Automatisiertes Kamera-Tracking zur Verbesserung der Lesbarkeit von Tafelanschrieben im Rahmen der Vorlesungsaufzeichnung an der Universität Stuttgart

Bachelorarbeit im Studiengang Audiovisuelle Medien

vorgelegt von: Philipp Lang

an der Hochschule der Medien Stuttgart

am: 20.12.2016

zur Erlangung des akademischen Grades eines Bachelor of Engineering

Erst-Prüfer: Prof. Uwe Schulz

Zweit-Prüfer: Jürgen Kössinger

Ehrenwörtliche Erklärung

Hiermit versichere ich, Philipp Lang, ehrenwörtlich, dass ich die vorliegende Bachelorarbeit mit dem Titel: „Automatisiertes Kamera-Tracking zur Verbesserung der Lesbarkeit von Tafelanschrieben im Rahmen der Vorlesungsaufzeichnung an der Universität Stuttgart" selbstständig und ohne fremde Hilfe verfasst und keine anderen als die angegebenen Hilfsmittel benutzt habe. Die Stellen der Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen wurden, sind in jedem Fall unter Angabe der Quelle kenntlich gemacht. Die Arbeit ist noch nicht veröffentlicht oder in anderer Form als Prüfungsleistung vorgelegt worden.

Ich habe die Bedeutung der ehrenwörtlichen Versicherung und die prüfungsrechtlichen Folgen (§ 24 Abs. 2 Bachelor-SPO (7 Semester) der HdM) einer unrichtigen oder unvollständigen ehrenwörtlichen Versicherung zur Kenntnis genommen.

Kurzfassung

Diese Arbeit beschäftigt sich mit dem Test, Programmanpassungen und -erweiterungen sowie der erstmaligen Installation eines automatisierten Kamera-Tracking-Systems an der Universität Stuttgart. Ziel dabei war es, die Aufzeichnung von Vorlesungen so zu verbessern, dass Tafelanschriebe auf Videoaufnahmen wesentlich besser lesbar werden. Im Laufe der Arbeit wird das Projekt „Hörsäle 2020" beschrieben, bei dem die Hörsäle an der Universität Stuttgart mit neuer Medientechnik ausgestattet werden und durch das diese Arbeit möglich wurde. Außerdem wird näher auf die Kamerasteuerung über VISCA und VISCA over IP eingegangen und der Aufbau sowie die Funktionsweise der Software LectureSight erklärt. Mit Hilfe dieser Software entstand ein Testaufbau, um zu prüfen, wie das Tracking funktioniert und welche Probleme dabei entstehen. Dieser Test wurde später auf einen Hörsaal ausgeweitet, um die Konfiguration abzuschließen und ein voll funktionsfähiges System zu erhalten. Durch mehrere Maßnahmen konnte das Tracking weiter verbessert werden, bis abschließend ein Test zur Lesbarkeit durchgeführt wurde, der bestätigt, dass mit einer bewegten Kamera und höherem Zoomfaktor die Deutlichkeit von Tafelanschrieben in den Videoaufzeichnungen drastisch zunimmt.

Abstract

This Bachelor thesis deals with the test, program modifications and extensions as well as the first installation of an automated camera tracking system at the University of Stuttgart. The aim was to improve the lecture recordings in such a way that the writing on the blackboard is much more readable on video recordings. In the course of this thesis, the "Hörsäle 2020" project will be described, in which the lecture halls at the University of Stuttgart will be equipped with new media technology and which made this thesis possible. In addition, the camera control via VISCA and VISCA over IP is discussed and the structure and functionality of the software LectureSight explained. With the help of this software, a test set-up was developed to test how the tracking works and to identify what problems arise. Later on this test was extended to a lecture hall to complete the configuration and to obtain a fully functional system. By several measures, the tracking was further improved and a test for readability was carried out, which confirms that the clarity of the writing on the blackboard increases drastically with a moving camera and a higher zoom factor.

Inhaltsverzeichnis

Ehrenwörtliche Erklärung ..... **I**
Kurzfassung ..... **II**
Abstract ..... **II**
Abkürzungsverzeichnis ..... **IV**
1 Zielsetzung **1**
2 Projektbeschreibung **3**
2.1 Vorlesungsaufzeichnung an der Universität Stuttgart 3
2.1.1 Lecturnity 3
2.1.2 Opencast Matterhorn 4
2.2 Projekt „Hörsäle 2020" 4
3 Kamerasteuerung **9**
3.1 VISCA 9
3.1.1 Aufbau des Protokolls 9
3.1.2 Zeitliche Abfolge 10
3.2 VISCA over IP 11
4 Kamera-Tracking **15**
4.1 Lösungsansätze und Auflösung 15
4.2 LectureSight 17
4.2.1 OSGi 18
4.2.2 Apache Maven 19
4.2.3 OpenCL 20
4.2.4 Funktionsweise 22
4.3 Forschung 24
5 Testaufbau **27**
5.1 Übersicht 27
5.2 Grafikkarte 28
5.3 LectureSight Konfiguration 30
5.3.1 Installation 30
5.3.2 GStreamer 31
5.3.3 Kamerasteuerung 33
5.3.4 Kamerakalibrierung 33
5.3.5 Szenenprofil 35
5.3.6 Test des Trackings 36
5.3.7 Probleme in LectureSight 39
5.4 Kamera 39
5.5 Crestron-Programmierung 42
6 Verbesserung des Trackings **46**
6.1 Erweiterung des virtuellen Kameraoperators 46
7 Fazit **53**
7.1 Lesbarkeit 53
7.2 Ausblick 58
Abbildungsverzeichnis **59**
Literaturverzeichnis **60**
Anhang A **65**

Abkürzungsverzeichnis

ACK	Acknowledge (Bestätigung in Protokollen)
API	Application Programming Interface (Programmierschnittstelle)
ASCII	American Standard Code for Information Interchange (Textcodierung)
CA	Capture Agent (Aufzeichnungsserver)
CPU	Central Processing Unit (Prozessor)
CU	Compute Units (Recheneinheit bei OpenCL-Geräten)
CUDA	Compute Unified Device Architecture (Programmiertechnik für GPUs)
CR	Carriage Return (Wagenrücklauf → Zeilenumbruch)
DSP	Digital Signal Processor (Prozessor zur digitalen Signalverarbeitung)
FPGA	Field-programmable Gate Array (integrierter Schaltkreis [IC], der nach der Herstellung noch konfiguriert werden kann)
GPU	Graphics Processing Unit (Grafikprozessor)
H.264	Standard zur Videokompression (H.264/MPEG-4 AVC)
HDMI	High Definition Multimedia Interface (digitales Videokabel)
HPC	High Performance Computing (Hochleistungsrechnen)
ICMP	Internet Control Message Protocol (Netzwerkprotokoll)
IP	Internet Protocol (Netzwerkprotokoll)
JAR	Java Archive (ausführbares Java-Programm)
JPEG	Joint Photographic Experts Group (Bildformat)
LAN	Local Area Network (internes Netzwerk)
LF	Line Feed (Zeilenvorschub → Zeilenumbruch)
MJPEG	Motion JPEG (Videoformat, besteht aus einzelnen JPEG-Bildern)
NFL	Neue Medien in Forschung und Lehre
OpenCL	Open Computing Language (Schnittstelle für Parallelrechner)
OSGi	Open Services Gateway initiative (Softwareplattform)
POM	Project Object Model (Konfigurationsdatei für Maven)
PT	Pan/Tilt (schwenken, neigen)
PTZ	Pan/Tilt/Zoom (schwenken, neigen, zoomen)
RS-232	Serielle Schnittstelle (offiziell: ANSI EIA/TIA-232-F-1997)
RS-422	Serielle Schnittstelle (offiziell: ANSI/TIA/EIA-422-B-1994)
RTSP	Real-Time Streaming Protocol (Protokoll für Videostreaming)
SIMD	Single instruction, multiple data (Klassifikation von Rechnerarchitekturen)
SM	Streaming Multiprocessor (Teil der GPU, auf der CUDA Kernel laufen)
SMM	Streaming Multiprocessor Maxwell
SMX	Streaming Multiprocessor Next Generation (Kepler)
TCP	Transmission Control Protocol (Netzwerkprotokoll, verbindungsorientiert)
TIK	Technische Informations- und Kommunikationsdienste
UBA	Universitätsbauamt
UDP	User Datagram Protocol (Netzwerkprotokoll, verbindungslos)
URL	Uniform Resource Locator (Webadresse)
VISCA	Protokoll von Sony zur Kamerasteuerung
VLAN	Virtual Local Area Network (logisches Teilnetz innerhalb eines Netzwerks)
XML	Extensible Markup Language (Dateiformat)

1 Zielsetzung

An der Universität Stuttgart gibt es seit 2004 ein System zur Aufzeichnung von Vorlesungen (Vass, 2014). Dieses ist zum heutigen Zeitpunkt weitgehend automatisiert und wird durch die Dozenten selbst bedient. Durch eine fest installierte HD-Kamera und durch den Einsatz von IPTV-Encodern, die das Signal der Kamera und der Beamer-Quelle zum Aufzeichnungsserver senden, ist zudem keine mobile Technik nötig.

Die Kamera hat bisher drei Voreinstellungen – Totale, Tafel und Redner. Die Totale filmt hierbei den Raum in seiner kompletten Breite (Abbildung 1), die Einstellung Tafel sorgt dafür, dass alle Tafeln im Bild sind (Abbildung 2) und die Einstellung Redner zeigt eine Nahaufnahme des Redners am Pult (Abbildung 3). Der Dozent muss sich bei der Aufnahme entscheiden, ob es wichtig ist seine Gestik und Mimik zu erkennen, was einen sehr eingeschränkten Aktionsradius bedeutet, oder ob der komplette Raum gefilmt werden soll, wobei in großen Hörsälen selbst mit der Einstellung „Tafel" die Lesbarkeit von Tafelanschrieben gering ist (Wulff, Fecke, Rupp & Hamborg, 2014, S. 185).

Abbildung 1: Einstellungsgröße Totale (Eigene Darstellung)

Abbildung 2: Einstellungsgröße Tafel (Eigene Darstellung)

Abbildung 3: Einstellungsgröße Redner (Eigene Darstellung)

Dies führt dazu, dass Vorlesungen mit vielen Tafelanschrieben nicht aufgezeichnet werden. Laut Wulffetal.‚ÉWulffetal.Wulff et al. (2014) betrifft das sowohl Vorlesungen aus der Mathematik, als auch aus der Physik und den Ingenieurwissenschaften. Da genau diese Studiengänge an der Universität Stuttgart stark vertreten sind, gehe ich in dieser Arbeit der Frage auf den Grund, wie ein automatisiertes Kamera-Tracking-System integriert werden kann und wie dieses die Lesbarkeit von Tafelanschrieben verbessert, um zukünftig eine möglichst uneingeschränkt nutzbare Vorlesungsaufzeichnung bieten zu können.

Hierzu habe ich mit Hilfe der Software LectureSight ein Tracking-System installiert und getestet, um das bestmögliche Resultat einer Vorlesungsaufzeichnung und eine vollautomatische Kameraführung zu erhalten.

Zu Beginn dieser Arbeit werde ich in Kapitel 2 im Detail auf die Gegebenheiten an der Universität Stuttgart eingehen und das Projekt beschreiben, aufgrund dessen ich mich mit diesem Thema beschäftige. Im darauffolgenden Teil werde ich erläutern, wie man die Voraussetzungen für das Kamera-Tracking durch die Steuerung einer PTZ-Kamera schafft. Anschließend gehe ich in Kapitel 4 genauer darauf ein, wie die Software LectureSight aufgebaut ist, die das Kamera-Tracking durchführt und in Kapitel 5 erläutere ich wie man ein Tracking-System mit LectureSight aufbaut, wie ich dieses getestet habe und welche Probleme dabei auftraten. Kapitel 6 dreht sich um mögliche Verbesserungen an der Software bevor ich im letzten Kapitel die vorhergehenden Teile zusammenfasse und einen Ausblick gebe.

2 Projektbeschreibung

2.1 Vorlesungsaufzeichnung an der Universität Stuttgart

Das an der Universität Stuttgart vorhandene System zur Vorlesungsaufzeichnung besteht im Wesentlichen aus drei Elementen: der Aufzeichnung, einem Videoserver und einer Lernplattform. Durch diese Arbeit wurde das Kamera-Tracking hinzugefügt, wofür abgesehen von der Installation neuer Software auch Änderungen an bestehender Software, wie beispielsweise der Crestron Mediensteuerung nötig waren. Abbildung 4 gibt eine Übersicht über die zur Vorlesungsaufzeichnung verwendete Software.

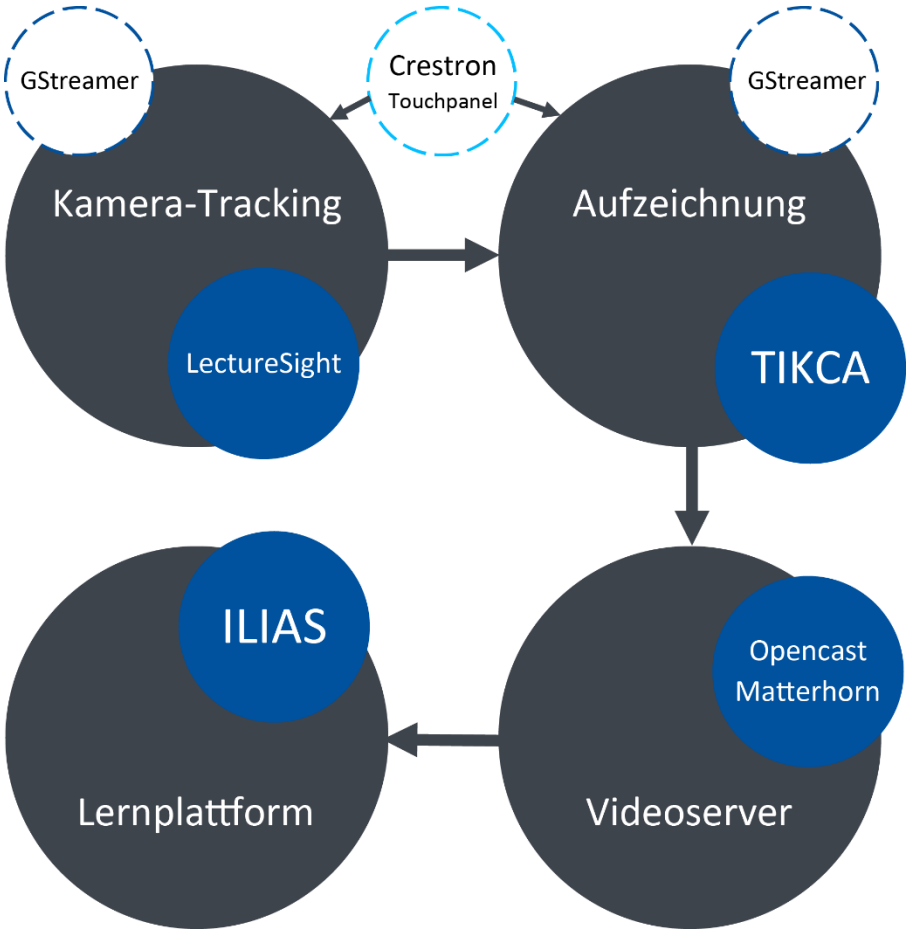

Abbildung 4: Software-Übersicht (Eigene Darstellung)

2.1.1 Lecturnity

Bis zum Jahr 2014 wurden mit Hilfe der Software Lecturnity die PowerPoint-Präsentationen der Dozenten aufgezeichnet. Hierzu musste diese Software auf dem Laptop jedes Dozenten installiert werden. Der Ton wurde separat aufgezeichnet.

Lecturnity wird jedoch nicht mehr weiterentwickelt und führt zu Problemen bei den aktuellen Versionen von Microsoft Office und Windows (TIK, 2015). Daher hat sich die Abteilung „Neue Medien in Forschung und Lehre" (NFL) der Technischen Informations- und Kommunikationsdienste der Universität Stuttgart (TIK) für eine neue Lösung entschieden, die einfacher zu bedienen und umzusetzen ist.

2.1.2 Opencast Matterhorn

Anstatt Software auf den Laptops zu verwenden werden nun in den Hörsälen sogenannte Capture Agents (CA) fest installiert. Diese leistungsfähigen Computer bekommen dasselbe Bildsignal, das auch der Beamer erhält und können dieses aufzeichnen. Zusätzlich wird den CAs auch der Ton aus der Beschallungsanlage zur Aufzeichnung zugeführt.

Ein Capture Agent ist ein Computerprogramm, dessen Aufgabe darin besteht, ein Audio- und Videosignal aufzuzeichnen, damit dieses anschließend für ein Videoportal verarbeitet werden kann. An der Universität Stuttgart wird hierbei eine Eigenentwicklung, der sogenannte TIKCA verwendet (Zurek, 2016). Der TIKCA ist ein Python-Skript, das mit der Mediensteuerung kommuniziert und abhängig von Benutzereingaben die Aufzeichnung startet beziehungsweise stoppt.

Als Videoportal wird seit 2014 die Software Opencast Matterhorn eingesetzt (TIK, 2015). Hierbei handelt es sich um eine Open Source Software, die offiziell 2009 erschienen ist (Brdiczka et al., 2010, S. 176). Opencast Matterhorn war der Arbeitstitel, der mit dem Release der Version 2.0.0 am 17. Juli 2015 (Kiesow, 2015) in Opencast geändert wurde (Opencast, 2016). An der Universität Stuttgart ist noch Version 1.6 und daher auch Matterhorn im Einsatz. Es wurde lediglich der „Theodul" Videoplayer auf die neueste Version aktualisiert.

2.2 Projekt „Hörsäle 2020"

Durch die im Jahr 2014 im Hochschulfinanzierungsvertrag „Perspektive 2020" bereitgestellten Mitteln des Landes Baden-Württemberg (Bergert, 2014, S. 11) wurde zusammen mit dem Universitätsbauamt (UBA), dem Dezernat VI (Technik und Bauten) der Universität Stuttgart, dem TIK und mehreren Fachplanern ein Konzept erstellt, um bis zum Ende des Jahres 2016 63 Hörsäle zu ertüchtigen und im Zuge dessen mit neuer Medientechnik auszustatten.

Abbildung 5: Übertragung (Wireworx GmbH, 2015b)

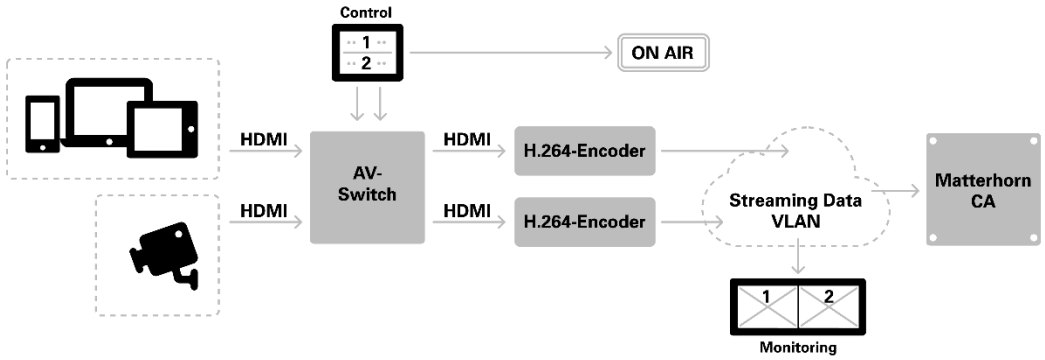

Abbildung 6: Hörsaalschema IPTV (Eigene Darstellung, Daten entnommen aus Grube, 2015)

Dieses Konzept sah vor, dass die Hörsäle zur Vereinfachung sowohl der finanziellen, als auch der technischen Planung in vier Raumtypen eingeteilt werden. Diese waren zu Beginn des Projekts die Hörsaalgrößen S, M, L und XL. Sie unterscheiden sich beispielsweise durch die Anzahl der Sitzplätze, der Anzahl an Videoprojektionen und ob es eine Sprachbeschallung oder auch eine „fest installierte IPTV-Übertragungs- und Aufzeichnungseinheit" (Wireworx GmbH, 2015a) geben sollte.

Da in die Konzeption auch das TIK, insbesondere die Abteilung NFL, integriert wurde, war es möglich, eine neue Struktur zur Vorlesungsaufzeichnung zu schaffen. Die Abteilung NFL betreut unter anderem die Lernplattform ILIAS sowie sämtliche Web-Dienste an der Universität Stuttgart.

Im Mai 2015 fand ein Startgespräch mit der Firma Wireworx, dem zuständigen Fachplaner für den Bereich Medientechnik, sowie Pascal Grube und Pascal Zurek vom TIK/NFL

statt. Hierbei entstand die folgende Idee: Jeder Hörsaal ab Größe M sollte eine Kamera erhalten, welche direkt einen IPTV-Stream erzeugt. Zusätzlich sollte ein weiterer IPTV-Encoder eingebaut werden, der von demselben Bild, mit dem auch der Beamer bespielt wird, ebenfalls einen Stream erzeugt. Diese beiden Streams werden über ein eigenes VLAN zum Capture Agent gesendet.

Über eine Mediensteuerung ist es nun möglich, das zu sendende Signal frei auszuwählen und den CA direkt selbst zu starten. Durch ein Vorschaubild auf dem Touchpanel erhält der Dozent eine Übersicht über die beiden Sendeleitungen (siehe Abbildung 5). Wenn der CA läuft, wird dies über eine ON AIR-Leuchte signalisiert – sowohl dem Dozenten, als auch den Studenten. Zusätzlich zeigt diese Leuchte auch an, dass die Kamera angeschaltet ist (siehe Abbildung 7). Aufgrund von Datenschutzbestimmungen muss die Kamera, wenn sie nicht verwendet wird, nicht nur komplett ausgeschaltet sein, sondern es muss außerdem physikalisch unmöglich sein, damit trotzdem ein Bild zu erzeugen. Daher sind die PTZ-Kameras in einem nur zu einer Seite geöffneten Kasten verbaut und drehen sich nach innen, wenn sie nicht verwendet werden. So ist auch auf den ersten Blick leicht zu erkennen, dass gerade nicht gefilmt werden kann.

Die ausgewählten IPTV-Kameras (Sony SRG-300SE) konnten kein Multicast-Streaming, das aber von Seite der Universität gefordert war, weshalb im Laufe des Projekts das Konzept leicht abgeändert wurde und ein anderer Kameratyp, die Sony SRG-300H, gewählt wurde. Dieser Typ hat nur einen HDMI-Ausgang und erzeugt selbst keinen Stream mehr. Daher wird die Kamera nun direkt auf einen zusätzlichen IPTV-Encoder geführt, der Multicast-Streaming unterstützt (siehe Abbildung 6).

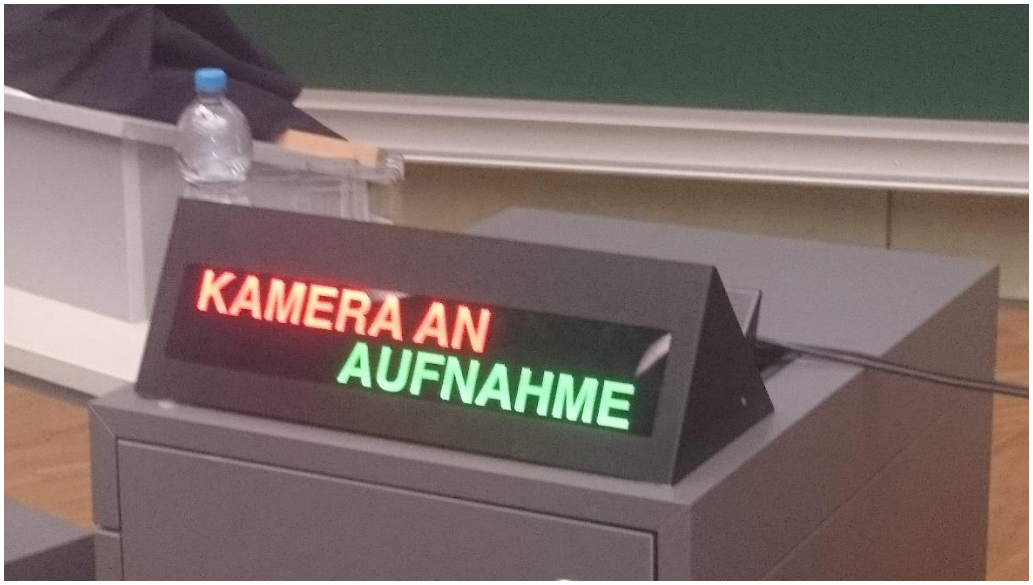

Abbildung 7: ON AIR-Leucte (Eigene Darstellung)

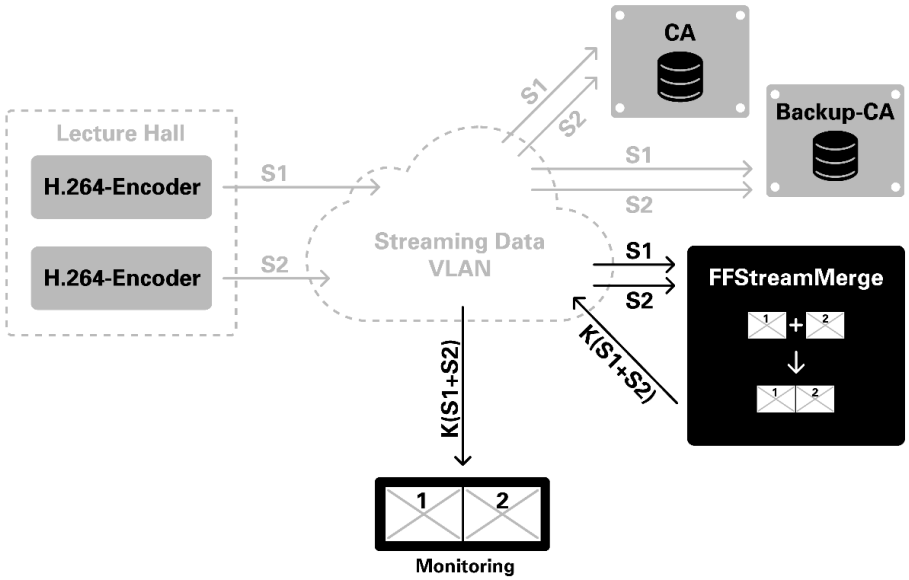

Abbildung 8: Erstellung der Vorschaubilder (Grube, 2015)

Da das Touchpanel der Mediensteuerung nur einen Stream darstellen kann, muss zuerst ein passender Stream erzeugt werden, der die beiden Bilder kombiniert. Dies wurde mit FFStreamMerge und einem MJPEG-Stream umgesetzt (siehe Abbildung 8).

