



GEORG-SIMON-OHM  
HOCHSCHULE NÜRNBERG

Georg-Simon-Ohm Hochschule Nürnberg

Fakultät Elektrotechnik Feinwerktechnik Informationstechnik efi

Projekt MatRigX

**Prüfungsstudienarbeit von**

**Armin Voit**

**Mat.Nr.: 2087399**

**Media Engineering (B-ME 6)**

**„Prüfungsstudienarbeit für das Projekt MatRigX im  
Rahmen des Studienprojektes in Media Engineering“**

Sommersemester 2012

# **Bestätigung gemäß § 35 (7) RaPO**

Armin Voit

Ich bestätige, dass ich die Prüfungsstudienarbeit mit dem Titel:

„Prüfungsstudienarbeit für das Projekt MatRigX im Rahmen des Studienprojektes in Media Engineering“

selbstständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt, sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Datum: Dienstag, 31. Juli 2012

Unterschrift: Armin Voit

# **Abstract**

Gemäß Paragraph 21, Absatz Eins der Rahmenprüfungsordnung, kurz RaPo, sind Prüfungsstudienarbeiten Prüfungsleistungen mit überwiegend zeichnerischem, gestalterischem oder sonstigem komplexen Inhalt und offenem Lösungsweg zum Nachweis kreativer Fähigkeiten, die sich wegen der umfassenden Aufgabenstellung und der Art der Aufführung in der Regel über einen längeren Zeitraum erstrecken.

Die einzelnen Prüfungsstudienarbeiten des Projektteams, bestehend aus Christian Neubauer, Johannes Kiesel, Christian Siemer und mir, welches sich mit der Realisierung einer Bullet Time mit acht Kameras über den Zeitraum des Sommersemesters 2012 beschäftigt hat, sollen einen individuellen Einblick auf die jeweiligen Anforderung, die es mit Lösungsstrategien zu bewerkstelligen galt, gewährleisten.

Diese, meine eigene Prüfungsstudienarbeit, will Auskunft über die Bedeutung der Rolle meiner Person in der gemeinsamen Teamarbeit verdeutlichen.

# Roadmap

## After Effects

Tätigkeit	Dokument	Beteiligung
Video Rendering der Aufnahmen	../Bilder/ ../Video/	Kiesel
Tracking und Kalibrierung der Frames	../Video/	Kiesel
Interpolation der Aufnahmen durch Twixtor	../Video	Kiesel

## Auslösen der Kameras

Tätigkeit	Dokument	Beteiligung
Recherche CHDK		Siemer/Voit
Installation CHDK	../CHDK/	Siemer/Voit
Erstellen von CHDK Skripten zum Auslösen der Kameras und zur Datenübertragung	../CHDK/	Siemer/Voit
Entwicklung eines Binärzählers	../Arduino/	Siemer
Auslösen mit Arduino	../Arduino/	Siemer

## Kalibrierung/Hardware

Tätigkeit	Dokument	Beteiligung
Konstruktion Multikamerarig		Neubauer
Bau Multikamerarig		Neubauer
Darstellung Grid		Siemer/Voit
Verifikation der Ausrichtungsgenaugkeit		Voit

## Kalibrierung/Software

Tätigkeit	Dokument	Beteiligung
Recherche openframeworks/ openCV	../OpenCV/	Kiesel/

		Neubauer/ Siemer/ Voit
Festlegen der Libraries und Installation	../Code/	Kiesel/ Neubauer/ Voit
Laden der Bilder in openCV	../Code/	Kiesel/ Voit
Farberkennung/ Markererkennung	../Code/	Kiesel/ Neubauer/ Voit
Morphologische Operationen zur Markerisolation	../Code/	Voit
Auslesen der Markerpositionen	../Code/	Kiesel/ Voit
Berechnung der affinen/perspektivischen Transformationsmatrix	../Code/	Kiesel /Neubauer/ Voit
Affine/perspektivische Transformation der Bilder	../Code/	Kiesel/ Neubauer/ Voit
Einbinden des Codes in openframeworks	../Code/	Neubauer
Interpolation der Einzelbilder zu einer Vorschauanimation	../Code/	Neubauer

## Präsentation des Projekts

Tätigkeit	Dokument	Beteiligung
Zwischenpräsentation	../dokumente/	Kiesel/ Neubauer/ Siemer/ Voit
Abschlusspräsentation	../dokumente/	Kiesel/ Neubauer/ Siemer/ Voit

## Projektorganisation

Tätigkeit	Dokument	Beteiligung
Ideenfindung	../dokumente/	Kiesel/ Neubauer/ Siemer/ Voit
Organisation	../dokumente/	Voit

Aufgabenverteilung	..../dokumente/	Voit
Protokollierung	..../dokumente/	Kiesel/ Neubauer/ Siemer/ Voit
Projektplan	..../dokumente/	Voit
Pflichten-/Lastenheft	..../dokumente/	Kiesel/ Neubauer/ Siemer/ Voit
Interne und externe Projektkommunikation	..../dokumente/	Voit
Gesamtdokumentation	..../dokumente/	Siemer/ Voit
Wiki-Dokumentation	Projekt-Server	Voit
Dokumentation des Codes	..../Code/	Kiesel/ Neubauer/ Voit
Materialbeschaffung		Neubauer/ Siemer

# Inhaltsverzeichnis

Bestätigung gemäß § 35 (7) RaPO .....	2
Abstract .....	3
Roadmap .....	4
1. Ausgangssituation .....	9
2. Projektidee .....	9
3. Projektorganisation .....	11
3.1 Rollenverteilung.....	11
3.2 Lastenheft.....	13
3.3 Zeitplan.....	13
3.4 Meetings.....	15
3.5 Blog .....	16
3.6 Dateiverwaltung .....	17
3.7 Wiki .....	18
3.8 Gesamtdokumentation.....	18
4.Grobkalibrierung.....	19
4.1 Recherche: Rigbau .....	19
4.2 Grid .....	21
4.3 Verifikation der Ausrichtungsgenauigkeit des Rigs .....	23
5. Canon Hack Development Kit (CHDK).....	24
5.1 Sprachen .....	25
5.2 Installation.....	26
5.3 Simultane Triggerung .....	28

5.4 Synchronitätsnachweis (ohne Arduino) .....	29
5.5 Datenübertragung.....	30
6. Feinkalibrierung.....	31
6.1 Recherche: openCV & openFrameworks .....	32
6.2 Image-load Funktion .....	33
6.3 Markererkennung.....	33
6.4 Morphologische Operationen.....	35
6.5 Transformationsmatrix.....	36
6.6 Codedokumentation .....	36
7. Vergleich mit professioneller Software .....	37
8. Soll/Ist-Vergleich.....	38
9. Resümee .....	39
Literaturverzeichnis .....	43
Internetquellen .....	43
Abbildungsverzeichnis.....	44

## 1. Ausgangssituation

Ausgangssituation für das Projekt mit dem Namen "MatRigX" war die, im 6. Semester des Studiengangs Media Engineering an der Georg-Simon-Ohm Hochschule vorgesehene Projektarbeit. Ziel der Projektarbeit war die Konzipierung und Realisierung einer Idee gemeinschaftlich innerhalb einer Gruppe. Hierbei wird im Gegensatz zu anderen Studiengängen ein besonderer Schwerpunkt auf die Gruppenarbeit gelegt. Die Projektarbeit erstreckte sich über das komplette Sommersemester 2012.

Angedacht war, dass die Studenten ihr bisher bestehendes Projekt aus dem Wintersemester 2011/2012 weiterführen und vertiefen, jedoch stand es den Studenten letztendlich dabei frei, ob sie ihr Projekt fortführen oder ein neues Thema wählen. Die, aus den Studenten Johannes Kiesel, Christian Neubauer, Christian Siemer und mir, gegründete Gruppe MatRigX hat sich wie jede andere Gruppe des Studiengangs Media Engineering dazu entschieden, ein neues Projekt zu beginnen, um somit den eigenen Horizont erweitern und neue Impressionen aus anderen technischen Bereichen einsammeln zu können.

Als Auftraggeber traten Prof. Dr. Stefan Röttger und Prof. Dr. Matthias Hopf in Erscheinung.

## 2. Projektidee

Seinen Ursprung hatte dieses Projekt im ersten gemeinsamen Treffen der Studenten des Studienfachs Media Engineering aus dem 6. Semester, einer Studentin des 4. Semesters, sowie den betreuenden Professoren, Prof. Dr. Stefan Röttger, Prof. Dr. Matthias Hopf, Prof. Dr. Heinz Brünig,

Prof. Dr.(USA) Ralph Lano und Prof. Dr. Hans-Georg Hopf. Nach einer regen Diskussion und zahlreichen Vorschlägen von Seiten der Studenten, aber auch der Professoren entwickelte sich unser Projekt ausgehend von der Idee eines Studenten. Der Vorschlag bestand in einer Art Kamerafahrt, die einen großen Wert auf eine künstlerische und gestalterische Komponente legte, was jedoch nicht den benötigten technischen Bestandteil aufweisen konnte, den es für den Studiengang bedarf. Diese Idee wurde dann von den Professoren weiterentwickelt und so angepasst, dass sie in einen Projektrahmen passte, der für Media Engineering angemessen war. Die Kamerafahrt blieb bestehen, jedoch wurde das gestalterische Element in den Hintergrund gerückt und der Fokus auf ein Anderes gelegt. Die Idee war nun die Nachbildung einer sogenannten Bullet Time.

