HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

PRAKTIKUMSBERICHT

Herr
Vladislav Förster

Eignung der HoloLens für die forensische Fallarbeit

2021

Fakultät **Computer- und Biowissenschaften**

PRAKTIKUMSBERICHT

Eignung der HoloLens für die forensische Fallarbeit

Autor:
Vladislav Förster

Studiengang:
Allgemeine und Digitale Forensik

Seminargruppe:
FO16w3

Erstprüfer:
Prof. Dr. rer. nat. Diek Labudde

Zweitprüfer:
Sven Becker, M.Sc.

Mittweida, Juni 2021

Bibliografische Angaben

Förster, Vladislav: Eignung der HoloLens für die forensische Fallarbeit, 16 Seiten, 5 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Computer- und Biowissenschaften

Praktikumsbericht, 2021

Referat

Die Arbeit soll Anwendungsmöglichkeiten der HoloLens in der Forensik näher betrachten. Dabei werden verschiedene Ideen, welche die forensische Arbeit effizienter oder präziser machen könnten, auf die Umsetzbarkeit geprüft. Als Ergebnis ging eine Unity Anwendung, die auf der HoloLens ausgeführt wird, und eine Python Konsolenanwendung, die auf einem externen Computer ausgeführt wird, hervor. Die Aufgabe der HoloLens ist es dabei Bilder aufzunehmen und an den externen Computer zu senden. Dieser speichert die Bilder und analysiert sie anschließend mit OpenPose.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

AR	Augmented-Reality
IMU	Inertiale Messeinheit
IR	Infrarot
TOF	Time of flight
UE4	Unreal Engine 4
VR	Virtual-Reality

1 Einleitung

1.1 Motivation

Die Forensik bedient sich vieler Neuerungen in der Technik um die Arbeit effizienter und präzieser durchzuführen. Mikroskopie wird benutzt um Spuren genauer zu untersuchen, DNA-Analyse hilft dabei Tätermerkmale zu bestimmen, Computersimulationen werden benutzt um Hypothesen auf ihre Plausibilität zu überprüfen und Virtual-Reality (VR) wird nun benutzt, um sich einen Überblick über einen Tatort in 3D verschaffen zu können. Eine Technik, mit der in immer mehr Bereichen experimentiert, ist Augmented-Reality (AR). Viele AR-Geräte verfügen nicht nur über Sensoren die RGB-Bilder produzieren, sondern beispielsweise auch Position, Lage oder Entfernung messen können. Aus diesen zusätzlichen Daten allein können bereits weitere Informationen gewonnen werden aber auch noch kombiniert werden. Um diese Informationen zu gewinnen können simple Berechnungen in Echtzeit auf dem Gerät durchgeführt werden, für komplexere Berechnungen aber, wie das Machine Learning, muss auf einen externen Computer zurückgegriffen werden.
Wie diese Technik auch im forensischen Bereich eine Anwendung finden könnte, soll diese Arbeit zeigen. In der Arbeit konnte die HoloLens 1 genutzt werden.

1.2 Problemstellung

In der Forensik ist eine korrekte Dokumentation besonders wichtig. Bei der Bilddokumentation von Tatorten aber auch der Verdächtigen benutzt man aktuell 3D-Scanner und Kameras [17]. Die Kameraaufnahmen dienen dazu, sich einen Überblick über den Tatort und die dort gefundenen Spuren zu verschaffen. Die Daten des 3D-Scanners werden benutzt um sich in VR am Tatort umzusehen und mögliche Szenarien aus unterschiedlichen Blickwinkeln anzusehen oder Simulationen durchzuführen. Bilder von den Tatverdächtigen können, mit Bildmaterial aus anderen Quellen, zur Gesichtserkennung oder Pose-Estimation benutzt werden.

1.3 Zielstellung

Aufgrund der verbauten Kameras kann man die HoloLens für die Bilddokumentation verwenden. Hinzu kommt das sie an das Netzwerk angeschlossen werden kann und Daten dadurch in kürzester Zeit auch an eine Datenbank übermittelt werden können, in dieser gespeichert, verarbeitet und die Ergebnisse zurück an die HoloLens geschickt werden können.

2 Theoretische Grundlagen

2.1 HoloLens

Die Hololens 1 war das erste AR-Gerät von Microsoft und hat sich an den Bereich Wissenschaft & Forschung gerichtet. Die Hololens 2 soll nun auch von Unternehmen genutzt werden. Um diesen Marktbereich anzusprechen, wurde die verbaute Hardware verbessert hat und ein Niveau erreicht hat, das Konsumenten und App Entwickler gewöhnt sind.

