FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

Depth Data Analysis for Fall Detection

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Christopher Pramerdorfer
Matrikelnummer 0626747

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: PD. Dr. Martin Kampel

Wien, 29.08.2013

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Depth Data Analysis for Fall Detection

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Christopher Pramerdorfer
Registration Number 0626747

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: PD. Dr. Martin Kampel

Vienna, 29.08.2013

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Christopher Pramerdorfer
Nußdorfer Straße 78, 1090 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Acknowledgments

I would like to thank my supervisor, Martin Kampel, for his support. Thanks are also due to Rainer Planinc, who kindly provided his fall database for evaluation purposes, and the members of the FEARLESS[1] project for the opportunity to test the proposed fall detection system in practice.

This work is dedicated to my mother, Christine, for her love and support.

[1] http://fearless-project.eu (last accessed on 20.08.2013)

Abstract

Falls are a leading cause for accidental deaths among persons aged 65 or older. Fall victims require immediate assistance in order to minimize morbidity and mortality rates. However, statistically every other fall victim is unable to get back up without help. The aim of fall detection is to ensure fast assistance by automatically informing caretakers in case of falls. Different methods have been proposed for this purpose, but there are still unresolved issues that limit the applicability of fall detection systems in practice.

This thesis introduces a new fall detection system using current depth-sensing technology, with the aim of addressing these limitations. This sensor technology has several advantages over alternatives; it is unobtrusive, preserves the privacy of subjects, and works independently of lighting conditions, allowing for continuous monitoring. The generated data enables powerful scene analysis and, consequently, reliable fall detection. The applicability of this technology is verified by means of a quantitative analysis, in order to assess its limitations and data quality. In contrast to existing methods, the proposed system emphasizes person detection and tracking, which allows for robust temporal analysis and thus improves fall detection performance. To this end, this work also proposes a new motion detection algorithm optimized for depth data and fall detection. It is shown that this algorithm performs better than the state of the art on this type of data. Furthermore, this work contributes to the research field of tracking by introducing effective means for tracking multiple persons. The proposed system is easy to set up, plug and play, and can run on inexpensive low-end hardware, promoting broad acceptance.

This work shows that reliable fall detection in depth data is possible, as verified on a comprehensive fall database. At the time of writing, the system is tested under practical conditions in four countries, with promising first results.

Kurzfassung

Stürze sind ein Hauptgrund für unfallbedingte Todesfälle in der Gruppe der Personen über 64. Im Falle eines Sturzes ist sofortige Hilfe notwendig, um die Morbiditäts- und Sterblichkeitsrate zu minimieren. Statistiken zeigen jedoch, dass jedes zweite Sturzopfer nicht in der Lage ist, ohne Hilfe wieder aufzustehen. Das Ziel der Sturzerkennung ist es, schnelle Hilfe zu garantieren, indem diese im Falle eines Sturzes automatisch angefordert wird. Sturzerkennung ist ein aktives Forschungsfeld; Limitationen aktueller Methoden schränken die Anwendbarkeit in der Praxis jedoch noch ein.

In dieser Diplomarbeit wird ein neues System zur Sturzerkennung vorgestellt, das versucht, diese Einschränkungen zu lösen. Dazu wird auf aktuelle Sensortechnologie zurückgegriffen, die Tiefendaten liefert. Diese hat mehrere Vorteile gegenüber Alternativen, insbesondere ist sie unauffällig, bewahrt die Privatsphäre und funktioniert unabhängig von Lichtverhältnissen, wodurch eine kontinuierliche Überwachung ermöglicht wird. Tiefendaten sind aussagekräftig und erlauben dadurch eine zuverlässige Sturzerkennung. Dieser Sachverhalt wird mittels einer quantitativen Analyse verifiziert, im Zuge derer auch die Einschränkungen der Technologie sowie die Qualität der generierten Daten bewertet wird. Das vorgestellte Sturzerkennungssystem legt einen Fokus auf effektive Algorithmen zur Erkennung und Verfolgung von Personen, da die dadurch ermöglichte temporäre Analyse die Leistungsfähigkeit der Sturzerkennung steigert. Zu diesem Zweck wurde ein neuer Algorithmus zur Bewegungserkennung in Tiefendaten entwickelt, der insbesondere hinsichtlich Sturzerkennung leistungsfähiger als aktuelle Algorithmen ist. Weiters stellt diese Diplomarbeit einen effizienten Ansatz zur Erkennung und Verfolgung mehrerer Personen in Tiefendaten vor. Das entwickelte Sturzerkennungssystem ist einfach zu installieren, muss nicht konfiguriert werden, und ist auf kostengünstiger Hardware lauffähig, was die Akzeptanz in der Praxis fördert.

Diese Diplomarbeit zeigt, dass Tiefendaten eine zuverlässige Sturzerkennung ermöglichen. Dies wird mittels einer umfangreichen Sturzdatenbank überprüft. Zum Zeitpunkt des Verfassens dieses Berichtes wird das vorgestellte System in vier Ländern in der Praxis evaluiert, vorläufige Resultate sind vielversprechend.

Contents

CHAPTER 1

Introduction

Falls are a major public health problem. Between 30% and 60% of US citizens of age 65 or older suffer from falls each year, and 10% to 20% of these falls result in serious injury, hospitalization, or death [78]. Falls are the leading cause for accidental death in this age group and among the leading causes of death in general [70]. Moreover, they are major contributors to immobility and premature nursing home placement and thus threaten the independence of elderly persons [91, 78]. Furthermore, falls can have significant psychological consequences. In fact, the fear of falling is a prevalent concern among fall victims and often leads to activity restriction, social withdrawal, and loss of independence [5, 26, 78]. These conditions decrease the quality of life and can have a negative effect on health, which may further increase the risk of falls [26, 78].

1.1 Motivation

Statistically, every other fall victim is unable to get back up without help [70]. However, immediate help and treatment of injuries is vital for minimizing morbidity and mortality rates [67, 64]. Fall victims that have to remain on the floor for prolonged periods of time face the risk of potentially life-threatening diseases such as dehydration, hypothermia, and pneumonia [70]. In a study conducted by Wild et al. [95] half of the fall victims that remained on the floor for one hour or longer died within six months thereafter. Falls are, therefore, particularly dangerous for older persons that live alone and thus cannot rely on quick help in case of a fall, especially if too injured to call. In consequence, means are required for ensuring help under these circumstances. Permanent personal assistance is not feasible in terms of required manpower and health care costs. Panic buttons are a proven method of enabling persons to call for help easily [62]. However, their applicability with respect to falls is limited because they require user intervention, which may not be possible due to unconsciousness, for example [69].

To this end, there has been active research in the field of fall detection, with the aim to develop technology that is able to detect fall incidents and inform caretakers

(e.g. family members or ambulance personnel) [64]. Such technology can help decrease morbidity and mortality rates among fall victims that live alone, raise their confidence in living independently, and consequently allow them to stay at home, which is what elderlies generally prefer [25]. Furthermore, fall detection has the potential to help cope with increasing health care costs [25, 64, 49]. This is of particular interest considering the ongoing demographic change; the percentage of older persons continues to increase and is expected to reach almost 30% in Western Europe in 2050, as illustrated in Figure 1.1.

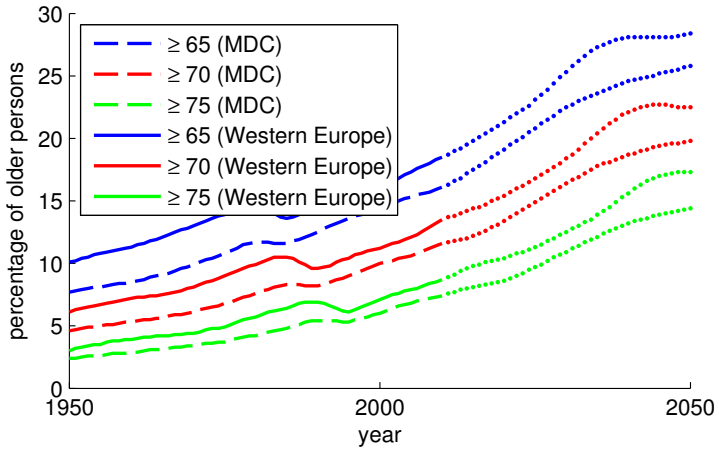


Figure 1.1: Estimated percentages of persons of ages $\{65, 70, 75\}$ or older, according to [93]. MDC includes Europe, Northern America, Australia, New Zealand, and Japan.

Consequently, research in this area (and the broader field of smart homes and ambient assisted living [25]) has been supported by governments and international organizations. At the time of writing, there are several research projects funded by the European Union.[1] For example, the FARSEEING[2] project aims to create a comprehensive fall repository, which allows for proper evaluation of fall detection systems. The goal of the FATE[3] project is to develop a reliable fall detection system using worn sensors, whereas the FEARLESS[4] project aims to accomplish this goal with vision-based sensors.

Fall detection systems must be reliable, that is they must exhibit a high sensitivity (percentage of falls correctly detected as such) and specificity (percentage of detected falls that actually happened). Moreover, they must be unobtrusive and respect the privacy of subjects in order to achieve broad acceptance [57]. To this end, such systems also have to be practicable, namely they must be inexpensive to acquire and maintain as well as easy to install and operate. However, the following literature review highlights that many existing fall detection methods and systems fall short in this regard.

[1] http://ec.europa.eu/digital-agenda/en/ict-and-fall-prevention-elderly (last accessed on 20.08.2013)
[2] http://farseeingresearch.eu (last accessed on 20.08.2013)
[3] http://project-fate.eu (last accessed on 20.08.2013)
[4] http://fearless-project.eu (last accessed on 20.08.2013)

1.2 Related Literature

Preliminary studies on fall detection were based on wearable accelerometers, with the aim of detecting falls by means of body acceleration [66]. Since then different approaches to fall detection have been proposed, which are here categorized in active and passive methods. The former are based on input from sensors that are worn or attached to the subject by other means, whereas the latter utilize sensors that are located in the environment of the subject. This review focuses on fall detection using depth data as current surveys on other approaches are available [64, 66, 100].

1.2.1 Active Methods

Methods based on wearable sensors are called active because they require the subject to actively participate by wearing sensors. Research in this area has been active for more than twenty years [57], with most methods relying on data from accelerometers or orientation sensors [66]. Bourke [14] and Bagalà [9] recently compared several approaches, the best achieved a sensitivity of 83% and a specificity of 97% under real-world conditions.

Active methods are cost-effective and easy to use [64]. Their main limitation is that they are intrusive because they require subjects to wear sensors, which is often opposed [100]. Even if subjects are willing to cooperate, the danger of forgetting to wear the sensors remains. Measures have been proposed to reduce these limitations, for example by embedding sensors into garments [65] or by utilizing mobile phone sensors [23]. However, the limitations cannot be precluded, rendering active methods unfavorable [64].

1.2.2 Passive Methods

Passive fall detection employs sensors that are located in the environment of the subject, hence user intervention is not required. Several methods of this kind are based on vibration sensors placed on the floor [1], microphones [55], or both [56]. A limiting factor of the former approach is that the performance depends on the floor type [55]. Recent advances in audio-based fall detection are promising, but performance is still affected by noise and other sound sources [54]. Another approach is to use pressure mats which are comparatively inexpensive but lack in terms of specificity [64, 100].

Other passive methods aim to detect falls by means of cameras and computer vision. A popular approach is to use background subtraction for person detection and to detect falls via features derived from person regions, such as size, orientation, or shape [75, 77]. Supervised learning is frequently used for classification purposes [3, 77]. Some methods assume that persons remain unmoving for several seconds after a fall [75, 77], which is arguably too restrictive in practice. Another approach is to detect lying persons as opposed to fall incidents [94]. More information on image-based fall detection is available in the survey of Mubashir et al. [64].

Camera images convey limited information in terms of scene geometry. In fact, Rougier et al. [76] argue that person detection and tracking in images is insufficient for reliable fall detection. Several methods have been proposed that aim to overcome this

limitation by operating in world coordinates, which allow for more distinctive and robust (view-independent) features. For example, the authors of [76] track the subject's head in 3D space by means of pose estimation and detect falls using velocity information. Another approach is to estimate world coordinates of objects via feature matching from multiple cameras (e.g. stereo matching [68, 40] or silhouette matching [4, 22]). Nevertheless, disadvantages inherent to the sensor technology limit the applicability of camera-based fall detection. These disadvantages mainly stem from the fact that images encode lighting conditions as opposed to scene geometry, hence scene changes are not necessarily reflected in the data and vice versa. Furthermore, cameras do not work in darkness, hence unobtrusive fall detection during nighttime is not possible. Feature matching techniques are comparatively limited in terms of resolution and accuracy. Moreover, their setup is complex because several cameras must be calibrated.

Time of flight sensors, which measure distances instead of lighting conditions (or both) do not have these limitations. This sensor type has the advantage of being active and thus works independently of lighting conditions, even during nighttime. Distance estimation is part of the sensor firmware, reducing computational requirements. Time of flight sensors are thus advantageous for fall detection and have been successfully used for this purpose [43, 27]. However, they are still too expensive for widespread use.[5]

1.2.3 Kinect-Based Methods

In late 2010 a depth sensor called Kinect was released by Microsoft. This sensor has the aforementioned advantages of time of flight sensors and estimates distances with a higher resolution and accuracy [82]. Moreover, it is inexpensive, costing less than 200 euros. Its depth-sensing technology was developed by PrimeSense [31] and is also used in other products such as the Asus Xtion Pro Live, which is shown in Figure 1.2.

Figure 1.2: The Asus Xtion Pro Live sensor.

Kinect sensors project an infrared pattern into the scene, which is recorded using a conventional camera. In every recorded image, this pattern is detected and matched with the projection. The obtained correspondences are used to compute disparities, similar to stereo matching. Details regarding the geometrical model of the sensor and how this estimation works can be found in [31, 82, 47]. Kinects produce disparity maps with resolutions of up to 640 by 480 pixels at up to 30 frames per second. The mapping from disparity maps to depth maps (i.e. from disparities to distances) is explained in [82, 47].

[5] The sensors used in [43, 27] cost more than 2000 euros, although cheaper sensors exist.

An alternative is to use middleware for this purpose, such as OpenNI[6] or KinectSDK[7]. Pixels for which disparity estimation failed are flagged accordingly by the sensor. Such pixels are called zero-pixels in this text.

Several methods for detecting falls using the Kinect sensor have been proposed, Table 1.1 contains a selection. All of those utilize background subtraction for person detection and, with the exception of [28, 60], height-based features for fall detection. Such features are naturally powerful for discriminating between upright and fallen persons because the perceived height of persons is significantly lower if they lie on the floor after a fall. All discussed methods except [69] also incorporate velocities, arguing that the vertical velocity of persons increases considerably during a fall due to gravity and that slow falls are not serious [8]. On the other hand, the authors of [69] reason that even slow falls can be dangerous and thus must be detected, which excludes velocity-based features. Furthermore, activities of daily living (e.g. sitting down) may be similar to falls in terms of observed velocities, causing false alarms and impacting specificity [74, 46].

Table 1.1: Kinect-based fall detection methods discussed in this section.

method	features used for fall detection
Auvinet & Meunier [8]	height and velocity of head and person centroids
Dubey et al. [28]	HU features of motion history image
Dubois & Charpillet [29]	height and velocity of centroid, point deviation
Kepski & Kwolek [46]	centroid, velocity and acceleration from worn device
Mastorakis & Makris [60]	change of bounding box
Planinc & Kampel [69]	height and orientation of the spine
Rougier et al. [74]	height and velocity of centroid

Apart from [28, 46], all discussed methods are conceptually similar and relatively simple; in all these cases persons are detected via background subtraction and the world coordinates of some representation of the person are obtained and analyzed in order to detect falls. Dubey et al. [28] follow a different approach, extracting HU features from motion history images and using a support vector machine for classification. In [46] information from both a Kinect and a worn device is analyzed.

All covered methods reportedly perform very well on simulated falls (higher than 96% in terms of sensitivity and specificity). Unfortunately, these methods were tested on custom test sequences, hence direct comparisons are not possible. This issue is attributed to the lack of publicly available Kinect fall datasets. To the knowledge of the author, the sole extensive dataset of this sort was presented only recently by Planinc and Kampel [69]. Nevertheless, the reported results suggest that Kinect-based methods have the potential for reliable fall detection in practice, while being relatively simple in concept and thus moderate in terms of computational requirements. This is due to the fact that Kinect's depth data is powerful for this kind of scene analysis.

[6] http://openni.org (last accessed on 20.08.2013)

[7] http://microsoft.com/en-us/kinectforwindows (last accessed on 20.08.2013)

However, the discussed methods have limitations that are expected to restrict their performance in practice. Velocity-based features are too restrictive if slow falls are considered important. Only Rougier et al. [74] account for falls that are not visible due to occlusions, which is considered mandatory in practice. Impending sunlight can have the same effect, a fact not taken into account. Furthermore, no method explicitly detects whether persons lie down on beds or other furniture (hereafter referred to as resting); discrimination is based solely on the features used for fall detection. This approach is error-prone because observed heights of resting persons may be similar to those of fallen persons, considering the height of some beds and couches. Particularly, height-based discrimination is not reliable if falls that end in a sitting position should be detected. While this is considered a requirement, some existing methods are tuned for persons that lie flat on the floor. For example, the method proposed by Rougier et al. [74] requires that the centroid height of fallen persons must be lower than 35.8cm, which is expectedly too low in this regard. Velocity-based features are more powerful for differentiating between fallen and resting persons, but imply the aforementioned restrictions.

Most of the covered methods [8, 28, 29, 46, 74] neglect person tracking, that is the association of detections between subsequent frames. It is thus expected that these methods fail in presence of multiple persons. Consequently, person tracking is considered mandatory for reliable fall detection. Only the fall detection methods introduced in [69] and [60] include tracking, which is accomplished by means of the NITE[8] middleware.

The approach by Dubey et al. [28] is considered limited because neither motion history images nor HU features are invariant in terms of sensor position, hence the performance is expected to decrease if the sensor is positioned differently than during training. However, this cannot always be avoided in practice because of environmental restrictions. This also applies to the work by Kepski and Kwolek [46], who use the height of the person centroid relative to the sensor height as a feature for fall detection.

Some of the discussed methods lack in terms of practicability. Those presented in [8] and [28] appear too complex to run in real-time on low-end hardware (unfortunately, information on computational requirements is missing in both cases). That introduced in [46] runs on a PandaBoard ES, a low-cost ARM platform, but requires a worn device, which is obtrusive. As mentioned before, [28] and [46] use features that restrict the sensor placement, which complicates the installation and hinders a proper setup.

In conclusion, fall detection using Kinect sensors is a promising approach. This is due to the fact that these sensors are inexpensive, unobtrusive, and provide data that allow for robust person detection and distinctive features. Moreover, the data are easily interpretable by humans, enabling verification of fall events by caretakers, for instance. Details of monitored persons are not exposed, hence their privacy is preserved [74]. Existing methods of this kind perform well on simulated falls, but often neglect circumstances that are considered important in practice. Furthermore, some methods seem too complex to run on low-end hardware or are inflexible in terms of sensor setup, which limits their practicability and may hinder broad application.

[8] http://openni.org/files/nite (last accessed on 20.08.2013)

1.3 Aim and Contributions

The aim of this thesis is to address limitations of existing Kinect-based fall detection methods, with the purpose of developing a fall detection system that is both reliable and practical. To this end, the system must detect falls reliably under practical conditions, which include slow and occluded falls as well as changing scene conditions. For this purpose, the limitations and data characteristics of the sensor must be known and evaluated in terms of implications for fall detection. To the knowledge of the author, such an analysis is missing in the literature. This thesis aims to fill this gap.

The algorithms involved must have low computational requirements in order to be applicable on low-cost platforms. With this in mind, they were designed to run on a PandaBoard ES[9], an inexpensive single-board computer. This platform features a dual-core ARM Cortex-A9 CPU with 1.2 GHz and 1 GiB of RAM. It has a low power-consumption and is thus inexpensive to operate, and is small enough to be unobtrusive (Figure 1.3). Another focus is easy setup and maintenance. To this end, flexibility in terms of sensor placement is required and the system must be plug and play.

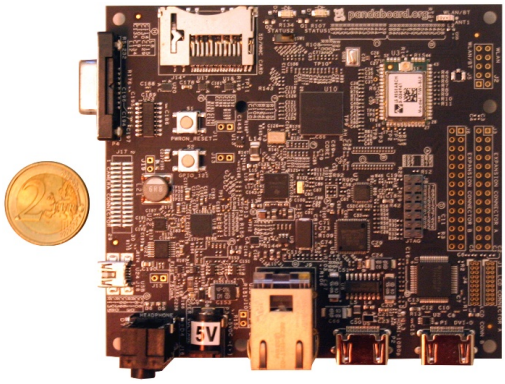


Figure 1.3: The PandaBoard ES, in comparison to a 2 euro coin.

The main contribution of this work is the introduction of a fall detection system that is reliable in practice, inexpensive, and easy to install and operate, in order to advance the research field of fall detection and ambient assisted living. Moreover, this thesis analyzes aspects regarding the data quality of the Kinect sensor that are not covered in the existing literature, information considered valuable in the design of Kinect-based solutions. Furthermore, several algorithms constituting the system are applicable for other tasks related to computer vision. In particular, this work proposes a new background subtraction algorithm designed for the Kinect sensor that allows for robust detection of persons and moving objects in general. The proposed fall detection system operates in a so-called plan-view space, which has been successfully used for person detection and tracking in the past [13, 33]. This work contributes to this research field by introducing

[9] http://pandaboard.org (last accessed on 20.08.2013)

an efficient approach to tracking persons in plan-view space that can cope with multiple persons, even if they move close together.

1.4 Structure

This thesis is structured as follows. Chapter 2 covers a quantitative analysis of the Kinect sensor, with a focus on implications for person detection and fall detection. On a high level of abstraction, the proposed fall detection system consists of three main parts, namely motion detection, person detection and tracking, and fall detection (Figure 1.4).

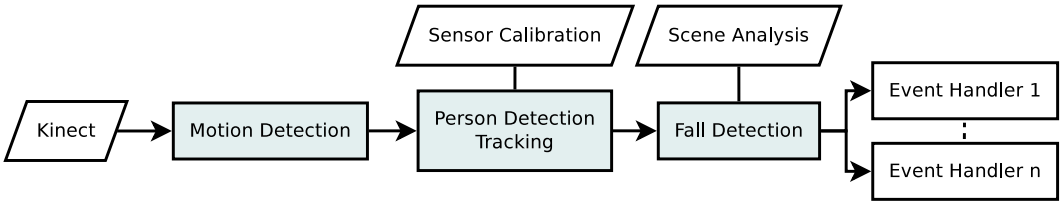


Figure 1.4: High-level overview of the structure of the proposed system.

During motion detection, moving objects are detected. For this purpose, a new background subtraction algorithm optimized for Kinect depth maps is presented in Chapter 3. Depth maps and detected motion constitute the input to the person detection and tracking stage, which is discussed in Chapter 4. During this stage, persons are distinguished from other objects and tracked over time. A central concept are so-called plan-view maps, which allow for efficient and robust person detection and scene analysis. For computing these coordinates, the height and orientation of the sensor with respect to the floor must be known. The system is able to estimate these properties automatically, which simplifies the setup. Chapter 5 covers fall detection in plan-view coordinates as well as scene analysis, namely automatic detection of sitting accommodations and regions in which reliable fall detection is possible. Chapter 6 concludes this work.

CHAPTER 2

Sensor Analysis

In order to be able to evaluate the Kinect sensor for fall detection, its limitations and data quality must be known. Such knowledge is also considered prerequisite for designing reliable fall detection algorithms. However, previous work in the field of Kinect-based fall detection does not address this topic in detail. Available literature on Kinect sensor analysis does not focus on fall detection and is thus insufficient in this regard.

The aim of this chapter is to provide a detailed analysis of the data quality of Kinect sensors and to discuss the results with respect to fall detection. A focus is on highlighting limitations of the technology and possible implications for fall detection. For this purpose, the sensor was evaluated in terms of the quality criteria (i) resolution, (ii) precision and random errors, (iii) reproducibility, and (iv) accuracy [89, 44]. All experiments were carried out on metrical data, as provided by OpenNI 2.1. This analysis is based on a paper by the author of this thesis [71].

