Technische Universität Ilmenau
Fakultät für Informatik und Automatisierung
Fachgebiet Neuroinformatik und Kognitive Robotik

Recherche und Evaluation von Features zur Detektion von gestürzten Personen in häuslichen Umgebungen

Masterarbeit zur Erlangung des akademischen Grades Master of Science

Friederike Schneemann

Betreuender wiss. Mitarbeiter:	Dipl.-Inf. Michael Volkhardt
Verantwortlicher Hochschullehrer:	Prof. Dr. H.-M. Groß

Die Masterarbeit wurde am 07.03.2013 bei der Fakultät für Informatik und Automatisierung der Technischen Universität Ilmenau eingereicht.

urn:nbn:de:gbv:ilm1-2013200135

Erklärung: „Hiermit versichere ich, dass ich diese Masterarbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe. Alle von mir aus anderen Veröffentlichungen übernommenen Passagen sind als solche gekennzeichnet."

Ilmenau, 07.03.2013 .....

Friederike Schneemann

Zusammenfassung

Die vorliegende Arbeit befasst sich mit der Recherche und Evaluation von Features zur Detektion gestürzter Personen in häuslichen Umgebungen über eine mobile Roboterplattform.

Etwa ein Drittel der Menschen über 65 Jahren stürzen mindestens einmal pro Jahr. Fast die Hälfte dieser Personen schafft es nach dem Sturz nicht, aus eigener Kraft wieder aufzustehen. Da eine lange Liegezeit zu ernsten Komplikationen führen kann, bedarf es einer zuverlässigen Methodik zur Erkennung gestürzter Personen, über die unverzüglich Hilfe angefordert werden kann. Die bisher kommerziell verfügbaren Produkte bieten jedoch nur bedingt eine Lösung. Großes Potential wird hier in der Service- und Assistenzrobotik gesehen. Vor allem die 3D-Daten der Tiefenkamera Kinect, mit der die meisten der aktuellen Serviceroboter ausgestattet sind, versprechen neue Möglichkeiten zur robusten Erkennung von Stürzen.

Diese Arbeit gibt einen ausführlichen Überblick über bestehende Verfahren zur Sturzdetektion, sowie zur merkmalsbasierten Detektion von Personen und Objekten in 3D-Daten. Die vorgestellten Ansätze werden bezüglich ihrer Eignung für den in dieser Arbeit betrachteten Anwendungsfall bewertet.

Auf Basis der Bewertungsergebnisse wird ein neuer Ansatz zur Evaluation geeigneter Features und zur Detektion gestürzter Personen über eine mobile Roboterplattform entwickelt. Die einzelnen Komponenten des neuen Ansatzes werden in der vorliegenden Arbeit ausführlich vorgestellt und die Details der Implementierung angesprochen.

Mit der Kinect einer mobilen Roboterplattform selbst aufgenommenes Testmaterial dient der Evaluation des neu entwickelten Verfahrens. Im Rahmen der Evaluation wird die Detektionsleistung verschiedener maschineller Lerntechniken und Merkmale miteinander verglichen. Die Ergebnisse der Evaluation werden in der Arbeit ausführlich vorgestellt und diskutiert. Es zeigt sich, dass das Histogram of Local Surface Normals in Kombination mit einer Support Vector Machine sehr gut geeignet ist, gestürzte Personen in den 3D-Daten einer mobile Roboterplattform zu detektieren.

Die Arbeit schließt mit einer Zusammenfassung der Ergebnisse und einem Ausblick für weiterführende Arbeiten.

Abstract

This thesis deals with the research and evaluation of features to detect fallen people in a home environment with a mobile robot.

About one third of people aged over 65 fall at least once a year. Half of the people can't manage to get up after the fall by own means. As lying on the floor for a long time can cause serious health risks, a reliable method to detect fallen people and to call for help is needed. Commercially available products provide only a limited solution. Robotic Assistance for the Elderly promises to have great potential in this field. Especially the depth data of a Kinect offers new possibilities for a reliable system that detects fallen people. The Kinect is a standard equipment of today's assistance robots.

This thesis gives an overview on known approaches for fall detection, as for feature-based people and object detection in 3D-Data and analyses the ability of the presented approaches for the application examined in this thesis.

A new approach to evaluate suitable features and to detect fallen people with a mobile robot is implemented. The individual components of the new approach are presented in details.

Data recorded by using the Kinect of a mobile robot, is used to evaluate the developed approach. The eavaluation compares the detection rate of different machine learning techniques and features. The achieved results are presented and discussed. The evaluation shows, that a Histogram of Local Surface Normals in combination with a Support Vector Machine is suitable to detect fallen people in 3D-Data generated by a mobile robot.

Finally, a summary on the results and suggestions for further research is given.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Etwa 50% aller älteren Menschen leiden unter der Angst in der häuslichen Umgebung zu fallen [Deshpande et al., 2009]. Diese Angst ist nicht unbegründet. Jeder Dritte über 65-Jährige stürzt mindestens einmal pro Jahr [Lord et al., 2003]. Bei den über 80-Jährigen sogar jeder Zweite [Noury et al., 2008]. Fast die Hälfte der älteren Personen schafft es nach dem Sturz nicht, aus eigener Kraft wieder aufzustehen [Alexander, 2010]. Eine lange Liegezeit kann jedoch zu ernsten Komplikationen wie Dehydration[1], Hypothermie[2] und einer daraus häufig resultierenden Pneumonie[3] führen. Ein Viertel aller Personen, die nach dem Sturz länger als eine Stunde auf dem Boden liegen, sterben innerhalb des folgenden Jahres, auch wenn der Sturz keine direkten physischen Verletzungen hervorruft [Wild et al., 1981]. Unverzügliche Hilfe ist somit ein entscheidender Faktor bei der Reduzierung der Sturzfolgen. Sie reduziert das Todesrisiko nach einem Sturz um 80% und das Risiko eines Krankenhausaufenthalts um 26% [Noury et al., 2008].

Um die Sicherheit allein lebender, älterer Personen zu erhöhen, bedarf es daher einer zuverlässigen Methodik zur Erkennung gestürzter Personen. Die bisher kommerziell verfügbaren Produkte bieten jedoch nur bedingt eine Lösung (Abschnitt 3.1).

[1] Wassermangel, [2] Unterkühlung, [3] Lungenentzündung

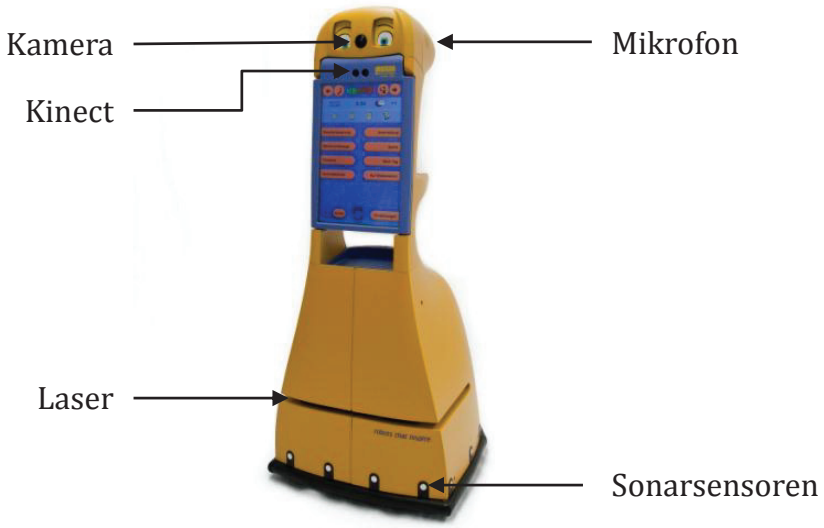


Abbildung 1.1: Sensorik von Service- und Assistenzrobotern
Um seine Umgebung wahrzunehmen ist der mobile Roboterassistent SERROGA mit einer Vielzahl verschiedener Sensoren ausgestattet.

Abhilfe kann die Service- und Assistenzrobotik schaffen, in der großes Potential zur Verbesserung der Lebenssituation älterer Menschen gesehen wird. Für die Zukunft sind mobile Roboterassistenten geplant, die ältere Personen bei ihren täglichen Aufgaben im Haushalt unterstützen und zur Unterhaltung beitragen. Durch die so geförderte selbstständige Lebensführung können Senioren länger ein selbstbestimmtes Leben in ihrem eigenen Zuhause führen, während der im Haushalt lebende Roboterassistent für die nötige Sicherheit sorgt und in Notsituationen Hilfe anfordern kann.

Im Rahmen des vom Thüringer Ministerium für Wirtschaft, Arbeit und Technologie geförderten Projekts SERROGA (Service-Robotik für die Gesundheitsassistenz) [Ilmenau, 2012] arbeitet das Fachgebiet Neuroinformatik und Kognitive Robotik der TU Ilmenau an der Entwicklung eines „Gesundheitsroboters“. Dieser soll älteren Personen als Kommunikationsassistent dienen, sie an die Einnahme von Medikamenten und an Termine erinnert, sowie zur gesundheitlichen Prävention animieren. Zudem soll die Funktionalität des Roboters auf die Erkennung kritischer Situationen erweitert werden, zu denen vor allem der Sturz in der häuslichen Umgebung gehört.

Aktuelle Serviceroboter sind zur Wahrnehmung ihrer Umgebung mit einer Vielzahl verschiedener Sensoren ausgestattet (Abb. 1.1). Im Gegensatz zu den von monokularen Kameras oder Lasern erfassten Daten, versprechen die 3D-Daten der Tiefenkamera Kinect gut geeignet zu sein, gestürzte Personen in der häuslichen Umgebung zu detektieren. Verglichen mit der monokularen Kamera weist die Kinect eine deutlich geringere Empfindlichkeit gegenüber Beleuchtungsvariationen auf. Zudem ermöglichen die zur Verfügung stehenden Tiefeninformationen eine verbesserte Segmentierung der Vordergrundobjekte vom Hintergrund. Die vom Laser erzeugten Daten sind nicht geeignet, da durch die hohe Einbaulage des Lasers nicht sichergestellt werden kann, dass die gestürzte Person in den eindimensionalen Daten erfasst wird.

1.2 Problemstellung

Bei der Detektion gestürzter Personen über eine mobile Roboterplattform ist zu berücksichtigen, dass der Sturz außerhalb des Sensorbereichs der mobilen Roboterplattform statt finden kann (z. B. wenn der Roboter sich zum Zeitpunkt des Sturzes in einem anderen Raum befindet). Daher reicht es in diesem Anwendungsfall nicht, den aktiven Sturzprozess zu detektieren. Die mobile Roboterplattform muss auch in der Lage sein, auf dem Boden liegende Personen zu erkennen. Im Vergleich zur Detektion stehender Personen bietet die Detektion liegender Personen eine deutlich größere Herausforderung. Durch die bei Stürzen häufig vorkommende Verdeckung einzelner Körperteile durch den Körper selbst oder durch andere Objekte (wie z. B. Tische und Sofas) gehen für die Detektion wichtige Charakteristiken verloren. Hierzu zählen beispielsweise die typische, von Kopf und Schulter gebildete Omegaform und die horizontale Symmetrie des Körpers. Zudem muss ein Detektor mit der deutlich höheren Abbildungsvielfalt liegender Personen umgehen können. Hier gestaltet sich die Aufnahme repräsentativen Trainingsmaterials als schwierig, da Stürze nur simuliert werden können und eine Abdeckung aller möglichen Posen nicht möglich ist. Bei 2-dimensionalen Bilddaten kommt hinzu, dass bei der 2D-Abbildung liegender Personen perspektivische Verzerrungen auftreten, die die Abbildungsvielfalt zusätzlich erhöhen [Wang et al., 2011].

1.3 Zielsetzung

Ziel dieser Masterarbeit ist die Recherche und Evaluation von Features zur Erkennung gestürzter Personen auf Basis der 3D-Daten der Tiefenkamera Kinect. Hierzu soll ein ausführlicher Überblick über geeignete Verfahren zur Detektion gestürzter Personen erstellt werden, der neben bekannten Verfahren zur Sturzdetektion auch etablierte Verfahren zu Detektion von Personen und Objekten in 3D-Daten vorstellt und diese hinsichtlich ihrer Eignung zur Detektion gestürzter Personen analysiert. Ausgehend von den Analyseergebnissen sollen vielversprechende Features implementiert und evaluiert werden. Hierzu ist eine valide Test-Datenbasis zu erstellen. Die Ergebnisse der Evaluation sind hinsichtlich der Erfüllung der Anforderungen zu bewerten. Schließlich soll ein Prototyp zur Sturzdetektion auf einem mobilen Roboter implementiert werden.

1.4 Überblick

In Kapitel 2 werden die für diese Arbeit benötigten Grundlagen und Hintergründe erläutert. Kapitel 3 enthält einen Überblick über bekannte Verfahren zur Sturzdetektion (Abschnitt 3.1), zur merkmalsbasierten Detektion von Personen (Abschnitt 3.2) und Objekten in 3D-Daten (Abschnitt 3.3) und die Bewertung der vorgestellten Verfahren hinsichtlich ihrer Eignung für den in dieser Arbeit betrachteten Anwendungsfall (Abschnitt 3.4). Auf Basis der Bewertungsergebnisse wird ein neuer Ansatz zur Detektion gestürzter Personen über eine mobile Roboterplattform entwickelt. Kapitel 4 beschreibt das Konzept des neuen Verfahrens und geht auf Details der Implementierung ein. In Kapitel 5 wird die durchgeführte Evaluation des neuen Ansatzes und die erreichten Ergebnisse vorgestellt. Kapitel 6 fasst die Ergebnisse schließlich kurz zusammen und gibt einen Ausblick für zukünftige Arbeiten.

Kapitel 2

Grundlagen

2.1 Objektklassifikation

Bei der Klassifikation werden Objekte anhand bestimmter Merkmale (engl. Features) durch einen Klassifikator in verschiedene Klassen eingeteilt. In dieser Arbeit wird ein binäres Klassifikationsproblem betrachtet, d. h. die Daten werden einer von zwei Klassen zugeordnet.

2.1.1 Klassifikatoren

Als Klassifikator können maschinelle Lerntechniken verwendet werden, die mit Hilfe repräsentativer Beispieldaten eine Entscheidungsregel erlernen. Da in dieser Arbeit die Klassenzugehörigkeit der Trainingsdaten bekannt ist, kann auf Methoden des überwachten Lernens (engl. supervised learning) zurückgegriffen werden [Alpaydin, 2008].

Nearest Neighbour

Zur Klassifikation eines unbekannten Objekts über die Nearest Neighbour Methode werden die Klassen der aus den Trainingsdaten bekannten k nächsten Nachbarn des Objekts betrachtet. Als Distanzmaß wird in der Regel die Euklidische Distanz verwendet. Die Klasse des unbekannten Objekts wird über eine Mehrheitsentscheidung bestimmt (Abb. 2.1 a). Das Nearest Neighbour Verfahren ist ein nicht-generalisierendes

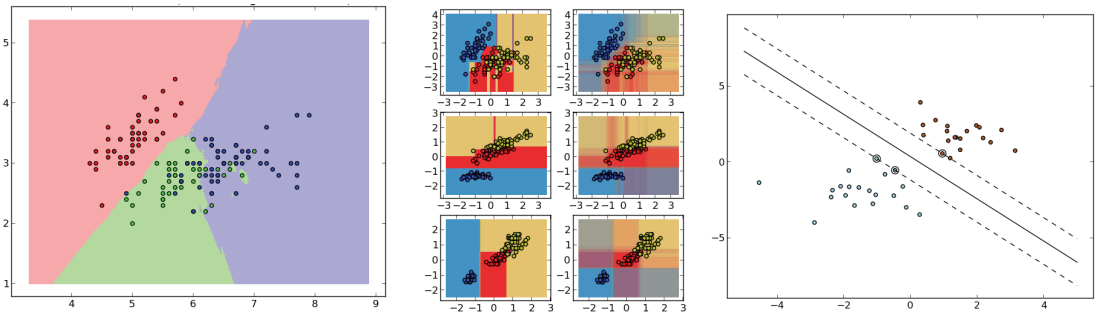


Abbildung 2.1: Entscheidungsgrenzen verschiedener Klassifikatoren [Pedregosa et al., 2011]

a) Nearest Neighbour für 3 Klassen mit $k = 15$. b) links: einfacher Entscheidungsbaumklassifikator, rechts: Random Forests. Jeweils für 3 Klassen c) binäre Support Vector Machine. Die Hyperebene liegt so, dass ein möglichst breiter, objektfreier Rand zu den Support Vectors entsteht.

maschinelles Lernverfahren, da es kein generelles Entscheidungsmodell erlernt, sondern einfach Instanzen der Trainingsdaten speichert. Die optimale Anzahl der berücksichtigten k Nachbarn ist datenabhängig. Generell unterdrückt ein größeres k den Einfluss von Rauschen, führt jedoch auch zu weniger präzisen Grenzen [Pedregosa et al., 2011]. In [Enas und Choi, 1986] wird $k \approx N^{2/8}$ oder $k \approx N^{3/8}$ empfohlen,wobei N die Anzahl der verwendeten Trainingsbeispiele beschreibt.

Random Forests

Ein Random Forests Klassifikator besteht aus mehreren, unkorrelierten Entscheidungsbäumen, die im Trainingsprozess zufällig gewachsen sind [Thom, 2012]. Ein Entscheidungsbaum besteht aus hierarchisch aufeinanderfolgenden Entscheidungsregeln, über die ein Modell zur Prädiktion der Klassenzugehörigkeit unbekannter Daten erstellt werden kann. Beim Random Forests werden die Entscheidungsregeln auf Basis einer zufällig ausgewählten Teilmenge der Trainingsdaten bestimmt. In jedem Knoten der Bäume werden zudem die Merkmale auf denen die Entscheidung basiert ebenfalls zufällig gewählt. Als Entscheidungsregel wird schließlich in jedem Knoten die Aufteilung genommen, die die meisten der zufällig ausgewählten Trainingsdaten richtig klassifi-

ziert. In der Detektionsphase kann die Klasse eines unbekannten Objekts entweder über eine Mehrheitsentscheidung der einzelnen Entscheidungsbäume oder über die Mittelung ihrer Prädiktionssicherheiten bestimmt werden (Abb. 2.1 b) [Pedregosa et al., 2011].

Support Vector Machine

Die Support Vector Machine (SVM) fällt in die Kategorie der mit Kernels arbeitenden Klassifikationsmethoden. Kernel Methoden projizieren die Daten in einen höherdimensionalen Raum, um in diesem eine einfachere Trennung der Klassen zu ermöglichen [Camastra und Vinciarelli, 2008]. Als Kernelfunktionen werden in der Regel:

- lineare Kernels: $k(x, y) = \langle x, y \rangle$
- polynomiielle Kernels: $k(x, y) = \langle x, y \rangle^d$, $d > 0$
- Kernels mit radialer Basisfunktion (RBF): $k(x, y) = \exp(-\gamma \|x - y\|^2)$, $\gamma > 0$

verwendet [Pedregosa et al., 2011].

Beim Training einer SVM wird in dem hochdimensionalen Merkmalsraum nach einer Hyperebene gesucht, die die Klassen der über Vektoren repräsentierten Trainingsobjekte trennt und einen maximalen Abstand zu den am nächsten liegenden Vektoren hat (Abb. 2.1 c). Diese werden als Support Vectors bezeichnet. Der beim Training entstehende breite, objektfreie Rand um die Hyperebene hilft, dass auch Objekte, die nicht genau den Trainingsdaten entsprechen, zuverlässig klassifiziert werden [Steinwart und Christmann, 2008].

AdaBoost

Beim AdaBoost, kurz für Adaptive Boosting, wird ein starker Klassifikator aus der gewichteten Kombination mehrerer schwacher Klassifikatoren gebildet. Jeder schwache Klassifikator muss dabei nur eine Entscheidungen treffen, die geringfügig besser ist als eine zufällige Wahl. In einer Folge von Trainingsrunden $t = 1, \ldots, T$ wird jeweils der schwache Klassifikator $h_t(e)$ ausgewählt, der die gewichteten Trainingsdaten e am besten klassifiziert. Hierzu wird im, in dieser Arbeit betrachteten 1-dimensionalen

Fall, pro Trainingsrunde für jede Dimension $j = 1, \ldots, M$ des Featurevektors f der Schwellwert θ_j und die Parität p_j bestimmt, mit denen der geringste Fehler bei der Klassifikation der Trainingsdaten über das Merkmal f_j gemacht wird. Über alle Dimensionen wird die Kombination $< f_j, p_j, \theta_j >$ als schwacher Klassifikator $h_t(e)$ der Runde t ausgewählt, die den geringsten Fehlerwert aufweist. Über eine Gewichtungsfunktion bekommen Trainingsbeispiele, die von den in den Vorrunden ausgewählten schwachen Klassifikatoren falsch klassifiziert wurden, bei der Fehlerberechnung der nächsten Runde ein größeres Gewicht.

Jeder schwache Klassifikator entscheidet mit:

$$h_j(e) = \begin{cases} +1 & \text{wenn } p_j f_j(e) < p_j \theta_j \\ -1 & sonst \end{cases} \tag{2.1}$$

über die Klassenzugehörigkeit eines Merkmals. Die Entscheidungsregel des starken Klassifikators ist schließlich über:

$$H(e) = \text{sign}(F(e)), \quad F(e) = \sum_{t=1}^{T} \alpha_t h_t(e) \tag{2.2}$$

gegeben, wobei α_t die im Training ermittelte Klassifikationssicherheit des schwachen Klassifikators $h_t(e)$ beschreibt [Arras et al., 2007].

2.1.2 Beurteilung von Klassifikationsalgorithmen

Um die Leistung eines Klassifikationsalgorithmus beurteilen zu können, muss eine Evaluation mit Daten durchgeführt werden, die nicht zum Klassifikatortraining verwendet wurden.

Es existiert eine Reihe von Leistungsmaßzahlen, über die die Zuverlässigkeit eines Klassifikators quantifiziert und die Leistung mehrerer Klassifikationsalgorithmen verglichen werden kann. Ausgangspunkt aller Maßzahlen ist die Anzahl der richtig positiv (True Positive, TP), der falsch positiv (False Positive, FP), der falsch negativ (False Negative, FN) und richtig negativ (True Negative, TN) detektierten Objekte. Diese werden über den Vergleich der Detektionsergebnisse mit einer Ground-Truth Annotation bestimmt und in Form einer Konfusionsmatrix dargestellt (Tab. 2.1) [Alpaydin, 2008].

		Tatsächliche Klasse	
		1	0
Ermittelte Klasse	1	True Positive (TP)	False Negative (FN)
	0	False Positive (FP)	True Negative (TN)

Tabelle 2.1: Konfusionmatrix einer Klassifikationsevaluation

Das einfachste Maß zur Evaluation der Klassifikationsqualität ist die Genauigkeit (engl. Accuracy). Diese beschreibt den Anteil korrekt klassifizierter Objekte an der Gesamtobjektzahl:

Accuracy (AC):

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

Da bei der Berechnung nicht zwischen falsch positiv und falsch negativ detektierten Objekten unterschieden wird, kann anhand der Accuracy nur die globale Güte des Klassifikators abgeschätzt werden. Je nach Anwendungsfall sind die beiden möglichen Fehlklassifikationen jedoch unterschiedlich wichtig. Daher wird ein Klassifikator häufig über seine Treferquote (engl. Recall) und seine Genauigkeit (engl. Precision) bewertet:

Recall (RC) und **Precision** (PR):

$$RC = \frac{TP}{TP + FN}, \quad PR = \frac{TP}{TP + FP}, \tag{2.4}$$

wobei Recall die Wahrscheinlichkeit beschreibt, dass ein gesuchtes Objekt detektiert wird und Precision die Wahrscheinlichkeit, dass ein detektiertes Objekt dem gesuchten entspricht.

Um die Bedeutung von Recall und Precision unterschiedlich zu gewichten, kann der $\mathcal{F}_\beta$-Wert mit unterschiedlichen Gewichtungsfaktoren β verwendet werden:

$\mathcal{F}_\beta$-Wert ($\mathcal{F}_\beta$):

$$\mathcal{F}_\beta = \frac{(1 + \beta^2) \cdot PR \cdot RC}{\beta^2 \cdot PR + RC} \tag{2.5}$$

$\mathcal{F}_1$ entspricht dem harmonischen Mittel aus Recall und Precision und kann wie die Accuracy als globales Gütemaß des Klassifikators eingesetzt werden. $\mathcal{F}_2$ gewichtet den Recall-Wert doppelt und bei $\mathcal{F}_{0.5}$ wird entsprechend der Precision-Wert doppelt gewichtet.

In der Praxis existiert häufig ein *Trade-off* zwischen Precision und Recall. Da dieser in der Regel mit veränderten Parametern des Klassifikationsalgorithmus manipuliert werden kann, wird zur Beurteilung eines Klassifikationsalgorithmus häufig eine **ROC-Kurve** verwendet, in der für jeden Precision-Wert der zugehörige Recall-Wert abgetragen wird. Die Position, an der der Recall- und Precision-Wert gleich ist, wird als **Equal Error Rate** (EER) bezeichnet. Die Fläche unter der Kurve entspricht der **Average Precision** (AP) und kann über:

$$AP = \int_0^1 PR(RC)dRC \tag{2.6}$$

bestimmt werden. Die EER und die AP sind weitere Möglichkeiten die globale Güte eines Klassifikators anzugeben.

Alle vorgestellten Leistungsmaßzahlen können Werte von $0 - 1$ annehmen, wobei höhere Werte eine bessere Detektorleistung bedeuten [Scharkow, 2012].

2.2 Grundlagen 3D-Daten

Die bei 3D-Daten verfügbaren Tiefeninformationen weisen bei der Verwendung zur visuellen Objektdetektion vor allem Vorteile bezüglich der Hintergrundsegmentierung auf. Im Gegensatz zu 2D-Daten können in 3D-Daten auch Objekte mit einer dem Hintergrund ähnlichen Farbgebung segmentievt werden. Bezüglich der Detektion gestürzter Personen ist zudem Vorteilhaft, dass auf Infrarotbasis arbeitende 3D-Kameras zwar nur in Indoor-Bereichen eingesetzt werden können, hier jedoch tagsüber wie auch nachts. Zudem bieten 3D-Daten einen gesteigerten Schutz der Privatsphäre, da sie keine visuellen Informationen enthalten, über die auf die Identität der beobachteten Person rückgeschlossen werden kann [Mastorakis und Makris, 2012].

Die Tiefeninformationen können in Form 2-dimensionaler Tiefenkarten oder 3-dimensionaler Punktwolken repräsentiert werden. Eine Tiefenkarte ist ein Bild, in dem

der Wert jedes Pixels die Tiefe des Szenenpunktes relativ zur Aufnahmeposition beschreibt. Über interne und externe Kalibrierungsparameter können diese in absolute Tiefenwerte transfomiert und als Sammlung von 3D-Koordinaten in einer Punktwolke dargestellt werden. Einige der im Folgenden vorgestellten Aufnahmetechniken nehmen neben den Tiefendaten die RGB-Daten der beobachteten Szene auf. Solche Daten werden als RGB-D Daten bezeichnet.

2.2.1 Aufnahme von 3D-Daten

Für die Aufnahme der Tiefeninformationen einer Szene stehen verschiedene Techniken zur Verfügung [Bernin, 2011]:

Laserscanner

Laserscanner tasten ihre Umgebung zeilen- oder rasterartig mit einem Laserstrahl ab. Die Oberflächengeometrie der betrachten Objekte wird über die Pulslaufzeit, Phasendifferenz im Vergleich zu einer Referenz oder durch Triangulation von Laserstrahlen erfasst. Um eine 3D Repräsentation zu erstellen, wird der Laserscanner in der Regel auf einer PTU (Pan Tilt Unit) angebracht und die 2-dimensionalen Scanlinien mehrerer Messungen zusammengefasst.

Stereo-Kameras

Stereo-Kameras nehmen eine Szene mit zwei benachbarten Kameras gleichzeitig auf. Über die Distanz zweier in den stereoskopischen Halbbildern korrespondierender Punkte kann mit Hilfe der Epipolargeometrie auf die Tiefe des Pixels rückgeschlossen werden.

Time-of-Flight Kameras

Time-of-Flight (TOF) Kameras messen die Abstände innerhalb einer Szene über die Laufzeit des von der Kamera gesendeten Lichts im Infrarotbereich. TOF-Kameras bieten jedoch nur eine sehr geringe Auflösung.

Abbildung 2.2: Kinect-Sensorleiste [LTD, 2013] und Infrarotmuster [STANFORD, 2013]
a) Die Kinect besitzt eine RGB- und eine IR-Kamera, sowie einen Projektor zur projektion eines IR-Musters. b) Die Kinect zählt zu den aktiven Triangulationssystemen, da sie die Tiefenstruktur einer Szene über die Differenz zwischen gesendetem und aufgenommenem Muster analysiert.

Aktive Triangulationssysteme

Aktive Triangulationssysteme projizieren ein Infrarotlicht mit einem bekannten Muster in die Szene, die wiederum mit einer infrarotempfindlichen Kamera observiert wird. Über die Differenz zwischen gesendetem und aufgenommenem Muster wird die Tiefenstruktur der Szene analysiert. Die bekannteste, auf dem aktiven Triangulationssystem arbeitende 3D-Kamera ist die von Microsoft im Jahr 2010 auf den Markt gebrachte **Kinect** (Abb. 2.2). Durch die Kinect steht erstmals der breiten Masse ein kostengünstiges System zur Aufnahme von 3D-Daten zur Verfügung. Ursprünglich als Hardware zur Steuerung von Videospielen entwickelt, haben viele Forscher das Potential der Kinect zur Detektion von Personen und Objekten in Anwendungen außerhalb des Videospiel-Sektors erkannten. Die Kinect enthält eine von der Firma PrimeSense [LTD, 2013] entwickelte Tiefensensorik.

2.2.2 Segmentierung von 3D-Daten

Um in 3D-Daten einzelne Objekte detektieren zu können, muss die Punktwolke in einzelne Segmente (engl. Cluster) segmentiert werden. Mit Hilfe eines Klassifikators

lassen sich die Cluster anschließend als Objekt einer bestimmten Klasse klassifizieren [Jaspers, 2012].

Modellbasierte Segmentierung

Bei der modellbasierten Segmentierung werden Objekte segmentiert, die bestimmten parametrischen Modellen entsprechen. Das können beispielsweise Ebenen, Zylinder oder Kugeln sein. Eines der bekanntesten modellbasierten Segmentierungsverfahren ist der RANSAC-Algorithmus (**Ran**dom **Sa**mple **C**onsensus). Dieser iterativ arbeitende Ansatz verwendet zur Bestimmung der Modellparameter lediglich so viele zufällig ausgewählte Punkte, wie zur Berechnung der Modellparameter nötig sind (im Fall einer Ebene drei Punkte). Anschließend wird aus den zufällig gewählten Punkten das Modell berechnet und die Anzahl der das Modell unterstützenden Punkte bestimmt. Liegt die Distanz eines Punktes zum Modell unter einem gegebenen Schwellwert t_{RANSAC}, so unterstützt er dieses. Schließlich werden die Modellparameter ausgewählt, deren Modell die meisten unterstützenden Punkte hat.

Die modellbasierte Segmentierung lässt sichr zur Detektion der Bodenfläche in einer Punktwolke einsetzen. Als Modell bietet sich eine Ebene an, deren Normalenvektor parallel zur y-Achse liegt [Rusu und Cousins, 2011].

Clustering

Beim Clustering werden alle Punkte, deren Ähnlichkeit unter einem gegebenen Schwellwert t_{Clust} liegt, zu einem Cluster zusammengefasst. Als Ähnlichkeitsmaß kann die Euklidische Distanz:

$$d_{\text{EuclDist}} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{2.7}$$

zwischen den Punkten $\boldsymbol{p}_1 = (x_1, y_1, z_1)$ und $\boldsymbol{p}_2 = (x_2, y_2, z_2)$ verwendet werden. Über die Größe des Schwellwerts t_{Clust} wird bestimmt, ob viele kleine Cluster ($t_{\text{Clust}} \to 0$) oder wenige große Cluster ($t_{\text{Clust}} \to \infty$) generiert werden. Das Euklidische Clustering ist ein einfaches, aber recheneffizientes Segmentierungsverfahren. Bei der Verwendung eines eindimensionales Distanzmaßes wird auch von einem Jump Distance Clustering (JDC) gesprochen.

