Informatics Inside

Zukunft mIT gestalten

Informatics Inside
Herbst 2022

Tagungsband

Herausgeber*innen:
Uwe Kloos, Hochschule Reutlingen
Natividad Martinez, Hochschule Reutlingen
Gabriela Tullius, Hochschule Reutlingen

Impressum

Anschrift:

Hochschule Reutlingen
Reutlingen University
Fakultät Informatik
Human-Centered Computing
Alteburgstraße 150
D-72762 Reutlingen

Telefon: +49 7121 / 271-4002
Telefax: +49 7121 / 271-4032

E-Mail: informatics.inside@reutlingen-university.de
Website: informatics.inside@reutlingen-university.de

Organisationskomitee:

Prof. Dr. rer. nat. Uwe Kloos, Hochschule Reutlingen
Prof. Dr.-Ing habil. Natividad Martinez, Hochschule Reutlingen
Prof. Dr. rer. nat. Gabriela Tullius, Hochschule Reutlingen

B. A. Kristina Wagner
B. Sc. Alex Pollok
B. Sc. Annea Skupnjak
B. Sc. Tim Mangliers
B. Eng. Arthur Christoph Krauß
B. Sc. Christian Michels
B. Eng. Niko Gaiser
B. Sc. Philipp Hackh
B. A. Florian Trah
B. A. Viola Schneider
B. Sc. Kai Müller
B. Sc. Reyhan Sancak
B. Sc. Patrick Schmutz
B. Sc. Lukas Bertsch
B. Sc. Sebastian Alois Eckl
B. Sc. Tim Jüstel

Verlag: Hochschule Reutlingen
ISBN 978-3-00-073892-0

Vorwort

Herzlich willkommen zur Informatics Inside Herbst 2022!

Die Konferenz wird von Studierenden des Masterprogramms „Human-Centered Computing“ im Rahmen ihrer Forschungsarbeit organisiert und durchgeführt. Das Masterprogramm sieht vor, dass die Studierenden eine wissenschaftliche Vertiefung in einem Forschungsgebiet ihrer Wahl erarbeiten. Dies kann in einer Firma, in einem Labor der Hochschule oder an ausländischen Partnerhochschulen stattfinden. Als Teil der Ausbildung müssen die Studierenden ihre wissenschaftliche Arbeit in Form eines Forschungsartikels veröffentlichen, präsentieren und mit dem Fachpublikum diskutieren. So entsteht ein Gewinn für alle Beteiligten: Einerseits können die Studierenden die Fähigkeiten trainieren, ihre Arbeit verständlich darzulegen und über den Inhalt zu diskutieren. Anderseits wird dem Publikum ein breites Spektrum der aktuellen Themen in der Informatik angeboten.

Das Motto in diesem Jahr lautet: „Zukunft mIT gestalten“. Die Beiträge sind ein Spiegelbild der menschenzentrierten Rolle der Informatik in der heutigen Welt. Sie zeigen u. a. Forschungen in Künstlicher Intelligenz, Mensch-Maschine-Interkation und Mixed-Reality mit Anwendungen in der Medizin, der Wirtschaft und der Gesellschaft. Ein besonderer Höhepunkt der Konferenz ist der abschließende Gastvortrag von Frau Prof. Dr. Claudia Müller-Birn zum Thema “Human-Centered Data Science”.

Ein letztes Wort möchte ich dem Organisationskomitee der Konferenz widmen: Der gesamte Prozess wurde von den Studierenden über mehrere Monate geplant. Vom Call-for-Papers, Gutachten, Proceedings-Erstellung bis zur Finanzierung und Technik wurde alles höchst professionell durchgeführt. An dieser Stelle möchte ich den Masterstudierenden meinen Glückwunsch aussprechen.

Ich wünsche allen für diesen Tag eine spannende Konferenz, neue Erkenntnisse und neue Kontakte sowie viel Spaß.

Reutlingen im November 2022

Prof. Dr.-Ing habil. Natividad Martinez

Inhaltsverzeichnis

Menschmodelle für ergonomische Analysen in der Radiologie

Marcel van Look
Hochschule Reutlingen
Reutlingen, Deutschland
Marcel_Andre.Van_look@Student.Reutlingen-University.de

Abstract

Das Ziel dieser Arbeit ist die Evaluierung verschiedener digitaler Menschmodelle in einem durch zwei Anwendungsfälle aufgestellten medizinischen Rahmen. Der erste Anwendungsfall befasst sich mit der Überprüfung der Lagerung von Patienten beim Röntgen in der Radiologie und der zweiter Anwendungsfall befasst sich mit der Überprüfung der ergonomisch korrekten Haltung eines Arztes während einer Operation. In Folge der Arbeit wurden verschiedene wissenschaftliche Veröffentlichungen und Bücher nach aktuell verwendeten digitalen Menschmodellen durchsucht und anschließend, falls sie für die Anwendungsfälle relevant waren, in eine Evaluierungstabelle eingetragen. Zusätzliche Informationen wurden falls notwendig den Herstellerwebseiten entnommen. Mit diesem Vorgehen konnten fünf Digitale Menschmodelle gefunden und bewertet werden. Das Vergleichssiegermodell ist hierbei Biomechanics of Bodies, die zweite Wahl The AnyBody Modeling System und die günstigste Alternative Santos. Für die gewählten Anwendungsfälle empfiehlt es sich entweder den Vergleichssieger oder die zweite Wahl zu nehmen, da diese Modelle einen größeren Funktionsumfang bieten und somit eine Erweiterung der Anwendungsfälle möglich wäre.

CCS Concepts

•Computing methodologies→Modeling and simulation→Simulation evaluation

Keywords

Digital human model, biomechanic human model, digital man model, mannequin Digitales Menschmodell, Biomechanisches Modell, Manikin

1 Einleitung

Das Ziel dieser Arbeit ist die Evaluierung verschiedener digitaler Menschmodelle in einem durch zwei Anwendungsfälle aufgestellten medizinischen Rahmen. Der erste Anwendungsfall befasst sich mit der Überprüfung der Lagerung beim Röntgen in der Radiologie. Ziel ist eine Verminderung der Strahlendosis auf den Patienten durch die Vermeidung von Mehrfachaufnahmen auf Grund von falscher Lagerung. Der zweite Anwendungsfall befasst sich mit der Überprüfung der ergonomisch korrekten Haltung eines interventionellen Radiologen wärend eines Eingriffs, um muskuloskelettale Schäden vorzubeugen. Digitale Menschmodelle (DMM) sind in vielen Fachgebieten, wie zum Beispiel dem Militär, der Luft- und Raumfahrttechnik, sowie der Automobile-

Betreuer Hochschule: Prof. Dr. Burgert
Hochschule Reutlingen
Oliver.Burgert@Reutlingen-University.de
Informatics Inside Herbst 2022
23. November 2022, Hochschule Reutlingen

branche im Einsatz. Auch in der Medizin haben die DMM bereits Erfolge erzielt und breiten sich immer weiter aus. Es gibt jedoch so viele DMM auf dem Markt, die alle unterschiedliche Spezifikationen und somit auch Stärken aufweisen, dass es schwierig ist, das Richtige für den Einsatz zu finden. Bei manchen medizinischen Anwendungsfällen wird zum Beispiel weniger Wert auf eine schöne Hüllfläche des Modells gelegt, dafür stehen ein vollständiges Skelett und die umhüllende Muskelstruktur im Vordergrund. Aber selbst in der Medizin gibt es Fachgebiete, wie zum Beispiel die plastische Chirurgie, bei denen es genau andersherum ist. In diesem Artikel werden mehrere DMM aufgezeigt, die sich für die Anwendung in Fachgebieten der Medizin eignen, die größeren Wert auf Skelettstruktur und Muskulatur legen. Wichtig ist hierbei der Fokus auf ein ganzkörper Modell. Das liegt daran, dass es viele DMM gibt, die nur die unteren Extremitäten abbilden, um Ganganalysen durchzuführen.

1.1 Die Anwendungsfälle

Für diesen Artikel wurden zwei Anwendungsfälle aufgestellt, die sich in dem Fachgebiet der Medizinischen Informatik eingliedern lassen.

Der erste Anwendungsfall befasst sich mit der Überprüfung der Lagerung beim Röntgen in der Radiologie.

Dazu wird angenommen, dass ein Patient in der Radiologie mit einer Verletzung am Arm geröntgt werden muss. Er wird mit den erforderlichen Sicherheitsmaßnahmen ausgestattet und muss seinen Arm unter dem Gerät platzieren. In diesem Anwendungsfall soll mithilfe eines DMM überprüft werden, ob der Arm ergonomisch korrekt gelagert wird oder ob es zu einem Diskomfort für den Patienten führt.

Der zweiter Anwendungsfall befasst sich mit der Überprüfung der ergonomisch korrekten Haltung eines Arztes während einer Operation.

Hierbei wird angenommen, dass ein Arzt am Operationstisch steht und während einer Operation über mehrere Stunden unterschiedliche Körperhaltungen einnehmen muss. Hinzu kommt, dass ein Radiologe eine Bleischürze, die ein hohes Eigengewicht aufweist, tragen muss. Dabei soll mithilfe eines DMM eine Überwachung seiner Haltung generiert werden. Diese kontrolliert, ob der Arzt ergonomisch korrekt steht. Gegebenenfalls soll eine schlechte Haltung erkannt werden, um diese dann möglichst schnell wieder ändern zu können.

1.2 Forschungsfragen

Forschungsfrage 1: Welche Digitalen Menschmodelle, die in den letzten vier Jahren wissenschaftlich relevant waren, gibt es momentan auf dem Markt, mit denen ein arbeitswissenschaftlicher biomechanischer Prototyp, der Anwendung in einem Krankenhaus findet, erstellt werden kann?

Forschungsfrage 2: Welche Spezifikationen muss ein Digitales Menschmodell aufweisen, um für die Anwendungsfälle relevant zu sein?

Forschungsfrage 3: Welche gefundenen digitalen Menschmodelle aus der ersten Forschungsfrage erfüllen auch die Kriterien aus Forschungsfrage 2?

2 Methodik

Das Ziel dieser Arbeit ist das Sammeln und Evaluieren verschiedener DMM. Bei der Suche wurde mit einfachen Suchtermen, wie in Tabelle 1 zu sehen, gearbeitet. Die Zahlen der gefundenen Artikel in der Tabelle können Dopplungen enthalten. Die Suche wurde mit den Suchmaschinen Google Scholar, PubMed und IEEE durchgeführt. Danach wurden die wissenschaftlichen Publikationen ausschließlich nach den Namen und falls vorhanden zusätzlichen Informationen der verwendeten DMM durchsucht. Als einzige Ausnahme wurde ein Buch [1] aus

Tabelle 1: Literaturrecherche

Suchterm	Suchmaschine	Datum	Ergebnisse
(Medizin) AND ((Mannequin) OR (Human Body Model))	IEEE	10.08.2022	4657
(Medizin) AND ((Mannequin) OR (Human Body Model))	PubMed	08.08.2022	23
(Mannequin) OR (Human Body Model) With at least one of the words (Medizin)	Google Scholar	14.08.2022	6150

dem Jahr 2016 in die Quellen mit aufgenommen, das bei der ersten Googelsuche zur Thematik der Arbeit gefunden wurde. Durch das Vorgehen konnten über 100 verschiedene DMM gefunden werden. Davon blieben nach dem Herausfiltern von veralteten und für medizinische Anwendungen unbrauchbaren Modellen elf DMM übrig. Darauf hin war es notwendig, die Herstellerwebseiten der DMM zurate zu ziehen, um die Informationen zu vervollständigen. Als letzter Schritt wurden die DMM daraufhin überprüft, ob sie die Anwendungsfälle durchführen können. Es ergab sich eine Liste von fünf DMM, die evaluiert werden konnten.

3 Digitale Menschmodelle

Digitale Menschmodelle können Aufgaben für Menschen virtuell überprüfen und testen. Bekannte Beispiele sind dafür die Crashtests oder das Einrichten eines ergonomisch korrekten Arbeitsplatzes. Im folgenden Kapitel werden die unterschiedlichen Arten von digitalen Menschmodellen und deren Einsatzgebiete aufgezeigt, sowie erklärt, wie sie funktionieren und welche Methoden sie anwenden können. [1,2]

3.1 Arten digitaler Menschmodelle

Digitale Menschmodelle lassen sich, je nach dem welchen Funktionsumfang sie aufweisen, in drei Kategorien untergliedern. In Abbildung 1 sind diese von oben nach unten zusehen: arbeitswissenschaftliche DMM, arbeitswissenschaftlich nutzbare DMM und sonstige DMM. Wie auch in der Abbildung zu erkennen, gehören die übergeordneten DMM immer zu den untergeordneten Gruppierungen und können somit bei Bedarf auch in deren Anwendungsgebieten eingesetzt werden. Wie sinnvoll das ist, muss von Fall zu Fall entschieden werden, da zum Beispiel die arbeitswissenschaftlichen Modelle sich aufgrund ihres Aussehens nicht immer für den Einsatz auf dem Gebiet der dritten Gruppierung eignen würden. Arbeitswissenschaftliche DMM dienen als professionelle Werkzeuge in der Produktgestaltung und bei einzelnen Prozessgestaltungen. Dabei wird das Augenmerk, eher auf Funktionalität als auf Design gelegt, da die Aufgaben der DMM hier auf dem Gebiet der Analyse verschiedener Szenarien sowie ergonomischer Sachverhalte liegen.

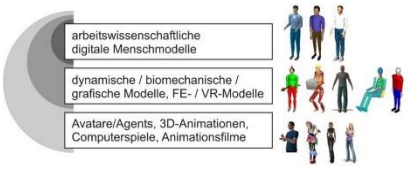

Abbildung 1: Arten digitaler Menschmodelle [1]

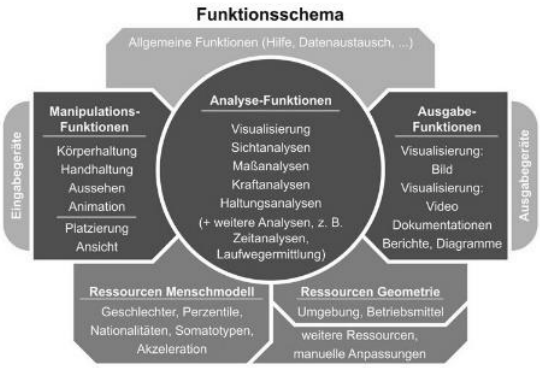

Abbildung 2: Funktionsschema digitaler Menschmodelle [1]

Arbeitswissenschaftlich nutzbare DMM sind, im Vergleich zu dem übergeordneten DMM, spezialisiert auf ein Aufgabengebiet und weisen unterschiedliche Vorzüge auf. So gibt es zum Beispiel DMM, die sich auf ein besseres Aussehen, autonome Prozesse, sowie ein flüssigeres Bewegungsverhalten fokussieren. Eine andere Gruppe von DMM spezialisiert sich beispielsweiße auf anatomische Strukturen wie Knochen und Muskeln. Da diese DMM spezialisiert sind, haben sie auch nicht den kompletten Funktionsumfang, den ein arbeitswissenschaftliches DMM hat.

Sonstige DMM sind Modelle, die ausschließlich graphischen Nutzen haben und keinerlei Analysefunktionen beinhalten. Unter diese Kategorie fallen sogar Modelle, die keine DMM im eigentlichen Sinn sind. Ein Beispiel für diese DMM wäre BioDigital Human. [1,6]

3.2 Funktionalität digitaler Menschmodelle

Die Funktionalität eines DMM lässt sich in drei Blöcke unterteilen, wie in Abbildung 2 zu sehen ist. Bevor mit diesen drei Blöcken gestartet wird, bearbeitet der User zuerst die Rahmenbedingungen. Das heißt, es wird das Menschmodell erstellt und es werden die gewünschten Eigenschaften eingestellt, also Geschlecht, Alter, Nationalität und dergleichen. Danach generiert der User die Umgebung, also beispielsweiße ein Arbeitsplatz, Geräte oder andere Gegenstände mit denen das DMM interagieren soll. Nachdem dadurch der Rahmen für das zu testende Szenario erstellt ist, werden zuerst Manipulationsfunktionen ausgeführt. Dadurch wird das DMM an die auszuführende Aufgabe angepasst. Dabei können Funktionen, wie in Abbildung 2 beschrieben, benutzt werden, um mit dem DMM zu interagieren und das Szenario durchzuführen. Darauf folgt der Analyseteil, in dem das Modell unterschiedliche ergonomische Methoden durchführt. Wenn von dem Szenario ausgegangen wird, dass ein ergonomisch korrekter Arbeitsplatz generiert werden soll, so wird zum Beispiel die Sichtanalyse und eine Maßanalyse durchgeführt, um festzustellen, ob der Monitor im Sichtfeld des DMM richtig platziert wurde. Zum Schluss werden im dritten Block mithilfe der Ausgabefunktionen die Ergebnisse präsentiert. Dabei kann es sich um verschiedene Bilder oder Videos handeln, die das durchgeführte Szenario beispielsweise aus Sicht des DMM beleuchten. Zusätzlich wird eine Dokumentation erstellt und es werden je nach Test visualisierte Berichte generiert. [1,6]

4 Ergebnisse

Dieses Kapitel befasst sich mit den aufgestellten Anforderungen, die für die Erfüllung

der Anwendungsfälle benötigt werden, den verglichenen DMM sowie dem Aufbau der Evaluierungstabelle.

4.1 Anforderungen an die digitalen Menschmodelle

Im folgenden Unterkapitel werden die Anforderungen erläutert, die im Rahmen der Anwendungsfälle an die DMM gestellt werden. Dabei werden die Funktionen in vier Kategorien untergliedert, unerlässliche Funktionen, nützliche Funktionen, nebensächliche Funktionen und unnötige Funktionen. Bis auf die letzte Kategorie werden alle aufgegriffen und die Funktionen der DMM eingegliedert und kurz erklärt.

4.1.1 Unerlässliche Funktionen

Es wurden fünf unerlässliche Funktionen identifiziert. Unerlässliche Funktionen sind eine Grundvoraussetzung für den weiteren Verlauf der Betrachtung und sind aufgrund dessen aus der Bewertung ausgeschlossen.

Es muss ein Volumenskelett vorhanden sein. Das Volumenskelett entspricht dem menschlichen Skelett und muss, um eine sinnvolle Bewegung darstellen zu können vorhanden sein.

Es müssen anatomische Strukturen vorhanden sein. Mithilfe der anatomischen Strukturen, genauer der Muskulatur, kann das DMM später in der Analyse die Werte der Haltung kalkulieren. Dadurch sind sie für das durch die Anwendungsfälle generierte Szenario unverzichtbar.

Das DMM muss sowohl Vorwärts- als auch Inverse Kinematik unterstützen. Durch die Vorwärts- und Inverse Kinematik können Winkel und Pose der Arme und Gelenke berechnet werden. Dadurch sind diese zwei Funktionen die Mindestanforderungen, die ein DMM erfüllen muss, um realistisch manipuliert werden zu können.

Das DMM muss eine Haltungsanalyse durchführen können. Diese stellt die Informationen bereit, die in den Anwendungsfällen gesucht werden.

4.1.2 Nützliche Funktionen

Durch den Einsatz der folgenden Funktionalitäten kann die Benutzung des DMM erleichtert oder gegebenenfalls sogar verbessert werden. Deshalb werden diese Funktionen in der Evaluation mit zwei Punkten bewertet.

Mithilfe von Motion Capture kann das DMM die Eingabe einer Bewegung des Patienten oder Arztes erleichtern oder gegebenen falls überhaupt erst möglich machen.

Bei den Analysefunktionen hilft die Teilkörperanalyse bei der Betrachtung einzelner Körperregionen. Demnach ist sie bei dem ersten Anwendungsfall nützlich. Dahingegen helfen die Bewegungsanalyse und die Muskelbelastungsanalyse bei dem zweiten Anwendungsfall.

Die beiden Funktionen Modellskalierung und frei bewegliches Modell sorgen für größere Freiheiten bei der Anpassung der DMM und generieren eine größere Vielseitigkeit.

4.1.3 Nebensächliche Funktionen

Die nebensächlichen Funktionen sind entweder für die Anwendungsfälle nicht notwendig, können aber eine interessante Erweiterung darstellen oder sind nur für den Entwickler eine Hilfe. Sie werden mit einem Punkt bewertet.

Die Anpassung des Alters und des Geschlechts ist zu geringem Maß interessant, da gegebenenfalls eine andere Belastung des Oberkörpers durch unterschiedlichen Körperbau, vorliegen kann.

Die Analysefunktionen der Kinematischen-, Kinetischen-, Kraft-, Erreichbarkeits-, Sportmedizinische- sowie Sitzkomfortanalyse geben weiter Informationen, die zwar nicht ausdrücklich im Anwendungsfall gefragt sind, aber dem Arzt situationsbedingt nützlich sein können.

Durch die Funktionen Muskelkalibrierung und Muskelmodellierung kann das DMM im Nachlauf noch angepasst werden. Dies sollte

zwar nicht notwendig sein, kann aber trotzdem von Nutzen sein.

Die Funktion der Muskelkennzeichnung ist gut für angehende Ärzte zum Lernen oder generell für das Wählen von Landmarken geeignet.

Die Funktionen, die für den Nutzer uninteressant sind, dem Entwickler wiederum helfen können, sind verschiedene Motion Capture Interfaces und Bibliotheken des Herstellers der DMM.

