HAW HAMBURG

BACHELOR THESIS
Ahmad Shapur Ghani

Roboterunterstützte KI-basierte Sturzvalidierung in Smart Home-Umgebungen

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Ahmad Shapur Ghani

Roboterunterstützte KI-basierte Sturzvalidierung in Smart Home-Umgebungen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Thomas Lehmann
Zweitgutachter: Prof. Dr. Kai von Luck

Eingereicht am: 12.03.2024

Ahmad Shapur Ghani

Thema der Arbeit

Roboterunterstützte KI-basierte Sturzvalidierung in Smart Home-Umgebungen

Stichworte

Soziale Robotik, Smart Home, Unterstütztes Wohnen, YOLOv8, ROS2

Kurzzusammenfassung

Die Bachelorarbeit bewegt sich im Rahmen der Living Place Forschungsgruppe und hat das Ziel, ein Szenario für die Suche und Rettung von gestürzten Personen in der Wohnung umzusetzen. Es soll auf Sturzmeldungen im Smart Home reagiert werden und ein Notfallszenario eingeleitet werden, um die gestürzte Person zu lokalisieren. Dieser Prozess umfasst die suche nach gestürzten Personen mit einem Roboter, eine Person- & Posen-Erkennung, sowie das Absetzen einer Notfallbenachrichtigung an Helfer und das öffnen aller möglichen Türen der Wohnung.

Ahmad Shapur Ghani

Title of Thesis

Robot-assisted AI-based fall validation in smart home environments

Keywords

Social robotics, smart home, assisted living, YOLOv8, ROS2

Abstract

The bachelor's thesis is part of the Living Place research group and aims to implement a scenario for the search and rescue of people who have fallen in the home. The aim is to respond to fall reports in the smart home and initiate an emergency scenario to find the person who has fallen. This process includes searching for people who have fallen using a robot, recognising a person & poses, sending an emergency notification to helpers and opening all possible doors in the home.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Problemstellung

Die demografische Veränderung in Europa wird uns vor viele Herausforderungen stellen. Ein bedeutendes Problem wird sein, dass immer mehr pflegebedürftige Personen mit weniger Personal versorgt werden müssen. Die Berichte der Europäischen Kommission und des Statistischen Bundesamts für Deutschland liefern Informationen darüber, mit wie vielen pflegebedürftigen Personen bis zum Jahr 2050 zu rechnen ist.

Um die Bevölkerung auf einem konstanten Niveau zu halten, ist eine durchschnittliche Geburtenrate von 2,1 erforderlich. Im Jahr 2020 lag die europäische durchschnittliche Geburtenrate bei 1,5 [1]. Im Jahr 2022 betrug die durchschnittliche Geburtenrate in Deutschland 1,46 [2]. Durch die zu niedrige Geburtenrate und die seit 2021 wieder steigende Lebenserwartung von Frauen und Männern wird es immer mehr ältere Menschen geben. Im Jahr 2050 werden voraussichtlich 30% der in der EU lebenden Menschen über 65 Jahre alt sein. Außerdem ist zu erwarten, dass die Anzahl der Haushalte steigt, während die durchschnittliche Größe der Haushalte abnimmt [1].

Mit der zunehmenden Lebenserwartung steigt auch die Anzahl der Personen, die pflegebedürftig sind. Im Jahr 1999 waren in Deutschland 2,02 Millionen Personen pflegebedürftig. Im Jahr 2021 hat sich diese Zahl mit 4,96 Millionen mehl als verdoppelt [2].

Von diesen knapp 5 Millionen Menschen, werden über 80% von zu Hause aus versorgt. Abbildung 1.1 zeigt auf, durch wen diese versorgt werden.

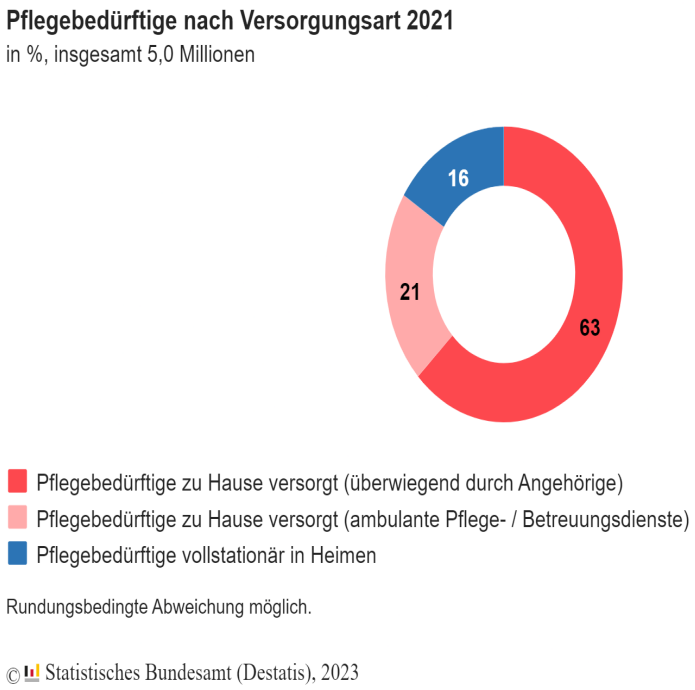


Abbildung 1.1: Pflege Versorgungsart [2]

Es sind häufig ältere Menschen, die den größten Teil ihrer Zeit allein zu Hause verbringen. Pflegekräfte oder Familienangehörige können oft nicht ständig vor Ort sein. Dies ist nicht selten von der pflegebedürftigen Person selbst auch überhaupt nicht gewünscht. Ein Sturz stellt zwar jeden Menschen ein alltägliches Risiko dar, aber Menschen über 65 Jahre haben ein deutlich höheres Risiko zu stürzen. Häufig fehlt älteren Menschen nach einem Sturz die Kraft, selbstständig aufzustehen [43]. Um ein langes Liegenbleiben zu vermeiden, existieren unterstützende Systeme wie beispielsweise der Hausnotruf des Deutschen Roten Kreuzes. Die Anwender tragen dabei entweder eine Notrufkette oder ein Armband mit einem Knopf, der im Notfall betätigt werden kann [3]. Solch ein System hat jedoch einen großen Nachteil: Der Anwender kann das Tragen der Kette oder des Armbandes vergessen.

Smart Home-Techniken können diesem Problem Abhilfe schaffen.

Diese Arbeit befasst sich mit der Frage, wie einer gestürzten Person geholfen werden kann, die nicht in der Lage ist, selbst aufzustehen oder um Hilfe zu rufen. Es wird an

einem vorhandenen Sturzerkennungssystem angesetzt, um dieses zu einem Verifikationssystem weiterzuentwickeln. Das Sturzerkennungssystem nutzt Kamerabilder, die häufig in Smart Home Umgebungen integriert sind. Durch die Analyse der Kamerabilder kann das System eine Schätzung der Körperhaltung der Bewohner vornehmen und einen Sturz feststellen.

Ab diesem Zeitpunkt setzt das Verifikationssystem ein und erweitert den Ablauf. Wenn ein Sturz erkannt wird, überträgt das Sturzerkennungssystem das Ereignis und die Position des Sturzes an einen Server. Der Server sendet dann einen Roboter zur Überprüfung und Suche aus. Der Roboter ist ebenfalls mit einer Kamera ausgestattet und kann sich fortbewegen. Seine Aufgabe besteht darin, zur gestürzten Person zu fahren und den Sturz zu verifizieren. Dies geschieht durch die Übertragung von Bildern an einen separaten Server, auf dem eine Personenerkennung und eine erneute Einschätzung der Körperhaltung durchgeführt wird. Sobald eine auf dem Boden liegende Person erkannt wurde, wird eine Notfallmeldung an zuvor bestimmte Helfer gesendet.

Das Verifikationssystem erweitert das vorhandene Sturzerkennungssystem und verbessert die Zuverlässigkeit des Gesamtsystems maßgeblich. Ein kritischer Aspekt der Sturzerkennung ist die Bestimmung der Schwellenwerte, die festlegen, wann ein Sturz registriert wird. In dieser Arbeit sind die Aspekte der falsch positiven (false positive) und falsch negativen (false negative) Erkennungen von großer Relevanz. Ein falsch positiver Fall tritt auf, wenn das System fälschlicherweise einen Sturz signalisiert, während ein falsch negativer Fall vorliegt, wenn ein tatsächlich erfolgter Sturz vom System nicht erkannt wird.

Das Verifikationssystem erlaubt es, die Schwellenwerte für das Sturzerkennungssystem herabzusetzen, wodurch zwar die Häufigkeit falsch-positiver Meldungen ansteigen kann, diese jedoch durch die nachgelagerte Überprüfungskette, bestehend aus Server, Roboter und weiteren Komponenten, effektiv abgefangen werden. Wenn sich herausstellt, dass kein Sturz stattgefunden hat, hat dies keine negativen Konsequenzen.

Gleichzeitig führt das Absenken des Schwellenwertes zu einer Verringerung der falsch-negativen Fälle. Dadurch steigt die Wahrscheinlichkeit, dass tatsächliche Stürze zuverlässig erkannt werden. Dieser Ansatz führt zu einer veritablen Präzisierung der Sturzerkennung. Damit wird das Sicherheitsnetz für Personen, die auf das System angewiesen sind, ungleich verbessert.

Das Verfikationssystem soll dazu beitragen, die Sicherheit und das Wohlbefinden von Personen in einem Smart Home zu verbessern. Insbesondere ältere oder kranke Menschen, die möglichst eigenständig wohnen möchten, sollen durch ein solches System Entlastung erfahren.

1.2 Aufbau der Arbeit

Die Arbeit ist in sechs Kapitel gegliedert. Im folgenden Kapitel wird der Stand der Technik behandelt. Hierbei wird ein Verständnis für Smart Home Umgebungen erarbeitet, inklusive ähnlicher Lösungsansätze und einer Vorstellung des in dieser Arbeit verwendeten Labors, dem Living Place an der HAW Hamburg. Anschließend erfolgt die Problemanalyse, in der die Ursachen und Folgen von Stürzen beschrieben und die Anforderungen an das Verifikationssystem festgelegt werden. Im vierten Kapitel wird die Umsetzung des Systems erläutert und dabei die verwendeten Techniken vorgestellt sowie ihre Auswahl begründet. In einem weiteren Kapitel werden die Ergebnisse präsentiert. Abschließend wird ein Ausblick gegeben und ein Fazit gezogen.

2 Stand der Technik

2.1 Was ist ein Smart Home?

Ein Smart Home ist eine vernetzte und automatisierte Wohnumgebung, die durch die Integration intelligenter Technologien und Geräte in der Lage ist, verschiedene Aspekte des Wohnens zu überwachen, zu steuern und zu optimieren. Die Grundidee besteht darin, durch den Einsatz von Sensoren, Aktoren und digitalen Schnittstellen eine effiziente Interaktion zwischen den Bewohnern und ihrer Wohnumgebung zu ermöglichen [24].

Smart Homes bieten eine Vielzahl von Funktionen zur Unterstützung der Bewohner:

1. **Sicherheit:** Überwachungssysteme, Einbruchssensoren und intelligente Türschlösser können die Sicherheit erhöhen
2. **Energieeffizienz:** Smarte Thermostate, Beleuchtungssysteme und Energiemonitoring ermöglichen eine effiziente Nutzung von Ressourcen, wodurch Energiekosten gesenkt werden können.
3. **Komfort:** Automatisierte Beleuchtung, Klimaregelung und intelligente Haushaltsgeräte optimieren den Wohnkomfort.
4. **Gesundheit:** Smarte Gesundheitsüberwachungssysteme und Assistenzgeräte können die Gesundheitsparameter überwachen und Warnungen bei Abweichungen senden.
5. **Konnektivität:** Die Integration von vernetzten Geräten ermöglicht eine nahtlose Kommunikation zwischen verschiedenen Komponenten des Haushalts.

Smart Home-Systeme werden in Zukunft in kaum einem Haushalt wegzudenken sein. Die Entwicklungsfortschritte und die Anpassungsfähigkeit der Systeme an den allgemeinen

technologischen Fortschritt sind bereits zwei nicht zu unterschätzende Aspekte. Spätestens aber die einfache Bedienung und unkomplizierte Anbindung an den eigenen Haushalt wird der Verbreitung bahnbrechen. Die Forschung konzentriert sich derzeit auf die Erweiterung der Funktionalitäten und die Optimierung des Zusammenspiels der Systemkomponenten. Beispiele für zukunftsrelevante Aspekte sind:

1. **Künstliche Intelligenz (KI):** Integration von KI-Algorithmen zur kontextbezogenen Anpassung von Systemfunktionen an die Bedürfnisse der Bewohner [38].
2. **Nachhaltigkeit:** Weiterentwicklung von Smart Home Systemen, um nachhaltiges Ressourcenmanagement und umweltfreundliche Praktiken zu fördern [38].
3. **5G-Technologie:** Nutzung fortschrittlicher Kommunikationsprotokolle wie 5G, um die Konnektivität und Datenübertragung in Smart Homes zu verbessern [29].
4. **Robotik:** Integration von Robotik für unterstützende Aufgaben im Haushalt [27].
5. **Gesundheitswesen:** Erweiterung von Smart Home Anwendungen im Gesundheitswesen, einschließlich telemedizinischer Dienste und Gesundheitsüberwachung [44].

Insgesamt ebnet die kontinuierliche Forschung und Entwicklung den Weg für eine breite Anwendbarkeit von Smart Home-Systemen in verschiedenen Lebensbereichen. Smart Home-Systeme werden in der zukünftigen Verbrauchertechnologie-Landschaft eine große Rolle spielen.

2.2 Vorstellung Living Place

Die HAW Hamburg besitzt ein eigenes Smart Home-Labor namens Living Place Lab (LP) [5], welches als Forschungsumgebung für Lehrende und Studierende dient. Das LP bietet zahlreiche Möglichkeiten für verschiedene Forschungsthemen, insbesondere im Bereich des Ambient Assisted Living (AAL). Das in dieser Arbeit entwickelte System ist für den Einsatz im LP vorgesehen. Es ergänzt die bestehenden Smart Home-Komponenten und soll einen Beitrag zum Forschungsbereich des AAL leisten. Es optimiert die Interaktion mit der Umgebung und unterstützt die Bewohner weiter.

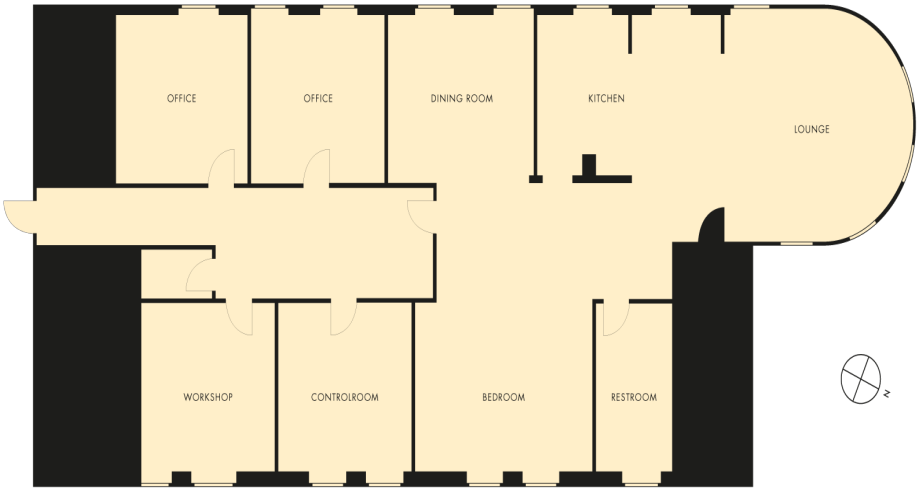


Abbildung 2.1: Living Place Grundriss [5]

2.3 Ambient Assisted Living

Ambient Assisted Living (AAL) umfasst eine Vielzahl von Systemen, Produkten und Dienstleistungen, die innovative Technologien nutzen, um insbesondere das Leben älterer Menschen zu verbessern [26]. AAL gewinnt angesichts des demografischen Wandels zunehmend an Bedeutung. Ziel ist es, alleinlebenden Senioren mehr Selbstbestimmtheit in den eigenen vier Wänden zu ermöglichen und dieses selbstbestimmte Leben sicherer zu gestalten. Die Palette der AAL-Systeme reicht von einfachen Lösungen wie Bewegungssensoren, die das Licht automatisch einschalten, bis hin zu komplexeren Systemen wie intelligenten Medikamentendispensern, die die Medikamenteneinnahme überwachen, also an die Einnahme erinnern und die Einhaltung der Einnahme kontrollieren [42]. Das Ziel ist, Unterstützung zu bieten, wo nötig und damit gleichzeitig die Lebensqualität und Autonomie zu fördern.

2.4 Verwandte Arbeiten

Roboter als Assistenten für ältere Menschen im Alltag sind bereits Gegenstand der Forschung. Relevant für die hiesige sind zwei Veröffentlichungen: *„RiSH: A robot-integrated smart home for elderly care“* [27] und *„Fallen people detection capabilities using assistive robot“* [34]. Diese Veröffentlichungen verfolgen teilweise ein ähnliches Ziel wie diese Arbeit. Im Falle eines Sturzes soll der Roboter zur gestürzten Person fahren und Hilfe anfordern.

„RiSH“ RiSH entwickelt ein roboterintegriertes Smart Home zur Unterstützung älterer Menschen. Das System kombiniert verschiedene Technologien wie Heimserviceroboter, Sensornetzwerke und Cloud-Dienste, um die Aktivitäten und das Wohlbefinden der Bewohner zu überwachen. Der Roboter nutzt Audiosignale und Bewegungssensoren, um menschliche Aktivitäten zu erkennen und auf Notfälle wie Stürze zu reagieren. Die Fähigkeit des Roboters, Stürze zu erkennen und darauf zu reagieren, wurde durch Experimente mit menschlichen Probanden getestet [27].

„Fallen people detection capabilities using assistive robot“ Beschäftigt sich mit der Entwicklung und dem Test eines Assistenzroboters, der gestürzte Personen erkennen kann. Der Roboter begibt sich auf Patrouille und analysiert eingehende Bilder. Dabei erfolgt die Sturzerkennung mittels Objekterkennung und die Meldung auf dem Roboter selbst [34].

2.5 Personenerkennung

Objekterkennung ist ein Teilbereich der Computer Vision, der sich mit der Identifikation und Klassifizierung von visuellen Objekten in digitalen Bildern oder Videos beschäftigt. Dabei geht es nicht nur um die Klassifizierung der Objekte, sondern auch um ihre genaue Lokalisierung. Moderne Objekterkennungssysteme sind in der Lage, mehrere Objekte und deren Klassen in einem Bild zu erkennen und gleichzeitig ihre Position durch Bounding Boxes anzuzeigen.

Funktionsweise der Objekterkennung: Objekterkennungsalgorithmen nutzen in der Regel maschinelles Lernen, um aus einer Vielzahl von Trainingsdaten zu lernen. Diese Algorithmen extrahieren Merkmale aus den Trainingsbildern, die zur Identifizierung

von Objekten beitragen. Die Merkmale können einfache Kanten oder Farben bis hin zu komplexen Mustern sein, die für spezifische Objekte charakteristisch sind [35].

Verschiedene Ansätze der Objekterkennung: Bei zweistufigen Detektoren, werden zunächst Objektregionen vorgeschlagen (z.B. mittels Region Proposal Network in Faster R-CNN), bevor die Klassifizierung und Bounding-Box-Regression für diese Vorschläge erfolgt [40].

Einstufige Detektoren wie YOLO und SSD überspringen den Region-Vorschlag-Schritt und prognostizieren Klassifizierung und Bounding Boxes in einem Durchgang, was zu einer schnelleren Verarbeitung führt [25].

Verschiedene Modelle der Objekterkennung:

- R-CNN und Varianten (Fast R-CNN, Faster R-CNN): Regionen mit CNN-Features, die Objekte in einem Bild lokalisieren und klassifizieren [40].
- YOLO (You Only Look Once): Schnelle Objekterkennung, die das Bild in einer einheitlichen Struktur analysiert, um gleichzeitig zu klassifizieren und lokalisieren [25].
- SSD (Single Shot MultiBox Detector): Ähnlich wie YOLO, verwendet es einen einzigen Durchlauf, um Objekte zu lokalisieren und zu klassifizieren [32].

Bewährte Techniken:

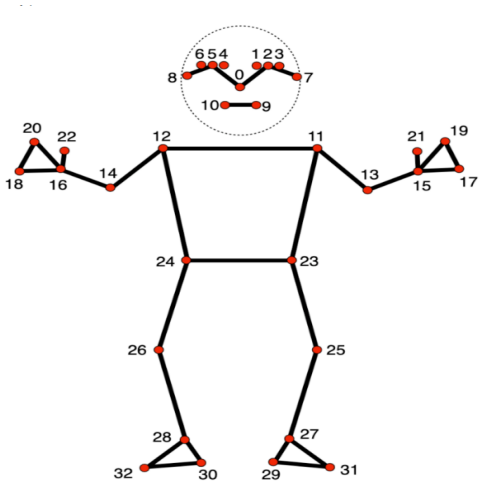
Konvolutionale neuronale Netze (CNNs) haben sich als besonders leistungsfähig erwiesen, da sie hierarchische Merkmalsdarstellungen für die Objekterkennung lernen können.

Für das Training der Modelle haben sich- Data Augmentation und Transfer Learning als die besten Varianten erwiesen. Sie erhöhen die Vielfalt und Menge der Trainingsdaten, was zu einer verbesserten Leistung führt [41].