Bullet Time(engl. bullet: Projektil und time: Zeit) bezeichnet in der Filmkunst einen Spezialeffekt, bei dem der Eindruck einer Kamerafahrt um ein in der Zeit eingefrorenes Objekt herum entsteht. Dieser Spezialeffekt erlaubt, schnelle Geschehnisse, zum Beispiel fliegende Pistolenkugeln, genau und von verschiedenen Blickwinkeln aus zu sehen. Ebenso sind Verlangsamungen bis hin zum Stillstand und sogar rückwärts laufende Zeit möglich. Neben der Verwendung in Actionfilmen wird Bullet Time auch als Spielelement in Videospielen verwendet.<sup>1</sup>

Wohingegen ein Bullet Time Effekt in einem Spiel relativ einfach erzeugt werden kann, da der 3D Raum meist schon besteht und lediglich die virtuelle Kamera, deren Bild letztendlich der Benutzer sieht, bewegt werden muss, ist bei einer cineastischen Variante der Bullet Time schon etwas mehr Aufwand nötig, um eine Fahrt im zeittoten Raum zu simulieren. Hierbei bedient man sich eines sogenannten Rigs mit einem Array von Kameras. Diese werden auf dem Rig, was frei übersetzt soviel heißt wie Aufbau, nebeneinander gestellt und zeitgleich oder leicht sukzessiv ausgelöst je nach Wunsch. Die eigentliche Kamerafahrt besteht nun darin, von einem Bild zum anderen zu springen, sprich die Bilder der

---

<sup>1</sup> [http://de.wikipedia.org/wiki/Bullet\\_Time](http://de.wikipedia.org/wiki/Bullet_Time) (Stand: 13.07.2012).

Kameras nacheinander ablaufen zu lassen, um so den gewünschten Kamerafahrteffekt zu erzeugen.

Der Film Matrix hat dieses Genre revolutioniert. Noch heute wird beim Betrachter eine Bullet Time mit den berühmten Szenen der Protagonisten des Films assoziiert. An den überragenden Aufnahmen des Films orientierend, haben wir uns ebenfalls dazu entschlossen solch eine Bullet Time zu simulieren.

### 3. Projektorganisation

Eine gelungene Projektdurchführung ist gekennzeichnet von einer akribischen Projektorganisation. Dies erforderte in unserem Fall regelmäßig veranstaltete Meetings, die Errichtung eines Blog, sowie das Erstellen einer Dropbox als Lösung einer gemeinsamen Dateiverwaltung und das Einrichten eines Gits zur Darstellung und Kollaboration unseres gemeinsamen Codes.

Diese gemeinsamen Meetings und Dokumentations- bzw. Organisationstools werden auf den folgenden Seiten näher erläutert.<sup>2</sup>

#### 3.1 Rollenverteilung

Die Rollenvergabe soll erzielen, komplexe Aufgaben, die sich über einen längeren Zeitraum erstrecken, zu strukturieren, um somit die Arbeit in einer Gruppe zu erleichtern und einen harmonischen Arbeitsfluss zu gewährleisten. Ist die Rollenverteilung erfolgt, ist es ebenfalls von Nöten

---

<sup>2</sup> <http://wiki.ohm-hochschule.de/roettger/index.php/Project-R3DS/Projektorganisation>  
(Stand: 02/2012)

die Kommunikation zu seinen Teammitgliedern aufrecht zu halten und neuste Fortschritte bei eigenen Aufgaben mit der Gruppe zu teilen. Werden diese Arbeitstechniken eingehalten, ist der Grundstein für eine gelungene Projektarbeit gesetzt.

Das Team des Projekts „MatRigX“ hat sich ebenfalls für eine, wenn auch nicht so strikte, Rollenvergabe entschieden. Meine Person wurde von den Projektmitgliedern einstimmig zum Projektleiter ernannt.<sup>3</sup> Aufgrund der Tatsache, dass diese die wichtigste Rolle ist, da der Projektleiter nach Außen sowohl Gruppe als auch das Projekt repräsentiert, wurde dieser Part von uns offiziell im ersten Sitzungsprotokoll festgehalten. Die restlichen Rollen wurden nicht formal niedergeschrieben, sondern in gemeinsamer Einigung mündlich definiert. Die gemeinsam ausgearbeitete Aufgabenverteilung sah folgendes vor.

Die Aufgabe eines Jeden von uns war es, darüber hinaus abwechselnd Sitzungsprotokolle der wöchentlich stattfindenden Teammeetings zu erstellen und diese daraufhin in digitaler Form den anderen Mitgliedern zur Verfügung zu stellen. Ebenfalls in den gemeinsamen Verantwortungsbereich fiel die Erstellung und Instandhaltung eines eigenen Blogs, um somit neuste Prozesse schriftlich festzuhalten und der Öffentlichkeit präsentieren zu können, sowie auf Seiten der nicht öffentlichen Arbeit die Errichtung und ständige Aktualisierung einer Dropbox, die alle Teammitglieder, sowie die Betreuer zeitnah auf den gleichen Stand bringt und die gemeinsame Nutzung einer eigens für das Projekt gegründeten Facebook Gruppe zum schnellen Informationsaustausch.<sup>4</sup>

---

<sup>3</sup> 1.Protokol\_10.04.12.pdf, S.1.

<sup>4</sup> <http://wiki.ohm-hochschule.de/roettger/index.php/Project-R3DS/Aufgabenverteilung> (Stand 02/2012)

## 3.2 Lastenheft

Das Lastenheft (teils auch *Anforderungsspezifikation*, *Anforderungskatalog*, *Produktskizze*, *Kundenspezifikation* oder *Requirements Specification*) beschreibt die Gesamtheit der Forderung des Auftraggebers an die Lieferungen und Leistungen eines Auftragnehmers.<sup>5</sup> In einem ersten Meeting hat das Projektteam gemeinsam die Grundlage für das Lastenheft gelegt und Themen spezifiziert. Zusammen mit Christian Siemer habe ich diese Spezifizierung dann in Worte gefasst und ein konkretes Lastenheft erstellt, in dem die Forderungen des Auftraggebers, in unserem Fall Prof. Dr. Stefan Röttger und Prof. Dr. Matthias Hopf, niedergeschrieben wurden.

## 3.3 Zeitplan

Die Erstellung und konsequente Einhaltung eines Zeitplan von Beginn eines Projektes an ist maßgeblich für das Gelingen eben dieses ausschlaggebend. Dabei müssen gleich mehrere Dinge berücksichtigt werden.

Zum einen muss sich der Projektleiter einen Überblick über den Umfang der Projektarbeit verschaffen. Dies kann jedoch nur ein grober Schätzwert werden, da im Laufe eines jeden Projekts immer neue Schwierigkeiten auftreten können, die zu Beginn nicht einsehbar waren und dann viel Zeit in Anspruch nehmen können. Auch Aufgaben, die zu Beginn als schnell vollendbar angesehen werden, können sich im Laufe des Projekts als größeres Hindernis darstellen, was wiederum eine entsprechende Anpassung des Zeitplans zur Folge hat, um gegebenenfalls wichtigere Aufgaben noch dementsprechend zu priorisieren. Natürlich kann auch der

---

<sup>5</sup> <http://de.wikipedia.org/wiki/Lastenheft> (Stand: 15.06.2012).

Fall eintreten, dass eine Aufgabe schneller als geplant abgearbeitet wird. Die, sich daraus ergebenden, frei werden Ressourcen müssen dementsprechend neu verwaltet werden und mit nächsten, eigentlich erst später vorgesehenen, Aufgaben beauftragt werden. Dies ist auch gleich ein weiterer Punkt, der bei der Zeitplanung berücksichtigt werden muss. Wie hoch ist die Anzahl der Ressourcen und über welche Kompetenzen verfügen diese? Zu Beginn muss sich der Projektleiter einen Überblick verschaffen, wie viele Personen er für welches Einsatzgebiet einteilen kann und über welche Kompetenzen, diese Personen verfügen. So ist es von größter Wichtigkeit schon vor dem eigentlichen Projekt im gemeinsamen Gespräch zu eruieren, in welchen Themengebieten die Projektmitglieder am versiertesten sind und so wissensspezifisch Aufgaben zu verteilen, um ein möglichst effektives und effizientes Arbeiten garantieren zu können. Ein Punkt, der sich auch oft, wie z.B. auch in unserem Fall, negativ auf den Zeitplan auswirken kann, sind äußere Faktoren. Hierbei kann sich der Projektleiter und das Team nur sehr schlecht bis gar nicht darauf einstellen, da das Projektteam keinen Einfluss auf diese äußeren Faktoren hat. In unserem Fall hieß das konkret, dass wir erst nach drei Monaten alle unsere Kameras zu Verfügung gestellt bekommen hatten. Da wir keinerlei Einfluss darauf hatten, wann die Kameras letztlich geliefert werden, konnten wir den Zeitplan auch nicht dementsprechend anpassen. Die äußeren Faktoren gelten demnach als das schwerste zu kalkulierende Element bei der Erstellung von Zeitplänen.

Ein Zeitplan ist also ein ständig zu überwachendes Dokument, auf das durchgehend reagiert werden muss.

Als Projektmanager wurde ich mit der Aufgabe vertraut einen Zeitplan für das Projekt MatRigX zu erstellen. Der von mir, gemeinsam mit meinem Team ausgearbeitete, Zeitplan sah vier Phasen für unser Projekt vor. Die erste Phase beschäftigte sich mit dem Rigbau. In dieser Phase wurde

bereits bestehende Rigs betrachtet und bewertet. Im Anschluss daran wurde unser eigenes Rig konzipiert und konstruiert.

Phase zwei sah vor, eine synchrone Auslösung zu bewerkstelligen, sowie die Realisierung einer Fernbedienung mittels dem Open-Source Mirco Controller Arduino.

