Rheinische
Friedrich-Wilhelms-
Universität Bonn

Institut für Informatik
Abteilung VI
Autonome Intelligente Systeme

Rheinische
Friedrich-Wilhelms-Universität Bonn

Bachelorarbeit

Semantische 3D Kartenerstellung auf Objektebene mit einem Smart Edge Sensor Netzwerk

Autor:
Julian Hau

Erstgutachter:
Prof. Dr. Sven Behnke

Zweitgutachter:
Dr. Volker Steinhage

Betreuer:
Simon Bultmann, M. Sc.

Datum: 7. August 2022

Eidesstattliche Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen verwendet habe. Die Stellen der Arbeit sowie evtl. beigefügte Abbildungen, Zeichnungen oder Grafiken, die anderen Werken dem Wortlaut oder Sinn nach entnommen wurden, habe ich unter Angabe der Quelle kenntlich gemacht.

_____ _____

Ort, Datum Unterschrift

Zusammenfassung

Um autonomen Robotern Interaktion zu ermöglichen, ist es wichtig, die Umgebung, in der die Roboter operieren, möglichst genau zu kennen. Dazu wird häufig eine volumetrische Karte der Umgebung mit geometrischen und semantischen Informationen generiert. Durch die Integration von Informationen auf Objektebene kann das Szenenverständnis weiter erhöht werden. In dieser Arbeit wird ein Smart Edge Sensor Netzwerk bestehend aus RGB Kameras und Intel RealSense RGB-D Kameras verwendet. Die Verarbeitung der Rohdaten und Segmentierung findet auf den Sensorboards statt, während die verarbeiteten Daten auf dem Backend fusioniert werden. Das bestehende Framework ermöglicht bereits eine allozentrische semantische Karte zu generieren. Das Ziel dieser Arbeit ist es, die Karte um Informationen auf Objektebene zu erweitern, damit Bewegungen erkannt und temporär verdeckte Objekte korrekt dargestellt werden. Dazu werden drei verschiedene Ansätze implementiert und evaluiert: Im ersten Ansatz wird jedes Objekt durch ein Ellipsoid, welches die Statistik der Punktmessungen abbildet, repräsentiert. Für eine alternative Methode werden Keypunkte auf dem Objekt detektiert, während ein entsprechendes Objekt-Mesh Informationen über die Geometrie des Objektes liefert. Der dritte Ansatz generiert pro Objekt eine Subkarte, welche die Positionen aller belegten Voxel im Objektkoordinatensystem speichert. Die Bestimmung der Objektpose geschieht mit Hilfe des PnP-Algorithmus auf Basis der Keypunkt-Detektionen. Alternativ kann für die Subkarten- und die Ellipsoid-Repräsentation die Objektpose mittels Hauptkomponentenanalyse der jeweiligen Punktwolke berechnet werden. Evaluiert werden die Methoden in Experimenten mit dem Sensor Netzwerk und auf dem Behave-Datensatz. Die Keypunkt-basierte Posenschätzung liefert bessere Ergebnisse als die Posenschätzung der Hauptkomponentenanalyse. Gleichzeitig benötigen Ellipsoide und Keypunkt-Skelette eine deutlich geringere Datenübertragungsrate zwischen Sensorboards und Backend als die Subkarten.

Inhaltsverzeichnis

1 Einleitung **1**

2 Grundlagen **3**
2.1 Occupancy Grid Map 3
2.2 6DoF-Pose 4
2.3 Iterative-Closest-Point Algorithmus 6
2.4 Lochkamera 7
2.5 Perspective-n-Point Algorithmus 8

3 Literaturübersicht **11**
3.1 Semantisches Mapping 11
3.2 Posenschätzung mit Hilfe von Keypoints 14
3.3 Behave-Datensatz 14

4 Methodik **17**
4.1 Semantische Karte mit expliziter Objektrepräsentation 17
4.2 Ellipsoide 18
4.2.1 Eingangsdaten 19
4.2.2 Fusion der Kameraperspektiven 19
4.2.3 Objekttracking 20
4.2.4 Clean-Up 21
4.3 Keypunkt-Skelette 22
4.3.1 Generieren von Trainingsdaten 22
4.3.2 Netzwerk 23
4.3.3 Eingangsdaten 24
4.3.4 Fusion der Kameraperspektiven 25
4.3.5 Objekttracking und Clean-Up 26
4.4 Subkarten 27
4.4.1 Eingangsdaten 27
4.4.2 Fusion der Kameraperspektiven 27
4.4.3 Objekttracking und Clean-Up 28

5 Evaluation **31**
5.1 Metriken . . . 31
5.1.1 Intersection-over-Union . . . 31
5.1.2 Translationsfehler . . . 32
5.1.3 Orientierungsfehler . . . 32
5.2 Auswertung auf dem Behave-Datensatz . . . 32
5.2.1 Posenevaluation . . . 33
5.2.2 Rückprojektion in die Kamerabilder . . . 40
5.3 Experimente mit dem Sensor Netzwerk . . . 44
5.3.1 Qualitative Ergebnisse . . . 45
5.3.2 Datenübertragungsrate . . . 48

6 Fazit **51**
6.1 Ausblick . . . 52

Anhang **53**

1 Einleitung

Die Tätigkeiten autonomer Roboter haben in den letzten Jahren stark zugenommen. Eine zentrale Voraussetzung für das Erfüllen einer Vielzahl von Tätigkeiten ist ein möglichst genaues Szenenverständnis. Dazu wird die Umgebung des autonomen Roboters mit verschiedenen Sensoren aufgezeichnet. Aus den gewonnenen Daten wird eine Karte erstellt, die den Roboter bei seiner Tätigkeit unterstützt.

In der Robotik Halle unseres Instituts befindet sich ein Smart Edge Sensor Netzwerk, bestehend aus 16 Sensorboards mit RGB-Kameras und 4 Sensorboards mit RGB-D Kameras. Mit diesen Sensoren wird eine globale allozentrische Karte [4] mit semantischen und geometrischen Informationen generiert und die Posen von Personen [5] werden bestimmt. In dieser Arbeit wird die Karte um drei verschiedene Objektrepräsentationen ergänzt, die Informationen über die einzelnen Objektinstanzen speichern. Dazu werden die Daten vierer Sensorboards mit RGB-D Kameras genutzt. Für die Objektrepräsentationen Ellipsoide, Keypunkt-Skelette und Subkarten werden auf den Sensorboards verschiedenen Eigenschaften der Punktwolken berechnet und an das Backend übermittelt. Die Fusion und Visualisierung der Daten erfolgt anschließend auf dem Backend. Die Posenschätzung wird entweder mit dem PnP-Algorithmus auf Basis von Keypunkt-Detektionen oder mit einer Hauptkomponentenanalyse des Punktwolkensegmentes durchgeführt. Durch das Zuordnen der Informationen zu Objektinstanzen sollen Bewegungen verfolgt werden. Außerdem soll eine robuste Darstellung auch bei temporärer Verdeckung möglich sein. Die generierten Objektrepräsentationen werden zusammen mit den Skeletten der Personen aus [5] in der globalen allozentrischen Karte [4] visualisiert.

Der Aufbau dieser Arbeit ist wie folgt: Im 2. Kapitel werden einige Grundlagen erklärt, die für die Methode relevant sind. Anschließend folgt in Kapitel 3 die Literaturübersicht. Dabei wird auf andere Verfahren zur Erstellung semantischer Karten eingegangen, und es werden die Arbeiten näher erläutert, auf denen diese Arbeit aufbaut. Zusätzlich wird der zur Evaluation verwendete Behave-Datensatz [2] vorgestellt. Danach werden in Kapitel 4 die Methode und Unterschiede zwischen den Objektrepräsentationen im Detail erklärt. In Kapitel 5 wird die Methode auf dem Behave-Datensatz [2] und in Experimenten mit dem Sensor Netzwerk evaluiert. Zum Schluss folgt im 6. Kapitel das abschließende Fazit und ein Ausblick auf mögliche zukünftige Arbeiten.

2 Grundlagen

In den folgenden Abschnitten werden einige Grundlagen erläutert, auf welchen diese Arbeit aufbaut. Dazu zählen die Occupancy Grid Maps (vgl. Abs. 2.1), die 6DoF-Pose (vgl. Abs. 2.2), der ICP-Algorithmus (vgl. Abs. 2.3), das Lochkameramodell (vgl. Abs. 2.4) und der PnP-Algorithmus (vgl. Abs. 2.5).

2.1 Occupancy Grid Map

Occupancy Grid Maps diskretisieren den Raum in Zellen mit einer festen Größe. Jede dieser Zellen korrespondiert dabei entweder zu einem Volumen in 3D oder zu einer Fläche in 2D, die entweder frei, belegt oder unbeobachtet sein kann. Somit können die Zellen als binäre Zufallsvariablen beschrieben werden. Abbildung 2.1 zeigt beispielhaft eine zweidimensionale Occupancy Grid Map. Schwarze Zellen sind belegt, während weiße Zellen frei und graue Zellen unbeobachtet sind.

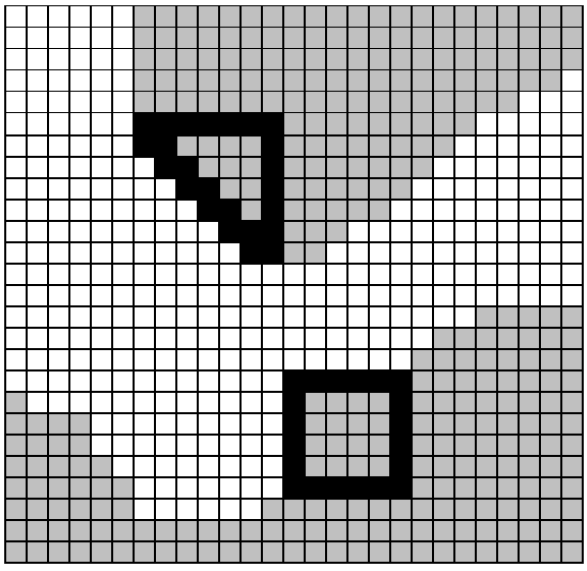


Abbildung 2.1: 2D Occupancy Grid Map [17]

Sofern die Sensordaten $z_{1:t}$ von Zeitpunkt 1 bis t und die Position des Sensors $x_{1:t}$ von Zeitpunkt 1 bis t bekannt sind, kann eine Aussage über den Zustand von Zelle m_i mit Index i getroffen werden. Die Zufallsverteilung der Karte ergibt sich dann als Produkt über alle Zellen:

$$p(m|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t}) \,. \tag{2.1}$$

Bei den in dieser Arbeit generierten Subkarten handelt es sich um Occupancy Grid Maps.

2.2 6DoF-Pose

Die Pose eines starren Körpers kann im dreidimensionalen Raum durch 6 Freiheitsgrade (6 degrees of freedom) beschrieben werden. Bei drei Freiheitsgraden handelt es sich um die Translationen entlang der x-, y- und z-Achse. Die Freiheitsgrade der Orientierung bestehen aus den drei Rotationen um die x-, y- und z-Achse (vgl. Abb. 2.2). Die Translation $\mathbf{t} \in \mathbb{R}^3$ und Orientierung $\mathbf{R} \in \mathbb{R}^{3\times3}$ beschreiben somit die Pose eines Objektes vollständig:

$$\mathbf{t} = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}, \quad \mathbf{R} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} .$$

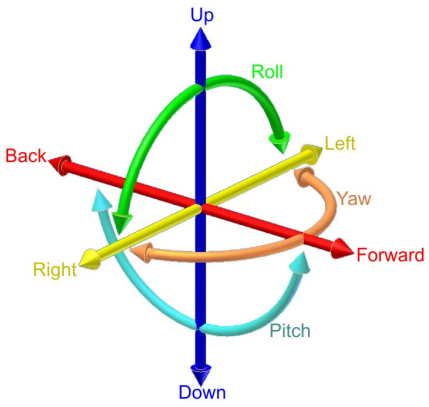


Abbildung 2.2: Die 6 Freiheitsgrade eines starren Körpers im 3D-Raum [27]

Mit der Transformation $\mathbf{T}$ kann ein Punkt $\mathbf{p} = (x, y, z, 1)^T$ in homogenen Koordinaten anschließend vom Weltkoordinatensystem in das Objektkoordinatensystem transformiert werden:

$$\mathbf{p_{Objekt}} = \mathbf{T} \cdot \mathbf{p_{Welt}} \tag{2.2}$$

mit

$$\mathbf{T} = \begin{pmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{pmatrix} . \tag{2.3}$$

Die Rücktransformation von Objekt- zu Weltkoordinaten erfolgt mit der inversen Transformationsmatrix:

$$\mathbf{p_{Welt}} = \mathbf{T}^{-1} \cdot \mathbf{p_{Objekt}} \tag{2.4}$$

mit

$$\mathbf{T}^{-1} = \begin{pmatrix} \mathbf{R}^{-1} & -\mathbf{R}^{-1} \cdot \mathbf{t} \\ 0 & 1 \end{pmatrix} . \tag{2.5}$$

Alternativ kann die Orientierung als Einheitsquaternion $\mathbf{q}$ dargestellt werden. Diese Darstellung hat den Vorteil, dass nur vier Werte zur Beschreibung der Rotation benötigt werden:

$$\mathbf{q} = \begin{pmatrix} q_x \\ q_y \\ q_z \\ q_w \end{pmatrix} = q_x \cdot i + q_y \cdot j + q_z \cdot k + q_w \, . \tag{2.6}$$

Dabei sind $q_x, q_y, q_z, q_w \in \mathbb{R}$ und i, j, k sind die Imaginärteile von $\mathbf{q}$, für die gilt

$$i^2 = j^2 = k^2 = ijk = -1 \, . \tag{2.7}$$

Ein Quaternion $\mathbf{q}$ wird normiert, indem es auf Einheitslänge skaliert wird:

$$\frac{\mathbf{q}}{\|\mathbf{q}\|_2} = \frac{q_x}{\|\mathbf{q}\|_2} i + \frac{q_y}{\|\mathbf{q}\|_2} j + \frac{q_z}{\|\mathbf{q}\|_2} k + \frac{q_w}{\|\mathbf{q}\|_2} \tag{2.8}$$

mit

$$\|\mathbf{q}\|_2 = \sqrt{q_x^2 + q_y^2 + q_z^2 + q_w^2} \, . \tag{2.9}$$

Die Pose der Objekte wird in dieser Arbeit durch Translationsvektor und Einheitsquaternion beschrieben. Für das Eintragen neuer Messergebnisse in die Subkartenstruktur ist eine Transformation in das jeweilige Objektkoordinatensystem notwendig. Die Visualisierung erfolgt durch Rücktransformation in die Weltkoordinaten.

2.3 Iterative-Closest-Point Algorithmus

Iterative-Closest-Point (ICP) ist ein Algorithmus, mit dem zwei korrespondierende Punktwolken $P = \{\mathbf{p_1}, ..., \mathbf{p_n}\}$ und $Q = \{\mathbf{q_1}, ..., \mathbf{q_n}\}$ iterativ aneinander angepasst werden können. Gesucht ist dabei die Translation $\mathbf{t}$ und Rotation $\mathbf{R}$, welche die Summe des quadratischen Fehlers E für alle korrespondierenden Punkte $\mathbf{p}_i$ und $\mathbf{q}_i$ minimieren:

$$E(\mathbf{R}, \mathbf{t}) = \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{p}_i - \mathbf{R}\mathbf{q}_i - \mathbf{t}\|_2^2 \,. \tag{2.10}$$

Der Algorithmus kann dabei in die folgenden Schritte unterteilt werden:

1. Finde für jeden Punkt in der Ausgangspunktwolke P den nächsten Nachbar als korrespondierenden Punkt in der Referenzpunktwolke Q.
2. Bestimme eine Kombination aus Translation $\mathbf{t}$ und Rotation $\mathbf{R}$, die die Punkt-zu-Punkt-Abstandsquadrate aller korrespondierenden Punkte minimiert.
3. Transformiere die Ausgangspunktwolke P mit der Translation $\mathbf{t}$ und Rotation $\mathbf{R}$.
4. Beende, wenn ein Terminationskriterium erreicht wurde, ansonsten wiederhole den Vorgang.

Die Kriterien für eine Termination des Algorithmus sehen im Allgemeinen wie folgt aus:

1. Die Anzahl der Iterationen hat die spezifizierte maximale Anzahl an Iterationen erreicht.
2. Die Differenz zwischen der aktuell berechneten und der vorherigen Transformation bestehend aus Translation $\mathbf{t}$ und Rotation $\mathbf{R}$ liegt unter einem angegebenen Schwellwert.
3. Die Summe des quadratischen Fehlers liegt unter einem angegebenen Schwellwert.

In Abbildung 2.3 sind drei verschiedene Iterationen des ICP-Algorithmus am Beispiel einer blauen Ausgangslinie und einer roten Referenzlinie dargestellt. Die Korrespondenzen zwischen beiden Linien sind mit schwarzen Linien markiert. Entsprechend kann ICP auch für eine Anpassung der Subkarten an die zugehörige neu detektierte Punktwolke verwendet werden.