	HoloLens 1	HoloLens 2
Kamera	1408x792 Foto, 1216x684 24fps Video, TOF-Kamera	3904x2196 Foto, 1920x1080 30fps Video, TOF-Kamera, 2 IR-Kameras
Sichtbereich	34°	52°
Architektur	x86 32-Bit	ARM 64-Bit
CPU-Kerne	4	8
RAM	2GB	4GB
Sensoren	IMU, 4 Mikofone, 1 Lichtsensor	IMU (Accelerometer, Gyroskop, Magnetometer), 5 Mikrofone

Tabelle 2.1: Vergleich relevanter Kenndaten HoloLens 1 und HoloLens 2 [10, 11]

Ein sehr interessantes Feature ist das sogennante Spatial-Mapping. Dabei wird der Raum mit Hilfe der Kameras gescannt und das Gerät generiert ein Mesh, mit dem platzierte Hologramme interagieren können. [13].

Microsoft bietet mit Azure auch eine Möglichkeit verschiedenste Berechnungen in der Cloud auszuführen [9]. Hierfür eignen sich besonders Anwendungen des Maschinellen Lernens. Aufgrund der Datenschutzbeschränkungen wird in dieser Arbeit jedoch nicht weiter darauf eingegangen.

Die Entwicklung für die HoloLens kann mit Hilfe der Unity-Engine, Unreal-Engine, OpenXR oder WebXR erfolgen [12]. Die Unterstützung für die Unreal Engine und OpenXR ist erst mit dem Erscheinen der HoloLens 2 unterstützt worden [5, 8].
Den Einstieg in die Mixed-Reality (MR) Entwicklung erleichtert Microsfot mit dem ”Mixed Reality Toolkit” (MRTK). Dabei handelt es sich um eine Sammlung von Werkzeugen wie z.B. C# Scripten, grafische Bauelemente oder Shadern [14]. Dieses Toolkit ist als Repository auf der microsoft Github-Seite für Unity und Unreal verfügbar [3].

Eine Besonderheit in der Hardware der HoloLens ist die Holographic Processing Unit (HPU). Dabei handelt es sich um eine spezielle, von Microsoft entwickelte, Verarbeitungseinheit. In Abb 2.1 kann man sehen, wie alle Sensordaten an die HPU übermittelt

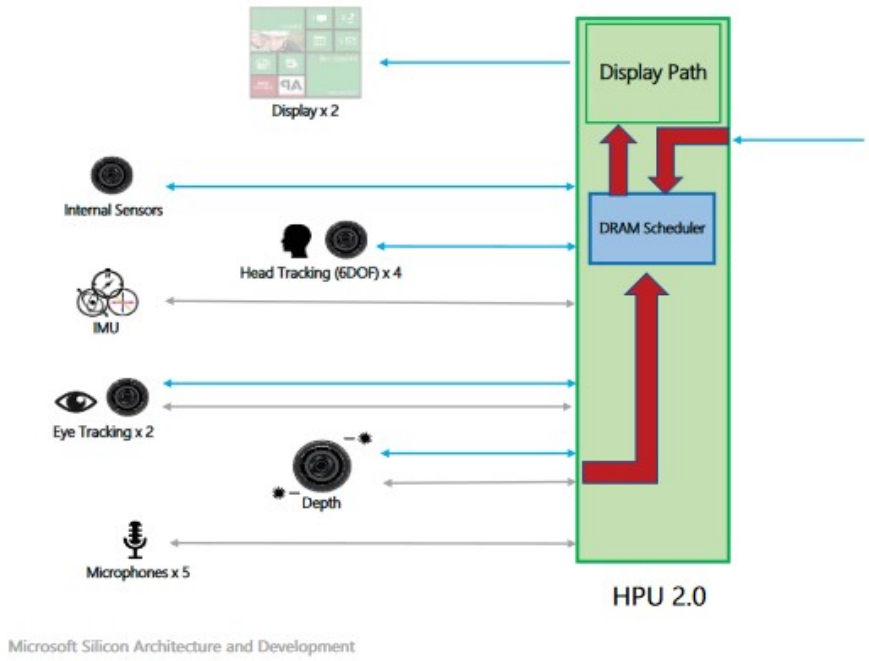


Abbildung 2.1: Verarbeitung der Sensordaten durch die HPU in der HoloLens 2 [18]

und verarbeitet werden, um die CPU zu entlasten [18].

