NTNU – Trondheim
Norwegian University of
Science and Technology

Fall Detection for the Elderly using Microsoft Kinect

Erlend Kvinge Jørgensen

Master of Science in Cybernetics and Robotics
Submission date: June 2014
Supervisor: Tor Engebret Onshus, ITK

Norwegian University of Science and Technology
Department of Engineering Cybernetics

Problem Description

The overall goal is to investigate the potential of using the Kinect from Microsoft in fall detection for elders. Existing literature mostly concludes that the results seem promising, but the falls analyzed are mostly standard falls starting with a person standing upright fully in Kinect view and ending with a person lying on the ground fully in Kinect view. This is not the case in reality; falls can have very different characteristics both when it comes to how quickly they happen and also start- and end-posture. There will also be occlusions such as chairs and couches along with persons being partially or fully out of view in the process of falling. Important parts are:

- Investigate the robustness of simple parameters suggested in literature when falls are more varying
- Test fall detection in more realistic scenarios
- Investigate tracker weaknesses that are introduced with more realistic scenarios and must be handled if necessary
- A complete fall detection scheme should be suggested, as a basis for further testing

Supervisor: Tor Onshus, ITK, NTNU

Preface

This thesis concludes my five years at the Norwegian University of Science and Technology studying Engineering Cybernetics. The theme and problem description was the result of a discussion with Kristian Stormo and Kristina Hoff Wanderås in which we were looking for ways of fall detection among the elderly.

I would like to thank Anders Kofod-Pedersen for lending me the Kinect. I would also like to thank my supervisor Tor Onshus.

In addition I would like to thank the persons I've been sharing office with these five months; Håkon Søhoel, Håkon Bøe, Sverre Kvamme, Kristian Stormo, Øyvind Ulvin Halvorsen and Simen Fuglaas. Along with many productive discussions, this time has also been joyful and humorous because of them.

Erlend Kvinge Jørgensen,

Trondheim, June 2014

Contents

List of Figures

Abstract

This thesis explores the use of Microsoft Kinect for fall detection of elders. Strengths and weaknesses are analyzed, and typical problems encountered when using the Kinect for fall detection in a home environment are discussed. At first, general information is presented; existing technologies, related work and sensor and software.

Seeing as most of the related work is based on simple properties such as centroid height or the bounding box, many of these simple properties are tested with short and specific fall- and neutral events. This is to shed light on different situations which might alter and give unsatisfactory results for some properties. All simple parameters are then compared, with special attention given to robustness. This is to account for more realistic scenarios in which there will be occlusions and a wide range of different situations.

After the simple parameters have been analyzed, more advanced techniques are suggested, based on the most robust of the simple parameters. This includes creating height- and inactivity maps to be able to separate sitting/lying areas from walking areas. If this is done successfully, simple parameters may be sufficient to detect falls and avoid a large amount of false positives.

A larger test is done where three scenarios are viewed; living room, hallway and bedroom. For each scenario a training period is recorded. A person is walking around, sitting in chairs, lying in bed etc., and the chosen properties and techniques are analyzed based on this training period. Fall detection is then investigated by executing five different falls in each scenario, and then comparing the response from these falls to the normal behaviour reflected by the training period.

Lastly, an algorithm is suggested, trying to maximize the number of detected falls while still minimizing the number of false positives. The algorithm is run throughout the training period to investigate false positives, and it is also run with the fall events after the training period to make sure these are detected. The results are discussed, and it seems that a working fall detection system can be constructed, especially considering the improvements in accuracy with the new Kinect 2.

Sammendrag

Målet med denne oppgaven er å utforske bruk av Kinect-sensoren fra Microsoft til å detektere fall blant eldre. Styrker og svakheter er analysert, og typiske problemer man vil oppleve ved bruk i et realistisk hjemme-scenario er satt fokus på. Først blir basis-informasjon presentert; eksisterende teknologi, tidligere arbeid og sensor og software.

Mesterparten av tidligere arbeid baserer seg på enkle parametre som høyden til sentroiden eller egenskaper ved personens "bounding box". Disse parametrene er testet med korte, spesifikke fall- og nøytrale sekvenser. Sekvensene er valgt for å sette fokus på svakheter ved de enkle parametrene, og for å finne ut hvilke som er mest robuste i forskjellige typer scenarioer. Alle parametrene er sammenlignet, og de mest robuste er funnet.

Etter de enkle parametrene har blitt analysert, blir noen mer avanserte fremgangsmåter foreslått, ved å bruke de mest robuste av de enkle parametrene. Å generere høyde- og inaktivitets-kart basert på normal oppførsel er foreslått. Dette er for å kunne bruke enkle parametre på en robust måte, ved å kunne skille områder hvor man ligger/sitter fra områder hvor man beveger seg. Hvis disse kartene er nøyaktige nok, kan man bruke enkle kriterier, og likevel unngå en stor mengde falske deteksjoner.

En større test har blitt utført, hvor tre scenarioer er gjennomgått; stue, gang og soverom. For hvert av disse scenarioene er en opplæringsperiode tatt opp. I hvert av scenarioene går en person rundt, sitter i stoler, ligger i sengen osv, og basert på denne opplæringsperioden er hver av de mer avanserte fremgangsmåtene diskutert. Falldeteksjons-delen er undersøkt ved å gjennomføre fem forskjellige fall i hvert scenario, og sammenligne denne reponsen med normal oppførsel tatt opp under opplæringsperioden.

Til slutt er en komplett algoritme foreslått, hvor antall falske deteksjoner er forsøkt minimert, og alle fall blir detektert. Algoritmen først blir kjørt mens systemet læres opp for å undersøke antall falske deteksjoner. Deretter blir algoritmen kjørt for hvert fall for å sjekke at disse blir detektert. Resultatene viser at et fungerende falldeteksjons-system virker gjennomførbart, spesielt med tanke på at den nye Kinect 2 fra Microsoft har langt bedre nøyaktighet, noe som trolig vil føre til bedre segmentering.

Conclusion

Being able to detect falls automatically would be a great safety improvement for elders, seeing as falls are a significant source for trauma and injuries. A working fall detection system would result in the possibility of elders staying at home longer, which would result in less resources needed per person. In addition, if a fall occurs, being able to send necessary help quickly will minimize damage.

The Kinect has great potential for use when it comes to short-distance surveillance indoors. The weaknesses such as limited view and faulty segmentation will hopefully be improved with the improved accuracy of the new Kinect 2.0, and do not pose critical problems. Seeing as a 3D image is generated, the amount of information is large, and segmentation is more easily done than in traditional camera approaches. Furthermore, the segmentation from the 3D image generated by the Kinect is lighting- and color-independent, proving a more stable and robust segmentation.

It seems that due to natural limitations in the Kinect, the most robust and informative of the simple features are height above ground and the derivative of this height along with the 2D floor position found from the centroid. Other approaches have been investigated, but seem less robust. Parameters based on covered area on the floor are sensitive towards persons falling on their knees/ending sitting, and partially occluded persons. The joint positions estimated by the skeleton tracker seem inaccurate and not robust enough, especially during and after a fall. The centroid height is naturally a more robust parameter, seeing as a large amount of points are used to generate it, but is sensitive towards occlusions from the ground up such as chairs and tables.

More advanced features have been suggested, based on the robust simple features. Both inactivity and low height are central parts of a person falling. Generating maps seems like a good and simple way of analyzing a room and learning normal behaviour. Furthermore, the central part of these maps is the 2D position and the height, two features which are quite robust. To get a somewhat accurate 2D position it is only necessary to be able to segment the person at all, and the height is only based on the point on the person with the largest distance from the floor, which is usually in view even though occlusions are present. The height velocity is less robust, but falls are also very often detected when using this as a parameter, which is why this is used when the maps are insufficient.

Testing showed that it is possible to detect walls using the Hough transform and that the technique of defining exit points where the person path crosses a wall line

gave very good results. In all three scenarios investigated the doors were correctly clustered resulting in clearly defined exit zones. Two ways in which exit zones can be used have been suggested. One is for determining if an occluded fall has occured, by analyzing if the point where the person track is lost is close to an exit zone. This made the algorithm able to detect falls ending fully occluded, which has been a problem in literature. However, it is necessary to demand a certain number of exit points for training before this analysis can be employed. The second is to avoid having a sensor in more private and smaller rooms by registering the normal absence duration values when exiting through the different exit zones. This feature has not been investigated enough, but as long as exit zones are separated in a satisfying manner, and doors leading out of the house are detected by the large absence duration variance for example, this seems promising. In addition enough slack should be added to minimize the amount of false positives.

Normal falls within view are easily detected; both inactivity and a drop in height is present. In general, if a fall is in view it seems like it will be detected, provided the maps are somewhat accurate. There are some challenges when it comes to exiting the Kinect view, in which a sudden drop in height may occur due to the fact that only some body parts are visible as the person is exiting. This will usually not cause false positives because of the maps and exit zones, but they still may occur. If this proves a problem, one could exclude all fall detection when exiting and leave it to absence duration instead. Furthermore, this will be a less significant problem with the Kinect 2.0.

Bedrooms are more complicated scenarios. There are larger height variations seeing as a person is undressing, sitting on the bed, lying in bed etc. In addition it seems that the segmentation algorithm acts unsatisfactory when a person is lying in bed, with blankets etc. This might be improved for the Kinect 2.0, but it seems that even though the segmentation is faulty, it is somewhat consistent, resulting in inaccurate but functional maps. As a result of this, all five different falls were detected in the bedroom also.

All in all the Kinect shows great potential for fall detection among elders. There is a lot of information in a 3D image, which makes segmenting robust. The detection criteria are based on simple features of a fall; height drop and inactivity, which seems to increase robustness. Of course, more realistic testing with a large amount of falls is necessary but it seems that due to the simple principles used, this will merely be a matter of tuning and adding/removing/adjusting parts of the algorithm. It is also important to recognize that if a fall is detected, a depth image snapshot can be sent to caretakers to make the final decision. In this way, false positives will not pose large problems if they do not happen very often, and the system can be tuned more “aggressively” to be able to detect a larger portion of falls. In other words, it seems that in general automatic fall detection would best work as a filter, sending only abnormal activity to a person making the final decision.

1. Introduction

This chapter will give an introduction to the assignment, motivation for choosing the assignment, and also go through and discuss related work regarding fall detection using Microsoft Kinect.

1.1. About the Project

One of the main challenges for enabling elders to stay at home for as long as possible, as independent as possible is being able to ensure their safety in a more automatic way. An important part of this is fall detection. According to Stevens et al.[2] "More than a third of older adults fall each year and 10% to 20% of falls cause serious injuries such as fractures or head traumas". The article concludes that "Fall related injuries among older adults, especially among older women, are associated with substantial economic costs. Implementing effective intervention strategies could appreciably decrease the incidence and healthcare costs of these injuries".

A system where a fall is detected automatically would significantly decrease the average time between a fall occurs and the necessary treatment is started, thereby decreasing the damage and costs a fall introduces. Moreover, automating and making the health care of the elderly more efficient will grow in importance seeing as the percentage of elders in Norway (and worldwide) will increase significantly during the next $20 - 30$ years[3]. Several technologies have been tested and used for fall detection, but each technology seems to have its own strengths and weaknesses, and a very robust and unintrusive solution seems to be abscent[4]. According to Yu[5] "...the existing methods have not completely solved the problem of fall detection for the elderly and patient".

The Kinect sensor introduces a new, unintrusive approach with more information available compared to a regular camera. This enables better accuracy and the possibility of developing new algorithms, as well as the possibility of combining camera with 3-D imaging.

1.2. About the Thesis

The focus in this thesis is investigating principles regarding fall detection with Kinect. Consequently, little emphasis has been put on implementation and de-

tails regarding how things are solved in practice. This can be found on the enclosed cd, and it is also rather straight forward in most cases. In cases where MATLAB functions have been available, these have been used without giving detailed descriptions.

In both chapter 3 and 4 testing and results are documented quite extensively. An effort has been made to avoid repitition, but in chapter 4 almost each fall during testing is viewed and commented on separately. This is an attempt to shed light on the many different fall scenarios one might encounter which must be accounted for when designing an algorithm.

All tests have been executed by the author, Erlend Kvinge Jørgensen.

The discussion will be spread throughout the thesis, not as a final paragraph in each chapter. This seems more natural, seeing as it in my opinion makes the discussion easier to follow and holds the text more together. However, conclusions will be added at the end of the chapter when needed, seeing as this is a good way to sum up a chapter.

The program used for analysis is MATLAB, version R2013a. The files on the enclosed CD are .m-files, which can be opened with MATLAB. Seeing as the running time is very dependent on implementation and operating system etc., only computational complexity has been evaluated, described by O(..) notation.

Due to potential formatting issues, and the fact that some figures might be hard to read, the thesis as a .pdf-file is also included on the CD. This will enable the ability to zoom and view the report exactly as intended.

1.3. Other Technologies

Yu[5] compares and classifies existing fall detection technologies, and can be viewed for more detail. The article divides existing technology into three categories; wearable device, ambience device and vision-based.

1.3.1. Wearable Device

Posture/Motion Device

Several approaches have been developed using a sensor measuring either posture or motion. Several accelerometers placed on the user either together or spread out seems to be a common approach, i.e. [6, 7]. This enables the sensors to estimate the posture/motion of the user. Furthermore, optical sensors, position tilt-swithes etc. are also used for the same purpose. Because of the very limited amount of information available and the necessary fixed position on the user which can easily be put off, these solutions are very prone to false alarms. This has been confirmed by

employees at St. Olavs Hospital in Trondheim, which state that the amount of both false alarms and missed detections are unsatisfactory, resulting in an unwillingness to use these devices.

Alarm Device

A common approach in Norway today is alarm devices worn by the user, often called safety alarms. Typically these devices are worn around the neck, and must be pressed actively for an alarm to go off. This, of course, makes it necessary for the user to be conscious and able to press a button after the fall event has occured. Furthermore it is necessary for the user to remember the presence of the sensor, and the necessity to press the button which can be a problem in combination with dementia and disorientation after a fall.

There are some advantages of wearable devices; they are cheap and simple to install. However, as mentioned, passive devices are often prone to false alarms and missed detections. Active devices have the need for the user to actively press a button, which is undesirable. Moreover, wearable devices are intrusive and requires the user to remember to wear the given device. "The general comment from practising doctors is that most of patients have low will to wear devices for detecting falls because they feel well before a fall occurs"[5].

1.3.2. Ambience Device

Ambience devices includes vibration sensors on the floor[8], IR-sensors[9] and sound sensors[10, 11]. Also, pressure sensors have been used especially for bed exit/entering/presence situations. These devices are non-intrusive and cheap. However, as in the previous section we have a limited amount of information; there is no way of knowing if the pressure/vibrations generated comes from a person or something else. As a result of this, the amount of false alarms is significant.

1.3.3. Camera-based Device

Traditionally these approaches have been used by one (or more) camera(s). These concepts will also be reviewed later in the context of using the Kinect instead of cameras, and will be discussed more thoroughly then.

Body-shape-change Analysis

During a fall the shape of a person will change. In other words, the box containing the person (bounding box) will go from a small floor area and large height to smaller

height and larger floor area. Several methods based on this principle have been proposed using Hidden Markov Models and different features of the person bounding box[12, 13].

Inactivity Detection

A fall will usually end with a period of inactivity. Therefore, if inactivity occurs in a certain context it is possible to draw the conclusion that a fall has occured. Several approaches for deciding context and determining inactivity have been proposed[14, 15].

3D Head Motion Analysis

If it is possible to detect the head position of the person it is possible to analyse changes in head position, especially in the vertical direction. This can indicate if a fall has occured or not.

The different camera-based techniques all have the general weaknesses of camera-based solutions, they are dependent on lighting conditions and the computational load can be large. Similar situations may look very different from different angles, and the algorithms may also be color- and shape-sensitive seeing as this is what is used to determine the characteristics of a person. In addition, one needs more than one camera to robustly determine depth information.

As mentioned, some of these techniques will be reviewed in the context of the Kinect later in this report. Because of the accurate 3D representation obtained by the kinect, and the already developed tracking algorithms, some of these approaches can be done more accurately and rather easily using the already developed Kinect SDK. It will also be investigated to combine these approaches to make a more robust detection algorithm.

1.4. Related Work

Many articles have been published with focus on fall detection using the Microsoft Kinect. The most important and relevant ones will be reviewed in this section, although additional publications may be referenced in later sections.

1.4.1. Human Fall Detection Using Kinect Sensor

Kepski and Kwolek[16] combine the Kinect with an accelerometer to create a more robust system for when the view of the Kinect is occluded. They use their own

algorithm for finding the person by using foreground extraction based on finding the largest connected component in a difference map. The static scene used for the difference map is updated by using the median of a given number of previous images to take into account changing scenery; moving chairs etc. They also have their own ground plane detection algorithm. Fall detection is based on if the distance from the 3D centroid of the person to the ground plane is below a certain threshold. They were able to detect all falls, but had very simple scenarios and very few different kinds of falls.

1.4.2. Image-based Fall Detection with Human Posture Sequence Modeling

Dai et al.[17] use the skeleton tracking feature of the Microsoft Windows SDK to extract the 3D coordinates of each of the 20 joints defined in the tracker. They use Hidden Markov Models to classify a sequence of joint coordinates as one of seven motion classes based on a training set of 130 clips. After a few steps of simplification and preparation including Principle Component Analysis, K-means clustering and training of Hidden Markov Models a calssificator is proposed. Experiments are run with each activity performed 10 times, and the results show a 95 % recognition rate of fall activities and an 87 % recognition rate for activities of daily living. They conclude that the approach has great potential seeing as the size of the training set is small compared to the size of the testing data, and we still see good results. However, as will be discussed in later sections, the position of each joint may not be accurate at all times, especially during a fall.

1.4.3. Fall Detection from Depth Map Video Sequences

Rougier et al.[18] employs an approach where the distance from the 3D centroid to the ground plane is used as the first priority decision parameter. However, if the view of the person is occluded the 3D body velocity prior to occlusion is used as the decision parameter. They use a training set, and decide a 97.5 % confidence interval based on the mean values and standard deviations for distance and velocity for normal activities. The success rate for non-occluded falls is 100 %, and the overall success rate including occluded falls is 98.7 %. They conclude that occlusions are a significant problem in such systems, as in any vision-based system, and must be addressed.

1.4.4. Fall Detection System Using Kinect's Infrared Sensor

Mastorakis and Makris[19] use the properties of the bounding box surrounding the person for determining if a fall has occured. The focus is on the first derivatives of

the size of each dimension of the bounding box, and thresholds for floor coverage area "velocity" and height "velocity" of the bounding box are set. If both of these thresholds are exceeded at the same time, a fall is suggested. After this, an inactivity test focusing on height is run, and if this test is positive, a fall is detected. This is based on the fact that a fall usually starts with an expansion in floor coverage area and a contraction in height, followed by a period of inactivity. Also here a training dataset was used to determine threshold values. The computational load is very low, and they were able to succesfully detect all falls and neglect all non-falls. Nevertheless, as will be discussed in later sections, using only one Kinect may cause the floor coverage area to be incorrect because of occlusions both from the person's body and from other objects.

2. Sensor and Software

2.1. The Microsoft Kinect Sensor

The Microsoft Kinect sensor has several interesting features. Originally used for the ability to interact without a remote controller on the Xbox 360, many other areas of use have been suggested. A large amount of applications have been developed, many available, open source, through the OpenNI website[20]. Microsoft have also developed their own SDK for creating own applications on a standard computer. The ability to get a 3D view of a scene yields more information readily available, and simplifies tasks such as hand gesture recognition, body pose detection etc. compared to camera-based solutions. The color- and lighting-sensitivity introduced by traditional camera-based solutions is also far less significant when using the kinect.

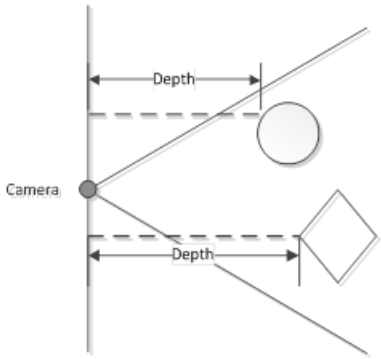
2.1.1. Technological Specifications

The Kinect consists of three different sensors; an RGB camera with a maximum resolution of 1280 x 960 at 30 fps, a multi-array microfone and an infrared camera/depth sensor, all developed by PrimeSense[21]. Both the camera and the infrared sensor have a field of view of 43° vertically and 57° horizontally. Seeing as the focus will be on the infrared depth sensor in this thesis, this is also what will be described in detail. Furthermore, an RGB camera and a multi-array microfone is rather self-explanatory.

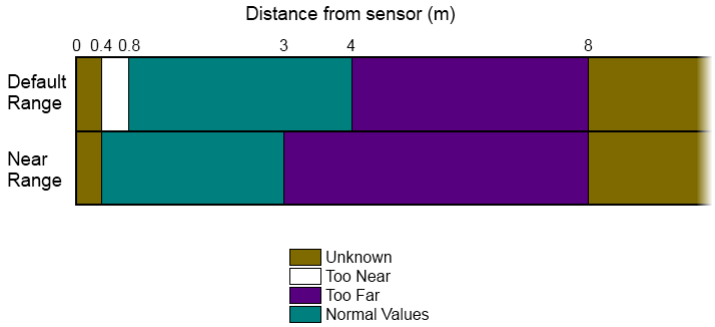
The depth sensor works in the way that an array of infrared beams is emitted from the device. The infrared camera then measures the intensity of each beam crossing-point thus extracting information about the depth at these points. See Fig. 2.1a for an illustration. This is sufficient to create a depth map, and the resolution for the sensor is 320 x 240 x 16 bpp, which is often sufficient, especially for small areas.

2.1.2. Limitations

It is clear that there are some limitations when using this technology, although some of these limitations will be less significant when the Kinect 2 (descibed in the following section) is available. However, because the Kinect 1 is the sensor used in this thesis, these limitations will be discussed.



(a) Illustration of depth values received



(b) Distance limitations for the Kinect

Figure 2.1.: Illustration and limitation of depth values received. Taken from Microsoft's web page[1]

The maximum accurate range of the sensor is given to be 4 m by Microsoft (see Fig. 2.1b), but during testing satisfactory segmenting was provided up to $7 - 8$ m. The minimum range can also cause problems; it is given to be 0.8 m. In a small room, this can pose problems. Furthermore, the rather small vertical field of view makes the minimum distance for viewing the whole person rather large in the context of a small room.

The fact that the depth value is based on the intensity of a light-beam suggests that the depth value is dependent on the reflective abilities of the object viewed. This has been mentioned in literature, but has not posed a problem during the testing and collection of data, and it seems this is limited to very specific situations.

A basic limitation for the 3D view technology is the lack of transparence. In other words, an object blocks the infrared camera view for objects behind it creating a shadow thus only revealing the object closest to the sensor. This results in different reponses based on the object's orientation. More specifically, if a persons is viewed sideways, it is impossible to determine attributes of the arm farthest away from the

camera seeing as this arm is in the shadow of the closer part of the person. This will be discussed in further sections.