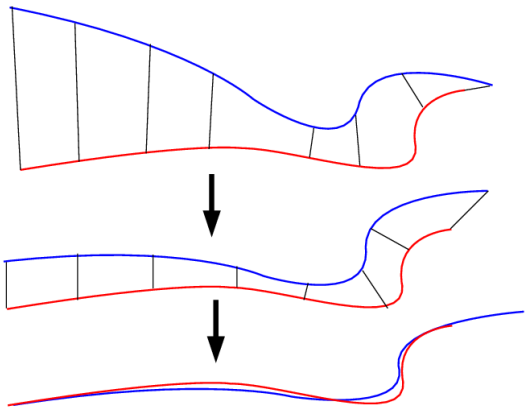


Abbildung 2.3: ICP angewendet auf zwei Linien [25]

2.4 Lochkamera

Bei der Lochkamera handelt es sich um ein einfaches Kameramodell, welches das Verhältnis von Punkten im 3D-Raum und korrespondierenden Punkten auf der 2D-Bildebene beschreibt (vgl. Abb. 2.4). Gleichung (2.11) beschreibt die verzerrungsfreie Projektion eines Punktes $\mathbf{P_W} = (x, y, z, 1)^T$ in homogenen 3D-Weltkoordinaten zu einem Pixel $\mathbf{p} = (u, v, 1)^T$ in der 2D-Bild-Ebene, wobei $\mathbf{K}$ die intrinsische Kameramatrix, $[\mathbf{R}\,\mathbf{t}]$ die Transformation von Weltkoordinaten in Kamerakoordinaten (vgl. (2.2)) und s ein Skalierungsfaktor ist.

$$s \cdot \mathbf{p} = \mathbf{K} \cdot [\mathbf{R}\,\mathbf{t}] \cdot \mathbf{P_W} \tag{2.11}$$

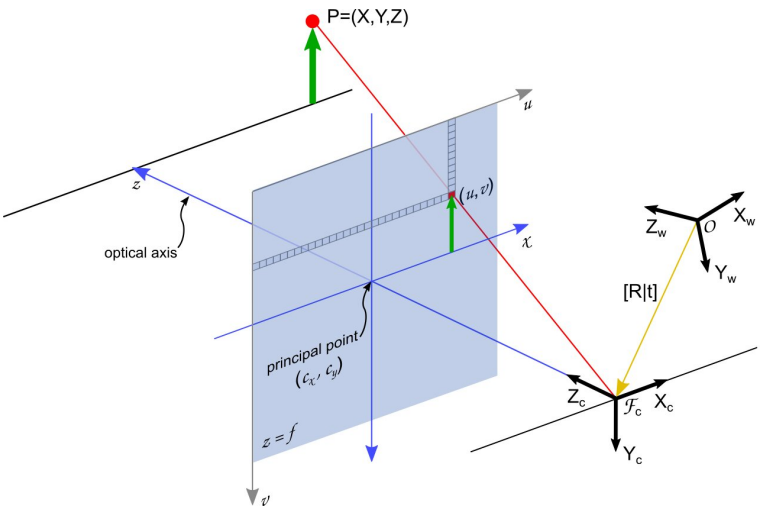


Abbildung 2.4: Lochkamera Modell [18]

Die intrinsische Kameramatrix $\mathbf{K}$ enthält die horizontalen und vertikalen Brennweiten f_x und f_y in Pixel-Einheit und die Koordinaten des optischen Zentrums (c_x, c_y), welche sich als Schnittpunkt von optischer Achse und Bildebene ergeben:

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} . \tag{2.12}$$

Die Transformationsmatrix $[\mathbf{R}\,\mathbf{t}]$ transformiert einen Punkt $\mathbf{P_W}$ in homogenen Weltkoordinaten zu einem Punkt $\mathbf{P_C} = (x_C, y_C, z_C)$ Kamerakoordinaten:

$$\begin{pmatrix} x_C \\ y_C \\ z_C \end{pmatrix} = [\mathbf{R}\,\mathbf{t}] \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} . \tag{2.13}$$

Gilt $z_C \neq 0$, dann kann Gleichung (2.11) zu folgender Gleichung vereinfacht werden:

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f_x\, x_C/z_C + c_x \\ f_y\, y_C/z_C + c_y \end{pmatrix} . \tag{2.14}$$

Das Modell der Lochkamera kommt in dieser Arbeit sowohl bei der Erstellung von Trainingsdaten für die Keypunkt-Posenschätzung (vgl. Abs. 4.3.1), als auch bei der Evaluation unserer Methode zum Einsatz (vgl. Abs. 5.2.2).

2.5 Perspective-n-Point Algorithmus

Mit dem PnP-Algorithmus kann die 6DoF-Pose eines Objektes anhand von 3D-Punkten in Welt-Koordinaten und korrespondierenden 2D-Punkten in einer Bildebene berechnet werden. Es werden die Translation $\mathbf{t}$ und Rotation $\mathbf{R}$ des Objektes gesucht, welche den Reprojektionsfehler der Objektpunkte von 3D zu 2D minimieren, wie in Abbildung 2.5 illustriert.

Die Projektion der 3D-Punkte in die Bildebene geschieht über die Gleichung (2.13) der Lochkamera. Dementsprechend ist die geschätzte Pose die Transformationsmatrix $[\mathbf{R}\,\mathbf{t}]$, bestehend aus Translation $\mathbf{t}$ und Rotation $\mathbf{R}$. Für die Minimierung des Reprojektionsfehlers gibt es verschiedene Varianten. In dieser Arbeit wird die iterative Methode, basierend auf der Levenberg-Marquardt Optimierung, verwendet. Die Resultierende Pose minimiert die Summe der Distanzquadrate zwischen den im Bild beobachteten 2D-Punkten und den reprojizierten Objektpunkten. Zusätzlich nutzt die verwendete Variante des Algorithmus das Random Sample Consensus (RANSAC) Verfahren, um robuster mit Ausreißern umgehen zu können.

Dazu wird die Posenschätzung auch mit zufälligen Teilmengen der Objektpunkte berechnet und Ausreißer werden entsprechen klassifiziert.

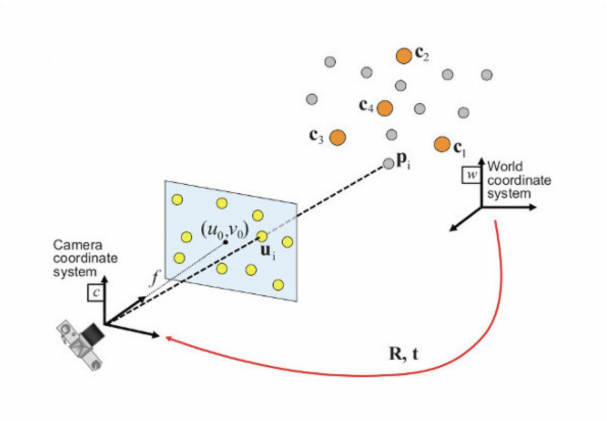


Abbildung 2.5: Illustration des PnP-Algorithmus [20]

3 Literaturübersicht

In den folgenden Abschnitten wird eine Übersicht über verwandte Arbeiten gegeben. In Abschnitt 3.1 werden Verfahren zur Erstellung semantischer Karten beschrieben. Dabei wird insbesondere genauer auf eine aktuelle Arbeit von Bultmann und Behnke [4] eingegangen, welche als Grundlage für diese Arbeit dient. Anschließend werden in Abschnitt 3.2 Verfahren zur Posenschätzung mit Keypoints erläutert. Danach folgt in Abschnitt 3.3 die Vorstellung des Behave-Datensatzes [2], auf dem diese Arbeit evaluiert wird.

3.1 Semantisches Mapping

Das Erstellen von dreidimensionalen Karten, welche für die Lokalisierung und Pfadplanung von mobilen Robotern genutzt werden, kann auf verschiedene Arten und Weisen erfolgen. Eine weitverbreitete Methode sind die Occupancy Grid Maps. Octomap [13], eines der bekanntesten 3D-Occupancy-Mapping Frameworks nutzt eine effiziente hierarchische Octree-Struktur, um die Belegungswahrscheinlichkeiten einzelner Voxel der Umgebung zu speichern. Eine andere weitverbreitete Kartenart sind die TSDF-Karten (Truncated Signed Distance Function). Eine TSDF-Karte besteht aus mehreren 3D Voxeln, die jeweils die Distanz zu der nächstgelegenen Oberfläche speichern. Ein Beispiel für ein Framework zum Erstellen TSDF-basierter Karten ist Voxblox [19]. Beide genannten Verfahren liefern allerdings noch keine semantischen Informationen in der Karte. Andere Arbeiten verwenden vergleichbare Ansätze und erweitern die Karte um semantische Informationen. Stückler et al. [26] fusionieren probabilistische Segmentierungen aus mehreren RGB-D Kameraperspektiven in eine Voxel-basierte 3D Karte. MaskFusion [21] ist ein RGB-D SLAM (Simultaneous Localisation and Mapping) System, das mehrere Objekte in einer Szene tracken und rekonstruieren kann, ohne die Modelle der Objekte zuvor genauer zu kennen. Die erstellte Karte basiert dabei auf Surfels (surface elements). Dabei handelt es sich um Elemente die neben der Position auch Oberflächeneigenschaften wie den Radius und die Oberflächennormale speichern. MID-Fusion [31] nutzt eine Octree-TSDF-Struktur, um SLAM mit einer RGB-D Kamera umzusetzen. Neben den geometrischen Daten werden

auch Farbe, Semantik und eine Vordergrund-Wahrscheinlichkeit in die Karte fusioniert. Voxblox++ [10] nutzt eine geometrische Segmentierung und eine semantische Instanz-Segmentierung, um eine semantische Karte mit Informationen auf Objektebene in statischen Umgebungen zu generieren. Mit TSDF++ [11] haben Grinvald et al. eine Methode entwickelt, die TSDF-Subvolumen für Objektinstanzen erstellt. Die Subvolumen werden wiederum von einer globalen volumetrischen Karte, welche an jeder Position mehrere Objekte speichern kann, referenziert, wodurch temporär verdeckte Objekte nicht neu rekonstruiert werden müssen und dynamische Umgebungen darstellbar sind.

Kürzlich wurde von Bultmann und Behnke [4] eine Methode vorgestellt, die die Perspektiven mehrerer RGB-D Kameras fusioniert und daraus ein dreidimensionale allozentrische Karte mit semantischen Informationen erstellt. Neben der semantischen Karte werden auch 3D Posen von Personen berechnet und in der Karte visualisiert [5]. Um die Datenübertragungsrate gering zu halten, werden die Rohdaten gänzlich auf den Sensorboards behandelt, während nur semantische Informationen über das Netzwerk an das Backend übermittelt werden. Zusätzlich werden bewegliche Objekte berücksichtigt, indem via Raytracing unbelegte Voxel freigezeichnet werden können. Im Folgenden wird der Aufbau und Ablauf der Pipeline (vgl. Abb. 3.1) genauer erläutert. Die Hardware des Systems besteht aus 20 Smart-Edge-Sensoren, wovon 16 Sensoren auf dem Google EdgeTPU Dev Board basieren und mit einer RGB Kamera ausgestattet sind. Die weiteren 4 Sensoren basieren auf dem Nvidia Jetson Xavier NX developer kit und sind jeweils mit einer Intel RealSense D455 RGB-D Kamera und einer FLIR Lepton 3.5 Wärmebildkamera verbunden. Die Personen- und Objektdetektion erfolgt unter Verwendung der MobileDet Architektur [30]. Trainiert wurde der RGB-Detektor auf dem COCO Datensatz [16] mit einer Input-Auflösung von 848×480 und der Wärmebild-

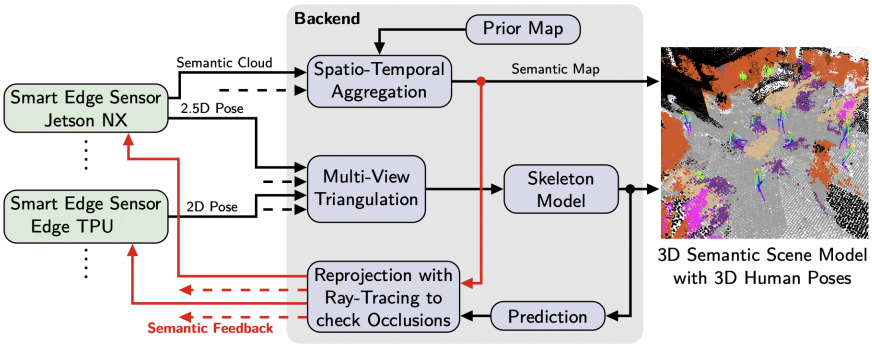


Abbildung 3.1: Pipeline zur Erstellung einer semantischen Karte und Bestimmung von Personen Posen [4]

Detektor auf dem ChaLearn IPHD Datensatz [9] mit einer Input-Auflösung von 160×120. Allerdings wird der Wärmebild-Detektor lediglich für die Detektion von Personen verwendet, da die Priorität der Pipeline auf dem Erkennen und Tracken von Personen liegt. Für die Umsetzung der semantischen Segmentierung verwenden Bultmann und Behnke die DeepLab v3+ [8] Architektur mit einem MobileNet v3 Backbone [14], trainiert auf den Indoor-Szenen des ADE20K Datensatzes [33]. Da die verwendeten Kameras ein Seitenverhältnis von 16:9 haben, wurde eine Input-Auflösung von 849×481 gewählt. Die Klassen wurden auf die für ein Indoor-Szenario relevanten reduziert und werden in Abbildung 3.2 dargestellt. Aus den aufgezeichneten Tiefenbildern wird eine geometrische Punktwolke generiert, die unter Verwendung von Detektionen und semantischer Segmentierung mit entsprechenden Klasseninformationen ergänzt wird. Um die Datenmenge zu reduzieren, wird ein Voxelgrid-Filter mit einer Auflösung von 5 cm und ein Statistical-Outlier-Filter angewendet. Die Ausgabepunktwolke, die von Sensor zu Backend gesendet wird, wird mit einer Frequenz von 1 Hz berechnet und enthält pro Punkt neben der Klassenfarbe auch einen Wahrscheinlichkeitsvektor für die Klassenzugehörigkeit. Auf dem Backend wird anschließend die allozentrische semantische Karte als Occupancy Grid Map mit einer Voxelgröße von 10 cm generiert. Als Datenstruktur dient ein Sparse-Voxel-Grid basierend auf Hashmaps, um eine effiziente Speicherung zu ermöglichen. Zusätzlich wird die Karte mit A-priori-Informationen in Form eines 3D Gebäudescans initialisiert. Punkte, die als Person klassifiziert wurden, werden nicht in die Karte integriert, da Personen bereits durch 3D Skelette mit einem höheren Detailgrad dargestellt werden. Der Raytracing-Schritt wurde in Form des Bresenham-Algorithmus [1] implementiert und sorgt dafür, dass durch Bewegung freigewordene Voxel mit der Zeit als frei markiert werden.

Abbildung 3.2: Die 16 semantischen Klassen der Methode [4]

3.2 Posenschätzung mit Hilfe von Keypoints

Um eine Keypoint-basierte Posenschätzung zu realisieren, müssen zunächst charakteristische Punkte auf dem zu schätzenden Körper definiert werden. Diese Keypunkte werden anschließend im Sensorbild detektiert, wodurch sich eine Pose durch die Lage der Punkte zueinander ergibt. Das Zuordnen der Keypunkte zu den Objektinstanzen erfolgt im Allgemeinen auf zwei Arten: bottom-up oder top-down. Bei dem bottom-up-Ansatz geschieht zuerst die Detektion der Keypunkte. Danach werden die einzelnen Punkte zu Instanzen verknüpft. Der top-down-Ansatz verwendet hingegen einen vorgeschalteten Detektor. Für jedes detektierte Objekt werden die Keypunkte innerhalb der Bounding Box gesucht. Während der top-down-Ansatz das Risiko birgt, dass durch falsche Detektionen Keypunkte verloren gehen, ist es beim bottom-up-Ansatz möglich, dass fälschlicherweise Keypunkte verschiedener Objektinstanzen miteinander verknüpft werden. Häufig haben top-down-Ansätze eine höhere Genauigkeit und langsamere Geschwindigkeit als bottom-up-Ansätze [6].

Zappel et al. [32] haben auf Basis des bottom-up-Ansatzes von OpenPose [6] Keypunkte auf verschiedenen Objekten des YCB-V-Datensatzes [28] detektiert. Die 6DoF-Pose der Objekte wurde mit Hilfe des PnP-RANSAC Algorithmus berechnet, welcher für 2D und 3D Korrespondenzen starrer Körper eine entsprechende Translation und Rotation ermittelt.

In der Arbeit von Bultmann und Behnke [4] wird ein top-down-Ansatz zur Bestimmung menschlicher Posen verwendet. Als Basis dient für die Posenerkennung die Architektur von Xiao et al. [29]. Der ResNet Backbone wurde jedoch als Anpassung für die Sensorboards durch MobileNet v3 [14] ersetzt. Durch Triangulation mehrerer Ansichten werden aus den 2D Keypunkten entsprechende 3D Skelette berechnet (vgl. Abb. 3.1).

3.3 Behave-Datensatz

Diese Arbeit wird auf Teilen des Behave-Datensatzes [2] ausgewertet. Der Datensatz umfasst 321 Videosequenzen mit insgesamt ca. 15k Frames, aufgezeichnet mit 4 Kinect RGB-D Kameras mit einer Framerate von 1 Hz. In den verschiedenen Szenarien interagieren 8 Personen mit 20 Objekten an 5 verschiedenen Umgebungen. Die Interaktionen umfassen alltägliche Tätigkeiten mit gewöhnlichen Gegenständen in natürlichen Umgebungen. Für jeden Frame existieren Kameraposen, pseudo-ground-truth Objekt-Fit und Masken für Objekt und Person. Die Objektsegmentierung und das Fitting wurden durchgeführt, indem die Objekte zuvor mit einem 3D-Scanner aufgezeichnet wurden. Die gescannten Meshes sind ebenfalls in

dem Datensatz enthalten. Um die 6D-Pose des gescannten Meshes im jeweiligen Frame zu bestimmen, wurden manuell Objekt-Keypunkte in den Bildern markiert. Der sich ergebende Objekt-Fit erzeugt, verglichen mit der Kinect-Punktwolke, eine Chamfer-Distanz von 2.42 cm. Die Segmentierungsmasken wurden durch Rückprojektion der Objektmeshes in die Kamerabilder bestimmt. Desweiteren enthält der Datensatz auch pseudo-ground-truth SMPL und Kontakt-Annotationen, die für unsere Auswertung jedoch unerheblich sind.

Abbildung 3.3: RGB-Bilder und Visualisierung verschiedener Szenarien des Behave-Datensatzes [2]

4 Methodik

In diesem Kapitel wird die in dieser Arbeit entwickelte Methode zur Repräsentation dynamischer Objekte in einer semantischen Karte genauer erläutert. In Absatz 4.1 wird die Grundlage dieser Arbeit beschrieben und es wird ein Überblick über den Ablauf der Methode gegeben. Anschließend wird in den Abschnitten 4.2, 4.3 und 4.4 näher auf die gewählten Objektrepräsentationen eingegangen: Ellipsoide, Keypunkt-Skelette und Subkarten. Dabei werden zunächst die für die Berechnung der jeweiligen Struktur benötigten Eingangsdaten genannt. Danach folgt die Erläuterung der Fusion der Perspektiven, des Trackings und des Clean-Up Schrittes. Für die Keypunkte wird zusätzlich das Generieren der Trainingsdaten und das verwendete Netzwerk erklärt.

4.1 Semantische Karte mit expliziter Objektrepräsentation

Als Basis für diese Arbeit dient die in Abschnitt 3.1 näher erläuterte Arbeit von Bultmann und Behnke [4] zur Erstellung einer allozentrischen semantischen Karte. In dieser Karte können bereits Objektbewegungen durch Raytracing und Freizeichnen unbelegter Voxel dargestellt werden. Allerdings geschieht dies nur langsam, da die Updatefrequenz lediglich 1 Hz beträgt. Darüber hinaus ist es häufig nicht möglich, Objekttrajektorien nachzuvollziehen.