4.2 Die digitalen Menschmodelle

Im folgenden Abschnitt werden die DMM, die in diesem Artikel evaluiert werden, kurz vorgestellt. Dabei wird auf den allgemeinen Rahmen der DMM eingegangen. Genauere Informationen zu dem Funktionsumfang der Modelle sind dann in den Tabellen 2 und 3 im Anhang aufgezeigt. Modelle, die nicht in die Endauswahl gekommen sind, sind Ramsis (kein Motion Capture, nicht kostenlos, Funktionsumfang zu gering), Human Builder (kein Motion Capture und kein Volumenskelett), MakeHuman (kein Motion Capture und kein Volumenskelett), Jack and Jill (kein Motion Capture und kein Volumenskelett), LifeMod (schwer zugänglich, suspekte offizielle Informationen) und Emma CC (beschränkt auf obere Extremitäten). [1,2,3,6] Diese Modelle sind zwar durchaus aktuell und fähig in einem medizinischen Szenario eingesetzt zu werden, allerdings entsprechen sie nicht den Anforderungen der gewählten Anwendungsfälle.

4.2.1 The AnyBody Modeling System

Bei dem DMM The AnyBody[1][1,5,8] Modeling System, in den folgenden Kapiteln als AnyBody abgekürzt, handelt es sich um ein simuliertes Skelett mit Muskeln, Bändern und Sehnen. Es wurde in Dänemark an der Universität in Aalborg erstellt und ab 2002 von dem dafür gegründeten Unternehmen AnyBody Technologies weiterentwickelt. Abgesehen von der fehlenden Haut, aufgrund des Fokus auf Muskel-Skelett Modellierung (MSM), ist das Modell trotzdem auf den meisten Gebieten einsatzfähig. Für diese Arbeit ist das Fehlen der Haut allerdings kein Problem, da das DMM dadurch bessere Einblicke in die biomechanischen Abläufe gewährt. Dementsprechend weist das DMM auch die richtigen Funktionalitäten auf und kann Muskelkräfte, Muskelermüdung und den Metabolismus analysieren und visualisieren. Ebenso ist es echtzeitfähig und kann über Motion Capture manipuliert werden. Das ist hinsichtlich des zweiten Anwendungsfalls gut. Hilfreich sind auch die Webinare die häufig stattfinden und danach auf dem YouTube Channel erscheinen.

4.2.2 Santos

Santos [2][1] ist ein Menschmodell das 1998 an der Universität von Iowa in Kooperation mit der US Army entstanden ist. Ziel war die Simulation eines Soldaten, der alle möglichen militärorientierten Aufgaben ausführen kann. Bei der Analyse wurde der Schwerpunkt vor allem auf Muskelbelastung und die Vitalwerte gelegt. Mit dem Modell ist es möglich, Informationen wie Muskelermüdung, Diskomfort, Stärke, Kraftaufwand, Blutdruck, Herzschlag und Metabolismusinformation zu generieren. Seit 2008 wurde die Entwicklung von Santos an das Spin-off Unternehmen SantosHuman Inc. der Universität von Iowa übergeben. Es wurden in den nächsten Jahren viele neue Einsatzbereiche wie zum Beispiel die Medizin, Luft und Raumfahrt sowie die Automobilbranche erschlossen. Durch den großen Funktionsumfang, der auch Motion Capture beinhaltet,

[1]https://www.anybodytech.com/software/anybodymodelingsystem/ - abgerufen: 05.10.2022

[2]https://www.santoshumaninc.com/ - abgerufen: 05.10.2022

lassen sich beide Anwendungsfälle ohne Probleme umsetzen.

4.2.3 Total Human Model for Safety

Das Total Human Model for Safety (THUMS)[3][4] [9] wurde seit den 90ern von der Toyota Motor Corporation und Toyota Central R&D Labs entwickelt und 2000 auf den Markt gebracht. Vertrieben wurde es in Deutschland eine Zeit lang von DYNAmore. Heute gibt es das Modell allerdings als free Access Software von Toyota. Das Interessante an diesem Modell ist, dass es seit 2010 in der Modellversion 4.0 sehr fein aufgebaut ist. Dazu wurde das komplette Modell auf CT-Scan Daten entwickelt. Dadurch kann es sowohl eine sehr feine Skelettstruktur als auch sehr detaillierte Muskelstränge und sogar modellierte Organe vorweisen. Dies ist optimal für den Einsatz in der Radiologie. Allerdings kann schon von den Herstellern abgeleitet werden, dass hierbei das Einsatzgebiet in der Automobilbranche liegt. Mithilfe des Funktionsumfangs lässt sich der erste Anwendungsfall ohne Probleme umsetzen, allerdings wird es bei dem Zweiten etwas schwieriger, da das DMM nicht Motion Capture fähig ist und man dahingehend schwierigere Ansätze verfolgen muss. Diese Tatsache wird sich in der folgenden Evaluation nochmals widerspiegeln.

4.2.4 Global Human Body Models

Das Global Human Body Models Consortium (GHBMC)[5] [4,9] wurde 2006 mit der Intention gegründet, eine Plattform zu erstellen, auf der weltweite Forschung zum Thema Menschmodellierung gebündelt werden kann. Heute nehmen viele namentlich bekannte Firmen daran teil. GHBMC kann eine ganze Modellfamilie von zehn DMM aufweisen und hat für jeden Gebrauch ein passendes Modell. Für dieses DMM gibt es eine kostenlose Lizenz für Universitäten und Forschungseinrichtungen. Da auch hier wieder hauptsächlich die Automobilbranche dahintersteht, wird sich vorwiegend auf die Simulation von Crashtest fokussiert. Dennoch sind die anatomischen Strukturen interessant und passend für den ersten Anwendungsfall. Hinzu kommt, dass die GHBMC Familie momentan in der dritten Entwicklungsphase ist und hier die Verbesserung eben jener interessanten anatomischen Strukturen priorisiert. Allerdings besteht auch bei diesem DMM wieder das analoge Problem mit dem zweiten Anwendungsfall wie bei THUMS.

4.2.5 Biomechanics of Bodies

Das DMM Biomechanics of Bodies[6], [5,7] im folgenden BoB genannt, ist genau genommen eine Familie aus fünf unterschiedlichen DMM, die auf explizite Anwendungsgebiete angepasst wurden. Diese Anwendungsgebiete sind Research, Teaching, Ergo, EMG und Xsens. Jedes der DMM hat einen unterschiedlichen Funktionsumfang. Für diesen Artikel werden die DMM Teaching Xsens und EMG nicht beachtet. Das begründet sich darauf, dass man bei der Lehrvariante keine eigenen Daten einspielen kann. Die Xsens Variante ist zwar sehr gut für den zweiten Anwendungsfall geeignet, allerdings kann das gleiche Ergebnis auch günstiger mit anderen Varianten erreicht werden. Zudem hat die EMG-Variante zu viele zusätzliche Funktionen, die nicht notwendig sind und somit den hohen Preis nicht rechtfertigen. Optisch betrachtet ist das Modell

[3] https://www.dynamore.se/de/produkte/modelle/menschmodell - abgerufen: 05.10.2022

[4] https://www.toyota.co.jp/thums/ - abgerufen: 05.10.2022

[5] http://www.ghbmc.com/ - abgerufen: 05.10.2022

[6] https://www.bob-biomechanics.com/ - abgerufen: 05.10.2022

einfach aufgebaut, da man auf eine Hüllfläche verzichtet hat. Mit den gegebenen Funktionen ist es aber optimal dafür geeignet sowohl den ersten als auch den zweiten Anwendungsfall umzusetzen.

4.3 Tabellenaufbau

Die Tabellen 2 und 3 dienen als Evaluierungstabellen. Sie sind nach dem gleichen Schema angelegt. Da beide Tabellen eine komplette Seite belegen und einen wichtigen Bestandteil der Arbeit ausmachen, sind sie im Anhang des Artikels zu finden. Dabei wird in Zeile Eins, über der eigentlichen Tabelle, ein Titel für die besten DMM festgelegt. In Zeile Zwei und Drei stehen der Name und die Firma des vorgestellten DMM. In Zeile Vier finden sich die graphischen Komponenten, die das DMM anzeigen kann. Zeile Fünf beschäftigt sich mit den Eigenschaften des Modells. Das heißt, was an dem digitalen Menschen im Voraus angepasst werden kann (Geschlecht u.Ä.). In Zeile Sechs befinden sich die Manipulationsarten, mit denen der User auf das Modell einwirken kann. In Zeile Sieben sind die Analysefunktionen zu finden, die das DMM durchführen kann. Zeile Acht beschäftigt sich mit Funktionen, die sich nicht in die vorherigen Kategorien eingliedern lassen, aber nennenswert sind. Zeile Neun und Zehn befassen sich mit der Systemintegration und den Modellkosten. Abschließend steht in Zeile Elf die vergebene Gesamtpunktzahl. Funktionen, die in den Anforderungen in Kapitel 4.1 aufgezählt wurden, werden in der Evaluation bewertet. Die vergebenen Punkte stehen dazu zur Nachvollziehbarkeit nochmals in Klammern hinter den Funktionen.

5 Diskussion

Wenn die Punkteverteilung der Evaluationstabelle betrachtet wird, ist zu sehen, dass die DMM THUMS (7P.) und GHBMC (9P.) auf den beiden letzten Plätzen sind. Einzeln betrachtet könnten sie die Anwendungsfälle durchführen, aber die anderen Modelle sind auf allen Gebieten im direkten Vergleich besser. Das Modell Santos(12P.) weist alle nötigen Funktionen auf, um die Anwendungsfälle ohne große Probleme umzusetzen. Mit der Kombination von Motion Capture als Eingabemittel und der Haltungsanalyse zur Überprüfung der Ergonomie erfüllt es alle Mindestanforderungen. Das DMM erhält dadurch und durch die Modellkosten den Titel der günstigsten Alternative. Allerdings lässt der Funktionsumfang nicht viele Erweiterungen der Anwendungsfälle zu, weshalb es in der Gesamtwertung auch nur den dritten Platz erreicht. Das DMM BoB(20P.) kann den ersten Platz vor dem AnyBody(14P.) Modell belegen, in dem es die Möglichkeit der Teilkörperanalyse vorweisen kann, ein paar nebensächliche Funktionen mehr hat und etwas günstiger ist. Die Teilkörperanalyse ist sowohl im ersten Anwendungsfall zur Betrachtung einzelner Körperpartien als auch, im zweiten Anwendungsfall bei einer Betrachtung eines bestimmten Körperteils nützlich. Beide Modelle haben die Möglichkeit aus zwei unterschiedlichen Motion Capture Ansätzen zu wählen und bieten somit eine größere Freiheit bei der Entscheidungsfindung. Ebenfalls können die Anwendungsfälle mit dem verbleibenden Funktionsumfang noch erweitert werden. Folglich ist ein Projekt, das mit diesen beiden DMM durchgeführt wird, zukünftig leicht zu erweitern.

6 Fazit

Für diesen Artikel wurden aus über 100 recherchierten Digitalen Menschmodellen fünf herausgefiltert, die sich für den Einsatz bei Anwendungsfällen medizinischer Art, mit einem Hauptaugenmerk auf den ganzen Körper, eignen. In Forschungsfrage 1 werden DMM gesucht, die in einem medizinischen Umfeld einsetzbar sind. Diese Frage lässt sich mit den sechs ausgeschiedenen DMM Ramsis, Human Builder, MakeHuman, Jack and Jill, LifeMod und Emma CC sowie den fünf Evaluierten Modellen AnyBody, Santos, THUMS, GHBMC und BoB beantworten. Für die zweite Forschungsfrage ist es

wichtig, dass ein DMM eine Haltungsanalyse bzw. eine ergonomische Analyse durchführen kann. Als Eingabemöglichkeiten kann das DMM entweder manuell an die Haltung des Patienten angepasst werden oder mithilfe von Motion Capture gesteuert werden. Wenn der zweite Ansatz eingesetzt wird, kann mit dem gleichen Setup an Funktionen auch der zweite Anwendungsfall durchgeführt werden. Demnach eignen sich die DMM AnyBody, Santos, THUMS, GHBMC und BoB für den Einsatz und beantworten somit die dritte Forschungsfrage. Zuletzt können nach der Evaluation drei Modelle für den Gebrauch bei diesen oder ähnlichen Anwendungsfällen empfohlen werden. Falls nicht damit zu rechnen ist, dass die Anwendungsfälle erweitert werden, dass das durchzuführende Projekt an einer Forschungseinrichtung oder einer Universität durchgeführt wird und möglichst wenig Geld investiert werden soll, so lässt sich die günstigste Alternative Santos empfehlen. Falls jedoch Erweiterungen für die Anwendungsfälle geplant sind, oder diesen zumindest der Funktionsumfang der DMM nicht im Weg stehen soll, dann lassen sich die Modelle BoB, der Vergleichssieger oder AnyBody die zweite Wahl empfehlen. Desweitern bieten die Hersteller dieser beiden Modelle mehr Hilfestellungen und Tutorials an als bei Santos.

Literaturverzeichnis

[1] A. Bullinger-Hoffmann, J.Mühlstedt. Homo Sapiens Digitalis – Virtuelle Ergonomie und digitale Menschmodelle. München, Springer Vieweg 2016, ISBN 978-3-662-50458-1, ISBN 978-3-662-50459-8 (eBook)

[2] A. Kumar et al. Musculoskeletal Modeling and Analysis of Trikonasana. DOI: 10.4103/ijoy.IJOY_1_18

[3] A. Nerot, W. Skalli, X. Wang. An assessment of the realism of digital human manikins used for simulation in ergonomics. DOI:10.1080/00140139.2015.1038306

[4] C. Untaroiu, W. Pak, Y. Meng, J.Schap. A Finite Element Model of a Midsize Male for Simulating Pedestrian Accidents DOI:10.1115/1.4037854

[5] J. Langholz, G. Westman und M. Karlsteen. Musculoskeletal Modelling in Sports - Evaluation of Different Software Tools with Focus on Swimming DOI: 10.1016/j.proeng.2016.06.27

[6] J. Mühlstedt und B. Spanner-Ulmer. Homo Sapiens Digitalis: über den Praxiseinsatz digitaler Menschmodelle. Zentrum Mensch-Maschine-Systeme der Technischen Universität Berlin, Chemnitz, ISBN 978-3-18-302922-8

[7] J. Shippen und B. May. BoB – biomechanics in MATLAB DOI:10.3846/biomdlore.2016.02

[8] L. Engelhardt, M. Melzner, L. Havelkova, P. Fiala. A new musculoskeletal AnyBody™ detailed hand model. DOI:10.1080/10255842.2020.1851367

[9] N.Trube, W. Riedel und M.Boljen. How muscle stiffness affects human body model behavior. DOI:10.1186/s12938-021-00876-6

7 Anhang

Tabelle 2: Digitale Menschmodelle 1-3

	Zweite Wahl	Günstigste Alternative	
Name	AnyBody	Santos	THUMS
Firma	AnyBody Technology	Santos Human Inc	Toyota Motor Corporation
Graphische Komponenten	-Skelett -Volumenskelett -Anatomische Strukturen	-Skelett -Volumenskelett -Anatomische Strukturen -Hüllfläche	-Skelett -Volumenskelett -Anatomische Strukturen -Hüllfläche
Eigenschaften		- Geschlecht (1)	-Geschlecht (1) -50 Perzentiles -Fußgänger und Insasse
Manipulationen	-Vorwärtskinematik -Inverse Kinematik -Motion Capture (2) -Muskelmodellierung (1)	-Vorwärtskinematik -Inverse Kinematik -Motion Capture (2)	-Vorwärtskinematik -Inverse Kinematik -Muskelmodellierung (1)
Analysen	-Sichtbarkeit -Lastenhandhabung -Haltung -Bewegung (2) -Kinematik (1)	-Sichtbarkeit -Lastenhandhabung -Kinematik (1) -Haltung -Bewegung (2) -Kraft (1)	-Lastenhandhabung -Haltung -Kinematik (1) -Aufprall -Sportmedizinisch (1)
Weitere Funktionen	-Bibliotheken (1) -AnyScript -Umfangreiches GUI -Interface für markerbasiertes Motion Capture System (1) -Interface für IMU Motion Capture System (1) -Modellskalierung(2) -Frei bewegliches DMM(2) -Routine zur Muskelkalibrierung (1)	-Bibliotheken (1) -GUI -Frei bewegliches DMM(2)	-Bibliotheken (1) -GUI -Drei vorgeschriebene Modellhaltungen -Verletzungsevaluierung
Systemintegration	Stand alone	Stand alone	Stand alone
Modellkosten	Floating:7950Euro Node- locked:5500Euro (0)	Kostenlos mithilfe des Santos University Programm (2)	Free Access (2)
Punkte	14	12	7

Tabelle 3: Digitale Menschmodelle 4 und 5

			Vergleichssieger	
Name	GHBMC		BoB	
Firma	Global Human Body Models Consortium		BoB Biomechanics	
Graphische Komponenten	-Skelett -Volumenskelett -Anatomische Strukturen -Hüllfläche		-Skelett -Volumenskelett -Anatomische Strukturen	
Eigenschaften	-Geschlecht (1) -Perzentiles -Proportionen -Kinder (1)			
Manipulationen	-Vorwärtskinematik -Inverse Kinematik -Muskelmodellierung (1)		-Vorwärtskinematik -Inverse Kinematik -Motion Capture (2) -Muskelmodellierung (1)	
Analysen	-Sichtbarkeit -Erreichbarkeit(1) -Lastenhandhabung -Haltung	-Kinematik (1) -Sitz Komfort (1) -Aufprall	-Sichtbarkeit -Erreichbarkeit (1) -Lastenhandhabung -Haltung -Kinematik (1)	-Kinetik (1) -Kraft (1) -EMG -Teilkörper (2) -Muskelbelastung (2)
Weitere Funktionen	-Bibliotheken (1) -Zehn vorgeschriebene Modellhaltungen -Verletzungsevaluierung		- Bibliotheken (1) -Umfangreiches GUI -Interface für markerbasiertes Motion Capture System (1) -Interface für IMU Motion Capture System (1) -Modellskalierung (2) -Frei bewegliches DMM (2) -EMG Schnittstelle -Muskelkennzeichnung (1)	
Systemintegration	Keine Angabe gefunden		MATLAB	
Modellkosten	Kostenlose Akademische Lizenz (2)		Research 2497 Euro(2400USD) Ergo 3641Euro(3500USD). (1)	
Punkte	9		20	

Genauigkeitsevaluation der HoloLens 2 Tiefensensorik

Michael Pietschmann
Hochschule Reutlingen
Reutlingen, Deutschland
Michael.Pietschmann@Student.Reutlingen-University.de

Zusammenfassung

Microsofts HoloLens 2 besitzt in vielen Anwendungsbereichen durch ihre umfassende Sensorenausstattung ein hohes Einsatzpotential. Die Genauigkeit des integrierten Tiefensensors für Objektrekonstruktionen ist jedoch noch unzureichend erforscht. In dieser Arbeit wird die Genauigkeit des Tiefensensors durch ein kontrolliertes Experiment untersucht. Zur Durchführung der Messungen für das Experiment mit 17 verschiedenen Objekten wurde ein 3D-Mesh-Rekonstruktionsverfahren implementiert. Die mit der Unity Game-Engine und dem Mixed Reality Toolkit entwickelte Anwendung konnte genutzt werden, um eine maximale durchschnittliche Abweichung des Tiefensensors im Bezug auf 3D-Rekonstruktionen von 5.67 cm ±2.14 cm zu ermitteln. Des Weiteren können Objekte mit einer Abmessung von bis zu 0,5 cm rekonstruiert werden, solange eine Dimension des Objekts ca. 30 cm überschreitet. Mit den in dieser Arbeit erarbeiteten Ergebnissen kann der Nutzen der HoloLens 2 Tiefensensor-Kapazitäten für weiter Anwendungen besser eingeschätzt werden.

CCS Concepts

• **Computing methodologies** → **Mixed / augmented reality**; **Mesh geometry models**.

Keywords

HoloLens 2, Tiefensensorexperiment, 3D-Mesh-Rekonstruktion

1 Einleitung

Zentraler Bestandteil von Augemented Reality (AR) und Mixed Reality (MR) Anwendungen sind die verwendeten 3D-Modelle und ihre Einbettung in die reale Welt. Um eine möglichst optimale Verbindung zwischen der realen Welt und der virtuellen Welt zu schaffen, werden Tiefensensoren eingesetzt, damit Informationen über die reale Umgebung der Nutzer*innen gewonnen werden können. Die HoloLens 2 (HL2) von Microsoft[1] ist ein sogenanntes Head-Mounted Display (HMD) und besitzt neben ihren Displays, welche für Einblendungen virtueller Hologramme genutzt werden, auch einen integrierten Tiefensensor. Seit 2019 ist die HL2 auf dem Markt erhältlich und stellt damit aktuell den Industriestandard für HMD MR Geräte dar. Da die HL2 damit relativ neu ist, wurde in einer ersten Recherche herausgefunden, dass die Genauigkeit des verbauten Tiefensensors noch nicht ausführlich getestet wurde. Eine Evaluation dieses Tiefensensors und der HL2 3D-Rekonstruktions-Kapazitäten wird daher in diesem Paper durchgeführt.

Forschungsfrage Zur Betrachtung der technischen Fähigkeiten im Bezug auf die 3D-Rekonstruktion der HL2 wurde folgende

[1] https://www.microsoft.com/de-de/d/hololens-2/91pnzzznzwcp?activetab=pivot:technischendatentab, 29.09.22

Betreuerin Hochschule: Prof. Dr. rer. nat. Gabriela Tullius
Hochschule Reutlingen
Gabriela.Tullius@Reutlingen-University.de
Informatics Inside Herbst 2022
23. November 2022, Hochschule Reutlingen

Forschungsfrage formuliert: ***RQ 1:*** Welchen Detailgrad unterstützt die HL2 beim Scannen und 3D-Rekonstruieren physischer Objekte?