2.2.3 Berechnung von Oberflächennormalen in 3D-Daten

Über die Orientierung der Oberflächennormalen lassen sich wichtige Eigenschaften der Oberfläche geometrischer Objekte beschreiben. Die Orientierung der Oberflächennormalen an einem bestimmten Punkt wird über den Vektor, der an diesem Punkt senkrecht zur Oberfläche steht, beschrieben. Zur Bestimmung des Vektors wird für jeden Punkt $\boldsymbol{p}_i$ eine Ebene an seine k nächsten Nachbarn angepasst. Der Normalenvektor der Ebene wird als Eigenvektor $\boldsymbol{n}_i$ mit dem kleinsten Eigenwert $\lambda_{i,0}$ der Kovarianzmatrix $C_i \in \mathbb{R}^{3\times3}$ der von den k Nachbarn gebildeten Nachbarschaft P_i bestimmt [Holz et al., 2011]. Die Anzahl der bei der Berechnung der Oberflächennormalen berücksichtigten Nachbarn k entscheidet dabei, wie viele feine Details der Oberflächenbeschaffenheit über die Normalen repräsentiert werden [Rusu und Cousins, 2011]. Neben der Angabe einer festen Anzahl von Nachbarn ist es in der Praxis auch üblich einen Radius anzugeben, innerhalb dem alle Punkte als Nachbarn berücksichtigt werden.

Kapitel 3

State of the Art

3.1 Sturzdetektion

Bekannte Ansätze zur Detektion gestürzter Personen sind grundsätzlich in zwei Kategorien zu unterteilen: Verfahren, die den aktiven Sturzprozess detektieren und solche, die nach einem Sturz liegende Personen detektieren. Die Ansätze zur Detektion des aktiven Sturzes können weiter nach der Art des verwendeten Sensors unterteilt werden:

- Tragbare Sensoren
- Externe Umgebungssensoren
- Audiobasierte Verfahren
- Bildbasierte Verfahren

Abbildung 3.1 zeigt ein Strukturgramm der im Folgenden betrachteten Verfahren. Eine Bewertung der Verfahren hinsichtlich ihrer Eignung für den in dieser Arbeit betrachteten Anwendungsfall erfolgt in Abschnitt 3.4.

3.1.1 Aktiver Sturz: Tragbare Sensoren

Tragbare Sensoren zur Detektion des aktiven Sturzprozesses basieren auf Beschleunigungssensoren, die bei einer Beschleunigung über einem festgesetzten Schwellwert

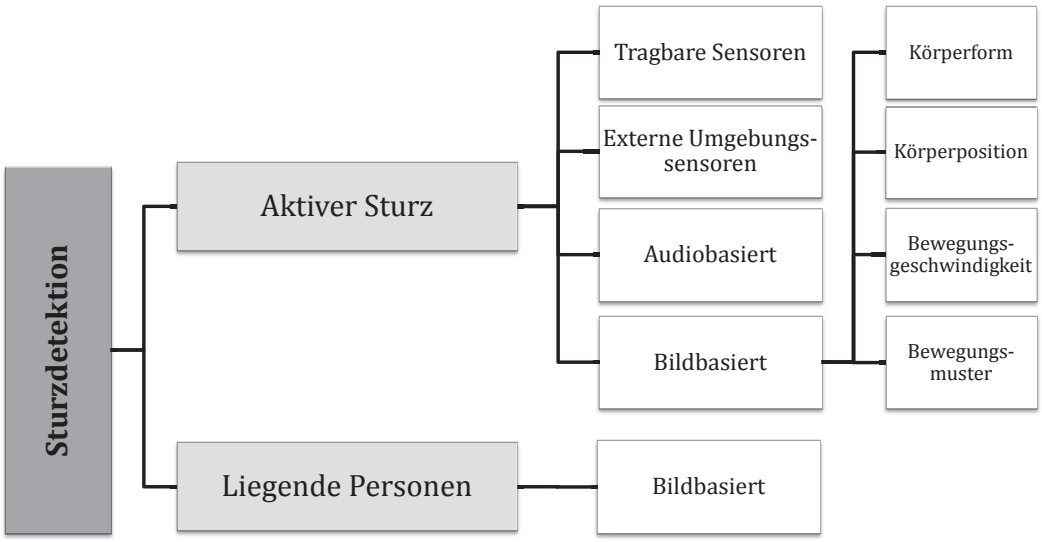


Abbildung 3.1: State of the Art der Sturzdetekion
Strukturgramm der vorgestellten Verfahren

Richtung Boden oder bei einer Abweichung der Sensorbewegung von einem erlernten Bewegungsmuster automatisch einen Alarm auslösen. Als Erweiterung der von vielen älteren Personen bereits genutzten Armbänder und Ketten mit einem Notfallknopf, bieten die Produkte mit einer automatischen Detektion des Sturzes den Vorteil, dass auch dann Hilfe angefordert wird, wenn das Betätigen des Notfallknopfes durch die gestürzte Person nicht möglich ist, z. B. bei einer durch den Sturz hervorgerufenen Bewusstlosigkeit. Für einen zuverlässigen Schutz muss der Sensor jedoch dauerhaft getragen werden. Dies wird vor allem von demenzkranken Personen häufig vergessen. Zudem generieren die Geräte eine hohe Anzahl von Fehlalarmen [Litvak et al., 2008].

Zu den tragbaren Sensoren zählt das von Philips patentierte *AutoAlert*-System [Electronics, 2012], das ähnlich einem Kettenanhänger um den Hals getragen wird (Abb. 3.2 a). Durch das Trageband entsteht jedoch ein zusätzliches Risiko der Strangulation, vor allem für Nutzer die im Rollstuhl sitzen. Dieses Risiko besteht bei dem vom Fraunhofer IIS entwickelten *MotionSENS*-System [IIS, 2011] nicht. Dieses kann dank seiner geringen Dicke wie ein Pflaster getragen oder in die Kleidung integriert werden (Abb. 3.2 b). Das System ist jedoch noch nicht kommerziell verfügbar.

a) b) c)

Abbildung 3.2: Sensoren zur Detektion eines aktiven Sturzes
*Tragbare Sensoren: a) Das AutoAlert-System von Philips wird wie ein Kettenanhänger getragen [*Electronics*, 2012]. b) Das MotionSENS-System vom Fraunhofer IIS kann ähnlich wie ein Pflaster getragen oder in die Kleidung integriert werden [*IIS*, 2011]. Externe Umgebungssensoren: c) Intelligente Fußmatte SensFloor der Future Shape GmbH [*Future-Shape*, 2010a]*

3.1.2 Aktiver Sturz: Externe Umgebungssensoren

Im Rahmen des vom Bundesministerium für Bildung und Forschung geförderten Verbundprojekt *SensFloor* [Future-Shape, 2010b] wird aktuell an der Entwicklung eines intelligenten Fußbodens gearbeitet, der das Bewegungsverhalten von Personen erkennt und analysiert, um im Fall eines Sturzes automatisch Hilfe anfordern zu können. Im Unterschied zu tragbaren Sensoren ist dieser externe Umgebungssensor nicht intrusiv und auch für Demenzpatienten geeignet, da kein Gerät dauerhaft mit sich getragen werden muss. Nachteilig ist der zurzeit noch bestehende Eingriff in das individuelle Wohnungsdesign, da das System aktuell von der Future Shape GmbH nur als max. 100 cm breite Matte in zwei verschiedenen Oberflächenausführungen verfügbar ist (Abb. 3.2 c) [Future-Shape, 2010a]. Für die Zukunft ist eine Integration des Systems in bzw. unter den normalen Fußbodenbelag geplant. Ein nachträglicher Einbau des Systems ist dann jedoch mit einem hohen Aufwand und hohen Kosten verbunden.

3.1.3 Aktiver Sturz: Audiobasiert

Der Sturz einer Person erzeugt in der Regel ein typisches Geräuschmuster. Dieses kann mit Hilfe maschineller Lerntechniken von anderen Geräuschen unterschieden werden.

Doukas et al. [Doukas und Maglogiannis, 2008] verwenden zur Aufnahme mit Mikrofonen und Bewegungssensoren ausgestattete Sensoren, die am menschlichen Körper (z. B. am Fuß) befestigt werden. Über eine SVM wird zwischen drei Bewegungen unterschieden: laufen, rennen und fallen. Nachteilig ist jedoch, dass die verwendeten Sensoren am Körper getragen werden und somit intrusiv sind.

Litvak et al. [Litvak et al., 2008] verwenden daher Sensoren, die in einer Ecke des Raumes platziert werden und neben Audiodaten die Vibration des Bodens über Bewegungssensoren aufnehmen. Zur Sturzdetektion wird zunächst die Energie der Bewegungssensordaten in einem festen Zeitfenster analysiert. Im Fall eines Vibrationsevents werden Features aus den Audio- und Bewegungssensordaten extrahiert und über ein Mustererkennungsverfahren klassifiziert. Zur Evaluation wurden Experimente mit einer Rettungspuppe durchgeführt. Das Verfahren erreicht hierbei Detektionsraten von 96%, allerdings sind die Sturzgeräusche einer Rettungspuppe mit denen einer echten Person nicht vergleichbar [Popescu und Mahnot, 2009].

Die Problematik, realistischen Soundbeispiele stürzender Personen zu erhalten, lösen Popescu et al. [Popescu und Mahnot, 2009] mit der Entwicklung eines Klassifikators, der für das Training nur Beispieldaten einer Klasse (z. B. nur Geräusche die keinen Sturz beschreiben) benötigt. Als Sensor werden drei vertikal angeordnete, an einer Wand montierte Mikrophone verwendet. Die vertikale Anordnung ermöglicht die Verwendung von Informationen über die Höhe der Schallquelle, was zu einer Reduktion der Falsch-Positiv Rate führt. Diese ist für die praktische Anwendung des Systems dennoch zu hoch.

3.1.4 Aktiver Sturz: Bildbasiert

Die auf Bilddaten basierenden Verfahren zur Detektion des aktiven Sturzes bestehen aus zwei Schritten: die Detektion der stehenden Person und dem Tracking dieser [Willems et al., 2009]. Als Datenbasis dienen 2D- oder 3D-Bilddaten, die in der Regel von statischen Kameras aufgenommen werden. Daher kann zur Detektion der stehenden Person auf bekannte Verfahren zur Hintergrundsubtraktion und Bewegungsdetektion zurückgegriffen werden. Der aktive Sturzprozess wird anschließend über:

- die **Veränderung der Körperform**,
- die **Körperposition**,
- die **Bewegungsgeschwindigkeit** oder
- das **Bewegungsmuster**

der getrackten Person detektiert. Die zugrunde liegende Annahme hierbei ist, dass ein Sturz charakterisiert wird durch die den Übergang von einer vertikalen zu einer horizontalen Pose mit einer unüblichen Steigerung der Geschwindigkeit und hierdurch ein typisches Bewegungsmuster entsteht [Zambanini et al., 2010]. Die im Folgenden vorgestellten Verfahren bilden eine repräsentative Auswahl aktueller Techniken zur bildbasierten Detektion des aktiven Sturzprozesses. Für einen Überblick über ältere Verfahren wird auf [Willems et al., 2009] verwiesen.

Analyse der Veränderung der Körperform in 2D-Daten

Zur Detektion des aktiven Sturzprozesses über die Veränderung der Körperform in 2D-Daten, werden einfache geometrische Hüllkörper an die Silhouette der getrackten Person angepasst. Über deren geometrischen Eigenschaften kann analysiert werden, ob sich eine Person in einer aufrechten oder liegenden Position befindet.

Williams et al. [Williams et al., 2007] verwenden als geometrischen Hüllkörper eine Bounding Box. Über deren Seitenverhältnis ($Breite/Höhe > 0,8$) wird eine gestürzte Personen detektiert. Um zu verhindern, dass eine stürzende Person von anderen Objekten verdeckt wird, fusionieren Williams et al. die Detektionsergebnisse mehrerer Kameras. Zusätzlich zu dem Seitenverhältnis der Bounding Box analysieren Vaidehi et al. [Vaidehi et al., 2011] den Neigungswinkel der detektierten Person (Abb. 3.3). Ein Sturz wird durch ein Seitenverhältnis $Breite/Höhe > 1$ und einen Neigungswinkel $\theta > 45°$ charakterisiert. Beide Verfahren können jedoch keine Stürze parallel zur optischen Achse der Kamera erkennen, da sich in diesem Fall das Seitenverhältnis der Bounding Box nicht stark genug ändert.

Rougier et al. [Rougier et al., 2007] verwenden daher als geometrischen Hüllkörper eine Ellipse. Diese bietet eine bessere Approximation der menschlichen Silhouette

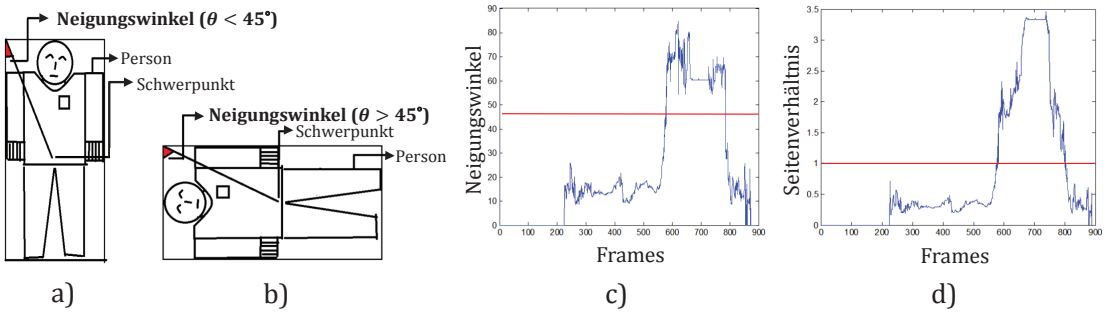


Abbildung 3.3: Seitenverhältnis und Neigungswinkel der Bounding Box einer Person [Vaidehi et al., 2011]
Neigungswinkel einer a) aufrecht stehenden Person und b) einer gestürzten Person. Videosequenz mit einem Sturz: c) Verlauf des Neigungswinkels d) Verlauf des Seitenverhältnis der Bounding Box. (Die rote Linie entspricht dem Schwellwert des Sturzdetektors)

(Abb. 3.4 a). Über ein Motion History Image (MHI) (Abb. 3.4 b) detektieren Rougier et al. sich schnell bewegende Objekte. Für diese wird die Standardabweichung der Ellipsenorientierung σ_θ, sowie die Standardabweichung des Verhältnisses der Hauptachsen σ_ρ bestimmt. Stürzt eine Person senkrecht zur optischen Achse der Kamera, ändert sich die Orientierung der Ellipse signifikant und σ_θ ist groß. Stürzt eine Person hingegen parallel zur optischen Achse der Kamera, ändert sich das Achsenverhältnis deutlich und σ_ρ ist groß. Bei einer laufenden Person sind beide Werte klein.

Um Stürze von sturzähnlichen Bewegungen, wie dem schnellen Hinlegen auf ein Sofa, unterscheiden zu können, schlagen Williams et al., wie auch Rougier et al. vor, bestimmte Bereiche im Raum als normale Inaktivitätszonen zu definieren. In diesen lösen sturzähnliche Bewegungen keinen Alarm aus.

Da eine manuelle Markierung solcher Inaktivitätszone bei jeder Positionsveränderung der Möbel oder der Kamera neu durchgeführt werden müsste, stellen Shoaib et al. [Shoaib et al., 2011] ein Verfahren zur automatischen Segmentierung der Szene in Aktivitäts- und Inaktivitätszonen vor. Über die geometrische Konstellation des Kopfes, der Körpermitte und der Füße klassifizieren Shoaib et al. die Bewegung einer Person in eine von vier Kategorien (Stehen/Laufen, Sitzen, Knien/Beugen, Liegen). Bereiche

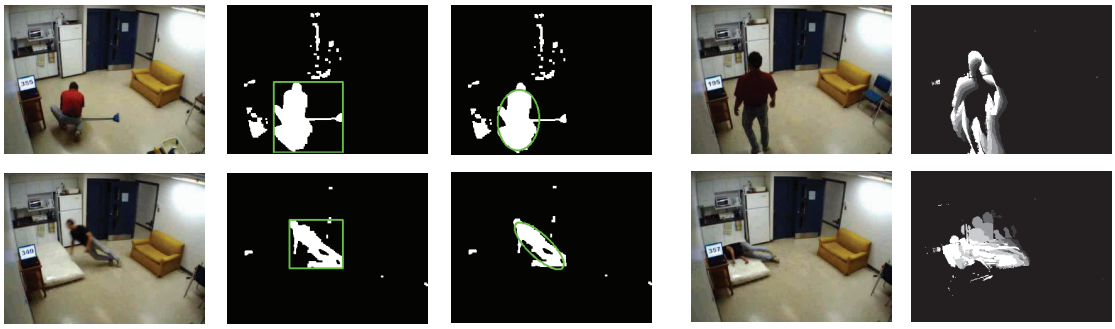


Abbildung 3.4: Körperform und MHI einer stürzenden Person [Rougier et al., 2007]

a) Ein ellipsenförmiger Hüllkörper bietet eine bessere Approximation an die menschliche Silhouette als eine Bounding Box. b) Über ein Motion History Image (MHI) können Informationen über die Bewegung in einer Videosequenz extrahiert werden.

in denen Personen laufen oder stehen werden als Aktivitätszonen markiert, die mit Sitzbewegungen als Inaktivitätszonen. Ein Liegen löst nur dann einen Sturzalarm aus, wenn sich die Person außerhalb einer Inaktivitätszone befindet.

Analyse der Körperposition in 3D-Daten

Um in 3D-Daten den aktiven Sturzprozess über die Körperposition zu detektieren, wird in der Regel der Abstand des Mittelpunkts der getrackten Person zum Boden analysiert. Liegt dieser unter einem bestimmten Schwellwert, gilt ein Sturz als detektiert. Die im Folgenden vorgestellten Ansätze unterscheiden sich vor allem in der Art der Hintergrundsegmentierung, der Bodendetektion und des verwendeten Mittelpunkts.

Jansen et al. ermitteln in den von einer kalibrierten TOF-Kamera (Abschnitt 2.2.1) aufgenommenen Tiefenkarten die 3D-Position des Zentrums einer an die Silhouette einer Person angepassten Ellipse. Über diese wird die Pose der Person in 3 Kategorien eingeteilt: stehen, sitzen ($z_h \leq 70$ cm) oder liegen ($z_h \leq 35$ cm). Diraco et al. [Diraco et al., 2010] analysieren neben der Distanz des Schwerpunkts der detektierten Person zum Boden auch das Verhalten der Person nach einem möglichen Sturz. Liegt

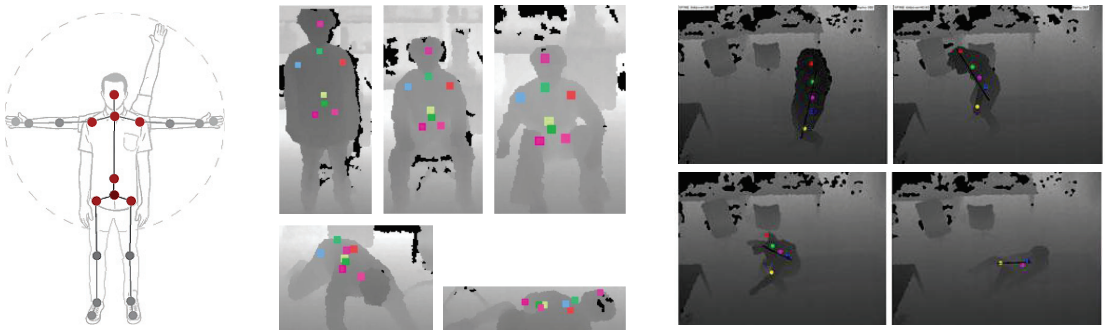
der Schwerpunkt unter 40 cm, gefolgt von einer Inaktivität der getrackten Person für mindestens 4 Sekunden, wird von einer gestürzten Person ausgegangen.

Statt einer TOF-Kamera wird von Rougier et al. [Rougier et al., 2011] zur Aufnahme der 3D-Daten erstmals die Tiefenkamera Kinect (Abschnitt 2.2.1) verwendet. Um nicht auf externe Kalibrierungsparameter angewiesen zu sein, bestimmen Rougier et al. den 3D-Schwerpunkt der über ihre Bewegung detektierten Person relativ zur Kamera und nicht in Weltkoordinaten wie Jansen et al. und Diraco et al. Über das V-Disparity Image [Zhao et al., 2007] detektieren Rougier et al. die Lage der Bodenfläche, die dadurch als Ebenengleichung ebenfalls relativ zur Kamera vorliegt. So kann der Abstand zwischen Boden und dem 3D-Schwerpunkt der Person über eine einfache Punkt-Ebene-Distanz bestimmt werden. Um auch stürzende Personen detektieren zu können, die am Ende des Sturzes durch andere Objekte verdeckt sind, kombinieren Rougier et al. dieses Abstandskriterium mit der 3D-Geschwindigkeit des fallenden Körpers. Liegt die Geschwindigkeit eines plötzlich verschwindenden Objekts in der letzten Sekunde vor dem Verschwinden über einem aus Trainingsdaten ermittelten Schwellwerts wird das Objekt als gestürzte Person klassifiziert.

Analyse der Veränderung der Körperform in 3D-Daten

Über das Microsoft Kinect SDK [Research, 2012] kann neben den Rohdaten, wie dem RGB- und Tiefenbild, auch auf die Ergebnisse der Kinect-internen Posenschätzung zurückgegriffen werden. Dieses extrahiert für zwei der maximal sechs im Tiefenbild getrackten Personen ein komplettes Skelett bestehend aus 20 sogenannten *Joints*, die die Position charakteristischer Körperteile und Gelenke beschreiben (Abb. 3.5 a). Über die relative Lage der *Joints* können Rückschlüsse auf die Veränderung der Körperform während einer getrackten Bewegung gezogen werden.

Zhang et al. [Zhang et al., 2012] verwenden zur Sturzdetektion nur die 8 *Joints* auf dem Kopf und Torso. Diese weisen bei einer stehenden oder sitzenden Person eine stabile Struktur auf. Bei einer stürzenden Person ist diese Struktur hingegen instabil (Abb. 3.5 b). Zhang et al. nutzen diese Eigenschaft, um aus den Deformationskosten zwischen zwei Strukturen Merkmale zur Detektion stürzender Personen zu extrahie-



a) b) c)

Abbildung 3.5: Sturzdetektion über die Kinect-interne Posenschätzung
*a) Die Kinect extrahiert für max. zwei Personen ein Skelett mit 20 Joints [*Microsoft*, 2012]. b) Die 8 Joints auf Kopf und Torso bilden bei stehenden und sitzenden Personen (oben) eine stabile Struktur. Bei stürzenden Personen (unten) ist diese Struktur hingegen instabil [*Zhang *et al., 2012]. c) Orientierung der Hauptachse einer Person auf Basis der Joints für Kopf, Schulter, Wirbelsäule, Hüfte und Knie [*Planinc *und* Kampel*, 2012]*

ren. Um auch Stürze außerhalb des Tiefenbereichs der Kinect (ab ~ 4 m) detektieren zu können, greifen Zhang et al. zusätzlich auf das von der Kinect aufgenommene RGB-Videomaterial zurück. In diesem wird das Seitenverhältnis der die Person umgebenden Bounding Box über eine Zeitsequenz in einem Histogramm abgetragen und als Klassifikationsmerkmal für Stürze außerhalb der Reichweite des Tiefensensors genutzt.

Auch Planinc et al. greifen in [Planinc und Kampel, 2012] auf die von der Kinect-internen Posenschätzung bereitgestellten Skelettinformationen zurück. Auf Basis der *Joint*-Positionen bestimmen Planinc et al. die Orientierung der Hauptachse der getrackten Person, sowie die Höhe der Wirbelsäule. Über die zusätzliche Höheninformation kann zwischen Stürzen und sturzähnlichen Bewegungen unterschieden werden, bei denen die Orientierung der Hauptachse zwar gleich ist, die Höhe jedoch nicht (z. B. das Hinlegen in ein Bett). Für die Bestimmung der Hauptachse wird eine Gerade an den *Joint* des Kopfes, der Schulter, Wirbelsäule, Hüfte und des Knies angepasst (Abb. 3.5 c).

Die Detektionsleistung der Verfahren von Zhang et al. und Planinc et al. sind abhängig von der Qualität der Kinect-internen Posenschätzung. Laut Mastorakis et al.

[Mastorakis und Makris, 2012] scheitert diese jedoch während eines Sturzes häufig und kann, wenn die gestürzte Person liegt, nicht wiederhergestellt.

Analyse der Bewegungsgeschwindigkeit in 3D-Daten

Um nicht von der in Sturzsituationen instabil arbeitenden Posenschätzung der Kinect abhängig zu sein, greifen Mastorakis et al. [Mastorakis und Makris, 2012] nur auf die vom Kinect-internen Personentracking zur Verfügung gestellte Breite, Höhe und Tiefe der die getrackte Person umgebenden Bounding Box zurück. Diese Parameter werden auch während eines Sturzes stabil extrahiert. Durch die Berücksichtigung der Tiefe der Bounding Box können auch stürzende Personen detektiert werden, die parallel zur optischen Achse der Kamera fallen. Hierzu summieren Mastorakis et al. die Breite und Tiefe zu einem Wert. Liegen die Geschwindigkeiten der Höhe bzw. der Breiten-Tiefen-Kombination in einer Sequenz aus acht Videoframes über den aus Trainingsdaten ermittelten Schwellwerten, wird die Bewegung als möglicher Sturz klassifiziert. Ein Alarm wird jedoch erst ausgelöst, wenn die Geschwindigkeit der Höhe anschließend für mind. zwei Sekunden unter einem bestimmten Schwellwert liegt und damit eine Inaktivität der Person analysiert wird.

Analyse von Bewegungsmustern

Ähnlich dem vorgestellten Verfahren von Williams et al. [Williams et al., 2007] verwenden auch Anderson et al. [Anderson et al., 2006] als geometrischen Hüllkörper eine Bounding Box, um über die Veränderung des Seitenverhältnisses den aktiven Sturzprozess zu detektieren. Eine Analyse dieses Merkmals über die Zeit zeigt, dass durch einen Sturz ein typisches Muster erzeugt wird, welches sich deutlich von alltäglichen Bewegungen unterscheidet (Abb. 3.6 a). Anderson et al. verwenden Sequenzen mit gehenden, knienden und fallenden Personen, um ein Hidden Markov Model (HMM) zu trainieren und über dieses in der Anwendung einzelne Videosequenzen zu klassifizieren.

Um unnormales Verhalten wie einen Sturz von den alltäglichen Bewegungen einer Person unterscheiden zu können, stellt Cai in [Cai, 2010] eine mobile Roboterplattform vor

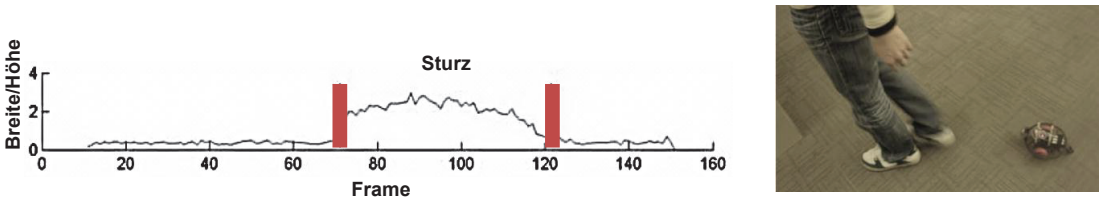


Abbildung 3.6: Analyse von Bewegungsmustern
*a) Das Seitenverhältnis der Bounding Box einer stürzenden Person [*Anderson *et al., 2006]. b) Die mobile Roboterplattform Dog 1.0 verfolgt die Person und detektiert unnormales Verhalten über das wahrgenommene Bewegungsmuster [*Cai*, 2010]*

(*Dog 1.0*, Abb. 3.6 b), die einer Person folgt, die wahrgenommenen Bewegungsmuster in Form einer Sequenz aus Symbolen speichert und von der Norm abweichendes Bewegungsverhalten detektiert. Zusätzlich kann der Roboter mit drei vertikal angebrachten Infrarotsensoren die Umrisse einer auf dem Boden liegenden Person messen und diese mit Hilfe von Mustererkennungsverfahren klassifizieren. Der Nachteil des *Dog 1.0* ist das, von manchen Nutzern als aufdringlich empfundene Verfolgen der Person.

3.1.5 Detektion liegender Personen

Um zur Sturzdetektion eine mobile Roboterplattform einsetzen zu können, die den Nutzer nicht dauerhaft beobachtet, muss anstatt des aktiven Sturzprozesses die liegende Person erkannt werden.

Der von Wang et al. [Wang et al., 2011] vorgestellte Ansatz zur Detektion liegender Personen basiert auf bekannten Ansätzen zur körperteilbasierten Detektion stehender Personen [Felzenszwalb et al., 2008] und zur Posenschätzung [Ferrari et al., 2009]. Wie in [Felzenszwalb et al., 2008] beschreiben Wang et al. eine Person über ein körperteilbasiertes, deformierbares Modell (Abb. 3.7 a) - b). Dieses besteht aus einem, die grobe äußere Form repräsentierenden *Root*-Filter und mehreren *Part*-Filtern, die wichtige Formdetails erfassen. Jedem *Part*-Filter ist ein Raum-Modell zugehörig. Dieses beinhaltet die ideale Position des *Part*-Filters relativ zum *Root*-Filter, sowie

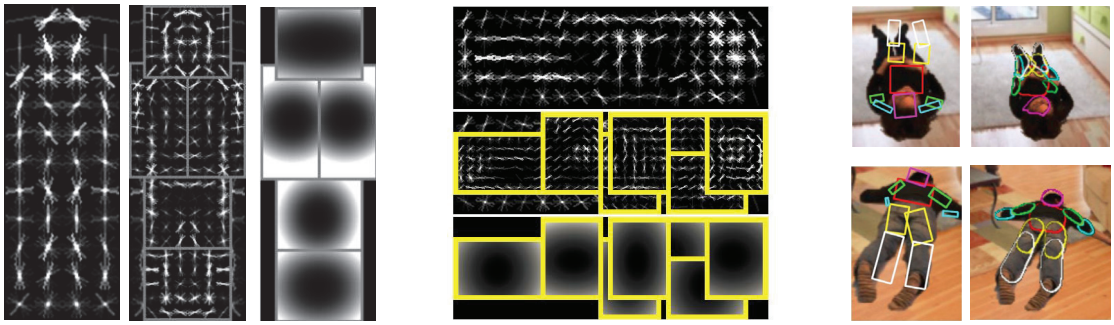


Abbildung 3.7: Detektion liegender Personen und Posenschätzung nach [Wang et al., 2011]

a) Das von Felzenszwalb et al. [Felzenszwalb et al., 2008] zur Detektion stehender Personen verwendete Modell. b) Eins der insg. acht von Wang et al. zur Detektion liegender Personen verwendeten Modelle. Von links nach rechts a) bzw. oben nach unten b): der Root-Filter, die Part-Filter und die zugehörige Deformationsfunktion. c) Ergebnisse der Posenschätzung: initiale Anordnung der Teilobjekte (links), finale Segmente (rechts)

die bei einer Abweichung von der idealen Position entstehenden Deformationskosten. Da die Körperorientierung liegender Personen wesentlich vielfältiger als die stehender Personen ist und dieses auf Grund von perspektivischen Verzerrungen zu signifikanten Unterschieden in der Erscheinung einer Person führt, verwenden Wang et al. eine Kombination aus acht getrennten körperteilbasierten Modellen, wovon jedes Modell auf eine Klasse von Körperorientierungen (eingeteilt in 45° Intervalle) spezialisiert ist. Alle Modelle führen die Detektion unabhängig voneinander aus. Zur Posenschätzung (Abb. 3.7 c) erweitern Wang et al. den Ansatz von Ferrari et al. [Ferrari et al., 2009]. Diese verwenden zur Repräsentation des menschlichen Körpers das *Pictorial Structure*-Modell [Felzenszwalb et al., 2004]. Dieses beschreibt eine Sammlung von semantischen, in einer Baum-Struktur angeordneten Teilobjekten, von denen jedes Teilobjekt die Erscheinung eines bestimmten Körperteils (z. B. einzelner Gliedmaßen) modelliert.

