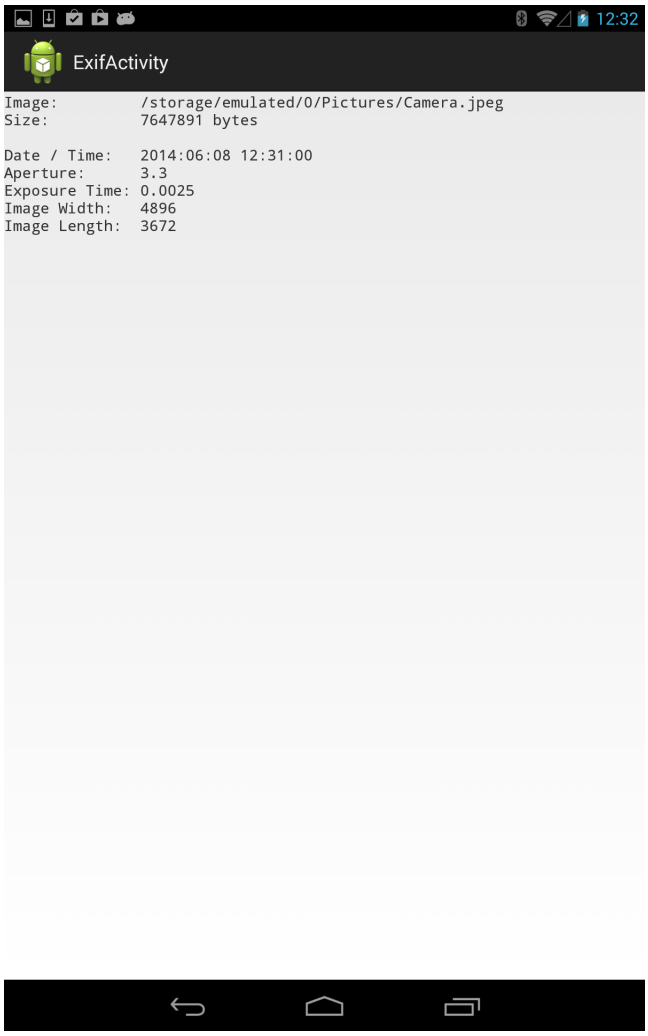

Abbildung 5.6: Metadatenansicht ExifActivity

Die Einstellungsansicht SettingsActivity dient zur Darstellung und Änderung von App- und Kamera-Einstellungen. Sie ist unterteilt in zwei Subansichten:

Network Settings: Einstellungen, die die Netzwerkverbindungen betreffen.

Camera Position: Position der Kamera im Stereo-System (Left oder Right).

Camera IP Address: IP-Adresse der Sony DSC-QX10 Kamera.

Camera Port: Port der Kamera.

Server IP Address: IP-Adresse des StereoServers.

Server Port: Port des StereoServers.

Camera Settings: Einstellungen, die das Verhalten der Kamera betreffen.

Still Size: Legt das Seitenverhältnis und die Auflösung des aufgenommenen Bildes fest.

Postview Image Size: Legt fest, ob das aufgenommene Bild in der Originalgröße oder als verkleinerte Zwei-Megapixel-Version von der Kamera heruntergeladen werden soll.

Exposure Mode: Legt den Belichtungsmodus fest.

Exposure Compensation: Wert der Belichtungskorrektur zwischen -2.0 und +2.0. Nur auswählbar, wenn Exposure Mode auf „Program Auto“ gesetzt ist.

ISO Speed Rate: Wert der ISO-Empfindlichkeit von 100 bis 3200 oder Auto. Nur auswählbar, wenn Exposure Mode auf „Program Auto“ gesetzt ist.

White Balance: Weißabgleichs-Modus. Nur auswählbar, wenn Exposure Mode auf „Program Auto“ gesetzt ist.

Beep Mode: Legt fest, ob die Kamera beim Auslösen ein Piep- und/oder Blenden-Geräusch ausgeben soll.

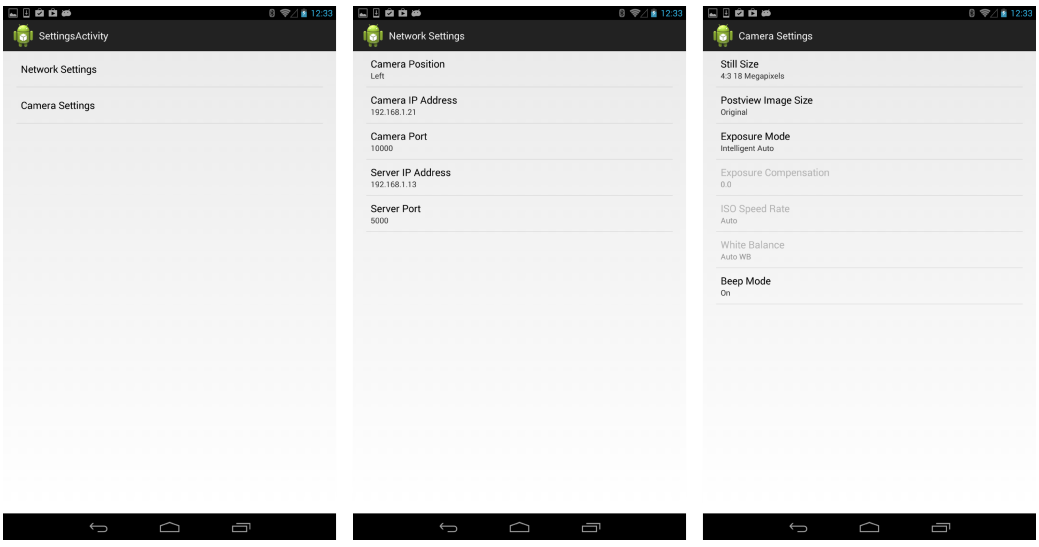

Abbildung 5.7: Einstellungsansicht SettingsActivity

Dokumentation

Es folgt eine Beschreibung der Klassen der CameraClient-App:

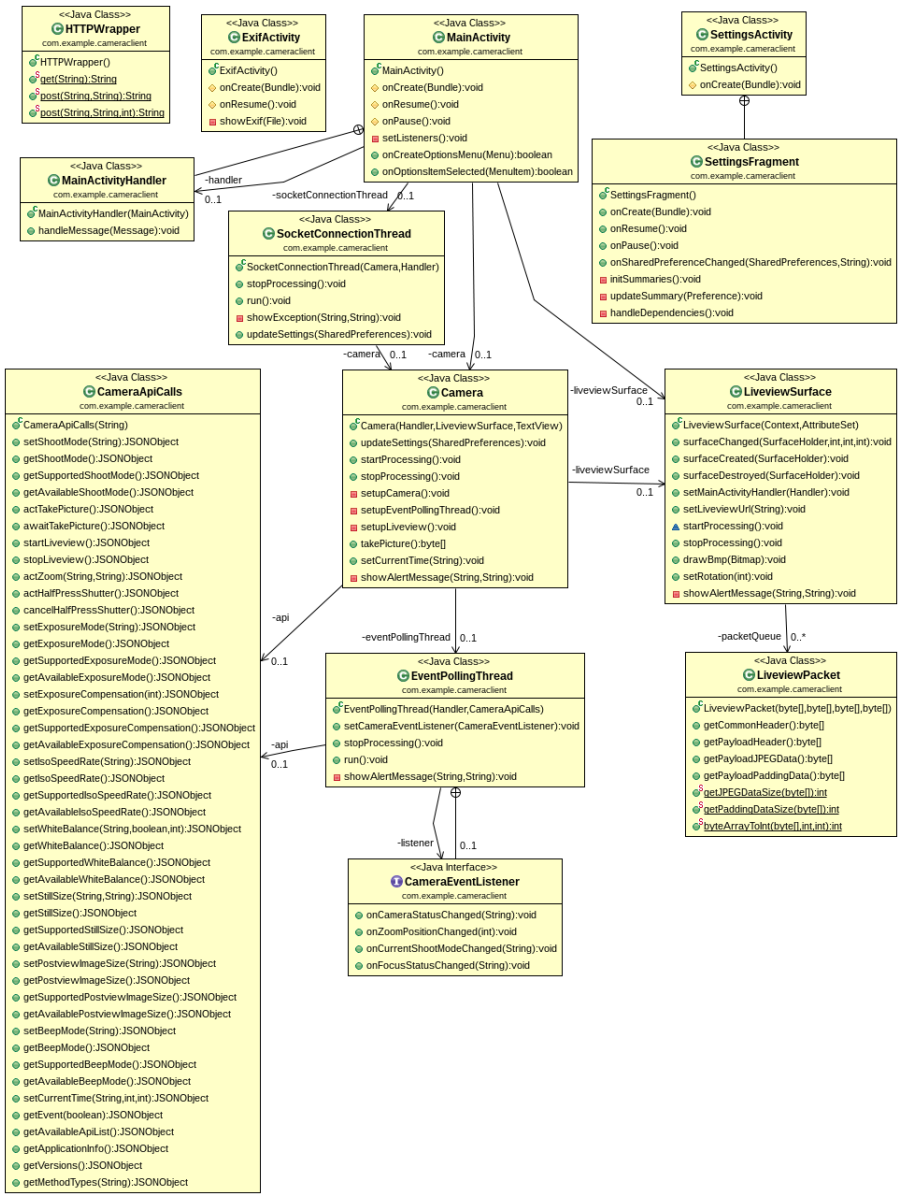

Abbildung 5.8: UML-Klassendiagramm

MainActivity: Eine Activity ist in Android eine Klasse, die ein Fenster bereitstellt, in dem das User-Interface dargestellt werden kann. Die MainActivity ist der Einsprungspunkt

in die App. Sie regelt den Lebenszyklus des Programms. Dafür stehen verschiedene Callback-Methoden zur Verfügung. OnCreate wird aufgerufen, wenn die App gestartet wird und initialisiert das User-Interface. Hier werden auch die Objekte initialisiert, die im späteren Verlauf benötigt werden, wie zum Beispiel Camera und LiveviewSurface. Wenn die App bereit ist mit dem Benutzer zu interagieren, wird OnResume aufgerufen. Hier werden alle Threads gestartet und die Listener registriert, die auf Tastendrücke reagieren. In onResume wird auch die Kamera initialisiert. OnPause wird aufgerufen, wenn die App in den Hintergrund gerät oder beendet wird. Hier werden alle Threads gestoppt.

ExifActivity: Ein JPEG-Bild, wie es von der Kamera geliefert wird, enthält in seinem Header Metadaten. Diese sind im Exif-Format (Exchangeable Image File Format) gespeichert. Die ExifActivity extrahiert die Metadaten und stellt sie dar. Angezeigt werden Größe, Auflösung, Belichtungszeit, Blendenzahl und Zeitstempel. Es werden immer die Informationen des zuletzt aufgenommenen Bildes dargestellt.

SettingsActivity: Die SettingsActivity verwaltet die Einstellungen der App und der Kamera. Die Einstellungen werden in einer Android-App in einem SharedPreferences-Objekt gespeichert.

HttpWrapper: Der HttpWrapper stellt die statischen Methoden get und post zur Verfügung. Diese entsprechen den HTTP-Zugriffsmethoden GET und POST. Post wird bei jedem API-Aufruf verwendet, um eine entfernte Methode der Kamera aufzurufen. Get kann benutzt werden, um Daten von der Kamera herunterzuladen.

CameraApiCalls: CameraApiCalls implementiert die wichtigsten Aufrufe der Sony Camera Remote API. Jedem Aufruf entspricht eine Methode. Für jeden Aufruf wird ein JSON-Objekt erzeugt, das per HttpWrapper.post an die Kamera übertragen wird. Die Methoden geben die Antwort der Kamera als JSONObject zurück.

Camera: Die Klasse Camera verwaltet die Kamera. Sie entnimmt die gewünschten Einstellungen aus den SharedPreferences und leitet sie durch API-Aufrufe an die Kamera weiter. Camera startet den EventPollingThread und die Ausgabe des Sucherbildes (Liveview). Wenn die Kamera einen API-Aufruf mit einer URL beantwortet (zum Beispiel startLiveview), enthält diese die IP-Adresse 10.0.0.1. Diese wird automatisch durch 192.168.1.21 beziehungsweise 192.168.1.22 ersetzt (siehe Kapitel 4). Camera stellt die Methoden takePicture und setCurrentTime zur Verfügung, um ein Bild aufzunehmen oder die interne Uhr der Kamera zu stellen.

EventPollingThread: Der EventPollingThread dient zur Anzeige von Statusinformationen der Kamera in der MainActivity. Er verwendet dazu den API-Aufruf getEvent. GetEvent blockiert solange, bis eine Änderung des Kamerastatus auftritt. Der EventPollingThread veranlasst dann mit Hilfe von Callback-Methoden, dass der geänderte Status in der Hauptansicht sichtbar wird und ruft wieder getEvent auf. Es werden der allgemeine Kamera-Status, der Fokus-Status und die momentane Zoom-Position ausgewertet.

LiveviewSurface: Das Sucherbild ist ein Datenstrom, der per HTTP-GET von der Kamera heruntergeladen wird. Der Aufbau dieses Stromes ist in Abbildung 5.9 dargestellt.