Methodik Zur Bildung einer Wissensgrundlage und der Beantwortung der Forschungsfrage wurden eine systematische Literaturrecherche durchgeführt, deren Ergebnisse in Kapitel 2 zusammengefasst sind. Es wurden Suchterme gebildet, die in den Datenbanken ACM Digital Library[2], IEEE Xplore[3], Springer Link[4] genutzt wurden, um nach passender Literatur zu suchen. Für die Suche nach bestehenden Projekten und Forschungsarbeiten wurden zusätzlich die Internetseiten GitHub[5], GitLab[6] und Twitter[7] eingesetzt. Die Genauigkeitsevaluation des HL2 Tiefensensors wurde mit einem kontrollierten Experiment unter Laborbedingungen durchgeführt. Zur Ausführung des Experiments wurde eine prototypische Anwendung mit der Unity Game-Engine zur 3D-Objektrekonstruktion implementiert.

2 Verwandte Arbeiten

Zur Beantwortung der Forschungsfrage RQ1 muss zunächst thematisiert werden, wie die Genauigkeit von Tiefensensoren grundsätzlich angegeben wird. Bei Betrachtungen von Produktseiten von Tiefensensoren wird die Genauigkeit des Sensors meist auf zwei verschiedene Arten angegeben. Die Genauigkeit wird entweder mit einer Prozentangabe[8], wie beispielsweise ±3%, oder in absoluten Werten mit der maximalen Abweichung in Millimeter[9] angegeben. Alle überprüften Dokumentationen nennen zudem die maximale Reichweite des Tiefensensors. Manche technischere Produktdokumentationen teilen darüber hinaus auch die Größe des Sensors in Pixeln und die maximale Abtastfrequenzrate mit[10]. Passende Fachliteratur nutzt zur Beschreibung der Genauigkeit häufig Diagramme, die die Messgenauigkeit in Abhängigkeit zur Distanz zum Objekt darstellen [2, Abb. 6, 8, Abb. 8, 10, Abb. 3 & 4, 16, Tab. 1]. Es wird also eine Angabe der Genauigkeit oder des Rauschen (engl. Noise) des Sensors in mehreren Schritten bzw. einer Kurve abhängig von der Distanz zum Objekt durchgeführt.

Mit dieser Grundlage wird folgend dargelegt, welche Angaben in der offiziellen HL2-Dokumentation[11] veröffentlicht sind und welche Ergebnisse bezüglich des Detailgrades bisher von anderen Forschungsgruppen für die HL2 identifiziert wurden. Die HL2 besitzt einen 1-Megapixel (MP) Time-of-Flight (ToF) Tiefensensor[12]. Die Aktive-Korrespondenzen-Analyse [13] wird für diese ToF-Technologie verwendet und basiert dabei auf dem Aussenden von Licht in einer bestimmten sichtbaren oder ultra-violetten Frequenz [13, 16]. Diese in einem spezifischen Muster ausgesendete Lichtfrequenzen werden von einer passenden Kamera aufgenommen [16]. Das aufgenommene Muster kann mit dem ursprünglichen Muster verglichen und damit eine Tiefeninformation für jedes Pixel der Kamera abgeleitet werden [16]. Auf diese Weise entsteht eine Tiefenmap, für die aktuelle Position der Kamera bzw. des Betrachters [13]. Durch mehrere Messungen an unterschiedlichen Positionen kann ein gesamtes dreidimensionales Abbild der Umgebung erzeugt werden [13]. Die HL2 nutzt diese Technologie, indem sie den Tiefensensor der Microsoft Kinect einsetzt[13].

[2] https://dl.acm.org/, 16.09.22
[3] https://ieeexplore.ieee.org/Xplore/home.jsp, 16.09.22
[4] https://link.springer.com/, 16.09.22
[5] https://github.com, 16.09.22
[6] https://gitlab.com, 16.09.22
[7] https://twitter.com, 16.09.22
[8] https://shop.m5stack.com/products/tof-sensor-unit, 20.09.22
[9] https://www.st.com/en/imaging-and-photonics-solutions/vl53l4cx.html, 20.09.22
[10] https://www.ti.com/lit/ds/sbas704b/sbas704b.pdf?ts=1659530925126&ref_url=https%253A%252F%252Fwww.google.com%252F, 20.09.22
[11] https://docs.microsoft.com/de-de/hololens/hololens2-hardware, 18.09.22
[12] https://www.microsoft.com/de-de/d/hololens-2/91pnzzznzwcp?activetab=pivot:technischendatentab, 18.09.22
[13] https://www.businesswire.com/news/home/20190917005239/en/Mixel, 18.09.22

Dieser Kinect Sensor ist mit zwei Modi[14] ausgestattet, sodass zwei Messbereiche[15] ausgelesen werden können. Ein Bereich ist auf der HL2 für die Handgestenerkennung optimiert[15] und mit einer Frequenz von 45 Bilder-Pro-Sekunde[14](engl. Frames-Per-Second, FPS) auf einen Bereich bis 1.0 Meter[14]eingestellt. Der zweite Bereich ist für die Fernwahrnehmung konstruiert, wird mit einer niedrigeren Frequenz von 1-5 FPS[14] ausgelesen und umfasst einen Sichtbereich von 1.0 bis 5.0 Meter. Zur Genauigkeit dieses Sensors wird über die Angabe der 1024x1024 Pixel (1 MP) Tiefenmap jedoch keine Aussage gemacht. Eine Verbesserung des Sensors im Bezug auf die Genauigkeit wird von Microsoft nur noch über einen Rauschreduzierungsalgorithmus ausgeführt[15].

Im Folgenden werden **verwandte Arbeiten** beschrieben, die Beobachtungen zur Genauigkeit der HL2 Tiefensensorik liefern.

Guan et al. [6] stellen in ihrer Arbeit ein Deep Neural Network (DNN) vor, welches Objekterkennung auf mobilen MR Headsets durchführen kann. Darin werden auch Versuche und Vergleiche mit der HL2 durchgeführt, indem die Tiefensensordaten als Eingabeparameter in das DNN eingegeben werden. Beim Auslesen der Tiefenmap wurde festgestellt, dass nur eine Tiefenbild-Auflösung von 360x360 Pixel für interaktive Anwendungen erreicht werden kann. Dies wird mit der verringerten FPS Frequenz des Sensors von 1-5 FPS für volle 1 MP Tiefenbilder begründet. Für performante interaktive 30 FPS Anwendungen beträgt die Tiefenbildgröße somit nur 360x360 Pixel, was die Genauigkeit der Messungen deutlich beeinflussen kann. Das Auslesen der Tiefensensordaten erfolgt dabei über den Research Mode der HL2, welcher direkten Zugriff auf die Sensordaten ohne Vorverarbeitung durch bereits implementierte Algorithmen bietet.

Teruggi et al. [15] nutzen die HL2 mit der integrierten Spatial Mapping Technologie zu 3D-Rekonstruktionstests von sehr großen und weitläufigen Architekturstrukturen. Bei den Experimenten in einem Kirchenschiff und dem Glockenturm-Treppenhaus wurden die aufgezeichneten Meshdaten mit dem Ergebnis von Laserscanaufnahmen verglichen. Als Genauigkeitsfehler ermitteln die Autoren eine maximale durchschnittliche Abweichung von 3 cm. Die Ground Truth Laserscanauflösung beträgt dabei 1 cm. Zudem ist die größte absolute Abweichung von der Ground Truth entlang der x-Achse mit 0.59 m beziffert. Des Weiteren kann ein maximaler Messradius der HL2 von sechs bis sieben Metern identifiziert werden. Im Treppenhaus des Turmes wird zudem herausgefunden, dass eine zu enge Aufnahmeumgebung von unter 0.5 m zu starken Unterschieden in den Messungen führen.

Kuan et al. [14] vergleichen in ihrer Arbeit die Performance drei verschiedener Tiefensensor-Systeme im Außenbereich. Der Kinect v2 Sensor wird hierbei ebenfalls im Vergleich verwendet. Da dieser Sensor in der HL2 zum Einsatz kommt, können die Messergebnisse in der vorliegenden Arbeit verwendet werden. Das Team um Kuan et al. [14] führt dabei Versuche durch mit jeweils variierender Distanz zu einer ebenen Fläche unter Variation der Intensität der Sonneneinstrahlung, der Materialbeschaffenheit, des Messwinkels, des Sonnenlicht-Einstrahlungswinkels und der Objektform. Für eine min. 90 prozentige Genauigkeit der Abstandsmessungen des Kinect v2 Sensores identifizieren Sie weniger als drei Meter für sehr geringen Lichteinfall, weniger als zwei Meter für mittlere Sonneneinstrahlung und weniger als 1.5 Meter für eine sehr starke Lichtintensität. Nur die Objektoberfläche hat noch einen Einfluss auf die Messgenauigkeit. Sie kann bei näheren Messungen detaillierter nachgebildet werden. Die anderen getesteten

[14]https://www.microsoft.com/en-us/research/blog/microsoft-hololens-2-improved-research-mode-to-facilitate-computer-vision-research/?ocid=msr_blog_hl2_eccv_tw, 19.09.22

[15]https://www.youtube.com/watch?v=S0fEh4UdtT8, 19.09.22

Einflussfaktoren haben dabei keinen signifikanten Einfluss auf die Messgenauigkeit.
Da die HL2 erst seit 2019 auf dem Markt verfügbar und somit relativ neu ist, konnte speziell für die HL2 keine weitere Literatur gefunden werden. Jedoch wurde das Themengebiet bereits auf der ersten Generation der HoloLens erforscht, welche aufgrund der Ähnlichkeit der Geräte in dieser Arbeit beachtet werden kann. Vor allem Hübner et al. haben im Bereich der Rekonstruktion von Innenräumen mit der HoloLens 1 drei Paper veröffentlicht [7–9]. In der ersten Veröffentlichung [9] wurden mehrere Scanvorgänge von Innenräumen mit der HoloLens 1 durchgeführt und mit dem Mesh eines Terrestrial Laserscanners (TLS) verglichen. Es konnte dabei eine starke Varianz der Genauigkeit von fast 0 cm Abweichung bis deutlich über 10 cm Differenz identifizierte werden. Wobei die stärksten Differenzen in den eher schwerer zugänglichen Positionen, wie den oberen Ecken eines Raumes zu finden sind [9, vgl. Abb. 8].
Hübner et al. haben darüber hinaus in ihrer Veröffentlichung im Februar 2020 [8] den Zeitfaktor der Benutzung der HoloLens zusätzlich untersucht. Sie identifizieren, dass die Messgenauigkeit in den ersten 40 Minuten nach Start des Gerätes starken Schwankungen unterliegt, jedoch im Bereich von 2 mm bis 6 mm Abweichung bleibt. Zwischen 40 und 60 Minuten wird eine stetige Abweichung von 6 mm gemessen und ab einer Stunde erhöhte sich die stetige Abweichung auf 8 mm. Zudem führen Hübner et al. in dieser Arbeit ebenso Messungen über die Distanz und das dabei entstehende Rauschen des Sensors durch. Sie können dabei herausfinden, dass ab einer Messdistanz von 2,5 m das Rauschen steil ansteigt. Wobei das Team ebenso identifiziert, dass das Rauschen stark abnimmt, wenn nur die besten 75% der Pixel für die Messung verwendet werden. Die Abweichung des Rauschens bleibt bei diesen Messungen unter 1 cm.
In der neusten Veröffentlichung aus dem Jahr 2020 [7] wird ein Voxel-basiertes-Verfahren angewendet mit einer Voxelgröße von 5 cm. Aufgrund der min. Mesh-Dreiecksgröße der HoloLens 1 von 5 cm, wurde diese Größendimension der Rekonstruktionsgenauigkeit von den Autoren bewusst festgelegt.
Aus der Literatur kann somit mit eine durchschnittliche Genauigkeit des rekonstruierten Objekt-Gitternetzes (engl. Mesh) von ±3-5 cm erwartet werden. Wobei die Sensorgenauigkeit der Rohdaten des Sensors prinzipiell im Bereich von 0.5 mm bis 1 cm liegen sollte.

Im weiteren Verlauf werden Literaturquellen und Projekte vorgestellt, die sich mit der 3D-Mesh-Rekonstruktion beschäftigen, da diese für das Genauigkeitsexperiment unabdingbar sind. Dazu wird die Frage formuliert, wie die Sensordaten der HL2 ausgelesen werden können. Dafür konnten drei Methoden identifiziert werden. Das Mixed Reality Toolkit (MRTK; aktuell Version 2)[16] stellt die erste Möglichkeit dar. Es handelt sich dabei um ein Toolkit von Microsoft, welches speziell für die Anwendungsentwicklung für die HoloLens (Gen. 1 & 2) veröffentlicht wurde. Jung et al. [12] nutzen das MRTK beispielsweise zur 3D-Model-Rekonstruktion, wie auch in Kapitel 3 vorgestellt wird. Es wird dabei auf die Spatial Awarness API des MRTK zugegriffen, um die Meshdaten des Spatial Mappings auszulesen. Das MRTK kann hierbei in Verbindung mit der Unity Game-Engine oder der Unreal Game-Engine eingesetzt werden. Auch Communityprojekte nutzen das MRTK, um beispielsweise Objekterkennung[17] durchzuführen. Eine weitere Möglichkeit bietet der Research Mode, der auf der HoloLens (Gen. 1 & 2) aktiviert werden kann. Auf die Daten des Research Mode können dabei mit der

[16] https://learn.microsoft.com/de-de/windows/mixed-reality/mrtk-unity/mrtk2/?view=mrtkunity-2022-05, 21.09.22
[17] https://gitlab.com/mpavlopoulou/hololensobjectdetectionapp/-/tree/main/, 21.09.22

HoloLens2ForCV[18] Bibliothek zugegriffen werden. Puljiz et al. [4] benutzen den Research Mode mit der HoloLensForCV Bibliothek beispielsweise für die Voxel-basierte Rekonstruktion einer Arbeitsumgebung mit der HoloLens 1, um einen Roboterarm sicher bewegen zu können. Die letzte identifizierte Möglichkeit für einen Zugriff auf die Sensordaten der HL2 stellt das Community geförderte Mixed Reality Toolkit XRTK[19] dar. Es bietet ebenso eine Spatial Awarness API Implementierung, welche Zugriff auf das erzeugte Raummesh gewährt. Sowohl das XRTK als auch das MRTK führen somit eine Vorverarbeitung der Sensordaten des Tiefensensors durch. Jedoch kann nur über den Research Mode der Rohdatenstream des Tiefensensors ausgelesen werden.

Nachdem thematisiert wurde wie die Sensordaten ausgelesen werden können, wird nun die 3D-Rekonstruktion betrachtet. Dabei wird spezifisch die Frage gestellt, welche anwendbaren Algorithmen existieren, um aus einem gescannten Mesh ein zusammengehörendes und vollständiges Objekt zu formen? Wie kann also eine sogenannte Formvervollständigung (engl. Shape completion) durchgeführt werden? Eine zufriedenstellende Formvervollständigung ist keine einfache Aufgabe, wie an den vielen präsentierten Möglichkeiten von Tao Ju [11] zu sehen ist. Die meisten Methoden basieren auf einer regelbasierten Triangulation der Vektoren und eventuell einem nachträglichen Glätten des gesamten Meshes. Darüber hinaus existieren Voxel-basierte Verfahren, wie zum Beispiel von Puljiz et al. [4] mit einer Voxel-Gitter-Filterung durchgeführt wird. Aktuell wird die Formvervollständigung von komplexen 3D-Meshes überwiegend mit Maschinellen Lernverfahren durchgeführt. Dai et al. [1] nutzen beispielsweise ein Convolutional Neural Network mit 3D-Encodern zur 95.5 prozentiger Formvervollständigung. Diese Verfahren besitzen jedoch meist einen begrenzten Einsatzbereich, da der Datensatz für das Training nur eine oder mehrere spezielle Modellkategorie, wie architektonische Strukturen umfasst [1, vgl. Kapitel 8]. Hier wird also ein Vorwissen über die zu scannenden Objekte benötigt.

3 3D-Mesh-Rekonstruktions-Anwendung

Zur Durchführung des Experiments auf der HL2 wird eine Anwendung benötigt, welche das zu scannende Objekt rekonstruieren kann. Diese Anwendung zur 3D-Mesh-Rekonstruktion wird im Folgenden geschildert.

3.1 Technologieauswahl

Wie in Kapitel 2 ausgeführt, existieren drei Möglichkeiten zur Sensorauslesung auf der HL2. Zur Implementierung der 3D-Rekonstruktionsanwendung für das Experiment wird das MRTK mit der Unity Game-Engine genutzt. Um diese Entscheidung treffen zu können, wurden alle drei Möglichkeiten mit passenden Beispielanwendungen oder Standardtutorials der Bibliotheken zum Auslesen der Tiefensensoren getestet. Dabei stellen sich die vorhandene Spatial Mapping Funktionalität[16] des MRTK, die fehlerfreie Installation und der einfache Zugriff auf die Sensordaten am effektivsten für diese Arbeit heraus. Aus einem Bildvergleich bei den durchgeführten Vortests ist zu schließen, dass eine Rohdatenauslesung über den Research Mode zu einem genaueren Ergebnis führen könnte. Aufgrund der stark erhöhten Komplexität der zu implementierenden Sensordaten-Verarbeitungsalgorithmen wird die schlechtere Genauigkeit des MRTK jedoch hingenommen.

[18] https://github.com/microsoft/HoloLens2ForCV, 21.09.22
[19] https://xrtk.io/README.html, 22.09.22

3.2 Technische Umsetzung

Zur Umsetzung des 3D-Mesh-Rekonstruktionsverfahrens wurde eine Unity 2021.3.8f1 LTS Anwendung aufgesetzt. Die Konfiguration für die HL2 und der Zugriff auf die Meshdaten erfolgte durch Integration des MRTK. Zur Interaktion, Konfiguration und Starten des Scanvorgangs besitzt die Unity-Szene des Experiments ein individuelles Einstellungsmenü. Durch das Auslesen der Meshdaten mit dem MRTK ergibt sich der in Abbildung 1 gezeigte technische Ablauf für die Rekonstruktion. Basierend darauf wurde eine Softwarearchitektur entworfen (siehe Klassendiagramm im Projektrepository[22]). Die Architektur ist zur einfachen Austauschbarkeit der Schritte mit dem Fassaden-Entwurfsmuster [5] in mehrfacher Kombination mit dem Strategie-Entwurfsmuster [5] designet.
Im Folgenden wird das Vorgehen der Implementierung jedes Einzelschritts aus Abbildung 1 kurz zusammengefasst.

Objekt scannen Beim Scannen geht es darum, das Mesh des Objektes zu erzeugen und somit die Hauptgrundlage der Rekonstruktion zu schaffen. Im MRTK wird dazu eine Instanz der Klasse SpatialObjectMeshObserver konfiguriert, indem das Update Intervall auf 0.25 Sekunden, die Observationsabmessungen auf 5 m erhöht und der Detailgrad (engl. Level of Detail, LoD) auf *unlimited* eingestellt wird. Diese Einstellungen führen somit zu einem automatischen Erfassen der 3D-Abmessungen des zu scannenden Objekts und werden mit dem Starten des Scans über das erstellte Menü angewendet.

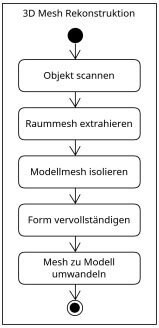

Abbildung 1: UML Aktivitätsdiagramm des 3D-Rekonstruktionsverfahrens

Raummesh extrahieren Um das Mesh des gescannten Raumes zu extrahieren, wird es aus dem SpatialObjectMeshObserver ausgelesen. Dabei findet eine Vereinigung der einzelnen sogenannten Submeshes statt, da das Spatial Mapping System das Raummesh in einzelne Abschnitte unterteilt.

Modellmesh isolieren Bei diesem Schritt findet die Filterung des Meshes des Zielobjekts aus dem gesamten Raummesh statt. Dafür wird ein virtueller Würfel genutzt, der in der Anwendung positioniert wird und die 3D-Begrenzung für das Zielobjekt darstellt. Alle Vektoren und Dreiecke des Raummeshes, die sich innerhalb dieser Begrenzungen befinden werden isoliert und in ein neues Meshobjekt kopiert. Zur Filterung werden einfache Koordinatenvergleiche mit den Vektoren des Meshes durchgeführt.

Form vervollständigen In vielen Fällen beinhaltet das im vorherigen Schritt extrahierte Mesh Löcher und somit fehlende Dreiecke, da das Zielobjekt oft nicht in alle drei Dimensionen ganzheitlich umrundet werden kann. Wie in Kapitel 2 bereits thematisiert existieren viele Möglichkeiten solche Löcher zu schließen. Maschinelle Lernverfahren wurden für diese Arbeit ausgeschlossen, da möglichst beliebige Objekte gescannt werden sollen und viele Verfahren Objektvorwissen voraussetzen. Zudem ist die Implementierung eines Maschinellen Lernverfahrens keine triviale Aufgabe, da die Hardwarebeschränkungen der HL2 berücksichtigt werden müssen. Es wird daher auf eine einfache Meshbibliothek namens Geometry3DSharp[20] zurückgegriffen, deren *SimpleHoleFiller* Klasse eine Triangulierung

[20] https://github.com/gradientspace/geometry3Sharp, 23.09.22

der Löcher durch eine Identifizierung der Kanten des Meshes ausführt.

Mesh zu Modell umwandeln Zur Verwendung des Meshes in der Unity Anwendung muss aus dem Mesh ein Unity GameObject erstellt werden. Durch eine Mittelpunktbestimmung des Meshes kann das Mesh dafür zum Koordinatenursprung verschoben werden, um die weitere Benutzung des Modells in der Anwendung zu optimieren. Zudem wird die Objektrekonstruktion abgeschlossen, indem eine Materialzuweisung stattfindet, welche die Gitterstruktur des Modells visualisiert. Für eine komfortable Betrachtung des Modells wird zudem eine Hand-Interaktions-Komponente hinzugefügt. Abbildung 2 zeigt abschließend eine Rekonstruktion eines Tisches.