2.2 Spatial Mapping für die Tatortvisualisierung

3D-Objekte werden als eine Menge von Punkten, die Dreiecke erzeugen, dargestellt und als *Mesh* bezeichnet. Früher wurden die Punkte, die notwendig sind um ein Objekte darzustellen, noch alle manuell hinzugefügt und an die richtige Position im 3D-Raum gesetz. Durch die Fortschritte in den bildgebenden Verfahren, kann man Objekte nun einfach mit Geräten scannen und automatisch ein Mesh erzeugen lassen. Damit kann man besonders detailreiche Objekte im 3D-Raum schnell und originalgetreu darstellen. In der Forensik ist das originalgetreue darstellen besonders wichtig. Aus diesem Grund werden 3D-Scanner von immer mehr Polizeistellen eingesetzt [17].
Um einen möglichst guten Scan zu erzeugen müssen die Lichtverhältnisse jedoch angepasst werden und weitere Tatortarbeit ist während des Scans nur bedingt möglich. Da die HoloLens auf dem Kopf sitzt und die Umgebung vor der tragenden Person aufnimmt, kann der Raum gescannt werden, während andere Tatortarbeiten ausgeführt werden.
In der Arbeit [7] wird ein Gebäude von innen mit der HoloLens gescannt um einen 3D-Gebäudeplan zu erzeugen. Das Ergebnis dieser Arbeit zeigt, dass Räume gut dargestellt werden können und die Punkte des erzeugten Mesh im Vergleich zu einer Punktwolke eines 3D-Scanner, im Durschnitt um 5 cm abweichen. Möbel oder etwaige klei-

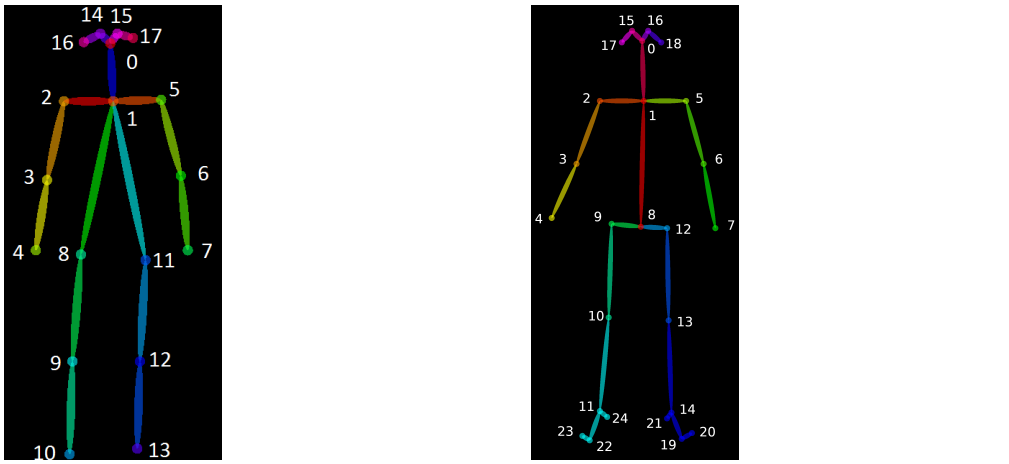
Abbildung 2.2: Ein von Hand modellierter Eimer, bestehend aus 2700 Dreicken [16]

nere Objekte werden in der genannten Arbeit jedoch nicht weiter betrachtet.
Die HoloLens kann maximal 2000 *TrianglesPerCubicMeter* erzeugen. Selbst bei der Annahme das alle Dreiecke einzigartige Punkte erhalten, ist die maximale Anzahl an Punkten in einem Kubikmeter nur 6000.

2.3 OpenPose

OpenPose ist ein System zur *Human Pose Estimation*. OpenPose führt die notwendigen Berechnungen am schnellsten auf einer Nvidia Grafikkarte mit CUDA Support aus, kann diese jedoch auch auf einer AMD Grafikkarte oder der CPU ausführen. Möchte man die CPU benutzen, werden mindestens 8 Kerne empfohlen [15]. Zur Programmierung werden eine C++ und Python-API als auch ein Unity-Plugin bereitgestellt.

