Eine Virtual-Reality Anwendung zur Simulation von Mensch-Roboter-Kollaboration

Sven Gadow, Alexander Nachtigall, Nico Pistel, Michael Potthoff, Bastian Schaffeld, Thorben von Horn, Tom Vierjahn

Westfälische Hochschule
Fachbereich Wirtschaft und Informationstechnik
Visual Computing / Mechatronik-Institut Bocholt
Campus Bocholt

E-Mail: tom.vierjahn@w-hs.de

Zusammenfassung: In Kollaboration mit dem Fachbereich Maschinenbau wurde eine prototypische Lösung zur Visualisierung der Interaktion zwischen Mensch und Maschine bei kooperativen Arbeiten entwickelt. Der Prototyp wurde in der Programmiersprache C++ auf Basis der *Unreal Engine 4* realisiert und soll als Grundlage für weitere Forschungen im Bereich der Mensch-Roboter-Kollaboration dienen. Zur Echtzeitsimulation der Maschinen wurde eine Anbindung basierend auf dem Protokoll *OPC UA* integriert, sodass eine Kopplung mit Enterprise-Applikationen wie *Siemens NX MCD* und *ABB RobotStudio* möglich ist. Für eine realitätsgetreue Darstellung können die Maschinen in der virtuellen Realität abgebildet werden. Die Visualisierung eines Menschen erfolgt durch Einbindung der *Microsoft Azure Kinect*, wodurch eine Person durch eine farbliche Punktwolke oder ein Skelett angezeigt werden kann.

Stichworte: Virtual-Reality, Robotik, Simulation

1 Einleitung

Bei der Herstellung von Produkten im Zeitalter der Industrie 4.0 werden auch immer mehr industrielle Roboter eingesetzt, um manuelle Vorgänge zu automatisieren (siehe [NFM20] für eine Diskussion der Entwicklung von Mensch-Maschine-Interaktion im Kontext der Industrie 4.0). Auf diese Weise lassen sich teilweise jedoch noch nicht alle Produktionsvorgänge vollständig ohne menschliches Zutun abbilden. Um trotzdem die Effizienz zu steigern, können in diesem Fall aber Teilaufgaben durch Roboter übernommen werden. Bei dieser Mensch-Roboter-Kollaboration können jedoch komplett neue Problemstellungen und Fragen auftreten, die in genauer Form im Voraus nicht absehbar sind. Damit solche Probleme nicht erst später im produktiven Betrieb auffallen, ist es vorteilhaft, solche industriellen Aufbauten virtuell nachzubauen, um diese in VR zu betrachten und konkrete Arbeitsabläufe zu simulieren.

In dieser VR-Simulation kann dann bereits im Voraus ein Eindruck der kooperativen

Arbeit zwischen Mensch und Maschine gewonnen werden. So werden schon in der Planungsphase menschliche Probleme, aber auch Probleme mit dem Roboter sichtbar. Sollte sich beispielsweise ein Mensch durch einen Roboter bedroht fühlen, wenn dieser dem Menschen zu nahe kommt, dann können auf Basis dieser Informationen Entscheidungen für den späteren Einsatz getroffen werden, um Arbeitsabläufe zu optimieren.

Zum Start des Projektes wurden zunächst Rahmenbedingungen und Werkzeuge festgelegt. Eines dieser Werkzeuge ist die Game-Engine, auf der die Anwendung basiert. Es wurde sich dabei für die Verwendung von Unreal Engine 4[1] entschieden, da diese mit der Bereitstellung umfangreicher Funktionen und Bausteine sowie einer relativ simplen Bedienung per Oberfläche die Entwicklung vereinfacht und zugänglicher macht. Als VR-Headset wird die HTC Vive Pro Eye[2] verwendet. Eine Betrachtung des ganzen Körpers ermöglicht die Azure Kinect[3], welche unter Einsatz von AI mehrere Personen unterscheiden und tracken kann. Zunächst werden damit lediglich Personen in der Anwendung als Skelett oder farbige Punktwolke dargestellt. Zukünftig sind diese Daten unter anderem für die Auswertung des Verhaltens relevant.

Ein weiterer Aspekt des Projektes, den es zu betrachten gilt, ist die Bedienung des Roboters. Die Forschung an Mensch-Roboter-Kollaboration erfordert ein realitätsnahes Verhalten und eine einfache Steuerung. Echte Roboter besitzen dazu eine Steuereinheit, welche Befehle unter Berücksichtigung physikalischer Gegebenheiten in Bewegungen umsetzen. Relevant sind hier beispielsweise Gewicht, Beschleunigung und Winkelpositionen der Gelenke. Es ist schnell zu erkennen, dass die Entwicklung einer eigenen Steuerung nicht sinvoll ist, da sie komplex und Roboter-spezifisch ist. Glücklicherweise gibt es Anwendungen für Roboter, die dazu dienen, Produktionsabläufe zu entwickeln und zu testen. In diesem Projekt wurde mit einem Roboter der Marke ABB gearbeitet, welche die Entwicklungsumgebung RobotStudio[4] verwenden. Die Simulation des Produktionsablaufs enthält hier eine Emulation der Steuereinheit, bei der die resultierenden Winkelpositionen ausgelesen werden können. Weiter wurde NX[5], ein CAD/CAM/CAE-System von Simens, für einen weiteren Roboter angebunden, um die Modularität der entwickelten Lösung zu demonstrieren.