Wang et al. beobachteten in ihren Experimenten eine starke Korrelation zwischen der detektierten Anordnung der *Part*-Filter und der Konfiguration der Teilobjekte des

Merkmal	***TP***	***FP***	***FN***
Seitenverhältnis	81%	9%	10%
Torsozentrum	78%	10%	12%

Tabelle 3.1: Detektionsergebnisse von Qingcong Lv [Lv, 2011]

Pictorial Structure-Modell. Ein ähnlicher Zusammenhang zwischen statistischen und semantischen Körperteilen wurde bereits beim Implicite Shape Model (ISM) [Leibe et al., 2008] beobachtet. Wang et al. nutzen diesen Zusammenhang aus, um die Effizienz und Genauigkeit der Posenschätzung über eine Initialisierung des Algorithmus mit der aus der Detektionsphase bekannten Anordnung der *Part*-Filter zu steigern.

Der Vergleich der Detektionsleistungen des spezialisierten Modells von Wang et al. ($AP = 74\%$) mit einem zur Detektion stehender Personen trainierten und anschließend rotierten Modell ($AP = 37\%$) betont, wie stark der Einfluss der bei liegenden Personen entstehenden perspektivischen Verzerrung auf die Erscheinung einer Person ist.

Auch Qingcong Lv adaptiert in [Lv, 2011] ein bekanntes Verfahren zur Detektion stehender Personen auf die Detektion gestürzter Personen. Über die von Bourdev et al. [Bourdev und Malik, 2009] vorgestellten *Poselets* detektiert Qingcong Lv zunächst alle im Bild abgebildeten Personen. Über die *Poselets* ist es zudem möglich, bestimmte Körperteile zu detektieren. Qingcong Lv nutzt dies, um neben der Bounding Box den Torso einer Person zu finden. Über das Seitenverhältnis der Bounding Box und dem Abstand des Torsozentrums zum unteren Rand der Bounding Box entscheidet Qingcong Lv, ob die detektierte Person steht oder liegt. Um gestürzte Personen von solchen unterscheiden zu können, die sich in ein Bett oder auf eine Couch gelegt haben, berücksichtigt Qingcong Lv zusätzlich die Interaktion der detektierten Personen mit solchen Objekten. Hierzu greift Qingcong Lv auf ein 2-Komponenten Mixture Modell [Maji et al., 2011] zurück, um die verschiedenen Charakteristika der Person und des Objekts (Bett oder Couch) zu modellieren.

Qingcong Lv evaluiert beide zur Sturzdetekion verwendeten Merkmale (Seitenverhältnis und Höhe Torsozentrum) getrennt. Wie Tab. 3.1 zeigt, sind die Detektionsleistungen beider Merkmale ähnlich. Angaben zur Detektionsleistung der kombinierten Merkmale werden nicht gemacht.

3.2 Personendetektion in 3D-Daten

Da es bisher kein Verfahren gibt, dass liegende Personen auf Basis von 3D-Daten detektiert, werden im Folgenden bekannte Verfahren zur Detektion stehender Personen in 3D-Daten vorgestellt und in Abschnitt 3.4 hinsichtlich ihrer Eignung zur Detektion liegender Personen bewertet. Dem Fokus dieser Arbeit entsprechend werden nur featurebasierte Verfahren betrachtet. Weitere Möglichkeiten sind die Personendetektion über Geodätische Extrema [Plagemann, 2010], [Schwarz, 2011] oder modellbasierte Ansätze [Xia, 2011], [Krishnamurthy, 2011].

Featurebasierte Verfahren erlernen das Modell des zu detektierenden Objekts über repräsentative Trainigsdaten aus denen mit Hilfe von Merkmalsextraktionsverfahren die charakteristischen Eigenschaften des Objekts bestimmt werden. Bekannte, zur Persondetektion in 3D-Daten verwendete Features, sind in zwei Kategorien zu unterteilen:

- Geometrische und statistische Features
- Histogrammbasierte Features

Die histogrammbasierten Features lassen sich weiter über die verwendete Datenbasis (Tiefenkarte oder Punktwolke) kategorisieren. Abbildung 3.8 zeigt ein Strukturgramm der im Folgenden betrachteten Features.

3.2.1 Histogrammbasierte Features

Histogrammbasierte Features basieren auf der Annahme, dass Objekte der selben Klasse (z. B. Personen) über die Häufigkeitsverteilung eines bestimmten Merkmals beschreibbar sind.

Histogrammbasierte Features: Datenbasis Tiefenkarte

Motiviert durch die Leistung des histogrammbasierten HOG-Deskriptors von Dalal und Triggs [Dalal und Triggs, 2005] bei der Detektion stehender Personen in 2D-Daten, existieren zahlreiche Adaptionen dieses Verfahrens für die Anwendung auf Tiefenkarten.

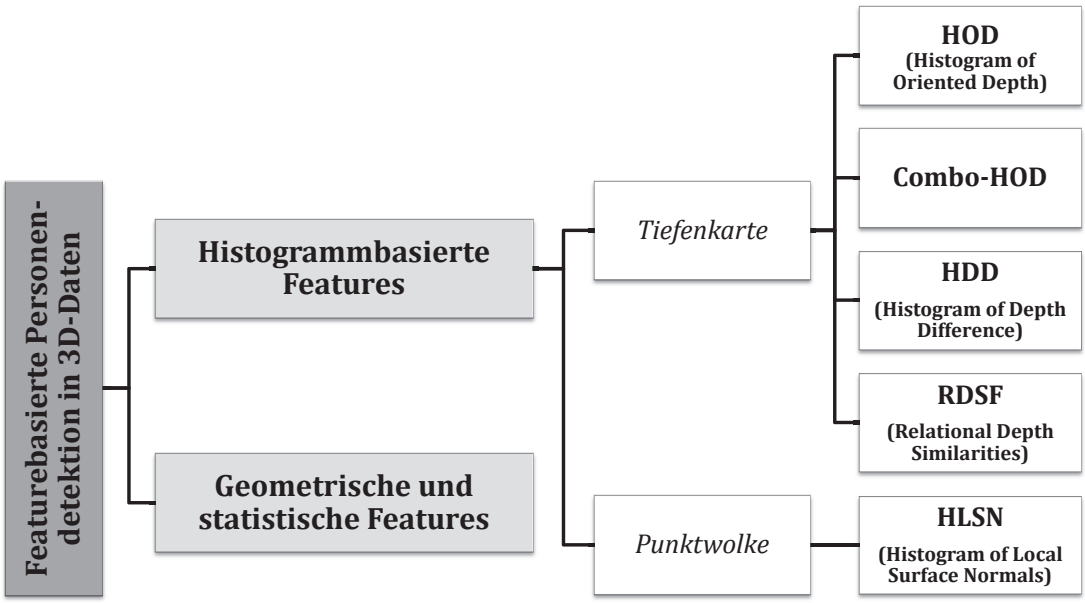


Abbildung 3.8: Featurebasierte Personendetektion in 3D-Daten
Strukturgramm der vorgestellten Verfahren

Der HOG-Deskriptor besteht aus normalisierten Histogrammen, die die Häufigkeitsverteilung der Gradientenorientierung beschreiben. Zur Erstellung des Deskriptors wird ein $B \times H$ großer Bildausschnitt in $n \times n$ große Zellen aufgeteilt, für die jeweils ein 1-dimensionales Gradientenhistogramm mit b Bins bestimmt wird. Die Zellenhistogramme werden über $m \times m$ große, sich überlappende Blöcke normalisiert, wodurch pro Block ein Merkmalsvektor mit m^2 Elementen entsteht. Die Merkmalsvektoren aller Blöcke werden schließlich hintereinander zu einem k-elementigen Endmerkmalsvektor gereit und dieser zum Training einer linearen SVM verwendet.

Histogram of Oriented Depth (HOD) Das dem HOG-Deskriptor am stärksten ähnelnde histogrammbasierte Feature ist das in [Spinello und Arras, 2011] vorgestellte Histogram of Oriented Depth. Dieses charakterisiert die Form und Erscheinung von Personen über die Häufigkeitsverteilung lokaler Tiefenänderungen in der Tiefenkarte. Das HOD verfolgt zur Erstellung des Deskriptors dieselbe Prozedur wie das HOG. Anders als bei der Anwendung auf 2D-Daten muss in der Detektionsphase jedoch nicht der gesamte Bereich der Tiefenkarte auf allen Skalenstufen durchsucht wer-

den. Über die Tiefeninformationen kann dieser Prozess so geleitet werden, dass nur Suchfenstergrößen untersucht werden müssen, die für die Abbildung einer Person in der enstprechenden Tiefe realistisch sind. Das von Spinello et al. als *Depth-Informed Scale Space Search* bezeichnete Verfahren führt zu einer gesteigerten Effizienz und Genauigkeit.

Neben dem rein auf Tiefendaten basierenden HOD stellen Spinello et al. mit dem **Combo-HOD** ein Feature vor, dass die Informationen des Tiefen- und des RGB-Sensors der Kinect kombiniert. Beim Combo-HOD werden die Detektionsergebnisse eines auf RGB-Daten seperat arbeitenden HOG mit denen eines auf Tiefenkarten arbeitenden HOD gewichtet fusioniert, wodurch die Vorteile beider Datenbasen kombiniert werden.

Die von Spinello et al. durchgeführte Evaluation zeigt, dass die Verwendung von Tiefendaten über das HOD im Vergleich zu RGB-Daten zu besseren Detektionsergebnissen führt. Die Kombination von RGB- und Tiefeninformationen im Combo-HOD kann diese Ergebnisse nochmal leicht verbessern.

Histogram of Depth Difference (HDD) Auch Wu et al. zeigen in [Wu et al., 2011], dass Tiefendaten zur Detektion von Personen besser geeignet sind als reine RGB-Daten. Als Deskriptor stellen Wu et al. das Histogram of Depth Difference vor, das lokale Tiefenvarianzen beschreibt. Hierzu wird für jedes Pixel einer 8×8 großen Zelle die Tiefenvarianz in x- und y-Richtung bestimmt:

$$\Delta_x = \frac{D(x+1,y) - D(x-1,y)}{2}, \quad \Delta_y = \frac{D(x,y+1) - D(x,y-1)}{2}, \tag{3.1}$$

mit $D(x,y)$ als Tiefenwert an der Stelle (x,y). Anschließend wird für jede Zelle ein Zellenhistogramm mit Bins von $0° - 360°$ gebildet, welches analog zu [Dalal und Triggs, 2005] über überlappende Blöcke normalisiert wird.

Relational Depth Similarities Feature (RDSF) Deutlichere Unterschiede zum originalen HOG-Deskriptor weist das erstmals in [Ikemura und Fujiyoshi, 2010] vorgestellte Relational Depth Similarities Feature auf. Dieses extrahiert das relative Tiefenverhältniss zweier Regionen, beschrieben über die Distanz ihrer Tiefenhistogramme. Zur Berechnung des RDSF wird das Tiefenbild in quadratische Zellen aufgeteilt,

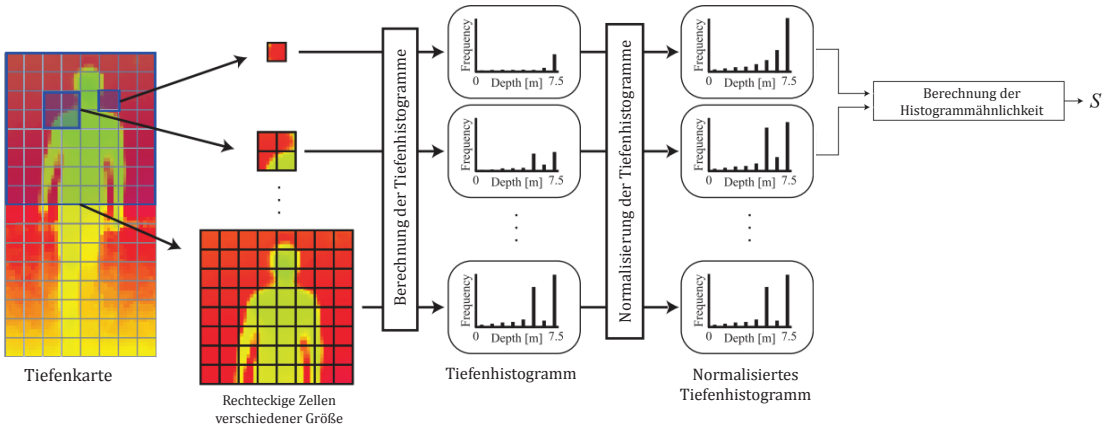


Abbildung 3.9: Relational Depth Similarities Feature (RDSF) [Ikemura und Fujiyoshi, 2010]

Zur Berechnung des RDSF wird das Suchfenster in rechteckige Zellen unterschiedlicher Größe aufgeteilt. Für jede Zelle wird ein normalisiertes Tiefenhistogramm berechnet, welches mit denen der anderen Zellen verglichen wird.

von denen jeweils zwei ausgewählt werden (Abb. 3.9). Aus den Tiefenwerten der Zellen wird jeweils ein Histogramm erstellt, normalisiert und die Ähnlichkeit der zwei normalisierten Tiefenhistogramme über die Bhattacharyya Distanz [Bhattacharyya, 1946] bestimmt. Für jedes Suchfenster werden die Distanzwerte aller möglichen Zellen-Zweier-Kombination ermittelt und in einen Featurevektor zusammengefasst. Ähnlich wie in [Spinello und Arras, 2011] reduzieren Ikemura et al. die Anzahl der zu überprüfenden Suchfenster, indem sie die Tiefendaten in Weltkoordinaten projizieren und nur ein Fenster der Größe 0.60×180 cm betrachten.

Auch die von Ikemeura et al. durchgeführte Evaluation zeigt, dass die Verwendung von Tiefen- im Vergleich zu RGB-Daten zu einer deutlichen Steigerung der Detektionsleistung führt.

Histogrammbasierte Features: Datenbasis Punktwolke

Wie in Abschnitt 3.4 beschrieben, ist das in diese Kategorie fallende Verfahren für den in dieser Arbeit betrachteten Anwendungsfall gut geeignet und wird im Folgenden daher detaillierter betrachtet.

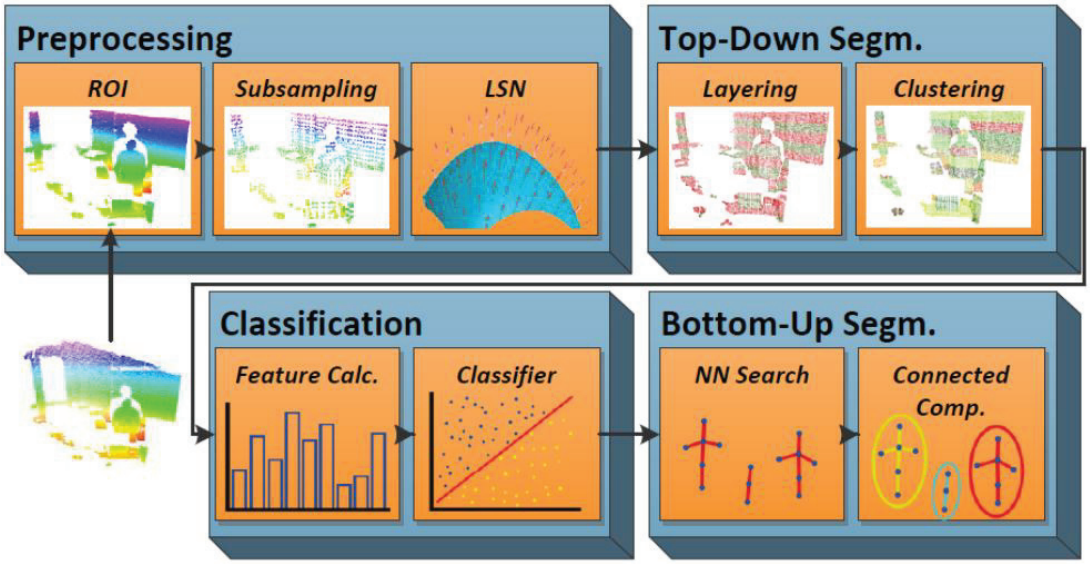


Abbildung 3.10: Personendetektion mit HLSN [Hegger et al., 2012]
Um über HLSNs Personen in 3D-Daten zu detektieren, führen Hegger et al. einen aus vier Phasen (blau) bestehenden Detektionsprozess durch. Jede Phase besteht aus weiteren Unterschritten (orange), die die eigentlichen Berechnungen auf den Eingangsdaten durchführen.

Histogram of Local Surface Normals (HLSN) Ein aus Punktwolken extrahiertes, histogrammbasiertes Feature ist das von Hegger et al. in [Hegger et al., 2012] vorgestellte Histogram of Local Surface Normals. Dieses basiert auf der lokalen Verteilung der Oberflächennormalen eines Objekts (Abschnitt 2.2.3). Die Motivation hinter der Verwendung von Oberflächennormalen ist, dass die Oberfläche einer Person eher unregelmäßig bzw. zylindrisch ist. Künstlich hergestellte Objekte, wie sie in der häuslichen Umgebung hauptsächlich vorkommen, haben im Gegensatz dazu eher eine regelmäßige Oberfläche. Diese Eigenschaft kann durch die Orientierung der Normalenvektoren einer Oberfläche beschrieben werden.

Um über HLSNs Personen in 3D-Daten zu detektieren, führen Hegger et al. einen aus vier Phasen bestehenden Detektionsprozess durch (Abb. 3.10). Die erste Phase beschreibt das ***Preprocessing*** der Punktwolke. In diesem wird die Größe der Eingangsdaten reduziert, um den Rechenaufwand in den folgenden Phasen zu verringern.

Da die Tiefendaten der Kinect in größeren Entfernungen sehr verrauscht sind und Personen in der Regel kleiner als 2 m sind, wird zunächst eine *Region of Interest* (*ROI*) definiert:

$$Pixel \in ROI, \text{wenn } 0.5 \text{ m} \leq \text{Tiefe} \leq 5.0 \text{ m und } 0.0 \text{ m} \leq \text{Höhe} \leq 2.0 \text{ m}.$$

Alle Punkte außerhalb der ROI werden aus der Punktwolke entfernt. Anschließend wird die Anzahl der übrig gebliebenen Punkte durch ein *Downsampling* reduziert, indem alle Punkte innerhalb einer $size_{grid} \times size_{grid} \times size_{grid}$ großen Zelle auf einen Punkt gemittelt werden. Hegger et al. verwenden eine Zellengröße von $size_{grid} = 3$ cm. Bevor die vorverarbeitete Punktwolke an die Segmentierungs-Phase gegeben wird, werden die lokalen Oberflächennormalen (engl. *Local Surface Normals* (*LSN*)) aller Punkte bestimmt.

Um in der auf die PreProcessing-Phase folgenden **Segmentierung** die Punktwolke mit möglichst geringem Rechenaufwand in einzelne Cluster zu segmentieren, verwenden Hegger et al. einen Top-Down Segmentierungsansatz. Hierzu wird die Punktwolke zunächst der Höhe nach in $M = 8$ jeweils $LH = 25$ cm große *Layer* aufgeteilt. Anschließend wird für jeden Layer l_j mit $j = 1 \ldots M$ über ein Euklidisches Clustering eine Sequenz kleiner Cluster $\mathbf{C} = \{c_1, \ldots, c_O\}$ generiert. Hegger et al. verwenden einen Distanzschwellwert von $t_{Clust} = 2 * size_{grid} = 6$ cm. Durch das dem feinstufigen Clustering vorgeschalteten Layering führt die einfache Euklidische Clustertechnik auch bei teilweise verdeckten Objekte, wie einer hinter einem Tisch sitzenden Person, zu keinen Problemen. Der vorgestellte Ansatz erstellt für beide Objekte mehrere kleine Cluster, die einzeln klassifiziert werden können.

Die Klassifikation der in der Segmentierungs-Phase generierten 3D-Cluster erfolgt in der **Klassifikations**-Phase. Diese beginnt mit einer *Feature-Berechnung*, bei der für jeden Cluster ein HLSN berechnet und mit zusätzlichen geometrischen Features zu einem Merkmalsvektor zusammengefügt wird. Da die meisten Klassifikatoren eindimensionale Eingangsdaten vorraussetzen, wird zur Bestimmung des HLSN für jede Achse der Normalenvektoren ($x, y,$ und z) ein seperates Histogramm mit $b = 7$ Bins erstellt. Zusätzlich wird die Breite, Tiefe, Anzahl der Punkte und die Distanz vom Cluster- zum Layerschwerpunkt als 22-tes bis 25-tes Element dem Featurevektor zuge-

Pose	Detektionsrate	Bewegung	Detektionsrate
Stehen	87,29%	keine Bewegung	87,29%
Sitzen	74,94%	Laufen	86,32%
Part. verdeckt	82,35%	Rennen	86,71%

Tabelle 3.2: Detektionsrate des auf HLSN-basierenden Verfahrens von Hegger et al. für verschiedene Posen und Bewegungen [Hegger et al., 2012]

fügt. Zur *Klassifikation* verwenden Hegger et al. einen Random Forests Klassifikator, der in Tests auf verschiedenen Datensätzen eine bessere Detektionsleistung erbracht hat als eine SVM oder AdaBoost.

Die letzte Phase des Detektionsprozesses von Hegger et al. beschreibt einen **Bottom-Up Segmentierungs**ansatz, der die Klassifikationsergebnisse der einzelnen Cluster zu einem Gesamtdetektionsergebnis fusioniert. Hierzu werden über einen *Nearest Neighbour*-Ansatz alle Cluster, deren Mittelpunktdistanz kleiner als $2 * LH = 0.5$ m ist, als zu einem Objekt zugehörig definiert. Ein Objekt gilt als Person, wenn mindesten drei einzelne Cluster des Objekts positiv als Person klassifiziert wurden.

Tabelle 3.2 zeigt die Detektionrate des Verfahrens von Hegger et al. für verschiedene Posen und Bewegungen. Da für das Klassifikatortraining nur Testdaten mit stehenden Personen verwendet wurden, können die beim Sitzen horizontal angeordneten Beine nicht detektiert werden, wodurch diese Pose deutlich schlechter detektiert wird. Auf Grund des layerbasierten Ansatzes kann das Verfahren von Hegger et al. auch mit partiell verdeckten Personen umgehen.

3.2.2 Geometrische und statistische Features

Geometrische und statistische Merkmale geben Informationen über die geometrische Form eines Objekts wieder. Sie ermöglichen es die Form, Lage und Größe eines Objekts zu kennzeichnen. Zudem können statistische Momente geometrisch interpretiert werden. Auf geometrischen und statistischen Features basierende Detektionsverfahren sind der menschlichen Wahrnehmung nachempfunden, da auch der Mensch Objekte vor allem über ihre Gestalt bzw. Kontur identifiziert [Hermes, 2005].

Nr.	Feature	Nr.	Feature
f_1	Breite	f_2	Gesamtanzahl Punkte
f_3	Rundheit	f_4	Linearität
f_5	Objektgrenzenlänge	f_6	Objektgrenzenregelmäßigkeit
f_7	Hauptwinkelabweichung	f_8	Hauptkrümmung
f_9	Quadratisches Splineinterpolation	f_{10}	Kubische Splineinterpolation
f_{11}	Standardabweichung bzgl. Schwerpunkt	f_{12}	Durchschn. Abweichung vom Median
f_{13}	Wölbung bzgl. Schwerpunkt	f_{14}	Radius
f_{15}	Verhältnis Hauptkomponenten	f_{16}	Größe Bounding Box
f_{17}	Größe Konvexe Hülle		

Tabelle 3.3: Geometrische und statistische Features zur Detektion von Personen in 3D-Punktwolken nach [Spinello et al., 2010]

Geometrische und statistische Features sind auch für die Detektion von liegenden Personen gut geeignet (Abschnitt 3.4). Da Teile der von Spinello et al. in [Spinello et al., 2010] und [Spinello et al., 2011] vorgestellten Ansätze in dem in dieser Arbeit entwickelten eigenen Ansatz verwendet werden (Kapitel 4), werden diese im folgenden Abschnitt detaillierter betrachtet. Weitere Arbeiten, die geometrische und statistische Feature zur Detektion von Personen verwenden sind [Arras et al., 2007], [Spinello et al., 2008], [Bajracharya et al., 2009] und [Navarro-Serment et al., 2010]. Die verwendeten Merkmale sind den im Folgenden vorgestellten jedoch ähnlich und werden daher nicht weiter im Detail betrachtet.

Spinello et al. stellen in [Spinello et al., 2010] einen körperteil- bzw. layerbasierten Ansatz vor, der vor allem die Problematik der partiellen Verdeckungen bei der Detektion von Personen adressiert. Ähnlich dem Verfahren von Hegger et al. [Hegger et al., 2012] unterteilen Spinello et al. eine stehende Person über verschiedene Höhenlevel in mehrere Layer (Abb. 3.11 a). Spinello et al. trainieren jedoch für jeden Layer einen spezialisierten Klassifikator. Hierzu wird für jeden, über ein Jump Distance Clustering (Abschnitt 2.2.2) entstehenden Cluster, ein Featurevektor bestehend aus 17 geometrischen und statistischen Merkmalen bestimmt (Tab. 3.3) und als positives Trainingsbeispiel verwendet. Nach einer *one-vs-all* Strategie werden alle Hintergrundsegmente,

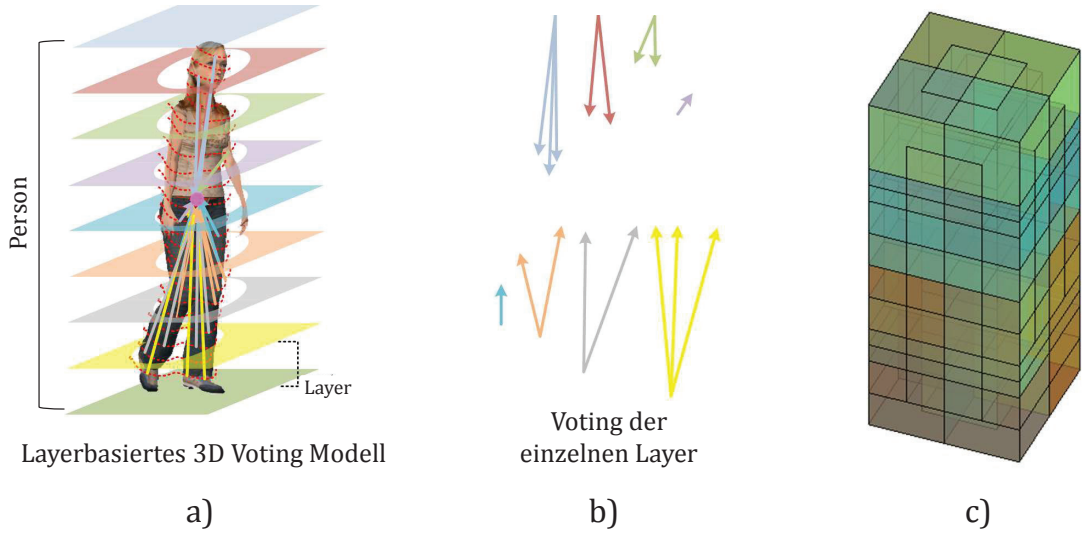


Abbildung 3.11: Personendetektion über Layer und Volumentesselierung
*a) Die Laserscandaten werden in mehrere Layer unterteilt, für die jeweils ein spezialisierter Klassifikator trainiert wird. b) In der Detektionsphase votet jeder Layer für den Personenmittelpunkt [*Spinello *et al., 2010]. c) Die aus Trainingsdaten erlernte optimale Volumentesselierung spiegelt die Anatomie des menschlichen Körpers wieder [*Spinello *et al., 2011].*

sowie alle Segmente der anderen Layer als negative Trainingsbeispiele verwendet. Als Klassifikator verwenden Spinello et al. AdaBoost mit 20 schwachen Klassifikatoren.

Zusätzlich zur Extraktion der Merkmale werden die geometrischen Beziehungen der Layer und ihrer Segmente analysiert. Für jeden Layer wird abhängig von der Versatzvariabilität seiner Segmente innerhalb der Trainingsdaten ein Gewicht ermittelt, das in der Detektionsphase den Anteil des Layers am Detektionsergebnis bestimmt. Zudem wird die Distanz des Layerzentrums zum Zentrum der Person ermittelt und in einem 3D-Personenmodell gespeichert.

In der Detektionsphase wird zunächst für jedes Segment der Eingangspunktwolke die Wahrscheinlichkeit bestimmt, dass das Segment zu einem der trainierten Layer gehört. Hierzu wird der Featurevektor des Segments über die korrespondierenden AdaBoost-Modelle der Layer klassifiziert. Die Wahrscheinlichkeiten werden mit dem in der Trainingsphase erstellten 3D-Personenmodell kombiniert und in einem Voting-Raum zu-

sammen gefasst (Abb. 3.11 a)-b). Lokale Maxima entsprechen dem Zentrum einer Person in 3D.

Die von Spinello et al. durchgeführte Evaluation zeigt, dass das Verfahren vor allem im Nahbereich (Entfernungen von 0 bis 10m) eine gute Detektionsleistung mit einer EER von bis zu 96% erreicht.

Motiviert von den guten Detektionsleitungen erweitern Spinello et al. in [Spinello et al., 2011] diesen Bottom-Up-Ansatz neben einer Methodik zum Tracken der detektierten Person, um eine Top-Down-Detektionsmethodik. Diese verwendet die Detektionen des Bottom-Up-Detektors als Eingangshypothese und verifiziert diese über ein erweitertes 3D-Personenmodell. Hierzu repräsentieren Spinello et al. die dreidimensionale Gestalt einer Person über ein tesselliertes Volumen, aus dem wiederum geometrische und statistische Features berechnet werden. Die Lernphase des Top-Down-Detektors besteht daher aus zwei Unterphasen: dem Lernen der besten Volumentesselierung, bei dem das Gesamtvolumen iterativ in kleinere Voxels unterteilt wird, und dem Lernen der Features, die aus den Voxels extrahiert werden.

Um die Suche nach der besten Volumentesselierung effizient zu gestalten, stellen Spinello et al. einige Bedingungen auf, die das Seitenverhältnis, die Form und die Größe der Voxels betreffen. Über einen iterativen Algorithmus werden zunächst Voxels aller die Vorbedingungen erfüllenden Größen erstellt, um anschließend zu testen, in welchen Kombinationen welche Voxels das gegebene Volumen lückenlos repräsentieren können und somit eine geeignete Tesselierung $\mathcal{T}_j$ darstellen. Für jedes Trainingsbeispiel entsteht so ein Set $\mathcal{T}$ mit möglichen Tesselierungen.

Anschließend wird für jeden Voxel $\mathcal{T}_j^i$ jeder Tesselierung $\mathcal{T}_j \in \mathcal{T}$ ein Vektor mit neun geometrischen und statistischen Features auf Basis der n in $\mathcal{T}_j^i$ enthaltenen Punkte $\mathcal{P} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ berechnet:

- f_1 - *Gesamtanzahl Punkte*: Kardinalität von $\mathcal{T}_j^i$, bezeichnet als n. $f_1(\mathcal{T}_j^i) = n$

- f_2 - *Rundheit*: Grad der Rundheit, bestimmt aus den Eigenwerten $\lambda_1, \lambda_2, \lambda_3$ der Kovarianzmatrix von $\mathcal{P}$. $f_2(\mathcal{T}_j^i) = 3\frac{\lambda_3}{\sum_i \lambda_i}$, mit $\lambda_1 > \lambda_2 > \lambda_3$.

- f_3 - *Flachheit*: Grad der Flachheit, bestimmt aus den Eigenwerten. $f_3(\mathcal{T}_j^i) = 2\frac{\lambda_2-\lambda_3}{\sum_i \lambda_i}$.

- f_4 - *Linearität*: Grad der Linearität, bestimmt aus den Eigenwerten. $f_4(\mathcal{T}_j^i) = \frac{\lambda_1-\lambda_2}{\sum_i \lambda_i}$.

