



Projektdokumentation

MArC
Mixed Reality Architecture Composer

von

Laura Anger (Matrikelnr. 11086356)
Vera Brockmeyer (Matrikelnr. xxxxxxxxxxxx)
Paul Berning (Matrikelnr. xxxxxxxxxxxx)
Lukas Kolhagen (Matrikelnr. 11084355)

Durchgeführt im
Master Medientechnologie
im SS 2016 und WS 2016/17

Betreuer:

Prof. Dr. Stefan Michael Grünvogel
Institut für Medien- und Phototechnik

Inhaltsverzeichnis

1 Einleitung	4
1.1 Anwendungskontext	4
1.2 Projektziel	4
2 Grundlagen	4
3 Projektmanagement	4
4 Materialien	4
4.1 Hardware	4
4.1.1 Computer zur Ausführung der Unity-Simulation	4
4.1.2 Computer zur Ausführung der Tracking-Anwendung	5
4.1.3 HTC Vive	5
4.1.4 Spielfeldkamera	5
4.1.5 Leap Motion	5
4.2 Marker	5
4.2.1 ArUco Marker	6
4.2.2 Trackingmarker	6
4.3 Obsolete Hardware	6
4.3.1 Ovrvision Pro	7
4.3.2 Webcam	8
4.4 Software	8
4.4.1 Unity	8
4.4.2 Visual Studio	8
4.4.3 OpenCV	9
4.4.4 Leap Motion SDK	9
4.4.5 Orion Beta Software	9
4.4.6 uEye SDK	9
4.4.7 Steam VR	9
5 System	10
5.1 Starten des Systems	10
5.2 Menüführung	10

5.2.1	Menüs	10
5.2.2	Ablauf der Menüführung	11
5.3	Kalibrierung	13
5.3.1	Kamerakalibrierung	13
5.3.2	Kalibrierung des Arbeitsbereichs	16
5.3.3	Kalibrierungsfehler	16
5.3.4	Mögliche Fehlerquellen	16
6	Auswertung	16
6.1	Testing durch unerfahrene Benutzer	16
6.2	Testing durch Architekten	16
7	Zusammenfassung	16

1 Einleitung

Virtuelle und erweiterte Realität ist bereits seit einiger Zeit in aller Munde. Mit dem Erscheinen von Oculus Rift, HTC Vive und anderen Virtual-Reality-Headsets rückt eine neue Art der Immersion beim Genuss von Videospielen in greifbare Nähe. Wenn es jedoch um die Nutzung dieser Technologien zur Effizienzsteigerung in professionellen Umgebungen geht, so sind verfügbare Anwendungen bisher noch Mangelware. Die Entwicklung von MArC, einem Mixed-Reality-System für die architektonische Planung bei Siedlungsbauten, soll dies ändern.

1.1 Anwendungskontext

Die frühe Konzeptionierung zur Erschließung von Wohngebieten findet heute in Architekturbüros meist noch so statt wie vor [??] Jahren. Dazu werden simple Modelle aus leicht zu verarbeitenden Materialien – wie etwa Styropor – erstellt und als Platzhalter für die zu planenden Gebäude bei dem Entwurf verwendet.

Diese Herangehensweise macht Änderungen an den Gebäuden aufwendig und führt zu dem Umstand, dass ein bestimmter Zustand der Planung nur umständlich wiederhergestellt werden kann – zum Beispiel durch Fotografieren und späteren manuellen Wiederaufbau.

1.2 Projektziel

2 Grundlagen

- Wissenschaftlicher Stand

3 Projektmanagement

4 Materialien

4.1 Hardware

Zur Ausführung der MArC-Software sind diverse Hardware-Komponenten Voraussetzung. Diese Komponenten werden nachfolgend beschrieben und deren Kontext im System näher erläutert.

4.1.1 Computer zur Ausführung der Unity-Simulation

Die Anwendung, welche aus Unity^[13] heraus erstellt wurde, benötigt einen Host-Computer, welcher sowohl mit dem HTC Vive Head-Mounted Display kompatibel,

als auch leistungsstark genug sein muss, um das Rendering der Simulation mit ausreichend hoher Bildrate ausführen zu können.