In den folgenden Abschnitten wird zunächst auf einige technische Grundlagen, besonders dem Import von Roboter-CAD-Modellen in die Anwendung eingegangen (Abschnitt 2). Weiter wird für die Simulation der Roboter die Anbindung von Enterprise-Software per OPC UA erläutert (Abschnitt 3). In Abschnitt 4 wird die Umsetzung der Mensch-Maschine-Interaktion mit VR besprochen. Abschließend wird ein Fazit zu bisherigen Ergebnissen sowie ein Ausblick auf zukünftige Ziele gegeben.

[1] `https://www.unrealengine.com/en-US/`
[2] `https://www.vive.com/eu/product/vive-pro-eye/overview/`
[3] `https://azure.microsoft.com/en-us/services/kinect-dk/`
[4] `https://new.abb.com/products/robotics/en/robotstudio`
[5] `https://www.plm.automation.siemens.com/global/en/products/nx/`

2 Technische Grundlagen

Dieser Abschnitt stellt die Basis der Anwendung dar. Dabei werden die folgenden Punkte beschrieben:

- Der Import eines Computer-Aided Design Modells in Unreal Engine 4
- Die Steuerung eines Roboters per Tastatur
- Die Steuerung eines Roboterwagens

Die bereitgestellte Software ist im Hinblick auf das Robotermodell ABB IRB-4600[6] optimiert. Dabei steht ebenfalls ein Roboterwagen per CAD-Modell bereit, welcher es dem Roboter ermöglicht, eine Bewegung zu einem entfernten Ort durchzuführen.

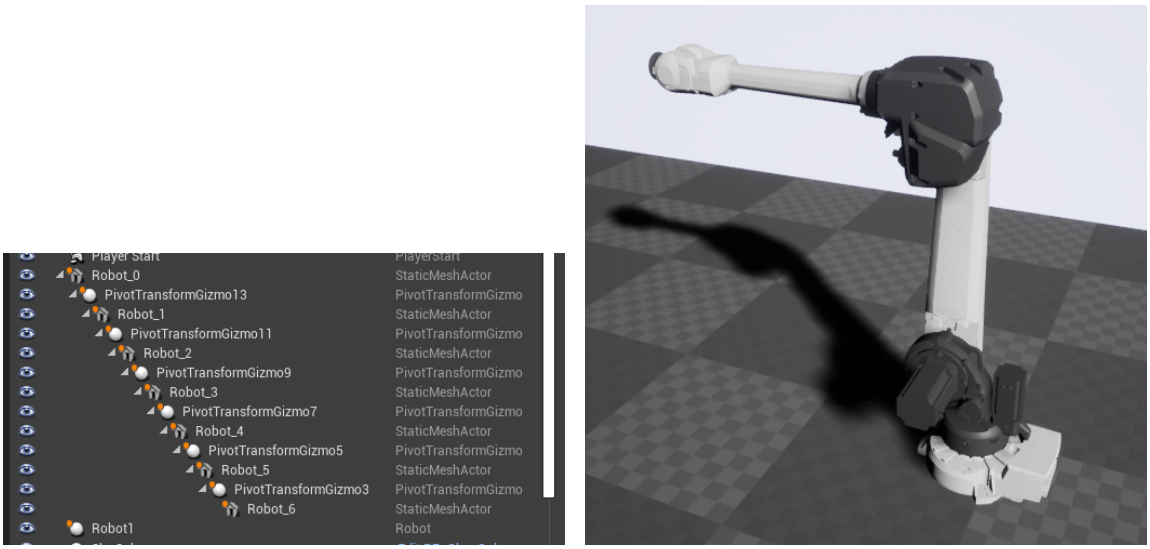
2.1 Import des CAD-Modells in Unreal Engine 4