Damit der Dozent selbst eine Aufnahme starten, pausieren oder stoppen kann, hat das TIK für die Kommunikation zwischen Mediensteuerung und CA ein eigenes Protokoll entwickelt (siehe Abbildung 9). Dieses kennt die Zustände Idle, Starting, Recording, Pausing, Paused, Unpausing, Stopping und Error. Der Status wird einmal pro Sekunde per UDP durch die Mediensteuerung abgefragt und auf dem Touchpanel dargestellt, um eine Rückmeldung über die Aufzeichnung zu erhalten. Zusätzlich schaltet der TIKCA, wenn dieser im Zustand Recording ist, zwei Ports zweier Netzwerksteckdosen ein und aktiviert damit die beiden Leuchten beim Dozenten und bei der Kamera, um allen Beteiligten zu signalisieren, dass die Vorlesung nun aufgezeichnet wird.

Die fertig aufgezeichneten Vorlesungen werden den zuständigen Dozenten nach dem Ingest, also der Einspielung in Opencast, in der E-Learning Plattform ILIAS angezeigt. Dort können diese zum Beispiel noch das Video trimmen, also Anfang und Ende entfernen, und anschließend veröffentlichen. Wahlweise kann auch nur eines von beiden Signalen veröffentlicht werden. Die Aufzeichnung steht den in den entsprechenden Kursen eingeschriebenen Studenten zur Verfügung.

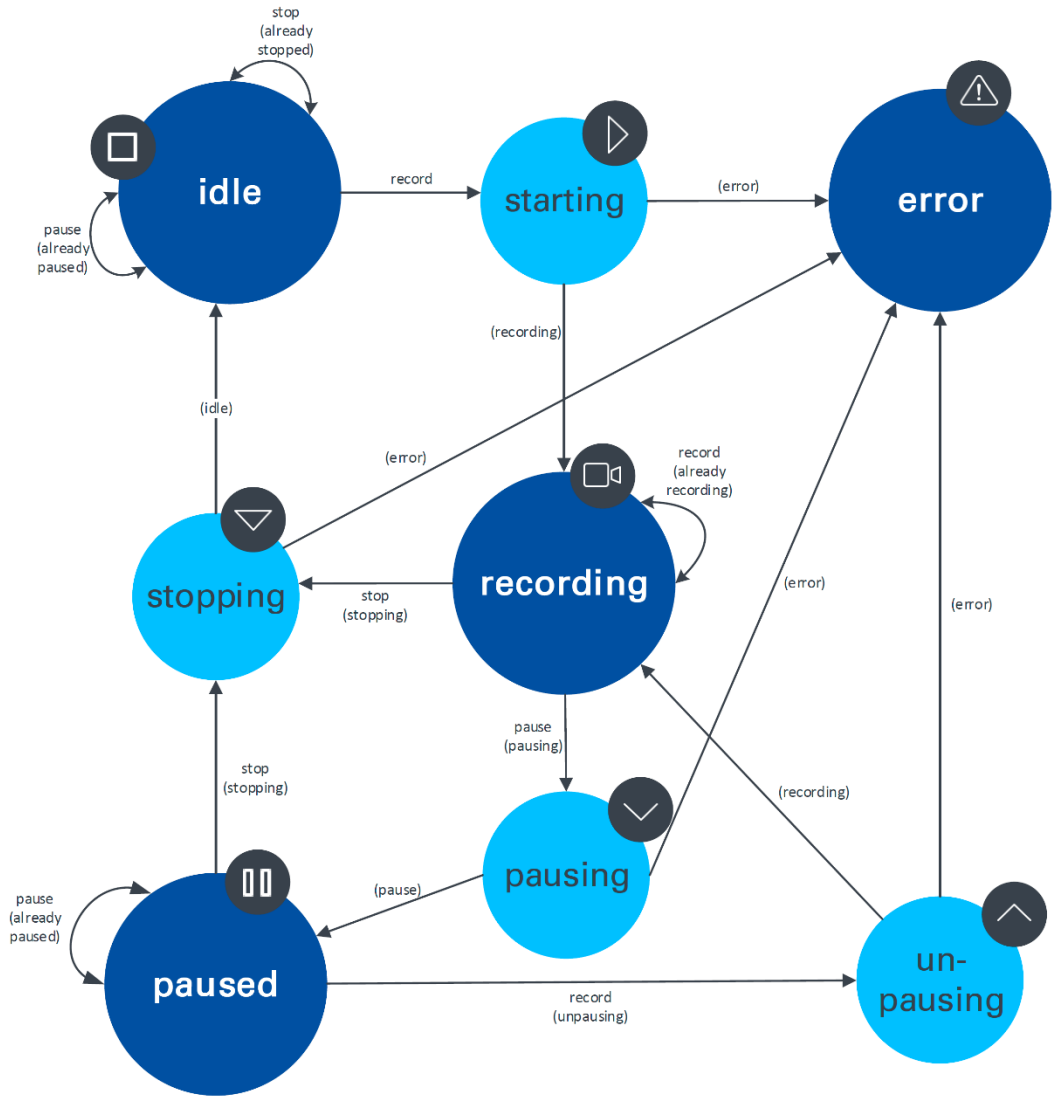

Abbildung 9: Protokoll zwischen Mediensteuerung und CA (Eigene Darstellung, Daten entnommen aus Grube, 2015 und Skiles, 2015)

Zusätzlich zu diesem Aufzeichnungskonzept wurde während des Startgespräches das Bestreben geäußert, eine vollautomatische Kameraführung zu erhalten. Zu diesem Zeitpunkt war bereits der Einsatz der Software LectureSight angedacht, die daher in dieser Arbeit verwendet wurde.

3 Kamerasteuerung

Zunächst muss die softwareseitige Kommunikation mit den Kameras geklärt werden, um das Kamera-Tracking der beweglichen Kameras sowie die Softwaresteuerung des Trackings zu ermöglichen.

Im Projekt „Hörsäle 2020" erhielten die Hörsäle der Größe S erhalten aus Kostengründen keine Kamera. In den größeren Hörsälen wurde die Kamera Sony SRG-300H installiert. Diese wird über das proprietäre, aber offen dokumentierte, Protokoll VISCA gesteuert. Bisher erfolgt die Steuerung jeweils über eine Mediensteuerung, die teilweise seriell an die Kamera angebunden ist und teilweise über Netzwerk mit dieser kommuniziert.

Da das VISCA-Protokoll für den weiteren Verlauf dieser Arbeit wichtig ist, wird dessen Aufbau im folgenden Kapitel näher erläutert.

3.1 VISCA

VISCA ist einerseits ein von Sony entwickeltes Protokoll zur Steuerung von PTZ-Kameras, andererseits bezeichnet VISCA auch ein Bussystem, mit dem seriell über RS-232 oder RS-422 bis zu sieben Kameras mit einem Controller (hier: Mediensteuerung) angesteuert werden können. Diese Topologie wird als Daisy Chain bezeichnet, da alle Geräte wie bei einer Gänseblümchenkette miteinander verbunden sind. Hierbei besitzt jede Kamera sowohl einen VISCA Input als auch einen VISCA Output (Sony, 2014).

3.1.1 Aufbau des Protokolls

Ein VISCA-Paket kann zwischen 3 und 16 Byte lang sein. Es beginnt immer mit dem Header und endet mit einem Terminator (0xFF), der das Ende eines Pakets markiert (siehe Abbildung 10).

Der Header besteht aus je 3 Bit Absender- und Empfängeradressen. Der eigentliche Payload kann unterschiedlich lang sein. (Sony, 2014, S. 24)

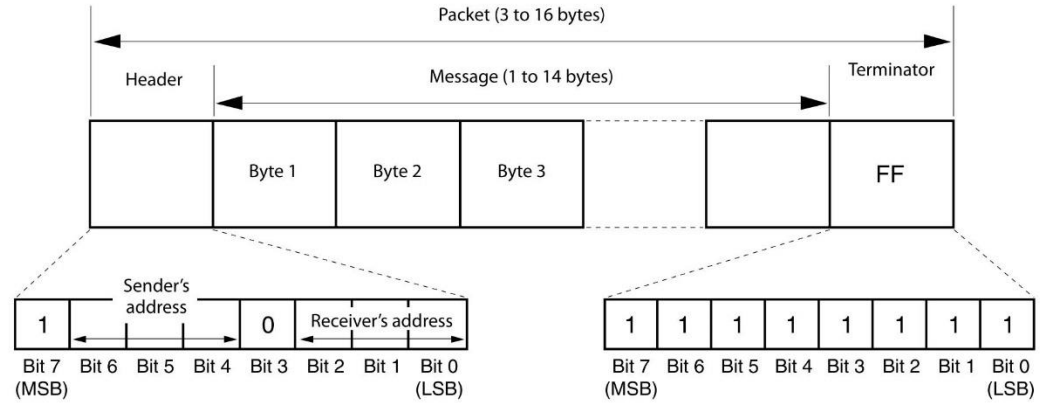

Abbildung 10: Paketstruktur von VISCA (Sony, 2014, S. 24)

3.1.2 Zeitliche Abfolge

Bevor der Controller eine neue Nachricht sendet, muss dieser warten, bis er eine Bestätigung (ACK) der Kamera erhält (Sony, 2014, S. 25).

Es können maximal zwei VISCA-Befehle an eine Kamera gesendet werden, da die SRG-300H zwei Sockets besitzt. Diese Sockets sind temporäre Speicher (Buffer), die einen Befehl solange beibehalten, bis dieser vollständig ausgeführt wurde. Dem Controller wird durch die ACK-Nachricht mitgeteilt, welcher der beiden Sockets verwendet wurde. Wenn beide Sockets benutzt sind, muss abgewartet werden, bis ein Befehl vollständig ausgeführt wurde (Completion, siehe Abbildung 11). Bei mehreren gesendeten Befehlen werden diese eventuell nicht der Reihe nach fertiggestellt.

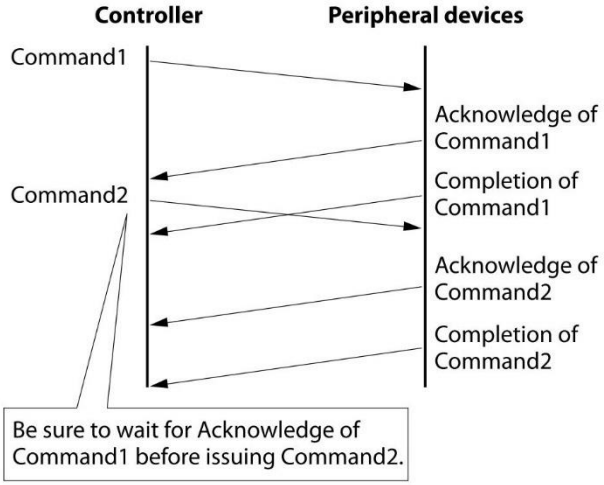

Abbildung 11: Zeitlicher Ablauf von VISCA (Sony, 2014, S. 25)

Gewisse Nachrichten, wie zum Beispiel Abfragen (inquiries), können immer gesendet werden, da sie keinen Socket verwenden. Diese Nachrichten erhalten keine Bestätigungs-, sondern nur die Completion-Nachricht mit dem Socket 0. (Sony, 2014, S. 26)

Auf Befehle oder Anfragen gibt es drei mögliche Antworten (Sony, 2014, S. 25):

- Acknowledge: Bestätigung des Befehls. Anfragen, CANCEL und „Device Setting Commands" erhalten keine Bestätigung.
- Completion: Bestätigt die Ausführung eines Befehls oder beinhaltet bei Anfragen die gewünschten Daten.
- Error: Fehlermeldungen können entweder anstelle von ACK oder Completion, oder bei manchen Befehlen auch nach der ACK-Nachricht auftreten. Man unterscheidet zwischen:
    - Message length error
    - Syntax Error
    - Command buffer full
    - Command cancelled
    - No socket (to be cancelled)
    - Command not executable (Sony, 2014, S. 25)

3.2 VISCA over IP

VISCA over IP ist ein bei neueren Modellen eingesetztes Protokoll, das auf VISCA basiert. Damit können mehr als sieben Kameras gleichzeitig gesteuert werden.

Dies erfordert einige Änderungen im Protokoll. Die Wichtigste ist dabei der zusätzliche „Message Header", der den Payload-Typ, die Payload-Länge und eine Sequenznummer beinhaltet (Sony, 2014, S. 30).

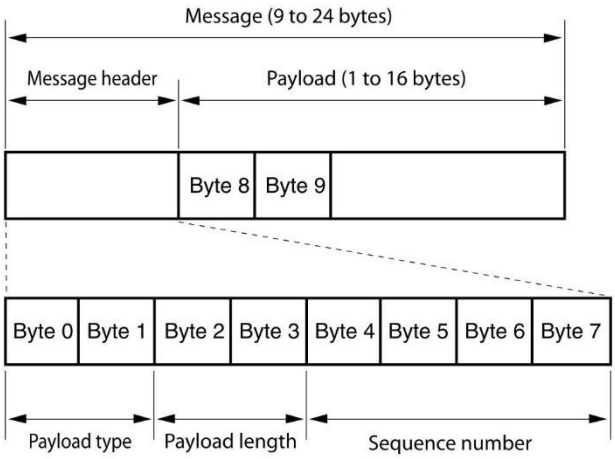

Abbildung 12: Aufbau des Message Headers von VISCA over IP (Sony, 2014, S. 30)

Die Kommunikation erfolgt über LAN (bis 100Base-TX) mittels IPv4 und UDP auf Port 52381. Die Topologie ist nun eine Stern-, anstatt einer Ring-Topologie bei der Verwendung von RS-232/RS-422, (Sony, 2014, S. 29).

Durch die Verwendung von IP ist es jetzt möglich schon im IP Header zu erkennen, wer Absender und Empfänger eines Paketes war, da dort die IP-Adresse angegeben ist. Daher wird die Empfängeradresse im VISCA-Protokoll fest auf 1 und die Absenderadresse auf 9 festgelegt. (Sony, 2014, S. 25)

Bei den neu eingeführten Payload-Typen unterscheidet man die Folgenden (Sony, 2014, S. 30):

- VISCA command: VISCA Steuerbefehl vom Controller zur Kamera
- VISCA inquiry: Anfragen des Controllers an die Kamera
- VISCA reply: ACK, Completion, Antwort oder Fehlermeldung von der Kamera zum Controller
- VISCA device setting command: Spezielle Kommandos, zum Beispiel:
    - IF_Clear: leert den Socket der Kamera (Sony, 2014, S. 26)
    - CAM_VersionInq: fragt Versionsinformationen der Kamera ab (Sony, 2014, S. 27)
- Control command: hier gibt es nur zwei mögliche Inhalte:
    - RESET: setzt die Sequenznummer auf 0 zurück, unabhängig von der soeben verwendeten Sequenznummer. Die nächste gültige Sequenznummer ist nun jedoch nicht 0, sondern 1.
    - ERROR: Fehler in der Sequenznummer (0x0F 0x01) oder fehlerhafte Nachricht (0x0F 0x02)
- Control reply: Antwort auf den RESET-Befehl. Acknowledgement, also Bestätigung des Resets (0x01), ohne Terminator

Da UDP verbindungslos ist, also den Zustand einer Verbindung nicht kennt, muss die Anwendung auf Controllerseite dafür sorgen, dass auf Nachrichten gewartet und bei einem Timeout entsprechend gehandelt wird, also zum Beispiel eine erneute Übertragung stattfindet. Falls das Paket verloren ging, kann nach einem Timeout das vorherige Paket mit der alten Sequenznummer erneut übermittelt werden.

Welche Fehler durch Verlust von Nachrichten entstehen können, zeigen die folgenden beiden Grafiken:

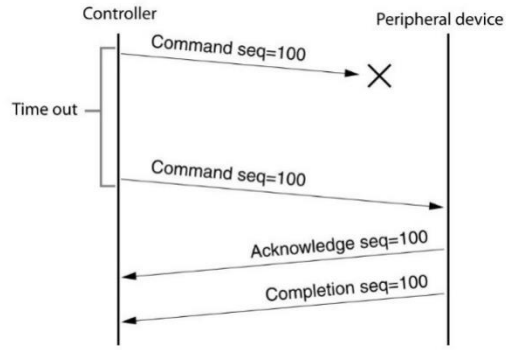

Abbildung 13: Verlust des VISCA-Befehls (Sony, 2014, S. 33)

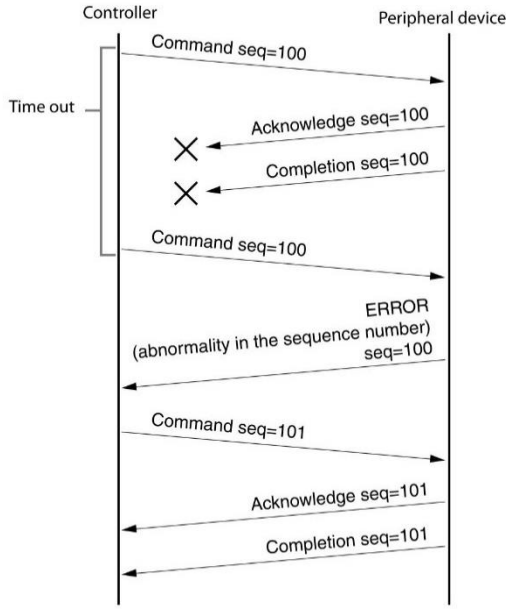

Abbildung 14: Verlust der ACK oder Completion Nachricht (Sony, 2014, S. 33)

Abbildung 13 zeigt den Verlust eines Pakets zur Kamera. Der Controller wartet eine Antwort ab, in diesem Fall die Bestätigung, dass die Nachricht ankam (ACK), aber erhält diese in diesem Fall nicht. Nach einem Time out, also einer gewissen Zeit, die abgewartet wird, wird das Paket erneut mit der gleichen Sequenznummer gesendet. Da die Kamera das vorige Paket nicht erhalten hat, stellt es kein Problem dar, dass die selbe Sequenznummer erneut verwendet wurde und die Kamera führt anschließend das gewünschte Kommando aus.

Abbildung 14 ist auf den ersten Blick für den Controller gleich, da dieser keine Bestätigung erhält. Hier hat die Kamera jedoch sehr wohl das gesendete Paket erhalten, aber die Antwort ging verloren. Daher erhält der Controller beim zweiten Versuch die Fehlermeldung „abnormality in the sequence number", da zwei Pakete mit derselben Sequenznummer bei der Kamera ankamen. Sobald eine neue Nachricht mit einer höheren Sequenznummer gesendet wird, bekommt der Sender eine positive Rückmeldung.

Problematisch ist hierbei, dass es keine Definition für das Time out gibt, sondern dass dies selbst vom Controller festgelegt werden muss. In einem Test wurde daher versucht herauszufinden, wie lange es dauert bis die Kamera eine eingehende Nachricht mit einem ACK bestätigt. Dazu wurde mit einem Python-Skript (Zurek & Pérez Vázquez, 2016) direkt eine Nachricht an die Kamera gesendet, auf die Antwort gewartet und die Nachrichten jeweils auf der Konsole mit Zeitstempel ausgegeben (siehe Abbildung 15). Dies waren durchschnittlich 33 Millisekunden, maximal 47 Millisekunden (siehe Abbildung 16). Jedoch verhält sich die hier verwendete Kamera Sony SRG-300H nicht immer wie erwartet und wenn gerade eine Aktion ausgeführt wird, verzögert sich eine Antwort entsprechend. Aufgrund dieser Erkenntnis war es möglich, die Implementierung des Protokolls in LectureSight anzupassen, um das korrekte Time out beim Senden und Empfangen der Kamerakommandos zu beachten und somit die Fehlerbehandlung zu verbessern. Hierfür wurde für das Time out ein Wert von 50 Millisekunden gewählt, da dieser im vorigen Test nicht überschritten wurde.

```
ls@nflmhlsv03axl:~/PTZ$ python camauftafel.py
Socket created
Socket bind complete
Seqnr bisher: 255
resetbefehl: 01ff
payload length 2
seqnr 255
Sequence No. 255
[00:00:53.687]: Sending message: 020000020000000ff01ff
[00:00:53.687]: Reply: 02 01 00 01 00 00 00 00 01
Seqnr nun: 100
Formatting header and payload  8101040002ff
payload length 6
seqnr 100
Sequence No. 100
Sending UDPheader, header and payload  0100000600000006481010400002ff
[00:00:53.766]: Sending message: 0100000600000006481010400002ff
[00:00:53.797]: Reply: 01 11 00 03 00 00 00 64 90 41 ff
ACK
[00:00:53.797]: Reply: 01 11 00 03 00 00 00 64 90 51 ff
COMPLETE
```

Abbildung 15: Konsolenausgabe des Python-Skripts (Eigene Darstellung)

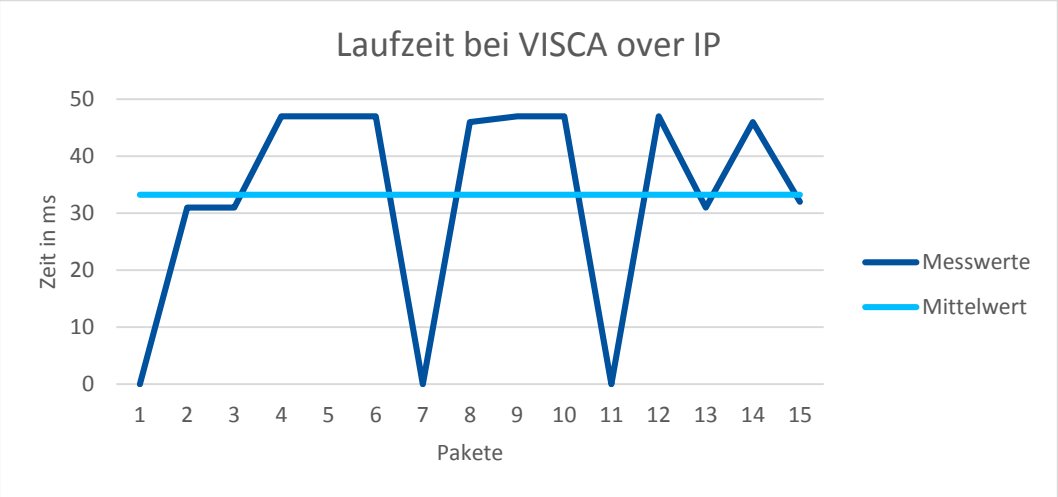

Abbildung 16: Laufzeit von VISCA over IP Paketen (Eigene Darstellung)

4 Kamera-Tracking

4.1 Lösungsansätze und Auflösung

Es gibt mehrere Möglichkeiten, um die Lesbarkeit der Tafelanschriebe zu verbessern. Im Projekt „Hörsäle 2020" wurde bereits zu Beginn der Planung die Option einer zweiten Kamera vorgesehen, um diese optional nachrüsten zu können. Mit Hilfe dieser zweiten Kamera sollte ein Übersichtsbild erzeugt werden, welches anschließend von einer Software analysiert wird. Damit kann dann eine Position berechnet werden, zu der die eigentliche Kamera bewegt wird.