2.1 Resolution

The resolution of a measuring device describes the smallest details it is able to resolve [44]. With regard to depth sensors it is desirable to distinguish between depth resolution and spatial resolution. In this text, the term depth resolution denotes the smallest difference in distance the sensor can distinguish, while spatial resolution refers to the minimum size of reliably detectable objects.

2.1.1 Depth Resolution

The depth resolution of the sensor restricts detectable scene changes. If the depth resolution is too low, fallen persons or parts thereof may not be distinguishable from the floor. This can cause fall detection algorithms that rely on a continuous detection of fallen persons to fail. Stone and Skubic [87] mentioned this issue, but to the knowledge of the author further analyses in this regard are missing in the literature.

Kinect sensors can distinguish between two object distances only if their difference is large enough. This is because the sensor can derive disparities with only limited accuracy. In fact, the sensor distinguishes between 1024 distinct disparities and thus distances [47], while the measuring range spans approximately 10m.

In [47] Khoshelham and Elberink modeled the depth resolution of the Kinect as a function of object distance. Their approach was to derive a mapping from disparity to distance, by comparing corresponding values. They found this mapping to be quadratic with respect to object distance. Consequently, the depth resolution of the sensor decreases quadratically with object distance. Figure 2.1 shows the graph as proposed by [47]. Other authors presented almost identical models [82, 2]. They employed a more direct approach, estimating the depth resolution directly from metrical data by calculating differences between adjacent measurements.

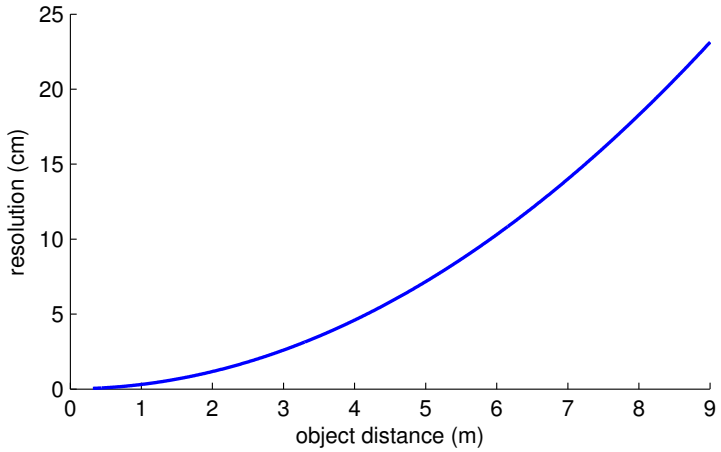


Figure 2.1: Depth resolution as a function of object distance.

The mapping from disparity to distance is carried out in software; therefore, the depth resolution of the sensor was expected to vary slightly with software package and version. In order to verify that the model shown in Figure 2.1 applies to the software used for fall detection (OpenNI 2.1), an experiment similar to those in [82, 2] was conducted. Distance measurements were recorded while the sensor was translated slowly away from a wall, up to a distance of 11m. This was done three times to ensure that the obtained dataset was complete, in the sense that it contained all values the sensor was able to produce. All unique measurement values were sorted, constituting a tuple $(d_1, d_2, \ldots, d_n)$, and the depth resolution at object distance d_i was estimated as $d_{i+1} - d_i$. The resulting mapping confirms with those of [47, 82, 2].

As shown in Figure 2.1, the depth resolution of the Kinect remains below 10cm for object distances closer than approximately 6m, but increases significantly thereafter. In consequence limbs of fallen persons may be indiscernible from the floor at larger distances, as illustrated in Figure 2.2. It is therefore suggested not to depend on a reliable detection of limbs for fall detection.

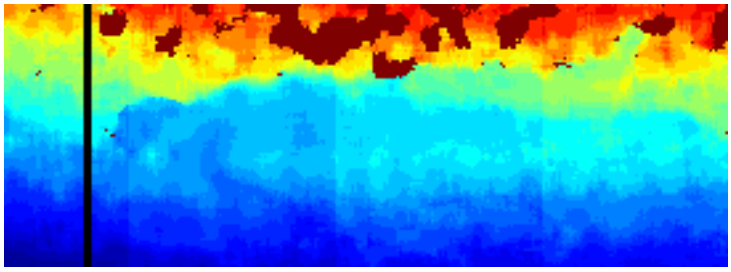


Figure 2.2: Illustration of the effect of limited depth resolution on fall detection. The image depicts a person lying flat on the floor at a distance of approximately 6m from the sensor. Distances are color-coded, blue colors represent closer distances. Due to the limited depth resolution, limbs are only partially distinguishable from the floor. This is illustrated by the fact that colors in regions of arms or legs do not always differ from those of the floor beneath them (left side of the vertical line).

2.1.2 Spatial Resolution

The minimum size objects must have in order to be reliably (i.e. continuously) detectable from a certain distance depends on the spatial resolution of the sensor. According to a datasheet[1] by the developer of the depth sensing technology of the Kinect, the spatial resolution is 3.4mm at an object distance of 2m. There exists only one publication that examines this criteria in more detail [12]. The authors estimated the spatial resolution by translating a planar test object with holes of different diameters away from the sensor until these holes were no longer continuously perceptible. Unfortunately, the experiment was limited to near-range (up to 2.5m).

One property that restricts the spatial resolution of an imaging device is the distance between adjacent sensor pixels, denoted as sampling pitch [88]. The smaller the sampling pitch, the higher the possible resolution. A model for the limitations imposed by this property is the Ground Sample Distance (GSD). The GSD corresponds to the minimum distance required between two objects in order for them to be separable from a certain distance [53]. It is defined as GSD $= dp/f$, where p is the sampling pitch, f is the focal length, and d is the object distance. The authors of [47] estimated $p = 9.3\mu$m and $f = 5.453$mm. At an object distance of 2m this amounts to a GSD of about 3.41mm, almost identical to the spatial resolution according the datasheet. Figure 2.3 (dotted line) shows the GSD as a function of object distance.

The GSD disregards characteristics such as quality of optics. Furthermore, it does not model complexities involved in distance estimation and hence represents the upper limit for the spatial resolution of the Kinect. It was thus expected that the GSD differs from the minimum size of reliably detectable objects. In order to verify this hypothesis, two square patches of cardboard with side lengths of 1.875cm and 3.75cm, respectively, were positioned in the scene, oriented towards the sensor. Two test objects were assumed to be

[1] http://primesense.com/wp-content/uploads/2013/02/PrimeSense_3DsensorsWeb.pdf (last accessed on 20.08.2013)

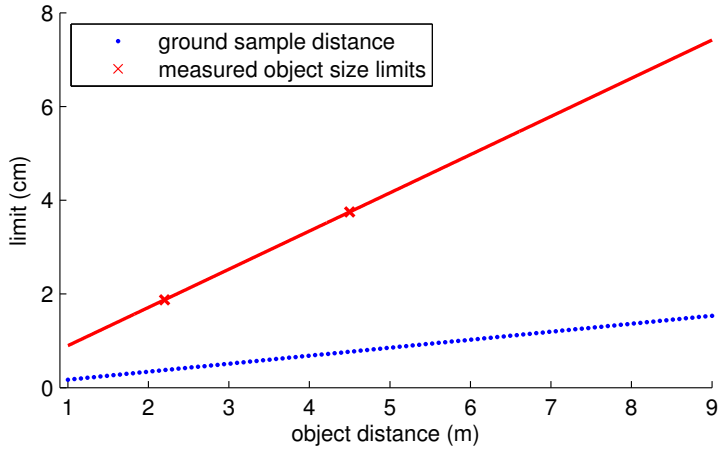


Figure 2.3: The dotted line illustrates the GSD of the Kinect. The crosses indicate the maximum distances at which the test objects were still reliably perceptible. The continuous line shows the estimated graph.

sufficient given the linearity of the GSD. The objects were located 30cm away from the background in order to preclude effects caused by the limited depth resolution. Maximum object distances were estimated by translating the sensor away from the objects until they were no longer reliably detectable by means of visual examination.

The results indicate a linear relation between object size and maximum distance. The maximum distances at which the test objects were still reliably detectable were much lower than predicted by the GSD, as shown in Figure 2.3. The size limits differ considerably from [12], according to which the minimum object diameter is 4cm at a distance of 1.5m. Discrepancies were expected, considering the different test methods.

The results presented in Figure 2.3 should be interpreted as clues for the sensor capabilities under optimal scene conditions. Further testing showed that object shape, surface properties, and lighting conditions can have a negative impact on performance. Particularly, depending on scene conditions arms and legs of standing persons are not reliably detectable at object distances beyond 6m. This circumstance underlines that fall detection methods should not depend on a reliable detection of limbs.

2.2 Precision

The precision describes the variability between multiple measurements of the same object under stable conditions [89]. With regard to the Kinect, the precision indicates the closeness of repeated distance measurements. This criteria is particularly important with respect to background subtraction algorithms, as their performance depends on data stability. Knowledge of the sensor precision allows for estimating the performance of such algorithms and aids in proper configuration.

2.2.1 Precision

Different methods have been proposed for estimating the precision of the Kinect sensor. The authors of [47] did so by analyzing single frames. They examined distance measurements of a planar surface that was oriented in parallel to the image plane of the sensor. Sensor measurements represent distances between points and the image plane (i.e. z-coordinates of observed points in the sensor coordinate system) [47, 2]. Therefore, deviations between these measurements were considered as measurement errors. More precisely, they fitted planes to the measurements and estimated the precision as the Root-Mean-Square Error (RMSE) between the measurements and the fitted planes. This approach is illustrated in Figure 2.4. The authors found the sensor precision at a particular object distance to be about half the depth resolution at that distance (Figure 2.5). Results presented in [20] support this hypothesis.

Andersen et al. [2] instead defined the precision as the range between consecutive measurements. Temporal analysis is considered important as fall detection is a continuous task. Unfortunately they examined the precision at only a single object distance.

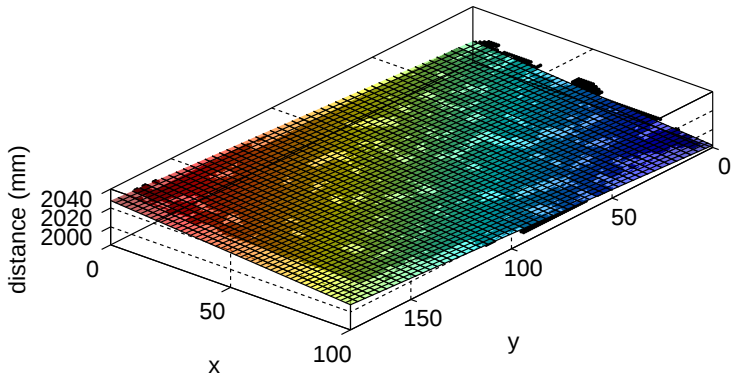


Figure 2.4: Distance measurements of a planar object from a distance of 2m. The plane was obtained via robust plane fitting, similar to [47] and [20]. The RMSE corresponds to the standard deviation between the fitted plane and the measurements [47].

In an effort to derive a model for the sensor precision that encompasses temporal variations, the following experiment was conducted. A cardboard plane with a width and height of 55cm by 99cm was used as the test object. The object was chosen because it was planar and had little absorption and specularity, and therefore represented an eminently suitable measurement target. The object was aligned parallel to the image plane of the sensor and 100 frames were recorded. This was done from distances between two and nine meters, in 1m steps. No significant infrared sources were present.

For every test distance, measurements of border regions were discarded as such regions contain measurement errors [2]. Zero-pixels were removed and the remaining values were consolidated to one set of values per distance d, denoted as $\mathbf{x}_d$. The cardinalities of $\mathbf{x}_d$ are summarized in Table 2.1. The sensor precision was estimated as the standard deviation

of $\mathbf{x}_d$, henceforth denoted as s_d. In order to be able estimate the precision as a function of object distance, a second order polynomial was fitted to the obtained (d, s_d) pairs.

Table 2.1: Cardinalities of $\mathbf{x}_d$ in thousand, for every test distance d.

d	2m	3m	4m	5m	6m	7m	8m	9m
$\lvert\mathbf{x}_d\rvert$	3572	1496	818	466	328	205	183	125

The resulting function is similar but greater than half the depth resolution, as visible in Figure 2.5. There are two primary causes for the differences, apart from temporal variations. First, multiple measurements of planar objects may only be treated as equal if the object is aligned perfectly in parallel to the sensor plane. Small alignment errors could not be avoided while obtaining $\mathbf{x}_d$. On the other hand, the plane-fitting approach employed by [47, 20] is able to correct for such errors. Second, it was found that plane-fitting RMSEs vary between frames. For example, the range over the RMSEs of the 100 frames used to generate $\mathbf{x}_5$ was almost 6mm. Under these circumstances, both models are comparable. It is thus considered applicable to approximate the spatio-temporal precision of the Kinect as half its depth resolution.

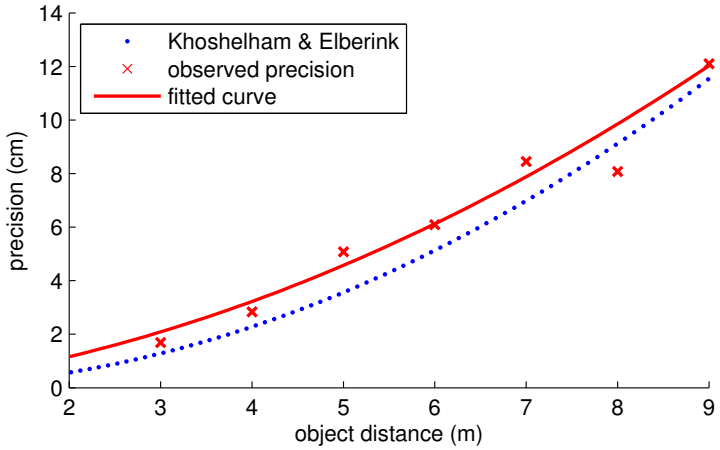


Figure 2.5: Sensor precision as a function of object distance. The dotted curve represents half the depth resolution of the sensor, which according to [47] and [20] approximates its precision. The obtained (d, s_d) pairs are shown in red, together with a fitted curve.

In conclusion, the spatio-temporal precision of the Kinect remains high compared to the measured distances throughout the measuring range. The following sections investigate the relation between the precision and properties of measurement deviations.

2.2.2 Measurement Deviations

Given the high sensor precision, it was postulated that the measurements are stable in general, with strong outliers occurring only infrequently. This assumption is supported

by [2] and [18], who found measurement histograms to be reminiscent of Gaussian distributions. Unfortunately, both [2] and [18] tested only at a single object distance.

In order to confirm that strong measurement deviations are infrequent and to estimate the measurement distribution, the ranges of $\mathbf{x}_d$ were computed as $r_d = \max(\mathbf{x}_d) - \min(\mathbf{x}_d)$. Histograms of $\mathbf{x}_d$ were computed and the bins were sorted by frequency, in descending order. The bins of every histogram were collected until the combined frequencies amounted to at least 90% of the total number of measurements. Then, for each test distance d, the range of the distances corresponding to the collected histogram bins was calculated. These ranges are denoted as a_d. Table 2.2 provides further information.

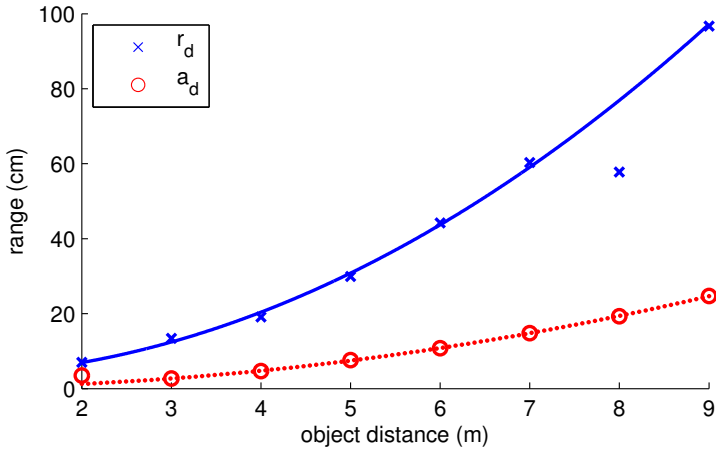


Figure 2.6: (d, r_d) in comparison with (d, a_d) and fitted curves. The reason for the outlier is apparent in Table 2.2 (only four distinct measurement values).

Figure 2.6 illustrates the obtained values for r_d and a_d, as well as fitted curves. It is visible that r_d reached substantial magnitudes at larger object distances. On the other hand a_d remained much smaller. This circumstance, the high goodness of fit (Figure 2.6), and the statistics summarized in Table 2.2 further support the hypothesis that strong outliers are infrequent. In fact, Table 2.2 shows that the numbers of unique measurements of both $\mathbf{x}_d$ (second row) and the collected histogram bins (third row) varied only slightly throughout the measuring range. Thus the measurement stability and distribution in terms of disparity levels was similar over all tested distances. This suggests that the primary reason for the decreasing sensor precision is the decreasing depth resolution.

2.2.3 Error Model

On this basis, it is possible to model the frequencies and magnitudes of outliers as a function of object distance. Such a model is considered valuable as it provides concise information on the data quality to expect at a given object distance.

The precision of the Kinect was verified to correspond to half its depth resolution. In consequence, the model can be estimated by analyzing magnitudes and frequencies of

Table 2.2: Statistics related to measurement deviations. The second row corresponds to the number of unique values of $\mathbf{x}_d$, from which r_d was derived. The third row represents the number of collected histogram bins at every object distance. The last row shows the percentages of measurements used for computing a_d.

object distance d	2m	3m	4m	5m	6m	7m	8m	9m
unique measurements (r_d)	7	6	5	5	5	5	4	5
unique measurements (a_d)	4	2	2	3	2	2	2	2
percentages used for a_d	90.8	92.6	93.4	98.6	95.8	96.4	93.9	99.2

measurement deviations in relation to the sensor precision. The analysis was carried out by regarding $\mathbf{x}_d$ as samples from a random variable X_d, whose properties were estimated based on $\mathbf{x}_d$ via statistical analysis. The expected value of X_d was estimated as the sample mean of $\mathbf{x}_d$ and is denoted as $\overline{x}_d$. The precision of the sensor at test distance d corresponds to the standard deviation of X_d and was estimated via s_d.

Measurement deviations were analyzed relative to the mean-subtracted data, $|\mathbf{x}_d - \overline{x}_d| = \mathbf{m}_d$. Different percentiles of this data were calculated. The pth percentile of a dataset is the value below which p percent of the data fall. With regard to $\mathbf{m}_d$ this means at least p percent of random errors had magnitudes of less than or equal to the pth percentile. Thus, percentiles of $\mathbf{m}_d$, henceforth denoted as $v_{d,p}$, provide concise information regarding the frequencies and magnitudes of deviations from the mean.

Measurement histograms suggest that X_d can be approximated by a Gaussian distribution, $X_d \sim N(\overline{x}_d, s_d^2)$. In order to verify this assumption, $v_{d,p}$ was compared with $\Phi_d^{-1}(q)$, with $\Phi_d^{-1}(\cdot)$ representing the quantile function corresponding to $X_d^0 \sim N(0, s_d^2)$ and $q = 1 - (1 - p/100)/2$.[2] As X_d^0 is Gaussian with a mean of zero, $\Phi_d^{-1}(q)$ represents the interval $[-\Phi_d^{-1}(q), \Phi_d^{-1}(q)]$ with a cumulative probability of $p/100$. Consequently, under the assumption that $X_d \sim N(\overline{x}_d, s_d^2)$ $v_{d,p}$ corresponds to $\Phi_d^{-1}(q)$.

Figure 2.7 (continuous curves) shows the graphs of second order polynomials fitted to $v_{d,90}$, $v_{d,95}$ and $v_{d,100}$, respectively. The graphs of the corresponding $\Phi_d^{-1}(\cdot)$ are depicted as dotted curves. The observed values are reasonably close to the predictions, supporting the above assumption. It is thus considered practicable to estimate the measurement distribution relative to the mean as $N(\overline{x}_d, s_d^2)$. An application for this approach is presented in Section 3.3.1. However, it is important to consider that the limited depth resolution causes the accuracy of the model to decrease with distance (Figure 2.7).

2.3 Reproducibility

The measurements used to examine the data quality in Section 2.2 were recorded in absence of strong infrared (IR) radiation and using a test object whose material had little absorption and reflectance. To be able to determine whether the results are generalizable, effects of IR radiation and different clothing materials must be tested.

[2] $q = 0.9999999$ was used for $p = 100$ as $\Phi_d^{-1}(1) = \infty$.

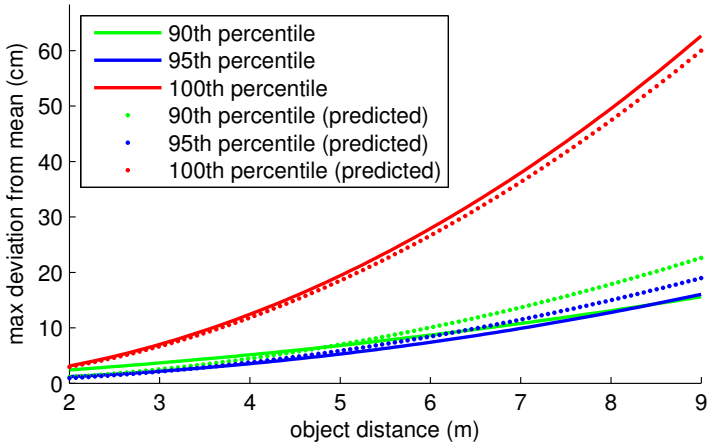


Figure 2.7: $v_{d,90}$, $v_{d,95}$ and $v_{d,100}$ (continuous curves) and predictions (dotted curves).

2.3.1 Clothing Materials

Specular, absorbing, or translucent surfaces can degrade the data quality or preclude measurements [47, 18]. Berger et al. [11] examined different materials and showed that the percentage of zero-pixels is correlated with object specularity. However, clothing materials were not tested. Other authors found that object materials can influence the data quality, but did not conduct tests [47, 2]. Literature on the effect of clothing materials is not available but considered necessary for assessing the applicability of the Kinect sensor for fall detection.

In order to investigate the impact of different clothing materials on the sensor quality, the experiment for estimating the sensor precision was repeated under identical test conditions. However, the surface of the test object was covered with different clothing materials, namely black cotton, red polyester, and black leather.

Figure 2.8 summarizes the results. The precisions for cotton and polyester were close to the reference (Figure 2.5), but those for leather differed considerably. At distances larger than 5m fractions of measurements of leather were zero-pixels (16% at a distance of 7m). No zero-pixels were observed for cotton and polyester.

Clothing worn at home is often made from cotton or materials with similar reflectivity properties, such as wool or linen. The test results indicate that the sensor quality is stable with respect to such materials. It is therefore assumed that common clothing materials have a negligible effect on the data quality.

2.3.2 Lighting Conditions

Strong IR radiation can influence the Kinect as it estimates distances based on a projected IR pattern. Chow et al. [20] found that fluorescent lamps did not impact performance significantly. According to own tests this is also applies to incandescent light bulbs. On the other hand, the IR radiation in sunlight is strong enough to obscure the projected

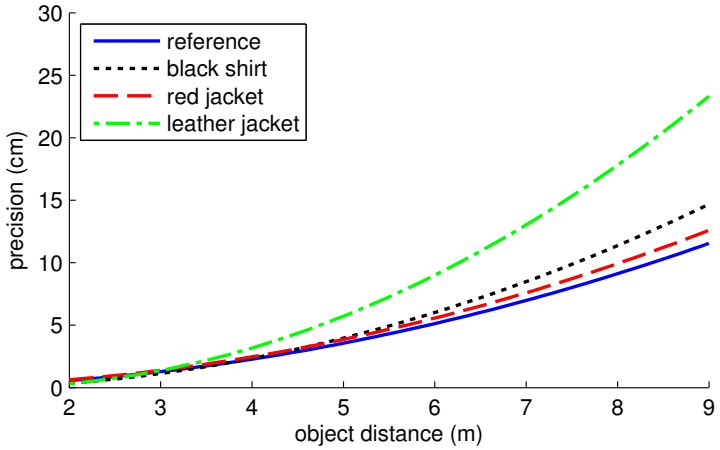


Figure 2.8: Sensor precision for different object materials and test distances.

pattern, degrading or preventing measurements. This effect is mentioned in [47, 2] but analyses are missing in the literature.