Die dritte Phase widmete sich der Kalibrierung unserer Bilder. Dies hieß eine eigene Software zu schreiben, mit Hilfe der freien Bibliotheken openCV und openFrameworks. Darin enthalten war eine Markererkennung und das Erstellen einer Transformationsmatrix.

Als vierte und letzte Phase war die Interpolation und die Ausgabe unserer bearbeiteten Bilder vorgesehen. Dazu wurde eine lineare Interpolation, auch Alpha-Blending genannt, erzeugt.

Die vier Phasen sollten dabei aber keineswegs so verstanden werden, dass sie von chronologischer Natur sind.

### 3.4 Meetings

Zur optimierten Kommunikation sowie Stärkung und Verbesserung der Teamarbeit legten wir gemeinsam fest, dass wir wöchentlich ein Treffen abhalten werden, in welchem neuste Ergebnisse präsentiert und wiederum neue Aufgaben verteilt werden können. Die Aufgabe eines Jeden von uns war es, darüber hinaus abwechselnd Sitzungsprotokolle der wöchentlich stattfindenden Teammeetings zu erstellen und diese daraufhin in digitaler Form den anderen Mitgliedern und betreuenden Professoren zur Verfügung zu stellen.<sup>6</sup>

Im späteren Verlauf trafen sich gezielt auch nur Teile des Projektteams, um so explizit die gemeinsame projektbezogene Vertiefungsrichtung zu

---

<sup>6</sup> <http://wiki.ohm-hochschule.de/roettger/index.php/Project-R3DS/Meetings> (Stand: 02/2012).

bearbeiten, z.B. Anwendung der Transformationsmatrix auf die von uns geschossenen Bilder.

Meine Aufgabe bestand im Konkreten in der Organisation und Leitung der Meetings.

## 3.5 Blog

Unsere Arbeit sollte sowohl für jetzige als auch zukünftige Zeiten Informationen über Fortschritt, Probleme, Schwierigkeiten, Lösungswege etc. öffentlich festhalten. Aus diesem Grund wurde uns und allen anderen Projektteams nahe gelegt unseren Projektfortschritt in einem öffentlichen Blog niederzuschreiben.<sup>7</sup>

Der Blog ist unter folgendem Link zu erreichen:

<http://www.matrigx.blogspot.de>

Meine Aufgabe bestand darin, einen Blog zu erstellen und ihn mit den neusten Erkenntnissen und Fortschritten unseres Projektes zu füllen. Da der Blog unser Projekt nach außen hin repräsentiert hat, habe ich stets, sobald ich neue Projektfortschritte erzielt hatte bzw. mir neue Projektfortschritte mitgeteilt wurden, diese daraufhin als neuen Post im Blog angelegt. Oft wurden die Posts auch mit Bildern vom Projekt unterlegt, um sich somit ein besseres Bild vom aktuellen Projektstand machen zu können.

Gegen Ende des Projekts wurde der Blog dann zusätzlich von den restlichen Teammitgliedern gepflegt, damit ich mich ebenfalls verstärkt in die Programmierung unserer Software einarbeiten konnte.

---

<sup>7</sup> <http://schorsch.efi.fh-nuernberg.de/mewiki/index.php/Project-BME-2012-07-MatRigX/Blog> (Stand: 07/2012).

## 3.6 Dateiverwaltung

Um ein effektives und effizientes Kollaborieren zu gewährleisten, entschieden wir uns für die Softwarelösung Dropbox. Dropbox ist eine Internetdienstleistung, mit der man Dateien auf verschiedenen Rechnern synchronisieren und bearbeiten kann. Jede Datei, die man in den freigegebenen Ordner auf dem Rechner speichert, wird sofort auf allen anderen Rechnern, die auch auf diesen Ordner zugreifen können, synchronisiert. Somit ist gewährleistet, dass jedem Teammitglied zu jeder Zeit die gleichen Daten zur Bearbeitung des Projekts zur Verfügung stehen.

Dropbox ist mittlerweile für alle gängigen Plattformen verfügbar. Dazu zählen unter anderem Windows, Mac OS, Linux, Android und iOS. Auch ein Zugriff per Webbrowser auf die Daten ist möglich.<sup>8</sup>

Im späteren Verlauf unseres Projekts wurde uns von unseren betreuenden Professoren geraten, unseren bisherigen Code in einem speziellen Verwaltungssystem, einem sogenannten Git abzulegen, um somit leichter kollaborieren zu können. Daraufhin entschieden wir uns für die Seite Github.com. Github ist ein webbasierter Hosting-Dienst für Software-Entwicklungsprojekte. Er verwendet namensgebenderweise das Versionsverwaltungs-System Git.<sup>9</sup> Meine Aufgabe war es nun ein Git, eine ordentliche Struktur für das Repository zu erstellen und unseren Code hochzuladen. Des Weiteren fiel es in meinen Aufgabenbereich neuere Versionen unseres Codes hochzuladen, um unser Git auf dem neusten Stand zu halten. Unser kompletter Code ist auf dem Git des Projektserver der Hochschule einzusehen

---

<sup>8</sup> <http://schorsch.efi.fh-nuernberg.de/mewiki/index.php/Project-BME-2012-07-MatRigX/Dateiverwaltung> (Stand: 07/2012).

<sup>9</sup> <http://de.wikipedia.org/wiki/GitHub> (Stand: 18.07.2012).

## 3.7 Wiki

Um unseren Projektverlauf und die -ergebnisse nachhaltig festzuhalten und sie für eine breite Masse z.B. nachfolgenden Studenten zugänglich zu machen, haben wir die wichtigsten Punkte in einem Wiki auf dem Projektserver der Ohm Hochschule dokumentiert.<sup>10</sup>

Für die Erstellung des Wikis habe ich mich als Projektleiter bereit erklärt. Wichtig war es hierbei einen Überblick über das Thema zu liefern, ohne sich dabei im Detail zu verlieren. Das Wiki soll als Anlaufstelle für zukünftige Projekte gedacht sein, die sich über unser Projekt informieren und gegebenenfalls darauf aufbauen wollen. Weiterführende und vertiefende Informationen zu unserer Projektarbeit soll die Gesamtdokumentation liefern.

## 3.8 Gesamtdokumentation

Von jedem Studenten soll nach dem Projekt eine sogenannte Prüfungsstudienarbeit erstellt werden. Darin werden die eigenen Leistungen innerhalb des Projektteams dokumentiert.

Die Zusammenfassung aller Leistungen wird in der Gesamtdokumentation geleistet. Diese soll das komplette Projekt möglichst genau dokumentieren.

Für die Erstellung der Gesamtdokumentation habe ich mich zusammen mit Christian Siemer bereit erklärt.

---

<sup>10</sup> <http://schorsch.efi.fh-nuernberg.de/mewiki/index.php/Projects/BME-2012-07-MatRigX> (Stand: 07/2012).

## 4. Grobkalibrierung

Um ein möglichst gutes Endresultat zu erzielen, haben wir uns darauf geeinigt und auch im Lastenheft festgehalten, dass wir unsere aufgenommenen Bilder, vor der Ausgabe als Video, kalibrieren müssen. Dies wurde auch ein Schwerpunkt unseres Projekts. Zuerst überlegten wir wie wir die Bilder kalibrieren könnten. Relativ schnell wurde uns klar, dass wir hierfür zwei Arten der Kalibrierung benötigen. Zuerst sollte eine Grobkalibrierung durchgeführt werden, welche schon vor dem eigentlichen Aufnehmen der Bilder von statten geht. Diese Kalibrierungsart wurde mittels unserer Hardwarekomponenten erzielt. Als zweiter Schritt zur Optimierung unseres Endprodukt, dem fertigen Videos, sahen wir eine Feinkalibrierung vor. Diese wurde dann auf Softwarebasis realisiert.

### 4.1 Recherche: Rigbau

Um unser Ziel, eine Bullet Time zu simulieren, in die Tat umsetzen zu können, benötigten wir eine Vorrichtung, mit der es möglich war, alle acht Kameras gleichzeitig darauf zu platzieren. Eine solche Vorrichtung wird in Fachkreisen Rig genannt. Dieses dient einer äquivalenten Ausrichtung der einzelnen Kameras. Im Bereich der professionellen Filmproduktion bestehen Rigs meist aus Aluminium oder Carbon. Aufgrund der Tatsache, dass diese Materialen sehr leicht sind, jedoch große Robustheit und Präzision aufweisen, erfreuen sich Rigs aus eben diesen Stoffen größter Beliebtheit. Da unser Projekt eine studentisches Projekt war und ohne finanzielle Zuschüsse auskommen musste, somit ein No-Budget-Projekt war, mussten wir einen Kompromiss zwischen Präzision und Preis beim Rigbau beachten. Wir haben uns daraufhin geeinigt, ein Rig aus Holz zu bauen. Nun stand die Frage im Raum, wie das Rig auszusehen hat. Für die

Konzeptionierung und die eigentliche Realisierung wurde Christian Neubauer beauftragt.

Bevor es jedoch soweit war, haben wir uns zuvor betrachtet, welche prinzipiellen Aufbauten es bei Rigs gibt. Nach einiger Recherche konnten wir zwei prinzipielle Bauweisen nachweisen. Zum einen gibt es den geradlinigen Aufbau. Hierbei werden die Kameras nebeneinander auf dem Rig platziert und alle parallel zueinander, in die gleiche Blickrichtung, ausgerichtet.



Abbildung 1: Geradliniges Rig mit parallelen Kameras

Die zweite Variante eines Rigs ist der kreisförmige Aufbau. Hierbei wird das Rig kreisförmig um das abzulichtende Objekt aufgestellt. Die Kameras sind hier, nicht wie zuvor parallel, sondern auf das zentrale Objekt ausgerichtet. Die wahrscheinlich bekannteste Szene, die mit einem zentral gerichteten Rig erstellt wurde ist aus dem Film Matrix.