Da Menschen bei der Objekterkennung einer Objekt-Klasse zugeordnet werden, müssen sie anhand von Datensätzen und Trainingsmethoden in Objekterkennungsmodellen so trainiert, dass sie in Bildern als Personen bzw. Menschen klassifiziert und dargestellt werden.

2.5.1 Pose Estimation

Pose Estimation ist ein Prozess in der Bildverarbeitung. Das Ziel ist es, die Position und Orientierung von Objekten oder Lebewesen innerhalb eines Bildes oder Videos zu identifizieren. Bei der Personenerkennung, welche eine häufige Anwendung der Pose Estimation darstellt, geht es darum, die Positionen von Körperteilen und die Haltung (Pose) einer oder mehrerer Personen zu erkennen. Oft werden diese in Form von sogenannten Keypoints dargestellt, welche Gelenke oder markante Körperpunkte repräsentieren.



(a) Pose Estimation Keypoints aus MediaPipe [33]

0 - nose	16 - right wrist
1 - left eye (inner)	17 - left pinky
2 - left eye	18 - right pinky
3 - left eye (outer)	19 - left index
4 - right eye (inner)	20 - right index
5 - right eye	21 - left thumb
6 - right eye (outer)	22 - right thumb
7 - left ear	23 - left hip
8 - right ear	24 - right hip
9 - mouth (left)	25 - left knee
10 - mouth (right)	26 - right knee
11 - left shoulder	27 - left ankle
12 - right shoulder	28 - right ankle
13 - left elbow	29 - left heel
14 - right elbow	30 - right heel
15 - left wrist	31 - left foot index
	32 - right foot index

(b) Pose Estimation Keypoints beschreibung aus MediaPipe [33]

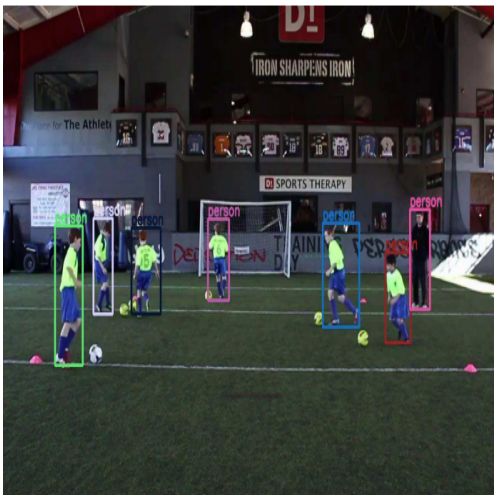
Abbildung 2.2: Keypoints und Beschreibung aus MediaPipe

Es bestehen zwei Hauptansätze zur Pose Estimation:

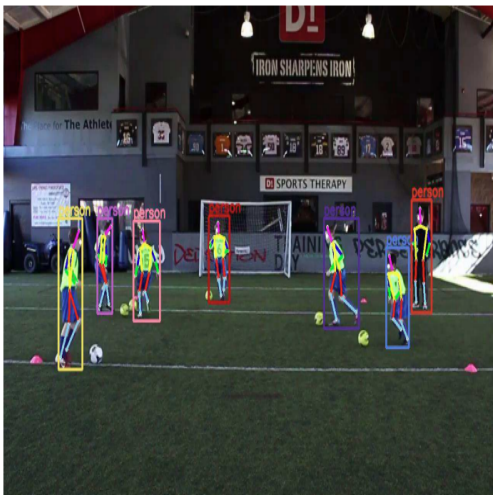
Bottom-Up-Ansatz: Dieser Ansatz beginnt damit, alle relevanten Keypoints im Bild zu identifizieren, ohne zu wissen, zu welcher spezifischen Person sie gehören. Nachdem die Keypoints erfasst wurden, werden sie mithilfe von Clustering- oder Matching-Algorithmen in sinnvolle Gruppen eingeteilt, um einzelne Personen und ihre Posen zu rekonstruieren [30]. Ein prominentes Beispiel für den Bottom-Up-Ansatz ist „MediaPipePose“ von Google [33].

Top-Down-Ansatz: Zunächst werden Personen im Bild mithilfe eines Objekterkennungssystems (z.B. YOLO) identifiziert. Um jede Person herum wird sodann eine Bounding Box gezeichnet. Anschließend wird für jede erkannte Bounding Box separat eine

Keypoint-Erkennung durchgeführt, um die Pose der Person innerhalb der Box zu bestimmen [37].



(a) Zunächst Personenerkennung [37]



(b) Pose Estimation der einzelnen Personen [37]

Abbildung 2.3: Personen und Posen Schätzung mit dem Top-Down-Ansatz

Index	Key point
0	Nose
1	Left-eye
2	Right-eye
3	Left-ear
4	Right-ear
5	Left-shoulder
6	Right-shoulder
7	Left-elbow
8	Right-elbow
9	Left-wrist
10	Right-wrist
11	Left-hip
12	Right-hip
13	Left-knee
14	Right-knee
15	Left-ankle
16	Right-ankle

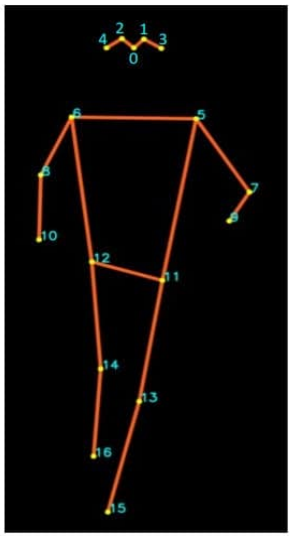


Abbildung 2.4: Keypoints YOLOv8 [23]

Beide Ansätze haben spezifische Vor- und Nachteile. Der Bottom-Up-Ansatz kann schneller sein, da er das Bild nur einmal verarbeiten muss, um alle Keypoints zu identifizieren. Der Top-Down-Ansatz kann dagegen bei komplexen Bildern mit vielen Personen genauer

sein, da er sich auf die einzelnen Personen fokussiert und Interaktionen zwischen den Posen verschiedener Personen minimiert.

Anwendungsgebiete der Pose Estimation sind vielfältig und umfassen unter anderem [28]:

- Sportanalyse: zur Beurteilung von Körperhaltungen und -bewegungen bei Sportlern.
- Überwachung und Sicherheit: Erkennung von Verhaltensweisen wie z.B. Stürzen, um schnelle Hilfeleistung zu ermöglichen.
- Gesundheitswesen: zur Unterstützung bei der Rehabilitation und Analyse von Körperhaltungen.
- Entertainment: in Videospielen und Filmproduktion, wo die Bewegungen von Schauspielern in digitale Modelle übertragen werden (Motion Capture).

3 Analyse

Das Ziel dieser Arbeit besteht darin, ein System zu entwerfen, das in der Lage ist, Stürze zu verifizieren, die durch ein im Smart Home integriertes Sturzerkennungssystem registriert wurden. Das System soll in Fällen eingesetzt werden, in denen die gestürzte Person nicht selbstständig in der Lage ist, Hilfe zu rufen. Anschließend werden Maßnahmen ergriffen, um die Situation zu bewerten und weitere Hilfsmaßnahmen einzuleiten.

Dieses Kapitel beginnt mit der Definition des Begriffs „Sturz“ (3.1). Anschließend werden die Grundlagen und die schwerwiegenden Folgen von Stürzen auf die Gesundheit und Lebensqualität Betroffener erläutert (3.2). Darauffolgt der Blick auf die kritische Rolle der schnellen Reaktionszeit nach einem Sturzereignis (3.3). Abschließend werden die Anforderungen an das System dargestellt, die die Grundlage für die Realisierung des Projektes bilden (3.4). Das Kapitel 3 bildet damit Grundlage für das Verständnis der Problemstellung und unterstreicht die Relevanz der Entwicklung eines solchen Systems.

3.1 Definition „Sturz“

Ein Sturz wird durch das Deutsche Netzwerk für Qualitätsentwicklung in der Pflege (DNQP) folgendermaßen definiert:

> „Ein Sturz ist ein Ereignis, bei dem der Betroffene unbeabsichtigt auf dem Boden oder auf einer anderen tieferen Ebene aufkommt.“ Hiermit sind auch Stürze gemeint, in deren Folge die Betroffenen den Boden oder die tiefere Ebene nicht mit dem gesamten Körper berühren, sondern dort beispielsweise sitzen oder hocken. [43]

3.2 Grundlagen und Folgen

Stürze stellen für alle Menschen ein alltägliches Risiko dar. Allerdings erhöht sich das individuelle Sturzrisiko durch den Verlust der Fähigkeit, Stürze zu vermeiden, sei es wegen fortgeschrittenen Alters oder durch Krankheit. Menschen ab 65 Jahren haben ein signifikant höheres Risiko zu stürzen [43]. Auch die Folgen eines Sturzes haben unterschiedliche Auswirkungen auf das allgemeine Wohlbefinden der gestürzten Person. Im Durchschnitt stürzen Menschen über 65 Jahre mindestens einmal pro Jahr. Frauen in diesem Alter haben dabei ein höheres Sturzrisiko als Männer [43].Etwa 15% dieser Stürze führen zu einem Krankenhausaufenthalt, unabhängig davon, ob die betroffene Person schwerkrank ist, zu Hause lebt oder in einem Pflegeheim untergebracht it. Die physischen und psychischen Verletzungen, die ein Sturz zur Folge hat, sind mit höherem Alter in der Regel auch gravierender. Das betrifft auch die Angehörigen der Betroffenen. Denn durch das erhöhte Sturzrisiko oder eine Sturzerfahrung entsteht Pflegebedürftigkeit und nicht selten soziale Abhängigkeit. Angehörige können Schuldgefühlen und Selbstvorwürfen ausgesetzt sein. Die Betroffenen leiden oft unter dem Verlust des Vertrauens in die eigenen Fähigkeiten und darunter, mangels der entsprechenden Fähigkeiten, Abhängigkeiten zu Lasten anderer zu erschaffen. [43]

3.3 Einflussfaktor Zeit

Das schnelle Auffinden einer gestürzten Person und ihre angemessene Behandlung sind von entscheidender Bedeutung, insbesondere wenn die gestürzte Person nicht in der Lage ist, selbst Hilfe zu rufen. Durch schnelles Handeln können schwerwiegende Verletzungen erkannt und behandelt, Komplikationen vermieden, Schmerzen gelindert und das psychische Wohlbefinden aller Betroffenen verbessert werden. Eine verzögerte Erkennung und Behandlung kann zu einer Verschlimmerung der Verletzungen, Infektionsrisiken, langfristigen Beeinträchtigungen und psychologischen Auswirkungen führen [36]. Ein effizientes Sturzerkennungs- und Verifikationssystem verschafft hier Abhilfe. Solche Systeme können in einer Smart-Home-Umgebung als unterstützende Systeme eingesetzt werden, um die Sicherheit und Gesundheit der Betroffenen zu erhöhen und ihre Lebensqualität und Unabhängigkeit zu erhalten. Eine schnelle Identifizierung und Unterstützung nach einem Sturz ist daher entscheidend, um negative Folgen zu minimieren und eine umfassende Versorgung sicherzustellen.

3.4 Anforderungen

Die Sturzerkennung von Personen mittels Wohnraumüberwachung erfährt im LP ein besonderes Augenmerk (Vgl. BA Schmidpeter [16]). Das hier untersuchte Verifikationssystem soll an diese Vorarbeit anknüpfen. Im Vordergrund steht hier der zum Unfallort fahrende Roboter und die dort durch ihn zu erfolgende Verifizierung des Sturzes sowie die Reaktion auf das Verifikationsergebnis. Im Falle eines erfassten Sturzes also die abzusendende Notfallnachricht.

In der Abbildung 3.1 ist die vorangegangene Sturzerkennung dargestellt.

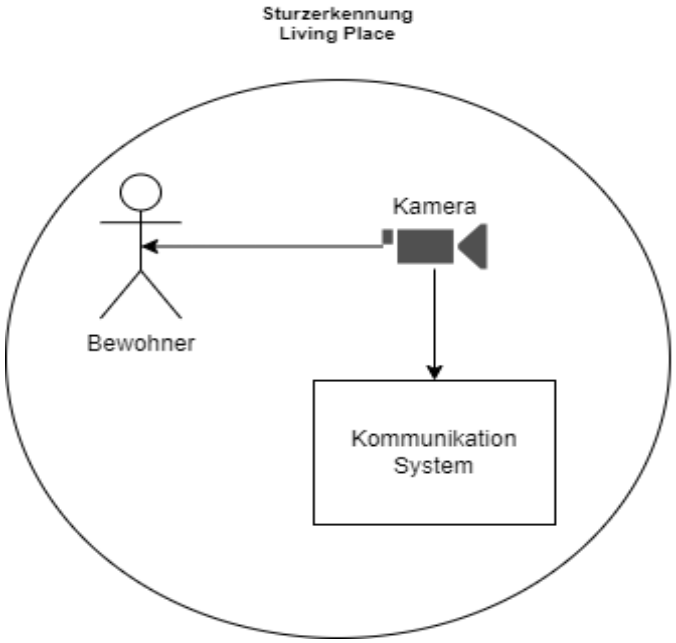


Abbildung 3.1: Kontextdiagramm Sturzerkennungssystem

Das Sturzerkennungssystem überwacht mithilfe der im LP verbauten Kameras die Bewohner im Wohnraum. Das System arbeitet mit einer pose estimation-Software, welche angepasst wurde, um Stürze der Bewohner zu registrieren. Das Kommunikationssystem wird nach einem erkannten Sturz benachrichtigt.

Die Abbildung 3.2 beschreibt die einzelnen Komponenten des Verifikationssystems und wie sie miteinander interagieren.

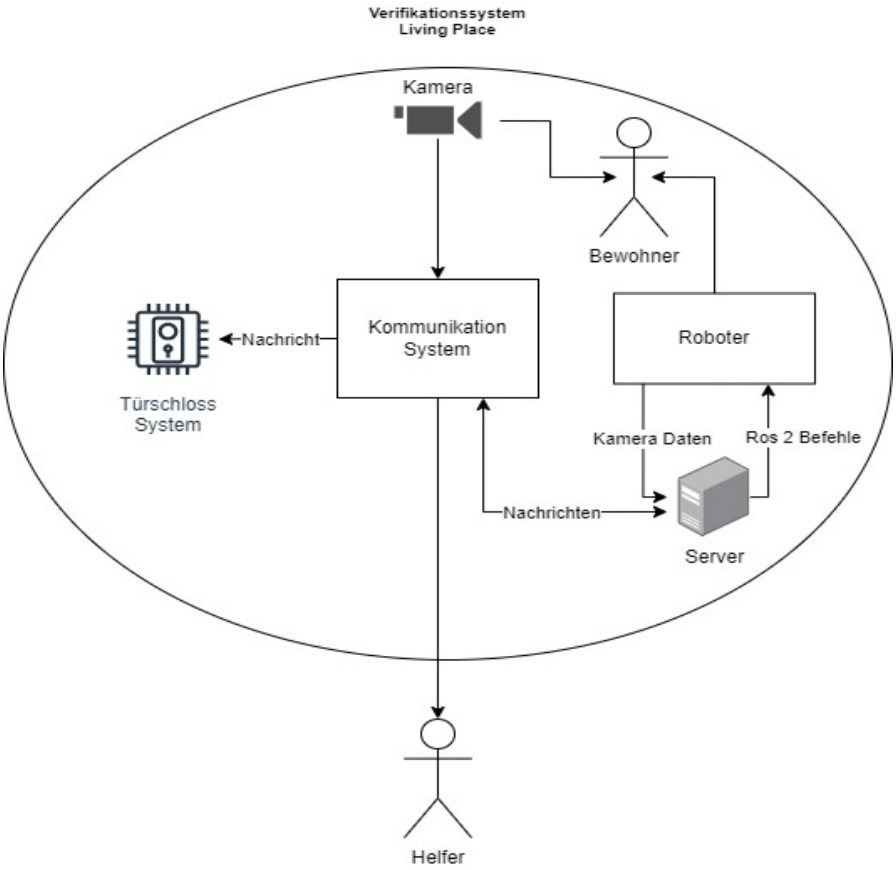


Abbildung 3.2: Kontextdiagramm Verfikationssystem

Kommunikationssystem:
Das Kommunikationssystem bildet das Grundgerüst des interaktiven Netzwerks innerhalb des Smart Home Systems. Es fungiert als Vermittler zwischen den verschiedenen Komponenten, um einen reibungslosen Informationsaustausch und eine effektive Koordination zu gewährleisten. Außerdem kann es bei bestimmten Ereignissen einen gewünschten Prozess auslösen.

Bewohner:
Die Bewohner sind die Hauptnutzer und die primäre Zielgruppe des Smart Home Systems. Ihre Rolle besteht darin, das Living Place zu bewohnen und die Funktionen des Systems zu nutzen. Im Falle eines Sturzes sind die Bewohner diejenigen, die von den Sicherheitsmechanismen des Systems profitieren, da es darauf abzielt, ihnen schnell und

effektiv zu helfen. Die Bewohner sind passive Akteure im System, deren Wohlbefinden und Sicherheit durch die automatischen Reaktionen des Systems unterstützt werden.

Kameras:
Die Kameras des Smart Home Systems sind für die kontinuierliche Überwachung der Wohnung zuständig. Die Hauptaufgabe in diesem Systemkontext besteht darin, Stürze zu erkennen, mithilfe des Sturzerkennungssystems. Wenn eine solche Situation erkannt wird, sendet das Sturzerkennungssystem Benachrichtigungen an das Kommunikationssystem, um eine Notfallreaktion auszulösen. Sie dienen als einzige Sensoren für die Erkennung von Stürzen im Wohnbereich und bilden eine wesentliche Grundlage für die Sicherheit und das Wohlbefinden der Bewohner.

Roboter:
Der Roboter fungiert als physische Einheit im Smart Home-System. Seine Aufgabe besteht darin, im Falle eines erkannten Sturzes zur übermittelten Position zu fahren und bei der Bilderfassung und -verarbeitung für den Server zu helfen.

Server:
Der für das Verifikationssystem bereitgestellte Server ist die zentrale Datenverarbeitungseinheit des Verifikationssystems. Seine Hauptaufgabe besteht darin, die von den Robotern aufgenommenen Bilder zu empfangen und eine Personenerkennung durchzuführen sowie zu verifizieren, dass es sich bei dem registrierten Sturz überhaupt um eine Person handelt. Ist die Überprüfung erfolgreich, löst der Server die Notfallsequenz aus. Diese besteht darin, dass Benachrichtigungen an die Helfer gesendet und eine Nachricht an das Kommunikationssystem des Living Place gesendet werden, um die Türschlösser der Wohnung zu entriegeln. Der Server spielt eine entscheidende Rolle bei der Sicherstellung einer schnellen und zuverlässigen Notfallreaktion im Smart Home System.

Helfer:
Helfer sind externe Akteure, die im Notfall über das Smart Home-System alarmiert werden. Ihre Hauptaufgabe ist die Unterstützung und Hilfe für die gestürzte Person.

Türschließsystem:
Das Türschließsystem ist für die Sicherung und Kontrolle des Zugangs zum Wohnraum verantwortlich. Im Notfall spielt es eine zentrale Rolle, indem es auf Anweisung die Türschlösser entriegelt. Dies ermöglicht den Rettungskräften den Zugang zur Wohnung, um schnell Hilfe leisten zu können. Das Türschlosssystem gewährleistet die physische Sicherheit und erleichtert die reibungslose Umsetzung der Notfallreaktion im Smart Home

System.

Anwendungsfälle:
Um die Funktionsweise des neu entwickelten Verifikationssystems zu erläutern und dessen Integration in das bestehende Sturzerkennungssystem zu veranschaulichen, wurden Use Cases konzipiert. Die Definition dieser Anwendungsfälle macht die Funktionalität des Verifikationssystems greifbar.

Use Case 1:	Nachricht an Kommunikationssystem senden
Akteur:	Sturzerkennungssystem
Ziel:	Übermittlung einer Nachricht an das Kommunikationssystem im Living Place, sobald ein Sturz erkannt wurde, inklusive Position des Sturzes.
Vorbedingung:	Ein Sturz wurde durch das Sturzerkennungssystem erfasst.
Hauptfluss:	- Das Sturzerkennungssystem erkennt einen Sturz. - Das System generiert eine Nachricht, dass ein Sturz erkannt wurde und die Position beinhaltet. - Die Nachricht wird an das Kommunikationssystem des Living Place gesendet.
Ausnahmefluss:	Keine Verbindung zum Kommunikationssystem möglich.

Tabelle 3.1: Nachricht an das Kommunikationssystemsenden

Use Case 2:	Nachrichtenempfang durch den Server
Akteur:	Server
Ziel:	Empfang der Nachricht vom Kommunikationssystem.
Vorbedingung:	Nachricht wurde vom Sturzerkennungssystem gesendet.
Hauptfluss:	- Das Kommunikationssystem empfängt die Nachricht. - Die Nachricht wird an den Server weitergeleitet. - Der Server empfängt die Nachricht.
Ausnahmefluss:	Nachricht kann nicht zugestellt werden.

Tabelle 3.2: Nachrichtenempfang auf dem Server

Use Case 3:	Unfallortbestimmung
Akteur:	Server
Ziel:	Bestimmung des Unfallorts aus der empfangenen Nachricht.
Vorbedingung:	Nachricht wurde vom Serevr empfangen.
Hauptfluss:	- Der Server analysiert die empfangene Nachricht. - Der Server extrahiert die Information über den Unfallort.
Ausnahmefluss:	Unfallort kann nicht bestimmt werden.