2.1.3. The Kinect 2

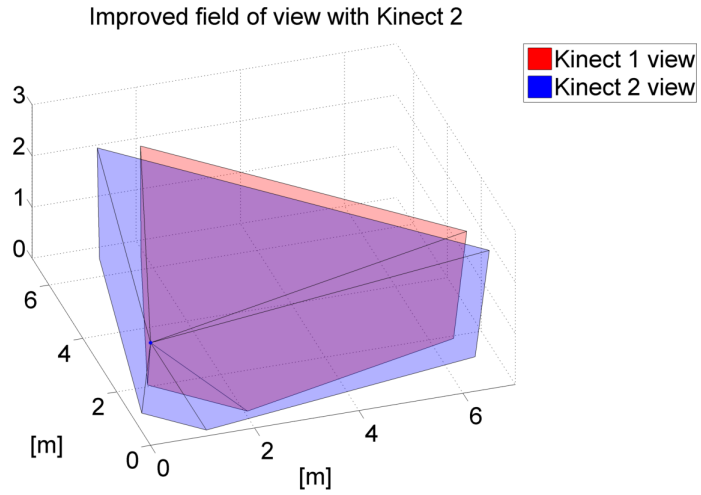


Figure 2.2.: Illustration of improved field of view for Kinect 2. Both Kinects are placed at $(0, 0, 2)$

As mentioned above, some of the limitations will be less significant if one uses the Kinect 2 instead. This is the second generation Kinect, made for the new Xbox One. The SDK for the Kinect 2 is not available yet, but is planned to be released summer 2014. The Kinect 2 depth sensor has improved from the original Kinect in almost all aspects:

- The resolution has increased to 512 x 424 x 16bpp
- The field of view has increased from 43° x 57° to 60° x 70° decreasing the minimum distance for viewing the whole person significantly. For an illustration, see Fig. 2.2.
- The technology for determining depth has changed from measuring the intensity of an emitted infrared beam to employing a time of flight-based approach similar to a radar gun. This yields far more accurate depth values thus the possibility to get far more accurate tracking and 3D representation.

There are also several more improvements which are not relevant for this thesis. It is clear that this will improve the performance of the fall detection system, making it more accurate and therefore most likely make the system more robust. Therefore, these limitations will not have a large focus in this thesis. However, as mentioned above, the basic limitations of 3D view technology will still be apparent, making this an important consideration.

2.2. OpenNI

The OpenNI framework[20] provides a simple and fast starting point for using the Kinect for PC. The website contains OpenNI SDK, middleware and applications to be able to get started very quickly with developing software using the Kinect. The SDK used is the OpenNI 2.2.0.33 Beta (x64). The middleware used is the NITE 2.2.0.11(x64) middleware which is developed by PrimeSense, and is by far the most popular middleware in the OpenNI framework with almost 66000 downloads. Here some sample programs are available, and one of these sample programs provide the basis for the software developed in this thesis. Microsoft have also developed their own SDK, but this seemed less intuitive and is also not open source.

2.2.1. NITE Middleware

As mentioned above, the NITE Middleware is the most popular middleware in the OpenNI framework. The sample program used as a basis for developing software is called UserViewer, and is a basic segmentation and skeleton tracking program using only the depth sensor. It segments persons, and tracks them estimating skeleton joint positions. When a person is first discovered, there is a calibration fase, in which the tracker is "unsure" if this really is a person. If further tracking suggests this is actually a person, the tracker switches from "calibrating" to "tracking" and the person is tracked. The API is well documented, and contains most of the functions needed for basic software development. Example depth frames in later sections are screenshots from the UserViewer program.

The built-in user tracker system works well for scenarios where a person is facing the sensor. Naturally, this gives a good impression of skeleton orientation, placement etc., and this is also the demanded scenario when the Kinect is used for games on the Xbox. However, a fall detection system must consider all orientations seeing as the person must be able to move freely in the viewed space. The segmentation usually gives an accurate impression of which depth values belong to a person and which don't, but the skeleton estimation struggles significantly when the person is standing in a more sideways manner. This is natural seeing as it is impossible to determine the properties of the parts of the person which are in the view shadow. Three cases with skeleton tracking enabled are illustrated in Fig. 2.3.

From Fig. 2.3 it is clear that when a person is lying on the ground, which is likely after a fall, both algorithms struggle; the skeleton estimation is very inaccurate whereas the segmentation algorithm is fairly accurate, but often struggles to cover the whole person because of a noise threshold used in separating the person from the floor. It seems both the segmentation and the skeleton tracking are simpler when a person is facing the viewer, and this is also the scenario in mind when the tracker was designed. An improvement is suggested in later sections, but some weaknesses are hard to eliminate, as will be discussed.

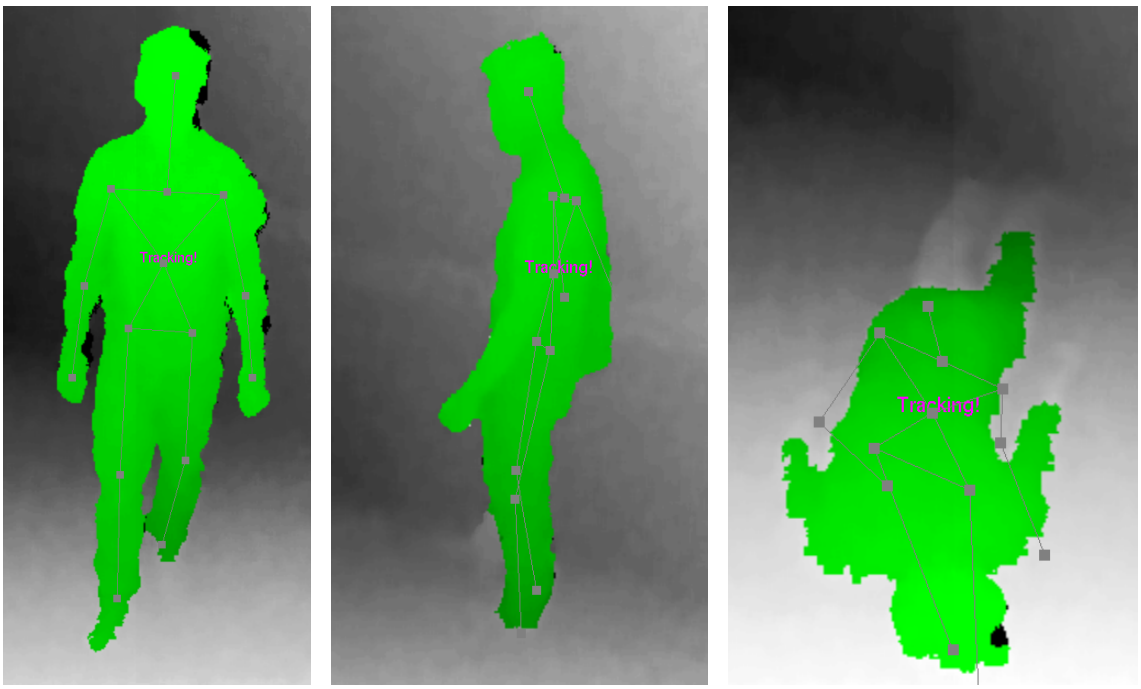


Figure 2.3.: Estimated skeleton for different scenarios; facing the sensor, sideways relative to the sensor and lying on the floor

For a more detailed description of both OpenNI and NITE, see the documentations available on the OpenNI website[20].

2.3. Kinect Studio

A part of the Microsoft SDK is a program called Kinect Studio, which enables the user to record sequences of input from the Kinect, and replay them as real input to an application. There are also several ways to view the data; a depth view and a 3D view giving a more descriptive way of analyzing the input data. This is very useful when developing applications, for both practical and analytical reasons. When analyzing changes in the algotihms it is useful to have the exact same input data for comparison.

3. Basic Properties

There are many properties of a person being tracked readily available (or with minor added computation) from the NITE segmenting and tracking algorithm. It is of interest to analyze these properties, seeing as a lot of them are used as important parts of fall detection algorithms in literature, and it is also these simple properties more advanced techniques will build upon. In this chapter, we will first go through the precomputation needed, and then compare the responses of each parameter in a larger test. This will give valuable information both for using the parameter alone as a fall detection criterion, but also regarding use of each parameter in more advanced approaches.

3.1. Precomputation

Some precomutation steps are necessary to make evaluation as general, robust and accurate as possible. The steps are quite simple and intuitive, and will be described in this section.

3.1.1. Coordinate transformations

Basic Coordinate Systems

In general, two coordinate systems are used in both OpenNI and NITE; real world coordinates and projective coordinates. Real world coordinates describe position in a coordinate system as illustrated in Fig. 3.1, whereas projective coordinates have the same y-value, which is the raw depth value, but the x- and z-coordinates are given from pixel coordinates in the depth view image. There are functions in the NITE library for converting between these two coordinate systems, and both are used as return values for different functions in the library, depending on what is most appropriate.

Common Coordinate system

To maximize the view area it is of interest to place the Kinect at a certain height and with a certain tilt, as illustrated earlier in Fig. 2.2. Because the tilt angle

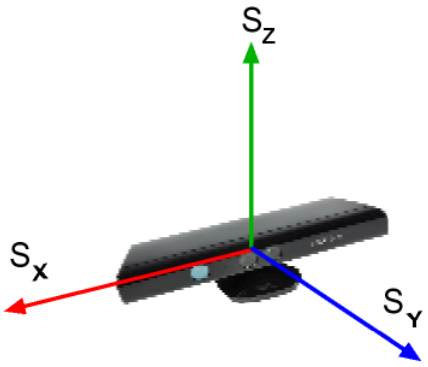


Figure 3.1.: World coordinate system used in OpenNI and NITE

and height for the Kinect can vary, it is of interest to have a common coordinate system independent of tilt angle and height. To get an accurate representation of the bounding box etc., the tilt angle must be compensated for. Furthermore, to get an accurate measurement of height above ground it is also of interest to compensate for the height above ground of the Kinect. Considering this, a common coordinate system is chosen, where the original world coordinate system used by NITE is rotated by the tilt angle around the x-axis and then translated by the negative height along the z-axis. An illustration of this is shown in Fig. 3.2. Because we have access to $\mathbf{p}^{kin}$, we must find $\mathbf{T}^0_{kin}$ given by:

$$\mathbf{T}^0_{kin} = \mathbf{T}_{trans_z,h} * \mathbf{T}_{rot_x,-\theta} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & h \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) & 0 \\ 0 & -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) & 0 \\ 0 & -\sin(\theta) & \cos(\theta) & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

where θ is the tilt angle and h is the camera height.

The obvious way to find these parameters is by measuring the height and tilt angle by hand. However, this seems rather inefficient and inaccurate. Fortunately, the NITE library has a function for finding floor plane parameters, getFloor(), which only takes a few sample frames in the start of the program to initialize. This function returns a point in the plane and the normal vector to the plane, $\mathbf{p}^{kin}_{plane}$ and $\mathbf{n}^{kin}_{plane}$. From basic vector algebra, we have

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}| \cdot |\mathbf{b}| \cdot \cos(\theta_a) \tag{3.2}$$

where θ_a is the angle between the vectors. Because we assume that x_{kin}- and x_{com}-axis are parallel, in other words that the kinect is only tilted by a rotation around

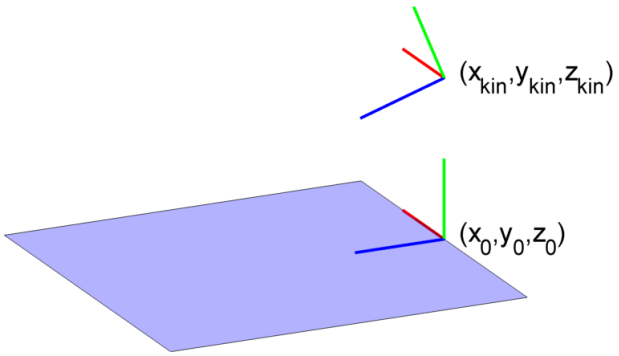


Figure 3.2.: Illustration of the NITE and common coordinate systems

the x-axis, we can assume that $n^{kin}_{plane_x} \approx 0$. Because the normal vector will be in the y-z-plane and the fact that it has unity length, we have

$$\theta = \arccos(\mathbf{n}^{kin}_{plane} \cdot (0,0,1)^T) = \arccos(n^{kin}_{plane_z}) \tag{3.3}$$

Calculating h is slightly more complicated, although this can also be done using basic vector algebra. By finding the point on z_{kin} that intersects the floor plane, $-z_{kin_{int}}$, the height is given by the scalar projection of the vector $(0,0,-z_{kin_{int}})^T$ on z_0. The general equation for a plane is given by

$$a(x-x_p) + b(y-y_p) + c(z-z_p) = 0 \tag{3.4}$$

where $(a,b,c)^T$ is the normal vector of the plane, and (x_p, y_p, z_p) is a point on the plane. $z_{kin_{int}}$is easily found by the equation

$$z_{kin_{int}} = -\frac{ax_p + by_p + cz_p}{c} \tag{3.5}$$

h is now given by

$$h = z_{kin_{int}} \cdot \cos(\theta) = z_{kin_{int}} \cdot n^{kin}_{plane_z} \tag{3.6}$$

We now have both h and θ, which is enough to find $\mathbf{T}^0_{kin}$, and we can now use $\mathbf{p}^0$ for each point instead, which gives a more accurate and tilt/height independent representation. The common coordinate system is what will be used in further sections.

Accuracy of Floor Plane Parameters

Because of the somewhat noisy depth signal given from the depth sensor, the floor plane representation and therefore the tilt angle and height estimates will most likely not be 100% correct. Tests showed that the tilt angle estimate was rather accurate, whereas the height estimate had some deviations, depending on tilt angle and real height. It seems that the height estimate error increased as the height increased.

The error was around 11 cm at a height around 2 m with $\theta \approx 20°$. However, because of inaccuracies in the tilt angle, using the computed values without compensating for the heigh error gave more realistic results. For example, the minimum z-value for the person was around 0 when using the values directly, and negative when the height error was compensated for. As a result of this, the computed values are used directly to compute $\mathbf{T}^0_{kin}$.

3.1.2. Additional Segmenting

Because the segmenting algorithm is based on a person standing with a significant distance from a wall, the noise threshold is rather large. This can easily be seen if a person is lying on the ground, as shown in Fig. 2.3. It is clear that the whole person is not segmented correctly, most likely due to the fact that a the segmenting algorithm assumes the legs of the person is part of the stationary scene, taking noise into account. Clearly, this gives a less accurate description of some person properties. Consequently, an additional segmenting algorithm is proposed, trying to compensate for this, and to give more accurate segmenting.

The algorithm is a recursive function given in Algorithm 3.1 , and is run after the

Algorithm 3.1 Additional segmenting algorithm

create background image by averaging the 10 first frames when the tracker is started

checkPixel(p_c)
  if pixel is marked as person
    for surrounding pixels p_s
      if p_s deviates from background by 10 cm and has less than
          15 cm depth difference from p_c
        set p_s to person pixel
        checkPixel(p_s)
      end
    end
  end
end

original segmenting algorithm. The basic idea is that a stationary image is stored before the person enters the scene. Then, if a depth image pixel most likely is a part of the segmented person, it is "added" to the already segmented person. By using the original tracker, the algorithm can be very simple, and can simply be added as a final step. In this way it is possible to take advantage of the already developed framework, but also add application specific functionality.

An illustration of this can be seen in Fig. 3.3, showing a sample frame from the same scenario as shown in Fig. 2.3 with the red pixels describing pixels added with

Algorithm 3.1. It is clear that the additional segmenting gives a more accurate segmenting algorithm. Seeing as it is desirable to have the best possible segmenting, this algorithm is used in this chapter. No significant flaws were discovered during testing. However, even though the segmenting is more accurate, it is not necessarily accurate. This is the case for example when parts of the person are hidden either by other parts of the person or by furniture etc. This has been discussed in sec. 2.1.2 and will be discussed in further sections. Furthermore, in a real, general scenario the static image generated by the 10 first frames must be updated to account for moving of furniture, doors etc. A way of doing this has been suggested in [22], and is based on extracting the median value of the previous frames.

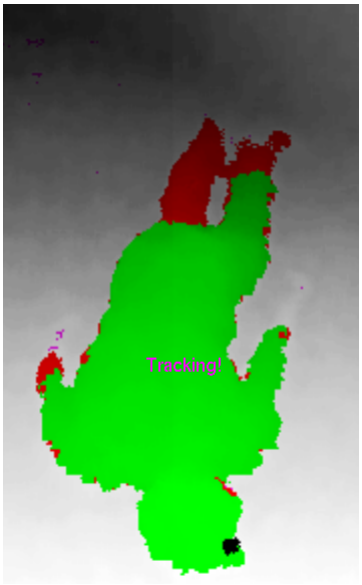


Figure 3.3.: Illustration of additional segmenting algorithm

3.1.3. Kalman Filter

As will be shown in later sections two states, z_{max} and WD, will be estimated along with its first derivatives. The measurements are somewhat noisy, even after spike removal described in sec. 3.2.1. Therefore, for this task a Kalman Filter is a natural choice. The Kalman Filter is a widely known optimal state estimator and can be reviewed in detail in [23], but the main parts of the algorithm are given in this section. The basic principle is to first predict the state and covariance based on the underlying model and the prior state estimates, and then update the predicted estimate and covariance when a measurement is available. The underlying system model is of the form

$$\mathbf{x}_{k+1} = \mathbf{F}_k \mathbf{x}_k + \mathbf{E}_k \mathbf{w}_k \tag{3.7}$$

$$\mathbf{y}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \tag{3.8}$$

Where $\mathbf{F}_k$, $\mathbf{E}_k$ and $\mathbf{H}_k$ are matrices and $\mathbf{w}_k$ and $\mathbf{v}_k$ are white noise vectors with covariances $\mathbf{Q}_k$ and $\mathbf{R}_k$, respectively. We assume gaussian distributions with zero mean for both $\mathbf{w}_k$ and $\mathbf{v}_k$, where $\mathbf{Q}_k$ and $\mathbf{R}_k$ are design matrices.

The prediction steps are given by

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} \tag{3.9}$$

$$\hat{\mathbf{P}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{P}}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{E}_k \mathbf{Q}_k \mathbf{E}_k^T \tag{3.10}$$

The update steps are given by

$$\mathbf{K}_k = \hat{\mathbf{P}}_{k|k-1} \mathbf{H}_k^T \left[\mathbf{H}_k \hat{\mathbf{P}}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \right]^{-1} \tag{3.11}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \left[\mathbf{y}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \right] \tag{3.12}$$

$$\hat{\mathbf{P}}_{k|k} = \left[\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right] \hat{\mathbf{P}}_{k|k-1} \left[\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right]^{-1} + \mathbf{K}_k \mathbf{R}_k \mathbf{K}_k^T \tag{3.13}$$

The initial parameters are estimates for the state and the covariance matrix, $\hat{\mathbf{P}}_{1|0}$ and $\hat{\mathbf{x}}_{1|0}$. The Kalman Filter is quite robust for the choice of initial parameters[23], and has convergence towards the real values as long as the system is observable.

State Model

In our case a very simple model is used, employing the Kalman Filter as a simple estimator of a state and the first derivative of this state. The second derivative is affected by white noise to account for changes in the states. This results in the following model

$$\mathbf{x} = \begin{bmatrix} x \\ \dot{x} \end{bmatrix}, \quad \dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w$$

where w is a independent, zero mean gaussian noise with variance σ_w^2 and x is either z_{max} or WD. The discrete model, when using first order Taylor approximation is given by

$$\mathbf{x}(k+1) = \begin{bmatrix} 1 & h \\ 0 & 1 \end{bmatrix} \mathbf{x}(k) + \begin{bmatrix} \frac{h^2}{2} \\ h \end{bmatrix} \mathbf{w}(k)$$

where h is the timestep, in this case $\frac{1}{30}$. The model could also be extended to three states, $\mathbf{x} = \begin{bmatrix} x & \dot{x} & a_x \end{bmatrix}^T$ which would result in white noise jerk instead of white noise acceleration. However, this did not yield better results in practice during testing. This may be due to the fact that the model is extremely simplified thus creating a need for a more quickly changing response. It seemed that adding a third state created more of a delayed response. Additionally, we get the following measurement equation

$$y(k) = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}(k) + v(k)$$

where $v(k)$ is zero mean gaussian white noise with variance σ_v^2.

Both $\sigma_{v_z}^2$ and $\sigma_{v_{WD}}^2$ have been estimated by having a person standing still in front of the Kinect and measuring the variance of the response. σ_{w_z} and $\sigma_{w_{WD}}$ are tuning parameters which have been chosen to give a sufficiently fast response for realistic movement. The values used in further sections are:

$$\sigma_{v_z}^2 = 5 \cdot 10^{-5}, \quad \sigma_{v_{WD}}^2 = 6.9 \cdot 10^{-4}, \quad \sigma_{w_z}^2 = 8 \cdot 10^{-4}, \quad \sigma_{w_{WD}}^2 = 1 \cdot 10^{-2}$$

3.2. Bounding Box

The bounding box is defined as the smallest possible box in space oriented along the coordinate axis containing the whole person, expressed by x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}. Seeing as the bounding box is used in several algorithms, it is of interest to have an accurate representation of it. There is a function in the NITE library computing the bounding box x- and z-values, but the depth-values are left out. The reason for this is most likely that the depth of a person is rather inaccurate seeing as it is impossible to tell how far the person streches in the view shadow. In other words, the minimum depth value is easily calculated whereas the maximum depth value is impossible to know accurately because the 3D view only sees the object closest to the sensor. This is also discussed in sec. 2.1.2, and illustrated in Fig. 2.3. However, because the Kinect is elevated and tilted, the person will usually have an angle towards it. Therefore, in most cases the maximum depth value of a person will be lower than the real value, but still give some impression of the space occupied in y-direction. As a result of this, we also find y_{min} and y_{max}. A simple for loop is run, calculating $\mathbf{p}^0$ for each depth pixel segmented as a person, and checking if any of the values are either maximum or minimum values.

3.2.1. Spike Removal

As can be seen in the following section, the min and max values are somewhat noisy, especially the y_{max} values. However, the noise is of a very specific character; spikes of varying magnitude depending on dimension (x,y or z), lasting one or two, maximum three sample frames. It is of interest to remove these spikes in a best possible manner to get a more accurate representation of the maximum and minimum values. There are several possible approaches; a simple low-pass filter, median filter, Kalman Filter etc. However, because the spikes are recognizable, two more ad-hoc algorithms are suggested, exploiting this.

Simple Algorithm

A spike-removal algorithm based on recognizing spikes is shown in Algorithm 3.2. It is very simple and intuitive, and exploits the fact that a spike will introduce a

Algorithm 3.2 Simple spike removal algorithm

set delta_max
set filtered vector to zero

for i = 3 to end
    if(abs($entry_i$ - $entry_{i-1}$)<delta_max
      and abs($entry_{i-1}$ - $entry_{i-2}$)<delta_max)
        $filtered_i = entry_i$
    else
        $filtered_i = filtered_{i-1}$
    end
end

large, unlikely jump in response. The fact that it simply copies the previous value may delay response, if the real value is changing during the spike, but the delay will be maximum three frames, which at 30 fps is 0.1 s.