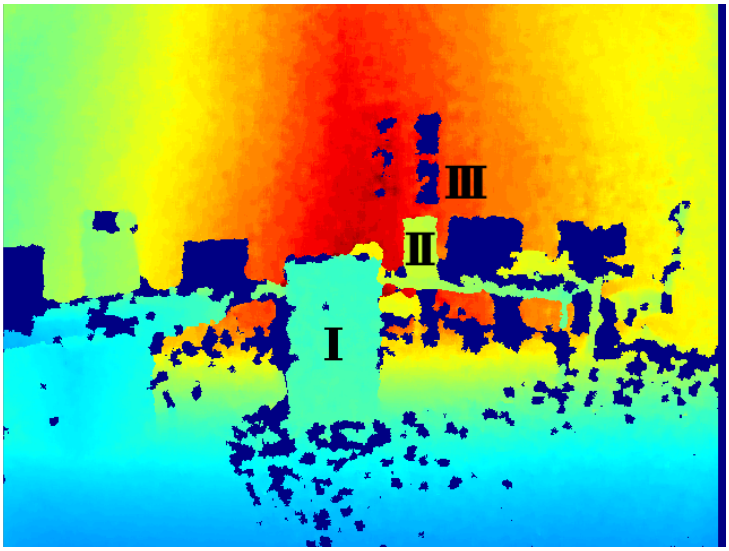


Figure 2.9: Color-coded depth map of the test scenario. The distances from the sensor to the test objects I, II, and III were 4m, 5.5m, and 7.5m, respectively.

In order to analyze the impact of sunlight in an indoor environment, a test setup was established in an office with three large windows. Three test objects in the form of cardboard planes were positioned in the scene as shown in Figure 2.9. The object materials were identical to that of the test object used in Section 2.2 and the objects were oriented towards the sensor. Every 30 minutes 100 frames were recorded, over a

duration of 21 hours. The scene remained static. An illuminance meter was used to measure illuminances at object centers, Table 2.3 summarizes the observed values. For every test object, the sensor precision was estimated as in Section 2.2.1.

Table 2.3: Measured illuminances on the test objects.

time	I	II	III
11:00	950	650	400
12:00	900	500	370
13:00	340	220	160
14:00	550	320	240
15:00	950	570	430
16:00	1700	700	530
16:30	3000	1050	450
17:00	250	150	90

The precision scores and consequently the measurement quality for the test objects I and II remained almost constant, as illustrated in Figure 2.10. This stability is remarkable considering the variances in illuminance. On the other hand, the precision for object III varied considerably. Figure 2.10 shows a correlation between sensor precision and illuminance, but there was a high variability even during nighttime.[3] It is therefore concluded that the sensor robustness decreases with distance but remains relatively high up to distances of around 6m. This seems reasonable as the intensity of the projected pattern decreases with distance because of speckle effects.

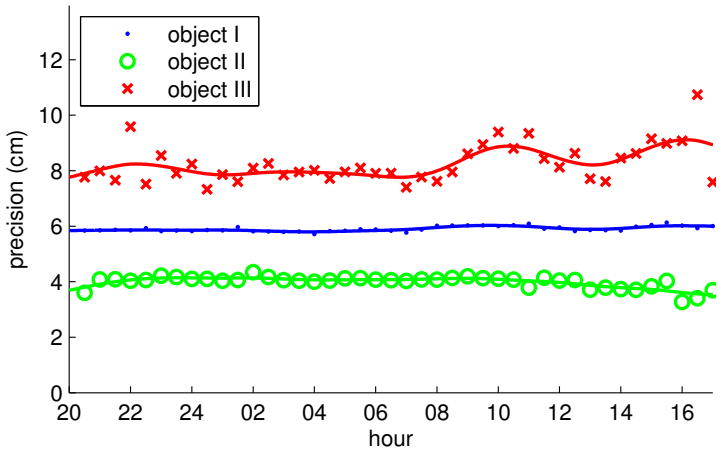


Figure 2.10: Sensor precision over time.

At 16:30 test object I was partially exposed to direct sunlight emitting through a

[3]Different datasets were tested in order to rule out erroneous data, but the variability persisted. A possible cause is internal sensor adaption [2].

window. The sensor was unable to measure distances to exposed parts.[4] The maximum illuminance for which measurements were still possible was approximately 6000 lux. Importantly, this limit was found to vary strongly with sensor distance, surface properties and reflection angles, and is therefore not universally applicable. The fact that direct sunlight exposure can prevent measurements is important with regard to fall detection. Emitting sunlight can cause large areas of zero-pixels on floors. If persons fall in such areas they may not be detectable, or only partially. In order to minimize negative effects on the detection of upright persons, sensors should be positioned at the opposite side of windows so that persons remain detectable because of self-shadowing.

2.3.3 Multiple Sensors

Multiple Kinect sensors with overlapping views can result in a decreased data quality. While the fall detection system presented in this text uses only a single sensor, knowledge of effects of multiple sensors is considered valuable. Berger et al. [11] found that the number of zero-pixels increases with sensor count and object specularity. Distances and angles between sensors can also have an impact [73]. In [2] and [11] pattern overlap caused a small fraction of zero-pixels, but no other effects.

Based on these findings it was expected that a second sensor has only minor effects with respect to fall detection. In order to obtain more information in this regard, a second sensor was present during the experiment described in Section 2.3.2. The second sensor faced test object I as illustrated in Figure 2.11. The other test objects were not in the sensor view and thus not affected by its projection.

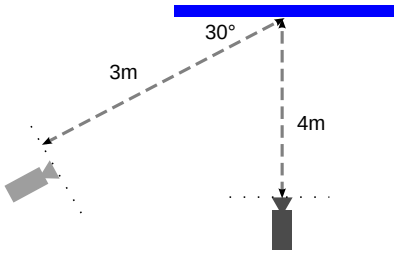


Figure 2.11: Sensor positions relative to test object I.

In presence of a second sensor at most 1.5% of measurements of object I were zero-pixels. In floor areas the percentages were larger, as visible in Figure 2.9. The floor had a higher specularity, thus the results agree with [11]. The measurement quality was also affected. The measured precision for test object I was about 6cm (Figure 2.10), considerably higher than the reference precision at the corresponding distance (about 2.2cm). These results contradict with [11, 2] in which no effects on quality were observed.

Further tests suggest that persons are reliably detectable in presence of a second sensor, but the measurement quality decreases. In consequence, the results derived in Section 2.2 are not applicable in this context.

[4] Zero-pixels were filtered in a preprocessing step and thus did not affect precision scores.

2.4 Accuracy

In this analysis, the accuracy $a(d)$ corresponds to the signed difference between the true distance d of an object and the average of a large number of distance measurements. Knowledge of $a(d)$ allows for correction, $d' = d - a(d)$. Available literature examines the accuracy of the Kinect in near-range only [82, 47, 73].

In order to estimate $a(d)$ a second-order polynomial was fitted to $\overline{x}_d - t_d$, with $\overline{x}_d$ and t_d representing the mean of $\mathbf{x}_d$ and the true object distance, respectively. The graph of $a(d)$, shown in Figure 2.12, resembles the depth resolution of the sensor. This seems reasonable considering that depth resolution is a major cause for the limited accuracy. It should be noted that the results may vary slightly based on the software used.

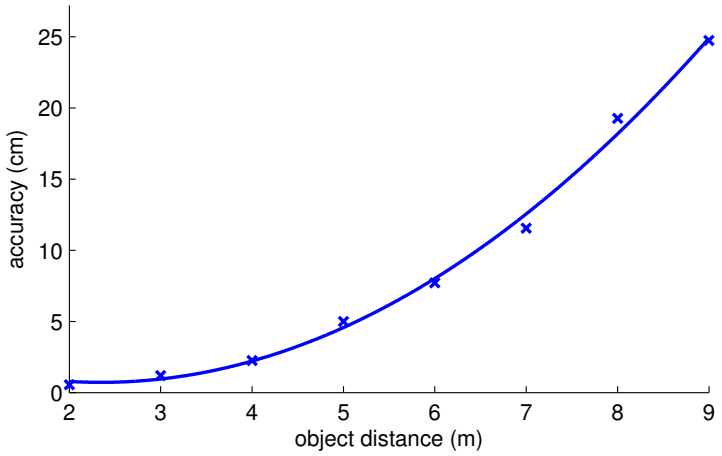


Figure 2.12: Sensor accuracy as a function of object distance $a(d)$.

The results confirm that the accuracy remains high throughout the measuring range, considering the impact of errors on the estimation of y-coordinates (object heights), which are frequently employed for detecting falls. If the height difference between the sensor and a considered point is h, the measured z-coordinate is m and the true value is d, then from similar triangles it follows that $\delta = h(m-d)/m$, with δ denoting the height estimation error induced by the limited accuracy.[5] Given the results shown in Figure 2.12, this error remains below 5cm for $h = 2.2$m, a reasonable height difference between the mounted sensor and a fallen person, and $m \leq 7$m.

2.5 Conclusions

The results presented in this chapter suggest that Kinect sensors are well-suited for fall detection. The resolution and precision were found to be sufficient to allow for a reliable detection of standing and fallen persons. However, fall detection algorithms should

[5] This applies on the condition that the sensor is a pinhole camera and thus represents an approximation.

not rely upon continuous detection of smaller body parts such as limbs. The sensor is expected to be robust in terms of common clothing fabrics and infrared radiation through sunlight for object distances up to approximately 6m. In conclusion it is considered advisable to constrain the region of interest to distances up to 6m from the sensor.

Sunlight emitting through windows can result in regions on the floor for which no measurements are available. Persons that fall in such areas may not be detectable in the data. Reliable fall detection algorithms must account for this possibility.

This chapter introduced an approach for modeling frequencies and magnitudes of outliers relative to the mean as a function of object distance. Knowledge of this model is valuable to the proper configuration of algorithms, as highlighted in Section 3.3.1.

CHAPTER 3

Motion Detection

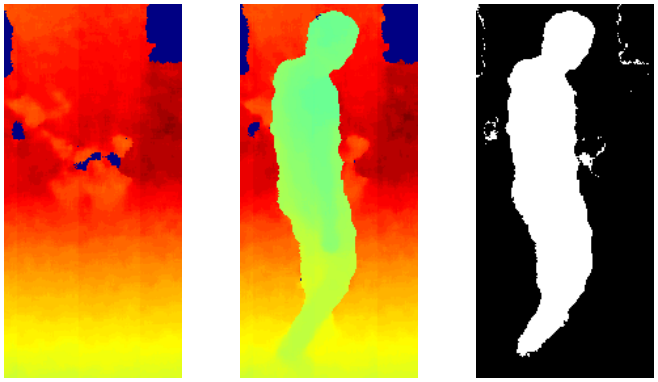
The first step of the proposed fall detection system is the localization of regions in which motion occurs. Motion detection is often the first step in the detection of objects that are expected to be moving. In particular, it is frequently utilized for person tracking [99]. A main incentive for employing motion detection is efficiency. Under the assumption that relevant objects are moving, the search can be restricted to regions in which motion occurs. In terms of pattern recognition, motion detection thus represents a method for search-space reduction [17].

Motion detection must be accurate and fast in order to be feasible for object detection. A family of methods that are able to fulfill these criteria under various circumstances are called Background Subtraction (BS) methods [99]. These methods detect motion in image sequences by comparing image frames to a background model, which is a representation of the static parts of the depicted scene. In the simplest case, a single frame of the same sequence is used (e.g. the most recent frame [42]). Image regions that differ significantly from the model are classified as containing movement. Such regions constitute the foreground, the rest represents the background. Each pixel of the input frame is classified as either foreground or background, thus the output is a binary image, called foreground mask [17]. Figure 3.1 illustrates this process.

3.1 Related Literature

Background subtraction has been used for motion detection for over three decades; an early algorithm was proposed by Jain and Nagel [42] in 1979. A seminal work on background subtraction was presented by Wren et al. [96], who modeled every pixel of the background as a single Gaussian, based on recently observed frames. This allows the background model to include sensor noise and to adapt to gradual scene changes such as lighting changes. Later approaches adopted this on-line approach to model adaption.

Wren et al. used the mean for representing central tendency. A more robust alternative is the median. However, maintaining medians is computationally complex as recently



(a) model (b) input frame (c) classification

Figure 3.1: Background subtraction. Regions of input frames (b) are classified as foreground or background (c), based on similarity with a background model (a).

observed values must be stored and kept sorted. McFarlane and Schofield [61] suggested an efficient shortcut. They showed that the background model converges towards the median if its pixels are incremented or decremented based on observed pixel values. This fact is the basis for the family of $\Sigma - \Delta$ BS algorithms [58, 52].

The aforementioned methods employ unimodal background models. Such models cannot encode sudden input changes that should be regarded as background (e.g. caused by waving trees or sudden lighting changes). Consequently, multimodal background models have been proposed. Stauffer and Grimson [86] improved on the method by Wren et al. by representing each model pixel as a mixture of Gaussians. Many variations and advancements of this method have been presented, a summary can be found in [15]. Elgammal and Davis [30] presented a more flexible non-parametric approach.

More recently, Barnich et al. [10] introduced ViBe, a BS algorithm with interesting properties. ViBe does not use an explicit model for describing variations in images. Instead, each pixel of the background model is comprised on N samples collected from previous frames. Whether an observed value is used to update the model and what sample it replaces is determined probabilistically. ViBe performs comparable or better than other state-of-the-art BS algorithms while being faster than most alternatives [17].

Despite considerable advances in this field, there are still scenarios in which modern BS algorithms perform poorly. For instance, most methods classify shadows cast by moving objects as foreground.[1] Objects with similar appearance than the background may not be reliably detectable, a frequent problem in surveillance. More information on this topic can be found [92, 17]. Many remaining problems with respect to BS in images stem from the fact that changes in the scene are not necessarily associated with significant lighting changes (which would be reflected in images) and vice versa.

[1]Several methods for shadow detection and removal have been proposed, [72] gives a survey.

3.2 Motion Detection in Depth Data

The aforementioned methods were designed for use with grayscale or color images. On the other hand, Kinect returns depth maps, which represent measured distances as opposed to lighting conditions. Remarkably, many BS methods for grayscale images also work with Kinect depth maps because mathematically both types of input are two-dimensional matrices. Therefore, such BS methods have been used to detect motion in depth data. For example, Rougier et al. [74] use the aforementioned method by Wren et al. to detect motion in Kinect depth maps for the purpose of fall detection.

Kinect depth maps are more powerful for use with BS methods than traditional images. This is mainly because there is a direct correspondence between scene geometry and depth maps. This circumstance leads to the following advantages with regard to BS:

- Significant scene changes are always reflected in depth maps. Particularly, Kinects' resolution and robustness with respect to clothing and lighting conditions allow for reliable detection of persons.
- Conversely, significant changes in depth maps are caused only by changes in the scene. For example, lighting changes and shadows have no significant effects.
- Measurement deviations due to sensor noise are unimodal, thus complex background models are not required. For example, the method by Wren et al. [96] is appropriate, given the distribution of measurement errors.
- Thresholds are intuitive as observed values represent distances.

Moreover, the information encoded in depth maps can be utilized to improve the detection performance. For example, foreground objects are always expected to be closer to the sensor than the background. However, to the knowledge of the author, BS methods that explicitly utilize depth information have not been proposed so far. To this end, Section 3.3 introduces a new BS method specifically designed for Kinect depth maps. This method builds on the findings presented in Chapter 2 and explicitly utilizes depth information in order to achieve a high detection performance of both upright and fallen persons. In fact, results presented in Section 3.4 show that the proposed method performs better than previous methods while being faster.

3.3 The DMD Motion Detector

In this section a new background subtraction algorithm for use with Kinect depth data is proposed. This algorithm is henceforth denoted as DMD (Depth map Motion Detector). DMD builds upon previous work in this field but utilizes distance information to increase the detection performance. The main design goals of DMD were:

- High detection performance with respect to upright and fallen persons
- Unmoving and lying persons should be detectable over prolonged time periods

- No learning phase required, fast bootstrapping and ghost removal
- Fast processing speed and low memory consumption
- No configuration required

3.3.1 Classification

DMD was designed based on two central assumptions, namely that (i) observed values represent distance measurements, and that (ii) objects constituting the background are rigid. These assumptions are general and not restrictive in practice.

Under these assumptions a pixel p can represent foreground only if its value is smaller (i.e. closer to the sensor) than the model $m \neq 0$. There are two possible causes for $p < m$: presence of a foreground object, and measurement deviations due to sensor noise. Therefore, the pixel should be classified as foreground only if $p < (m - t)$, with $t > 0$ accounting for sensor noise. A special case is $p = 0$, that is an observed zero-pixel. In this case there is no reliable way to determine the class of the pixel, based solely on comparison with m. DMD always regards such pixels as background. Algorithm 1 shows the resulting classification method.

```
for every depth map pixel do
    if value not 0 and value < (model - threshold) then
        classify as foreground
    else
        classify as background
    end
end
```

Algorithm 1: Pixel classification.

In contrast to existing BS methods, DMD computes t dynamically based on observed distances. This approach has two advantages. First, it allows DMD to optimize t so that foreground objects are detected more accurately even if they are close to the background, which is particularly important for fall detection. Second, it frees the user from selecting appropriate thresholds based on particular scene conditions.

DMD computes dynamic thresholds based on the model presented in Section 2.2.3. According to this model the distribution of the sensor noise is approximately Gaussian, with a known distance-dependent standard deviation σ. Consequently, 99.7% of deviations due to noise deviate from the mean by less than 3σ, which suggests that $t = 3\sigma$ is a suitable choice. Tests confirmed this, except for small object distances. To this end, DMD assigns no thresholds lower than $t_{min} = 10$cm.

For increased efficiency, DMD computes dynamic thresholds only once and stores them in a lookup table for fast retrieval. More precisely, t is calculated and stored for all distances d for which $t > t_{min}$ and $d < d_{max}$, with d_{max} representing the maximum distance observed during initialization.

3.3.2 Model Initialization

Many BS methods require an initial learning phase in which the background model is initialized from observed frames. This increases robustness but delays operability. Furthermore, learning the background over a longer time period can be counterproductive if the scene changes during that time or soon afterwards. An alternative is to initialize the model based on the first observed frame and to rely on on-line learning for improving the model over time. How this initialization is performed depends on the background model. For example, the $\Sigma - \Delta$ family of methods simply use the first frame as the model [52]. However, single frames do not encode temporal variability. ViBe employs an interesting approach to account for this limitation. The method assumes that adjacent pixels depict the same object and uses these pixels during initialization [10].

The model of DMD is a single matrix that represents central tendency of observed measurements, similar to [61, 52]. DMD employs a single-frame approach to model initialization that is similar to ViBe. More precisely, during initialization each model pixel is set to the minimum non-zero value of the corresponding pixel in the first frame and its 4-neighborhood. The minimum is used in order to decrease the likelihood of noise being detected as foreground. The approach of treating measurements from adjacent pixels as equal can lead to significant errors at object borders. DMD is able to detect and correct such errors quickly, as discussed below.

Classification, as shown in Algorithm 1, works correctly only if the model does not contain zero-pixels. Therefore, DMD interpolates model values during initialization. Different methods for this purpose have been proposed [18, 48]. Inpainting methods [21, 90] are feasible as well. However, the focus of these methods appears to be a high interpolation quality, often at the cost of speed. On the other hand, DMD does not require high quality because errors are corrected at runtime.

DMD thus employs a custom interpolation method with sufficient quality and low computational requirements. This method is illustrated in Figure 3.2. Initially, the current pixel and its four nearest neighbors are examined and the minimum non-zero value is obtained. This step corresponds to the model initialization method described above. If no such value exists, the next four nearest neighbors are examined analogously. This iterative algorithm continues until the first non-zero value is found, or until a predefined iteration limit.[2] This results in a star-like sampling, originating from the current pixel. The method is accurate for small areas of zero-pixels as the sampling is dense initially. The algorithm converges quickly even for larger regions because the number of comparisons per iteration remains constant.

3.3.3 Model Adaption

The model adaption scheme of DMD consists of two parts, namely gradual updates and ghost detection. These parts are independent but complement each other.

[2] In the current implementation, this limit is defined as 1/8th of the smallest depth map dimension.

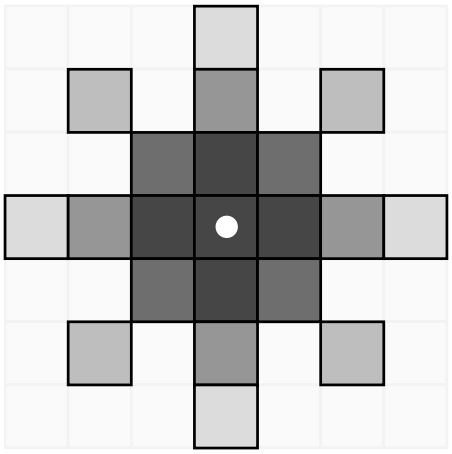


Figure 3.2: Interpolation of zero-pixels during model initialization (first five iterations). Pixels that are examined together have the same color. Darker pixels are processed first. The white circle marks the current pixel.

Gradual Updates

DMD employs the method proposed by McFarlane and Schofield [61] for incorporating information from observed frames into the background model. More precisely, DMD periodically compares each observed value p with the corresponding model value m. m is incremented if $m < p$, otherwise it is decremented. This causes p to converge towards the temporal median of the previously observed values [61]. This model is appropriate, given the distribution of the sensor noise.

Measurements are given in mm, hence m changes by 1mm per update. If the model is updated every α-th frame, this corresponds to a maximum change of f/α mm per second, where f is the framerate. The larger α, the slower the model adapts to changes in the scene. In the context of fall detection, a slow adaption rate is feasible as it allows for a longer detection of fallen persons. However, if the adaption rate is too slow it takes a long time before changes in the scene are reflected in the model, which can cause persistent errors. It was found applicable to to choose α so that $f/\alpha \approx 10$. This allows for detection of fallen persons over prolonged periods of time, while small scene changes are incorporated into the model within few seconds. Scene changes that are significant in terms of measured distances are accounted for by means of ghost detection.

Ghost Detection

A background object that changes its position is detected twice by most BS methods, once at the new position and once at the original position. The erroneous detection at the original position is often referred to as a ghost [10]. A common cause for ghosts are foreground objects that are present during model initialization. In this case the model must be refined gradually, an approach called bootstrapping [92, 17]. In practice, it

cannot be ensured that no foreground objects are present during initialization. Moreover, the background in indoor environments is generally not static over long periods of time (e.g. moved chairs). Therefore, fast ghost detection and correction as well as a high bootstrapping performance were considered mandatory when designing DMD.

Gradual updates alone are insufficient in this regard. If a background object with measured distance d_o was moved, it would take $\alpha/f \cdot |d_o - d_n|$ seconds for the model to converge to the new measured distance d_n. Therefore, DMD specifically detects and corrects for ghosts. For this purpose, it periodically compares each model pixel m with the currently observed measurement $p \neq 0$. Under the assumption that background objects are rigid, p is expected not to exceed $m + t$, unless the background has changed. Consequently, each model pixel is classified as a ghost if $(p - m) > 2t$. The coefficient 2 is added in order to ensure that sensor noise has no effect. Model pixels that are classified as ghosts are set to p. By default, ghost detection is carried out once every two seconds.

3.4 Results and Discussion

In order to evaluate the performance of DMD, the algorithm was applied to two test sequences described below. Ground-Truths (GTs) for both sequences were generated by first processing the sequences with DMD. A low threshold was used (5-7cm, depending on object distance) in order to detect persons accurately, apart from regions that were not present in the data because of the limited sensor resolution and precision. Errors were removed manually in a post-processing step. This data-centric approach was chosen in favor of others because it results in more significant evaluation scores. More precisely, on this basis a perfect motion detection algorithm would be able to score 100% in terms of detection performance. If the GTs were obtained by other means (e.g. based on RGB information as in [19]) this would not be the case as the data would be insufficient. One GT was obtained for every second (30 frames) of the sequences.