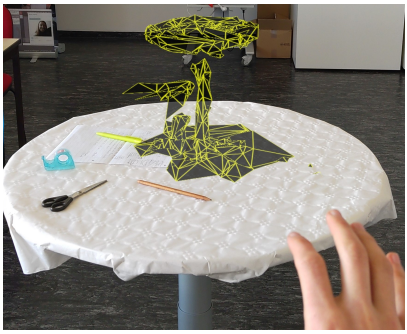

Abbildung 2: Beispielhafte Rekonstruktion eines Tisches (Modell skaliert)

4 Tiefenscan Experiment

Zur Evaluation der Genauigkeit der 3D-Mesh-Rekonstruktion wird in diesem Kapitel das Experiment geschildert mit dem Genauigkeitsmessungen durchgeführt wurden. Zur geregelten Durchführung des kontrollierten Experiments wurden dabei die Schritte von Cash et al. [3] und Rebecca Bevans Blogeintrag[21] befolgt.

[21] https://www.scribbr.com/methodology/experimental-design/, 23.09.22

4.1 Versuchsplanung

Für das Experiment wurden die unabhängigen, die abhängigen und die äußeren Einflussfaktoren herausgearbeitet. Die unabhängigen und abhängigen Variablen sind in Tabelle 1 aufgelistet und ihre Kontrollparameter im Experiment zugeordnet. Als äußere Haupteinflussfaktoren konnten die 360° Umrundung mit zwölf Messpunkten in 30° Abständen, die konstante Messhöhe und das konstant gehaltene virtuelle Messverfahren identifiziert werden. Alle weiteren äußeren Einflussfaktoren, welche die Messwerte deutlich beeinflussen, können im Projektrepository[22]. nachgeschlagen werden. Aus diesen Variablen wurden dann drei Hypothesen abgeleitet. $H0_1$: Die Distanz zum Objekt hat keinen Einfluss auf die Genauigkeit der Rekonstruktion (m=0). $H1_1$: Eine höhere Distanz zum Objekt führt zu einer ungenaueren Rekonstruktion (m>=0). $H0_2$: Die Größe des Objekts hat keinen Einfluss auf die Genauigkeit der Rekonstruktion (m=0). $H1_2$: Ein kleineres Objekt führt zu einer ungenaueren Rekonstruktion (m>=0). $H0_3$: Unterschiedliche Formen des Objekts haben einen Einfluss auf die Genauigkeit der Rekonstruktion (d>10%). $H1_3$: Die Form des Objektes hat keinen Einfluss auf die Genauigkeit der Rekonstruktion (d<=10%). Die Erhöhung des üblichen Schwellenwerts bei Hypothesentests von 5% auf 10% ist dabei durch die starke Varianz der Objekte in die Y-Dimension zu erklären. Mit diesen Hypothesen konnten die Kontrollparameter, wie Tabelle 1 zeigt, festgelegt werden.

4.2 Versuchsdurchführung

Für jedes Objekt bzw. für jede Umrundung des Objekts mit der festen Distanz muss die Anwendung immer neu gestartet werden, um in den Windowseinstellungen die vorherigen Meshdaten löschen zu können. Abbildung 3 zeigt den gewählten Versuchsaufbau mit den implementierten Hilfsmitteln. Auf dem zu sehenden Tisch wurden bei der Durchführung

Tabelle 1: Abhängige und unabhängige Variablen des Experiments

Name	Einheit	Kontrollparameter
Unabhängige Variablen		
Distanz zum Objekt	m	0.5, 0.75, 1, 1.5, 2, 3 –> zufällige Reihenfolge
Größe des Objekts	cm	Quaderbreite: 30, 20, 10, 6.6, 3.3, 1.6, 1, 0.7 Zylinderdurchmesser: 20, 12, 9.25, 5, 2.75, 2, 1.4, 1, 0.5
Form des Objekts		Zwei Messreihen: 8 Quader, 9 Zylinder
Abhängige Variablen		
Rekonstruierte Breite (x-Achse)	m	Virtuelles Raycasting Messverfahren, Genauigkeit: 10^{-7}
Rekonstruierte Höhe (y-Achse)	m	Virtuelles Raycasting Messverfahren, Genauigkeit: 10^{-7}
Rekonstruierte Tiefe (z-Achse)	m	Virtuelles Raycasting Messverfahren, Genauigkeit: 10^{-7}

somit die Objekte platziert. Durch das blaue, virtuell eingeblendete, radiale Netz auf dem Boden kann die Umrundung in konstantem Abstand durchgeführt werden. Nachdem alle Messpunkte abgelaufen sind, kann der Scanprozess beendet werden, der Begrenzungswürfel platziert und die Rekonstruktion bis hin zu Ausgabe des Modells als eigenständiges Objekt gestartet werden. Danach findet die Messung mit dem selbst implementierten virtuellen Messwerkzeug in alle drei Dimensionen statt. Pro Dimension wurden zwei Messpunkte gewählt, die jeweils die minimalen und maximalen Ausmaße der Rekonstruktion kennzeichneten. Damit die virtuellen Messwerte mit den realen Ausmaßen der Objekte verglichen werden können, wurde zudem ein Kalibrierungswürfel mit 1x1x1 Meter Ausmaßen der Szene hinzugefügt und überprüft. Ein Meter entspricht so genau einer Messeinheit im virtuellen Koordinatensystem.

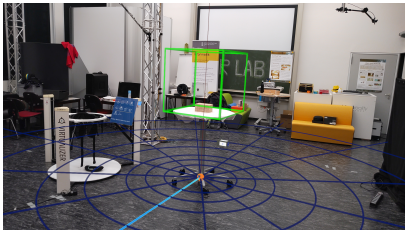

Abbildung 3: Aufbau des Experiments mit allen implementierten Hilfsmitteln

4.3 Beobachtungen

B1: Das erzeugte Mesh variiert während einer Umrundung bzw. Messung stark. Teilweise konnten während der Umrundung genauere Rekonstruktionsergebnisse über die Gitternetzvisualisierung beobachtet werden, als am Ende der Messung als Ergebnis dokumentiert wurde. B2: Zudem wurde beobachtet, dass die Größe der Mesh Dreiecke von Messung zu Messung stark variierten. Es konnte nicht identifiziert werden, ob die Varianz nur durch die immer minimal abweichenden Messungen oder durch eine nicht vorgesehene Änderung der LoD MeshObserver Einstellung Zustande kommt. B3: Die Meshgenauigkeit wurde bei manchen Messungen durch die verringerte Messhöhe verbessert und manchmal verschlechtert. Diese Veränderungen wurden nur über die Live-Visualisierung des Meshes wahrgenommen und es konnte keine Systematik festgestellt werden, wann eine Verbesserung bzw. eine Verschlechterung eintritt.

4.4 Messergebnisse

Einige Ergebnisse der Messungen sind in den Abbildungen 4-6 visualisiert. In Kapitel 5 findet die Diskussion dieser Ergebnisse statt. Die zu den Diagrammen gemessenen Rohdaten können im Projektrepository[22] eingesehen werden. Über dieser Ausarbeitung hinaus werden die Rohdaten noch ausführlicheren Untersuchungen unterzogen, um weitere Ergebnisse zu extrahieren.

5 Diskussion

Im Bezug auf die Anwendung des Experiments kann die Machbarkeit einer 3D-Mesh-Rekonstruktion mit dem gewählten Ansatz in Kapitel 3 gezeigt werden. Für größere Objekte ist die Qualität der Rekonstruktion rein visuell sehr zufriedenstellend. Verbessert werden könnte vor allem das

[22] https://gitlab.reutlingen-university.de/pietscmi/3dreconstructionhololens2,

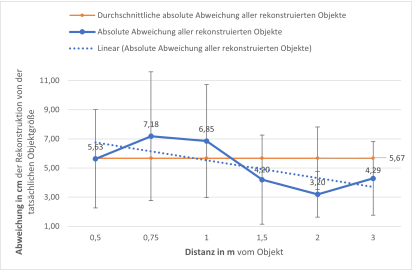

Abbildung 4: Abweichungen der Rekonstruktionen abhängig von der Distanz

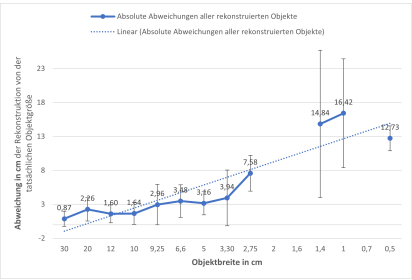

Abbildung 5: Abweichungen der Rekonstruktionen abhängig von der Objektgröße

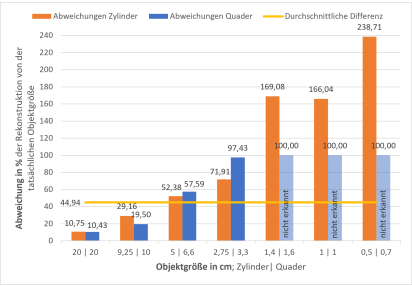

Abbildung 6: Vergleich der Abweichungen der Rekonstruktionen der verschiedenen Objektformen

Formvervollständigungsverfahren, da dieses nur die nötigste Schließung der Löcher im Mesh übernimmt. Ein komplexerer Algorithmus oder der Einsatz eines Maschinellen Lernverfahrens könnte die Qualität des Gesamtmodells dabei verbessern. Eine Verbesserung der Genauigkeit des rekonstruierten Meshes, könnte weiterhin der Einsatz des Research Mode zum Auslesen der Tiefendaten bringen. Da die vom MRTK implementierte Spatial Mapping Technik hauptsächlich für das Rekonstruieren eines gesamten Raumes optimiert ist, könnte durch ein anderes Mesherzeugungsverfahren eventuell eine Verbesserung des Meshes erzielt werden.

Folgende Erkenntnisse können darüber hinaus aus dem Experiment gewonnen werden. Zunächst kann die Beurteilung und Evaluation der Hypothesen aus Kapitel 4 durchgeführt werden. Fehlende Messwerte durch nicht erkannte Objekte wurden bei den Berechnungen hierfür ausgelassen, wie beispielsweise Abbildung 5 zeigt. Für die Hypothesen eins und zwei wurde dazu eine lineare Regressionsanalyse durchgeführt, welche die Regressionskoeffizienten m1 = -0,6091 (H_1) und m2 = 1,134 (H_2) ergaben. Durch den negativen Koeffizienten und wie das Diagramm in Abbildung 4 zeigt, muss $H1_1$ abgelehnt werden. Jedoch kann $H0_1$ auch nicht unbedingt angenommen werden, da der Regressionskoeffizient negativ und nicht Null ist. Es könnte hierbei sogar die Hypothese formuliert werden, dass eine weitere Entfernung zu einer höheren Genauigkeit führt. Das müsste jedoch in einer getrennten Untersuchung nochmals bestätigt werden. Zudem lässt sich aus den Messungen des Hypothesenpaars H_1 ableiten, dass die HL2 die höchste Genauigkeit in einer Entfernung von zwei Meter mit einer maximalen durchschnittlichen Abweichung von 3,20 cm erzielt.

Die Null-Hyphothese $H0_2$ muss aufgrund des positiven m2-Koeffizienten abgelehnt werden. Im Gegensatz zur ersten Alternativhypothese kann Hypothese $H1_2$ jedoch angenommenen werden, wie Abbildung 5 zeigt.

Im Bezug auf die Form der Objekte, wurde die mittlere Differenz d vergleichbarer Objekte in Prozent berechnet. Diese Differenz von 44,94%, welche größer als der festgelegte Schwellenwert von 10% ist, bestätigt somit die Hypothese $H0_3$. Die erwartete

Alternativhypothese $H1_3$ kann also nicht bestätigt werden, sodass diese abgelehnt und $H0_3$ weiterhin angenommen werden muss. Abbildung 6 zeigt hierfür den Vergleich der beiden Messreihen.
Im Bezug auf die Beobachtungen B1-B3 sollten noch folgende Überlegungen in Betracht gezogen werden. Um den Scanvorgang zu optimieren und somit B1 zu berücksichtigen, sollte das Vorgehen zur Auswahl der Messstellen nach Möglichkeit überarbeitet werden. Dabei könnte eventuell die Aggregation von vier oder mehr Einzelmessungen zu einem insgesamt genaueren Ergebnis der Messungen führen. Für Beobachtung B2 wäre eine Überprüfung des gesetzten LoD zu empfehlen. Zur Optimierung des Messverfahrens im Bezug auf Beobachtung B3 könnten getrennte Messreihen mit unterschiedlichen Messhöhen Klarheit verschaffen. Dadurch könnte identifiziert werden, inwieweit der durch die Messhöhe entstehende Messwinkel Einfluss auf die Messgenauigkeit nimmt.
Insgesamt sollten auch noch folgende Einsichten berücksichtigt werden. Die maximale durchschnittliche Abweichung aller Messungen von 5,67 cm ±2,14 cm ist festzuhalten. Zudem kann die Erkennung und somit die erfolgreiche Rekonstruktion von sehr schmalen Objekten, wie der 0,5 cm Zylinder, durchaus durchgeführt werden, solange das Objekt in eine Dimension ein bestimmte Länge überschreitet. Diese Länge kann aus dem Experiment aktuell nur auf ca. 30 cm geschätzt werden. Darüber hinaus weisen die rekonstruierten Modelle teilweise nicht mehr die gleiche Form auf, wie ihre gescannten Objekte in der realen Welt. Abbildung 7 zeigt beispielsweise, dass eine Ähnlichkeit zu dem verwendeten Holzzylinder nicht zwangsläufig wiederzuerkennen ist. Die Rekonstruktionen besitzen ähnliche Abmessungen, aber die Form muss daher ab einer Größe von unter zehn Zentimetern nicht unbedingt übereinstimmen. Um eine detailliertere Genauigkeitsaussage in diese Richtung treffen zu können, sollte

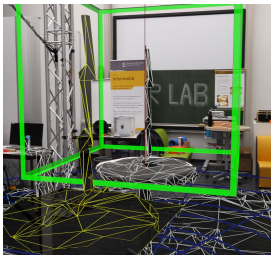

Abbildung 7: Ergebnis der Rekonstruktion des 1,4 cm Holzzylinders

ein Vergleich des Meshes mit einem TLS erzeugen Mesh ausgeführt werden.
Die Limitierungen der Ergebnisse des kontrollierten Experiments liegen vor allem in der Auswahl der Objekte, dem gewählten Umrundungs- bzw. Scanprozessverfahrens und dem selbst implementierten Messinstruments. So können die Ergebnisse nur für ähnliche Objekte übertragen werden, die auch mit einem ähnlichen Scanprozess vermessen wurden. Des Weiteren fehlt eine Validierung der Genauigkeit des Messvorgang mit dem verwendeten virtuellen Messinstrument Auch eine Wiederholung der Messreihen würden die Aussagekraft der Ergebnisse noch verbessern. Die Messergebnisse sollten daher nur unter Berücksichtigung dieser Punkte verwendet werden, bieten jedoch sicherlich gute Richtwerte.

6 Fazit

In dieser Ausarbeitung wird ein erfolgreicher 3D-Mesh-Rekonstruktionsansatzes unter Einsatz der HL2, des MRTK und der Meshbibliothek Geometry3DSharp vorgestellt. Dieser Ansatz wurde daraufhin in einem Experiment verwendet, um die Genauigkeit der Rekonstruktionen zu evaluieren. Aus den 17 rekonstruierten Objekten und den digitalen Vermessungen der erzeugten Meshes kann eine maximale durchschnittliche Abweichung von 5,67 cm ±2,14 cm festgestellt werden. Des Weiteren wurde identifiziert, dass die genauesten Messungen in einer Distanz von zwei Metern erzielt werden, die Form der

gescannten Objekte eine Rolle spielt und die Genauigkeit mit der Größe der Objekte abnimmt. Aus den Ergebnissen des Experiments können nachfolgende Experimente abgeleitet werden, um die Genauigkeit der HL2 im Bezug auf 3D-Rekonstruktionen weiter zu evaluieren. Zudem kann das hier vorgestellte Verfahren der 3D-Rekonstruktion vor allem durch die Verwendung des Research Modes noch verbessert werden.

Literatur

[1] A. Dai, C. R. Qi und M. Nießner. „Shape Completion Using 3D-Encoder-Predictor CNNs and Shape Synthesis". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, S. 6545–6554. DOI: 10.1109/CVPR.2017.693.

[2] Giacomo Borghi u. a. „Sub-3 mm, near-200 ps TOF/DOI-PET imaging with monolithic scintillator detectors in a 70 cm diameter tomographic setup". In: *Physics in medicine and biology* 63.15 (2018), S. 155006. DOI: 10.1088/1361-6560/aad2a6.

[3] Philip Cash, Tino Stanković und Mario Štorga. *Experimental design research: Approaches, perspectives, applications*. Springer, 2016. ISBN: 978-3-319-33779-1.

[4] D. Puljiz u. a. „What the HoloLens Maps Is Your Workspace: Fast Mapping and Setup of Robot Cells via Head Mounted Displays and Augmented Reality". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, S. 11445–11451. DOI: 10.1109/IROS45743.2020.9340879.

[5] Karl Eilebrecht und Gernot Starke. *Patterns kompakt: Entwurfsmuster für effektive Softwareentwicklung*. 5. Aufl. IT kompakt. Berlin und Heidelberg: Springer Vieweg, 2019. ISBN: 978-3-662-57937-4.

[6] Yongjie Guan u. a. „DeepMix". In: *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*. ACM Digital Library. 2022, S. 28–41. DOI: 10.1145/3498361.3538945.

[7] P. Hübner, M. Weinmann und S. Wursthorn. „Voxel-Based indoor reconsturction from HoloLens triangle meshes". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. 2020, S. 79–86. DOI: 10.5194/isprs-annals-V-4-2020-79-2020.

[8] Patrick Hübner u. a. „Evaluation of HoloLens Tracking and Depth Sensing for Indoor Mapping Applications". In: *Sensors (Basel, Switzerland)* 20.4 (2020). DOI: 10.3390/s20041021.

[9] Patrick Hübner u. a. „Evaluation of the Microsoft HoloLens for the Mapping of Indoor Building Environments". In: *Publikationen der DGPF* 28 (2019).

[10] Ryan M. Jans, Adam S. Green und Lucas J. Koerner. „Characterization of a Miniaturized IR Depth Sensor With a Programmable Region-of-Interest That Enables Hazard Mapping Applications". In: *IEEE Sensors Journal* 20.10 (2020), S. 5213–5220. DOI: 10.1109/JSEN.2020.2971595.

[11] Tao Ju. „Fixing Geometric Errors on Polygonal Models: A Survey". In: *Journal of Computer Science and Technology* 24.1 (2009), S. 19–29. DOI: 10.1007/s11390-009-9206-7.

[12] Younhyun Jung u. a. „Model Reconstruction of Real-World 3D Objects: An Application with Microsoft HoloLens". In: *Intelligent Scene Modeling and Human-Computer Interaction*. Springer Int. Publishing und Imprint Springer, 2021, S. 89–104. DOI: 10.1007/978-3-030-71002-6_6.

[13] Reinhard Koch und Johannes Bruenger. „Depth Estimation". In: *Computer Vision: A Reference Guide*. Springer International Publishing, 2021, S. 290–294. DOI: 10.1007/978-3-030-63416-2_125.

[14] Yau Weng Kuan, Ng Oon Ee und Lee Sze Wei. „Comparative Study of Intel R200, Kinect v2, and Primesense RGB-D Sensors Performance Outdoors". In: *IEEE Sensors Journal* 19.19 (2019), S. 8741–8750. DOI: 10.1109/JSEN.2019.2920976.

[15] S. Teruggi und F. Fassi. „HoloLens 2 Spatial Mapping capabilities in vast monumental heritage environments". In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLVI-2/W1-2022 (2022), S. 489–496. DOI: 10.5194/isprs-archives-XLVI-2-W1-2022-489-2022.

[16] Huimin Yao u. a. „Adaptive Multi-window Matching Method for Depth Sensing SoC and Its VLSI Implementation". In: *Advances in multimedia information processing - PCM 2016*. Bd. 9916. Springer, 2016, S. 428–437. DOI: 10.1007/978-3-319-48890-5_42.

Anwendung eines Machine Learning Modells in Unity - Evaluierung von Unity Barracuda

Bruno Cunha Teixeira
Hochschule Reutlingen
Reutlingen, Deutschland
Bruno.Cunha_Teixeira@student.reutlingen-university.de

Zusammenfassung

Methoden des Machine Learning (ML) werden heutzutage in verschiedenen Anwendungsgebieten und Umgebungen genutzt, um komplexe Aufgabenstellungen zu lösen. Durch Simulationen können ML-Modelle interaktiv getestet werden, um so Schwachstellen offenzulegen. Mit der Inference-Engine Unity Barracuda können in Unity Modelle ausgeführt und so eine Simulationsumgebung entwickelt werden. In dieser Arbeit wird Barracuda anhand von zwei ML-Modellen evaluiert und in einer Literaturrecherche die Vorteile von grafischen Simulationen im Bezug zu ML-Modellen ermittelt. Dabei können Vorteile im Bereich der Datenbeschaffung und dem Testen der ML-Modelle entstehen. Beispielsweise können gefährliche Testszenarien im Kontext des autonomen Fahrens virtuell durchgeführt werden, ohne dabei Verkehrsteilnehmer zu gefährden. Bei der Beschaffung von Daten können diese in der Simulation erstellt werden und so Kosten und Zeit sparen. Die zwei ML-Modelle konnten in Unity integriert und interaktiv getestet werden. In der Evaluierung wurde die Genauigkeit und die Anzahl der FPS gemessen. Im Vergleich zur Ausführung in Python verringerte sich die Genauigkeit um ca. 10% und abhängig vom Modell konnten bis zu durchschnittlich 133 FPS erzielt werden.

CCS Concepts

• **Computing methodologies** → **Machine learning**; **Modeling and simulation**.