More Advanced Algorithm

Algorithm 3.3 More advanced spike removal algorithm

set delta_max
set spike = 0
set filtered vector to zero, and $filtered_1 = entry_1$

for i = 2 to end

    if(spike = 0 and $entry_i - entry_{i-1}$>delta_max)
        spike detected
        set spike = 1, $spike_i = i$,$spike_{prevval} = entry_{i-1}$

    else if(spike = 1 and $entry_i$ is within likely range from $spike_{prevval}$)
        spike over
        set spike = 0, $filtered_i = entry_i$
        place $filtered_{spike_i}...filtered_{i-1}$ on a line between $spike_{prevval}$ and $entry_i$

    else if(spike = 0)
        normal run
        $filtered_i = entry_i$
    end
end

As mentioned above, simply copying the previous value if a spike is detected may delay the response if the real value is changing. Therefore, a more complicated spike removal algorithm is suggested, given in pseudocode in Algorithm 3.3. Instead of

copying the previous value, the values are linearly interpolated between the start and end-values of the spike. Because the values are set when the spike is over, the algorithm introduces the need for delaying the rest of the system a few frames, until the spike is over, and the value is calculated. Of course, a maximum spike length should be set, for example to three of four frames, and therefore the maximum delay at 30 fps is around 0.13 s, which should be acceptable.

The two algorithms have been compared throughout the work on this thesis, and it seems (naturally) that the more advanced algorithm performs slightly better. Large weaknesses have not been encountered, and therefore Algorithm 3.3 will be used as a default in further sections.

3.2.2. Sample Run

To give an impression of accuracy, the results from a sample run is shown. In the scenario a person is standing on the same spot, rotating with about 17 rpm with a distance of around 2.5 m to the Kinect. By comparing the responses before and after Algorithm 3.2 and Algorithm 3.3, shown in Fig. 3.4, it can be seen that running spike removal is very effective, and yields a far more accurate reponse. It is also apparent, as discussed earlier, that if the real value is changing during the spike, the more advanced algorithm will perform better, depending on the amount of change. The values after more advanced spike removal will be used as a default in this chapter.

It is clear that y_{max} is the parameter introducing the largest spikes and therefore the largest uncertainty. This does not come as a suprise, given the depth value uncertainties. There are also some spikes in x-direction, which is also natural because x- and y-directions are connected to an outlying depth value. The z-direction however, is very stable. This is also natural, seeing as the spikes in x/y-direction most likely will be below z_{max} and above z_{min} and therefore not affect these values. One can also note that $z_{max} - z_{min} = 1.84\,\text{m}$, which is an error in the person height of about 4 cm.

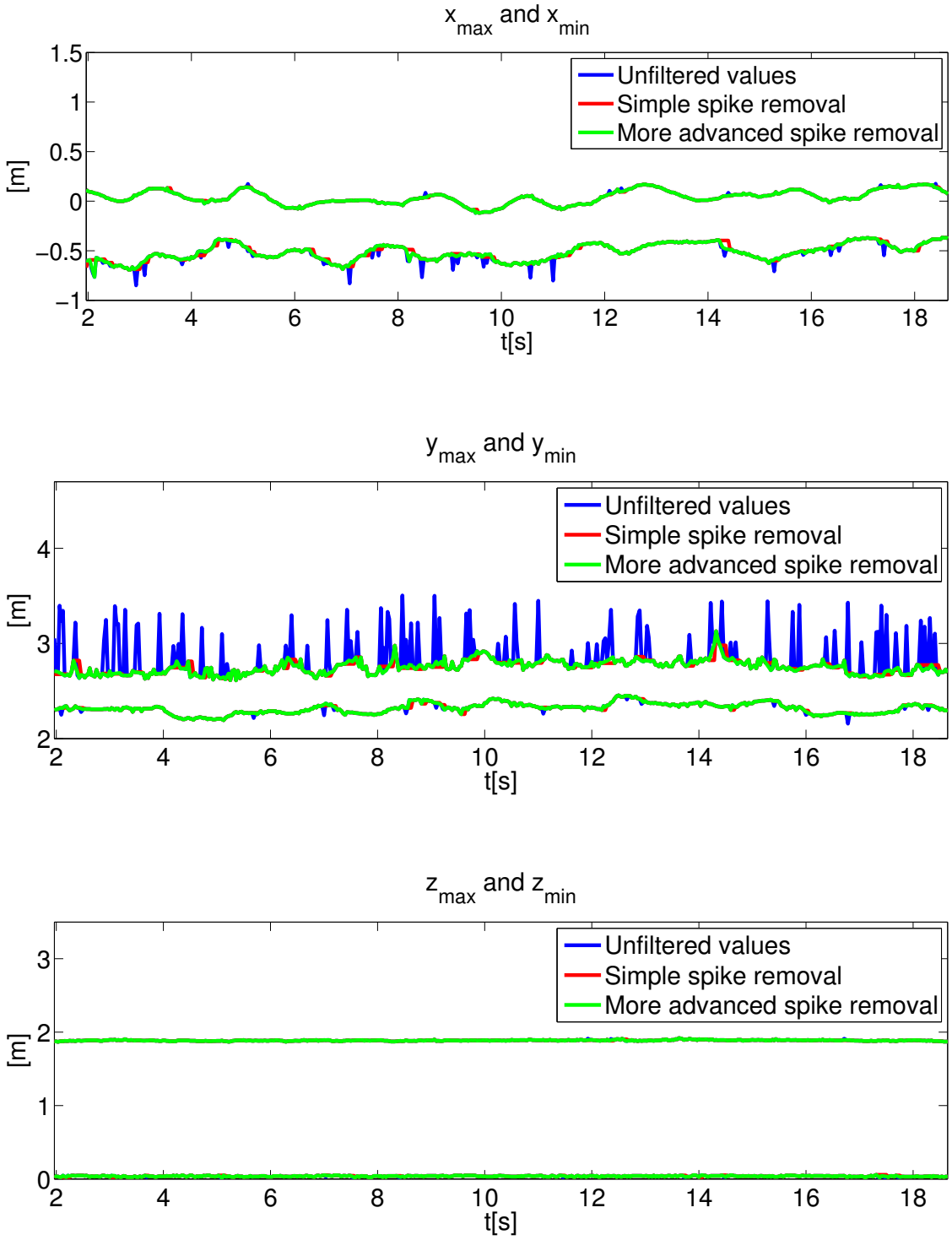


Figure 3.4.: Bounding box values, with filtering

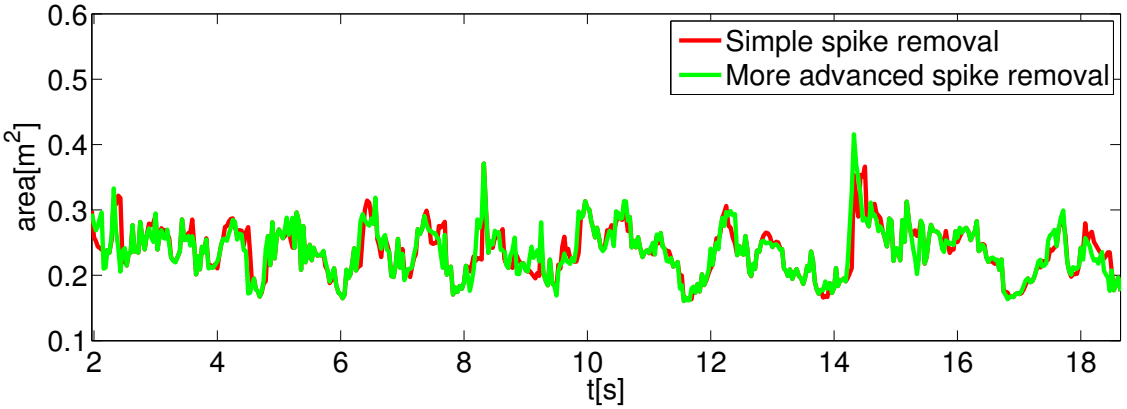


Figure 3.5.: Bounding box area on floor for the two spike removal algorithms

The calculated area projected by the person on the floor, given by $(y_{max} - y_{min}) \cdot (x_{max} - x_{min})$ is shown in Fig. 3.5. As one can see, the two spike removal algorithms give very similar results, both having a mean of around $0.25\,\mathrm{m}^2$, which does not seem too far off. However, the area is in reality fairly constant during the rotation, but the calculated area is rather noisy and after low pass filtering, quite "wavy". This is due to the fact that the bounding box is along the coordinate axis, which means that the floor projection will vary in accuracy being the most accurate when the person is oriented along one of the coordinate axis. An illustration of this is seen in Fig. 3.6.

3.3. Ellipse Approximation

As mentioned in sec. 3.2.2, the bounding box approximation of person area projected on the floor is quite inaccurate, and dependent on person orientation. Rougier et al.[24] suggests using an ellipse instead of a box. This is in the context of 2D imaging, where the properties of the ellipse, more specifically orientation and minor/major axis ratio will be used to determine if a fall has occured. Furthermore, because of the 3D imaging of the Kinect, fitting ellipses could be used in another way. By projecting each person depth pixel onto the floor plane (when the coordinate transformation has been done, it is simply x^0 and y^0), an estimate of the area covered can be seen. Even though it is just the part of the person facing the Kinect that is represented, the projected points still yields an impression of projected area. In general, the area of a person standing up projected on the floor can be approximated as an ellipse, and two ways of doing this will be discussed; one using only edge points, and one using all points to generate the ellipse.

3.3.1. Convex Hull Method

A way of generating the describing ellipse is by using the pripciple of convex hulls[25], which may be visualized as the shape formed by a rubber band stretched around the point cloud. By fitting an ellipse to the points describing the convex hull, a good estimate of the person projected area will often be acquired. An illustration of this can be seen in Fig. 3.6. The MATLAB function *convhull* will be used to find the convex hull, and the function $fit\_ellipse$[26] will be used for fitting the ellipse to the convex hull. This function uses a least squares criterion for finding the ellipse parameters. The convex hull is very sensitive to outlier spikes seeing as these points will expand the convex hull, thereby affecting the ellipse significantly. As a consequence of this, we run a variance-based spike removal algorithm, where the criterion for including the point p is given by

$$|p - \bar{p}| \leq 3\sqrt{\sigma_x^2 + \sigma_y^2} \tag{3.14}$$

where $\bar{p} = \begin{bmatrix} x_{avg} & y_{avg} \end{bmatrix}^T$ and $\sigma_{\cdot\cdot}^2$ is the variance of the points in each direction. This criterion could also be used instead of the spike removal algorithms described in sec. 3.2.1. It seemed that using the algorithms suggested in sec. 3.2.1 gave a more significant response to falls, but comparing the two algorithms at a later stage could also be done to possibly improve performance seeing as this criterion is more mathematically supported.

3.3.2. Moments Method

Instead of using the convex hull, Rougier et al.[24] suggests using all points in generating the ellipse moments, and using these moments to estimate an ellipse. Using an ellipse generated from all points labeled as the person creates a far more robust representation, especially considering spikes seeing as an outlier will not have to be removed, but will instead introduce a minor disturbance in the computed ellipse. Furthermore, arm and leg movement etc. will increase the ellipse size less than in the convex hull method seeing as the density will still be concentrated around the rest of the body. The computations for generating the ellipse are all based on the central moments given by

$$\mu_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (x - \bar{x})^p (y - \bar{y})^q f(x, y) \mathsf{dxdy} \tag{3.15}$$

which more specifically for this application can be written

$$\mu_{pq} = \sum_{person\, pixels} (x - \bar{x})^p (y - \bar{y})^q \tag{3.16}$$

where $\bar{x}$ and $\bar{y}$ are the centroids in x- and y-direction. The normalized moments are given by

$$\mu_{pq}^{'} = \mu_{pq}/\mu_{00} \tag{3.17}$$

From the moments the properties of the ellipse are[24]:

$$x_0 = \bar{x} = \frac{\sum_{person\,pixels} x}{n_{person\,pixels}}, \quad y_0 = \bar{y} = \frac{\sum_{person\,pixels} y}{n_{person\,pixels}} \tag{3.18}$$

$$\theta = \frac{1}{2}\arctan\left(\frac{2\mu_{11}^{'}}{\mu_{20}^{'} - \mu_{02}^{'}}\right) \tag{3.19}$$

$$a = \left(\frac{4}{\pi}\right)^{1/4}\left[\frac{(I_{max})^3}{I_{min}}\right]^{1/8}, \quad b = \left(\frac{4}{\pi}\right)^{1/4}\left[\frac{(I_{min})^3}{I_{max}}\right]^{1/8} \tag{3.20}$$

where I_{min} and I_{max} are given by

$$I_{min} = \frac{\mu_{20}^{'} + \mu_{02}^{'} - \sqrt{(\mu_{20}^{'} - \mu_{02}^{'})^2 + 4\left(\mu_{11}^{'}\right)^2}}{2}$$

$$I_{max} = \frac{\mu_{20}^{'} + \mu_{02}^{'} + \sqrt{(\mu_{20}^{'} - \mu_{02}^{'})^2 + 4\left(\mu_{11}^{'}\right)^2}}{2}$$

3.3.3. Sample Run

An illustration with two sample frames from the scenario described in sec. 3.2.2 along with the person points, bounding box and two ellipses are shown in Fig. 3.6. The calculated areas are shown in Fig. 3.7. It is clear that the moment based area is larger, but is less varying when the person is spinning. In other words, it seems that the moment based ellipse area is less varying, thereby yielding a more realistic floor area derivative.

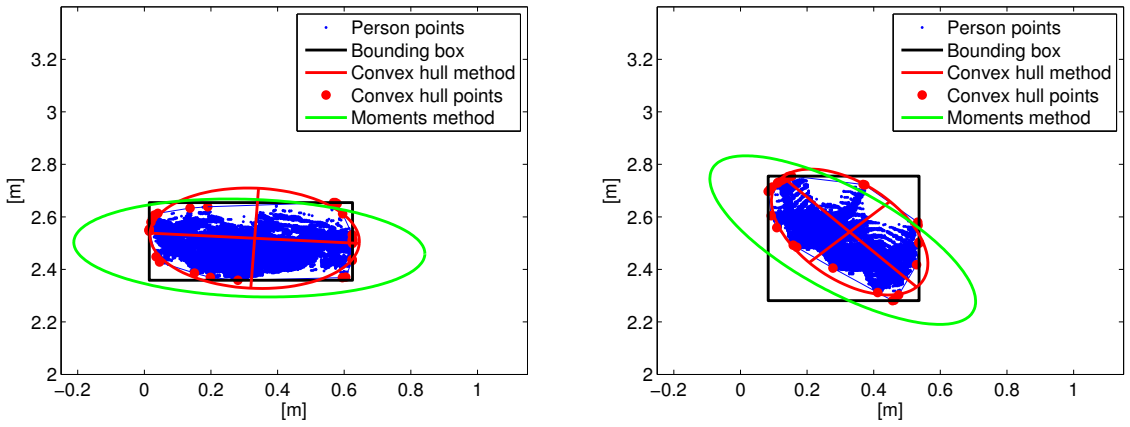


Figure 3.6.: Two sample frames from a person spinning

A significant drawback of both ellipse methods is added algorithm complexity. In general between 4000 - 30000 pixels will cover the person, in other words a person

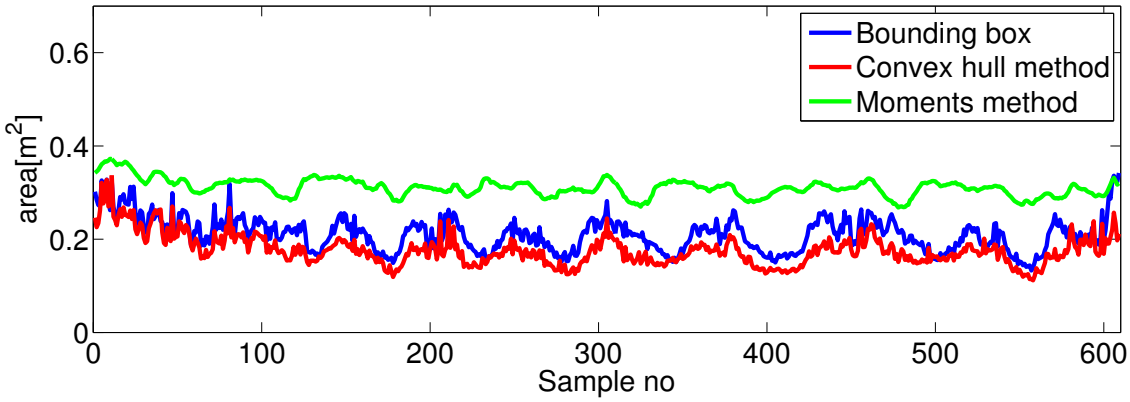


Figure 3.7.: Calculated area using different approaches

will be represented by 4000 - 30000 points on the floor plane and these points must be considered when employing one of the algorithms above. However, efficient convex hull algorithms have been proposed, with a running time of $O(n \log(h))$[27], where n is the number of points and h is the number of points on the convex hull (which is usually a small number compared to the number of points, between 10 and 40). The running time for the moment calculations in sec. 3.3.2 is $O(n)$ seeing as we need to iterate through all points and generate moments based on this. Both running times are linear (or close to linear), and the bounding box running time is also linear (due to the fact that one must iterate through all points also here). This suggests that neither of the ellipse methods will increase computational complexity by a significant amount.

3.4. Basic Property Analysis

In this section the properties described above, in addition to some other readily available features will be reviewed. This is to give an impression of the raw data we have to work with, and also to give an impression of the accuracy of each of these parameters. A large part of the literature regarding fall detection with Kinect uses one or more of these features, which makes it very relevant to analyze the accuracy and robustness.

Some of the information is based on the raw, segmented depth-frame, whereas some of the information is from the skeleton tracker in the NITE library. Both parts could be improved by improving the segmenting and skeleton tracking algorithms, but one can assume that competent people have been working on this for a while, both for Microsoft and PrimeSense. Therefore, this is not a focus in this thesis (except for the minor addition mentioned in sec. 3.1.2), and the assumption is that the segmenting and tracking algorithms are hard to improve further than suggested, especially given the accuracy of the sensors. When it comes to the skeleton tracker, a confidence

measurement is also given, describing the confidence for each joint. This will also be considered in analysis. Furthermore, the performance of both algorithms will most likely improve when the Kinect 2 is released with following SDK, and the analysis done is only valid for Kinect 1.

Some example scenarios will be analyzed, based on [28], which covers the most basic and essential scenarios. To get a quick and simple introduction of the responses, only one test is run for each scenario, with the person facing the camera. Test have shown that a person facing the camera is usually the hardest scenarios to analyze, seeing as the person might block parts of its own body, as discussed in earlier sections. The responses will be compared, and it seems that even though one run is not enough to get a full impression, one run of each scenario will be quite descriptive.

One can also employ the logic of necessary and sufficient conditions; it is necessary but not sufficient that one run gives satisfactory results. In other words, if one sample run gives unsatisfactory results, this is a clear sign that the given property might not be appropriate for fall detection. If one sample run gives satisfactory results, there is a demand for further analysis to be able to categorize this property as a useful and robust property.

3.4.1. Properties Analyzed

Bounding Box Properties

Mastorakis and Makris[19] suggest using the properties of the bounding box for fall detection. These properties will also be analyzed in this thesis. However, some small modifications will be done; instead of using the height of the bounding box, in other words $z_{max} - z_{min}$, only z_{max} will be looked at. With the coordinate transformation described in sec. 3.1.1, z_{max} is the maximum height of the person above the floor. The following values are defined

$$d_x = x_{max} - x_{min}, \quad d_y = y_{max} - y_{min}, \quad d_z = z_{max} - z_{min}$$

The reason for using d_z instead of z_{max} is avoiding the need for calculating the floor plane. However, the floor plane is readily available from the NITE library. Data about the robustness of this algorithm was hard to acquire, but there are other robust floor plane detection algorithms, i.e. Zhao et al.[29]. This suggests that the floor plane is detectable in most cases. If in some cases this is shown to be problematic, the height and tilt angle could be measured, or d_z could be used instead.

The strength of only using z_{max} is robustness against items blocking the lower part of a person; if a table etc. blocks the view of the feet, a jump in bounding box height will be seen, whereas z_{max} will remain stable. Seeing as the probability of a person being blocked by a chair, table etc. is significant in a real scenario this

seems like a more robust measurement. It could be argued that this approach is sensitive to the upper part of a person being blocked, but this will also disturb d_z and could be considered far less likely that the lower part being blocked in a typical home scenario.

A second measurement suggested in [19] is

$$WD = \sqrt{d_x^2 + d_y^2} \tag{3.21}$$

This is to give a measure of occupied space in the floor plane. The natural measure of occupied space in the floor plane would be the area given by

$$A_{pl} = d_x \cdot d_y \tag{3.22}$$

However, as discussed in previous sections, this is rather inaccurate due to the nature of the depth image; only the closest object is described by the depth value. It seems that inaccuracies will be less disturbing when using WD seeing as the two bounding box sizes are separated. In other words, a large d_x combined with a small, inaccurate d_y will affect WD less than A_{pl}. This is illustrated in Fig. 3.8. It is clear that WD has a more even increase than A_{pl} which, due to inaccuracies in values, in this case is desired. As a result of this, z_{max} and WD are the bounding box properties that will be considererd. Furthermore, the first derivatives of these two properties will also be included, which are decision parameters in [19]. As mentioned above, a Kalman Filter will be used, and it is $\hat{z}_{max_{KF}}$, $\hat{\dot{z}}_{max_{KF}}$, $\hat{WD}_{KF}$ and $\hat{\dot{WD}}_{KF}$ that will be evaluated and used in later sections.

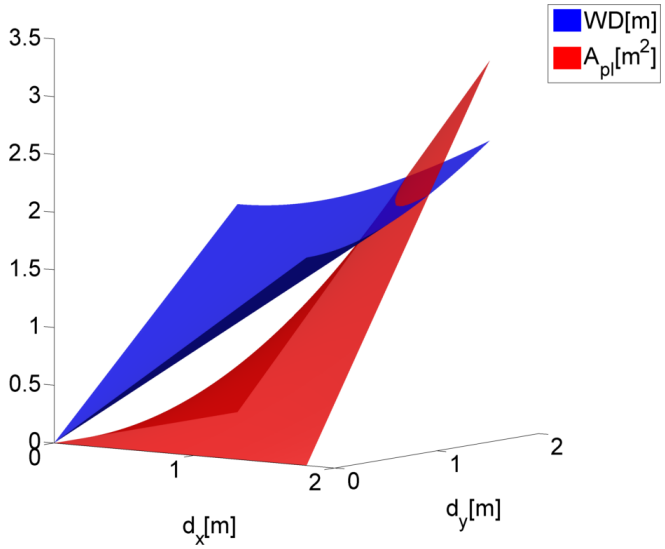


Figure 3.8.: Comparison of WD and A_{pl}

Joint Positions

Dai et al.[17] use the skeleton tracking feature of the Microsoft Windows SDK to extract the 3D coordinates of each of the 20 joints defined in the tracker. The joints

extracted are shown in Fig. 3.9. All joint positions during a fall sequence are stored, simplified and used in a classification algorithm involving Hidden Markov Models. In [17] this yields good results. However, as mentioned, by observing joint positions during a fall, and especially right after a fall this approach seems rather inaccurate. Seeing as the skeleton tracker is based on a person standing upright in front of the Kinect, facing it, a person falling is not something the tracker is prepared for handling. Furthermore, determining the 3D position of 20 joints during a fall seems like a far more complicated task intuitively, than doing the same thing on a standing person facing the Kinect.