Somit ist grundsätzlich die automatische Kameraführung möglich und man kann nun die in Abbildung 3 (Seite 2) gezeigte Einstellungsgröße „Redner" beibehalten, aber diesen Redner dabei auch verfolgen. Durch die mit dem optischen Zoom erreichte höhere relative Auflösung der Tafel, das heißt das Objekt wird bei gleichbleibender Auflösung der Kamera größer dargestellt, sollte für den späteren Betrachter auch ein Tafelanschrieb leicht erkennbar sein.

Ein weiterer Lösungsansatz für dieses Problem ist es, die Auflösung der Tafel absolut zu erhöhen. Dies wäre mit einer 4K-Kamera möglich, die eine Totale filmt. Das Bild einer 4K-Kamera ist doppelt so hoch und breit wie das einer HD-Kamera, da die Auflösung hier 3840 × 2160 Pixel beträgt. Insbesondere bei der Fernseh- und Kinofilmproduktion existieren auch Kameras mit einer Auflösung von 4096 × 2160 Pixeln, bei denen das Bild also noch etwas breiter ist. Dabei wird hier ebenfalls dieselbe vertikale Auflösung erreicht.

Das klingt vorerst nach einer deutlichen Verbesserung im Gegensatz zum vorigen Lösungsansatz, jedoch muss man sich am Ende überlegen, wie der Vorteil der höheren Auflösung des 4K-Materials ausgenutzt werden kann. Die beiden Studien von Statista (2016) und von HbbTV (2016) zeigen, dass 2015 nur 13,5 Prozent des weltweiten Absatzes an Fernsehern aus 4K-Geräten bestand. Daher kann davon ausgegangen werden, dass bisher nur die wenigsten Nutzer ein kompatibles Endgerät besitzen und somit die Tafelanschriebe ebenfalls nicht besser erkennen könnten.

Daher bleibt nur die Möglichkeit einen Videoplayer zur Verfügung zu stellen, der einen digitalen Zoom ermöglicht, um die native Auflösung der Kamera auszunutzen. Das Problem hierbei ist jedoch, dass im Vergleich zum HD-Stream bei 4K eine vierfache Datenrate benötigt wird. Dies wiederum fordert ein adaptives Streaming, bei dem verschiedene

Qualitätsstufen und dadurch auch unterschiedliche Datenraten zur Verfügung gestellt werden, damit die Nutzung auch bei schlechten Internetverbindungen mit geringer Bandbreite möglich ist. Ein solches System wird unter anderem an der Universität zu Köln eingesetzt (Rolf, 2016).

Der Konsum dieser Aufzeichnungen wird durch den Videoplayer mit digitalem Zoom ebenfalls deutlich anspruchsvoller, da die Nutzer nun in einer passenden Zoomstufe ständig im Player hin- und herfahren müssen. Außerdem ist dies nicht auf mobilen Endgeräten möglich, da Videostreams dort auf nativen Playern dargestellt werden und somit nicht der Videoplayer mit digitalem Zoom genutzt werden kann. Somit bleibt das Problem von nicht erkennbaren Tafelanschrieben teilweise noch bestehen.

Eine weitere Lösung wäre es, das 4K-Video nach der Aufzeichnung zu verarbeiten und dabei ein Tracking durch die Erkennung von Personen durchzuführen. Dies wäre qualitativ die beste Lösung, da es die positiven Aspekte aus beiden anderen Lösungsansätzen kombiniert. Außerdem wäre es im Gegensatz zum Live-Tracking möglich, durch eine ausführliche Analyse zu einem besseren Tracking-Ergebnis zu kommen. Jedoch ist dies auch die kostenintensivste Variante, da sowohl ein leistungsfähiger Server, als auch die teureren 4K-Kameras benötigt werden.

An der Universität von Kapstadt wurde am Centre for Innovation in Learning and Teaching das System Track4K entwickelt, das anstatt PTZ-Kameras eine unbewegliche, und dadurch günstigere, 4K Überwachungskamera benutzt und genau dieses System umgesetzt hat (Marais, Marquard, Fitzhenry, Hahn & Khtieb, 2016). An der Universität Stuttgart wäre der Einsatz dieser günstigeren Überwachungskameras jedoch aufgrund der Datenschutzbestimmungen nicht möglich (siehe Kapitel 2.2).

Mit 4K-Kameras kann jedoch nicht unbedingt eine höhere Detailauflösung erreicht werden. Diese besitzen zwar die doppelte horizontale und vertikale Auflösung einer HD-Kamera, doch durch den optischen Zoom einer HD-Kamera kann ein weitaus größerer Vergrößerungsfaktor erreicht werden. Außerdem bringt der Einsatz von 4K-Kameras durch die größere Bandbreite und die Probleme bei den Nutzern einige Nachteile mit sich. Deshalb wurde in diesem Projekt die erstgenannte Variante verfolgt. Hierfür ist außer den beiden HD-Kameras ein Server mit einer passenden Software nötig. Durch die Universität Stuttgart wurde hierfür die Software LectureSight vorgegeben, die sowohl das Tracking als auch die Kamerasteuerung übernimmt. Warum sich die Universität für LectureSight entschieden hat, wird im folgenden Kapitel erläutert.

4.2 LectureSight

LectureSight ist eine Open Source Software, die in Java geschrieben und ursprünglich von Benjamin Wulff entwickelt wurde (Wulff, 2012). LectureSight arbeitet mit dem OSGi Framework, nutzt OpenCL, um die aufwändige Bildanalyse auf der Grafikkarte auszuführen, und verwendet Maven als Build-Umgebung. Was diese drei Begriffe im Einzelnen bedeuten, wird in den nachfolgenden Kapiteln genauer erläutert.

LectureSight unterstützt sämtliche für das Tracking notwendigen Funktionen und wird bereits an mehreren Universitäten, unter anderem an der Universität Osnabrück, der Universität von Manchester und der Universität von Kapstadt, in Verbindung mit Opencast eingesetzt (Kleinefeld, Lehmann & Ottow).

LectureSight ist nicht die einzige Software, die sich mit Tracking beschäftigt. Speziell für Vorlesungen gibt es bereits kommerzielle Software, die in der Regel sehr teuer ist (Wulff et al., 2014, S. 186). Aufgrund positiver Erfahrungen der anderen Universitäten und weil LectureSight eine Open Source Software ist, hat sich die Universität Stuttgart dazu entschlossen ebenfalls LectureSight einzusetzen.

Bisher konnte LectureSight nur über VISCA mit der Kamera kommunizieren und es existierte nur eine Standalone-Version. Das bedeutet, dass jeder Hörsaal, der mit Kamera-Tracking ausgerüstet werden sollte, einen eigenen Computer haben müsste. Alleine aufgrund der Vielzahl an Hörsälen und durch den beschränkten Platz gerade in kleineren Hörsälen wäre die Umsetzung an der Universität Stuttgart nicht möglich gewesen.

Daher hat die Universität Stuttgart bereits vor Beginn dieser Arbeit den Entwickler damit beauftragt, die Software weiter zu entwickeln und die gewünschten Features, wie VISCA over IP und eine Serverversion mit mehreren Instanzen zu implementieren. Durch die Verwendung des Internet Protokolls ist es möglich, einen zentralen Server zu verwenden, der für das Kamera-Tracking zuständig ist und somit, im Gegensatz zur Verwendung jeweils eines Servers pro Hörsaals, Geld einzusparen.

Bei der Programmierung gab es jedoch Verzögerungen, sodass die Implementierung von VISCA over IP nur in einer Vorabversion geliefert wurde und die Serverversion bis zum Abschluss dieser Arbeit noch nicht fertiggestellt werden konnte.

4.2.1 OSGi

Eine Möglichkeit um komplexe Software zu strukturieren ist die Modularisierung. Das bedeutet, dass Software in „weitestgehend autonome Softwareeinheiten (Module)" (Schmidt-Casdorff & Vogel, 2009, S. 10) aufgeteilt wird. Dies bietet den Vorteil, dass die einzelnen Module im Anschluss unabhängig voneinander entwickelt werden können. In einem Komponentenmodell ist definiert, welche Eigenschaften die Module besitzen. Sie müssen

- Schnittstellen veröffentlichen, über die auf sie zugegriffen werden kann
- Abhängigkeiten zu anderen Komponenten definieren
- klar identifizierbar sein, sich also beispielsweise in Namen, Version und Eigenschaften unterscheiden

Es können unterschiedliche Versionen einer Komponente im gleichen Framework installiert sein und Komponenten können unabhängig von anderen Komponenten (de-)installiert werden. Die Laufzeitumgebung (das Komponentenframework) entfernt beziehungsweise installiert Komponenten und verantwortet die Abhängigkeiten zu anderen Komponenten (Schmidt-Casdorff & Vogel, 2009, S. 10–11).

Es ist jedoch zum jetzigen Stand nicht möglich, dieses Modulkonzept mit Java selbst umzusetzen (Schmidt-Casdorff & Vogel, 2009, S. 11). Daher wird zur Umsetzung eine zusätzliche Plattform benötigt, die sogenannte Open Services Gateway initiative (OSGi).

OSGi erfüllt sämtliche zuvor genannten Kriterien des Komponentenmodells. Die einzelnen Module werden hier Bundles genannt, man kann diese zur Laufzeit installieren und deinstallieren – dies nennt sich Bundle Life Cycle (Schmidt-Casdorff & Vogel, 2009, S. 12).

Jedes Bundle besitzt eine Manifest-Datei, in der, abgesehen vom Namen und der Versionsnummer, auch die Abhängigkeiten zu anderen Bundles definiert sind. Hierin werden nicht nur die Namen der benötigten Bundles genannt, sondern auch die gewünschte Version. Damit kann das Framework das gewünschte Package bereitstellen (Schmidt-Casdorff & Vogel, 2009, S. 17–19).

Bei LectureSight handelt es sich um eine OSGi-Anwendung, die Apache Felix als OSGi-Framework benutzt (Wulff & Fecke, 2013, S. 6). Diese besteht aus diversen Bundles, wie

beispielsweise die für die Universität Stuttgart programmierte VISCA over IP Schnittstelle, die beim Build von LectureSight ausgewählt werden können. Wie dieser Build genau abläuft wird im folgenden Kapitel beschreiben.

4.2.2 Apache Maven

Maven ist ein Softwaretool der Apache Software Foundation für den automatisierten Build und die Verwaltung von Java Projekten (The Apache Software Foundation, 2016b).

In der POM (Project Object Model), der zentralen Konfigurationsdatei, sind sämtliche Einstellungen hinterlegt. Die Datei pom.xml enthält beispielsweise das Verzeichnis „target", in das der Build gespeichert wird, das „sourceDirectory", in welchem der Quellcode liegt und das „testSourceDirectory", in welches die Testdateien gespeichert werden. Als Mindestanforderungen muss diese Datei folgenden Parameter enthalten (The Apache Software Foundation, 2016a):

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1</version>
</project>
```

Die „modelVersion" gibt die POM-Version an; hier wird aktuell nur Version 4.0.0 unterstützt (The Apache Software Foundation, 2016c). Die „groupId" ist eine eindeutige Bezeichnung einer Organisation, die „artifactId" ist der Name des Projekts und „version" die Versionsnummer. Zusammen ergeben diese drei Felder den sogenannten „fully qualified artifact name" (The Apache Software Foundation, 2016a).

Viele Projekte hängen von anderen ab und sind ohne diese nicht lauffähig. Maven kann diese Abhängigkeiten, die hier Dependencies genannt werden, verwalten. Beim Build werden alle Abhängigkeiten aus dem jeweiligen Repository heruntergeladen, wobei auch transitive Abhängigkeiten aufgelöst werden (The Apache Software Foundation, 2016c). Das bedeutet: Wenn ein Projekt C ein anderes Projekt B benötigt und dieses von Projekt A abhängig ist, werden sowohl Projekt A als auch Projekt B automatisch heruntergeladen. Ein Repository ist ein spezielles Verzeichnis, das häufig für die Speicherung

von Software genutzt wird, und neben Beschreibung der Software oftmals auch eine Versionskontrolle beinhaltet.

Die Dependencies werden ebenfalls in der POM gespeichert und sind darin mit den folgenden Eigenschaften definiert (The Apache Software Foundation, 2016c):

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.0</version>
    <type>jar</type>
    <scope>test</scope>
    <optional>true</optional>
  </dependency>
...
</dependencies>
```

Eine POM kann ihre Eigenschaften auch an Untermodule vererben, indem diese die übergeordnete POM als „parent" festlegen (The Apache Software Foundation, 2016a).

Nach der Erstellung einer POM kann nun ein Build-Prozess gestartet werden. Dabei gibt es drei verschiedene Varianten, sogenannte Lifecycles: Default, Clean und Site (Porter, 2016). Der Default Lifecycle besteht aus den Phasen „validate", „compile", „test", „package", „verify", „install" und „deploy" (Porter, 2016). Die erste Phase kontrolliert, ob alle Informationen verfügbar sind und das Projekt korrekt definiert ist. Anschließend wird der Quellcode kompiliert. Das kompilierte Objekt wird nun getestet und, falls die Tests erfolgreich waren, beispielsweise in eine JAR-Datei gepackt. Im Anschluss daran folgt ein Integrationstest und die Installation in das lokale Repository, bevor im letzten Schritt das finale Package in das externe Repository hochgeladen wird (Porter, 2016).

4.2.3 OpenCL

LectureSight arbeitet mit dem Framework OpenCL in Version 1.1, das unter anderem dafür zuständig ist, auf modernen Grafikkarten auch nichtgraphische Berechnungen auszuführen.

Die Open Computing Language (OpenCL) entstand aufgrund einer hohen Heterogenität im Bereich der Prozessoren. Durch die verschiedensten Arten von Prozessoren wird die Entwicklung von Software ungemein erschwert, wenn für all diese effiziente Software entwickelt werden soll (Gaster, Howes, Kaeli, Mistry & Schaa, 2013, S. 1).

OpenCL ist ein offener, lizenzfreier Standard, der von der Khronos Group verwaltet wird (The Khronos Group, 2016) und am 9.12.2008 in Version 1.0 veröffentlicht wurde (The Khronos Group, 2008).

Der OpenCL-Standard beinhaltet vier Modelle, das „platform model", das „execution model", das „memory model" und das „programming model" (Gaster et al., 2013, S. 15–16).

Im ersten Teil des Standards, dem Plattform-Modell, wird beschrieben, dass ein OpenCL-System aus einem Host besteht, der eines oder mehrere Geräte hat, die OpenCL C Code ausführen können, sogenannte Devices (Gaster et al., 2013, S. 16). Ein Device hat eine Anzahl an unabhängigen Recheneinheiten, sogenannten Compute Units, welche wiederum in Processing Elements unterteilt sind. (Gaster et al., 2013, S. 20).

Ein Device kann eine CPU, GPU, DSP oder ein FPGA sein (Trevett, 2012), Compute Units (CUs) sind deren Kerne (Gaster et al., 2013, S. 121–122). Bei Grafikkarten sind dies die „Multithreaded SIMD Processors", die beispielsweise bei NVIDIA Grafikkarten „Streaming Multiprocessor" (SM) genannt werden. Auf den sogenannten Processing Elements, die bei Grafikkarten den „SIMD Lanes" entsprechen und bei NVIDIA „Thread Processor" heißen, wird der Code ausgeführt (Hennessy, Patterson & Asanović, 2012, S. I-90).

Das Execution-Modell legt fest, wie die Programme, die Kernel genannt werden, auf den Devices ausgeführt werden (Gaster et al., 2013, S. 16). Kernel werden in der Programmiersprache OpenCL C geschrieben, die auf der ISO C99 basiert (Gaster et al., 2013, S. 15).

Das Speicher-Modell („memory model") beschreibt verschiedene Speicher-Typen, die der Kernel benutzt (Gaster et al., 2013, S. 16). Dabei wird zwischen dem globalen Speicher (global memory), dem konstanten Speicher (constant memory), dem lokalen Speicher (local memory) und dem privaten Speicher (private memory) unterschieden, die in einem hierarchischen Modell angeordnet sind (Gaster et al., 2013, S. 29).

Der letzte Teil des Standards beschreibt mit dem Programming Model wie parallele Algorithmen in OpenCL abgebildet werden (Munshi, 2012, S. 24).

Die ersten OpenCL Grafikkartentreiber für OpenCL wurden von NVIDIA im September 2009 veröffentlicht (NVIDIA, 2009).

4.2.4 Funktionsweise

LectureSight ist in mehrere Bundles aufgeteilt, die verschiedene Aufgaben übernehmen.

Es gibt die Videoanalyse, die für das Tracking zuständig ist, und die im linken Bereich von Abbildung 17 beschrieben wird. Die Kommunikation mit der Kamera findet über die Bundles PTZ Service und PTZ Driver statt und zur Kamerasteuerung gibt es die Bundles Steering Worker und den Virtual Director. Zusätzlich gibt es die Möglichkeit mit Hilfe von Skripten die Bewegung der Kamera zu beeinflussen (Wulff et al., 2014, S. 188).

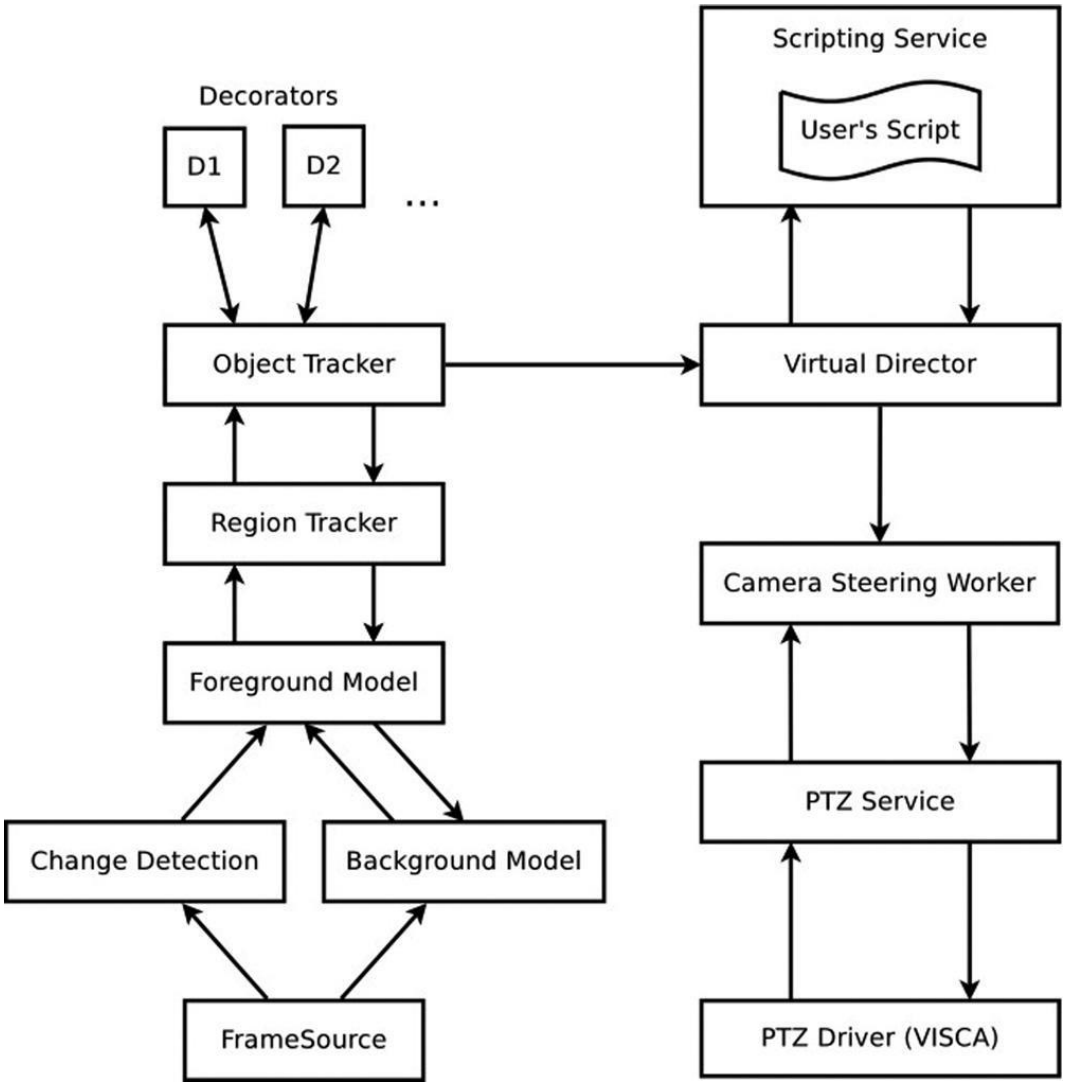

Abbildung 17: Softwarearchitektur von LectureSight (Wulff et al., 2014, S. 188)

Die Analyse besteht, wie in Abbildung 17 zu erkennen, aus mehreren Schritten. Zuerst wird auf dem Video sowohl eine Bewegungserkennung durchgeführt, als auch ein Modell des Hintergrunds erstellt (Wulff et al., 2014, S. 189). Wenn sich nun etwas im Bild bewegt, also eine Änderung zwischen mehreren Frames stattfindet, löst dies die Erstellung eines Vordergrundmodells aus (Wulff et al., 2014, S. 189). Die Änderung zwischen zwei Frames kann durch weiße Pixel im Fenster „change" in der Software dargestellt werden. Im Gegensatz zu anderer Software behält LectureSight das Vordergrundmodell über eine längere Zeit, um einer zeitlichen Dynamik entgegenzuwirken (Wulff et al., 2014, S. 189).

An dieser Stelle sollte angemerkt werden, dass sich die Softwarearchitektur mittlerweile geändert hat und aktuell ein neueres Bundle zur Analyse existiert, welches in dieser Arbeit verwendet wurde. In diesem sogenannten „Video Analysis Template" wird kein Background-Modell mehr erstellt, sondern zusammen mit der Bewegungserkennung werden Blobs und damit Objekte erkannt. Ein Blob ist dabei die Region eines Bildes, die sich beispielsweise im Kontrast vom Rest des Bildes unterscheidet (Lindeberg, 1993, S. 283). Außerdem gibt es nun Templates, was Bildteile sind, die als Objekt erkannt wurden und nach denen in zukünftigen Frames gesucht wird, um die Zuverlässigkeit des Trackings zu erhöhen (Wulff & Marquard, 2016c).

Im Anschluss an die initiale Analyse wird im Region Tracker analysiert, ob Objekte zu klein oder zu groß sind, um Menschen zu sein. Diese werden verworfen (Wulff et al., 2014, S. 189). Der Region Tracker kümmert sich außerdem um falsch erkannte Objekte, indem für jede Region die Anzahl an geänderten Pixeln gezählt wird. Falls diese Anzahl unter einem definierten Schwellwert liegt, werden die Objekte in dieser Region verworfen (Wulff et al., 2014, S. 189).

Mit dem Object Tracker ist es möglich einem Objekt zusätzlich sogenannte „Decorators" zuzuordnen. Diese speichern zusätzliche Informationen, wie zum Beispiel ein Histogramm eines Objekts. Sollte sich dieses nicht zu stark und zu schnell ändern, können damit verschiedene Objekte voneinander unterschieden werden (Wulff et al., 2014, S. 190). Mit den Decorators wurde außerdem versucht die Position des Kopfes zu speichern, jedoch geschah dies mit einer bereits in OpenCL vorhandenen Funktion. Bei steilen Hörsälen, bei denen die Übersichtskamera oft nur den Oberkörper eines Dozenten sieht, stößt diese Funktion jedoch an ihre Grenzen und funktioniert nicht mehr zuverlässig. Daher wird diese Funktion heutzutage nicht mehr in LectureSight verwendet (Wulff & Marquard, 2016c).

Die Steuerung der Kamera ist in zwei Bundles aufgeteilt. Es gibt den Camera Operator, der die Zielposition der Kamera berechnet und in Abbildung 17 Virtual Director genannt wird. Hierfür kann man zwischen dem „LectureSight pan-only Camera Operator", der ausschließlich Schwenkbewegungen ausführt und dem „Scripted Camera Operator", der die Möglichkeit bietet Skripte einzubinden, auswählen.

Das zweite für die Steuerung zuständige Bundle ist der Steering Worker, welcher sich um die Kamerabewegung kümmert. Dabei vergleicht er permanent die aktuelle Kameraposition mit der errechneten Zielposition des Camera Operators und entscheidet, wann sich die Kamera mit welcher Geschwindigkeit in welche Richtung bewegen soll. Es werden keine absoluten Positionen an die Kamera gesendet, sondern sie wird lediglich in eine Richtung gelenkt. Dabei werden beispielsweise kleine Bewegungen ignoriert und bei schnellen Schwenks wird kurz vor dem Erreichen des Ziels abgebremst (Wulff et al., 2014, S. 190).