Keywords

Unity Barracuda, Inference engine, ONNX, Deep Learning

1 Einleitung

Machine Learning (ML) stellt die Basis von intelligenten Systemen dar, die in ihrem Aufgabenfeld menschenähnliche oder gar übermenschliche kognitive Fähigkeiten erreichen [11]. Oft agieren solche Systeme in Umgebungen, in denen es viele Szenarien gibt und es damit schwierig wird solche Systeme zu testen. Abhilfe können Simulationen schaffen, in denen solche Szenarien in kontrollierter Umgebung von den ML-Modellen durchlaufen werden [4]. Ein Beispiel dafür wäre das Testen eines ML-Modells für das autonome Fahren in einem Szenarium in dem ein Wildtier auf die Straße springt. Zudem können in Simulationen Modelle interaktiv getestet werden, um so Schwächen zu identifizieren. Solche Simulationen benutzen oft Spiele-Engines um eine virtuelle Umgebung zu erschaffen [16][4]. Mit der Inference-Engine Unity Barracuda ist ML auch in der Spiele-Engine Unity vertreten. Eine Inference-Engine

Betreuer/-in Hochschule: Prof. Dr. Cristóbal Curio, Markus Rehmann, Michael Brunner, Hochschule Reutlingen, Vorname.Nachname@Reutlingen-University.de

Informatics Inside Herbst 2022
23. November 2022, Hochschule Reutlingen

ermöglicht das Ausführen von Modellen in einer fremden Umgebung. Somit könnte Unity mit Barracuda als Simulation für ML-Modelle eignen. Ziel dieser Arbeit ist die Evaluierung von Unity Barracuda. Dafür sollen zwei ML-Modelle in Unity integriert und das Testen interaktiv gestaltet werden.

1.1 Methodik

Aus den zuvor beschrieben Zielen wurden Forschungsfragen (RQ) abgeleitet, die in Tabelle 1 zu sehen sind. Für die Beantwortung der Fragen wurde eine Literaturrecherche und eine Evaluierung durchgeführt.

Tabelle 1: Forschungsfragen

ID	Forschungsfrage
RQ1	Wie kann ein ML-Modell mit Unity Barracuda ausgeführt werden?
RQ2	Welche Vorteile bietet die Ausführung von ML-Modellen in Simulationen?
RQ3	Wie wirkt sich die Ausführung von ML-Modellen in Unity auf die Performance aus?

RQ1 beschäftigt sich mit der Implementierung eines ML-Modells mithilfe von Unity Barracuda. In RQ2 sollen die Vorteile von 3D-Simulationen für ML-Modelle ermittelt werden. RQ3 befasst sich mit der Performance der Modelle in Unity Barracuda.

1.2 Aufbau

In Kapitel zwei werden die Grundlagen erklärt, die für das Verständnis der darauf folgenden Kapitel dient. Das dritte Kapitel umfasst den Stand der Wissenschaft im Bezug auf Unity Barracuda und 3D-Simulationen für ML-Modelle. Im vierten Kapitel wird die Implementierung von zwei ML-Modellen mithilfe von Barracuda in Unity beschrieben. Darauf folgt im fünften Kapitel die Evaluierung der Implementierung. Im sechsten Kapitel werden die Ergebnisse diskutiert. Den Abschluss stellt das siebte Kapitel mit dem Fazit der Arbeit dar.

2 Grundlagen

In diesem Kapitel werden die Grundlagen erläutert. Im ersten Unterkapitel wird der Aufbau von künstlichen neuronalen Netzen beschrieben, welche eine Rolle im Kapitel zur Implementierung spielen. Das zweite Unterkapitel umfasst das in Barracuda verwendete Format für ML-Modelle ONNX.

2.1 Aufbau von künstlichen neuronalen Netzen

Unter dem Begriff ML wird die Fähigkeit von Systemen beschrieben durch problembezogene Daten zu lernen, um dadurch Aufgaben zu lösen. Deep Learning (DL) stellt dabei einen beliebten Ansatz des ML dar und basiert auf künstlichen neuronalen Netzen mit mehreren sogenannten Hidden-Layers [11]. Da in dieser Arbeit zwei convolutional neural networks (CNN) verwendet werden, beschreiben die folgenden Unterkapitel den Aufbau von CNNs und feedforwards neural networks (FNN), welche ein Bestandteil von CNNs sind.

2.1.1 Feedforward neural network

Goodfellow et al. bezeichnet FNN als das Fundament von DL-Modellen [7]. In Abbildung 1 ist der vereinfachte Aufbau eines FNN zu sehen. Diese bestehen aus einer Input-Schicht, mindestens einer Hidden-Schicht und einer Output-Schicht. Die einzelnen Schichten bestehen wiederum aus sogenannten Neuronen. Als Eingabe wird ein Vektor benötigt [5]. Neuronen empfangen von der Input-Schicht oder von anderen Neuronen einen Input, welcher mit Gewichten und einem Bias zu einer gewichteten Summe verrechnet wird. Diese Summe wird an eine Aktivierungsfunktion weitergeleitet. Überschreitet die Summe den Schwellenwert der Aktivierungsfunktion wird dieser Output an das nächste Neuron oder an die Output-Schicht weitergeleitet [5].

Die Neuronen sind bei einem FNN mit den Neuronen der nächsten Schicht (ggf. mehrfach) verbunden. Wie der Name FNN impliziert, gibt es keine Rückkoppelung, d.h. der Informationsfluss verläuft von hinten nach vorne [7]. Da das Trainieren von neuronalen

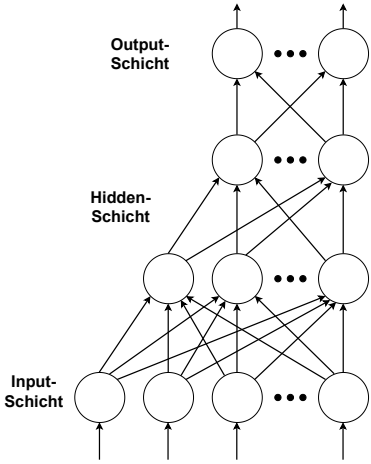

Abbildung 1: Vereinfachte Darstellung eines FNN nach Ernst et al. [5]

Netzen nicht im Fokus dieser Arbeit steht, wird dies nur kurz und oberflächlich erklärt. Im Training werden die Gewichte der Neuronen anfangs zufällig initialisiert. Danach werden die Daten an das Modell geschickt und das Ergebnis mit dem Output des Modells abgeglichen. Basierend darauf werden die Gewichte der Neuronen von vorne nach hinten geändert, so dass der Output dem richtigen Ergebnis stärker ähnelt[5].

2.1.2 Convolutional neural network

In seiner Arbeit definiert Goodfellow et al. ein CNN als ein FNN mit mindestens einer Schicht, die eine Faltungsoperation durchführt[7]. Da ein Bild aufgrund der drei Farbkanäle durch eine dreidimensionale Matrix dargestellt werden kann und CNNs eine Matrix als Eingabe akzeptieren, sind CNNs für die Bildverarbeitung attraktiv. Wie in der Abbildung 2 zu sehen ist besteht ein CNN aus unterschiedlichen Schichten.
Die Input-Schicht gibt die Größe der Matrix vor und empfängt diese. Darauf folgt eine sogenannte Convolution-Schicht in der die Faltungsoperation stattfindet. Ziel der Faltung ist das Extrahieren von Merkmalen, wie zum Beispiel Kanten und Formen. Dies wird durch einen sogenannten Kernel (zu dt. Filter) realisiert, der die Matrix durchläuft und mit seinen Werten verrechnet. Der Output des Kernels wird als Feature Map bezeichnet und ist eine Matrix, die als Input für die Pooling-Schicht dient [7]. Ziel der Pooling-Schicht ist die Verkleinerung der Matrix durch die Verwerfung von überflüssigen Informationen. Es existieren verschieden Pooling-Ansätze, wie z.B. das Max-Pooling, bei dem die Matrix unterteilt und innerhalb der Unterteilung der größte Wert als Repräsentation in der neuen verkleinerten Matrix verwendet wird. In der Regel haben CNNs mehrere Convolution- und Pooling-Schichten[7]. Die Fully Connected-Schicht (FC-Schicht) ist ein FNN und folgt nach der letzten Pooling-Schicht. Aus diesem Grund muss die Matrix in ein Vektor umgewandelt werden. Dieser Vorgang wird als Flattening bezeichnet. Am Ende der FC-Schicht wird durch die Aktivierungsfunktion die Zugehörigkeit zu einem Output in Form einer Wahrscheinlichkeit wiedergegeben [5].

2.2 ONNX

Das Open Neural Network Exchange (ONNX)[1] ist ein offenes Format für ML-Modelle, sowohl für traditionelle ML-Modelle als auch für DL. Durch den Export der Modelle in dieses Format sollen diese in verschiedenen Frameworks und Umgebungen ausführbar gemacht werden[2]. Im ONNX-Format wird die oberste Ebene als Modell bezeichnet, mit dem Zweck die Metadaten mit einem Graphen zu

[1] https://onnx.ai/ (Zuletzt aufgerufen am 05.10.2022)
[2] https://github.com/onnx/onnx (Zuletzt aufgerufen am 05.10.2022)

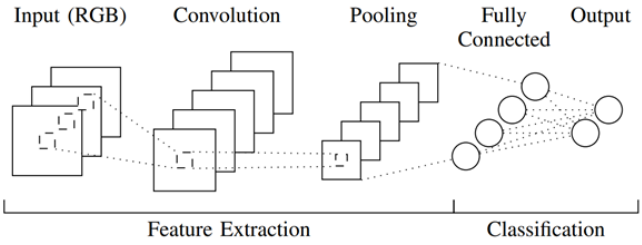

Abbildung 2: Aufbau eines CNN[10]

verknüpfen [13]. Der Graph enthält dabei alle ausführbare Elemente[3] und besteht deswegen aus Listen mit Knoten, Ein- und Ausgaben sowie Initialisierungen. Ein Knoten enthält den Namen des Operators, der aufgerufen werden soll und zusätzlich dessen Ein- und Ausgaben. Operatoren werden in ONNX in Funktionen und primitive Operatoren unterteilt. Funktionen sind dabei Operatoren, deren Berechnungen wiederum durch einen Untergraphen beschrieben werden [13]. Ein Beispiel für ein primitiven Operator ist der *Abs*-Operator[4], der einen Tensor als Input annimmt, den Betrag jedes Elements davon berechnet und in einem neuen Tensor als Output wiedergibt.

3 State of the Art

In diesem Kapitel wird der Stand der Wissenschaft im Bezug zu Unity Barracuda und dem Ausführen von ML-Modellen in Simulationen wiedergegeben.

3.1 Unity Barracuda

Neben der Entwicklung von Videospielen, wird Unity auch für Visualisierungen und interaktive 3D-Anwendungen verwendet. Dies spiegelt sich auch in der Verwendung von Barracuda wieder. Im Folgenden werden drei Arbeiten vorgestellt in denen Unity Barracuda verwendet wird.

Kaisar et al. entwickelten 2022 ein generationenübergreifendes Fitnesspiel mit Unity [12]. Senioren und junge Menschen müssen in den drei unterschiedlichen Level zusammenarbeiten und über die Körperbewegung die Spielfigur steuern. Junge Menschen steuern die Spielfigur, welche als Vogel dargestellt wird über das Balancieren auf einer Platte mit darunterliegenden Halbkugeln (NEO-NEO Balance Ball). Senioren hingegen steuern den Vogel über ihre Armbewegungen, die über eine Webcam aufgenommen wird. Die Autoren nutzen Unity Barracuda für die Ausführung eines nicht näher beschriebenen ML-Modells, um die Pose auf dem Bild zu erkennen. Abhängig von der erkannten Pose wurde die Spielfigur vertikal bewegt. Für die horizontale Steuerung, war die junge Person mit dem Balancieren auf der Platte verantwortlich [12].

Neben der Nutzung für die Entwicklung von Videospielen kann Unity auch für die Visualisierung verwendet werden. Nagasaka et al. beschreiben in ihrer Arbeit aus dem Jahr 2021 ein System zur Visualisierung von DL-Model -len in der virtuellen Realität (VR) [14]. Ziel der Visualisierung ist das Verhalten der Modelle nachvollziehbarer für die Nutzer zu machen. Das System lädt ein ONNX-Modell mithilfe von Barracuda in die Scene und basierend auf den Schichten werden zwei- und dreidimensionale Visualisierungen erzeugt. Mit diesen Visualisierungen kann der Nutzer mit einer VR-Brille interagieren. Neben den verschiedenen Outputs der jeweiligen

[3] https://github.com/onnx/onnx/blob/main/docs/IR.md (Zuletzt aufgerufen am 05.10.2022)
[4] https://github.com/onnx/onnx/blob/main/docs/Operators.md#Abs (Zuletzt aufgerufen am 05.10.2022)

Schicht zeigen die Visualisierungen auch Performance-Metriken wie loss und Genauigkeit des Modells an. In einem kleinen Nutzertest mit fünf Personen, die Erfahrung in DL besitzen, konnte positives Feedback gesammelt werden [14].
Castanyer et al. beschäftigen sich in ihrer Arbeit mit der Integrierung von CNNs in mobilen Anwendungen [3]. Dabei konzentrieren sich Castanyer et al. auf die Herausforderungen bei der Integration und dem Tradeoff zwischen Genauigkeit und Komplexität der Modelle. Besonders bei mobilen Endgeräten ist die Performance ein wichtiger Aspekt, da eine niedrigere Rechenleistung zur Verfügung steht und die Energieversorgung durch ein Akku beschränkt ist. In ihre Studie trainierten sie ein MobileNetV2 und ein ResNet34 für die Klassifizierung von Verkehrsschildern und integrierten diese mithilfe von Barracuda in eine Unity-App. Als Benchmark-Datensatz diente der bereits in der Literatur oft verwendete GTSRB-Datensatz [17]. Insgesamt konnten die Autoren fünf verschiedene Herausforderungen in ihrer Studie ermitteln [3]. Die erste Herausforderung ist der frühe Entwicklungsstand der Frameworks. Während der Entwicklung fiel auf, dass einige Operatoren nicht durch ONNX oder Barracuda unterstützt werden. Aufgrund der Weiterentwicklung der Frameworks entschärft sich diese Herausforderung im Laufe der Zeit. Beispielsweise wird Barracuda weiterentwickelt und bittet in ihrer Dokumentation[5] um die Erstellung eines Tickets auf GitHub, falls ein benötigter Operator nicht unterstützt wird. Die zweite Herausforderung ist die Abhängigkeit gegenüber den eingespeisten Daten. In der Studie schwankten die Ergebnisse der Anwendung stark bei realen Daten. Als weitere Herausforderung wurde die Zuverlässigkeit der Daten identifiziert, welche im Zusammenhang mit der vorherigen Herausforderung steht. Um eine gewisse Zuverlässigkeit der Daten zu gewährleisten, implementierten die Autoren die Augmentation für reale Daten. Die Kompatibilität bzw. die Abhängigkeit zwischen den verwendeten Frameworks gestaltet sich ebenfalls als eine Herausforderung. So muss das exportierte Modell mit der Inference-Engine kompatibel sein und das Preprocessing der Eingabedaten muss in verschiedenen Umgebungen identische Ergebnisse liefern. Die letzte ermittelte Herausforderung stellt die Perfomance dar. Komplexere Modelle die genauere Ergebnisse liefern benötigten mehr Speicherplatz, was beim mobilen Endgerät nicht zu vernachlässigen ist [3].

3.2 3D-Simulationen für ML-Modelle

In der Bildverarbeitung können 3D-Simulationen für die Erstellung von Daten [1] und für das Testen von Modellen verwendet werden [4]. In der Simulation können gefährliche oder selten auftretende Szenarien durchlaufen werden. Beispielsweise können so Modelle für das autonome Fahren trainiert oder getestet werden, ohne dabei andere reale Verkehrsteilnehmer zu gefährden. Weitere positive Aspekte finden sich in der Organisation. In der Simulation werden keine Genehmigungen benötigt, die beispielsweise bei einem Flug einer Drohne ab einer bestimmten Höhe anfallen. Zudem kann die Erstellung von Daten in der Simulation zu einer Kosten- und Zeitersparnis führen [1]. Im folgenden werden die zwei Simulationsumgebungen CARLA und AirSim untersucht.
Im Jahr 2017 veröffentlichten Dosovitskiy et al. in ihrer Arbeit einen freien Fahrsimulator für die Erprobung und Forschung von Modellen für das autonome Fahren namens Car Learning to Act (CARLA) [4]. Dafür wurde

[5] https://docs.unity3d.com/Packages/com.unity.barracuda@3.0/manual/SupportedOperators.html (Zuletzt aufgerufen am 05.10.2022)

in der Unreal Engine zwei Städte mit verschiedenen Fahrzeugen, Gebäuden, Passanten und weiteren Objekten erstellt (siehe Abbildung 3). Das Verhalten der Fahrzeuge und der Passanten wird simuliert und beinhaltet auch den Spurwechsel oder das Überqueren einer Straße durch Passanten. Dies ist besonders für die Evaluierung der Modelle wichtig, da in der Simulation gefährliche Situationen sicher durchlaufen werden können [4]. Die Simulation ist nach dem Server-Client Prinzip aufgebaut. Auf dem Server läuft eine Instanz der Simulation und durch eine Client-API können Modelle mit der Simulation interagieren. Dabei schickt der Server Sensordaten vom Auto (z. B. Kamerabilder) an den Client bzw. an das Modell und erhält dann die Steuerbefehle für das Auto vom Client. CARLA kann dabei nicht nur für die Ausführung der Modelle benutzt werden, sondern auch für das Training und Optimieren verwendet werden. Dafür trainierten die Autoren beispielsweise einen Agenten zum Steuern eines Autos auf Basis von Reinforcement Learning [4].

Abbildung 3: Ausschnitt vom Simulator CARLA mit unterschiedlichem Wetter

Eine weitere Simulation ist AirSim, die 2017 von Shah et al. veröffentlicht worden ist [16]. Ähnlich wie bei CARLA wurde eine Simulation mit der Unreal Engine entwickelt, um die Forschung im Bereich autonome Fahrzeuge zu unterstützen. Anfangs beschränkte sich AirSim auf autonome Drohnen, mittlerweile werden auch Autos und andere Fahrzeuge unterstützt[6]. Über eine API können Modelle mit der Simulation interagieren und so beispielsweise eine Drohne steuern. In einem Experiment verglichen Shah et al. den Flug einer Drohne in AirSim und in der realen Welt. Dafür flog die Drohne zuerst in der Simulation in Kreisen und Vierecken und dann in der realen Welt. Um die Vergleichbarkeit zu gewährleisten, wurden die Steuerbefehle der Simulation für den Flug in der Realität verwendet. Der Unterschied zwischen Realität und Simulation lag beim Kreis bei 1,47 Meter und beim Viereck bei 0,65 Metern. Als Ursache vermuten die Autoren unter anderem leichte Winde und Fehler beim Approximieren des Drohnenmodells. Ein praktischer Anwendungsfall ist die Zählung und Lokalisierung von Wildtieren. Auf Basis von AirSim wurde von Bondi et al. AirSim-W [1] entwickelt, um Wildtiere und Wilderer in der afrikanischen Savanne zu erkennen. Durch die in der Simulation erstellten Daten konnten die Entwickler im Vergleich zu echten Daten die Kosten von 8000$ auf 5000$ und den zeitlichen Aufwand von 800 Stunden auf 200 Stunden reduzieren [1].

4 Implementierung

In diesem Kapitel wird die Implementierung und Ausführung der Modelle in Unity erläutert. Zu Beginn werden die zwei verwendeten CNNs kurz beschrieben. Darauf aufbauend wird die Verwendung der Modelle in Unity mithilfe von Unity Barracuda beschrieben.

4.1 Maskenmodell

Die Implementierung des Maskenmodells in Unity soll Nutzern die Möglichkeit bieten,

[6] https://microsoft.github.io/AirSim (Zuletzt aufgerufen am 05.10.2022)

verschiedene Szenarien zu simulieren und so das Verhalten des Modells zu untersuchen. Das Modell klassifiziert Bilder von Gesichtern in die Kategorien *Person mit Mund-Nasen-Schutz (MNS)* und *Personen ohne MNS*. In der Unity-Scene kann eine virtuelle Maske im Bild bewegt werden, um so verschiedene Szenarien zu erstellen.
Das verwendete CNN wurde zuvor im Rahmen einer studentischen Arbeit erstellt und basiert auf der MobileNetV2-Architektur[15]. Für die Implementierung wurde hauptsächlich Tensorflow und Keras verwendet. Dabei wurde die FC-Schicht ersetzt und mit einem Datensatz neu angelernt. Als Datensatz wurden ca. 11000 Bilder verwendet, die aus dem Datensatz von Cabani et al. [2] und einem öffentlichen GitHub-Repository[7] stammen. Zum Trainieren des CNNs wurden 80% der Bilder vom Datensatz verwendet. Die restlichen 20% wurden für die Validierung verwendet. Auf dem Validierungsdatensatz konnte das Modell ein F1-Score von 97% erreichen und zeigt dabei keine Anzeichen von Overfitting an. Das Modell akzeptiert als Input ein mindestens 224x224 großes Bild mit einer Normalisierung der Farbwerte zwischen -1 und 1. Das Output des Modells besteht aus einer 1x2 Matrix, welche die Wahrscheinlichkeit der zwei Klassen beinhaltet. Für die Konvertierung des Modells in das ONNX-Format wurde das Tool *tf2onnx*[8] verwendet.