Der Import eines Computer Aided Design (kurz: CAD) Modells stellt eine wichtige Basis für die Nutzung der Anwendung dar. Die vorhandene Softwarelösung bietet die Möglichkeit, Roboter als Gesamtkonstruktion in das Projekt zu importieren. Dabei bestehen diese Roboter aus mehreren Komponenten. Darunter sind die einzelnen Bauteile als grafische Darstellungen, die Nullstellen der Bauteile und die Verbindungspunkte. Die Nullstellen werden benötigt, damit die Rotationen und Translationen erfolgreich durchgeführt werden können. Die Verbindungspunkte sorgen dafür, dass die Bauteile zusammengeschraubt sind und die Rotationen oder Translationen an die weiteren Komponenten übergeben werden. Da bei dem ersten Import des Modells die Nullpunkte und die Ausrichtung der Bauteile nicht notwendigerweise korrekt sind, werden zunächst zum Ausgleich der fehlenden Information Pivotknoten verwendet. Hierbei werden die einzelnen Komponenten zunächst richtig ausgerichtet und richtig positioniert. Die Pivotknoten bestehen aus einer Rotationsmatrix und einem Translationsvektor. Wird beides in der homogenen Schreibweise geschrieben und anschließend verknüpft, so entsteht beispielsweise die folgende Matrix, die eine Rotation um die z-Achse sowie eine Translation zu einem Punkt (x, y, z) beschreibt:

$$\begin{pmatrix} \cos(\varphi) & -\sin(\varphi) & 0 & x \\ \sin(\varphi) & \cos(\varphi) & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Die Pivotknoten in Unreal Engine haben eine ähnliche Form und können um weitere Rotationen um die x- und y-Achse erweitert werden. Die Teile sind jedoch noch nicht miteinander verbunden, deshalb müssen anschließend sogenannte Sockets erstellt werden (vgl. [Gam21] für die Erstellung und Verwendung von Sockets). Diese verbinden die einzelnen Komponenten miteinander und sind für die Ausführung von verknüpften Rotationen des Roboters

[6] `https://new.abb.com/products/robotics/de/industrieroboter/irb-4600`



(a) Hierarchie (b) Roboter

Abbildung 1: Importieren des ABB IRB-4600 in Unreal Engine

notwendig. Das bedeutet, dass sich das gesamte Robotermodell entsprechend mitrotiert, falls beispielsweise eine Rotation über die Bodenplatte stattfindet. Dieses Vorgehen begünstigt den allgemeinen Import von Robotermodellen jeglicher Art. Weiterhin entsteht im Laufe des Importprozesses eine Hierarchie. Im Fall des ABB IRB-4600 sieht diese so aus, wie es Abbildung 1 zeigt.

Man erkennt, dass die Hierarchie in Abbildung 1 sehr einfach gehalten ist, indem die Bodenplatte (Robot_0) ganz oben Einfluss auf jedes andere Bauteil im Modell hat. Weiter absteigend nimmt die Abhängigkeit immer mehr ab. Das Werkzeug am Roboterarm (Robot_6) ist beispielsweise unabhängig von anderen Bauteilen. Dazwischen erkennt man die besagten Pivotknoten.

2.2 Tastatursteuerung

Um einfache Bewegungsabläufe zu visualisieren und zu testen, ohne notwendigerweise die Anbindung an eine Enterprise-Software wie ABB RobotStudio oder Siemens NX zur Simulation zu benötigen, kann ein importierter Roboter manuell gesteuert werden. Diese manuelle Tastatursteuerung erfolgt über die individuelle Bewegung der entsprechenden Gelenke. Hierbei ist jedem Gelenk jeweils ein Tastenpaar zur Rotation in beide mögliche Richtungen zugeordnet.

Jeder Gelenkskomponente können eigene Werte für Rotationseigenschaften zugewiesen werden. Diese Eigenschaften sind die *maximale Achsengeschwindigkeit* in Grad pro Sekunde, die *Beschleunigung* und die *Arbeitsbereiche*, welche die maximal und minimal mögliche Rotation in Grad darstellen. Um natürlich zu wirken, wird bei der Bewegung einer Achse eine Beschleunigung auf das entsprechende Gelenk ausgeübt. Um diese zu simulieren, nähert sich die Winkelgeschwindigkeit $\omega(t)$ der Achsen mit einer definierbaren konstanten

Winkelbeschleunigung α an die Maximalgeschwindigkeit ω_{max} an:

$$\omega(t) = \min(\omega(t - \Delta t) + \alpha \cdot \Delta t,\ \omega_{max}).$$

Dabei ist $\omega(t)$ die Geschwindigkeit zum aktuellen Zeitpunkt t. Δt ist die vergangene Zeit seit der Berechnung von $\omega(t - \Delta t)$, der vorherigen Berechnung von $\omega(t)$. Hört die Bewegung eines Gelenks in eine Richtung auf, wird die Berechnung mit einer negativen Beschleunigung durchgeführt, bis die Winkelgeschwindigkeit 0°/s beträgt, um das Abbremsverhalten darzustellen.

Die vorwärtskinematische Berechnung von Folgegliedern in der Gelenkshierarchie bei der Bewegung einer untergeordneten Komponente wird von der Engine übernommen. Zur Darstellung wird lediglich bei jedem Tick für jede Komponente die lokale Rotation aktualisiert.

2.3 Wagen

Neben der manuellen Steuerung von importierten Robotern ist es auch möglich, einen Wagen mit vier nicht-schwenkbaren Rädern manuell zu steuern. Dieser kann als Basis eines beliebigen Roboters verwendet werden, wobei der Roboter an die Wurzelkomponente (Wagen) angebunden wird.