- f_5 - *Standardabweichung bzgl. Schwerpunkt*: als Maß der Kompaktheit. $f_5(\mathcal{T}_j^i) = \sqrt{\frac{1}{n-1}\sum_i |\boldsymbol{x}_i - \bar{\boldsymbol{x}}|^2}$, mit $\bar{\boldsymbol{x}}$ als Schwerpunkt.

- f_6 - *Wölbung bzgl. Schwerpunkt*: als 4. zentralisierter Moment der Datenverteilung in $\mathcal{T}_j^i$. $f_6(\mathcal{T}_j^i) = \sum_i |\boldsymbol{x}_i - \bar{\boldsymbol{x}}|^4 / f_5(\mathcal{T}_j^i)$, mit $\bar{\boldsymbol{x}}$ als Schwerpunkt.

- f_7 - *Durschn. Abweichung vom Median*: als alternatives Maß der Kompaktheit. $f_7(\mathcal{T}_j^i) = \frac{1}{n}\sum_i |\boldsymbol{x}_i - \tilde{\boldsymbol{x}}|$, mit $\tilde{\boldsymbol{x}}$ als unabhängiger Median $\tilde{\boldsymbol{x}} = (\tilde{x}, \tilde{y}, \tilde{z})$.

- f_8 - *Normalisierter Fehler der Flachheit*: als alternatives Maß der Flachheit. Bestimmt über die Summe des quadratischen Fehlers einer an die Daten angepassten Ebene, normalisiert über n. $f_8(\mathcal{T}_j^i) = \sum_i^n (ax_i + by_i + cz_i + d)^2$, mit a, b, c, d als Ebenenparameter, bestimmt aus den Eigenwerten der Kovarianzmatrix.

- f_9 - *relative Gesamtanzahl Punkte*: als Maß der Punktdichte im Voxel, relativ zum Gesamtvolumen $\mathcal{B}$. $f_9(\mathcal{T}_j^i) = \frac{n}{\mathcal{B}}$

Dies führt zu Trainingsvektoren der Form

$$\boldsymbol{s} = (f_1^1, \ldots, f_9^1, f_1^2, \ldots, f_9^2, f_1^V, \ldots, f_9^V) \tag{3.2}$$

mit V als Gesamtanzahl aller Voxels im Set aller Tesselierungen $\mathcal{T}$, die zum Training 20 schwacher Klassifikatoren über AdaBoost verwendet werden. Dieses Training erfüllt beide Ziele der Trainingsphase. Zum einen werden die am besten zur Klassifikation geeigneten Features ausgewählt und zum anderen wird indirekt die optimale Tesselierung $\mathcal{T}_{opt}$ des Volumen ausgewählt. Indirekt, da $\mathcal{T}_{opt}$ über das Set von Voxels definiert wird, aus denen die besten Features extrahiert wurden. Abb. 3.11 c) zeigt die erlernte optimale Tessellierung. Interessanterweise spiegelt die erlernte Unterteilung die Anatomie

des menschlichen Körpers wieder. In der Detektionsphase wird schließlich die Featureberechnung nur für die Tesselierung $\mathcal{T}_{opt}$ durchgeführt und diese über die trainierten AdaBoost-Klassifikatoren klassifiziert.

Die von Spinello et al. durchgeführte Evaluation zeigt, dass der BUTD-Detektor im Nahbereich ($0 - 10$ m) leicht bessere Ergebnisse als der BU-Detektor (EER von bis zu 97.6%). In weiteren Entfernungen (15 m bzw. 20 m) ist eine deutliche Leistungssteigerung zu erkennen.

3.3 Objektdetektion in 3D-Daten

Da die Personendetektion eine Unterkategorie der allgemeineren Objektdetektion ist, werden im Folgenden bekannte, featurebasierte Verfahren zur Detektion von Objekten in 3D-Daten vorgestellt. Wie in Abschnitt 3.2.1 besprochen, bietet die Orientierung der Oberflächennormalen eine vielversprechende Unterscheidungsmöglichkeit zwischen Personen und anderen Objekten. Daher werden im Folgenden nur auf der Orientierung der Normalenvektoren basierende Verfahren betrachtet, die sich nach der verwendete Datenbasis (Tiefenkarte oder Punktwolke) kategorisieren lassen.

3.3.1 Datenbasis: Tiefenkarte

Histogram of Oriented Normal Vectors (HONV) Das in [Tang, 2011] vorgestellte Histogram of Oriented Normal Vectors ist dem in Abschnitt 3.2.1 vorgestellten HLSN sehr ähnlich. Im Gegensatz zum HLSN wird beim HONV die Orientierung der Oberflächennormale jedoch direkt aus der Tiefenkarte bestimmt (Abb. 3.12 a). Analog zum originalen HOG-Verfahren [Dalal und Triggs, 2005] verwendet Tang ein Sliding Window, das in $n \times n$ große Zellen aufgeteilt wird. In jeder Zelle wird für jeden Pixel die Orientierung des Normalenvektors $\mathbf{N}$ als Tupel aus Azimut- und Zenitwinkel (φ, θ) aus den Gradienten des Tiefenbilds bestimmt (Abb. 3.12 a)-b):

$$\varphi = \arctan\left(\frac{\partial d(x,y)}{\partial y} / \frac{\partial d(x,y)}{\partial x}\right) \tag{3.3}$$

$$\theta = \arctan\left(\left(\frac{\partial d(x,y)}{\partial y}\right)^2 + \left(\frac{\partial d(x,y)}{\partial x}\right)^2\right)^{\frac{1}{2}}. \tag{3.4}$$

Pro Zelle wird ein 2-dimensionales Histogramm über φ und θ gebildet (Abb. 3.12) c), das erst zellenintern und anschließend über mehrere benachbarte Zellen geglättet wird. Der finale HONV-Deskriptor wird schließlich durch die Aneinanderreihung aller Zellenhistogramme des Suchfensters erstellt.

Tang stellt zudem mit dem **RIHONV** eine rotationsinvariante Version des HONV vor, das unabhängig vom Betrachtungswinkel ist. Hierzu wird die dominante Oberflä-

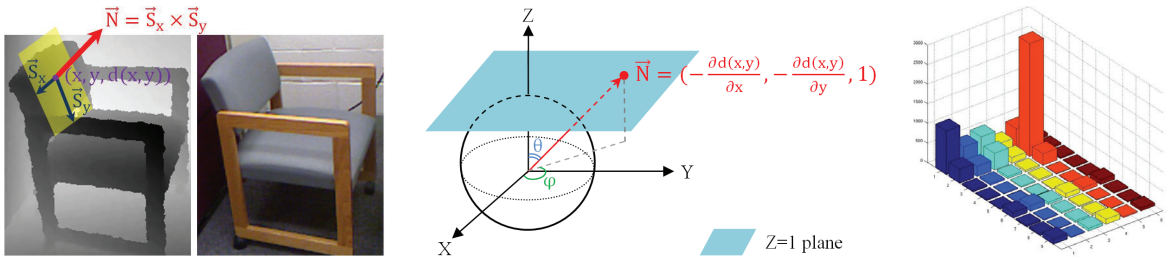


Abbildung 3.12: Histogram of Oriented Normal Vectors (HONV) [Tang, 2011] *a)-b)Die Orientierung des Normalenvektors* **N** *wird als Tupel aus Azimut- und Zenitwinkel* (φ, θ) *aus den Gradienten des Tiefenbilds bestimmt. c) Pro Zelle wird ein 2-dimensionales Histogramm über* φ *und* θ *gebildet.*

chenorientierung des Objekts ermittelt und alle anschließend bestimmten Zellenhistogramme über diese normalisiert.

3.3.2 Datenbasis: Punktwolke

Wie in Abschnitt 3.4 beschrieben, sind die in diese Kategorie fallende Verfahren für den in dieser Arbeit betrachteten Anwendungsfall gut geeignet und werden im Folgenden daher detaillierter betrachtet.

Point Feature Histogram (PFH) Ein vor allem in der Objekterkennung zum Matching zweier Punktwolken verwendetes Feature ist das von Rusu [Rusu, 2009] entwickelte Point Feature Histogram. Das PFH analysiert für jeden Punkt p_q einer Punktwolke die Differenz zwischen der Orientierung seines Normalenvektors und dem Normalenvektor aller k Nachbarn. Als Nachbar gelten alle Punkte $\boldsymbol{p}_{ki}$, die innerhalb einer Kugel mit dem Radius r um den Punkt $\boldsymbol{p}_q$ liegen (Abb. 3.13 a). Zur Berechnung der Differenz zwischen zwei Punkten $\boldsymbol{p}_t$ und $\boldsymbol{p}_s$ und deren Normalen $\boldsymbol{n}_t$ und $\boldsymbol{n}_s$ wird ein festes Koordinatensystem an einem der Punkte definiert (Abb. 3.13 b):

$$\boldsymbol{u} := \boldsymbol{n}_s, \qquad \boldsymbol{v} := \boldsymbol{u} \times \frac{(\boldsymbol{p}_t - \boldsymbol{p}_s)}{||\boldsymbol{p}_t - \boldsymbol{p}_s||_2}, \qquad \boldsymbol{w} := \boldsymbol{u} \times \boldsymbol{v} \tag{3.5}$$

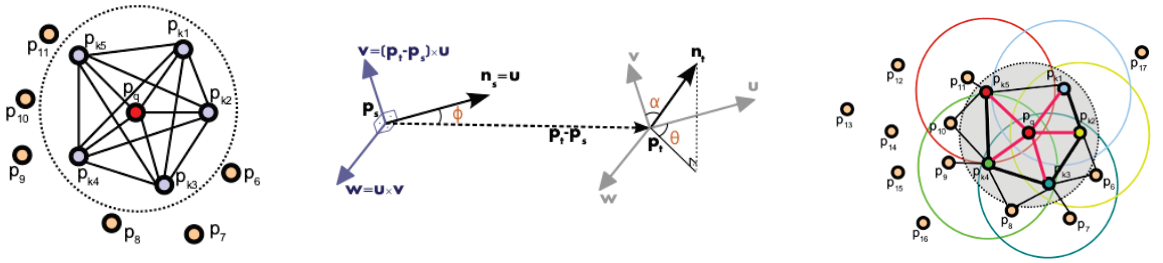


Abbildung 3.13: (Fast) Point Feature Histogram ((F)PFH) [Rusu, 2009]
PFH: a) Einflussbereich zur Berechnung des PFH am Punkt $\boldsymbol{p}_q$. b) Zur Berechnung des PFH verwendetes Koordinatensystem und Winkel. FPFH: c) Einflussbereich zur Berechnung des FPFH am Punkt $\boldsymbol{p}_q$

Unter der Verwendung dieses Koordinatensystems beschreibt Rusu die Differenz zwischen den zwei Normalen $\boldsymbol{n}_t$ und $\boldsymbol{n}_s$ über drei Winkelmaße:

$$\alpha = \boldsymbol{v} \cdot \boldsymbol{n}_t, \qquad \phi = \boldsymbol{u} \cdot \frac{(\boldsymbol{p}_t - \boldsymbol{p}_s)}{d}, \qquad \theta = \arctan(\boldsymbol{w} \cdot \boldsymbol{n}_t, \boldsymbol{u} \cdot \boldsymbol{n}_t) \tag{3.6}$$

und einem Abstandsmaß d als Euklidsche Distanz zwischen den Punkten $\boldsymbol{p}_t$ und $\boldsymbol{p}_s$. Das Quadrupel $< \alpha, \phi, \theta, d >$ wird für jedes Punktepaar in der Nachbarschaft berechnet und in ein Histogramm mit b^4 Bins eingetragen. Je nach Anwendungsfall wird die Distanz d nicht berücksichtigt. Die Berechnung der PFH für eine Punktwolke mit n Punkten hat eine Komplexität von $O(nk^2)$. Vor allem für sehr große oder dichte Punktwolken ist dies zu hoch, um eine echtzeitfähige Implementierung zu erreichen.

Fast Point Feature Histogram (FPFH) Daher stellt Rusu mit dem Fast Point Feature Histogram eine vereinfachte Version des PFH vor, deren Berechnungsalgorithmus eine reduzierte Komplexität von $O(nk)$ aufweist. Zur Berechnung des FPFH wird zunächst analog zum PFH für jeden Punkt p_q einer Punktwolke das Tupel $< \alpha, \phi, \theta >$ zwischen sich und seinen k Nachbarn berechnet und daraus ein Simplified Point Feature Histogram (SPFH) bestimmt. Anschließend wird für jeden Punkt p_q das FPFH berechnet, indem die SPFH der benachbarten Punkte gewichtet summiert werden:

$$FPFH(\boldsymbol{p}_q) = SPFH(\boldsymbol{p}_q) + \frac{1}{k} \sum_{i=1}^{k} \frac{1}{w_k} \cdot SPFH(\boldsymbol{p}_k). \tag{3.7}$$

Das Gewicht w_k repräsentiert die Distanz zwischen dem betrachten Punkt $\boldsymbol{p}_q$ und dem Nachbarn $\boldsymbol{p}_k$. Da die SPFH der Nachbarn auch auf Basis von Punkten berechnet wurden, die außerhalb der eigentlichen Nachbarschaft von $\boldsymbol{p}_q$ liegt, bezieht das FPFH einen größeren Nachbarschaftsbereich ein als das PFH (Abb. 3.13 c). Bei der Berechnung des Histogramms wird beim FPFH im Gegensatz zum PFH eine dekorreliertes Schema angewendet, d.h. die Werte jedes Differenzmaß werden unabhängig von den anderen Maßen in b Bins eingeteilt.

Surface Entropie (SURE) Eine weitere Möglichkeit die Komplexität des PFH/FPFH-Algorithmus zu senken ist die Reduktion der n Punkte, für die der Deskriptor berechnet wird. Statt den Deskriptors für jeden Punkt in einer Punktwolke zu berechnen, bestimmen Fiolka et al. [Fiolka et al., 2012] zunächst m Interest Points ($m \ll n$), die möglichst invariant gegenüber Veränderungen der Beleuchtung, des Betrachtungswinkels, der Skalierung etc. sind. Anschließend wird an diesen Punkten ein FPFH-ähnlicher Deskriptor extrahieren, der die Objekteigenschaften am jeweiligen Interest Point beschreibt. Ihre Kombination aus Interest Point Detektor und Deskriptor bezeichnen Fiolka et al. als SURE (Surface Entropie). Als Interest Points werden Punkte extrahiert, die eine starke lokale Variation der Oberflächenorientierung aufweisen. Als Vergleichsmaß wird die Entropie H einer Region $\mathcal{E}$ über ein Histogramm $X_\mathcal{E}$ der Azimutwinkel aller Normalen in der Nachbarschaft bestimmt:

$$H(X_\mathcal{E}) = -\sum_{x \in X_\mathcal{E}} p(x) \log p(x), \tag{3.8}$$

mit $p(x)$ als Binhäufigkeit des Histogrammbins x. Zur Erstellung der Histogrammbins führen Fiolka et al. eine gleichmäßige Dekomposition der kugelförmigen Nachbarschaftsregion eines Punktes durch (Abb. 3.14 a). Hierzu werden zunächst t äquidistante Neigungswinkel $\theta_i = \frac{\pi \cdot i}{t}$, mit $i \in \{0, \ldots, t-1\}$ ausgewählt, um anschließend für jeden dieser Neigungswinkel äquidistante Azimutwinkel $a_{\theta_i} := \lfloor 2t \sin(\theta_i) + 1 \rfloor \propto C(\theta_i)$ zu bestimmen. Hierdurch ist die Abtastdichte des Azimutwinkels proportional zum Kreisumfang $C(\theta_i)$. Jedes der so eingeteilten Histogrammbins wird durch einen auf das Binzentrum zeigenden, normalisierten Vektor $\boldsymbol{v}_{i,j}$ repräsentiert. Zur Berechnung

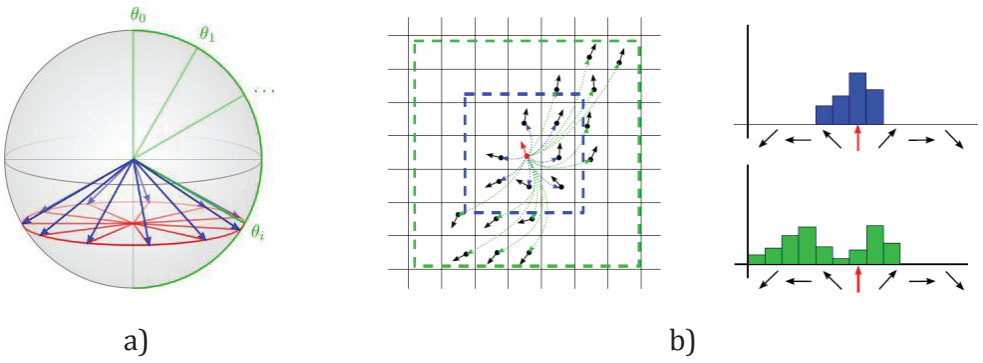


Abbildung 3.14: Surface Entropie (SURE) Interes Point Detektor und Deskriptor [Fiolka et al., 2012]
a) Gleichmäßige Dekomposition der Nachbarschaftsregion zur Erstellung der Histogrammbins. b) Der Deskriptor wird aus zwei Histogrammen unterschiedlicher Nachbarschaftsbereiche (einen Inneren (blau) und einen Äußeren (grün)) erstellt.

der Binhäufigkeiten wird für jede Oberflächennormale $\boldsymbol{n}(\boldsymbol{q}_m)$ eines Punktes $\boldsymbol{q}_m \in \mathcal{E}$ der Beitrag zum Histogrammbin $x_{i,j}$ über das Gewicht:

$$w_{i,j} = \begin{cases} 0, & \text{wenn } \boldsymbol{n}(\boldsymbol{q}_m) \cdot \boldsymbol{v}_{i,j} < \cos\alpha \\ \frac{\boldsymbol{n}(\boldsymbol{q}_m)\cdot\boldsymbol{v}_{i,j} - \cos\alpha}{1-\cos\alpha}, & \text{sonst} \end{cases} \tag{3.9}$$

bestimmt, mit α als maximal betrachteten Winkelbereich. Fiolka et al. extrahieren einen Interest Point dort, wo das Entropiemaß nach Formel 3.8 lokale Maxima aufweist. Interest Points die innerhalb der Nachbarschaft eines anderen Interest Points mit einer größeren Entropie liegen, werden entfernt.

Der anschließend an jedem Interest Point extrahierte Deskriptor ist dem FPFH sehr ähnlich. Fiolka et al. berechnen das Quadrupel $< \alpha, \phi, \theta, \geq >$ jedoch nur zwischen dem Interest Point und seinen Nachbarn, statt alle möglichen Zweier-Punktkombinationen in einer Nachbarschaft zu betrachten. Außerdem betrachten Fiolka et al. pro Interest Point zwei Nachbarschaftsbereiche (einen inneren und ein äußeren Bereich) für die jeweils ein Histogramm berechnet wird (Abb. 3.14 b). Abhängig davon, ob neben den Tiefendaten auch RGB-Daten vorhanden sind, integrien Fiolka et al. zu den Informationen über die Form auch farbige Texturinformationen in den Deskriptor. Hierzu wird der den Interest Point umgebende Bildbereich zum einen in den HSL-Farbraum

transformiert und ein Histogramm über den Farbton und die Sättigung gebildet. Zum anderen wird die Luminanzverteilung des Bereichs in einem Histogramm erfasst. In beiden Fällen wird wieder für den inneren und äußeren Nachbarschaftsbereich ein eigenes Histogramm erstellt.

Zur Evaluation des Interest Point Detektors testen Fiolka et al. die Wiederholbarkeit der Interest Point Detektionen unter unterschiedlichen Betrachtungswinkeln und vergleichen ihre Ergebnisse mit dem bis dahin als State-of-the-Art geltenden Normal Aligned Radial Feature (NARF) [Steder et al., 2010]. Damit zwei Interest Points unterschiedlicher Punktwolken übereinstimmen, muss ihre Euklidsche Distanz unterhalb des Kugelradius r der bei der Histogrammberechnung betrachteten Nachbarschaft liegen. Zudem darf kein anderer Interest Point innerhalb dieses Abstands liegen (*Unique Repeatability*). Außerdem evaluieren Fiolka et al. die Leistung des Deskriptors zwei korrespondierende Punkte zu finden. Um die Distanz zwischen den Deskriptoren zweier Interest Points $d(\boldsymbol{q}_1, \boldsymbol{q}_2)$ zu messen, kombinieren Fiolka et al. die gemittelte Euklidische Distanz der Winkelhistogramme mit der Earth Mover's Distanz [Rubner et al., 2000] der Farb- und der Luminanz-Histogramme zu einem Distanzwert. Über den Anteil der Interest Points, die zwischen zwei Bildern über den Deskriptor korrekt gematched wurden, kann ein *Matching Score* bestimmt werden. Der SURE-Deskriptor übertrifft den NARF-Deskriptor in beiden Experimenten.

3.4 Bewertung

Bei der Detektion gestürzter Personen über eine mobile Roboterplattform ist zu berücksichtigen, dass der Sturz außerhalb des Sensorbereichs der mobilen Roboterplattform stattfinden kann (Abschnitt 1.2). Daher sind die vorgestellten Verfahren zur Detektion des aktiven Sturzprozesses (Abschnitt 3.1.4) für den hier betrachteten Anwendungsfall nicht geeignet. Die bestehenden Verfahren zur Detektion liegender Personen (Abschnitt 3.1.5) erreichen jedoch keine ausreichende Detektionsleistung, um älteren Menschen im praktischen Einsatz Sicherheit zu bieten. Zudem sind die bestehenden Verfahren sehr komplex, wodurch aktuell eine echtzeitfähige Implementierung nicht möglich ist.

Die Verwendung von 3D-Daten versprechen hier neue, verbesserte Möglichkeiten. Neben den generell geltenden Vorteilen (Abschnitt 2.2) kann beispielsweise die Abbildungsvielfalt liegender Personen über eine Normalisierung der Objektorientierung reduziert werden, was zu einer verbesserten Detektionsleistung führen kann. Zudem lässt sich durch die Verwendung von 3D-Punktwolken der Rechenaufwand reduzieren. Über Szeneninformationen, wie die Lage der Bodenfläche und bestimmten Vorbedingungen für die zu segmentierenden Objekte, ist die Anzahl der zu untersuchenden Objekte deutlich einschränkbar.

Bei den bestehenden Features zur Detektion von Personen und Objekten in 3D-Punktwolken versprechen vor allem die auf der Orientierung der Oberflächennormalen basierenden Features eine gute Möglichkeit, zwischen Personen und anderen in der häuslichen Umgebung typischerweise vorkommenden Objekten zu unterscheiden (Abschnitt 3.2.1). Hierzu zählen das Histogram of Local Surface Normals (HLSN) (Abschnitt 3.2.1), das Fast Point Feature Histogram (FPFH) (Abschnitt 3.3.2) und der SURE Interest Point Detektor und Deskriptor (Abschnitt 3.3.2).

Außerdem ist zu erwarten, dass die Verwendung von geometrischen und statistischen Features (Abschnitt 3.2.2) zur Detektion liegender Personen ähnlich gute Ergebnisse liefert, wie bei der Detektion stehender Personen. Denn die geometrischen und statistischen Eigenschaften einer liegenden Person sollten denen einer stehenden Person ähnlich oder identisch sein.

Um eine Stabilität gegenüber den bei gestürzten Personen häufig auftretenden partiellen Verdeckungen zu erreichen (Abschnitt 1.2), empfiehlt sich die Verwendung eines körperteil- bzw. layerbasierten Ansatzes, wie er von Hegger et al. (Abschnitt 3.2.1) und Spinello et al. (Abschnitt3.2.2) vorgestellt wurde.

Kapitel 4

Featurebasierte Sturzdetektion in 3D-Daten: Eigener Ansatz

Auf Basis der Bewertungsergebnisse bisheriger Verfahren zur Detektion von Stürzen, Personen und Objekten (Abschnitt 3.4) wurde ein neuer Ansatz zur Evaluation geeigneter Features und zur Detektion gestürzter Personen über eine mobile Roboterplattform entwickelt. Der folgende Abschnitt gibt einen Überblick über das Konzept des eigenen Ansatzes. Anschließend werden die einzelnen Komponenten des neuen Verfahrens detailliert vorgestellt und die Details der Implementierung angesprochen.

4.1 Überblick

Der eigene Ansatz basiert auf den 3D-Daten einer an einer mobilen Roboterplattform befestigten Kinect. Die Kinect gibt ein Tiefenbild aus, das in eine Punktwolke konvertiert wird. In einer Vorverarbeitungseinheit (Preprocessing) wird die Anzahl der Punkte in der Punktwolke über ein Downsampling und die Begrenzung auf eine Region of Interest (ROI) reduziert. Zudem wird die Punktwolke bezüglich der Einbaulage der Kinect transformiert. In der anschließenden Ground Plane Estimation wird die Lage der Bodenfläche detektiert und alle nicht zum Boden zugehörigen Punkte bestimmt. Zur Segmentierung dieser Punkte wird anschließend ein Euklidisches Clustering angewendet. Über eine Verifizierung ihrer Höhe werden nur Cluster weiter betrachtet,

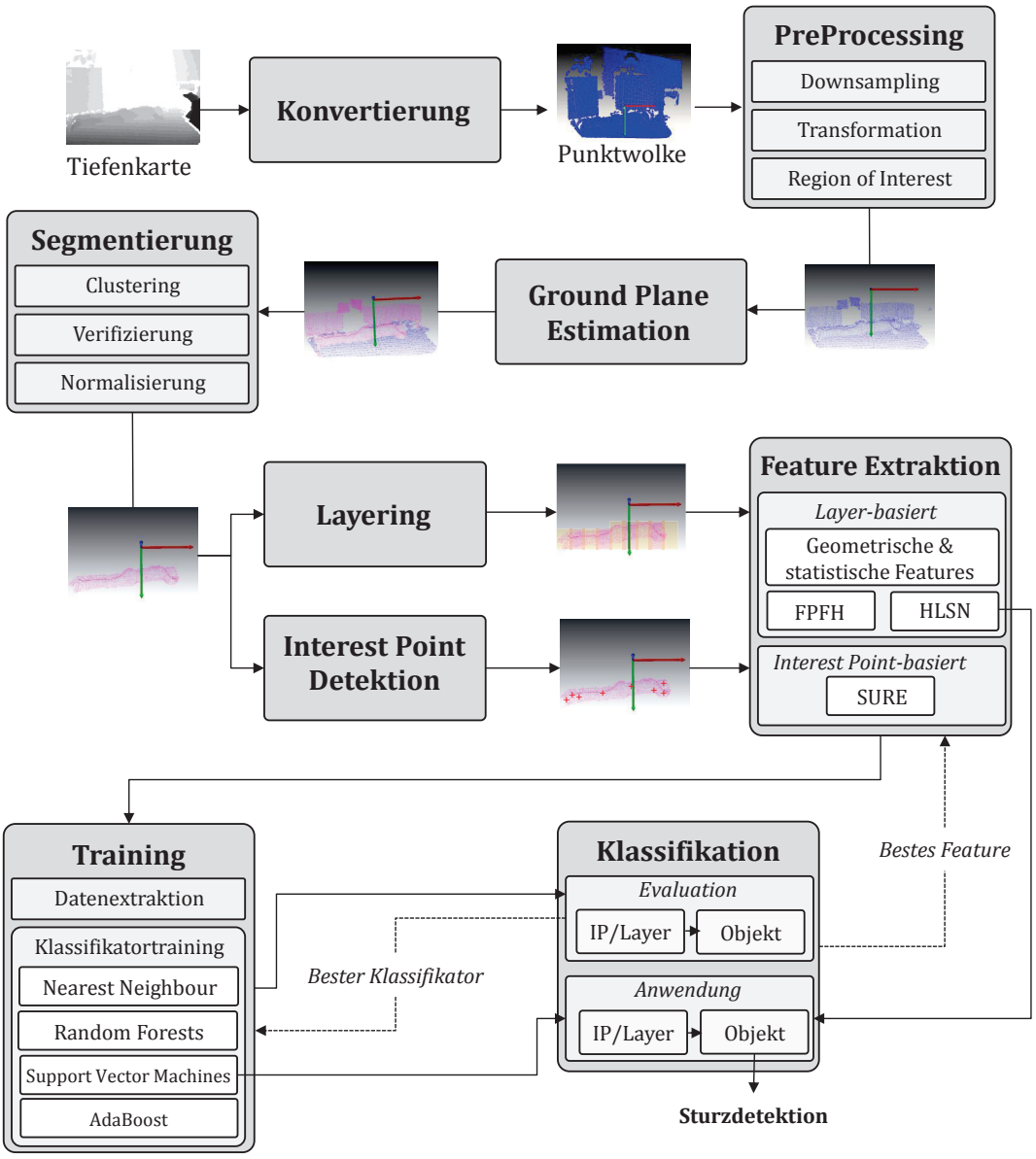


Abbildung 4.1: Neuer Ansatz zur featurebasierten Sturzdetektion in 3D-Daten

Schematische Darstellung des Ablaufs des neuen Ansatzes

die durch die Grenzen der ROI nicht abgeschnitten wurden. Diese werden bezüglich ihrer Lage und Orientierung normalisiert. Abhängig davon, ob Interest Points oder auf Layer basierende Features verwendet werden, wird jedes segmentierte Objekt an einen Interest Point Detektor oder an eine Layering-Einheit gesendet. In der anschließenden Feature Extraktion wird pro Interest Point bzw. Layer ein Featurevektor extrahiert. Als Features stehen geometrische und statistische Features, das HLSN, das FPFH, sowie der SURE-Deskriptor zur Verfügung (Kapitel 3). Die Featurevektoren werden, je nachdem ob sich der neue Ansatz in der Trainings- oder Detektionsphase befindet, zum Training eines binären Klassifikators oder zur Klassifikation des Interest Points bzw. Layers in eine von zwei Klassen (Person oder Objekt) verwendet. Als Klassifikationsalgorithmen stehen die Methoden der Nearest Neighbour, Random Forests, Support Vector Machines und AdaBoost zur Verfügung. Über eine ausführliche Evaluation wird die beste Feature-Klassifikator-Kombination gefunden. Mit dieser wird in der Anwendung des neuen Ansatzes die Klassenzugehörigkeit aller Interest Points bzw. Layer eines Clusters bestimmt, um über die Anzahl der positiv klassifizierten Interest Points bzw. Layer zu entscheiden, ob es sich bei dem Cluster um eine Person oder ein Objekt handelt. Abbildung 4.1 illustriert den schematischen Ablauf.

4.2 Implementierung

Zur Nutzung des neuen Ansatzes auf einer Roboterplattform des Fachgebiets Neuroinformatik und Kognitive Robotik der TU Ilmenau wurde die Implementierung in der Programmiersprache C++ vorgenommen und in die Middleware MIRA (Middleware for Robotic Applications) [Einhorn, 2012] eingebunden. Die im Folgenden beschriebenen Komponenten sind als einzelne MicroUnits umgesetzt worden. Innerhalb der Units werden Funktionen der PCL (Point Cloud Library) [Rusu und Cousins, 2011] verwendet. Die PCL ist ein Open Source Projekt, das Algorithmen und Datenstrukturen zur Verarbeitung von Punktwolken zur Verfügung stellt. Die als Klassifikatoren verwendeten maschinellen Lerntechniken sind zum Großteil in der Programmiersprache Python unter Verwendung der scikit-learn Bibliothek [Pedregosa et al., 2011] implementiert.

Einzig für den AdaBoost-Klassifikator wurde eine am Fachgebiet Neuroinformatik und Kognitive Robotik bereits bestehende C++ Implementierung optimiert.

4.2.1 Konvertierung

Um das Tiefenbild der Kinect in eine Punktwolke zu konvertieren, wird die im MIRA-Framework bereits zur Verfügung stehende `DepthToPointCloud`-Unit verwendet. In dieser wird über ein Lochkameramodell und den intrinsischen Kalibrierungsparametern der Kamera für jeden Pixel des Tiefenbildes der korrespondierende 3D-Punkt ermittelt und in einer Punktwolke gespeichert.