Tabelle 3.3: Unfallortbestimmung

Use Case 4:	Navigation starten
Akteur:	Server und Roboter
Ziel:	Navigation Framework auf dem Roboter starten.
Vorbedingung:	Server und Roboter sind mit einander verbunden.
Hauptfluss:	- Der Server sendet ein Kommando welcher die Navigation auf dem Roboter startet.
Ausnahmefluss:	Keine verbindung zwischen Server und Roboter

Tabelle 3.4: Navigation starten

Use Case 5:	Übermittlung von Start- und Zielposition
Akteur:	Server
Ziel:	Befehlsübertragung der Start- und Zielposition an den Roboter
Vorbedingung:	- Server und Roboter sind mit einander verbunden. - Navigation Framework gestartet
Hauptfluss:	- Der Server teilt dem Roboter mit an welcher Startposition er sich befindet. - Der Server teilt dem Roboter mit an welche Zielposition er sich bewegen soll.
Ausnahmefluss:	Keine verbindung zwischen Server und Roboter.

Tabelle 3.5: Start- und Zielposition übermitteln

Use Case 6:	Start- und Zielposition setzen.
Akteur:	Roboter
Ziel:	Roboter nutzt Start- und Zielposition für die Navigation.
Vorbedingung:	- Server und Roboter sind mit einander verbunden. - Navigation Framework gestartet
Hauptfluss:	- Der Roboter teilt der Navigaton seine Startposition mit. - Der Roboter der Navigaton seine Zielposition mit.
Ausnahmefluss:	Keine verbindung zwischen Server und Roboter.

Tabelle 3.6: Start- und Zielposition setzen

Use Case 7:	Anfahrt zum Unfallort
Akteur:	Roboter
Ziel:	Fahrt zum Unfallort.
Vorbedingung:	- Navigation Framework funktioniert - Unfallort wurde der Navigation mitgeteilt.
Hauptfluss:	- Der Roboter plant die Route zum Unfallort. - Der Roboter fährt zum Unfallort.
Ausnahmefluss:	Roboter kann den Unfallort nicht erreichen.

Tabelle 3.7: Fahrt zum Unfallort

Use Case 8:	Bilderaufnahme
Akteur:	Roboter
Ziel:	Der Roboter nimmt Kamerabilder während seiner Fahrt auf.
Vorbedingung:	- Roboter fährt zum Unfallort - Kamera ist funktionsfähig.
Hauptfluss:	- Der Roboter überträgt seine Kamerabilder - Der Roboter fährt zum Unfallort.
Ausnahmefluss:	Kamera funktioniert nicht.

Tabelle 3.8: Bilderaufnahme

Use Case 9:	Bildübertragung an den Server
Akteur:	Roboter
Ziel:	Übertragung der Bilder an den Server.
Vorbedingung:	Roboter nimmt Bilder auf.
Hauptfluss:	- Der Roboter initiiert die Übertragung der Bilder. - Bilder werden erfolgreich an den Server gesendet.
Ausnahmefluss:	Übertragung fehlschlägt.

Tabelle 3.9: Bildübertragung an den Server

Use Case 10:	Personenerkennung auf dem Server
Akteur:	Server
Ziel:	Durchführung einer Personen erkennung auf den empfangenen Bildern.
Vorbedingung:	Bilder werden an den Server gesendet.
Hauptfluss:	- Der Server empfängt die Bilder - Der Server führt eine Personenerkennung auf den Bildern durch.
Ausnahmefluss:	Es wird keine Person auf den Bildern erkannt.

Tabelle 3.10: Personenerkennung auf dem Server

Use Case 11:	Versenden einer Notfallnachricht
Akteur:	Server
Ziel:	Senden einer Notfallnachricht an Helfer, wenn eine Person erkannt wird.
Vorbedingung:	Personenerkennung wurde durchgeführt.
Hauptfluss:	- Eine Person wird auf den Bildern erkannt. - Eine Notfallnachricht wird generiert und an Helfer gesendet.
Ausnahmefluss:	Keine Person erkannt.

Tabelle 3.11: Notfallnachricht versenden

Use Case 12:	Türschlossentriegelung
Akteur:	Server
Ziel:	Entriegelung der Türen im Living Place, um Helfern Zugang zu ermöglichen.
Vorbedingung:	Notfallnachricht wurde an Helfer gesendet.
Hauptfluss:	- Der Server sendet ein Signal an das Kommunikationssystem im Living Place. - Die Türen werden entriegelt.
Ausnahmefluss:	Türen können nicht entriegelt werden.

Tabelle 3.12: Türschlossentriegelung

3.4.1 Anforderungsanalyse

Funktionale Anforderungen an das Verifikationssystem:

Basierend auf den bisherigen Anforderungen und Use Cases lassen sich die funktionalen Anforderungen an das Verifikationssystem ableiten.

Bilderkennung und -verarbeitung:

1. Das System muss in der Lage sein, Bilder von einer Kamera in Echtzeit zu empfangen.
2. Es verwendet eine Bibliothek zur Erkennung von Personen auf den Bildern

Kommunikation zwischen den Komponenten:

1. Das System verwendet ein passendes Nachrichtenprotokoll.
2. Das verwendete Kommunikationssystem muss in der Lage sein, auf Nachrichten zu reagieren.

Steuerung des Roboters:

1. Das System beinhaltet die Nutzung eines Roboters. Es muss ermöglicht werden, diesen zu Steuern und seine Kamera anzubinden.
2. Der Roboter muss sich in einem Raum navigieren können.

Annahmen und Einschränkungen für die Entwicklung:

Annahmen:

Gestürzte Person: Das Verifikationssystem zielt darauf ab, eine auf dem Boden liegende Person zu erkennen, um die zuvor erfolgte Sturzmeldung zu verifizieren. Es wird angenommen, dass die Person nicht mehr dazu in der Lage ist sich selbständig aufzurichten und Hilfe zu rufen.

Sturzerkennung: Es wird angenommen, dass das Sturzerkennungssystem, funktionsfähig ist und verlässliche Daten liefert.

Funktionsfähigkeit des Kommunikationssystems: Es wird angenommen, dass das Kommunikationssystem im Living Place ordnungsgemäß funktioniert und Nachrichten zuverlässig übertragen kann.

Funktionsfähiger und konfigurierbarer Roboter: Es wird angenommen, dass der Roboter einsatzbereit und für den Anwendungsfall konfigurierbar ist.

Verbindung zum Server: Es wird angenommen, dass eine stabile Netzwerkverbindung zwischen dem Roboter und dem Server besteht, um Bilder und Informationen effektiv zu übertragen.

Personenerkennung: Es wird angenommen, dass die Personenerkennung auf dem Server eine angemessene Genauigkeit aufweist, um sicherzustellen, dass nur echte Personen erkannt werden.

4 Lösungsansatz

Basierend auf den vorstehenden benannten Anforderungen wurde das Verifikationssystem konzipiert. Die Anforderungen bilden die Grundlage für die Auswahl und Implementierung der notwendigen Komponenten. Ein wesentlicher Bestandteil ist ein Server, der die zentrale Datenverarbeitung und -kommunikation übernimmt. Zusätzlich wird ein mobiler Roboter benötigt, der mit einer Kamera ausgestattet ist, navigieren und mit dem Server kommunizieren kann.

4.1 Auswahl der Hardware und Software

4.1.1 Roboter

Der Roboter, der für die Umsetzung des Verifikationssystems benötigt wird, muss navigieren können und mit einer Kamera ausgestattet sein, um Bilder an den Server zu übertragen. Hierfür ist der Turtlebot3 Waffle Pi von Robotis [19] besonders geeignet.

Der Turtlebot3 Waffle Pi von Robotis ist aufgrund seiner Ausstattung und Mobilität für das Projekt geeignet. Er ist mit einem Raspberry Pi, einem Lidar-Sensor und einer Kamera ausgestattet und erfüllt somit die unter 3.4 definierten Anforderungen. Die Kombination dieser Komponenten ermöglicht eine Navigation und Umgebungserfassung, die für das Verifikationssystem essentiell sind. Die Integration des Raspberry Pi und die Installation von ROS2 Foxy [12] bietet eine flexible Plattform für die Programmierung und Datenverarbeitung. Der Lidar-Sensor ermöglicht eine detaillierte Kartierung der Umgebung und die Kamera erweitert die Fähigkeiten des Roboters um visuelle Datenverarbeitung. Insgesamt ist der Turtlebot3 Waffle Pi eine umfassende Lösung, die den technischen Anforderungen des Projekts gerecht wird und eine zuverlässige Basis für die Implementierung des Verifikationssystems bietet. Die Abbildung 4.1 zeigt den Turtlebot3 Waffle Pi, mit seinen Komponenten:

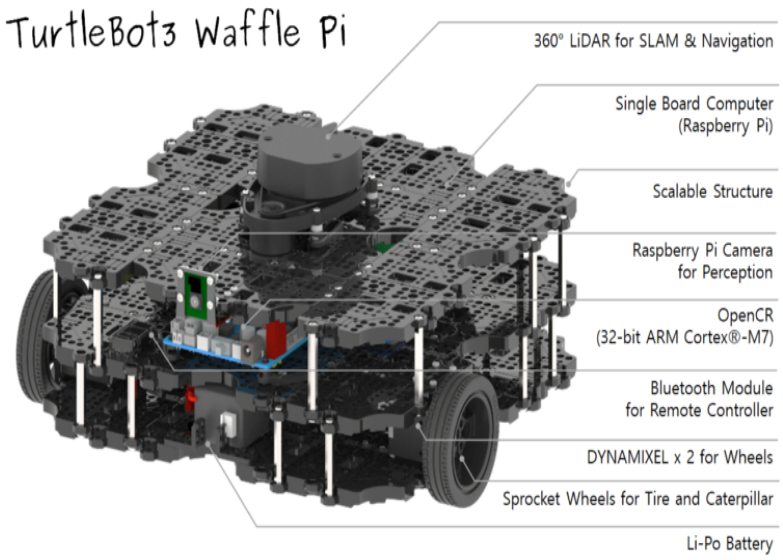


Abbildung 4.1: Turtlebot3 Waffle Pi[21]

4.1.2 Server

Der zentrale Server im Verifikationssystem spielt eine entscheidende Rolle, da er die Bildverarbeitung durchführen, Nachrichten mit dem Kommunikationssystem austauschen und mit dem Roboter interagieren muss. Für eine optimale Zusammenarbeit mit dem Turtlebot3 Waffle Pi, der unter ROS2 Foxy läuft, wird die Installation eines Linux-basierten Betriebssystems empfohlen. Linux wird bevorzugt, da es einen breiteren Support und eine umfangreichere Dokumentation für ROS bietet, was die Entwicklung und Fehlerbehebung erleichtert. Ubuntu Linux - Focal Fossa (20.04) ist nach dieser Maßgabe für ROS2 Foxy am besten geeignet.

In der Entwicklungsphase des Verifikationssystems wird keine hohe Rechenleistung für die Bildverarbeitung und Kommunikation benötigt, was den Einsatz eines Laptops der Mittelklasse ermöglicht. Dieser eignet sich gut für die Entwicklung, da er eine schnelle und einfache Implementierung sowie das Testen des Codes ermöglicht. Damit wird eine effiziente und agile Entwicklungsumgebung geschaffen, die eine solide Basis für den Betrieb und die Weiterentwicklung des Verifikationssystems bietet.

In dieser Arbeit wird ein Lenovo Thinkbook 14 ITL mit folgender Ausstattung verwendet:

- Betriebssystem: Ubuntu Linux - Focal Fossa (20.04)
- Prozessor: 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.80 GHz
- Arbeitsspeicher: 16 GB RAM
- Festplatte: 1TB SSD
- Grafikeinheit: Intel® Iris® Xe Graphics
- Systemtyp: 64-Bit-Betriebssystem, x64-basierter Prozessor

4.1.3 Kommunikationssystem

Im Living Place ist ein Kommunikationssystem integriert, das den Austausch von MQTT-Nachrichten ermöglicht. MQTT (Message Queuing Telemetry Transport) [6] ist ein leichtgewichtiges Messaging-Protokoll, das auf Publish-Subscribe basiert. Es wurde speziell für die effiziente Kommunikation in Netzwerken mit geringer Bandbreite und Geräten mit begrenzten Ressourcen wie IoT-Geräten entwickelt. Es ermöglicht eine zuverlässige und schnelle Übertragung von Nachrichten zwischen Geräten. Dieser Austausch wird durch einen MQTT-Broker realisiert, der als zentrale Schaltstelle für die Nachrichtenübermittlung zwischen den einzelnen Komponenten dient. Zusätzlich steht Node-RED [8] zur Verfügung, eine visuelle Programmierumgebung, mit der Datenflüsse erstellt und bearbeitet werden können. Die Hauptfunktionen von Node-RED sind das Sammeln, Verarbeiten und Weiterleiten von Informationen. Im Rahmen dieser Arbeit wird Node-RED genutzt, um auf bestimmte MQTT-Nachrichten zu reagieren und daraufhin Skripte auf dem Server zu starten. Dadurch können automatisierte Aktionen und Prozesse effizient umgesetzt werden können.

4.1.4 Türschlosssystem

Das Living Place bietet auch die Möglichkeit, die Haustürschlösser bei Eingang einer bestimmten MQTT-Nachricht auf das passende Topic zu öffnen oder zu schließen.

4.1.5 ROS 2

Das Robot Operating System (ROS) wird zur Steuerung des Turtlebot3 Waffle Pi eingesetzt. In dieser Arbeit wird die Version ROS2 Foxy verwendet. ROS ist sowohl auf dem Roboter als auch auf dem Server installiert, um die Kommunikation und Steuerung zwischen den Komponenten zu ermöglichen. Zudem bietet ROS eine Vielzahl nützlicher Visualisierungstools, mit denen sich die Zustände des Roboters überwachen und verfolgen lassen.

ROS 2 (Robot Operating System 2) ist ein Open-Source-Framework zur Entwicklung und Steuerung von Robotern. Es nutzt eine verteilte Architektur, in der Komponenten als Knoten (Nodes) agieren. Diese Knoten führen spezifische Aufgaben aus und kommunizieren miteinander über Topics, Services und Actions.

- Nodes: Autonome Einheiten mit spezifischen Funktionen, die den Roboter steuern und Informationen über Topics austauschen.
- Topics: Kanäle, die den asynchronen Nachrichtenaustausch zwischen Nodes ermöglichen, um Daten wie Sensorinformationen zu übertragen.
- Services: Ermöglichen synchrone Interaktionen, bei denen ein Node einen Service anbietet und andere diesen aufrufen, um Aufgaben zu erfüllen.
- Actions: Unterstützen die asynchrone Durchführung langwieriger Aufgaben, indem sie eine Reihe von Aktionen zwischen Nodes definieren.

4.1.6 OpenCV

In dieser Arbeit wird OpenCV verwendet, um vorhandene Bilder zu manipulieren und mit ihnen zu arbeiten. OpenCV ist eine umfangreiche Open-Source-Bibliothek für Computer Vision, die es ermöglicht, Bildverarbeitungsaufgaben effizient durchzuführen, wie zum Beispiel das Erkennen und Verarbeiten von Bildern, um nützliche Informationen daraus zu extrahieren. Diese Funktionalität ist entscheidend, um dem Roboter die Wahrnehmung seiner Umgebung und entsprechende Interaktionen zu ermöglichen.

ROS2 erleichtert diesen Prozess durch die Bereitstellung einer CV-Bridge, die eine nahtlose Übertragung von Bildern vom Roboter zum Server ermöglicht. Die CV-Bridge konvertiert ROS-Bildnachrichten in ein Format, das von OpenCV verwendet werden kann

und umgekehrt. Dadurch wird die Integration von Bildverarbeitungsfunktionen in das ROS2-System vereinfacht. Durch die Synergie zwischen OpenCV und ROS2 können Bilder direkt vom Roboter erfasst, zum Server übertragen, dort verarbeitet und für eine Vielzahl von Anwendungen genutzt werden, in dieser Arbeit werden die empfangenen Bilder mit YOLOv8 zur Erkennung von Personen und Posen verwendet.

4.1.7 YOLOv8 Pose Estimation

Wie bereits in Kapitel 2 erläutert, gibt es verschiedene Methoden zur Durchführung einer Pose Estimation. In Zusammenhang mit dem Verifikationssystem, dessen primäres Ziel es ist, einen Sturz zweifelsfrei zu verifizieren und somit sicherzustellen, dass tatsächlich eine Person gestürzt ist, wird in dieser Arbeit eine Pose Estimation Methode verwendet, die dem Top-Down-Ansatz folgt. Dieser Ansatz beginnt mit einer ganzheitlichen Analyse des Bildes, um eine Person zu identifizieren und eine Bounding Box um sie herum zu erstellen. Anschließend werden die Körper-Keypoints innerhalb dieser Bounding Box detektiert.

Für diesen spezifischen Ansatz ist die Verwendung von YOLOv8 Pose besonders geeignet, da sie präzise und effizient die erforderlichen Informationen aus dem Bild extrahiert, um eine genaue Pose Estimation zu ermöglichen.

YOLOv8 [45] ist die achte Iteration von „You Only Look Once“ (YOLO), einem fortschrittlichen Deep-Learning-Modell, das für die Erkennung von Objekten in Bildern und Videos konzipiert ist. Wie seine Vorgänger kombiniert YOLOv8 Geschwindigkeit und Genauigkeit, indem es die Objekterkennung in nur einem Durchlauf durch das neuronale Netzwerk ermöglicht. Dadurch ist es deutlich schneller als herkömmliche zweistufige Erkennungssysteme.

Die Pose Estimation-Funktion in YOLOv8 ist ein erweitertes Feature, das darauf abzielt, die Haltung oder Position von Personen in Bildern oder Videos zu erkennen und zu analysieren. Diese Funktion nutzt das leistungsstarke Objekterkennungsframework YOLOv8, um einzelne Körperteile zu identifizieren und ihre räumlichen Beziehungen zueinander zu bestimmen. Dadurch kann YOLOv8 nicht nur erkennen, wo sich eine Person im Bild befindet, sondern auch ihre Pose bestimmen. Dies ist besonders nützlich für Anwendungen wie Sportanalyse, fortgeschrittene Überwachungssysteme oder Interaktionen zwischen Mensch und Computer.

Außerdem können die einzelnen Modelle von YOLOv8 auf eigenen Datensätzen trainiert werden. Dies ist ein weiterer Grund, auf diese Technologie zu setzen. Sie bietet die Möglichkeit, individuelle Ideen in Bezug auf Pose Estimation und Personenerkennung zu entwickeln.

4.1.8 Navigation

Der Turtlebot3 Waffle Pi ist dazu in der Lage das Navigation2 Framework von ROS 2 zu nutzen. Navigation2 (Nav2) ist ein erweitertes modulares und portables Navigation Framework, das für ROS 2 entwickelt wurde. Es bietet eine Sammlung von Paketen, die eine breite Palette von Navigationstätigkeiten für mobile Roboter abdecken, von der Wegplanung über die Hindernisvermeidung bis hin zur Geschwindigkeitsregelung. Damit die Navigation mit Nav2 erfolgreich funktioniert, sind mehrere Schritte und Komponenten involviert, welche zusammenwirken [7].

- Kartenbereitstellung: Zunächst benötigt Nav2 eine Karte der Umgebung. Diese kann entweder im Vorfeld kartiert und dann bereitgestellt werden oder durch Prozesse wie SLAM (Simultaneous Localization and Mapping) in Echtzeit erstellt werden.

- Lokalisierung: Der Roboter muss seine Position auf der Karte kennen. Dies wird typischerweise durch Lokalisierungspakete wie AMCL (Adaptive Monte Carlo Localization) erreicht. Diese nutzen Sensorinformationen, um die Position und Orientierung des Roboters auf der Karte zu schätzen.

- Sensorintegration: Nav2 benötigt Daten von Sensoren des Roboters, wie Lidar, Kamera, Ultraschall oder Radencoder, um die Umgebung wahrzunehmen und die Position des Roboters zu bestimmen.

- Wegplanung: Der Global Planner in Nav2 verwendet die Karte und die aktuelle Position des Roboters, um einen optimalen Weg zum Ziel zu finden. Dieser Weg muss Hindernisse vermeiden und kann unter Verwendung verschiedener Algorithmen, berechnet werden.

- Controller (Steuerung): Ein lokaler Controller nutzt den vom globalen Planer berechneten Pfad, um Geschwindigkeitsbefehle (linear und angular) zu generieren und den Roboter entlang des Pfades zu steuern. Dabei werden dynamische Hindernisse vermieden.

- Recovery-Verhalten: Wenn der Roboter feststellt, dass er nicht weiterkommen kann (zum Beispiel, wenn er in einer Sackgasse steckt), können Recovery-Strategien aktiviert werden, um den Roboter zu befreien und die Navigation fortzusetzen.
- Lifecycle-Management: Nav2 verwendet das ROS 2 Lifecycle-Modell für den Betrieb seiner Knoten. Das bedeutet, dass jeder Knoten in Nav2 bestimmte Zustände wie Konfiguration, Aktivierung, Inaktivität usw. durchläuft.
- Benutzerinteraktion: Für die Navigation muss der Benutzer (oder ein übergeordnetes System) ein Ziel setzen. Dies geschieht üblicherweise durch Senden eines NavigateToPose-Ziels über eine ROS 2-Action-Schnittstelle.