Die in dieser Arbeit implementierten Objektrepräsentationen sollen diese Nachteile ausbessern, sodass Objektbewegungen erkannt und temporär verdeckte Objekte korrekt dargestellt werden. Folglich können Klassen, für die eine detailreichere Darstellung gewünscht ist, aus der allozentrischen semantischen Karte entfernt und separat visualisiert werden. So wie es auch für Personen erfolgt (vgl. Abb. 3.1). Ellipsoide, Keypunkte und Subkarten werden mit Hilfe von 4 Jetson NX Smart Edge Sensorboards mit Intel RealSense RGB-D Kameras ermittelt, die auch in [4] verwendet wurden. Die Verarbeitung der RGB- und Tiefenbilder erfolgt ebenfalls wie in [4]. Der Ablauf des Objekt-Mappings ist in Abbildung 4.1 dargestellt. Nach der semantischen Segmentierung [4] werden die Punktwolken mit dem Euclidean Cluster Extraction Algorithmus [22], gefolgt von einem

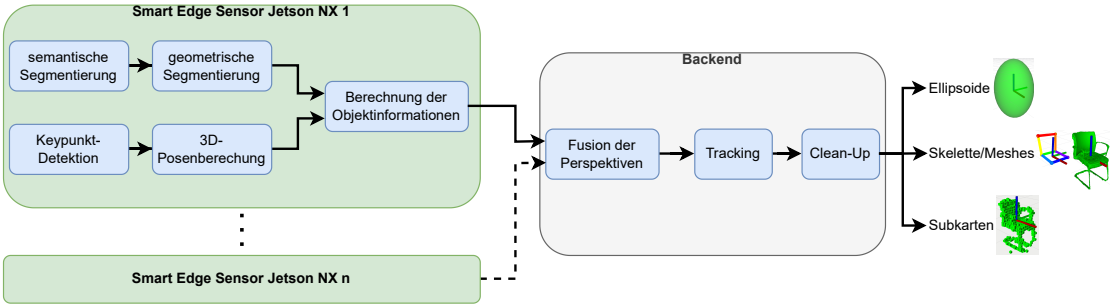


Abbildung 4.1: Ein Überblick über die Pipeline des Objekt-Mappings

Statistical-Outlier-Filter [22], geometrisch segmentiert, um ein Punktwolkensegment pro detektiertem Objekt zu erhalten. Gleichzeitig findet die Detektion von Keypunkten in den RGB-Bildern statt. Für jedes detektierte 2D-Keypunkt-Skelett wird anschließend eine entsprechende 6DoF-Pose mit dem PnP-RANSAC Algorithmus berechnet. Die erhaltenen Keypunkt-basierten Posen werden den semantisch und geometrisch segmentierten Punktwolken zugeordnet. Nun werden verschiedene Objekteigenschaften bestimmt. Diese Eigenschaften unterscheiden sich je nach gewählter Datenstruktur und werden in den Abschnitten 4.2.1, 4.3.3 und 4.4.1 genauer beschrieben. Die berechneten Daten werden von den Sensorboards an das Backend gesendet, wo sie als Eingangsdaten für das Objekt-Mapping dienen. Auf dem Backend werden die Daten der verschiedenen Sensoren fusioniert, indem die Objektinformationen identischer Objekte kombiniert werden. Im Tracking Schritt werden die neuen Beobachtungen zuvor beobachteten Objektinstanzen zugeordnet. Dies ermöglicht Trajektorien nachzuvollziehen und eine robuste Darstellung bei Verdeckung. Abschließend folgt der Clean-Up Schritt, wodurch fehlgeschlagene Zuordnungen bereinigt werden sollen. Die Ausgabe in Form von Ellipsoiden, Keypunkt-Skeletten und Subkarten kann in der globalen allozentrischen Karte visualisiert werden.

4.2 Ellipsoide

Die Ellipsoide speichern neben der Position und Orientierung des Objektes auch die Statistik der Punktmessungen, die getätigt wurden. Dadurch kann die Objektpose und ein Ellipsoid-Volumen, in dem sich das Objekt befindet, ermittelt werden. Diese Darstellung liefert allerdings keine genauere Beschreibung der Form des Objekts.

4.2.1 Eingangsdaten

Zu den übermittelten Daten für die Ellipsoid-Darstellung gehören die Translation, die Orientierung, die statistische Verteilung und die Anzahl der Punkte des Punktwolkensegmentes. Die Translation ergibt sich als geometrischer Schwerpunkt des Punktwolkensegmentes in Kamerakoordinaten. Um die Orientierung zu berechnen, wird eine Hauptkomponentenanalyse der Punktwolke durchgeführt. Dazu wird die Kovarianzmatrix bestimmt und die Eigenvektoren und Eigenwerte werden extrahiert. Als Hauptachse des Objektkoordinatensystems wird der Eigenvektor mit dem höchsten Eigenwert gewählt. Zudem wird die Annahme gemacht, dass die Z-Achse des Objektkoordinatensystems der Z-Achse des Weltkoordinatensystems entspricht, da die Objekte im Szenario größtenteils auf einer Ebene stehen und sich auch nur in dieser Ebene bewegen. Die Hauptachse wird dementsprechend projiziert. Die erhaltene Rotationsmatrix wird in ein Quaternion umgewandelt und normalisiert. Mit der Orientierung kann die mittlere Varianz des Punktwolkensegmentes in X-, Y- und Z-Richtung bestimmt werden, woraus sich wiederum die 2-σ-Umgebung als statistische Verteilung berechnen lässt.

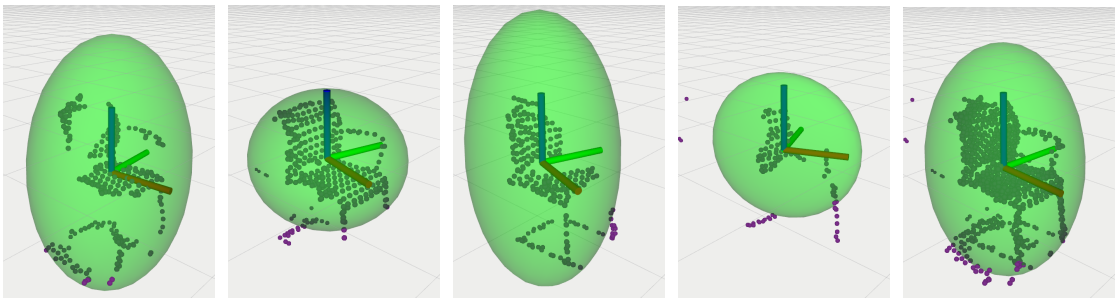
Nachdem diese Daten auf den Sensorboards berechnet wurden, werden sie an das Backend übermittelt, wo die Fusion der Daten aller Sensoren erfolgt.

4.2.2 Fusion der Kameraperspektiven

Als Erstes werden Translation und Orientierung der Segmente $\boldsymbol{S}$ aller Kameraperspektiven vom Kamerakoordinatensystem in das Weltkoordinatensystem transformiert. Anschließend wird für jedes Segment $\boldsymbol{s} \in \boldsymbol{S}$ der nächste Nachbar $\boldsymbol{s}^*$ in der Menge aller bereits verarbeiteten Cluster $\boldsymbol{S}^*$ gesucht, wobei $\boldsymbol{S}^*$ zu Beginn jedes Frames leer ist. Befindet sich der geometrische Schwerpunkt von $\boldsymbol{s}$ in der Bounding-Box von $\boldsymbol{s}^*$, wird angenommen, dass die Segmente zu dem selben Objekt gehören. Die Bounding-Box wird dabei aus der statistischen Verteilung der Punktmessungen gebildet. Folglich werden Translation, Orientierung und statistische Verteilung von $\boldsymbol{s}^*$ aktualisiert, indem der Mittelwert gewichtet mit der Anzahl der jeweiligen Punktmessungen n berechnet wird. Somit ergibt sich für die Translation

$$\mathbf{c}_{s^*,1:m} = \frac{\mathbf{c}_{\boldsymbol{s},m} \cdot n_{\boldsymbol{s},m} + \mathbf{c}_{\boldsymbol{s}^*,1:m-1} \cdot n_{\boldsymbol{s}^*,1:m-1}}{n_{\boldsymbol{s},m} + n_{\boldsymbol{s}^*,1:m-1}}\,, \tag{4.1}$$

wobei m den Index der Kamera beschreibt. Die statistische Varianz wird analog berechnet. Für die Berechnung des Einheitsquaternions wird die sphärische lineare Interpolation genutzt. Zudem wird die Anzahl der Punktmessungen von $\boldsymbol{s}^*$ um die



(a) (b) (c) (d) (e)

Abbildung 4.2: Fusion der Daten von 4 Punktwolken aus verschiedenen Perspektiven in die Ellipsoid-Struktur: (a), (b), (c) und (d) zeigen die Punktwolkensegmente eines Stuhls aus 4 verschiedenen Blickwinkeln und die daraus berechnete statistische Verteilung dargestellt als Ellipsoid mit der dazugehörigen Pose. In (e) sieht man das auf dem Backend fusionierte Ellipsoid, die fusionierte Objektpose und die Punktwolkensegmente aus allen Perspektiven. Die Punktwolken werden nicht zum Backend übertragen und dienen lediglich der Visualisierung. Die Input-Daten stammen aus dem Behave-Datensatz [2].

Anzahl der Punktmessungen von $\boldsymbol{s}$ erhöht:

$$n_{s^*,1:m} = n_{\boldsymbol{s},m} + n_{\boldsymbol{s^*},1:m-1}\,. \tag{4.2}$$

Sollte kein korrespondierendes Cluster in $\boldsymbol{S^*}$ für $\boldsymbol{s}$ gefunden werden, wird ein neues Cluster durch Hinzufügen von $\boldsymbol{s}$ zur Menge $\boldsymbol{S^*}$ initialisiert. Abbildung 4.2 zeigt exemplarisch die auf den Sensorboards berechneten Ellipsoide vierer Kameraperspektiven und das daraus auf dem Backend fusionierte Ellipsoid.

4.2.3 Objekttracking

Im nächsten Schritt wird mit einem Constant Velocity Model eine Vorhersage über die aktuelle Position aller bereits bekannten und beobachteten Objekte $\boldsymbol{O}$ getroffen. Dazu wird der geometrische Schwerpunkt $\mathbf{c}_{\boldsymbol{o}}$ und die aktuelle Geschwindigkeit $\mathbf{v}$ aller Objekte $\boldsymbol{O}$ mit der vergangenen Zeit Δt kombiniert:

$$\mathbf{c}_{\text{pred}} = \mathbf{c}_{\boldsymbol{o}} + \mathbf{v} \cdot \Delta t\,. \tag{4.3}$$

Vergleichbar mit der Fusion der Kameraperspektiven wird nun für jedes fusionierte Segment $\boldsymbol{s^*}$ der nächste Nachbar $\boldsymbol{o} \in \boldsymbol{O}$ basierend auf der vorhergesagten Position $\mathbf{c}_{\text{pred}}$ gesucht. Zusätzlich wird ein prozentualer Schwellenwert $\tau_{\text{Punktanzahl}}$ definiert, der verhindern soll, dass eine Zuordnung bei deutlich kleinerer Punktwolkengröße stattfindet. Damit $\boldsymbol{o}$ als nächster Nachbar berücksichtigt wird, muss Gleichung (4.4) erfüllt sein, wobei $n^{\boldsymbol{s^*}}_{\text{Punkte}}$ die Anzahl der Punkte des fusionierten

Segmentes und $n^{\boldsymbol{o}}_{\text{Punkte}}$ die Anzahl der Punkte des Objektes im Mittel beschreibt. Hierdurch wird vermieden, dass weder eine fälschliche Zuordnung zwischen Objekten mit deutlichem Größenunterschied stattfindet, noch bei partieller Beobachtung eines stark verdeckten Objektes Translation und Orientierung aktualisiert werden, da der geometrische Schwerpunkt und die Orientierung der partiellen Punktwolke nicht die Translation und Orientierung des gesamten Objektes widerspiegeln.

$$n^{\boldsymbol{s}^*}_{\text{Punkte}} \geq \tau_{\text{Punktanzahl}} \cdot n^{\boldsymbol{o}}_{\text{Punkte}} \tag{4.4}$$

Zusätzlich wird überprüft, ob die Distanz d zwischen $\mathbf{c}_{\text{pred}}$ und dem geometrischen Schwerpunkt von $\boldsymbol{s}^*$ die maximale Tracking-Distanz τ_{Tracking} überschreitet. Wenn das nicht der Fall ist, werden alle Informationen von $\boldsymbol{s}^*$ in $\boldsymbol{o}$ integriert. Dazu werden die Translation $\mathbf{c}_{\boldsymbol{s}^*}$ und Orientierung $\mathbf{q}_{\boldsymbol{s}^*}$ von $\boldsymbol{s}^*$ übernommen und die aktuelle Geschwindigkeit $\mathbf{v}$ von $\boldsymbol{o}$ wird folgendermaßen berechnet:

$$\mathbf{v} = \frac{\mathbf{c}_{\boldsymbol{s}^*} - \mathbf{c}_{\boldsymbol{o}}}{\Delta t}\,, \tag{4.5}$$

wobei Δt die vergangene Zeit zwischen der letzten Beobachtung von $\boldsymbol{o}$ und der aktuellen ist. Die neue statistische Verteilung von $\boldsymbol{o}$ ergibt sich als Mittelwert aus der bisherigen statistischen Verteilung von $\boldsymbol{o}$ und der statistischen Verteilung von $\boldsymbol{s}^*$ unter Berücksichtigung der Anzahl bisheriger Observationen, analog zu Gleichung (4.1). Zuordnungen mit geringer Distanz d werden vorrangig behandelt. Sollte kein passendes Objekt $\boldsymbol{o}$ für $\boldsymbol{s}^*$ gefunden werden, dann wird für $\boldsymbol{s}^*$ in der Menge $\boldsymbol{O}$ eine neue Objektinstanz mit der Geschwindigkeit $\mathbf{v} = (0, 0, 0)^T$ initialisiert.

4.2.4 Clean-Up

Im Clean-Up Schritt wird für jedes Objekt $\boldsymbol{o} \in \boldsymbol{O}$ geprüft, ob ein weiteres Objekt $\boldsymbol{o'} \in \boldsymbol{O}$ existiert, in dessen Bounding-Box sich der geometrische Schwerpunkt von $\boldsymbol{o}$ befindet. Sollte das der Fall sein, wird das Objekt mit der geringeren Anzahl an Beobachtungen entfernt. Dadurch werden Objekte, die beispielsweise durch partielle Beobachtungen nicht der korrekten Objektinstanz zugeordnet und als neues Objekt registriert wurden (vgl. (4.4)), wieder entfernt. Außerdem werden Objekte, deren letzte Beobachtung zu lange zurückliegt, entfernt, um mit Objekten, die den Sichtbereich verlassen oder nicht verfolgt werden konnten, umgehen zu können.

(a) 1 (b) 2 (c) 3 (d) 4 (e) 5 (f) 6

(g) 7 (h) 8 (i) 9 (j) 10 (k) 11 (l) 12

Abbildung 4.3: Die im Trainingsdatensatz verwendeten Stuhl 3D-Modelle

4.3 Keypunkt-Skelette

Die Keypunkt-Skelette speichern, wie die Ellipsoide, die Objektpose bestehend aus Translation und Orientierung. Zusätzlich ist auf den Sensorboards und auf dem Backend die Objektgeometrie in Form von zuvor definierten Keypunkten bekannt, wodurch eine detailreichere Darstellung im Vergleich zu den Ellipsoiden stattfinden kann. Sollte neben den Keypunkten auch das Mesh des Objektes, zum Beispiel als Scan, bekannt sein, lässt sich der Detailgrad weiter steigern. In dieser Arbeit werden nur Keypunkt-Skelette für die Objektklasse *Stuhl* betrachtet. Die Methode kann jedoch entsprechend auf weitere Objektklassen erweitert werden.

4.3.1 Generieren von Trainingsdaten

Um Keypunkte auf Objekten zu erkennen, ist ein Detektor notwendig. Dieser Detektor muss mit entsprechenden Trainingsdaten, die Ground Truth Keypunkte für die Objekte enthalten, trainiert werden. In dieser Arbeit wurden synthetische Trainingsdaten unter Verwendung von *sl-cutscenes* [3], einer Erweiterung für *stilllleben* [23], generiert. Dabei handelt es sich um ein Framework, welches das Erstellen und Rendern realistischer indoor Szenen mit physikalisch interagierenden Objekten ermöglicht. Erstellt wurde ein Datensatz, bestehend aus 110 Trainings-Szenarien und 20 Test-Szenarien mit jeweils 120 Frames. In den Szenarien bewegen sich zwischen 3 und 6 Stühle mit einer zufälligen linearen Geschwindigkeit und

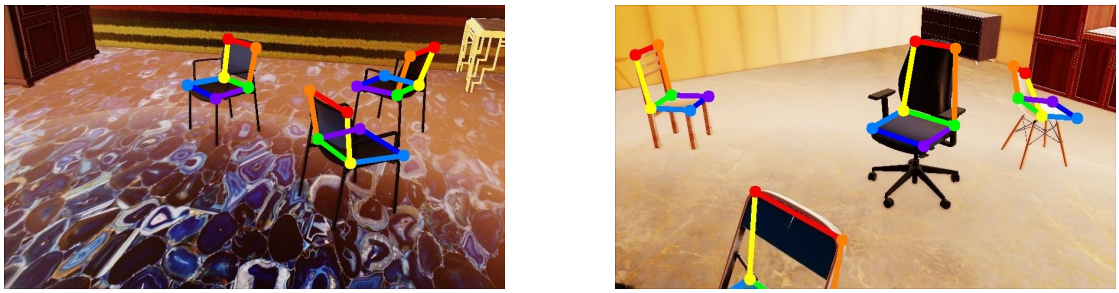


Abbildung 4.4: Einzelne Frames aus zwei verschiedenen Trainings-Szenarien und die dazugehörigen Ground-Truth Keypunkt-Annotationen

Rotationsgeschwindigkeit durch einen Raum mit fester Größe. Die Raumtexturen und Stuhl-Meshes werden dabei zufällig ausgewählt. Zur Auswahl stehen 10 Bodentexturen, 14 Wandtexturen und 12 Stühle. Für die Meshes wurde der Fokus auf Bürostühle gelegt, da diese Art von Stühlen hauptsächlich im Einsatzbereich des Smart Edge Sensor Netzwerk vorkommen (vgl. Abb. 4.3). Stuhl 1, 2 und 3 stammen aus externen Dateien die in *sl-cutscenes* [3] enthalten sind. Bei Stuhl 4 handelt es sich um einen 3D-Scan eines Bürostuhles aus unserem Labor. Die übrigen Meshes sind öffentlich verfügbar und stammen von 3D-Modell-Webseiten wie [7] und [24]. Zusätzlich wurden an zufälligen Positionen in Wandnähe Dekorationsobjekte platziert. Für die generierten Bilder wurde eine Auflösung von 848 $\times$ 480 gewählt. Die Kameraposition und Orientierung variiert nur leicht und soll einen ähnlichen Blickwinkel wie die Kameras der Smart Edge Sensoren aufweisen.