Abbildung 2: Kreisförmiges Rig aus dem Film "Matrix"

Ausgehend von der Popularität der Matrix Filme und den sensationellen Bildern, die noch ein Jeder im Gedächtnis hat, haben wir uns dafür entschieden, unser Rig als ein Kreisförmiges zu gestalten. Das Rig hatte fortan den Zweck, dass alle Kameras auf einer Ebene ausgerichtet sind und sich in ihrer horizontalen Ausrichtung im Raum parallel zum Rig befinden. Da unser Rig aus Kostengründen aus minderwertigem Holz hergestellt wurde und zusätzlich noch in einem sehr warmen Raum gelagert wurde, konnten wir leider nicht verhindern, dass es sich mit der Zeit stark in seiner Form veränderte. Bedingt durch den aufgeheizten Raum und die starke Sonneneinstrahlung verzog sich das Rig sehr schnell und stark. Dies machte es uns leider schon nach kurzer Zeit immer schwieriger genaue Messungen am Rig durchzuführen und mit dem Rig zu arbeiten, da sich ständig die Resultate unterschieden und die Ergebnisse immer ungenauer wurden.

## 4.2 Grid

Unser fertiges Rig war lediglich für eine horizontale Ausrichtung unserer Kameras ausgelegt. Damit die Kameras das zu aufzunehmende Objekt

möglichst alle im gleichen Bildbereich haben – in unserem Fall stets der Bildmittelpunkt - um ein späteres Springen bei der Bilderfolge zu verhindern, haben wir uns dazu entschieden ein Grid auf der Kamera anzeigen zu lassen. Ein Grid ist ein Raster, das den Bildschirm in einzelne Partitionen unterteilt. Die bekanntesten Grids, oder auch Raster, sind der goldene Schnitt<sup>11</sup> und das 3x3 Grid<sup>12</sup>. Da wir für unsere Zwecke einen möglichst konkreten Punkt des aufzunehmenden Objektes anvisieren und nicht nur das aufzunehmende Objekt in einen bestimmten Bereich auf dem Display einordnen wollten, da uns dies zu ungenau erschien, entschieden wir uns deshalb, ein Grid zu erstellen, welches einem Fadenkreuz gleicht. Die Auswahl und Erstellung des Grids, welches wir letztlich verwendeten wurde von mir vollzogen. Das „Fadenkreuz“ unserer Kameras bestand aus einer horizontalen und einer vertikalen Linie und hatte seinen Schnittpunkt in der Bildmitte.



Abbildung 3: Grid auf dem Display einer Kamera

So konnten wir nun einen konkreten Punkt in unserem Objekt mit allen Kameras anvisieren und somit erzielen, dass das Objekt bei allen Kameras in gleichen Bildschirmbereich auftritt.

---

<sup>11</sup> Definition: Unter Goldener Schnitt versteht man das Teilungsverhältnis einer Strecke, bei der die größere Teilstrecke zur gesamten Strecke im gleichen Verhältnis steht, wie die kleinere Teilstrecke zur Größeren.

<sup>12</sup> Definition: Aufteilung des Bildschirms in 9 gleich große Rechtecke.

## 4.3 Verifikation der Ausrichtungsgenauigkeit des Rigs

Nachdem unser Rig gebaut und unsere Kameras mit Grids versehen wurden, konnte ich eine quantitative Verifikation der Ausrichtungsgenauigkeit unserer Konstruktion vornehmen. Hierfür nahm ich zusammen mit Johannes Kiesel eine Szene mitsamt dem Rig und allen acht Kameras auf. Objekt der Szene war ein Lot, welches in der Mitte einen blauen Marker hatte und orthogonal zum Boden hing. Mit Hilfe der Grids visierten wir diesen blauen Marker mit jeder Kamera an. Nachdem wir die Fotos aufgenommen hatten, hatten wir also acht Einzelbilder von verschiedenen Perspektiven mit jeweils dem gleichen Motiv. Daraufhin bearbeitete ich diese Bilder so, dass sie alle mit halber Transparenz übereinander lagen und ein neues Bild erzeugten.

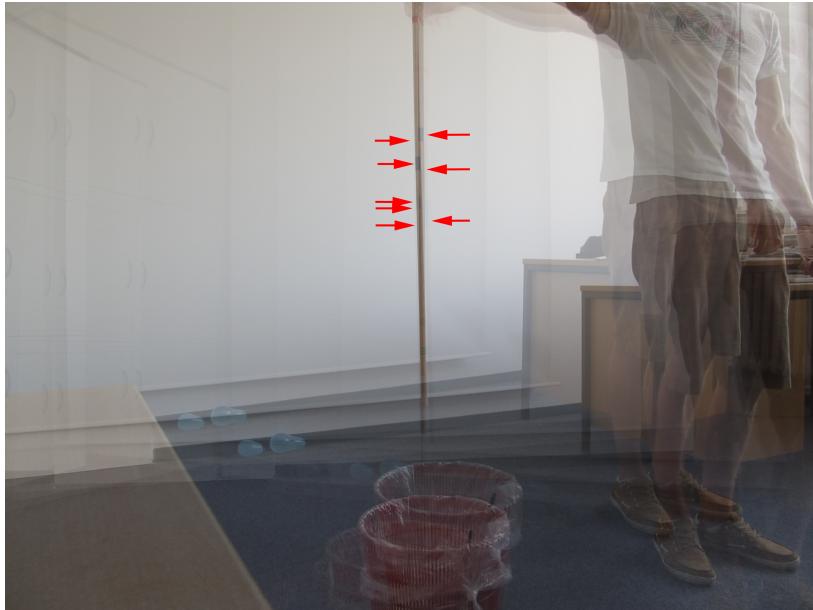


Abbildung 4: Verifikation der Ausrichtungsgenauigkeit des Rigs

Wie in der oberen Abbildung deutlich zu sehen, trat bei den Kameras ein deutlicher Unterschied auf, wo der Marker im Bild zu sehen ist. Dies hatte seine Folgen, in der zuvor beschriebenen Ungenauigkeit des Rigs. Dadurch dass sich das Rig extrem verformt hatte, standen die Kameras leider alle in einem leicht anderen Winkel auf dem Rig. Die roten Pfeile geben dabei

die Lage des Markers wieder. Ein Nachmessen des Rig mit einer Wasserwaage hat ergeben, dass die Kameras einen maximalen Winkelunterschied von 5° Grad aufweisen. Dies ergibt bei einem Abstand der Kameras vom Objekt, hier das Markierungslot, von 2,50 Meter einen maximalen Höhenunterschied der Marker von knapp 22cm. Berechnet wird dies über den Tangens. Da wir den Abstand von Kamera zu Objekt wissen, sowie den Winkelunterschied der Kameras zueinander können wir die Gleichung des Tangens nach der gesuchten Gegenkathete auflösen und erhalten so das Ergebnis von ca. 22cm maximaler Höhenunterschied bei den Markern. Die nächste Abbildung soll dies veranschaulichen.

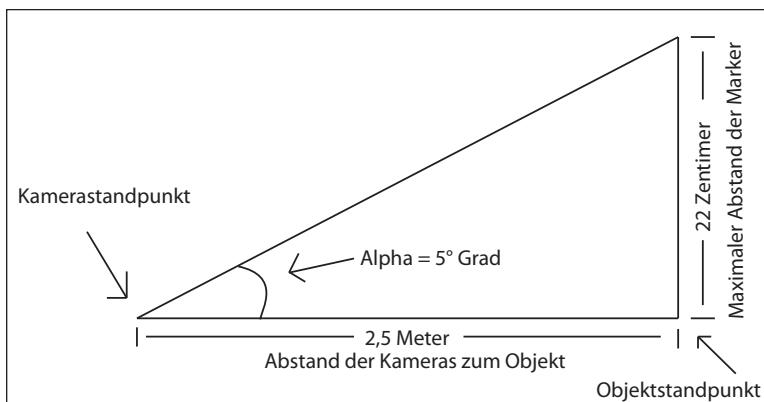


Abbildung 5: Berechnung der Gegenkathete

## 5. Canon Hack Development Kit (CHDK)

Das Canon Hack Development Kit ist ein Firmware-Aufsatz für eine große Anzahl an Canon Kameras, welcher auf Open Source Basis entstand. CHDK wird nicht direkt auf die Kamera, sondern auf eine einführbare Speicherkarte geschrieben, um so die Original Firmware unangetastet zu lassen. Durch die Verwendung von CHDK sind enorme Funktionserweiterungen bei den verwendeten Canon Kameras möglich.<sup>13</sup>

---

<sup>13</sup> [http://chdk.wikia.com/wiki/CHDK\\_in\\_Brief](http://chdk.wikia.com/wiki/CHDK_in_Brief).

So ist es z.B. bei Digitalkameras möglich auf die sonst für den Benutzer nicht zur Verfügung stehenden .raw-Dateien zuzugreifen, anstatt auf die üblichen Bilder im .jpg Format. Für unsere Zwecke war jedoch eine andere Eigenschaft von CHDK von besonderem Nützen. Und zwar haben wir uns die Möglichkeit zu eigen gemacht, mittels CHDK unsere Kamera fernzusteuern. Dazu gleich mehr in Punkt 5.2 Simultane Triggerung.