Die Bewegung des Wagens verläuft über einen Differenzialantrieb (vgl. [Ron15] für eine Darstellung der Kinematik von mobilen Robotern). Dabei können die Räder auf beiden Seiten des Wagens verschieden schnell und in unterschiedlichen Richtungen rotiert werden. Das Resultat ist eine entsprechende Drehung des Wagens. Um aus den Geschwindigkeiten der verschiedenen Seiten eine korrekte Drehung des Wagens abzubilden, werden zur physikalischen Simulation manuell bei jedem Tick Rotationspunkt und Rotationswinkel berechnet.

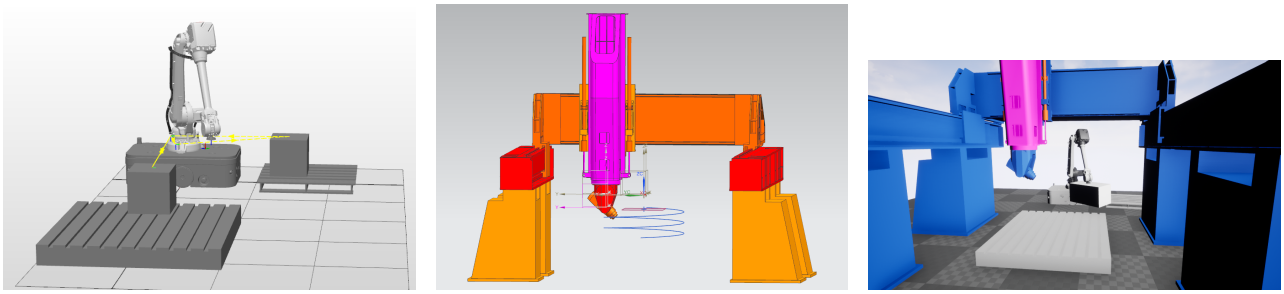
3 Anbindung an Enterprise-Software

Zur Simulation der Bewegungsabläufe von Maschinen und Robotern werden in der Industrie verschiedenste Enterprise-Anwendungen genutzt. Unsere Anwendung soll auf diesen etablierten Software-Lösungen zur Maschinensimulation aufbauen und eine zusätzliche Visualisierung in *Unreal Engine* bereitstellen. Eines der Hauptziele ist damit eine einheitliche Kommunikation zwischen unserer Lösung und gängigen Simulationsprogrammen zu schaffen.

Im Folgenden werden insbesondere *ABB RobotStudio*[7] und *Siemens NX*[8] betrachtet. Diese übernehmen Berechnungen der Bewegungsabläufe der Maschinen (gegebenenfalls auch unter Betrachtung von weiteren Randbedingungen wie etwa dem Umfahren von Hindernissen). Sowohl Positionsdaten der einzelnen Gelenke des Roboters als auch deren Ausrichtung werden dann regelmäßig über das Netzwerk an unsere Lösung übertragen. Dort werden diese Daten entsprechend gesetzt, um eine 1-zu-1 Abbildung der Maschinenbewegung in Unreal-Engine zu erhalten. Abbildung 2 zeigt dieses Verhalten in Aktion.

[7] `https://new.abb.com/products/robotics/robotstudio`

[8] `https://www.plm.automation.siemens.com/global/en/products/nx/`



(a) ABB RobotStudio (b) Siemens NX (c) Unreal Engine

Abbildung 2: Bewegungsabläufe von RobotStudio (a) und NX (b) in Unreal Engine (c)
Eine Live-Demo ist unter `https://youtu.be/Cz2i3gZp_pM` einsehbar.

Zentrale Komponente ist dabei das Netzwerkprotokoll, welches zur Übertragung der Daten dient. Dabei sollte möglichst keine neue eigene Protokollarchitektur definiert werden. Stattdessen wird auf eine bereits bestehende Protokollarchitektur der *Open Platform Communications Foundation (OPC Foundation)*[9] gesetzt: *OPC Unified Architecture (OPC UA)*[10].

3.1 OPC UA

OPC UA ist ein TCP/IP Netzwerkprotokoll zur Kommunikation zwischen Maschinen. Es wurde ursprünglich zur industriellen Automatisierung von Maschinen und Robotern entwickelt (vergleiche [LM06] für eine Diskussion verschiedener Einsatzgebiete von OPC UA in industriellen Anwendungen). Dementsprechend hat sich OPC UA zum de-facto Standardprotokoll für industrielle Maschinenkommunikation entwickelt und wird demzufolge von einem Großteil aller modernen Anwendungen im Bereich des industriellen Maschinenbaus unterstützt. Darunter fallen auch die von uns betrachteten Programme ABB RobotStudio und Siemens NX.