The performance of DMD was assessed by means of the PCC-score, a prominent metric for evaluating binary classifiers [81]. The PCC-score yields the percentage of correct classification as

$$\text{PCC-score} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}. \tag{3.1}$$

In the context of background subtraction, the terms of (3.1) have the following meaning:

- TP: number of pixels correctly classified as foreground
- TN: number of pixels correctly classified as background
- FP: number of pixels incorrectly classified as foreground
- FN: number of pixels incorrectly classified as background

DMD was tested with default settings. To allow for comparison, ViBe was tested as well, using the parameters $N = 12$, $t = 200$, $\#_{min} = 2$ and model adaption rates $\phi = \{2, 16\}$. N was set to correspond to about twice the expected data variability due

to sensor imprecision (Table 2.2). t was chosen in correspondence with the maximum assigned threshold of DMD in both scenes. $\phi = 16$ and $\#_{min} = 2$ were defaults in [10]. The detector output was evaluated directly, without post-processing.

An own implementation of ViBe was used, based on the pseudo-code presented in [10]. In contrast to the reference implementation, random numbers were generated using a fast xorshift-algorithm [59]. Given the indeterministic nature of ViBe, the results are thus expected to differ slightly from the reference implementation.

3.4.1 Fall Detection

In order to assess the performance of DMD with respect to fall detection, a sequence was recorded that shows a typical fall. In this sequence a person walks into the view after ten seconds, falls on the floor at a distance of 3.5m from the sensor and remains there, lying flat on the back without moving. The sensor was located at a height of 2.5m with a tilt of 25 degrees. The total duration of the sequence is 30 seconds.

Figure 3.3 shows the PCC-scores obtained for this sequence. For the first ten seconds DMD performed slightly better than ViBe. During this period no foreground objects were present, thus errors are due to false positives. DMD was more robust than ViBe in this regard because DMD never classifies pixels as foreground that cannot possibly represent foreground, regardless of value.

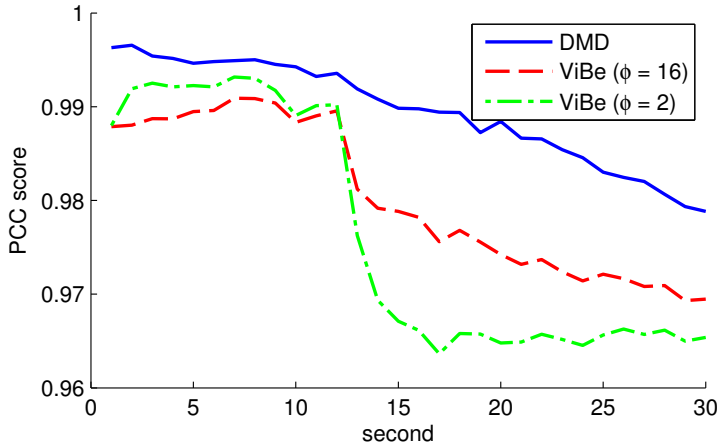


Figure 3.3: PCC-scores for the fall sequence.

During and immediately after the fall (seconds 12 to 15) DMD performed best for two reasons, (i) the aforementioned robustness with respect to measurement noise, and (ii) the smaller threshold t. In fact, the dynamically computed thresholds of DMD were in the range of 10cm at the distance of the fallen person, whereas the threshold of ViBe was set to 20cm. At $t = 20$cm parts of the lying person were not detected. This threshold was chosen based on scene geometry; a smaller threshold would have resulted in a better

detection of the person at the expense of more false positives and consequently a lower PCC-score. DMDs' dynamic thresholds do not have this limitation.

After the fall the scores decreased with time due to model adaption. This effect is highlighted in Figure 3.4, which visualizes the speed at which foreground pixels representing the fallen person were incorporated into the model. DMD incorporated foreground pixels much slower in the model than ViBe.

Proper configuration of update rates depend on the task at hand. For example, a slow update rate is required if the employed fall detection algorithm requires detection of fallen persons over prolonged periods of time. However, a slow update rate may cause problems related to scene changes, particularly slow removal of ghosts. DMD is able to detect and remove ghosts independently, thus fast update rates are not required.

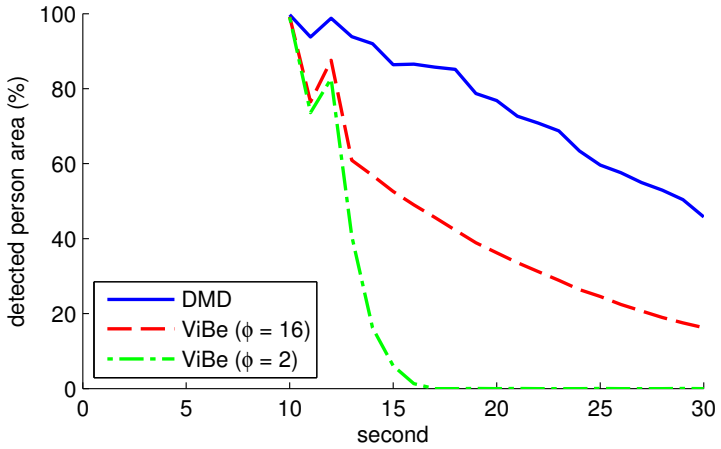


Figure 3.4: Percentage of detected foreground pixels of the lying person over time.

3.4.2 Ghost Removal

The performance of ghost detection and removal was examined by means of a test sequence showing a group of persons in an office environment. In this sequence, five persons are standing still initially for four seconds and then walk around for 26 seconds. The sensor height and tilt were 2.5m and 25 degrees, respectively.

Figure 3.5 visualizes the obtained PCC-scores. The scores were low initially because no persons were detected, as they were part of the background models. DMD performed very well in terms of bootstrapping; the negative effects disappeared almost instantaneously as soon as the persons started to walk. This has two reasons. First, DMD never classifies pixels that are larger (and thus represent further distances) than the model as foreground. Therefore, ghosts cannot cause false positives.[3] Second, the ghost removal algorithm, which runs every two seconds in the default configuration, is able to correct for all ghosts immediately, except for regions that are obscured by foreground objects.

[3]However, they may obscure other objects, causing false negatives.

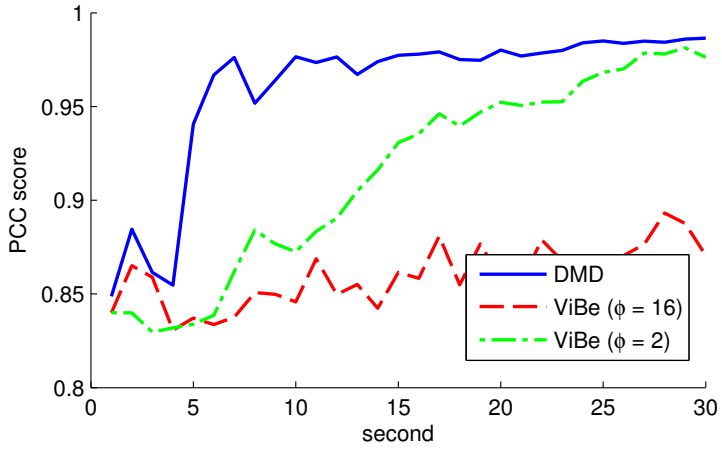


Figure 3.5: PCC-score of the group scene.

The time it takes ViBe to remove ghosts depends on ϕ. With $\phi = 16$ the performance not improve considerably throughout the sequence, as visible in Figure 3.5. With $\phi = 2$ the performance improved faster, but it still took about 25 seconds until all ghosts were removed.[4] These results highlight the importance of fast ghost removal.

At the end of the sequence, the performances of DMD and ViBe with $\phi = 2$ were similar. This was expected as the higher sensitivity of DMD is not beneficial with regard to foreground objects that are distant from the background. In fact, algorithm and configuration choices were found to have only little impact on the detection performance of standing or moving persons.

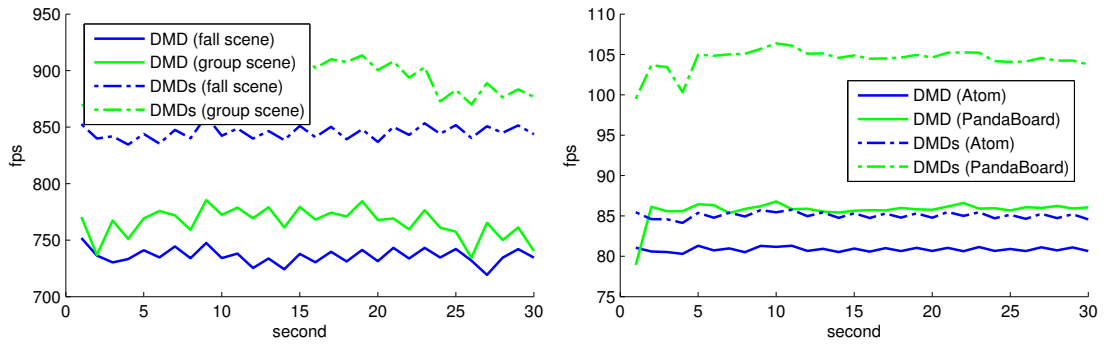
3.4.3 Speed

Motion detection algorithms must be fast even on low-end hardware in order to be applicable for the proposed fall detection system. The speed of DMD was measured using the fall and group sequences. The time required to process every frame was measured and the results of 30 consecutive frames were averaged. All tests were repeated three times, the results reported below represent the mean. Three systems with the following CPUs were tested: Intel Core i7 2600, Intel Atom D2500, and Dual ARM Cortex A9 (PandaBoard ES). Ubuntu 12.04 with GCC 4.7.2 was used on all systems. DMD was compiled directly on all systems with the optimization flags `-O3 -march=native`.

Figure 3.6 (a) shows the measured speed of DMD on the Intel i7 system. The continuous lines depict the speed of DMD in default configuration. The average frame rate over both sequences was 751 fps. This is about three times as fast as ViBe on similar data and hardware [10]. To the knowledge of the author, DMD is faster than all other state-of-the-art background subtraction algorithms.

[4] The reason why the person disappeared more quickly in Section 3.4.1 is that its detected region was much smaller initially than some of the areas representing ghosts.

The dashed lines show the performance of an alternative version denoted DMDs, which uses static thresholds. This limits the applicability for fall detection but has negligible effects with respect to standing or walking persons. DMDs achieved an average framerate of 874 fps, which amounts to an improvement of 16.4%.



(a) Intel i7 2600

(b) Intel Atom & PandaBoard ES

Figure 3.6: Speed of DMD and DMDs on an Intel i7 2600 desktop computer (a) as well as on an Intel Atom system and a PandaBoard ES (b).

Figure 3.6 (b) illustrates the speed on the Intel Atom system and the PandaBoard ES on the group sequence. DMD was fast enough for real-time processing on both systems. The Cortex A9 CPU of the PandaBoard ES was faster than the Atom CPU.

In order to estimate the speed of model-initialization, the times required to initialize the model for the fall and group sequences were measured. The first frame of the fall sequence contained mostly small zero-pixel areas, as summarized in Table 3.1. Model initialization completed in 0.97ms on the Intel i7 system, which corresponds to a framerate of 103fps. The first frame of the group sequence contained four large zero-pixel areas (windows). The initialization speed remained relatively high, given the much larger zero-pixel areas. This confirms that interpolation scales well in this regard.

Table 3.1: Interpolation speed of zero-pixel regions. # denotes the number of zero-pixel regions and q_i represents the ith quantile of region sizes (number of pixels).

sequence	#	$q_{0.25}$	$q_{0.5}$	$q_{0.75}$	$q_{0.9}$	q_1	fps
fall	104	1.0	9.5	56.5	177.0	5211.0	103
group	101	2.0	38.0	101.7	383.8	21522.0	54

CHAPTER 4

Person Detection and Tracking

The aim of person detection and tracking is to detect persons in image sequences and associate these detections correctly between frames. Person detection and tracking is a complex problem due to the nonrigid nature of persons, complex motion, and occlusions, among others [99]. With grayscale or color images, another complication is the fact that the perspective projection of the scene during image formation causes a loss of information. Kinect and other depth sensors allow for a partial reconstruction of the scene geometry, which facilitates the problem. The proposed system utilizes this fact and detects and tracks persons in so-called plan-view maps, concise representations of the scene as viewed from above.

4.1 Related Literature

Person detection and tracking has been an active research field for decades; comprehensive reviews are available in [99, 63, 39]. This section reviews a selection of recent methods designed for or applicable to Kinect sensors, categorized based on the data representation they operate on (depth maps, point clouds, or plan-view maps).

Depth Maps

Methods operating on depth maps frequently utilize histogram-based features and supervised learning for person detection [85, 41, 97]. Conceptually, these methods are similar to a seminal work by Dalal and Triggs [24] that introduced histograms of oriented gradients as powerful features for person detection in grayscale images. These features model the local appearance of objects by means of gradient distribution and, when consolidated to larger blocks, constitute powerful feature descriptors. These descriptors are then classified as (not) representing a person using a support vector machine. Spinello and Arras [85] as well as Wu et al. [97] proposed similar descriptors for depth maps, which model the local distribution depth gradient orientations. The approach by Ikemura and

Fujiyoshi [41] is similar in terms of block generation and classification, but uses pairwise similarities between image regions as features.

These methods reportedly perform well (detection rates of up to 99% at error rates below 10% [41]) but have high computational requirements. In fact, [85] employ GPU programming to achieve real-time performance, while [41] report a framerate of only 10fps on a high-end system. One reason is that they process frames completely and independently; neither motion detection nor tracking are used. Furthermore, they must be trained on a comprehensive database in order to achieve a high performance in practice, but such databases are hard to compile, especially in the context of fall detection.

Xia et al. [98] follow a different approach, first detecting heads of persons via shape matching. Based on these detections, persons are segmented via region growing and tracked based on the coordinates and velocities of their centroids. This approach has the advantage of being less complex and not requiring a training phase, which increases its flexibility. However, its head detection algorithm is expected to fail for lying persons.

Point Clouds

Another approach is to first reproject depth map pixels to world coordinates and to operate on the resulting point cloud. Kelly et al. [45] follow this approach, clustering the point cloud using an iterative top-down approach based on 3D proximity tests. Thresholds are computed dynamically from the observed maximum height and the golden ratio, which allows for estimating the proportion of persons based on their height. Subsequently the obtained clusters are analyzed with respect to under- and over-segmentation via ellipse fitting on shoulder and head height, respectively. This method is able to cope with significant occlusions but is prohibitively slow (less than 1fps including tracking). Furthermore, the assumptions used for clustering do not apply to fallen persons.

Hegger et al. [37] compute surface normals from subsampled point clouds and cluster points using an efficient top-down method that results in small clusters of adjacent points. These clusters are then classified as human or non-human by processing normal vector histograms of the constituting points with a Random Forest classifier. Subsequently, nearby clusters classified as representing a person are merged. According to the authors, the system is too slow for real-time applications and achieves a detection rate of 84%, which is considered too low for reliable fall detection.

Plan-View Maps

A reason why methods operating on point clouds are slow is the large number of points (up to 307200) and the fact that clustering is a complex task. Hegger et al. [37] alleviate this problem by subsampling the point cloud. This is accomplished by discretizing the continuous space into cubic cells, which significantly reduces the number of points at the expense of resolution. A similar yet more extreme approach to data reduction is to utilize so-called plan-view maps, two-dimensional representations of the scene as viewed from the top and under orthographic projection [13]. Plan-view maps are computed from point clouds by subsampling X and Z coordinates (assuming the scene geometry presented in

Section 4.2) and calculating scalar statistics over all points that belong to the same cell. This represents a significant data reduction (up to two orders of magnitude, depending on the configuration). Furthermore, plan-view maps are two-dimensional matrices and hence efficient to process. On the other hand, they convey much less information than the original point clouds. Whether this is a problem depends on the task at hand.

Beymer [13] first showed that plan-view maps allow for efficient detection and tracking of persons in depth data. He introduced occupancy maps, which represent the number of points that belong to each cell, and tracked persons in these maps via Gaussian mixtures and Kalman filtering. Harville [33] computed occupancy and height maps from foreground pixels obtained via background subtraction and tracked persons by means of plan-view templates and Kalman filtering. Height maps encode the largest Y coordinate observed for each cell and thus are powerful representations of the scene geometry. Later works improved the tracking stage and introduced new plan-view maps. Harville [35] used an algorithm reminiscent of particle filtering for tracking. Muñoz-Salinas [79] introduced color maps and employed the CONDENSATION algorithm for tracking.

Conclusions

Considering the aim of this thesis, plan-view maps have several advantages over alternatives. They can be computed comparatively quickly, especially if only foreground pixels are considered. Furthermore, they represent powerful cues for person detection and tracking. Most importantly, (i) person sizes do not depend on the sensor distance, (ii) persons are better separable because of the virtual top-view, and (iii) height maps are robust with respect to partial occlusions. Moreover, they allow for robust features for fall detection, as discussed in Section 5.1. Regarding tracking, complex algorithms are considered superfluous. This is because the number of simultaneously visible persons is expected to be small (three persons at most), considering the purpose of the system.

4.2 Sensor-World Geometry

In order to be able to compute plan-view maps, the relation between points in the scene and pixels of the depth map must be known. Kinect uses a traditional CCD sensor for acquiring IR images, from which disparities are estimated. It is thus applicable to regard depth maps as images from a camera, which simplifies computations.

4.2.1 Sensor Model

A simple model for describing how points in the scene are mapped to an image is the pinhole camera model [32]. This model assumes the coordinate system $(O; \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ with $O(0|0|0)$, $\mathbf{e}_1 = (1,0,0)^T$, $\mathbf{e}_2 = (0,1,0)^T$, $\mathbf{e}_3 = (0,0,1)^T$ and has a single parameter $f > 0$. This parameter, called focal length, defines the image plane, $Z = -f$. The image coordinates of a point $P(X|Y|Z)$ are the (X, Y) coordinates of the intersection point between the image plane $E : (\mathbf{x} - \mathbf{a})^T \mathbf{e}_3 = 0$ and the line $l : \mathbf{x} = \lambda \overrightarrow{OP} = \lambda P$ $(\lambda \in \mathbb{R})$.

Combining both equations yields

$$(\lambda P - \mathbf{a})^T \mathbf{e}_3 = 0 \iff \lambda P^T \mathbf{e}_3 - \mathbf{a}\mathbf{e}_3 = 0 \iff \lambda = -\frac{f}{Z},$$

hence the image coordinates of P are $\mathbf{p} = (x, y)^T = (-fX/Z, -fY/Z)^T$. Figure 4.1 illustrates this model and the resulting mapping, which is called central projection.

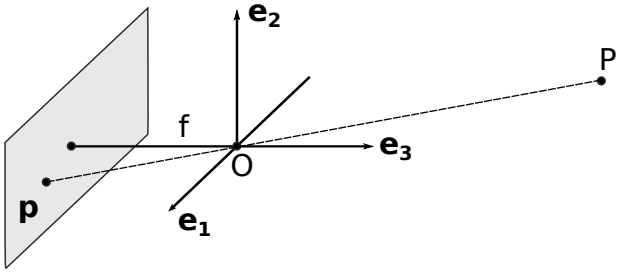


Figure 4.1: The pinhole camera model.

The intersection point of the image plane and the line $l : \mathbf{x} = \lambda \mathbf{e}_3$ ($\lambda \in \mathbb{R}$) is called principal point. The pinhole model assumes that this point has the image coordinates $(x_0, y_0)^T = (0, 0)^T$, but in practice this is not always the case. To this end, the mapping becomes $P(X|Y|Z) \mapsto (-fX/Z + x_0, -fY/Z + y_0)^T$. Moreover, CCD sensors have a finite resolution, hence image coordinates must be discretized accordingly. This can be accomplished by multiplying x and y with the number of pixels per unit distance, m_x and m_y. In practice, this step is usually carried out during central projection. Doing so yields the mapping $P(X|Y|Z) \mapsto (\alpha_x X/Z + x'_0, \alpha_y Y/Z + y'_0)^T$ with $\alpha_x = -m_x f, \alpha_y = -m_y f, x'_0 = m_x x_0$, and $y'_0 = m_y y_0$ [32]. This mapping can be written as

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \mapsto \begin{pmatrix} \alpha_x X + Z x'_0 \\ \alpha_y Y + Z y'_0 \\ Z \end{pmatrix} = \begin{pmatrix} \alpha_x & 0 & x'_0 \\ 0 & \alpha_y & y'_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \mathcal{K} P = \tilde{\mathbf{p}}, \tag{4.1}$$

with $\tilde{\mathbf{p}}$ representing the homogeneous image coordinates of P. The calibration matrix $\mathcal{K}$ can be obtained by means of camera calibration. Different methods for calibrating the Kinect have been proposed [82, 47, 38]. According to Smisek et al. [82] the parameters of the infrared sensor are $\alpha_x = \alpha_y = -585.6$px, $x'_0 = 316$px and $y'_0 = 247.6$px.

Infrared images and depth maps are shifted by about three pixels in both directions [82]. This shift can be corrected by adapting x'_0 and y'_0 accordingly. The resulting model approximates the mapping between points in the scene and depth map pixels. This model was found sufficiently accurate with respect to fall detection, in spite of neglecting factors such as lens distortions.

4.2.2 World Coordinates

In practice, the camera coordinate system varies depending on how the camera is positioned in the scene. It is thus practicable to regard $(\mathbf{0}; \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$ as the so-called

world coordinate system, and to examine how both coordinate systems are related. Two coordinate systems with origins O_1, O_2 are always related by a translation by $O_2 - O_1$ and a rotation R. Consequently, the mapping from world to camera coordinates is $P_c = R(P - O)$ [32]. This mapping can be written in terms of homogeneous coordinates,

$$\tilde{P}_c = \begin{pmatrix} R & -RO \\ \mathbf{0}^T & 1 \end{pmatrix} \tilde{P} = \mathcal{E}\tilde{P}. \tag{4.2}$$

Image coordinates are thus obtained by mapping points from world to camera coordinates before applying $\mathcal{K}$ (4.3). The matrix $\mathcal{C}$ is denoted camera matrix.

$$\begin{pmatrix} \tilde{\mathbf{p}} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathcal{K} & 0 \\ \mathbf{0}^T & 1 \end{pmatrix} \mathcal{E}\tilde{P} = \mathcal{C}\tilde{P} \tag{4.3}$$

4.3 Sensor Calibration

$\mathcal{E}$ varies with the sensor position and orientation, which are not known beforehand. The proposed system is able to estimate $\mathcal{E}$ automatically during startup, which simplifies the system setup. This is accomplished by detecting the floor and fitting a plane to the corresponding points in camera coordinates, which yields the equation of the ground plane in camera coordinates. Subsequently the sensor position and height can be estimated by comparing this equation with the known ground plane equation in world coordinates.

4.3.1 Floor Detection

The first step of estimating $\mathcal{E}$ is the detection of depth map pixels that constitute the floor. A fast and reliable approach for this purpose is to use so-called v-disparity images. These scene representations are popular for road and obstacle detection [51, 16] but also allow for detection of the floor in indoor scenes [74]. V-disparity images are obtained by computing a histogram with n bins for each depth map row r, considering only values inside some interval $[z_{min}, z_{max}]$. This results in an image whose pixels encode the frequency of measurements of particular distance ranges. On the condition that the sensor is positioned correctly (not flipped upside-down and with little rotation along the $\mathbf{e}_3$-axis), distance measurements of the floor change approximately linearly with respect to r. Consequently, depth map pixels that represent the floor constitute a line in the v-disparity image [51]. Figure 4.2 shows an example.

The floor can thus be found by means of line detection in the v-disparity image. The current implementation uses Hough transform for this purpose, as in [51]. Detected lines are examined in terms of length, slope as well as position and the line that most likely represents the floor plane is selected. The criteria the floor line must fulfill are that (i) it must exceed some minimum length, (ii) the slope between its normal vector and $(1, 0)^T$ must be $0 < \alpha < \pi/2$, and (iii) it must appear rightmost in the v-disparity image. These criteria allow for reliable detection of the floor plane in presence of other lines (which originate from desks and walls, for example), as illustrated in Figure 4.2.