## 5.1 Sprachen

Wird CHDK auf dem Kamera installiert, so sind schon vorgeschriebene Programme in den Skriptsprachen uBasic und Lua vorhanden. Auch wenn man nun selbst etwas programmieren möchte, hat man die Möglichkeit zwischen beiden Sprachen zu unterscheiden. uBasic war die Skriptsprache, die zusammen mit CHDK eingeführt wurde. Nach einiger Zeit, als die Skripte immer komplexer wurden und die Programmiermöglichkeiten an ihre Grenzen der sehr simplen Sprache uBasic gestoßen sind, fand im April 2008 die Integration von Lua in CHDK statt.<sup>14</sup> Lua ist eine Skriptsprache, die deutliche Vorteile zu uBasic liefert. Dazu zählt z.B., dass nun auch Fehlermeldungen auf dem Display ausgegeben werden können. Dies war mit uBasic bisher nicht möglich. Des Weiteren ist die Syntax konsistenter geworden und erstmals kann man auch eigene Funktionen definieren. Der Nachteil von Lua kommt mit seiner Komplexität und seiner späten Einführung in CHDK. Viele Nutzer von CHDK sind keine versierten Programmierer. Dies ist auch keineswegs notwendig, solange man sich nur mit uBasic beschäftigt. Die Syntax bei uBasic ist auch für einen Nicht-Programmierer verständlich. Jedoch sollten sich Neulinge in CHDK von Anfang an mit der Syntax in Lua anfreunden. Auch wenn es anfangs sehr frustrierend erscheinen mag, eigene Skripte

---

<sup>14</sup> <http://chdk.setepontos.com/index.php/topic,1194.0.html> (Stand 04/2008).

zu erstellen, so sind doch die Möglichkeiten, wenn man erst einmal den Grundaufbau verstanden hat, um einiges größer als mit uBasic.

## 5.2 Installation

Um CHDK auf unseren Canon Powershot SX 130 IS installieren zu können, reichte es nicht aus lediglich den Namen der Kamera zu wissen, sondern man benötigte die konkrete Versionsnummer der darauf gespielten Firmware. Die Versionsnummer ist für den Endkunden eigentlich nicht einsehbar. Um an diese Nummer zu kommen, gibt es verschiedene Möglichkeiten. Ich werde im Folgenden nur die Methode beschreiben, die ich gewählt habe, um unsere Kameras mit der korrekten CHDK Version zu bespielen.

Ich habe mich dazu entschieden das Programm ACID<sup>15</sup> zu benutzen. ACID ist eine plattformunabhängige Java Applikation, die entwickelt wurde, um auf einfache Art herauszufinden, welche Firmware-Version auf einer Canon Kamera installiert ist.<sup>16</sup> Dabei geht man wie folgt vor:

Zuerst wird ein Bild mit der entsprechenden Kamera gemacht von der man die Firmware-Version erfahren möchte. Darauf hin zieht man das Foto via Drag 'n Drop in das Programm, welches nun das Bild analysiert und aus den Exif<sup>17</sup> Daten des Bildes Metadaten zum Kameramodel und dessen Firmware-Version ausliest. Folgend ein Beispielbild zur Version 1.09 von ACID.

---

<sup>15</sup> ACID: **A**utomatic **C**amera **I**entifier and **D**ownloader.

<sup>16</sup> <http://chdk.wikia.com/wiki/ACID>.

<sup>17</sup> Definition: Das **E**xchangeable **I**mage **F**ormat ist ein Standart der Japan Electronic and Information Technology Industries Association für das Dateiformat, in dem moderne Digitalkameras Informationen über die aufgenommenen Bilder (Metadaten) speichern.

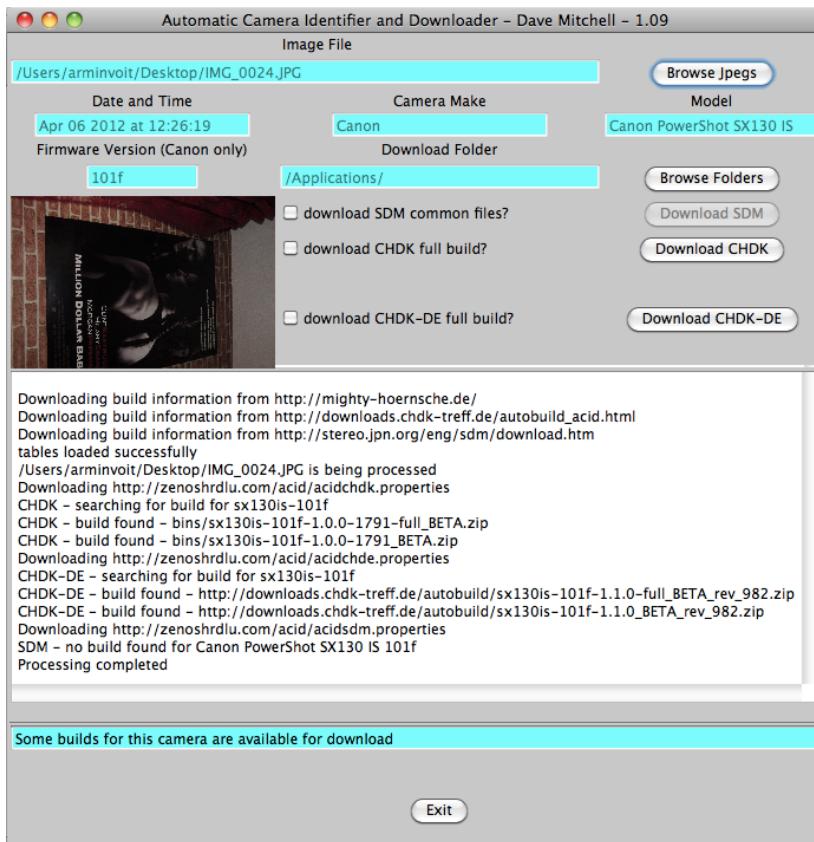


Abbildung 6: ACID nach dem Auslesen der Exif-Daten

Ausgehend von den nun gewonnen Informationen – im konkreten Fall hat die Kamera die Firmware-Version 101f – konnte die entsprechende CHDK Version heruntergeladen werden und auf die SD Karte gespielt.

Jedoch ist das alleinige Kopieren der korrekten CHDK Version auf die SD Karte nicht ausreichend. Damit die Kamera in der Lage ist, den Firmware Aufsatz zu laden, ist es zuvor nötig eine bootfähige SD Karte zu erzeugen. Auch hierfür gibt es mehrere Möglichkeiten. Da ich auf Mac OS gearbeitet habe, habe ich mich hierbei für das Programm SDM Install<sup>18</sup> entschieden. Mit diesem Programm ist es möglich gleichzeitig eine bootfähige Partition auf der SD Karte zu erstellen und CHDK auf der SD Karte zu installieren, so dass hierdurch CHDK auf der Kamera gebootet werden kann. Folgend ein Screenshot der Version 1.24.

<sup>18</sup> SDM: Stereo Data Maker

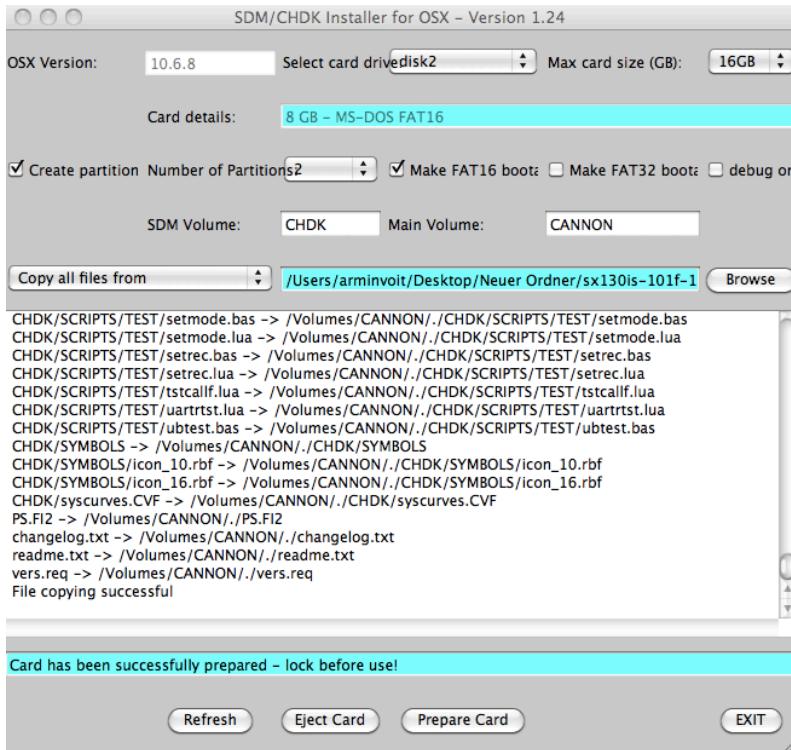


Abbildung 7: SDM/CHDK Installer für OS X

Nach diesen beiden Schritten ist die Kamera nun in der Lage CHDK selbstständig zu booten und anzuwenden. Möchte man CHDK nicht verwenden, es aber auch nicht von der SD Karte löschen, so kann man einfach den SD Lock auf „unlocked“ stellen und die Kamera ist nicht mehr in der Lage CHDK zu laden.

## 5.3 Simultane Triggerung

Hauptgrund für die Verwendung von CHDK für unser Projekt war das Erreichen einer simultanen Triggerung, sprich einem simultanen Auslösen aller Kameras. Um dies zu bewerkstelligen, mussten wir alle Kameras mit dem gleichen Skript ausstatten. Unser CHDK Skript, welches wir letztlich auch bei der Projektpräsentation verwendet haben, ist folgendermaßen aufgebaut.