Um die für das Training notwendigen Ground-Truth Keypunkt-Annotationen zu generieren, wurde, wie in [32], ein modifiziertes BOP-Toolkit [12] verwendet. Zunächst wurden pro Objektmodell 6 Keypunkte in Objektkoordinaten manuell platziert. Davon befinden sich 4 Keypunkte auf den Eckpunkten der Sitzfläche und 2 Keypunkte auf der Oberkante der Lehne. Anschließend konnten die Keypunkte mit der, in den generierten Daten enthaltenen, Ground-Truth Objekttranslation und Rotation über die Gleichung der Lochkamera (vgl. (2.11)) in das Kamerabild projiziert werden. Zwei beispielhafte Szenen mit annotierten Keypunkten sind in Abbildung 4.4 dargestellt. Die Speicherung der Annotationen erfolgt im COCO-Format (Common Objects in Context) [16].

4.3.2 Netzwerk

Zur Detektion der Keypunkte wird der gleiche top-down-Ansatz verwendet, der in [4] zur Bestimmung menschlicher Posen dient. Da die erstellten synthetischen Trainingsdaten kein Bildrauschen enthalten, wurden während des Trainings ver-

schiedene Augmentationen genutzt, um Rauschen nachträglich hinzuzufügen. Die Detektion der Keypunkte erfolgt auf den RGB Bildern der 4 RGB-D Kameras und findet auf den Sensorboards statt.

4.3.3 Eingangsdaten

Der geometrischen Schwerpunkt, die statistischen Verteilung und die Anzahl der Punkte eines jeden Punktwolkensegmentes werden, wie für die Ellipsoide (vgl. Abs. 4.2.1), auf den Sensorboards berechnet und an das Backend übermittelt. Zusätzlich wird mittgeteilt, ob eine gültige Keypunkt-Detektion für das Segment getätigt wurde, welche Rotation und Translation aus den Keypunkten geschätzt wurden und die mittlere Distanz zwischen den Keypunkten und dem nächstgelegenen Punkt im Punktwolkensegment. Dazu wird zunächst für alle detektierten Keypunkt-Skelette mit dem PnP-RANSAC Algorithmus, wie in [32], die Translation und Rotation des Objektmodells bestimmt. Dazu ist es notwendig, die Ground-Truth Keypunkte im Objektkoordinatensystem zu kennen, um sie den detektierten Keypunkten im Kamerabild zuordnen zu können. Anschließend ergeben sich die 3D-Keypunkte in Kamerakoordinaten durch Transformation der Ground-Truth Keypunkte mit der berechneten Objektpose (vgl. Abb. 4.5). Nun wird versucht die 3D-Keypunkt-Skelette $\boldsymbol{K}$ den zugehörigen Punktwolkensegmenten $\boldsymbol{S}$ zuzuordnen. Dies wird realisiert, indem durch alle Segmente iteriert wird und für jeden Keypunkt der nächste Nachbar in dem jeweiligen Punktwolkensegment gesucht wird.

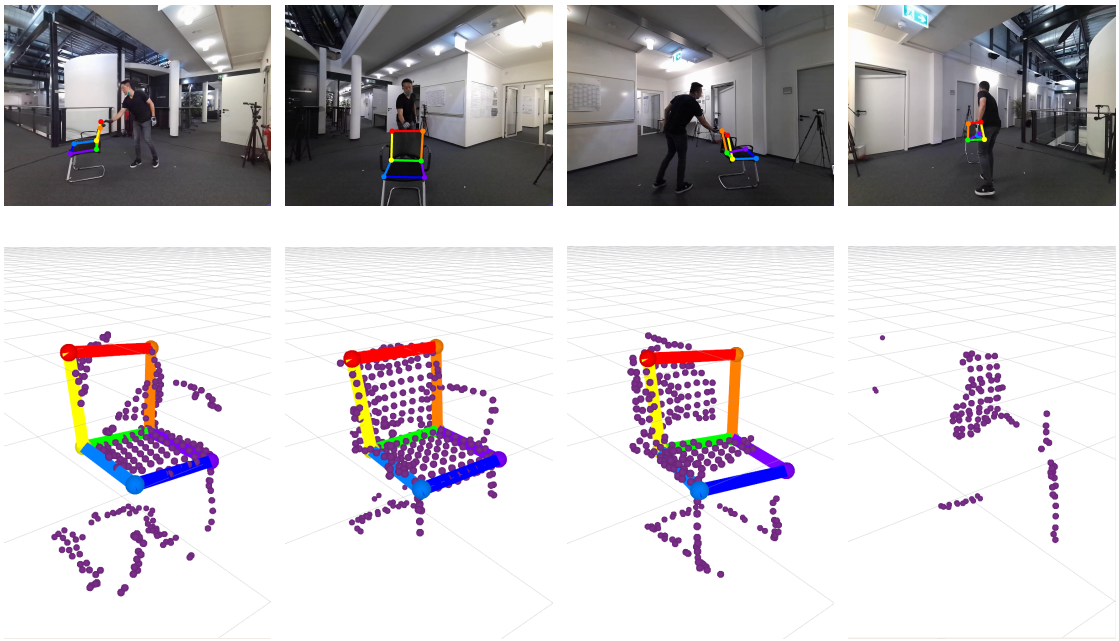
Sei $\mathbf{x}_{i,\boldsymbol{k}}$ der i-te Keypunkt von Skelett $\boldsymbol{k}$, $\mathbf{y}_i$ der nächste Nachbar von $\mathbf{x}_{i,\boldsymbol{k}}$ in Segment $\boldsymbol{s}$, n die Anzahl der Keypunkte und $\|\mathbf{x}-\mathbf{y}\|_2$ die euklidische Distanz zwischen den Punkten $\mathbf{x}$ und $\mathbf{y}$, dann bezeichnet $\boldsymbol{k}_{\min}$ das Keypunkt-Skelett mit der niedrigsten mittleren Distanz zu Segment $\boldsymbol{s}$:

$$\boldsymbol{k}_{\min} = \min_{\forall \boldsymbol{k} \in \boldsymbol{K}} \left(\frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_{i,\boldsymbol{k}} - \mathbf{y}_i\|_2 \right). \tag{4.6}$$

Der Schwellenwert τ_{Distanz} beschreibt die maximal erlaubte mittlere Distanz zwischen den Keypunkten und ihren nächsten Nachbarn:

$$\tau_{\text{Distanz}} \geq \frac{1}{n} \sum_{i=1}^{n} \|\mathbf{x}_{i,\boldsymbol{k}_{\min}} - \mathbf{y}_i\|_2 \,. \tag{4.7}$$

Wenn Gleichungen (4.6) und (4.7) erfüllt sind, dann wird Keypunkt-Skelett $\boldsymbol{k}_{\min}$ Segment $\boldsymbol{s}$ zugeordnet. Sollten für ein Punktwolkensegment kein gültiges Keypunkt-Skelett existieren, dann werden die Objektinformationen, wie in 4.2.1 beschrieben, ohne Keypunktdaten an das Backend übermittelt, um das Objekttracking auch



(a) (b) (c) (d)

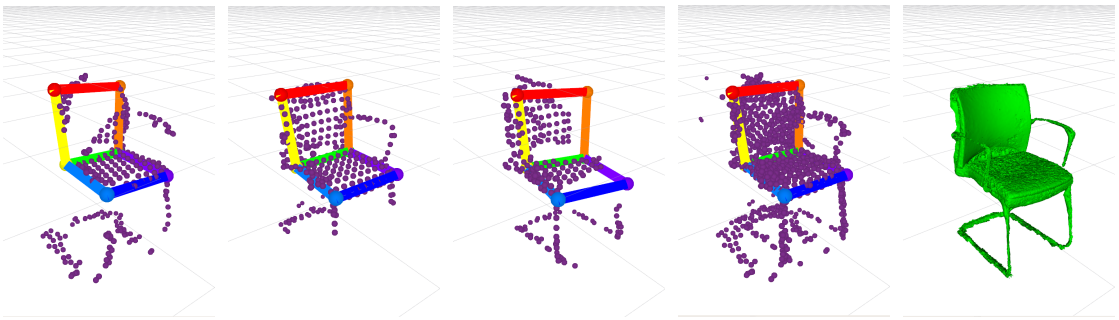
Abbildung 4.5: 2D-Keypunkt Detektionen und die mit PnP-RANSAC berechneten 3D-Posen: Die oberen Bilder zeigen die detektierten Keypunkte für einen Frame des Behave-Datensatzes [2] aus den Perspektiven (a), (b), (c) und (d). Darunter sind die Ergebnisse des PnP-RANSAC Algorithmus und die zugeordneten Punktwolkensegmente zu sehen. Für die 2D-Keypunkte des oberen Bildes von (d) konnte der Algorithmus keine Pose finden. Daher können dem Punktwolkensegment keine Keypunktdaten zugeordnet werden. Die Punktwolkensegmente werden nicht zum Backend übertragen und verbleiben auf den Sensorboards.

mit fehlgeschlagener Keypunktdetektion gewährleisten zu können.

4.3.4 Fusion der Kameraperspektiven

Auf dem Backend erfolgt die Fusion der Kameraperspektiven unter Verwendung von Translation und statistischer Verteilung der Segmente, wie bei den Ellipsoiden (vgl. Abs. 4.2.2). Darüber hinaus werden auch Keypunkt-spezifische Daten des Segmentes $\boldsymbol{s}$ in das zugehörige fusionierte Segment $\boldsymbol{s}^*$ integriert. Die mittlere Distanz d zwischen den Keypunkten und dem nächsten Nachbar in der Punktwolke wird als inverses Gewicht w verwendet:

$$w = \frac{1}{1+d}\,. \tag{4.8}$$



(a) (b) (c) (d) (e)

Abbildung 4.6: Fusion von Keypunkt-Skeletten aus drei verschiedenen Perspektiven: (d) zeigt das auf dem Backend fusionierte Keypunkt-Skelett der Skelette aus (a), (b) und (c). Die 2D-Keypunkte der einzelnen Skelette sind in Abb. 4.5 dargestellt. Die Punktwolken werden nicht an das Backend übermittelt und dienen lediglich der Visualisierung. In (e) sieht man das Ground-Truth Mesh transformiert mit der fusionierten Pose.

Gleichungen (4.9) und (4.10) beschreiben die Berechnung der fusionierten Translation $\mathbf{t}_{s^*,1:m}$ und des Gewichtes $w_{s^*,1:m}$ mit Kameraindex m.

$$\mathbf{t}_{s^*,1:m} = \frac{\mathbf{t}_{s,m} \cdot w_{s,m} + \mathbf{t}_{s^*,1:m-1} \cdot w_{s^*,1:m-1}}{w_{s,m} + w_{s^*,1:m-1}} \tag{4.9}$$

$$w_{s^*,1:m} = w_{s,m} + w_{s^*,1:m-1} \tag{4.10}$$

Die Orientierung als Einheitsquaternion wird mit der sphärischen linearen Interpolation mit dem Gewicht w berechnet. Abbildung 4.6 zeigt die Fusion der Posendaten aus drei Perspektiven. Das Resultat kann entweder als Keypunkt-Skelett oder, wenn vorhanden, als Mesh visualisiert werden.

4.3.5 Objekttracking und Clean-Up

Das Objekttracking, die Datenassoziation und der Clean-Up Schritt funktionieren wie bei den Ellipsoiden (vgl. Abs. 4.2.3, 4.2.4) mit der Ausnahme, dass die Objektrotation und Objekttranslation durch die Keypunkt-Daten beschrieben werden. Die Visualisierung erfolgt wahlweise durch die 3D-Keypunkte oder durch das 3D-Modell, welches für die manuelle Definition der Keypunkte in Objektkoordinaten benutzt wurde, transformiert mit der entsprechenden fusionierten Objektpose.

4.4 Subkarten

Die Subkartenstruktur speichert die Objektpose und die Position aller belegten Voxel in Objektkoordinaten. Die Auflösung der Subkarte kann dabei unabhängig von der globalen allozentrischen Karte gewählt werden. Außerdem hat diese Darstellung den Vorteil, dass die geometrischen Eigenschaften des Objektes gespeichert werden können, ohne dass zuvor ein Objektmodell bekannt sein muss. Verglichen mit Ellipsoid oder Keypunkt-Skelett ist jedoch eine deutlich höhere Datenübertragung zwischen Sensorboards und Backend notwendig. Die Struktur der Subkarten orientiert sich an der globalen allozentrischen Karte. Dementsprechend wird das in [4] implementierte Sparse-Voxel-Grid basierend auf Hash-maps verwendet.

4.4.1 Eingangsdaten

Für das Erstellen der Subkarten muss auf den Sensorboards keine Berechnung von Punktwolkeneigenschaften durchgeführt werden, da die gesamte semantisch und geometrisch segmentierte Punktwolke von den Sensorboards auf das Backend übertragen wird. Um die Keypunkt-Posenbestimmung auch für die Subkarten nutzen zu können, werden, wie in Absatz 4.3.3 beschrieben, Keypunkt-Skelette den Punktwolkensegmenten zugeordnet. Die Keypunkt-spezifischen Daten werden ebenfalls übermittelt.

4.4.2 Fusion der Kameraperspektiven

Zur Fusion der Kameraperspektiven werden zunächst alle Punktwolkensegmente $\boldsymbol{S}$ von Kamerakoordinaten in Weltkoordinaten transformiert. Anschließend wird, ähnlich wie bei Ellipsoid (vgl. Abs. 4.2.2) und Keypunkt-Skelett (vgl. Abs. 4.3.4), für jedes Segment $\boldsymbol{s} \in \boldsymbol{S}$ geprüft, ob es in der Menge der bereits verarbeiteten fusionierten Segmente $\boldsymbol{S}^*$ ein Cluster gibt, das zum gleichen Objekt gehört. Dazu wird der geometrische Schwerpunkt von $\boldsymbol{s}$ berechnet und es wird der nächste Nachbar $\boldsymbol{s}^* \in \boldsymbol{S}^*$ bestimmt. Nun wird geprüft wieviele Punkte von $\boldsymbol{s}$ sich in der Bounding-Box von $\boldsymbol{s}^*$ befinden. Überschreitet die Anzahl der Punkte innerhalb der Bounding-Box n_{innen} im Verhältnis zu der Gesamtanzahl der Segmentpunkte n_{total} einen festgelegten prozentualen Schwellenwert τ_{fusion}, dann wird angenommen, dass es sich um das gleiche Objekt handelt:

$$n_{\text{innen}} \geq \tau_{\text{fusion}} \cdot n_{\text{total}} \,. \tag{4.11}$$

Folglich werden die Punktwolken von $\boldsymbol{s}$ und $\boldsymbol{s}^*$ konkateniert und die Punktwolkeneigenschaften, wie geometrischer Schwerpunkt und Orientierung, werden neu

berechnet (vgl. Abs. 4.2.1). Die Keypunktdaten werden, wie in Absatz 4.3.4 beschrieben, fusioniert. Sollte kein zugehöriges Cluster in $\boldsymbol{S}^*$ für $\boldsymbol{s}$ gefunden werden, dann werden die Punktwolkeneigenschaften von $\boldsymbol{s}$ bestimmt und Segment $\boldsymbol{s}$ wird in $\boldsymbol{S}^*$ eingefügt und damit als neues Cluster initalisiert.

4.4.3 Objekttracking und Clean-Up

Das Finden einer zugehörigen Objektinstanz $\boldsymbol{o}$ für ein fusioniertes Segment $\boldsymbol{s}^*$ funktioniert wie in Absatz 4.2.3 erklärt. Objekttranslation und Objektrotation werden den Daten von $\boldsymbol{s}^*$ entnommen. Die Translation wird dabei immer durch den geometrischen Schwerpunkt beschrieben. Sofern gültige Keypunktdaten verfügbar sind, wird die Orientierung des Objektes durch diese beschrieben. Andernfalls wird die Orientierung des Punktwolkensegmentes, welche mit einer Hauptkomponentenanalyse berechnet wurde, als neue Objektorientierung verwendet. Zusätzlich wird eine Anpassung der Subkarte, transformiert mit der neuen Objektpose, an das Punktwolkensegment mit dem Iterative-Closest-Point Algorithmus durchgeführt. Die neue Objektpose wird mit der via ICP erhaltenen Translation und Rotation angepasst. Die Segmentpunktwolke von $\boldsymbol{s}^*$ wird unter Verwendung der neuen Objektpose in das Objektkoordinatensystem transformiert. Danach werden die einzelnen Punkte in das Subkartenvolumen von $\boldsymbol{o}$ eingetragen. Da es sich bei der verwendeten Subkartenstruktur um ein Sparse-Voxel-Grid [4] handelt, wird nur für Bereiche, in denen eine Messung getätigt wurde, Speicher allokiert und nicht für das gesamte Volumen. Darüber hinaus werden nicht alle Punktmessungen gespeichert. Stattdessen wird pro Voxel nur ein Punkt gespeichert, der die durchschnittliche Lage aller Puntkmessungen, die in diesem Voxel getätigt wurden, beschreibt. Zusätzlich wird pro Voxel m_i die Anzahl der Punktmessungen n_{m_i} gespeichert. Damit ein Voxel als belegt angenommen wird, muss die Anzahl der Punktmessungen n_{m_i} in einem Voxel einen gewissen Prozentsatz τ_{obs} aller Beobachtungen des Objektes n_{obs} erreichen. Der Zustand des Voxels m_i kann dann wie folgt beschrieben werden:

$$p(m_i) = \begin{cases} 1 & \text{für } n_{m_i} \geq \tau_{\text{obs}} \cdot n_{\text{obs}} \\ 0 & \text{für } n_{m_i} < \tau_{\text{obs}} \cdot n_{\text{obs}} \end{cases} . \tag{4.12}$$

Eine fusionierte Punktwolke und das Ergebnis nach dem Eintragen der Punktwolke in die entsprechende Subkarte ist in Abbildung 4.7 beispielhaft dargestellt. Sollte noch kein passendes Objekt $\boldsymbol{o}$ für $\boldsymbol{s}^*$ existieren, dann wird eine neue Objektinstanz erstellt und die Punkte werden in eine leere Subkarte eingetragen.

Der folgende Clean-Up Schritt unterscheidet sich nur geringfügig von dem Clean-Up Schritt der Ellipsoide und der Keypunkt-Skelette (vgl. Abs. 4.2.4). Anstatt nur zu prüfen, ob sich der geometrische Schwerpunkt eines Objektes in der Bounding-Box eines anderen Objektes befindet, wird die Anzahl der Punkte innerhalb der Bounding-Box eines anderen Objektes bestimmt. Auch hier wird das Objekt mit der geringeren Anzahl von Beobachtungen entfernt, sofern die Anzahl der Punkte einen prozentualen Schwellenwert überschreitet (vgl. (4.11)). Visualisiert werden die Subkarten in der globalen allozentrischen Karte, indem die Subkarten mit der entsprechenden Objektpose transformiert werden.