Das Kommunikationsmodell bei OPC UA basiert auf einer Client-Server-Architektur. Konkret werden die Gerätedaten von einem OPC-Server bereitgestellt und von einem OPC-Client konsumiert. OPC UA bietet Funktionalitäten zum Durchsuchen von hierarchischen Namespaces, welche konkrete Messdaten (Positionsdaten, Winkel, Temperaturen o. ä.) der Maschine enthalten, und ermöglicht es Clients, diese Daten zu lesen und zu schreiben und sie gegebenenfalls auch auf Datenänderungen zu überwachen.

OPC UA unterstützt zwei grundlegende Protokollarten: Ein Binärprotokoll und ein Protokoll für Web-Services basierend auf *SOAP*. Das binäre Protokoll bietet die beste Leistung und den geringsten Overhead, da es minimale Ressourcen benötigt. Daher basiert die hier vorgestellte Lösung auf dem Binärprotokoll von OPC UA. Außerdem unterstützt OPC UA verschiedene Sicherheitseinstellung: Unverschlüsselt, signiert und verschlüsselt.

[9] `https://opcfoundation.org/`
[10] `https://opcfoundation.org/about/opc-technologies/opc-ua/`

Für OPC UA stehen viele verschiedene APIs in allen gängigen Programmiersprachen zu Verfügung. Aufgrund der Tatsache, dass in Unreal-Engine entwickelt wird, bietet sich eine C++ API für die Programmierung eines OPC-UA-Clients an. Dementsprechend wurde die Open-Source C/C++ API des Projektes *open62541*[11] gewählt.

3.2 ABB RobotStudio

ABB RobotStudio ist eine Simulations- und Offline-Programmiersoftware für Roboter und Maschinen des Herstellers ABB. Die Software ermöglicht die Programmierung von Robotern am PC, ohne dass dafür Produktivsysteme heruntergefahren oder unterbrochen werden müssen. Somit lassen sich Aufgaben wie Training, Programmierung und Optimierung der Roboter durchführen, ohne dass dabei die Produktion gestört wird.

ABB RobotStudio unterstützt mehrere verschiedene (auch virtuelle) Controller des Herstellers ABB. Im Mittelpunkt zur Kommunikation über OPC UA steht dabei der *Industrial Robot Controller 5 (IRC5)*[12]. Dieser Controller unterstützt OPC UA und ABB stellt mit dem *IRC5 OPC UA Server*[13] eine hauseigene OPC-UA-Serversoftware zur Verfügung, welche eine einfache grafische Konfiguration eines OPC-UA-Servers ermöglicht.

In der IRC5 OPC-UA-Server-Software lassen sich somit neue (virtuelle oder reale) Controller anlegen, welche dann eine eigene Oberkategorie in der OPC-UA-Hierarchie bilden. So lassen sich die Controller in der von ABB RobotStudio konkret erstellten Hierarchie unter dem Punkt *DeviceSet* finden. Jeder Controller verfügt dann weiterhin über sogenannte *MotionDevices*, welche den Robotern bzw. den Maschinen entsprechen. Diese umfassen dann jeweils entsprechende Messdaten wie Winkel und Positionen der einzelnen Gelenke.

Abbildung 3 zeigt eine Visualisierung der OPC-UA-Hierarchie eines Roboters in ABB RobotStudio an.

3.3 Siemens NX

NX ist eine umfangreiche *CAD/CAM/CAE*-Software der Firma Siemens. Die Software ermöglicht die Modellierung und das Designen von Oberflächen und geometrischen Körpern, die Analyse von mechanischen Vorgängen und unterstützt in der Produktion von Maschinen. Die Software hat als CAD/CAM/CAE-Lösung damit einen größeren Funktionsumfang als ABB RobotStudio, bietet jedoch auch die Möglichkeit, Bewegungsvorgänge von Robotern und Maschinen innerhalb der Software zu simulieren.

Da Siemens NX von Haus aus keine OPC-UA-Server-Software anbietet, wurde ein eigener rudimentärer OPC-UA-Server geschrieben. Dazu wurde die Programmiersprache Python genutzt mit einer offenen OPC-UA-Implementation des *FreeOpcUa*[14] Projektes: *opcua-asyncio*[15]. Diese beinhaltet High-Level Abstraktionen für sowohl OPC-UA-Clients als auch

[11] `https://open62541.org/`
[12] `https://new.abb.com/products/robotics/controllers/irc5`
[13] `https://new.abb.com/products/robotics/controllers/irc5/irc5-options/opc-ua`
[14] `https://freeopcua.github.io/`
[15] `https://github.com/FreeOpcUa/opcua-asyncio`

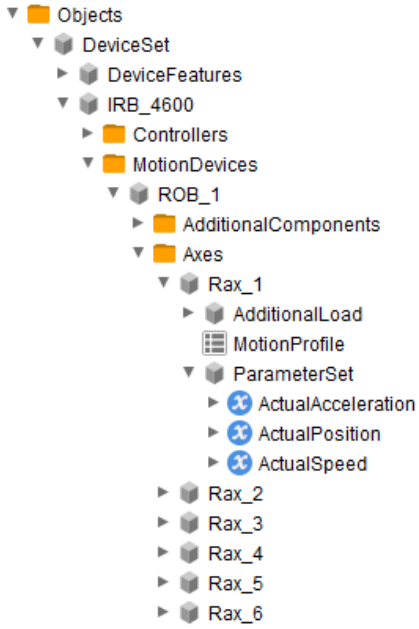


Abbildung 3: OPC-UA-Hierarchie in ABB RobotStudio

OPC-UA-Server, welche direkt vom Anwendungsprogrammierer genutzt werden können oder bei Bedarf erweitert werden können.