Die *Pose Estimation* kann *Top-Down* oder *Bottom-Up* erfolgen. Bei der *Top-Down-Methode* wird eine Person in dem Eingabebild gesucht und anschließend werden die Keypoints für die gefundene Person gesucht. Dabei kann es Probleme geben, wenn die Person nur teilweise sichtbar ist. Die *Bottom-Up-Methode*, den OpenPose nutzt, sucht im Gegensatz dazu zuerst nach den Keypoints und nutzt anchließend *Part Affinity Fields*, um sie korrekt zu verbinden [1].
Bei der *Pose Estimation* kann man zwischen den Modellen *Body 25*, *COCO* oder *MPI* wählen. Die Standardimplementation nutzt das *Body 25* Modell und in der Dokumentation wird dazu geraten bei dieser, aufgrund der höheren Genauigkeit und Geschwindigkeit, zu bleiben [2]. Für GPUs mit 2GB VRAM wird empfohlen *COCO* zu benutzen, da es weniger Speicher benötigt als *Body 25*. *MPI* hat zwar die geringste Genauigkeit, benötigt aber auch die wenigsten Resourcen [4].



(a) Visualisation des COCO Models [20] (b) Visualisation des Body 25 Models [19]

Abbildung 2.3: Vergleich von COCO und Body 25

Die HoloLens besitzt eine 32-Bit Architekur. Einige Abhängigkeiten von OpenPose sind jedoch nur für 64-Bit Systeme verfügbar. Ein Beispiel für solch eine Abhängigkeit war *Caffe*. Diese benötigt Bibliotheken, die für das Maschinelle Lernen in OpenPose notwendig sind.
Bei näherer Betrachtung von OpenPose wurden als Mindestvoraussetzungen eine GPU mit mindestens 1,5 GB Speicher sowie CUDA und cuDNN genannt oder, im Falle das OpenPose ohne Verwendung der Grafikkarte ausgeführt werden soll, mindestens 2GB RAM und eine CPU mit mindestens 8 Kernen [15].

Diese Voraussetzungen führten dazu, dass OpenPose nicht auf der HoloLens ausgeführt werden konnte und auf einen externen Computer ausgewichen wurde. Für die Umsetzung auf dem Computer wurde sich für Python entschieden. Um die Python API von OpenPose verwenden zu können, müssen in der CMmake-GUI Variablen geändert werden.

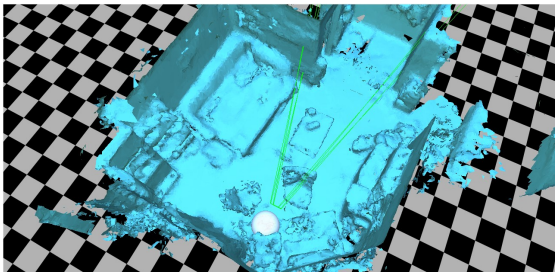
3 Methodische Vorgehensweise

Verwendet wurde Unity in der Version 2019.4.25f1 und die Unity-Pakete aus dem MRTK 2.6.1 . Diese Unity Version wurde gewählt, da es sich um eine LTS Version handelt die bis zum Sommer 2022 Updates erhält und diese auch mit der HoloLens 1 kompatibel ist. In der Unity 2020 LTS Version wurden Änderungen vorgenommen und machen sie inkompatibel mit der HoloLens 1. Das MRTK ist rückwärtskompatibel und kann auch in der neuesten Version mit der HoloLens 1 verwendet werden. Es muss lediglich darauf geachtet werden, das einige Feature nicht mit der HoloLens 1 verwendet werden können.