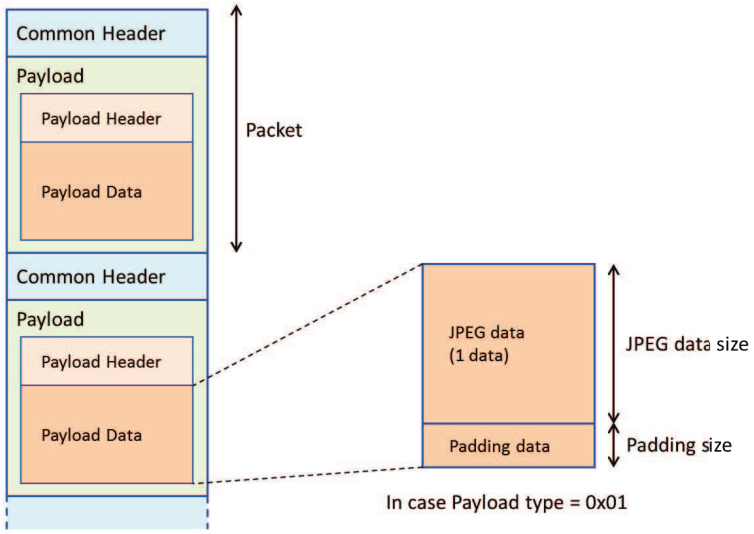

Abbildung 5.9: Format des Liveview-Streams
Quelle: (Sony Corporation, 2014a)[S. 112, Abb. 1]

LiveviewSurface beinhaltet zwei Threads, den getLifeviewDataThread und den drawBmpThread. Der getLifeviewDataThread lädt den Liveview-Stream herunter, zerlegt ihn in einzelne Pakete und verpackt diese in ein LiveviewPacket. Das LiveviewPacket wird einer Queue hinzugefügt, aus der es der drawBmpThread entnimmt. Der drawBitmapThread entnimmt dem LiveviewPacket die JPEG-Daten, konvertiert diese ins Bitmap-Format und zeichnet sie auf den Bildschirm.

LiveviewPacket: Die Klasse LiveviewPacket dient als Daten-Container für ein Paket aus dem Liveview-Datenstrom. Es erlaubt den Zugriff auf die einzelnen Bestandteile des Paketes.

SocketConnectionThread: Der SocketConnectionThread stellt die Verbindung zum Stereo-Server her. Über ein einfaches Textprotokoll nimmt er Anfragen des Servers entgegen und leitet diese an die Kamera weiter. Dazu wird ihm im Konstruktor eine Referenz auf das Kamera-Objekt übergeben.

Die Dokumentation des Quellcodes wurde mit Javadoc erstellt.

5.2.2 StereoServer

Der StereoServer ist eine Linux-Anwendung, die in der Programmiersprache C geschrieben wurde. Für die Auswertung der Bilddaten kommt OpenCV zum Einsatz. OpenCV ist eine quelloffene, plattformunabhängige Bibliothek zur Echtzeitverarbeitung von Bilddaten. Durch die Plattformunabhängigkeit von OpenCV kann der StereoServer relativ leicht auf andere Betriebssysteme als Linux portiert werden. Es muss dann im Wesentlichen der Netzwerkcode angepasst werden.

User Interface

Der StereoServer wird zunächst per Kommandozeile gesteuert. Nach dem Start wird auf Verbindungen der CameraClients gewartet:

```
Waiting for CameraClients...
```

Wenn sich beide CameraClients verbunden haben, stehen folgende Möglichkeiten zur Verfügung:

```
Incoming Connection from 192.168.1.18, Port 44371
Left CameraClient connected
Incoming Connection from 192.168.1.11, Port 55230
Right CameraClient connected
Both CameraClients are connected.

Enter t to take pictures.
Enter a to analyze pictures.
Enter s to synchronize clocks.
Enter e to exit.
```

t: Take pictures: Veranlasst die CameraClients Bilder aufzunehmen, lädt diese herunter und speichert sie zur späteren Verarbeitung auf der Festplatte.

a: Analyze Pictures: Startet die Stereo-Distanzmessung anhand zweier zuvor gespeicherter Bilder.

s: Synchronize clocks: Veranlasst die CameraClients, die internen Uhren der Kameras mit der aktuellen Uhrzeit des StereoServers zu synchronisieren.

e: Exit. Beendet den StereoServer.

Durch die Eingabe von „a“ wird die Stereo-Distanzmessung gestartet. Die Bedienung erfolgt ab hier grafisch mit der Maus. Im ersten Schritt wird das Vorschaufenster geöffnet. Es zeigt eine herunterskalierte Version des linken Kamerabildes an.

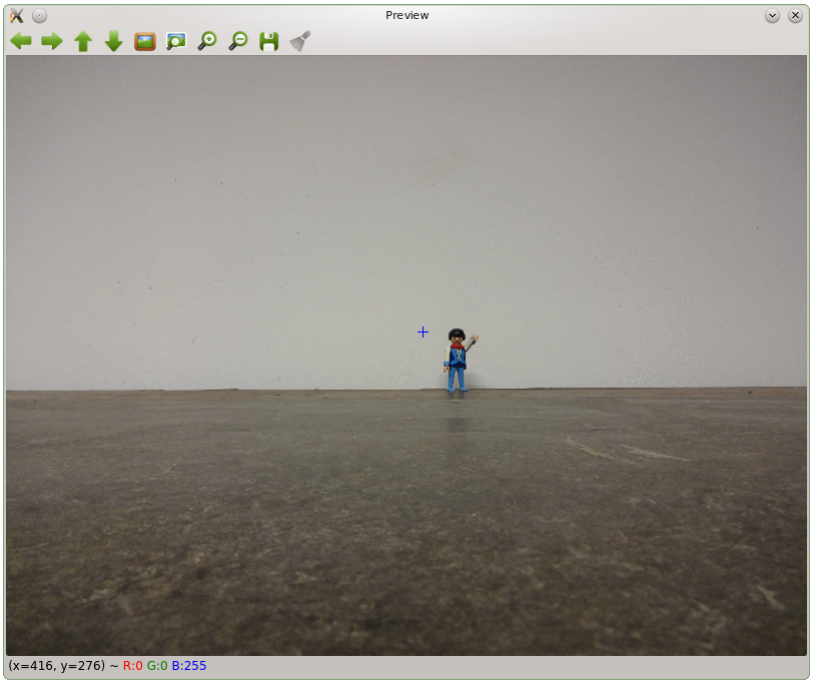

Abbildung 5.10: Vorschaufenster mit Fadenkreuz

Mit der Maus wird in die Nähe des zu untersuchenden Objekts geklickt. Dadurch wird ein Ausschnitt definierter Größe um den angeklickten Punkt gewählt. Das Vorschaufenster schließt sich, und der gewählte Ausschnitt des linken Bildes wird in Originalauflösung dargestellt. Jetzt muss mit der Maus ein Rechteck um das zu untersuchende Objekt gezogen werden.