Für eine erfolgreiche Nutzung von Nav2 müssen alle Komponenten korrekt konfiguriert und gestartet werden. Der Roboter muss mit der entsprechenden Sensorik und den nötigen Rechenressourcen ausgestattet sein.

4.2 Systementwurf

Anhand der Anforderungen aus Kapitel 3 und der Auswahl an Hard- und Software aus 4.1 kann ein Flussdiagramm entworfen werden, welches den Ablauf des Verifikationssystems beschreibt.

Die Abbildung 4.2 beschreibt das Szenario, welches durch das Verifikationssystem umgesetzt werden soll. Das Flussdiagramm zeigt den groben Ablauf des gesamten Systems, ohne dabei detailliert auf notwendige Zwischenschritte und Interaktionen der einzelnen Komponenten einzugehen. Diese werden jedoch in den folgenden Kapiteln erarbeitet und dargestellt.

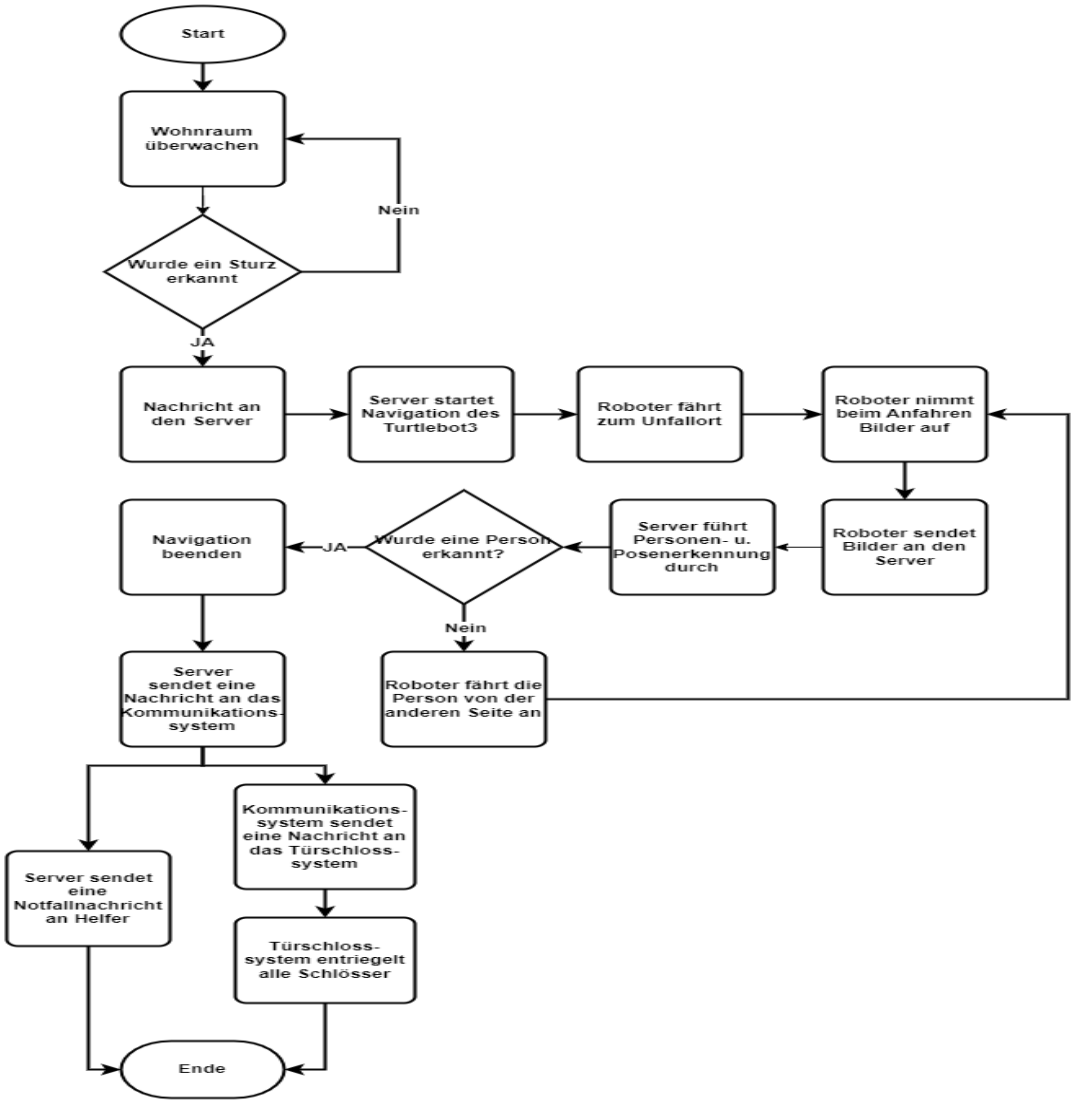


Abbildung 4.2: Veranschaulichung des Verifikaitonssystems als Flussdiagramm

In den folgenden Abschnitten wird die Architektur und das Design des Verifikationssystems detailliert durch ein Komponenten- und Klassendiagramm veranschaulicht. Diese Diagramme spielen eine wesentliche Rolle bei der Verdeutlichung der Struktur und Interaktionen innerhalb des Systems. Das Komponentendiagramm 4.2.1 bietet zunächst einen Überblick über die einzelnen Bestandteile des Systems und wie sie miteinander interagieren. Anschließend liefert das Klassendiagramm 4.2.2 eine tiefere Einsicht in die spezifischen Bausteine des Systems. Es zeigt, wie die verschiedenen Klassen konzipiert sind, um die benötigten Funktionalitäten des Verifikationssystems zu implementieren.

4.2.1 Komponentendiagramm

Die Auswahl von Hardware und Software unter 4.1 sowie das Flussdiagramm 4.2 veranschaulichen die notwendigen Bausteine für die Implementierung des Verifikationssystems. Das System interagiert mit einigen bereits bestehenden Systemen wie dem Türschloss- und Kommunikationssystem. Allerdings sind zusätzliche Entwicklungen erforderlich, um alle Funktionalitäten zu gewährleisten. Der Turtlebot3 Waffle Pi muss effizient mit dem Server kommunizieren können. Dazu muss der Server angepasst werden, um einen reibungslosen Datenaustausch mit dem Roboter zu ermöglichen. Auf dem Server werden zentrale Funktionen wie Bildverarbeitung, präzise Navigation des Roboters und das Versenden von Notfallnachrichten implementiert, um das Verifikationssystem vollständig zu realisieren.

Die Abbildung 4.3 gibt einen Überblick über die Pakete und Komponenten. Besonders wichtig sind die Server-Komponenten, die im Rahmen dieser Arbeit entwickelt wurden, um die Funktionalität des Verifikationssystems zu ermöglichen.

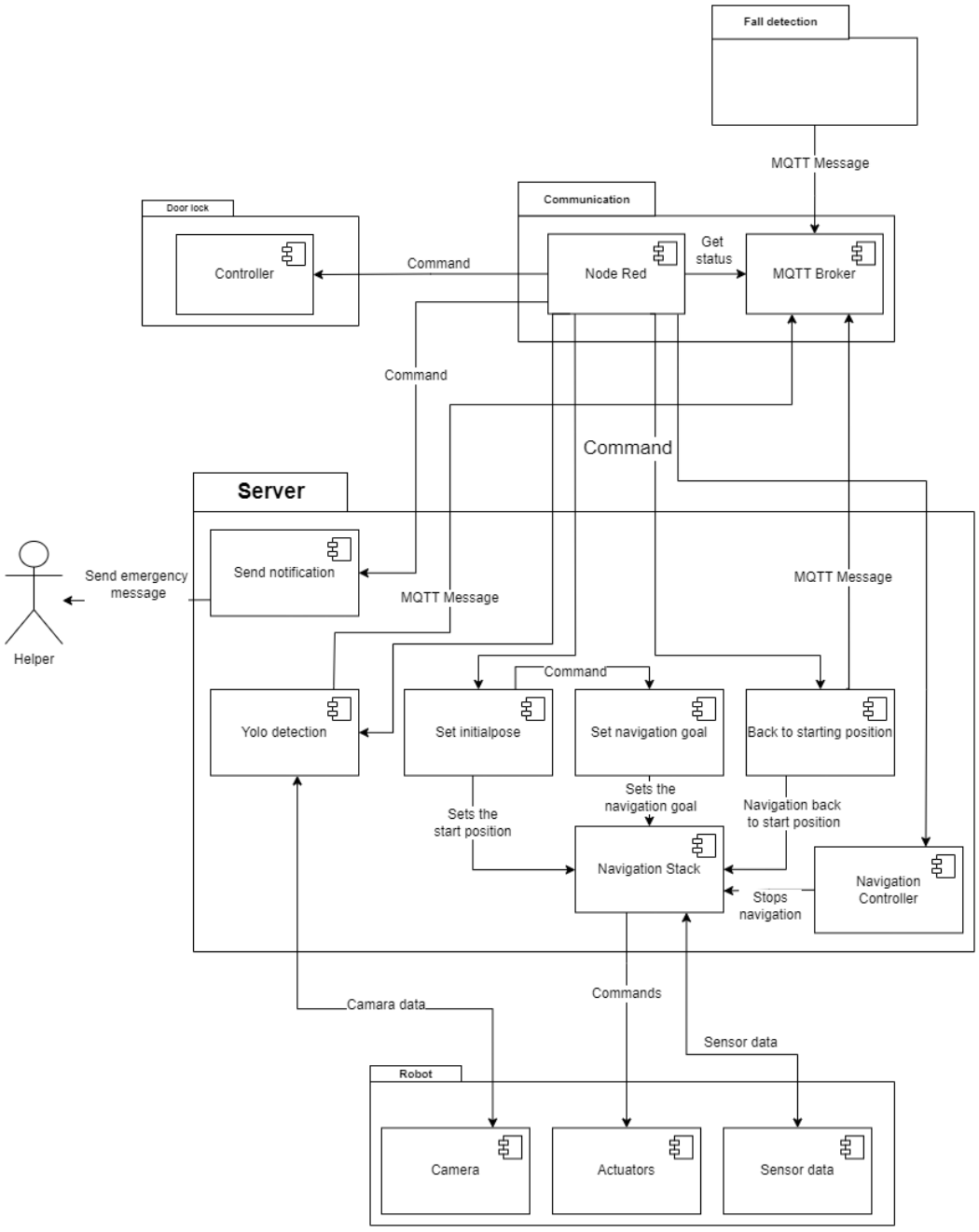


Abbildung 4.3: Komponentendiagramm des Verifikationssystems

Die vorhandenen Pakete **„Door lock“, „Communication“ und „Fall detection“** sowie ihre jeweiligen Komponenten sind bereits einsatzbereit. Die Serverkomponenten wurden speziell entwickelt, um eine effiziente Kommunikation mit dem Kommunikationssystem zu gewährleisten. Ebenso verfügen die Roboter-Komponenten **„Camera“, „Actuators“ und „Sensor data“** über Schnittstellen, die für die Entwicklung auf dem Server von entscheidender Bedeutung sind.

Die Serverkomponenten gliedern sich in drei Hauptpakete. Das **Notification-Paket** umfasst die Komponente *„Send notification“*, während *„Yolo detection“* dem **Image processing-Paket** zugeordnet ist. Das **Navigation-Paket** besteht aus den Komponenten *„Set initialpose“, „Set navigation goal“, „Back to starting position“, „Navigation Stack“ und „Navigation controller“*. Dabei ist zu beachten, dass der *„Navigation Stack“*, der eine Sammlung von ROS 2-Tools innerhalb des Nav2-Frameworks umfasst, nicht neu entwickelt werden muss. Um den Navigation Stack effektiv zu nutzen, müssen ihm Informationen aus den anderen Komponenten zugeführt werden, um eine präzise Navigation und Steuerung des Roboters zu ermöglichen, wie bereits in Kapitel 4.1.8 beschrieben wurde.

Das Komponentendiagramm gibt eine Übersicht darüber, wie die einzelnen Bestandteile des Systems miteinander interagieren. Das Sturzerkennungssystem nutzt MQTT, um im Falle eines Sturzes eine Nachricht an das Kommunikationssystem zu senden. *Node Red* reagiert auf den Eingang dieser Nachricht, indem es interne Prozesse anstößt und Kommandos zur Initiierung der Navigation und Bildverarbeitung an den Server sendet. Die Komponenten Yolo Detection, Set Initialpose und Set Navigation Goal werden aktiviert.

- *Set initialpose*: teilt dem Navigation Stack die Startposition des Roboters mit und leitet die Komponente Set Navigation Goal ein.
- *Set navigation goal:* bestimmt das Navigationsziel und kommuniziert dieses an den Navigation Stack.
- *Yolo detection:* tauscht Kameradaten mit dem Roboter aus und analysiert diese, um festzustellen, ob eine Person im Bild erkannt wird.

Wenn *Yolo Detection* eine Person erfolgreich erkennt, wird eine Benachrichtigung an das Kommunikationssystem gesendet. Dadurch werden weitere Prozesse in *Node Red* aktiviert und die Komponenten *Send Notification* und *Back to Starting Position* gestartet.

- *Send notification:* ist zuständig für das Versenden einer Notfallnachricht an definierte Kontakte.
- *Back to starting position:* navigiert den Roboter zurück zur Ausgangsposition und signalisiert das Erreichen dieser an das Kommunikationssystem.

Nachdem die Rückkehr bestätigt ist, löst *Node Red* ein Kommando zur Aktivierung der *Navigation Controller* Komponente aus, welche die Navigation stoppt und alle zugehörigen Prozesse beendet.

Der *Navigation Stack* steht während des gesammten Prozesess in kontinuierlichem Austausch mit den Aktuator- und Sensordaten des Roboters, um eine präzise Navigation sicherzustellen.

4.2.2 Klassendiagramm

Vorangegangene Ausführungen lieferten wichtige Einsichten in die verschiedenen Komponenten des Verifikationssystems. Sie identifizierten jene, die noch entwickelt werden mussten und boten eine kurze Übersicht über deren Funktionen.

Nachfolgend wird das Verständnis der Serverkomponenten vertieft. Es wird gezeigt, wie diese implementiert wurden, um die Funktionalitäten des Verifikationssystems zu realisieren. Ein detailliertes Klassendiagramm veranschaulicht die Struktur und Interaktion dieser Komponenten. Im Folgenden werden die spezifischen Klassen und ihre Funktionen erläutert, um ein umfassendes Verständnis dafür zu vermitteln, wie sie gemeinsam das Verifikationssystem bilden und seine Funktionalität sicherstellen.

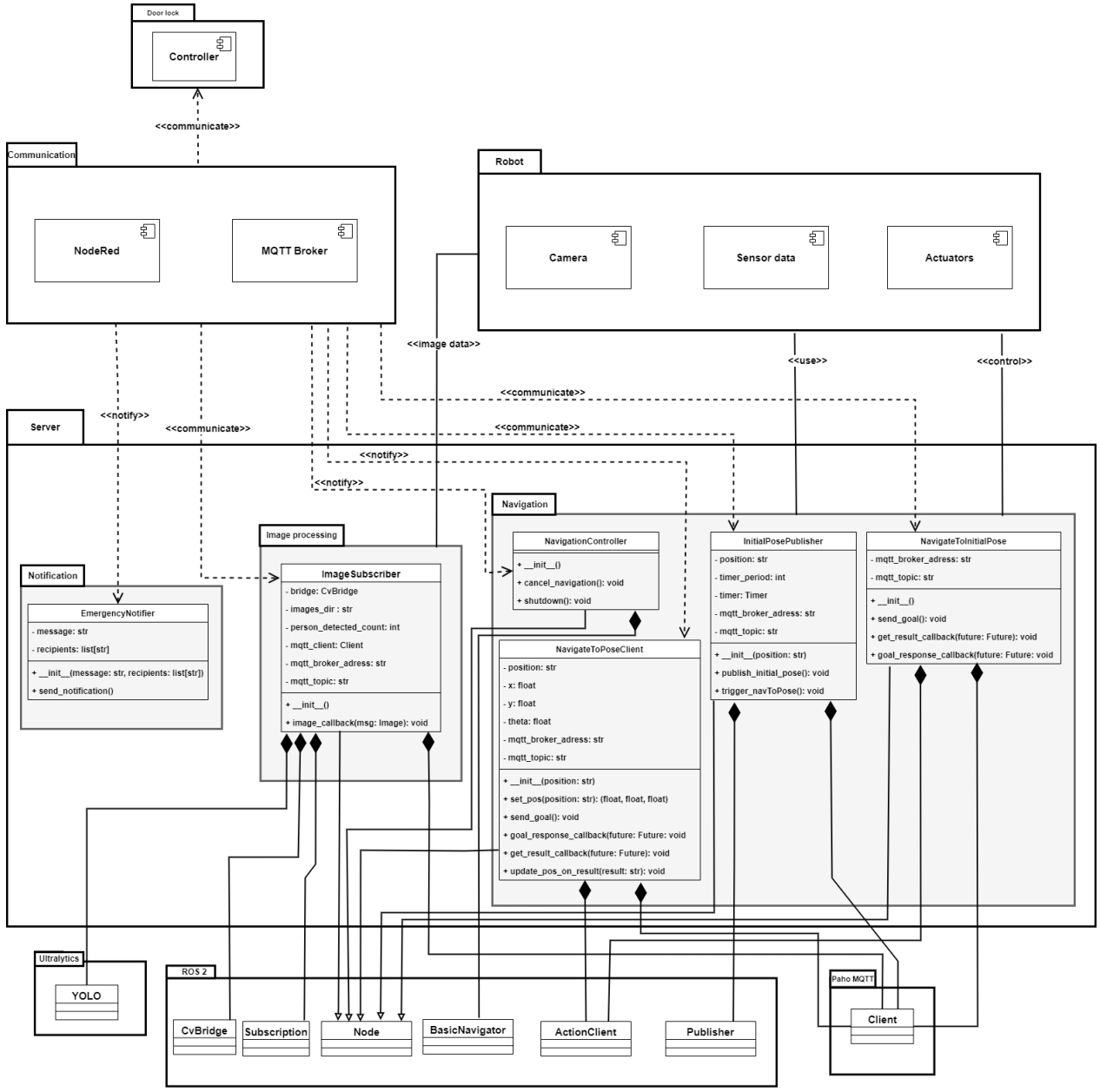


Abbildung 4.4: Klassendiagramm des Verifikationssystems

Beim Eingang einer Sturzmeldung initiiert das Kommunikationssystem über Node Red einen Prozess, der dem Server ein Signal sendet. Dieser Prozess startet zunächst die ROS2-Navigation und aktiviert dann spezifische Skripte, um die Navigation des Roboters effektiv zu steuern. Diese Skripte umfassen 1. den *InitialPosePublisher*, der die

Startposition des Roboters festlegt, 2. den *NavigateToPoseClient*, der dem Roboter sein Ziel mitteilt, und 3. den *ImageSubscriber*, der die Bildaufnahmen des Roboters abruft.

Die Klasse **InitialPosePublisher** ist ein ROS 2-Node, der dazu dient, eine initiale Pose (Position und Orientierung) für einen Roboter zu veröffentlichen. Sie ist in Python geschrieben und nutzt die ROS 2 Python-Bibliothek *rclpy*. Die Klasse veröffentlicht wiederholt eine initiale Position und führt dann einen weiteren Befehl aus, wenn eine bestimmte Bedingung erfüllt ist. Die initiale Position muss dem Navigation Stack mitgeteilt werden, damit dieser weiß, wo sich der Roboter auf der Karte befindet. Es gibt einen Counter und Timer, weil die Startposition mehrmals veröffentlicht werden muss, damit der Navigation Stack sie annimmt. Im Folgenden wird die Funktionsweise der Klasse und ihrer Methoden erläutert:

- *__init__(position_x):* Dies ist der Konstruktor der Klasse. Hier wird der Node initialisiert und ein Publisher für das Topic „/initialpose“ erstellt. Außerdem wird ein Timer gesetzt, um die Pose regelmäßig zu veröffentlichen, und einige Initialwerte werden gesetzt, wie die gemeldete Sturzposition und der Counter, der zählt, wie oft die Pose veröffentlicht wurde.

- *publish_initial_pose():* Die Methode wird regelmäßig vom Timer aufgerufen. Dabei wird ein „PoseWithCovarianceStamped-Objekt“ mit vordefinierten Werten für die Pose erstellt und auf dem */initialpose* Topic veröffentlicht. Dies geschieht bis zu fünf Mal und wird durch den Counter überwacht. Nachdem die Pose fünf Mal veröffentlicht wurde, wird der Timer gestoppt und die *trigger_navToPose()* Methode aufgerufen.

  Im Kontext von ROS2 wird die Pose eines Roboters durch drei Komponenten beschrieben: *x, y* und *Theta*. Diese Komponenten stellen die Position und Orientierung des Roboters im zweidimensionalen Raum dar. Theta ist der Winkel und repräsentiert die Orientierung oder Ausrichtung des Roboters im Raum.

- *trigger_navToPose():* Diese Methode protokolliert die aktuelle Position und führt anschließend einen Subprozess aus. Dieser Subprozess startet das *NavigateToPoseClient* Skript. Erst nach Übermittlung der Startposition in der Navigation darf ein Navigationsziel basierend auf der zuvor vom Kommunikationssystem empfangenen Position übergeben werden. Nach Ausführung dieses Befehls wird *rclpy.shutdown()* aufgerufen, um den Node sauber herunterzufahren.