Der PTZ Service stellt eine API zur Kamerasteuerung bereit. Dafür gibt es mehrere Driver, wie beispielsweise den VISCA Camera Driver, zur seriellen Steuerung von Sony-Kameras, oder den in dieser Arbeit verwendeten VISCA over IP Camera Driver.

4.3 Forschung

Durch die mit LectureSight möglich gewordene neue Einstellungsgröße „Tracking" werden nicht nur die Tafelanschriebe sichtbar, sondern auch die Gestik und Mimik des Dozenten (siehe Anhang A). Welche Auswirkungen diese „Sichtbarkeit von Gestik und Mimik [...] auf die Lernleistung und Motivation von Lernenden" (Rupp, Wulff & Hamborg, 2014, S. 5) hat, zeigt eine Studie von Lisa Rupp an der Universität Osnabrück.

In dieser Studie wurde die selbe Vorlesung mit zwei Kameras gleichzeitig aufgezeichnet. Die eine Kamera hat hierbei für die Kontrollgruppe eine Totale aufgezeichnet (siehe Abbildung 18), die andere Kamera wurde mit LectureSight gesteuert und hat für die Experimentalgruppe eine Halbtotale gefilmt (siehe Abbildung 19). Diese Videos wurden mittels Opencast Matterhorn zusammen mit der Präsentation bereitgestellt, wobei die Darstellung des Videoplayers nicht geändert werden konnte (Rupp et al., 2014, S. 11). Dieses Video konnten sich die Probanden am ersten Tag der Studie an einem Ort und zu einer Zeit ihrer Wahl ansehen. Am Tag darauf fand ein Leistungstest in Form einer Klausur statt (Rupp et al., 2014, S. 14).

Abbildung 18: Stimulusvideo Kontrollgruppe (Rupp et al., 2014, S. 12)

Abbildung 19: Stimulusvideo Experimentalgruppe (Rupp et al., 2014, S. 13)

Für den Begriff der Lernleistung, die hier unter anderem betrachtet wurde, wurden im Detail zwei Fälle unterschieden: zum einen die Recognition, als die „Wiedererkennung von Informationen" (Rupp et al., 2014, S. 6), zum anderen der Recall, als die „freie Reproduktion von Informationen" (Rupp et al., 2014, S. 6).

Die durchschnittliche Recognition, also die Punktzahl für die entsprechenden Aufgaben, war bei der Experimentalgruppe ca. 20 % höher als bei der Kontrollgruppe. Damit konnten sich Probanden, die das Tracking-Video gesehen hatten, besser an den Inhalt erinnern. Viel erstaunlicher ist jedoch, dass der durchschnittliche Recall, also somit auch das Verständnis, mit 150 % deutlich höher war (siehe Abbildung 20), obwohl die Recall-Aufgaben deutlich schwieriger zu beantworten sind als die Multiple-Choice Fragen zur Recognition (Wulff et al., 2014, S. 196).

Recognition

	N	*Mdn*	*U*	*z*	*p*
LectureSight	27	14,00	257,00	-2,849	0,004
Kontrollgruppe	30	11,67			

Recall

	N	*Mdn*	*U*	*z*	*p*
LectureSight	27	10	111,00	-4,907	0,000
Kontrollgruppe	30	4			

Abbildung 20: Lernleistung (Rupp et al., 2014, S. 15)

Die Studie kommt daher zu dem Ergebnis, dass man sich an „signifikant mehr Informationen erinnern [kann], wenn diese während des Lernprozesses von Gestik und Mimik begleitet präsentiert werden" (Rupp et al., 2014, S. 18).

Dieses Forschungsergebnis zeigt, dass Kamera-Tracking nicht nur eingesetzt werden sollte, um Tafelanschriebe besser erkennen zu können, sondern dass es auch aus pädagogischer Sicht durchaus sinnvoll ist.

5 Testaufbau

Im Rahmen des Projekts „Hörsäle 2020" wurden nicht nur bestehende Hörsäle umgebaut, sondern auch drei neue Testräume geschaffen. Diese sind im Gebäude des TIK/NFL, dem Allmandring 3a, untergebracht und entsprechen den Hörsälen der Kategorie S, M und XL. Damit wurde die perfekte Grundlage für einen ausführlichen Test der Software LectureSight und auch für den nachfolgend nötigen Umbau der Medientechnik geschaffen.

Der für diesen Test verwendete Demohörsaal XL besteht im Wesentlichen aus den folgenden Komponenten:

- zwei Beamer: Canon WUX400ST
- zwei PTZ-Kameras: Sony SRG-300H sowie LevelOne FCS-6020
- eine zentrale Video- und Audiomatrix: Crestron DM-MD8x8
- eine zentrale Steuerung: Crestron CP3
- ein Touchpanel: Crestron TSW-1052
- eine Audio-DSP: Yamaha DME64
- ein Mikro-Empfänger: Shure ULxD4D

Für den praktischen Test von LectureSight wird zusätzlich noch ein Computer benötigt, welches in diesem Fall ein FUJITSU Desktop ESPRIMO P920 E85+ ist.

Für den Einsatz von LectureSight in mehreren Hörsälen soll LectureSight auf einem dedizierten Server ausgeführt werden.

5.1 Übersicht

Es gibt grundsätzlich drei Bildquellen. Zum einen ein beliebiges Gerät des Dozenten, das an die Mediensteuerung angeschlossen wird und zur Aufnahme von dieser an einen H.264-Encoder gesendet wird. Dieser erzeugt einen Videostream und sendet ihn an den Capture Agent. Das gleiche gilt auch für die PTZ-Kamera (1), die ebenfalls an der Mediensteuerung angeschlossen ist und deren Videosignal auch an einen Encoder gesendet wird (siehe Abbildung 21).

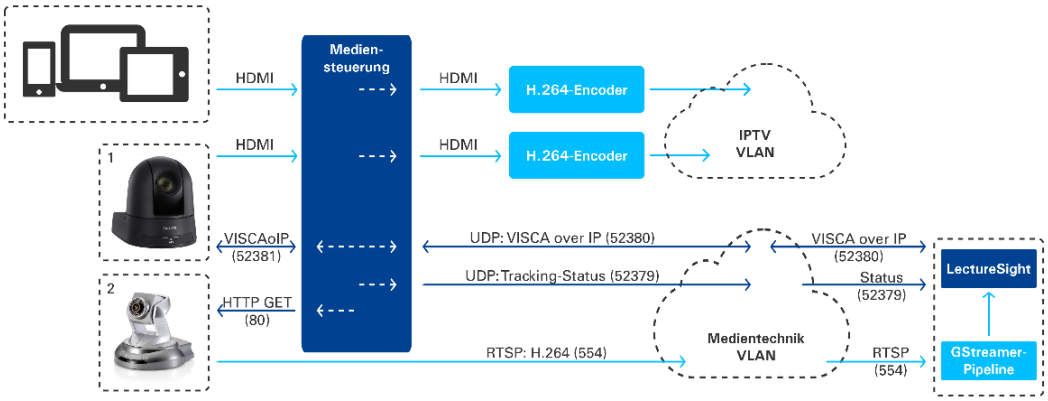

Abbildung 21: Übersicht zum Testaufbau im Demohörsaal XL (Eigene Darstellung, Daten entnommen aus Sony, 2014, S. 1 und LevelOne, 2014, S. 1)

Die dritte Bildquelle ist eine PT-Kamera (2), die einen H.264-Stream erzeugt und diesen über RTSP dem LectureSight-Server bereitstellt. Die Mediensteuerung sendet Befehle an die beiden Kameras, um diese auf verschiedene Positionen zu fahren; zur Sony-Kamera (1) mit VISCA over IP und zur Übersichtskamera (2) mittels HTTP GET.

Damit LectureSight direkt vom Dozenten gestartet werden kann, sendet die Mediensteuerung über UDP den aktuellen Tracking-Status zu LectureSight. Dieser ändert sich von „Tracking Off" zu „Tracking On", sobald der Dozent auf dem Touchpanel die Einstellungsgröße „Tracking" auswählt, die die bisherige Größe „Redner" ersetzt. In der Serverversion soll LectureSight später, sobald es den Status „Tracking On" erhält, eine neue Instanz für den jeweiligen Hörsaal starten.

Zum Schluss bleibt noch die Kamerasteuerung, die es sowohl durch die Mediensteuerung gibt, um beispielsweise Presets zu speichern, aber auch durch LectureSight für das automatische Kamera-Tracking.

5.2 Grafikkarte

Für den Echtzeit-Betrieb von LectureSight, also für Live-Video, wird empfohlen, eine Grafikkarte mit mindestens sechs OpenCL Compute Units zu verwenden (Wulff & Marquard, 2016a). Diese CUs könnten, wie bereits in Kapitel 4.2.3 beschrieben, Kerne bei Mehrkernprozessoren sein oder in diesem Fall Streaming Multiprocessors (SMs) einer NVIDIA Grafikkarte.

Zu Beginn wurde eine bereits vorhandene Grafikkarte verwendet, die NVIDIA GeForce 9400 GT, die jedoch nur 16 CUDA-Kerne hat (NVIDIA, 2016b). CUDA ist eine Architektur

von NVIDIA, die die Berechnung auf Grafikkarten erlaubt. CUDA-Kerne sind jedoch nicht mit OpenCL Compute Units gleichzusetzen. Eine CU entspricht einem SM und dieser besteht aus mehreren CUDA-Kernen.

LectureSight geht in seiner Dokumentation davon aus, dass eine CU immer aus 8 CUDA-Kernen besteht: „For NVIDIA GPUs, the number of CUDA units divided by eight yields the number of OpenCL compute units." (Wulff & Marquard, 2016a). Für die bereits erwähnte GeForce 9400 GT mag diese Information noch gelten; hier sind bei 16 CUDA-Kernen 2 SMs, also 2 CUs vorhanden. Doch bei heutigen Architekturen ist die Anzahl an CUDA-Kernen pro SM deutlich gestiegen.

Von Wulff und Marquard (2016a) wird unter anderem die NVIDIA GeForce GTX 750 Ti empfohlen, da sie bereits mit LectureSight getestet wurde und gut funktioniert. Diese hat laut deviceQuery des CUDA Toolkits (Version 8.0) 5 SMs (siehe Abbildung 22). Hier kommen die sogenannten SMMs (Maxwell) der ersten Generation (NVIDIA, 21.02.2014/2014) zum Einsatz, die pro SM vier separate Processing Blocks mit je 32 CUDA-Kernen haben, also insgesamt 182 CUDA-Kerne pro SM (NVIDIA, 2014, S. 6).

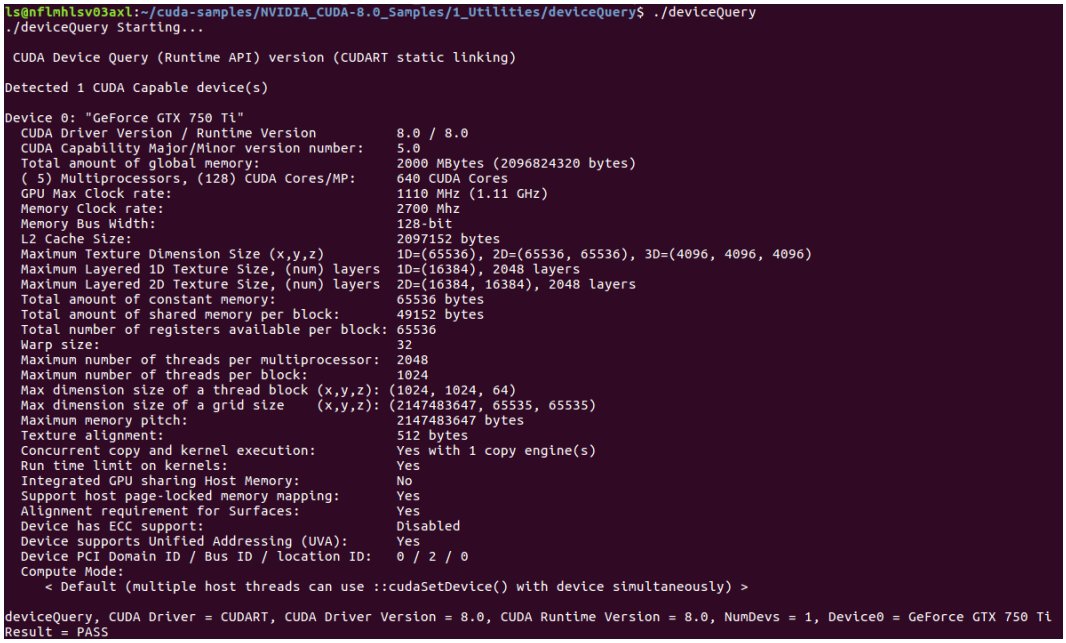

Abbildung 22: deviceQuery der NVIDIA GeForce GTX 750 Ti (Eigene Darstellung)

Für den Server, auf dem die finale Installation laufen soll, werden NVIDIA K40 mit 2880 CUDA-Kernen verwendet (NVIDIA, 2013, S. 2). Auf dieser Grafikkarte, die speziell für High Performance Computing (HPC) entwickelt wurde, befinden sich SMs der Kepler-Architektur, die nun SMX (Next Generation Streaming Multiprocessors) genannt werden. Diese bestehen aus je 192 CUDA-Kernen (NVIDIA, 2012, S. 1). Daher besitzt die K40 also

15 CUs. Die aktuell „leistungsfähigste Grafikkarte der Welt" (NVIDIA, 2016a) heißt K80 und enthält gleich zwei Grafikprozessoren mit je 13 SMX, also insgesamt 26 CUs und 4992 CUDA-Kerne (NVIDIA, 2015, S. 2).

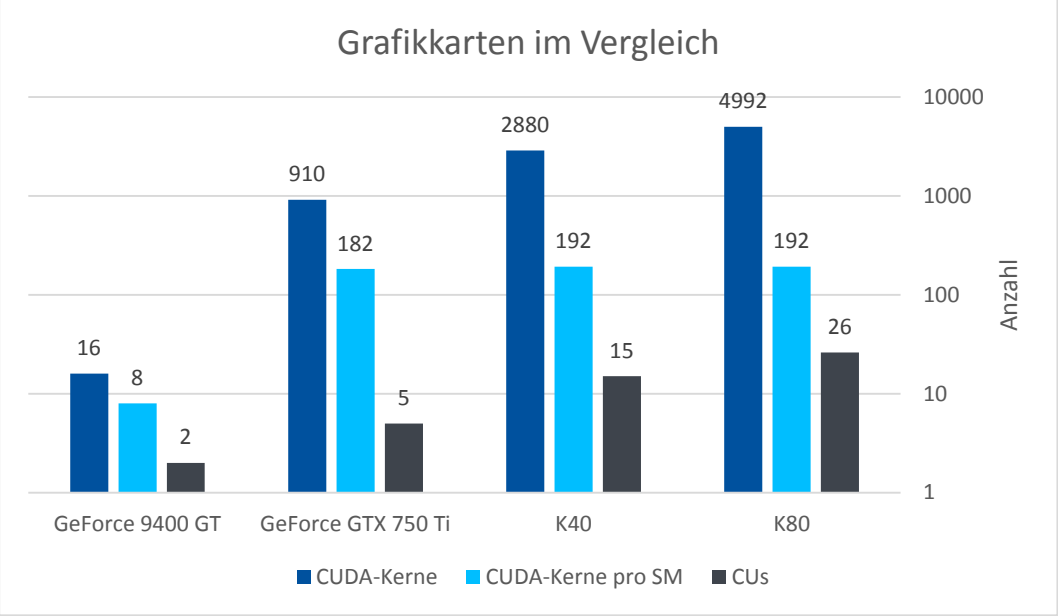

Abbildung 23: Grafikkarten im Vergleich (Eigene Darstellung, Daten entnommen aus NVIDIA, 2016b, NVIDIA, 2014, S. 6, NVIDIA, 2013, S. 2, NVIDIA, 2012, S. 1 und NVIDIA, 2015, S. 2)

LectureSight empfiehlt eine GPU mit mindestens sechs CUs und 512 MB Grafikspeicher zu verwenden (Wulff & Marquard, 2016a). Obwohl bei der für den Testaufbau verwendete GeForce 750 Ti mit nur fünf CUs und 2048 MB Grafikspeicher von dieser Empfehlung abgewichen wurde, funktioniert diese einwandfrei. Wie bereits erwähnt geht die Dokumentation von LectureSight noch von veralteten GPUs aus, die nur 8 CUDA-Kerne pro SM oder CU haben. Daher war es für die GeForce 750 Ti kein Problem für eine flüssig laufende Software zu sorgen.

5.3 LectureSight Konfiguration

5.3.1 Installation

Softwareseitig wurde auf dem PC Ubuntu 16.04.1 LTS verwendet.

Da der Computer im HSMEDIA-VLAN (Medientechnik) ist und somit keinen Internetzugriff hat, musste für die nachfolgenden Installationen der Ubuntu-Mirror der Universität Stuttgart unter /etc/apt/sources.list eintragen werden, der es ermöglicht, über die Paketverwaltung apt-get die nötige Software zu installieren:

- Oracle Java 8 (oracle-java8-installer), welches zusammen mit
- GStreamer (libgstreamer0.10-0) für LectureSight benötigt wird. GStreamer ist ein freies Multimedia-Framework, das heißt mit dieser Software können beispielsweise Videodatenströme bearbeitet werden.
- Git, als Versionsverwaltung
- Maven, zum Build von LectureSight

Zusätzlich wurde als Entwicklungsumgebung Eclipse installiert. Anschließend wurde mit Git das Repository von LectureSight, das unter anderem den Source Code enthält, geklont und das Programm mit Maven gebaut. Hierbei muss einerseits beachtet werden, in der pom.xml alle benötigten Bundles auszuwählen – zum Beispiel zur Nutzung von GStreamer als Videoquelle „lecturesight-framesource-gst" anstelle von „lecturesight-framesource-v4l" – und mit dem Kommandozeilenparameter „-DdeployTo" den korrekten Dateipfad zu den Bundles anzugeben: „runtime/bundles/application/".

Nun können in der Konfigurationsdatei (/runtime/conf/lecturesight.properties) diverse Parameter gesetzt werden. Am wichtigsten ist hier, dass zu Beginn eine Videoquelle angegeben wird, die LectureSight zur Analyse verwenden kann. Da dies im konkreten Fall keine Webcam ist, sondern ein Videostream, wird das Programm GStreamer benötigt, welches den Stream empfängt und LectureSight bereitstellt.

5.3.2 GStreamer

Die GStreamer Pipeline, die in der Dokumentation von LectureSight (Wulff & Marquard, 2014) beispielhaft beschrieben wird, konnte nicht eingesetzt werden, da diese ein anderes Setup voraussetzt. Deshalb wird in diesem Kapitel die hier verwendete Pipeline und deren Funktionen näher erläutert.

Die finale Konfigurationsdatei beinhaltet als Videoquelle die folgende GStreamer-Pipeline:

```
cv.lecturesight.framesource.input.mrl=
gst://rtspsrc location=rtsp://root:admin@172.25.107.10/live1.sdp !
decodebin !
videocrop top=300 left=690 right=590 bottom=300 !
ffmpegcolorspace !
gaussianblur sigma=3 !
ffmpegcolorspace ! video/x-raw-rgb !
autovideoconvert
```

Eine Pipeline setzt sich aus einzelnen Befehlen zusammen, die mit Ausrufezeichen verknüpft werden. In der oben genannten Pipeline gibt es die folgenden Elemente:

- „cv.lecturesight.framesource.input.mrl" ist der Parameter für die Videoquelle, die LectureSight zur Analyse und Bewegungserkennung nutzt.
- „rtspsrc" ist eine Bildquelle über RTSP, also ein Videostream. Die „location" gibt hierbei die URL zur Kamera an.
- „decodebin" erkennt, welches Format am Einfang anliegt und wählt automatisch den passenden Decoder.
- „videocrop" schneidet das Video zu und erstellt in diesem Fall aus dem Ausgangsvideo mit einer Auflösung von 1920 x 1080 Pixeln ein neues Video mit einer Auflösung von 640 x 480 Pixeln.
- „ffmpegcolorspace" ändert den Farbraum des Videos.
- „gaussianblur" fügt einen Gaußschen Weichzeichner hinzu.
- „autovideoconvert" wählt automatisch den korrekten Color Space Converter.

Trotz der neuen Grafikkarte ist das System noch nicht leistungsstark genug, um HD-Video mit einer Framerate von 30 Hz zu verarbeiten. Daher wurde das Video bei gleichbleibender Framerate auf 640 x 480 Pixel zugeschnitten, um die Datenmenge, die LectureSight verarbeiten muss, stark zu reduzieren. Da die Übersichtskamera ein Weitwinkel-Objektiv hat und somit einen weit größeren Bildausschnitt zur Verfügung stellt, als benötigt, führte dies in diesem Fall ohne Qualitätsverlust zum Erfolg. Des Weiteren wurde darauf geachtet, dass der benötigte Bildausschnitt in der Bildmitte ist, da die Kameraoptik dort die geringste Verzerrung aufweist.

Ein Problem, das später beim Tracking entstand, war, dass das Video viele Artefakte hat, die als Trackingpunkte erkannt wurden. Dieses Problem konnte gelöst werden, indem der Stream von MJPEG auf H.264 umgestellt wurde, da dieser Codec bei der hier eingesetzten Kamera deutlich weniger Artefakte erzeugt. Es gibt jedoch nicht nur Kompressionsartefakte, sondern auch ein Bildrauschen der Kamera selbst. Es wurde deshalb ein Gaußscher Weichzeichner mit geringer Varianz, also mit geringer Wirkung, angewendet. Somit verliert das Bild zwar etwas Detailgenauigkeit, doch dies ist für die Bewegungserkennung irrelevant und das Bildrauschen sowie die zufällig auftretenden Fehler verschwinden.

Da das GStreamer-Plugin „gaussianblur" nicht mehr über die Ubuntu-Paketverwaltung erhältlich ist, wurden die Source Packages für „gst-plugins-bad0.10" und einige weitere

Dependencies direkt installiert. Da LectureSight jedoch noch mit der alten Version GStreamer 0.10 arbeitet, ist es unwahrscheinlich, dass es für diese Plugins weitere Updates geben wird und es ist somit unproblematisch diese Plugins nicht über die Paketverwaltung zu installieren.

5.3.3 Kamerasteuerung

Wie bereits in Kapitel 4 erwähnt, gab es beim VISCA over IP-Modul, das zur Kamerasteuerung benutzt wird, einige Probleme. Beispielsweise war hier noch nicht vorgesehen, die IP-Adresse der Kamera einzugeben und der Port zur Steuerung war fest im Quellcode verankert. Diese beiden Parameter wurden in die Konfigurationsdatei ausgelagert, sodass hier nun die beiden folgenden Werte definiert werden können:

com.wulff.lecturesight.viscaoverip.device.ip=mm-steuerung-v03a37xl-1.rus.loc
com.wulff.lecturesight.viscaoverip.device.port=52380

Außerdem muss vor dem Build im Ordner Profiles ein Kameraprofil angelegt werden, das unter anderem die Model-ID und einige physikalische Eigenschaften, wie zum Beispiel die Maximalgeschwindigkeit oder die Schwenk- und Neige-Grenzwerte, enthält.

5.3.4 Kamerakalibrierung

Der nächste Schritt, nachdem LectureSight nun das Videosignal der Übersichtskamera sieht, ist die Software mit der Tracking-Kamera zu kalibrieren.

LectureSight besitzt intern ein normalisiertes Koordinatensystem. Dieses hat den folgenden Definitions- und Wertebereich:

$$D = [-1; 1]$$
$$W = [-1; 1]$$

Die Übersichtskamera wird automatisch in dieses Koordinatensystem gelegt. Das heißt, dass die Bildkanten nun jeweils die Werte 1 beziehungsweise -1 haben. Dies definiert den Aktionsradius des Trackings. Sämtliche Punkte, die außerhalb des Koordinatensystems liegen, können nicht angefahren werden, da sie nicht von der Übersichtskamera gefilmt werden.

Wenn jetzt Objekte erkannt werden, wird ein Punkt im Koordinatensystem gesetzt. Damit die Kamera auch den richtigen Punkt anfährt, muss diese korrekt kalibriert werden.

Das Ziel ist es, vier Positionen zu erhalten, die den Grenzen der Übersichtskamera entsprechen. Dazu wird in LectureSight im Modul „PTZ Camera Control" die aktuelle Position der Kamera mit ihren internen Koordinaten angezeigt. Diese aktuelle Position entspricht der Bildmitte.

Für die nächsten Schritte muss das Tracking zunächst deaktiviert werden, da sich ansonsten, falls sich etwas im Bild bewegt, die Kamera bewegt. Hierzu ist es am einfachsten im Modul „Scene Profile Editor" das komplette Bild der Übersichtskamera zu ignorieren. Im nächsten Kapitel wird dieser Editor nochmals detaillierter erklärt.