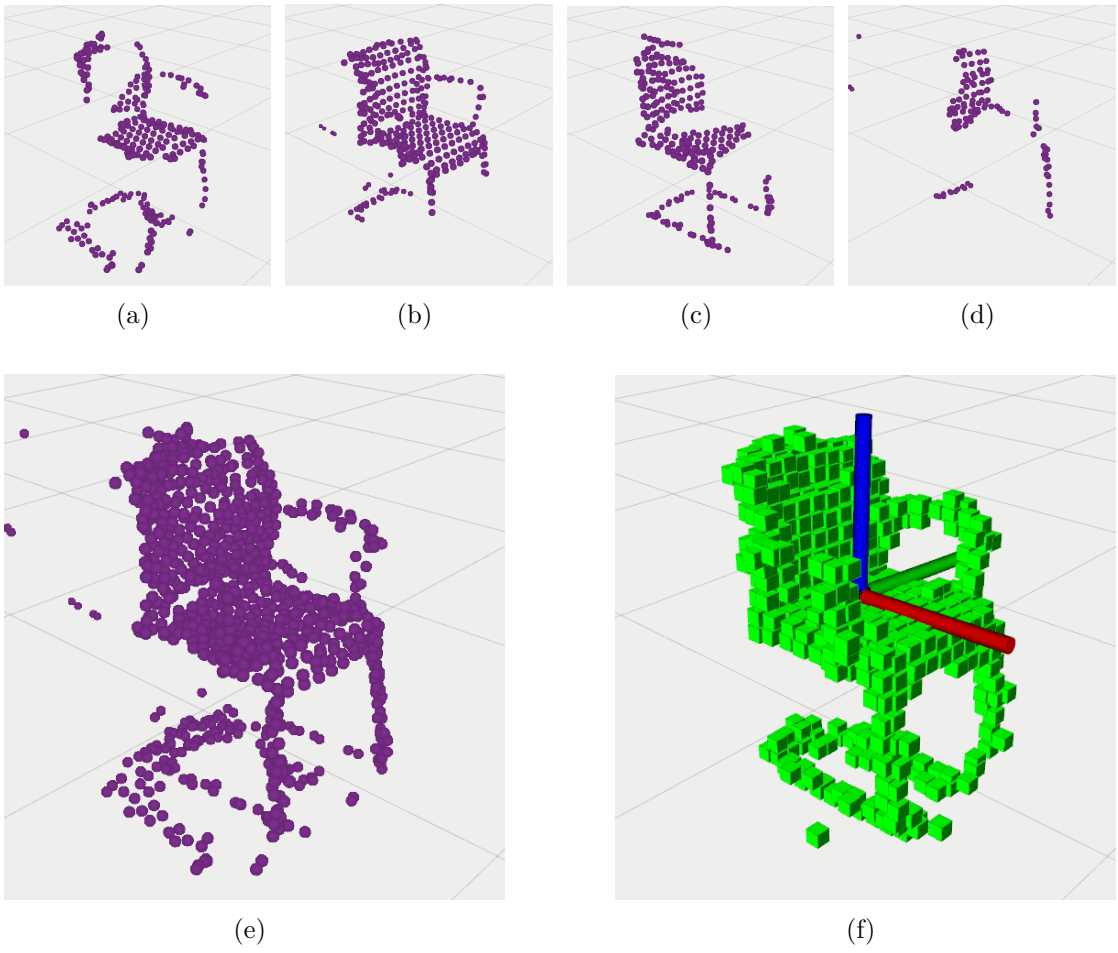


Abbildung 4.7: Fusion von 4 Punktwolken und Eintragen in die Subkarte: Die Punktwolken (a), (b), (c) und (d) aus vier verschiedenen Perspektiven werden an das Backend übermittelt, wo sie zu (e) konkateniert werden. In (f) sieht man die Subkarte und Objektpose der zugeordneten Objektinstanz, nachdem die Punktwolke in das Subkartenvolumen eingetragen wurde. Die Voxelgröße der Subkarte beträgt 5 cm.

5 Evaluation

In diesem Kapitel wird die vorgestellte Methode evaluiert. Dazu werden in Abschnitt 5.1 die verwendeten Metriken vorgestellt. Danach werden in Abschnitt 5.2 die Ergebnisse der Evaluation auf dem Behave-Datensatz [2] präsentiert, gefolgt von den Ergebnissen der Experimente mit dem Sensor Netzwerk in Abschnitt 5.3.

5.1 Metriken

Die in dieser Arbeit verwendet Metriken sind Intersection-over-Union, ein Translationsfehler und ein Orientierungsfehler. Die genaue Definition der einzelnen Metriken folgt in den kommenden Abschnitten.

5.1.1 Intersection-over-Union

Intersection-over-Union (IoU) ist eine Metrik, welche die Genauigkeit von Bounding-Boxen oder Masken beschreiben kann. Dazu wird die Region, in der sich vorhergesagte Maske und Ground-Truth Maske überschneiden, durch die Vereinigung beider Masken geteilt. Sei $\mathbf{M}_{\text{pred}}$ eine vorhergesagte binäre Maske und $\mathbf{M}_{\text{gt}}$ eine binäre Ground-Truth Maske. Wenn $\mathbf{M}(x, y)$ der Wert der Maske an den Pixelkoordinaten x und y ist, dann gilt für die Schnittmenge

$$\mathbf{M}_I(x, y) = \mathbf{M}_{\text{pred}}(x, y) \wedge \mathbf{M}_{\text{gt}}(x, y) \tag{5.1}$$

und für die Vereinigung

$$\mathbf{M}_U(x, y) = \mathbf{M}_{\text{pred}}(x, y) \vee \mathbf{M}_{\text{gt}}(x, y)\,. \tag{5.2}$$

Der IoU-Wert wird dann mit der Anzahl $n(\mathbf{M})$ der Pixel, für die gilt $\mathbf{M}(x, y) = 1$, berechnet:

$$E_{\text{IoU}} = \frac{n(\mathbf{M}_I)}{n(\mathbf{M}_U)}\,. \tag{5.3}$$

Das Ergebnis liegt im Interval $[0, 1]$. Höhere Werte entsprechen einer höheren Genauigkeit.

5.1.2 Translationsfehler

Für die Auswertung der Translation der Pose verwenden wir die euklidische Distanz. Sei $\mathbf{t}_{\text{pred}}$ die vorhergesagte Position und $\mathbf{t}_{\text{gt}}$ die Ground-Truth Position in Weltkoordinaten. Dann gilt für den Translationsfehler

$$E_{\text{trans}} = \|\mathbf{t}_{\text{pred}} - \mathbf{t}_{\text{gt}}\|_2 = \sqrt{(x_{\text{pred}} - x_{\text{gt}})^2 + (y_{\text{pred}} - y_{\text{gt}})^2 + (z_{\text{pred}} - z_{\text{gt}})^2} \quad (5.4)$$

mit

$$t_{\text{i}} = (x_i, y_i, z_i)^T . \quad (5.5)$$

Im folgenden wird der Translationsfehler einer Posenschätzung in cm angegeben.

5.1.3 Orientierungsfehler

Die Orientierung wird in dieser Arbeit als Einheitsquaternion angegeben. Dementsprechend muss für die Auswertung auch eine Quaternion-spezifische Metrik verwendet werden. Basierend auf [15] wird der Fehler zwischen der geschätzten Orientierung $\mathbf{q}_{\text{pred}}$ und der Ground-Truth Orientierung $\mathbf{q}_{\text{gt}}$ folgendermaßen bestimmt:

$$E_{\text{rot}} = \arccos(2\langle \mathbf{q}_{\text{pred}}, \mathbf{q}_{\text{gt}} \rangle^2 - 1) , \quad (5.6)$$

wobei $\langle \mathbf{q}_{\text{pred}}, \mathbf{q}_{\text{gt}} \rangle$ das innere Produkt beider Quaternionen ist:

$$\langle \mathbf{q}_{\text{pred}}, \mathbf{q}_{\text{gt}} \rangle = x_{\text{pred}} x_{\text{gt}} + y_{\text{pred}} y_{\text{gt}} + z_{\text{pred}} z_{\text{gt}} + w_{\text{pred}} w_{\text{gt}} \quad (5.7)$$

mit

$$\mathbf{q} = xi + yj + zk + w . \quad (5.8)$$

Das Resultat beschreibt den Winkel zwischen den beiden Rotationen und liegt im Interval $[0, \pi]$.

5.2 Auswertung auf dem Behave-Datensatz

Zur Auswertung werden vier verschiedene Szenarien des Behave-Datensatzes [2] verwendet: *Date01_Sub01_chairblack_hand*, *Date02_Sub02_chairblack_sit*, *Date03_Sub03_chairwood_hand*, *Date03_Sub04_chairwood_sit*. Da die Keypunkt-Detektion in dieser Arbeit ausschließlich für die Klasse Stuhl realisiert wurde, beschränken wir uns auf Szenarien, in denen Stühle genutzt werden. In den vier Szenarien kommen zwei verschiedene Stuhlmodelle (*chairblack*, *chairwood*) zum Einsatz (vgl. Abb. 5.1). Für beide Stuhlmodelle liegt dem Datensatz ein 3D-Scan

(a) *chairblack*

(b) *chairwood*

Abbildung 5.1: Die im Behave-Datensatz verwendeten Stühle

bei. Die Meshes der Scans werden für die Visualisierung genutzt. Darüber hinaus werden auf den Modellen jeweils manuell Keypunkte definiert, die als Eingabe für den PnP-Algorithmus dienen. Es gibt jeweils ein Szenario, in dem das Objekt durch den Raum bewegt wird (*hand*), und ein Szenario, in dem eine Person auf dem Stuhl sitzt und somit das Objekt größtenteils verdeckt (*sit*).

5.2.1 Posenevaluation

Für die Evaluation der Pose ist es zunächst erforderlich, die Ground-Truth Orientierungen der Objekte des Behave-Datensatzes mit einer konstanten Transformation zu transformieren, da die Orientierung der Achsen nicht der in dieser Arbeit gewählten Konvention entspricht (x-Achse: nach vorne, z-Achse: aufwärts). Dazu wird die Transformation manuell bestimmt und auf alle Ground-Truth Orientierungen angewendet. Zudem ist es notwendig, die konstante Verschiebung zwischen der Translation von Ellipsoid und Subkarte und der Ground-Truth Translation zu ermitteln, da diese auf unterschiedliche Arten und Weisen definiert sind. Dazu wird die geschätzte Objekttranslation in das Ground-Truth Objektkoordinatensystem transformiert und die Verschiebung zwischen den Translationen über alle Frames mit gültigen Observationen gemittelt. Die ermittelte konstante Verschiebung wird bei dem Vergleich mit der Ground-Truth Pose berücksichtigt. Eine Anpassung der Translation der Keypunkt-Skelette ist nicht notwendig, da die Keypunkte auf dem Ground-Truth Mesh definiert wurden, wodurch der Ursprung der Keypunkte mit dem Ursprung der Ground-Truth Objektkoordinaten übereinstimmt. Anschließend wird pro Frame der Translationsfehler E_{trans} und der Orientierungsfehler E_{rot} zwi-

Tabelle 5.1: Einfluss der Keypunkt-basierten Posenschätzung auf die Ellipsoide

Szenario	$\overline{E_{\text{rot}}}$ ohne Keypunktdaten (°)	$\overline{E_{\text{rot}}}$ mit Keypunktdaten (°)
chairblack_hand	35.69	**6.16**
chairblack_sit	28.72	**14.64**
chairwood_hand	18.76	**12.73**
chairwood_sit	23.35	**21.48**

Tabelle 5.2: Einfluss der Keypunkt-basierten Posenschätzung auf die Subkarten

Szenario	$\overline{E_{\text{rot}}}$ ohne Keypunktdaten (°)	$\overline{E_{\text{rot}}}$ mit Keypunktdaten (°)
chairblack_hand	10.88	**4.60**
chairblack_sit	9.29	**9.28**
chairwood_hand	9.13	**6.83**
chairwood_sit	9.01	**8.84**

schen Ground-Truth Pose und geschätzter Pose für alle Objektrepräsentationen bestimmt. Daraus wird wiederum der mittlere Fehler $\overline{E}$ und die Standardabweichung σ berechnet. Tabelle 5.1 und Tabelle 5.2 zeigen die Evaluation des Rotationsfehlers mit und ohne Verwendung der Keypunktdaten für die Ellipsoid, bzw. Subkartenrepräsenation für die vier Szenarien aus dem Behave Datensatz unter Verwendung der Ground-Truth Segmentierung. Man erkennt, dass mit der Keypunkt-basierten Orientierungsschätzung in allen Szenarien ein niedrigerer Orientierungsfehler erzeugt wird. Daher wird im Folgenden für Ellipsoide und Subkarten die Keypunkt-basierte Orientierungsschätzung genutzt, sofern verfügbar. Um zu Überprüfen wie sehr die Methode von der Qualität der Segmentierung und Detektion abhängt, werden für die Berechnung der Fehler zum einen die Ground-Truth Segmentierungen des Datensatzes genutzt und zum anderen die Segmentierungen und Bounding-Box Detektionen des CNN aus [4].

Ground-Truth Segmentierung

Mit den pseudo Ground-Truth Masken und den Tiefenbildern des Datensatzes werden Punktwolken gebildet, die als Eingabe für die vorgestellte Methode dienen. Zusätzlich wurden aus den Masken Ground-Truth Bounding-Boxen berechnet, in denen die Keypunkte detektiert werden. Die Tabellen 5.3, 5.4, 5.5 und 5.6 zeigen die Ergebnisse der Posenevaluation für die vier gewählten Szenarien unter Verwendung der Ground-Truth Segmentierung.

Tabelle 5.3: Posendaten für *chairblack_hand* mit Ground-Truth Segmentierung

	Translationsfehler (cm)		**Orientierungssfehler** (°)	
Typ	$\overline{E_{\text{trans}}}$	σ	$\overline{E_{\text{rot}}}$	σ
Ellipsoid	**3.73**	0.21	6.16	1.02
Skelett	5.13	0.36	**4.57**	0.50
Subkarte	**3.73**	0.21	4.60	0.37

Tabelle 5.4: Posendaten für *chairblack_sit* mit Ground-Truth Segmentierung

	Translationsfehler (cm)		**Orientierungssfehler** (°)	
Typ	$\overline{E_{\text{trans}}}$	σ	$\overline{E_{\text{rot}}}$	σ
Ellipsoid	**4.83**	0.38	14.64	1.72
Skelett	5.83	0.42	**7.28**	0.57
Subkarte	5.01	0.36	9.28	0.97

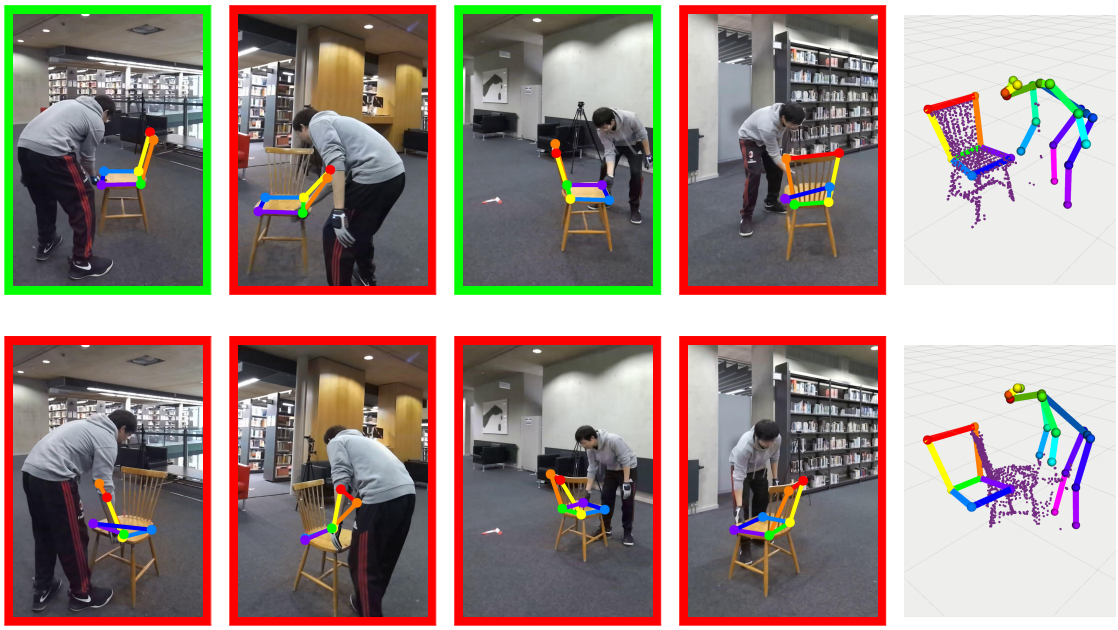
Tabelle 5.5: Posendaten für *chairwood_hand* mit Ground-Truth Segmentierung

	Translationsfehler (cm)		**Orientierungssfehler** (°)	
Typ	$\overline{E_{\text{trans}}}$	σ	$\overline{E_{\text{rot}}}$	σ
Ellipsoid	**3.41**	0.10	12.73	3.25
Skelett	12.18	0.96	16.68	2.62
Subkarte	3.43	0.10	**6.83**	0.95

Tabelle 5.6: Posendaten für *chairwood_sit* mit Ground-Truth Segmentierung

	Translationsfehler (cm)		**Orientierungssfehler** (°)	
Typ	$\overline{E_{\text{trans}}}$	σ	$\overline{E_{\text{rot}}}$	σ
Ellipsoid	**2.64**	0.12	21.48	4.06
Skelett	6.58	0.29	**4.81**	0.36
Subkarte	2.65	0.13	8.84	0.53

Betrachtet man den Translationsfehler in den vier Szenarien, wird deutlich, dass die Translation der Keypunkt-basierten Posenschätzung eine höhere Ungenauigkeit aufweist als die Schätzung der Translation über den geometrischen Schwerpunkt des Punktwolkensegments, wie sie für Ellipsoid und Subkarte genutzt wird. Das liegt insbesondere daran, dass es Frames gibt, in denen keine gültige Keypunkt-basierte Posenschätzung existiert, das Objekt jedoch verschoben wird. Durch die Posenschätzungen über die Hauptkomponentenanalyse werden die Posen von Ellipsoid und Subkarte in diesen Fällen trotzdem aktualisiert. Der Translationsfehler von Ellipsoid und Subkarte unterscheidet sich nur geringfügig. Die unterschiedlichen Orientierungsfehler kommen ebenfalls zustande, da nicht in allen Frames eine gültige Keypunkt-Detektion gemacht wird. Im Falle des Keypunkt-Skeletts geschieht dann keine Aktualisierung, während bei Ellipsoid und Subkarte die über die Hauptkomponentenanalyse erhaltene Pose genutzt wird, welche in der Regel ungenauer ist (vgl. Tab. 5.1, 5.2). Alle Szenarien zeigen, dass die Orientierung der Subkarte genauer ist als die der Ellipsoide, weil die Hauptkomponentenanalyse der konkatenierten Punktwolke eine bessere Posenschätzung liefert als die gemittelte Posenschätzung der Hauptkomponentenalayse der Punktwolken aus den einzelnen Perspektiven. In den Szenarien *chairblack_hand* (vgl. Tab. 5.3), *chairblack_sit* (vgl. Tab. 5.4) und *chairwood_sit* (vgl. Tab. 5.6) liefert die Skelett-Repräsentation die beste Orientierung. Dies entsteht durch Situationen, in denen durch ungültige oder ausbleibende Keypunkt-Detektion die letzte verfügbare Keypunkt-basierte Orientierung bessere Werte liefert als die aktuelle Orientierungsschätzung durch die Hauptkomponentenanalyse. Insbesondere in den Szenarien mit Verdeckung und fehlenden Keypunkt-Detektionen können häufig nicht alle verdeckten Punktwolkensegmente über die Anzahl der Punkte (vgl. (4.4)) herausgefiltert werden und verfälschen die Orientierungsschätzung. Diese Beobachtung legt nahe, sich gänzlich auf die Orientierungsschätzung der Keypunkte zu verlassen. Die Ergebnisse des Szenarios *chairwood_hand* (vgl. Tab. 5.5) zeigen jedoch, dass es vorkommt, dass bei Orientierungsänderungen die Detektionen der Keypunkte fehlschlagen (vgl. Abb. 5.2). Hier ist das Nutzen der Pose der Hauptkomponentenanalyse von Vorteil. Für den PnP-RANSAC Algorithmus wurden strenge Grenzwerte gewählt. Dadurch ist sichergestellt, dass gültige Keypunkt-Posenschätzungen eine hohe Genauigkeit haben und eine Verbesserung gegenüber der Posenschätzung der Hauptkomponentenanalyse bieten (vgl. Tab. 5.1, 5.2) und auch für Ellipsoide und Subkarten genutzt werden können. Ein Nachteil ist jedoch, dass für eine gültige Pose sehr akkurate Detektionen erfolgen müssen (vgl. Abb. 5.2).