Die Klasse dient hauptsächlich dazu, eine initiale Position für den Roboter zu setzen und darauf aufbauend eine weitere Aktion auszulösen, die von der initialen Position abhängt.

Die nächste Komponente, die **NavigateToPoseClient**-Klasse, wird gestartet, nachdem die Startposition festgelegt wurde. Die NavigateToPoseClient Klasse ist ein ROS 2 Node, der entwickelt wurde, um den Turtlbot3 Waffle Pi bei der Navigation zu vorgegebenen Positionen zu unterstützen. Dies geschieht durch Kommunikation mit dem Navigation 2 (Nav2) Stack.

Nachfolgend wird der Aufbau und die Funktionsweise dieser Klasse beschrieben:

- *__init__(position):* Der Konstruktor der Klasse. Er initialisiert den Node, erstellt einen Action-Client für die *NavigateToPose-Action* und legt die anfängliche Zielposition fest. Die Zielkoordinaten (x, y und theta) werden durch Aufruf der Methode *determine_position* festgelegt.
- *determine_position(position):* Die Methode wählt die Zielposition anhand des übergebenen Parameters aus einem vordefinierten Positionsverzeichnis aus. Die Zielposition ist ein Koordinaten-Tupel (x, y, theta), welches die Zielposition und -orientierung des Roboters darstellt.
- *send_goal():* Diese Methode sendet ein Navigationsziel an den Nav2 Action Server. Es wird eine NavigateToPoseGoal-Nachricht mit den aktuellen Zielkoordinaten und der Orientierung (x, y, theta) erstellt und über den Action-Client gesendet. Zusätzlich wird ein Callback für die Antwort des Action-Servers registriert.
- *goal_response_callback(future):* Dieser Callback wird aufgerufen, sobald eine Antwort vom Nav2 Action Server eintrifft. Es wird überprüft, ob das Ziel akzeptiert wurde. Anschließend wird ein weiterer Callback registriert, um das Ergebnis der Navigation abzurufen.
- *goal_result_callback(future):* Dieser Callback wird aufgerufen, wenn das Ergebnis der Navigationsanfrage vorliegt. Die Methode bestimmt abhängig vom Ergebnis (erfolgreich oder abgebrochen) die nächste Zielposition und startet eine neue Navigationsanfrage.
- *update_position_based_on_result(result):* Diese Methode aktualisiert die Zielposition, nachdem das vorherige Navigationsziel abgeschlossen wurde. Basierend auf dem aktuellen Ergebnis (erfolgreich oder abgebrochen) und der aktuellen Position

wird die nächste Position bestimmt, die der Roboter ansteuern soll. Nachdem alle Positionen abgefahren wurden, kehrt der Roboter zu seiner Startposition zurück und meldet, dass kein Sturz festgestellt werden konnte. Es gibt außerdem einen 60-Sekunden-Counter, bevor ein neues Navigationsziel ausgewählt wird. Der Counter hat den Zweck, der paralellel laufenden Bildverarbeitung, genügend Zeit zu bieten. Diese muss schließlich Bilder aufnehmen, verarbeiten sowie eine Entscheidung treffen.

Zusammenfassend ermöglicht diese Klasse dem Roboter, eine vordefinierte Abfolge von Punkten zu navigieren. Nach dem Erreichen oder Abbrechen eines Ziels wird automatisch das nächste Ziel ausgewählt und angesteuert, um sicherzustellen, dass ein Ziel nicht nur von einer Seite aus angefahren wird. Wenn alle vordefinierten Positionen abgefahren wurden und keine Unterbrechungen während der Ausführung aufgetreten sind, fährt der Roboter zur Startposition zurück und meldet, dass kein Sturz verifiziert werden konnte. Sollte der Roboter zur Startposition zurückkehren müssen, würde dies im Kommunikationssystem eine Meldung auslösen, die einen Prozess startet, um das Skript Navigation Controller zu starten.

Die Klasse **ImageSubscriber** ist ein ROS 2-Node, der Bilder von der Kamera des Turtlebot3 Waffle Pi abonniert. Diese Bilder werden mit einem vortrainierten YOLOv8-Modell analysiert, um Personen und ihre Pose zu erkennen. Basierend auf der Erkennung führt der Node spezifische Aktionen aus. Im Folgenden werden die Methoden der Klasse sowie ihre Funktionsweise erläutert:

- *__init__():* Der Konstruktor der Klasse initialisiert den Node, erstellt ein CvBridge-Objekt (um ROS-Bilder in OpenCV-Bilder zu konvertieren), richtet einen Subscriber für das Kamerabild-Topic des Roboters ein, lädt das YOLOv8-Modell, erstellt ein Verzeichnis für erkannte Bilder und initialisiert den MQTT-Client.

- *image_callback(msg):* Die Methode wird jedes Mal aufgerufen, wenn ein neues Bild vom Kamerabild-Topic empfangen wird. Es wird ein Bild pro Sekunde empfangen. Das ROS-Bild wird in ein OpenCV-Bild konvertiert, welches dann in ein PyTorch-Tensor umgewandelt wird. Anschließend wird die Personen- und Posen-Erkennung durchgeführt, um zu prüfen, ob Personen im Bild erkannt wurden. Wenn eine Person erkannt wird, wird das Bild mit Annotationen gespeichert. Sobald eine Person mindestens 20 Mal mit einer Konfidenz von 70% oder höher erkannt wurde, wird

eine MQTT-Nachricht an das Kommunikationssystem gesendet, die bestätigt, dass ein Sturz stattgefunden hat. Anschließend wird der Node heruntergefahren.

Wie beschrieben wird eine Nachricht gesendet, sobald ein Sturzereignis verifiziert wurde. Diese Nachricht löst neue Prozesse im Kommunikationssystem aus. Dabei werden Befehle an den Server gesendet, um die Klassen *EmergencyNotifier* und *NavigateToInitialPose* zu starten.

Die Klasse *EmergencyNotifier* ist dafür vorgesehen, Notfallbenachrichtigungen an eine vordefinierte Liste von Empfängern zu senden.

- *__init__(message, recipients):* Dies ist der Konstruktor der Klasse. Er initialisiert zwei Hauptattribute: message und recipients. Message enthält die zu sendende Notfallnachricht und recipients ist eine Liste von Empfängern, die die Nachricht erhalten sollen.
- *send_notification():* Diese Methode ist dafür verantwortlich, die Notfallbenachrichtigung an alle Empfänger zu senden. In der aktuellen Implementierung wird die Benachrichtigung nicht tatsächlich gesendet. Stattdessen wird für jeden Empfänger in „self.recipients“ eine Nachricht in der Konsole ausgegeben, die simuliert, dass die Nachricht gesendet wird.

Zusammenfassend dient diese Klasse als Framework für ein Notfallbenachrichtigungssystem. Die tatsächliche Logik zum Senden der Nachricht muss, je nach Anforderung, an die reale Welt angepasst werden.

Die Klasse **NavigateToInitialPose** ist ein ROS 2-Node, der entwickelt wurde, um den Roboter nach einem Sturz zur definierten Anfangsposition zurückzuführen. Sie nutzt den Navigation2-Action-Server, um eine Navigationsaufgabe zu starten, und benachrichtigt das Kommunikationssystem, sobald der Roboter die Startposition erreicht hat und somit das Navigationsziel erreicht wurde.

- *__init__():* Im Konstruktor der Klasse wird der Node initialisiert und ein Action-Client für die NavigateToPose-Action eingerichtet. Außerdem wird die vordefinierte initiale Position als Zielposition gesetzt, zu der der Roboter navigieren soll. Zusätzlich wird der MQTT-Client initialisiert und mit dem Broker verbunden.

- *send_ goal():* Die Methode wartet darauf, dass der Navigation2-Action-Server verfügbar ist und sendet dann das Navigationsziel (die initiale Position). Zusätzlich wird ein Callback für die Antwort des Action-Servers registriert.

- *goal_ response_ callback(future):* Dieser Callback wird aufgerufen, sobald eine Antwort vom Action-Server eingeht. Es wird geprüft, ob das Ziel akzeptiert wurde. Falls nicht, wird eine entsprechende Nachricht protokolliert.

- *get_ result_ callback(future):* Dieser Callback wird aufgerufen, sobald das Ergebnis der Navigationsaufgabe verfügbar ist. Es wird das Ergebnis protokolliert und eine Nachricht an das Kommunikationssystem veröffentlicht, die signalisiert, dass die Startposition erreicht wurde. Nach dem Versenden der Nachricht wird der Node heruntergefahren.

Die Klasse funktioniert ähnlich wie die NavigateToPoseClient Klasse. Sie ermöglicht es, das aktuelle Navigationsziel zu überschreiben, falls in diesem Moment eine Aktion ausgeführt wird. Dies geschieht, wenn der Turtlebot3 bereits auf dem Weg zum Ziel eine Person erkannt und somit seine Aufgabe erfüllt hat.. Anschließend fährt er zurück zur Startposition, um auf neue Sturzmeldungen reagieren zu können. Sobald die Startposition erreicht wurde, wird eine Nachricht abgesendet. Diese Nachricht löst einen Prozess im Kommunikationssystem aus, der zur Folge hat, dass ein Befehl an den Server gesendet wird, um die Klasse Navigation Controller zu starten.

Die Klasse **NavigationController** dient dazu, den *BasicNavigator* aus dem Paket *nav2_ - simple_ commander* in ROS2 zu verwenden. Ihr Ziel ist es, den zu Beginn gestarteten Navigation Stack sauber herunterzufahren.

- *__ init__ ():* Hier wird *„rclpy“* initialisiert, was für ROS 2-Knoten notwendig ist, um mit dem ROS-Graphen zu kommunizieren. Anschließend wird der BasicNavigator instanziiert. Dieser bietet eine hohe Abstraktionsebene über den Navigation2-Stack und vereinfacht die Verwaltung von Navigationsaufgaben. Um sicherzustellen, dass der Navigation2-Stack vollständig aktiv und bereit ist, bevor Befehle gesendet werden, wird „waitUntilNav2Active“ aufgerufen.

- *cancel_ navigation():* Diese Methode ruft cancelTask auf dem BasicNavigator auf. Sie wird verwendet, um die aktuelle Navigationsaufgabe abzubrechen, falls der Roboter navigiert und die Navigation aus irgendeinem Grund gestoppt werden soll.

- *shutdown():* Diese Methode gewährleistet ein ordnungsgemäßes Herunterfahren des Navigation Stack. Zunächst wird destroyNode auf dem BasicNavigator aufgerufen, um den Navigator-Node sauber zu zerstören. Anschließend wird rclpy.shutdown aufgerufen, um die rclpy-Kontexte zu schließen und alle ROS-bezogenen Ressourcen freizugeben.

Da die Klasse NavigationController erst aufgerufen wird, sobald die Startposition erreicht wurde, ist sichergestellt, dass der Navigation Stack während der Navigation zum Ziel nicht heruntergefahren wird.

Im nächsten Kapitel werden die Ergebnisse des entwickelten Verifikationssystems dargestellt.

5 Ergebnisse

Dieses Kapitel liefert Ergebnisse der zuvor entwickelten Komponenten und deren Interaktion. Zunächst wird die Navigation des Roboters betrachtet, gefolgt von den Ergebnissen der Personenerkennung.

5.1 Navigation des Turtlebot3 Waffle Pi

Wie in Kapitel 4 beschrieben, sind einige Bausteine erforderlich, die die Navigation des Roboters ermöglichen. Zunächst benötigt der Roboter eine Karte seiner Umgebung. Anschließend muss dem Roboter mitgeteilt werden, wo er sich zum Start auf der Karte befindet. Danach können Navigationsziele angegeben werden, zu denen er sich dann selbstständig navigieren kann.

Das Erstellen einer Karte erfolgt mit hilfe der „SLAM“ Technik und den Sensoren des Turtlebot3 Waffle Pi.

SLAM (Simultaneous Localization and Mapping) ist eine bedeutende Technik in der Robotik. Sie ermöglicht einem Roboter, sich in einer unbekannten Umgebung zu orientieren und gleichzeitig eine Karte dieser Umgebung zu erstellen. Für den Turtlebot3 Waffle Pi, läuft dieser Prozess wie folgt ab [11]:

Sensordaten sammeln: Der Turtlebot3 nutzt seine Sensoren, insbesondere einen LIDAR-Sensor, um Informationen über seine Umgebung zu sammeln. Diese Informationen umfassen Entfernungsdaten zu umgebenden Objekten, die als 'Scan-Informationen' bezeichnet werden.

Odometrie und TF (Transform): Die Odometrie nutzt Daten von Bewegungssensoren, um im Laufe der Zeit eine Schätzung der Veränderung der Position des Roboters zu erhalten [13]. TF (Transform) in ROS ist ein System, das die räumlichen Beziehungen zwischen verschiedenen Koordinatensystemen verwaltet [15]. Diese Informationen helfen

dem Roboter, seine eigene Bewegung und Position, relativ zur Startposition zu verstehen.

Kartenerstellung (Mapping): Der Turtlebot3 nutzt während seiner Bewegung die gesammelten Scan-Daten, zusammen mit der Odometrie und TF-Informationen, um eine Karte der Umgebung zu erstellen. Die Karte wird jedes Mal aktualisiert, wenn der Roboter neue Bereiche erkundet [11].

Visualisierung in RViz: RViz ist ein Tool in ROS [14], das die Visualisierung von Sensordaten, Roboterzuständen und -operationen ermöglicht. Die vom Turtlebot3 erstellte Karte wird in Echtzeit in RViz dargestellt. So können Benutzer den Fortschritt der Kartenerstellung verfolgen.

Nachdem eine Karte der gewünschten Umgebung aufgezeichnet wurde, kann diese abgespeichert werden, um sie für spätere Aufgaben erneut zu verwenden.

Die Abbildung 5.4 zeigt den Vorgang zur Erstellung einer wiederverwendbaren Karte einer Umgebung in Zwischenschritten.

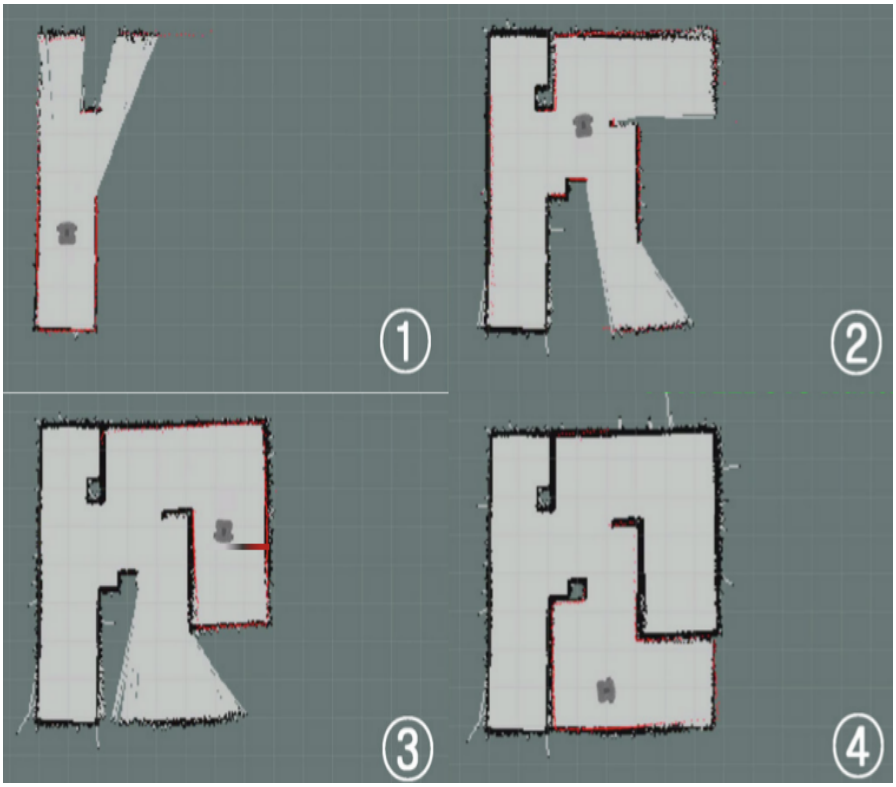
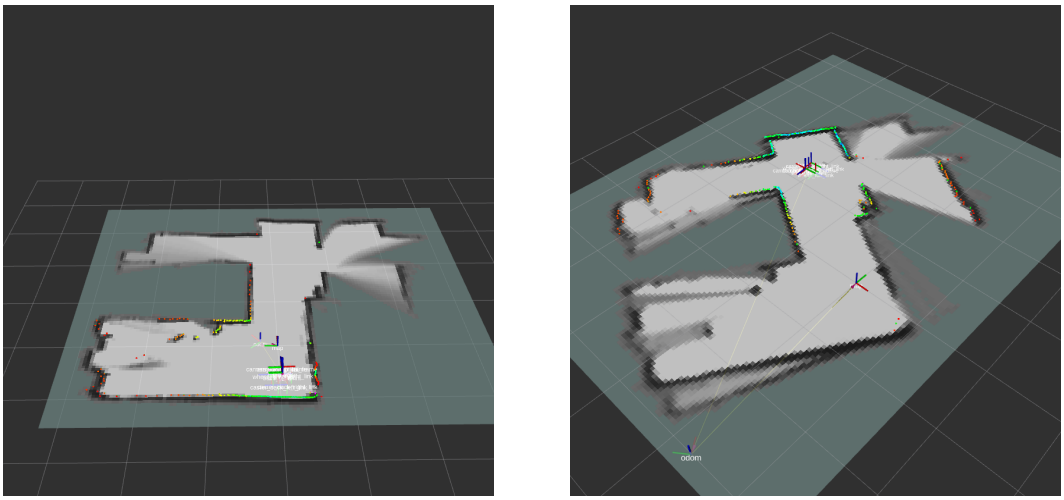


Abbildung 5.1: Erfolgreiche Kartierung mit Zwischenschritten [22]

5.1.1 Probleme bei der Karten erzeugung

Wie oben beschrieben besteht der Prozess zur Erstellung einer Karte für den Roboter aus der Zusammenarbeit mehrerer Techniken. Trotz Bemühungen konnte weder für das LP noch für eine andere Umgebung eine wiederverwendbare Karte mithilfe des Turtlebot3 Waffle Pi erzeugt werden. Die Abbildung 5.2 zeigt den Versuch, eine Karte aufzunehmen. Dabei zeigt Abbildung 5.2a den Startzustand, der noch gut aussieht. Abbildung 5.2b zeigt den Zustand, nachdem der Roboter ein Stück nach vorne gefahren ist. Im Verlauf der Abbildungen ist eine Verschiebung in der Karte zu erkennen, insbesondere unten links in Abbildung 5.2b.

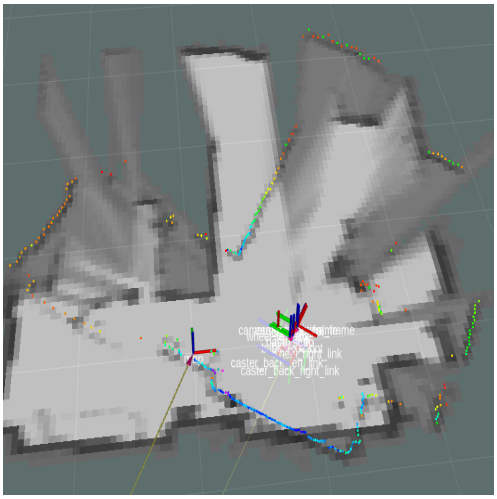


(a) Startposition bei Karten erstellung.

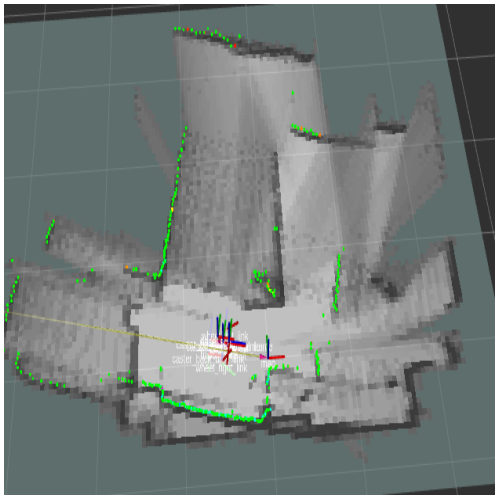
(b) Karte nach vorwärtsbewegung.

Abbildung 5.2: Verschiebung in der Orientierung des Roboters

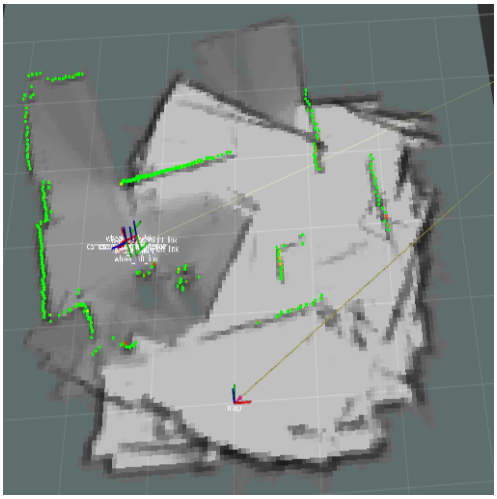
Abbildung 5.2 zeigt eine leichte Verschiebung in der Orientierung des Roboters nach einer Vorwärtsbewegung. Die folgenden Abbildungen aus 5.3 zeigen die Karten, nachdem der Roboter sich nach links oder rechts gedreht hat und den Raum durchfahren hat.