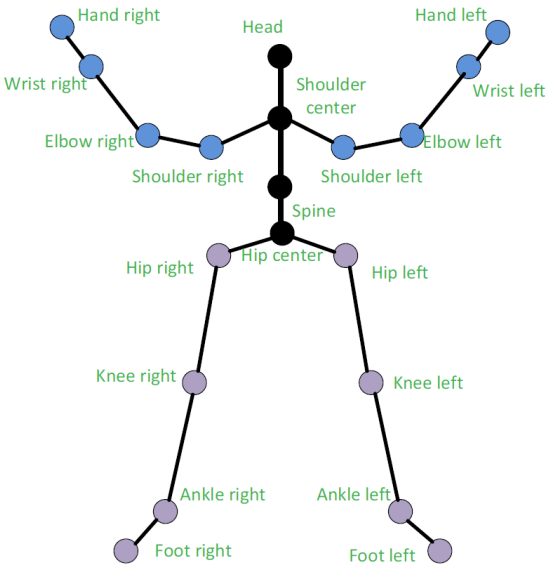


Figure 3.9.: Skeleton joints tracked by Microsoft skeleton tracker

From Fig. 2.3 it is clear that the accuracy of each joint position is very unpredictable during and after a fall. On the other hand, it might be that the inaccurate parts are removed during the simplification stages before the classification algorithm seeing as these are very similar for all falls, and the point of simplification is to extract the information separating one case from the other cases. Even though this might be the case to some degree, it seems like an over-complicated approach given the accuracy of the information available and the highly varying response depending on Kinect- and person orientation. Moreover, the running time of this approach is significantly larger than other, simpler approaches.

It seems that the most robust joint estimated is the head. This makes sense seeing as it usually "stands out" from the body, and it usually is the joint furthest from the floor, even in the fall process. Of course, when the person is lying down after the fall the head might be lower than other parts of the body, but this is after the fall process has happened. This is also reflected in the joint position confidence. The most apparent parameter for fall detection when it comes to head position is z_{head},

seeing as this describes the height above ground. Based on this I have also chosen to include z_{head} in the properties considered.

Centroid Height

Several approaches utillizing the person centroid, maybe in combination with other properties have been proposed[16, 18]. These approaches use the distance from the centroid to the floor plane, in this case z_{centr}. This is a more robust measurement than z_{head} seeing as the basic requirement is only actually being able to segment the person, in other words if a person is detected, a somewhat accurate centroid measurement is available. However, z_{centr} will be sensitive to occlusions seeing as the whole person should be viewed to create realistic centroid coordinates. Nevertheless, z_{centr} will also be a property considered.

Ellipse Approximations

Both ellipse approximations mentioned above will be considered. The property extracted from both approaches is the ellipse area. One could look at other parameters, such as the ellipse ratio $\rho = a/b$ which gives an impression of the shape of the ellipse, or the orientation of the ellipse. However, seeing as the ellipse is generated by the points on the person projected on the floor, the area seems like the most describing and robust parameters.

3.4.2. Properies Not Analyzed

There are some properties that have been omitted for different reasons. As mentioned above, positions of other joints, such as shoulder, knee etc. have been excluded due to inaccuracies in estimation. Furthermore, x- and y-position of the head could be used, but also these estimates seemed inaccurate. More complicated analysis using the whole segmented person depth image could also be tested, such as fitting an ellipse around the person and viewing the orientation of this ellipse, or filling ratio of the rectangles making up the bounding box.

H-W Ratio

Kepski and Kwolek[22] combine several parameters; height/width ratio, $z_{max}/z_{max_{tot}}$, z_{centr} and $max(\sigma_x, \sigma_z)$ from single depth images. These features are used as input to different classifiers. z_{centr} is already included. $z_{max}/z_{max_{tot}}$ is merely a more robust version of z_{max}, seeing as it takes into account the height of the person, which in a normal scenario will be $z_{max_{tot}}$. However, in the simple scenarios run in this section, it is the same person doing all falls, and therefore $z_{max_{tot}}$ will be the same in all scenarios, and we can conclude that if z_{max} is appropriate, $z_{max}/z_{max_{tot}}$ also is.

$max(\sigma_x, \sigma_z)$ did not seem to give a good separation between falls and non-falls, and height/width ratio seemed inaccurate due to the difficulties of width measurements which will be discussed in the following section.

3.4.3. Testing and Results

Backward fall	a) ending sitting
	b) ending lying
Forward fall	c) ending on knees
	d) with forward arm protection
	e) ending lying
Lateral fall to the left	f) ending lying
Lateral fall to the right	g) ending lying
Neutral	h) sit down on a chair
	i) lie down on bed
	j) bend down, catch something
	k) person spinning

Table 3.1.: Fall scenarios in simple comparison

As mentioned above, a series of scenarios, inspired by [28] have been carried out. Seeing as it seemed that the most problematic orientation was when the person was facing the Kinect, this is the case in all scenarios. Futhermore, only one run of each scenario has been done. This is to keep the amount of data at a somewhat acceptable level and also to give a simple introduction of the different basic properties. The scenarios included are given in Tab. 3.1. Sample frames from the different scenarios are shown in sec. B.1.

Following is a comparison of each basic property for each scenario, ending in a matrix describing how appropriate each property is for each scenario. The decision is based on if the property follows what is expected of the property; the height should decrease during a fall, WD should increase and so on. Moreover, the abillity to separate the response from the opposite responses will be a part of the decision. A rating from 0 to 2 will be given, where 0 is inappropriate, and 2 is very appropriate.

It is important to realize that this score is based solely on the response of one fall for each scenario, and, as mentioned above, only gives an impression of appropriateness and especially inappropriateness of each scenario.

$\hat{\mathbf{WD}}_{\mathbf{KF}}$

The calculated $\hat{WD}_{KF}$ for the different scenarios can be seen in Fig. 3.10. It is clear that this property is not always informative, but gives a fair impression of floor area

in some cases. Events a), c) and d) give unsiatisfactory results. Event a) and c) are rather intuitive, seeing as the actual covered floor area is far less than falling flat out. In contrast, event d) is not that intuitive. What has happened here is that the person becomes "discontinuous"; one can only see the front part and the legs, which do not seem connected in space from the view of the Kinect. Therefore the segmentation algorithm decides that only the front part is labeled as a person. The added segmentation in sec. 3.1.2 will for the same reason not include the legs either; the "discontinuity" is too large, and therefore not included in this step. It is also intuitive that lying down on a bed (event i)) is very similar to falling flat out when it comes to area projected onto the floor, which is reflected in the response.

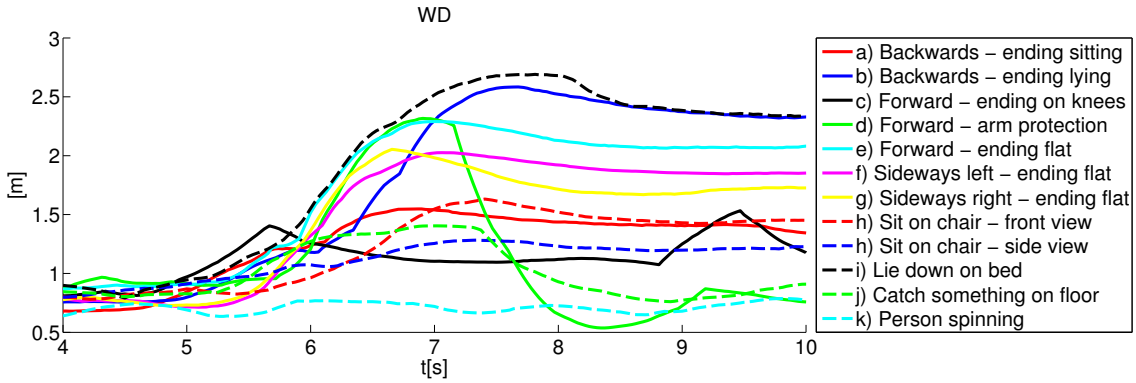


Figure 3.10.: WD for different scenarios

$\hat{WD}_{KF}$

The reponse is shown in Fig. 3.11. Here parts of the reason for also viewing the time derivative of WD is apparent. The maximum value of $\hat{WD}_{KF}$ is separating the responses, and even though event d) quickly drops to an unsatisfactory level when viewing $\hat{WD}_{KF}$, $\hat{WD}_{KF}$ has a satisfactory response. However, events a), c) and i) are not detectable by using $\hat{WD}_{KF}$.

$\hat{z}_{max_{KF}}$

The response is shown in Fig. 3.12, and is far more promising than $\hat{WD}_{KF}$. As can be seen from the results, there is a clear separation between neutral and fall events (if we omit i), which has proven to be problematic to distinguish based solely on simple properties). As mentioned, this property will also be far more robust when it comes to occlusions such as chairs, tables etc.

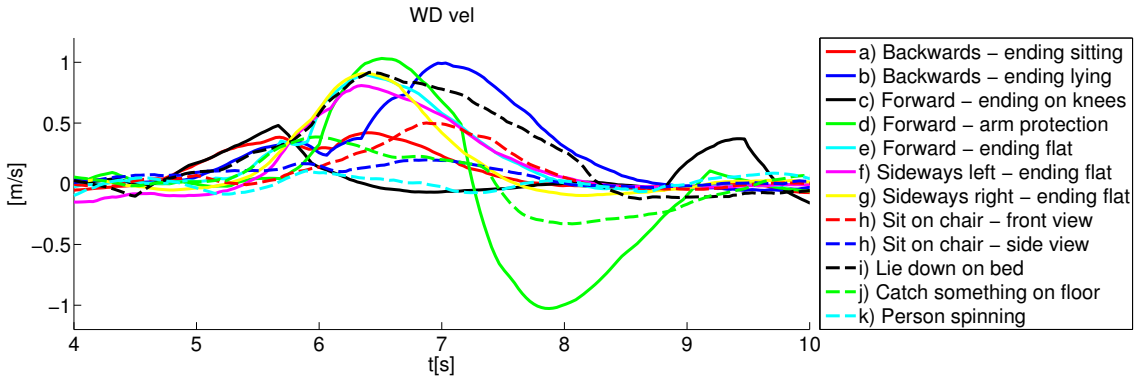


Figure 3.11.: WD derivative for different scenarios

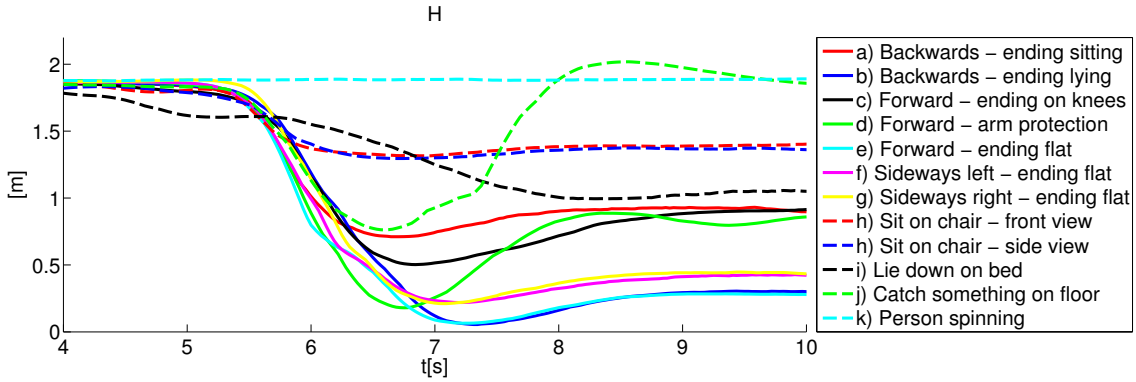


Figure 3.12.: Person height for different scenarios

$\hat{\dot{\mathbf{z}}}_{\mathbf{max_{KF}}}$

The response is shown in Fig. 3.13. Here we actually have a threshold in which all neutral events are separated from all fall events. If we look away from case j), which will easily be handled with inactivity detection, the differences in $\hat{\dot{z}}_{max_{KF}}$ is significant. The difference between the largest neutral event $\hat{\dot{z}}_{max_{KF}}$ and the smallest fall event $\hat{\dot{z}}_{max_{KF}}$ is $0.428\,\mathrm{m/s}$, which is significant. We are also able to separate event i) from the rest, which has shown to be problematic with the previous properties.

$\mathbf{z_{head}}$

The results can be seen in Fig. 3.14. The response is similar to $\hat{z}_{max_{KF}}$, but far more noisy in some scenarios. This is natural, seeing as the skeleton tracker is struggling to find the head position when the head is not necessarily the "joint" on top of the body whereas the body point furthest away from the floor is simple to find. Also here, lying down on the bed is creating problems. By looking at the joint position confidence shown in Fig. 3.15 it is also clear that falls create a loss in confidence. The events start at $t \approx 5.7\,\mathrm{s}$. It is clear that, especially for the fall scenarios the

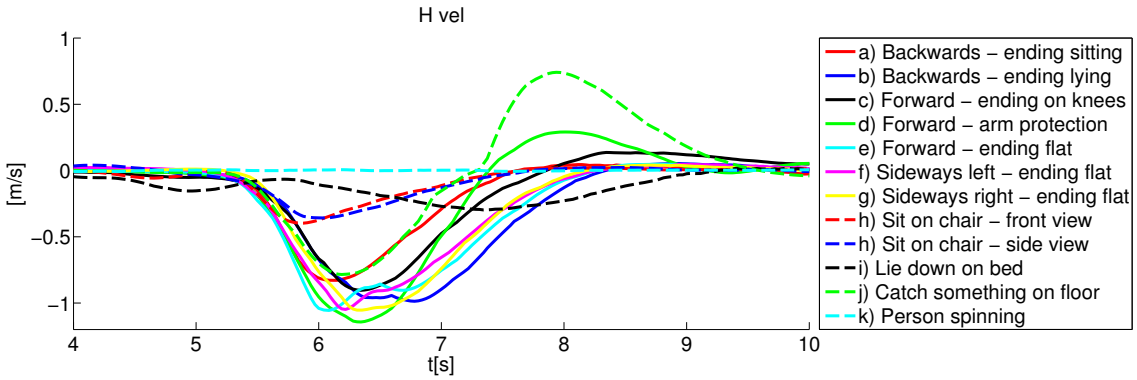


Figure 3.13.: Height derivative for different scenarios

confidence changes to 0 after a few seconds and often varying between 0 and 0.7. It is undesirable to use a parameter with this much uncertainty. One could suggest that a drop in head position confidence may suggest a fall has occured, but in case a) for example, the confidence stays at 0.7 long after the fall has occured.

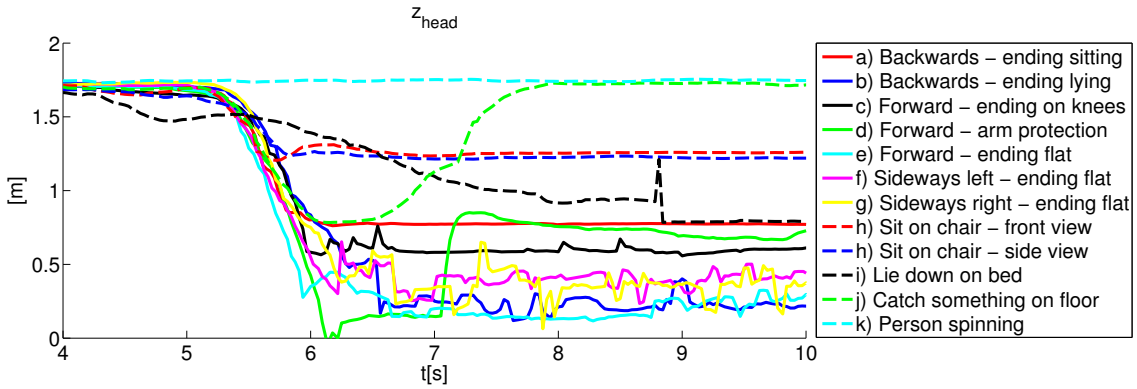


Figure 3.14.: Estimated head height for different scenarios

z_{centr}

The response is shown in Fig. 3.16. Similar to $\hat{z}_{max_{KF}}$ we have a space separating fall events from neutral events. In this case it is around 0.1 m, which is larger than the space when viewing $\hat{z}_{max_{KF}}$, even for event i). However, if we omit case i), the difference in $\hat{z}_{max_{KF}}$ is 0.3 m, which is more significant. Compared to $\hat{z}_{max_{KF}}$, z_{centr} will be more robust to limb movement, seeing as lifting an arm for example will increase $\hat{z}_{max_{KF}}$ to the height of the arm whereas z_{centr} will be less affected. On the other hand, $\hat{z}_{max_{KF}}$ will be more robust when the person is partially occluded, especially from items on the floor such as chairs; when parts of the person is occluded, z_{centr} will increase seeing as it is only the visible parts that are used to calculate z_{centr}. A test showed that when a person of height around 1.8 m was occluded by an

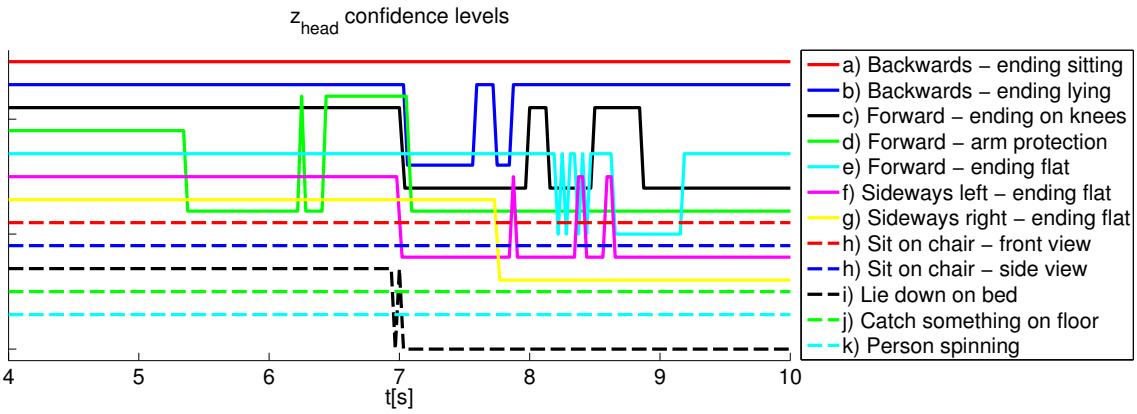


Figure 3.15.: Head height confidence for different scenarios. Shifted for illustrative purposes. All start at 0.7

object with height around 0.9 m, the centroid shifted around 0.2 m, or 19 % upwards, whereas $\hat{z}_{max_{KF}}$ remained stable.

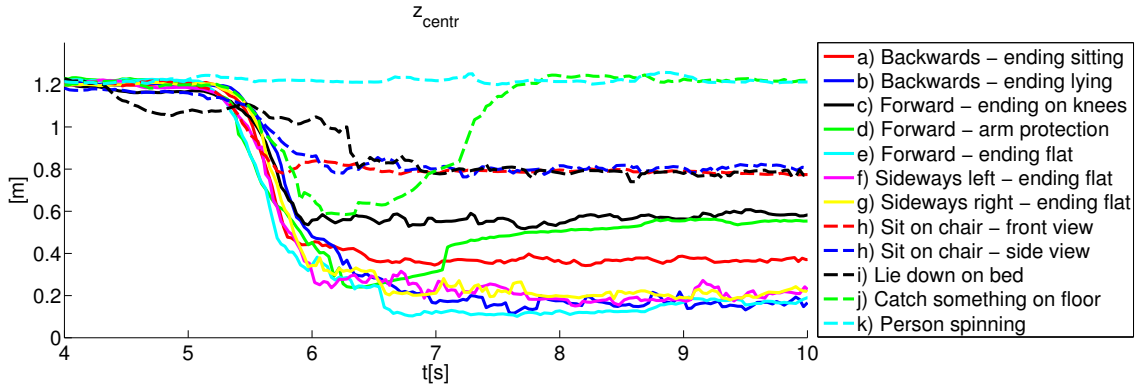


Figure 3.16.: Centroid height for different scenarios

Ellipse Area

Ellipse areas are shown in Fig. 3.17 and Fig. 3.18. It is clear that the same problems as with $\hat{WD}_{KF}$ are apparent. It is also clear that the moment based ellipse is far less noisy than the convex hull generated ellipse. Consequently, even though the moment ellipse is far less noisy and varying than $\hat{WD}_{KF}$ when the person is spinning, it is not more straightforward to detect falls using the moment ellipse criterion.