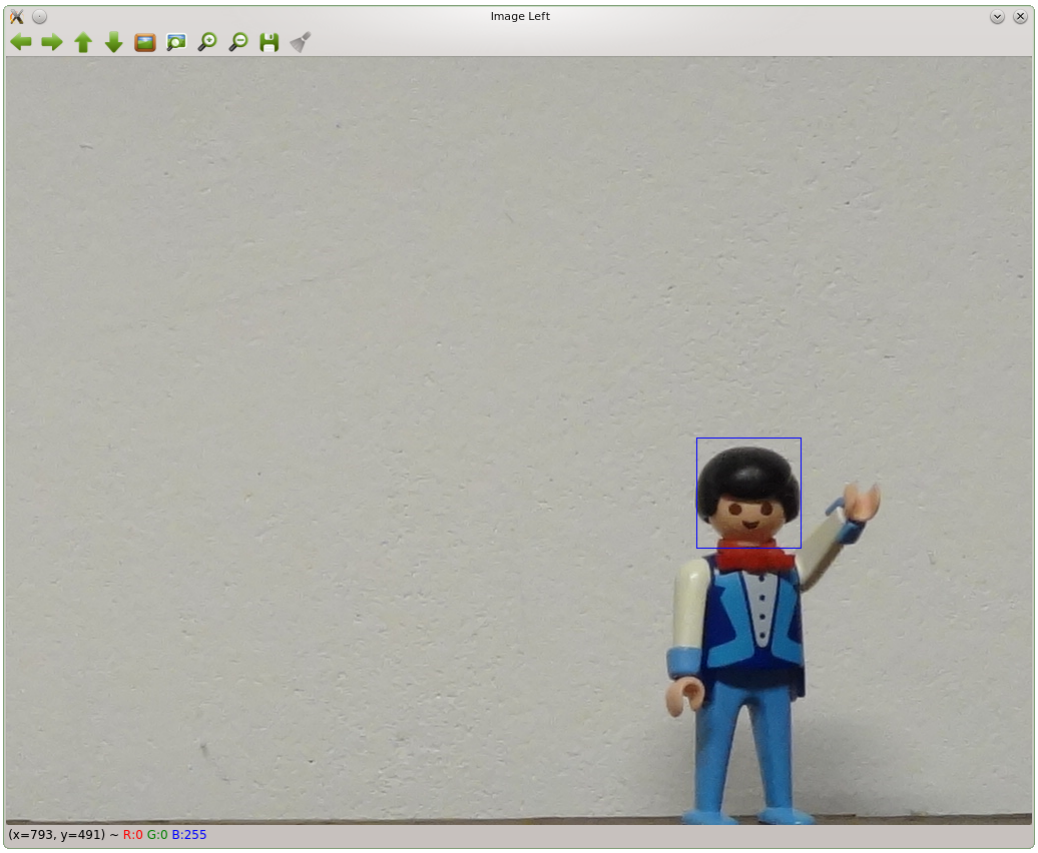

Abbildung 5.11: Linkes Bild mit ausgewähltem Objekt

Der StereoServer versucht jetzt, das auf dem linken Bild ausgewählte Objekt auf dem rechten Bild wiederzufinden und markiert es mit einem Rechteck.

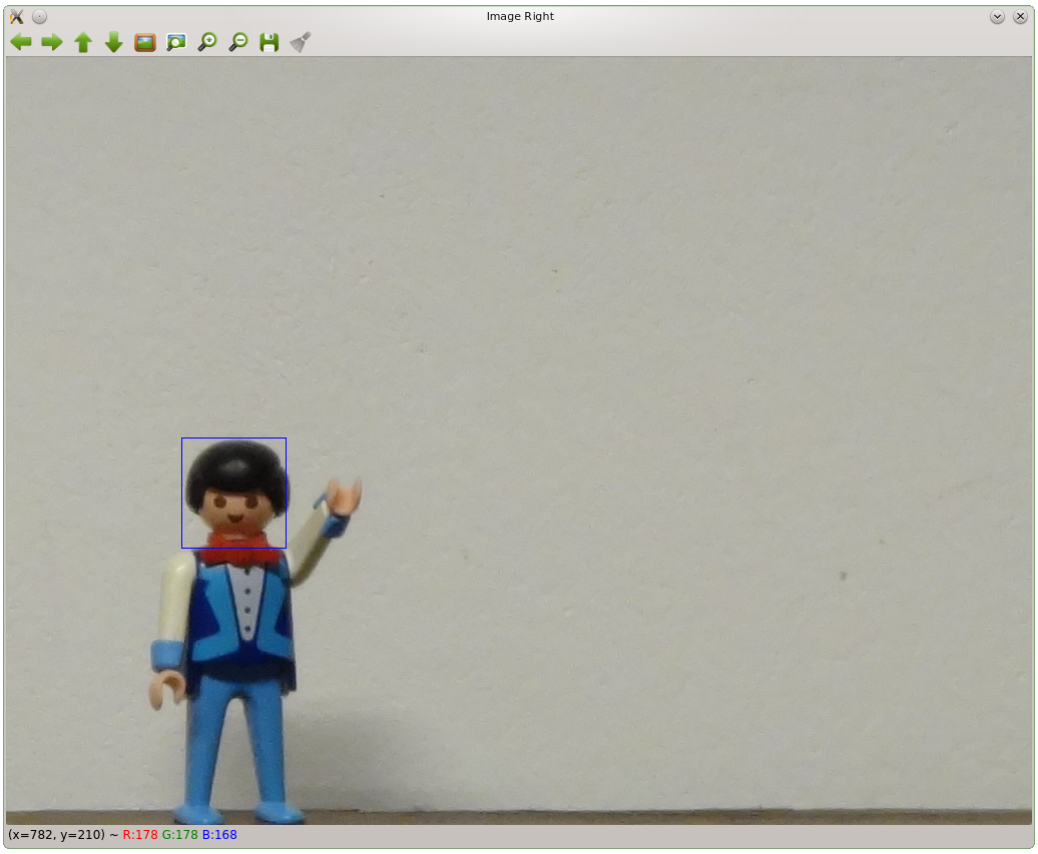

Abbildung 5.12: Rechtes Bild mit erkanntem Objekt

Da nun die Position des Objekts auf beiden Bildern bekannt ist, kann dessen horizontaler Abstand zwischen dem linken und rechten Bild ermittelt werden. Jetzt sind alle Werte bekannt, die benötigt werden um mit Hilfe der Gleichung 5.12 den Abstand zwischen der gedachten Verbindungslinie der beiden Kameras und dem Objekt auszurechnen. Das Ergebnis wird auf der Konsole ausgegeben.

```
1 Starting stereo distance measurement.
2 Select area of interest from the preview image.
3 Select the object to be analyzed from the left image.
4 Distance to object: 675 mm
```

Dokumentation

Der Quellcode des StereoServers ist in drei C-Module aufgeteilt:

Main-Modul: Enthält die Hauptschleife und stellt Funktionen bereit, um CameraClient-Verbindungen zu verwalten, Bilder anzufordern und die Uhren zu synchronisieren.

Server-Modul: Enthält Funktionen zum Auf- und Abbau der Socket-Verbindungen und zum Senden und Empfangen von Daten über diese.

Stereo-Modul: Beinhaltet die Funktionen für die Stereo-Distanzmessung.

Die Dokumentation des Quellcodes wurde mit Doxygen erstellt.

Programmablauf In der Hauptschleife des Programms, wie in Abbildung 5.16 dargestellt, werden die Netzwerkverbindungen verwaltet sowie Tastatureingaben verarbeitet. Um dies ohne die Verwendung von Threads zu erreichen, wird IO-Multiplexing mit Hilfe der POSIX-Funktion select verwendet. Die Socket-Deskriptoren des Servers und der beiden CameraClients sowie der File-Descriptor der Standardeingabe werden einem FD_SET hinzugefügt. Select blockiert solange, bis auf einem der Deskriptoren ein Ereignis auftritt. Anschließend wird auf das Ereignis reagiert, und die Schleife beginnt von Neuem.