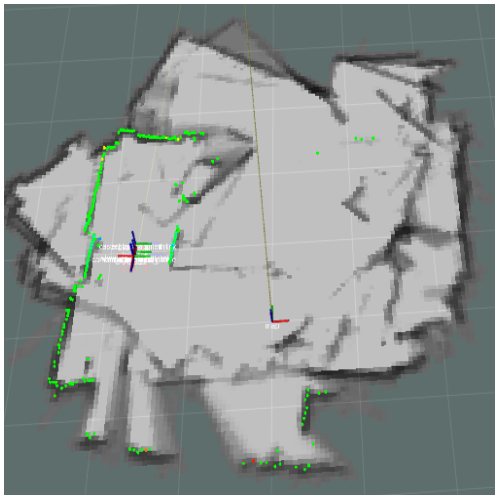
(a) Startposition - Rotation nach rechts.



(b) Bewegung nach links.



(c) Karte nach Bewegung durch den Raum.



(d) Wiederholtes umherfahren im Raum.

Abbildung 5.3: Orientierungsverlust des Roboters auf der Karte

Aus den Abbildungen 5.3 wird deutlich, dass der Roboter Schwierigkeiten hat, seine Umgebung korrekt wahrzunehmen und in eine Karte zusammenzuführen. Wenn der Roboter rotiert, verliert er seine Orientierung und „springt“ auf der Karte herum, um sich neu zu positionieren und beginnt dann, eine neue Karte ab dieser Position zu zeichnen, anstatt die bisherigen Daten zu verwenden, um seine Position relativ zur eigentlichen Karte zu bestimmen.

Es können verschiedene Ursachen bestehen, die den Roboter während des Karten-Erstellungsprozesses mit SLAM seine Orientierung verlieren und ein unzuverlässiges Kartenmaterial produzieren lassen.

Schlechte Sensorqualität oder -konfiguration: Wenn die Sensoren des Roboters, insbesondere der LIDAR oder andere Entfernungsmessgeräte, nicht korrekt kalibriert oder von schlechter Qualität sind, können sie ungenaue Daten liefern, die zu einer fehlerhaften Lokalisierung führen.

Schnelle oder abrupte Bewegungen: Schnelle oder ruckartige Bewegungen können die Position des Roboters beeinträchtigen, insbesondere wenn die Sensordaten nicht schnell genug aktualisiert werden oder die Bewegungen die Grenzen der Sensoren überschreiten.

Softwarefehler: Fehler in der SLAM-Software oder Konfigurationsprobleme können zu einer schlechten Leistung führen. Dies kann von falschen Einstellungen bis hin zu Fehlern im SLAM-Algorithmus selbst reichen.

Fehler in der Odometrie: Ungenauigkeiten in der Odometrie können zu inkonsistenten Schätzungen der Bewegung und Position des Roboters führen. Diese Ungenauigkeiten können durch Schlupf der Räder, unebene Böden oder andere mechanische Probleme verursacht werden.

Durch verschiedene Tests konnten zumindest einige der genannten Punkte ausgeschlossen werden. Der LIDAR-Sensor erkennt Hindernisse zuverlässig und bildet sie auf der Karte ab. Dies wurde unter anderem durch das Aufstellen großer Hindernisse getestet. Es konnte auch ausgeschlossen werden, dass die Kartenerstellung aufgrund zu schneller Bewegungen nicht ordnungsgemäß funktioniert, da diese Fehler auch bei geringster Geschwindigkeit auftreten. Es könnte sich um einen Software- oder Konfigurationsfehler handeln. Eine alternative Methode zur Erstellung einer Karte wurde zwar getestet. Sie führte jedoch zu denselben Ergebnissen. Der Roboter wurde auf die Werkseinstellungen zurückgesetzt und die einzelnen Bauteile wurden überprüft, um sicherzustellen, dass sie gemäß der Dokumentation [20] eingerichtet sind. Es ist wahrscheinlich, dass ein Fehler in der Odometrie zu den Sprüngen auf der Karte führt. Diese Annahme wird durch die Fehlersuche-Seite in der ROS-Dokumentation gestützt [10]. Die drei beschriebenen Tests dienen zur Feststellung, ob ein Roboter eine gute Odometrie hat, um Navigationsaufgaben zu ermöglichen. Leider hat keiner dieser drei Tests mit dem ausgewählten Turtlebot3

Waffle Pi erfolgreich funktioniert, um eine fehlerhafte Odometrie als Fehlerquelle auszuschließen.

Diese Probleme haben nicht nur zur Folge, dass keine Karte mit dem Roboter selbst erzeugt werden konnte, sondern auch, dass eine Navigation mit dem Turtlebot3 Waffle Pi nicht möglich ist. Selbst wenn eine korrekte Karte verwendet würde, hätte die fehlerhafte Odometrie dennoch Auswirkungen auf die Navigation des Turtlebot3 Waffle Pi, da sich seine Position während der Bewegungen so stark ändert, dass eine Orientierung nicht mehr möglich ist.

5.1.2 Navigation in Gazebo

Um die in Kapitel 4 entwickelten Komponenten, die die Navigation des Turtlebot3 Waffle Pi ermöglichen sollen, dennoch zu testen, wurde die Simulationsumgebung Gazebo [4] verwendet.

Für den Turtlebot3 Waffle Pi und andere Modelle der Turtlebot3-Serie stellen die Entwickler mehrere Simulationsumgebungen in Gazebo zur Verfügung.

Zum Testen der Navigationskomponenten wurde die Turtlebot3 World Map verwendet. Im linken Teil der Abbildung ist die Simulationsumgebung und im rechten Teil die generierte Karte zu sehen.

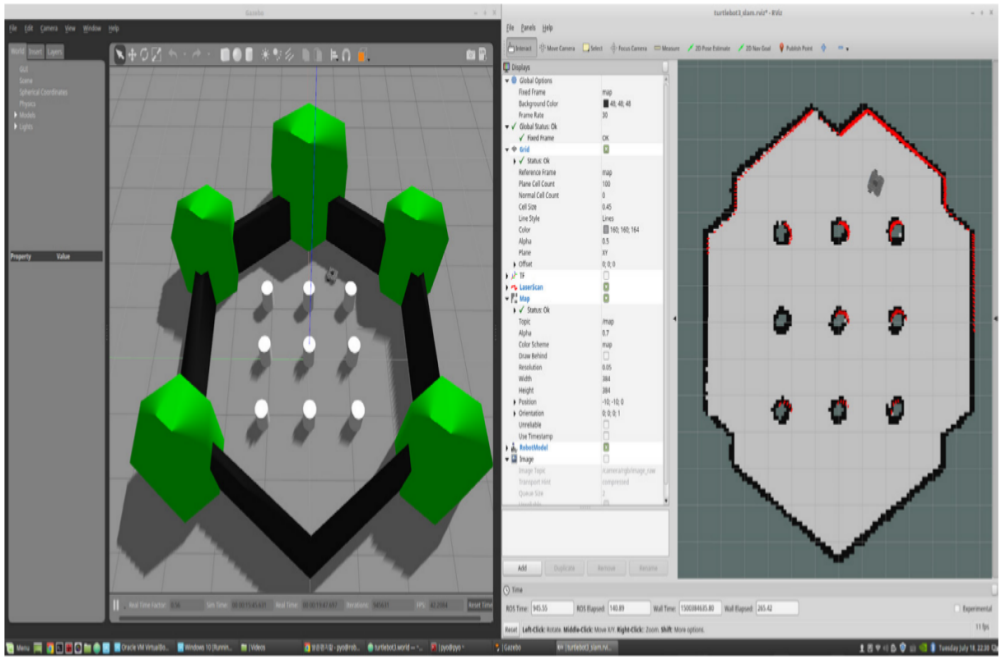


Abbildung 5.4: Turtlebot3 World + Erzeugte Karte mittels SLAM [22]

Die Abbildung 5.5 dient zum Verständnis und zur Visualisierung der entscheidenden Punkte auf der Karte. Der grüne Kreis symbolisiert die Startposition des Roboters, die blaue und gelbe Raute symbolisieren die Sturzpositionen und die Pfeile drum herum symbolisieren alternative Positionen zum Sturzpunkt. Die beiden grauen Linien symbolisieren den Weg, der zurückgelegt werden muss, um die angegebene Sturzposition zu erreichen.

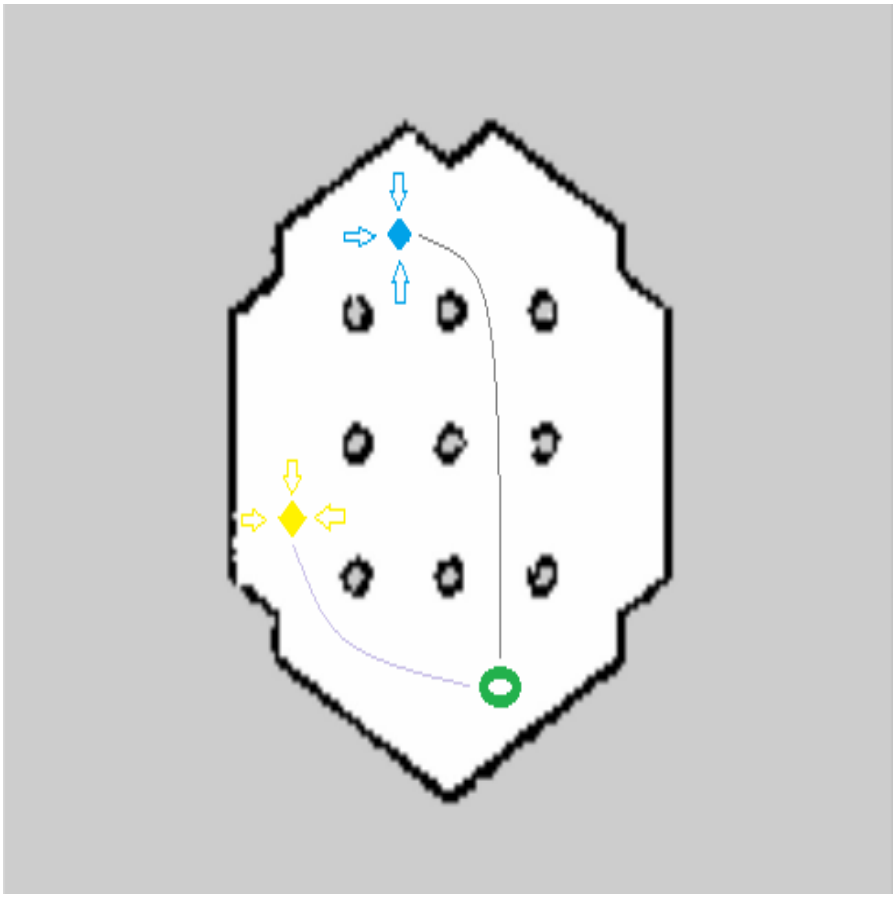


Abbildung 5.5: Karte mit eingezeichneten Zielen und Startposition

InitialPosePublisher

Das Ziel des Skriptes *InitialPosePublisher* ist es, die Startposition des Turtlebot3 Waffle Pi auf der Karte zu bestimmen, damit der Roboter sich in der Umgebung navigieren kann.

Mit dem Rviz Visualisierungs Tool, würde das Setzen der Startposition wie folgt aussehen:

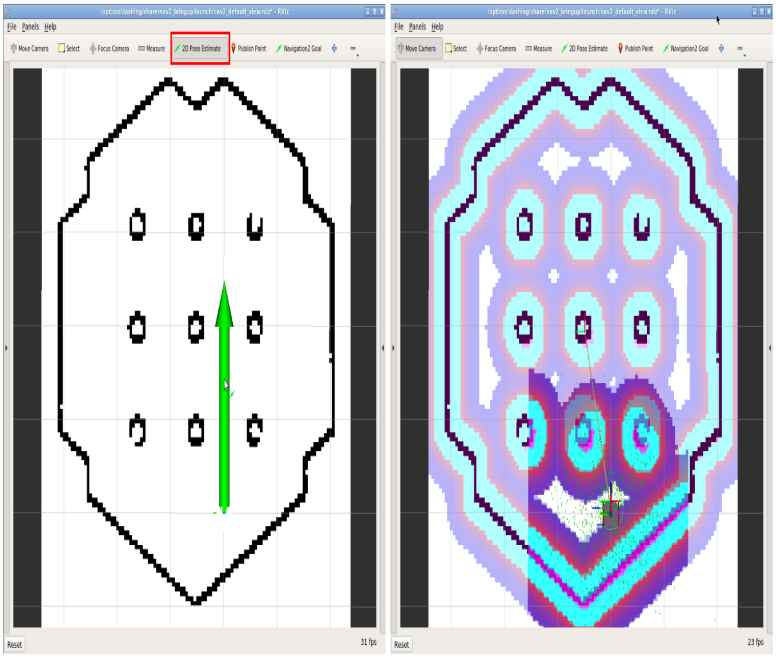


Abbildung 5.6: Startposition festlegen mit Rviz [9]

Das gleiche Verhalten kann mit dem InitialPosePublisher automatisiert werden. Es wird ein Node erzeugt, der auf das Topic */initialpose* publizieren kann. Da der Node */amcl* das Topic */initialpose* aboniert hat und somit auf Nachrichten lauscht, empfängt er die publizierte Nachricht des */initial_pose_publishers* und trägt die Position in die Karte ein. Die Abbildung 5.7 zeigt den rqt-Graphen und die Beziehungen an. Die Abbildung 5.8, zeigt das Veröffentlichen der Startposition und das eintragen dieser in die Karte. Das Publizieren der Startposition erfolgt fünf Mal (5.8a), da es empfohlen wird, die Startposition mehrmals zu setzen, um die Genauigkeit während der Navigation zu erhöhen.

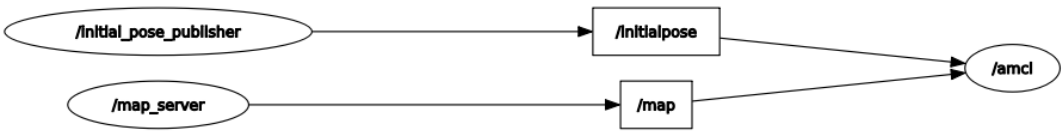
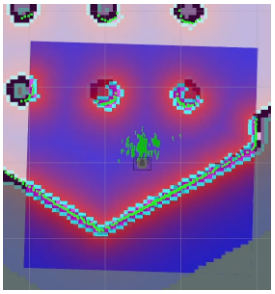


Abbildung 5.7: Rqt Graph - InitialPosePublisher

(a) Startposition veröffentlichen



(b) Startposition erfolgreich angenommen.

Abbildung 5.8: Veröffentlichung und setzen der Startposition

NavigateToPoseClient

Diese Klasse dient dazu, um dem Navigation Stack, Ziele mitzuteilen die angefahren werden sollen. Das Skript wird erst ausgeführt, nachdem die Startposition festgelegt wurde. Es wird ein ActionClient erstellt, der mit dem Action-Server von Nav2 für die Aktion NavigateToPose kommuniziert.

Die Abbildung zeigt einen kleinen Ausschnit aus dem rqt-Graphen, welcher die gesammte Navigation umfasst.

/navigate_to_pose ist das Haupttopic, unter dem die Aktion stattfindet.

/navigate_to_pose_client ist der Node, der durch die NavigateToPoseClient Klasse repräsentiert wird.

/navigate_to_pose/action ist das Topic, das verwendet wird, um die Navigationsziele zu senden.

/navigate_to_pose/action/status ist das Topic, auf dem der Status der Aktion veröffentlicht wird, der anzeigt, ob die Aktion akzeptiert, abgelehnt oder beendet wurde.

/navigate_to_pose/action/feedback ist das Topic, das während der Ausführung der Navigationsaktion Rückmeldungen gibt.

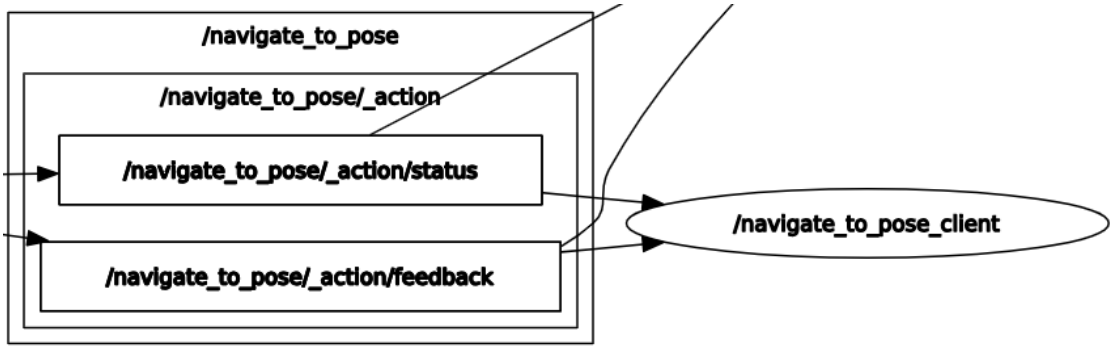


Abbildung 5.9: rqt-Graph - NavigateToPoseClient

Die NavigateToPoseClient-Klasse interagiert mit diesen Topics und dem Action-Server, um die Navigation des Roboters zu koordinieren. Wenn ein Ziel gesendet wird, wartet sie auf die Antwort des Action-Servers, verarbeitet das Ergebnis und aktualisiert gegebenenfalls das nächste Ziel.

Um besser zu verstehen, welche Schritte durch die Implementierung dieser Klasse und der *InitialPosePublisher* Klasse automatisiert wurden, empfiehlt sich das Video der Entwickler von TurtleBot3. Es zeigt, welche Schritte mithilfe von Rviz durchgeführt werden müssen, um die Navigation zu einem Ziel zu starten [17].

NavigateToInitialPose

Diese Klasse hat die gleiche Funktionalität wie die NavigateToPoseClient-Klasse, mit dem Unterschied, dass die Startposition vordefiniert ist und diese als Ziel an den Action Server übergeben wird.

5.2 Personen- und Posenerkennung

5.2.1 ImageSubscriber

Die Klasse ImageSubscriber initialisiert einen Node namens */image_ subscriber* und abonniert das Topic */camera/image_ raw/uncompressed*, welches die Livebilder der im Roboter eingebauten Kamera veröffentlicht. Auf diese Weise erhält der Server die Bilder, die in der Bildverarbeitung verwendet werden. Die Beziehung wird durch die Abbildung 5.10 dargestellt.

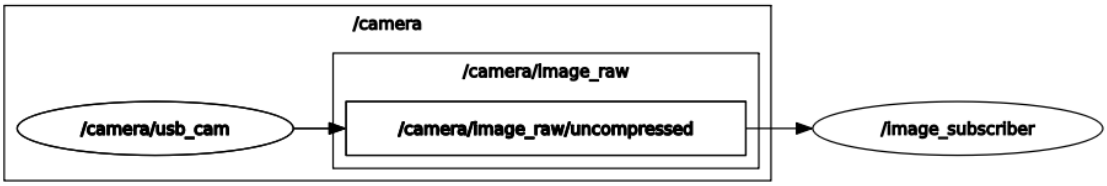


Abbildung 5.10: Rqt Graph - ImageSubscriber

5.2.2 Ergebnisse der Bildverarbeitung

Jedes Mal, wenn ein Bild vom Topic */camera/image_ raw/uncompressed* empfangen wird, ruft der Node *ImageSubscriber* die interne Methode *image_ callback* auf.

In der Methode wird das Bild mithilfe von CvBridge in ein für OpenCV kompatibles Format konvertiert. Anschließend wird das Bild in einen PyTorch-Tensor transformiert, um es durch das YOLO-Modell verarbeiten zu können.

Das YOLO-Modell führt auf jedem empfangenen Bild eine Pose Estimation durch, die die Personenerkennung durch den verwendeten Top-Down-Ansatz beinhaltet.

Sobald das Modell eine Person erkennt und eine Pose schätzen kann, werden die Bilder abgespeichert. Das Programm terminiert jedoch erst, wenn die Erkennung insgesamt 20 Mal mit einer Konfidenz von mindestens 70% erfolgreich war. Dabei muss die Erkennung nicht am Stück erfolgen, da sich durch die Bewegung des Roboters der Kamerawinkel immer verändert.

Das Modell erkennt Personen bereits bei der Anfahrt des Roboters ans Ziel aus einer Entfernung von ca. 3-5 Metern. Allerdings haben die Bilder in der Regel eine Konfidenz von unter 50.

Die Abbildung 5.11 zeigt Bilder und ihre Verarbeitung durch das YOLOv8 Pose-Estimationsmodell während der Anfahrt zum Zielort. Die Genauigkeit wird nicht nur durch die Distanz und Lichtverhältnisse beeinträchtigt, sondern auch durch die Bewegungen des Roboters während der Fahrt.