Zu Beginn werden alle Kameras und CHDK gestartet und in den Aufnahmemodus geschaltet. In diesem Zustand warten die Kameras darauf, das, vorher manuell an der Kamera angewählte, Skript zu starten. Alle Kameras sind über USB Kabel und USB Hubs an einer Stromquelle, in unserem Fall der USB Port des PCs, verbunden, um so gleichzeitig ein Stromimpuls zu erhalten. Wird nun das USB Kabel, welches mit allen Kameras verbunden ist, in den PC eingesteckt. Somit bekommen die Kameras einen Stromimpuls. Mit dieser positiven Spannungsflanke startet das CHDK Skript. Die Kameras beginnen nun alle zu fokussieren und verharren daraufhin in dieser Position. Da der Fokus bei allen Kameras unterschiedlich schnell funktioniert, sollte man hier mindestens 2 Sekunden warten. Wird nun das USB Kabel wieder herausgenommen, führt das dazu, dass der Stromimpuls wieder verschwindet und eine negative Flanke bei den Kameras erscheint. Diese negative Flanke bringt die Kameras in ihrem Skript zum nächsten Schritt und bewirkt, dass alle Kameras auslösen. Da die negative Flanke bei allen Kameras simultan auftritt, wird auch das Foto simultan ausgelöst.

## 5.4 Synchronitätsnachweis (ohne Arduino)

Nachdem ich zusammen mit Christian Siemer das zeitgleiche Auslösen der Kameras realisiert hatte, habe ich mich nun an die Arbeit gemacht, nachzuweisen, wie genau unsere simultane Triggerung wirklich funktioniert. Dazu habe ich erst einen Versuchsaufbau mit zwei Kameras und anschließend einen mit allen Kameras durchgeführt.

Beim ersten Versuch wollte ich herausfinden, ob die Kameras prinzipiell synchron sind. Dazu waren zwei Kameras ausreichend. Denn wären hier schon Probleme aufgetreten, so wäre das Ergebnis mit acht Kameras nicht anders verlaufen. Ich habe nun also zwei Kameras so aufgestellt, dass sie

beide auf eine Stoppuhr<sup>19</sup> gerichtet waren. Das, mit beiden Kameras, geschossene Foto würde nun beweisen, ob die Kameras synchron sind oder nicht. Die beiden Kameras zeigten das exakt gleiche Bild auf die hundertstel Sekunde genau an und somit war bewiesen, dass die Kameras synchron ausgelöst wurden. Jedoch hatten wir eine Sache nicht berücksichtigt. Das Handydisplay arbeitet lediglich mit einer Bildwiederholungsrate von 30 Hertz. Somit war unser Messergebnis lediglich auf ~34ms genau.

Das langte uns allerdings noch nicht und somit suchten wir nach Möglichkeiten unser Messergebnis noch genauer darzustellen. Bei der zweiten Durchführung, die Synchronität nachzuweisen, besorgten wir uns einen Desktopmonitor, der mit einer Bildwiederholungsrate von 60 Hertz arbeitete. Für diesen Test verwendeten wir alle acht Kameras. Auch hier richteten wir wieder die Kameras auf die Stoppuhr<sup>20</sup>. Das Ergebnis war das gleiche wie zuvor mit den zwei Kameras. Auch hier zeigten alle Kameras die exakt gleiche Zeit auf der Stoppuhr an. Somit hatten wir nachgewiesen, dass unsere Kameras mindestens auf 17ms genau auslösten.

Spätere Test bis hin zu 1ms Genauigkeit, wurden dann zusätzlich von Christian Siemer mittels Arduino im späteren Projektverlauf durchgeführt.

## 5.5 Datenübertragung

Das Thema Datenübertragung habe ich ebenfalls mit Christian Siemer bearbeitet. Anfänglich war unser Ziel, einen automatisierten Ablauf mit den Kameras mit Hilfe eines Skripts zu erzeugen. Nach vielen Recherchen im Internet und zahlreichen Skriptvarianten, haben sich zwei

---

<sup>19</sup> Darstellung auf einem Samsung Galaxy S2

<sup>20</sup> Darstellung auf einem Fujitsu Siemens Monitor

entscheidende Probleme herauskristallisiert. Das erste Problem war, beim Versuch alle acht Kameran an den PC anzuschließen, dass dieser stets sobald er eine Kamera erkannt hat, keine weiteren mehr gesucht hat und sich quasi mit einer einzigen zufrieden gegeben hat. Um dieses Problem zu lösen, hätte man Änderungen am USB Treiber vornehmen müssen, da es prinzipiell möglich ist alle acht Kameran gleichzeitig zu erkennen, jedoch hätte eine solche Treibermodifizierung den zeitlichen Rahmen bei Weitem gesprengt. Das zweite Problem, welches sich aufgetan hat war, dass es zwar prinzipiell möglich war ein Skript zum Auslösen der Kameran und ein Skript zur Bildübertragung in einem gemeinsamen Skript zu kombinieren, jedoch gab es einen entscheidenden Faktor der sich zu diesem Vorhaben antagonistisch verhielt. Eine einzelne Einstellung in der Kamera verhindert nämlich, dass das Auslösen und Übertragen in einem „Rutsch“ möglich ist. Möchte man die Kameran auslösen, so muss man direkt in den CHDK Einstellungen auf der Kamera unter dem Punkt *Skripteinstellungen* den Punkt *Fernbedienung* deaktivieren. Möchte man nun das geschossene Bild auf seinen Rechner ziehen, so ist dies nur möglich, wenn man den eben beschriebenen Punkt aktiviert.

Allein über das Skript hat man keine Möglichkeit auf diesen Punkt zuzugreifen. Aus diesen Gründen habe wir uns dazu entschlossen, die Bildübertragung, nach dem Schießen der Fotos semiautomatisch ablaufen zu lassen. Dabei hatten wir sukzessiv für jede Kamera den Punkt *Fernbedienung* aktiviert und waren in der Lage die Bilder der Kameran herunterzuladen ohne dabei ein uneffektives Entnehmen der Speicherplatte vorzunehmen.

## 6. Feinkalibrierung

Da einer unserer Projektschwerpunkte auf der Kalibrierung der aufgenommenen Bilder lag haben wir natürlich schon mit dem Rig eine

Grobkalibrierung vorgenommen. Da diese nur sehr grob war, führte kein Weg daran vorbei, noch eine softwarebasierte Feinkalibrierung auf die Bilder anzuwenden. Ziel war es ein Springen des Objektes von Bild zu Bild möglichst zu vermeiden. Unser Programm sollte dabei wie folgt ablaufen. Zu Beginn wird die Szene noch ohne Objekt aber mit einem Marker abfotografiert. Dieser Marker dient dann später dazu, alle Bilder mit Objekt auf die gleiche Position zu transferieren. Im Anschluss an die Fotos mit den Markern werden die Fotos mit dem Objekt gemacht. Danach werden die Bilder von den Kameras sukzessiv auf den Rechner in einen gemeinsamen Ordner geladen. Von diesem Ordner aus, werden die Bilder automatisch von unserem Programm geladen, transformiert und in einem Video ausgegeben.

## 6.1 Recherche: openCV & openFrameworks

Für unser Projekt wurde von unserem Auftraggeber der Wunsch geäußert, unsere Kalibrierungssoftware mit den freien Bibliotheken openCV und openFrameworks zu realisieren. Da keiner von uns bisher mit diesen Bibliotheken gearbeitet hat, hat die Recherche- und Einarbeitungsphase relativ viel Zeit in Anspruch genommen. Zu Beginn unseres Projekts war auch noch nicht klar, dass wir mit diesen Bibliotheken arbeiten werden, weshalb diese Phase auch relativ spät erst in Angriff genommen wurde. Ich habe mir, wie meine Teammitglieder auch, sehr schwer getan bei den ersten Versuchen mit openCV und openFrameworks. Allein alle Komponenten der Bibliotheken zu installieren und diese wiederum korrekt in Xcode zu integrieren und zu verknüpfen, war ein anfangs sehr frustrierender und langwieriger Prozess. Erst nach zwei Wochen haben wir es geschafft auf allen Mac Rechnern die beiden Bibliotheken erfolgreich einzubinden. Nachdem nun erste Beispielprogramme geverviewt wurden

und wir ganz langsam ein Gefühl für die C++ Syntax bekamen, konnten wir uns daran wagen, erste Komponenten unserer Software zu schreiben.

## 6.2 Image-load Funktion

Da wir einen komplett automatisierten Prozess innerhalb unserer Software angestrebt hatten, war es unser Ziel, dass natürlich auch das Laden der Bilder voll automatisch abläuft. Hierzu haben Johannes und ich eine Image-load Funktion geschrieben, welche in einer Schleife alle Bilder aus dem Bilderordner lädt. Hierzu laden wir die Bilder mit dem Objekt in einer ersten und die Bilder mit den Markern in einer zweiten Schleife. Damit die Bilder nicht verloren gehen, werden sie in einer Array namens *buffer\_mrk[]* und *buffer\_src[]* temporär gespeichert.

## 6.3 Markererkennung

Die Markererkennung ist der Ausgangspunkt unserer Software. Zu Beginn haben wir uns überlegt, wie wir es am Besten schaffen, dass das spätere Video keinerlei Sprünge aufweist. Dazu brauchten wir die Information, wo genau das Objekt auf jedem Bild vorkommt, um anschließend die Bilder aufeinander anzupassen. Wir entschieden uns dafür, die gesuchten Werte mittels Farbmarker aus dem Bild auszulesen. Nachdem wir dieses Vorgehen gemeinsam entschieden hatten, machten sich Johannes Kiesel und ich an die Umsetzung einer Markererkennung.