(a) (b) (c) (d) (e)

Abbildung 5.2: Die 2D-Keypunkt-Detektionen und 3D-Skelette zweier aufeinanderfolgender Frames: (a), (b), (c) und (d) zeigen die 2D-Keypunkt-Detektionen aus den vier Kameraperspektiven. In (e) ist das resultierende Keypunkt-Skelett und die von den Sensoren gemessene Punktwolke visualisiert. Gültige Keypunkt-Detektionen sind grün umrandet und ungültige rot. Während im ersten Frame in zwei von vier Bildern eine gute Detektion getätigt wird, kann im folgenden Frame keine ausreichende Detektion erfolgen. Dadurch kann für den ersten Frame eine 3D-Pose berechnet und die Objektbewegung korrekt registriert werden. Im zweiten Frame kann basierend auf den Keypunkt-Detektionen keine Objektpose bestimmt werden, wodurch das Skelett nicht aktualisiert wird.

Online CNN Segmentierung

Um zu testen, wie die Methode mit Daten mit höherem Rauschen umgeht, wurde die Segmentierung und Bounding-Box Detektion aus [4] genutzt, um die Punktwolken und Keypunkt-Detektionen aus den RGB-Bildern und Tiefenbildern des Datensatzen zu erhalten. Die Tabellen 5.7, 5.8, 5.9 und 5.10 zeigen die Ergebnisse der Posenevaluation für die vier gewählten Szenarien unter Verwendung der Online CNN Segmentierung und Detektion.

Tabelle 5.7: Posendaten für *chairblack_hand* mit Online CNN Segmentierung

	Translationsfehler (cm)		**Orientierungssfehler** (°)	
Typ	$\overline{E_{\text{trans}}}$	σ	$\overline{E_{\text{rot}}}$	σ
Ellipsoid	7.30	0.43	8.98	3.10
Skelett	**7.19**	0.76	6.69	1.13
Subkarte	7.62	0.38	**4.81**	0.39

Tabelle 5.8: Posendaten für *chairblack_sit* mit Online CNN Segmentierung

	Translationsfehler (cm)		**Orientierungssfehler** (°)	
Typ	$\overline{E_{\text{trans}}}$	σ	$\overline{E_{\text{rot}}}$	σ
Ellipsoid	7.07	0.32	59.79	8.35
Skelett	**5.85**	0.46	**6.31**	0.73
Subkarte	7.78	0.35	56.94	7.23

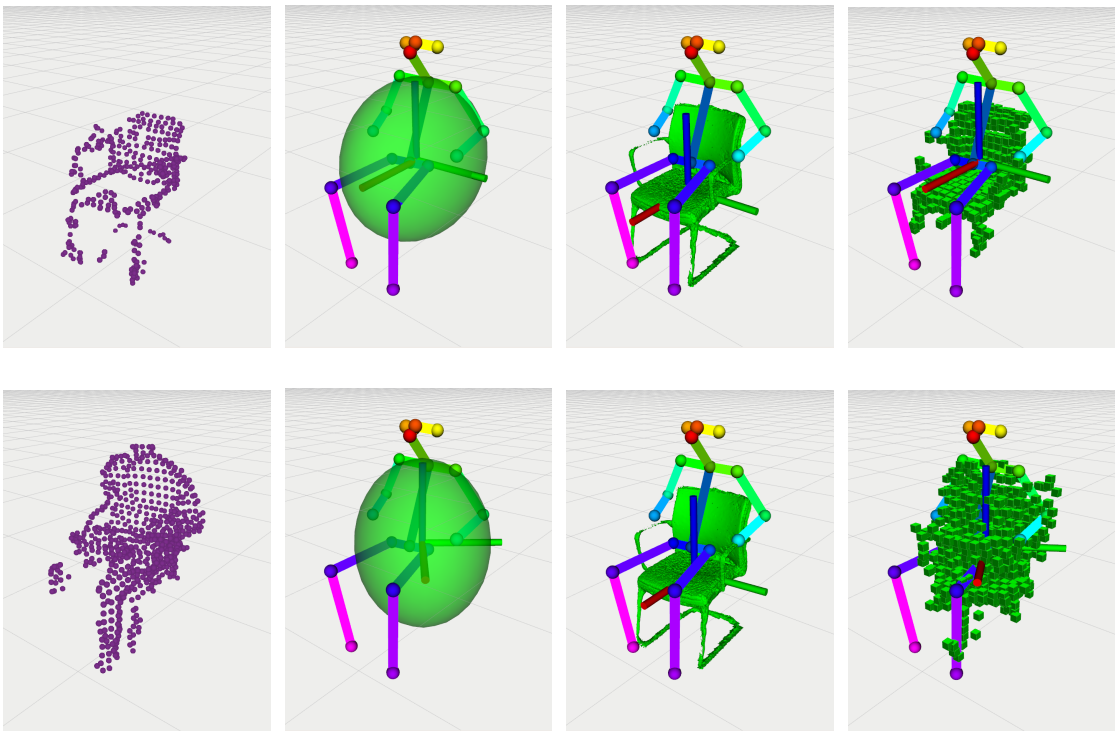
Tabelle 5.9: Posendaten für *chairwood_hand* mit Online CNN Segmentierung

	Translationsfehler (cm)		**Orientierungssfehler** (°)	
Typ	$\overline{E_{\text{trans}}}$	σ	$\overline{E_{\text{rot}}}$	σ
Ellipsoid	7.35	0.35	12.99	3.93
Skelett	8.74	0.77	8.65	1.70
Subkarte	**7.22**	0.34	**7.75**	1.69

Tabelle 5.10: Posendaten für *chairwood_sit* mit Online CNN Segmentierung

	Translationsfehler (cm)		**Orientierungssfehler** (°)	
Typ	$\overline{E_{\text{trans}}}$	σ	$\overline{E_{\text{rot}}}$	σ
Ellipsoid	9.62	0.27	28.05	4.55
Skelett	**5.09**	0.40	**9.88**	0.77
Subkarte	8.58	0.29	22.52	4.76

Die Ergebnisse zeigen, dass die Orientierungsschätzung der Ellipsoide und Subkarten in den Szenarien *chairblack_sit* (vgl. Tab. 5.8) und *chairwood_sit* (vgl. Tab. 5.10) stark fehlerhaft ist. Da in den Szenarien Person und Stuhl nicht richtig segmentiert werden können und große Teile der Person als Stuhl erkannt werden, nimmt die Punktwolkengröße bei Verdeckung nicht erkennbar ab. Die Segmente können folglich nicht herausgefiltert werden (vgl. (4.4)). Da keine Keypunkte für das verdeckte Objekt detektiert werden, wird die Pose von Ellipsoid und Subkarte mit der Hauptkomponentenanalyse aktualisiert. Diese Pose spiegelt allerdings nicht die tatsächliche Objektpose wider. Der Ablauf dieses Verhaltens ist in Abbildung 5.3 dargestellt und wird mit dem Ergebnis des gleichen Szenarios unter Verwendung der Ground-Truth Segmentierung verglichen, welches das gewünschte Verhalten zeigt. In den Szenarien *chairblack_hand* (vgl. Tab. 5.7) und *chairwood_hand* (vgl. Tab. 5.9) ist die Differenz zwischen den Translationsfehlern von Ellipsoid/Subkarte und Skelett-Repräsentation, verglichen mit der Ground-Truth Segmentierung, deutlich geringer. Das liegt daran, dass die Punktwolken deutlich mehr Rauschen enthalten, was einen direkten Einfluss auf die Translationsberechnung über den geometrischen Schwerpunkt der Punktwolken hat. Des Weiteren profitieren die Subkarten von der Nutzung der Hauptkomponentenanalyse in Fällen, in denen keine Keypunkt-Detektion zur Verfügung steht, die Ellipsoide jedoch nicht. Vergleicht man die einzelnen Ergebnisse der Online Segmentierung (vgl. Tab. 5.7, 5.8, 5.9, 5.10) mit den Resultaten der Ground-Truth Segmentierung (vgl. Tab. 5.3, 5.4, 5.5, 5.6), dann fällt auf, dass sich die Posenschätzungen von Ellipsoid und Subkarte in allen Szenarien durch die Daten mit höherem Rauschen der Online Segmentierung verschlechtert haben. In den Szenarien *chairblack_hand* (vgl. Tab. 5.3, 5.7) und *chairwood_hand* (vgl. Tab. 5.5, 5.9), in denen die Objekte verschoben aber kaum verdeckt werden, ist die Verschlechterung nur gering. Die Ergebnisse sind trotz höherem Rauschen brauchbar. Die Posenschätzung der Keypunkt-Skelette verändert sich ebenfalls, da die top-down Keypunkt-Detektionen von den Bounding-Boxen abhängen. Im Falle der Online CNN Segmentierung wurden die Bounding-Boxen nicht mehr aus den Ground-Truth Masken berechnet, sondern stammen von einem Online CNN. In den Szenarien *chairblack_hand* (vgl. Tab. 5.3, 5.7) und *chairwood_sit* (vgl. Tab. 5.6, 5.10) ist eine leichte Verschlechterungen erkennbar. Allerdings können auch Verbesserungen durch die Online Bounding-Box Detektion festgestellt werden. Im Szenario *chairblack_sit* (vgl. Tab. 5.4, 5.8) hat sich die Orientierungsschätzung leicht verbessert. Das Szenario *chairwood_hand* (vgl. Tab. 5.5, 5.9) zeigt eine deutlich verbesserte Posenschätzung. Die Anzahl der gültigen Keypunkt-Detektionen hat durch die Online Bounding-Box Detektion in diesem Szenario zugenommen.

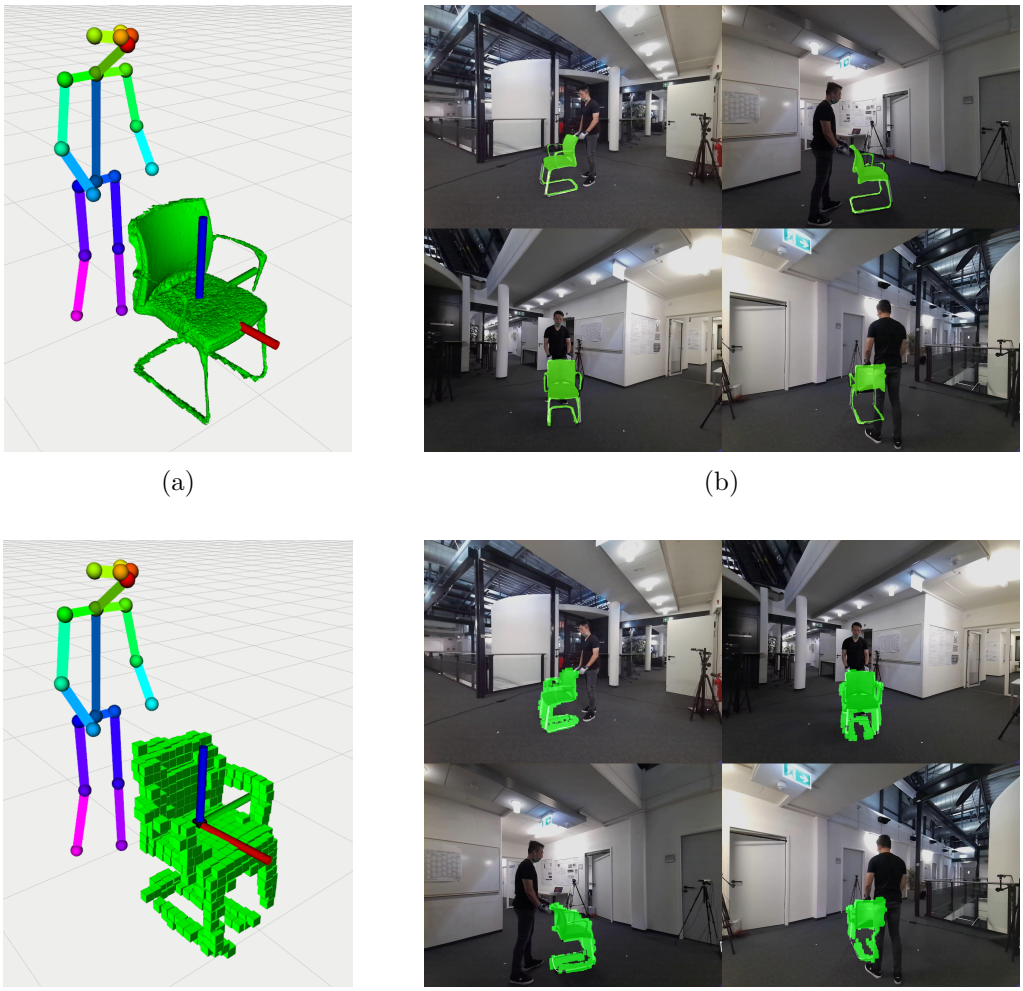


(a) (b) (c) (d)

Abbildung 5.3: Die verschiedenen Objektrepräsentationen bei Verdeckung: (a) zeigt ein Punktwolkensegment für die Ground-Truth Segmentierung (oben) und für die Online Segmentierung (unten). Die Anzahl der Punkte bei der Ground-Truth Segmentierung nimmt signifikant ab, wodurch ein fälschliche Aktualisieren der Objektpose verhindert wird. Bei der Online Segmentierung können Stuhl und Person nicht korrekt getrennt werden und die Objektpose von Ellipsoid (b) und Subkarte (d) wird mit der Pose des Punktwolkensegmentes aus (a) aktualisiert. Diese Pose spiegelt allerdings nicht die tatsächliche Pose des Objektes wider. Das Mesh (c) behält die vorherige Pose, da keine Keypunkte detektiert werden konnten.

5.2.2 Rückprojektion in die Kamerabilder

Um die Genauigkeit von Keypunkt-Skelett und Subkarte weiter zu überprüfen, werden die 3D-Objektrepräsentationen in die Kamerabilder projiziert. Anschließend erfolgt ein Vergleich mit der pseudo Ground-Truth Maske des Datensatzes durch Berechnung des IoU-Wertes. Die Keypunkt-Skelette werden projiziert, indem das zugehörige Mesh mit der entsprechenden Pose aus Sicht der Kameraperspektive gerendert wird. Die Subkarten werden projiziert, indem die Eckpunkte eines jeden Voxels in das Kamerabild projiziert werden und anschließend die konvexe Hülle der Punkte gefüllt wird. Die Ergebnisse einer solchen Projektion sind in Abbildung 5.4 visualisiert. Zudem wird auf die Mesh-Maske und die Ground-Truth



(a) (b)

(c) (d)

Abbildung 5.4: Rückprojektion in die Kamerabilder: Für das Keypunkt-Mesh (a) und die Subkarte (b) wurden das Objekt in die Kamerabilder zurück projiziert. (b) und (d) zeigen die Ergebnisse. Die Masken passen sehr genau zum Bild. Da es sich bei dem reprojizierten Mesh um einen 3D-Scan des verwendeten Objektes handelt, stimmen auch feine Elemente überein. Die Subkarte kann aufgrund der diskreten Auflösung feine Elemente wie Armlehnen oder Beine nicht akkurat darstellen.

Maske vor der Berechnung der IoU-Werte eine Dilatation mit einem 10×10 Kernel angewendet, da die verwendeten Stühle (vgl. Abb. 5.1) viele dünne Elemente enthalten und die Subkarte aufgrund der Voxelgröße diese nicht darstellen kann. Die Rückprojektion wird für jede Kameraperspektive der vier Szenarien einmal mit der Ground-Truth Segmentierung und einmal mit der Online CNN Segmentierung durchgeführt. Angegeben wird der mittlere IoU-Wert $\overline{E_{\text{IoU}}}$ und die Standardabweichung σ.

Tabelle 5.11: IoU-Werte für *chairblack_hand* mit Ground-Truth Segmentierung

	Kamera 1		**Kamera 2**		**Kamera 3**		**Kamera 4**		**Gesamt**
Typ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$
Mesh	0.64	0.13	0.69	0.12	0.61	0.14	0.56	0.12	**0.63**
Subkarte	0.62	0.05	0.66	0.05	0.57	0.07	0.57	0.06	0.60

Tabelle 5.12: IoU-Werte für *chairblack_sit* mit Ground-Truth Segmentierung

	Kamera 1		**Kamera 2**		**Kamera 3**		**Kamera 4**		**Gesamt**
Typ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$
Mesh	0.53	0.15	0.56	0.16	0.53	0.12	0.54	0.08	**0.54**
Subkarte	0.45	0.13	0.39	0.14	0.43	0.13	0.64	0.08	0.48

Tabelle 5.13: IoU-Werte für *chairwood_hand* mit Ground-Truth Segmentierung

	Kamera 1		**Kamera 2**		**Kamera 3**		**Kamera 4**		**Gesamt**
Typ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$
Mesh	0.46	0.19	0.42	0.19	0.52	0.17	0.48	0.21	0.47
Subkarte	0.64	0.06	0.60	0.07	0.67	0.04	0.65	0.04	**0.64**

Tabelle 5.14: IoU-Werte für *chairwood_sit* mit Ground-Truth Segmentierung

	Kamera 1		**Kamera 2**		**Kamera 3**		**Kamera 4**		**Gesamt**
Typ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$
Mesh	0.60	0.10	0.54	0.17	0.49	0.15	0.47	0.18	0.53
Subkarte	0.58	0.07	0.57	0.07	0.52	0.11	0.54	0.09	**0.55**

Ground-Truth Segmentierung

Die Tabellen 5.11, 5.12, 5.13 und 5.14 zeigen die IoU-Werte für die 4 Kameraperspektiven in den vier gewählten Szenarien unter Verwendung der Ground-Truth Segmentierung. Man kann erkennen, dass in den Szenarien mit starker Verdeckung (*sit*) geringere Werte im Vergleich zu den anderen Szenarien (*hand*) erzielt wurden. Die Werte des Meshes im Szenario *chairwood_hand* scheinen dieser Beobachtung zu widersprechen, allerdings ist in diesem Szenario die Keypunkt-basierte Posenschätzung, welche für die Pose des Meshes genutzt wird, besonders schlecht (vgl. Tab. 5.13), da bei einigen wichtigen Frames gültige Keypunkt-Detektionen ausbleiben.