4.2.2 Preprocessing

Da die Kinect ein hochaufgelöstes Tiefenbild mit 640×480 Pixeln bereitstellt, ist die Anzahl der Punkte in der resultierenden Punktwolke mit 307.200 Punkten sehr hoch. Um den Berechnungsaufwand der Folgeschritte zu reduzieren, wird daher die Anzahl der Punkte über ein Downsampling und über die Begrenzung auf eine ROI verringert. Zudem werden die Punktkoordinaten innerhalb der `PreProcessing`-Unit bezüglich der Einbaulage der Kinect transformiert. Abbildung 4.2 zeigt den algorithmischen Ablauf des Preprocessings.

Downsampling

Zum Downsampling der Punktwolke wird der `VoxelGrid`-Filter der PCL verwendet. Dieser erstellt ein 3-dimensionales Gitter aus quadratischen Voxeln mit der Kantenlänge $downsample\_leaf\_size$ und mittelt alle Punkte innerhalb eines Voxels auf einen Punkt. Analog zu [Hegger et al., 2012] (Abschnitt 3.2.1) wird eine Voxelgröße von 3 cm verwendet.

Transformation

Bei der Konvertierung der Tiefenkarte werden nur die intrinsischen Kalibrierungsparameter der Kinect berücksichtigt. Deshalb muss die resultierende Punktwolke noch be-

Eingaben

1 $P_{in} = \{(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)\}$ *// Punktwolke mit n Punkten*
2 d *// Voxelgröße beim Downsampling*
3 h_k *// Höhe der Kinect*
4 α_k *// Roll-Winkel der Kinect*
5 y_{max} *// Maximale Höhe der ROI*
6 y_{min} *// Minimale Höhe der ROI*

Algorithmus

Downsampling: **// Reduktion der Punktwolkendichte**

7 Erstelle ein Gitter VG aus Voxeln V_j der Größe $d \times d \times d$;
8 Lege VG über P_{in};
9 Für jeden $V_j \in VG$;
10 Bestimme p_j als Mittelwert der Punkte $p_i \in V_j$;
11 Speicher p_j in P_d;

Transformation: **// Normalisierung bezüglich Einbaulage Kinect**

12 Bestimme die Transformationsmatrix T zur Rotation um α_k;
13 Bestimme P_t über die Transformation von P_d mit T;

Region of Interest: **// Begrenzung auf relevante Region**

14 $P_{ROI} = \{p_i \in P_t(x, y, z) \,|\, y \in [-y_{min} + h_k,\, -y_{max} + h_k]\}$;

Rückgabe

15 $P_{out} = P_{ROI}$ *// Vorverarbeitete Punktwolke*

Abbildung 4.2: Preprocessing Algorithmus

züglich der Einbaulage der Kinect über eine Transformation normalisiert werden. Hier ist vor allem der Neigungswinkel um die x-Achse (Roll-Winkel) von Bedeutung, da bei der anschließenden Extraktion der Region of Interest und der Ground Plane Estimation von einer parallel zur x-z-Ebene liegenden Bodenfläche ausgegangen wird. Über die PCL-interne Transformationsfunktion `transformPointCloud` wird die Punktwolke um den aktuellen Roll-Winkel der Kinect rotiert. Zur Reduzierung des Berechnungsaufwands wird keine zusätzliche Translation der Punktwolke bezüglich der Einbauhöhe

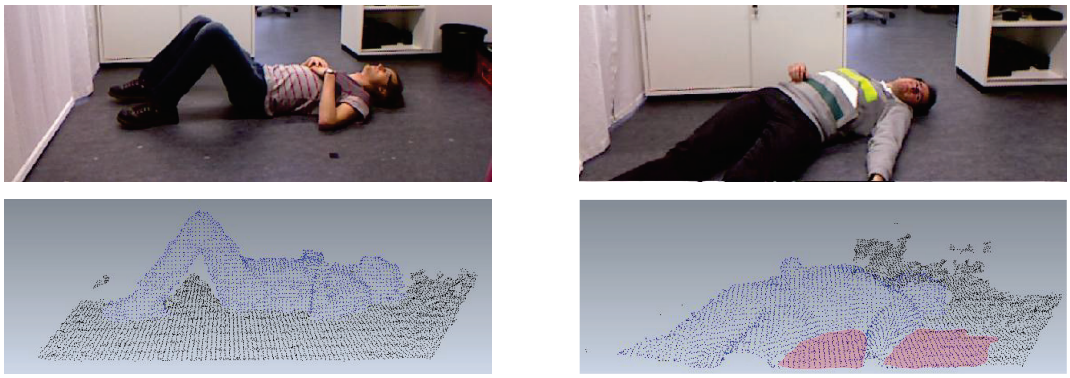


Abbildung 4.3: Beispielhafte Situationen, die Einfluss auf die Auswahl der Parametereinstellung bei der ROI und der Segmentierung haben
a) ROI: Durch das Abwinkeln der Beine kann die Punktwolke einer gestürzten Person relativ hoch sein. b) Clustering: Eine fehlerhafte Segmentierung der gestürzten Person führt zu einer erhöhten Anzahl an Punkten in der Punktwolke.

der Kinect durchgeführt. Stattdessen werden alle im Folgenden vorgestellten, sich auf die Höhe beziehenden Parameter, innerhalb des Codes mit der Einbauhöhe der Kinect ($kinect\_high = 90$ cm) verrechnet.

Region of Interest

Für die Detektion gestürzter Personen ist nur der untere Bereich der Punktwolke relevant. Daher können alle Punkte oberhalb der Höhe ROI_high_max aus der Punktwolke entfernt werden. Hierzu wird auf die Funktionalität des `PassTrough`-Filters der PCL zurückgegriffen. Über diesen werden alle Punkte aus der Punktwolke entfernt, deren y-Koordinate nicht innerhalb des Bereichs [ROI_high_min, ROI_high_max] liegt. Damit auch Personen berücksichtigt werden, die auf ein niedriges Objekt wie z. B. einen Staubsauger gestürzt sind oder die nach einem Sturz ihre Beine anwinkeln (Abb. 4.3 a) wird als Maximum eine Höhe von 60 cm verwendet. Das Minimum wird auf -10 cm gesetzt, da bei der Konvertierung der Tiefendaten eine Fehlertoleranz zu berücksichtigen ist.

4.2.3 Ground Plane Estimation

Zur Bestimmung der Bodenfläche wird auf die PCL-Implementierung des modellbasierten Segmentierungsverfahrens RANSAC (Abschnitt 2.2.2) zurückgegriffen. Als Modell wird das `SAC_MODEL_NORMAL_PARALLEL_PLANE` verwendet. Dieses definiert eine Ebene, deren Normalenvektor parallel zu einer vom Nutzer vorgegebenen Achse liegt. Bei der Ermittlung der das Modell unterstützenden Punkte wird zusätzlich zur Euklidischen Distanz zwischen Punkt und Modell, der Winkel zwischen der Punkt- und der Ebenennormalen betrachtet. Beides muss unter einem Schwellwert liegen, damit der Punkt das Modell unterstützt. Mit den Schwellwerten $RANSAC\_distance\_max = 10$ cm und $RANSAC\_angle\_max = 2°$ können in den Testdaten gute Ergebnisse erzielt werden.

Die PCL-Implementierung nutzt zur effizienten Suche nach den besten Modellparameter einen Kd-Suchbaum. Dieser speichert die Eingangsdaten in einer baumartigen Datenstruktur und unterstützt so effiziente Bereichsanfragen. Für eine detaillierte Betrachtung wird auf [Maneewongvatana und Mount, 1999] verwiesen.

Damit in den folgenden Schritten die zum Boden zugehörigen Punkte von den restlichen Punkten unterschieden werden können, gibt die `GroundPlaneEstimation`-Unit eine Punktwolke aus, in der den Punkten unterschiedliche Labels zugewiesen sind (0 = Ground Plane, 1 = Rest).

4.2.4 Segmentierung

Clustering

Um die in der Punktwolke abgebildeten Objekte einzeln klassifizieren zu können, werden alle nicht zum Boden zugehörige Punkte über ein Euklidisches Clustering (Abschnitt 2.2.2) segmentiert. Hierzu wird auf die Implementierung der PCL zurückgegriffen. Wie bei der RANSAC-Implementierung wird auch hier zur effizienten Gestaltung der Suche ein Kd-Suchbaum verwendet. Als Parameter ist der maximale Schwellwert für die euklidische Distanz zweier Punkte eines Clusters $Clust\_dist\_max$, sowie die minimale $Clust\_size\_min$ und maximale Anzahl $Clust\_size\_max$ der Punkte pro

Cluster zu definieren. In den Testdaten, die unterschiedlich große Personen in verschiedenen Posen, Tiefen ($< 4m$) und unterschiedlich starken, partiellen Verdeckungen beinhalten, bestehen die Punktwolken aller Personen aus weniger als 3800 und mehr als 700 Punkten. Um Fehler der Segmentierung zu tolerieren, wird die minimale Anzahl an Punkten pro Cluster auf 500 und die maximale auf 5000 gesetzt. Dadurch werden beispielsweise auch Personen berücksichtigt, die mit einem Teil des Bodens zusammen segmentiert werden und deren Punktwolke daher aus einer größeren Anzahl an Punkten besteht (Abb. 4.3 b). Als Schwellwert für die euklidische Distanz wird ein Maß von 3 cm verwendet. Den algorithmischen Ablauf des Clusteringprozesses zeigt Abbildung 4.4.

Eingaben

1 $P_{in} = \{(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)\}$ *// Punktwolke mit n Punkten*
2 d_{max} *// Maximale Distanz zweier Punkte*
3 c_{min} *// Minimale Anzahl an Punkten pro Cluster*
4 c_{max} *// Maximale Anzahl an Punkten pro Cluster*

Algorithmus

5 Für jeden $p_i \in P_{in}(x, y, z)$; *// für jeden Punkt der Punktwolke*
6 Überprüfe, ob ein Cluster C_k, $k = 1, \ldots, K$ mit einem Punkt p_j existiert, für den gilt: $euclDist(p_i \rightarrow p_j) \leq d_{max}$;
7 Wenn *ja*: füge p_i zu C_k hinzu;
8 Wenn *nein*: erstelle neuen Cluster C_{K+1}, $p_i \in C_{K+1}$;
9 Für jeden $C_k \in \mathbf{C}$; *// für jeden segmentierten Cluster*
10 Überprüfe, ob die Kardinalität von $C_k \in [c_{min}, c_{max}]$;
11 Wenn *nein*: entferne C_k aus $\mathbf{C}$;

Rückgabe

12 $\boldsymbol{C}$ *// Vektor mit Punktwolken der einzelnen Cluster*

Abbildung 4.4: Segmentierung: Algorithmus zum Clustering einer Punktwolke

Verifizierung

Durch die Verwendung des `PassTrough`-Filters zur Begrenzung der Punktwolke auf die ROI werden Objekte mit einer Höhe $> ROI\_high\_max$ abgeschnitten, wodurch der untere Teil des Objekts weiterhin Teil der Punktwolke ist. Da ausgeschlossen werden kann, dass diese Objekte einer liegenden Person entsprechen, werden alle segmentierten Cluster über ihre Höhe verifiziert. Um eine Unabhängigkeit von einzelnen Ausreißern zu gewährleisten, wird die Höhe jedes Clusters über den Mittelwert der höchsten 1% seiner Punkte bestimmt. Liegt diese oberhalb dem Schwellwert:

$$ROI\_high\_max - Clust\_high\_diff = 55 \text{ cm}$$

wird davon ausgegangen, dass das Objekt durch die ROI abgeschnitten wurde und daher nicht weiter als möglicher Kandidat für eine gestürzte Person betrachtet. Abbildung 4.5 zeigt den algorithmischen Ablauf der Verifizierung.

Eingaben

1 $\boldsymbol{C}$ // *Vektor mit Punktwolken der einzelnen Cluster*
2 y_{max} // *Maximale Höhe der ROI*
3 h_d // *Maximale Höhendifferenz zur ROI*
4 h_k // *Höhe der Kinect*

Algorithmus

5 Für jeden $C_k \in \mathbf{C}$; // *für jeden segmentierten Cluster*
6 Speicher die absteigend sortierten y-Koord. aller $p_i \in C_k$ in $\mathbf{y}_k$;
7 Bestimme den Mittelwert h_{c_k} des ersten 1% der Elemente von $\boldsymbol{y}_k$;
8 Wenn $h_{c_k} > -(y_{max} - h_d) + h_k$;
9 Entferne C_k aus $\boldsymbol{C}$;

Rückgabe

10 $\boldsymbol{C}$ // *Vektor mit Punktwolken der einzelnen Cluster*

Abbildung 4.5: Segmentierung: Algorithmus zur Verifizierung der Clusterhöhe

Normalisierung

Um die Position und Ausrichtung eines Objekts für das folgende Layering oder die Detektion von Interest Points zu normalisieren, wird es am Koordinatenursprung und entlang der x-Achse ausgerichtet. Hierzu wird der Schwerpunkt und die Hauptorientierung des Objekts bestimmt. Die Hauptorientierung wird über den Eigenvektor mit dem größten Eigenwert der über alle Punkte des Objekts gebildeten Kovarianzmatrix ermittelt. Über eine affine Transformation wird der Schwerpunkt des Objekt zunächst in den Koordinatenursprung verschoben und anschließend an diesem um den Winkel zwischen x-Achse und Hauptorientierung rotiert. Den algorithmischen Ablauf des Normalisierungsprozesses zeigt Abbildung 4.6.

Eingaben

1 $\boldsymbol{C}$ *// Vektor mit Punktwolken der einzelnen Cluster*

Algorithmus

2 Für jeden $C_k \in \boldsymbol{C}$; *// für jeden segmentierten Cluster*
3 Bestimme den Schwerpunkt s_{c_k} der Punkte $p_i \in C_k$;
4 Bestimme die Kovarianzmatrix ζ_{c_k} in s_{c_k} über die Punkte $p_i \in C_k$;
5 Bestimme die Eigenvektoren υ und die Eigenwerte λ von ζ_{c_k};
6 Verwende den Eigenvektor υ_n mit dem größten Eigenwert λ_n als Hauptorientierung von C_k;
7 Bestimme den Winkel α zwischen υ_n und der x-Achse;
8 Bestimme die Transformationsmatrix T_t zur Translation um s_{c_k};
9 Bestimme P_{C_k} über die Transformation der Punkte $p_i \in C_k$ mit T_t;
10 Bestimme die Transformationsmatrix T_{rot} zur Rotation um α;
11 Transformiere P_{C_k} mit T_r;

Rückgabe

12 $\boldsymbol{P} = \{P_{C_1}, \ldots, P_{C_K}\}$ *// Vektor mit Punktwolken der K Cluster*

Abbildung 4.6: Segmentierung: Algorithmus zur Normalisierung einer Clusterpunktwolken

Die `Segmentation`-Unit liefert als Ausgabe einen Vektor, der die normalisierten Punktwolken der segmentierten Cluster enthält. Zusätzlich kann zu Visualisierungszwecken eine gelabelte Punktwolke ausgegeben werden, die die segmentierten Cluster in ihrer ursprünglichen Position und Orientierung jeweils mit einem eigenen Label enthält.

4.2.5 Layering

Zum Erreichen einer Robustheit gegenüber partiellen Verdeckungen, wird allen Features, die nicht auf der Detektion von Interest Points basieren, vor der Feature Extraktion eine Layering-Stufe vorgeschaltet, die ähnlich der bei [Hegger et al., 2012] (Abschnitt 3.2.1) und [Spinello et al., 2010] (Abschnitt 3.2.2) arbeitet. Um auch bei partiell verdeckten Personen eine stabile Layergröße zu erhalten, wird im neuen Ansatz statt einer festen Layeranzahl standardmäßig eine feste Layergröße verwendet. Zur Festlegung dieser wurden Cluster von komplett abgebildeten, liegenden Personen analog zu Hegger et al. in 8 Layer aufgeteilt. Die Mittelung der Layergrößen mehrerer Testdaten ergab eine Größe von $layer\_size = 22.52$ cm.

Anders als bei Hegger et al. und Spinello et al. werden in diesem Ansatz sich überlappende Layer verwendet (Abb. 4.8). Hierdurch kann die Varianz der Größe verschiedener Personen und die daraus resultierende Varianz der Layerinhalte etwas ausgeglichen werden. Mit einer Überlappung von $layer\_overlap = 2.5$ cm beinhaltet jeder Layer zu etwa 10% den gleichen Bereich wie sein Nachbar.

Für die Umsetzung des Layering wird die normalisierte Punktwolke von links nach rechts iterativ durchlaufen. Für jeden Layer wird über die Layernummer l, die Layergröße, den Layer-Overlap und den minimalen x-Wert des Clusters $cluster\_x\_min$ seine linke $boundary\_l$ und rechte Grenze $boundary\_r$ bestimmt:

$$boundary\_l = cluster\_x\_min + (l-1) * (layer\_size - layer\_overlap) \quad (4.1)$$

$$boundary\_r = boundary\_left + layer\_size \quad (4.2)$$

Alle Punkte innerhalb der Grenzen werden über die Anwendung des `PassTrough`-Filters der PCL extrahiert und in einer eigenen Punktwolke gespeichert. Abbildung 4.7 zeigt den algorithmischen Ablauf des Layering-Prozesses.

Eingaben

1 $P_k = \{(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)\}$ *// Punktwolke des k-ten Clusters, // bestehend aus n Punkten*

2 l_s *// Größe jedes Layers*

3 l_o *// Größe der Überlappung zweier Layer*

Initialisierung

4 $x_{min} \leftarrow min(x_i \in P)$; *// linke Clustergrenze*

5 $x_{max} \leftarrow max(x_i \in P)$; *// rechte Clustergrenze*

6 $l_n \leftarrow \lfloor (x_{max} - x_{min}) / l_s \rfloor$; *// Anzahl der Layer*

Algorithmus

7 `Für` $l = 1, \ldots, l_n$`;` *// Für jeden Layer l*

8 `Bestimme die linke Layergrenze:` $b_l = x_{min} + (l-1) * (l_s - l_o)$`;`

9 `Bestimme die rechte Layergrenze:` $b_r = b_l + l_s$`;`

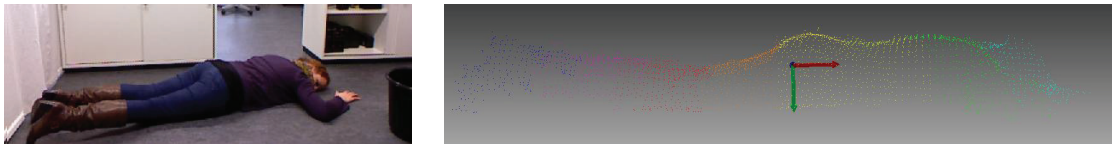
10 `Erstelle eine Punktwolke mit den zum Layer zugehörigen Punkten:` $P_l = \{p_i \in P_i(x, y, z) \,|\, x \in [b_l, b_r]\}$`;`

Rückgabe

11 $\boldsymbol{P}_k = \{P_1, \ldots, P_{l_n}\}$ *// Vektor mit Punktwolken der l_n Layer des k-ten Clusters*

Abbildung 4.7: Layering Algorithmus

Im neuen Ansatz wird analog zu Hegger et al. ein nicht layerspezifischer Klassifikator verwendet. Um die Möglichkeit jedoch offen zu halten, den Ansatz in Zukunft um ein layerspezifisches Verfahren wie bei Spinello et al. zu erweitern, erlaubt die `Layering`-Unit neben der Verwendung einer festen Layergröße auch die Verwendung einer festen Layeranzahl. Hierdurch kann gewährleistet werden, dass auch bei Personen unterschiedlicher Größe der gleiche Körperbereich dem gleichen Layer zugewiesen wird. Über Punktwolken, die ausschließlich komplett abgebildete Personen enthalten, lassen sich so layerspezifische Trainingsdaten generieren, mit denen sich anschließend ein spezialisierter Klassifikator für jeden Layer trainieren lässt. Das Setzen der boolschen



a) b)

Abbildung 4.8: Layering

Im neuen Ansatz wird die Punktwolke in sich überlappen Layer unterteilt. Die Layer haben eine feste Größe. a) Originalbild. b) Punktwolke nach dem Layering.

Variable *constant_layer_number* ermöglicht den Wechsel in diesen Modus. Über die gegeben Anzahl der festen Layer *layer_number* und die Breite der Punktwolke wird in diesem zunächst die individuelle Layergröße bestimmt, um mit dieser anschließend das oben beschriebene iterative Layering-Verfahren durchzuführen.

Die Ausgabe der `Layering`-Unit ist ein Vektor, der pro segmentierten Cluster wiederum einen Vektor enthält, indem die Punktwolken der einzelnen Layer des Clusters enthalten sind. Zudem kann die in Layer aufgeteilte Punktwolke eines einzelnen Objekts ausgegeben werden.

4.2.6 Interest Point Detektion

Neben der Verwendung einzelner Layer ist die Extraktion von Features an einzelnen Interest Points eine weitere Möglichkeit, eine Robustheit des neuen Ansatz gegenüber partiellen Verdeckungen zu erreichen. Zur Detektion der Interest Points wird der in [Fiolka et al., 2012] vorgestellte SURE-Interest Point Detektor (Abschnitt 3.3.2) verwendet. Dieser selektiert in Punktwolken Regionen mit einer hohen Oberflächenvarianz. Zur Implementierung wurde der unter [Fiolka, 2012b] öffentlich zur Verfügung stehende Quellcode verwendet.

Parameter, die die Performance des Detektors beeinflussen, sind:

- Abtastrate (*SURE_sampling_rate*),
- Histogrammskala (*SURE_hist_scale*),
- Normalenabtastrate (*SURE_normals_sampling_rate*),
- Normalenskala (*SURE_normals_scale*).

Abtastrate	Histogrammskala	Normalenskala	Normalenabtastrate
1	3	1	0.5

Tabelle 4.1: Optimale Verhältnis der Parameter des SURE-Interest Point Detektors [Fiolka, 2012a]

Die Abtastrate beschreibt die Genauigkeit, mit der die Punktwolke für die Suche nach Maxima in der Entropie aufgelöst wird. Die Histogrammskala bezeichnet die Größe der Region um einen Punkt, in der die Oberflächennormalen zur Erstellung des Entropiehistogramms berücksichtigt werden. Über die Normalenabtastrate wird die Genauigkeit festgelegt, mit der die Normalen geschätzt werden. Die Normalenskala legt fest, wie groß der kubische Bereich ist, in dem die Normalen geschätzt werden, daher sollte der Wert der Normalenskala immer über der Normalenabtastrate liegen. Tabelle 4.1 zeigt das von Fiolka in [Fiolka, 2012a] ermittelte optimale Verhältnis der vier Parameter.

Analog zur Nachbarschaftsgröße bei der Normalenberechnung *normals_radius* der zwei anderen auf Normalenhistogrammen basierenden Features (HLSN und FPFH, Abschnitt 4.2.7), wird im neuen Ansatz eine Normalenskala von 4 cm verwendet. Entsprechend der Tabelle 4.1 ergeben sich daraus eine Abtastrate von 4 cm, eine Histogrammskala von 12 cm und eine Normalenabtastrate von 2 cm. Diese Werte bewertet auch Fiolka als optimalen Kompromiss zwischen Leistung und Berechnungsaufwand.

Welche der über den Detektor gefundenen Entropiemaxima schließlich als Interest Points verwendet werden, wird über den Entropieschwellwert $SURE\_entropie\_min$ und die Eckigkeit $SURE\_cornerness\_min$ beeinflusst. Der Entropieschwellwert ist der Grenzwert, ab dem ein lokales Maximum als Interest Point in Betracht gezogen wird. Über die Eckigkeit können Entropiemaxima an Kanten ausgeschlossen werden. Für beide Parameter werden die im neuen Ansatz die von Fiolka empfohlenen Werte verwendet: $SURE\_entropie\_min = 0.6$ und $SURE\_cornerness\_min = 0.15$.

Da die Implementierung von Fiolka et al. die Berechnung der Interest Points mit der Extraktion des Deskriptors in einer Funktion koppelt, wird im neuen Ansatz der SURE-Interest Point Detektor mit der SURE-Deskriptor Extraktion in der `FeatureExtractionPoint`-Unit zusammengefasst.

4.2.7 Feature Extraktion

Entsprechend den Bewertungsergebnissen bekannter Features zur Detektion von Personen und Objekten (Abschnitt 3.4) wurden im neuen Ansatz vier verschiedene Features implementiert, die in einer ausführlichen Evaluation (Kapitel 5) auf ihre Leistung zur Detektion gestürzter Personen untersucht werden:

- Geometrische und statistische Features nach [Spinello et al., 2011]
- Histogram of Local Surface Normals (HLSN) [Hegger et al., 2012]
- Fast Point Feature Histogram (FPFH) [Rusu, 2009]
- SURE-Deskriptor [Fiolka et al., 2012]

Die ersten drei Features werden layerbasiert implementiert, d. h. es wird pro Layer ein Featurevektor mit d Merkmalen extrahiert. Da der SURE-Deskriptor dem FPFH sehr ähnlich ist, bietet er in Kombination mit dem SURE-Interest Point Detektor eine gute Möglichkeit, den Einfluss von Layer- und Interest Point-basierter Featureextraktion auf die Detektionsleistung zu untersuchen.

Geometrische und statistische Features (GSF)

Im neuen Ansatz werden die von Spinello et al. in [Spinello et al., 2011] zur Klassifikation einzelner Voxels vorgestellten geometrischen und statistischen Features verwendet. Pro Layer L_j wird ein Featurevektor $\boldsymbol{f}$ mit 9 Merkmalen f_i extrahiert. Die Berechnung der einzelnen Merkmale erfolgt analog zur Beschreibung in Abschnitt 3.2.2 auf Seite 37. f_1 entspricht der Anzahl der Punkte n der Punktwolke des Layers L_i und kann leicht über das `size`-Attribut der Punktwolke bestimmt werden. Zur Berechnung der für die Merkmale $f_2 - f_4$ benötigten Eigenwerte wird über die PCL-Funktion `computeCovarianceMatrix` zunächst die 3×3 große Kovarianzmatrix der Punktwolke des Layers L_j bestimmt. Mit der PCL-Funktion `eigen33` lassen sich anschließend die Eigenwerte dieser bestimmen. Die Kovarianzmatrix wird weiter verwendet, um die für das Merkmal f_8 benötigten Ebenenparameter über die PCL-Funktion `solvePlaneParamters` zu bestimmen. Diese Funktion ermittelt die Ebenenparameter

a, b, c und d über die Eigenwerte und Eigenvektoren der Kovarianzmatrix mittels der Methode der kleinsten Quadrate [Blow, 1960]. Zudem gibt die Methode ein Maß für die Rundheit der Punktwolke aus, dass bis auf einen Faktor 3 dem Merkmal f_2 entspricht. Um den für das Merkmal f_7 benötigte, unabhängige Median $\tilde{\boldsymbol{x}}$ zu bestimmen, werden die x-, y- und z-Koordinaten aller Punkte zunächst in drei Vektoren $\boldsymbol{v}_{x,y,z}$ kopiert und diese nach aufsteigenden Werten sortiert. Der unabhängige Median $\tilde{\boldsymbol{x}}$ ergibt sich schließlich aus:

$$\tilde{\boldsymbol{x}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} = \begin{cases} \begin{pmatrix} \boldsymbol{v}_x[(n+1)/2] \\ \boldsymbol{v}_y[(n+1)/2] \\ \boldsymbol{v}_z[(n+1)/2] \end{pmatrix} & \text{wenn } n \text{ ungerade} \\ \begin{pmatrix} (\boldsymbol{v}_x[n/2-1] + \boldsymbol{v}_x[n/2+1])/2 \\ (\boldsymbol{v}_y[n/2-1] + \boldsymbol{v}_y[n/2+1])/2 \\ (\boldsymbol{v}_z[n/2-1] + \boldsymbol{v}_z[n/2+1])/2 \end{pmatrix} & \text{wenn } n \text{ gerade.} \end{cases} \tag{4.3}$$

Für die Berechnung der Punktdichte im Merkmal f_9 werden über die PCL-Funktion `getMinMax3D` die minimalen und maximalen x, y und z-Koordinaten der Punktwolke bestimmt und über:

$$\mathcal{B} = (x_{max} - x_{min}) * (y_{max} - y_{min}) * (z_{max} - z_{min}) \tag{4.4}$$

das Volumen der die Punktwolke umgebenden Bounding Box bestimmt. Dieses wird entsprechend der Formulierung auf Seite 37 mit der Anzahl der Punkt n ins Verhältnis gesetzt.

Histogram of Local Surface Normals (HLSN)

In dieser Arbeit wird das von Hegger et al. zur Detektion stehender Personen entwickelte HLSN [Hegger et al., 2012] auf die Detektion liegender Personen adaptiert. Wie bei Hegger et al. werden die Oberflächennormalen der Punktwolke über das in Abschnitt 2.2.3 beschriebene Verfahren objekt- und nicht layerweise bestimmt. Sonst wäre die Genauigkeit der Normalenberechnung für die Punkte, die an dem Rand eines Layers liegen, deutlich reduziert. Denn ein für die Normalenorientierung wichtiger Teil

der Nachbarschaft kann bereits zu einem anderen Layer gehören und würde bei der Normalenberechnung nicht berücksichtigt werden. Die Berechnung der Oberflächennormalen wird über Funktionen der PCL realisiert. Diese verwenden auch hier zur effizienten Berechnung einen Kd-Suchbaum. Zur Bestimmung der bei der Berechnung der Oberflächennormalen berücksichtigten Nachbarschaft wird analog zu Hegger et al. ein Radius von $normals\_radius = 4$ cm verwendet. Die Berechnung der Normalenhistogramme erfolgt analog zu dem in Abschnitt 3.2.1 beschriebenen Verfahren.

Zusätzlich zu den 3×7 Histogrammwerten, der Tiefe und der Anzahl der Punkte, wird im neuen Ansatz statt der Breite, die Höhe des Layers in dem 24-elementrigen Featurevektor berücksichtigt. Da in dem neuen Ansatz, anders als bei Hegger et al., jeder Layer nur aus einem Cluster besteht, wird auf die Distanz vom Cluster- zum Layerzentrum verzichtet. Hierdurch besteht der in dieser Arbeit verwendete HLSN-Featurevektor aus einem Element weniger als der Featurevektor von Hegger et al.

Fast Point Feature Histogram (FPFH)

Das FPFH ist ein punktbasiertes Feature (Abschnitt 3.3.2), d. h. für jeden Punkt einer Punktwolke wird ein eigener Deskriptor berechnet. Um bei der layerbasierten Feature Extraktion trotz Verwendung des FPFH pro Layer nur einen Featurevektor zu extrahieren, werden in dem neuen Ansatz die FPFHs aller n Punkte eines Layers aufsummiert und über die Division durch n normalisiert. Als Alternative könnten die Histogramme eines Layers beispielsweise über ein kMeans-Clustering auf k Clusterzentren beschränkt und diese in einem Featurevektor hintereinander gereiht werden. In der Detektionsphase müssten dann jedoch alle Permutationen der k Clusterzentren überprüft werden. Dies führt zu einer deutlichen Steigerung der Komplexität und des damit verbundenen Rechenaufwands. Daher wird diese Alternative in dieser Arbeit nicht weiter betrachtet.

Die PCL bietet bereits eine Implementierung des FPFH, die mit 11 Bins pro Differenzmaß für jeden Punkt einen 33-dimensionalen Featurevektor extrahiert. Aus den oben genannten Gründen erfolgt auch beim FPFH die Berechnung der Oberflächennormalen objekt- und nicht layerweise. Als Nachbarschaft zur Bestimmung der Oberflächennor-

malen wird, wie beim HLSN, ein Bereich mit einem Radius von 4 cm betrachtet. Entsprechend der Empfehlung von Rusu [Rusu, 2009] wird zur Berechnung der Normalenhistogramme ein Nachbarschaftsbereich mit dem Radius

$$FPFH\_neighbors\_radius = 2 * normals\_radius = 8 \text{ cm}$$

berücksichtigt.

Die drei beschriebenen Features sind in der `FeatureExtractionLayer`-Unit implementiert. Diese gibt einen Vektor aus, der für jeden Cluster der segmentierten Punktwolke einen Vektor mit den Featurevektoren der Layer des Clusters enthält.