Ein Ereignis auf dem Server-Socket kündigt eine eingehende Verbindung eines Clients an. Ein Client muss sich mit seiner ID (LEFT oder RIGHT) identifizieren. Es kann jeweils nur eine Verbindung für jede Kameraseite aufgebaut werden. Die Client-Socket-Deskriptoren werden überwacht, um auf einen Verbindungsabbruch zu reagieren. Durch die Überwachung der Standardeingabe können Tastatureingaben verarbeitet werden.

Algorithmus zum Objektvergleich Um die Formel 5.12 zur Distanzbestimmung anwenden zu können, muss der horizontale Abstand $(x_1 - x_2)$ zwischen den zu untersuchenden Objekten auf dem linken und rechten Kamerabild ermittelt werden. Der Algorithmus sieht vor, dass das zu untersuchende Objekt im linken Bild manuell selektiert wird. Das ensprechende Objekt im rechten Bild wird dann automatisch gesucht. Durch Subtraktion der rechten x-Koordinate von der linken x-Koordinate erhält man dann den gesuchten Abstand.

Man beginnt damit, dass man im linken Bild ein Rechteck um das zu suchende Objekt zieht. Anschließend wird für den gewählten Bereich das Histogramm über alle drei Farbkanäle gebildet und zwischengespeichert. Im nächsten Schritt wird das rechte Bild betrachtet, und es wird ein Array mit der Größe der Bildbreite erstellt. Die Box aus dem linken Bild wird nun auf das rechte Bild übertragen. Da die Kameras sich auf gleicher Höhe befinden, wird die y-Koordinate beibehalten. In einer Schleife wird jetzt die Box, beginnend bei x=0, pixelweise nach rechts durchs Bild geschoben. Dabei wird jedes Mal das Histogramm für die aktuelle Box

berechnet. Mit Hilfe der OpenCV-Funktion cvCompareHist wird die Abweichung zum zuvor gespeicherten Histogramm der linken Box ermittelt und in das Array eingetragen.

Die Box, deren Histogramm am wenigsten vom gesuchten Histogramm abweicht, umschließt das gesuchte Objekt, deshalb wird im letzten Schritt das Minimum des Arrays ermittelt, um die Box mit der größtmöglichen Übereinstimmung zu finden. Der Index des kleinsten Elements des Arrays ist die x_2-Koordinate.

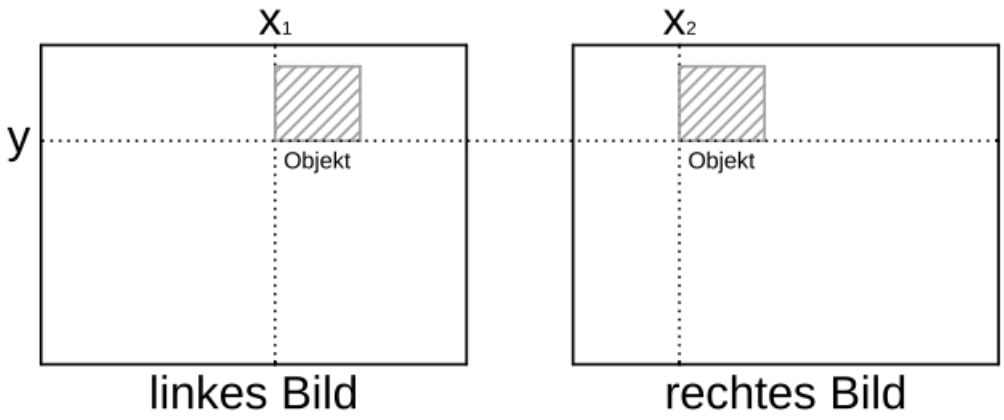

Abbildung 5.13: Objekt auf linkem und rechtem Bild

Netzwerk-Protokoll Der StereoServer übermittelt Befehle an die CameraClients. Dazu steht ihm ein textbasiertes Protokoll zur Verfügung. Jeder Befehl wird mit einem Zeilenvorschub („\n“) abgeschlossen. Eine beispielhafte Bildanforderung ist in Diagramm 5.14 dargestellt. Diagramm 5.15 zeigt den Zeitabgleich. Es folgt eine Beschreibung der Kommandos.

SEND POSITION: Ein CameraClient, der sich mit dem StereoServer verbindet, wird durch dieses Kommando angewiesen, seine Position im Stereo-System zu übermitteln. Gültige Antworten sind LEFT und RIGHT.

TAKE PICTURE: Veranlasst den CameraClient, ein Bild aufzunehmen und von der Kamera herunterzuladen. Es wird keine Antwort erwartet.

IS PICTURE READY: Der StereoServer pollt den CameraClient solange, bis dieser mit der Antwort PICTURE READY signalisiert, dass er das Bild von der Kamera heruntergeladen hat. Ist dies noch nicht der Fall, lautet die Antwort PICTURE NOT READY. Eine positive

Rückmeldung bedeutet, dass das Kamerabild jetzt vom CameraClient heruntergeladen werden kann.

SEND PICTURE SIZE: Damit der StereoServer Speicher für das herunterzuladende Bild reservieren kann, fordert er die Bildgröße an. Der CameraClient antwortet mit einem Integer-Wert mit der Größe des Bildes in Bytes.

SEND PICTURE: Veranlasst den CameraClient, die Bilddaten an den Server zu schicken. Der Server lädt die Menge an Bytes herunter, die er zuvor durch SEND PICTURE SIZE ermittelt hat.

SET TIME: Der StereoServer sendet seine aktuelle Systemzeit an den CameraClient. Dieser synchronisiert mit Hilfe von API-Aufrufen die internen Uhren der Kameras.