Summary

It is clear that some properties are more robust than others. In fact, a larger part of the investigated properties seemed to give unsatisfactory results, even properties

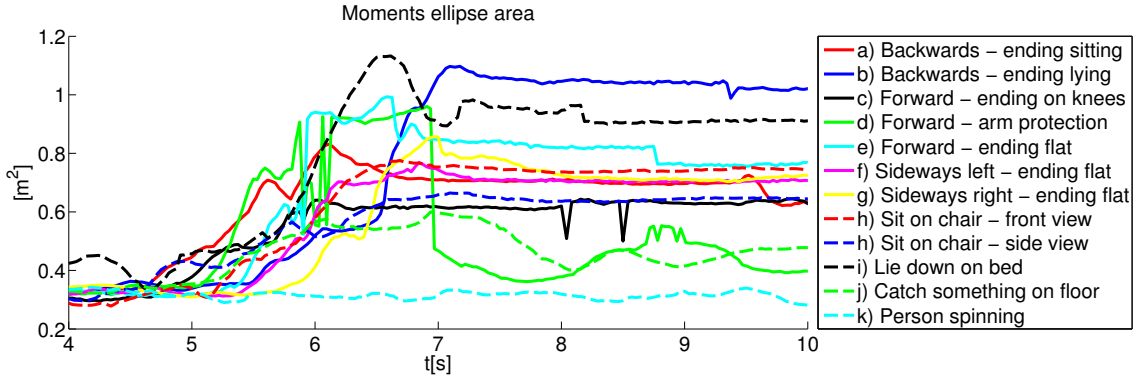


Figure 3.17.: Moments generated ellipse area for different scenarios

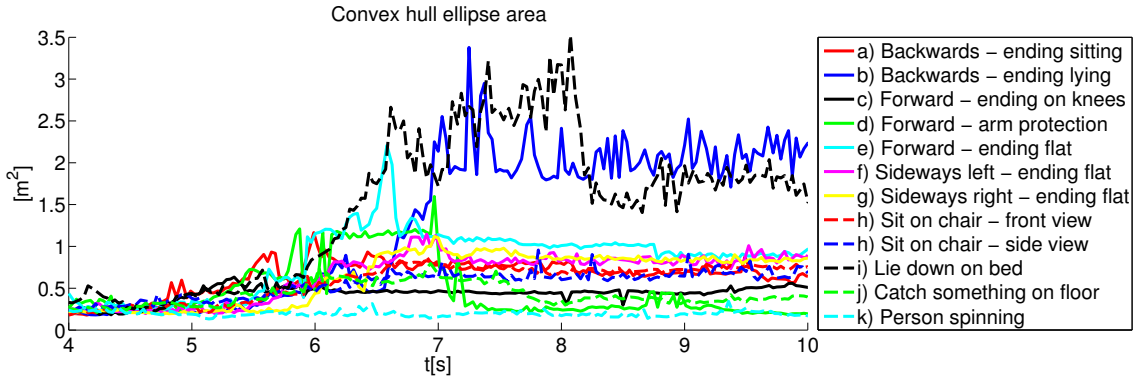


Figure 3.18.: Convex hull generated ellipse area for different scenarios

suggested in literature. As mentioned, to sum up the results of this section, the appropriateness based on a single run of each scenario is summarized in Tab. 3.2. An important consideration taken into account when determining the score, is that it is assumed that additional inactivity detection is also employed after the property has been considered. This seems like a necessary addition, seeing as this will prevent event j) among others from being detected as a fall. It is also not necessary to alarm someone if the user is able to get up after a fall and can call someone if necessary etc. Including inactivity detection is also supported in literature[19, 14, 15], and can be done in different ways, from very simple to very complicated. More details about the inactivity detection will be given in further sections.

The table shows that it is event c) and i) that are creating the most problems for finding a confident, simple-property fall detection algorithm. This is because the parameters are based on floor area or height, and both of these are somewhat in a middleground between standing up and falling. It is also important to realize that the problems of separating event i) from falls are also dependent on the height of the bed. In the test scenario the "bed" used is a table, and the height is therefore larger than a typical bed height would be. As a consequence of this, a more realistic bed height might change the appropriateness of $\hat{z}_{max_{KF}}$ from 2 to 1 or even 0. This

	$\hat{WD}_{KF}$	$\hat{\dot{W}D}_{KF}$	$\hat{z}_{max_{KF}}$	$\hat{\dot{z}}_{max_{KF}}$	z_{head}	z_{centr}	Ellipse Area
a)	0	0	1	2	1	2	0
b)	2	2	2	2	2	2	2
c)	0	0	1	2	2	1	0
d)	0	1	1	2	1	1	0
e)	2	2	2	2	2	2	1
f)	2	2	2	2	2	2	1
g)	1	2	2	2	2	2	1
h)	1	1	2	2	2	2	1
i)	0	0	1	2	0	2	0
j)	1	1	2	2	2	2	2

Table 3.2.: Appropriateness of each parameter for each scenario

is of course not desired, and it seems the only property able to classify all events in the test is $\hat{\dot{z}}_{max_{KF}}$. This is backed up by the intuitive idea that a fall will at some point have an uncontrolled acceleration towards the floor.

4. More Advanced Techniques

In this chapter several more advanced techniques and features will be discussed. Most of the more advanced techniques are based on simple machine learning schemes, and require the need for training. The approaches will be presented in different sections, and a larger test will be run in which all features will be reviewed. This larger test is described and discussed in sec. 4.5.

In many of the following methods, the 2D person floor position is a central part. The 2D position is found by transforming each depth pixel segmented as a person to world coordinates, and then to the common, floor plan coordinate system. From these points the centroid is found, and used as 2D floor position. After this transformation a Kalman Filter could be used to better the estimate, but as will be seen in further sections, the position estimate is not very noisy. This is mostly due to the large amount of points (usually between 5000 and 40000 depending on distance to the Kinect) used for calculating the centroid, creating a robust estimate.

4.1. Inactivity Detection

There are several ways of defining and approaching inactivity detection, and the amount of detail and complexity is very varying in literature.

A very simple form of inactivity detection is given in Mastorakis and Makris[19]. Here the inactivity detection is simply checking whether v_H, the bounding box height velocity is below a certain threshold for a certain amount of time after the fall has occured.

Cuddihy et al.[30] uses simple infrared motion sensors to get an impression of activity. The parameter chosen is elapsed duration of time that no acitivity has been observed, and based on these data an alert line is generated using points for each 30 min of the day. For robustness, changes in schedule, small variations in behaviour etc. are taken into account.

Jansen and Deklerck[15] employ an approach closer to what might be expected using the Kinect. They use a 3D camera to estimate the orientation of the torso in combination with inactivity detection as input to a context model to determine if a fall has occured. Furthermore, inactivity zones proposed in Nait-Charif and Mckenna[14] are discussed, concluding that this relatively simple approach will have both false positives and negatives. The idea that a context model "must take at

least location, time and duration of the inactivity into account" is introduced. This opens up for more complex and robust inactivity detection schemes. The way this is handled in [15] is a very simple context model where the ground floor is discretized and a historgram is associated with each cell in the grid. When a person is located in a cell for d time units, and then moves away, one is added to the d'th bin in the histogram. In this way, after a learning period, a probability distribution of duration for each cell on the floor is obtained, and can be used to calculate probability for a given duration.

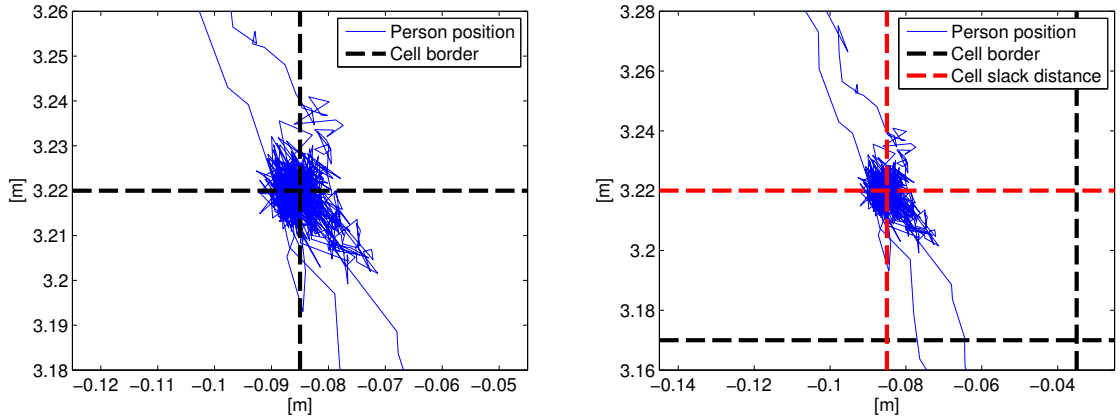
It seems that the method suggested in Jansen and Deklerck[15] has a lot of potential. Taking location, duration and time into account could not only result in an inactivity detection algorithm which generates less false positives and detects more real positives when it comes to falls, but could also detect other unusual events, such as a person passing out in a chair, bed, couch etc. Another way of using the same methodology for exit/entry analysis will be suggested in sec. 4.3.

4.1.1. Noise Handling

One significant weakness when it comes to this approach is due to the fact that the 2D position estimate is always noisy. The magnitude of this noise depends on the sensor. The fact that it is the amount of time spent in one cell before moving over to another cell that is taken into account, and not an accumulated time introduces the need for handling noise. Because of the noise, a scenario could be encountered where the 2D position "jumps" back and forth between 2-4 cells. This is illustrated in Fig. 4.1a, where a typical position of a person sitting in a chair is shown, along with the discretization of the floor. It is clear that even though the person position is only varying with around $2 - 3\,\text{cm}$ which should be considered stationary and a period of inactivity, each cell can end up with only small durations due to the fact that the position is changing between cells. The probability of this happening decreases with increased cell sizes, but the worst case scenario is independent of cell sizes, and is clearly not desired.

Current Cell Priority Algorithm

To handle the issue discussed above, a simple algorithm is suggested, shown in pseudocode in Algorithm 4.1. The algorithm is based, like the name suggests, on prioritizing the current cell. In other words, a "slack distance" is defined, in which the current cell will not be changed, even if the 2D position moves over to another cell. The size of this slack distance is a tuning parameter, along with the grid size. This is illustrated in Fig. 4.1b. When Algorithm 4.1 is used, the worst case scenario is a lot less damaging for the inactivity detection scheme; if the distance varies around the line describing the "slack distance", the first time it crosses $cell_{curr}$ will be updated, and the rest of the observations will be associated with the new $cell_{curr}$.



(a) Without slack distance **(b)** With slack distance

Figure 4.1.: Worst case scenario for inactivity detection

Algorithm 4.1 Current Cell Priority algorithm

if current position is outside of frame defined by $cell_{curr}$
and slack distance around it

store $t - t_{cell_{curr}}$ in $cell_{curr}$ values
set $cell_{curr}$ to current position
set $t_{cell_{curr}} = t$

end

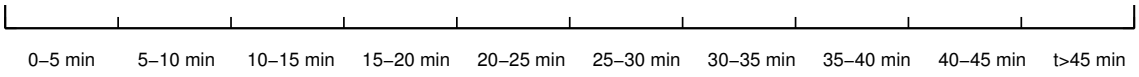
4.1.2. Duration Analysis

There are several ways of analyzing and representing the duration for each position cell. What seems necessary is that durations must be represented in a way such that some measurement of probability can be found for the current duration. This is an essential part of detecting unusual behaviour. Two approaches will be suggested. The two approaches are similar, but have some advantages/disadvantages which will be analyzed in this section.

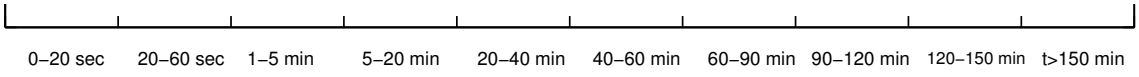
Histogram

In [15] it is suggested using one histogram for each cell. This is to keep the necessary memory at a low level, seeing as the histogram has a finite and choosable amount of partitions. Furthermore, the histogram partitions can have different "lengths", which can be chosen based on normal behaviour; normal walk duration is a few seconds, whereas a person can sit in a chair for over an hour. Taking this into account, it is possible to have "walking-parts" and "stationary-parts" in the histogram. An illustration of this is shown in Fig. 4.2.

The strength of using different lengths, is that it is possible to take into account irregular activity with a response time based on the normal amount of activity. When partitioning as in Fig. 4.2a, if a person falls during walking, it takes five minutes(or more if the partitioning intervals are larger) before the duration is considered irregular in the histogram. In contrast, if the "walking-part" of the histogram is partitioned with smaller intervals, as in Fig. 4.2b, it will take far less time before a fall is considered irregular.



(a) Evenly partitioned



(b) Unevenly partitioned

Figure 4.2.: Different histogram partitions

Storing Exact Values

It is also possible to store exact values, with a maximum amount of stored values, of course. This gives a more accurate description, and eliminates the need for determining histogram partition sizes. However, as mentioned above, the need for memory is larger for this approach. Seeing as the amount of observations is not a problem during testing, storing exact values is the approach chosen for further analysis.

Probability Density Approximation

The most basic way of using the histograms/exact values is finding the probability of the current duration interval d by the simple formula

$$P(d \geq T) = \frac{n_{d \geq T}}{n_{tot}} \tag{4.1}$$

However, it seems like a more advanced approach might yield better results seeing as this creates a more continuous and robust probability density. For example, if d is larger than all others when storing exact values, using 4.1 yields $P(d \geq T) = 0$ whereas generating a probability density might give a more realistic approximation of probability. This is similar for the histogram approach.

To approximate the probability density it is necessary to assume a probability distribution. It seems that using a Gaussian Mixture Model (GMM) is a good place

to start. This is used in [14], and is based on the assumption that the probability distribution is described by[31]

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k p(\mathbf{x}|k, \mu_k, \mathbf{\Sigma}_k) \tag{4.2}$$

where $\sum_{k=1}^{K} \pi_k = 1$ and each mixture component is a Gaussian density function with mean μ_k and covariance $\mathbf{\Sigma}_k$. The log likelihood is given by

$$L(\chi|\theta) = \log \prod_{n=1}^{N} p(\mathbf{x}^n|\theta) = \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k p(\mathbf{x}^n|, \mu_k, \mathbf{\Sigma}_k) \tag{4.3}$$

where $\chi = \left\{\mathbf{x}^1, ..., \mathbf{x}^N\right\}$ is a set of i.i.d. realisations of $\mathbf{x}$ and θ is the model parameters. It is possible to find the optimal values of θ by maximising $L(\chi|\theta)$. This is a nonlinear problem and is therefore not trivial to solve. However, the EM algorithm[32] has been widely used to deal with this problem. The algorithm is a two-step iterative algorithm which searches for the local maximum of the log likelihood. It has been known to be sensitive to initial conditions, but we will assume the sets are simple enough for this not to be a problem.

It is important to realize that K in 4.2 is also unknown; we do not know a priori how many Gaussian densities the probability distribution consists of. However, according to Roberts et al.[33], by using the Minimum Description Length (MDL)[34] principle it is possible to get a good estimate of the number of Gaussians. The principle is stated as *select the model that gives the shortest description of the data set*[31]. The method is based on choosing K which minimizes ζ given by

$$\zeta = -L(\chi|\hat{\theta}) + \frac{1}{2} M \ln N \tag{4.4}$$

where $\hat{\theta}$ is the model parameter estimates and $M = 2dK + (K - 1)$ is the number of free parameters. The first term is simply negating the log likelihood rewarding large values, whereas the second term penalizes large models.

4.2. Height Map

As discussed in sec. 3.4, z_{max} (also referred to as the person height, or simply h in further sections) is an interesting parameter, but is sensitive to persons lying down on beds, couches etc. However, by employing the same mindset as in sec. 4.1 where the floor is discretized into cells, a more robust approach is suggested. By creating a height map where each cell value is the average of all z_{max}-values when the person is located in this cell we can get a good impression of the usual height values in this position. In this way, a more appropriate threshold for z_{max} can be defined where the usual value can be taken into account. A more complicated

approach can also be employed where, instead of using only the average value, a probability distribution can be generated by using all registered values. Unlike inactivity detection, a histogram is easily employed for storing height values; the range of values is far less varying and minimum and maximum values are easily defined.

When a height map is succesfully generated, using z_{max} for fall detection is more robust. In the case of a bed/couch the height map will reflect a lower height in these places, thus taking into account the fact that a person is lying down on the bed making it less likely to trigger an alarm. One could say that this also will make it less likely to trigger an alarm if a fall occurs close to the bed/couch, but as long as the discretization of the floor is fine enough, this will not be a problem.

A way of handling no registered heights in a floor square will also be needed. This can be solved by averaging surrounding floor squares with registered heights, median filtering, defining a general minimum normal height or by simply excluding the height map from fall detection if there are few or no registered heights for the given 2D position.

4.3. Entry/exit Analysis

The use for finding entry/exit zones when it comes to fall detection might not be intuitive. Where a person enters and exits is not necessary knowledge for the methods and properties discussed above. However, when used correctly this could compensate for lack of view. If it is possible to have some impression of how long a person usually is gone when exiting to a certain area, this could give valuable information. For example, if the toilet lacks a sensor (for privacy and space reasons etc.) a fall or loss of conscience in the toilet could be detected (though somewhat delayed) by registering that the person is in this room for a longer period than usual. In other words, unusual absence duration when exiting in a certain exit zone might suggest unusual behaviour. Furthermore, if a room has a blind spot, and a person falls in this blind spot, after a while the person will be in this blind spot for an irregular amount of time. This can be detected with entry/exit analysis. If a person falls behind a fully occluding object such as a couch etc., this will also consitute as an exit point, and will be an irregular exit point which can often be detected (unless it is very close to a door/already defined entry/exit point).

In [31] 1D entry and exit zones are defined on the edges of the picture. This is because the camera is in the roof facing down, and therefore the zones will be on the edge of the image. This is not the case for the Kinect, and it is therefore of more use to employ 2D entry/exit zones. One could employ the concept of GMM for finding entry/exit zones also, as suggested in [31]. However, it seems like a simpler approach is sufficient in this case. Seeing as it is only necessary to classify and separate the entry/exit zones for the purpose of this analysis, applying a clustering algorithm for entry/exit points might be adequate.

4.3.1. Bisecting K-Means Clustering

The Bisecting K-means algorithm is a simple, intuitive clustering algorithm with runtime $O(n)$ where n is the number of points. It uses the K-means clustering algorithm, but adds a bit to it, compensating for known weaknesses. The basic K-means clustering algorithm is shown in Algorithm 4.2[35].

Algorithm 4.2 Basic K-means clustering algorithm

1. Select K points as the initial centroids
2. Assign all points to the closest centroid
3. Recompute the centroid of each cluster
4. Repeat steps 2 and 3 until the centroids don't change

The K-means algorithm can be sensitive when choosing initial points. Usually, these are chosen at random which may yield different results for consecutive runs with the same dataset. Furthermore; because the number of clusters is unknown in our case, it would be of advantage to have an algorithm which add to the number of clusters until a certain criterion is met. The Bisecting K-means clustering algorithm can easily be modified to fulfil this criterion. The (slightly modified from [35]) Bisecting K-means algorithm is shown in Algorithm 4.3.

Algorithm 4.3 Bisecting K-means clustering algorithm

1. Pick a cluster to split
2. Find 2 sub-clusters using the basic K-means algorithm
3. Repeat step 2, the bisecting step, for ITER times and take the split that produces the clustering with the highest overall similarity
4. Repeat steps 1,2 and 3 until the maximum distance from a point to the cluster centroid is below a certain threshold, d_{max}

Because K-means clustering is run ITER amount of times and the best one is chosen, the probability for an unsatisfactory clustering due to bad initial points will be small. d_{max} must be tuned to separate door openings from each other. The natural choice would be around 50 cm which is around half the width of a normal indoors door. However, because of noisy measurements this threshold must probably be somewhat increased. This will be investigated during testing.

4.3.2. Wall Detection

Because a door opening gives some view of the room behind the door when the door is open, the tracker could keep tracking when a person exits the room through a door opening. This could cause trouble for the clustering algorithm above; if the

tracker follows after the person exits the room, the exit points could be far apart, resulting in a faulty exit zone representation. Therefore, if it is possible to detect the walls of a room, one could view the "crossing" of the wall as an exit point, instead of the actual point where the track is lost.

One could use the same approach as floor detection for wall detection. Calculating disparity and creating a V-disparity image has been suggested in [22]. Another relevant technique is employing the Hough transform[36], which is used to find arbitrary shapes in images and described in the following section. This is actually used as a part of the V-disparity approach in [22] to find lines in an image. The disatvantage of the Hough transform is the exponentially growing complexity with number of parameters determining the shape. In the case of a plane, which is the shape we're trying to detect, there are three parameters given by the plane equation (if the equation is divided by the x-coefficient)

$$x + by + cz + d = 0 \tag{4.5}$$

Three parameters makes for a rather computationally expensive Hough transform. However, this method can be simplified, seeing as the floor is detected and the parameters are readily available from the NITE library. If these parameters are available (which they seem to be throughout testing in different scenarios), a simpler approach can be employed.

It is clear that a wall will form a line when projected onto the floor. Because of the coordinate transformation described in sec. 3.1.1, all depth pixels from the Kinect can be projected onto the floor. Moreover, if one only reviews the points above a certain height, the walls will be more apparent. This makes sense seeing as the walls will be more visible above the height of chairs, tv's etc. Furthermore, seeing as the parametrization of a line has only two parameters, the Hough tranform is greatly simplified.

Using the Hough Transform for Finding Lines

The main principle behind the Hough transform is converting the complex problem of finding shapes into a far simpler problem of finding a global maxima in a n-dimensional grid, where n is the number of parameters used to describe the shape. The way this is done, is that the parameter grid is discretized, and each point adds 1 to all grid cells which describe shapes that could contain the current point. This can be viewed as a voting procedure, and the grid cell with the most votes will describe the parameters for the shape which fits the most points.The specifics will be given for a line, seeing as this is the problem at hand.

The most common description of a line is the following equation

$$y + ax + b = 0 \tag{4.6}$$

However, this description has singularities for vertical lines (when $a \to \infty$). Therefore, several other approaches have been suggested. The chosen approach is describing a line by two other parameters, r and θ as illustrated by Fig. 4.3. In other words,

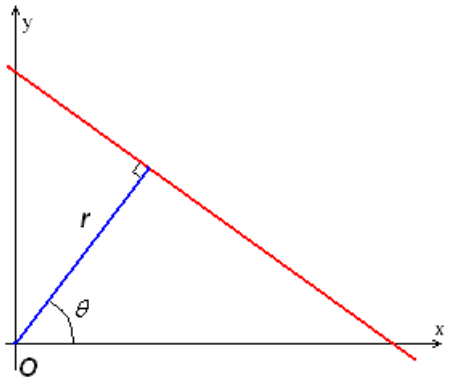


Figure 4.3.: Illustration of (r, θ) line representation

the length and angle of the line perpendicular to the current line going through the point. The equations are given by

$$r = x\cos(\theta) + y\sin(\theta) \tag{4.7}$$

$$y = \left(-\frac{\cos(\theta)}{\sin(\theta)}\right)x + \left(\frac{r}{\sin(\theta)}\right) \tag{4.8}$$

By using this approach, there is still a singularity in eq. 4.8 at $\theta = 0$. This is simply a vertical line, and is easily identified. Each point (x, y) will represent a sinusoidal in the (r, θ)-space. There are of course an infinite amount of representations, but if we choose $r \geq 0$ and $\theta \in [0, 2\pi)$ or $\theta \in [0, \pi)$ and $r \in \mathbf{R}$ a single representation is available. These limitations also result in a finite grid size. The line will be described by the point in (r, θ)-space where most sinusoidals cross, and is given by eq. 4.8. A detailed example can be seen in [37].

Implementation

To account for noisy measurements, some functionality is added based on practical aspects; instead of just adding 1 to the grid point which corresponds to the calculated r given a θ, 1 is also added to all 8 points surrounding the point. This is to account for noise. Furthermore, some functionality is added because often more than one wall is in view. To try to detect all walls in view, the points used to generate the most dominant wall are removed, and the Hough transform is run again. This is done until the most dominant point is below a certain threshold, either a constant or a function of the first peak, number of initial points etc. When removing points, all points in which the r-value has a "grid-distance" of less than 20 are removed. This is also to account for noise.