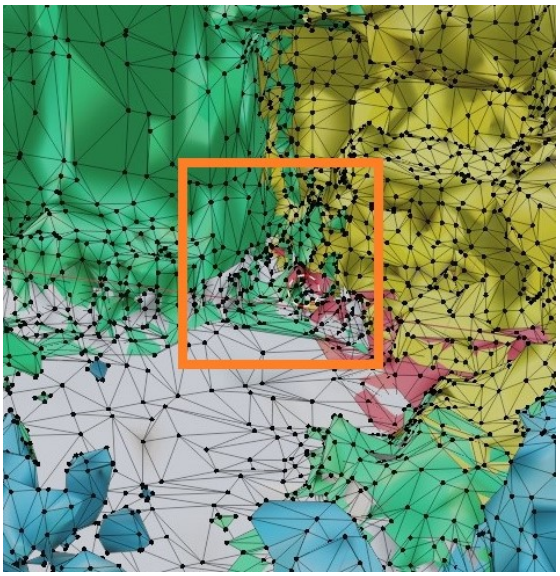
3.1 Spatial Mapping

Das Spatial-Mapping von Microsoft bietet, innerhalb des MRTK, eine Möglichkeit das *Level Of Detail* einzustellen. Dabei kann man zwischen verschiedenen voreingestellten Stufen, aber auch einen konkreten Wert für *TrianglesPerCubicMeter* setzen. 500 ergab den kleinsten und 2000 den größten Wert. Werte außerhalb dieses Intervalls ergaben keine sichtbare Änderung in dem erzeugten Mesh.
Um Einen Raum mit der Hololens einzuscannen wurde, mit Aufgesetzter HoloLens, durch den Raum gelaufen. Die Bilder in Abb. 3.1 sind aus einem kurzem drei minütigem laufen und umsehen im Raum entstanden. Keine Gegenstände wurden im genaueren angeschaut um die Scans zu verbessern, da solche Versuche einzeln durchgeführt wurden.

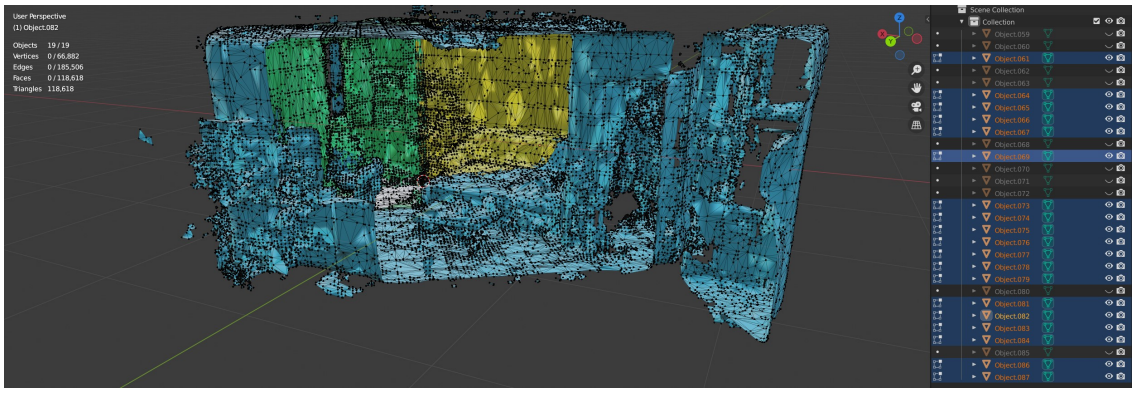
Um sich einen Eindruck über das Mesh eines 18m^2 gescannten Raumes zu verschaffen, wurde es als OBJ-Datei über das *Device Portal* heruntergeladen und in Blender importiert. Das gesamte Mesh besteht aus 29 Objekten, diese werden durch 84.184 Punkte bzw. 148.477 Dreiecke dargestellt. In Abb. (c) sind 19 Objekte sichtbar, diese sind im *Outliner* blau hervorgehoben und werdem mit 66.882 Punkten bzw. 118.618 Dreicken dargestellt. Zehn Objekte wurden zum besseren Verständnis ausgeblendet.
In Abb. 3.1 (b) kann man sehen wie sich vier verschiedene Objekte überschneiden und viele redundante Punkte erzeugen. Diese überschneidenden Flächen haben einen zum Teil stark abweichenden Normalenvektor und bereiten beim Kontakt mit einem bewegenden Objekt Schwierigkeiten eine realistische Simulation zu erzeugen. Um dieses Problem zu lösen, müssen Nachbearbeitungsschritte durchgeführt werden um die überschneidenden Flächen zu entfernen.
Eine Möglichkeit wäre es diese in Blender durchzuführen. Zuerst müssen alle Objekte zusammengeführt werden. Anschließend können die problematischen Punkte im Mesh markiert werden und der Operator *Merge by Distance* verwendet werden. Dieser Operator bestimmt den Abstand der Punkte zueinander und führt diese - sollte der berechnete



(a) Anzeige der Spatial Map für einen Raum über das Device Portal



(b) Redundante Punkte in der Spatial Map



(c) Mesh aus Abb. (a), importiert in Blender

Abbildung 3.1

Wert kleiner sein als der vom Nutzer eingestellte - zusammen.

In den weiteren Versuchen wurde festgestellt, dass kleinere Objekte nicht im Mesh erkennbar sind. Konkrete Beispiele waren eine 1L PET-Flasche und ein Gemälde mit einem 3,3cm tiefem Rahmen. Bei geringer Beleuchtung konnten Bereiche nicht gescannt werden und blieben leer. Objekte aus Glas oder solche mit metallischen Oberflächen werden fehlerhaft gescannt.