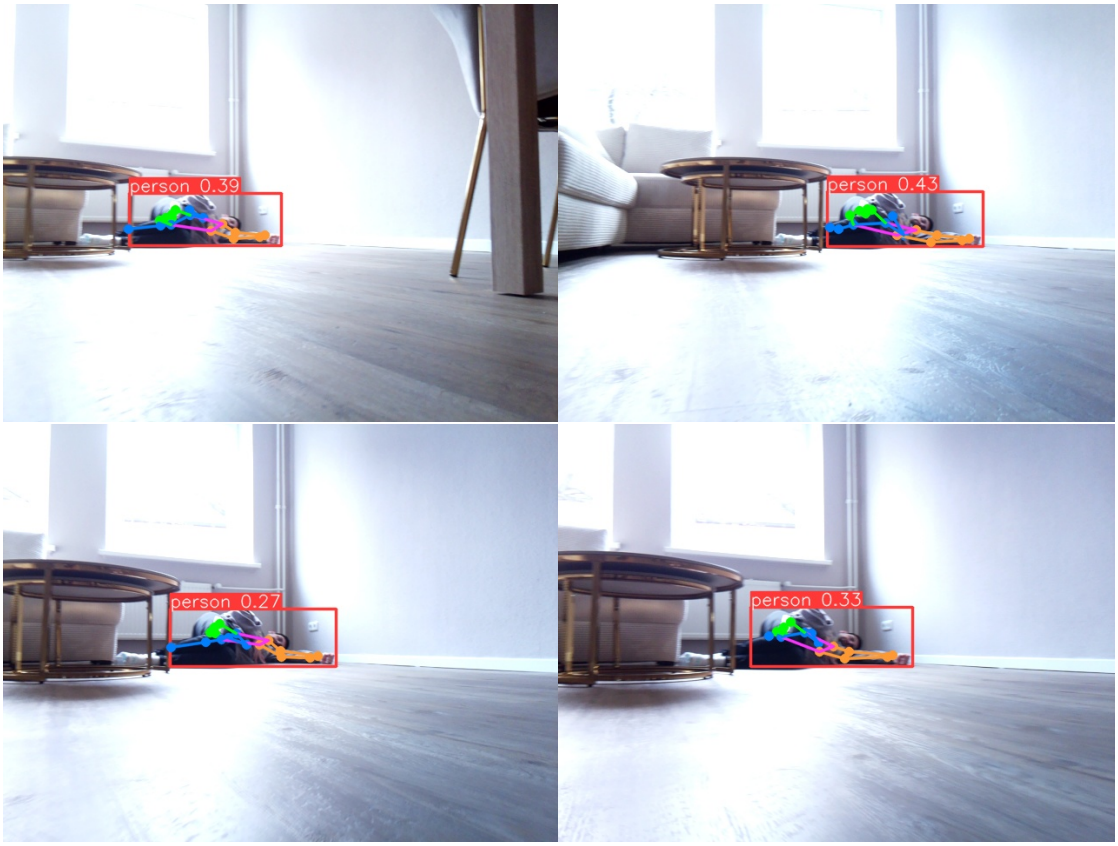


Abbildung 5.11: Schätzung während der Anfahrt zum Ziel

Dieses Verhalten zieht sich durch alle Versuche (Vgl. Abb. 5.12). Wenn das Bild stabil ist, die Lichtverhältnisse gut sind und der ganze Körper gut zu erkennen ist, bestehen einige Ausreißer nach oben (Vgl. Abb. 5.13).

Abbildung 5.12: Ergänzend - Schätzung während der Anfahrt zum Ziel

Abbildung 5.13: Ausreißer nach oben, während der Anfahrt

In den Versuchen hat sich ergeben, dass eine Distanz zur gestürzten Person idealerweise 1-2 Meter beträgt. Am besten bewertet das Modell Bilder, in denen der ganze Körper gut zu erkennen ist. Jedoch ist es auch noch zuverlässig, wenn Teile des Körpers auf dem Bild fehlen. Die Abbildung 5.14 zeigt in der ersten Reihe noch zwei Fotos, in denen der Turtlebot3 in bewegung ist. Die übrigen Bilder zeigen die Verarbeitung und ihre Konfidenz-Werte. Durch den in der Navigation zum Ziel implementierten Timer hat die Bildverarbeitung ausreichend Zeit, um Bilder zu verarbeiten und zu bewerten.

Abbildung 5.14: Zielposition erreicht - Bilder werden verarbeitet

Die Abbildung 5.15 zeigt die Erkennung von Personen und Posen in verschiedenen Körperhaltungen und Ausrichtungen zur Kamera.

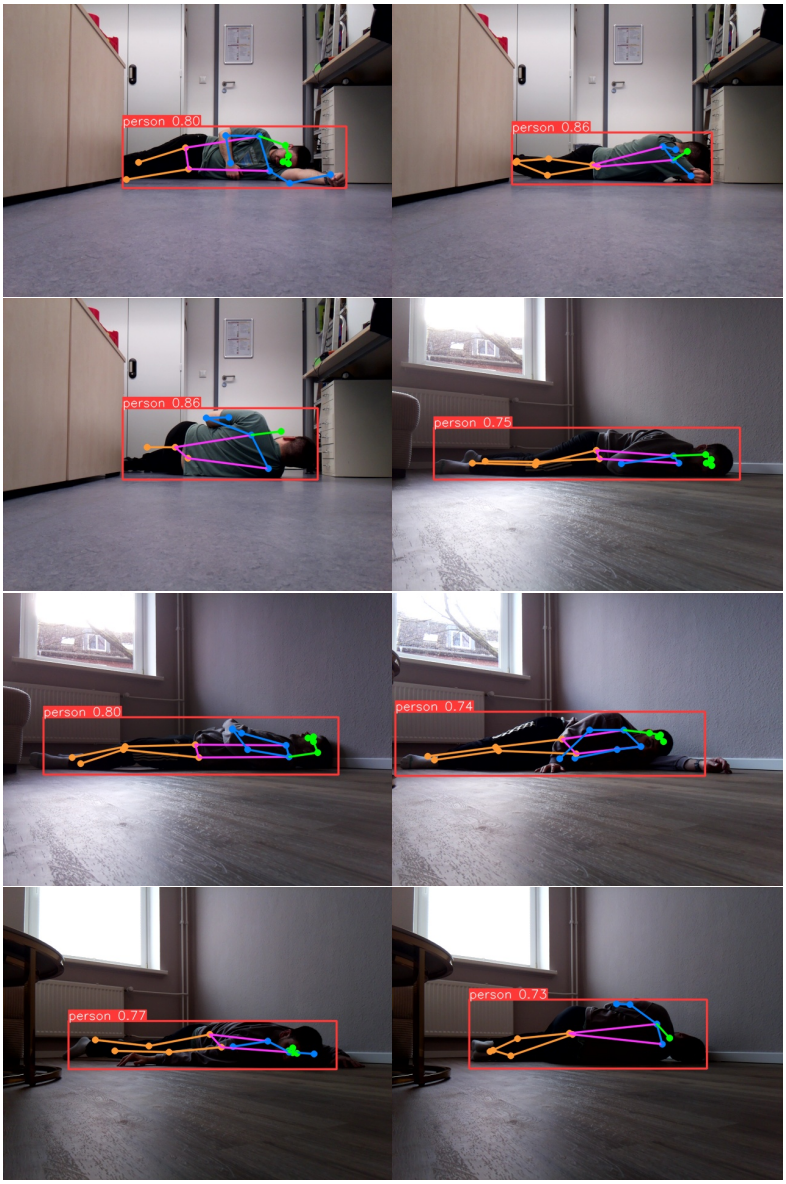


Abbildung 5.15: Umgang mit unterschiedlichen Körperhaltungen

Die Personenerkennung funktioniert auch unter schwierigen Bedingungen zuverlässig, wie die unterschiedlichen Abbildungen zeigen. Allerdings hat die Pose Estimation Schwierigkeiten, die Körper-Keypoints richtig zu interpretieren und liefert teilweise falsche Informationen. Dennoch kann mit diesem Modell sicher bestimmt werden, ob eine Person auf dem Boden liegt oder nicht.

6 Diskussion

Im diesem Kapitel wird ein Blick auf die Zukunft und die weiterführenden Potenziale des in dieser Arbeit entwickelten Systems geworfen. Dabei wird der Nutzen kurz reflektiert und das Entwicklungspotenzial zur Verbesserung des Gesamtsystems aufgezeigt. Darüber hinaus wird der gesellschaftliche Mehrwert eines solchen Systems – von der Stärkung der individuellen Unabhängigkeit bis hin zur Entlastung von Pflegeinstitutionen – diskutiert. In einer Welt, in der demografische Veränderungen und gesundheitliche Herausforderungen zunehmend in den Vordergrund rücken, kann die Technologie einen Beitrag zum Wohl zukünftiger Pflegebedürftiger leisten. Außerdem werden die Ergebnisse und ihre Aussagekraft in Bezug auf die verwendete Bildverarbeitungsmethode diskutiert.

6.1 Überblick

Das in dieser Arbeit entwickelte Verifikationssystem knüpft an das bereits entworfene Sturzerkennungssystem im Living Place an. Wenn ein Sturz registriert wurde, wird eine Nachricht mit der Position des Sturzes an einen Server gesendet. Dieser Server verarbeitet die Nachricht und sendet einen Roboter zur angegebenen Position. Der Roboter sendet während seiner Fahrt kontinuierlich Bilder an den Server, der eine Schätzung darüber ausführt, ob es sich bei dem registrierten Sturz tatsächlich um eine Person handelt. Sobald eine Person erkannt und somit der Sturz verifiziert wurde, stoppt der Roboter. Der Server sendet dann eine Nachricht an ausgewählte Helfer und veranlasst das Öffnen der Türschlösser in der Wohnung.

6.2 Diskussion der Ergebnisse

In diesem Abschnitt werden die Ergebnisse der Personenerkennung aus Kapitel 5.2.2 diskutiert. Es wird erläutert, welche Grundlage die Ergebnisse liefern, um eine Aussage

darüber zu treffen, ob es sich bei dem gemeldeten Sturz tatsächlich um eine Person handelt oder nicht.

Das verwendete YOLOv8-Modell liefert zuverlässige Ergebnisse, die zur Identifizierung von Personen genutzt werden können. Es gibt jedoch Schwierigkeiten, Personen zu erkennen, wenn die Sicht des Roboters durch Hindernisse 6.1 blockiert ist. Außerdem liefert das Modell eine geschätzte Pose der Person, wobei es jedoch häufig zu einer ungenauen Schätzung der Körper-Keypoints kommt. Dies liegt vor allem daran, dass das YOLOv8-Modell auf dem COCO-Keypoints-Datensatz 2017[31] trainiert und getestet wurde, der nicht für das Einzeichnen von Keypoints von auf dem Boden liegenden Personen ausgelegt ist.

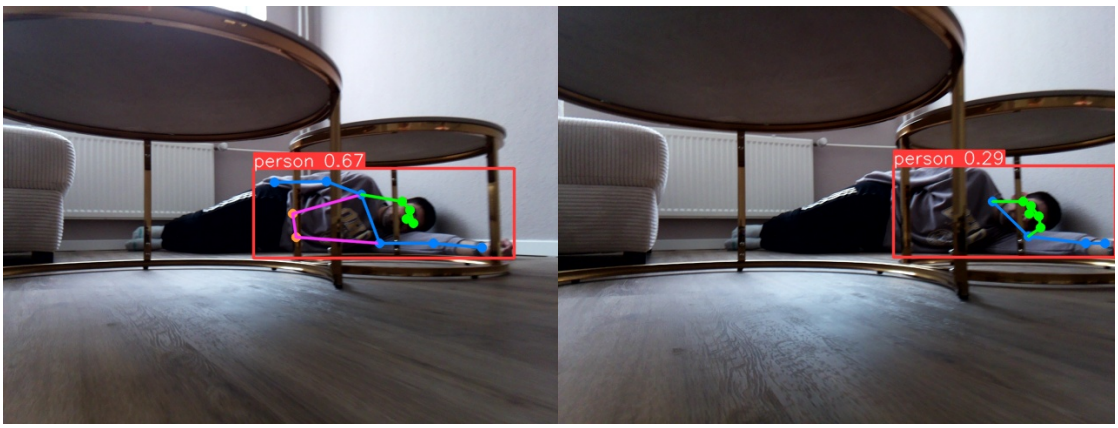


Abbildung 6.1: Sicht durch Hinternisse eingeschränkt

6.2.1 Pose Estimation

Wie bereits in Kapitel 2.5.1 beschrieben, gibt es unterschiedliche Ansätze zur Pose Estimation: den Top-Down- und den Bottom-Up-Ansatz.

Für diese Arbeit wurde die Entscheidung getroffen, ein Modell zu wählen, das den Top-Down-Ansatz verwendet, was sich als richtig erwiesen hat. Die Funktionsweise sieht vor, dass zunächst eine Objekterkennung durchgeführt wird, um Personen zu identifizieren. Erst im zweiten Schritt werden die Körper-Keypoints eingezeichnet. Dies liefert das für diese Arbeit entscheidende Ergebnis, ob es sich um eine Person handelt oder nicht. Für die Arbeit und die Analyse der Körper-Keypoints in weiterführenden Arbeiten ist die Schätzung der Körperhaltung eine gute Grundlage.

Das ausgewählte Modell bietet die Möglichkeit, es mit eigenen Daten zu trainieren. Dadurch kann ein Trainingsdatensatz erstellt werden, der die Körper-Keypoints von am Boden liegenden Personen besser erfasst. Das Modell kann daraufhin trainiert werden, um genauere Ergebnisse zur Pose der Person zu liefern.

Die folgenden Abbildungen zeigen den Unterschied zwischen der Pose Estimation von YOLOv8 (Top Down) und von MediaPipePose (Bottom Up).

Zu nächst ein einfaches Bild im Stand:

Abbildung 6.2: MediaPipePose und YOLOv8 im Vergleich

Beide Modelle erkennen die Pose und die einzelnen Keypoints sehr gut, wenn die Person steht.

Um unterschiedliche Szenarien und Ergebnisse zu vergleichen, wurden die Ergebnisse der beiden Modelle übereinander gezeichnet, als die Person auf dem Boden lag. YOLOv8 verwendet farbige Verbindungen, um die Pose zu schätzen, während MediaPipePose weiße

Verbindungen verwendet. Beide Modelle liefern schlechte Einschätzungen der Keypoints, aber YOLOv8 identifiziert das Objekt als eine Person.

Abbildung 6.3: Schlechte Posenschätzung unter schwierigen Bedingungen

MediaPipePose hat Schwierigkeiten, eine Person und ihre Pose zu bestimmen, wenn diese nicht frontal zur Kamera ausgerichtet ist und/oder schwierige Lichtbedingungen herrschen. In Abbildung 6.4 sind vier Fälle dargestellt, in denen YOLOv8 mit einer Konfidenz von mindestens 70% eine Person erkennen konnte, während MediaPipePose keine Posenschätzung vornehmen konnte.

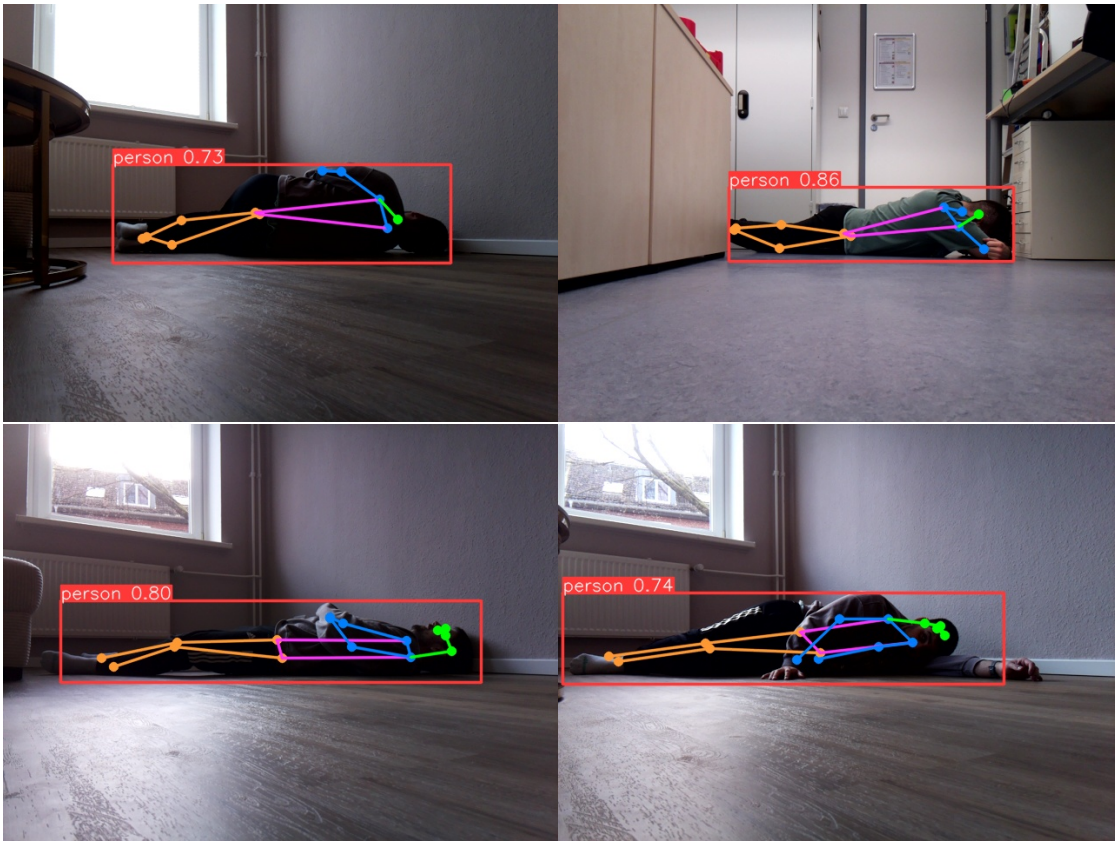


Abbildung 6.4: Keine Ergebnisse aus MediaPipePose, aber aus YOLOv8

Unter den richtigen Bedingungen liefert MediaPipePose gute Ergebnisse. Für den Anwendungsfall, wie er in dieser Arbeit verfolgt wird, ist es jedoch keine gute Wahl. Sehr oft werden keine Posen geschätzt. Außerdem bedarf einer weiteren Verarbeitung, in des die Keypoints interpretiert werden, um sicherzustellen, dass die erkannten Keypoints tatsächlich einer Person zugeordnet wurden und nicht einem anderen Objekt. Wie die Abbildung 6.3 zeigt, sind die Keypoints nicht gut eingezeichnet. Es wäre kühn, anhand solcher Daten eine zuverlässige Aussage darüber zu treffen, ob es sich um eine Person handelt oder nicht. Leider gibt es auch keine Möglichkeit, das Modell von MediaPipePose auf einem eigenen Trainingsdatensatz zu trainieren, um bessere Ergebnisse zu erzielen, im Gegensatz zu YOLOv8.

6.3 Empfehlung für weiterführende Arbeiten

Das entwickelte System bietet in seinen einzelnen Komponenten mehrere Möglichkeiten zur Weiterentwicklung an. Folgend werden einige erörtert.

Navigation des Roboters

In den Kapiteln zur Implementierung und zu den Ergebnissen wird das Funktionsmodell des Roboters vorgestellt, das auf festen Positionen basiert. Der Roboter beginnt seine Suche an einem definierten Startpunkt und navigiert zu einem vorbestimmten Ziel, um nach gestürzten Personen zu suchen. Die Möglichkeit, den Roboter auch zu alternativen Positionen zu steuern, um den Blickwinkel zu erweitern und damit die Effizienz der Suche zu erhöhen, wurde bereits erwähnt. Diese Methodik erfüllt die aktuellen Anforderungen der Arbeit. Für eine reale Einsatzumgebung ist jedoch eine Weiterentwicklung und Verfeinerung erforderlich.

Aus dieser Ausgangslage ergeben sich zwei mögliche Entwicklungsszenarien, die einzeln oder in Kombination umgesetzt werden könnten.

Eine Weiterentwicklung der Sturzerkennung könnte darin bestehen, dass das System nicht nur das Ereignis eines Sturzes identifiziert, sondern auch in der Lage ist, die Wohnung in vordefinierte Bereiche einzuteilen. Im Falle eines Sturzes würde dann nicht nur eine Nachricht abgesendet werden, in der steht, dass ein Sturz registriert wurde, sondern auch der spezifische Bereich, in dem der Sturz stattgefunden hat. Langfristig könnte man auch die genaue Position des Sturzes innerhalb der Wohnung ermitteln und kommunizieren.

Ein zweiter Ansatz wäre die Implementierung von Machine-Learning-Methoden, die es dem Roboter ermöglichen, sich autonom in der Wohnung zu orientieren. Bei erfolgreicher Umsetzung würde der Roboter eigenständig nach der gestürzten Person suchen, ohne auf vorher festgelegte Start- und Zielkoordinaten angewiesen zu sein. Trotz der Autonomie wäre eine Bereichsangabe sinnvoll, um die Effizienz der Suche zu maximieren und unnötige Verzögerungen zu vermeiden.

Letzteres kann mithilfe der Ros Kinetic Version auf dem Turtlebot3 Waffle Pi umgesetzt werden [18]. In der Abbildung 6.5 werden die möglichen Funktionen des Turtlebot3 Waffle Pi und ihrer Kompatibilität mit unterschiedlichen Ros Versionen dargestellt. Funktionen, die zur Verfügung stehen, sind mit einem Kreis markiert. Die nicht zur Verfügung stehenden Funktionen sind mit einem Kreuz versehen.