4.3.3. Exit Zone to Point Association

By looking at the covariance matrix of the points of a certain exit zone it is possible to determine a measure of probability for a certain point to be a part of this exit zone. A 2x2 covariance matrix is often represented as an ellipse, and a confidence region can be defined based on an assumed probability distribution. Seeing as the distribution is unknown, a simpler approach is suggested, not taking into account distribution, but giving a measurement based simply on the covariance matrix. $\mathbf{p} = \begin{bmatrix} p_1 & p_2 & ... & p_n \end{bmatrix}$ are the points defining the exit zone and $\mathbf{R}$ is the covariance of these points. The ellipse is defined by

$$\frac{(\cos(\phi)(\tilde{x}) + \sin(\phi)(\tilde{y}))^2}{a^2} + \frac{(-\sin(\phi)(\tilde{x}) + \cos(\phi)(\tilde{y}))^2}{b^2} = 1 \tag{4.9}$$

$$a = \sqrt{\lambda_1}, \qquad b = \sqrt{\lambda_2}, \qquad \phi = \text{atan2}(v_y, v_x) \tag{4.10}$$

where ϕ is the axis angle, $\tilde{x} = x - x_0$, $\tilde{y} = y - y_0$, x_0 and y_0 are the centroids of $\mathbf{p}$, λ_i are the eigenvalues of $\mathbf{R}$ and $v_1 = [v_x, v_y]^T$ is the eigenvector for λ_1.

For a given point $p = [x, y]^T$, a "conformity coefficient", k, can be given by switching the 1 in eq. 4.9 with k^2 giving

$$k = \sqrt{\frac{(\cos(\phi)(\tilde{x}) + \sin(\phi)(\tilde{y}))^2}{a^2} + \frac{(-\sin(\phi)(\tilde{x}) + \cos(\phi)(\tilde{y}))^2}{b^2}} \tag{4.11}$$

Now, k will give an impression of how much the size of the ellipse must increase to contain p thus giving a value for determining if a point is part of a cluster. This can be used for determining if an exit point is abnormal.

4.4. Occlusions

It is important to take occlusions into account seeing as rooms in houses often have objects that can cause occlusions. As discussed in earlier sections, some features are more sensitive to occlusions than others. It is clear that if the person is totally occluded, it is only the process before total occlusion that can be used to interpret the response. However, if the person is partially occluded, the different features will have different amounts of disturbance.

The inactivity detection algorithm along with the entry/exit zones uses only 2D floor position as input. The 2D floor position is a fairly robust feature, seeing as the 2D position is not far off (for a person standing up) as long as the person is in sight at all. This is especially considering the fact that the occlusions often will be from the floor up to a certain height. Furthermore, occlusions from stationary objects will be "stationary occlusions", meaning that the 2D position will be similar,

even if somewhat inaccurate, when a person is moving in similar patterns around the occlusion.

As discussed in sec. 3.4.3, z_{max} is one of the parameters that is most robust towards occlusions. This is intuitive, seeing as most occlusions in a home environment is covering lower and not upper parts of the body yielding no disturbance in z_{max}. This naturally leads to a small probabillity of disturbance for $\dot{z}_{max}$ as well.

4.5. Testing and Results - Learning Period

In this section three scenarios will be used to test the performance of the different more advanced techniques. Throughout the section, each technique will be evaluated in each scenario. Seeing as a lot of the analysis is similar for all three scenarios, the analysis for scenario 1 will be the most extensive. Here comments that are valid for all scenarios will be added, and the analysis of the other scenarios will refer to scenario 1. When it comes to generating GMM pdf's this could be done for all scenarios, but this is also mostly done for scenario 1 to avoid repitition.

4.5.1. Scenarios

In order to reveal strengths and weaknesses with the approaches proposed above, three scenarios are reviewed. This is to test performance in different situations, seeing as a sensor should work in both living room and kitchen for example. The performances are dependent on different parameters, such as number of doors and size of room.

For each scenario an intensified activity level is used for training purposes. Consequently, the length of periods out of room, sitting in chairs etc. may not be realistic seeing as the main purpose is getting enough data to evaluate the learning methods. The person is walking around a lot more than one might do in a realistic scenario to generate training data in a small amount of time. The reason for this is keeping the data size reasonable, and the lack of a completely realistic testing area with elders is unavailable thus giving a somewhat unrealistic testing scenario regardless. It is important to realize that this is merely an evaluation of appropriateness and robustness, and the length of periods of inactivity is not realistic, but if these are registered correctly, this will also most likely be the case for longer periods.

Scenario 1: Living Room

The first scenario is a living room, around $5.80\,\mathrm{m} \times 3.89\,\mathrm{m}$. Both a depth image and two color images can be seen in sec. B.2, with a red ellipse around the Kinect. We can see that there are some occlusions; a couch, a table, some chairs and an

ironing board. There are two door openings in view, and an additional opening to a veranda out of view. The whole room is not covered by the sensor, and therefore some tests have been done by sitting in a part of the couch outside of camera view. The training set is based on walking out of view through doors, walking around in the living room, and sitting in chairs and couches.

The training period lasted around 1 hour, and all registered 2D position tracks can be seen in Fig. 4.4, in total 61. From this figure some parts of the living room can clearly be distinguished from the others; the couch can be seen as a series of inactivity points around $(-1.2\,,\,3)$, the chair creates a lot of inactivity around $(-0.1\,,\,3.2)$ and the door with tracks ending around $(-0.8\,,\,4.4)$ among others.

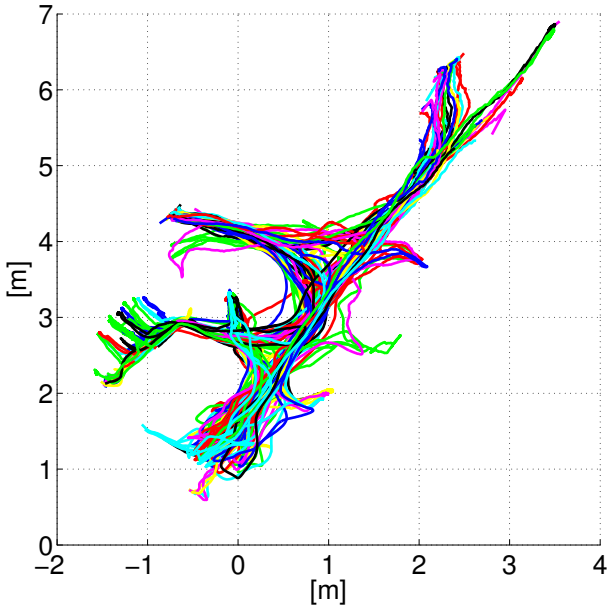


Figure 4.4.: 2D tracks generated from living room training set

The training set is edited to account for sensor weaknesses, and to create a more representative dataset. Because the noise increases for longer distances with Kinect 1, the tracker sometimes keeps tracking even when the person has exited through the door opening on the right in the depth image. This results in a lack of entry/exit point, and instead the track is maintained until the person returns. The tracking data for this inactivity is removed, generating entry and exit points.

However, this will not be a problem in practice, seeing as it does not affect inactivity detection map or the height map. The only technique affected is entry/exit analysis, because both entry and exit points fail to be registered. However, this does not happen everytime, and when the training set increases in size, it is assumed that the amount of entry/exit points is satisfactory either way. Consequently, the editing is only to create more entry/exit points for analysis.

Scenario 2: Hallway

One of the most important weaknesses of the Kinect is the minimum distance. Because of the relatively small vertical spread angle, the floor is not in sight before around 2.25 m away from the Kinect. However, because the analysis is based on height and 2D position, it is not necessary to see the floor the person is standing on in order to analyze movement. The centroid is available and rather accurate even if only the upper half of the person is showing. This will be investigated in scenario 2.

Scenario 1 is rather straight forward. Much space, two doors far away from each other, and most of the movement happens at a comfortable distance away from the Kinect, which leads to being able to view the whole person. It is therefore of interest to test in a less comfortable scenario. The hallway is smaller than the living room, 2.58 m × 2.13 m with sample depth image and two color images shown in sec. B.2. As can be seen from the pictures, there are 5 doors in this hallway, with a minimum distance of around 0.9 m between each other. The small area and many doors will make it harder for entry/exit analysis.

The training period lasted around 13 minutes, and consists of a person walking through the hallway, between rooms, along with some standing in the hallway. All registered 2D position tracks are shown in Fig. 4.5, in total 63.

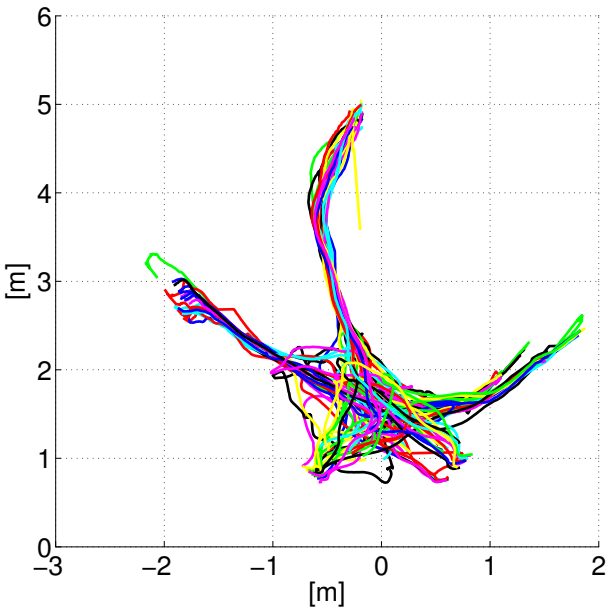


Figure 4.5.: 2D tracks generated from hallway training set

An important thing to notice in the training period is that the doors are open and stationary at all times. This is to make the problem simpler. It seems that the person segmenting algorithm in the NITE library registers moving doors as persons and starts tracking them. This can create problems for the analysis, seeing as it will create unrealistic and unpredictable tracks. Changes to handle this must be added,

either in the tracker or the learning algorithm. The door tracking problem will be further analyzed in sec. 5.1.5.

Scenario 3: Bedroom

The bedroom is also a more complicated scenario than scenario 1; a large part of the floor area is covered by a bed, the room is small and there are occlusions such as closets and dressers. It is also interesting to investigate how the tracker responds to a person lying in bed, lying under a blanket and getting in and out of bed. As in scenario 2 the problem of not having the whole person in view due to small distances is present.

The size of the bedroom is around 2.84 m × 2.56 m with sample depth image and two color images shown in sec. B.2. The training period lasted around 31 minutes, where the person is getting in and out of bed, walking around and bending down to get a book from the shelf on the right in Fig. B.18. All registered 2D position tracks are shown in Fig. 4.6, in total 19, and also here the door is always open. Closing and opening doors will be investigated in sec. 5.1.5.

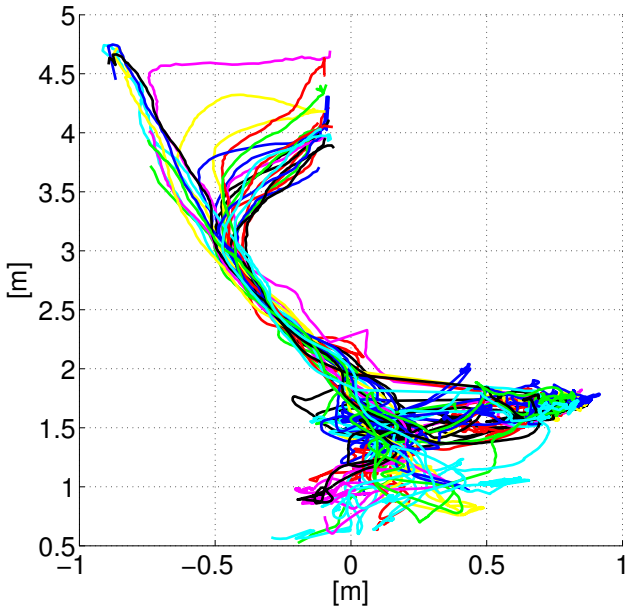


Figure 4.6.: 2D tracks generated from bedroom training set

When a person is in bed, with a blanket pulled over, a large part of the bed is segmented as a person. This is because there are movement in large parts of the bed when the person is pulling the blanket over. This is not necessarily a problem, seeing as the 2D position will be in the bed regardless and the maximum height will also be realistic. However, problems may arise if a fall happens when the person is getting out of the bed. A sample frame is shown in Fig. 4.7. As we can see, both

the person and the bed are considered as the same person. After the person has gotten up, the tracker reinitializes, and the person is tracked correctly. However, a fall might occur in the process of getting out of the bed. This might pose problems, for example fall 3 in sec. 4.6.3, which will be discussed in more detail in this section. The reason for this faulty segmentation is that the segmentation algorithm is most likely not made for such scenarios where a person is lying in a bed, and a blanket has created movement in the whole bed.

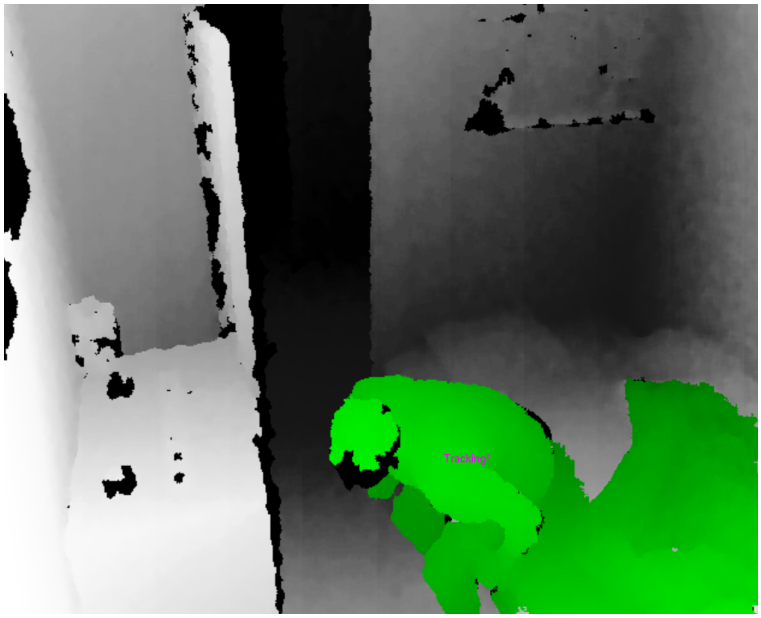


Figure 4.7.: Faulty segmenting when getting out of bed

This must be solved preferably by improving the tracking algorithm, or taking this into account in software. Furthermore, according to Microsoft, the precision of the Kinect 2.0 is far better, and the tracking algorithm is also greatly improved. Consequently, one could hope the Kinect 2.0 segmentation algorithm is able to segment such scenario better. If not, the problem of lying in bed must be considered.

4.5.2. Inactivity Detection

The inactivity detection scheme is hard to illustrate, seeing as there is a probability distribution connected with each of the many grid squares. However, it is possible to view some sample grid frames which can illustrate the approach and analyze its efficiency. As mentioned above, the durations a person is in a cell may in a more realistic scenario be significantly longer, but the sample frames can still show if the approach seems promising.

Current Cell Priority

An important part of the inactivity detection suggested is choosing cell size and slack distance. These are tuning parameters, and must be balanced; too large and a fall might occur in the same grid cell as an inactivity zone, too small and the inactivity zones might not be recognized seeing as the position will jump between cells. In the following analysis I have chosen grid size of 20 cm × 20 cm, and a slack distance of 5 cm. The choice is based on the observed variance when a person is sitting in a chair, in which the position estimate is varying with around 5 cm.

Scenario 1: Living Room

As mentioned above, some sample grid cells can be used to evaluate the results. The sample cells are marked with squares on top of the 2D tracks map, shown in Fig. 4.8, along with a closer look of the cells. The cell border and the slack distance is also shown. Cell 1 is on one of the chairs, whereas cell 2 is in a typical activity area; a place where one only walks through. The number of registered durations is 11 for cell 1 and 59 for cell 2. The reason it looks less for cell 2 is because of the time resolution at 30 fps, which means that there are several measurements with the same time duration value.

The registered consecutive amounts of time spent in the two frames is shown in Fig. 4.8, along with estimated GMM pdf's for different K's defined in 4.2. It is clear that there is a significant difference between the two cells, both in expected value and variance. The chair zone (cell 1) durations range from 0.5 s to 53 s whereas activity zone (cell 2) durations range from 0.16 s to 1.4 s. Even though the values might be different in a more realistic scenario, it is clear that there are significant differences between inactivity zones and activity zones. Consequently, the inactivity detection scheme described above seems promising based on scenario 1.

The reason that the Minimum Description Length defined in 4.4 has been omitted above is because this method works better with a larger number of observations. However, ζ_{min} was found when $K = 2$ for both distributions, which seems correct intuitively.

Scenario 2: Hallway

The same approach as in scenario 1 is employed; choosing sample frames for analysis shown on top of the 2D tracks map. Seeing as this is a hallway, there are no clear inactivity zones. Consequently, only one cell is chosen. The number of registered durations is 24, and the points are shown in Fig. 4.9. The response is similar to cell 2 in scenario 1, and this is a typical activity zone.

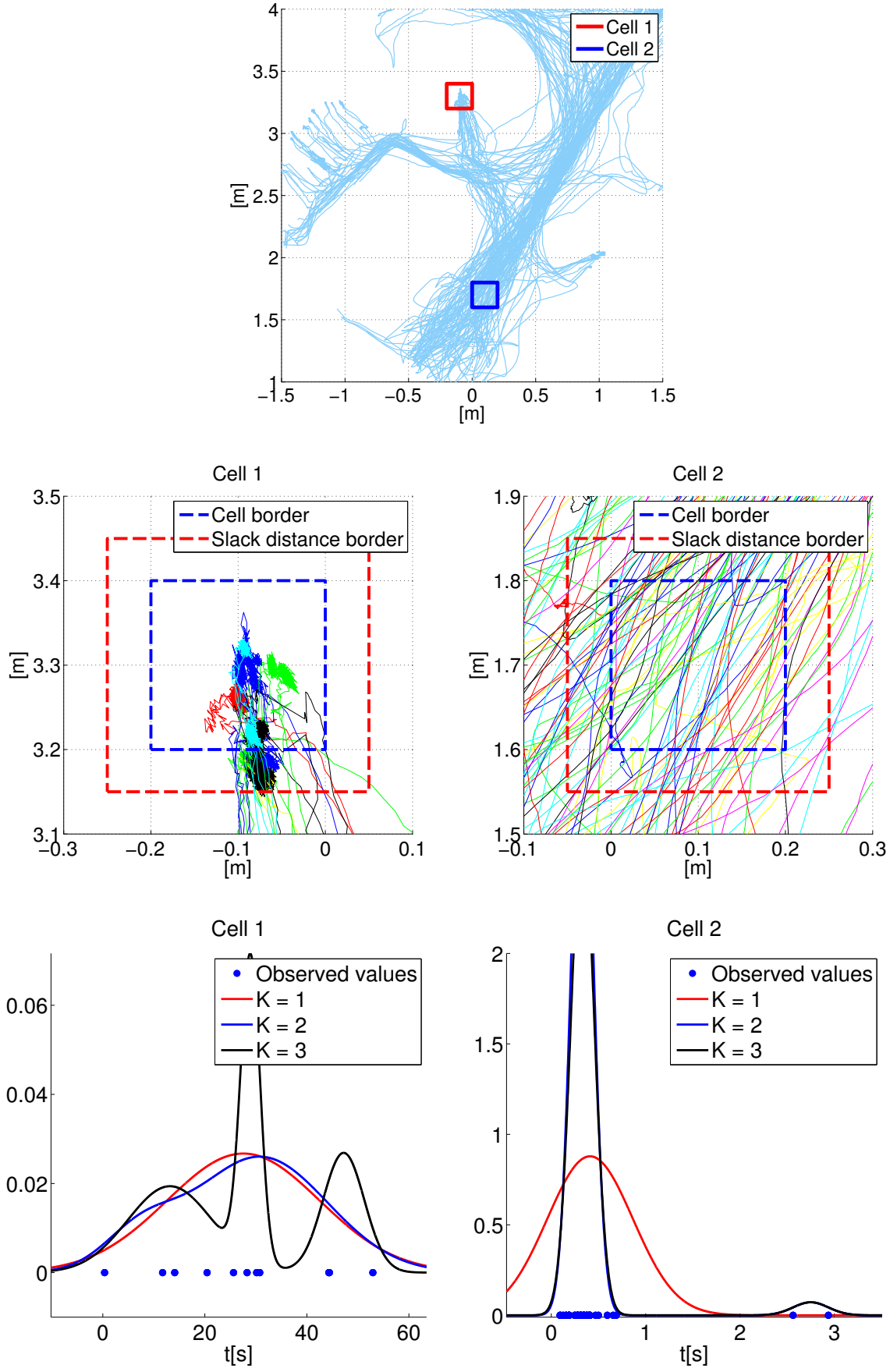


Figure 4.8.: Living room inactivity samples with generated pdf's

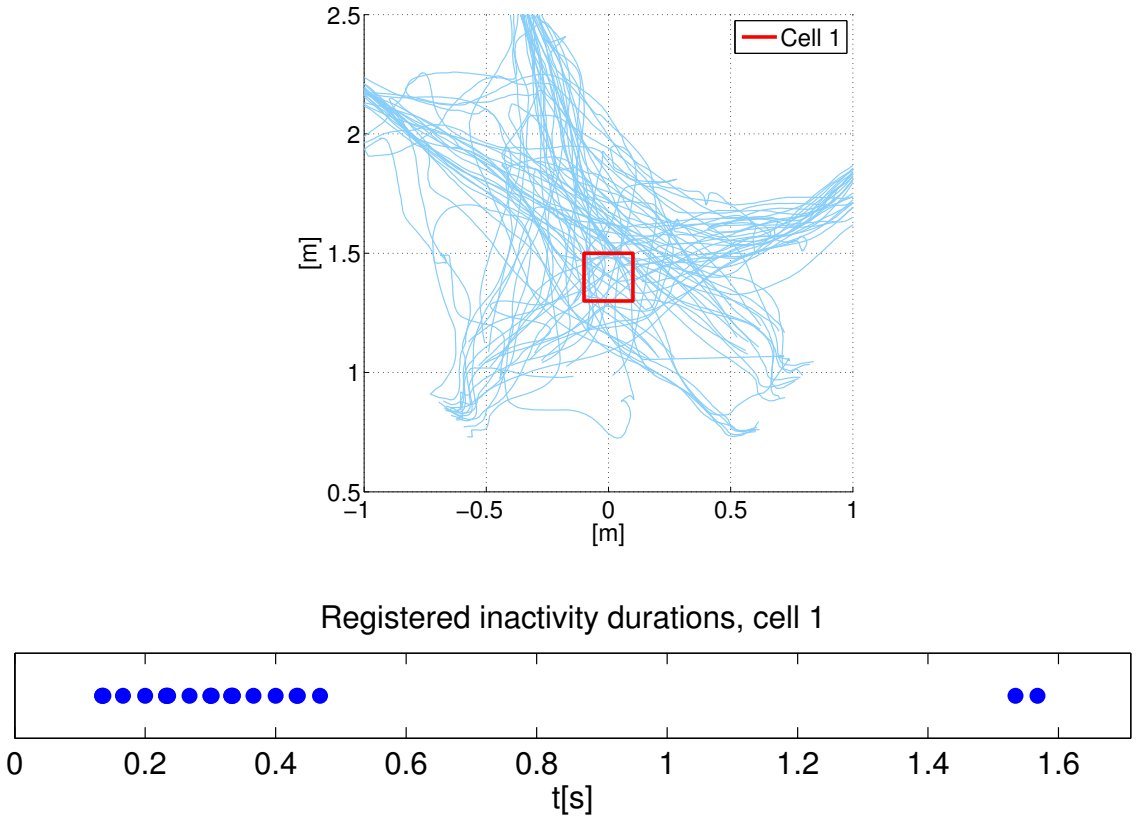


Figure 4.9.: Hallway inactivity sample

Scenario 3: Bedroom

Because this is a bedroom, the most significant inactivity zone is the bed. Furthermore, some inactivity occurs when a person is standing next to the bed, undressing and getting ready for bed. Three sample frames are chosen based on this, shown in Fig. 4.10, and one can see that there are clear differences between the duration in the two inactivity zones and an activity zone.