Damit wurde ein simpler OPC-UA-Server für Siemens NX entwickelt, in dem sich Variablen (beispielsweise die Position eines Robotergelenks) mit Namen, Datentyp und weiteren Informationen definieren lassen, welche dann vom OPC-UA-Client (in diesem Fall Siemens NX) geschrieben und gelesen werden können. Aus diesen Variablen ergibt sich dann die OPC-UA-Hierarchie des Servers.

Innerhalb von Siemens NX lassen sich Signale anlegen, welche sich mit Laufzeitparametern eines Physik-Objektes in Siemens NX verbinden lassen (beispielsweise die x-Koordinate des Masseschwerpunktes eines Robotergelenks). Wird der OPC-UA-Server mit Siemens NX verbunden, so lassen sich Zuordnungen zwischen den internen Signalen von Siemens NX (welche sich auf ein Physik-Objekt beziehen) und den externen Signalen des Servers (die Variablen, die dort angelegt wurden) bilden. Somit lässt sich eine entsprechende 1-zu-1 Zuordnung der Signale innerhalb von Siemens NX konfigurieren, womit dann die aktuellen Daten der Laufzeitparameter des Physik-Objektes in die Variablen des Servers geschrieben werden. Die Ausprägungen dieser Variablen werden wiederum vom OPC-UA-Client unserer Unreal-Engine Lösung herangezogen, um die Bewegung des Physik-Objektes innerhalb der Unreal-Engine darzustellen.

3.4 Unreal-Engine-4-Plugin

Das Unreal-Engine-4-Plugin stellt die zentrale Basis der entwickelten Softwarelösung dar. Hierbei vereint es die notwendigen Komponenten zum Anbinden der Modelle, die im vorherigen Abschnitt importiert wurden, sowie die Anbindung an OPC UA. Nachdem das Modell

entsprechend Abschnitt 2 importiert wurde, kann jede Einzelkomponente mit einer sogenannten *OPC Movement Component* ausgestattet werden. Diese Komponente ermöglicht es Bewegung von ABB RobotStudio oder Siemens NX MCD durch einen Server zu übertragen. Dabei muss in jeder Komponente angegeben werden, auf welchen Pfad im OPC-UA-Server zugegriffen werden soll. Ebenfalls muss mithilfe eines Multipliers die Richtung der Bewegung festgelegt werden und durch ein Offset muss bestimmt werden, ob die Bewegung örtlich versetzt stattfinden soll.

Nachdem diese Komponente einzeln konfiguriert worden sind, müssen diese mit einer weiteren selbstentwickelten Softwarekomponente verbunden werden. Dabei handelt es sich um den sogenannten OPC-UA-Client-Actor. Diese Komponente wird in der Welt von Unreal Engine 4 instanziiert und anschließend konfiguriert. Dieser OPC-UA-Client-Actor wurde entwickelt, damit nicht jede einzelne Komponente auf den OPC-UA-Server zugreifen muss und der Server nicht unnötig belastet wird. Die Konfiguration beinhaltet dabei die Adresse des Servers, zu welchem sich der Client Actor verbinden soll. Ebenfalls ist die Angabe des Security Modes notwendig. Die Software beinhaltet darüber hinaus für den Zugriff auf eine signierte oder verschlüsselte Quelle ein Zertifikat, welches diesen bei dem jeweiligen Server verifiziert.

Im Allgemeinen stellt das Plugin die Bewegungen von Roboter und Roboterwagen bereit sowie die OPC UA Komponenten und die weiteren Komponenten für die virtuelle Realität.

4 Mensch-Roboter-Kollaboration in der VR

Nachdem bereits in den vorhergehenden Abschnitten auf die Maschinen-Komponente eingegangen wurde und nun die vollständige Darstellung und realistische Simulation einer Maschine möglich ist, fehlt für die Mensch-Machine-Interaktion noch der Faktor Mensch.

Um diesen einen Teil der Simulation werden zu lassen, wurden zwei verschiedene Ansätze verfolgt, die sich letztendlich kombinieren lassen und im Folgenden erläutert werden.

4.1 Virtual-Reality

Der offensichtlichste Schritt, um zumindest passiv ein Teil der Simulation zu werden, ist die Integration von VR, sodass Arbeitsabläufe aus einer realistischen Perspektive betrachtet werden können, als sei man direkt in der Fertigung vor Ort.