Die Markererkennung wurde von uns dabei wie folgt konzipiert:

Zuerst wird ein, zuvor mit der Kamera aufgenommenes Bild, welches sich im RGB-Farbraum befindet, in den HSV<sup>21</sup>-Farbraum übersetzt. Dies hat den Vorteil, dass die Markererkennung robuster wird und weniger fehleranfällig bei unterschiedlichen Lichtverhältnissen ist, als im RGB-Farbraum. Anschließend werden unsere vier unterschiedlich farbigen Marker aus dem Bild gefiltert und jeweils in einem eigenen Vorschaufenster dargestellt. Die Darstellung erfolgt dabei als Binärbild. Dies bedeutet, dass auf den Vorschaufenstern lediglich zwei Werte (S/W) zu sehen sind. Die Marker werden weiß dargestellt und alles andere, was nicht zum Marker gehört schwarz. Diese Kontrolle hat den Vorteil, dass man im Vorschaufenster genau erkennen kann, welcher Marker von der Software gut erkannt wird und welcher nicht.

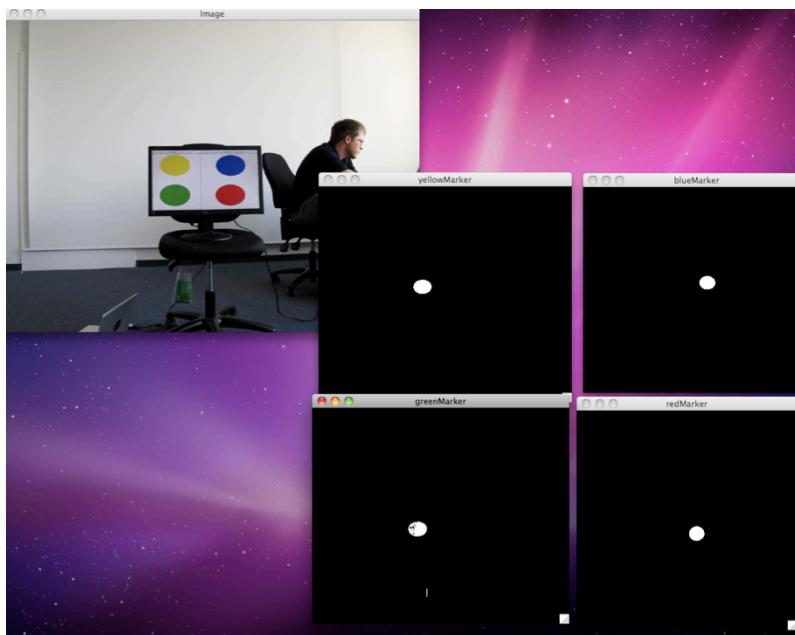


Abbildung 8: Markererkennung

Wie im obigen Bild deutlich zu erkennen, werden die vier Marker sehr gut von unserer Software ausgelesen. Anfangs haben wir noch mit drei Markern auf einem Stab gearbeitet. Dies hat sich jedoch als ungeeignet für die Berechnung der späteren Matrix herausgestellt, da uns so eine Dimension gefehlt hat.

---

<sup>21</sup> HSV: **Hue**, **Saturation**, **Value**

## 6.4 Morphologische Operationen

Obwohl unsere Markererkennung schon sehr gut funktionierte, war das Ergebnis immer noch verbesserungsfähig. So tauchten in den binären Vorschaubildschirmen der einzelnen Marker vereinzelt Störpixel auf, welche nicht zu den Markern gehörten. Dies lag daran, dass sich der Farbton der Marker teilweise auch in anderen Elementen, die auf dem Bild im Hintergrund zu sehen waren, wiederspiegeln.

Um diese Störpixel zu entfernen, bediente ich mich der morphologischen Operatoren `cvErode()` und `cvDilate()`. Diese Operatoren sind standartmäßig 3x3-Masken.

Bei der Funktion `cvErode()` fährt nun die Maske über jeden Pixel im Bild. Mit dem mittleren Pixel in der Maske wird der aktuelle Pixel im Bild ausgewählt. Sind nun alle anderen acht Pixel der Maske mit weißen Pixeln belegt, so bleibt der ausgewählte Pixel weiß. Ist dieser schwarz, so wird er auf weiß gesetzt. Ist auch nur ein Pixel der äußeren acht Pixel auf der Maske schwarz, also nicht belegt, so wird der ausgewählte Pixel im Maskeninneren auf schwarz gesetzt, sofern er weiß war. War er schon schwarz, so ändert sich nichts.

Die Funktion `cvDilate()` funktioniert genau entgegengesetzt wie `cvErode()`. Auch hier fährt die 3x3-Maske über jeden einzelnen Pixel im Bild. Mit dem mittleren Pixel in der Maske wird wieder der aktuelle Pixel im Bild ausgewählt. Ist nun mindestens ein Pixel der äußeren acht Pixel der Maske auf weiß gesetzt, so wird der ausgewählte Pixel ebenfalls auf weiß gesetzt. Ist kein Pixel der äußeren acht Pixel der Maske mit weiß belegt, so wird der ausgewählte Pixel in der Mitte der Maske auf schwarz gesetzt. Durch eine Verbindung dieser zwei Operatoren lässt sich sicherstellen, dass kleinere Störpixel beseitigt, jedoch der eigentlich Marker nicht wesentlich verändert wird.

## 6.5 Transformationsmatrix

Nachdem wir die Markererkennung realisiert hatten, galt es nun die daraus gewonnenen Informationen auf unsere Kalibrierung anzuwenden. Die Transformationsmatrix wurde dabei auf folgende Weise gewonnen: Zuerst berechneten wir den Schwerpunkt aus den gefilterten Markern. Mit diesen Schwerpunkten hatten wir nun in jedem der acht Bilder vier fixe Werte. Um eine Matrix zu gewinnen, mussten wir nun diese Fixwerte in Relation zueinander stellen. Dies geschah, indem wir die vier Schwerpunkte der Marker eines Bildes mittels der Funktion *getAffineTransform()* auf die vier Schwerpunkte der Marker des vorherigen Bildes<sup>22</sup> transferierten. Dabei erfuhren die veränderten Schwerpunkte eine Translation, eine Rotation, eine Skalierung und eine Scherung.

Diese Operationen wurden dann in einer Matrix, in unserem Fall *map\_matrix1* gespeichert und so auf jeden einzelnen Bildpunkt des Bildes angewendet. Dieser Vorgang wurde dann bei jedem Bild wiederholt. Da das erste Bild in der Bildabfolge keinen Vorgänger hat, wurde auf dieses Bild keine Matrix angewendet, sprich es blieb auch bei der Vorschau völlig unverändert.

## 6.6 Codedokumentation

Eine weitere Aufgabe meinerseits bestand darin, zusammen mit Christian Neubauer und Johannes Kiesel unseren Code ordnungsgemäß zu dokumentieren. Prof. Dr. Ralph Lano sagte einst: „Codedokumentation müsse sich wie Prosa lesen. Auch jemand, der keine Ahnung vom Programmieren hat, müsse anhand der Dokumentation verstehen, was in dem Programm, wann vor sich geht.“

---

<sup>22</sup> „Vorherig“ bezieht sich hierbei auf die aneinander gereihte finale Abfolge der Bilder als Film

Daran orientierend, haben wir unseren Code so gut und ausführlich wie möglich dokumentiert, um auch für zukünftige Projektteams, die eventuell auf unserem Code aufbauen oder auch nur Teile davon verwenden möchten, einen leichten Einstieg in unsere Software zu ermöglichen.

## 7. Vergleich mit professioneller Software

Das Ziel unseres Projektes war ein Vergleich unseres kalibrierten und linear interpolierten Videos mit einem, mit professioneller Software erstellten, Video.

Beide Videos konnten im Laufe des Projekts erfolgreich erzeugt werden. Beim Vergleich beider Videos wird ersichtlich, dass sich unsere Kalibrierung nur wenig von der Kalibrierung, die professionell mit Adobe After Effects erstellt wurde, unterscheidet. Jedoch weist die Kalibrierung in After Effects bei qualitativer Betrachtung einen höheren Grad an Genauigkeit auf.

Bei der linearen Interpolation der beiden Videos sind keine qualitativen Unterschiede zu erkennen.

In Anbetracht der Tatsache, dass wir unser Produkt an einer professionellen Software gemessen haben, kann sich unser Ergebnis durchaus sehen lassen. Für weitere Projekte in diesem Bereich, bieten wir hiermit eine solide Basis zur Weiterentwicklung.

## 8. Soll/Ist-Vergleich

SOLL	IST
Rig für min. 8 Kameras	✓
CHDK Skript - Auslösen - Bildübertragung	✓ ✓
Hardware-Kalibrierung - Grid (Sucherzielkreuz) - Nachweis Synchronität	✓ ✓
Software-Kalibrierung - Markererkennung - Übereinanderlegen der Bilder - lineare Interpolation mit Alphablending	✓ ✓ ✓
Qualitativer Vergleich mit professioneller Software	✓
Quantitativer Vergleich mit professioneller Software	✗
Affine Transformation	✓
Perspektivische Transformation	✗
<b>optional</b>	
Fernauslöser mit Hilfe von Arduino	✓

Ausgehend vom Projektplan, haben wir alle bis auf die Anwendung der perspektivischen Transformation, sowie dem quantifizierten Vergleich des professionellen Videos mit unserem, durch unsere Software produziertem, Video, alle unsere gesteckten Projektziele erreicht.

## 9. Resümee

Dieses Resümee soll versuchen, das komplette Projekt, mit all seinen Facetten und besonders meine Erfahrungen bezüglich des Projekts und den Projektmitgliedern zu reflektieren.