4.5.3. Height Map

Grid size is also a question when it comes to the height map. Unlike inactivity detection, grids that are too small is not as crucial seeing as it is not accumulated time inside a grid cell that counts, a value is simply mapped to a cell. The most important downside of a small grid size will be the fact that some cells may have no measured heights at all. This can be handled by using the surrounding values, but still introduces uncertainties in the height map. Similar to inactivity detection, if the cell is too large a situation might occur where a fall happens within the same cell as a chair etc.

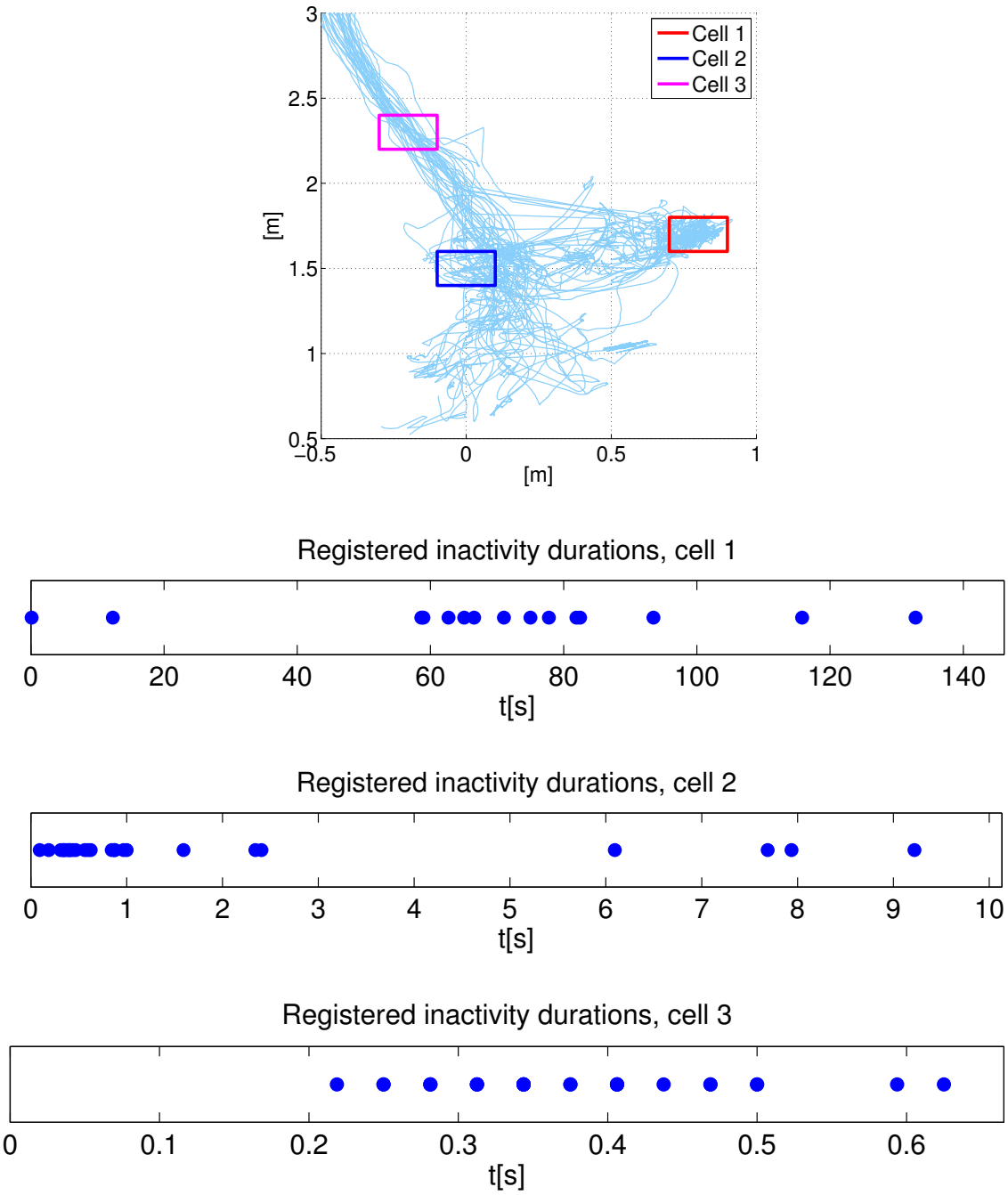


Figure 4.10.: Bedroom inactivity sample

Scenario 1: Living Room

As mentioned above, there will more blue squares for smaller cell sizes, meaning no height has been registered for the particular cell. The accuracy is of course better for smaller grid sizes, but needed accuracy is not necessarily high seeing as the important part is for example separating a fall next to a chair with a perrson sitting

in the chair. It seems that the cell size should be smaller than in the inactivity detection case, and 10 cm × 10 cm seems like a good place to start. The average heights for 10 cm × 10 cm cells are shown in Fig. 4.11.

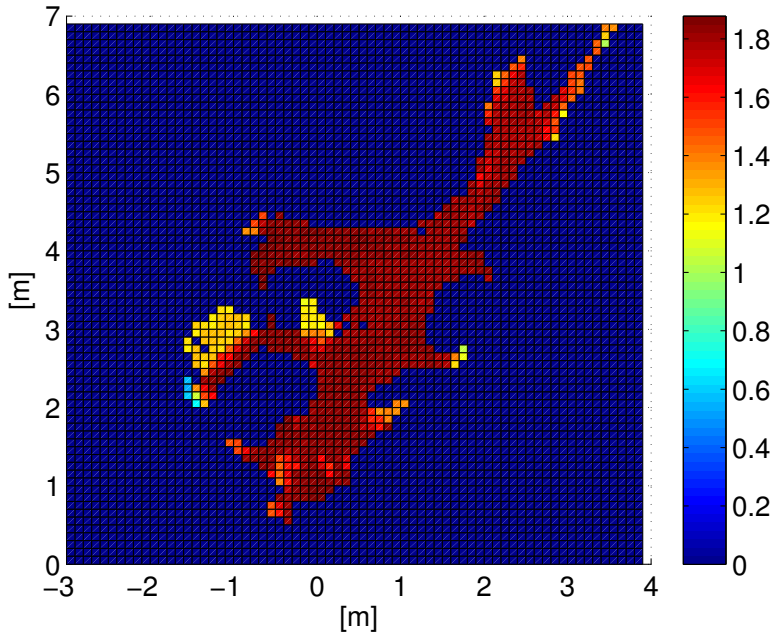


Figure 4.11.: Living room height map, in meters

It is clear that the heights registered are rather accurate, and the average value gives a good impression of the scenario; the chairs are easily seen as yellow zones, and the height is quite stable where the person walks around. The lower averages near the edges in the lower part of the map is due to the person bending down to view the computer recording the data.

It is also of interest to look at the distribution of height values, and not just the average. Two different cells are chosen, one with highly varying height values and one with more stable height values, shown in Fig. 4.12. The number of observations is 6095 for cell 1, and 1026 for cell 2. This is large compared to sec. 4.5.2, which is natural due to the fact that every time frame (at 30 fps) when the person is present results in a stored height. As mentioned in sec. 4.2 the amount of observations does not pose a problem because a histogram can easily be employed.

The observed values along with a GMM pdf with appropriate K are shown in Fig. 4.12. Also here both the variance and the expected values differ. The values in cell 1 are centered around 1.82 m whereas the values in cell 2 have two centers. This is due to the fact that the person is in the process of sitting down in this cell seeing as this cell is on the border of one of the chairs. The variance would most likely be smaller with smaller cell sizes, but it is apparent that even with larger cell sizes there is a clear spectrum of normal values for a given cell, and based on scenario

1 generating a height map seems promising for learning normal height values in a room and detecting deviations from this normal.

Scenario 2: Hallway

The height map for 10 cm × 10 cm cells is shown in Fig. 4.13. The reason for the lower average height on the right part of the height map is because the person is moving out of the Kinect view. As can be seen from the sample depth image in Fig. B.17, the view is skewed relative to the door opening on the edge of the depth image. This results in tracking a smaller and smaller part of the person when the person is exiting the view, and creates an unrealistic "normal height". Other than this weakness, it is clear that the height map also works in the a more difficult scenario, even when the whole person is not in view.

Scenario 3: Bedroom

The height map for 10 cm × 10 cm cells is shown in Fig. 4.14. It seems that the average values are more varied in scenario 3. This is natural; getting in and out of bed, bending down to pick up a book, undressing etc. leads to a less homogenous height map. The variance of registered heights in each cell is also significantly larger. Two sample frames are chosen, also shown in Fig. 4.14. The large variance of cell 2 is because the person is both standing up and bending down at this point. The large variance for cell 1 is less intuitive; the lowest points are below 1 m, and this must mean that the person is both standing and lying down on the bed when in this cell. This might be the case, but the tracker weakness mentioned in sec. 4.5.1 can also explain the large variance. Because both the person and the bed is segmented together as one person, the person height is large, whereas the 2D centroid is inaccurate leading to an inaccurate height map.

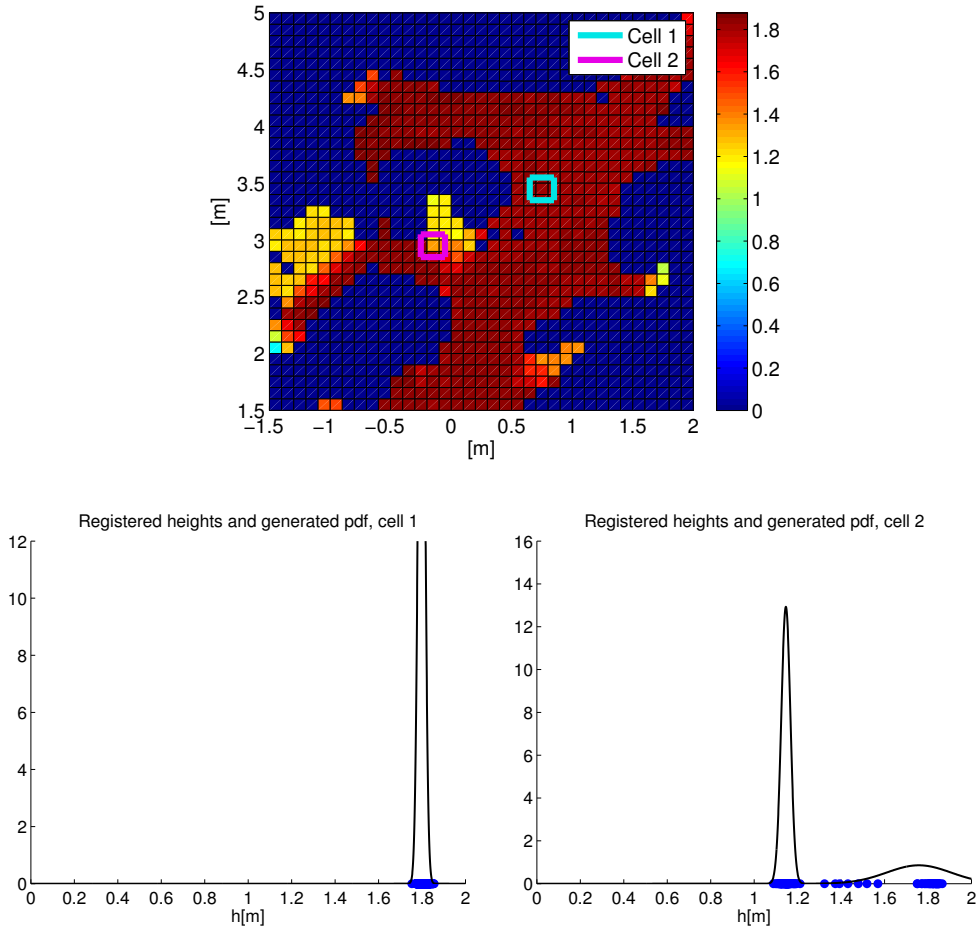


Figure 4.12.: Living room height samples

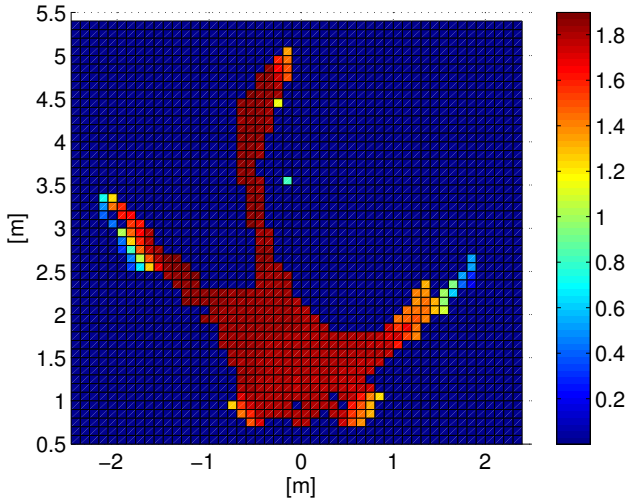


Figure 4.13.: Hallway height map, in meters

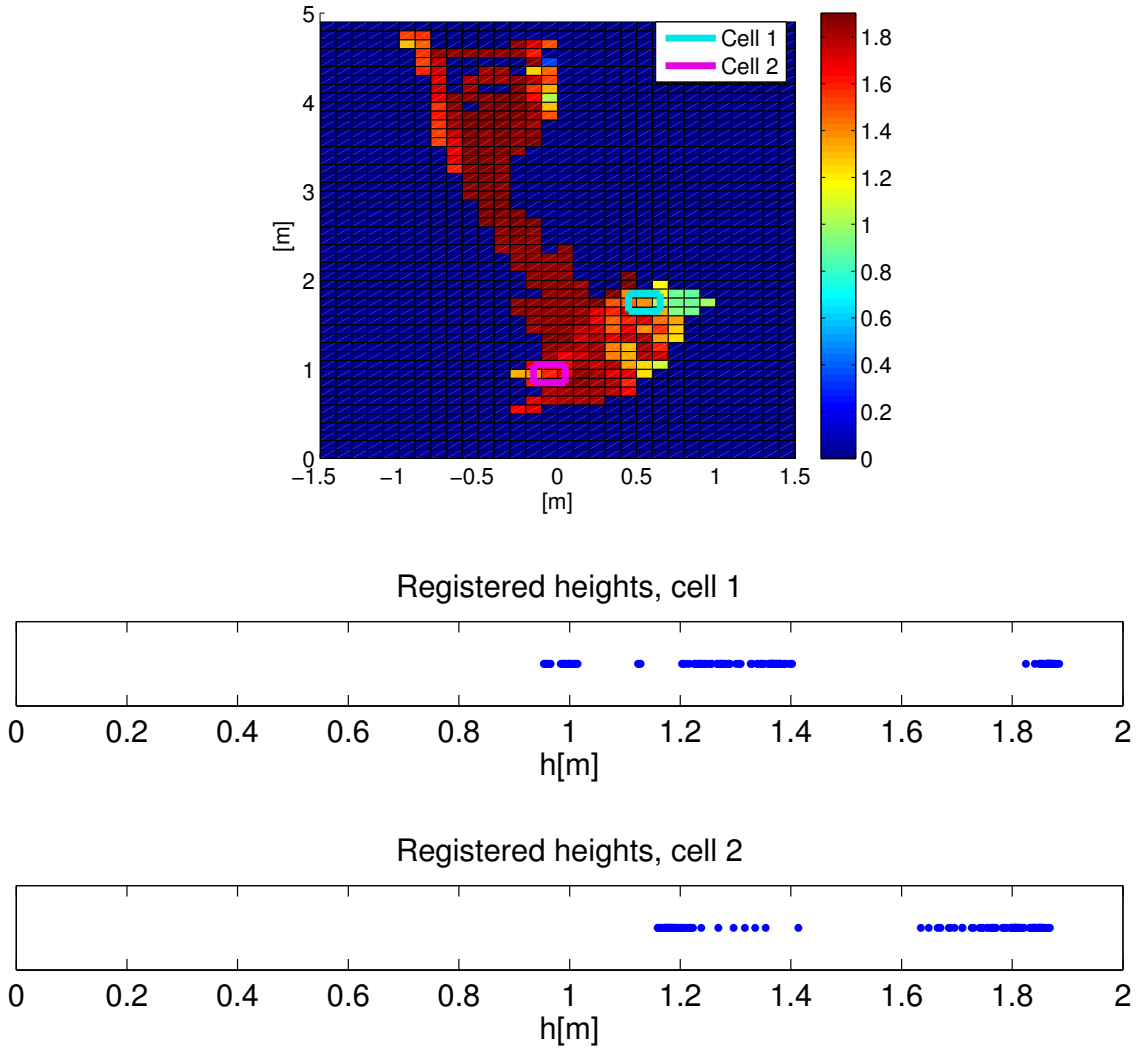


Figure 4.14.: Bedroom height map and samples

4.5.4. Wall Detection

As mentioned in sec. 4.3.2, it seems that storing all 3D points above a certain height threshold, T_h, will give a simpler extraction of walls. Naturally, walls will reach from floor to ceiling and furniture will not, creating a much simpler point image if only points above T_h are used.

Scenario 1: Living Room

It is of interest to compare using all values versus using only values above $T_h = 1.8\,\text{m}$. The results are shown in Fig. 4.15 and Fig. 4.16. Fig. 4.15 shows points used projected onto the floor with calculated walls, and Fig. 4.16 shows the Hough

transform result with the points as input, both iterations. It is clear that using only points above T_h yields far fewer points, and a larger fraction of the points are wall points. This can also be seen in the Hough transform images, especially after the most dominant wall has been removed. It seems that the walls are detected either way, but the most robust solution is apparent when only using values above T_h, at least in this scenario.

Scenario 2: Hallway

Because there are more door openings in scenario 2 compared to scenario 1, there are fewer points that should be associated with the wall, making the problem of wall detection more difficult. The number of values above T_h is roughly the same as in scenario 1, around 41000. Using all values and using only values above $T_h = 1.8\,\mathrm{m}$ is compared in Fig. 4.17 and Fig. 4.18. In this case, due to the reduced number of wall points, the walls are wrongly determined when using all points whereas using points above T_h results in correct detection of the walls. As can be seen in Fig. 4.18, the wall are two very clear spikes in the accumulator array for points above T_h. The conclusion from scenario 2 is that wall detection is possible, and it is best to only use points above a certain height threshold.

Scenario 3: Bedroom

In the bedroom there is only one wall. The Hough transform for points above $T_h = 1.8\,\mathrm{m}$ is shown in Fig. 4.19, and it is clear that the wall is detected easily by the very clear spike around the correct (r, θ).