4.2 Tiermodell

Das zweite Modell ist ebenfalls ein CNN und klassifiziert Bilder in neun verschiedene Tierarten. Das Modell wurde mit Pytorch erstellt und basiert auf der ResNet50-Architektur[9]. Die Auswahl der Architektur und Technologie wurde aus folgenden Gründen getroffen. Pytorch ist neben Tensorflow ein beliebtes Framework für ML und daher sollte der Umgang von Unity Barracuda mit Pytroch-Modellen untersucht werden. Im Vergleich zu MobileNetV2 ist ResNet50 nicht auf Performance ausgelegt und soll hingegen eine höhere Genauigkeit besitzen. Dadurch soll der Umgang von Unity Barracuda mit komplexeren Modellen beobachtet werden. Außerdem wird die Architektur nicht offiziell von Unity Barracuda unterstützt. Nichtsdestotrotz kann das Modell in das ONNX-Format exportiert werden und in Unity Barracuda ausgeführt werden. Hierbei soll untersucht werden, wie die Inference Engine mit nicht unterstützten Modellen agiert.
Ähnlich wie bei dem Maskenmodell wurde die FC-Schicht des vortrainierten ResNet50 ersetzt und mit einem neuen Datensatz angelernt. Aus dem Datensatz Caltech101 [6] wurden dafür die Bilder für die benötigten Klassen extrahiert. Insgesamt kann das Modell ein Bild in folgende Tiere klassifizieren: Leopard, Schmetterling, Krokodil, Elefant, Flamingo, Igel, Kangaroo, Okapi und Nashorn. Dabei wurden 75% der Daten zum Trainieren und jeweils 12,5% für die Validierung und das Testen verwendet. Das Modell erreicht auf dem Testdatensatz eine Genauigkeit von 96,3%. Als Input wird ein mindestens 224x244 großes Bild mit einer Normalisierung der Farbwerte zwischen 0 und 1 benötigt. Der Output besteht aus einer 1x9 Matrix, welche die Wahrscheinlichkeiten der Klasse beinhaltet.

4.3 Implementierung in Unity

Für die Implementierung wurde die Unity-Version 2021.2.6f1 und die Barracuda-Version 3.0 verwendet. Für die zwei zuvor beschriebenen Modelle wurde jeweils eine Unity-Scene erstellt, die bis auf das Pre- und Postprocessing nahezu identisch sind. Abbildung 4 zeigt den Ablauf des Programms anhand eines Aktivitätsdiagramms. Im ersten Schritt wird die

[7] https://github.com/chandrikadeb7/Face-Mask-Detection (Zuletzt aufgerufen am 05.10.2022)
[8] https://github.com/onnx/tensorflow-onnx (Zuletzt aufgerufen am 05.10.2022)

Schnittstelle *IWorker* von Barracuda initialisiert und das Modell geladen. Dabei bietet Barracuda verschiedene Backends an, die entweder die CPU oder GPU für die Ausführung verwenden[9].

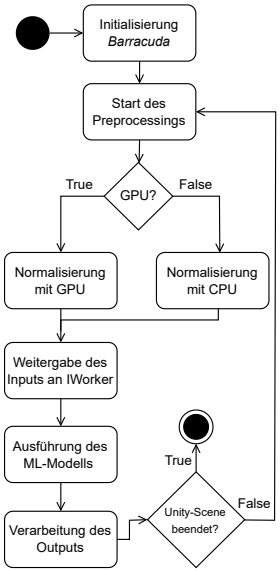

Abbildung 4: Aktivitätsdiagram der Implementierung

Im zweiten Schritt werden die Daten für das Modell vorbereitet. Als Input für die CNNs wird das Bild der Kamera benutzt. Dafür wird das Bild auf die Größe 224x224 skaliert und normalisiert. Für die Normalisierung stehen zwei verschiedene Optionen zur Verfügung. Die erste Option nutzt die CPU und parallele Threads zur Normalisierung. Bei der zweiten Option wird ein Shader für die Berechnung verwendet. Hauptsächlich werden Shader für die Berechnung von Reflexionen und anderen grafischen Effekten verwendet und werden dabei auf der GPU ausgeführt. Unity bietet mit dem *Compute shader* die Möglichkeit Quellcode auf der GPU außerhalb der normalen Rendering Pipeline auszuführen[10]. Das normalisierte Bild wird in einem *Tensor*

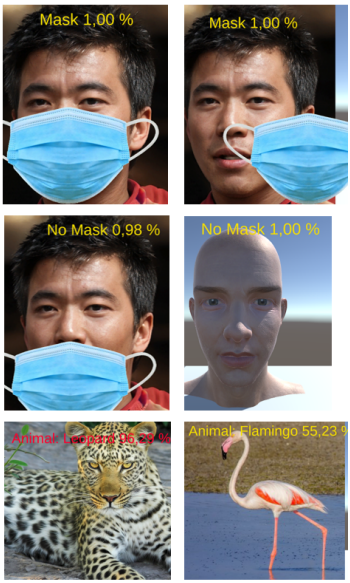

Abbildung 5: Anwendung der ML-Modelle in Unity

abgespeichert, der als Input dient. Die Klasse *Tensor* ist in Barracuda integiert und ähnelt einem mehrdimensionalen Array, welches als In- und Output verwendet wird. Der Input wird als Übergabeparamter an die *Execute*-Funktion der *IWorker*-Schnittstelle übergeben. Diese Funktion führt das ML-Modell mit dem Input aus. Als Output wird ein *Tensor* zurückgegeben und für die Darstellung in der Scene weiterverarbeitet. Wird die Unity-Scene nicht geschlossen, wiederholt sich der ganze Prozess ab dem zweiten Schritt und das nächste Bild der Kamera wird vorbereitet.
Die Abbildung 5 zeigt die Klassifizierung der ML-Modelle in Unity. In der Unity-Scene kann das Bild bzw. das Gesicht und zusätzlich beim Maskenmodell die Position des MNS verändert werden. Dadurch können verschiedene

[9] https://docs.unity3d.com/Packages/com.unity.barracuda@3.0/manual/Worker.html (Zuletzt aufgerufen am 05.10.2022)

[10] https://docs.unity3d.com/Manual/class-ComputeShader.html (Zuletzt aufgerufen am 05.10.2022)

Szenarien erstellt werden und so beispielsweise das Verhalten des Modells analysiert werden. Befindet sich die Maske seitlich vom Gesicht, erkennt das Modell die Person fälschlicherweise als Person mit MNS.

5 Evaluierung

In der Evaluierung wird die Performance der beiden Modelle in der Barracuda-Umgebung gemessen. Dafür wurde die Genauigkeit der Modelle in verschiedenen Umgebungen ermittelt und die Bildwechselfrequenz (FPS) in Unity erfasst. Für die Evaluierung der Genauigkeit wurden die Modelle in verschiedenen Formaten und Umgebungen ausgeführt. Die dafür verwendeten Datensätze waren die Testdatensätze der jeweiligen Modelle. Die Metrik Genauigkeit ist allerdings anfällig gegenüber unausgewogenen Datensätzen [8]. Bei einer starken Überrepräsentation einer Klasse könnten die Ergebnisse verzerrt werden, da möglicherweise das Modell bei einer bestimmten Klasse genauer oder ungenauer klassifiziert. Aus diesem Grund wurden die Testdatensätze gleichmäßig aufgestellt. Neben der Ausführung der Modelle in ihrem Framework ohne die Exportierung in das ONNX-Format, wurde die ONNX-Varianten auch in der ONNX-Runtime (ONNX-RT) und in Barracuda getestet. Die Ereignisse der Evaluierung sind in der Tabelle 2 zu sehen.

Tabelle 2: Genauigkeit der Modelle in verschiedenen Umgebungen

	Maskenmodell	Tiermodell
Tensorflow/ Pytorch	97%	90,01%
ONNX-RT	90,35%	89,78%
Barracuda	87%	81,03%

Bei der Evaluierung der FPS wurden die Modelle in Unity mit zwei verschiedenen Backends ausgeführt und die durchschnittlichen FPS ermittelt. Das erste Backend *CSharpBurst* wird auf der CPU ausgeführt und wird von Barracuda zu den effizienteren CPU-Backends gezählt[11]. Das zweite wird auf der GPU ausgeführt, trägt die Bezeichnung *ComputePrecompiled* und wird ebenfalls von Barracuda als effizienteres GPU-Backend aufgeführt. Neben den unterschiedlichen Backends wurde auch mit unterschiedlichen Preprocessing-Arten evaluiert, die in Kapitel 4.3 erläutert wurden. Die aufgerundeten Ergebnisse sind in Tabelle 3 zu sehen.

Tabelle 3: Durchschnittliche FPS

Backend/ Preprocess	Maskenmodell	Tiermodell
CPU/CPU	31 FPS	10 FPS
CPU/GPU	34 FPS	11 FPS
GPU/CPU	102 FPS	59 FPS
GPU/GPU	133 FPS	81 FPS

6 Diskussion

Die Implementierung zeigt, dass ML-Modelle in Unity nicht nur ausgeführt, sondern auch interaktiv getestet werden können. Am Beispiel des Maskenmodells konnte in Unity das Verhalten des Modells bei unterschiedlichen Positionen des MNS getestet und der Übergang zwischen den zwei Klassifikationsklassen beobachtet werden. Mithilfe von Barracuda könnte in Unity eine Simulationsumgebung, ähnlich wie bei AirSim oder CARLA geschaffen werden, mit dem Unterschied, dass die Modelle direkt in Unity ausgeführt werden könnten. Dadurch wären Vorteile bei der Performance denkbar, da im Vergleich zu CARLA keine Client-API nötig wäre. Jedoch müsste dies in einer weitere Studie belegt werden. Neben dem Einsatz in der Simulation kann Unity Barracuda auch in Videospielen oder in mobilen Applikationen verwendet werden. Beispielsweise könnte in einer mobilen Anwendung die Kamera-App dadurch

[11] https://docs.unity3d.com/Packages/com.unity.barracuda@3.0/manual/Worker.html (Zuletzt aufgerufen am 05.10.2022)

erweitert werden, dass fremde Menschen im Videostream erkannt und sofort unkenntlich gemacht werden, um in einem Livestream den Datenschutz zu erhöhen.
Die Genauigkeit der in Barracuda ausgeführte Modelle sank im Vergleich zur Ausführung in Tensorflow bzw. Pytorch. Der Unterschied beträgt beim Maskenmodell 10% und beim Tiermodell 8,98%. Wie bereits erwähnt, basiert das Maskenmodell auf der MobileNetV2 Architektur und wird offiziell von Unity Barracuda unterstützt. Das auf ResNet50 basierte Tiermodell hingegen wird nicht offiziell unterstützt. Es ist also nicht auszuschließen, dass bestimmte Operatoren deswegen gar nicht oder nur teilweise korrekt ausgeführt werden. Dies könnte eine Erklärung für den größeren Unterschied zwischen der Ausführung in der ONNX-RT und der in Barracuda sein. Beim Export des Maskenmodells in das ONNX-Format sank die Genauigkeit um ca. 7%. In diesem Fall wäre eine Optimierung des Exports sinnvoll, da der Unterschied zwischen ONNX-RT und Barracuda vergleichsweise klein ist. Die Wahl des Backends hatte keine Auswirkung auf die Genauigkeit.
Bei der Messung der FPS hatte die Wahl des Prozessors einen großen Einfluss auf die Performance. Wurde das Modell und das Preprocessing auf der GPU ausgeführt so konnten beim Maskenmodell eine ca. vier mal höhere FPS erzielt werden. Beim Tiermodell sogar eine ca. acht mal höhere FPS. Diese Zahlen sind jedoch von der verwendeten Hardware abhängig und könnten bei einer leistungsstärkeren CPU geringer ausfallen. Interessant ist jedoch, dass beim GPU-Backend die Wahl des Preprocessings eine größere Wirkung hatte. Beim Vergleich der beiden Modelle erreicht das Maskenmodell eine höhere FPS. Grund dafür ist, wie in Kapitel 4 bereits erwähnt die Architektur der Modelle. Das Maskenmodell basiert auf der MobileNetV2-Architektur, die auf eine schnelle Ausführung optimiert ist [15]. Die in Tabelle 3 erreichten Zahlen wurden in einer minimalistisch gehaltenen Unity-Scene erreicht. In einer echten Anwendung würde die Anzahl der FPS niedriger ausfallen, da Ressourcen für anderen Berechnungen benötigt werden. Dies könnte jedoch entschärft werden, indem das Modell nicht mehr in jedem Frame ausgeführt wird.

7 Fazit

Diese Arbeit befasst sich mit der Inference-Engine Unity Barracuda und der Ausführung von ML-Modellen in grafischen Simulationen. Ziel war es ein ML-Modell in Unity auszuführen, die Performance zu untersuchen und die Vorteile von Simulationen für ML-Modelle zu erfassen. Neben einer Literaturrecherche wurden auch zwei ML-Modelle mithilfe von Barracuda in Unity integriert und dann evaluiert. Das erste Modell erkennt, ob Personen eine Maske tragen und das zweite Modell klassifiziert Bilder in neun verschiedene Tierarten. In der Unity-Scene des ersten Modells kann eine virtuelle Maske über Gesichter gelegt und bewegt werden, um so die Ergebnisse des Modells zu untersuchen. Dadurch konnten Grenzfälle simuliert werden und das Verhalten der Modelle dabei beobachtet werden. Dieser Vorteil wurde auch in der Literaturrecherche ermittelt. Neben der Ausführung der Modelle können die Simulationen auch Daten für das Training generieren. Dadurch können finanzielle Mittel und Zeit gespart werden. Außerdem können gefährliche Situationen simuliert werden. In einer Evaluierung wurde die Implementierung der beiden Modelle in Unity im Hinblick auf die Genauigkeit und der Performance der FPS untersucht. Die Genauigkeit der Modell sank insgesamt um ca. 10% im Vergleich zur Ausführung in Python. In der Evaluierung konnte das leichtgewichtige Maskenmodell bis zu durchschnittlich 133 FPS erzielen, jedoch ist die FPS stark davon abhängig, ob eine GPU oder CPU für die Ausführung verwendet wird.

Literatur

[1] Elizabeth Bondi u. a. „AirSim-W: A Simulation Environment for Wildlife Conservation with UAVs“. In: *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*. COMPASS '18. Menlo Park und San Jose, CA, USA: Association for Computing Machinery, 2018. DOI: 10.1145/3209811.3209880.

[2] Adnane Cabani u. a. „MaskedFace-Net – A dataset of correctly/incorrectly masked face images in the context of COVID-19“. In: *Smart Health* 19 (2021). DOI: https://doi.org/10.1016/j.smhl.2020.100144.

[3] Roger Creus Castanyer, Silverio Martínez-Fernández und Xavier Franch. „Integration of Convolutional Neural Networks in Mobile Applications“. In: *2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)*. 2021, S. 27–34. DOI: 10.1109/WAIN52551.2021.00010.

[4] Alexey Dosovitskiy u. a. „CARLA: An Open Urban Driving Simulator“. In: *Proceedings of the 1st Annual Conference on Robot Learning*. 2017, S. 1–16.

[5] Hartmut Ernst, Jochen Schmidt und Gerd Beneken. „Maschinelles Lernen: Deep Learning mit neuronalen Netzen“. In: *Grundkurs Informatik: Grundlagen und Konzepte für die erfolgreiche IT-Praxis – Eine umfassende, praxisorientierte Einführung*. Wiesbaden: Springer Fachmedien Wiesbaden, 2020, S. 801–833. ISBN: 978-3-658-30331-0. DOI: 10.1007/978-3-658-30331-0_18.

[6] Li Fei-Fei, R. Fergus und P. Perona. „Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories“. In: *2004 Conference on Computer Vision and Pattern Recognition Workshop*. 2004, S. 178–178. DOI: 10.1109/CVPR.2004.383.

[7] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[8] Margherita Grandini, Enrico Bagli und Giorgio Visani. *Metrics for Multi-Class Classification: an Overview*. 2020. DOI: 10.48550/ARXIV.2008.05756.

[9] Kaiming He u. a. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385.

[10] Lars Hertel u. a. „Deep Convolutional Neural Networks as Generic Feature Extractors“. In: *CoRR* abs/1710.02286 (2017). DOI: 10.48550/ARXIV.1710.02286.

[11] Christian Janiesch, Patrick Zschech und Kai Heinrich. „Machine learning and deep learning“. In: *Electronic Markets* 31.3 (2021), S. 685–695. ISSN: 1422-8890. DOI: 10.1007/s12525-021-00475-2.

[12] Emiran Kaisar u. a. „Designing Social Exergame to Enhance Intergenerational Interaction and Exercise“. In: *HCI in Games*. Hrsg. von Xiaowen Fang. Cham: Springer International Publishing, 2022, S. 530–541.

[13] Tung D. Le u. a. „Compiling ONNX Neural Network Models Using MLIR“. In: *CoRR* abs/2008.08272 (2020).

[14] Hikaru Nagasaka und Motoya Izuhara. „Interactive Visualization of Deep Learning Models in an Immersive Environment“. In: 2021. DOI: 10.1145/3489849.3489956.

[15] Mark Sandler u. a. „MobileNetV2: Inverted Residuals and Linear Bottlenecks“. In: (2018). DOI: 10.48550/ARXIV.1801.04381.

[16] Shital Shah u. a. „AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles“. In: *CoRR* abs/1705.05065 (2017). arXiv: 1705.05065.

[17] J. Stallkamp u. a. „Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition“. In: *Neural Networks* 32 (2012). Selected Papers from IJCNN 2011, S. 323–332. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2012.02.016.

Geometric Deep Learning: Eine gruppenbasierte Sichtweise

Tim Mangliers
Hochschule Reutlingen
Reutlingen, Deutschland
Tim.Mangliers@Student.Reutlingen-University.de

Zusammenfassung

In der vorliegenden Arbeit wird eine gruppenbasierte Sichtweise auf das Geometric Deep Learning gegeben. Das Geometric Deep Learning besitzt viele praxisrelevante Anwendungen in den Bereichen der Computergrafik und der Netzwerkanalyse. Es erfolgt eine Konstruktion von Ansatzfunktionen mit bestimmten geometrischen Eigenschaften. Die Konstruktion erfolgt mithilfe des Geometric Deep Learning Blueprints. Es wird das theoretische Minimum bereitgestellt um diesen adäquat zu motivieren und zu verstehen. Zur Veranschaulichung wird die Modellarchitektur eines Convolutional Neural Networks eingeordnet. Für die konstruierten Ansatzfunktionen wird zudem eine Fehlerabschätzung im Rahmen der empirischen Risikominimierung durchgeführt.

CCS Concepts

• **Computing methodologies** → **Supervised learning**; **Neural networks**.

Keywords

Machine Learning, Supervised Learning, Geometric Deep Learning, Convolutional Neural Networks

1 Einleitung

In den letzten Jahren hat sich im Bereich des Machine Learning insbesondere das Deep Learning (DL) durch erfolgreiche praktische Anwendungen basierend auf Bild-, Video- und Audiodaten hervorgetan [3]. Beispielsweise wird DL in der Windows Spracherkennung verwendet um Audio- und Videodateien mithilfe der menschlichen Stimme und Sprache zu suchen oder bei der Google-Suchmaschine für die Funktion des Bilder-Suchdienstes eingesetzt [8, 18]. Bild-, Video- und Audiodaten sind Beispiele für euklidische Daten.

Ebenfalls haben sich in den letzten Jahren Problemstellungen im DL ergeben die auf Daten ohne zugrunde liegender euklidischer Struktur beruhen. Wird DL auf Daten ohne euklidischer Struktur betrieben spricht man von Geometric Deep Learning (GDL) [3, 4]. Die Anwendungsgebiete des GDLs sind vielfältig. Beispiele sind im Kontext der sozialen Netzwerke zu finden. Das soziale Netzwerk kann mithilfe von Graphen und auf deren Knoten definierten Funktionen modelliert werden [3]. Die Knoten werden mit Mitgliedern des sozialen Netzwerkes identifiziert. Die Kanten geben eine Bekanntschaft der Mitglieder an. Die Funktion auf den Knoten beschreibt das Alter der Mitglieder. Mithilfe von GDL kann dann das Alter eines Mitglieds anhand seiner Bekanntschaften geschätzt werden. Auch können mit GDL Netzwerkanalysen ausgeführt werden. Mitglieder mit gemeinsamen Interessen können detektieren werden. Ferner können Empfehlungsdienstverfahren umgesetzt werden. Dabei

Betreuer Hochschule Bernhard Mößner
Hochschule Reutlingen
Bernhard.Moessner@Reutlingen-University.de
Informatics Inside Herbst 2022
23. November 2022, Hochschule Reutlingen

werden Objekte vorgeschlagen die Mitgliedern mit einer hohen Wahrscheinlichkeit gefallen. Überdies finden Anwendungen in der Teilchenphysik und der Chemie statt, denn Partikelsysteme oder Moleküle können als Graphen modelliert werden. Hier existieren Anwendungen im Bereich des Wirkstoffdesigns. In diesen Anwendungen werden physikalische, chemische sowie biologische Eigenschaften anhand der betrachteten Molekülstruktur vorhergesagt [3]. Eine weitere Anwendung ist die Verkehrsprognose. Es werden Straßennetze als Graphen modelliert und die Ankunftszeit bezüglich einer bestimmten Route geschätzt [4].
Im Bereich der Computergrafik und der Computer Vision treten Begrenzungsflächen von dreidimensionalen Objekten auf [15]. Diese Begrenzungsflächen werden mithilfe Riemannscher Mannigfaltigkeiten modelliert [3]. Eine Riemannsche Mannigfaltigkeit ist eine mathematische Beschreibung einer Oberfläche. Auch im Bereich der Virtual Reality, der Augmented Reality und beim Motion Capture dienen Riemannsche Mannigfaltigkeiten als Datengrundlage [4, 5]. Beispielsweise werden beim Motion Capture die erfassten Bewegungen eines Akteurs auf ein dreidimensionales Modell transferiert. Weitere nichteuklidische Daten in der Computergrafik sind Punktwolken oder Polygonnetze.
In der Computergrafik wird GDL bei Klassifikationsproblemen eingesetzt. Zum Beispiel wenn zwischen einem dreidimensionalen Modell eines Hundes und eines Menschen unterschieden werden soll. Die prominentesten Vertreter von Daten mit nichteuklidischer Struktur sind Graphdaten und Mannigfaltigkeitsdaten.