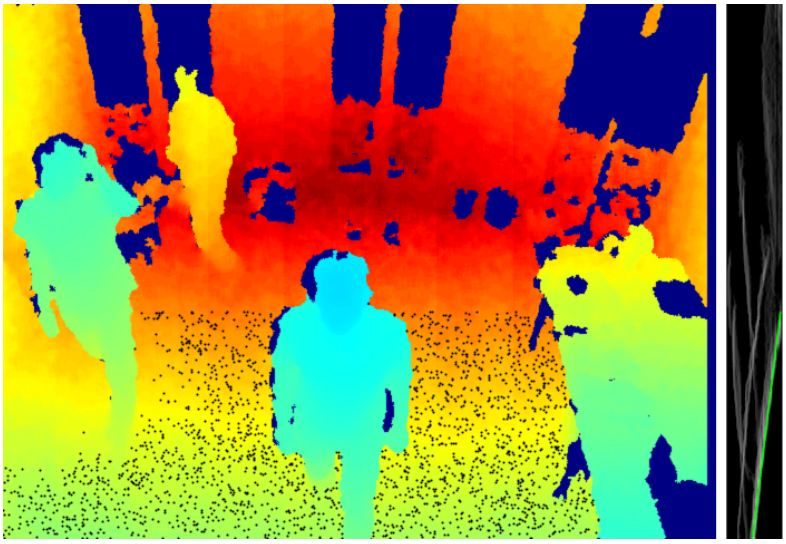


Figure 4.2: A color-coded depth map with samples from the floor (left), as obtained via detection of the floor line in the corresponding v-disparity image (right).

Once the floor line has been found, depth map pixels representing the floor are obtained on a row-by-row basis. This is accomplished by selecting pixels that fall in the interval $[z_r - \varepsilon, z_r + \varepsilon]$, with z_r being the distance represented by the floor line at row r. The current implementation uses $\varepsilon = 25$cm, which is appropriate considering the quality characteristics of the sensor.

4.3.2 Sensor Calibration

After floor detection a subset of the floor pixels is selected randomly and projected to camera space, $P_k = \mathcal{K}^{-1}\tilde{\mathbf{p}}_k$. The projected points comprise a point cloud in camera coordinates from which the equation of the floor plane can be estimated via plane fitting. There are no outliers (ε is small compared to the observed distances), hence robust techniques such as RANSAC are not beneficial. Thus total least squares is employed, that is the goal is to find the plane $E : (\mathbf{x} - \mathbf{a})^T\mathbf{n} = 0$ with $||\mathbf{n}|| = 1$ that minimizes $\sum_k (P_k - \mathbf{a})^T\mathbf{n}$. It can be shown that E contains the barycenter $\mathbf{p} = 1/m \sum_k P_k$ of the projected points [80], hence E satisfies $\mathbf{y}^t\mathbf{n} = 0$ with $\mathbf{y} = \mathbf{x} - \mathbf{p}$ and thus minimizes

$$\sum (Q_k^T\mathbf{n})^2 = \sum Q_k^T\mathbf{n}Q_k^T\mathbf{n} = \mathbf{n}^T \underbrace{\left(\sum Q_k Q_k^T\right)}_{=\Sigma\in\mathbb{R}^{3\times 3}} \mathbf{n} \qquad (Q_k = P_k - \mathbf{p}) \tag{4.4}$$

under the constraint $||\mathbf{n}|| = 1$ (which implies $\mathbf{n}^T\mathbf{n} = 1$). The desired minimum can be calculated using Lagrange multipliers, that is by solving for $\partial F/\partial(\cdot) = 0$ with $F = \mathbf{n}^T\Sigma\mathbf{n} - \lambda(\mathbf{n}^T\mathbf{n} - 1)$. One obtains $\partial F/\partial\mathbf{n} = 2\Sigma\mathbf{n} - 2\lambda\mathbf{n} = 0$ (Σ is symmetric by definition) and thus $\Sigma\mathbf{n} = \lambda\mathbf{n}$. This implies that $\mathbf{n}$ is a normalized eigenvector of Σ.

On the condition that $||\mathbf{n}|| = 1$ the term $-\lambda(\mathbf{n}^T\mathbf{n} - 1)$ vanishes, hence $\mathbf{n}$ minimizes $\mathbf{n}^T\Sigma\mathbf{n} = \mathbf{n}^T(\lambda\mathbf{n}) = \lambda\mathbf{n}^T\mathbf{n} = \lambda$. Consequently, $\mathbf{n}$ is the normalized eigenvector of Σ with the smallest eigenvalue λ. This eigenvalue analysis of Σ (which represents the covariance of the samples around their mean) is called principal component analysis [88].

A viable alternative for obtaining $\mathbf{n}$ is by means of singular value decomposition of $Q = (Q_1 \cdots Q_m)^T$. Namely, $\mathbf{n}$ corresponds to the right singular vector of Q with the smallest singular value [88, 83]. Singular value decomposition has the advantage of being numerically stable [84]. Computation times were found similar for $m = 2000$.

Another optimization is to estimate the hyperplane coordinates $\tilde{\mathbf{e}} = (a, b, c, d)^T$ of E in projective space, which is possible directly from $P = (P_1 \cdots P_m)^T$. Similar to before $\tilde{\mathbf{e}}$ corresponds to the right singular vector of P with the smallest singular value.

The matrix $\mathcal{E}$ is obtained directly from $\tilde{\mathbf{e}}$. On the condition that $b \geq 0$ (which can always obtained by computing $\tilde{\mathbf{e}} = -\tilde{\mathbf{e}}$ if necessary) the distance between the origin (the sensor location) and the floor plane is $s = d/n$ with $n = ||(a, b, c)^T||$. The sensor orientation corresponds to the rotation that maps $\mathbf{e}_2 = (0, 1, 0)^T$ (the normal vector of the floor plane in world coordinates) to $\mathbf{n} = (a, b, c)^T/n$. By definition the angle φ between $\mathbf{e}_2$ and $\mathbf{n}$ satisfies $\cos(\varphi) = \mathbf{e}_2^T\mathbf{n}$. Consequently $\mathbf{e}_2$ can be transformed to $\mathbf{n}$ by rotating $\mathbf{e}_2$ by $\varphi = \arccos(\mathbf{e}_2^T\mathbf{n})$ along the axis $\mathbf{v} = \mathbf{e}_2 \times \mathbf{n}$. The rotation matrix corresponding to this so-called axis-angle representation $(\mathbf{v}, \varphi) = (v_1, v_2, v_3, \varphi)$ can be calculated by means of Rodrigues' formula [88], $R = I + \sin(\varphi)K + (1 - \cos(\varphi))K^2$, with

$$K = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix}. \tag{4.5}$$

Subsequently $\mathcal{E}$ is obtained as

$$\mathcal{E} = \begin{pmatrix} R & -sR\mathbf{e}_2 \\ \mathbf{0}^T & 1 \end{pmatrix}. \tag{4.6}$$

4.3.3 Parameter Smoothing

$\mathcal{E}$ is obtained from several frames in order to increase robustness. This is accomplished by estimating $\tilde{\mathbf{e}}_1 \cdots \tilde{\mathbf{e}}_m$ over m consecutive frames (currently $m = 30$) and computing $\mathcal{E}$ based on $\mathbf{n} = \text{median}(\mathbf{n}_1, \ldots, \mathbf{n}_m)$ and $s = \text{median}(s_1, \ldots, s_m)$.

4.3.4 Results and Discussion

Table 4.1 summarizes test results regarding calibration performance in the office environment illustrated in Figure 4.2. The results were obtained by applying the algorithm ten times in succession for each tested sensor position. The mean calibration errors (difference between the actual sensor height and the mean of the estimations) were approximately 5cm and the standard deviations of the estimations were below 1cm. The values of the last column were obtained by applying the estimated matrices $\mathcal{E}_1 \cdots \mathcal{E}_{10}$ on a vector and calculating the standard deviation of the angles between the results. The values thus

indicate the variance between successive estimations of R. The results suggest that the proposed calibration method is able to estimate $\mathcal{E}$ accurately and reliably, and that it is robust with respect to sensor height and tilt. However, the sensor rotation along the $\mathbf{e}_3$-axis of the world coordinate system should be small to ensure that the floor appears as a distinctive line in v-disparity images. The processing time required for estimating $\mathcal{E}$ is less than two seconds on a PandaBoard ES for $m = 30$.

Table 4.1: Calibration performance in an office environment at different sensor positions and tilts (negative rotation along the $\mathbf{e}_1$-axis).

sensor height	sensor tilt	mean error	stddev	angle stddev
1.5m	15°	5.3cm	0.9cm	0.22°
2.0m	25°	5.0cm	0.3cm	0.12°
2.5m	35°	4.9cm	0.5cm	0.05°
3.0m	45°	5.9cm	0.9cm	0.20°

Evaluation in practice has confirmed the reliability of the proposed method. Moreover, the method is flexible in terms of scene conditions as long as (i) the sensor rotation along the $\mathbf{e}_3$-axis is small and (ii) the floor is visible. A correct sensor placement can be ensured by positioning the sensor on top of furniture or other planar objects. The latter requirement is potentially restrictive but was found less so in practice. This is because the floor is detectable as long as larger parts are visible, even in cluttered environments. However, the method may fail if only a small fraction of the floor is visible. Improvements in this regard are planned, for example by iteratively adapting parameters for ground line detection or using a more flexible line detection algorithm.

4.4 Person Detection

The proposed system detects persons using occupancy and height maps that are generated from regions classified as foreground during motion detection. Given the high data quality, robust person detection can be achieved by classifying obtained occupancy and height map regions using simple geometrical properties. Features that are too restrictive for detecting fallen persons are used only initially.

4.4.1 Computation of Plan-View Maps

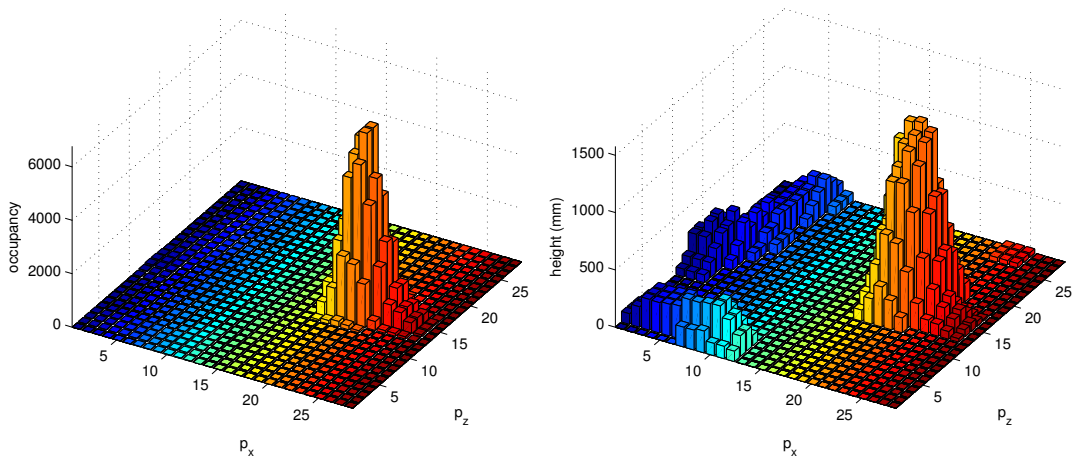
Algorithm 2 shows how occupancy $\mathcal{O}$ and height maps $\mathcal{H}$ are computed from a depth map and a foreground mask. Firstly, the world coordinates (X, Y, Z) of each foreground pixel $\mathbf{p} = (x, y, Z)^T$ are obtained. This is accomplished by multiplying the homogeneous version $\tilde{\mathbf{p}}$ with the inverse of the camera matrix $\mathcal{C}$. On this basis, the plan-view coordinates of the pixel are $(|(X - X_{min})/\delta|, |(Z - Z_{min})/\delta|) = (p_x, p_z)$.[1] Only points within a certain region of interest $([X_{min}, X_{max}], [Y_{min}, Y_{max}], [Z_{min}, Z_{max}])$ are considered. $Y_{min} = -20$cm and

[1] The symbol $||$ means „round to nearest integer".

$Y_{max} = 2$m are reasonable choices for person detection (the negative lower limit accounts for imprecise calibration). The other limits are selected automatically based on scene analysis, as described in Section 5.3. Proper selection of δ depends on the data quality and resolution requirements. Setting δ too low causes significant noise due to the limited sensor resolution and precision. Considering the data quality of Kinect, an appropriate choice in terms of noise robustness is to define $\delta = 7.5$cm and apply a 3×3 box filter. Figure 4.3 illustrates an occupancy map and a height map.

Set $\mathcal{O}(\cdot) = \mathcal{H}(\cdot) = 0$
for *every depth map pixel* **p do**
  if *classified as foreground* **then**
    map to world coordinates: $\tilde{P} = \mathcal{C}^{-1}\tilde{\mathbf{p}}$
    if *P lies within the region of interest* **then**
      compute plan-view coords.: $p_x = |(X - X_{min})/\delta|, p_y = |(Y - Y_{min})/\delta|$
      update occupancy map: $\mathcal{O}(p_x, p_z) = \mathcal{O}(p_x, p_z) + (Z_c/1000)^2$
      update height map: $\mathcal{H}(p_x, p_z) = \max(\mathcal{H}(p_x, p_z), Y)$
    end
  end
end

Algorithm 2: Computation of plan-view maps.



(a) occupancy map

(b) height map

Figure 4.3: Occupancy and height maps generated from a depth map. The peaks towards the right represent a person.

Occupancy and height maps are complementary with regard to person detection. The former are robust in terms of noise, but not in terms of occlusions. On the other hand, the latter are less affected by partial occlusions but susceptible to noise, as visible

in Figure 4.3 (b). To this end, occupancy maps are frequently utilized to height maps, by thresholding height maps based on occupancy [33, 34, 79]. The occupancy threshold t_0 must be selected carefully as occupancies of lying persons are naturally lower than those of upright persons. An appropriate threshold for the given plan-view configuration, $t_0 = 300$, was found by examining occupancy statistics of fallen persons from different distances and angles. This threshold did not affect regions corresponding to fallen persons significantly, while removing most errors due to sensor noise.

4.4.2 Detection of Candidate Regions

In [33, 35] plan-view regions that ought to represent persons are detected in a greedy fashion. In each iteration the plan-view cell with the largest occupancy is located and a quadratic region with a fixed size around this cell is selected as a person candidate region. Subsequently, all occupancies in this region are set to zero and the next iteration starts, until the maximum occupancy decreases below a threshold. A similar method is employed in [79]. This approach is inadequate for fall detection because the area occupied by persons varies on an individual basis and increases significantly if they are lying. Setting the area large enough to encompass lying persons increases the likelihood of undersegmentation in presence of multiple persons.

The proposed system detects candidate regions via connected component analysis of the binary version of the height map, $\mathcal{H} > 0$. This approach is suitable for fall detection because regions of persons are stable in plan-view space, as verified using the fall database kindly provided by the authors of [69]. This database is henceforth referred to as „the fall database". Connected components have the advantage of representing regions occupied by persons more precisely than templates with predetermined shapes. For computation the algorithm presented in [36] was optimized with regard to the characteristics of the input (the number of connected components is small in general).

4.4.3 Feature Selection

The resulting candidate regions $R_1 \cdots R_n$ must be analyzed for whether they represent persons. From a pattern recognition standpoint, this represents a binary classification problem and the goal of feature selection is to find features that are distinctive in this regard. For this purpose, it is assumed that persons are upright when they enter the field of view. This assumption enables powerful features for classifying new objects but is too restrictive with regard to fallen persons. Therefore, some features are used only initially, that is until the object in question has been classified as a person for several frames.

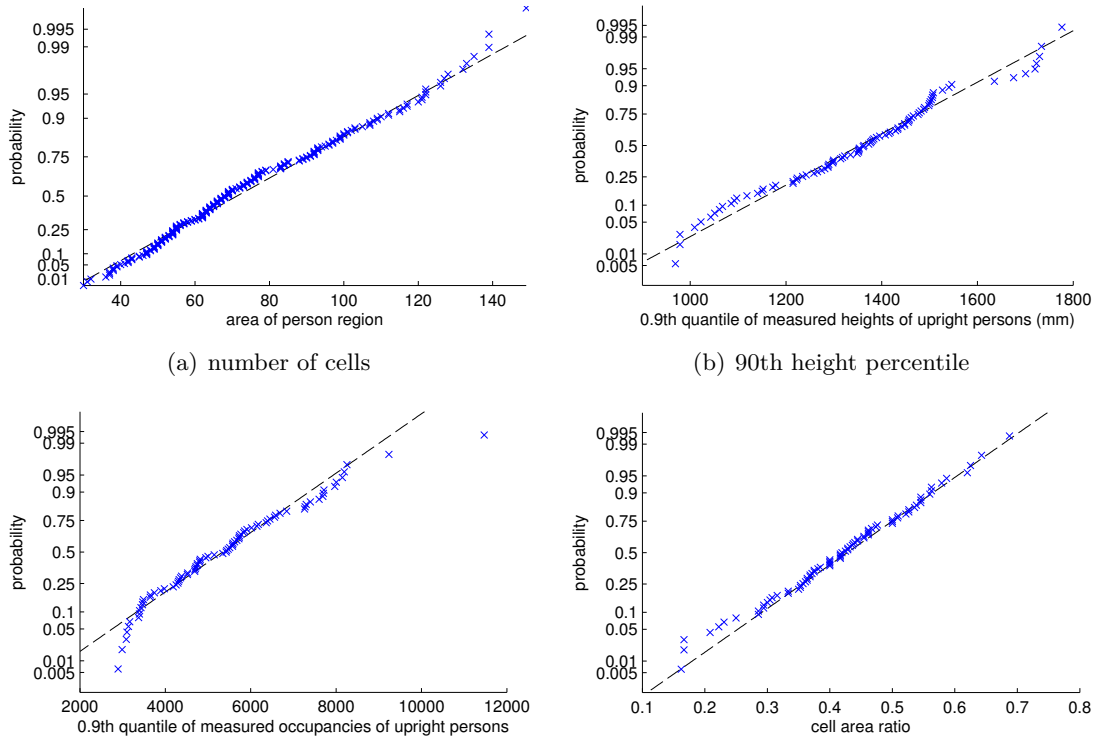
One used feature is the number of cells of R_i, which represents the area occupied by the object. An analysis of 250 frames from different scenes depicting standing, sitting, and lying persons from different distances and angles suggests that this feature follows a Rayleigh distribution with $\hat{\mu} \approx 68$ (Figure 4.4 (a)).

On the condition that persons appear upright, a naturally distinctive feature is the maximum height observed for R_i [35]. Figure 4.4 (b) shows the distribution of the 90th percentiles of regions representing upright and sitting persons, obtained from the

aforementioned dataset. As expected, this distribution is approximately Gaussian with maximum likelihood estimates of $\hat{\mu} = 1341$ and $\hat{\sigma} = 205$. The 90th percentile is used in favor of the maximum because it is a more robust statistic.

Another intuitively suitable feature is the 90th percentile of occupancies, an indicator for object size (Figure 4.4 (c), $\hat{\mu} = 5426$, $\hat{\sigma} = 1661$). However, this feature is susceptible to measurement noise and partial occlusions. Furthermore, it is not completely stable with respect to object distance from the sensor, in spite of the fact that occupancies are normalized. These reasons explain the high standard deviation.

The fourth feature employed is a simple shape feature, namely the number of cells with heights of at least $m - 20$cm divided by those with heights of at least $m - 50$cm, with m representing the observed maximum. The rationale for using this feature is that the head of a person has a smaller size than the upper body, thus the ratio is expected to be below 1. Results obtained from the test set confirm this hypothesis (Figure 4.4 (d), $\hat{\mu} = 0.42$, $\hat{\sigma} = 0.11$). This feature is general, robust with respect to partial occlusions, and rotational invariant.



(a) number of cells

(b) 90th height percentile

(c) 90th occupancy percentile

(d) cell area ratio

Figure 4.4: Distribution plots of features obtained from candidate regions that represent persons. Plot (a) shows the occupied area of upright, sitting, and fallen persons in comparison with a Rayleigh distribution. The other plots depict features obtained from upright or sitting persons in comparison with Gaussian distributions.

4.4.4 Classification

The task of classification is to examine the features derived from R_i and deduce whether R_i represents a person or not. This is often accomplished by means of supervised learning, that is by training a classifier on a database of feature vectors with known class labels. This allows the classifier to learn decision boundaries between classes in feature space, which in turn are used for classification. To this end, person detection in plan-view maps can be achieved by manually labeling a large number of candidate regions as positive (representing a person) or negative, and use the resulting database to train a classifier, which is then used to classify R_i (the histogram-based methods reviewed in Section 4.1 follow this approach). Negative samples can also be obtained in large numbers by generating candidate regions randomly, on the condition that no persons are present.[2]

In order to assay the performance of this approach for person detection in plan-view maps, a dataset containing 80 positive and 800 negative regions was obtained from the fall database. Linear discriminant analysis applied on this dataset using the features b, c, d (Figure 4.4) achieved an average classification sensitivity of 0.99 at a specificity of 0.77 (obtained via ten-fold stratified cross-validation). This analysis is flawed (e.g. sampling bias and the fact that the negative features were not approximately Gaussian) but nonetheless suggests that supervised learning is suitable, on the condition that an extensive training database is available.

However, it is hard to compile such a database because of the large variation in the appearance of persons, particularly in combination with assistive equipment such as rollators. Furthermore, manually obtaining an extensive set of negative samples (as opposed to random sampling, which is flawed because it generates synthetic regions) is time consuming and arguably unfruitful. To this end, feature vectors $\mathbf{x} = (a, b, c, d)$ are classified solely based on comparison with positive samples. For this purpose, it is assumed that the features are statistically independent, thus $P(\mathbf{x}) = P(a)P(b)P(c)P(d)$. This is a strong assumption that does not actually apply, but it allows for an efficient component-wise classification. The rationale is that $\mathbf{x}$ can be rejected as soon as any factor in the above equation is found to be below a threshold.

To further increase the classification speed, the current implementation does not compute probabilities but utilizes step functions $S(a), S(b), S(c), S(d)$. These functions are defined based on the estimated feature distributions of positive samples. For example, considering Figure 4.4 (a) a conservative choice for $S(a)$ with an expected false rejection rate of about 1% is given in equation (4.7). The other features are approximately normally distributed, hence appropriate thresholds are $[-2.5z, 2.5z]$, with z denoting the z-score.

$$S(a) = \begin{cases} 1, & 30 \leq a \leq 160 \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

On this basis, $\mathbf{x}$ is classified as representing a person if $S(a)S(b)S(c)S(d) = 1$. This implies that $\mathbf{x}$ does not represent a person if any factor is zero, which increases efficiency; the system computes and evaluates features in succession and aborts as soon as $S(\cdot) = 0$.

[2] This approach is used in [85].

4.4.5 Results and Discussion

The training error (i.e. the fraction of misclassified samples) of the proposed classification algorithm on the aforementioned dataset is 0.18 (sensitivity: 0.99, specificity: 0.8).

In order to obtain an objective estimate of the classification performance, one frame was extracted every second for sequences from the fall database that depict upright persons. The person detection algorithm was applied to each of the 625 frames obtained this way. It was able to detect all persons correctly (530 regions), as assessed via visual comparison of the input and the detector output. Only three detected regions did not represent persons but were caused by sensor noise. These results confirm the applicability of the person proposed detection method. Future work will focus on improving the training database by incorporating data recorded in practice.