3.2 HoloLens und OpenPose

Es wurde versucht die C++ Bibliothek OpenPose für Pose Estimation zu verwenden. Dabei wurde eine Möglichkeit gesucht die Bibliothek in einer in Unity erstellten Anwendung zu verwenden. Alle notwendigen Bibliotheken mit Abhängigkeiten wurden dafür mit Hilfe von CMake-GUI und Visual Studio kompiliert. Anschließend konnten die er-

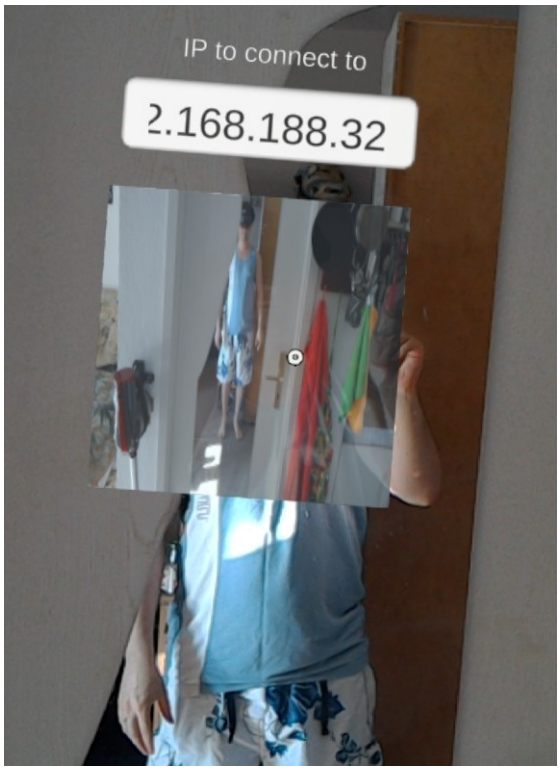
zeugten DLLs in einen Ordner mit dem Namen 'Plugins' in Unity kopiert werden und eine Pose Estimation mit dem Body 25 Model konnte auf dem PC ausgeführt werden. Bei dem Versuch die Anwendung auf der HoloLens auszuführen gab es jedoch aufgrund der inkompatibilität der kompilierten Bibliothken und der 32-Bit Architektur der HoloLens Fehlermeldungen.

3.3 OpenPose auf einem externen Computer

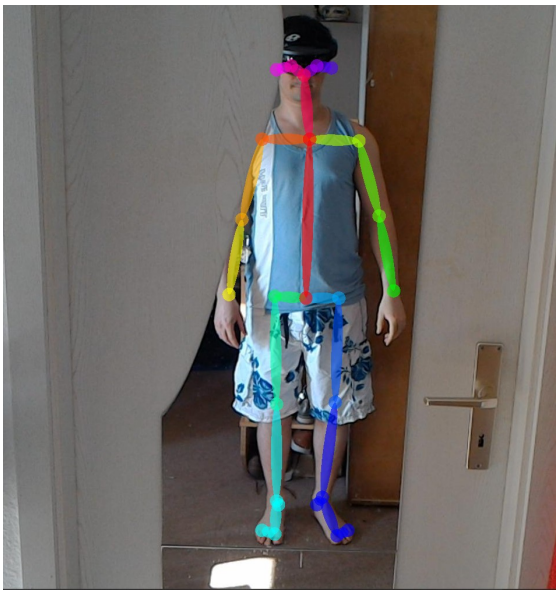
Man kann Bilder auf der HoloLens aufnehmen und diese über das Netzwerk an einen externen Computer schicken, um diese dann mit OpenPose zu verarbeiten. Um den PC auszuwählen mit dem man sich verbinden möchte, wurde in Unity ein Script geschrieben, das durch die Eingabe einer IP den Computer spezifiziert. Anschließend kann mit der Tap-Geste ein Bild aufgenommen werden. Dieses Bild ist als BRGA32 Array verfügbar und mehrere MB groß. Da OpenPose mit JPG umgehen kann, der Alpha-Channel nicht notwendig ist, das Bild nur noch einige KB groß ist und die Genauigkeit unbeeinträchtigt bleibt, wurde sich entschieden es als solches an den Computer zu schicken.
Auf dem Computer wartet ein TCP-Socket auf eine eingehende Verbindung von der HoloLens. Um zu wissen wie viele Bytes gelesen werden müssen, schickt die HoloLens vor den eigentlichen Bilddaten einen 4-Byte langen Header, um die Länge der Bilddaten anzugeben. Anschließend wird die angegebene Anzahl an Bytes eingelesen, als JPG auf dem Computer gespeichert und ein Fenster mit dem Bild und den Keypoints wird angezeigt. Der Großteil des Codes für die OpenPose Funktion ist dem Beispiel "01_body_from_image.py" entnommen.