1.1 Motivation und Ziele

Wie bereits ausgeführt treten in den Anwendungsgebieten des GDL verschiedene Arten von Daten auf. Euklidische Daten besitzen eine bewegungssymmetrische Struktur. Bewegungen sind zum Beispiel Translationen oder Rotationen. Nichteuklidische Daten hingegen besitzen andere Symmetrien. Nichteuklidische Daten besitzen keine Translationssymmetrie [7]. Symmetrien führen zu Invarianzen, also zu Funktionen die sich unter einer Symmetrie nicht ändern. Im Kontext des DL wird durch die Struktur der Daten der Hypothesenraum festgelegt. Das heißt der Raum, der die Funktionen enthält die die Daten erklären können. Der Hypothesenraum für Daten mit bestimmter Symmetriestruktur besitzt durch die Daten vorgegebene symmetrische Eigenschaften. Beispielsweise kann eine Eigenschaft der Hypothesen bei der Einzelobjektklassifizierung auf zweidimensionalen Bildern eine Rotationsinvarianz sein. Dieses Beispiel ist in Abbildung 1 veranschaulicht. Beim DL werden Hypothesenräume mit

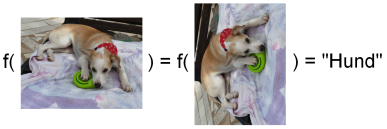

Abbildung 1: Rotationsinvarianz um 90° bei der Einzelobjektklassifizierung

Ansatzfunktionen approximiert. Nun stellt sich die Frage ob Ansatzfunktionen gefunden werden können die genau die symmetrischen Eigenschaften besitzen wie der durch die Daten vorgegebene Hypothesenraum. Da die symmetrischen Eigenschaften unterschiedlich ausfallen können, wäre ein allgemeines Konstruktionsprinzip der Ansatzfunktionen wünschenswert, das Fähig ist Ansatzfunktionen mit beliebigen symmetrischen Eigenschaften zu erzeugen.
Beim Lern- bzw. Approximationsprozess im DL kommt es unweigerlich zu einem Fehler zwischen der approximierten Hypothese und der eigentlich gesuchten Funktion. Wenn eine Konstruktion angegeben werden kann, die Ansatzfunktionen mit bestimmten Eigenschaften erzeugt, stellt sich die Frage

wie sich die Konstruktion auf den Fehler auswirkt.

1.2 Forschungsfragen

Die oben stehenden Fragestellungen sind mit den folgenden zwei Forschungsfragen konkretisiert:

RQ1 Es gibt Hypothesenräume mit bestimmten Eigenschaften. Lassen sich Ansatzfunktionen erzeugen die genau diese Eigenschaften besitzen?

RQ2 Können aus der Konstruktion von Ansatzfunktionen mit vorgegebenen Eigenschaften Rückschlüsse auf den Fehler beim Lernprozess gezogen werden?

Das Ziel der Arbeit ist die Beantwortung der oben stehenden Fragestellungen und den daraus spezifizierten zwei Forschungsfragen.

1.3 Methodik

Die Beantwortung der zwei Forschungsfragen erfolgt mithilfe einer systematischen Literaturrecherche. Zusätzlich wurde eine unsystematische Literaturrecherche mit dem Schneeballsystem bezüglich des Artikels [4] von Bronstein et al. durchgeführt. Hier wurde der Artikel [3] von Bronstein et al. gefunden und als relevant eingestuft. Chronologisch erfolgte erst die unsystematische Literaturrecherche und anschließend die systematische Literaturrecherche mit der unten erläuterten Suchstrategie.

1.3.1 Suchstrategie

Der Suchterm *"Geometric Deep Learning" AND "Survey"* ergab in den Datenbanken ACM Digital Library, arXiv, IEE Xplore, ScienceDirect, SpringerLink und dem Web of Science im Zeitraum von 2017 bis 2022 insgesamt 199 Treffer. Die genaue Anzahl der Ergebnisse pro Datenbank ist in Tabelle 1 dargestellt. Der Zeitraum ergibt sich daher, dass der Begriff GDL erstmals im Jahr 2017 im Artikel [3] von Bronstein et al. verwendet wird. Aufgrund der ausreichend gefundenen Literatur wurde auf synonyme Suchterme mit dem OR Operator wie Manifold Learning oder Graph Learning verzichtet.

Tabelle 1: Anzahl der Literaturergebnisse pro Datenbank

Datenbank	Ergebnisse
ACM Digital Library	31
arXiv	16
IEEE Xplore	2
ScienceDirect	65
SpringerLink	28
Web of Science	57

1.3.2 Literaturauswahl

Neben dem Einschlusskriterium das die Artikel im Zeitraum von 2017 bis 2022 publiziert wurden, wurden die Ergebnisse weiter danach gefiltert das sie in deutscher oder englischer Sprache vorliegen. Anschließend wurden Duplikate entfernt. Daraufhin wurden die Titel überprüft und es verblieben elf Artikel. Diese elf Artikel wurden dann weiter auf ihr Abstract hin überprüft und für die vorliegende Arbeit als relevant eingestuft. Zusammen mit den Artikeln [3, 4] von Bronstein et al. werden demnach insgesamt 13 Artikel als relevant befunden. Diese 13 Artikel werden für die Ergebnisse dieser Arbeit berücksichtigt und werden im folgenden Kapitel dargestellt.

1.4 Verwandte Arbeiten

Mit dem Artikel [3] von Bronstein et al. ist ein 2017 publizierter Übersichtsartikel des Forschungsfeldes GDL gegeben. Hierin werden in komprimierter Form die mathematischen Grundlagen für das Lernen auf Graphen und Riemannschen Mannigfaltigkeiten bereitgestellt und Verallgemeinerungen der euklidischen Convolutional Neural Network (CNN) Modellarchitektur auf nichteuklidische Bereiche dargestellt.

Im Artikel [22] von Zhang et al. werden Graph Neural Networks (GNNs), insbesondere Graph CNNs nach Anwendungen und zugrunde liegendem Faltungsmechanismus kategorisiert.
Aufgrund der Schnelllebigkeit des Forschungsfeldes publizierten Cao et al. mit dem Artikel [5] der im Jahr 2020 publiziert wurde einen zum Artikel [3] von Bronstein et al. verwandten Übersichtsartikel. Dieser schließt die bis dahin aufgrund neu entwickelten Modellarchitekturen entstandene Lücke.
Im Artikel [25] von Zhou et al. stellen diese eine generelle Designpipeline für GNN Modellarchitekturen vor.
In dem Artikel [20] von Xiao et al. werden die Vor- und Nachteile verschiedener Repräsentationen von 3D Formdaten wie Punktwolken oder Polygonnetze bezüglich unterschiedlicher Modellarchitekturen untersucht.
Der Artikel [21] von Yuan et al. stellt sowohl traditionelle als auch DL Methoden für die 3D Form Editierung, insbesondere Deformationstechniken für Polygonnetze vor.
Der Artikel [1] von Abadal et al. stellt Grundlagen der GNNs vor und gibt eine historische Einordnung bezüglich verschiedener GNN Modellarchitekturen. Außerdem werden effiziente Rechentechniken bezüglich spezieller Hardware präsentiert.
Im prototypischen Buch [4] das von Bronstein et al. im Jahr 2021 veröffentlicht wurde wird der GDL Blueprint vorgestellt. Mit diesem können viele existierende Modellarchitekturen eingeordnet werden.
Der Artikel [6] von Cao et al. beschäftigt sich neben einer kurzen Einführung in das GDL sowie einer Einordnung von gängigen Anwendungen hauptsächlich mit den bekannten Problemen bisheriger Modellarchitekturen sowie zu bewältigenden Herausforderungen des Forschungsfeldes.
Im Artikel [7] von Cao et al. wird die Historie des Forschungsfeldes GDL beleuchtet und Benchmark Datensätze für verschiedene Modellarchitekturen vorgestellt. Zudem werden potenzielle Forschungsrichtungen und Herausforderungen des Forschungsfeldes besprochen.
Der Artikel [11] von Keramatfar et al. stellt eine bibliometrische Übersicht über Artikel die GNNs behandeln seit 2004 dar. Es werden Forschungstrends sowohl quanti- als auch qualitativ evaluiert.
Zhou et al. evaluieren und kategorisieren im Artikel [26] insgesamt 327 Artikel im Bereich GNNs .
Der Artikel [19] von Ward et al. stellt ein Tutorial bezüglich GNNs dar und stellt praxisnahe Beispiele vor.

1.5 Aufbau der Arbeit

In Kapitel 2 wird der Rahmen festgelegt in dem die Ergebnisse eingeordnet werden. Es wird auf das überwachte Lernen eingegangen und die empirische Risikominimierung (ERM) der statistischen Lerntheorie erläutert. Weiter umfasst das Kapitel technische Erläuterungen zu Gruppen. Im Kapitel 3 werden die Ergebnisse vorgestellt. Hier wird der GDL Blueprint erläutert und anhand der CNN Modellarchitektur verdeutlicht. Daraufhin wird eine Fehlerzerlegung der ERM präsentiert. Das Kapitel 4 enthält eine Diskussion der Ergebnisse bezüglich der gestellten Forschungsfragen und Das Kapitel 5 beinhaltet ein abschließendes Fazit.

2 Grundlagen

Im folgenden werden die Grundlagen erläutert die zum Verständnis der Ergebnisse benötigt werden. Ziel ist es einen Rahmen bereitzustellen indem die Ergebnisse eingeordnet und diskutiert werden können.

2.1 Überwachtes Lernen

Es sei die Menge $\mathcal{X}$ der Attribute und die Menge $\mathcal{Y}$, der Label gegeben. Dann heißt die Menge $\mathcal{D} = \{(x_i, y_i) : x_i \in \mathcal{X}, y_i \in$

$\mathcal{Y}$, für $i = 1, ..., n\}$ bestehend aus $n \in \mathbb{N}$ Elementen die Menge der Daten. Weiter wird eine unbekannte Funktion $f : \mathcal{X} \to \mathcal{Y}$ angenommen. Deren unabhängige Variable entspricht $x \in \mathcal{X}$ und deren abhängige Variable entspricht gerade $y \in \mathcal{Y}$. Die Funktion f wird Hypothese genannt. Das Problem des überwachten Lernens entspricht dann dem Bestimmen der Hypothese f mithilfe der Daten $\mathcal{D}$ [4].

2.2 Empirische Risikominimierung

Beim statistischen Lernen wird eine statistische Sichtweise auf das maschinelle Lernen eingenommen. Eine Verfahrensweise um die Güte einer Hypothese einzuschätzen ist die ERM. Daten werden als Observationen einer zugrunde liegenden Wahrscheinlichkeitsverteilung interpretiert. Sei sonach eine Wahrscheinlichkeitsverteilung P über $\mathcal{X} \times \mathcal{Y}$ definiert und die Observationen bezüglich $P(x, y)$ unabhängig und identisch verteilt.[1] Sei außerdem $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}, (y_1, y_2) \mapsto L(y_1, y_2)$ eine Verlustfunktion die die Differenz zwischen dem vorhergesagten Label $f(x)$ und dem tatsächlichen Label y misst. Dann bezeichnet:

$$\mathcal{R}(f) = \int_{\mathcal{X}\times\mathcal{Y}} L(f(x), y)\, dP(x, y) \quad (1)$$

den durchschnittlichen Verlust, auch das Risiko genannt.[1] Mit n Observationen $\mathcal{D}$ ist das empirische Risiko mit:

$$\widehat{\mathcal{R}}(f) = \frac{1}{n} \sum_{i=1}^{n} L(f(x_i), y_i) \quad (2)$$

definiert.[2] Ziel der ERM ist es $\mathcal{R}(f)$ zu minimieren. Da $P(x, y)$ unbekannt ist wird anstelle von $\mathcal{R}(f)$ das empirische Risiko $\widehat{\mathcal{R}}(f)$ auf den bekannten Observationen ausgewertet und mithilfe eines Lernalgorithmus minimiert [2].[1] Ziel ist es, dass das DL Architekturmodell gut generalisiert, demgemäß nicht nur auf den bekannten Observationen eine bestmögliche Approximation leistet sondern auch auf bisher unbekannten Daten.

2.3 Gruppen

Informell gesprochen ist eine Symmetrie eines Objektes oder eines Systems eine Transformation die das Objekt oder das System unverändert oder anders ausgedrückt invariant lässt [4]. Eine Gruppe ist ein algebraisches Objekt das Symmetrien codiert und wird im folgenden definiert.

2.3.1 Gruppen

Ein geordnetes Paar $(G, \circ)$, bestehend aus einer Menge G und einer inneren, zweistelligen Verknüpfung
$\circ : G \times G \to G : (g_1, g_2) \mapsto g_1 \circ g_2$ mit den folgenden Axiomen:

- Für alle $g_1, g_2, g_3 \in G$ gilt:
  $(g_1 \circ g_2) \circ g_3 = g_1 \circ (g_2 \circ g_3)$.
- Es gibt ein Element $e \in G$, so dass für alle $g \in G$ gilt:
  $g \circ e = e \circ g = g$.
- Für jedes Element $g \in G$ gibt es ein Element g^{-1}, so dass gilt:
  $g \circ g^{-1} = g^{-1} \circ g = e$.

wird als Gruppe bezeichnet [10]. Das Element e wird als neutrales Element bezeichnet. Wenn möglich wird im folgenden die Verknüpfung in der Notation unterdrückt, somit G anstelle von $(G, \circ)$ geschrieben. Als Beispiel ist hier die Diedergruppe D_3, die Symmetriegruppe eines ebenen gleichseitigen Dreiecks angeführt. Die Gruppe enthält 6 Elemente, nämlich drei Rotationen r_1, r_2 und r_3 um 0°=360°, 120° und 240°, sowie die Spiegelungen s_1, s_2, s_3 an den drei Mittelsenkrechten. Der Zusammenhang ist in der Abbildung 2 dargestellt. Werden die Ecken des Dreiecks nummeriert so vertauschen sie sich unter einer Rotation oder Spiegelung entsprechend.

[1] https://www-m5.ma.tum.de/foswiki/pub/M5/Allgemeines/MA4801_2021S/ML.pdf, zuletzt abgerufen am 29.10.2022.
[2] https://geometricdeeplearning.com/lectures/, zuletzt abgerufen am 29.10.2022.

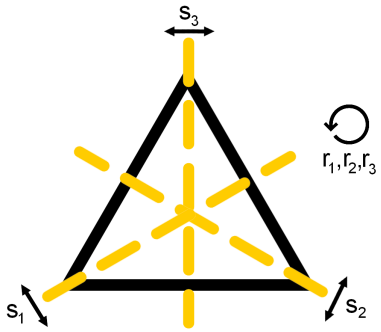

Abbildung 2: Die Diedergruppe D_3 als Symmetriegruppe eines gleichseitigen Dreiecks mit Rotationen r_1, r_2, r_3 und Spiegelungen s_1, s_2, s_3

Fasst man die Ecken als Menge auf operiert die Gruppe auf der Menge. Eine solche Verknüpfung wird Gruppenaktion genannt.

2.3.2 Gruppenaktionen

Sei G eine Gruppe und $\mathcal{X}$ eine nichtleere Menge. Dann operiert die Gruppe G auf der Menge $\mathcal{X}$ mit der äußeren zweistelligen Verknüpfung $\triangleright : G \times \mathcal{X} \to \mathcal{X}, (g, x) \mapsto g \triangleright x$, wenn folgende Axiome erfüllt sind:

- Für alle $x \in \mathcal{X}$ und das neutrale Element $e \in G$ gilt: $e \triangleright x = x$.
- Für alle $g_1, g_2 \in G$ und $x \in \mathcal{X}$ gilt: $(g_1 \circ g_2) \triangleright x = g_1 \triangleright (g_2 \triangleright x)$.

Die Verknüpfung $\triangleright$ wird Gruppenaktion genannt [10].
Operiert nun eine Gruppe auf Elemente die einer Funktion als Argumente dienen und ändert sich der Funktionswert nicht unter der Gruppenaktion, spricht man von einer invarianten Funktion bezüglich dieser Gruppe [4].

2.3.3 Invariante Funktionen

Eine Funktion $f : \mathcal{X} \to \mathcal{Y}$ wird G invariant genannt, wenn gilt: $f(g \triangleright x) = f(x)$, für alle $g \in G$ und $x \in \mathcal{X}$.

Solche Invarianzen sind im DL allgegenwärtig. Beispielsweise beim Bestehen einer Rotationsinvarianz um 90° bei der Einzelobjektklassifizierung auf zweidimensionalen Bildern. Dies ist in der Abbildung 1 dargestellt. Ein anderes Beispiel ist in der computergestützten Chemie zu finden. Hier sollen Eigenschaften von Molekülen unabhängig ihrer Orientierung im Raum vorhergesagt werden. Eine erklärende Hypothese f soll hier folglich Rotationsinvariant sein. Weißen Daten bestimmte Symmetrien auf, soll die Hypothese die die Daten erklärt diese Symmetrien respektieren. Die Hypothese soll G invariant bezüglich einer Gruppe G sein. Durch die Daten sind Hypothesenräume infolgedessen mit $\mathcal{F} \subseteq \{f : \mathcal{X} \to \mathcal{Y}, f \text{ ist G invariant}\}$ gegeben. Wird nun der Hypothesenraum mittels Ansatzfunktionen approximiert sollen die Ansatzfunktionen auch die Eigenschaften des Hypothesenraums besitzen.

3 Ergebnisse

Im folgenden wird vorgestellt wie sich G invariante Ansatzfunktionen im allgemeinen erzeugen lassen. Anschließend wird ein konkretes Beispiel vorgeführt und hiernach eine Fehlerabschätzung im Kontext der ERM vorgenommen.

3.1 Geometric Deep Learning Blueprint

Der im Artikel [4] von Bronstein et al. vorgestellte GDL Blueprint erlaubt es G invariante Ansatzfunktionen zu erzeugen. Dies wird im folgenden dargestellt.

3.1.1 Signale auf geometrischen Bereichen

Sei Ω eine Menge und hier Bereich genannt. DL Architekturmodelle operieren auf Funktionen die auf Bereichen definiert sind und die im folgenden Signale genannt werden. Sei $\mathcal{X}(\Omega, C) = \{x : \Omega \to C\}$ der Raum der vektorwertigen Signale auf Ω, mit einem Hilbertraum C. Außerdem sei der Raum

$\mathcal{X}(\Omega, C)$ selbst ein Hilbertraum. Ein Hilbertraum ist ein Vektorraum mit einem Skalarprodukt, der vollständig ist. Demnach ein Raum in dem jede Cauchy Folge konvergiert.

3.1.2 Gruppenaktionen auf Signalen

Sei mit $\triangleright : G \times \Omega \to \Omega, (g, u) \mapsto g \triangleright u$ eine Gruppenaktion auf Ω gegeben. Dann ist mit $\dot{\triangleright} : G \times \mathcal{X}(\Omega, C) \to \mathcal{X}(\Omega, C), (g, x) \mapsto (g \dot{\triangleright} x)(u) = x(g^{-1} \triangleright u)$ eine Gruppenaktion auf $\mathcal{X}(\Omega, C)$ gegeben. Weiterhin ist $\dot{\triangleright}$ linear.

3.1.3 Equivariante Funktionen

Eine Funktion $f : \mathcal{X}(\Omega, C) \to \mathcal{X}(\Omega, C)$ wird G equivariant genannt, falls gilt: $f(g \dot{\triangleright} x) = g \dot{\triangleright} f(x)$.

3.1.4 Geometric Deep Learning Blueprint

Seien Ω und Ω' Bereiche und G eine Gruppe die auf Ω operiert. Des Weiteren sei Ω' eine gröbere Version von Ω. Dann sind die folgenden Bausteine definiert:

- Eine lineare G equivariante Funktion $B : \mathcal{X}(\Omega, C) \to \mathcal{X}(\Omega', C')$, für die $B(g \triangleright x) = g \dot{\triangleright} B(x)$, für alle $g \in G$ und $x \in \mathcal{X}(\Omega, C)$ gilt.
- Ein nichtlineare Funktion $\sigma : C \to C'$, elementweise angewandt als $((\dot{\sigma})(x))(u) = \sigma(x(u))$.
- Eine vergröbernde Funktion $P : \mathcal{X}(\Omega, C) \to \mathcal{X}(\Omega', C)$
- Eine G invariante Funktion $A : \mathcal{X}(\Omega, C) \to \mathcal{Y}$, mit $A(g \dot{\triangleright} x) = A(x)$, für alle $g \in G$ und $x \in \mathcal{X}(\Omega, C)$.

Diese Bausteine ermöglichen es G invariante Funktionen $f : \mathcal{X}(\Omega, C) \to \mathcal{Y}$, der Form:

$$f = A \circ \dot{\sigma}_n \circ B_n \circ P_{n-1} \circ \ldots \circ P_1 \circ \dot{\sigma}_1 \circ B_1 \quad (3)$$

zu konstruieren. Mit $\circ$ ist hier die übliche Komposition von Funktionen bezeichnet.