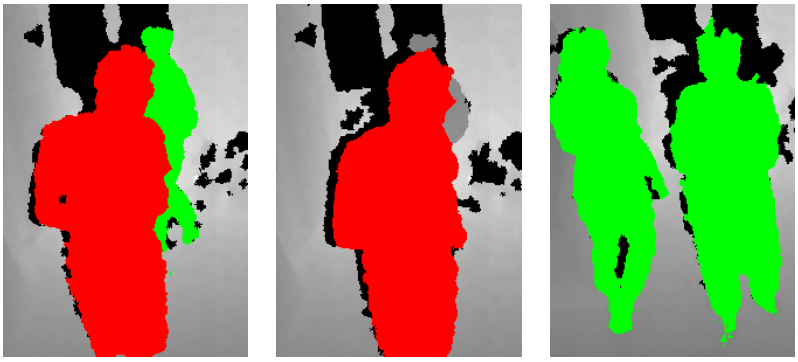
The detection performance in presence of multiple persons was assayed on two test sequences, which were recorded in an office environment at a sensor height of 2.2m. In these sequences, two and three persons, respectively, are walking around without restrictions at distances of up to 5m from the sensor. Both sequences have a duration of 60 seconds. For evaluation purposes, three frames per second were extracted from each sequence. The sensitivity decreased with an increasing number of persons but remained above 94%. Table 4.2 summarizes the person detection performance.

Table 4.2: Performance of the proposed person detection method on test frames depicting one, two, and three persons, respectively.

# persons	positives	true positives	sensitivity	false positives
one person	530	530	100%	3
two persons	322	317	98.5%	1
three persons	492	465	94.5%	0

There are two causes for the decreasing sensitivity. First, the likelihood of occlusions increases with the number of present persons. The person detection method is robust with respect to partial occlusions, but significant occlusions can cause misdetections. This circumstance is illustrated in Figure 4.5. Second, persons that stand close together can cause the corresponding plan-view regions to merge, which leads to undersegmentation (Figure 4.5 (c)). This problem is accounted for during tracking.

The connected component algorithm employed for detecting person candidate regions has a time complexity of $O(n)$, with n representing the number of plan-view cells [36]. The time complexity for computing the features for a candidate region is $O(m \log m)$ (features b and c require sorted data), with m denoting the number of occupied cells. Erroneous region candidates due to sensor noise are often small and thus rejected early, reducing the time complexity for computing features to $O(1)$ (feature a can be computed in constant time). In summary, the person detection method has moderate computational requirements and scales well with the number of visible persons. Quantitative test results summarized in Figure 4.9 confirm this fact.



(a) partial occlusion (b) partial occlusion (c) undersegmentation

Figure 4.5: Visualization of the detector output; each detected person is shown in a different color. The proposed person detection method is robust with respect to partial occlusions (a) but fails if the degree of occlusion becomes too high (b). The second cause for errors is undersegmentation of close persons (c).

4.5 Tracking

For the purpose of tracking, each region R_j is represented as a point in three-dimensional plan-view space, denoted as $\mathbf{r}_j = (x_j, z_j, h_j)$. $\mathbf{c}_j = (x_j, z_j)$ is the center of mass of R_j, calculated as stated in Equation 4.8. The center of mass is used in favor of the centroid because it is more stable with respect to moving limbs. This is because the corresponding regions have lower occupancies and hence contribute less to $\mathbf{c}_j$. $h_j = \mathcal{H}(\mathbf{c}_j)/\delta$ represents the height of R_j at the position of its center of mass in plan-view coordinates. If $\mathbf{c}_j \notin R_j$, which is possible unless R_j is convex, the maximum observed height is used instead.

$$\mathbf{c}_j = \frac{1}{\sum \mathcal{O}(R_j)} \sum_{\mathbf{p} \in R_j} \mathcal{O}(\mathbf{p})\mathbf{p} \tag{4.8}$$

Similarly, each track is represented by a tuple $\mathbf{t}_i = (x_i, z_i, h_i)$. The problem is thus to find, in each frame, the optimal association between the known tracks $\mathbf{t}_1 \cdots \mathbf{t}_m$ and the detected person regions $\mathbf{r}_1 \cdots \mathbf{r}_n$. An association is optimal if it minimizes the association costs $\sum w_{ij}$. On the condition that $m = n$ and that each track must be assigned to a different region, this represents a linear assignment problem, which can be solved in polynomial time using the Hungarian algorithm [50].

4.5.1 Association Costs

Association costs are calculated based on distance and velocity constraints. More precisely, the cost w_{ij} to associate track T_i with region R_j is calculated as the Euclidean distance between the the estimated position of the former and the point representation the latter, $||\mathbf{t}_i - \mathbf{r}_j||$. This applies unless $||(x_i, z_i) - \mathbf{c}_j|| > t$, in which case $w_{ij} = \infty$. This distance

constraint is motivated by the fact that the velocity of persons is limited. Heights are not used in this context because they are less stable, especially during falls. The current implementation uses a threshold of $t = 5$, which corresponds to 37.5cm, in order to account for the limited stability of $\mathbf{c}_j$ with respect to changes in pose.

$\mathbf{t}_i$ is estimated under the assumption that the velocity remains constant between frames and that there are no measurement errors. In this case $\mathbf{t}_i = 2\mathbf{t}'_i - \mathbf{t}''_i$, with $\mathbf{t}'_i$ and $\mathbf{t}''_i$ denoting the track position in the most recent and second most recent frame, respectively. More sophisticated means for state estimation exist, such as the popular Kalman and particle filters [6]. However, the employed method was found sufficient for fall detection and has the advantage of being computationally simple.

Computing the association costs between all tracks and regions results in a $m \times n$ cost matrix (w_{ij}). On the condition that $m = n$ this matrix is square and the Hungarian algorithm can be applied directly. However, in general $m \neq n$ because persons may enter or leave the view at any time, or due to occlusions. This problem is solved by introducing dummy tracks and regions with zero association costs.

4.5.2 Association of Tracks and Regions

The Hungarian algorithm yields pairs $(i, j)_k$ so that $\sum_k w_{ij}$ is minimized. It does not assign multiple tracks to the same region (and vice versa), which is required if multiple person regions become merged due to proximity. This section outlines a simple algorithm to overcome this limitation.

The first step of this algorithm is to create two Boolean $m \times n$ matrices (o_{ij}) and (a_{ij}), with o_{ij} encoding whether the estimated track position (x_i, z_i) lies within the bounding box of R_j. Similarly, (a_{ij}) encodes whether track i and region j are part of the found optimal association. (a_{ij}) is obtained by initializing it with zeros and subsequently setting $a_{ij} = 1$ for every $(i, j)_k$ for which $i \leq m$, $j \leq n$, and $w_{ij} \neq \infty$.

On the condition that no rows of the matrix (a_{ij}) are zero vectors, it represents the sought optimal association between tracks and regions. In this case, the track positions are updated, $\mathbf{t}''_i = \mathbf{t}'_i$, $\mathbf{t}'_i = \mathbf{r}_j$, and new tracks are generated for all regions not assigned to any track (indicated by columns that are zero vectors).

If the ith row of (a_{ij}) is a zero vector, no association was found for T_i. There are two possible reasons for this circumstance; the corresponding region was either not detectable or merged with other regions. (o_{ij}) is a hint for the present reason; T_i lies within the bounding box of region j if $o_{ij} = 1$. In this case, it is assumed that T_i and R_j correspond and that the reason for $a_{ij} = 0$ is that R_j was already assigned to another track k. Consequently, R_j is split between T_i and T_k (and in general all other tracks assigned this way) via nearest neighbor clustering. Subsequently, the positions of T_i and T_k are set to the center of mass of the resulting regions.

If, on the other hand, there is no such component j, the position of T_i is updated under the assumption that the velocity remains constant and a counter is incremented that encodes for how many frames T_i has not been associated with any region. If this counter exceeds a threshold (ten seconds in the current implementation), T_i is discarded.

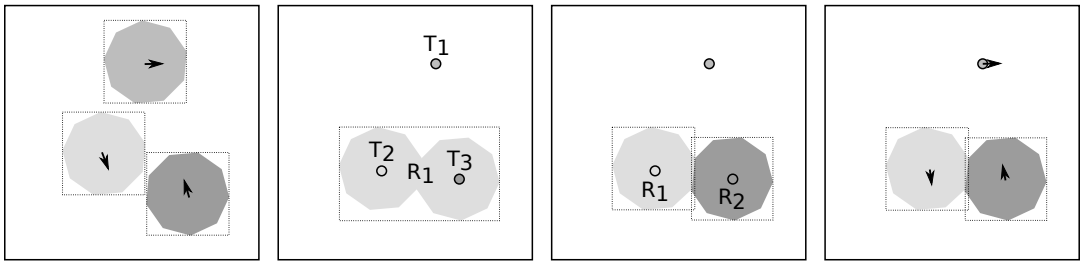


Figure 4.6: Correction for undersegmentation. Image (a) illustrates the situation in frame $t-1$. There are three tracked regions, the arrows represent their center of mass and estimated velocity. In frame t one region is no longer detected and the two other regions are merged (b). $a_{11} = 0$ and $o_{11} = 0$, hence track T_1 is not associated to any region. On the other hand, $a_{31} = 0$ and $o_{31} = 1$ (T_3 is not associated to region R_1 but lies within its bounding box). Consequently, R_1 is split among T_2 (to which it was assigned to originally) and T_3 via nearest neighbor clustering (c). Finally, the positions of T_2 and T_3 are set to the center of mass of the corresponding split regions (d). The position of T_1 is updated under the constant velocity assumption.

4.5.3 Tracking-Based Classification

As discussed before, some features used for classifying candidate regions are too restrictive with respect to fallen persons. Therefore, these features are used only for classifying regions that correspond to new tracks. The rationale behind this approach is that persons are expected to be upright when they enter the view. This is implemented as follows. First, only feature (a) is evaluated, and all candidate regions for which $S(a) = 0$ are discarded. The remaining regions are input to the tracking stage, and the other features are tested only for regions for which $k_i \leq K$, with k_i denoting the number of frames the corresponding tracks have been known. If a region is not classified as representing a person on this basis, both the region and the associated track are discarded. Consequently, objects must be classified as persons for K consecutive frames before they are regarded as such. Only tracks with $k_i > K$ are considered, which implies a delay of K frames. Small values for K were found to be sufficient (e.g. corresponding to one second).

4.5.4 Results and Discussion

In order to evaluate the tracking performance in the context of fall detection, the person detection and tracking algorithms discussed in this chapter were tested on the fall database. This database comprises 144 sequences with durations between 15 and 40 seconds. In each sequence a single person performs an activity of daily living or a simulated fall. No tracking errors (e.g. losing track of the person) were observed.

The performance in presence of multiple persons was assayed on the two test sequences introduced in Section 4.4.5. All persons were tracked correctly even though some were not

detected over several frames due to occlusions. This confirms that the tracking algorithm can cope with temporary missed detections. Merged person regions were detected and corrected successfully, as illustrated in Figures 4.7 and 4.8.

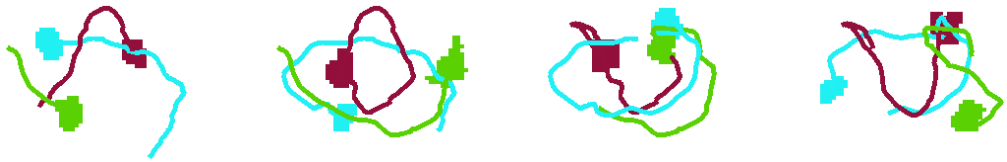
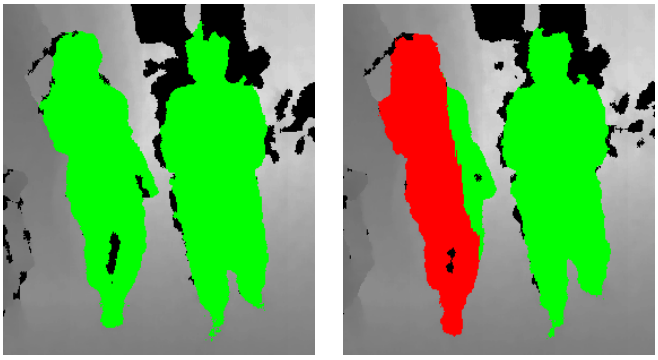


Figure 4.7: Illustration of the tracking performance in presence of three persons, in five second intervals. All persons were tracked correctly over time, even if their occupied regions overlapped in plan-view space (third illustration). The smooth trajectories highlight the stability of the center of mass as a region representation. For clarity, only trajectories corresponding to recent observations are shown.



(a) without correction (b) with correction

Figure 4.8: Visualization of two tracked persons without (a) and with (b) the proposed method for correcting for undersegmentation due to proximity.

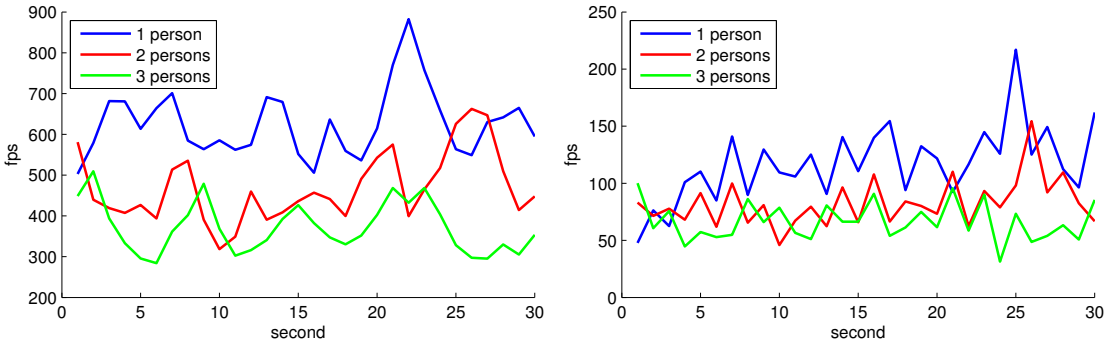
In order to ensure a high specificity of the fall detection system, the tracking algorithm must be able to cope with situations in which a person moves a chair while sitting on it before standing up and moving away. This situation can cause two candidate regions, one for the person and one for the moved chair. It is important that the person is tracked correctly in order to rule out false alarms. This is the main reason why object height contributes to association costs. Doing so ensures reliable tracking in such situations. This was verified using a test sequence in which a person performs the aforementioned action 40 times with four different chairs. No tracking errors occurred.

In summary, these results confirm the applicability of the proposed person detection and tracking algorithms for fall detection. However, there is a possibility that persons that fall out of their beds are not detected. This is due to the height feature used classifying candidate regions, which causes regions with heights below 83cm to be discarded. If a

person rolls out of the bed without rising up, this height criterion might be dissatisfied. On the other hand, sitting up on the bed is sufficient for a correct detection, and if the person falls afterwards (e.g. while attempting to stand up) he or she is tracked correctly.

The tracking algorithm can be improved by employing an advanced motion model (e.g. a particle filter) or multiple state hypotheses [99]. Furthermore, the simple clustering method can cause temporary classification errors (Figure 4.8 (b)) which, however, were found not to impact the fall detection performance. Unfortunately, the use of more complex methods is aggravated by the limited processing power of the target system.

Figure 4.9 illustrates the combined speed of person detection and tracking, as measured using three test sequences depicting one, two, and three persons, respectively. The test methods and systems were identical to those presented in Section 3.4.3. The speed on the test sequences decreased considerably with an increasing number of visible persons. This is mainly attributed to the increasing number of foreground pixels and hence slower plan-view computation time. Nevertheless, the framerate remained fast enough for real-time processing even on low-end hardware; the average framerates on the PandaBoard ES were 117fps, 83fps, and 66fps, respectively.



(a) Intel Core i7 2600 (b) PandaBoard ES

Figure 4.9: Speed of person detection and tracking on the test system with the Intel Core i7 CPU (a) and on the PandaBoard ES (b).

CHAPTER 5

Fall Detection

The proposed system detects falls by means of height map analysis. A finite state machine is used to model the state of each tracked person over time; state transitions are governed by height map statistics. Falls and other events are detected based on state transitions as well as temporal analysis. The system is able to estimate fall confidence scores, which allow caretakers to balance the trade-off between sensitivity and specificity according to their needs. Regions in which reliable fall detection is possible are detected automatically, based on the sensor position and scene geometry. This aids in sensor placement and improves the specificity. For flexibility, detected falls and other events are processed and relayed by so-called event handlers, which interface with the outside world.

5.1 State Detection

The proposed system classifies persons as being in one of the following states.

- Active: the person is standing, walking, or sitting on a sitting accommodation.
- Fallen: the person is lying or sitting on the floor.
- Resting: the person is lying on a bed or couch (or other objects).
- Unknown: the person has not been detected for several frames.

This classification is based on height map analysis because person heights are naturally well-suited for this task (active persons clearly differ from fallen persons in terms of height), as verified in the pertinent literature [8, 29, 69, 74]. Furthermore, height maps are robust with respect to partial occlusions and can be processed quickly. On the other hand, occupancy maps are susceptible to partial occlusions, hence features based on occupancy are not used.

5.1.1 Discrimination Between Active and Inactive Persons

Figure 5.1 illustrates 250 90th percentiles of height map regions representing active, fallen, and resting persons, obtained from the fall database and own test sequences. It is visible that this feature is powerful for classifying persons as active or inactive (fallen or resting). In fact, 17 features (percentiles as well as different measures of central tendency and dispersion) were evaluated in this regard and the 90th percentile was found the feature of choice in terms of discriminability. Based on this data, the classification threshold was defined as $\eta = 775$, which maximizes the margin. This nonparametric approach was employed because the two subsets are not distributed normally, unlike in [74].

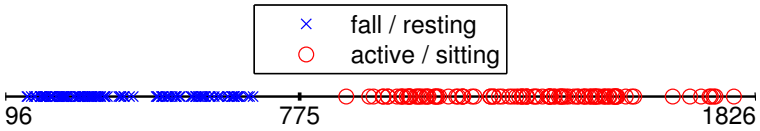


Figure 5.1: 90th percentiles of height map regions representing persons.

The 90th percentile is the only feature used for differentiating between active an inactive persons because it is powerful and robust with respect to partial occlusions and misdetections of limbs, and because features with similar characteristics were found to be highly correlated. Furthermore, it is already computed during person detection. Features representing velocities are not used because of their limitations, particularly the fact that they are too restrictive for detecting slow falls. Another type of feature employed in the existing literature is the orientation of the main axis [68, 69]. While this feature is reportedly well-suited for classification, it is not used for state detection because it is expected to be of limited robustness with respect to falls that end in sitting positions.

The main motivation for using only a single feature for differentiating between active an inactive persons is efficiency, which is important considering the limited computational capabilities of the target system. The fraction of time during which a person falls or is lying on the floor is very small in general. Consequently, the efficiency can be improved dramatically by performing fall analysis only if an actual incident is likely. This is accomplished by testing for whether $\eta < 775$, which is possible in constant time, and proceeding only if this is the case for several frames.

For this purpose, the system assigns one instance of the finite state machine visualized in Figure 5.2 to each tracked person, hence fall detection in presence of multiple persons is supported. State transitions depend on the person state, which is computed in each frame. For example, the state changes to Fallen if the person is classified as such for n consecutive frames. Transitions to the other states are defined analogously. $n = 2$ seconds proved to be a valid compromise in terms of transition delay and robustness.

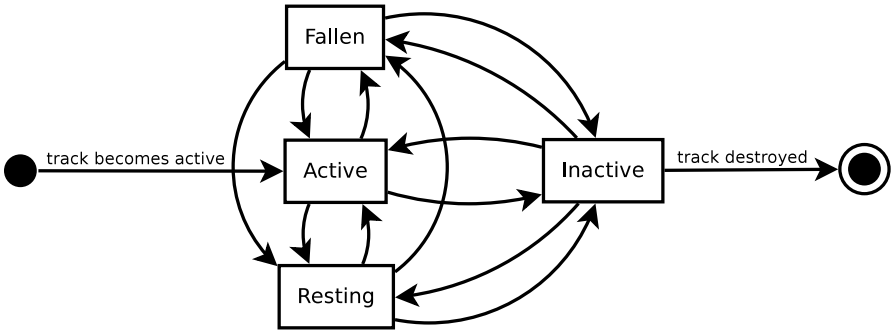


Figure 5.2: The finite state machine used to model the state of a person.

5.1.2 Discrimination Between Fallen and Resting Persons

In order to achieve a high specificity, fall detection systems must be able to differentiate between persons that have fallen and those that are resting on beds or couches, for example. Differentiation based on velocities is not reliable in practice because falls can happen slowly and, conversely, people might sit or lie down quickly [69, 74, 46]. Features derived from plan-view maps are considered impracticable, given the variety of sitting accommodations. Particularly, the 90th percentile of heights is not suitable for classifying between fallen and resting persons because observed heights of fall victims that end up sitting on the floor are similar to those of resting persons, as visible in Figure 5.1.

In consequence, the proposed system differentiates between fallen and resting persons based on an explicit localization of sitting accommodations during startup. More precisely, objects (or parts thereof) are detected if they have a height between 30cm and 85cm and occupy an area of at least 0.75m^2 (the area of an object with a size of 150cm times 50cm). These criteria are general enough to apply to most sitting accommodations that are large enough to lie on. No further classification is performed because doing so would increase the complexity and arguably decrease the sensitivity in this regard. Instead, the proposed method detects not only sitting accommodations but also other furniture that matches these criteria, such as desks. This is not considered a limitation because such objects cannot cause classification errors for reasons described shortly. On the other hand, the method is fast and was found robust in practice.

Given the simple criteria, sitting accommodations can be detected reliably and quickly using height maps. For this purpose a height map of the scene is computed during system initialization. This is accomplished as stated in Algorithm 2, but without restricting to foreground pixels. Morphological closing [84] with a 3×3 circular structuring element is applied to the result. Closing is advantageous over linear filtering in this context because it better corrects for local minima that originate from the limited sensor resolution. On this basis, sitting accommodations can be localized by thresholding the height map. Subsequently, objects that are too small to lie on are removed via connected component analysis and area-based classification. This results in a binary plan-view map that encodes the locations of furniture, as illustrated in Figure 5.3.

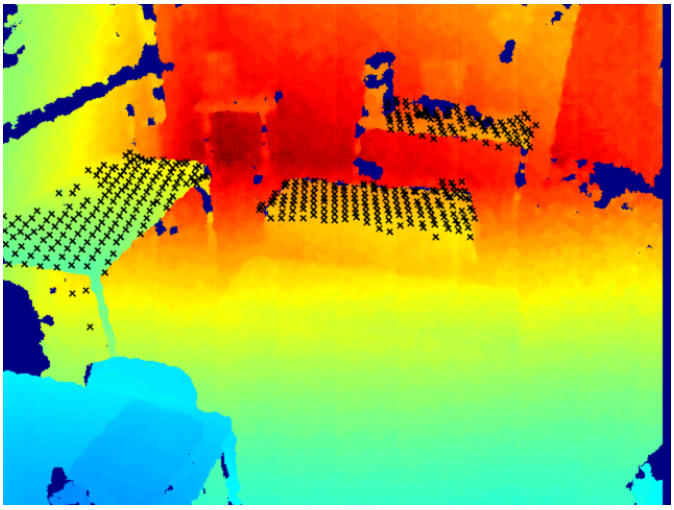


Figure 5.3: Furniture detection via height map analysis. The black crosses represent plan-view cells classified as being part of an object. The minimum object height was set to 20cm for generating this figure because of the low height of the mattress (actual beds are expected to be higher, hence the default threshold of 30cm).

Once the location and extent of sitting accommodations is known, robust discrimination between fallen and resting persons is possible based on area overlap in plan-view space. More precisely, during state detection the percentage of overlap between the area occupied by the tracked person and localized furniture is calculated. Only overlapping regions in which the observed height of the person is larger than the corresponding object height are considered. This ensures that persons are only classified as resting if they lie on top of objects, not underneath them (which may happen with desks, for example). If the overlap percentage is below 75% the person is classified as fallen, otherwise as resting. Using a threshold of 75% as apposed to 100% accounts for imprecise object detection while being restrictive enough to ensure that falls are detected as such.

5.2 Fall Detection

Falls and other events are initiated by state transitions, as summarized in Table 5.1. More precisely, the fall detection system distinguishes between Fall, LikelyFall, and Recovery events. A Fall event signals that a fall was observed, while a LikelyFall event indicates a situation in which a fall is likely even though it could not be observed explicitly due to occlusions, for example. State transitions from Fallen to Active trigger a Recovery event, which signals that a person that had suffered from a fall was able to recover. Transitions from Inactive to Fallen cause a Fall event only if the second last state differs from Fallen in order to suppress multiple Fall events that represent the same incidence.