Features	Kinetic	Melodic	Noetic	Dashing	Foxy	Galactic	Humble
Teleop	O	O	O	O	O	O	O
SLAM	O	O	O	O	O	O	O
Navigation	O	O	O	O	O	O	O
Simulation	O	O	O	O	O	O	O
Manipulation	O	O	O	O	O	△	O
Home Service Challenge	O	O	O	X	X	X	X
Autonomous Driving	O	X	O	X	X	X	X
Machine Learning	O	O	X	O	X	X	X

Examples	Kinetic	Melodic	Noetic	Dashing	Foxy	Galactic	Humble
Interactive Markers	O	X	X	X	X	X	X
Obstacle Detection	O	X	X	O	X	X	X
Position Control	O	X	X	O	X	X	X
Point Operation	O	X	X	O	X	X	X
Patrol	O	X	X	O	X	X	X
Follower	O	X	X	X	X	X	X
Panorama	O	X	X	X	X	X	X
Auto Parking	O	X	X	O	X	X	X
Auto Parking(Vision)	O	X	X	X	X	X	X
Multi TurtleBot3	O	X	X	X	X	X	X

Abbildung 6.5: Mögliche Funktionen unter den verschiedenen Ros Versionen [18]

Interaktion mit dem Roboter

Das aktuelle Verifikationssystem geht davon aus, dass eine gestürzte Person nicht in der Lage ist, sich eigenständig aufzurichten oder um Hilfe zu rufen. Eine zukunftsorientierte Weiterentwicklung könnte eine interaktive Kommunikationsschnittstelle beinhalten, die eine direkte Interaktion mit dem Roboter ermöglicht. Durch den Einsatz fortschrittlicher Methoden der natürlichen Sprachverarbeitung (NLP) und Text-to-Speech-Technologien könnte der Roboter bei Eintreffen am Einsatzort verbal erfragen, ob der Betroffene in Ordnung ist. Die Antwort würde dann analysiert werden, um zu entscheiden, ob weitere Hilfe erforderlich ist.

Zudem ließe sich das System durch den Einbau von Lautsprechern und Mikrofonen so erweitern, dass eine telefonische Verbindung zu einer Hilfsperson hergestellt werden kann. Über diese Verbindung könnte die Situation bewertet und die nächsten Schritte koordiniert werden. Diese Weiterentwicklung würde nicht nur die Effizienz des Systems erhöhen, sondern auch den Betroffenen eine zusätzliche Ebene der Sicherheit und des Komforts bieten.

Identifikation der Bewohner

Das aktuelle System kann eine Person erkennen, ohne jedoch ihre Identität zu bestimmen. Eine Weiterentwicklung des Personenerkennungsmodels zu einem Identifizierungssystem, das die Bewohner eines Haushalts spezifisch erkennen kann, würde einen signifikanten

Fortschritt darstellen. Ein solches Feature wäre insbesondere in Pflegeeinrichtungen oder Haushalten mit mehreren sturzgefährdeten Personen von großem Wert.

Die Integration eines präzisen Identifikationsmoduls in das Notfallsystem könnte zur Effizienz der Ersthelfer maßgeblich beitragen. Bei Auslösung eines Notfallsignals könnten neben der Alarmierung auch wichtige personenbezogene Informationen übermittelt werden, wie beispielsweise Geschlecht, Körpergröße, Alter, Gewicht und medizinische Besonderheiten, bisherige Krankheitsverläufe oder gar Sensordaten von Gesundheitstrackern. Solche Daten könnten für das Rettungspersonal wichtig sein und dazu beitragen, eine zielgerichtete und effektive Erstversorgung zu gewährleisten. Die Anpassung und der Austausch sensibler Daten erfordert eine sorgfältige Abwägung. Insbesondere bei der Benachrichtigung im Falle eines Sturzes, aus dem sich die betroffene Person nicht selbst befreien kann, ist es wichtig, die Frage zu klären, an wen die Notfallnachricht gesendet wird. Wenn es sich um einen Nachbarn handelt, ist es wahrscheinlich, dass die betroffene Person nicht möchte, dass dieser Einblick sensible personenbezogene Daten erhält. Allerdings könnte es für den Nachbarn relevant sein zu wissen, ob die Person an Osteoporose leidet, da dies impliziert, dass beim Aufhelfen besondere Vorsicht geboten ist.

Diese Anforderung unterstreicht die Notwendigkeit einer differenzierten Handhabung personenbezogener Daten. Je nach Kontext der Hilfeleistung müssen Informationen so angepasst und geteilt werden, dass einerseits die Privatsphäre der betroffenen Person gewahrt wird und andererseits den Helfern genügend Kontext geboten wird, um adäquat reagieren zu können. Der Umgang mit diesen sensiblen Informationen erfordert daher ein hohes Maß an Vorsicht und eine klare Regelung, welche Daten in welcher Situation mit welchen Helfern geteilt werden dürfen. Es muss gewährleistet sein, dass die Übermittlung dieser Daten unter strenger Einhaltung der gesetzlichen Datenschutzvorschriften erfolgt, um die Privatsphäre und Sicherheit der betroffenen Personen zu schützen.

6.4 Gesellschaftlicher Nutzen

Angesichts des demografischen Wandels und der daraus resultierenden Herausforderungen, die in der Einleitung thematisiert wurden, sowie der Problematiken, Konsequenzen und der Notwendigkeit eines raschen Eingreifens bei Sturzereignissen, die unter 3.1, 3.2 und 3.3 erläutert wurden, präsentiert das in dieser Arbeit entwickelte System einen möglichen Ansatz, um diesen Herausforderungen zu begegnen. Dieser Ansatz kann als

unterstützende Maßnahme dazu beitragen, die Versorgung der alternden und zunehmend pflegebedürftigen Bevölkerung in Zukunft zu gewährleisten. Das vorgeschlagene Verifikationssystem adressiert nicht nur die unmittelbaren Risiken von Sturzverletzungen, sondern bietet auch eine proaktive Strategie zur Verbesserung der Sicherheit und Unabhängigkeit im häuslichen Umfeld. Durch die Verknüpfung von technologischer Innovation mit den realen Bedürfnissen einer Gesellschaft, die sich den Herausforderungen einer älter werdenden Bevölkerung stellen muss, kann das System dazu beitragen, die Lebensqualität der Betroffenen zu erhöhen und gleichzeitig das Gesundheitssystem zu entlasten.

Obwohl das vorgestellte System keine direkte medizinische Versorgung anbieten kann, stellt es dennoch eine bedeutende Erleichterung für Pflegebedürftige dar, insbesondere für diejenigen, die zu Hause leben. Die Angst vor Stürzen kann die tägliche Lebensführung stark beeinträchtigen. Die Gewissheit, dass im Falle eines Sturzes ein zuverlässiges System vorhanden ist, welches selbstständig Hilfe rufen kann, bietet den Betroffenen ein erhebliches Maß an Sicherheit und trägt dazu bei, ihre Unabhängigkeit zu bewahren.

In Pflegeeinrichtungen könnte dieses System zu einer wesentlichen Ressource werden, um das Pflegepersonal zu unterstützen und zu entlasten. Angesichts der demografischen Entwicklung, die zu einer wachsenden Anzahl von Pflegebedürftigen bei gleichzeitigem Mangel an Pflegekräften führt [39], könnten technologische Hilfsmittel den Druck von den Schultern des Personals nehmen. Pflegekräfte stehen oft vor der Herausforderung, das Sturzrisiko zu minimieren und gleichzeitig eine hochwertige Betreuung aller Patienten sicherzustellen. Ein intelligentes Assistenzsystem kann eine Schlüsselrolle spielen, indem es die Überwachung der am stärksten gefährdeten Personen unterstützt und so mehr Zeit für die Pflege aller Patienten ermöglicht. Sollte ein Sturz eintreten, der aus unterschiedlichsten Gründen nicht sofort bemerkt wird, können Schuldgefühle und Selbstvorwürfe des Pflegepersonals entstehen. Ein automatisiertes Hilfesystem kann hier präventiv wirken, indem es die sofortige Erkennung von Stürzen und die Einleitung von Maßnahmen ermöglicht, was zur psychischen Entlastung des Personals beiträgt [43].

Dadurch würde das Pflegepersonal entlastet werden und die Sicherheit der Pflegebedürftigen erhöht. Ein solches System hätte das Potenzial, den Pflegestandard insgesamt zu verbessern und somit einen Beitrag zur Qualität und Effizienz der Pflegeleistungen zu leisten.

Die Einführung eines solchen Verifikationssystems für Sturzereignisse ist jedoch mit mehreren Herausforderungen verbunden. Besonders ältere oder pflegebedürftige Menschen

könnten sich von der Technologie überwacht fühlen, was ihre Nutzungsbereitschaft einschränkt. Auch Datenschutzbedenken sind signifikant, da der Umgang mit sensiblen Daten wie Gesundheitsinformationen, strengen Vorschriften unterliegt. Zusätzlich belastend für Einrichtungen und Endnutzer sind die Kosten für Entwicklung, Implementierung und Wartung. Technologische Barrieren wie die Notwendigkeit kompatibler Infrastruktur sowie der Schulungsbedarf des Personals und der Nutzer stellen weitere Hindernisse dar. Ethische Bedenken bezüglich der automatisierten Entscheidungsfindung müssen ebenfalls berücksichtigt werden. Um diese Herausforderungen zu bewältigen, ist es wichtig, Bedenken ernst zu nehmen und Maßnahmen wie öffentliche Aufklärung, Einbindung von Datenschutzexperten und die Sicherstellung einer benutzerfreundlichen Handhabung zu ergreifen.

6.5 Fazit

In dieser Arbeit wurde ein Verifikationssystem für Sturzereignisse im Rahmen eines Smart Homes entworfen. Das System erweitert ein Sturzerkennungssystem, indem es nach der Erkennung eines Sturzes automatisch ein Notfallszenario einleitet. Dabei wird ein Roboter aktiviert, der zur gestürzten Person navigiert, Bilder aufnimmt und diese zur Verifikation an einen Server sendet. Die erfolgreiche Identifikation einer gestürzten Person führt zur Auslösung einer Notfallmeldung. Um dieses Szenario zu realisieren, mussten mehrere Module entwickelt werden, die untereinander kommunizieren, um das Szenario ohne manuelle Eingriffe zu automatisieren. Die Herausforderungen bestanden vor allem darin, die Navigation des Roboters zu automatisieren und eine Methode zu entwickeln, die zuverlässig validieren kann, ob es sich bei der gestürzten Person tatsächlich um einen Menschen handelt oder nicht.

Das entworfene System bietet viele Möglichkeiten zur Weiterentwicklung in den Teilbereichen, wie in unter 6.3 beschrieben wurde. Es kann in den einzelnen Modulen so weiterentwickelt werden, dass es mehr Szenarien der realen Welt berücksichtigt und die einzelnen Bereiche für eine bessere Gesamtsystemleistung verbessert werden.

Literaturverzeichnis

[1] *Auswirkungen des demografischen Wandels in Europa - Die Macht der Demografie.* https://commission.europa.eu/strategy-and-policy/priorities-2019-2024/new-push-european-democracy/impact-demographic-change-europe_de#:~:text=Bev%C3%B6lkerungsalterung%20in%20Europa,-Die%20Europ%C3%A4er%2Finnen&text=Bei%20den%2015%2D%20bis%2029,in%20l%C3%A4ndlichen%20Regionen%20noch%20verst%C3%A4rken.&text=Steigender%20Seniorenanteil-,2050%20werden%20rund%2030%20%25%20der%20Bev%C3%B6lkerung%20%C3%BCber%2065%20Jahre,sein%20(heute%3A%2020%20%25).. – Accessed: 2023-11-28

[2] *Demografischer Wandel in Deutschland - Mehr Pflegebedürftige.* https://www.destatis.de/DE/Themen/Querschnitt/Demografischer-Wandel/_inhalt.html#. – Accessed: 2023-11-28

[3] *DRK - Hausnotruf.* https://www.drk.de/hilfe-in-deutschland/senioren/altersgerechtes-wohnen/hausnotruf/. – Accessed: 2023-11-28

[4] *Gazebo.* https://gazebosim.org/homev. – Accessed: 2024-03-09

[5] *HAW Hamburg - Living Place.* https://livingplace.haw-hamburg.de/. – Accessed: 2023-11-28

[6] *MQTT.* https://mqtt.org/. – Accessed: 2024-03-05

[7] *Nav2.* https://navigation.ros.org/concepts/index.html#behavior-trees. – Accessed: 2024-03-05

[8] *Node Red Information.* https://nodered.org/. – Accessed: 2024-03-05

[9] *Robotis - Navigation Simulation Initial Pose.* https://emanual.robotis.com/docs/en/platform/turtlebot3/nav_simulation/. – Accessed: 2024-03-09

[10] *Ros Navigation Troubleshooting.* http://wiki.ros.org/navigation/Troubleshooting. – Accessed: 2024-03-09

[11] *Ros2 Cartograher.* http://wiki.ros.org/cartographer. – Accessed: 2024-03-05

[12] *Ros2 Foxy Dokumentation.* https://docs.ros.org/en/foxy/index.html. – Accessed: 2024-03-05

[13] *Ros2 Odom.* http://wiki.ros.org/navigation/Tutorials/RobotSetup/Odom. – Accessed: 2024-03-05

[14] *Ros2 Rviz.* http://wiki.ros.org/rviz. – Accessed: 2024-03-05

[15] *Ros2 TF2.* http://wiki.ros.org/tf2. – Accessed: 2024-03-05

[16] *Sturzerkennung mittels Pose Estimation im Kontext von Smart Home Umgebungen.* https://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/bachelor/schmidpeter.pdf. – Accessed: 2024-03-11

[17] *TurtleBot3 59 ROS2 Dashing Diademata Navigation2.* https://www.youtube.com/watch?v=VtyqUuuZAFA. – Accessed: 2024-03-09

[18] *Turtlebot3 Feature Übersicht.* https://emanual.robotis.com/docs/en/platform/turtlebot3/quick-start/. – Accessed: 2024-02-28

[19] *Turtlebot3 Waffle Pi.* https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/#notices. – Accessed: 2024-03-05

[20] *Turtlebot3 Waffle Pi Hardware Assembly.* https://emanual.robotis.com/docs/en/platform/turtlebot3/hardware_setup/#hardware-assembly. – Accessed: 2024-03-09

[21] *Turtlebot3 Waffle Pi Komponenten Übersicht.* https://emanual.robotis.com/docs/en/platform/turtlebot3/features/#specifications. – Accessed: 2024-03-05

[22] *Turtlebot3 Waffle Pi Mapping.* https://emanual.robotis.com/docs/en/platform/turtlebot3/slam/#run-slam-node. – Accessed: 2024-03-07

[23] *YOLOv8 Keypoints.* https://learnopencv.com/wp-content/uploads/2021/05/fix-overlay-issue.jpg. – Accessed: 2024-03-10

[24] Balakrishnan, Sumathi ; Vasudavan, Hemalata ; Murugesan, Raja K.: Smart Home Technologies: A Preliminary Review. In: *Proceedings of the 6th International Conference on Information Technology: IoT and Smart City*. New York, NY, USA : Association for Computing Machinery, 2018 (ICIT '18), S. 120–127. – URL https://doi.org/10.1145/3301551.3301575. – ISBN 9781450366298

[25] Bochkovskiy, Alexey ; Wang, Chien-Yao ; Liao, Hong-Yuan M.: *YOLOv4: Optimal Speed and Accuracy of Object Detection*. 2020

[26] Braun, Andreas ; Kirchbuchner, Florian ; Wichert, Reiner: *Ambient Assisted Living*. S. 203–222. In: Fischer, Florian (Hrsg.) ; Krämer, Alexander (Hrsg.): *eHealth in Deutschland: Anforderungen und Potenziale innovativer Versorgungsstrukturen*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2016. – URL https://doi.org/10.1007/978-3-662-49504-9_10. – ISBN 978-3-662-49504-9

[27] Do, Ha M. ; Pham, Minh ; Sheng, Weihua ; Yang, Dan ; Liu, Meiqin: RiSH: A robot-integrated smart home for elderly care. In: *Robotics and Autonomous Systems* 101 (2018), S. 74–92. – URL https://www.sciencedirect.com/science/article/pii/S0921889017300477. – ISSN 0921-8890

[28] Holte, Michael B. ; Tran, Cuong ; Trivedi, Mohan M. ; Moeslund, Thomas B.: Human Pose Estimation and Activity Recognition From Multi-View Videos: Comparative Explorations of Recent Developments. In: *IEEE Journal of Selected Topics in Signal Processing* 6 (2012), Nr. 5, S. 538–552

[29] Huseien, Ghasan F. ; Shah, Kwok W.: A review on 5G technology for smart energy management and smart buildings in Singapore. In: *Energy and AI* 7 (2022), S. 100116. – URL https://www.sciencedirect.com/science/article/pii/S2666546821000653. – ISSN 2666-5468

[30] Kim, Jong-Wook ; Choi, Jin-Young ; Ha, Eun-Ju ; Choi, Jae-Ho: Human Pose Estimation Using MediaPipe Pose and Optimization Method Based on a Humanoid Model. In: *Applied Sciences* 13 (2023), Nr. 4. – URL https://www.mdpi.com/2076-3417/13/4/2700. – ISSN 2076-3417

[31] Lin, Tsung-Yi ; Maire, Michael ; Belongie, Serge ; Bourdev, Lubomir ; Girshick, Ross ; Hays, James ; Perona, Pietro ; Ramanan, Deva ; Zitnick, C. L. ; Dollár, Piotr: *Microsoft COCO: Common Objects in Context*. 2015

[32] Liu, Wei ; Anguelov, Dragomir ; Erhan, Dumitru ; Szegedy, Christian ; Reed, Scott ; Fu, Cheng-Yang ; Berg, Alexander C.: *SSD: Single Shot MultiBox Detector*. S. 21–37. In: *Lecture Notes in Computer Science*, Springer International Publishing, 2016. – URL http://dx.doi.org/10.1007/978-3-319-46448-0_2. – ISBN 9783319464480

[33] LLC., Google: *MediaPipe Pose*. https://developers.google.com/mediapipe/solutions/vision/pose_landmarker/. 2024. – Accessed: 2024-02-28

[34] Maldonado-Bascón, Saturnino ; Iglesias-Iglesias, Cristian ; Martín-Martín, Pilar ; Lafuente-Arroyo, Sergio: Fallen People Detection Capabilities Using Assistive Robot. In: *Electronics* 8 (2019), Nr. 9. – URL https://www.mdpi.com/2079-9292/8/9/915. – ISSN 2079-9292

[35] Mitera, Hannah ; Im, Chanjong ; Mandl, Thomas ; Womser-Hacker, Christa: *Objekterkennung in historischen Bilderbüchern: Eine Evaluierung des Potenzials von Computer-Vision-Algorithmen*. S. 137–150. In: Schmideler, Sebastian (Hrsg.) ; Helm, Wiebke (Hrsg.): *BildWissen – KinderBuch: Historische Sachliteratur für Kinder und Jugendliche und ihre digitale Analyse*. Stuttgart : J.B. Metzler, 2021. – URL https://doi.org/10.1007/978-3-476-05758-7_9. – ISBN 978-3-476-05758-7

[36] Nikolaus, T.: *Stürze und Folgen*. S. 113–127. In: *Basiswissen Medizin des Alterns und des alten Menschen*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2013. – URL https://doi.org/10.1007/978-3-642-28905-7_7. – ISBN 978-3-642-28905-7

[37] Ning, Guanghan ; Liu, Ping ; Fan, Xiaochuan ; Zhang, Chi: A Top-down Approach to Articulated Human Pose Estimation and Tracking. In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018

[38] Oliver Hinz, Maximilian L. von ; Mihale-Wilson, Cristina: Smart Living: Der Mensch im Zentrum. . – Accessed: 2024-03-05

[39] Reinboth, Christian: Tag der Demografiefolgenforschung 2017. In: *Tag der Demografiefolgenforschung 2017*. Wernigerode : Hochschule Harz, 2017. – URL https://hcommons.org/deposits/objects/hc:40508/datastreams/CONTENT/content. – Conference Organized by Hochschule Harz

[40] Ren, Shaoqing ; He, Kaiming ; Girshick, Ross ; Sun, Jian: *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016

[41] Roboflow, Ultralytics: *What is YOLOv8? The Ultimate Guide. [2024]*. https://blog.roboflow.com/whats-new-in-yolov8/#yolov8-architecture-a-deep-dive. 2024. – Accessed: 2024-02-28

[42] Sailer, M. ; Mahr, A. ; Reichstein, C. u. a.: *Lösungsansätze*. https://library.oapen.org/bitstream/handle/20.500.12657/51484/9783658343354.pdf?sequence=1#page=153. 2021. – Accessed: 2024-03-08

[43] Schmidt, Simone: *Expertenstandard Sturzprophylaxe in der Pflege*. S. 87. In: *Expertenstandards in der Pflege - eine Gebrauchsanleitung*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2016. – URL https://doi.org/10.1007/978-3-662-47727-4_5. – ISBN 978-3-662-47727-4

[44] Strese, Hartmut ; Seidel, Uwe ; Knape, Thorsten ; Botthof, Alfons: Smart home in Deutschland. In: *Institut für Innovation und Technik (iit)* 46 (2010), S. 13. – Accessed: 2024-03-05

[45] Ultralytics: *Ultralytics YOLOv8 Docs - Pose Estimation*. https://docs.ultralytics.com/de/tasks/pose/#models. 2024. – Accessed: 2024-03-08

A Anhang

- Dokumentation: Diese Arbeit als Latex und PDF Dokument
- Diagramme: Alle selbsterstellen Diagramme, als Bild und Bearbeitungsdatei
- Code: Das vollständige Arbeitsverzeichniss

Erklärung zur selbstständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original