4.1.2 Computer zur Ausführung der Tracking-Anwendung

4.1.3 HTC Vive

Bei der HTC Vive handelt es sich um ein Head-Mounted Display, welches von HTC Laura in Kooperation mit Valve^[17] produziert wird. Vorgestellt wurde dieses am 1. März 2015 im Vorfeld der Mobile World Congress^[7].

Die Auflösung des Displays beträgt insgesamt 2160×1200 Pixel, was $1080x1200$ Pixeln pro Auge entspricht. Die Brille bietet ein Sichtfeld von bis zu 110° bei einer Bildwiederholrate von 90 Hz ^[3]. Zur Positionsbestimmung im Raum wird die Lighthousetechnologie von Valve genutzt. Zusätzlich sind neben einem Gyrosensor auch ein Beschleunigungsmesser und ein Laser-Positionsmesser verbaut. Mittels speziellen Game-Controllern wird eine Interaktion mit virtuellen Objekten ermöglicht.

4.1.4 Spielfeldkamera

IDS uEye 164xLE

Vera

4.1.5 Leap Motion

Bei der Leap Motion^[4] handelt sich um ein $7,6 \times 3 \times 1,3\text{ cm}$ großes Gerät, welches Laura es mit Hilfe von Sensoren möglich macht, Hand- und Fingerbewegungen als Eingabemöglichkeit zu nutzen. Die Idee dahinter ist, eine Eingabegerät analog zu Maus zu schaffen, welches keinen direkten Kontakt bzw. keine Berührung benötigt. Hergestellt wird die Leap Motion von der Firma Leap Motion, Inc., die ihren Hauptsitz in Amerika hat. Gegründet wurde die Firma am 1. November 2010.

Wie auf Abbildung 2 erahnt werden kann, besteht das Gerät im wesentlichen aus zwei integrierten weitwinkel Kameras und drei einfachen Infrarot LEDs. Die LEDs haben jeweils eine Wellenlänge von 850 nm . Der durch die beiden Kameras aufgespannte Interaktionsraum der Leap Motion ähnelt einer umgedrehten Pyramide, mit einem Flächeninhalt von knapp 243 cm^2 .

Für das Projekt wurde die Orion beta software, die in 4.4.5 näher beschrieben wird. Diese Software ermöglicht unter anderem eine Erweiterung der Reichweite der Leap Motion von 60 cm auf 80 cm . Diese Reichweite ist durch die Ausbreitung der LED Lichter räumlich begrenzt. Die Lichtintensität der LEDs ist wiederum durch den maximalen Strom, der über die USB-Verbindung fließt beschränkt.

4.2 Marker

Vera

4.2.1 ArUco Marker

Da sowohl für die Marker, die für den eigentlichen Trackingalgorithmus verwendet werden, als auch für sämtliche Marker, die zur Kalibrierung des Systems zum Einsatz kommen ArUco Marker verwendet werden, werden diese im Folgenden kurz erläutert. ArUco Marker bestehen ähnlich wie QR-Codes aus einer zweidimensionalen Matrix, aus schwarzen und weißen Quadranten, die die kodierten Daten binär darstellen. Die ArUco Bibliothek kann für Augmented Reality Anwendungen genutzt werden und basiert ausschließlich auf der OpenCV Bibliothek.

4.2.2 Trackingmarker

Das System umfasst zwölf Marker über die das Tracking realisiert wird. Alle Marker stimmen in Form und Farbe, sowie Material und Oberflächenbeschaffenheit überein. Sie sind würfelförmig und haben eine Kantenlänge von 46 mm. Die Kanten sind in einem Winkel von 45° angefast. Die Marker bestehen aus Aluminium, welches glasperlgestrahlt ist um eine matte Oberfläche zu erzeugen. Auf die Oberseite des Markers ist mittig ein grünes Quadrat mit einer Kantenlänge von 40 mm aufgebracht. Auf diesem ist, ebenfalls mittig, ein 35 mm großer ARUCO-Marker, welcher aus dem DICT_4X4_50 generiert wurde und einen Rand von einem bit hat. Jeder Marker hat einen einzigartigen ARUCO-Marker, der einer Id von 1–12 entspricht.