Außerdem sollten jetzt die Videoquellen der beiden Kameras dargestellt werden. Die Übersichtskamera besitzt eine Weboberfläche und den Stream der PTZ-Kamera kann beispielsweise mit dem VLC Media Player angezeigt werden. Um die Bildmitte hier besser erkennen zu können, ist es möglich, unter „Werkzeuge, Effekte und Filter, Videoeffekte, Overlay, Text hinzufügen" ein „+" einzugeben und als Position „Zentriert" anzugeben (siehe Abbildung 24). Somit ist ein Fadenkreuz in der Bildmitte vorhanden und die Kamera kann nun durch Klicken in das Koordinatensystem im Modul „PTZ Camera Control" bewegt werden.

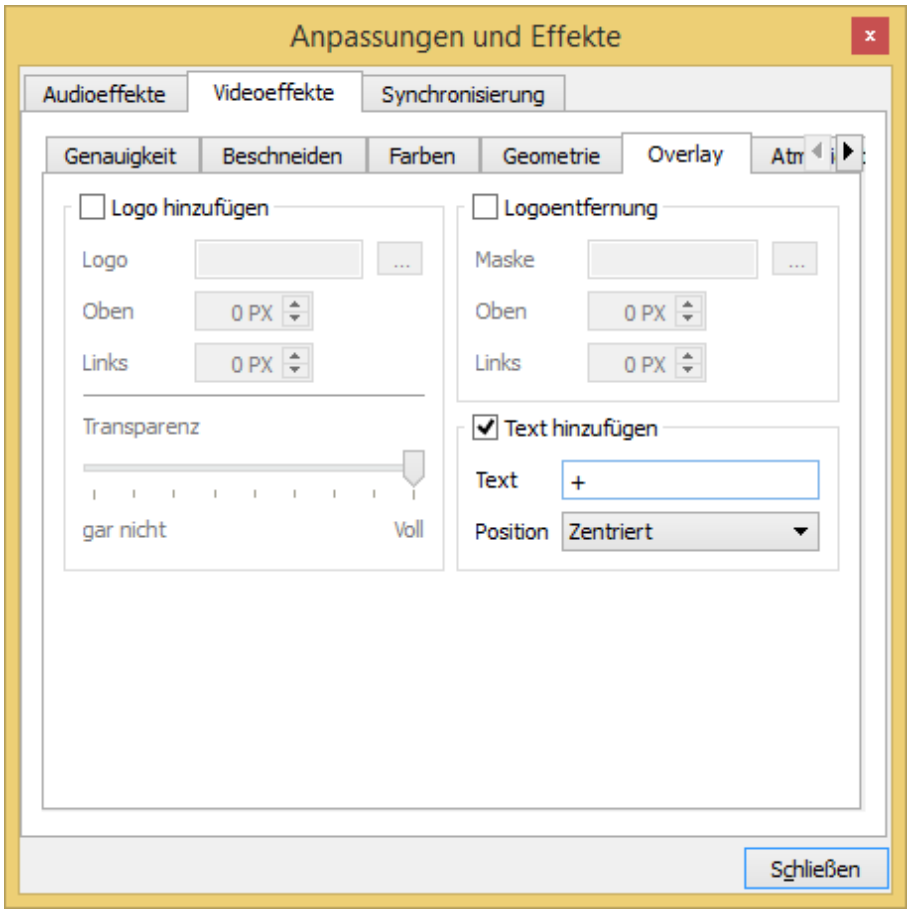

Abbildung 24: Overlay im VLC Media Player (Eigene Darstellung)

Zuerst wird die Kamera so weit nach links gefahren, bis der Bildmittelpunkt auf das gleiche Objekt zeigt, das in der Übersichtskamera am linken Bildrand ist. Bevor dies für die verbleibenden drei Seiten wiederholt wird, muss die x-Koordinate dieser Position notiert werden.

Anschließend werden die x- und y-Werte als Scene Limits in der Konfigurationsdateige-speichert. Hier sind das die folgenden Werte:

```
cv.lecturesight.ptz.steering.worker.relativemove.scene.limit.left=-450
cv.lecturesight.ptz.steering.worker.relativemove.scene.limit.right=950
cv.lecturesight.ptz.steering.worker.relativemove.scene.limit.top=210
cv.lecturesight.ptz.steering.worker.relativemove.scene.limit.bottom=-1000
```

Die Dokumentation ist veraltet (Wulff, 2013), da diese beispielsweise falsche Namen für die Konfigurationsdatei vorgibt und auch die dort beschriebene Vorgehensweise, um das Tracking zu deaktivieren, nicht funktioniert.

5.3.5 Szenenprofil

Im bereits erwähnten Scene Profile Editor ist es nun möglich, den Bereich, in dem getrackt werden soll, einzuschränken. Dazu wird ein neues Profil erstellt, in welchem mit Rechtecken direkt auf dem Bild der Übersichtskamera zunächst „ignore"-Bereiche definiert werden, die rote Rechtecke erzeugen und beim Tracking ignoriert werden (siehe Abbildung 25). Das sorgt dafür, dass das Videobild, das am Eingang anliegt, in den gewählten Bereichen schwarze Pixel erhält und somit dort nicht analysiert wird. Dies ist besonders für den Projektionsbereich wichtig, da es dort ein bewegtes Bild gibt, das nicht zur Analyse verwendet werden soll. Ebenso sollte der Sitzbereich der Studenten im Hörsaal sowohl wegen möglicher Störungen des Trackings als auch aus Datenschutzgründen ignoriert werden.

Über das Lineal-Symbol ist es möglich, die ungefähren Umrisse einer Person mittels eines blauen Rechtecks darzustellen. Dies hilft der Analyse zu erkennen, ob es sich bei einem erkannten Objekt um eine Person handelt oder nicht (Wulff, 2015). Der Trigger-Bereich, der mit dem Schalter-Symbol erzeugt werden kann, löst, sobald sich ein Objekt in diesen Bereich bewegt, einen Event aus, der mittels eines Skriptes verarbeitet werden kann (Wulff, 2015).

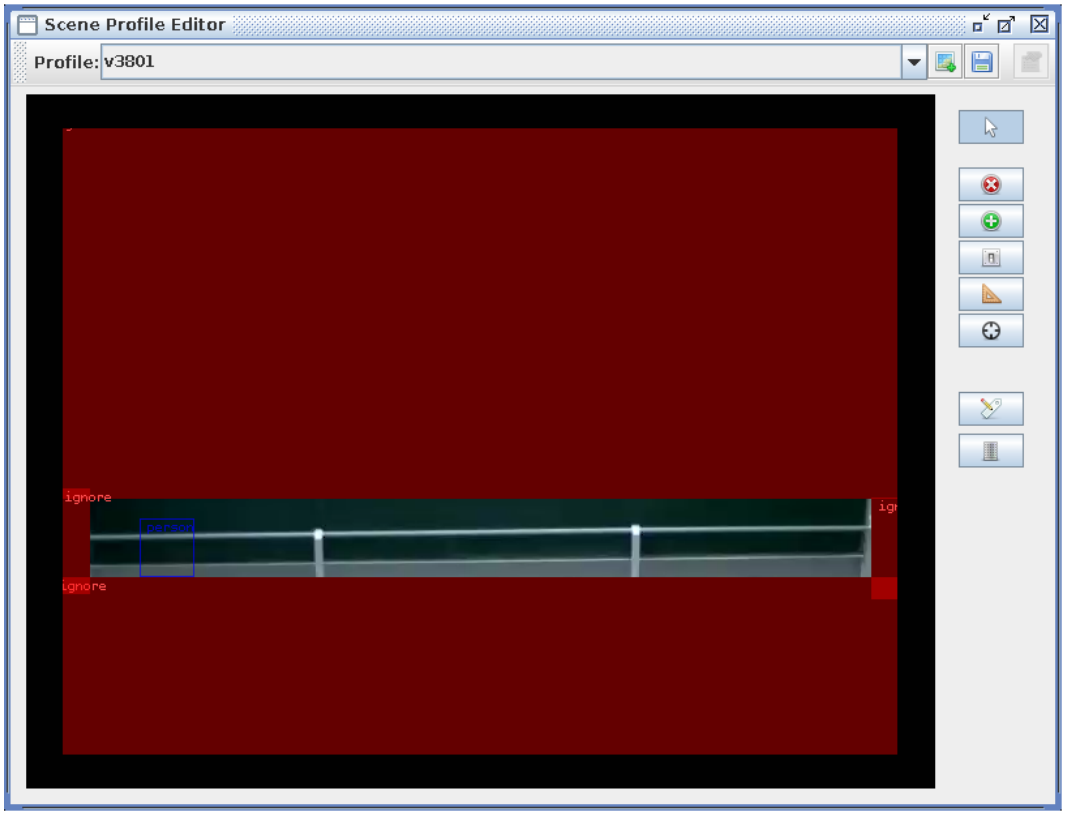

Abbildung 25: LectureSight Scene Profile Editor (Eigene Darstellung)

Beim Speichern eines Szenenprofils wird ein Name vergeben, der wiederum in der Konfiguration eingespeichert werden muss, damit das gewählte Profil beim Start aktiviert wird:

cv.lecturesight.profile.manager.active.profile=demoxl

5.3.6 Test des Trackings

Um die Software vollständig einzurichten fand zunächst ein Test im Demohörsaal XL statt. Da hier jedoch der Abstand zwischen Kamera und Dozent gering ist, unterscheidet sich dieser Raum sehr stark von den Hörsälen, in denen das System am Ende eingesetzt werden soll. Es musste daher ein Test in einem richtigen Hörsaal durchgeführt werden. Für diesen Test wurde der Hörsaal V38.01 ausgewählt, da dieser bereits im Rahmen des Projekts „Hörsäle 2020" umgebaut wurde und häufig für Vorlesungsaufzeichnungen genutzt wird.

Für die finale Konfiguration und Feinjustierung wurden zunächst die wichtigsten Konfigurationsparameter untersucht. Diese sind größtenteils im Quellcode (Wulff & Marquard, 2016b) dokumentiert. Für fast jedes Modul gibt es eine Reihe an Einstellungsmöglichkeiten.

Um zu kleine Objekte von der Analyse auszuschließen kann die Mindestgröße (in Pixeln) eines gültigen Blobs beispielsweise auf 50 gesetzt werden:

cv.lecturesight.blobfinder.blobsize.min=50

Dieser Wert hat sich im Hörsaal V38.01 als gut erwiesen, da er ungefähr der Größe einer Person entspricht. Ein weiterer Parameter, der gesetzt werden kann, ist die maximale Größe (blobsize.max) eines gültigen Blobs. Beim Test im Hörsaal lag dieser bei einem Wert von 100 und verhindert, dass die Tafel getrackt wurde, sobald sich diese bewegt.

Es gibt noch diverse weitere Parameter zur Videoanalyse, die jedoch bei diesen Tests auch in ihren Default-Werten gut funktioniert haben. Es kann beispielsweise festgelegt werden, ab wann ein Pixel als bewegt angesehen wird, wie viele Pixel pro Zelle geändert werden müssen, um diese zu aktivieren oder auch ab wann eine Menge an aktiven Zellen als Tracking-Ziel gesehen wird (Wulff & Marquard, 2016b). Eine Zelle ist dabei die Menge an Pixeln, in die ein Bild zur Analyse unterteilt wird.

Die Einstellungen des Camera Operators beinhalten die Startposition (cv.lecturesight.cameraoperator.panonly.pan bzw. tilt), den initialen Zoomwert und das „Target Timeout" (Wulff & Marquard, 2016b). Letzteres legt die Zeit fest, nach der ein Objekt nicht mehr getrackt wird, nachdem zuletzt eine Bewegung festgestellt wurde. Diese Arbeit verwendet den „LectureSight pan-only Camera Operator", der ausschließlich Schwenkbewegungen ausführt, da bei einer Vorlesung aus Datenschutzgründen nur der Bereich vor der Tafel gefilmt werden darf. Somit können weniger Fehler auftreten, als wenn ebenfalls eine Neigebewegung ausgeführt werden würde. Der zuvor bereits erwähnte Scripted Camera Operator könnte im Gegensatz dazu mittels JavaScript beispielsweise steuern, dass die Kamera den Zoom-Bereich vergrößert sobald mehr als ein Objekt erkannt wird. Alternativ könnte festgelegt werden, was passieren soll, wenn ein Trigger-Bereich betreten wird (Wulff, 2015). Dieses Modul wurde während dieser Arbeit nicht ausführlich getestet, jedoch wäre dies für die zukünftigen Installationen in weiteren Hörsälen eine interessante Erweiterung.

Der Steering Worker kann die Geschwindigkeit der Kamera ab einer definierten Entfernung zum Ziel reduzieren (cv.lecturesight.ptz.steering.worker.relativemove.move.alpha.x bzw. y) und es kann die Genauigkeit angegeben werden, mit der das Ziel erreicht werden soll (move.stop.x bzw. y). Außerdem ist es möglich, die Geschwindigkeitsdämpfung anzugeben und somit die Maximalgeschwindigkeit zu reduzieren (move.damp.pan

bzw. tilt), und eine Verzögerung festzulegen (move.initial.delay), bis zu der keine Bewegung ausgeführt wird (Wulff & Marquard, 2016b). All dies sorgt für eine ruhigere Kameraführung, sodass auch hektische Bewegungen der getrackten Objekte abgefangen werden. Des Weiteren kann der Steering Worker das Koordinatensystem auch spiegeln (xflip bzw. yflip), falls die Kamera kopfüber montiert wird.

Im VISCA over IP Modul wird festgelegt, wie häufig die Kamera nach ihrer aktuellen Position gefragt wird (com.wulff.lecturesight.visca.updater.interval). Zusätzlich ist hier die in Kapitel 5.3.3 bereits erwähnte IP-Adresse und der Port der Kamera verortet.

Um die Auswirkungen dieser Einstellungen zu testen und die für diesen Hörsaal passenden Parameter zu finden, bedarf es einiger ausführlicher Tests. Hierfür wurde zunächst das Tracking mit einer Person getestet. Dabei wurde die Funktion bei schnellem hin- und hergehen sowie dem heraustreten aus der definierten Szene getestet. Zudem wurden die Tafeln bewegt und an die Tafeln geschrieben.

Dabei zeigte sich, dass das Tracking selbst dann gut funktioniert, wenn sich die Testperson aus der Szene entfernt und diese wieder betritt. Die Tafelbewegung hatte das Tracking jedoch gestört, sodass hier noch die maximale Blob-Größe angepasst werden musste. Die schnellen Bewegungen waren für die Kamera zwar kein Problem, doch das Bild war anschließend zu hektisch und unruhig, sodass die Maximalgeschwindigkeit gesenkt werden musste. Um den schnellen Bewegungen entgegenzuwirken hilft es außerdem die Genauigkeit des Steering Workers zu reduzieren. Somit konnte bei geringen Bewegungen ein ruhigeres Bild erreicht werden.

Mehrere Personen im Bild waren zunächst kein Problem, solange die zuerst erkannte Person nicht zu lange ruhig stand. Das größere Problem waren Phantomobjekte, also falsch positive Diagnosen. Der erste Versuch, die erkannten Objekte über das Szenenprofil zu ignorieren, hat sich als impraktikabel erwiesen: Es wurden nicht immer die gleichen Phantomobjekte erkannt, denn auch bei Bewegungen der Testperson blieb das getrackte Objekt stehen, obwohl sich die Person weiterbewegt hatte. Dabei hat sich gezeigt, dass dieses Problem zum einen mit der Videokompression zusammenhängt, als auch mit dem Kamerarauschen. Wie bereits in Kapitel 5.3.2 beschrieben wurde, hat ein Wechsel des Videocodecs zu H.264 und der Einsatz eines Gaußschen Weichzeichners zum Erfolg geführt.

5.3.7 Probleme in LectureSight

Beim Build von LectureSight gab es diverse Fehlermeldungen, da hierbei Repositories angegeben waren, die mittlerweile offline sind und somit einige Abhängigkeiten nicht aufgelöst werden konnten.

Außerdem war im Standardprofil des Builds das Bundle „lecturesight-framesource-v4l" ausgewählt, das Video4Linux Quellen verwendet und beispielsweise für Webcams gedacht ist. In diesem Fall funktioniert dieses Bundle nicht, da die Übersichtskamera nicht direkt an dem Computer angeschlossen ist, sondern das Videosignal über einen Netzwerkstream geliefert wird. Daher wird das Bundle „lecturesight-framesource-gst" benötigt, welches als Videoquelle eine GStreamer Pipeline akzeptiert.

Die bereits erwähnte Vorabversion der VISCA over IP Steuerung hatte zu Beginn einige Fehler, die zunächst behoben werden mussten, bevor der Test beginnen konnte. Hier wurden einige Konzepte noch direkt aus VISCA übernommen, da beispielsweise zu Beginn per Broadcast nach Kameras gesucht wurde, wie dies nur bei seriellen Systemen üblich und auch möglich war. Daher wurde dieses Bundle überarbeitet und die IP und den Port zur VISCA over IP-Steuerung in die Konfigurationsdatei ausgelagert.

Die häufigsten Fehler in diesem Bundle waren Byte-Offsets bei der Paketanalyse. Hier wurde der Header vergessen und nur der Payload berücksichtigt, da es durchgehend einen Versatz von 7 Bytes gab. Es wurden außerdem nicht sämtliche möglichen Message Types implementiert, was sich jedoch relativ einfach nachrüsten lies, da die einzelnen Befehle nur als Konstanten in einer extra Klasse definiert waren. Hier erschien beispielsweise beim ersten Build auch eine Compiler-Fehlermeldung, da die Message aus int-Arrays zusammengesetzt, jedoch byte-Arrays erwartet wurden.

Die genannten Fehler hatten jedoch keinen negativen Einfluss auf den weiteren Testvorgang, sodass dieser fortgesetzt werden konnte.

5.4 Kamera

Die hier verwendete Sony SRG-300H besitzt 23 Tilt und 24 Pan Speeds (Sony, 2014, S. 50–51), die jedoch, wie in Abbildung 26 zu erkennen, keinen linearen Verlauf haben. Bei dem Test im Hörsaal V38.01 hat sich gezeigt, dass eine Maximalgeschwindigkeit der Stufe 8 ideal ist, da die Kamerabewegung sonst zu abrupt ist.

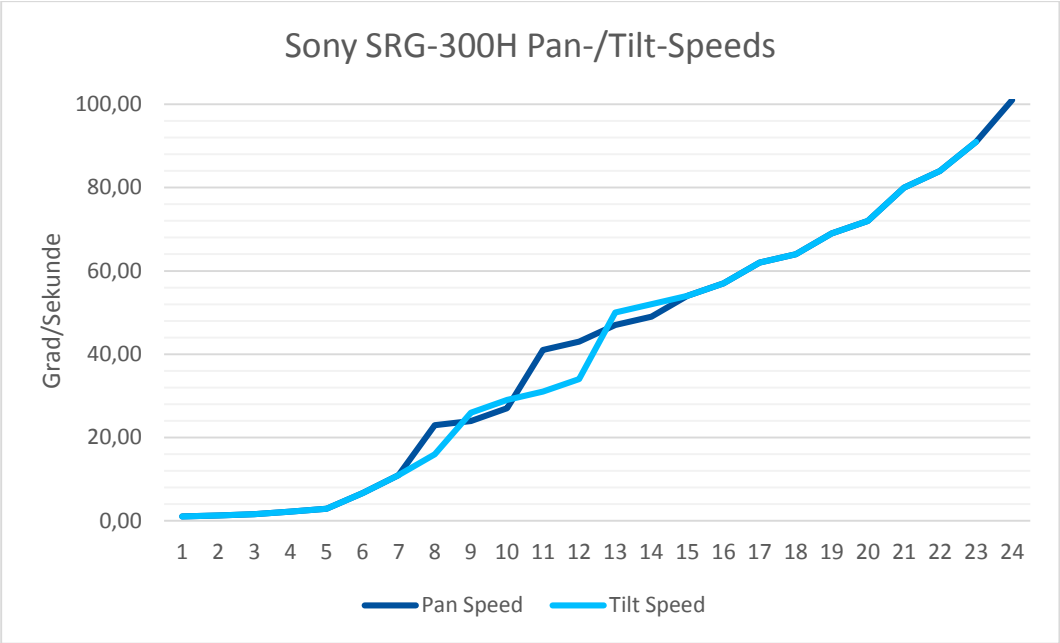

Abbildung 26: Geschwindigkeit Sony SRG-300H (Eigene Darstellung, Daten entnommen aus Sony, 2014, S. 50–51)

Gerade bei Stufe 8 ist erkennbar, dass die Geschwindigkeit des Pan Speeds bis zu Stufe 5 noch stark abnimmt, doch sich diese dann nur noch sehr wenig verringert. In der Praxis schwenkt die Kamera bei langen Strecken mit ihrer Maximalgeschwindigkeit und bremst ab einer definierten Entfernung stark ab, um dann zum Ende hin sehr langsam zum Zielpunkt zu fahren.

Eine Lookup Table sollte diesem Verhalten entgegenwirken und die Werte der Stufen 1 bis 8 gleichmäßiger verteilen (siehe Abbildung 27).

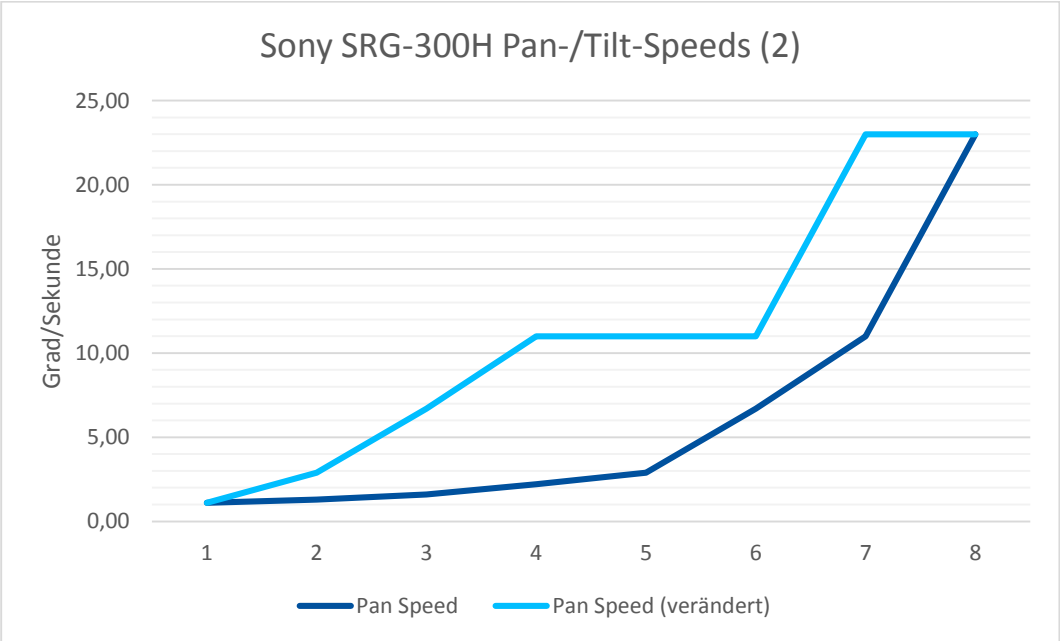

Abbildung 27: Geschwindigkeit Sony SRG-300H, verändert (Eigene Darstellung, Daten entnommen aus Sony, 2014, S. 50–51)

Das hat nur bedingt zu einer Besserung geführt, da der Bremsvorgang immer noch abrupt ist. Die Kamera besitzt jedoch zusätzlich zu den bereits genannten Pan und Tilt Speeds einen sogenannten Slow Mode (Sony, 2014, S. 50). Dieser besteht aus 127 Werten, die auf zwei Abschnitten zwei verschiedenen Kurven entsprechen (siehe Abbildung 28). Der erste Abschnitt bis Stufe 24 stellt am ehesten eine exponentielle Funktion dar, der zweite Abschnitt von Stufe 25 bis 127 ist linear.