SURE-Deskriptor

Im Gegensatz zu den drei oben beschriebenen Features wird der SURE-Deskriptor an einzelnen Interest Points extrahiert. Wie in Abschnitt 4.2.6 beschrieben, wurde zur Implementierung des SURE-Detektors und Deskriptors der originale Quellcode von Fiolka et al. verwendet. Ähnlich dem FPFH wird bei dem verwendeten SURE-Deskriptor pro Interest Point der in Abschnitt 3.3.2 beschriebene, 33-elementrige Featurevektor extrahiert. Der SURE-Deskriptor ist mit dem SURE-Interest Point Detektor in `FeatureExtractionPoints`-Unit zusammengefasst. Die Datenstruktur der Ausgabe entspricht der `FeatureExtractionLayer`-Unit, d. h. es wird ein Vektor ausgegeben, der für jeden Cluster der segmentierten Punktwolke einen Vektor mit den Featurevektoren der Interest Points des Clusters enthält.

4.2.8 Training

Trainingsdatenextraktion

Um über die extrahierten Featurevektoren die Klassenzugehörigkeit eines Layers oder eines Interest Points zu bestimmen, muss zunächst ein Klassifikator mit Hilfe eines maschinellen Lernverfahrens trainiert werden. Ausgangspunkt des Trainings ist ein Datensatz mit positiven und negativen Trainingsdaten. Diese lassen sich über die

`extractFeatureData`-Unit aus repräsentativen Trainingsvideos mit Hilfe des miracenters generieren.

Wie in Abschnitt 4.2.5 beschrieben, soll die Möglichkeit gewahrt werden, den neuen Ansatz auf die Verwendung eines layerspezifischen Klassifikators zu erweitern. Daher extrahiert die Unit die Trainingsdaten nicht objektunabhängig, sondern bewahrt die Zuordnung der Featurevektoren jedes Layers zu seinem Objekt.

Klassifikatortraining

Da das optimal geeignete Klassifikationsverfahren von den verwendeten Daten und dem verwendeten Feature abhängig ist, wird in dem neuen Ansatz die Leistung vier verschiedener Klassifikatoren untersucht (Kapitel 5):

- Nearest Neighbour (NN)
- Random Forests (RF)
- Support Vector Machines (SVM)
- AdaBoost (AB)

Für die ersten drei Klassifikatoren (NN, RF, SVM) wird die phyton-Bibiliothek scikit-learn [Pedregosa et al., 2011] verwendet. Auf Grund der Invarianz der SVM gegenüber der Skalierung der Eingangsdaten, werden alle Trainingsdaten vor dem Trainieren der Klassifikatoren so skaliert, dass der Mittelwert ihrer Amplituden $\mu = 0$ und ihre Varianz $\sigma = 1$ entspricht. Die dazu verwendeten Skalierungsparameter μ_{scale} und σ_{scale} werden zur Anwendung in der Detektionsphase gespeichert. Zur Gewährleistung der Unabhängigkeit des Klassifikators von den verwendeten Trainingsdaten erfolgt das Klassifikatortraining im Rahmen einer Kreuzvalidierung. Bei dieser werden die Trainingsdaten in n_{cv} Runden in unterschiedliche Trainings- und Testsets unterteilt. Pro Runde wird ein Klassifikator mit dem aktuellen Trainingsset trainiert, anschließend mit dem zugehörigen Testset getestet und die Detektionsleistung über einen Klassifikations-Score beschrieben. Dieser entspricht der in Abschnitt 2.1.2 vorgestellte Accuracy. Als finaler Klassifikator wird der mit dem höchsten Klassifikations-Score

verwendet. Zur Aufteilung der Daten in n_{cv} Trainings- und Testsets wird im neuen Ansatz die `StratifiedKFold`-Methode verwendet, bei der der Anteil an positiven und negativen Daten in jedem Trainings- bzw. Testset gleich dem im gesamten Datenset ist. Die Kreuzvalidierung wird in $n_{cv} = 5$ Runden vorgenommen. Abbildung 4.9 zeigt den algorithmischen Ablauf des Trainings der drei beschriebenen Klassifikatoren.

Eingaben

1 $S = [(e_1, l_1), \ldots, (e_N, l_N)]$ *// Set mit Trainingsbeispielen*
// $l_n = +1$ für positive Beispiele
// $l_n = -1$ für negative Beispiele
2 $P = [p_1, ..., p_M]$ *// Parametereinstellungen des Klassifikators*
3 n_{cv} *// Anzahl der Kreuzvalidierungsrunden*

Initialisierung

4 $score_{max} \leftarrow -\infty$; *// Bester Klassifikations-Score*
5 $clf_{best} \leftarrow 0$; *// Bester Klassifikator*

Algorithmus

6 Bestimme den Mittelwert μ_{scale} und die Varianz σ_{scale} von S;
7 Skaliere S mit: $e_m = (e_n - \mu) * \sigma$; *// Skalierung*
8 Für jedes $p_j \in P$; *// Für jede Parametereinstellung*
9 Teile S in n_{cv} Trainingssets $S_{i_{train}}$ und Testsets $S_{i_{test}}$;
10 Für $i = 1, \ldots, n_{cv}$; *// Kreuzvalidierung*
11 Trainiere den Klassifikator $clf_i(p_j)$ über $S_{i_{train}}$; *// Training*
12 Bestimme den Klassifikations-Score $score$ von $clf_i(p_j)$ über $S_{i_{test}}$;
13 Wenn $score > score_{max}$;
14 $clf_{best} = clf_i(p_j)$;

Rückgabe

15 clf_{best} *// Bester Klassifikator*
16 μ_{scale} *// Skalierungsparameter: Amplitudenmittelwert*
17 σ_{scale} *// Skalierungsparameter: Varianz*

Abbildung 4.9: Algorithmus zum NN-, RF- und SVM-Training

Eingaben

1 $S = [(e_1, l_1), \ldots, (e_N, l_N)]$ // *Set mit N Trainingsbeispielen*

2 $N = a + b$ // *a positive Beispiele mit* $l_n = +1$
// *b negative Beispiele mit* $l_n = -1$

3 T // *Anzahl Trainingsrunden*

Initialisierung

4 $D_1(n) \leftarrow \frac{1}{2a}$; // *für* $l_n = +1$

5 $D_1(n) \leftarrow \frac{1}{2b}$; // *für* $l_n = -1$

Algorithmus

6 Für $t = 1, \ldots, T$;

7 Normalisiere die Gewichte: $D_t(n) = \frac{D_t(n)}{\sum_{i=1}^{N} D_t(i)}$;

8 Für jedes Merkmal f_j;

9 [Trainiere einen schwachen Klassifikator h_j, verwende D_t];

10 Für jeden schwachen Klassifikator h_j;

11 Berechne: $r_j = \sum_{n=1}^{N} D_t(n) l_n h_j(e_n)$, mit $h_j(e_n) \in [+1, -1]$;

12 Wähle den schwachen Klassifikator h_j, der $|r_j|$ maximiert;

13 Setze $(h_t, r_t) = (h_j, r_j)$;

14 Aktualisiere die Gewichte:
$D_{t+1}(n) = D_t(n) exp(-\alpha_t l_n h_t(e_n))$, mit $\alpha_t = \frac{1}{2} ln(\frac{1+r_t}{1-r_t})$;

Rückgabe

15 $H(e) = sign(\sum_{t=1}^{T} \alpha_t h_t(e))$ // *Starker Klassifikator*

Abbildung 4.10: AdaBoost-Trainingsalgorithmus

Für den AdaBoost-Klassifikator wurde eine am Fachgebiet Neuroinformatik und Kognitive Robotik bereits bestehende C++ Implementierung optimiert. Diese basiert auf dem in [Arras et al., 2007] beschriebenen AdaBoost-Verfahrens und ist Teil der `BoostedLegDetector`-Unit. Die bestehende Implementierung wurde für den neuen Ansatz so geändert, dass sie als eigenständiges Programm, unabhängig vom Anwendungshintergrund, zum generellen AdaBoost-Training verwendet werden kann. Zudem wurden die Anzahl der benötigten Berechnungsschritte reduziert, indem die Sortierung

der Trainingswerte zur Bestimmung der Schwellwerte und der Paritäten (Abschnitt 2.1.1) vor der iterativen Bestimmung der schwachen Klassifikatoren durchgeführt wird. Abbildung 4.10 zeigt den algorithmischen Ablauf des AdaBoost-Trainings.

4.2.9 Klassifikation

Evaluation

Im Rahmen der Evaluation werden die über die `extractFeatureData`-Unit extrahierten Testdaten mit Hilfe der scikit-learn Bibliothek klassifiziert. Analog zum Training werden die Testdaten zunächst über die Parameter μ_{scale} und σ_{scale} skaliert. Abhängig davon, ob eine objektunspezifische oder eine objektspezifische Evaluation der Detektionsleistung durchgeführt wird (Abschnitt 5.2.2), wird nur die Gesamtanzahl der positiv und negativ klassifizierten Interest Points bzw. Layer über die Anwendung der trainierten Klassifikatoren ermittelt, oder für jedes Objekt einzeln. Über die objektspezifische Evaluation kann neben der besten Feature-Klassifikator-Kombination auch die optimale Anzahl der Interest Points bzw. Layer ermittelt werden, die positiv klassifiziert werden müssen, damit ein Cluster schließlich als liegende Person gilt. Abbildung 4.11 zeigt den algorithmischen Ablauf der objektunspezifischen Evaluation.

Anwendung

Neben der Evaluation der besten Feature-Klassifikator-Kombination ist das Ziel dieser Arbeit, einen Prototypen zur Sturzdetektion auf einem mobilen Roboter zu implementieren. Daher werden in der `Classification`-Unit die in der `FeatureExtraction`-Unit extrahierten Featurevektoren über einen trainierten Klassifikator klassifiziert. Die in Kapitel 5 beschriebene Evaluation zeigt, dass eine SVM mit einer radialen Basisfunktion als Kernel in Kombination mit dem HLSN für den hier betrachteten Anwendungsfall am besten geeignet ist. Daher nutzt die `Classification`-Unit zur Klassifikation die C++ Implementierung dieser SVM aus der LIBSVM-Bibliothek [Chang und Lin, 2011].

Eingaben

1 $S = [(v_1, l_1), \ldots, (v_N, l_N)]$ // *Set mit Testbeispielen*
// *$l_n = +1$ für positive Beispiele*
// *$l_n = -1$ für negative Beispiele*

2 clf // *Klassifikator*

3 μ_{scale} // *Skalierungsparameter: Amplitudenmittelwert*

4 σ_{scale} // *Skalierungsparameter: Varianz*

5 $score_c$ // *Klassifikationssicherheit*

Initialisierung

6 $TP \leftarrow 0$; // *Anzahl richtig positiv klassifizierter Objekte*

7 $TN \leftarrow 0$; // *Anzahl richtig negativ klassifizierter Objekte*

8 $FN \leftarrow 0$; // *Anzahl falsch negativ klassifizierter Objekte*

9 $FP \leftarrow 0$; // *Anzahl falsch positiv klassifizierter Objekte*

Algorithmus

10 `Skaliere` S `mit:` $v_m = (v_n - \mu) * \sigma$; // *Skalierung*

11 `Für jedes` $v_j \in S$;

12 `Klassifiziere` v_j `über` clf:; // *Klassifikation*

13 `Wenn` $clf(v_j) > score_c$: $l_{pred} = +1$,;

14 `Sonst:` $l_{pred} = -1$;

15 `Wenn` $l_{pred} = l_j = +1$: $TP + 1$;

16 `Wenn` $l_{pred} = l_j = -1$: $TN + 1$;

17 `Wenn` $(l_{pred} \neq l_j) \,\&\, (l_j = +1)$: $FN + 1$;

18 `Wenn` $(l_{pred} \neq l_j) \,\&\, (l_j = -1)$: $FP + 1$;

Rückgabe

19 $RC = \frac{TP}{TP+FN}$ // *Recall*

20 $PR = \frac{TP}{TP+FP}$ // *Precision*

21 $AC = \frac{TP+TN}{TP+TN+FP+FN}$ // *Accuracy*

22 $F_{0.5} = \frac{(1+0.5^2) \cdot PR \cdot RC}{0.5^2 \cdot PR + RC}$ // *$F_{0.5}$-Wert*

Abbildung 4.11: Zur objektunspezifischen Evaluation angewendeter Klassifikationsalgorithmus

Pro Cluster wird ermittelt, wie viele seiner Interest Points bzw. Layer positiv klassifiziert werden. Liegt diese Anzahl über der im Rahmen der Evaluation ermittelten Anzahl $min\_pos\_layer = 3$, gilt der segmentierte Cluster als gestürzte Person.

Über das Ergebnisse der `Classification`-Unit kann in der praktischen Anwendung eine Alarm-Unit gesteuert werden, die nach der Detektion einer gestürzten Person entsprechende Notfallmaßnahmen einleitet. Abbildung 4.12 zeigt den algorithmischen Ablauf der in der Anwendung eingesetzten Klassifikation.

Eingaben

1 $\mathbf{F}_k = \{\mathbf{f}_1, \ldots, \mathbf{f}_{l_n}\}$ *// Vektor mit Featurevektoren der l_n Layer des k-ten Clusters*
2 clf *// Klassifikator*
3 μ_{scale} *// Skalierungsparameter: Amplitudenmittelwert*
4 σ_{scale} *// Skalierungsparameter: Varianz*
5 l_{min} *// Mindestanzahl an Layern, die bei einer // Person positiv klassifiziert werden müssen*

Initialisierung

6 $l_{pos} \leftarrow 0$; *// Anzahl positiv klassifizierten Layer des Clusters*
7 $Fall \leftarrow false$; *// Boolsche Variable zur Anzeige, ob eine gestürzte Person detektiert wurde*

Algorithmus

8 `Für jeden` $\mathbf{f}_j \in \mathbf{F}_k$`;`
9 `Skaliere` $\mathbf{f}_j$ `mit:` $f_m = (f_n - \mu) * \sigma$`;` *// Skalierung*
10 `Klassifiziere` $\mathbf{f}_j$ `über` clf`:;` *// Klassifikation*
11 `Wenn` $clf(v_j) > 0.5$`:` $l_{pos} + 1$`,;`
12 `Wenn` $l_{pos} \geq l_{min}$`:` $Fall = true$`;`

Rückgabe

13 $Fall$

Abbildung 4.12: In der Anwendung eingesetzter Klassifikationsalgorithmus

Unit	Parameter	Wert
`PreProcessing`	*downsample_leaf_size*	3 cm
`PreProcessing` `Segmentation`	*kinect_high*	90 cm
`GroundPlaneEstimation`	*RANSAC_distance_max*	10 cm
`GroundPlaneEstimation`	*RANSAC_angle_max*	2°
`Segmentation`	*clust_size_min*	500
`Segmentation`	*clust_size_max*	5000
`Segmentation`	*clust_dist_max*	3 cm
`Segmentation`	*clust_high_diff*	5 cm
`Layering`	*layer_size*	22.52 cm
`Layering`	*layer_overlap*	2.5 cm
`FeatureExtractionPoints`	*SURE_normals_scale*	4 cm
`FeatureExtractionPoints`	*SURE_sampling_rate*	4 cm
`FeatureExtractionPoints`	*SURE_hist_scale*	12 cm
`FeatureExtractionPoints`	*SURE_normals_sampling_rate*	4 cm
`FeatureExtractionPoints`	*SURE_entropie_min*	0.6
`FeatureExtractionPoints`	*SURE_cornerness_min*	0.15
`FeatureExtractionLayer`	*normals_radius*	4 cm
`FeatureExtractionLayer`	*FPFH_neighbors_radius*	8 cm
`Classification`	*min_pos_layer*	3

Tabelle 4.2: Parametereinstellungen des neuen Ansatzes

Tabelle 4.2 fasst alle im neuen Ansatz verwendeten Parametereinstellungen zusammen.

Kapitel 5

Evaluation und Ergebnisse

Zur Bestimmung der für die Detektion gestürzter Personen am besten geeigneten Feature-Klassifikator-Kombination wurde eine ausführliche Evaluation durchgeführt. Im folgenden Kapitel werden die verwendete Datenbasis vorgestellt, die zur Evaluation angewendete Methodik beschrieben und die erreichten Ergebnisse diskutiert.

5.1 Datenbasis

Zum Training der Klassifikatoren und zur Evaluation des Verfahrens wurden mit der Kinect der mobilen Roboterplattform Scitos-G3 Videos aufgenommen, die die Tiefendaten der Kinect beinhalten. Die Mehrheit der Videos enthält zudem die korrespondierenden RGB-Daten.

Die Basis der positiven Trainings- und Testdaten bilden Videos, die auf dem Boden liegende Personen in verschiedenen Posen zeigen. Für die Trainingsdaten wurden nur Videos verwendet, in denen der Körper der Person komplett abgebildet ist. Dadurch wird gewährleistet, dass alle Körperbereiche den selben Anteil an den Trainingsdaten haben und keine Überspezialisierung des layerunspezifischen Klassifikators an einen bestimmten Körperbereich erfolgt. Die für die Testdaten verwendeten Videos beinhalten vor allem partiell verdeckte Personen, da dieses die in der praktischen Anwendung bestehende Situation realistischer abbildet als komplett abgebildete Personen (Abschnitt 1.2). Da der Fokus dieser Arbeit auf der Evaluation und Bewertung geeigneter Featu-

Positiv	Anzahl Videos	Dauer [hh:mm:ss]	Anzahl Personen
Training	11	00:06:09	9
Test	13	00:06:33	10
Gesamt	24	00:12:42	12

Samplinrate Kinect: 30 Hz

Negativ	Anzahl Videos	Dauer [hh:mm:ss]	
AWO	6	00:07:39	5 Whg. + 1 Raum
HomeLab	9	00:10:49	
Hunde	5	00:04:05	2 Hunde
Gesamt	20	00:22:33	

Tabelle 5.1: Eigenschaften der als Datenbasis verwendeten Videos

res und nicht auf der Segmentierung liegt, werden ausschließlich Videos verwenden, in denen die Person leicht vom Hintergrund zu trennen ist. Die partiellen Verdeckungen werden daher nur durch den gewählten Bildbereich, durch die Person selbst oder durch von der Person weit entfernte Objekte hervorgerufen.

Zur Generierung realistischer negativ Daten wurden Videos verwendet, die in fünf verschiedenen Wohnungen und einem Gemeinschaftsraums eines Seniorenwohnheims der AWO in Erfurt aufgenommen wurden. Zum Schutz der Privatsphäre der Bewohner stehen in diesen Videos keine RGB-Daten zur Verfügung. Szenen, in denen der Roboter lange auf derselben Stelle steht, wurden aus den Videos rausgeschnitten. Dadurch kommen die Featurevektoren keines Objekts überdurchschnittlich oft in den Trainings- oder Testdaten vor. Die Videos der AWO enthalten jedoch nur wenige auf dem Boden liegende Objekte, deren Größe und Form einer liegenden Person ähnlich sind und somit eine größere Herausforderung bei der Detektion gestürzter Personen darstellen. Daher wurden im HomeLab des Fachgebiets für Neuroinformatik und Kognitive Robotik der TU Ilmenau zusätzlich Videos mit Objekten aufgenommen, deren extrahierte Punktwolke in Form und Größe ähnlich der einer gestürzten Person ist. Da vor allem große (liegende) Hunde nicht nur eine Größe aufweisen, die gestürzten Personen ähnlich ist, sondern durch ihre unregelmäßige Form auch eine ähnliche Orientierung der Oberflächennormalen aufweisen, wurden zusätzlich Videos mit zwei größeren Hunden in verschiedenen Posen aufgenommen.

Tabelle 5.1 gibt Details zu den verwendeten Videos und Abbildung 5.1 zeigt einzelne Beispielbilder bzw. Tiefenkarten und die korrespondierenden Punktwolken.

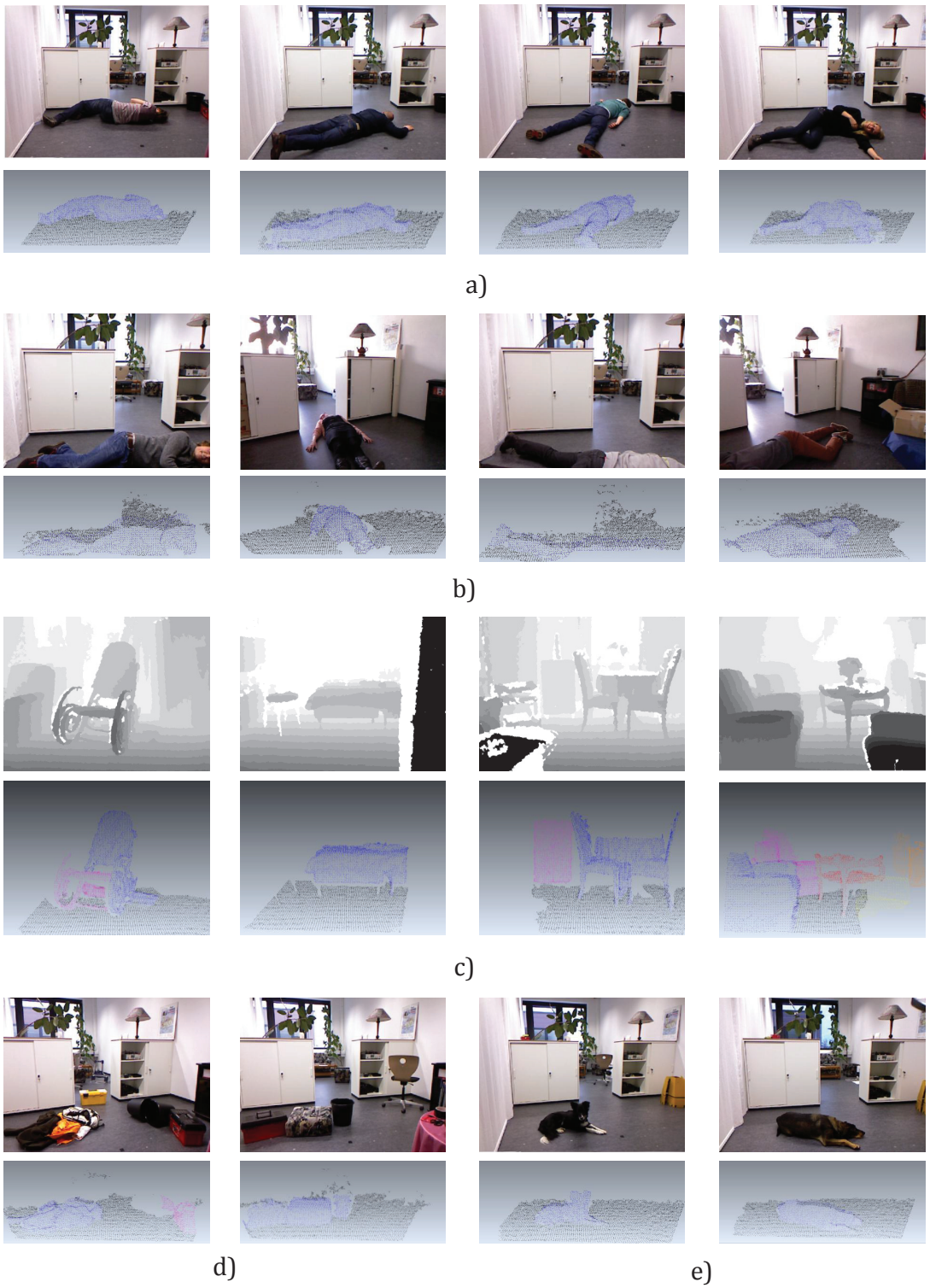


Abbildung 5.1: Beispielbilder und segmentierte Punktwolken der verwendeten Trainings- und Testdaten

Positiv: a) Training, b) Test. Negativ: c) AWO, d) HomeLab, e) Hund.

		Layer / IP		Objekte	
		pos	neg	pos	neg
GSF	Training	41.920	50.026	5.240	16.969
	Test	11.943	16.245	1.906	5.507
HLSN	Training	44.384	45.747	5.548	15.598
	Test	13.493	14.445	2.174	5.074
FPFH	Training	36.912	37.747	4.614	12.676
	Test	14.587	12.236	2.368	4.130
SURE	Training	35.368	42.667	3.850	10.537
	Test	12.906	13.722	2.106 +26 o. IP	3.407 + 664 o. IP

Tabelle 5.2: Anzahl der extrahierten Layer bzw. Interest Points und der berücksichtigten Objekte

Mit Hilfe der `extractFeatureData`-Unit wurden aus den beschriebenen Videos die Daten der Featurevektoren für alle vier Features extrahiert und in Trainings- und Testdaten aufgeteilt. Wie oben beschrieben gelten bei den positiven Daten für die Inhalte der Trainings- und Testvideos unterschiedliche Anforderungen. Daher lassen sich die positiven Daten entsprechend ihrer Videozugehörigkeit aufteilen. Die negativen Daten wurden so aufgeteilt, dass etwa 2/3 der extrahierten Objekte den Trainings- und das andere 1/3 den Testdaten zugewiesen wird. Um sicherzustellen, dass die Featurevektoren des selben Objekts nicht in den Trainings- und Testdaten gleichzeitig vorkommen, wurden die Featurevektoren von 90 in einem Video zeitlich nacheinander extrahierten Objekten in einem Block zusammengefasst und zusammen entweder den Trainings- oder den Testdaten zugewiesen. Da der Roboter sich bei der Aufnahme der negativen Videos bewegte bzw. das aufgenommene Objekt (Hund) sich bewegt hat, kann davon ausgegangen werden, dass sich die beobachtete Szene nach einer kurzen Zeit verändert hat. Über das Blocking werden somit die zum selben Objekt zugehörigen und in mehreren Frames mehrfach extrahierten Featurevektoren den Trainings- oder Testdaten zusammen zugewiesen.

Tabelle 5.2 zeigt die Anzahl der über die Layer bzw. Interest Points (IP) extrahierten Featurevektoren und die Anzahl der berücksichtigten Objekte für alle vier Fea-

tures. GSF bezeichnet dabei die geometrischen und statistischen Features, HLSN das Histogram of Local Surface Normals, FPFH das Fast Point Feature Histogram und SURE den Interest Point-basierten SURE-Deskriptor. Die Anzahl der berücksichtigten Videoframes ist abhängig von der Berechnungsdauer der einzelnen Features. Um für die vier Features eine vergleichbare Anzahl an Beispieldaten zu erhalten, wurden die unterschiedlichen Berechnungszeiten über die Steuerung der Abspielgeschwindigkeit des Videos ausgeglichen. Da die in der AWO aufgenommenen Videos mit negativ Beispielen nur sehr wenig auf dem Boden liegende Objekte enthalten, wurde bei der Extraktion der Daten die Höhe der Region of Interest auf 130 cm erhöht, um so eine größere Zahl an Trainings- und Testdaten zu generieren.

5.2 Evaluation

Zur Bestimmung der besten Feature-Klassifikator-Kombination wurde eine ausführliche Evaluation durchgeführt, in der alle vier Features (Abschnitt 4.2.7) mit den vier Klassifikationsverfahren (Abschnitt 4.2.8) kombiniert und getestet wurden. Dabei wurden zusätzlich einzelne Parameter der Klassifikatoren variiert.

5.2.1 Evaluierte Parameter

Bei dem Nearest Neighbour Klassifikator (NN) wurde die Anzahl der k Nachbarn in fünf Stufen variiert:

$$k_{param} = [5,\ 10,\ N^{\frac{2}{8}},\ N^{\frac{3}{8}},\ 100], \tag{5.1}$$

mit N als Dimension des jeweiligen Featurevektors.

Für die Support Vecto Machine (SVM) wurden die drei in Abschnitt 2.1.1 beschriebenen Kernelfunktionen getestet: linearer Kernel ($linear$), polynomiieller Kernel ($poly$) und ein Kernel mit einer radialen Basisfunktion (rbf),

$$kernel_{param} = [linear,\ poly,\ rbf]. \tag{5.2}$$

Alle drei Kernelfunktionen wurden mit unterschiedlichen Strafparametern C des Fehlerterms getestet:

$$C_{param} = [1, 10, 100, 1000]. \tag{5.3}$$

Da das Training der linearen SVM mit $C = 1000$ mehr als 48 Stunden dauert und die Ergebnisse beim getesteten HLSN keine deutlichen Unterschiede zur linearen SVM mit $C = 100$ aufweisen, wurde diese Parametervariation aus zeitlichen Gründen für F1, F3 und F4 nicht weiter getestet.

Zusätzlich zum C-Parameter wurde bei der polynomiiellen Kernelfunktion der Grad d der Funktion in 3 Stufen variiert:

$$d_{param} = [2, 3, 4]. \tag{5.4}$$

Bei Kernels mit einer radialen Basisfunktion wurde zwei verschiedene γ-Werte getestet:

$$\gamma_{param} = [1e^{-3}, 1e^{-4}] \tag{5.5}$$

Der Random Forests Klassifikator (RF) wurde über die Anzahl der Bäume nT in vier Stufen variiert:

$$nT_{param} = [10, 50, 100, 500] \tag{5.6}$$

Die verwendeten Variationen entsprechen den Empfehlungen der scikit-learn Bibiliothek bzw. beim Nearest Neighbour zusätzlich den Empfehlungen aus [Enas und Choi, 1986] (Abschnitt 2.1.1).

Für den AdaBoost Klassifikator (AB) wurden analog zu [Spinello et al., 2010] 20 schwache Klassifikatoren verwendet (Abschnitt 2.1.1).

5.2.2 Evaluationsmethodik

Im Rahmen der Evaluation wurden alle 33 bzw. 34 Klassifikatoren (SVM, *linear*, $C = 1000$ nur für das HLSN) für jedes Feature mit den in Tabelle 5.2 beschriebenen Trainingsdaten trainiert.

Objektunspezifische Evaluation Um zunächst die grundsätzliche Leistung jedes Klassifikators bezüglich der Trennung der zwei Klassen (Person und Objekt) zu untersuchen, wurde zunächst die objektunspezifische Detektionsleistung evaluiert. Hierbei wurden die Interest Points bzw. Layer aller in Tabelle 5.2 beschriebenen Testdaten über die trainierten Klassifikatoren einer der zwei Klassen zugeordnet und die Anzahl der richtig und falsch klassifizierten Interest Points bzw. Layer objektunabhängig aufsummiert.

Objektspezifische Evaluation Für den beste Klassifikator jedes Features wurde weiter die objektspezifische Detektionsleistung untersucht. Hierzu wurde analysiert, wie viele Interest Points bzw. Layer pro Objekt welcher Klasse zugeordnet werden. Bei dem Interest Point basierten SURE-Deskriptor wurde bei der Berechnung der FN- und TN-Detektionen zusätzlich die Anzahl der Objekte berücksichtigt, auf denen keine Interest Points detektiert wurden. Über die Variation der Mindestanzahl von Interest Points bzw. Layern, die positiv klassifiziert werden müssen, damit ein Cluster als gestürzte Person gilt, lässt sich die optimale Feature-Klassifikator-Kombination mit der optimalen Parametereinstellung ermitteln.

Berechnungszeit Da für die praktische Anwendung des Verfahrens eine Echtzeitfähigkeit vorausgesetzt wird, wurde neben der Detektionsleistung auch die zur Klassifikation benötigte Rechenzeit jedes Klassfikators bestimmt. Hierzu wurde die Zeit gemessen, die der Klassifikator braucht, um alle Featurevektoren der Testdaten zu klassifizieren. Diese wurde mit der Anzahl der klassifizierten Featurevektoren ins Verhältnis gesetzt. Über die so bestimmten Zeiten kann zwar keine Aussage getroffen werden, wie hoch die Berechnungszeit eines Klassifikators in der praktischen Anwendung beispielsweise pro Frame ist, sie bietet jedoch die Möglichkeit, die Berechnungszeiten der Klassifikatoren untereinander zu vergleichen. Einzig die Zeiten des AdaBoost-Klassifikators sind mit denen der anderen Klassifikatoren nicht vergleichbar, da dieser nicht über die scikit-Bibiliotheken in phyton evaluiert wurde, sondern in Matlab. Um einen guten Kompromiss zwischen Berechnungsdauer und Klassifikationsgenauigkeit

zu finden, wurde die Feature-Klassifikator-Kombination mit der höchsten Detektionsleistung mit dem schnellsten Klassifikator dieses Features verglichen.