Abbildung 1: Trackingmarker mit der ID ??.

4.3 Obsolete Hardware

Im Laufe eines Projekts nach Art von MArC ist es kaum vermeidbar, dass die gesetzten Projektziele reevaluiert werden müssen. Die Gründe hierfür können vielfältig sein. Beispielsweise könnte die Fertigstellung eines bestimmten Teils des Projekts deutlich länger gedauert haben als geplant, oder es könnte sich herausgestellt haben, dass bestimmte Komponenten zueinander nicht kompatibel sind.

Im vorliegenden Projekt trat eine Kombination der beiden oben genannten Gründe auf. Das Betreiben der Ovrvision Pro am US-Bus verschiedener während der

Örtliche Auflösung pro Auge	Zeitliche Auflösung	Bildwinkel	
		Horizontal	Vertikal
2560 × 1920	15 fps	115°	105°
1920 × 1080	30 fps	87°	60°
1280 × 960	45 fps	115°	105°
1280 × 800	60 fps	115°	90°
960 × 950	60 fps	100°	98°
640 × 480	90 fps	115°	105°
320 × 240	120 fps	115°	105°

Tabelle 1: Bildmodi der Ovrvision Pro Stereokamera.[\[11\]](#)

Entwicklung verwendeter Rechner stellte sich als unberechenbar und damit leider unbenutzbar heraus. Die Kamera sorgte während der Ausführung von Unity dafür, dass mit allen anderen Geräten, die ebenfalls per USB angeschlossen waren, unterschiedlichste Probleme auftraten. Als die Situation nach dem Verbinden der Kamera in der teilweisen Zerstörung eines Mainboards gipfelte, wurde die Entscheidung getroffen, die Ovrvision nicht länger als Gerät in der Entwicklung zu verwenden.

Stattdessen war zu diesem Zeitpunkt die Idee, eine gewöhnliche Webcam zu verwenden, um die Realisierung von Augmented Reality dennoch zu ermöglichen, wenn auch ohne den Stereo-3D-Effekt, welchen die Ovrvision nativ bereitgestellt hätte.

Im weiteren Verlauf des Projekts führte eine lange Zeit ungeklärte, starke Abweichung der Positionen der realen und virtuellen Marker zur Neuordnung der Projekt-prioritäten. Dies hatte zur Folge, dass letztendlich auch die Webcam als Plattform für die Umsetzung der AR-Fähigkeiten von MArC aufgegeben wurde.

Nachfolgend werden die Eigenschaften und technischen Daten beider Geräte kurz beschrieben.

4.3.1 Ovrvision Pro

Die Ovrvision Pro ist eine kompakte Stereokamera, welche über USB 3.0 mit dem Rechner verbunden werden kann^[9]. Sie ist kompatibel mit Programmen wie Unity, welches für das Projekt benutzt wurde und in [4.4.1](#) beschrieben wird.

Die Bildmodi der Kamera sind in Tabelle 1 aufgeführt.



Abbildung 2: Explosionszeichnung der Leap Motion.^[5]

4.3.2 Webcam

4.4 Software

4.4.1 Unity

Paul oder Lukas

Unity ist eine sogenannte Spiel-Engine, also eine Entwicklungs- und Laufzeitumgebung, die speziell auf die Entwicklung von 3D-Spielen ausgelegt ist. Die Software wurde am 6. Juni 2005 veröffentlicht^[2] und wird von Unity Technologies^[13] entwickelt und vertrieben. In der Spieleentwicklung ist Unity weit verbreitet, so werden beispielsweise 34 % der kostenfreien Top-1000-Spiele im mobilen Sektor mit Unity entwickelt^[15].

Unity bietet eine sehr breite Plattformunterstützung^[14] und erlaubt ebenso die Entwicklung für Head-Mounted-Displays, wie etwa die Oculus Rift^{[8][16]} oder auch die in diesem Projekt verwendete HTC Vive.^[16]

Die zu Beginn des Projekts verwendete Stereo-Kamera Ovrvision Pro stellt ein Software-Development-Kit (SDK) für Unity (Version 5) zur Verfügung.^[10] Da das endgültige Resultat des Projekts die Verwendung der Ovrvision Pro nicht mehr vorsieht, wie in 4.3 beschrieben, wird auf eine weitere Beschreibung dieses SDKs verzichtet.