Tabelle 5.15: IoU-Werte für *chairblack_hand* mit Online CNN Segmentierung

	Kamera 1		**Kamera 2**		**Kamera 3**		**Kamera 4**		**Gesamt**
Typ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$
Mesh	0.61	0.16	0.68	0.18	0.58	0.18	0.56	0.17	**0.61**
Subkarte	0.51	0.08	0.61	0.08	0.48	0.09	0.54	0.10	0.54

Tabelle 5.16: IoU-Werte für *chairblack_sit* mit Online CNN Segmentierung

	Kamera 1		**Kamera 2**		**Kamera 3**		**Kamera 4**		**Gesamt**
Typ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$
Mesh	0.48	0.13	0.43	0.17	0.60	0.17	0.56	0.08	**0.52**
Subkarte	0.30	0.12	0.33	0.13	0.34	0.10	0.56	0.08	0.38

Tabelle 5.17: IoU-Werte für *chairwood_hand* mit Online CNN Segmentierung

	Kamera 1		**Kamera 2**		**Kamera 3**		**Kamera 4**		**Gesamt**
Typ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$
Mesh	0.51	0.16	0.48	0.16	0.55	0.15	0.57	0.17	**0.53**
Subkarte	0.50	0.10	0.47	0.10	0.52	0.07	0.54	0.07	0.51

Tabelle 5.18: IoU-Werte für *chairwood_sit* mit Online CNN Segmentierung

	Kamera 1		**Kamera 2**		**Kamera 3**		**Kamera 4**		**Gesamt**
Typ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$	σ	$\overline{E_{\text{IoU}}}$
Mesh	0.55	0.14	0.45	0.12	0.45	0.12	0.51	0.13	**0.49**
Subkarte	0.41	0.13	0.35	0.09	0.32	0.09	0.34	0.11	0.35

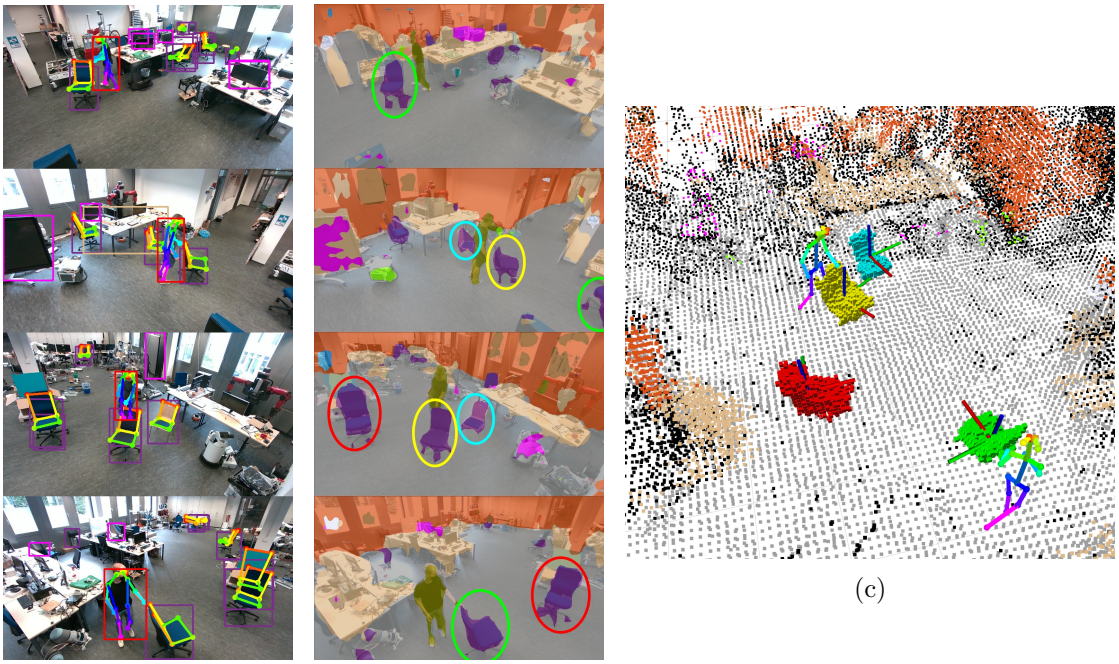
Online CNN Segmentierung

Die Tabellen 5.15, 5.16, 5.17 und 5.18 zeigen die IoU-Werte für die 4 Kameraperspektiven in den vier gewählten Szenarien unter Verwendung der Online CNN Segmentierung. Die Beobachtung für die Ground-Truth Segmentierung, dass in den Szenarien mit geringer Verdeckung (*hand*) bessere IoU-Werte erzielt werden, wird bestätigt. Mit der Online CNN Segmentierung werden für das Mesh in allen Szenarien bessere Werte berechnet als für die Subkarte. Das liegt daran, dass die Rekonstruktionsqualität der Subkarte stark vom Rauschen der segmentierten Punktwolke abhängt, während das Mesh lediglich über die Posenschätzung der Keypunkte bestimmt wird. Insgesamt liefert die Subkarte in den Szenarien *chairblack_sit* und *chairwood_sit* mit der Online CNN Segmentierung die schlechtesten IoU-Werte. Wie in Abschnitt 5.2.1 festgestellt, schlägt die Orientierungsschätzung in diesen Szenarien fehl. Im direkten Vergleich mit den IoU-Werten der Ground-

Truth Segmentierung (vgl. Tab. 5.11, 5.12, 5.13, 5.14) wird deutlich, dass sich die Werte für Subkarte und Mesh verschlechtert haben, wenn auch teilweise nur geringfügig. Eine Ausnahme ist das Mesh im Szenario *chairwood_hand* (vgl. Tab. 5.13, 5.13). Hier hat, wie in Absatz 5.2.1 beschrieben, die Anzahl der gültigen Keypunkt-Detektionen mit den Bounding-Boxen des Online CNN zugenommen, was in einer besseren Posenschätzung und Reprojektion resultiert.

5.3 Experimente mit dem Sensor Netzwerk

Die entwickelte Methode wurde zusätzlich in real-world Szenarien in der Robotik Halle unseres Instituts ausgewertet. Die Halle hat eine Fläche von ca. 237m^2 und eine Höhe von 3.2m. Über die gesamte Halle verteilt befinden sich 20 Smart Edge Sensoren. Dabei handelt es sich um 16 Sensoren basierend auf dem Google EdgeTPU Dev Board ausgestattet mit einer RGB-Kamera, die für die Erkennung und Posenschätzung von Personen genutzt werden [5] und um 4 Sensoren basierend



(a) (b) (c)

Abbildung 5.5: Keypunkt-Detektion, Segmentierung und die resultierende Karte: Mit den Informationen der Keypunkt-Detektionen (a) und der Segmentationsmasken (b) aus den vier Perspektiven werden die Objektrepräsentationen gebildet. Zusätzlich sind in (b) die Objektinstanzen durch farbige Ellipsen gekennzeichnet. In (c) werden die Subkarten in der globalen allozentrischen Karte [4] mit den menschlichen Posen [5] visualisiert.

auf dem Nvidia Jetson Xavier NX Developer Kit ausgestattet mit einer RGB-D Kamera, die neben dem Schätzen von menschlichen Posen auch für das Erstellen einer globalen allozentrischen Karte genutzt werden [4]. Die Daten der 4 Jetson NX Sensoren werden auch für die in dieser Arbeit vorgestellte Methode zum Erstellen von Objektrepräsentationen genutzt. Für die Visualisierung werden die Resultate des Objekt-Mappings mit der globalen allozentrischen Karte und den menschlichen Posen kombiniert (vgl. Abb. 5.5). Dazu wird die Objektklasse Stuhl aus der globalen allozentrischen Karte entfernt.

5.3.1 Qualitative Ergebnisse

In den folgenden Abschnitten werden die Ergebnisse von drei verschiedenen Szenarien gezeigt:

- Szenario 1: Die Stühle werden durch den Raum geschoben.
- Szenario 2: Die Stühle werden durch eine sitzende Person verdeckt.
- Szenario 3: Die Stühle werden während dem Sitzen bewegt.

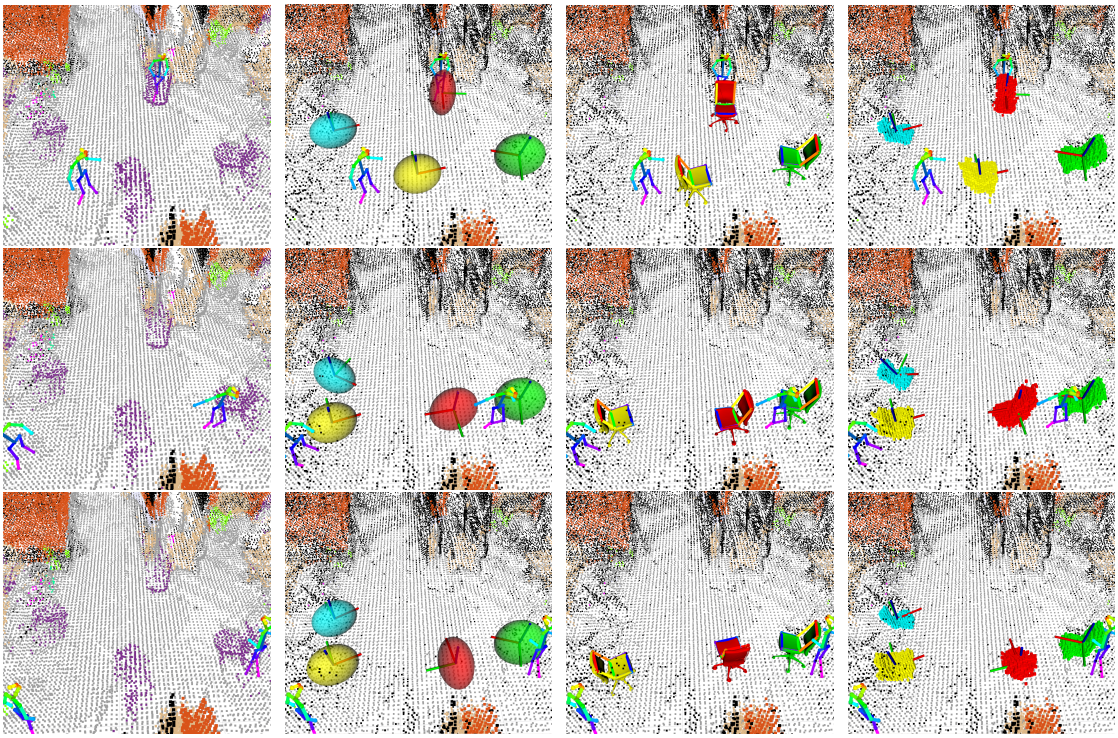
Verwendet wurden drei Bürostühle und ein Holzstuhl. Zwei der Bürostühle entsprechen dabei einem Stuhlmodell, für das ein 3D-Scan aufgezeichnet wurde und auf dem manuell Keypunkte ermittelt wurden (vgl. Abs. 4.3.1). Die Keypunkte dieses Stuhls dienen ebenfalls als Eingabe für den PnP-Algorithmus zur Ermittlung der 6DoF-Pose. Das gescannte Mesh wird für die Visualisierung der Keypunkt-Skelette verwendet.

Szenario 1

Abbildung 5.6 zeigt die Resultate der Bewegungsaktualisierung durch Raytracing aus [4] und die Resultate der aus dieser Arbeit stammenden Objektrepräsentation durch Ellipsoide, Keypunkt-Skelette und Subkarten zu drei verschiedenen Zeitpunkten. Der rote Stuhl wird von oben nach unten bewegt und währenddessen um 180° rotiert. Der gelbe Stuhl wird von der Mitte zum linken Bildrand bewegt. Die weiteren Stühle werden nicht bewegt. Bei den Objektinstanzen in Rot und Grün handelt es sich um die Bürostühle, die mit dem gescannten Stuhl, dessen Keypunkte für die Posenbestimmung genutzt werden, übereinstimmen. Der gelbe Stuhl ist ein weiterer Bürostuhl mit ähnlichen geometrischen Eigenschaften. Für alle drei Bürostühle liefert die Posenschätzung aus den Keypunktdaten gültige Ergebnisse. Die geometrischen Eigenschaften des Holzstuhls (blau) weichen zu stark von dem gescannten Stuhl ab, sodass die Posenschätzung via PnP unter

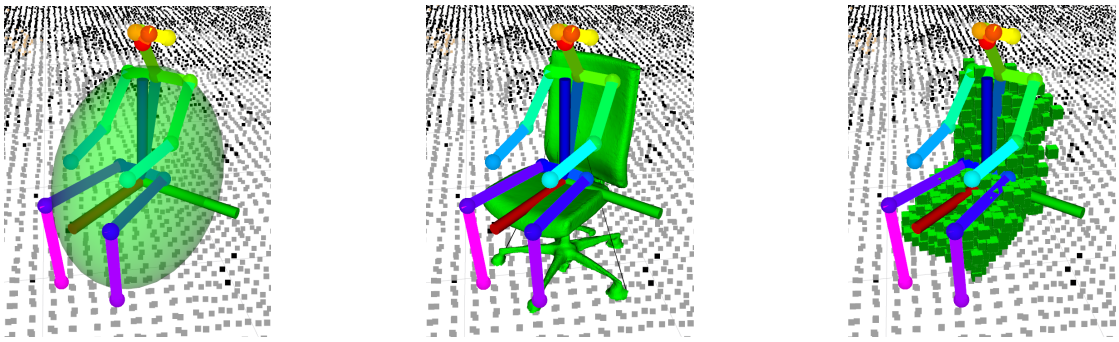
Verwendung der Keypunkte des gescannten Stuhls als Eingabe keine gültigen Ergebnisse erzeugt. Dementsprechend wird an dieser Position kein Mesh visualisiert. Für die Ellipsoide und Subkarten wird stattdessen die Posenschätzung über die Hauptkomponentenanalyse verwendet.

Die Berechnung der Posen- und Geometrie-Updates der Objektinstanzen läuft in Echtzeit, allerdings mit einer Latenz. Diese kommt durch die Verarbeitung der Daten und insbesondere die Synchronisierung der unterschiedlichen Perspektiven zustande, da die Fusion der Daten erst stattfindet, wenn aus allen Perspektiven eine neue Beobachtung vorliegt. Im direkten Vergleich mit dem Freizeichnen unbelegter Voxel durch Raytracing fällt auf, dass die zeitliche Verzögerung deutlich geringer ist und die Trajektorien der Objekte nachvollziehbar sind. Ein weiterer Vorteil der Subkarten ist, dass die Auflösung unabhängig von der globalen Karte gewählt werden kann. Im Beispiel wurde für die Subkartenstruktur eine Voxelgröße von 5cm gewählt, während die globale Karte eine Voxelgröße von 10cm hat.



(a) Globale Karte mit Raytracing (b) Ellipsoid (c) Mesh (d) Subkarte

Abbildung 5.6: Szenario 1: Stühle werden durch den Raum geschoben.



(a) Ellipsoid (b) Mesh (c) Subkarte

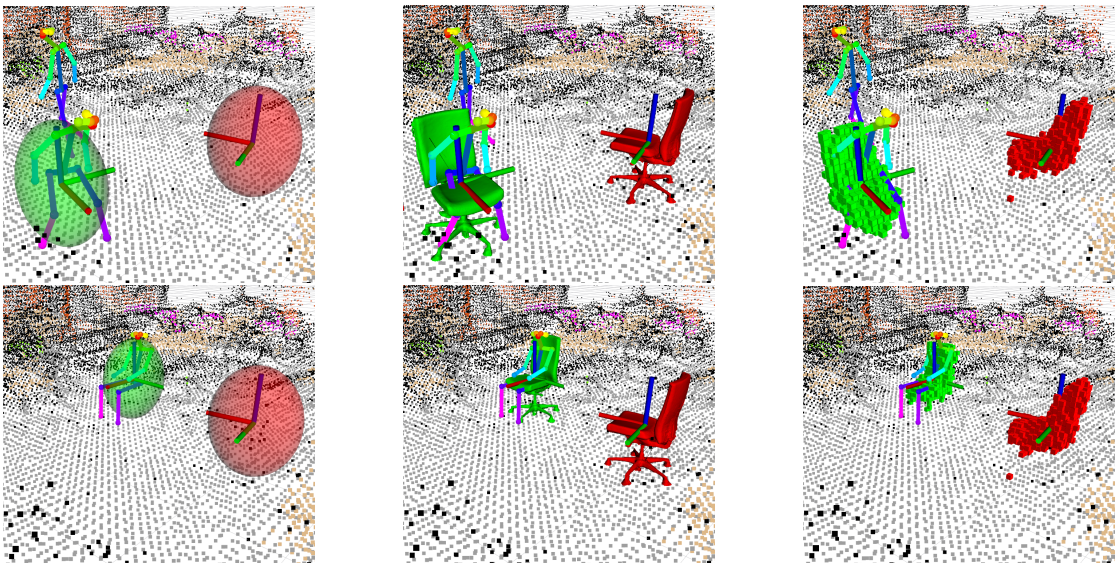
Abbildung 5.7: Szenario 2: Ein Stuhl wird durch eine sitzende Person verdeckt.

Szenario 2

In Abbildung 5.7 sind die Objektrepräsentationen, während ein Stuhl partiell durch eine sitzende Person verdeckt wird, exemplarisch dargestellt. Durch die Verdeckung werden keine guten Keypunkt-Detektionen gemacht und die mit der Hauptkomponentenanalyse berechnete Pose der partiellen Punktwolke stimmt nicht mit der tatsächlichen Objektpose überein. Indem die Detektionen von Personen vorrangig behandelt werden, ist sichergestellt, dass bei Verdeckung durch eine Person die Anzahl der Punkte der fusionierten Punktwolke des Objektes erkennbar abnimmt. Folglich werden partielle Beobachtungen aussortiert und die Objektpose wird nicht mit der Hauptkomponenten-Pose des partiellen Clusters verfälscht. Die letzte valide Beobachtung bleibt erhalten, wodurch die Objektpose trotz Verdeckung stabil bleibt. Dieses Vorgehen birgt das Risiko, dass gültige Keypunkt-Detektionen verworfen werden, da die Posen der Keypunkte den Punktwolkensegmenten zugeordnet werden. Jedoch werden im Allgemeinen, sobald eine Person einen Stuhl verdeckt, keine Keypunkte von ausreichender Qualität detektiert, da der verwendete Trainingsdatensatz (vgl. Abs. 4.3.1) keine entsprechenden Szenarien enthält. Der Verlust gültiger Keypunkt-Detektionen ist also vernachlässigbar. Das beschriebene Verhalten funktioniert gut, solange das verdeckte Objekt statisch ist. Ein Ansatz für das Tracken der Bewegungen verdeckter Objekte wird im Folgenden betrachtet.