Die Integration selber in Unreal Engine ist sehr komfortabel und erfolgt durch eine Reihe von Plugins, die für die verschiedenen VR-Hersteller zur Verfügung stehen. Da im Verlauf dieses Projektes nur eine HTC Vive Pro verwendet wurde, nutzt das Projekt die SteamVR-Integration. Des Weiteren muss am Level konfiguriert werden, wie das Verhältnis der Unreal Größeneinheit zu den Maßeinheiten der realen Welt ist, damit die relative Größe zu einem Menschen in VR auch realistisch wirkt.

Zur Steuerung in VR gibt es auch wieder verschiedene Möglichkeiten. So kann zum einen die Bewegung über Controller erfolgen, zum anderen kann aber auch im konkreten Beispiel

mit der HTC Vive Pro die VR-Brille im Raum durch Laser lokalisiert werden, sodass sich eine Person dort frei bewegen kann und diese Bewegungen dann auch cm-genau in der virtuellen Welt abgebildet werden. Im Projekt wird beides unterstützt, sodass eine Person sich im Raum frei bewegen und zusätzlich den Bereich in der Simulation teleportieren kann. Notwendig ist dies, da der Bereich, in dem sich in der realen Welt bewegt werden kann, oftmals nicht groß genug ist, um den vollständigen virtuellen Raum abzudecken. Auch diese Umsetzung ist in Unreal Engine sehr einfach, da es dafür bereits fertige Komponenten gibt, die per Drag & Drop ins Projekt übernommen werden können.

4.2 Microsoft Azure Kinect

Das Ziel des Projektes ist die Analyse von Umgang und Verhalten mit und um Roboter. Dazu ist eine detaillierte Betrachtung des ganzen Körpers notwendig und erfolgt durch die *Azure Kinect*. Diese bietet die Möglichkeit, einzelne Personen in 3D zu tracken und Positionsdaten der Körperteile zu ermitteln. Sie verfügt für diesen Zweck über eine *RGB-Kamera* und eine *Tiefenkamera* als Sensoren. Die Verarbeitung der Daten erfolgt mittels des *Azure Kinect Body Tracking SDK*, welches *Deep-Neural-Networks* verwendet, um Personen über mehrere Bilder hinweg eindeutig zu identifizieren und die Position und Orientierung wichtiger Körperteile zu extrahieren.

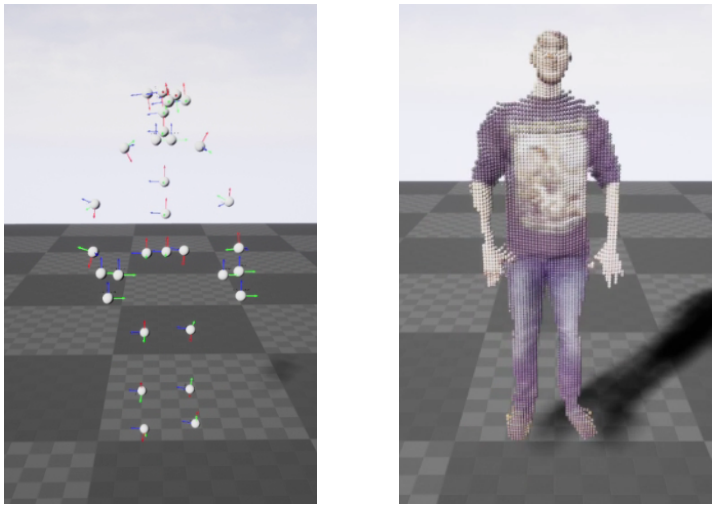
Für eine erste Visualisierung wird das *Azure Kinect Unreal Plugin*[16] verwendet, welches die erfassten Positionen als *Skelett* darstellt. Eigentliches Ziel ist die detaillierte Darstellung des ganzen Körpers unter Verwendung der Tiefen- und RGB-Daten. Mit diesen ist es möglich, eine *Punktwolke* zu erzeugen, die Form und Farben der Person aus Sicht der Kinect abbildet. Problem ist bei dieser Aufgabe die reine Anzahl der Punkte, welche in den Tausenden liegt. Diese Anzahl von Punkten mit individuellen Static Meshes zu erzeugen und bewegen erfordert zu viel Rechenleistung, um eine brauchbare Lösung zu bieten. Eine erste Idee war der Gebrauch von Partikeleffekten, nach einiger Recherche wurde sich jedoch für die *Hierarchical Instanced Static Mesh Component* (HISMC) entschieden. Die HISMC bietet die Möglichkeit, performant große Mengen gleicher Static Meshes darzustellen. Ein angegebenes Mesh wird einmal instanziiert und mit einer Reihe von Transformationen gerendert. Diese Transformationen können zur Laufzeit hinzugefügt, entfernt oder angepasst werden. Mit diesem HISMC ist es nun möglich, die Punktwolke darzustellen, jedoch zunächst ohne Farbe.