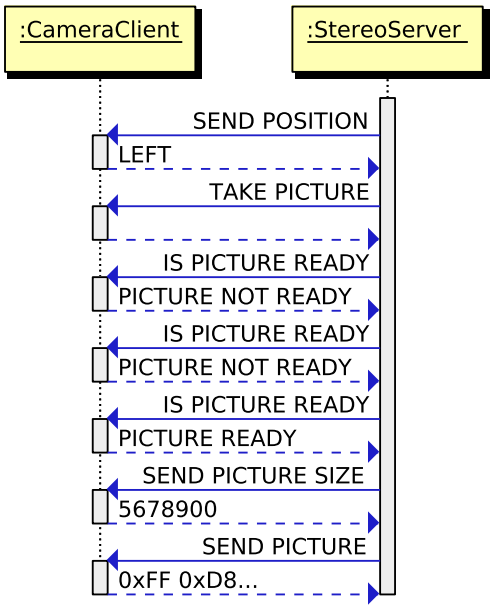

Abbildung 5.14: Ablauf der Bildanforderung

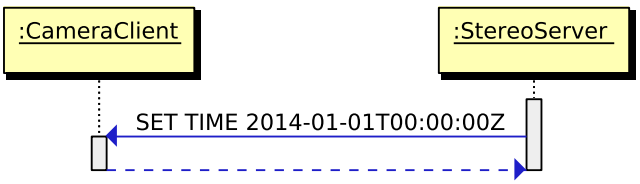

Abbildung 5.15: Ablauf des Zeitabgleichs

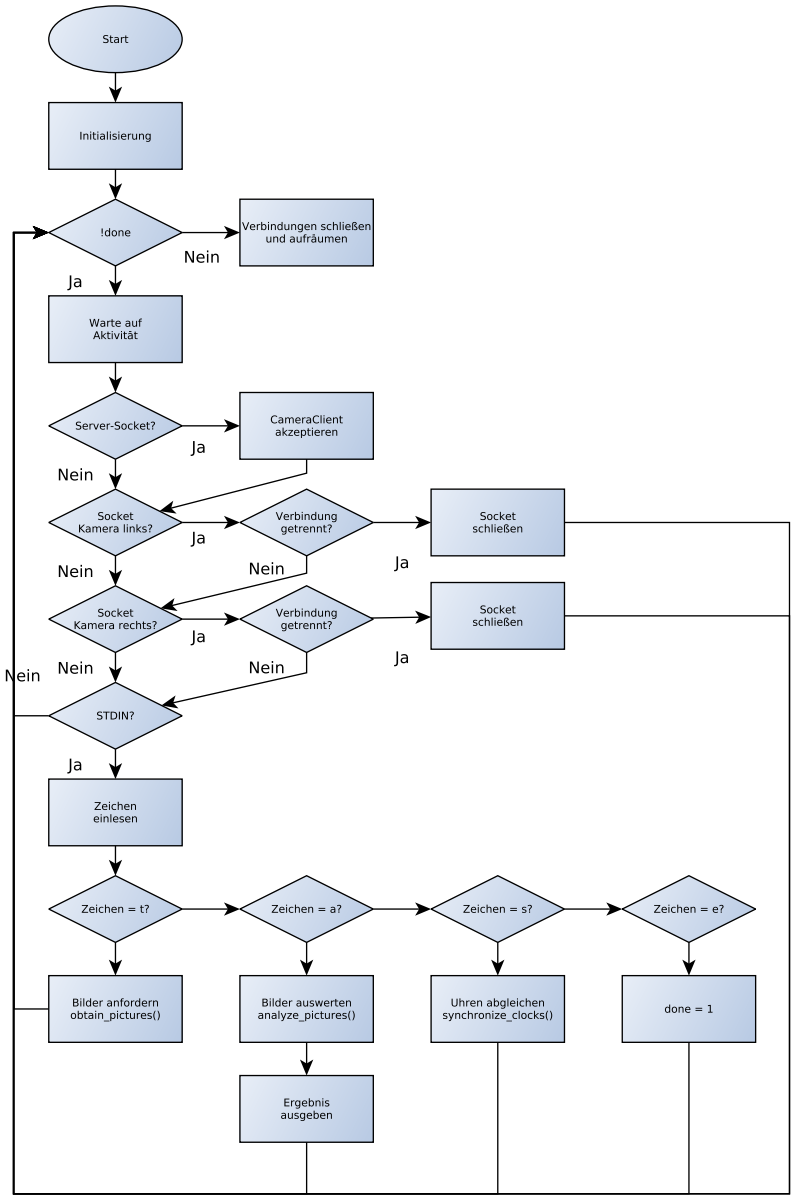

Abbildung 5.16: Grober Programmablauf des StereoServers

5.3 Zeitbetrachtung

Wenn bewegte Objekte in einem Stereo-System erfasst werden sollen, ist es wichtig, dass die beiden Bilder möglichst zeitgleich aufgenommen werden. Hier soll untersucht werden, welche Genauigkeit mit dem vorgestellten System erzielbar ist.

Dazu wird ein binärer 16-Bit LED-Zähler verwendet. Dieser wird mit fester Frequenz inkrementiert und so platziert, dass er im Sichtfeld beider Kameras liegt. Mit Hilfe des StereoServers werden Bilder aufgenommen. Aus der Differenz der abfotografierten Zählerstände im linken und rechten Bild kann der zeitliche Versatz zwischen den Aufnahmen bestimmt werden.

5.3.1 Zähler

Der Zähler ist auf Basis eines 8-Bit `Atmel AVR ATmega644P` Microcontrollers aufgebaut. Mit Hilfe zweier Schieberegister werden die 16 LEDs angesteuert. Es handelt sich dabei um Low-Current-LEDs, da die verwendeten Schieberegister `74HC595` nur 70mA treiben können. Durch Vorwiderstände wird der Strom pro LED auf ca. 8mA begrenzt. Die LEDs sind bei diesem Strom bereits gut ablesbar.

Die Zählfrequenz wird durch einen Timer-Interrupt generiert. In der ISR wird eine 16-Bit Zählervariable inkrementiert und in die Schieberegister geschoben.

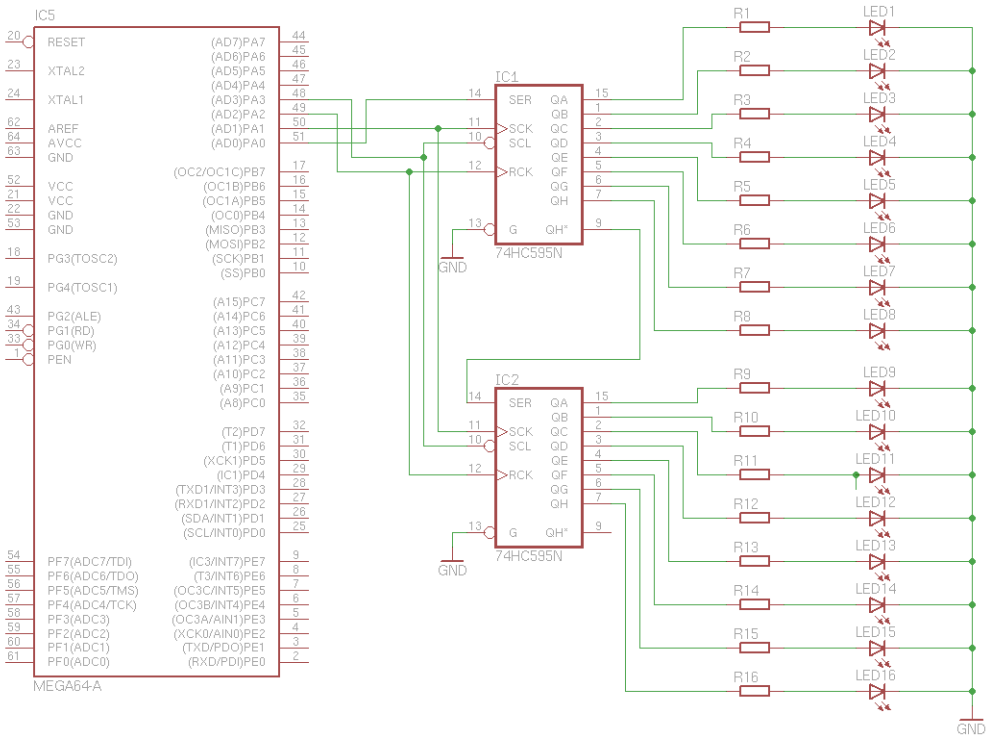

Abbildung 5.17: Vereinfachter Schaltplan des Zählers

5.3.2 Auswertung

Tests haben ergeben, dass die Belichtungszeit bei den verwendeten Kamera-Einstellungen und Lichtverhältnissen immer 1/30s beträgt. Der Zähler wird mit einer Frequenz von 20Hz betrieben. So kann die Zeitabweichung auf 50ms genau bestimmt werden. Dabei kann es zu maximal einem Wechsel des Zählerstands während der Belichtungszeit kommen. Dies stellt eigentlich ein Problem dar, jedoch kann man auf Grund der unterschiedlichen Belichtungen der LEDs einen solchen Fall erkennen und die Auswertung dennoch vornehmen. Hierbei lässt sich der Zählerstand nicht eindeutig bestimmen, sondern es kommen zwei Werte in Frage. Für die Auswertung wird immer von der längsten möglichen Verzögerung ausgegangen.