Szenario 3

Das Verschieben der Stühle während dem Sitzen erweist sich als schwierig für die Methode. Wie bereits beschrieben sind im stark verdeckten Zustand die Posenschätzungen häufig unbrauchbar oder nicht vorhanden, sodass das Tracking der Objektpose mit der vorgestellten Methode nicht möglich ist. Um trotzdem die Objektinstanz verfolgen zu können, wird, sofern keine neue gültige Beobachtung



(a) Ellipsoid (b) Mesh (c) Subkarte

Abbildung 5.8: Szenario 3: Der grüne Stuhl wird während dem Sitzen verschoben.

vorliegt, geprüft, ob eine Person in unmittelbarer Nähe des Stuhls erkannt wurde. Sollte das der Fall sein, wird eine Zuordnung zwischen Objektinstanz und Person getroffen. Die Objektpose richtet sich nach der Pose der Hüfte der Person. Abbildung 5.8 zeigt eine solche Situation. Der grüne Stuhl wird von links unten an den oberen Rand des Bildes bewegt. Durch die Zuordnung zu der entsprechenden Personeninstanz kann die Bewegung verfolgt werden.

5.3.2 Datenübertragungsrate

Die verschiedenen Objektrepräsentationen unterscheiden sich erheblich in der notwendigen Datenübertragungsrate zwischen den Sensorboards und dem Backend. Während die Ellipsoide und die Keypunkt-Skelette mit einer festen Datenmenge pro Objekt auskommen, variiert die benötigte Datenmenge für die Subkarten mit der Größe der Punktwolke. Da die Keypunkt-basierte Orientierungsschätzung auch für die Ellipsoide genutzt wurde, benötigen beide Strukturen ungefähr die gleiche Datenmenge. Die Datenmenge besteht aus: Translation der Punktwolke, Orientierung der Punktwolke, statistische Verteilung der Punktwolke, Anzahl der Punkte, PnP-Translation, PnP-Orientierung, mittlerer Abstand zwischen den Keypunkten und dem nächsten Nachbarn in der Punktwolke und einem Boolean, das angibt ob eine valide Keypunkt-Detektion getätigt wurde. Damit beträgt die Datenmenge pro Objekt 145 Bytes. Für die Ellipsoide wird die PnP-Translation nicht benötigt, wodurch sich eine Datenmenge von 121 Bytes ergibt.

Für die Subkarten kann auf die Berechnung von Punktwolken-spezifischen Daten auf den Sensorboards verzichtet werden. Stattdessen wird das gesamte Punktwolkensegment inklusive der Keypunkt-spezifischen Daten an das Backend übermittelt. Für ein Objekt mit einer mittleren Punktwolkengröße von 250 Punkten liegt die zu übertragene Datenmenge bereits bei ca. 9 Kilobyte.

In unserem Smart Edge Sensor Netzwerk findet die Datenübertragung zwischen Sensorboards und Backend mit einer Frequenz von 1 Hz statt. Sollte der Fokus auf einer geringen Datenübertragungsrate liegen, ist eine Objektrepräsentation zu präferieren, die nicht die Übertragung der Punktwolke zum Backend erfordert.

6 Fazit

In dieser Arbeit wurde eine globale allozentrische Karte [4], welche mit einem Smart Edge Sensor Netzwerk generiert wird, um verschiedene Objektrepräsentationen erweitert, die Instanz-spezifische Informationen speichern. Die Methode ermöglicht es, Objekte zu tracken und mit den aufgezeichneten Daten Ellipsoide, Keypunkt-Skelette und Subkarten zu konstruieren. Dadurch können Bewegungen verfolgt werden und Beobachtungen bleiben bei temporärer Verdeckung erhalten. Verglichen mit dem Freizeichnen unbelegter Voxel durch Raytracing in der globalen Karte, kann zusätzlich eine geringere Latenz festgestellt werden. Da bisher die Keypunkt-basierte Posenschätzung nur für die Objektklasse Stuhl umgesetzt wurde, haben wir uns auf diese Klasse beschränkt. Die Methode kann allerdings unter Verwendung der in dieser Arbeit genutzten Frameworks zur Erstellung synthetischer Trainingsdaten analog auf weitere Klassen erweitert werden. Die für die Subkarten und Ellipsoide genutzte Posenschätzung über die Hauptkomponentenanalyse hängt stark von der Qualität der Segmentierung ab. Falsche Segmentierungen können dazu führen, dass die Posenschätzung verdeckter Objekte verfälscht wird. Im Vergleich dazu ist die Posenschätzung auf Basis der Keypunkt-Detektionen genauer und weniger rauschanfällig. Jedoch liegt nicht in allen Fällen ein gültiges Ergebnis vor. Voraussetzung für die Keypunkt-basierte Posenschätzung ist es, die geometrischen Eigenschaften des zu schätzenden Objektes in Form von Ground-Truth Keypunkten zu kennen, die der PnP-Algorithmus als Eingabe neben den detektierten Keypunkten benötigt. Sollten die detektierten Keypunkte zu stark abweichen, beispielsweise durch andere geometrische Eigenschaften des Objektes oder ungenaue Detektionen, dann kann keine oder nur eine schlechte Pose gefunden werden. Subkarten und Ellipsoide sind in der Darstellung unterschiedlicher Objekte einer Klasse flexibler als die Keypunkt-Skelette, da das Resultat nicht nur von den Keypunkten, sondern von den detektierten Punktwolkensegmenten abhängt. Ein Nachteil der Subkarten ist allerdings die deutlich höhere Datenübertragungsrate zwischen Sensorboards und Backend, verglichen mit Ellipsoiden und Keypunkt-Skeletten.

6.1 Ausblick

Die Keypunkt-basierte Posenschätzung sollte als nächstes auf weitere Klassen erweitert werden. Dabei kommen Klassen in Frage, deren Instanzen über eine geometrische Segmentierung zu finden sind und häufig ihre Position verändern. Beispiele für solche Klassen in Bezug auf das Smart Edge Sensor Netzwerk sind mobile Computer oder Roboter. Objekte, die nur selten ihre Position ändern, werden bereits ausreichend über das Raytracing in der globalen allozentrischen Karte behandelt.

Des Weiteren kann die Zuverlässigkeit der Keypunkt-basierten Posenschätzung weiter ausgebaut werden. Dies wäre möglich, indem die Keypunkt-Detektion auch auf den Sensoren ohne Tiefenkamera, die bereits zur Detektion von menschlichen Posen [5] genutzt werden, stattfindet. Eine weitere Möglichkeit wäre das Ergänzen des synthetischen Trainingsdatensatzes um manuell annotierte Bilder aus dem Sensor Netzwerk. Darüber hinaus könnte auch mit Szenarien trainiert werden, in denen Personen mit den Objekten interagieren, um beispielsweise auch zuverlässig Keypunkt-Detektionen zu tätigen, wenn eine Person auf einem Stuhl sitzt. Dazu würde sich der Behave-Datensatz [2] ebenfalls eignen. Außerdem kann durch systematisches Ausprobieren verschiedener Grenzwerte für den PnP-RANSAC Algorithmus eine Konfiguration gefunden werden, die möglichst genaue Posen in möglichst vielen Bildern findet, robuster mit Außreißern umgeht und gleichzeitig der Posenschätzung der Hauptkomponentenanalyse überlegen bleibt.

Als Alternative zur Posenschätzung mit dem PnP-Algorithmus könnte, wie in [5], über trigonometrische Berechnungen die 3D-Position der Keypunkte bestimmt werden. Dadurch wäre keine A-priori-Information über die Ground-Truth Keypunkte der Objekte mehr notwendig. Diese Methode erfordert jedoch eine Detektion aus mindestens zwei Perspektiven.

Abbildungsverzeichnis

2.1 2D Occupancy Grid Map [17] 3
2.2 Die 6 Freiheitsgrade eines starren Körpers im 3D-Raum [27] 4
2.3 ICP angewendet auf zwei Linien [25] 7
2.4 Lochkamera Modell [18] 7
2.5 Illustration des PnP-Algorithmus [20] 9

3.1 Pipeline zur Erstellung einer semantischen Karte und Bestimmung von Personen Posen [4] 12
3.2 Die 16 semantischen Klassen der Methode [4] 13
3.3 RGB-Bilder und Visualisierung verschiedener Szenarien des Behave-Datensatzes [2] 15

4.2 Fusion der Daten von 4 Punktwolken aus verschiedenen Perspektiven in die Ellipsoid-Struktur 20
4.3 Die im Trainingsdatensatz verwendeten Stuhl 3D-Modelle 22
4.4 Einzelne Frames aus zwei verschiedenen Trainings-Szenarien und die dazugehörigen Ground-Truth Keypunkt-Annotationen 23
4.5 2D-Keypunkt Detektionen und die mit PnP-RANSAC berechneten 3D-Posen 25
4.6 Fusion von Keypunkt-Skeletten aus drei verschiedenen Perspektiven 26
4.7 Fusion von 4 Punktwolken und Eintragen in die Subkarte 29

5.1 Die im Behave-Datensatz verwendeten Stühle 33
5.2 Die 2D-Keypunkt-Detektionen und 3D-Skelette zweier aufeinanderfolgender Frames 37
5.3 Die verschiedenen Objektrepräsentationen bei Verdeckung 40
5.4 Rückprojektion in die Kamerabilder 41
5.5 Keypunkt-Detektion, Segmentierung und die resultierende Karte . . 44
5.6 Szenario 1: Stühle werden durch den Raum geschoben. 46
5.7 Szenario 2: Ein Stuhl wird durch eine sitzende Person verdeckt. . . 47
5.8 Szenario 3: Der grüne Stuhl wird während dem Sitzen verschoben. . 48

Tabellenverzeichnis

5.1 Einfluss der Keypunkt-basierten Posenschätzung auf die Ellipsoide . 34
5.2 Einfluss der Keypunkt-basierten Posenschätzung auf die Subkarten 34
5.3 Posendaten für *chairblack_hand* mit Ground-Truth Segmentierung 35
5.4 Posendaten für *chairblack_sit* mit Ground-Truth Segmentierung . . 35
5.5 Posendaten für *chairwood_hand* mit Ground-Truth Segmentierung 35
5.6 Posendaten für *chairwood_sit* mit Ground-Truth Segmentierung . . 35
5.7 Posendaten für *chairblack_hand* mit Online CNN Segmentierung . 38
5.8 Posendaten für *chairblack_sit* mit Online CNN Segmentierung . . . 38
5.9 Posendaten für *chairwood_hand* mit Online CNN Segmentierung . 38
5.10 Posendaten für *chairwood_sit* mit Online CNN Segmentierung . . . 38
5.11 IoU-Werte für *chairblack_hand* mit Ground-Truth Segmentierung . 42
5.12 IoU-Werte für *chairblack_sit* mit Ground-Truth Segmentierung . . 42
5.13 IoU-Werte für *chairwood_hand* mit Ground-Truth Segmentierung . 42
5.14 IoU-Werte für *chairwood_sit* mit Ground-Truth Segmentierung . . 42
5.15 IoU-Werte für *chairblack_hand* mit Online CNN Segmentierung . . 43
5.16 IoU-Werte für *chairblack_sit* mit Online CNN Segmentierung . . . 43
5.17 IoU-Werte für *chairwood_hand* mit Online CNN Segmentierung . . 43
5.18 IoU-Werte für *chairwood_sit* mit Online CNN Segmentierung . . . 43

Literatur

[1] John Amanatides und Andrew Woo. „A fast voxel traversal algorithm for ray tracing“. In: *In eurographics '87*. 1987, S. 3–10.

[2] Bharat Lal Bhatnagar, Xianghui Xie, Ilya Petrov, Cristian Sminchisescu, Christian Theobalt und Gerard Pons-Moll. „Behave: dataset and method for tracking human object interactions“. In: *IEEE conference on computer vision and pattern recognition (cvpr)*. IEEE. Juni 2022.

[3] Andreas Boltres, Angel Villar-Corrales, Jan Nogga und Peer Schütt. *sl-cutscenes*.

[4] Simon Bultmann und Sven Behnke. *3d semantic scene perception using distributed smart edge sensors*. 2022. URL: https://arxiv.org/abs/2205.01460.

[5] Simon Bultmann und Sven Behnke. „Real-time multi-view 3d human pose estimation using semantic feedback to smart edge sensors“. In: *Robotics: science and systems XVII*. Robotics: Science und Systems Foundation, Juli 2021. URL: https://doi.org/10.15607%2Frss.2021.xvii.040.

[6] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei und Yaser Sheikh. *Openpose: realtime multi-person 2d pose estimation using part affinity fields*. 2018. URL: https://arxiv.org/abs/1812.08008.

[7] *Cgtrader*. URL: https://www.cgtrader.com.

[8] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff und Hartwig Adam. *Encoder-decoder with atrous separable convolution for semantic image segmentation*. 2018. URL: https://arxiv.org/abs/1802.02611.

[9] Albert Clapés, Julio C. S. Jacques Junior, Carla Morral und Sergio Escalera. „Chalearn lap 2020 challenge on identity-preserved human detection: dataset and results“. In: *2020 15th ieee international conference on automatic face and gesture recognition (fg 2020)*. 2020, S. 801–808.

[10] Margarita Grinvald, Fadri Furrer, Tonci Novkovic, Jen Jen Chung, Cesar Cadena, Roland Siegwart und Juan I. Nieto. „Volumetric instance-aware semantic mapping and 3d object discovery“. In: *Corr* abs/1903.00268 (2019). arXiv: 1903.00268. URL: http://arxiv.org/abs/1903.00268.

[11] Margarita Grinvald, Federico Tombari, Roland Siegwart und Juan Nieto. *Tsdf++: a multi-object formulation for dynamic object tracking and reconstruction*. 2021. URL: https://arxiv.org/abs/2105.07468.

[12] Tomas Hodan. *Bop-toolkit*. URL: https://github.com/thodan/bop_toolkit.

[13] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss und Wolfram Burgard. „OctoMap: an efficient probabilistic 3D mapping framework based on octrees“. In: *Autonomous robots* (2013). Software available at https://octomap.github.io. URL: https://octomap.github.io.

[14] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le und Hartwig Adam. *Searching for mobilenetv3*. 2019. URL: https://arxiv.org/abs/1905.02244.

[15] Du Huynh. „Metrics for 3d rotations: comparison and analysis“. In: *Journal of mathematical imaging and vision* 35 (Okt. 2009), S. 155–164.

[16] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick und Piotr Dollár. *Microsoft coco: common objects in context*. cite arxiv:1405.0312Comment: 1) updated annotation pipeline description and figures; 2) added new section describing datasets splits; 3) updated author list. 2014. URL: http://arxiv.org/abs/1405.0312.

[17] Rasoul Mojtahedzadeh. *Robot obstacle avoidance using the kinect*. 2011. URL: https://www.researchgate.net/figure/A-sample-Occupancy-Grid-representation-of-the-environment-White_fig13_257541372.

[18] Nvidia. *Pinhole camera model*. URL: https://docs.nvidia.com/vpi/appendix_pinhole_camera.html.

[19] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart und Juan Nieto. „Voxblox: incremental 3d euclidean signed distance fields for on-board MAV planning“. In: *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, Sep. 2017. URL: https://doi.org/10.1109%2Firos.2017.8202315.

[20] OpenCV. *Perspective-n-point (pnp) pose computation*. URL: https://docs.opencv.org/4.x/d5/d1f/calib3d_solvePnP.html.

[21] Martin Rünz, Maud Buffier und Lourdes Agapito. *Maskfusion: real-time recognition, tracking and reconstruction of multiple moving objects*. 2018. URL: https://arxiv.org/abs/1804.09194.

[22] Radu Bogdan Rusu und Steve Cousins. „3d is here: point cloud library (pcl)“. In: *2011 ieee international conference on robotics and automation*. 2011, S. 1–4.

[23] Max Schwarz und Sven Behnke. „Stillleben: realistic scene synthesis for deep learning in robotics“. In: *Corr* abs/2005.05659 (2020). arXiv: 2005.05659. URL: https://arxiv.org/abs/2005.05659.

[24] *Sketchfab*. URL: https://sketchfab.com.

[25] Erik Smistad, Thomas Falch, Mohammadmehdi Bozorgi, Anne Elster und Frank Lindseth. *Medical image segmentation on gpus - a comprehensive review*. 2015. URL: https://www.researchgate.net/figure/Illustration-of-the-iterative-closest-point-method-to-align-two-lines-A-set-of-points-is_fig5_269703385.

[26] Jörg Stückler, Nenad Biresev und Sven Behnke. „Semantic mapping using object-class segmentation of rgb-d images“. In: Okt. 2012.

[27] Wikipedia. *Sechs freiheitsgrade*. URL: https://en.wikipedia.org/wiki/Six_degrees_of_freedom#/media/File:6DOF.svg.

[28] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan und Dieter Fox. *Posecnn: a convolutional neural network for 6d object pose estimation in cluttered scenes*. 2017. URL: https://arxiv.org/abs/1711.00199.

[29] Bin Xiao, Haiping Wu und Yichen Wei. *Simple baselines for human pose estimation and tracking*. 2018. URL: https://arxiv.org/abs/1804.06208.

[30] Yunyang Xiong, Hanxiao Liu, Suyog Gupta, Berkin Akin, Gabriel Bender, Yongzhe Wang, Pieter-Jan Kindermans, Mingxing Tan, Vikas Singh und Bo Chen. „Mobiledets: searching for object detection architectures for mobile accelerators“. In: *Proceedings of the ieee/cvf conference on computer vision and pattern recognition (cvpr)* (2021).

[31] Binbin Xu, Wenbin Li, Dimos Tzoumanikas, Michael Bloesch, Andrew Davison und Stefan Leutenegger. *Mid-fusion: octree-based object-level multi-instance dynamic slam*. 2018. URL: https://arxiv.org/abs/1812.07976.

[32] Moritz Zappel, Simon Bultmann und Sven Behnke. „6d object pose estimation using keypoints and part affinity fields“. In: (2021). URL: https://arxiv.org/abs/2107.02057.

[33] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso und Antonio Torralba. „Semantic understanding of scenes through the ade20k dataset“. In: *International journal of computer vision* 127 (März 2019).