Gütemaß In der praktischen Anwendung des Verfahrens ist geplant, dass der Roboter Kontrollfahrten vornimmt, bei denen er die Wohnung des Nutzers abfährt und diese auf kritische Situationen untersucht. Eine hohe FP-Rate würde zu regelmäßigen Fehlalarmen führen, die in der Praxis voraussichtlich eine Abschaltung des Systems zur Folge hätten. Im Gegensatz dazu ist eine leicht erhöhte FN-Rate in der praktischen Anwendung nicht so kritisch, da der Roboter auf seiner Kontrollfahrt die gestürzte Person mehrfach und im Regelfall auch aus mehreren Ansichten sieht, d. h. er hat mehrere Chancen die gestürzte Person zu detektieren. Daher wird als Gütemaß in dieser Arbeit der $F_{0.5}$-Wert verwendet (Abschnitt 2.5), der den Precision-Wert doppelt so stark gewichtet wie den Recall-Wert. Den FP-Detektionen wird somit eine höhere Bedeutung zugeschrieben.

5.3 Ergebnisse

5.3.1 Ergebnisse: objektunspezifische Evaluation

Die umfangreichen Ergebnisse aller 133 Feature-Klassifikator-Kombinationen, die im Rahmen der objektunspezifische Evaluation (Abschnitt 5.2.2) getestet wurden, können den Tabellen in Anhang B entnommen werden.

Tabelle 5.3 zeigt die Ergebnisse der besten Parametereinstellung jedes Klassifikators für alle vier Features bezüglich der objektunspezifischen Evaluation. Die Ergebnisse zeigen, dass der optimale Klassifikator von dem verwendeten Feature abhängig ist und keine allgemeine Aussage getroffen werden kann, welche Klassifikationsmethodik die beste ist. Bei den ersten drei Features schneidet der Random Forests Klassifikator mit 500 Bäumen, der Nearest Neighbour Klassifikator mit einer kleinen Nachbarschaft, sowie die SVM mit polynomiellen Kernel gut ab. Die am wenigsten geeignete Methode ist AdaBoost. Diese liefert bei allen vier Features die geringste Detektionsrate.

Geometrische und statistische Features (GSF)										
Klassifikator				TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
RF	$nT500$			7653	1886	14359	4290	80,23%	64,08%	78,09%	76,38%
NN	$k10$			7164	1975	14270	4779	78,39%	59,98%	76,04%	73,86%
SVM	*poly*	$C1000$	$d4$	9037	4685	11560	2906	65,86%	75,67%	73,07%	67,61%
AB	$wc20$			8884	6066	10179	3059	59,42%	74,39%	67,63%	61,92%

Histogram of Local Surface Normals (HLSN)										
Klassifikator				TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
RF	$nT500$			10955	236	14209	2538	97,89%	81,19%	90,07%	94,02%
NN	$k17$			11465	532	13913	2028	95,57%	84,97%	90,84%	93,24%
SVM	*poly*	$C10$	$d4$	10986	470	13975	2507	95,90%	81,42%	89,34%	92,60%
AB	$wc20$			10374	1794	12651	3119	85,26%	76,88%	82,41%	83,44%

Fast Point Feature Histogram (FPFH)										
Klassifikator				TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
NN	$k10$			10707	1168	11068	3979	90,16%	72,91%	80,88%	86,09%
SVM	*poly*	$C1000$	$d3$	10554	1154	11082	4032	90,14%	72,36%	80,67%	85,92%
RF	$nT500$			10153	1110	11126	4433	90,14%	69,61%	79,33%	85,12%
AB	$wc20$			10386	2756	9480	4200	79,03%	71,21%	74,07%	77,33%

Surface Entropie (SURE)										
Klassifikator				TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
SVM	*rbf*	$C1$	$\gamma 1e^{-4}$	99951	4687	9035	2955	67,98%	77,10%	71,30%	69,63%
RF	$nT100$			3445	3079	10643	9461	52,81%	26,69%	52,91%	44,16%
NN	$k67$			3958	4208	9514	8948	48,47%	30,67%	50,59%	43,43%
AB	$wc20$			3771	5229	8493	9135	41,90%	29,22%	46,06%	38,55%

Tabelle 5.3: Ergebnisse der objektunspezifischen Evaluation

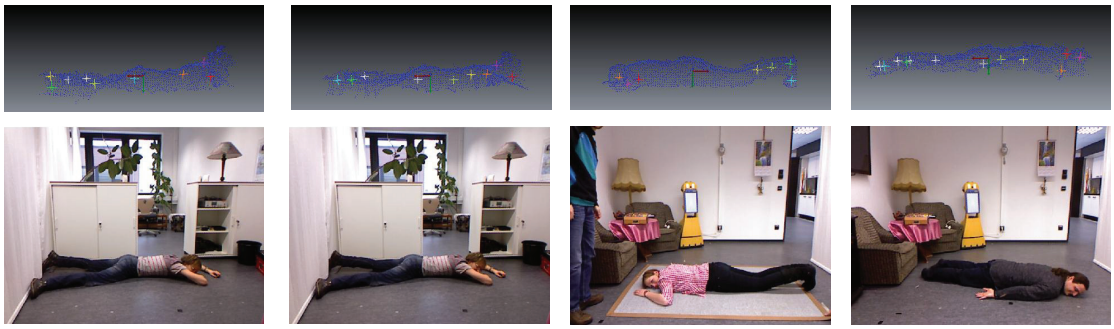


Abbildung 5.2: Interest Point Detektion

Die Positionen der detektierten Interest Points sind bei liegenden Personen instabil.

Auffällig ist, dass der auf Interest Points basierende SURE-Deskriptor eine deutlich schlechtere Detektionsleistung erreicht, als die drei auf Layer basierenden Features. Da der SURE-Deskriptor dem FPFH sehr ähnlich ist, lässt sich die deutlich schlechtere Leistung auf die verwendeten Interest Points zurückführen. Wie Abbildung 5.2 zeigt, ist die Detektion der Interest Points bei liegenden Personen instabil. Die Positionen der Interest Points konzentrieren sich zwar auf einzelne Bereiche (hauptsächlich auf den unteren Beinbereich und den oberen Kopf-/Armbereich), innerhalb dieser Bereiche variieren die als Interest Points detektierten Punkte jedoch stark.

Wie die objektunspezifische Evaluation weiter zeigt, ist die Detektionsleistung des HLSN deutlich besser, als die der anderen drei Features. Selbst die Leistung des schlechtesten HLSN-Klassifikators (AdaBoost, $F_{0.5} = 83.44\%$) ist höher oder vergleichbar mit denen der besten Klassifikatoren der anderen drei Features (GSF: $F_{0.5} = 76,38\%$, FPFH: $F_{0.5} = 86,09\%$, SURE: $F_{0.5} = 69,63\%$).

Abbildung 5.3 illustriert den Vergleich der objektunspezifischen Detektionsleistungen der verschiedenen Feature-Klassifikator-Kombinationen über eine *ROC*-Kurve. Diese wurde über die Variation der Klassifikationssicherheit generiert. Verglichen werden die besten Klassifikatoren des GSF, FPFH und SURE, sowie der beste und der schnellste Klassifikator des HLSN. Die über die Fläche unter der Kurve ermittelten *AP*-Werte (s. Legende) bestätigen die auf Basis des $F_{0.5}$-Werts getätigten Aussagen.

	Klassifikator				Min pos. L./IP.	TP	FP	RC	PR	AC	$F_{0.5}$
GSF	RF	$nT500$			3	1464	63	76.81%	95.87%	93.19%	91.34%
HLSN	RF	$nT500$			2	2086	25	95.95%	98.82%	98.44%	98.23%
HLSN	SVM	poly	$C1000$	$d4$	3	1895	5	87.17%	99.74%	96.08%	96.94%
FPFH	NN	$k10$			3	1953	50	82.47%	97.50%	92.84%	94.08%
SURE	SVM	rbf	$C1$	$\gamma 1e^{-4}$	4	1175	285	55.79%	80.48%	77.94%	73.94%

Tabelle 5.4: Ergebnisse der objektspezifischen Evaluation

5.3.2 Ergebnisse: Objektspezifische Evaluation

Die für den besten Klassifikator jedes Features durchgeführte objektspezifische Evaluation (Abschnitt 5.2.2) bestätigt die auf Basis der Ergebnisse der objektunspezifischen Evaluation vorgenommenen Einschätzungen. Abbildung 5.3 zeigt die *ROC*-Kurve der besten Klassifikatoren des GSF, FPFH und SURE, sowie die des besten und des schnellsten Klassifikators des HLSN. Die objektspezifische Detektionsleistung ist abhängig von der Anzahl der Layer bzw. Interest Points, die pro Cluster mindestens positiv klassifiziert werden müssen, damit der Cluster als gestürzte Person gilt. Daher wurden diese Kurve über die Variation des Parameters min_pos_layer/ip generiert. Tabelle 5.4 zeigt die zugehörigen Leistungsmaßzahlen der betrachteten Feature-Klassifikator Kombinationen, jeweils für den min_pos_layer/ip-Wert, bei dem der $F_{0.5}$-Wert maximal ist. Die Leistung des HLSN übertrifft auch bei der objektspezifischen Evaluation die der anderen drei Features. Daher wird das HLSN als am besten zur Detektion gestürzter Personen geeignetes Feature bewertet.

5.3.3 Ergebnisse: Berechnungszeit

Wie Tabelle A.8 im Anhang dieser Arbeit zeigt, ist ein Random Forests Klassifikator mit 500 Bäumen nach der in dieser Arbeit verwendeten Evaluationsmethodik der beste Klassifikator für das HLSN. Die zur Klassifikation benötigte Rechenzeit ist mit 20.16 ms/Objekt jedoch sehr hoch. Der schnellste HLSN-Klassifikator ist eine SVM mit polynomiellen Kernel, $C = 1000$ und $d = 4$. Mit 0.178 ms/Objekt benötigt diese

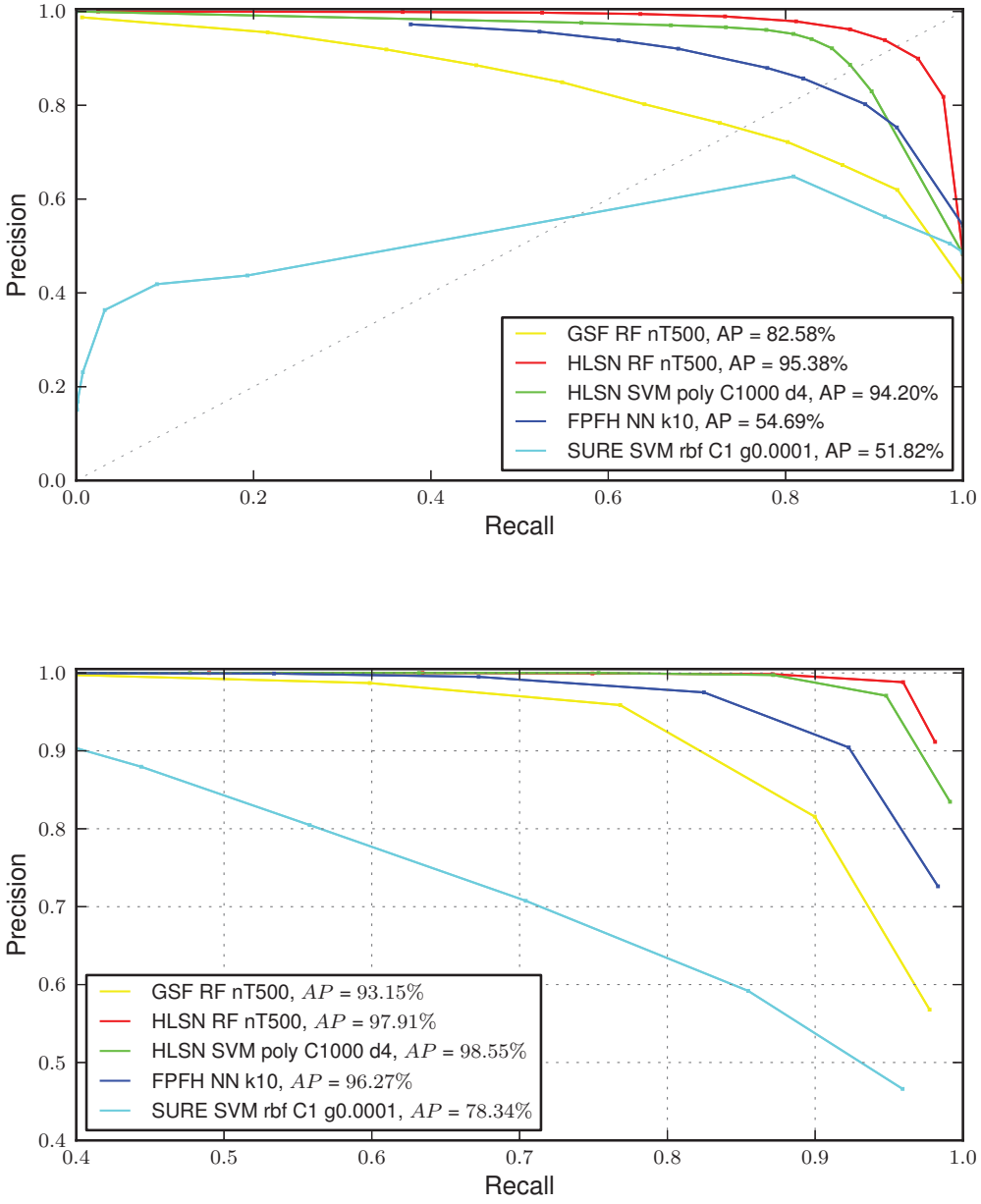


Abbildung 5.3: ROC-Kurven der objektunspezifischen (oben) und objektspezifischen (unten) Evaluationsergebnisse

Verglichen werden die besten Klassifikatoren des GSF, FPFH und SURE, sowie der beste und der schnellste Klassifikator des HLSN.

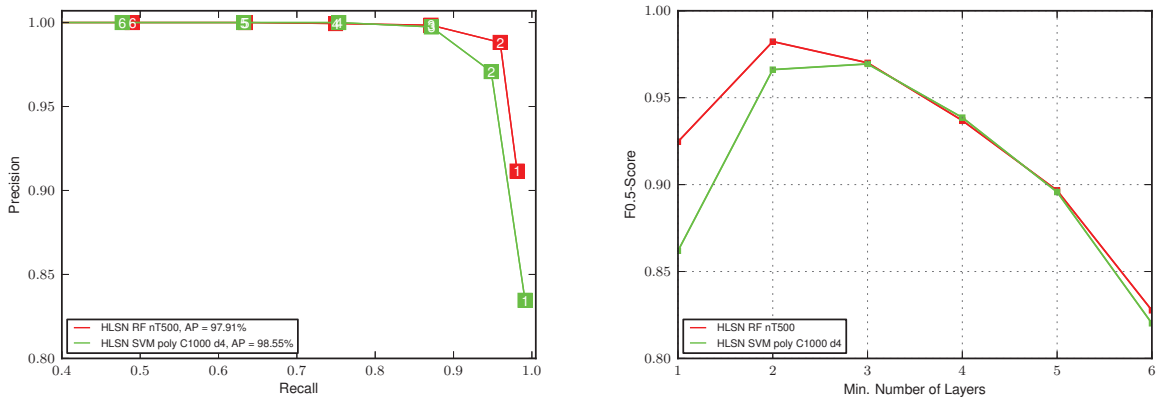


Abbildung 5.4: Vergleich des besten und des schnellsten HLSN-Klassifikators
Links: ROC-Kurve. Rechts: Abhängigkeit des $F_{0.5}$-Werts vom min_pos_layer-Wert.

zur Klassifikation der Testdaten nur 1% der Zeit des Random Forests Klassifikators. Wie Tabelle 5.4 zeigt, liegt die Detektionsleistung der SVM nur leicht unter der des Random Forests Klassifikators. Abbildung 5.4 zeigt die *ROC*- und $F_{0.5}$-Kurve der beiden Klassifikatoren in Abhängigkeit des *min_pos_layer*-Werts. Da der Leistungsverlust im Vergleich zur Reduktion des Rechenaufwands gering ist, wird die SVM für den in dieser Arbeit betrachteten Anwendungsfall als besser geeignet bewertet, als der Random Forests Klassifikator.

5.3.4 Ergebnisse: Mindestanzahl positiver Layer

Wie Tabelle 5.4 und die $F_{0.5}$-Kurve in Abbildung 5.4 zeigen, erreicht die SVM mit $min\_pos\_layer = 3$ den höchsten $F_{0.5}$-Wert. Daher wird das HLSN in Kombination mit einer SVM mit polynomiellen Kernel, $C = 1000$ und $d = 4$, sowie einer Mindestanzahl von 3 Layern, die pro Cluster positiv klassifiziert werden müssen, als zur Detektion von gestürzten Personen über eine mobile Roboterplattform am besten geeignet bewertet. Mit einem $F_{0.5}$-Wert von 96.94% und einer AP von 98.55% erreicht das vorgestellte System eine sehr gute Detektionsleistung.

Kapitel 6

Schlussfolgerung und Ausblick

Die in Abschnitt 5.3 vorgestellten Ergebnisse zeigen, dass die in diesem Ansatz entwickelte Methodik geeignet ist zur Detektion gestürzter Personen mit Hilfe einer mobilen Roboterplattform. Das Histogram of Local Surface Normals (HLSN) in Kombination mit einer SVM mit polynomiellen Kernel, $C = 1000$ und $d = 4$ bei einer Mindestanzahl von 3 Layern, die pro Cluster positiv klassifiziert werden müssen, erreicht mit einem $F_{0.5}$-Wert von 96.94% und einer $AP = 98.55\%$ eine sehr gute Detektionsleistung. Die geringe FP-Rate von 0.1% führt zu einer sehr geringen Anzahl an Fehlalarmen, was für den praktischen Einsatz des Verfahrens von erheblicher Bedeutung ist.

Bei diesen Ergebnissen ist jedoch zu beachten, dass dem Fokus dieser Arbeit geschuldet, nur Testdaten verwendet wurden, in denen die liegende Person leicht von ihrer Umgebung zu segmentieren ist. Für eine allumfassende Aussage zur Detektionsleistung des vorgestellten Ansatzes im praktischen Einsatz müsste zusätzlich die Leistung des Segmentierungsverfahrens analysiert werden. Diese Analyse würde den Rahmen dieser Arbeit jedoch sprengen und ist eine interessante Fragestellung für Folgebetrachtungen.

Das euklidische Clustering ist zur Segmentierung der Punktwolke, für die in dieser Arbeit durchgeführte Evaluation von Features zur Detektion gestürzter Personen, geeignet. Dies zeigen die Trainings- und Testdaten, in denen die Person leicht vom Hintergrund zu trennen ist. In der praktischen Anwendung wird dieses relativ einfache Verfahren vermutlich jedoch nicht ausreichen, um gute Segmentierungsergebnisse zu erzielen. Bei partiellen Verdeckungen durch Objekte die nah an der Person stehen,

würde das verwendete Verfahren diese zusammen mit der Person als ein Cluster segmentieren. Abhilfe könnte hier die zusätzliche Verwendung von RGB-Daten schaffen. Wie in [Taylor und Cowley, 2011] lässt sich die mit einem Objekt in einem Cluster segmentierte Person über eine zusätzliche Analyse der Farbkante zwischen Objekt und Person von diesem trennen, was in einer weiterführenden Arbeit noch zu belegen wäre.

Der im Rahmen dieser Arbeit entwickelte Ansatz basiert auf einem layerunspezifischen Klassifikator, der trotz seiner geringen Komplexität zu einer sehr guten Detektionsleistung führt. Es ist weiterhin denkbar, einen layerspezifischen Klassifikator zu trainieren und diesen in Kombination mit einem, dem ISM ähnlichen Voting-Modell [Leibe et al., 2008] anzuwenden. Die Spezialisierung des verwendeten Modells würde zu einer weiteren Reduzierung der FP-Detektionen führen. Die FP-Rate des in dieser Arbeit entwickelten Ansatzes ist mit 0.1% jedoch bereits sehr gering. Die geringe zusätzliche Leistungssteigerung über die Verwendung eines layerspezifischen Detektionsverfahrens ist daher mit einer deutlich gesteigerten Komplexität und einem damit verbundenen erhöhten Rechenaufwand verbunden.

Zusammenfassend ist festzustellen, dass der neue Ansatz die in der vorliegenden Arbeit gestellten Anforderungen erfüllt. Der entwickelte Ansatz und die durchgeführte Evaluation bieten eine sehr gute Basis für einen praxistauglichen Sturzdetektor auf einer mobilen Roboterplattform. Ältere Menschen bekommen durch diesen die Möglichkeit, in ihrer häuslichen Umgebung sicherer zu leben: ihre Lebensqualität steigt.

Anhang A

Überblick über die Evaluationsergebnisse

Die Tabellen auf den folgenden Seiten zeigen die vollständigen Ergebnisse der objektunspezifischen Evaluation, bei der die Anzahl der richtig und falsch klassifizierten Interest Points bzw. Layer objektunabhängig aufsummiert wurden.

NN - GSF									
Klassifikator		TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
NN	$k = 5$	7496	2256	13989	4447	76,87 %	62,76 %	76,22 %	73,56 %
NN	$k = 10$	7164	1975	14270	4779	78,39 %	59,98 %	76,04 %	73,86 %
NN	$k = 17$	7662	2380	13865	4281	76,30 %	64,15 %	76,37 %	73,52 %
NN	$k = 73$	7782	2689	13556	4161	74,32 %	65,16 %	75,70 %	72,29 %
NN	$k = 100$	7754	2726	13519	4189	73,99 %	64,93 %	75,47 %	71,98 %

Tabelle A.1: Objektunspezifische Evaluation:
Nearest Neighbour (NN) - Geometrische und statistische Features (GSF)

AB - GSF									
Klassifikator		TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
AB	$T = 20$	8884	6066	10179	3059	59,42 %	74,39 %	67,63 %	61,92 %

Tabelle A.2: Objektunspezifische Evaluation:
AdaBoost (AB) - Geometrische und statistische Features (GSF)

RF - GSF									
Klassifikator		TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
RF	$nT = 10$	8082	2391	13854	3861	77,17 %	67,67 %	77,82 %	75,06 %
RF	$nT = 50$	7705	1973	14272	4238	79,61 %	64,51 %	77,97 %	76,05 %
RF	$nT = 100$	7677	1927	14318	4266	79,94 %	64,28 %	78,03 %	76,22 %
RF	$nT = 500$	7653	1886	14359	4290	80,23 %	64,08 %	78,09 %	76,38 %

Tabelle A.3: Objektunspezifische Evaluation:
Random Forests (RF) - Geometrische und statistische Features (GSF)

SVM - GSF										
Klassifikator				TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
SVM	*linear*	$C1$		8801	7047	9198	3142	55,53 %	73,69 %	63,85 %	58,41 %
SVM	*linear*	$C10$		8797	7040	9205	3146	55,55 %	73,66 %	63,86 %	58,42 %
SVM	*linear*	$C100$		8796	7042	9203	3147	55,54 %	73,65 %	63,85 %	58,41 %
SVM	*rbf*	$C1$	$g1e^{-3}$	9245	7269	8976	2698	55,98 %	77,41 %	64,64 %	59,26 %
SVM	*rbf*	$C10$	$g1e^{-3}$	9095	6158	10087	2848	59,63 %	76,15 %	68,05 %	62,33 %
SVM	*rbf*	$C100$	$g1e^{-3}$	8774	5729	10516	3169	60,50 %	73,47 %	68,43 %	62,71 %
SVM	*rbf*	$C1000$	$g1e^{-3}$	8834	5591	10654	3109	61,24 %	73,97 %	69,14 %	63,42 %
SVM	*rbf*	$C1$	$g1e^{-4}$	9337	7657	8588	2606	54,94 %	78,18 %	63,59 %	58,42 %
SVM	*rbf*	$C10$	$g1e^{-4}$	9238	7947	8298	2705	53,76 %	77,35 %	62,21 %	57,25 %
SVM	*rbf*	$C100$	$g1e^{-4}$	9036	7089	9156	2907	56,04 %	75,66 %	64,54 %	59,10 %
SVM	*rbf*	$C1000$	$g1e^{-4}$	9075	6123	10122	2868	59,71 %	75,99 %	68,10 %	62,38 %
SVM	*poly*	$C1$	$d2$	8666	5673	10572	3277	60,44 %	72,56 %	68,25 %	62,53 %
SVM	*poly*	$C10$	$d2$	8698	5552	10693	3245	61,04 %	72,83 %	68,79 %	63,08 %
SVM	*poly*	$C100$	$d2$	8613	5444	10801	3330	61,27 %	72,12 %	68,87 %	63,17 %
SVM	*poly*	$C1000$	$d2$	8666	5469	10776	3277	61,31 %	72,56 %	68,97 %	63,27 %
SVM	*poly*	$C1$	$d3$	9363	6044	10201	2580	60,77 %	78,40 %	69,41 %	63,63 %
SVM	*poly*	$C10$	$d3$	9021	5401	10844	2922	62,55 %	75,53 %	70,47 %	64,78 %
SVM	*poly*	$C100$	$d3$	8807	5080	11165	3136	63,42 %	73,74 %	70,85 %	65,25 %
SVM	*poly*	$C1000$	$d3$	8669	4849	11396	3274	64,13 %	72,59 %	71,18 %	65,66 %
SVM	*poly*	$C1$	$d4$	9669	6647	9598	2274	59,26 %	80,96 %	68,35 %	62,62 %
SVM	*poly*	$C10$	$d4$	9265	5685	10560	2678	61,97 %	77,58 %	70,33 %	64,57 %
SVM	*poly*	$C100$	$d4$	9046	5084	11161	2897	64,02 %	75,74 %	71,69 %	66,06 %
SVM	*poly*	$C1000$	$d4$	9037	4685	11560	2906	65,86 %	75,67 %	73,07 %	67,61 %

Tabelle A.4: Objektunspezifische Evaluation:

Support Vector Machine (SVM) - Geometrische und statistische Features (GSF)

NN - HLSN									
Klassifikator		TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
NN	$k = 5$	11236	474	13971	2257	95,95 %	83,27 %	90,22 %	93,12 %
NN	$k = 10$	11124	446	13999	2369	96,15 %	82,44 %	89,92 %	93,05 %
NN	$k = 17$	11465	532	13913	2028	95,57 %	84,97 %	90,84 %	93,24 %
NN	$k = 72$	11714	697	13748	1779	94,38 %	86,82 %	91,14 %	92,77 %
NN	$k = 100$	11778	767	13678	1715	93,89 %	87,29 %	91,12 %	92,49 %

Tabelle A.5: Objektunspezifische Evaluation:
Nearest Neighbour (NN) - Histogram of Local Surface Normals (HLSN)

AB - HLSN									
Klassifikator		TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
AB	$T = 20$	10374	1794	12651	3119	85,26 %	76,88 %	82,41 %	83,44 %

Tabelle A.6: Objektunspezifische Evaluation:
AdaBoost (AB) - Histogram of Local Surface Normals (HLSN)

RF - HLSN									
Klassifikator		TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
RF	$nT = 10$	11113	445	14000	2380	96,15 %	82,36 %	89,89 %	93,03 %
RF	$nT = 50$	10930	267	14178	2563	97,62 %	81,00 %	89,87 %	93,77 %
RF	$nT = 100$	10967	268	14177	2526	97,61 %	81,28 %	90,00 %	93,84 %
RF	$nT = 500$	10955	236	14209	2538	97,89 %	81,19 %	90,07 %	94,02 %

Tabelle A.7: Objektunspezifische Evaluation:
Random Forests (RF) - Histogram of Local Surface Normals (HLSN)

SVM - HLSN										
Klassifikator				TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
SVM	*linear*	$C1$		12100	2688	11757	1393	81,82 %	89,68 %	85,39 %	83,28 %
SVM	*linear*	$C10$		12079	2668	11777	1414	81,91 %	89,52 %	85,39 %	83,33 %
SVM	*linear*	$C100$		12082	2668	11777	1411	81,91 %	89,54 %	85,40 %	83,33 %
SVM	*linear*	$C1000$		12077	2674	11771	1416	81,87 %	89,51 %	85,36 %	83,29 %
SVM	*rbf*	$C1$	$1e^{-3}$	12058	2307	12138	1435	83,94 %	89,36 %	86,61 %	84,97 %
SVM	*rbf*	$C10$	$g1e^{-3}$	11633	1541	12904	1860	88,30 %	86,22 %	87,83 %	87,88 %
SVM	*rbf*	$C100$	$g1e^{-3}$	11121	1042	13403	2372	91,43 %	82,42 %	87,78 %	89,48 %
SVM	*rbf*	$C1000$	$g1e^{-3}$	10848	813	13632	2645	93,03 %	80,40 %	87,62 %	90,19 %
SVM	*rbf*	$C1$	$g1e^{-4}$	12454	3064	11381	1039	80,26 %	92,30 %	85,31 %	82,41 %
SVM	*rbf*	$C10$	$g1e^{-4}$	12122	2694	11751	1371	81,82 %	89,84 %	85,45 %	83,30 %
SVM	*rbf*	$C100$	$g1e^{-4}$	12011	2276	12169	1482	84,07 %	89,02 %	86,55 %	85,01 %
SVM	*rbf*	$C1000$	$g1e^{-4}$	11607	1547	12898	1886	88,24 %	86,02 %	87,71 %	87,79 %
SVM	*poly*	$C1$	$d2$	10451	1339	13106	3042	88,64 %	77,45 %	84,32 %	86,15 %
SVM	*poly*	$C10$	$d2$	10393	1256	13189	3100	89,22 %	77,03 %	84,41 %	86,48 %
SVM	*poly*	$C100$	$d2$	10400	1197	13248	3093	89,68 %	77,08 %	84,64 %	86,84 %
SVM	*poly*	$C1000$	$d2$	10378	1228	13217	3115	89,42 %	76,91 %	84,45 %	86,60 %
SVM	*poly*	$C1$	$d3$	11224	617	13828	2269	94,79 %	83,18 %	89,67 %	92,22 %
SVM	*poly*	$C10$	$d3$	11052	509	13936	2441	95,60 %	81,91 %	89,44 %	92,51 %
SVM	*poly*	$C100$	$d3$	11030	498	13947	2463	95,68 %	81,75 %	89,40 %	92,53 %
SVM	*poly*	$C1000$	$d3$	11034	530	13915	2459	95,42 %	81,78 %	89,30 %	92,34 %
SVM	*poly*	$C1$	$d4$	11160	873	13572	2333	92,74 %	82,71 %	88,52 %	90,55 %
SVM	*poly*	$C10$	$d4$	10986	470	13975	2507	95,90 %	81,42 %	89,34 %	92,60 %
SVM	*poly*	$C100$	$d4$	10868	440	14005	2625	96,11 %	80,55 %	89,03 %	92,53 %
SVM	*poly*	$C1000$	$d4$	10792	494	13951	2701	95,62 %	79,98 %	88,56 %	92,02 %

Tabelle A.8: Objektunspezifische Evaluation:
Support Vector Machine (SVM) - Histogram of Local Surface Normals (HLSN)

NN - FPFH									
Klassifikator		TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
NN	$k = 5$	10865	1268	10968	3721	89,55 %	74,49 %	81,40 %	86,07 %
NN	$k = 10$	10707	1168	11068	3979	90,16 %	72,91 %	80,88 %	86,09 %
NN	$k = 17$	11182	1472	10764	3504	88,37 %	76,14 %	81,52 %	85,62 %
NN	$k = 67$	11507	1819	10417	3179	86,35 %	78,35 %	81,44 %	84,62 %
NN	$k = 100$	11517	1964	10272	3069	85,43 %	78,96 %	81,24 %	84,05 %

Tabelle A.9: Objektunspezifische Evaluation:
Nearest Neighbour (NN) - Fast Point Feature Histogram (FPFH)

AB - FPFH									
Klassifikator		TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
AB	$T = 20$	10386	2756	9480	4200	79,03 %	71,21 %	74,07 %	77,33 %

Tabelle A.10: Objektunspezifische Evaluation:
AdaBoost (AB) - Fast Point Feature Histogram (FPFH)

RF - FPFH									
Klassifikator		TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
RF	$nT = 10$	10605	1626	10610	3981	86,71 %	72,71 %	79,10 %	83,49 %
RF	$nT = 50$	10241	1214	11022	4345	89,40 %	70,21 %	79,27 %	84,77 %
RF	$nT = 100$	10192	1147	11089	4394	89,88 %	69,88 %	79,34 %	85,02 %
RF	$nT = 500$	10153	1110	11126	4433	90,14 %	69,61 %	79,33 %	85,12 %