Um zu untersuchen, ob der Belichtungsmodus der Kamera eine Auswirkung auf die Zeitverzögerung hat, werden Testreihen für die drei möglichen Belichtungsmodi „Intelligent Auto“, „Superior Auto“ und „Program Auto“ durchgeführt. Pro Modus werden 10 Auswertungen durchgeführt, deren Ergebnisse in den Tabellen 5.1, 5.2 und 5.3 aufgeführt sind. Die Zählerstände beziehen sich immer nur auf die niederwertigsten, sich unterscheidenden Bits, da die höherwertigen Bits sich nicht unterscheiden.

Die Abweichungen folgen keinem festen Muster und sind sowohl positiv als auch negativ. Eine konstante Abweichung könnte kompensiert werden, indem die Auslösung einer der beiden Kameras verzögert würde. Die Auswertung der Testreihen hat ergeben, dass der Belichtungsmodus „Superior Auto“ im Mittel die geringste Abweichung aufweist. Dazu wurden die Beträge der Abweichungen addiert und jeweils der Mittelwert gebildet.

- „Intelligent Auto“: $\frac{650\,\text{ms}}{10} = 65\,\text{ms}$
- „Superior Auto“: $\frac{400\,\text{ms}}{10} = 40\,\text{ms}$
- „Program Auto“: $\frac{650\,\text{ms}}{10} = 65\,\text{ms}$

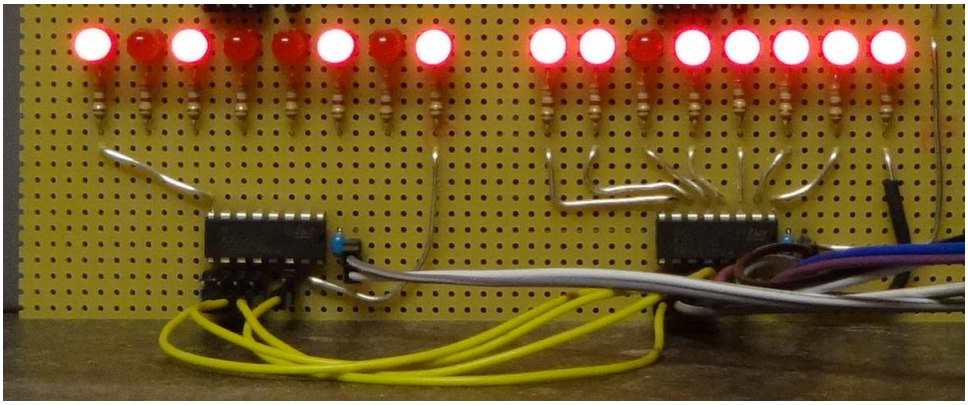

Abbildung 5.18: LED-Zähler in Betrieb

Auswertung	Zählerstand linkes Bild	Zählerstand rechtes Bild	Differenz
1	5	3	100ms
2	3	1	100ms
3	1	2	-50ms
4	3	3	0ms
5	7	8	-50ms
6	4	7	-150ms
7	7	7	0ms
8	8	7	50ms
9	3	1	100ms
10	1	0	50ms

Tabelle 5.1: Auswertung für den Belichtungsmodus „Intelligent Auto“

Auswertung	Zählerstand linkes Bild	Zählerstand rechtes Bild	Differenz
1	7	8	-50ms
2	2	3	-50ms
3	3	3	0ms
4	15	16	-50ms
5	3	2	50ms
6	15	16	-50ms
7	3	4	-50ms
8	6	7	-50ms
9	1	1	0ms
10	1	0	50ms

Tabelle 5.2: Auswertung für den Belichtungsmodus „Superior Auto“

Auswertung	Zählerstand linkes Bild	Zählerstand rechtes Bild	Differenz
1	1	2	-50ms
2	0	1	-50ms
3	0	3	-150ms
4	1	2	-50ms
5	1	3	-100ms
6	6	8	-100ms
7	3	2	50ms
8	3	3	0ms
9	6	7	-50ms
10	1	2	-50ms

Tabelle 5.3: Auswertung für den Belichtungsmodus „Program Auto“

5.3.3 Beispielrechnung

Anhand eines konkreten Beispiels soll überprüft werden, welche Abweichung eine Auslösungsverzögerung von 150ms bei der Distanzmessung zur Folge hat. 150ms entsprechen der maximalen Verzögerung, die sich aus der obigen Auswertung ergeben hat.

Es wird ein Stereosystem betrachtet, bei dem die beiden Kameras mit 100m Abstand auf zwei Hochhäusern installiert sind. Die Kameras sind nach Süden hin ausgerichtet. In einer Entfernung von 150m bewegt sich eine Drohne mit $50\ \frac{\mathrm{km}}{\mathrm{h}}$ von Westen nach Osten südlich an den Hochhäusern vorbei.

In 150ms legt die Drohne folgende Distanz zurück:

$$d_0 = 50\,\frac{\text{km}}{\text{h}} * 150\,\text{ms} \approx 2.08\,\text{m} \tag{5.16}$$

Anhand des Bildwinkels der Kamera von 70° kann die Breite des Sichtfeldes in 150m Entfernung bestimmt werden.

$$2 * (\tan\frac{\alpha_0}{2} * 150\,\text{m}) = 2 * (\tan 35° * 150\,\text{m}) = 210.06\,\text{m} \tag{5.17}$$

Da die Breite des Sichtfeldes jetzt bekannt ist, kann mit Hilfe der Bildbreite $x_0 = 4896px$ ausgerechnet werden, wievielen Pixeln die Entfernung von 2,08m entspricht:

$$\begin{aligned} 210.06\,\text{m} &\hat{=} 4896px \\ 2.08\,\text{m} &\hat{=} 48,48px \end{aligned} \tag{5.18}$$

Als nächstes wird $(x_1 - x_2)$ ausgerechnet. Dafür wird die Formel 5.12 umgestellt:

$$(x_1 - x_2) = \frac{Bx_0}{2\tan\frac{\alpha_0}{2}D} = \frac{100\,\text{m} * 4896px}{2\tan 35° * 150\,\text{m}} = 2330,74px \tag{5.19}$$

Jetzt wird die Abweichung von 48,48px, die in den 150ms entsteht, zu $(x_1 - x_2)$ addiert und in die Gleichung 5.12 eingesetzt:

$$\begin{aligned} D = \frac{Bx_0}{2\tan\frac{\alpha_0}{2}(x_1 - x_2)} &= \frac{100\,\text{m} * 4896px}{2\tan 35°((x_1 - x_2) + 48,48px)} \\ &= \frac{100\,\text{m} * 4896px}{2\tan 35° * 2379,22px} = 146.94\,\text{m} \end{aligned} \tag{5.20}$$

Die Abweichung in der Distanz beträgt $150\,\text{m} - 146.94\,\text{m} = 3,06m$. Dies entspricht einem relativen Fehler von $2,04\%$.