4.4.2 Visual Studio

Laura

4.4.3 OpenCV

Vera

Wo führen wir die dlls auf die du erzeugt hast??

4.4.4 Leap Motion SDK

Paul

4.4.5 Orion Beta Software

4.4.6 uEye SDK

Vera

4.4.7 Steam VR

Paul oder Lukas

5 System

Die Benutzung von MArC ist in der dem Programm mitgelieferten ReadMe-Datei beschrieben. Darin wird erklärt, welche Hard- und Software komponenten erforderlich sind, wie das System gestartet und kalibriert wird. Des weiteren enthält die ReadMe eine Übersicht über die enthaltenen Quellcode-Dateien.

5.1 Starten des Systems

Nachdem sichergestellt wurde, dass alle in der ReadMe-Datei beschriebenen Voraussetzungen bestehen, kann das System gestartet werden, indem zunächst die Tracking-Anwendung (auf dem einen Computer) und anschließend die aus Unity heraus erstellte Anwendung (auf dem anderen Computer) gestartet wird. Auf letzterem Computer beginnt darauffolgend die Menüführung, welche in [5.2](#) beschrieben ist.

5.2 Menüführung

Die Menüführung dient dazu, den Benutzer durch alle notwendigen Schritte zu leiten, die vor dem Starten der eigentlichen Simulation erforderlich sind. Im nachfolgenden Abschnitt [5.2.1](#) werden alle verfügbaren Menüs der Anwendung aufgelistet und kurz beschrieben, während im Abschnitt [5.2.2](#) der Ablauf der Menüführung erläutert wird.

5.2.1 Menüs

Die folgenden Menüs sind Bestandteil der Menüführung:

Screenshots der Menüs einfügen?

CalibDone: Wird aufgerufen, wenn die Kalibrierung des Arbeitsbereichs abgeschlossen ist. Es informiert den Benutzer, dass die Kalibrierung erfolgreich war und der Vorgang fortgesetzt werden kann.

CalibrateOrNot: Erscheint nach dem Verlassen des **Welcome**-Menüs und erlaubt dem Benutzer eine Kalibrierung durchzuführen oder eine bereits durchgeführte Kalibrierung zu laden.

ControllerNotFound: Warnt den Benutzer nach dem Starten der Kalibrierung, dass der HTC Vive Controller, welcher für die Kalibrierung benötigt wird, nicht eingeschaltet ist. Während das Menü angezeigt wird, kann der Benutzer den Controller einschalten und anschließend auf **Continue** klicken.

doPlaneCalibInVS: Dient dem Benutzer als Anleitung für die Durchführung der Arbeitsbereich-Kalibrierung. Diese wird in [5.3.2](#) genauer beschrieben.

doPoseCalibInVS: Dient dem Benutzer als Anleitung für die Durchführung der Kamera-Kalibrierung. Diese wird in [5.3.1](#) genauer beschrieben.

SelectCalibrationTarget: Erlaubt die Auswahl der Art der Kalibrierung. Es kann hier entweder nur der Arbeitsbereich oder sowohl der Arbeitsbereich, als auch die Kamera kalibriert werden. Die Kalibrierung ist näher in [5.3](#) beschrieben.

SetScale: Stellt das letzte Menü vor dem Starten der Simulation dar. In diesem kann der Benutzer den Maßstab der Gebäudesimulation einstellen und anschließend die Simulation starten.

SocketNotReady: Warnt den Benutzer nach dem Verlassen des **Welcome**-Menüs, dass die Netzwerkverbindung zum Computer, auf dem die Tracking-Anwendung ausgeführt wird, nicht bereit ist. Nach Bestätigung dieses Hinweises durch einen Klick auf **Continue**, kehrt der Benutzer zum **Welcome**-Menü zurück. Anschließend kann der Vorgang fortgesetzt werden, wenn die Netzwerkverbindung hergestellt wurde. Andernfalls erscheint wieder **SocketNotReady**.

Welcome: Erscheint als erstes Menü. Hier erhält der Nutzer eine kurze Information darüber, wie die Anwendung heißt und wozu sie dient.