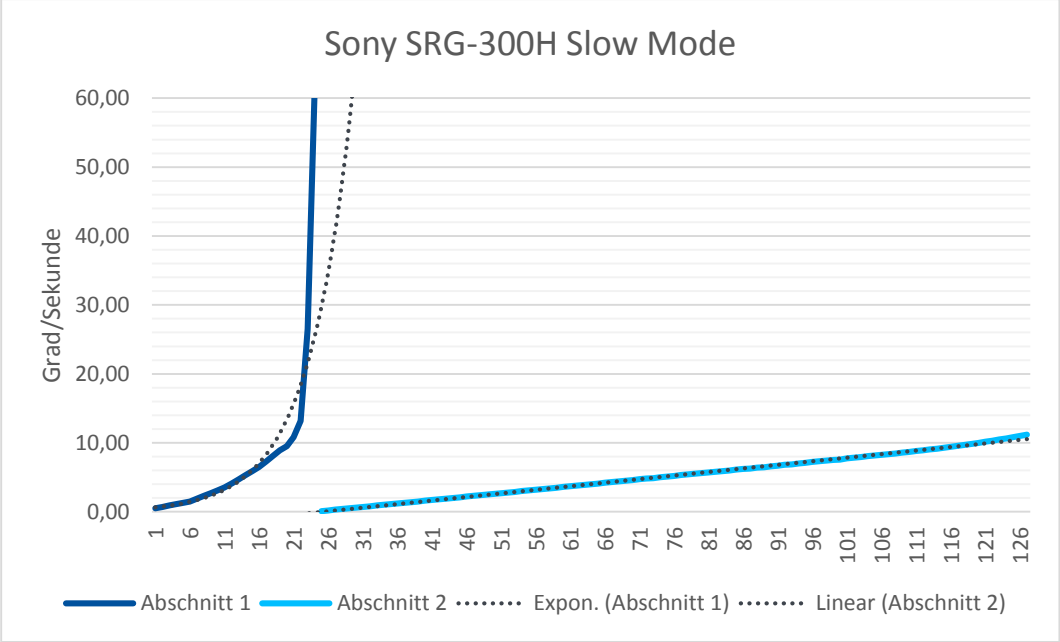

Abbildung 28: Slow Mode Sony SRG-300H (Eigene Darstellung, Daten entnommen aus Sony, 2014, S. 50–51)

Aufgrund dieser Werte wurde der Slow Mode dauerhaft aktivieret und die maximale Geschwindigkeit auf eine höhere Stufe (beispielsweise Stufe 23) gestellt. Damit konnte eine ähnliche Maximalgeschwindigkeit wie bisher zu erreicht werden. Sobald ein Bremsvorgang stattfindet, werden die Stufen linear reduziert und somit findet ein exponentieller Bremsvorgang statt. Da dieser in der Praxis ein deutlich ruhigeres Bild als der lineare Bremsvorgang aus Abbildung 27 erzeugt, wurde dies so in LectureSight im Camera Operator Modul implementiert.

Beim Testaufbau ist aufgefallen, dass die SRG-300H eine falsch implementierte Netzwerkschnittstelle hat. Der Kamera können eine IP-Adresse, eine Subnetzmaske und ein Gateway gegeben werden, jedoch wird dieses Gateway nicht wie erwartet verwendet.

Der eigentliche Zweck des Gateways ist es, den Netzwerkverkehr, der das Subnetz verlässt, weiterzuleiten, damit ein Router diesen an andere Subnetze senden kann. In diesem Fall funktioniert das für einen ICMP-Request (Ping) aus einem gerouteten Netz, der mit einer Reply beantwortet wird. Sobald jedoch aus einem gerouteten Netz über VISCA

over IP beispielsweise die Statusinformation abgefragt werden, wird keine Antwort empfangen. Mit Hilfe des Protokollanalyse-Programms Wireshark konnte gezeigt werden, dass eine fehlerhafte Implementierung vorliegt, da das Paket an die Kamera zwar gesendet wird, diese aber nicht antwortet. Sobald der Controller, der die Nachricht an die Kamera sendet, im gleichen Subnetz ist, funktioniert die Kommunikation auch über VISCA over IP.

Die Hörsäle an der Universität Stuttgart sind in verschiedene Gebiete aufgeteilt. In jedem Gebiet gibt es ein eigenes Subnetz für die Medientechnik, in dem auch die Kameras sind. All dies führt letztendlich dazu, dass ein zentraler LectureSight-Server nicht direkt mit allen Kameras kommunizieren könnte. Als Lösung bietet sich hier die Mediensteuerung an, die in jedem Hörsaal vorzufinden ist. Diese kann auch mit einem Gateway kommunizieren und somit auch vom LectureSight-Server erreicht werden.

Im folgenden Kapitel wird die Umsetzung dessen dargestellt.

5.5 Crestron-Programmierung

Um die bisherige Programmierung der Mediensteuerung für das Kamera-Tracking mit LectureSight vorzubereiten, sind daran einige Änderungen notwendig.

Die Programmierung wird mit Logikbausteinen, auch Module genannt, in der Software SIMPL Windows durchgeführt. Diese sind beispielsweise Logikoperationen wie „AND", „OR" oder „NOT". „AND" bezeichnet beispielsweise, dass sämtliche booleschen Variablen am Eingang des Moduls wahr sein müssen, damit die Variable am Ausgang wahr wird. Es gibt aber auch andere Bausteine wie zum Beispiel „Serial Send", das Kommandos an einen seriellen Port senden kann.

In SIMPL Windows gibt es drei verschiedene Typen von Variablen: digital, analog und seriell. Ein digitales Signal besteht aus einem Bit und sendet immer nur kurze Impulse. Ein analoges Signal besteht aus zwei Byte und hat einen Wertebereich von 0-65535, der wahlweise auch hexadezimal oder in Prozent angegeben werden kann. Ein serielles Signal ist ein Text oder String mit maximal 255 Byte. Diese entsprechen 255 Zeichen und können entweder in ASCII oder ebenfalls hexadezimal angegeben werden (Crestron, 2016).

Für die Kommunikation mit LectureSight müssen zwei neue UDP/IP-Verbindungen erstellt werden. Eine ist nötig, um der Software den aktuellen Tracking-Status mitzuteilen. Hierfür wird Port 52379 verwendet. Die andere wird für das VISCA over IP Protokoll verwendet, wozu Port 52380 verwendet wird (siehe auch Abbildung 21 auf Seite 28). Es wurde bewusst die Lösung gewählt, sowohl VISCA, als auch LectureSight je einen eigenen Port zuzuweisen, da die Kamerasteuerung durch LectureSight nicht zuverlässig funktioniert, wenn zwischen den Antworten der Kamera über VISCA over IP ein beliebiger Text gesendet wird. Außerdem sollte nicht derselbe Port wie bei VISCA over IP verwendet werden, da die Mediensteuerung auf diesem bereits mit der Kamera kommuniziert und es dadurch eventuell zu Störungen kommt.

Die Statusmeldung sollte jede Sekunde von Crestron an LectureSight gesendet werden, damit die Software noch schnell genug reagieren kann, aber nicht unnötig viele Daten gesendet werden müssen. Dafür wird ein Oszillator-Modul benötigt, welches jede Sekunde den Impuls zum Senden liefert.

Sobald auf dem Touchpanel der Button „Tracking" ausgewählt wird, meldet das Touchpanel diesen Zustand in einer Feedback-Variablen, die „true" gesetzt ist. Durch einen AND-Logikbaustein wird geprüft, ob ein Status gesendet werden soll und ob das Feedback vorhanden ist. Falls ja, wird das Kommando „:TRACK ON\n" mit einem „Serial Send"-Modul an den Port 52379 gesendet. Ein NOT-Logikbaustein hilft dabei im gegenteiligen Fall das Kommando „:TRACK OFF\n" zu senden. Der Doppelpunkt vor „TRACK" leitet den Beginn des Befehls ein, das „\n" steht in diesem Fall für ein CR LF, also einen Zeilenumbruch, und besagt das Ende des Befehls. In vielen Programmiersprachen, beispielsweise in C, steht die Escape-Sequenz „\n" nur für „new line" und erst in Verbindung mit „\r", das für „carriage return" steht, wird daraus ein kompletter Zeilenumbruch (ISO/IEC 9899:201x, S. 24).

Durch diese Befehle getriggert soll nun LectureSight eine neue Instanz starten beziehungsweise stoppen. Da jedoch die verwendeten Sony-Kameras, wie bereits im vorigen Kapitel erwähnt, nicht außerhalb eines Subnetzes erreichbar sind, muss die Mediensteuerung als Proxy dienen. Das bedeutet, dass sobald das Tracking über das Touchpanel aktiviert wird, die von LectureSight empfangenen Daten an die Kamera weitergeleitet werden und entsprechend auch die von der Kamera empfangenen Daten an LectureSight gesendet werden. Diese Verbindung ist in Crestron mit einem „Serial Buffer" möglich.

Nun ist zwar eine Verbindung von LectureSight zur Kamera hergestellt, jedoch kann die Mediensteuerung selbst noch nicht mit der Kamera kommunizieren, da diese zu Beginn des Projekts häufig noch über VISCA angesteuert wurden. Für VISCA over IP gibt es von Crestron bereits ein fertiges Softwaremodul, das logischerweise nicht exakt dem bereits vorhandenen für VISCA entspricht. Beispielsweise gibt es bisher pro Preset, also bestimmten gespeicherten Positionen, die die Kamera anfahren kann, einen eigenen Save-Befehl. Im neuen IP-Modul gibt es nur noch einen gemeinsamen Save-Befehl, der für das jeweils aktuell ausgewählte Preset gilt. Über einen OR-Logikbaustein kann eine Implementierung dessen aber leicht umgesetzt werden.

Die Unterscheidung zwischen den zu Beginn der Arbeit definierten Einstellungsgrößen Totale, Tafel und Tracking (zuvor „Redner") erfolgt in der Mediensteuerung über Presets. Die Kamera hatte bisher drei definierte Presets. Das erste ist die Home-Position, die dafür sorgt, dass die Kamera bei abgeschalteter Anlage in den Kamerakasten zeigt und somit der Datenschutz gewährleistet ist. Die zweite und dritte entsprechen den beiden soeben genannten Einstellungsgrößen. Da die Position „Redner" in der Programmierung noch nicht vorhanden war, musste ein viertes Preset angelegt werden, welches aufgerufen wird, sobald das Tracking aktiviert wird. Wie dieses in der Kamera selbst definiert wird ist dabei nebensächlich, da die Startposition bereits in LectureSight definiert wurde.

Zusätzlich zur Tracking-Kamera muss auch die Übersichtskamera gesteuert werden. Da es sich hierbei um ein kostengünstiges Modell handelt, besitzt diese kein eigenes Steuerprotokoll, sondern lediglich eine Weboberfläche, die mittels Basic Authentication (Basisauthentifizierung) erreichbar ist. Bei dieser Art der Authentifizierung werden Benutzername und Passwort in Base64 codiert und im Klartext übertragen. In diesem Fall erscheint beim Aufruf der Weboberfläche ein Pop-up, das einen Benutzernamen und ein Passwort anfordert. Um dieses Pop-up zu unterdrücken kann die gewünschte Seite nach dem Schema http://username:passwort@ip.adresse auch direkt erreicht werden.

Zuerst wurde hier die Home-Position eingestellt und ein neues Preset definiert, das den gewünschten Bildausschnitt zeigt. Nun ist es möglich, die Kamera über die Weboberfläche zu den beiden Positionen zu fahren. Mit Hilfe des Protokollanalyse-Programms Wireshark konnte der Klartext des HTTP Requests dargestellt werden.

Sofern in SIMPL Windows ein TCP/IP Client auf Port 80 angelegt wurde, können diese HTTP Befehle nun direkt mit einem Serial Send dorthin gesendet werden. Bei der hier verwendeten Kamera von LevelOne lauten diese folgendermaßen:

- GET /cgi-bin/camctrl.cgi?move=3&speedpan=30&speedtilt=30 HTTP/1.1\nHost: mm-sensor-v3801-1.rus.loc\nAuthorization: Basic cm9vdDo=\n\n
- GET /cgi-bin/preset.cgi?point=1 HTTP/1.1\nHost: mm-sensor-v3801-1.rus.loc\nAuthorization: Basic cm9vdDo=\n\n

Der erste Befehl setzt die Kamera auf ihre Home-Position, der zweite auf das Preset 1. Hier muss ebenfalls beachtet werden, dass Crestron mit der Escape-Sequenz „\n" bereits einen kompletten Zeilenumbruch beschreibt, Wireshark hingegen die gängige Bezeichnung „\r\n" verwendet. In den beiden dargestellten Befehlen wurde daher „\r\n" durch „\n" ersetzt.

Sobald das Tracking-Feedback aktiv ist, wird die Übersichtskamera auf das erste Preset gefahren. Wenn dieses beendet wird, soll sie zurück zur Home-Position fahren.

Beim Abschalten der Anlage werden sowohl die Tracking-Kamera als auch die Übersichtskamera zurück zur Home-Position gefahren. Dazu wird in einem Modul namens „Stepper with Progress & Reset", das zeitverzögert mehrere Befehle auslöst, zunächst das Tracking deaktiviert und anschließend über das Serial Send-Modul an beide Kameras der Befehl gesendet, diese zur Home-Position zu fahren. Anschließend werden die restlichen Geräte im Raum abgeschaltet und teilweise auch stromlos geschaltet.

Wenn die Programmierung abgeschlossen ist, muss diese noch kompiliert und anschließend zur Steuerung gesendet werden. Mit Hilfe der Software „Toolbox" ist es möglich, den Status der Signale während des Betriebs einzusehen und somit mögliche Fehler zu entdecken.

6 Verbesserung des Trackings

6.1 Erweiterung des virtuellen Kameraoperators

Bei dem vorigen Test wurde LectureSight mit dem Bundle „lecturesight.cameraoperator.panonly" gebaut. Das bedeutet, dass in der Konfigurationsdatei ein fester Y-Wert (=Neigewinkel) angegeben wird und die Kamera während des Trackings nur geschwenkt und nicht geneigt wird. Bei einem großen Abstand und einer starken Neigung der Kamera im Bezug zur Tafel führt dies aber dazu, dass die Kamera anstatt eines waagerechten einen bogenförmigen Schwenk ausführt.

Da sich das Bild der Kamera wie eine Kugel im Raum verhält, können folgende Kugelvariablen definiert werden (Bronštejn, Musiol & Mühlig, 2005, S. 216):

$$x = R \cdot \sin\theta \cdot \cos\varphi \quad [1]$$

$$y = R \cdot \sin\theta \cdot \sin\varphi \quad [2]$$

$$z = R \cdot \cos\theta \quad [3]$$

Die Kamera steht dabei im Ursprung O, schwenkt mit dem Azimutwinkel φ (Phi) und neigt mit dem Polarwinkel θ (Theta). Orthogonal zur x-Achse befindet sich die Tafel, der Bildausschnitt ist in negativer z-Richtung verschoben. Befände sich die Kamera nicht in der Mitte des Raumes, wäre sie in y-Richtung verschoben. R ist der Radius der Kugel und damit auch der Abstand zur Tafel (siehe Abbildung 29).

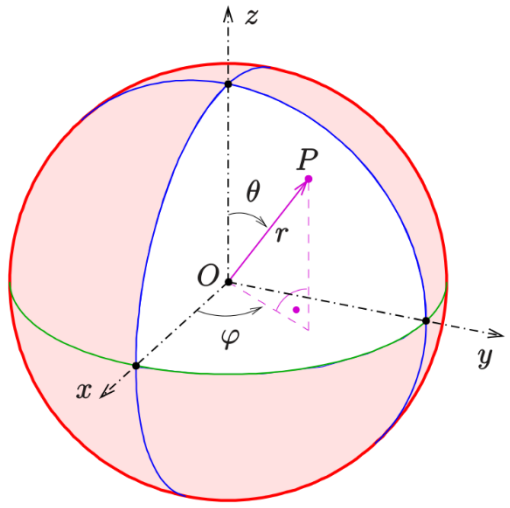

Abbildung 29: Kugelkoordinaten (Ag2gaeh, 2015)

Der x-Wert des Punktes P, der den Bildmittelpunkt beschreibt, wird als Konstante d gesetzt, dessen Wert sich nicht mehr ändert.

$$x_P = d = R \cdot \sin\theta \cdot \cos\varphi$$
$$\longrightarrow R = \frac{d}{\sin\theta \cdot \cos\varphi}$$

Wenn man R in Formel [2] einsetzt, erhält man:

$$y = \frac{d}{\sin\theta \cdot \cos\varphi} \cdot \sin\theta \cdot \sin\varphi = d \cdot \frac{\sin\varphi}{\cos\varphi} = d \cdot \tan\varphi$$

Ebenso kann man R in Formel [3] einsetzen:

$$z = \frac{d}{\sin\theta \cdot \cos\varphi} \cdot \cos\theta = d \cdot \frac{\cos\theta}{\sin\theta} \cdot \frac{1}{\cos\varphi}$$

$$= d \cdot \cot\theta \cdot \frac{1}{\cos\varphi} \qquad [4]$$

Um nun bei einem geänderten Azimutwinkel φ eine konstante Höhe des Kamerabildes im Raum zu bekommen, setzt man $z = z_0 = const.$

Daraus ergibt sich, dass nach der Formel $\theta(\varphi)$ gesucht werden muss. Diese erhält man, indem man die Formel [4] nach θ auflöst:

$$\cot\theta = \frac{z}{d} \cdot \cos\varphi$$
$$\theta = \cot^{-1}\left(\frac{z}{d} \cdot \cos\varphi\right)$$

Damit lässt sich nun berechnen, welcher Polarwinkel an der Kamera eingestellt werden muss, abhängig vom Azimutwinkel und bei konstantem Abstand d und z.

Die Lösung des Arkuskotangens ist kompliziert, da es in unterschiedlichen Programmen verschiedene Wertebereiche für diese Funktion gibt. Wie diese zustande kommen wird deutlich, wenn die Bildung einer Arkusfunktion genauer betrachtet wird.

Eine Arkusfunktion (auch zyklometrische Funktion) ist die Umkehrfunktion der trigonometrischen Funktion. Hierfür wird die ursprüngliche Funktion, in diesem Fall der Kotangens, in Monotonieintervalle zerlegt und die Spiegelung an der Winkelhalbierenden liefert die Umkehrfunktion (Bronštejn et al., 2005, S. 85).

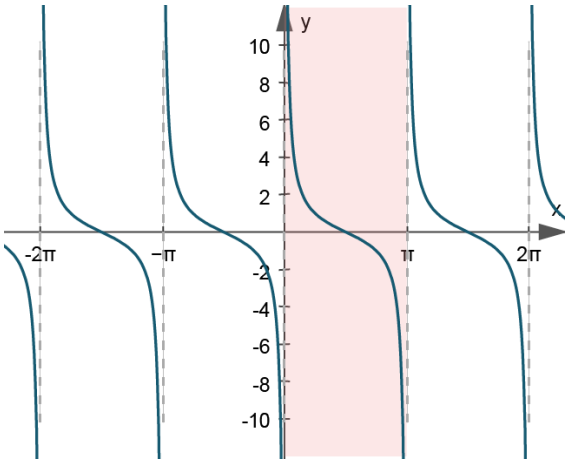

Abbildung 30: Kotangens von 0 bis π (Bourne, 2016)

Wird für den Kotangens ein Monotonieintervall von $0 < x < \pi$ gewählt (siehe rote Fläche in Abbildung 30), hat der Arkuskotangens den Wertebereich $k \cdot \pi < y < (k + 1) \cdot \pi$, wobei $k = 0$ der Hauptwert ist. Daraus ergibt sich ein Wertebereich von $0 < y < \pi$ (Bronštejn et al., 2005, S. 85–86) sowie die folgende stetige Funktion:

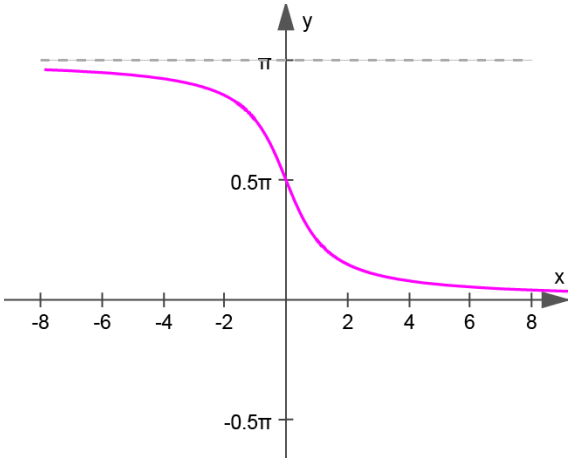

Abbildung 31: Arkuskotangens mit Wertebereich von 0<y<π (Bourne, 2016)

Eine weitere Möglichkeit wäre es, ein Monotonieintervall von $-\frac{\pi}{2} < x < \frac{\pi}{2}$ zu wählen (siehe rote Fläche in Abbildung 32), was den Nachteil hat, dass die Arkusfunktion keine stetige Funktion darstellt (siehe Abbildung 33). Für negative Werte von $\cot^{-1} x$ ist es hier möglich, zu diesen π zu addieren, um das korrekte Ergebnis zu erhalten.

Unter anderem verwenden die Webseite Wolphram|Alpha und die Software Matlab diese Definition, die Software Maple verwendet beispielsweise die selbe Definition wie Bronštejn et al. (2005) (Bourne, 2011).

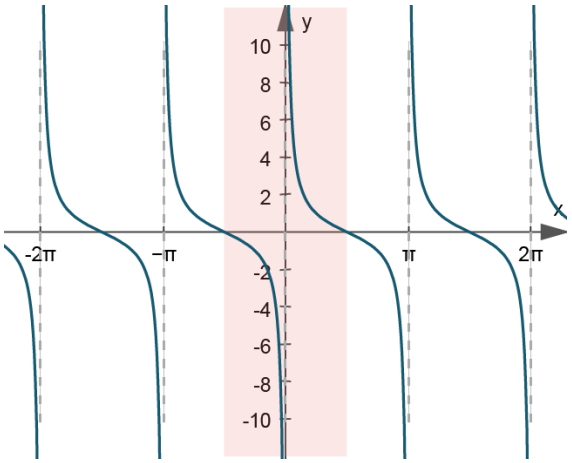

Abbildung 32: Kotangens von -π/2 bis π/2 (Bourne, 2016)

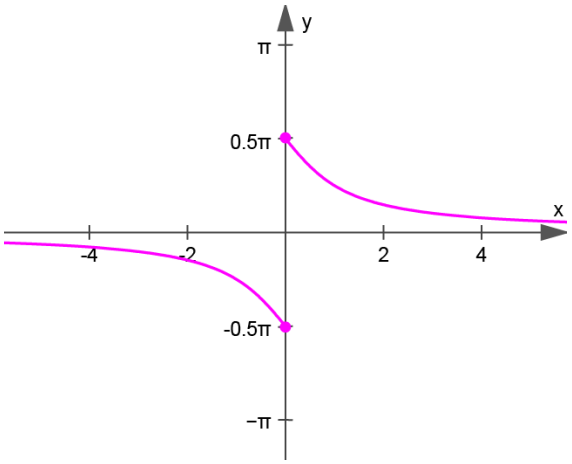

Abbildung 33: Arkuskotangens mit Wertebereich von -π/2<y<π/2 (Bourne, 2016)

Um diese unterschiedlichen Implementierungen zu umgehen, kann der Arkuskotangens auch, unabhängig vom Vorzeichen des x-Wertes, mit folgender Formel berechnet werden (Bronštejn et al., 2005, S. 87):

$$\cot^{-1} x = \frac{\pi}{2} - \tan^{-1} x \qquad [5]$$

Um außerdem einen Azimutwinkel zu erhalten, der nicht ab der z-Achse misst, sondern ab der Waagerechten (xy-Ebene), wurde der Winkel ϑ abhängig von θ definiert:

$$\theta = \frac{\pi}{2} - \vartheta$$

Somit gilt:

$$\frac{\pi}{2} - \vartheta = \frac{\pi}{2} - \tan^{-1}\left(\frac{z}{d} \cdot \cos\varphi\right)$$

$$\Rightarrow \quad \vartheta = \tan^{-1}\left(\frac{z}{d} \cdot \cos\varphi\right) \qquad [6]$$

Im Hörsaal V38.01, in welchen LectureSight getestet wurde, ist der Abstand zwischen Kamera und Tafel $d = 21{,}17\ m$ und die Bildoberkante von der optischen Achse $z_0 = -6{,}53\ m$ entfernt.

Daraus ergibt sich

$$\vartheta(\varphi_0) = \vartheta(0°) = \vartheta(0) = -0.2992\ rad = -17{,}14°$$

Dies entspricht exakt den in AutoCAD gemessenem Wert (siehe Abbildung 34).