4 Ergebnisse

Es konnte festgestellt werden das, dass Spatial Mapping viele Details nicht darstellt, es durch die Vielzahl an seperaten Objekten zu Überschneidungen kommt und dadurch viele redundante Punkte vorhanden sind. Es ist damit unmöglich, dass erzeugte Mesh für eine sorgfälltige Dokumentation des Tatorts zu benutzen.
Das Ausführen von OpenPose auf der HoloLens war aufgrund der Hardware nicht möglich. Die HoloLens konnte jedoch mit einem externen Computer über TCP verbunden werden und aufgenommene Bilder übertragen werden. Die Bilder konnten auf dem externen Computer analysiert und gespeichert werden.
Anhang A zeigt den, auf dem Desktop, ausgeführtem Python-Code. Das Unity-Anwendung kann auf der git Seite [6] heruntergeladen werden.



(a) Aufnahme der Unity-Anwendung (Zugeschnitten)



(b) Screenshot der Bildausgabe durch den Python-Code (Zugeschnitten)

Abbildung 4.1

5 Diskussion

Die 2019 erschienene HoloLens 2 hat bereits viele Hardwareverbesserungen die eine Umsetzung der Ideen verbessern würden. Es kommen jedoch auch immer mehr AR-Geräte auf den Markt. Sobald die Rechenleistung ausreicht und Machine Learning direkt auf den Geräten ausgeführt werden können, kommen sehr viele Anwendungsmöglichkeiten in Frage. Die wichtigste Komponente stellt dabei die GPU dar, da sie für parallele Verarbeitung von Daten und damit auch für Computer Vision essenziell ist.
Die Alternative ist, die Daten extern zu verarbeiten und die Ergebnisse zurück an das AR-Gerät zu schicken. Der Nachteil bei diesem Vorgehen ist die Notwendigkeit eine sichere Verbindung zu dem externen Gerät aufbauen zu können.
Die TOF-Kamera wurde in dieser Arbeit nicht weiter betrachtet, könnte für trassologische Untersuchungen jedoch relevant sein.

Anhang A: Python-Code

```
import sys
import socket
import numpy
import os
import cv2
from sys import platform
import argparse

tcp_port = 51512
tcp_ip = '0.0.0.0'
buf_size = 1024

def processImage(image):
    try:
        dir_path = os.path.dirname(os.path.realpath(__file__))
        sys.path.append(dir_path + '/../../python/openpose/Release');
        os.environ['PATH'] = os.environ['PATH'] + ';' + dir_path +
            ↪ '/../../x64/Release;' + dir_path + '/../../bin;'
        import pyopenpose as op

        # Flags
        parser = argparse.ArgumentParser()
        parser.add_argument("--image_path", default=image, help="
            ↪ Process an image. Read all standard formats (jpg, png,
            ↪ bmp, etc.).")
        args = parser.parse_known_args()


        # Custom Params (refer to include/openpose/flags.hpp for more
            ↪  parameters)
        params = dict()
        params["model_folder"] = "../../../models/"


        # Starting OpenPose
        opWrapper = op.WrapperPython()
        opWrapper.configure(params)
        opWrapper.start()
```

```
        # Process Image
        datum = op.Datum()
        imageToProcess = cv2.imread(args[0].image_path)
        datum.cvInputData = imageToProcess
        opWrapper.emplaceAndPop(op.VectorDatum([datum]))

        try:
            # Display Image
            print("Body keypoints: \n" + str(datum.poseKeypoints))
            cv2.imshow("OpenPose 1.7.0", datum.cvOutputData)
            cv2.waitKey(0)
        except Exception as e2:
            print("No Keypoints")
    except Exception as e:
        print(e)
        sys.exit(-1)


while True:
    # define header
    header = numpy.empty(4, dtype=bytes)
    bytesLeft = 4;
    # setup socket
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((tcp_ip,tcp_port))
    s.listen()
    c,addr = s.accept()
    # read header
    while(bytesLeft > 0):
        bytesRead = c.recv_into(header, bytesLeft)
        bytesLeft -= bytesRead

    # convert bytes to integer
    bytesLeft = numpy.frombuffer(header, dtype=int)[0]

    data = bytearray()
    i = 0
    fileName = "Image"
    fileEnding = ".jpg"
    fileWritten = False
    while(fileWritten == False):
        fullFileName = fileName + str(i) + fileEnding
        try:
```

```
            with open (fullFileName, "xb") as f:
                while(bytesLeft > 0):
                    data.extend(c.recv(1024))
                    bytesLeft -= bytesRead
                f.write(data)
                fileWritten = True
        except Exception as e:
            print(e)
            i += 1
    processImage(fullFileName)


c.close()
s.close()
```