5.2.2 Ablauf der Menüführung

Der Ablauf der Menüführung von MArC ist in Abbildung [3](#) dargestellt. Die einzelnen Menüs sind bereits in [5.2.1](#) beschrieben worden.

Nach dem Starten der Anwendung wird zunächst das Menü **Welcome** angezeigt. Dieses enthält nur einen Button *Get started*. Sobald dieser gedrückt wird, prüft die Anwendung, ob eine Netzwerkverbindung zu dem Computer mit der Tracking-Anwendung besteht. Sollte dies nicht der Fall sein, wird das Menü **SocketNotReady** angezeigt. Dieses verlässt der Benutzer über einen Klick auf **Continue**, anschließend wird erneut das Menu **Welcome** angezeigt. Wenn zu diesem Zeitpunkt die Netzwerkverbindung korrekt hergestellt wurde, gelangt der Benutzer zum Menü **CalibrateOrNot**, anderenfalls wird wiederholt **SocketNotReady** angezeigt.

In **CalibrateOrNot** hat der Benutzer die Auswahl zwischen den Schaltflächen *Yes* und *No*. Bei einem Klick auf *Yes* wird anschließend **SelectCalibrationTarget** angezeigt, bei einem Klick auf *No* lädt das System eine zuvor durchgeführte Kalibrierung und das Menü **SetScale** wird geöffnet.

SelectCalibrationTarget stellt den Benutzer vor die Wahl entweder nur den Arbeitsbereich (*Workspace*) oder sowohl den Arbeitsbereich als auch die Kamera zu kalibrieren (*Camera and Workspace*). Außerdem besteht die Möglichkeit über *Cancel* zum Menü **CalibrateOrNot** zurückzukehren.

Wählt der Benutzer *Camera and Workspace* in **SelectCalibrationTarget** aus, so informiert die Anwendung die Tracking-Anwendung auf dem anderen Computer und wartet anschließend darauf, dass von dort die Bestätigung gesendet wird, dass