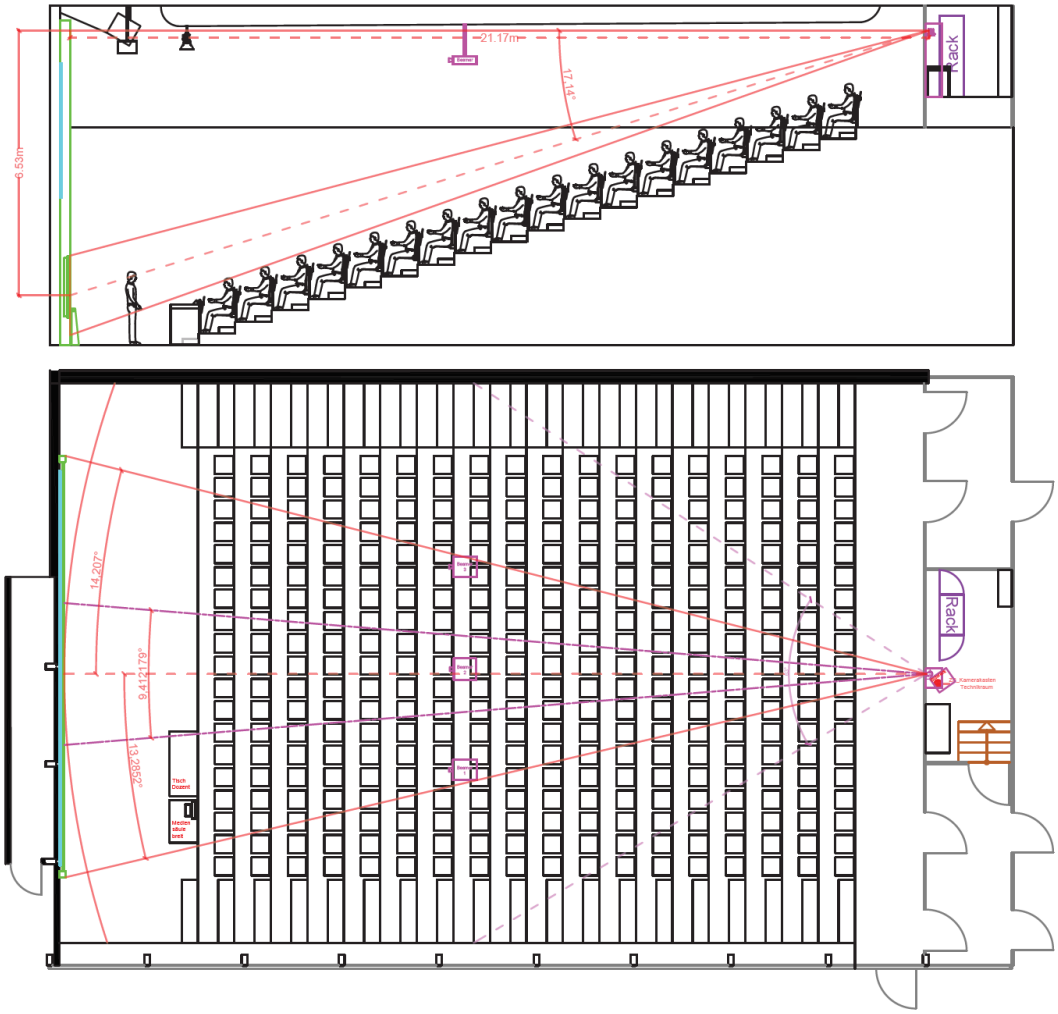

Abbildung 34: Dreiseitenansicht von V38.01 (Eigene Darstellung)

Für die maximale linke Auslenkung der Kamera von $13{,}2852° \triangleq 0{,}2318704818\ rad$ ergibt sich ein Winkel

$$\vartheta(13{,}2852°) = \tan^{-1}\left(\frac{-6{,}53\ m}{21{,}17\ m} \cdot \cos 0{,}2318704818\ rad\right) = -0{,}2916\ rad = -16{,}71°.$$

Das wäre eine Differenz zu $\vartheta(0°)$ von $0{,}43°$.

Um den Höhenunterschied bei dieser Entfernung zu berechnen wird zunächst der Winkel α des Dreiecks ABC mit $a = 21{,}17\ m;\ \beta = 0{,}43°;\ \gamma = 90°$ benötigt:

$$\alpha = 180° - \beta - \gamma \qquad \text{(Bronštejn et al., 2005, S. 145)}$$

$$= 180° - 90° - 0{,}43° = 89{,}57°$$

Da nun sämtliche Winkel bekannt sind, kann die gesuchte Seitenlänge ausgerechnet werden:

$$\frac{a}{\sin \alpha} = \frac{b}{\sin \beta} \qquad \text{(Bronštejn et al., 2005, S. 146)}$$

$$b = \frac{a \cdot \sin \beta}{\sin \alpha} = \frac{21{,}17\ m \cdot \sin 0{,}43°}{\sin 89{,}57°} = 0{,}1589\ m = 15{,}89\ cm$$

Dies entspricht ebenfalls den Beobachtungen vor Ort.

Die maximale rechte Auslenkung beträgt laut AutoCAD-Zeichnung $14{,}207°$. Daraus ergibt sich ein Winkel ϑ von $-16{,}65°$ und somit eine Differenz von $0{,}49°$ zu $\vartheta(0°)$ beziehungsweise ein Höhenunterschied von $18{,}1\ cm$.

Für die Implementierung dessen im Bundle „lecturesight.cameraoperator.panonly" wird zunächst der Winkel φ benötigt. Dieser kann berechnet werden, wenn die aktuelle x-Position bekannt ist. In LectureSight ist die aktuelle Position als absoluter Wert P_{Sony}, im Koordinatensystem der Kamera, und als relativer Wert P_{rel} im normalisierten Koordinatensystem vorhanden. Mit folgender Formel kann die hier gesuchte x-Position aus dem normalisierten Koordinatensystem heraus berechnet werden:

$$x_{Sony} = x_{rel} \cdot \frac{|l_{right}| + |l_{left}|}{2} + \left(\frac{|l_{right}| + |l_{left}|}{2} - |l_{left}| \right) \qquad [7]$$

Dabei ist l_{left} das „scene.limit.left", l_{right} entsprechend das „scene.limit.right".

Im ersten Teil der Formel [7] wird durch die Addition der beiden Limits zunächst die Breite der Szene berechnet. Diese wird anschließend halbiert, da das normalisierte Koordinatensystem von -1 bis +1 definiert ist und somit eine Breite von 2 hat. Wird diese Breite nun mit dem relativen Wert multipliziert, erhält man einen absoluten Wert, der nur korrekt ist, wenn der Betrag der beiden Limits identisch ist. Daher wird im zweiten Teil der Formel [7] berechnet, wie groß der Versatz aus der Mitte ist und dieser anschließend zum vorigen Produkt addiert.

Der Azimutwinkel φ kann mit den folgenden Formeln berechnet werden:

$$\varphi = \begin{cases} \frac{x_{Sony}}{pan_{max}} \cdot \varphi_{pan_{max}} \, f\ddot{u}r \, x > 0 \\ \frac{x_{Sony}}{pan_{min}} \cdot \varphi_{pan_{min}} \, f\ddot{u}r \, x < 0 \\ 0 \, f\ddot{u}r \, x = 0 \end{cases} \quad [8]$$

Formel [8] ergibt sich, indem das Verhältnis des aktuellen Werts x_{Sony} zu dem maximal oder minimal möglichen Pan-Wert (pan_{max} beziehungsweise pan_{min}), den die Kamera anfahren kann, gesetzt und anschließend mit dem maximalen oder minimalem Azimutwinkel der Kamera ($\varphi_{pan_{max}}$ beziehungsweise $\varphi_{pan_{min}}$) multipliziert wird.

In der Klasse java.lang.Math gibt es für die Berechnung des Winkels ϑ aus Formel [6] die Methode atan(double a) und cos(double a). Hier ist zu beachten, dass für beide Funktionen die beiden Winkel a im Bogenmaß benötigt werden. Mit der Funktion java.lang.Math.toRadians können die Winkel entsprechend umgerechnet werden. Das Ergebnis kann mit java.lang.Math.toDegree zurück in Grad gewandelt werden (Oracle, 2016).

Um nun durch den berechneten Winkel ϑ auf die gesuchte y-Position zu kommen, welche durch die Kamera angefahren werden muss, kann analog zu Formel [8] die folgende Formel hergeleitet werden:

$$\vartheta = \begin{cases} \frac{y_{Sony}}{tilt_{max}} \cdot \vartheta_{tilt_{max}} \, f\ddot{u}r \, y > 0 \\ \frac{y_{Sony}}{tilt_{min}} \cdot \vartheta_{tilt_{min}} \, f\ddot{u}r \, y < 0 \\ 0 \, f\ddot{u}r \, y = 0 \end{cases}$$

$$y_{Sony} = \begin{cases} \frac{tilt_{max} \cdot \vartheta}{\vartheta_{tilt_{max}}} \, f\ddot{u}r \, \vartheta > 0 \\ \frac{tilt_{min} \cdot \vartheta}{\vartheta_{tilt_{min}}} \, f\ddot{u}r \, \vartheta < 0 \\ 0 \, f\ddot{u}r \, \vartheta = 0 \end{cases}$$

Zur Bestimmung der y-Position im relativen Koordinatensystem aus der y-Position in Sony-Koordinaten wird analog zu Formel [7] die folgende Formel angewendet:

$$y_{Sony} = y_{rel} \cdot \frac{|l_{top}| + |l_{bottom}|}{2} + \left(\frac{|l_{top}| + |l_{bottom}|}{2} - |l_{bottom}| \right)$$

$$y_{rel} = \frac{|l_{bottom}| + 2 \cdot y_{Sony} - |l_{top}|}{|l_{top}| + |l_{bottom}|}$$

Somit ist nun die Zielposition bekannt, zu der die Kamera gefahren werden muss, um den bogenförmigen Schwenk auszugleichen.

7 Fazit

7.1 Lesbarkeit

In Kapitel 4.1 wurde unter anderem auf die Auflösung der Tafel eingegangen und dabei vermutet, dass die durchgeführte Lösung des Kamera-Trackings mit LectureSight eine höhere Detailauflösung schafft als der Einsatz einer 4K-Kamera.

Um dies zu überprüfen wurden drei Aufnahmen mit den verschiedenen Einstellungsgrößen erstellt und anschließend die Höhe der Tafel gemessen (siehe Abbildungen 35 bis 37). In der Einstellung „Totale" ist die Tafel 132 Pixel hoch, in der Einstellung „Tafel" beträgt die Höhe 259 Pixel und während des Trackings 718 Pixel. Somit wird durch die Einstellung „Tracking" im Vergleich zur Totale eine ca. 5,4-fache Bildhöhe erreicht, im Vergleich zur Einstellung „Tafel" beträgt der Faktor ca. 2,8. Die Tafel wäre mit einer 4K-Kamera bei der gleichen Einstellungsgröße jedoch nur doppelt so hoch. Daher wurde hiermit eine deutliche Verbesserung geschaffen.

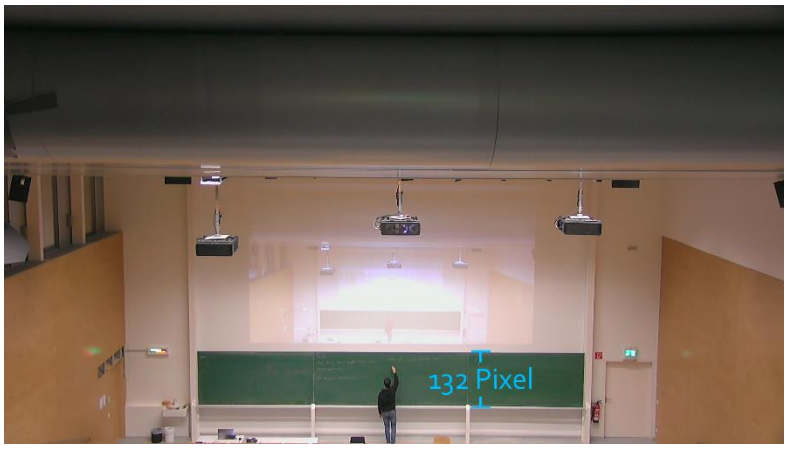

Abbildung 35: Einstellung Totale, Höhe der Tafel: 132 Pixel (Eigene Darstellung)

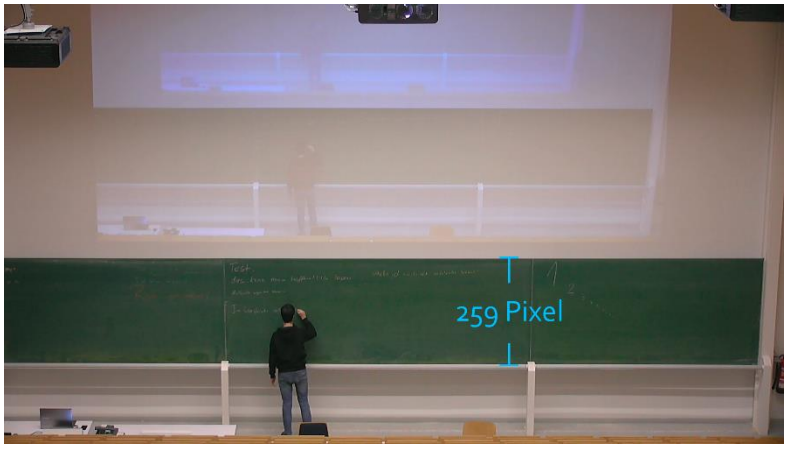

Abbildung 36: Einstellung Tafel, Höhe der Tafel: 259 Pixel (Eigene Darstellung)

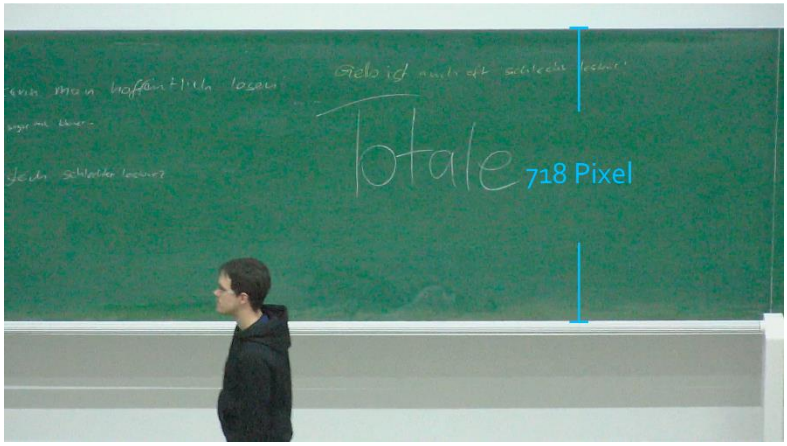

Abbildung 37: Einstellung Tracking, Höhe der Tafel: 718 Pixel (Eigene Darstellung)

Da nun die Auflösung der Tafel bekannt ist, kann berechnet werden, welche Größe ein Pixel auf der Tafel hat. Die Höhe der Tafel beträgt 1,5 m und somit gilt für die Totale:

$$1500\ mm = 132\ Pixel$$
$$1\ Pixel = \ 11{,}4\ mm$$

Für die Einstellungsgröße „Tafel" gilt weiterhin:

$$1500\ mm = 259\ Pixel$$
$$1\ Pixel = \ 5{,}8\ mm$$

Und für die Einstellungsgröße „Tracking" gilt:

$$1500\ mm = 718\ Pixel$$
$$1\ Pixel = \ 2{,}1\ mm$$

Dies legt fest, wie breit ein Strich minimal sein darf, damit dieser noch erkannt wird. Da es sich bei Buchstaben jedoch um komplexe Zeichen handelt, kann auf das ISO-Testzeichen Nr. 1 (siehe Abbildung 38) zurückgegriffen werden. Dieses Testzeichen, im Original „ISO character" genannt, ist so definiert, dass das kleinste Element eines Buchstabens gleicher Breite c noch wahrgenommen werden kann, wenn die parallelen Linien der Breite $\frac{c}{7}$ noch als einzelne Linien erkannt werden (ISO 446:2004(E), S. 2). Ursprünglich stammt dieses Zeichen aus der DIN 19051-1 und wurde beispielsweise für den Test von Projektionen entworfen.

Somit erhält man durch Multiplikation der obigen drei Werte mit dem Faktor 7 jeweils die minimale Breite eines Buchstabens.

$$b_{Totale} = 11{,}4\ mm * 7 = 79{,}8\ mm$$
$$b_{Tafel} = 5{,}8\ mm * 7 = 40{,}6\ mm$$
$$b_{Tracking} = 2{,}1\ mm * 7 = 14{,}7\ mm$$

Um zu überprüfen, ob die Lesbarkeit eines Tafelanschriebs durch die Tracking-Kamera in höherer Zoomstufe tatsächlich besser ist als in der Einstellungsgröße „Tafel", wurde ein Testbild (siehe Abbildung 39) erstellt, auf DIN A0 ausgedruckt und an die Tafel gehängt. Darin sind sowohl in Bruchteilen der Höhe und Breite des Papierformats aufgeteilte weiße und schwarze Rechtecke, als auch das ISO-Testzeichen Nr. 1 in selber Höhe wie die der Rechtecke platziert (siehe Abbildung 38).

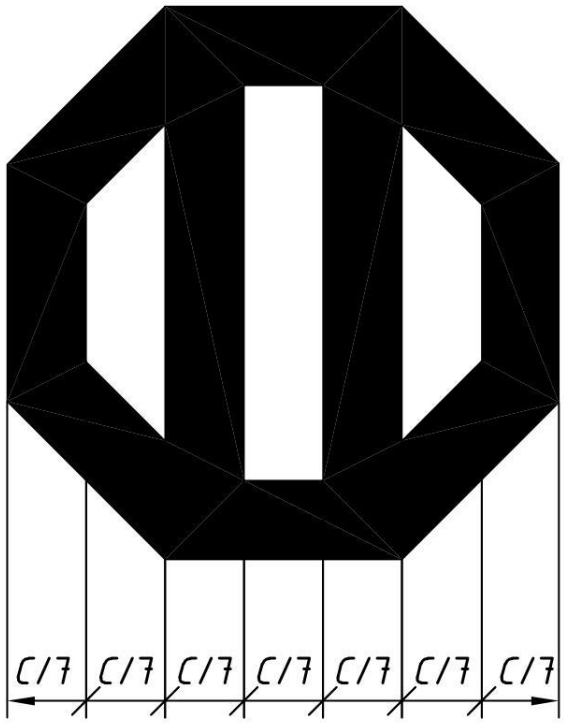

Abbildung 38: ISO Character (ISO 446:2004(E), S. 2)

Die Höhen der Rechtecke betragen:

148,625 mm (links oben), 74,312 mm (rechts oben),
37,156 mm (links unten), 18,578 mm (rechts unten),
9,289 mm (Mitte)

Die Höhen und Breiten der ISO-Testzeichen betragen (von oben nach unten):

74,312 mm, 37,156 mm, 18,578 mm, 9,289 mm

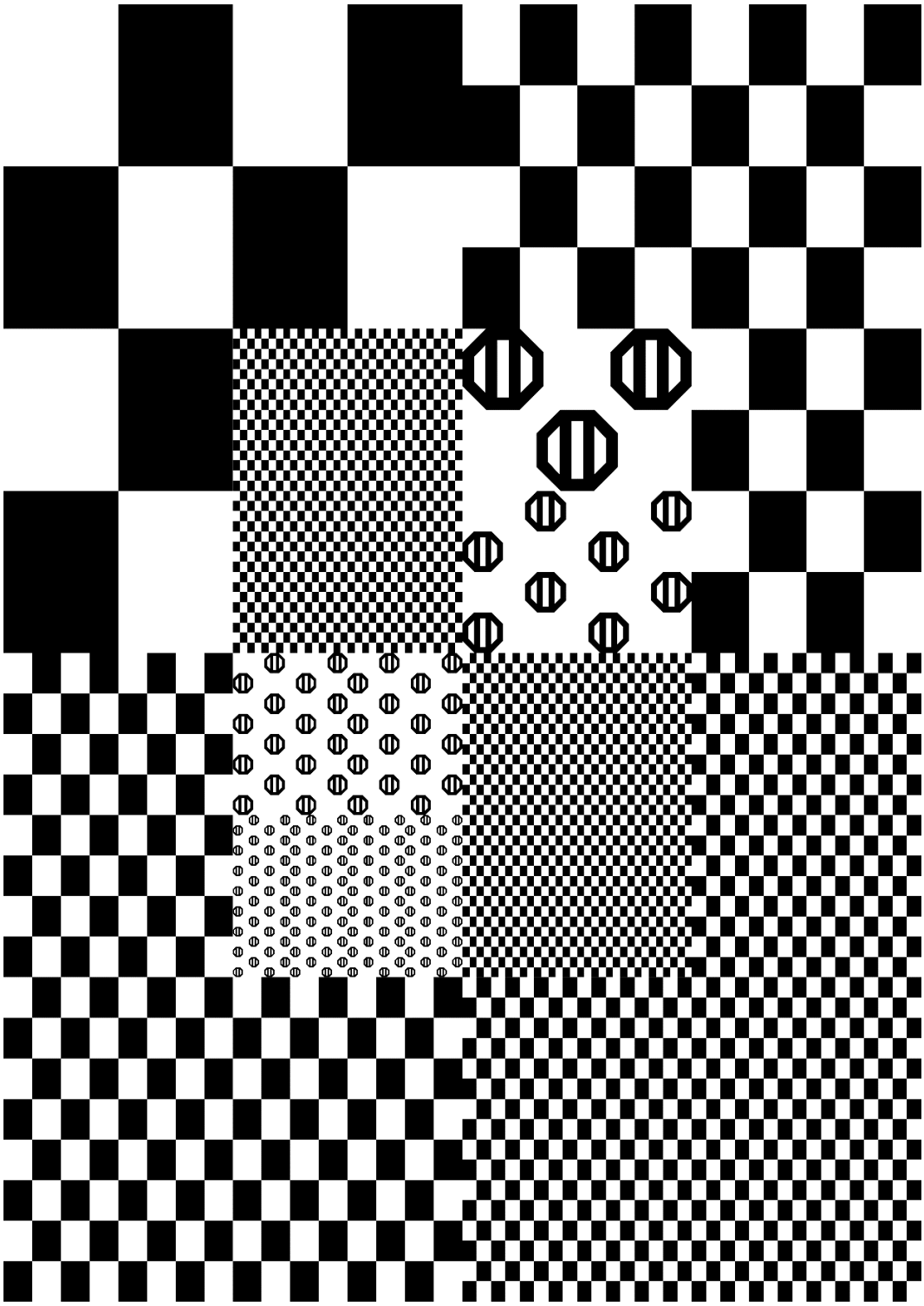

Abbildung 39: Testbild zur Lesbarkeit (Eigene Darstellung)

In Abbildung 40 sind drei Screenshots dargestellt, die durch Skalierung mit Pixelwiederholung auf gleiche Größe vergrößert wurden. Darunter sind mit derselben Skalierungsmethode Details aus der Mitte des Testbildes abgebildet.

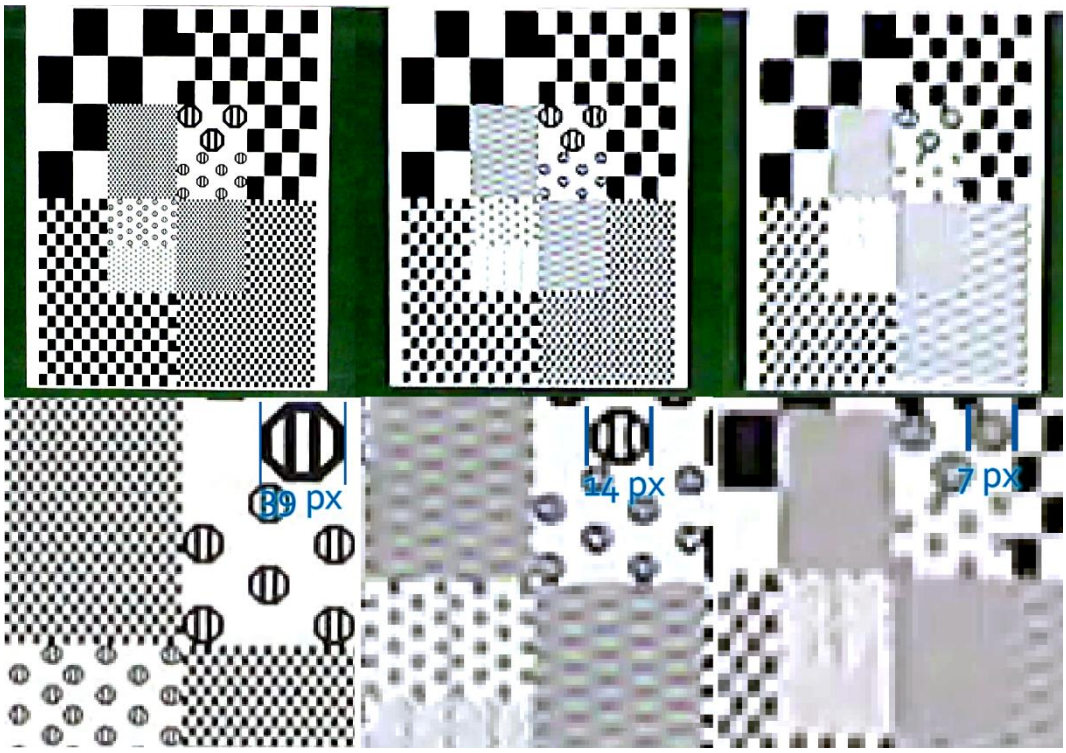

Abbildung 40: Testbild bei den Einstellungsgrößen "Tracking", "Tafel" und "Totale" (Eigene Darstellung)

Wie bereits zuvor erwähnt, müssen die einzelnen parallelen Linien des ISO-Testzeichens noch klar voneinander zu unterscheiden sein, was theoretisch bei einer Breite von 7 Pixeln der Fall wäre. Da hier jedoch eine Abtastung stattfindet, ist hierbei das Nyquist-Shannon-Abtasttheorem anzuwenden und die Abtastfrequenz muss daher mindestens doppelt so hoch sein wie die kleinste im Bild vorkommende Frequenz (Kotelnikov, 1933). Soll also ein sieben Pixel breites Symbol dargestellt werden, muss dieses mit mindestens 14 Pixeln abgetastet werden, da sonst die Linienpaare mit einer großen Wahrscheinlichkeit nicht exakt dem Raster der Kamera entsprechen. Dadurch lässt sich nun berechnen, ab welcher Mindestbreite ein Zeichen sicher abgebildet werden kann:

$$b_{Tracking} = \frac{74{,}312\ mm}{39\ px} \cdot 14\ px = 26{,}676\ mm$$

$$b_{Tafel} = \frac{74{,}312\ mm}{14\ px} \cdot 14\ px = 74{,}312\ mm$$

$$b_{Totale} = \frac{74{,}312\ mm}{7\ px} \cdot 14\ px = 148{,}624\ mm$$

Diese Werte stimmen, im Rahmen der Messungenauigkeit, sowohl mit der zuvor berechneten Größe eines Pixels als auch mit den Vergrößerungsfaktoren überein. Somit gilt, dass mit der Einstellungsgröße „Tracking" nun eine Schriftgröße von ca. 27 mm Breite abgebildet werden kann.