Table 5.1: Observed state transitions and effects (rows: old state; columns: new state). Events written inside brackets are triggered only under certain conditions.

	Active	Fallen	Resting	Unknown
Active		(Fall)		(LikelyFall)
Fallen	Recovery			
Resting		Fall		
Unknown		(Fall)		

5.2.1 Detection of Falls

State transitions from Active to Fallen do not directly cause a Fall events because the governing criterion was chosen general enough to ensure a high sensitivity (Figure 5.1) at the expense of specificity. Therefore, Fall events are triggered only after verification by means of temporal analysis, which is possible due to the robust person detection and tracking algorithms. For this purpose, the system maintains a list $\mathbf{p} = (p_1 \cdots p_{2n})$ of the $2n$ most recently observed 90th height percentiles for every tracked person, with n denoting the state transition delay introduced in Section 5.1.1. $\mathbf{p}$ encodes the temporal height progression of a person, from which discriminative features can be derived.

One such feature is the temporal height change. This feature is well-suited for fall detection because person heights decrease significantly in the course of falls. It is computed by first smoothing $\mathbf{p}$ with a median filter for noise reduction. If a state transition from Active to Fallen occurs, the most recent occurrence of $p_i \geq \eta$ happened n frames ago and indicates the frame during which the (possible) fall was in progress. Consequently, the height of the person before and after the fall is estimated as $h_{max} = \max(\{p_1, \ldots, p_{n-1}\})$ and $h_{min} = \min(\{p_{n+1}, \ldots, p_{2n}\})$, respectively. The temporal height change is calculated as the difference between the person height before and after the fall, $\Delta h = h_{max} - h_{min}$.

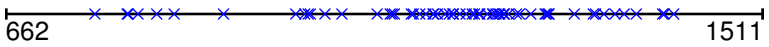


Figure 5.4: Observed temporal height changes due to falls, in millimeters.

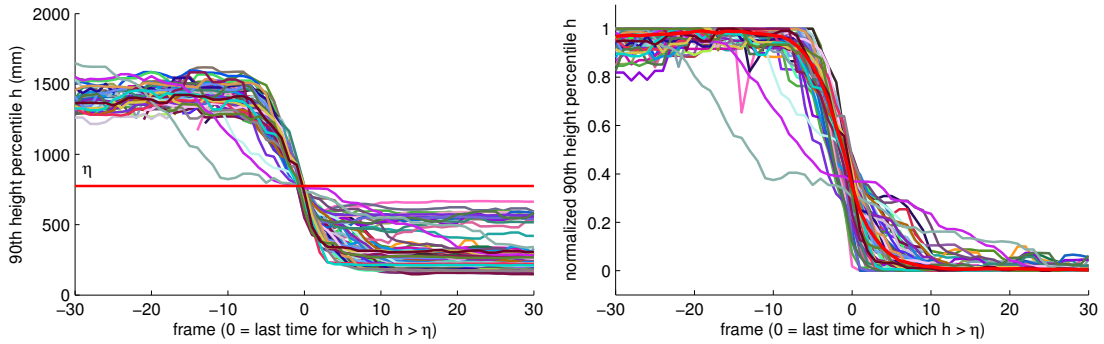
Figure 5.4 shows the distribution of this feature among 60 falls, as obtained from the fall database. The data were recorded at a rate of 15 frames per second and with $n = 30$. The smallest recorded value was 762mm, measured during an fall incident that ended in a kneeing position. Falls ending in lying poses caused significantly larger values ($\mu = 1152$mm, $\sigma = 157$mm). On this basis, the minimum temporal height change due to falls was defined as $\gamma = 600$mm. This is a conservative choice that is expected to guarantee a high sensitivity independently of person size and ending pose. Moreover, this criterion is robust with respect to fall speed as long as the fall duration in frames does not exceed $2n$. The current implementation uses $n = 2$ seconds, that is falls are expected not to last longer than four seconds. This assumption is comparatively general, for example [100] assumes a maximum duration of only two seconds.

The temporal height change is the only criterion that determines whether a Fall event is triggered. Instead of utilizing more criteria of this kind, the system provides fall confidence scores. This approach, which to the knowledge of the author has not been followed in this context so far, is considered superior because it enables caretakers to balance the trade-off between system sensitivity and specificity according to their needs.

Two factors contribute to these confidences, the first one is the observed temporal height change Δh, which is normalized according to γ and μ (5.1). Consequently, $c_{\Delta h} \in [0.2, 1]$ represents the likelihood that a fall occurred based on the temporal height change, optimized so that falls from the test database achieve $c_{\Delta h} = 1$ on average.

$$c_{\Delta h} = \min\left(1, 0.2 + 0.8\frac{\Delta h - \gamma}{\mu - \gamma}\right) \tag{5.1}$$

The second factor is obtained as follows. Figure 5.5 (a) illustrates the median-filtered height progressions of the 60 aforementioned falls. It is visible that these progressions are similar during falls (at frame indices around 0), except for slow falls. On the other hand, the progressions vary before and afterwards, depending on person size and pose. These variations can be corrected via normalization according to h_{min} and h_{max}, after which $p_i \in [0, 1]$. Figure 5.5 (b) shows the resulting normalized progressions $\mathbf{q} = (q_1 \cdots q_{2n})$ along with their average $\mathbf{a} = (a_1 \cdots a_{2n})$, which was calculated as the trimmed mean of the individual progressions.



(a) height progressions

(b) normalized height progressions

Figure 5.5: Height progressions of falls before (a) and after (b) normalization, centered at frame n (the time of the last incident of $p_i \geq \eta$). The average normalized height progression is shown in red in Figure (b).

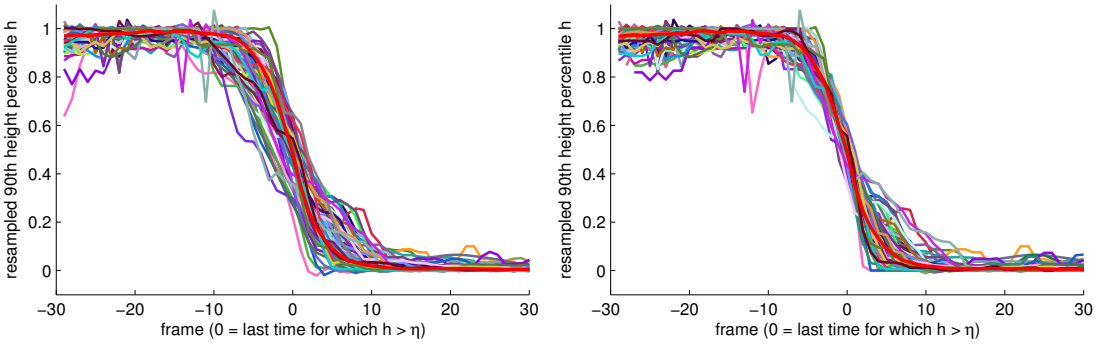
The influence of fall speed is accounted for by resampling $\mathbf{q}$ accordingly. For this purpose, the frames that are expected to represent the begin and end of the fall are estimated by searching for the last and first occurrences of $q_i > 0.95$ and $q_i < 0.05$, respectively. Subsequently, $\mathbf{q}$ is resampled so that the indices of these occurrences match those of $\mathbf{a}$. Aliasing effects are suppressed via lowpass filtering. Figure 5.6 (a) illustrates the resulting progressions $\mathbf{r} = (r_1 \cdots r_{2n})$ (missing values are interpolated). It is visible that progressions of slow falls no longer differ considerably from the average.

Alignment errors are corrected via template matching. For this purpose, $\mathbf{q}$ and $\mathbf{r}$ are matched to $\mathbf{u} = (a_{n-n/3} \cdots a_{n+n/3})$, that is regarding only central indices, which are expected to represent the actual fall event. The sought displacements q' and r' minimize the sum of absolute differences [88], $\arg\min_{q'} \sum_k |q_{k+q'} - u_k|$ and $\arg\min_{r'} \sum_k |r_{k+r'} - u_k|$. Figure 5.6 shows the height progressions after resampling and alignment.

On this basis, the second factor $c_{\Delta t}$ is calculated as stated in Equation 5.2, that is using the sums of absolute differences of both the normalized and resampled progressions. This increases the stability of the algorithm with respect to resampling errors. The scaling factor 1.68 results in an average score of $c_{\Delta t} = 1$ for falls from the test database.

$$c_{\Delta t} = \min\left(1.68 \Big/ \left(1 + \min\left(\sum_k |q_{k+q'} - u_k|, \sum_k |r_{k+r'} - u_k|\right)\right), 1\right) \tag{5.2}$$

The fall confidence is calculated as the weighted mean $c = 2c_{\Delta h}/3 + c_{\Delta t}/3$. $c_{\Delta h}$ has a greater impact on the confidence because it is considered a more robust statistic.



(a) resampled height progressions

(b) final height progressions

Figure 5.6: Height progressions after resampling (a) and alignment (b).

5.2.2 Detection of Likely Falls

Fall detection systems must cope with the fact that person detection might fail in order to be reliable in practice. In the context of Kinect-based fall detection there are two major causes for detection errors that can lead to unrecognized falls, namely occlusions and exposure to direct sunlight. The proposed system thus aims to detect such incidences, which is accomplished similarly to the detection of visible falls. More precisely, each time a state transition from Active to Inactive occurs, the temporal height change is estimated as introduced in Section 5.2.1. In order to obtain a proper threshold γ, the aforementioned 80 test falls were rerecorded and an occluding object with a height of 60cm was simulated by increasing the motion detector threshold accordingly. Figure 5.7 shows the resulting height progressions. The minimum and average temporal height changes were 380mm and 616mm, respectively. In consequence, LikelyFall events are triggered if Δh exceeds $\gamma = 300$mm.

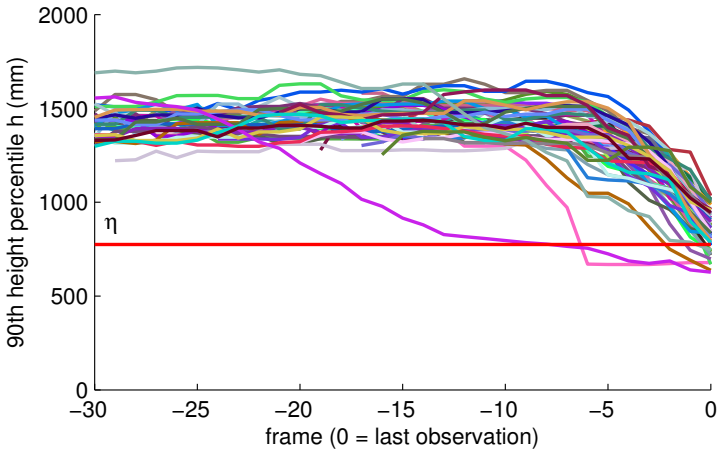


Figure 5.7: Height progressions of occluded falls.

The system is able to compute robust confidence scores for LikelyFall events by detecting regions that are occluded, exposed to direct sunlight, or not visible to the sensor for other reasons. This is accomplished in plan-view space, by maintaining a list of all cells $\mathbf{c}_i$ occupied by each tracked person during the last half-second. If a LikelyFall is triggered, an occupancy map $\mathcal{O}$ is calculated without restricting to foreground pixels and the confidence amounts to the fraction of recently occupied cells for which $\mathcal{O}(\mathbf{c}_i) = 0$.

To the best knowledge of the author, only Rougier et al. [74] aim to detect falls that are not entirely visible as well. However, they use velocity for this purpose, which, as they mention, can lead to false alarms and may fail to detect slow falls.

5.3 View Frustum Analysis

The proposed system is able to automatically determine scene regions in which reliable person detection is possible. This further increases the specificity and enables the system to provide feedback that aids in sensor placement. Furthermore, it allows the system to compute the optimal region of interest without manual intervention (unless desired), which simplifies the system setup. This functionality is achieved by analyzing the viewing frustum of the sensor, that is the volume that encompasses those parts of the scene that are visible to the camera. Viewing frustums are commonly used in rendering engines for culling objects that are outside the camera view [7].

The viewing volume of a pinhole camera corresponds to a frustum of a pyramid, with the camera being located at the apex and the base being parallel to the image plane (this follows directly from central projection, Section 4.2.1). The frustum originates from two planes that intersect this pyramid parallel to the base. These planes are called near and far plane and represent the minimum and maximum distance of visible objects from the camera. Suitable distances for the Kinect sensor depend on the required data quality, the current implementation uses $z_{min} = 0.5$m and $z_{max} = 7$m. The extent of the pyramid

depends on the field of view of the camera, which is 57.5 times 45 degrees according to the datasheet (Page 11). Given a focal length of 585.6px [82], the precise field of view is

$$585.6 = \frac{r_{x,y}}{2\tan(f_{x,y}/2)} \iff f_x = 57.31^\circ \ , \ f_y = 44.57^\circ, \tag{5.3}$$

with $r_{x,y}$ denoting the depth map dimensions (640 by 480 pixels) [88].

If ϕ_x and ϕ_y are the horizontal and vertical field of views in radians and the plane distance from the sensor is z, then from simple geometry it follows that the width of the plane is $w_z = 2z\tan(\phi_x/2)$ and the height is $h_z = 2z\tan(\phi_y/2)$. The extents of the near and far plane are thus $w_{z_{min}} \approx 55$cm times $h_{z_{min}} \approx 41$cm and $w_{z_{max}} \approx 765$cm times $h_{z_{max}} \approx 574$cm, respectively. Subsequently the corner points of the near and far planes can be calculated and transformed to world coordinates. The resulting viewing frustum in world coordinates is shown in Figure 5.8.

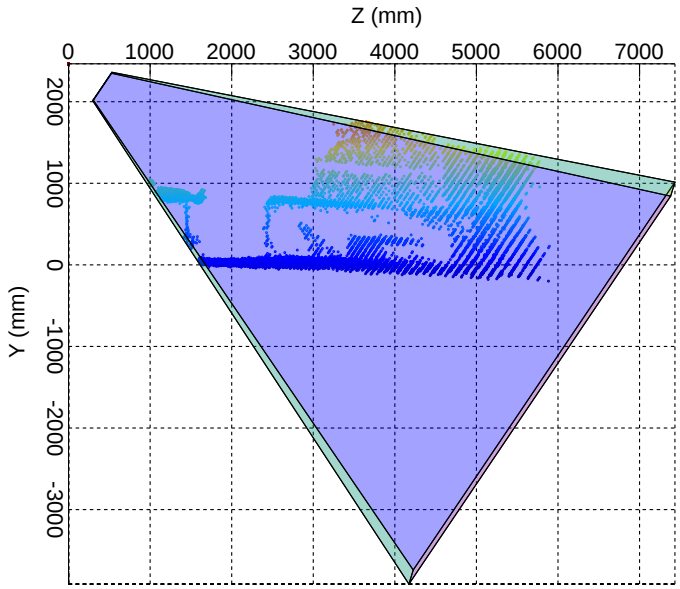


Figure 5.8: The viewing frustum of the Kinect sensor and the point cloud representation of a scene, in world coordinates. It is visible that the sensor is tilted downwards and rotated slightly along $\mathbf{e}_3$.

In Figure 5.8 the floor is not visible for $Z < 1.6$m. Conversely, upright persons are not fully visible for $Z > 4$m. For reliable state detection persons must be visible if they are lying and the largest visible Y-coordinate must be at least $\eta+\varepsilon$ (the height threshold used for detecting falls plus some value to increase the robustness, the current implementation uses $\varepsilon = 30$cm). These criteria are the basis for an automatic determination of the region of interest. Let (x_1, y_1, z_1) denote the coordinates of a corner point of the near plane and (x_2, y_2, z_2) the coordinates of the corresponding point of the far plane. The equation of the line that connects these points is $(z_2 - z_1)(y - y_1) = (y_2 - y_1)(z - z_1)$ and thus $z = (z_2 - z_1)(y - y_1)/(y_2 - y_1) + z_1$. There are four such lines, which correspond to the

edges of the frustum, and the range $[Z_1, Z_2]$ for which reliable fall detection is possible is obtained by evaluating the corresponding equations for $y = 0$ respectively $y = \eta + \varepsilon$. On this basis the region of interest is defined as $(X_{min}, X_{max}, \max(Z_1, Z_{min}), \min(Z_2, Z_{max}))$, with the variables denoting the smallest and largest observed world coordinates. This region of interest is optimal in the sense that it is the smallest rectangular region that encompasses all visible areas in which reliable fall detection is possible, and thus is used by default for computing plan-view maps (Section 4.4.1).

However, this region is not restrictive enough because the aforementioned criteria depend on both X and Z. An alternative would be to regard only those pixels during person detection for which both $(X, 0, Z)$ and $(X, \eta + \varepsilon, Z)$ lie inside the viewing frustum, that is to employ frustum culling. This approach, illustrated in Figure 5.9, is precise but has high computational requirements, which is prohibitive in practice. To this end, the proposed system operates in plan-view space. More precisely, during system initialization the above test is conducted for each plan-view cell (with known (X, Z) coordinates) and pixels are classified on this basis. This results in a binary plan-view map that allows for fast filtering of plan-view pixels for which reliable fall detection is possible via masking.

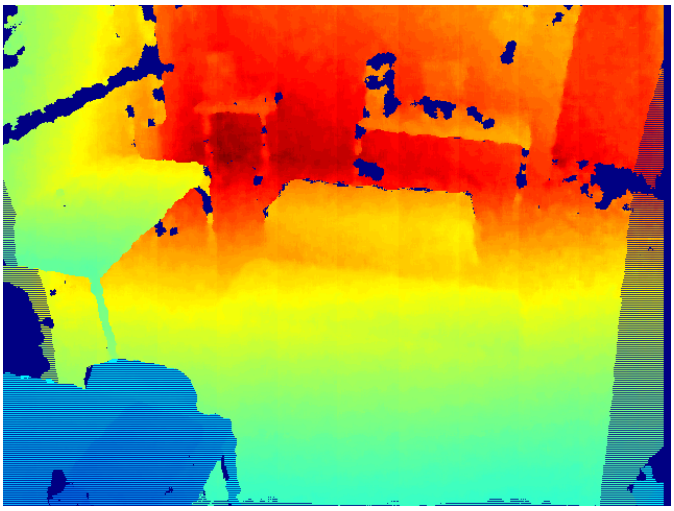


Figure 5.9: View frustum culling for pixel classification. Dashed regions mark pixels for which $(X, 0, Z)$ or $(X, \eta + \varepsilon, Z)$ are not visible to the sensor.

5.4 Event Handling

The system communicates via event handlers, which process and relay system information to different receivers. This increases its flexibility because support for new platforms can be added independently of the core system. Event handlers can be added and removed dynamically, and multiple event handlers can be registered simultaneously. This allows for flexible event reporting, particularly in combination with fall confidence scores. For

example, one event handler could relay all events to an e-mail address, while another event handler could additionally relay fall events with high confidences to a mobile phone, in order to ensure that falls are noticed quickly.

At the time of writing, the proposed fall detection system is being evaluated as part of the FEARLESS project. For this purpose, an event handler was implemented that visualizes and relays fall events and status notifications to the FEARLESS web platform. Figure 5.10 shows visualizations of a Fall and a Recovery event, as generated by the event handler. These visualizations convey enough information to enable caretakers to quickly assess the situation, while being abstract enough to preserve the privacy of subjects [74].

(a) Fall event (b) Recovery event

Figure 5.10: Visualizations of Fall and Recovery events.

5.5 Results and Discussion

The fall detection performance under experimental conditions was evaluated on three test databases. The evaluation was carried out by examining the system output for missed falls as well as false alarms. The remainder of this section presents and discusses the results. The default configuration was used for all tests, unless stated otherwise.

5.5.1 Detection of Visible Falls

The detection performance of visible falls was tested using all sequences from the fall database. This dataset consists of 144 sequences in which four persons perform different activities of daily living (e.g. sitting, lying) as well as simulated falls. To the knowledge of the author, this database is the only extensive and publicly available fall database recorded with a Kinect sensor. For testing purposes, the parameter that represents the minimum height of sitting accommodations was reduced to 20cm in order to account for the low height of the mattress used for simulating a bed (Figure 5.3).

Table 5.2 outlines the test results. The system was able to detect all falls that happened while standing or walking correctly; the reported fall confidence was 0.91 on

average. Two falls originating from a sitting position were not detected. These falls ended in a kneeing position, thus the observed temporal height change was too small (480mm and 390mm, respectively). All eight falls out of the bed were detected correctly. This amounts to an overall system sensitivity of 97.5%. Only a single false alarm was observed, which occurred because a resting person was not classified as such. This happened because the legs of the person were located outside the mattress, which caused the overlap ratio to drop below the threshold. This is expected not to be an issue in practice because beds provide enough space for persons to lie on.

Table 5.2: Test results on the test database introduced in [69].

sequence type	# sequences	# errors	average fall confidence
fall from standing or walking	64	0	0.91
fall from sitting	8	2	0.40
fall from resting	8	0	–
activity of daily living	64	1	–

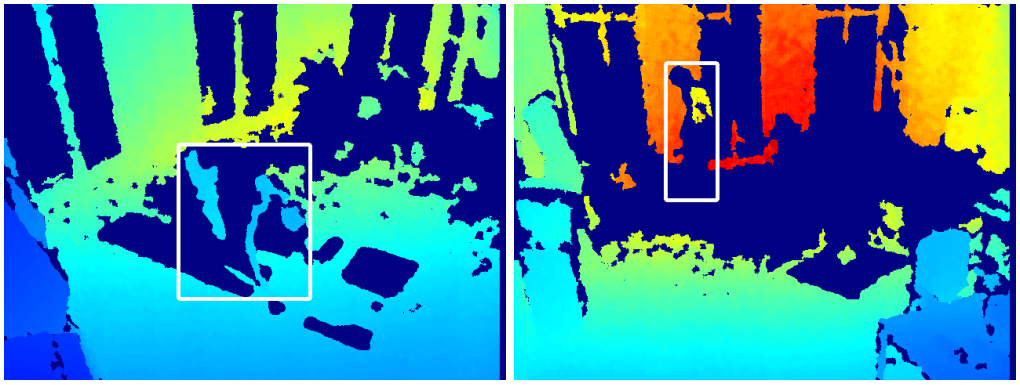
The results indicate a high sensitivity and specificity as well as robust confidence scores for falls that originate from an upright position. Two falls that both originated and ended in a sitting position were not detected. However, such falls are arguably unlikely in practice. It should be noted that, while all falls from the bed were detected, such falls may not be reliably detectable in practice. This is attributed to the criteria used for classifying new tracks, as discussed in Section 4.5.4. If falls that occur because the victim rolls out of the bed without ever sitting up are to expect, this opposes a limitation of the proposed system that could be addressed by employing pressure mats around the bed.

The performance on this test database is similar to the method presented in [69], which reportedly detected all falls correctly and produced only a single false alarm. However, the method was tested on only half the number of sequences. Unfortunately, a meaningful comparison with other existing methods is not possible because these methods were tested on custom sequences.

5.5.2 Detection of Invisible Falls Due to Sunlight

In order to evaluate the performance with respect to falls that are (partially) invisible due to sunlight exposure, an own set of test sequences was compiled. This set depicts backward, forward, and lateral falls as described in [69] in regions subjected to direct sunlight exposure from five different sensor positions, for a total number of 35 falls. Figure 5.11 illustrates two frames from these sequences.

The system detected 32 of 35 falls, which corresponds to a sensitivity of 91%. Four of these falls caused a Fall event, that is person detection did not fail despite the sunlight interferences. The remaining falls were classified as likely falls, with an average confidence of 0.58. Three falls were not detected because the detection failed while the person was still upright due to the significant quality degradation, as visible in Figure 5.11 (b). The



(a) sensor position 1 (b) sensor position 2

Figure 5.11: Illustration of falls performed in regions with direct sunlight exposure. Areas colored dark-blue represent regions in which no measurements are available.

results indicate a high sensitivity of the system with respect to such falls, on the condition that persons are detectable while they are still upright.

To the knowledge of the author, the proposed fall detection system is the only one that is able to explicitly detect such falls, and that can do so robustly by providing reliable and intuitive confidence scores.