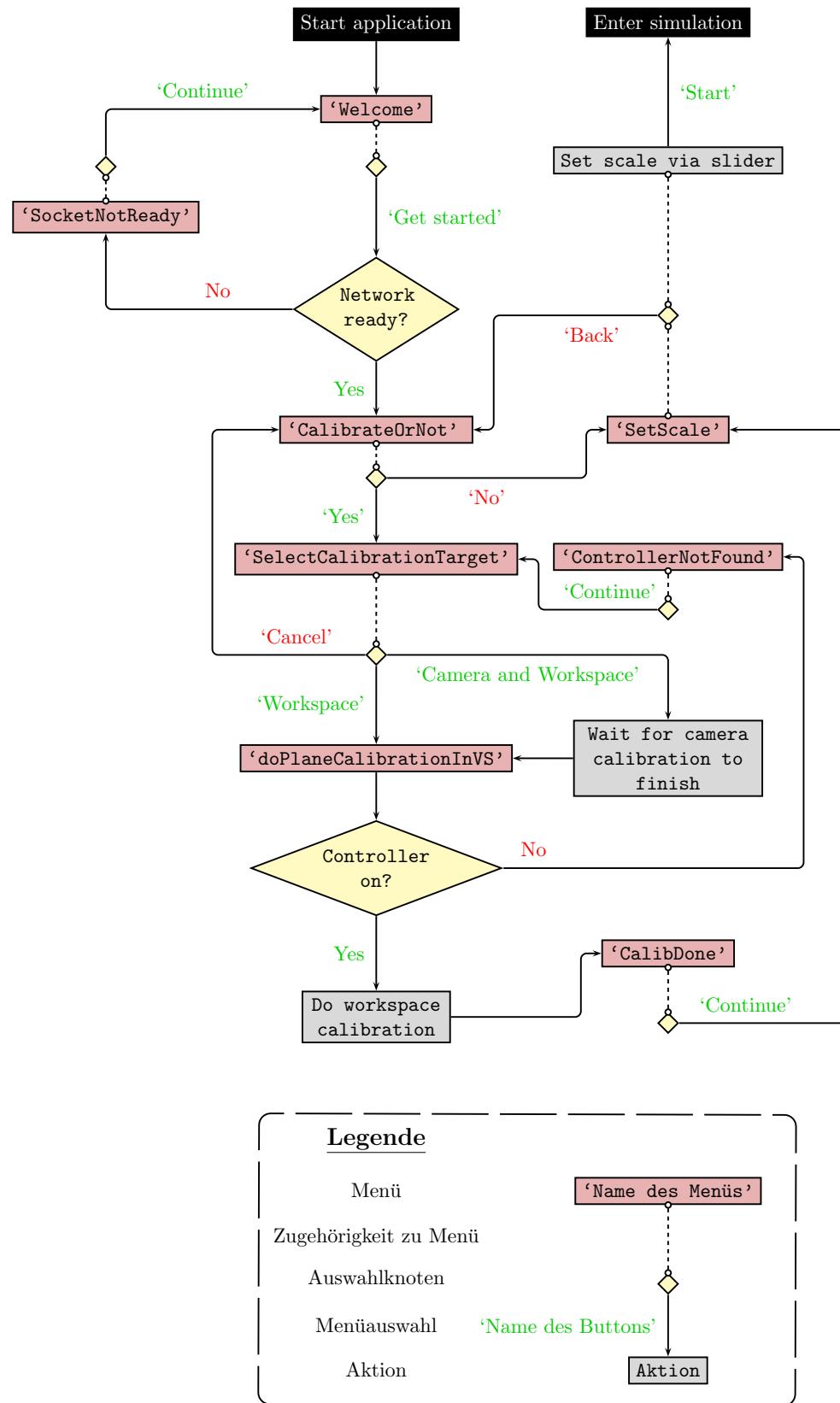


Abbildung 3: Flussdiagramm der Menüführung.

die Kamerakalibrierung abgeschlossen ist. Anschließend wird das Menü `doPlaneCalibrationInVS` angezeigt, welches auch aufgerufen wird, wenn der Benutzer `Workspace` in `SelectCalibrationTarget` wählt.

Im Menü `doPlaneCalibrationInVS` wird zunächst geprüft, ob der für die Kalibrierung notwendige HTC Vive Controller eingeschaltet ist. Sollte dies nicht der Fall sein, wird `ControllerNotFound` aufgerufen. Dieses kann mit einem Klick auf `Continue` verlassen werden, woraufhin wieder `SelectCalibrationTarget` angezeigt wird. Sofern der HTC Vive Controller beim Aufruf von `doPlaneCalibrationInVS` eingeschaltet ist, wird nach Durchführung der Kalibrierung des Arbeitsbereichs das Menü `CalibDone` angezeigt.

`CalibDone` kann über einen Klick auf `Continue` verlassen werden und führt den Nutzer anschließend zu `SetScale`. Aus diesem Menü kann über den Button `Back` entweder zu `CalibrateOrNot` zurückgekehrt oder die Simulation mit dem im Menü über den Slider eingestellten Maßstab gestartet werden.

5.3 Kalibrierung

Um das Ziel von MArC zu erreichen, an den Positionen der Aluminiumwürfel im Arbeitsbereich in der virtuellen Realität von Unity gerenderte Würfel darzustellen, muss das System kalibriert werden. Die Kalibrierung hat zum Ziel, eine Koordinatentransformation zu finden, die Positionen im Kamera-Koordinatensystem in das Unity-Koordinatensystem transformiert.

Zu diesem Zweck muss eine zweistufige Kalibrierung durchgeführt werden. Zunächst sorgt die Kamerakalibrierung dafür, dass Bildkoordinaten auf dem Sensor der Kamera in das 3D-Kamera-Koordinatensystem transformiert werden. Dafür wird sich einiger OpenCV-Funktionen in Verbindung mit Aruco-Markern bedient. Dieser Vorgang wird nachfolgend in 5.3.1 genauer beschrieben.

Der nächste Schritt, die Kalibrierung des Arbeitsbereichs, bestimmt über Punkt-Korrespondenzen – also in zwei verschiedenen Koordinatensystemen bekannte Punkte – eine affine 3D-Transformation, welche die Abbildung vom Kamera-Koordinatensystem auf das Unity-Koordinatensystem ermöglicht. Dieser Kalibrierungsschritt wird nachfolgend in 5.3.2 näher beschrieben.