3.2 Convolutional Neural Networks

Im folgenden werden die Bausteine des GDL Blueprints wie im Artikel [4] von Bronstein et al. ausgeführt den Bausteinen der CNN Modellarchitektur zugeordnet. Damit lässt sich zeigen das der GDL Blueprint nicht nur theoretisch G invariante Ansatzfunktionen erzeugt, sondern das solche Ansatzfunktionen auch tatsächlich existieren. Es werden quadratische, skalarwertige diskretisierte Bilder als Datengrundlage betrachtet. Dann sei der zugrunde liegende zweidimensionale geometrische Bereich durch $\Omega = \{(i, j) : i = 0, \ldots, d-1 \text{ und } j = 0, \ldots, d-1\}$ mit $u = (u_1, u_2) \in \Omega$ gegeben. Die skalarwertigen Signale sind mit $\mathcal{X}(\Omega, \mathbb{R})$ gegeben. Der geometrische Bereich Ω bildet ein Gitter. Mit zusätzlich geforderten periodischen Randbedingungen sei die zugrunde liegende Gruppe das direkte Produkt $G = \mathbb{Z}_d \times \mathbb{Z}_d$ der zyklischen Gruppe $\mathbb{Z}_d$. Die Gruppe G wird als diskrete Translationsgruppe bezeichnet. Sei $C_\Theta \in \mathbb{R}^{n \times n}$ eine zyklische Matrix, also eine Matrix der Gestalt:

$$C_\Theta = \begin{pmatrix} \theta_0 & \theta_{n-1} & \cdots & \theta_1 \\ \theta_1 & \theta_0 & \cdots & \theta_2 \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{n-1} & \theta_{n-2} & \cdots & \theta_0 \end{pmatrix} \quad (4)$$

Eine zyklische Matrix erhält man dadurch das die jeweils nächste Spalte durch zyklisches Verschieben der Spalte davor gewonnen wird. Die erste Spalte besteht aus dem Vektor $\Theta = (\theta_0, ..., \theta_{n-1})^T \in \mathbb{R}^n$. Das Produkt $C_\Theta x$ einer zyklischen Matrix C_Θ und eines Vektors $x \in \mathbb{R}^n$ kann als diskreter Faltungsoperator:

$$(x * \Theta)_i = \sum_{j=0}^{n-1} x_{j \bmod n}\, \theta_{i-j \bmod n} \quad (5)$$

interpretiert werden. Es gilt demnach $C_\Theta x = (x * \Theta)$. Die spezielle Wahl von $\Theta = (0, 1, \ldots, 0)^T$ führt zu der zyklischen Matrix die hier mit S bezeichnet ist. Die Matrix S heißt Translationsoperator. Weiter ist das Produkt von zyklischen Matrizen kommutativ. Es gilt demzufolge $C_\Theta C_H = C_H C_\Theta$ für

$\Theta, H \in \mathbb{R}^n$. Mit $C_H = S$ ist der Faltungsoperator equivariant bezüglich des Translationsoperators $C_\Theta S(x) = SC_\Theta(x)$.
Sei der Faltungskern Θ der Größe $l \times m$, interpretiert als Vektor aus $\mathbb{R}^{l \cdot m}$, in Form der Linearkombination $\alpha_{11}\Theta_{1,1} + \cdots + \alpha_{l \cdot m}\Theta_{l,m}$ mit $\alpha_{vw} \in \mathbb{R}, \Theta_{v,w} \in \mathbb{R}^{l \cdot m}$ gegeben. Dann ist mit der konkreten Wahl $\Theta_{vw}(u_1, u_2) = \delta(u_1 - v, u_2 - w)$ des Faltungskerns jede lokale lineare equivariante Funktion darstellbar als:

$$B(x) = \sum_{v=1}^{l} \sum_{w=1}^{m} \alpha_{vw} C_{\Theta_{v,w}} x \tag{6}$$

mit Patch $x \in \mathbb{R}^{l \times m}$, ebenso interpretiert als Vektor aus $\mathbb{R}^{l \cdot m}$. Für die konkrete Wahl $\Theta_{vw}(u_1, u_2)$ ist $\delta(0) = 1$, ansonsten ist $\delta(u_1 - v, u_2 - w) = 0$. Dies führt in Koordinaten zur Formel für die diskrete Faltung in zwei Dimensionen:

$$B_{uv}(x) = \sum_{a=1}^{l} \sum_{b=1}^{m} \alpha_{ab} x_{u+a, v+b} \tag{7}$$

Die populärste elementweise angewandte nichtlineare Aktivierungsfunktion σ der CNN Modellarchitektur ist die Rectifier Aktivierungsfunktion $\sigma(x) = \max(x, 0)$ [4].
Das Vergröbern dient hier als Gittervergröberung $P : \mathcal{X}(\Omega, C) \to \mathcal{X}(\Omega', C)$. Im Kontext der CNNs werden solche Vergröberungsstrategien als Pooling bezeichnet. Bekannte Poolingstrategien sind das Max Pooling oder das Average Pooling.
Eine zu den Kompositionen $P_i \circ \dot{\sigma}_i \circ B_i$ korrespondierende Schicht eines CNNs kann insgesamt als $L_i = P_i(\sigma_i(B_i(x)))$ angegeben werden.
Die G invariante Funktion A, im Kontext der CNNs auch translationsinvariantes globales-Pooling genannt, kann mit der Mittelwertsfunktion als average Pooling umgesetzt werden.

3.3 Fehlerzerlegung

Im Kontext der ERM kann der Fehler des Lernprozesses in verschiedene Komponenten zerlegt werden. Gibt man die Ansatzfunktionen vor lassen sich mit der Fehlerzerlegung Rückschlüsse auf den Fehler insgesamt ziehen.
Sei $\mathcal{R}^* = \inf_{f \in \mathcal{X} \times \mathcal{Y}} \mathcal{R}(f)$ das optimale Risiko und $\mathcal{R}_\mathcal{F} = \inf_{f \in \mathcal{F}} \mathcal{R}(f)$ das optimale Risiko eines festen Hypothesenraumes $\mathcal{F}$. Sei eine Funktion $\hat{f} \in \mathcal{F}$ gegeben die das empirische Risiko minimiert, also $\widehat{\mathcal{R}}(\hat{f}) \leq \widehat{\mathcal{R}}(f)$, für alle $f \in \mathcal{F}$. Dann kann nach Bottou und Bousquet in [2] die Differenz zwischen dem Risiko einer Ansatzfunktion f die von einem Lernalgorithmus approximiert ist und dem optimalen Risiko zerlegt werden in:

$$\begin{aligned} \mathcal{R}(f) - \mathcal{R}^* = \quad & \underbrace{(\mathcal{R}(f) - \mathcal{R}(\hat{f}))}_{\text{Optimierungsfehler}} + \\ & \underbrace{(\mathcal{R}(\hat{f}) - \mathcal{R}_\mathcal{F})}_{\text{Schätzungsfehler}} + \\ & \underbrace{(\mathcal{R}_\mathcal{F} - \mathcal{R}^*)}_{\text{Approximationsfehler}} \end{aligned} \tag{8}$$

Dabei stellt f eine Annäherung an $\hat{f}$ dar.
Der Optimierungsfehler hängt davon ab wie gut der Lernalgorithmus der zu f geführt hat bezüglich idealer ERM ist. Der Schätzungsfehler misst wie gut der empirische Risikominimierer $\hat{f} \in \mathcal{F}$ im Vergleich zum optimalen Risikominimierer abschneidet. Er hängt von der Anzahl der Observationen und der Größe des Hypothesenraumes ab. Größere bzw. komplexere Hypothesenräume führen zu größeren Schätzungsfehlern. Der Approximationsfehler hingegen ist unabhänig von den Observationen sowie der Ansatzfunktion f. Er gibt an wie gut der Hypothesenraum $\mathcal{F}$ ist. Komplexerere bzw. reichhaltigere Hypothesenräume führen zu kleineren Approximationsfehlern. Da der Hypothesenraum durch den GDL Blueprint auf Ansatzfunktionen eingeschränkt wird die bestimmte Eigenschaften besitzen verringert sich die Komplexität des Hypothesenraumes. Der Schätzungsfehler sollte folglich im Vergleich zu

Ansatzfunktionen die die bestimmten Eigenschaften nicht erfüllen kleiner werden. Da der Approximationsfehler unabhängig von den Ansatzfunktionen ist und die Eigenschaften der Ansatzfunktionen (da sie mit dem GDL Blueprint konstruiert sind) die Struktur der Daten respektieren, sollte sich der Approximationsfehler nicht verschlechtern.[2]

4 Diskussion

Mithilfe des GDL Blueprints lassen sich bereits existierende Architekturen die sowohl auf euklidischen als auch auf nichteuklidischen Daten agieren einordnen. Der Fall der CNN Modellarchitektur auf euklidischen Daten ist im Kapitel 3 behandelt. Ist der Bereich hingegen eine Riemannsche Mannigfaltigkeit und die Symmetriegruppe die Gruppe der Isometrien lassen sich damit ebenfalls Modellarchitekturen einordnen [4]. GNNs können dadurch eingeordnet werden das der Bereich ein Graph darstellt und die Symmetriegruppe gerade die Grupper aller Permutationen.

DL Modellarchitekturen treffen Annahmen über noch nicht verarbeitete Daten die nicht aus den Informationen der bereits gesehenen Daten hergeleitet werden können [17]. Dies wird Inductive Bias genannt. Findet dies in der Form statt, dass der Hypothesenraum eingeschränkt wird wird dies als Declarative Bias bezeichnet [17]. Die Invarianzeigenschaft der Ansatzfunkionen ist damit als Declarative Bias zu verstehen. Daten werden aus der echten Welt gewonnen. Diese verhält sich nach gewissen physikalischen Gesetzmäßigkeiten, die wiederum bestimmten Symmetrien unterliegen. Diese Symmetrien sollten sich sonach auch in den Daten widerspiegeln. Die Ansatzfunkionen dementsprechend zu wählen erscheint somit sinnvoll.

Komplexe Daten weißen Korrelationen auf, die sich über mehrere Skalen hinweg erstrecken. Eine Modellarchitektur sollte deshalb dazu in der Lage sein Signale auf mehreren Skalen zu repräsentieren, um Zusammenhänge dieser Art erfassen zu können. Der GDL Blueprint setzt diese Eigenschaft mithilfe der Vergröberungsfunktionen um. Ist f' auf Signalen $x \in \mathcal{X}(\Omega', C)$ definiert genügt $f \approx f' \circ P$. Der Sachverhalt ist in Abbildung 3 illustriert. Mit mehreren Schichten und dazugehörigen Vergröberungsfunktionen werden sowohl Korrelationen die auf einer feinen Auflösung bestehen als auch diese die auf gröberen Auflösungen bestehen erkannt. Dies geschieht indem die Korrelationen lokal erfasst werden und dann an die gröberen Auflösungen weitergereicht werden [4]. Ein Beispiel für Langzeitkorrelationen sind Zeitreihen des menschlichen Herzschlags [14].

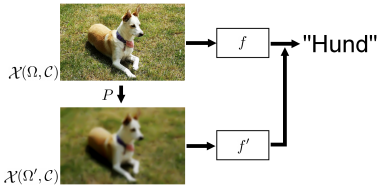

Abbildung 3: Vergröberung am Beispiel der Einzelobjektklassifizierung

Eine weitere Eigenschaft die mit dem GDL Blueprint umgesetzt wird ist die Deformationsstabilität. Die Eigenschaft meint das die Modellarchitektur sich invariant bezüglich hinreichend kleiner Deformationen verhalten soll. Sowohl Signale als auch der geometrische Bereich an sich kann Deformationen unterliegen. Beispielsweise kann ein Bild das mit entsprechender Belichtungszeit durch eine Fotokamera aufgenommen wird eine Bewegungsunschärfe enthalten. Trotzdem sollte beim Beispiel der Einzelobjektklassifizierung immer noch dasselbe Objekt identifiziert werden. Im Kontext der Sozialen Netzwerke können zu zwei unterschiedlichen Zeitpunkten unterschiedliche Bekannschaftsbeziehungen vorliegen. Hier liegt demgemäß eine Änderung des Bereichs vor [4]. Deformationsstabilität wird umgesetzt indem lokal

Deformationsequivariante Funktionen angesetzt werden.
Der GDL Blueprint ist schichtweise bzw. kompositionell aufgebaut. Unter der Annahme das mithilfe der Kompositionalität komplexe Sachverhalte wie sie in der echten Welt vorkommen gut erklärt werden können ist dies ebenfalls eine sinnvolle Eigenschaft.[3]
Konkrete CNN Modellarchitekturen die das generelle Schema das im Kapitel 3 vorgestellt wird erfüllen sind LeNet, AlexNet, ResNet, sowie UNet [9, 12, 13, 16].
Es existieren CNN Modellarchitekturen in denen keine Translationsinvarianz gegeben ist [4]. Solche Modellarchitekturen lassen sich ohne auch den zugrunde liegenden Bereich Ω zu ersetzen nicht mit dem GDL Blueprint ableiten. Gängige algorithmische Erweiterungen von DL Modellarchitekturen wie die Batch Normalization lassen sich ebenfalls nicht mit dem GDL Blueprint abbilden [4].
Außerdem sind keine Bausteine durch den GDL Blueprint gegeben die andere Transformationen wie Illuminationseffekte oder Farbänderungen erhalten [4]. Diese Informationen können aber mithilfe Datenaugmentation ohne architekturelle Änderungen in die Modellarchitektur inkooperiert werden [4].
Für die Fehlerzerlegung muss eine ERM vorliegen. Die ERM kann jedoch zu Overfitting führen. Also dazu das die gelernte Hypothese gut auf den bekannten Observationen agiert, jedoch keine ungesehenen Daten gut erklärt bzw. vorhersagt [23].
Um bei der ERM trotzdem eine gute Generalisierung zu realisieren kann angelehnt an das Sparsamkeisprinzip auch Ockhams Rasiermesser genannt, wobei die am wenigsten komplexe Hypothese ausgewählt wird, die Regularisierung eingeführt werden.
Sei dazu der Hypothesenraum mit einem nichtnegativen Komplexitätsmaß $\gamma : \mathcal{F} \to \mathbb{R}$ ausgestattet. Ein Komplexitätsmaß ist beispielsweise die Anzahl der Neuronen eines neuronalen Netzes. Dann kann im Kontext der ERM die Regularisierung durchgeführt werden mit der Funktion $\tilde{f} \in \mathcal{F}$ die das gegebene Komplexitätsmaß γ minimiert, also für die $\gamma(\tilde{f}) \leq \gamma(f)$, für alle $f \in \mathcal{F}$ gilt. Das Modell wird so auf eine bestimme Komplexität beschränkt. Ein komplexes Modell tendiert zu Overfitting da es eher das Rauschen in den bekannten Daten lernt als die wirklich relevanten Zusammenhänge [24]. Eine Verwirklichung der Regularisierung findet damit so statt, dass die ERM in einer eingeschränkten Form mit dem restringierten Hypothesenraum $\mathcal{F}_\delta = \{f \in \mathcal{F} : \gamma(f) \leq \delta\}$ als $\hat{f}_\delta = \arg\min_{f \in \mathcal{F}_\delta} \widehat{\mathcal{R}}(f)$ ausgeführt wird [4].

5 Fazit

Mithilfe des GDL Blueprints können G invariante Ansatzfunktionen konstruiert werden. Für diese Kontruktion müssen auf entsprechenden Bereichen zu bestimmten Gruppen equivariante und invariante Funktionen sowie nichtlineare und vergröbernde Funktionen definiert werden. Damit können Modellarchitekturen die auf euklidischen sowie Modellarchitekturen die auf nichteuklidischen Daten agieren beschrieben werden.
Außerdem setzt der GDL Blueprint weitere Eigenschaften um unter denen komplexe Hypothesen voraussichtlich gut approximiert werden können.
Im Kontext der ERM kann eine Fehlerzerlegung in drei Komponenten durchgeführt werden. Bei Ansatzfunktionen die durch den GDL Blueprint erzeugt werden sollte eine Fehlerkomponente nicht verschlechtert und eine zweite Fehlerkomponente verbessert werden.

[3] https://cfcs.pku.edu.cn/docs/2019-10/20191010101730652831.pdf, zuletzt abgerufen am 29.10.2022.

Literatur

[1] Sergi Abadal u. a. „Computing graph neural networks: A survey from algorithms to accelerators“. In: *ACM Computing Surveys (CSUR)* 54.9 (2021), S. 1–38. DOI: 10.1145/3477141.

[2] Léon Bottou und Olivier Bousquet. „The Tradeoffs of Large Scale Learning“. In: *Advances in Neural Information Processing Systems*. Hrsg. von J. Platt u. a. Bd. 20. Curran Associates, Inc., 2007. ISBN: 9781605603520. URL: https://proceedings.neurips.cc/paper/2007/file/0d3180d672e08b4c5312dcdafdf6ef36-Paper.pdf.

[3] Michael M. Bronstein u. a. „Geometric Deep Learning: Going beyond Euclidean data“. In: *IEEE Signal Processing Magazine* 34.4 (2017), S. 18–42. DOI: 10.1109/MSP.2017.2693418.

[4] Michael M. Bronstein u. a. „Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges“. In: *CoRR* abs/2104.13478 (2021). arXiv: 2104.13478. URL: https://arxiv.org/abs/2104.13478.

[5] Wenming Cao u. a. „A Comprehensive Survey on Geometric Deep Learning“. In: *IEEE Access* 8 (2020), S. 35929–35949. DOI: 10.1109/ACCESS.2020.2975067.

[6] Wenming Cao u. a. „Geometric deep learning: progress, applications and challenges“. In: *Science China Information Sciences* 65.2 (2022), S. 1–3. DOI: 10.1007/s11432-020-3210-2.

[7] Wenming Cao u. a. „Geometric machine learning: research and applications“. In: *Multimedia Tools and Applications* (2022), S. 1–53. DOI: 10.1007/s11042-022-12683-9.

[8] Mehdi Gheisari, Guojun Wang und Md Zakirul Alam Bhuiyan. „A Survey on Deep Learning in Big Data“. In: *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*. Bd. 2. 2017, S. 173–180. DOI: 10.1109/CSE-EUC.2017.215.

[9] Kaiming He u. a. „Deep Residual Learning for Image Recognition“. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, S. 770–778. DOI: 10.1109/CVPR.2016.90.

[10] Christian Karpfinger und Kurt Meyberg. *Algebra - Gruppen - Ringe - Körper*. 4. Aufl. Berlin Heidelberg New York: Springer-Verlag, 2017. ISBN: 978-3-662-54722-9.

[11] Abdalsamad Keramatfar, Mohadeseh Rafiee und Hossein Amirkhani. „Graph Neural Networks: A bibliometrics overview“. In: *Machine Learning with Applications* 10 (2022), S. 100401. DOI: 10.1016/j.mlwa.2022.100401.

[12] Alex Krizhevsky, Ilya Sutskever und Geoffrey E Hinton. „Imagenet classification with deep convolutional neural networks“. In: *Communications of the ACM* 60.6 (2017), S. 84–90. DOI: 10.1145/3065386.

[13] Yann LeCun u. a. „Gradient-based learning applied to document recognition“. In: *Proceedings of the IEEE* 86.11 (1998), S. 2278–2324. DOI: 10.1109/5.726791.

[14] Danuta Makowiec u. a. „Long-range dependencies in heart rate signals—revisited“. In: *Physica A: Statistical Mechanics and its Applications* 369.2 (2006), S. 632–644. DOI: 10.1016/j.physa.2006.02.038.

[15] Henry Parker Manning. *Introductory non-euclidean geometry*. Courier Corporation, 2005. ISBN: 0-486-44262-4.

[16] Olaf Ronneberger, Philipp Fischer und Thomas Brox. „U-net: Convolutional networks for biomedical image segmentation“. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, S. 234–241. DOI: 10.1007/978-3-319-24574-4_28.

[17] „Inductive Bias“. In: *Encyclopedia of Machine Learning*. Hrsg. von Claude Sammut und Geoffrey I. Webb. Boston, MA: Springer US, 2010, S. 522–522. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_390.

[18] Pramila P. Shinde und Seema Shah. „A Review of Machine Learning and Deep Learning Applications“. In: *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*. 2018, S. 1–6. DOI: 10.1109/ICCUBEA.2018.8697857.

[19] Isaac Ronald Ward u. a. „A Practical Tutorial on Graph Neural Networks“. In: *ACM Computing Surveys (CSUR)* 54.10s (2022), S. 1–35. DOI: 10.1145/3503043.

[20] Yun-Peng Xiao u. a. „A survey on deep geometry learning: From a representation perspective“. In: *Computational Visual Media* 6.2 (2020), S. 113–133. DOI: 10.1007/s41095-020-0174-8.

[21] Yu-Jie Yuan u. a. „A revisit of shape editing techniques: From the geometric to the neural viewpoint“. In: *Journal of Computer Science and Technology* 36.3 (2021), S. 520–554. DOI: 10.1007/s11390-021-1414-9.

[22] Si Zhang u. a. „Graph convolutional networks: a comprehensive review“. In: *Computational Social Networks* 6.1 (2019), S. 1–23. DOI: 10.1186/s40649-019-0069-y.

[23] Xinhua Zhang. „Empirical Risk Minimization“. In: *Encyclopedia of Machine Learning*. Hrsg. von Claude Sammut und Geoffrey I. Webb. Boston, MA: Springer US, 2010, S. 312–312. DOI: 10.1007/978-0-387-30164-8_251.

[24] Xinhua Zhang. „Regularization“. In: *Encyclopedia of Machine Learning*. Hrsg. von Claude Sammut und Geoffrey I. Webb. Boston, MA: Springer US, 2010, S. 845–849. DOI: 10.1007/978-0-387-30164-8_712.

[25] Jie Zhou u. a. „Graph neural networks: A review of methods and applications“. In: *AI Open* 1 (2020), S. 57–81. DOI: 10.1016/j.aiopen.2021.01.001.

[26] Yu Zhou u. a. „Graph Neural Networks: Taxonomy, Advances, and Trends“. In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 13.1 (2022), S. 1–54. DOI: 10.1145/3495161.

Autorenverzeichnis

Hochschule Reutlingen
Reutlingen University
Fakultät Informatik
Human-Centered Computing
Alteburgstraße 150
D-72762 Reutlingen

Telefon: +49 7121 / 271-4002
Telefax: +49 7121 / 271-4032

E-Mail: informatics.inside@reutlingen-university.de
Website: informatics.inside@reutlingen-university.de

ISBN
978-3-00-073892-0