5.5.3 Detection of Occluded Falls

The performance with regard to occluded falls was tested on test sequences in which a person performs 35 simulated falls behind a desk that has a height of 75cm. The person ends up partially or completely occluded after each fall (Figure 5.12).

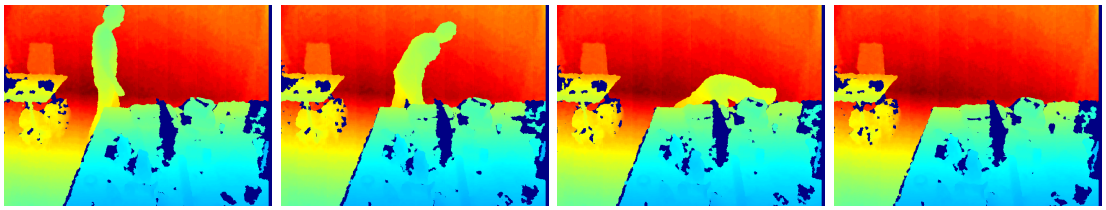


Figure 5.12: Illustration of an occluded fall.

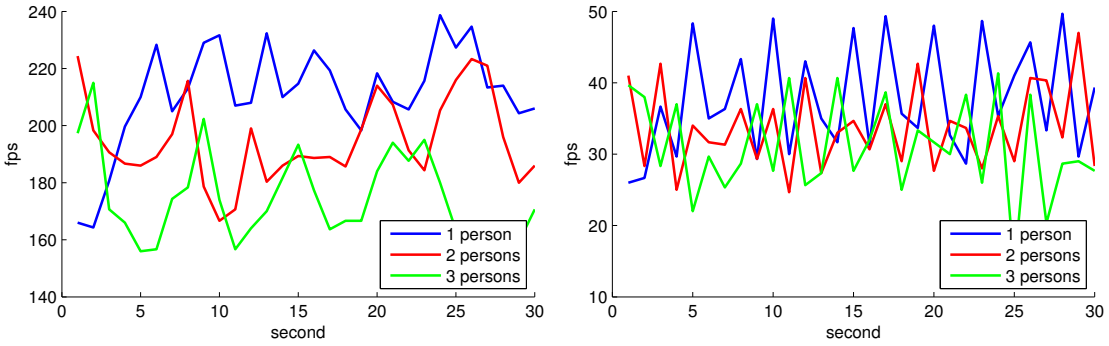
The proposed system was able to detect all occluded falls correctly. Six partially occluded falls caused Fall events, the remaining falls triggered LikelyFall events with an average fall confidence of 0.86. The only existing method that is able to detect occluded falls [74] employs fall velocities, which may fail to detect slow falls.

5.5.4 Performance in Practice

At the time of writing, the proposed system is evaluated under practical conditions in four countries (Austria, Germany, Italy, and Spain). Preliminary results indicate a high specificity in practice; the number of false alarms varies between 0 and 2 per day on average, depending on confidence thresholds.

5.5.5 Speed

Figure 5.13 outlines the overall speed of the proposed fall detection system, that is including motion detection, person detection and tracking, and fall detection. The measurements were obtained using the test methods and systems presented in Section 3.4.3 and using the test sequences introduced in Section 4.4.5. The results confirm that the system is fast enough for real-time operation on low-end hardware, even in presence of multiple persons. The moderate computational requirements allow for multiple instances to be run concurrently on desktop computers, which is beneficial in cases in which fall detection must span multiple rooms.



(a) Intel Core i7 2600

(b) PandaBoard ES

Figure 5.13: Overall speed of the fall detection system on an Intel Core i7 2600 desktop computer (a) and a PandaBoard ES (b), as measured on three test sequences.

CHAPTER 6

Conclusions

Depth data from Kinect sensors allow for reliable and practical fall detection. This is because these sensors are inexpensive, unobtrusive, privacy-preserving, and independent of lighting conditions. The produced data are of high quality and robust with respect to external infrared radiation and common clothing fabrics, as verified by means of a quantitative sensor analysis. However, it has been shown that limbs of fallen persons are not reliably detectable at larger distances and that direct sunlight exposure prevents measurements. These aspects are important to the design of reliable fall detection algorithms. The introduced error model aids in proper a configuration of algorithms.

The proposed fall detection system operates in plan-view space, which allows for robust detection and tracking of persons and enables distinctive features for fall detection. Operating in this space leads to a significant data reduction, which is required to ensure real-time application on inexpensive hardware. Furthermore, this space allows for efficient scene analysis, particularly the detection of sitting accommodations and regions in which no measurements are available. It has been demonstrated that this information can be utilized to increase the fall detection performance. The system is able to estimate the parameters required for computing plan-view coordinates automatically. Moreover, it detects regions in which reliable fall detection is possible, which aids in sensor placement and increases its efficiency. In consequence, the system does not require any configuration, which simplifies its setup and is expected to promote its acceptance in practice.

In contrast to previous works, fall detection builds upon reliable person detection and tracking algorithms, which allow for robust temporal analysis and enable operability in presence of multiple persons. On this basis, falls are detected and verified based on temporal height progressions. The proposed system follows a novel approach by estimating fall confidences, which enable caretakers to balance the trade-off between sensitivity and specificity based on their individual needs. An advantage of the proposed system over existing alternatives is that invisible falls can be detected reliably.

The high fall detection performance of the system under experimental conditions was verified on a comprehensive test database comprising 214 falls and activities of daily

living. At the time of writing, the system is evaluated under practical conditions in four countries; first results indicate a high specificity in practice. In conclusion, the results suggest that reliable and practical fall detection using Kinect sensors is possible. Future work will focus on extending the fall database with data recorded under practical conditions in order further evaluate and optimize the system.

The proposed system is future-proof because it supports all sensors that produce depth maps. The motion detector is the only component that must be modified in this case. A study on how different sensors impact the fall detection performance is planned.

The contributions of this work to the field of person detection and tracking can be applied in contexts other than fall detection. Particularly, this work has presented a new background subtraction algorithm for Kinect depth maps that performs better than the state of the art. Two variants of this algorithm have been discussed, one optimized for fall detection and one for general person or object detection tasks.

Bibliography

[1] M. Alwan, P. Rajendran, S. Kell, D. Mack, S. Dalal, M. Wolfe, and R. Felder, A Smart and Passive Floor-Vibration Based Fall Detector for Elderly, *Proc. Int. Conf. Information & Communication Technologies*, 2006, 1003–1007.

[2] M. Andersen, T Jensen, P Lisouski, A. Mortensen, M. Hansen, T. Gregersen, and P. Ahrendt, *Kinect Depth Sensor Evaluation for Computer Vision Applications*, tech. rep. ECE-TR-6, Aarhus University, 2012.

[3] D. Anderson, J. M. Keller, M. Skubic, X. Chen, and Z. He, Recognizing falls from silhouettes, *Proc. IEEE Conf. Engineering in Medicine and Biology Society*, 2006, 6388–6391.

[4] D. Anderson, R. H. Luke, J. M. Keller, M. Skubic, M. Rantz, and M. Aud, Linguistic summarization of video for fall detection using voxel person and fuzzy logic, *Computer Vision and Image Understanding*, 113 (1), 2009, 80–89.

[5] C. L. Arfken, H. W. Lach, S. J. Birge, and J. P. Miller, The prevalence and correlates of fear of falling in elderly persons living in the community, *American Journal of Public Health*, 84 (4), 1994, 565–570.

[6] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking, *IEEE Trans. Signal Processing*, 50 (2), 2002, 174–188.

[7] U. Assarsson and T. Moller, Optimized view frustum culling algorithms for bounding boxes, *Journal of Graphics Tools*, 5 (1), 2000, 9–22.

[8] E. Auvinet and J. Meunier, Head detection using Kinect camera and its application to fall detection, *Proc. Int. Conf. Information Science, Signal Processing and their Applications*, 2012, 164–169.

[9] F. Bagalà, C. Becker, A. Cappello, L. Chiari, K. Aminian, J. M. Hausdorff, W. Zijlstra, and J. Klenk, Evaluation of accelerometer-based fall detection algorithms on real-world falls, *PLoS ONE*, 7 (5), 2012.

[10] O. Barnich and M. Van Droogenbroeck, ViBe: a universal background subtraction algorithm for video sequences, *IEEE Trans. Image Processing*, 20 (6), 2011, 1709–1724.

[11] K. Berger, K. Ruhl, C. Brümmer, Y. Schröder, A. Scholz, and M. Magnor, Markerless Motion Capture using multiple Color-Depth Sensors, *Proc. Vision, Modeling and Visualization*, 2011, 317–324.

[12] T. Bernhard, A. Chintalapally, and D. Zukowski, *A Comparative Study of Structured Light and Laser Range Finding Sensors*, tech. rep., University of Colorado, 2012.

[13] D. Beymer, Person counting using stereo, *Proc. Workshop on Human Motion*, 2000, 127–133.

[14] A. Bourke, P Van de Ven, M Gamble, R O'Connor, K Murphy, E Bogan, E McQuade, P Finucane, G OLaighin, and J Nelson, Evaluation of waist-mounted tri-axial accelerometer based fall-detection algorithms during scripted and continuous unscripted activities, *Journal of Biomechanics*, 43 (15), 2010, 3051–3057.

[15] T. Bouwmans, F. El Baf, B. Vachon, et al., Background modeling using mixture of gaussians for foreground detection – a survey, *Recent Patents on Computer Science*, 1 (3), 2008, 219–237.

[16] A. Broggi, C. Caraffi, R. Fedriga, and P. Grisleri, Obstacle Detection with Stereo Vision for Off-Road Vehicle Navigation, *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005, 65–65.

[17] S. Brutzer, B. Hoferlin, and G. Heidemann, Evaluation of background subtraction techniques for video surveillance, *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2011, 1937–1944.

[18] M. Camplani and L. Salgado, Adaptive Spatio-Temporal Filter for Low-Cost Camera Depth Maps, *Proc. IEEE Conf. Emerging Signal Processing Applications*, 2012, 33–36.

[19] M. Camplani and L. Salgado, Background foreground segmentation with RGB-D Kinect data: An efficient combination of classifiers, *Journal of Visual Communication and Image Representation*, 2013.

[20] J. Chow, K. Ang, D. Lichti, and W. Teskey, Performance Analysis of a Low-Cost Triangulation-Based 3D Camera: Microsoft Kinect System, *Int. Archives Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIX (B5), 2012, 175–180.

[21] A. Criminisi, P. Perez, and K. Toyama, Region filling and object removal by exemplar-based image inpainting, *IEEE Trans. Image Processing*, 13 (9), 2004, 1200–1212.

[22] R. Cucchiara, A. Prati, and R. Vezzani, A multi-camera vision system for fall detection and alarm generation, *Expert Systems*, 24 (5), 2007, 334–345.

[23] J. Dai, X. Bai, Z. Yang, Z. Shen, and D. Xuan, Mobile phone-based pervasive fall detection, *Personal and Ubiquitous Computing*, 14 (7), 2010, 633–643.

[24] N. Dalal and B. Triggs, Histograms of oriented gradients for human detection, *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2005, 886–893.

[25] G Demiris, Smart homes and ambient assisted living in an aging society, *Methods of Information in Medicine*, 47 (1), 2008, 56–57.

[26] N. Deshpande, E. J. Metter, S. Bandinelli, F. Lauretani, B. G. Windham, and L. Ferrucci, Psychological, physical, and sensory correlates of fear of falling and consequent activity restriction in the elderly: the InCHIANTI study, *American Journal of Physical Medicine and Rehabilitation*, 87 (5), 2008, 354–362.

[27] G Diraco, A Leone, and P Siciliano, An active vision system for fall detection and posture recognition in elderly healthcare, *Design, Automation & Test in Europe Conference & Exhibition*, 2010, 1536–1541.

[28] R. Dubey, B. Ni, and P. Moulin, A Depth Camera Based Fall Recognition System for the Elderly, *Image Analysis and Recognition*, Lecture Notes in Computer Science vol. 7325, 2012, 106–113.

[29] A. Dubois and F. Charpillet, Automatic Fall Detection System with a RGB-D Camera using a Hidden Markov Model, *Proc. Int. Conf. Smart Homes and Health Telematics*, 2013, 259–266.

[30] A. Elgammal, D. Harwood, and L. Davis, Non-parametric Model for Background Subtraction, *Europ. Conf. Computer Vision*, 2000, 751–767.

[31] B. Freedman, S. Alexander, M. Machline, and Y. Arieli, Depth Mapping Using Projected Patterns, *US patent 2010/0118123 A1*, 2010.

[32] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2004.

[33] M. Harville, Fast, integrated person tracking and activity recognition with plan-view templates from a single stereo camera, *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2004, 398–405.

[34] M. Harville, Stereo person tracking with short and long term plan-view appearance models of shape and color, *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, 2005, 522–527.

[35] M. Harville, Stereo person tracking with adaptive plan-view templates of height and occupancy statistics, *Image and Vision Computing*, 22 (2), 2004, 127–142.

[36] L. He, Y. Chao, K. Suzuki, and K. Wu, Fast connected-component labeling, *Pattern Recognition*, 42 (9), 2009, 1977–1987.

[37] F. Hegger, N. Hochgeschwender, G. Kraetzschmar, and P. Ploeger, People Detection in 3D Point Clouds Using Local Surface Normals, *Robot Soccer World Cup 2012*, Lecture Notes in Computer Science vol. 7500, 2013, 154–165.

[38] C. D. Herrera, J. Kannala, and J. Heikkila, Accurate and Practical Calibration of a Depth and Color Camera Pair, *Computer Analysis of Images and Patterns*, Lecture Notes in Computer Science vol. 6855, 2011, 437–445.

[39] W. Hu, T. Tan, L. Wang, and S. Maybank, A survey on visual surveillance of object motion and behaviors, *IEEE Trans. Systems, Man, and Cybernetics*, 34 (3), 2004, 334–352.

[40] M. Humenberger, S. Schraml, C. Sulzbachner, and A. N. Belbachir, Embedded Fall Detection with a Neural Network and Bio-Inspired Stereo Vision, *Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshops*, 2012, 60–67.

[41] S. Ikemura and H. Fujiyoshi, Real-time human detection using relational depth similarity features, *Proc. Asian Conf. Computer Vision*, 2011, 25–38.

[42] R. Jain and H.-H. Nagel, On the Analysis of Accumulative Difference Pictures from Image Sequences of Real World Scenes, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 1 (2), 1979, 206–214.

[43] B. Jansen, F. Temmermans, and R. Deklerck, 3D human pose recognition for home monitoring of elderly, *Proc. IEEE Conf. Engineering in Medicine and Biology Society*, 2007, 4049–4051.

[44] Joint Committee for Guides in Metrology, International vocabulary of metrology – Basic and general concepts and associated terms (VIM), 2008.

[45] P. Kelly, N. E. O'Connor, and A. F. Smeaton, Robust pedestrian detection and tracking in crowded scenes, *Image and Vision Computing*, 27 (10), 2009, 1445–1458.

[46] M. Kepski and B. Kwolek, Fall Detection on Embedded Platform Using Kinect and Wireless Accelerometer, *Proc. Int. Conf. Computers Helping People with Special Needs*, 2012, 407–414.

[47] K. Khoshelham and S. Elberink, Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications, *Sensors*, 12 (2), 2012, 1437–1454.

[48] S.-Y. Kim, M. Kim, and Y.-S. Ho, Removal of Mixed, Lost, and Noisy Depth Pixels in Time-of-flight RGB-D Cameras, *IEEE Trans. Multimedia*, 2012.

[49] T. Kleinberger, M. Becker, E. Ras, and A. Holzinger, Ambient Intelligence in Assisted Living: Enable Elderly People to Handle Future Interfaces, *Universal Access in Human-Computer Interaction*, Lecture Notes in Computer Science vol. 4555, 2007, 103–112.

[50] H. W. Kuhn, The Hungarian method for the assignment problem, *Naval Research Logistics Quarterly*, 2 (1), 1955, 83–97.

[51] R. Labayrade, D. Aubert, and J.-P Tarel, Real time obstacle detection in stereovision on non flat road geometry through v-disparity representation, *Proc. IEEE Intelligent Vehicle Symposium*, 2002, 646–651.

[52] L. Lacassagne, A. Manzanera, and A. Dupret, Motion detection: Fast and robust algorithms for embedded systems, *IEEE Int. Conf. Image Processing*, 2009, 3265–3268.

[53] J. C. Leachtenauer and R. G. Driggers, *Surveillance and Reconnaissance Systems: Modeling and Performance Prediction*, Artech House, 2001.

[54] Y. Li, K. C. Ho, and M. Popescu, A microphone array system for automatic fall detection, *IEEE Trans. Bio-Medical Engineering*, 59 (5), 2012, 1291–1301.

[55] Y. Li, Z. Zeng, M. Popescu, and K. C. Ho, Acoustic fall detection using a circular microphone array, *Proc. IEEE Conf. Engineering in Medicine and Biology Society*, 2010, 2242–2245.

[56] D. Litvak, Y. Zigel, and I. Gannot, Fall detection of elderly through floor vibrations and sound, *Proc. IEEE Conf. Engineering in Medicine and Biology Society*, 2008, 4632–4635.

[57] C. Lord and D. Colvin, Falls In The Elderly: Detection And Assessment, *Proc. IEEE Conf. Engineering in Medicine and Biology Society*, 1991, 1938–1939.

[58] A. Manzanera and J. C. Richefeu, A new motion detection algorithm based on Σ–Δ background estimation, *Pattern Recognition Letters*, 28 (3), 2007, 320–328.

[59] G. Marsaglia, Xorshift RNGs, *Journal of Statistical Software*, 8 (14), 2003, 1–6.

[60] G. Mastorakis and D. Makris, Fall detection system using Kinect's infrared sensor, *Journal of Real-Time Image Processing*, 2012.

[61] N. McFarlane and C. Schofield, Segmentation and tracking of piglets in images, *Machine Vision and Applications*, 8 (3), 1995, 187–193.

[62] F. G. Miskelly, Assistive technology in elderly care, *Age and Ageing*, 30 (6), 2001, 455–458.

[63] T. B. Moeslund, A. Hilton, and V. Krüger, A survey of advances in vision-based human motion capture and analysis, *Computer Vision and Image Understanding*, 104 (2), 2006, 90–126.

[64] M. Mubashir, L. Shao, and L. Seed, A survey on fall detection: Principles and approaches, *Neurocomputing*, 100, 2012, 1–9.

[65] K. Niazmand, C. Jehle, L. T. D'Angelo, and T. C. Lueth, A new washable low-cost garment for everyday fall detection, *Proc. IEEE Conf. Engineering in Medicine and Biology Society*, 2010, 6377–6380.

[66] N Noury and A Fleury, Fall detection – principles and methods, *Proc. IEEE Conf. Engineering in Madicine and Biology Society*, 2007, 1663–1666.

[67] N. Noury, P. Rumeau, A. Bourke, G. ÓLaighin, and J. Lundy, A Proposal for the Classification and Evaluation of Fall Detectors, *IRBM*, 29 (6), 2008, 340–349.

[68] A. Nowakowska, Recognition of a vision approach for fall detection using a biologically inspired dynamic stereo vision sensor, MA thesis, Vienna University of Technology, 2011.

[69] R. Planinc and M. Kampel, Robust Fall Detection by Combining 3D Data and Fuzzy Logic, *Computer Vision – ACCV 2012 Workshops*, Lecture Notes in Computer Science vol. 7729, 2013, 121–132.

[70] R. S. Porter, *The Merck Manual of Diagnosis and Therapy*, Wiley, 2011.

[71] C. Pramerdorfer, Evaluation of Kinect Sensors for Fall Detection, *Proc. IASTED Conf. Signal Processing, Pattern Recognition and Applications*, 2013.

[72] A. Prati, I. Mikic, M. Trivedi, and R. Cucchiara, Detecting moving shadows: algorithms and evaluation, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 25 (7), 2003, 918–923.

[73] N. Rafibakhsh, J. Gong, and M. Siddiqui, Analysis of XBOX Kinect Sensor Data for Use on Construction Sites: Depth Accuracy and Sensor Interference Assessment, *Constitution Research Congress*, 2012, 848–857.

[74] C. Rougier, E. Auvinet, J. Rousseau, M. Mignotte, and J. Meunier, Fall Detection from Depth Map Video Sequences, *Proc. Int. Conf. Smart Homes and Health Telematics*, 2011, 121–128.

[75] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, Fall Detection from Human Shape and Motion History Using Video Surveillance, *Proc. Int. Conf. Advanced Information Networking and Applications*, 2007, 875–880.

[76] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, Monocular 3D head tracking to detect falls of elderly people., *Proc. IEEE Conf. Engineering in Medicine and Biology Society*, 2006, 6384–6387.

[77] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, Robust Video Surveillance for Fall Detection Based on Human Shape Deformation, *IEEE Trans. Circuits and Systems for Video Technology*, 21 (5), 2011, 611–622.

[78] L. Z. Rubenstein, Falls in older people: epidemiology, risk factors and strategies for prevention, *Age and Ageing*, 35, 2006, 1137–1141.

[79] R. Muñoz Salinas, A Bayesian plan-view map based approach for multiple-person detection and tracking, *Pattern Recognition*, 41 (12), 2008, 3665–3676.

[80] C. M. Shakarji, Least-squares fitting algorithms of the NIST algorithm testing system, *Journal of Research of the National Institute of Standards and Technology*, 103, 1998, 633–641.

[81] Shireen, Khaled, and Sumaya, Moving Object Detection in Spatial Domain using Background Removal Techniques – State-of-Art, *Recent Patents on Computer Science*, 1, 2008, 32–34.

[82] J. Smisek, M. Jancosek, and T. Pajdla, 3D with Kinect, *Proc. IEEE Conf. Computer Vision Workshops*, 2011, 1154–1160.

[83] L. Smith, *A tutorial on principal components analysis*, tech. rep., Cornell University, 2002.

[84] M. Sonka, V. Hlavac, and R. Boyle, *Image processing, analysis, and machine vision*, Cengage Learning, 2007.

[85] L. Spinello and K. Arras, People detection in RGB-D data, *Proc. Int. Conf. Intelligent Robots and Systems*, 2011, 3838–3843.

[86] C. Stauffer and W. Grimson, Learning patterns of activity using real-time tracking, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22 (8), 2000, 747–757.

[87] E. E. Stone and M. Skubic, Evaluation of an Inexpensive Depth Camera for Passive In-Home Fall Risk Assessment, *Proc. Int. Conf. Pervasive Computing Technologies for Healthcare*, 2011, 71–77.

[88] R. Szeliski, *Computer Vision: Algorithms and Applications*, Springer, 2010.

[89] B. Taylor, C. Kuyatt, and J. Lyons, *Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results*, Diane Publishing, 1994.

[90] A. Telea, An image inpainting technique based on the fast marching method, *Journal of Graphics Tools*, 9 (1), 2004, 23–34.

[91] M. E. Tinetti and C. S. Williams, Falls, injuries due to falls, and the risk of admission to a nursing home, *New England Journal of Medicine*, 337 (18), 1997, 1279–1284.

[92] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, Wallflower: principles and practice of background maintenance, *Proc. IEEE Int. Conf. Computer Vision*, 1999, 255–261.

[93] United Nations, Department of Economic and Social Affairs, Population Division, World Population Prospects: The 2012 Revision, http://esa.un.org/wpp, June 2013.

[94] S. Wang, Lying Pose Recognition for Elderly Fall Detection, *Robotics: Science and Systems*, 7, 2012, 345–352.

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

[96] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, Pfinder: real-time tracking of the human body, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19 (7), 1997, 780–785.

[97] S. Wu, S. Yu, and W. Chen, An attempt to pedestrian detection in depth images, *Proc. Chinese Conf. on Intelligent Visual Surveillance*, 2011, 97–100.

[98] L. Xia, C.-C. Chen, and J. Aggarwal, Human detection using depth information by Kinect, *Proc. IEEE Conf. Computer Vision and Pattern Recognition Workshops*, 2011, 15–22.

[99] A. Yilmaz, O. Javed, and M. Shah, Object Tracking: A Survey, *ACM Computing Surveys*, 38 (4), 2006.

[100] X. Yu, Approaches and principles of fall detection for elderly and patient, *Proc. Int. Conf. e-health Networking Applications and Services*, 2008, 42–47.