Trotz einer sorgfältigen Umsetzung der in den nächsten beiden Kapiteln beschriebenen Kalibrierungsschritte, ist es nicht gelungen, den Aluminiumwürfel und den gerenderten Würfel vollständig zur Deckung zu bringen. Der entstandene Fehler sowie mögliche systembedingte Fehlerquellen werden in 5.3.3 und in 5.3.4 beschrieben.

5.3.1 Kamerakalibrierung

Es gibt viele verschiedene wissenschaftliche Ausführungen über die Durchführung einer Kamerakalibrierung, wie z.B. [12], [18] und [1]. Dabei unterscheidet man häufig zwischen automatischen und manuellen Kalibrierungen.

In Abbildung 4 sind die Zusammenhänge zwischen dem Projektionszentrumkoordinatensystem der Kamera, sowie deren Bildebene und dem Weltkoordinatensystem

zu sehen. Im Gegensatz zu der Abbildung und den erwähnten Kalibrierungsansätzen reicht die Umrechnung von Bildkoordinaten in Weltkoordinaten nicht aus. Es muss eine Umrechnung der Bildkoordinaten in den Unity Raum erfolgen, um zu gewährleisten, dass die gerenderten Würfel nachher deckungsgleich mit den Aluminiumwürfeln sind. Koordinaten des Unity Raumes werden im Folgenden Unitykoordinaten genannt.

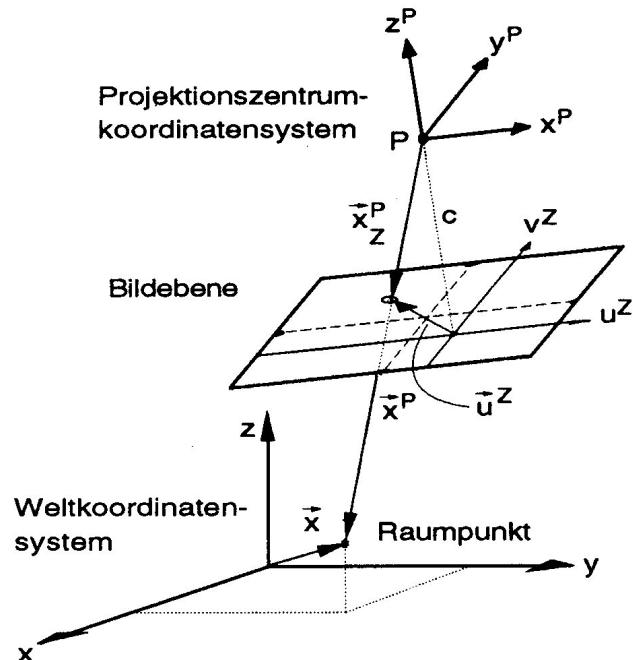


Abbildung 4: Lage des Kamerakoordinatensystems in Bezug auf Projektionsebene und Weltkoordinatensystem. [6]

Die Kamerakalibrierung, die diesem Projekt zu Grunde liegt, ist als manuell einzustufen und wird, wenn gewünscht bzw. benötigt zu Beginn der eigentlichen Anwendung durchgeführt. Um die verschiedenen Koordinatensysteme auseinander zu halten, wird zu Beginn eine Notation festgelegt, die in Tabelle 2 eingesehen werden kann. Zusätzlich können in der Tabelle sowohl die einzelnen Koordinatensysteme, als auch die einzelnen Berechnungsschritte der Kamerakalibrierung nachvollzogen werden, die im Folgenden erläutert werden.

$$M_{intrinsisch} = \begin{pmatrix} f & 0 & 0 & p_u \\ 0 & f & 0 & p_v \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (1)$$

$$M_{intrinsisch} \in \mathbb{R}^{3x4}$$

	Koordinaten	Komponenten	Transformation
\vec{u}	Sensorkoordinaten	u, v	
\vec{x}^V	Verkippungskoordinaten	$u^V, v^V, -c^V$	Bildhauptpunktverschiebung
\vec{x}_D^P	Verzeichnungskoordinaten	$u^D, v^D, -c$	Bildebenenverkippung
\vec{x}_Z^P	Projektionszentrumkoordinaten	$u^Z, v^Z, -c$	Linsenverzeichnung
\vec{x}^P	Projektionszentrumkoordinaten	x^P, y^P, z^P	Projektion
\vec{x}	Weltkoordinaten	x, y, z	Koordinatentransformation
\vec{x}^U	Unitykoordinaten	x^U, y^U, z^U	Koordinatentransformation