Schon unser Projektstart verlief ziemlich zäh und sollte sich auch über das komplette Projekt hinweg erstrecken. Das Semester begann Mitte März, jedoch wurde ein erstes Treffen mit den betreuenden Professoren, zur Ideenfindung und -vorstellung erst Ende März einberufen. Damals noch mit anderen Ideen und möglichen Projektgruppen, wurden unsere Vorschläge jedoch schnell als nicht projekttauglich abgestuft. Nachdem kein Vorschlag unsererseits von den Professoren akzeptiert wurde, haben wir einen vielversprechenden Projektvorschlag von Herrn Röttger, der ursprünglich in stark abgeänderter Version von Benjamin Kuckuk stammte, als zu bearbeitendes Projekt ausgewählt.

Da das Projekt als solches nicht von uns kam, hatten wir auch diesbezüglich keine konkreten Projektziele vor Augen zu diesem Zeitpunkt. Auch zwei Wochen später, bis zum ersten Treffen mit unserem Betreuer, war nicht wirklich klar, ob das Projekt wirklich umgesetzt wird und werden kann. Zwischenzeitlich haben wir uns sogar schon Gedanken gemacht über alternative Projektideen, da wir eben bis zu diesem Zeitpunkt nicht wussten, was wir bei diesem Projekt erreichen sollen und ob diesbezüglich überhaupt Equipment vorrätig sein wird.

Nach dem ersten Treffen wurden uns dann erste Projektziele definiert und versprochen, dass in kürzester Zeit acht Kameras für unser Projekt bereitstehen. Dies war Anfang April.

Mitte April bekamen wir dann die erste Kamera. Nach einem zähen Start konnten wir also leider auch nicht schneller voranschreiten, da uns das Equipment fehlte. Wir machten die Arbeiten, die mit einer Kamera möglich waren, wie das Auslösen und die Datenübertragung. Als Mitte April die zweite Kamera eintraf, konnten wir erst mit den Tests beginnen, ob

unsere Kameras synchron auslösen. In der Zwischenzeit hatten wir unser Rig gebaut und alle Formalitäten wie Lastenheft, Zeitplan, Recherchen etc. abwickeln können. Jedoch verschaffte uns das Warten auf unsere Projektutensilien einen immer größer werdenden Abstand im Projektfortschritt zu den anderen Gruppen. Des Weiteren schwand die Motivation stetig bei mir und meinen Teammitgliedern, aufgrund der Tatsache, dass wir einerseits Ergebnisse erzielen wollten, andererseits diese Ergebnisse nur theoretisch vorbereiten und nicht praktisch ermitteln konnten. Erst Mitte Juni, also drei Monate nach dem offiziellen Semesterstart, trafen die restlichen sechs Kameras ein. Für Mitte Juli war die Abschlusspräsentation angesetzt. Dies hatte zu bedeuten, dass wir lediglich einen Monat Zeit hatten, unser Rig, unsere Software, unserer Arduino Controller und die synchrone Auslösung gleichzeitig mit allen acht Kameras zu testen. Natürlich tauchten in dieser Phase viele Fehler auf, die speziell mit allen Kameras erst entdeckt werden konnten, auf die wir dann aufgrund der stark fortgeschrittenen Zeit schnellstens reagieren mussten. So war z.B. der Arduino Controller mit zwei Kameras bestens gelaufen, für acht Kameras, mit insgesamt drei zwischengeschalteten USB Hubs ist jedoch nicht mehr genug Strom an den Kameras angekommen, um hier eine entsprechende positive Flanke zu erzeugen. Auch konnten erste Aufnahmen von aussagekräftigen Motiven erst zu diesem Zeitpunkt gestartet werden, da man mit zwei Kameras im Grunde nur statische Objekte aufnehmen kann, um einen Bullet Time Effekt zu erzielen. Durch diese Fehler, von denen wir überhaupt erst sehr spät erfahren konnten, schoss unser Arbeitspensum natürlich in die Höhe. Der Aufwand bei der Fehlersuche, die wir in der Mitte des Semesters eingeplant hatten und für die wir zu diesem späten Zeitpunkt eigentlich keine Zeit hatten, ließ unser Projekt nur sehr langsam voranschreiten. Zu diesem Zeitpunkt war die gesamte Stimmung im Team sehr schlecht. Die schier unendliche Arbeit, die sich im letzten Monat des Projekts aufgetan hat und die sich bis zum Schluss aufstaute, da sich der Liefertermin der Kameras immer wieder nach hinten verschob, bis kurz vor den Prüfungszeitraum, ließ unser aller

Motivation rapide sinken. Nichtsdestotrotz haben wir uns den sehr widrigen äußereren Umständen gestellt, ihnen getrotzt und unser Projekt noch zu einem durchaus vertretbaren und respektablen Ende geführt.

Die Teamarbeit empfand ich stets als sehr angenehm. Die Harmonie im Team hat gestimmt, auch wenn es manchmal kleinere Unstimmigkeiten gab, wurden diese immer schnell behoben, um ein effektives und effizientes kollaboratives Arbeiten zu gewährleisten.

Die nicht ganz so erfolgreich verlaufene Kommunikation sowohl mit den Professoren als auch teamintern, ist wohl darauf zurück zu führen, dass ich mich zu stark in die meisten Projektarbeiten vertieft habe. Anstatt dass ich mich als Projektleiter ausschließlich auf die Organisation des Projektes konzentrierte, wie dies in anderen Projektengruppen der Fall war, habe ich mich verstärkt auch im Bereich der Programmierung eingebracht, um einen schnelleren Fortschritt des Projekts zu erzwingen. Dies führte manchmal dazu, dass mir die Arbeit über den Kopf wuchs und ich nicht mehr meinen Pflichten als Projektleiter nachkommen konnte. Ich habe stets versucht allen meinen Aufgaben und darüber hinaus gerecht zu werden, nicht immer mit Erfolg, aber zumindest im Großteil des Projekts.

Rückblickend kann man sagen, dass das Projekt sehr schleppend angefangen hat, bedingt durch mehrere Faktoren. Im Laufe des Semesters hat es etwas stagniert. Die Ursachen hierfür wurden bereits ausführlich beschrieben. Die Arbeitsleistung schoss jedoch gegen Ende gleich einer exponentiellen Leistungskurve nach oben. Gemittelt über das ganze Semester ergab dies weit über die, für ein solches Projekt angedachten, 300 Mannstunden Arbeit meinerseits. Dies kann ich auch von meinen übrigen Teammitgliedern behaupten. Der Aufwand, den wir für dieses Projekt aufgewendet haben, mag, verglichen mit den anderen Projektteams, anfangs noch hinter denen gelegen haben, jedoch gegen Ende haben wir unseren Rückstand wieder aufgeholt. Jeder bei uns im

Team hat durchgehend mitgearbeitet. Von unseren gesteckten Zielen haben wir den Großteil erreicht.

Als diese Faktoren sollten auch bei einer objektiven Beurteilung dieser Projektarbeit ins Gewicht fallen.

## Literaturverzeichnis

Bradski, Gary/ Kaehler, Adrian: Learning openCV, Sebastopol 2008,  
O'Reilly Media

## Internetquellen

[http://de.wikipedia.org/wiki/Bullet\\_Time](http://de.wikipedia.org/wiki/Bullet_Time) (Stand: 13.07.2012).

<http://wiki.ohm-hochschule.de/roettger/index.php/Project-R3DS/Projektorganisation> (Stand: 02/2012)

<http://wiki.ohm-hochschule.de/roettger/index.php/Project-R3DS/Projektorganisation> (Stand: 02/2012)

<http://wiki.ohm-hochschule.de/roettger/index.php/Project-R3DS/Aufgabenverteilung> (Stand 02/2012)

<http://wiki.ohm-hochschule.de/roettger/index.php/Project-R3DS/Meetings> (Stand: 02/2012).

<http://schorsch.efi.fh-nuernberg.de/mewiki/index.php/Project-BME-2012-07-MatRigX/Blog> (Stand: 07/2012).

<http://schorsch.efi.fh-nuernberg.de/mewiki/index.php/Project-BME-2012-07-MatRigX/Dateiverwaltung> (Stand: 07/2012)

<http://de.wikipedia.org/wiki/GitHub> (Stand: 18.07.2012)

<http://schorsch.efi.fh-nuernberg.de/mewiki/index.php/Projects/BME-2012-07-MatRigX> (Stand: 07/2012)

[http://chdk.wikia.com/wiki/CHDK\\_in\\_Brief](http://chdk.wikia.com/wiki/CHDK_in_Brief). (Stand: k. A.)

<http://chdk.setepontos.com/index.php/topic,1194.0.html> (Stand 04/2008).

<http://chdk.wikia.com/wiki/ACID>. (Stand: k. A.)

# Abbildungsverzeichnis

Abbildung 1: Geraemliniges Rig mit parallelen Kameras (Quelle: <a href="http://files.petapixel.com/assets/uploads/2010/10/5darray.jpg">http://files.petapixel.com/assets/uploads/2010/10/5darray.jpg</a> [Stand: k. A.]) .....	20
Abbildung 2: Kreisförmiges Rig aus dem Film "Matrix" (Quelle: <a href="http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/Pelicula_s/FX/imagenes/ejemplos/bulletTimeMatrix1.gif">http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/Pelicula_s/FX/imagenes/ejemplos/bulletTimeMatrix1.gif</a> [Stand: k. A.] .....	21
Abbildung 3: Grid auf dem Display einer Kamera (Quelle: Intern, 2012)	22
Abbildung 4: Verifikation der Ausrichtungsgenauigkeit des Rigs (Quelle: Intern, 2012) .....	23
Abbildung 5: Berechnung der Gegenkathete (Quelle: Intern, 2012).....	24
Abbildung 6: ACID nach dem Auslesen der Exif-Daten (Quelle: Intern, 2012) .....	27
Abbildung 7: SDM/CHDK Installer für OS X (Quelle: Intern, 2012) .....	28
Abbildung 8: Markererkennung (Quelle: Intern, 2012).....	34