7.2 Ausblick

Der Test zur Lesbarkeit bestätigt, dass mit einer durch LectureSight bewegten Kamera und einem höheren Zoomfaktor die Deutlichkeit von Tafelanschrieben in den Videoaufzeichnungen drastisch zunimmt und somit auch klassische Vorlesungen aufgezeichnet werden können, die nicht ausschließlich die Projektion verwenden. Daher ermöglichte LectureSight den Aufbau eines geeigneten Setups für automatisches Kamera-Tracking und die Aufzeichnung von Vorlesungen mit Tafelanschrieben.

In der Vergangenheit bestanden viele Vorlesungsaufzeichnungen oft nur aus den Vortragsfolien und der Stimme des Dozenten, da dies bis vor einiger Zeit mit Lecturnity gar nicht anders möglich war und es zudem in einigen Hörsälen keine Kamera gab. Die Ergebnisse der Studie aus Kapitel 4.3 zeigen eine erhebliche Verbesserung der Lernleistung, alleine durch eine bewegte Kamera, da auf deren Bild die Dozenten besser erkannt werden können.

Durch den Einsatz von Kamera-Tracking wird durch die bessere Lesbarkeit von Tafelanschrieben und durch die Möglichkeit, die Gestik und Mimik von Dozenten zu erkennen, insgesamt ein besseres Lernergebnis geschaffen. Daher ist eine Installation des getesteten LectureSight-Systems in weiteren Hörsälen als Ergebnis dieser Arbeit uneingeschränkt zu empfehlen. Die technischen Voraussetzungen konnten in deren Rahmen erfolgreich implementiert und getestet werden, sodass nun die Grundlage geschaffen ist, dieses System auch in weiteren Hörsälen unkompliziert einzusetzen.

Abbildungsverzeichnis

Abbildung 1: Einstellungsgröße Totale 1
Abbildung 2: Einstellungsgröße Tafel 1
Abbildung 3: Einstellungsgröße Redner 2
Abbildung 4: Software-Übersicht 3
Abbildung 5: Übertragung 5
Abbildung 6: Hörsaalschema IPTV 5
Abbildung 7: ON AIR-Leucte 6
Abbildung 8: Erstellung der Vorschaubilder 7
Abbildung 9: Protokoll zwischen Mediensteuerung und CA 8
Abbildung 10: Paketstruktur von VISCA 10
Abbildung 11: Zeitlicher Ablauf von VISCA 10
Abbildung 12: Aufbau des Message Headers von VISCA over IP 11
Abbildung 13: Verlust des VISCA-Befehls 13
Abbildung 14: Verlust der ACK oder Completion Nachricht 13
Abbildung 15: Konsolenausgabe des Python-Skripts 14
Abbildung 16: Laufzeit von VISCA over IP Paketen 14
Abbildung 17: Softwarearchitektur von LectureSight 22
Abbildung 18: Stimulusvideo Kontrollgruppe 25
Abbildung 19: Stimulusvideo Experimentalgruppe 25
Abbildung 20: Lernleistung 26
Abbildung 21: Übersicht zum Testaufbau im Demohörsaal XL 28
Abbildung 22: deviceQuery der NVIDIA GeForce GTX 750 Ti 29
Abbildung 23: Grafikkarten im Vergleich 30
Abbildung 24: Overlay im VLC Media Player 34
Abbildung 25: LectureSight Scene Profile Editor 36
Abbildung 26: Geschwindigkeit Sony SRG-300H 40
Abbildung 27: Geschwindigkeit Sony SRG-300H, verändert 40
Abbildung 28: Slow Mode Sony SRG-300H 41
Abbildung 29: Kugelkoordinaten 46
Abbildung 30: Kotangens von 0 bis π 48
Abbildung 31: Arkuskotangens mit Wertebereich von 0<y<π 48
Abbildung 32: Kotangens von -π/2 bis π/2 49
Abbildung 33: Arkuskotangens mit Wertebereich von -π/2<y<π/2 49
Abbildung 34: Dreiseitenansicht von V38.01 50
Abbildung 35: Einstellung Totale, Höhe der Tafel: 132 Pixel 53
Abbildung 36: Einstellung Tafel, Höhe der Tafel: 259 Pixel 53
Abbildung 37: Einstellung Tracking, Höhe der Tafel: 718 Pixel 54
Abbildung 38: ISO Character 55
Abbildung 39: Testbild zur Lesbarkeit 56
Abbildung 40: Testbild bei den Einstellungsgrößen "Tracking", "Tafel" und "Totale" ... 57
Abbildung 41: Sequenz 1, ohne Tracking 65
Abbildung 42: Sequenz 2, mit Tracking 65

Literaturverzeichnis

Ag2gaeh (Wikimedia Commons, Hrsg.). (2015). *File:Kugelkoord-def.svg.* Zugriff am 29.11.2016. Verfügbar unter https://commons.wikimedia.org/wiki/File:Kugelkoord-def.svg

Bergert, I. (2014, 27. Oktober). *Bericht zum Staatshaushaltsplan für 2015 / 2016* (Ministerium für Wissenschaft, Forschung und Kunst Baden-Württemberg, Hrsg.). Zugriff am 17.12.2016. Verfügbar unter https://mwk.baden-wuerttemberg.de/fileadmin/redaktion/m-mwk/intern/dateien/pdf/Kerndaten/GB_2015-2016_Gesamttext.pdf

Bourne, M. (Interactive Mathematics, Hrsg.). (2011, 3. Mai). *Which is the correct graph of arccot x?* Zugriff am 30.11.2016. Verfügbar unter http://www.intmath.com/blog/mathematics/which-is-the-correct-graph-of-arccot-x-6009

Bourne, M. (Interactive Mathematics, Hrsg.). (2016). *7. The Inverse Trigonometric Functions.* Zugriff am 30.11.2016. Verfügbar unter http://www.intmath.com/analytic-trigonometry/7-inverse-trigo-functions.php#arccot

Brdiczka, O., Knipping, L., Ludwig, N., Mertens, R., Ketterl, M., Schulte, O. A. et al. (2010). Opencast Matterhorn. *Interactive Technology and Smart Education, 7* (3), 168–180. http://dx.doi.org/10.1108/17415651011071631

Bronštejn, I. N., Musiol, G. & Mühlig, H. (2005). *Taschenbuch der Mathematik* (6., vollst. überarb. und erg. Aufl.). Frankfurt am Main: Deutsch.

Crestron. (2016) SIMPL Windows [Computer software]. Verfügbar unter https://www.crestron.com/resources/software

Gaster, B. R., Howes, L., Kaeli, D. R., Mistry, P. & Schaa, D. (2013). *Heterogeneous computing with OpenCL* (Rev. OpenCL 1.2 ed.). Amsterdam: Elsevier/Morgan Kaufmann.

Grube, P. (2015, März). *Stuttgart's Lecture Recording Setup using Matterhorn and TIKCA*. Vortrag beim Opencast Summit 2015, Manchester. Zugriff am 18.10.2016. Verfügbar unter https://www.youtube.com/watch?v=8MLrweOwvVs

HbbTV. (2016). *Absatz von Fernsehgeräten weltweit von 2008 bis 2015 und Prognose bis 2017 (in Millionen Stück).* Zugriff am 17.12.2016. Verfügbar unter https://de.statista.com/statistik/daten/studie/193715/umfrage/absatz-von-fernsehgeraeten-weltweit/

Hennessy, J. L., Patterson, D. A. & Asanović, K. (2012). *Computer architecture. A quantitative approach* (5th ed.). Waltham, MA: Morgan Kaufmann.

ISO, 446:2004(E) (15.03.2004). *Micrographics — ISO character and ISO test chart No. 1 — Description and use*. Genf: International Organization for Standardization.

ISO/IEC, 9899:201x (12.04.2011). *Programming languages — C*.

Kiesow, L. (2015). *Opencast 2.0.0 has been released–Google Groups.* Opencast Announcements. Zugriff am 18.10.2016. Verfügbar unter https://groups.google.com/a/opencast.org/forum/#!topic/announcements/RN08aG4Z4KU

Kleinefeld, N., Lehmann, C. & Ottow, S. *ELAN e.V. | LectureSight.* Zugriff am 24.11.2016. Verfügbar unter https://www.elan-ev.de/produkte_av_lecture_sight.php

Kotelnikov, V. A. (1933). *On the carrying capacity of the ether and wire in telecommunications. Material for the First All-Union Conference on Questions of Communication, Izd. Red. Upr. Svyazi RKKA*. Moskau. Zugriff am 19.11.2016. Verfügbar unter http://ict.open.ac.uk/classics/1.pdf

LevelOne. (2014, 22. Juni). *FCS-6020 User Manual. Version 1.0.* Zugriff am 01.12.2016. Verfügbar unter http://download.level1.info/manual/FCS-6020-V1_UM_V1.0.pdf

Lindeberg, T. (1993). Detecting salient blob-like image structures and their scales with a scale-space primal sketch: A method for focus-of-attention. *International Journal of Computer Vision, 11* (3), 283–318. http://dx.doi.org/10.1007/BF01469346

Marais, P. A/Prof., Marquard, S., Fitzhenry, C., Hahn, M. & Khtieb, M. T. (2016) Track4K [Computer software]. Kapstadt: University of Cape Town. Verfügbar unter http://track4k.co.za/

Munshi, A. (2012). *OpenCL programming guide* (OpenGL series). Upper Saddle River, NJ: Addison-Wesley.

NVIDIA. (2009). *OpenCL for NVIDIA.* Zugriff am 26.10.2016. Verfügbar unter https://web.archive.org/web/20091125133728/http://www.nvidia.com/object/cuda_opencl.html

NVIDIA. (2012). *Datasheet: NVIDIA Kepler Next-Generation Cuda Compute Architecture.* Zugriff am 27.10.2016. Verfügbar unter http://www.nvidia.com/content/PDF/kepler/NV_DS_Tesla_KCompute_Arch_May_2012_LR.pdf

NVIDIA. (2013). *Tesla K40 GPU Active Accelerator. Board Specification.* Zugriff am 27.10.2016. Verfügbar unter http://www.nvidia.com/content/PDF/kepler/Tesla-K40-Active-Board-Spec-BD-06949-001_v03.pdf

NVIDIA. (2014). *GeForce GTX 750 Ti Whitepaper.* Zugriff am 27.10.2016. Verfügbar unter http://international.download.nvidia.com/geforce-com/international/pdfs/GeForce-GTX-750-Ti-Whitepaper.pdf

NVIDIA. (2014). *Maxwell Architecture. NVIDIA Developer.* Zugriff am 04.11.2016. Verfügbar unter https://developer.nvidia.com/maxwell-compute-architecture

NVIDIA. (2015). *Tesla K80 GPU Accelerator. Board Specification.* Zugriff am 27.10.2016. Verfügbar unter http://images.nvidia.com/content/pdf/kepler/Tesla-K80-BoardSpec-07317-001-v05.pdf

NVIDIA. (2016a, 21. Oktober). *NVIDIA Tesla K80.* Zugriff am 27.10.2016. Verfügbar unter http://www.nvidia.de/object/tesla-k80-de.html

NVIDIA. (2016b). *GeForce 9400 GT | Specifications | GeForce.* Zugriff am 27.10.2016. Verfügbar unter http://www.geforce.com/hardware/desktop-gpus/geforce-9400-gt/specifications

Opencast. (2016). *Opencast 2.0: Release Notes. Opencast Matterhorn becomes simply Opencast.* Zugriff am 18.10.2016. Verfügbar unter https://docs.opencast.org/r/2.0.x/admin/release.notes/

Oracle. (2016). *Math (Java Platform SE 8).* Zugriff am 19.12.2016. Verfügbar unter https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html#atan-double-

Porter, B. (The Apache Software Foundation, Hrsg.). (2016, 17. November). *Maven – Introduction to the Build Lifecycle.* Zugriff am 04.12.2016. Verfügbar unter https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

Rolf, R. (2016, September). *Erfahrungen mit 4K*. Vortrag bei Opencast DACH 2016, Stuttgart. Zugriff am 24.11.2016. Verfügbar unter https://www.tik.uni-stuttgart.de/dienste/elearning/aufzeichnungen/oc-dach-2016/

Rupp, L., Wulff, B. & Hamborg, K.-C. (2014, Oktober). *Veranstaltungsaufzeichnungen mit LectureSight. Effekte auf Lernen und Akzeptanz*. Vortrag bei Teaching Trends 2014: Offenen für neue Wege - Digitale Medien in der Hochschule, Oldenburg. Zugriff am 05.11.2016. Verfügbar unter https://www.uni-oldenburg.de/fileadmin/user_upload/c3l/Teaching_Trends_2014/Dokumentation/TT2014_Track3_Rupp.pdf

Schmidt-Casdorff, C. & Vogel, T. (2009). *OSGi. Einstieg und Überblick* (1. Aufl.) [S.l.]: entwickler.press.

Skiles, M. (1001FreeDownloads.com, Hrsg.). (2015, 10. Februar). *Lynny Icons - Full.* Zugriff am 03.12.2016. Verfügbar unter http://www.1001freedownloads.com/free-vector/lynny-icons-full

Sony. (2014, 15. Juli). *SRG-300H Technical Manual (EN)* (Sony, Hrsg.). Verfügbar unter http://www.sony.de/pro/support/operation-manual/1237493003156

Speth, S. (2016, Dezember). *Programmierung und Softwareentwicklung*. Vortragsübung, Universität Stuttgart.

Statista. (2016). *Absatz von 4K-/UHD-Fernsehgeräten weltweit von 2013 bis 2015 und Prognose bis 2020 (in Millionen Stück).* Zugriff am 17.12.2016. Verfügbar unter https://de.statista.com/statistik/daten/studie/270070/umfrage/prognose-zum-absatz-von-ultra-hd-fernsehgeraeten-weltweit/

The Apache Software Foundation (Hrsg.). (2016a, 17. November). *Maven – Introduction to the POM.* Zugriff am 04.12.2016. Verfügbar unter https://maven.apache.org/guides/introduction/introduction-to-the-pom.html

The Apache Software Foundation (Hrsg.). (2016b, 17. November). *Maven – Introduction.* Zugriff am 04.12.2016. Verfügbar unter http://maven.apache.org/what-is-maven.html

The Apache Software Foundation (Hrsg.). (2016c, 17. November). *Maven – POM Reference.* Zugriff am 04.12.2016. Verfügbar unter https://maven.apache.org/pom.html

The Khronos Group. (2008). *The Khronos Group Releases OpenCL 1.0 Specification*. Singapur. Zugriff am 04.12.2016. Verfügbar unter https://www.khronos.org/news/press/the_khronos_group_releases_opencl_1.0_specification

The Khronos Group. (2016). *OpenCL - The open standard for parallel programming of heterogeneous systems.* Zugriff am 04.12.2016. Verfügbar unter https://www.khronos.org/opencl/

TIK (Hrsg.). (2015, 24. September). *TIK-Dienst: Veranstaltungsaufzeichnungen. TIK-Dienste | Technische Informations- und Kommunikationsdienste | Universität Stuttgart.* Zugriff am 14.10.2016.

Trevett, N. (2012, Dezember). *OpenCL Overview*. Vortrag bei SIGGRAPH Asia. Zugriff am 26.10.2016. Verfügbar unter https://www.khronos.org/assets/uploads/developers/library/2012-siggraph-asia/Trevett-OpenCL-Overview_SIGGRAPH-Asia-Dec12-nt2.pdf

Vass, M. (2014, 17. April). *Veranstaltungsaufzeichnungen | Universität Stuttgart,* Uni Stuttgart - Rechenzentrum. Zugriff am 16.12.2016. Verfügbar unter https://media.uni-stuttgart.de/uploadportal/ViewerPortal.php

Wireworx GmbH. (2015a, 19. Februar). *010 Projektbeschreibung. MT 2016 Universität Stuttgart. Rahmenvertrag zur Erneuerung der Medientechnik und Wartungsvertrag*. Stuttgart.

Wireworx GmbH. (2015b). *Softwarebeschreibung Uni Stuttgart 2016. Großer Hörsaal (10 Zoll Touchpanel)*. Stuttgart.

Wulff, B. (2012). *GPU-based real-time Video Analysis for Lecture Recordings*. Bachelorarbeit, Universität Osnabrück. Osnabrück.

Wulff, B. (2013). *3.3. Calibration - LectureSight - Opencast Projects Wiki,* LectureSight. Zugriff am 02.12.2016. Verfügbar unter https://opencast.jira.com/wiki/display/LECTURESIGHT/3.3.+Calibration

Wulff, B. (2015, März). *LectureSight Workshop*. Vortrag beim Opencast Summit 2015, Manchester. Zugriff am 05.12.2016. Verfügbar unter https://www.youtube.com/watch?v=V8qWUPGqi24

Wulff, B. & Fecke, A. (2013, April). *LectureSight. Automatic Camera Control in Lecture Recordings*. Vortrag bei 7th TF-Media Task Force meeting, Paris. Verfügbar unter https://www.terena.org/activities/media/meeting7/slides/20130403-LectureSight.pdf

Wulff, B., Fecke, A., Rupp, L. & Hamborg, K.-C. (2014). LectureSight. An open source system for automatic camera control for lecture recordings. *Interactive Technology and Smart Education, 11* (3), 184–200. http://dx.doi.org/10.1108/ITSE-07-2014-0019

Wulff, B. & Marquard, S. (2014) [*LS-60] Create general GSt FrameSource - Opencast.* Zugriff am 02.12.2016. Verfügbar unter https://opencast.jira.com/browse/LS-60

Wulff, B. & Marquard, S. (2016a). *1.1. Hardware - LectureSight - Opencast Projects Wiki,* LectureSight. Zugriff am 26.10.2016. Verfügbar unter https://opencast.jira.com/wiki/display/LECTURESIGHT/1.1.+Hardware

Wulff, B. & Marquard, S. (2016b) LectureSight [Computer software]. Verfügbar unter https://bitbucket.org/bwulff/lecturesight

Wulff, B. & Marquard, S. (2016c, März). *LectureSight in Action*. Vortrag beim Opencast Summit, Köln. Zugriff am 05.12.2016. Verfügbar unter https://www.youtube.com/watch?v=PibOj6Pt3kU

Zurek, P. (2016, September). *TIKCA*. Vortrag bei Opencast DACH 2016, Stuttgart. Zugriff am 18.10.2016. Verfügbar unter https://www.tik.uni-stuttgart.de/dienste/elearning/aufzeichnungen/oc-dach-2016/

Zurek, P. & Pérez Vázquez, R. (2016) Pysca [Computer software]. Köln: Pérez Vázquez, Rubén. Verfügbar unter https://bitbucket.org/uni-koeln/pysca/

Anhang A

Elektronischer Anhang

Der elektronische Anhang zeigt Ausschnitte zweier Aufnahmen der Vortragsübung „Programmierung und Softwareentwicklung" an der Universität Stuttgart aus dem Hörsaal V38.01 (Speth, 2016). Die Datei „Anhang A.mp4" besteht aus zwei Sequenzen, die denselben Dozenten zuerst ohne (Abbildung 41) und anschließend mit Kamera-Tracking durch die Software LectureSight zeigen (Abbildung 42).

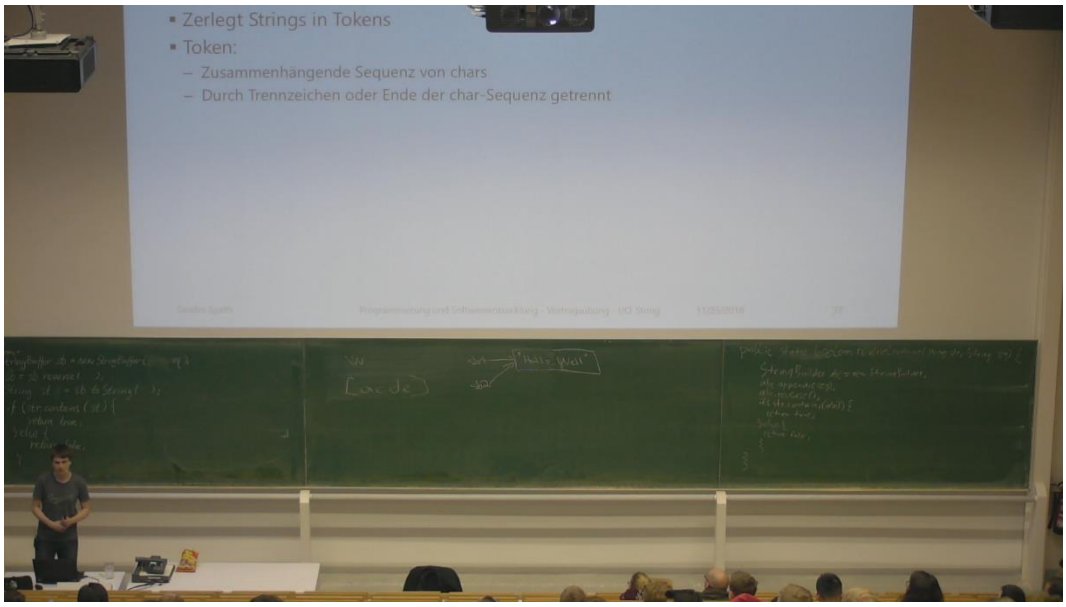

Abbildung 41: Sequenz 1, ohne Tracking (Speth, 2016)

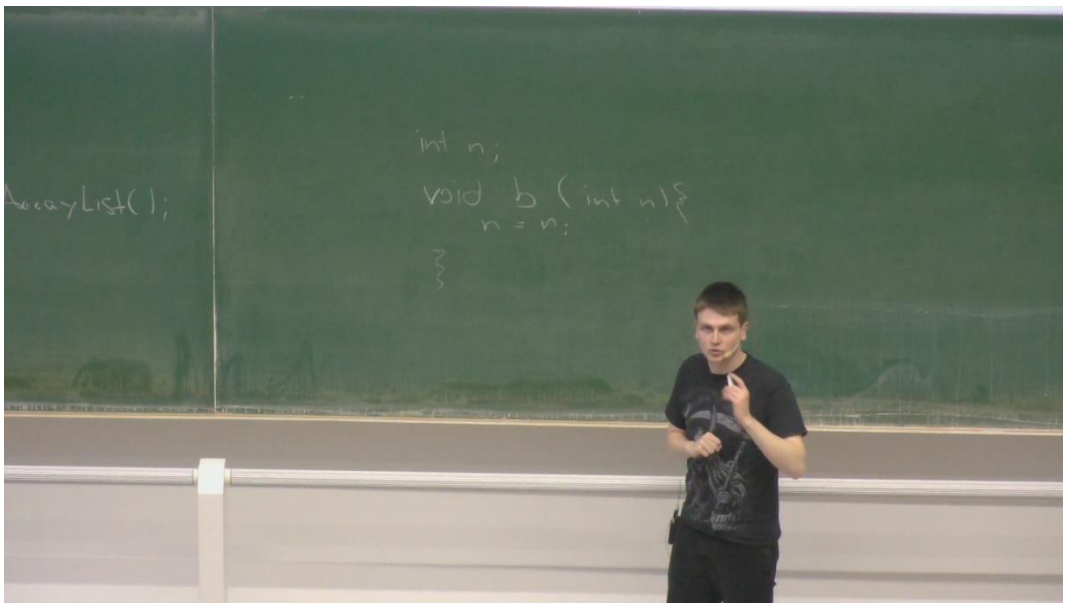

Abbildung 42: Sequenz 2, mit Tracking (Speth, 2016)