Ein HISMC verfügt weiter über die Option, eine Anzahl von Float-Werten pro Instanz zur Verfügung zu stellen. Wird dem HISMC ein Material zugewiesen, kann hier auf die Werte zugegriffen werden. Daher wurde ein Material erstellt, welches die drei Float-Werte als RGB-Werte interpretiert.

Ein letztes Problem, das beim Testen auffiel, ist die Punktdichte der Tiefenkamera. Aufgrund der Inverse-Square Law nimmt die Menge der verfügbaren Datenpunkte einer Person ab, wenn sich diese von der Kinect entfernt. Um dem entgegenzuwirken, werden die einzelnen

[16] `https://github.com/secretlocation/azure-kinect-unreal`

Punkte in Abhängigkeit vom Abstand zur Kinect skaliert.



(a) Skelett (b) Punktwolke

Abbildung 4: Darstellung einer Person durch die Kinect

5 Fazit und Ausblick

Im Rahmen des Projektes entstand eine prototypische Lösung zur Visualisierung von Mensch-Roboter-Kollaboration in der virtuellen Realität. Dabei wurde mithilfe des Machine-to-Machine-Protokolls OPC UA eine Anbindung an die praxisrelevanten Enterprise-Anwendungen ABB RobotStudio und Siemens NX geschaffen. Um eine für den Benutzer möglichst interaktive Erfahrung zu bieten, wird durch die Microsoft Azure Kinect zusätzlich zu den Maschinen auch ein Abbild des Bedieners in Form von einer farbigen Punktwolke oder eines Skelettes angezeigt. Alternativ zu der automatisierten Steuerung der Maschinen durch Anbindung an eine externe Anwendung kann der Anwender auch eine frei konfigurierbare manuelle Steuerung per Tastatur verwenden.

Um dem User das Importieren und Konfigurieren von beliebig komplexen Maschinenstrukturen zu ermöglichen, wird dieser sowohl bei dem Importieren des CAD-Modells als auch bei der Festlegung der Robotergelenke von unserer Lösung unterstützt. Dementsprechend lassen sich die einzelnen Komponenten der Gesamtstruktur hierarchisch anordnen. Gegebenenfalls können Position und Ausrichtung einer Komponente mithilfe von Pivotknoten korrigiert werden. Außerdem ermöglicht eine designierte OPC-Movement-Component eine individuelle Konfiguration der Bewegung von jeder Komponente.

Nachdem im Laufe des Projektes die Grundlagen für eine Lösung zur Visualisierung der Mensch-Roboter-Kollaboration gelegt wurden, kann diese Basis in Zukunft auch noch um eine Reihe von Erweiterungen vervollständigt werden.

Da große Industrieroboter bei nahem Kontakt mit einem Menschen auch durchaus eine Gefahr darstellen können, gibt es Sicherheitseinrichtungen, die beispielsweise durch Radar-

technologie eine Person erkennen können und den Roboter dann anhalten. Da durch die Kinect die reale Position einer Person ermittelt werden kann, könnten solche Sicherheitseinrichtungen auch in der simulierten Umgebung abgebildet werden und ein Signal durch OPC zurück an die Industrieapplikation gemeldet werden.

Ein oft genannter Wunsch ist weiter die Aufnahme von Bewegungsabläufen der Maschine. In der aktuellen Version werden die Bewegungen ausschließlich durch externe Anwendungen vorgegeben. Diese aufeinander abgestimmten Bewegungen könnten aber zur Laufzeit auch aufgezeichnet und dann später ohne die externen Anwendungen wieder abgespielt werden.

Acknowledgement

Diese Arbeit wurde teilweise aus zentralen Qualitätsverbesserungsmitteln der Westfälischen Hochschule finanziert und als Kollaboration der Fachbereiche "Wirtschaft und Informationstechnik" und "Maschinenbau" am Standort Bocholt im Projekt "Interaktiver VR-Einsatz in der CAD-Planung (VR in CAD)" durchgeführt.

Literatur

[Gam21] Games, Epic: *Setting Up and Using Sockets With Static Meshes.* `https://docs.unrealengine.com/en-US/WorkingWithContent/Types/StaticMeshes/HowTo/Sockets/index.html`, 2021.

[LM06] Leitner, Stefan-Helmut und Wolfgang Mahnke: *OPC UA–service-oriented architecture for industrial applications.* ABB Corporate Research Center, 48:61–66, 2006.

[NFM20] Nardo, M Di, D Forino und T Murino: *The evolution of man–machine interaction: The role of human in Industry 4.0 paradigm.* Production & Manufacturing Research, 8(1):20–34, 2020.

[Ron15] Ronsse, Renaud (École polytechnique de Louvain): *Mobile Robot Kinematics and Control.* `https://perso.uclouvain.be/renaud.ronsse/BEST2015/2015_slides_L2_2pages.pdf`, Juli 2015.