6 Fazit und Ausblick

Durch die Implementierung einer beispielhaften Stereobildauswertung konnte gezeigt werden, dass es grundsätzlich möglich ist, mit relativ günstiger, massenproduzierter Hardware ein verteiltes System zur mehrdimensionalen Bilddatenverarbeitung zu realisieren.

Die verwendeten Kameras lassen sich leicht und flexibel über offene Protokolle, wie HTTP und JSON, steuern. Zu Beginn der Arbeit stand über die Sony Camera Remote API noch nicht der volle Funktionsumfang der Kameras zur Verfügung, jedoch wurde während der Bearbeitungszeit mit der API-Version 1.50 der volle Funktionsumfang freigegeben.

Für ein Stereosystem wäre es wünschenswert, dass sich beide Kameras genau gleich einstellen lassen. Bei der DSC-QX10 lassen sich einige wichtige Parameter, wie zum Beispiel die Blendenzahl und die Belichtungszeit, nicht manuell einstellen, sondern werden automatisch bestimmt.

Die erzielbare Synchronität bei der Aufnahme von Bildern ist für viele Anwendungsfälle ausreichend. Sollen aber schnellere Objekte getrackt werden, wäre eine geringere Zeitabweichung wünschenswert. Ein gewisser Zeitversatz ist nicht vermeidbar, da die eingesetzten Betriebssysteme nicht echtzeitfähig sind und die verwendeten Netzwerkprotokolle keine garantierten Antwortzeiten bieten.

Die Tablets sind in Verbindung mit den Kameras grundsätzlich für den Einsatz geeignet, allerdings bleibt die Android-Software-Einschränkung, die die gleichzeitige Verwendung der WLAN- und der Mobilfunk-Schnittstelle verbietet, bestehen. Diese Einschränkung könnte man jedoch umgehen, indem man sich Root-Zugriff auf das Tablet verschafft. Die Frage, ob dabei die Garantie erlischt, ist allerdings umstritten.

Die in der Arbeit vorgestellte Stereobildauswertung ist nur beispielhafter Natur. Der Algorithmus ist relativ langsam und erlaubt nur die Bestimmung einer Distanz. Sollte das System tatsächlich zum Einsatz kommen, wäre ein elaborierterer Algorithmus wünschenswert, sodass auch Geschwindigkeit, Fluglage, Position usw. ermittelt werden können.

Um hardwarespezifische Einschränkungen zu verringern, könnten die DSC-QX10-Kameras in Zukunft durch die DSC-QX100 ersetzt werden, da diese unter anderem Einstellmöglichkeiten für die Belichtungszeit und die Blendenzahl bieten.

Zur Verringerung der Zeitdifferenz beim Auslösen der Kameras könnten in Zukunft echtzeitfähige Betriebssysteme auf dem Tablet und dem Auswertungscomputer zum Einsatz kommen.

Sollte in Zukunft die gleichzeitige Verwendung von WLAN und Mobilfunknetz möglich sein, wäre es auch denkbar, komplett auf den Auswertungscomputer zu verzichten. Hierzu könnte die Auswertung verteilt auf den Tablets vorgenommen werden, die zu diesem Zweck über LTE miteinander verbunden wären. Um die Menge der zu übertragenden Daten gering zu halten, könnte auf den Tablets eine Vorverarbeitung der Bilddaten stattfinden. Mit Bezug auf den in Kapitel 5 vorgestellten Algorithmus bedeutet dies, dass anstatt der Übertragung des gesamten linken Bildes an das rechte Tablet nur das Histogramm und die Koordinaten des gewählten Bildausschnitts übertragen werden müssen. Die Tablets müssen miteinander kommunizieren, um einen gemeinsamen Auslösezeitpunkt festzulegen. Hierfür müssen die internen Uhren der Tablets immer synchron sein. Da die Uhren driften können, müssen sie regelmäßig synchronisiert werden. Dies kann zum Beispiel mit Hilfe des vorgestellten LED-Zählers täglich erfolgen. Da in diesem Szenario kein Auswertungscomputer zur Verfügung steht, müssen die Tablets die Bildauswertung vornehmen. Die hierfür benötigten OpenCV-Funktionen, wie zum Beispiel die Histogrammbildung und der Histogrammvergleich, stehen mit der OpenCV4Android-Bibliothek bereits für Android-Geräte zur Verfügung, sodass eine Portierung mit relativ geringem Aufwand durchführbar sein sollte.

7 Anhang

Inhalt der beiliegenden CD:

1. Arbeit als PDF-Datei
2. CameraClient
    a) Java-Quellcode als Eclipse-Projekt
    b) Javadoc-Dokumentation
    c) Compilierte App im APK-Format
3. StereoServer
    a) C-Quellcode als Eclipse-Projekt
    b) Doxygen-Dokumentation
4. Counter
    a) C-Quellcode als Eclipse-Projekt
    b) Doxygen-Dokumentation
5. Router
    a) Shell-Skript zur Konfiguration des Routers
    b) Benötigte Konfigurationsdateien

Literaturverzeichnis

[Internet Engineering Taskforce 1999a] Internet Engineering Taskforce: *Hypertext Transfer Protocol – HTTP/1.1*. Juni 1999. – URL http://tools.ietf.org/html/rfc2616. – (Stand: 16.06.2014)

[Internet Engineering Taskforce 1999b] Internet Engineering Taskforce: *Simple Service Discovery Protocol/1.0 Operating without an Arbiter*. Oktober 1999. – URL ftp://ftp.pwg.org/pub/pwg/ipp/new_SSDP/draft-cai-ssdp-v1-03.txt. – (Stand: 16.06.2014)

[Internet Engineering Taskforce 2006] Internet Engineering Taskforce: *The application/json Media Type for JavaScript Object Notation (JSON)*. Juli 2006. – URL http://www.ietf.org/rfc/rfc4627.txt. – (Stand: 16.06.2014)

[Jernej Mrovlje, Damir Vrančić 2008] Jernej Mrovlje, Damir Vrančić: *Distance measuring based on stereoscopic pictures*. 9th International PhD Workshop on Systems and Control: Young Generation Viewpoint. Oktober 2008

[JSON-RPC.ORG 2005] JSON-RPC.ORG: *JSON-RPC 1.0 Specifications*. 2005. – URL http://json-rpc.org/wiki/specification. – (Stand: 16.06.2014)

[Linux Wireless] Linux Wireless: *hostapd Linux documentation page*. – URL http://wireless.kernel.org/en/users/Documentation/hostapd. – (Stand: 16.06.2014)

[Sony Corporation 2014a] Sony Corporation: *API references for Camera Remote API beta*. April 2014

[Sony Corporation 2014b] Sony Corporation: *How to use the Camera Remote API beta*. April 2014

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 17. Juni 2014 Mario Glaser