Tabelle 2: Parameter und Berechnungsschritte der Kamera Kalibrierung.[6]

$$R = R_\gamma R_\beta R_\alpha = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \quad (2)$$

$$R \in \mathbb{R}^{3x3}$$

$$t = (t_x \ t_y \ t_z) \quad (3)$$

$$t \in \mathbb{R}^{3x1}$$

$$M_{extrinsisch} = (R \ | \ t) \quad (4)$$

$$M_{extrinsisch} \in \mathbb{R}^{3x4}$$

DIST auf Null gesetzt.

5.3.2 Kalibrierung des Arbeitsbereichs

5.3.3 Kalibrierungsfehler

5.3.4 Mögliche Fehlerquellen

6 Auswertung

6.1 Testing durch unerfahrene Benutzer

6.2 Testing durch Architekten

7 Zusammenfassung

- Wissenschaftlicher Stand

Literatur

- [1] O. Faugeras. *Three-dimensional Computer Vision: A Geometric Viewpoint*. Artificial intelligence. MIT Press, 1993.
 - [2] John Haas. *A History of the Unity Game Engine*. PhD thesis, Worcester Polytechnic Institute, 2014.
 - [3] HTC. HTC Vive. <https://www.vive.com/>. Aufgerufen: 30. November 2016.
 - [4] Leap Motion. Leap Motion. <https://www.leapmotion.com/>. Aufgerufen: 30. November 2016.
 - [5] Leap Motion. Leap Motion Blog. <http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/>. Aufgerufen: 2. Januar 2017.
 - [6] Andreas Meisel. *3D-Bildverarbeitung für feste und bewegte Kameras*. PhD thesis, Braunschweig [u.a.], 1994. Zugl.: Aachen, Techn. Hochsch., Diss., 1993 u.d.T.: Meisel, Andreas: 3D-Bildverarbeitung für feste und bewegte Kameras auf photogrammetrischer Basis.
 - [7] Mobile World Congress. Mobile World Congress. <https://www.mobileworldcongress.com/>. Aufgerufen: 30. November 2016.
 - [8] Oculus. Oculus – Unity Intro. <https://developer3.oculus.com/documentation/game-engines/latest/concepts/unity-intro/>. Aufgerufen: 14. März 2017.
 - [9] Ovrvision Pro. Ovrvision Pro. <http://ovrvision.com/>. Aufgerufen: 30. November 2016.
 - [10] Ovrvision Pro. Ovrvision Pro – Informationen für Entwickler. <http://ovrvision.com/setup-en/>. Aufgerufen: 14. März 2017.
 - [11] Ovrvision Pro. Ovrvision Pro – Produktdetails. <http://ovrvision.com/product-en/>. Aufgerufen: 9. März 2017.
 - [12] T. Rahman and N. Krouglicof. An efficient camera calibration technique offering robustness and accuracy over a wide range of lens distortion. *IEEE Transactions on Image Processing*, 21(2):626–637, Feb 2012.
 - [13] Unity Technologies. Unity. <https://unity3d.com/de>. Aufgerufen: 8. März 2017.
 - [14] Unity Technologies. Unity – Multiplatform. <https://unity3d.com/unity/multiplatform>. Aufgerufen: 14. März 2017.
 - [15] Unity Technologies. Unity – Public Relations. <https://unity3d.com/public-relations>. Aufgerufen: 14. März 2017.
-

- [16] Unity Technologies. Unity – VR Overview. <https://unity3d.com/de/learn/tutorials/topics/virtual-reality/vr-overview>. Aufgerufen: 14. März 2017.
 - [17] Valve. Valve Software. <http://www.valvesoftware.com/>. Aufgerufen: 30. November 2016.
 - [18] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov 2000.
-