Literatur

[1] Zhe Cao u. a. “Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields”. In: *CVPR*. 2017.

[2] *COCO and MPI Models*. URL: https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/md_doc_installation_2_additional_settings.html#autotoc_md196 (besucht am 10. 07. 2021).

[3] *Difference between Body 25 COCO and MPI*. URL: https://github.com/microsoft (besucht am 10. 07. 2021).

[4] *Difference between Body 25 COCO and MPI*. URL: https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/md_doc_05_faq.html (besucht am 10. 07. 2021).

[5] Epic. *Unreal Engine 4.25 released!* URL: https://www.unrealengine.com/en-US/blog/unreal-engine-4-25-released (besucht am 07. 06. 2021).

[6] Vladislav Förster. *HoloLens2Desktop*. URL: https://github.com/BoomRaccoon/HoloLens2Desktop (besucht am 14. 07. 2021).

[7] Kourosh Khoshelham, Ha Tran und Debaditya Acharya. “INDOOR MAPPING EYEWEAR: GEOMETRIC EVALUATION OF SPATIAL MAPPING CAPABILITY OF HOLOLENS”. In: Bd. XLII-2/W13. Juni 2019, S. 805–810. DOI: 10.5194/isprs-archives-XLII-2-W13-805-2019.

[8] Khronos. *Multiple Conformant OpenXR Implementations Ship Bringing to Life the Dream of Portable XR Applications*. URL: https://www.khronos.org/news/press/multiple-conformant-openxr-implementations-ship-bringing-to-life-the-dream-of-portable-xr-applications (besucht am 07. 06. 2021).

[9] Microsoft. *Azure mixed reality cloud services overview*. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/develop/mixed-reality-cloud-services (besucht am 07. 06. 2021).

[10] Microsoft. *HoloLens-Hardware*. URL: https://docs.microsoft.com/de-de/hololens/hololens1-hardware (besucht am 07. 06. 2021).

[11] Microsoft. *HoloLens2-Hardware*. URL: https://docs.microsoft.com/en-us/hololens/hololens2-hardware (besucht am 07. 06. 2021).

[12] Microsoft. *Introduction to Mixed Reality development*. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/develop/development?tabs=unity (besucht am 07. 06. 2021).

[13] Microsoft. *Map physical spaces with HoloLens*. URL: https://docs.microsoft.com/en-us/hololens/hololens-spaces (besucht am 07. 06. 2021).

[14] Microsoft. *What is the Mixed Reality Toolkit*. URL: https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/?view=mrtkunity-2021-05 (besucht am 07. 06. 2021).

[15] *OpenPose Library - Compilation and Installation*. URL: https://github.com/tramper2/openpose/blob/master/doc/installation.md (besucht am 07. 07. 2021).

[16] plaggy. *CC0 - Bucket 3*. URL: https://sketchfab.com/3d-models/cc0-bucket-3-cb484683659d465796ed5af8664cf58f (besucht am 14. 07. 2021).

[17] DSc Prof. Wieczorek T. PhD. "INNOVATIVE INVESTIGATIONS OF THE CRIME SCENE USING 3D SCANNERS". In: *International Scientific Journal Security & Future*. Bd. Year II Issue. 2018, S. 40–42.

[18] Elene Terry. "Silicon at the Heart of HoloLens 2". In: *2019 IEEE Hot Chips 31 Symposium (HCS)*. 2019, S. 1–26. DOI: 10.1109/HOTCHIPS.2019.8875669.

[19] *Visualisation of the Body 25 model*. URL: https://cmu-perceptual-computing-lab.github.io/openpose/web/html/.github/media/keypoints_pose_25.png (besucht am 10. 07. 2021).

[20] *Visualisation of the COCO model*. URL: https://cmu-perceptual-computing-lab.github.io/openpose/web/html/.github/media/keypoints_pose_18.png (besucht am 10. 07. 2021).

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 14.07.2021