Tabelle A.11: Objektunspezifische Evaluation:
Random Forests (RF) - Fast Point Feature Histogram (FPFH)

SVM - FPFH										
Klassifikator				TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
SVM	*linear*	$C1$		10834	2130	10106	3752	83,57 %	74,28 %	78,07 %	81,53 %
SVM	*linear*	$C10$		10838	2126	10110	3748	83,60 %	74,30 %	78,10 %	81,56 %
SVM	*linear*	$C100$		10841	2123	10113	3745	83,62 %	74,32 %	78,12 %	81,58 %
SVM	*rbf*	$C1$	$g1e^{-3}$	10986	2092	10144	3600	84,00 %	75,32 %	78,78 %	82,11 %
SVM	*rbf*	$C10$	$g1e^{-3}$	10750	1805	10431	3836	85,62 %	73,70 %	78,97 %	82,94 %
SVM	*rbf*	$C100$	$g1e^{-3}$	10690	1515	10721	3896	87,59 %	73,29 %	79,83 %	84,30 %
SVM	*rbf*	$C1000$	$g1e^{-3}$	10575	1285	10951	4011	89,17 %	72,50 %	80,26 %	85,25 %
SVM	*rbf*	$C1$	$g1e^{-4}$	10817	2800	9436	3769	79,44 %	74,16 %	75,51 %	78,32 %
SVM	*rbf*	$C10$	$g1e^{-4}$	10935	2272	9964	3651	82,80 %	74,97 %	77,92 %	81,10 %
SVM	*rbf*	$C100$	$g1e^{-4}$	10919	1980	10256	3667	84,65 %	74,86 %	78,95 %	82,49 %
SVM	*rbf*	$C1000$	$g1e^{-4}$	10732	1744	10492	3854	86,02 %	73,58 %	79,13 %	83,21 %
SVM	*poly*	$C1$	$d2$	10502	1855	10381	4084	84,99 %	72,00 %	77,86 %	82,03 %
SVM	*poly*	$C10$	$d2$	10323	1547	10689	4263	86,97 %	70,77 %	78,34 %	83,16 %
SVM	*poly*	$C100$	$d2$	10144	1440	10796	4442	87,57 %	69,55 %	78,07 %	83,25 %
SVM	*poly*	$C1000$	$d2$	10093	1413	10823	4493	87,72 %	69,20 %	77,98 %	83,26 %
SVM	*poly*	$C1$	$d3$	11071	1761	10475	3515	86,28 %	75,90 %	80,33 %	83,98 %
SVM	*poly*	$C10$	$d3$	10748	1436	10800	3838	88,21 %	73,69 %	80,34 %	84,87 %
SVM	*poly*	$C100$	$d3$	10581	1206	11030	4005	89,77 %	72,54 %	80,57 %	85,70 %
SVM	*poly*	$C1000$	$d3$	10554	1154	11082	4032	90,14 %	72,36 %	80,67 %	85,92 %
SVM	*poly*	$C1$	$d4$	11431	3244	8992	3155	77,89 %	78,37 %	76,14 %	77,99 %
SVM	*poly*	$C10$	$d4$	10819	2039	10197	3767	84,14 %	74,17 %	78,35 %	81,94 %
SVM	*poly*	$C100$	$d4$	10603	1485	10751	3983	87,72 %	72,69 %	79,61 %	84,23 %
SVM	*poly*	$C1000$	$d4$	10462	1198	11038	4124	89,73 %	71,73 %	80,16 %	85,44 %

Tabelle A.12: Objektunspezifische Evaluation:
Support Vector Machine (SVM) - Fast Point Feature Histogram (FPFH)

NN - SURE									
Klassifikator		TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
NN	$k = 5$	3738	4431	9291	9168	45,76 %	28,96 %	48,93 %	41,00 %
NN	$k = 10$	3454	3536	10186	9452	49,41 %	26,76 %	51,22 %	42,26 %
NN	$k = 17$	3936	4261	9461	8970	48,02 %	30,50 %	50,31 %	43,07 %
NN	$k = 67$	3958	4208	9514	8948	48,47 %	30,67 %	50,59 %	43,43 %
NN	$k = 100$	3905	4170	9552	9001	48,36 %	30,26 %	50,54 %	43,19 %

Tabelle A.13: Objektunspezifische Evaluation:
Nearest Neighbour (NN) - SURE-Deskriptor

AB - SURE									
Klassifikator		TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
AB	$T = 20$	3771	5229	8493	9135	41,90 %	29,22 %	46,06 %	38,55 %

Tabelle A.14: Objektunspezifische Evaluation:
AdaBoost (AB) - SURE-Deskriptor

RF - SURE									
Klassifikator		TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
RF	$nT = 10$	3369	3672	10050	9537	47,85 %	26,10 %	50,39 %	41,02 %
RF	$nT = 50$	3425	3249	10473	9481	51,32 %	26,54 %	52,19 %	43,24 %
RF	$nT = 100$	3445	3079	10643	9461	52,81 %	26,69 %	52,91 %	44,16 %
RF	$nT = 500$	3307	2899	10823	9599	53,29 %	25,62 %	53,06 %	43,82 %

Tabelle A.15: Objektunspezifische Evaluation:
Random Forests (RF) - SURE-Deskriptor

SVM - SURE										
Klassifikator				TP	FP	TN	FN	PR	RC	AC	$F_{0.5}$
SVM	*linear*	$C1$		10004	4793	8929	2902	67,61 %	77,51 %	71,10 %	69,38 %
SVM	*linear*	$C10$		10004	4793	8929	2902	67,61 %	77,51 %	71,10 %	69,38 %
SVM	*linear*	$C100$		10043	4858	8864	2863	67,40 %	77,82 %	71,00 %	69,25 %
SVM	*rbf*	$C1$	$g1e^{-3}$	4051	5043	8679	8855	44,55 %	31,39 %	47,81 %	41,10 %
SVM	*rbf*	$C10$	$g1e^{-3}$	4099	4620	9102	8807	47,01 %	31,76 %	49,58 %	42,89 %
SVM	*rbf*	$C100$	$g1e^{-3}$	4133	4514	9208	8773	47,80 %	32,02 %	50,10 %	43,51 %
SVM	*rbf*	$C1000$	$g1e^{-3}$	4170	4533	9189	8736	47,91 %	32,31 %	50,17 %	43,69 %
SVM	*rbf*	$C1$	$g1e^{-4}$	9951	4687	9035	2955	67,98 %	77,10 %	71,30 %	69,63 %
SVM	*rbf*	$C10$	$g1e^{-4}$	10107	4850	8872	2799	67,57 %	78,31 %	71,27 %	69,48 %
SVM	*rbf*	$C100$	$g1e^{-4}$	4133	4514	9208	8773	47,80 %	32,02 %	50,10 %	43,51 %
SVM	*rbf*	$C1000$	$g1e^{-4}$	4108	4632	9090	8798	47,00 %	31,83 %	49,56 %	42,91 %
SVM	*poly*	$C1$	$d2$	3617	4336	9386	9289	45,48 %	28,03 %	48,83 %	40,44 %
SVM	*poly*	$C10$	$d2$	3691	4471	9251	9215	45,22 %	28,60 %	48,60 %	40,51 %
SVM	*poly*	$C100$	$d2$	3701	4490	9232	9205	45,18 %	28,68 %	48,57 %	40,52 %
SVM	*poly*	$C1000$	$d2$	3729	4496	9226	9177	45,34 %	28,89 %	48,65 %	40,70 %
SVM	*poly*	$C1$	$d3$	11167	6745	6977	1739	62,34 %	86,53 %	68,14 %	66,03 %
SVM	*poly*	$C10$	$d3$	4559	6367	7355	8347	41,73 %	35,32 %	44,74 %	40,27 %
SVM	*poly*	$C100$	$d3$	4466	6086	7636	8440	42,32 %	34,60 %	45,45 %	40,52 %
SVM	*poly*	$C1000$	$d3$	4487	6074	7648	8419	42,49 %	34,77 %	45,57 %	40,68 %
SVM	*poly*	$C1$	$d4$	3122	3222	10500	9784	49,21 %	24,19 %	51,16 %	40,78 %
SVM	*poly*	$C10$	$d4$	4017	4922	8800	8889	44,94 %	31,13 %	48,13 %	41,27 %
SVM	*poly*	$C100$	$d4$	4321	5918	7804	8585	42,20 %	33,48 %	45,53 %	40,11 %
SVM	*poly*	$C1000$	$d4$	4208	5531	8191	8698	43,21 %	32,60 %	46,56 %	40,57 %

Tabelle A.16: Objektunspezifische Evaluation:
Support Vector Machine (SVM) - SURE-Deskriptor

Anhang B

Weiterführende Anmerkungen zum Quellcode

B.1 Punktwolken: Datentypkonvertierung

Zum Versenden der 3D-Punkte in Form einer Punktwolke zwischen zwei MircoUnits muss der MIRA-konforme Datentyp `maps::GenericPointCloud` verwendet werden. Da innerhalb der MicroUnits auf Funktionen der PCL zurückgegriffen wird, wird die Punktwolke am Anfang jeder MicroUnit, die auf Basis der Punktwolke arbeitet, punktweise in den PCL-konformen Datentyp `pcl::PointCloud<PointXYZ>` gewandelt, um am Ende der MicroUnit den Datentyp vice versa zu ändern. Die Entwicklung eines Datentyps, der für das Versenden auf den Channels, als auch für die Anwendung der PCL-Funktionen geeignet ist, würde diese aufwendige Konvertierung unnötig machen und somit voraussichtlich zu einer deutlichen Reduzierung der Berechnungszeiten führen.

B.2 Datenextraktion

Zur Evaluation des neuen Ansatzes wurden die Featuredaten aus den Trainings- und Testvideos extrahiert (Abschnitt 4.2.8). Wie in Abschnitt 4.2.5 beschrieben, soll hierbei die Möglichkeit gewahrt werden, den neuen Ansatz auf die Verwendung eines layerspe-

zifischen Klassifikators zu erweitern. Daher extrahiert die `extractFeatureData`-Unit die Trainingsdaten nicht objektunabhängig, sondern bewahrt die Zuordnung der Featurevektoren jedes Layers zu seinem Objekt. Hierzu erstellt die Unit eine vom Nutzer angegebene Anzahl an TXT-Dateien ($training\_num\_files$) in denen die Daten der Featurevektoren Interest Point- bzw. layerweise gespeichert werden. Für jedes segmentierte Objekt wird eine neue Zeile in den TXT-Dateien angelegt. TXT-Dateien, in die aufgrund der Anzahl der Layer bzw. Interest Points des Objekts keine Zeile mit Featuredaten geschrieben wird, werden mit einer Zeile mit d Nullen aufgefüllt. Dadurch ist gewährleistet, dass die Featuredaten der i-ten Zeile aller TXT-Dateien zum selben Objekt gehören. Zudem wird eine TXT-Datei erstellt, die zeilenweise die Anzahl der Interest Points bzw. Layer pro Objekt enthält. Um zu garantieren, dass die Featurevektoren aller Interest Points bzw. Layer extrahiert werden, muss $training\_num\_files$ größer oder gleich der in den Trainingsdaten vorkommenden maximalen Anzahl an Interest Points bzw. Layer pro Objekt sein. In dieser Arbeit wurde für die layerbasierten Features $training\_num\_files = 20$ und für den auf Interest Points basierende SURE-Deskriptor $training\_num\_files = 30$ verwendet.

Da für das Training eines Interest Point- bzw. layerunspezifischen Klassifikators die Objektzugehörigkeit der Interest Points bzw. Layer nicht relevant ist, werden nach der Extraktion aller Trainingsdaten diese mit Hilfe eines Matlab-Scripts in drei TXT-Dateien zusammengefasst. Eine Datei enthält alle positiven Trainingsdaten, die Zweite alle negativen Trainingsdaten mit allgemeinen Objekte und die Dritte alle negativen Trainingsdaten mit Hunden. Durch die Trennung der Daten nach allgemeinen Objekte und Hunden ist eine Evaluation des Einflusses der Hundedaten auf die Gesamtdetektionsleistung möglich.

Die Aufteilung der Trainingsdaten erfolgt analog zur in Abschnitt 5.1 beschrieben Methodik. Auch diese wurde über ein Matlab-Script realisiert.

Literaturverzeichnis

[Alexander, 2010] Alexander, N.B. (2010). *Merck Manual of Geriatrics: Falls in the Elderly.* Online unter: `http://www.merckmanuals.com/professional/geriatrics/falls_in_the_elderly/falls_in_the_elderly.html`. 1

[Alpaydin, 2008] Alpaydin, E. (2008). *Maschinelles Lernen.* Oldenbourg Wissenschaftsverlag, München. 5, 8

[Anderson et al., 2006] Anderson, D., J. Keller, M. Skubic, X. Chen und Z. H. (2006). *Recognizing Falls from Silhouettes.* In: *Proc. of the 28th Annual Int. Conf. on Engineering in Medicine and Biology Society (EMBS)*, S. 6388–6391, New York City, USA. 24, 25

[Arras et al., 2007] Arras, K., M. Mozos und W. Burgard (2007). *Using Boosted Features for the Detecion of People in 2D Range Data.* In: *Proc. of the IEEE Int. Conf. on Robotics and Automation.* 8, 35, 69

[Bajracharya et al., 2009] Bajracharya, M., B. Moghaddam, A. Howard, S. Brennan und L. Matthies (2009). *Results from a Real-time Stereo-based Pedestrian Detection System on Moving Vehicle.* In: *Proc. of the IEEE ICRA 2009 Workshop on People Detection and Tracking*, Kobe, Japan. 35

[Bernin, 2011] Bernin, A. (2011). *Einsatz von 3D-Kameras zur Interpretation von räumlichen Gesten im Smart Home Kontext.* Masterarbeit, Hochschule für Angewandte Wissenschaften Hamburg. 11

[Bhattacharyya, 1946] Bhattacharyya, A. (1946). *On a measure of divergence between two statistical populations defined by probability distributions.* The Indian Journal of Statistics (1933-1960), 7(4):401–406. 31

[Blow, 1960] Blow, D.M. (1960). *To fit a plane to a set of points ba least squares.* Acta Crystallographica, 13(2):186. 64

[Bourdev und Malik, 2009] Bourdev, L. und J. Malik (2009). *Poselets: Body Part Detectors trained using 3D Human Pose Annotations.* In: *Int. Conf. on Computer Vision (ICCV'09).* 27

[Cai, 2010] Cai, Y. (2010). *Mobile Intelligence.* Journal of Universal Computer Science, 16(12):1650–1665. 24, 25

[Camastra und Vinciarelli, 2008] Camastra, F. und A. Vinciarelli (2008). *Machine Learning for Audio, Image and Video Analysis: Theory and Applications.* Springer-Verlag, London. 7

[Chang und Lin, 2011] Chang, C.C. und C. Lin (2011). *LIBSVM: A library for support vector machines.* ACM Transactions on Intelligent Systems and Technology, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 70

[Dalal und Triggs, 2005] Dalal, N. und B. Triggs (2005). *Histograms of oriented gradients for human detection.* In: *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR'05)*, S. 886–893. 28, 30, 40

[Deshpande et al., 2009] Deshpande, N., E. Metter, F. Lauretani, S. Bandinelli und L. Ferrucci (2009). *Interpretating Fear of Falling in the Elderly: What Do We Need to Consider?.* Journal of Geriatric Physical Therapy, 32(3):91–96. 1

[Diraco et al., 2010] Diraco, G., P. Leone und P. Siciliano (2010). *An active vision system for fall detection and posture recognition in elderly healthcare.* In: *Design, Automation Test in Europe Conference Exhibition (DATE)*, S. 1536–1541. 21

[Doukas und Maglogiannis, 2008] Doukas, C. und I. Maglogiannis (2008). *Advanced patient or elder fall detection based on movement and sound data.* In: *2nd Int. Conf. on Pervasive Computing Technologies for Healthcare*, S. 103–107, Tampere, Finnland. IEEE. 18

[Einhorn, 2012] Einhorn, E. (2012). *MIRA - Middleware for Robotic Applications.* In: *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'12).* 51

[Electronics, 2012] Electronics, Philips (2012). *AutoAlert Pendant*. Online unter: `http://www.lifelinesys.com/content/lifeline-products/personal-help-buttons/auto-alert-pendant`. 16, 17

[Enas und Choi, 1986] Enas, G.G. und S. Choi (1986). *Choice of the smoothing parameter and efficiency of k-nearest neighbor classification*. Computers & Mathematics with Applications, 12(2, Part A):235 – 244. 6, 80

[Felzenszwalb et al., 2004] Felzenszwalb, P., D. McAllester und D. Ramanan (2004). *Pictorial Structures for Object Recognition*. Int. Journal of Computer Vision, 61(1):55–79. 26

[Felzenszwalb et al., 2008] Felzenszwalb, P., D. McAllester und D. Ramanan (2008). *A discriminatively trained, multiscale, deformable part model*. In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'08)*, S. 1–8. 25, 26

[Ferrari et al., 2009] Ferrari, V., M. Marin-Jimenez und A. Zisserman (2009). *Pose search: Retrieving people using their pose*. In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'09)*, S. 1–8. 25, 26

[Fiolka, 2012a] Fiolka, T. (2012a). *Entropiebasierte interessante Regionen in 3D Punktwolken*. Masterarbeit, Rheinische Friedrich-Wilhelms-Universität Bonn. 62

[Fiolka, 2012b] Fiolka, T. (2012b). *sure-3d-features*. Online unter: `https://code.google.com/p/sure-3d-features/`. 61

[Fiolka et al., 2012] Fiolka, T., J. Stückler, D. Klein, D.A. und Schulz und S. Behnke (2012). *SURE: Surface Entropy for Distinctive 3D Features*. In: *Spatial Cognition*, Kloster Seeon. 43, 44, 61, 63, 113

[Future-Shape, 2010a] Future-Shape, GmbH (2010a). *SensFloor - Hightech für mehr Lebensqualität*. Online unter: `http://www.future-shape.de/download/6/Brosch%C3%BCre+SensFloor.pdf`. 17

[Future-Shape, 2010b] Future-Shape, GmbH (2010b). *Verbundprojekt SensFloor*. Online unter: `http://sensfloor.de/index.html`. 17

[Hegger et al., 2012] Hegger, F., N. Hochgeschwender, K. Kraetzschmar und P. Ploeger (2012). *People Detection in 3d Point Clouds using Local Surface Normals*.

In: *Proc. of the Robocup Symposium 2012*, Mexico City, Mexico. 32, 34, 35, 52, 59, 63, 64, 113

[Hermes, 2005] Hermes, T. (2005). *Digitale Bildverarbeitung.* Carl Hanser Verlag München Wien. 34

[Holz et al., 2011] Holz, D., S. Holzer, R. Rusu und S. Behnke (2011). *Real-Time Plane Segmentation using RGB-D Cameras.* In: *Proc. of the 15th Robo Cup International Symposium*, Istanbul, Türkei. 14

[IIS, 2011] IIS, Fraunhofer (2011). *MotionSENS - Bewegungssensor mit Sturzerkennung.* Online unter: `http://www.iis.fraunhofer.de/de/bf/med/mss/motionsens.html`. 16, 17

[Ikemura und Fujiyoshi, 2010] Ikemura, S. und H. Fujiyoshi (2010). *Real-Time Human Detection using Relational Depth Similarity Features.* In: *Proc. of the 10th Asian Conf. on Computer Vision (ACCV'10)*, S. 25–38. 30, 31, 113

[Ilmenau, 2012] Ilmenau, Technische Universität (2012). *Ziele und Einordnung der Forschergruppe SERROGA.* Online unter: `http://www.serroga.de/`. 2

[Jaspers, 2012] Jaspers, H. (2012). *Kombination von Bild- und Tiefeninformationen für Keypoint-basierte Objekterkennung.* Masterarbeit, Technische Universität Dortmund. 13

[Krishnamurthy, 2011] Krishnamurthy, S.N. (2011). *Human detection and extraction using kinect depth images.* Bournemouth University, England. 28

[Leibe et al., 2008] Leibe, B., A. Leonardis und B. Schiele (2008). *Robust Object Detection with Interleaved Categorization and Segmentation.* Int. Journal of Computer Vision, 77(1-3):259–289. 27, 90

[Litvak et al., 2008] Litvak, D., Y. Zigel und I. Gannont (2008). *Fall detection of elderly through floor vibrations and sound.* In: *30th Annual Int. Conf. of the IEEE engineering in medicine and biology society. (EMBS '08)*, S. 4632–4635, Vancouver, Kanada. 16, 18

[Lord et al., 2003] Lord, S.R., C. Sherrington und H. Menz (2003). *Falls in Older People: Risk Factors and Strategies for Prevention.* Injury Prevention, 9(1):93–94. 1

[LTD, 2013] LTD, PrimeSense (2013). *PrimeSense Firmenhomepage*. Online unter: `http://www.primesense.com/`. 12, 113

[Lv, 2011] Lv, Qingcong (2011). *A Poselet-based Approach for Fall Detection*. In: *Int. Symposium on IT in Medicine and Education (ITME'11)*, S. 209–212. 27

[Maji et al., 2011] Maji, S., L. Bourdev und J. Malik (2011). *Action Recognition from a Distributed Representation of Pose and Appearance*. In: *24th IEEE Conf. on computer Vision and Pattern Recognition (CVPR'11)*, S. 3177–3184. 27

[Maneewongvatana und Mount, 1999] Maneewongvatana, S. und D. Mount (1999). *It's okay to be skinny, if your friends are fat*. In: *4th Annual CGC Workshop on Computational Geometry*. 55

[Mastorakis und Makris, 2012] Mastorakis, G. und D. Makris (2012). *Fall detection system using Kintect's infrared sensor*. Real-Time Image Processing. 10, 24

[Microsoft, 2012] Microsoft (2012). *MSDN Library: Joint Orientation - Bones Hierarchy*. Online unter: `http://i.msdn.microsoft.com/dynimg/IC584386.png`. 23

[Navarro-Serment et al., 2010] Navarro-Serment, L.E., C. Mertz und M. Hebert (2010). *Pedestrian Detection and Tracking Using Three-dimensional LADAR Data*. The Int. Journal of Robotics Research, Special Issue on the 7th Int. Conf. on Field and Service Robots, 29(12):1516–1528. 35

[Noury et al., 2008] Noury, N., P. Rumeau, A. Bourke, G. ÓLaighin und J. Lundy (2008). *A proposal for the classification and evaluation of fall detectors*. IRBM, 29(6):340–349. 1

[Pedregosa et al., 2011] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot und E. Duchesnay (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12:2825–2830. 6, 7, 51, 67, 113

[Plagemann, 2010] Plagemann, C. (2010). *Real-time identification and localization of body parts from depth images*. In: *IEEE Int. Conf. on Robotics and Automation (ICRA)*, S. 3108–3113. 28

[Planinc und Kampel, 2012] Planinc, R. und M. Kampel (2012). *Introducing the use of depth data for fall detection.* Personal and Ubiquitous Computing. 23

[Popescu und Mahnot, 2009] Popescu, M. und A. Mahnot (2009). *Acoustic fall detection using one-class classifier*. In: *Annual Int. Conf of the IEEE Engineering in Medicine and Biology Society (EMBC '09)*, S. 3505–3508. 18

[Research, 2012] Research, Microsoft (2012). *Windows Kinect SDK*. Online unter: `http://www.microsoft.com/en-us/kinectforwindows/develop/overview.aspx`. 22

[Rougier et al., 2011] Rougier, C., E. Auvinet und J. Rousseau (2011). *Fall Detection from Depth Map Video Sequences.* In: *Towards Useful Services for Elderly an People with Disabilities: 9th Int. Conf. on Smart Homes and Health Telematics (ICOST'11)*, S. 121–128, Montreal, Kanada. 22

[Rougier et al., 2007] Rougier, C., J. Meunier, A. St-Arnaud und J. Rousseau (2007). *Fall Detection from Human Shape and Motion History Using Video Surveillance.* In: *21st Int. Conf. on Advanced Information Networking and Applications Workshop (AINAW '07)*, S. 875–880. 19, 21, 113

[Rubner et al., 2000] Rubner, Y., C. Tomasi und L. Guibas (2000). *The earth mover's distance as a metric for image retrieval.* Int. Journal of Computer Vision, 40:99–121. 45

[Rusu, 2009] Rusu, R.B. (2009). *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments.* Doktorarbeit, Computer Science department, Technische Universität München. 41, 42, 63, 66, 113

[Rusu und Cousins, 2011] Rusu, R.B. und S. Cousins (2011). *3D is here: Point Cloud Library (PCL).* In: *IEEE Int. Conf. on Robotics and Automation (ICRA'11).* 13, 14, 51

[Scharkow, 2012] Scharkow, M. (2012). *Automatische Inhaltsanalyse und maschinelles Lernen.* epubli GmbH, Berlin. 10

[Schwarz, 2011] Schwarz, L.A. (2011). *Estimating human 3D pose from Time-of-Flight images based on geodesic distances and optical flow.* In: *IEEE Int. Conf. and Workshop on Automatic Face & Gesture Recognition.* 28

[Shoaib et al., 2011] Shoaib, M., R. Dragon und J. Ostermann (2011). *Context-aware visual analysis of elderly activity in a cluttered home environment.* EURASIP Journal on Advances in Signal Processing. 20

[Spinello und Arras, 2011] Spinello, L. und K. Arras (2011). *People detection in RGB-D data.* In: *IEEE/RSI Int. Conf. on Intelligent Robots and Systems (IROS).* 29, 31

[Spinello et al., 2010] Spinello, L., K. Arras, R. Triebel und R. Siegwart (2010). *A Layered Approach to People Detection in 3D Range Data.* In: *Proc. of the 24th AAAI Conf. on Artificial Intelligence (AAAI'10)*, Atlanta, USA. 35, 36, 59, 80

[Spinello et al., 2011] Spinello, L., M. Luber und K. Arras (2011). *Tracking people in 3D using a bottom-up top-down detector*. In: *IEEE Int. Conf. on Robotics and Automation (ICRA)*, S. 1304–1310. 35, 36, 37, 63

[Spinello et al., 2008] Spinello, L., R. Triebel und R. Siegwart (2008). *Multimodal people detection and tracking in crowded scenes.* In: *AAAI Conf. in Artificial Intelligence (AAAI'08).* 35

[Stanford, 2013] Stanford (2013). *Kinect IR.* Online unter: `http://graphics.stanford.edu/~mdfisher/Images/KinectIR.png`. 12, 113

[Steder et al., 2010] Steder, B., R. Rusu, K. Konolige und W. Burgard (2010). *NARF: 3D range image features for object recognition.* In: *Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics at the IEEE/RSJ Int. Conf. on Intelligent Robots ans Systems (IROS).* 45

[Steinwart und Christmann, 2008] Steinwart, I. und A. Christmann (2008). *Support Vector Machines.* Springer, New York. 7

[Tang, 2011] Tang, S. (2011). *Visual recognition using hybrid cameras.* Masterarbeit, University of Missouri, USA. 40, 41, 113

[Taylor und Cowley, 2011] Taylor, C. und A. Cowley (2011). *Segmentation and Analysis of RGB-D data.* In: *RSS 2011 Workshop on RGB-D Cameras.* 90

[Thom, 2012] Thom, P.R. (2012). *Vorhersage kurzfristiger Aktienkursrenditen: Entwicklung eines maschinellen Lernverfahrens.* Josef Eul Verlag GmbH, Lohmar-Köln. 6

[Vaidehi et al., 2011] Vaidehi, A., K. Ganapathy, K. Mohan, A. Aldrin und K. Nirmal (2011). *Video based automatic fall detection in indoor environment.* In: *Proc. of the Int. Conf. on Recent Trends in Information Technology (ICRTIT)*, S. 1016–1020. 19, 20, 113

[Wang et al., 2011] Wang, S., S. Zabir und S. Leibe (2011). *Lying Pose Recognition for Elderly Fall Detection.* In: *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA. 3, 25, 26, 113

[Wild et al., 1981] Wild, D., U. Nayak und B. Isaacs (1981). *How dangerous are falls in old people at home?.* British Medical Journal, 282(6260):266–268. 1

[Willems et al., 2009] Willems, J., G. Debard, B. Bonroy, V. B. und T. Goedemé (2009). *How to detect human fall in video? An overview.* In: *Positioning and context-aware international conference (POCA)*, S. 6388–6391, New York City, USA. 18, 19

[Williams et al., 2007] Williams, A., D. Ganesan und A. Hanson (2007). *Aging in Place: Fall Detection and Localization in a Distributed Smart Camera Network.* In: *Proc. of the 15th Int. Conf. on Multimedia*, S. 892–901, New York City, USA. 19, 24

[Wu et al., 2011] Wu, S., S. Yu und W. Chen (2011). *An attempt to pedestrian detection in depth images.* In: *3rd Chines Conf. on Intelligent Visual Surveillance (IVS'11)*, S. 97–100. 30

[Xia, 2011] Xia, Lu (2011). *Human detection using depth information by Kinect.* In: *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW)*, S. 15–22. 28

[Zambanini et al., 2010] Zambanini, S., J. Machajdik und M. Kampel (2010). *Early versus Late Fusion in a Multiple Camera Network for Fall Detection.* In: *Proc. of the 34th Annual Workshop of the Austrian Association for Pattern Recognition (AAPR)*, S. 15–22, Zwettl, Österreich. 19

[Zhang et al., 2012] Zhang, C., Y. Tian und E. Capezuti (2012). *Privacy Preserving Automativ Fall Detection for Using RGBD Camers.* In: *Proc. of the 13th Int. Conf. on Computers Helping People with Special Needs (ICCHP'12)*, S. 625–633, Linz, Österreich. 22, 23

[Zhao et al., 2007] Zhao, J., J. Katupitiya und J. Ward (2007). *Global Correlation Based Ground Plane Estimation Using V-Diparity Image*. In: *IEEE Int. Conf. on Robotics and Automation (ICRA'07)*, S. 529–534. 22

Abbildungsverzeichnis

Abkürzungsverzeichnis

AB	AdaBoost
AC	Accuracy
BUTD	Bottom-Up Top-Down
EER	Equal Error Rate
FN	False Negative
FP	False Positive
FPFH	Fast Point Feature Histogram
GSF	Geometrische und statistische Features
HDD	Histogram of Depth Difference
HLSN	Histogram of Local Surface Normals
HONV	Histogram of Oriented Normal Vectors
HMM	Hidden Markov Model
HOD	Histogram of Oriented Depth
HOG	Histogram of Oriented Gradients
JDC	Jump Distance Clustering
LSN	Local Surface Normals
MHI	Motion History Image
MIRA	Middleware for Robotic Applications
NARF	Normal Aligned Radial Feature
NN	Nearest Neighbour
PFH	Point Feature Histogram
PR	Precision
RANSAC	Random Sample Consensus

RBF	Radiale Basisfunktion
RC	Recall
RDSF	Relational Depth Similarities Feature
RF	Random Forests
ROI	Region of Interest
SERROGA	Service-Robotik für die Gesundheitsassitenz
SURE	Surface Entropie
SVM	Support Vector Machine
TN	True Negative
TOF	Time-of-Flight
TP	True Positive

Thesen

- Zur Erhöhung der Sicherheit älterer, alleinlebender Personen bedarf es einer zuverlässigen Methodik zur Erkennung gestürzter Personen in der häuslichen Umgebung.
- Gegenwärtig kommerziell verfügbare Produkte zur Detektion von Stürzen bieten älteren Personen kein ausreichendes Maß an Sicherheit.
- Die 3D-Daten einer, an einer mobilen Roboterplattform befestigten Kinect, bieten neue Möglichkeiten zur robusten Erkennung gestürzter Personen.
- Vor allem auf der Orientierung der Oberflächennormalen basierende Features sind gut geeignet, um in der häuslichen Umgebung zwischen Objekten und Personen zu unterscheiden.
- Es wurde experimentell nachgewiesen, dass das Histogram of Local Surface Normals in Kombination mit einer Support Vector Machine sehr gut geeignet ist, gestürzte Personen in den 3D-Daten einer mobile Roboterplattform zu detektieren.

Ilmenau, 07.03.2013

.....

Friederike Schneemann