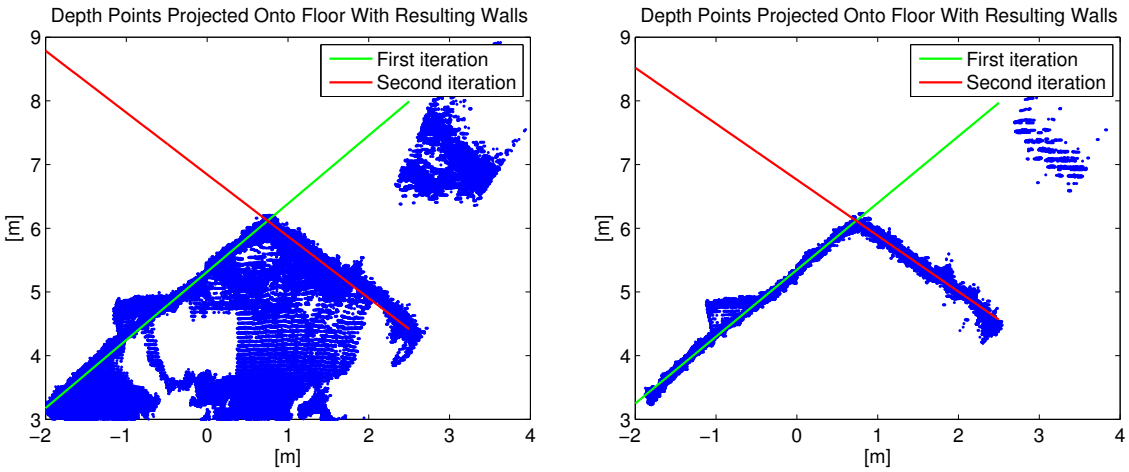


Figure 4.15.: Living room depth points projected onto floor, with resulting calculated walls. Left figure is all points, right is above T_h

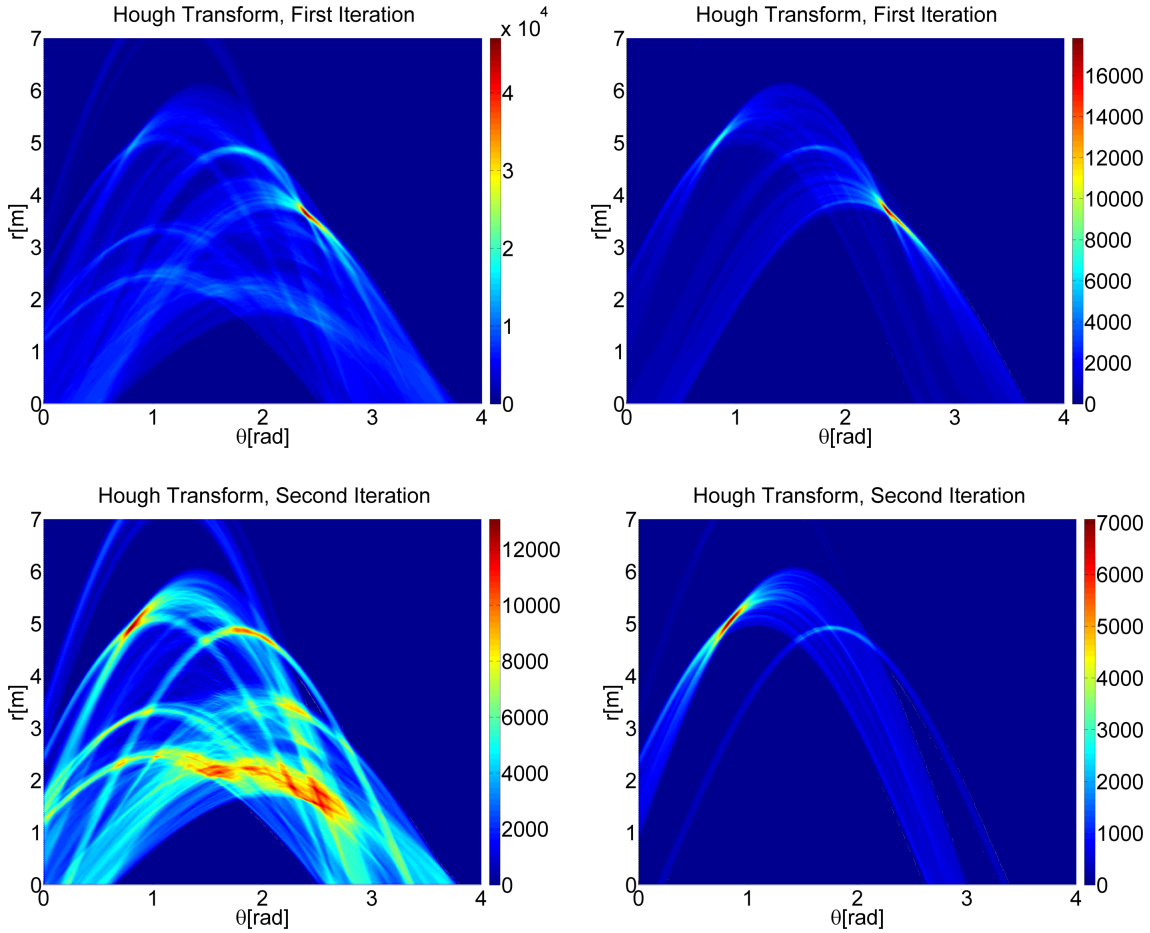


Figure 4.16.: Result of running Hough transform for living room depth points. Left column is all points, right is above T_h

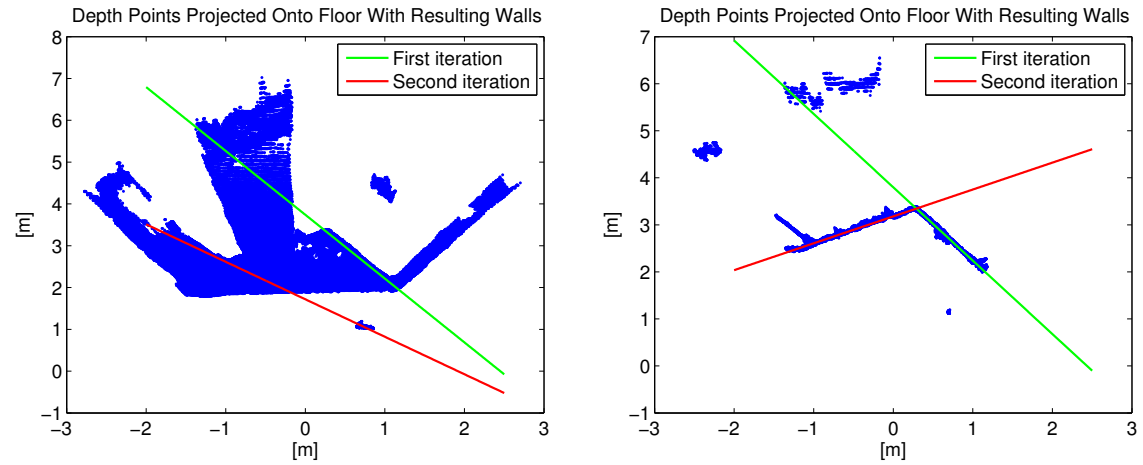


Figure 4.17.: Hallway depth points projected onto floor, with resulting calculated walls. Left figure is all points, right is above T_h

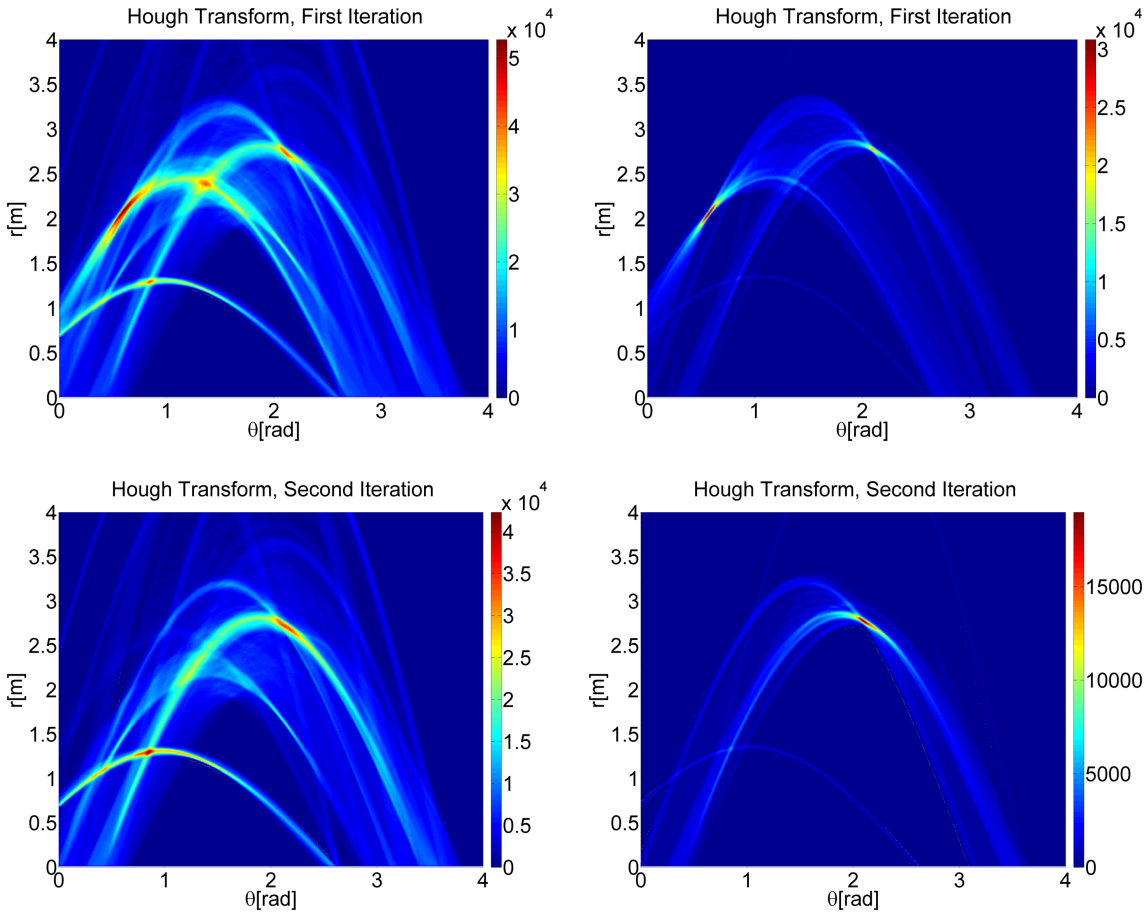


Figure 4.18.: Result of running Hough transform for hallway depth points. Left column is all points, right is above T_h

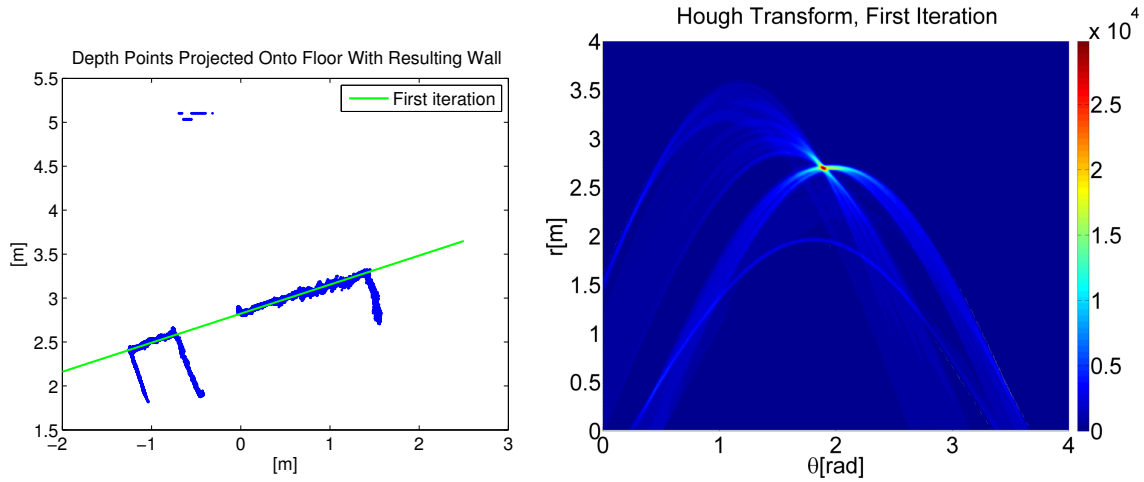


Figure 4.19.: Bedroom depth points above T_h, with resulting calculated wall and Hough transform

4.5.5. Entry/exit Analysis

Seeing as the tracker might take some time to recognize a person, it is of interest to compare using only exit points, to using both entry and exit points for determining entry/exit zones. This is because for exit points, the person is already tracked and therefore the exit point is when the person disappears. This is more accurate than the entry point, which is when the tracker starts tracking the person.

Scenario 1: Living Room

A comparison of using both entry and exit points, and using only exit points is shown in Fig. 4.20, where the points are emphasized. It is clear that using only exit points yields fewer points, but a more accurate representation of entry/exit zones; the points are more clustered, and more easily separated.

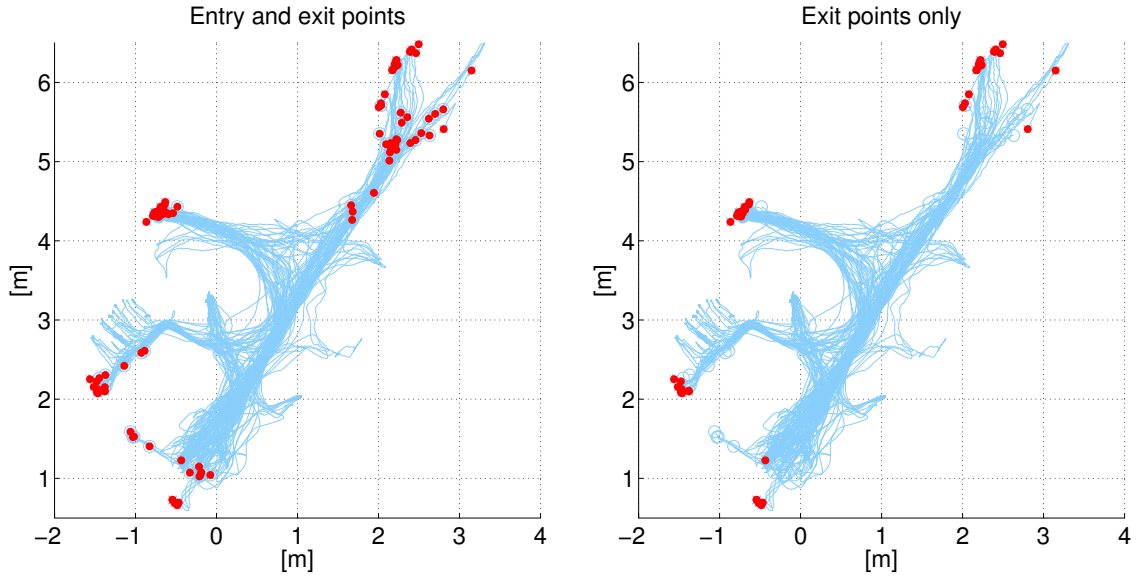


Figure 4.20.: Comparison of using both entry and exit points and using only exit points for entry/exit analysis

As discussed in sec. 4.3.2, if door openings are detected indirectly from wall detection, a more accurate approach can be taken; by generating exit points not when the person exits the tracker view, but when the person crosses the wall lines. As shown above in sec. 4.5.4, the wall is detected correctly. The generated exit points are shown along with cluster numbering using bisecting k-means clustering and the calculated wall from using points above T_h in Fig. 4.21. These are promising results; the door is apparent, and the exit points generate four separated clusters with small inter-cluster distances. By setting d_{max} in Algorithm 4.3 to 1 m the exit zones are easily clustered. The interval for d_{max} in which the clustering would be correct is

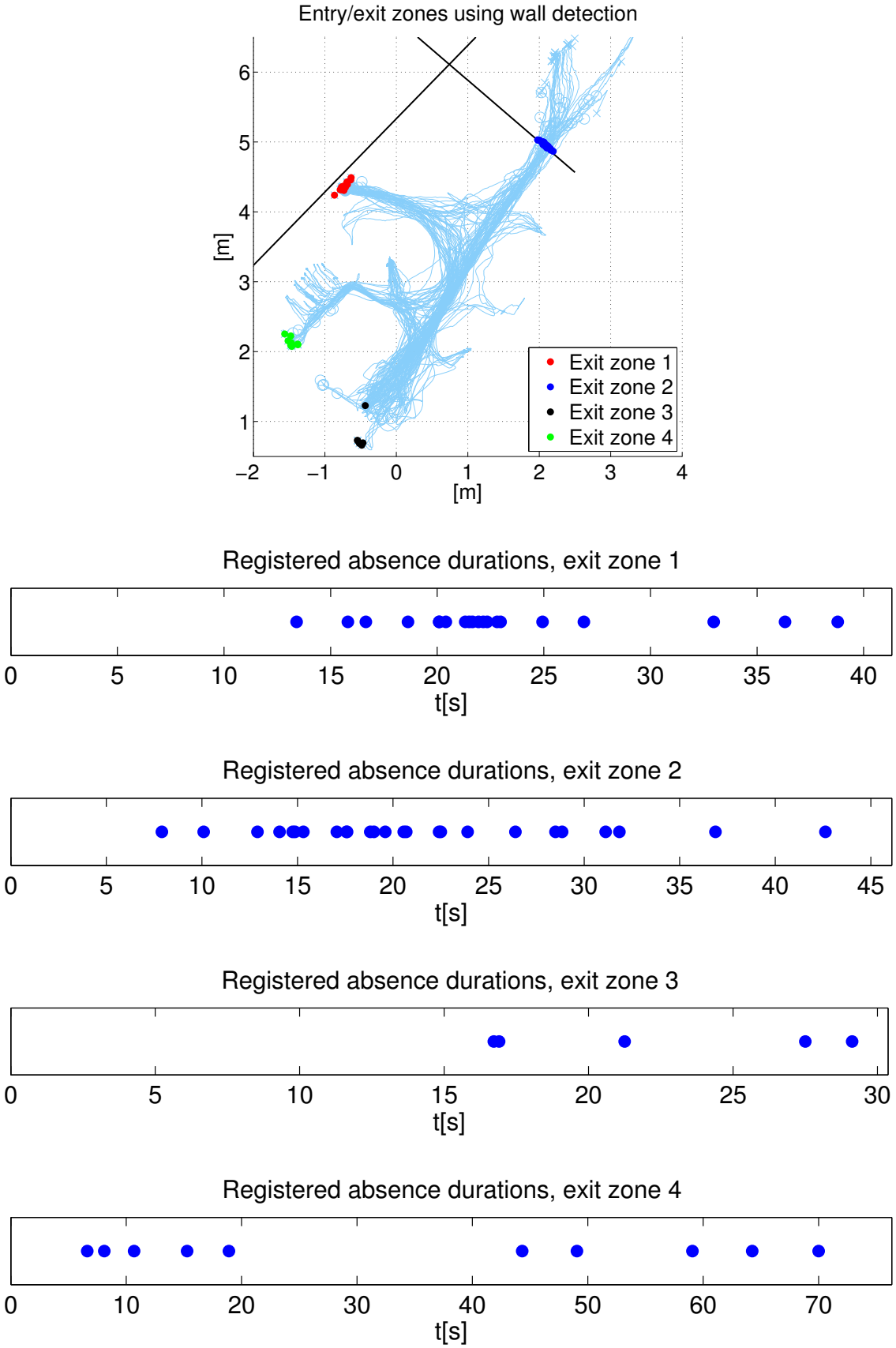


Figure 4.21.: Living room entry/exit analysis using wall detection with registered absence durations

large in this case, but in cases with doors close to each other, the interval giving correct clustering is smaller, and must be determined.

The registered time durations after exiting through an exit zone until a person is tracked in the room are shown in Fig. 4.21. One could also here generate GMM pdf's for the durations to give an estimated pdf of duration. It is clear that, given correctly defined exit zones, entry/exit analysis gives a way of analyzing the different exit zones and the absence duration associated with them.

Scenario 2: Hallway

As mentioned in sec. 4.5.1, the entry/exit analysis seems more difficult in scenario 2; there are more doors, and the doors are closer to each other. The generated exit points are shown along with cluster numbering using $d_{max} = 0.5\,\text{m}$ and the calculated wall from using points above T_h in Fig. 4.22. Once again, the results are promising. The number of doors is recognized, and even though only 3 out of 5 doors are in the Kinect view, the remaining doors are recognized based on the 2D points where the person exits the view. In other words, even if the doors are out of view, if the view is close enough to the doors, a person will exit the view at different places when going through different doors.

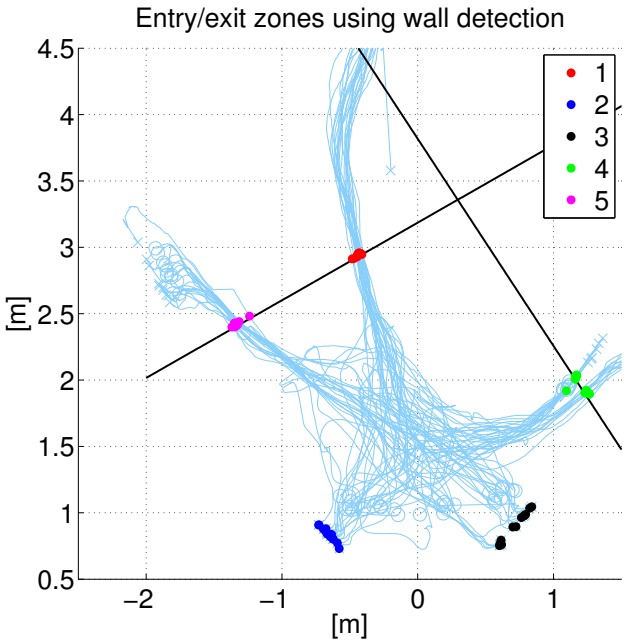


Figure 4.22.: Hallway entry/exit analysis using wall detection

Scenario 3: Bedroom

There are two entry/exit zones in scenario 3; the door and the area where the person is bending down out of Kinect view. These are easily clustered, seeing as the distance between them is large, and the inter-cluster distance is small. The clusters are shown in Fig. 4.23, along with the wall calculated from the Hough transform.

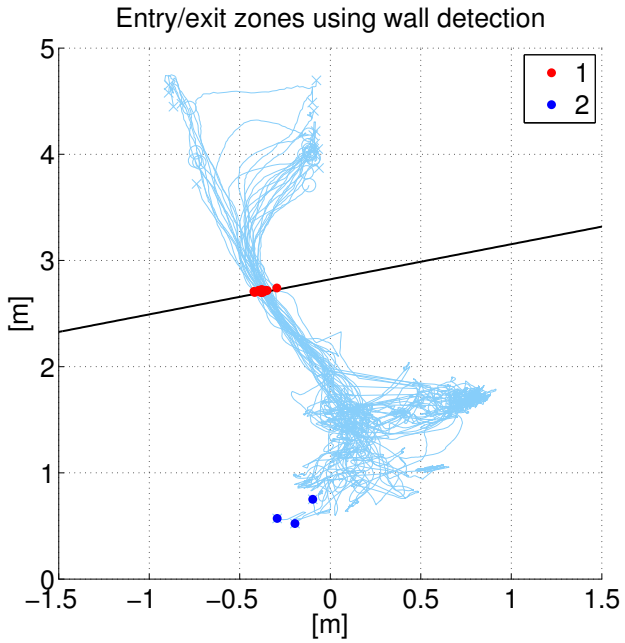


Figure 4.23.: Bedroom entry/exit analysis using wall detection

4.5.6. Summary and Conclusion

It seems that the scenarios leave using all of the more advanced parameters with promising results. It must be accounted for that the movement recorded is realistic when it comes to walking speed, inactivity zones and entry/exit zones, but unrealistic when it comes to duration in the inactivity zones and duration away from the room. The need for more realistic testing is apparent, seeing as a training set in which the person is not thinking about creating a training set might yield different results. It might take longer to have enough data seeing as persons mostly sit in their chairs/couches, but this must be determined by using a realistic training period.

The techniques suggested are, of course, dependent on a functioning segmentation algorithm. However, it seems the algorithms suggested has some robustness, seeing as the map is learned over time, giving little weigth to a few faulty segmentations in the long run.

Inactivity Detection

Given that the training set used is realistic enough, it seems that inactivity detection is successfully able to separate activity zones from chairs, beds etc. The size of each cell and slack radius must also be reconsidered in a more realistic scenario, but the principle seems functional.

Height Map

The height map learning is succesful in all scenarios, but it seems bedrooms and places with a smaller floor space will give larger variances. However, this should not be a problem if one is able to combine the height map with other features.

Entry/exit Analysis

It seems that the entry/exit zones are clearly defined, especially combined with wall detection. Using this for direct fall detection will be discussed in the following section, and if a more realistic training set yields entry/exit zones as clear as in the scenarios analyzed above, entry/exit zones might prove useful.

Absence duration analysis is more of an uncertain concept which requires more testing. There must definitely be an upper variance in which the absence duration is omitted from fall detection, taking into account doors which lead out of the living area; a person should be able to go on a vacation, a long visit etc. It seems an upper duration limit could be set, to focus on rooms with only one entry/exit, such as most bathrooms and many kitchens. Of course, the time before unusual activity is detected is longer when the person is not in view, but regardless it seems better than no detection.

4.6. Testing and Results - Fall Detection

After the system has had a training period, it is natural to look at falls, and how these compare to the normal behaviour registered by the system. After all, this is the purpose of the system; to detect falls. Five falls are analyzed for each scenario, and the goal is that these falls will shed light on some strengths and weaknesses of the suggested focus features even though a much larger number of sample falls must be gone through to examine robustness of the system. Taking the results of sec. 3.4 into account, $\hat{\dot{z}}_{max_{KF}}$ will also be investigated, seeing as this was the only feature (along with $\hat{z}_{max_{KF}}$ if a height map is present) that gave promising results. For simplicity, $\hat{h}$ and $\hat{\dot{h}}$ will be used, instead of $\hat{z}_{max_{KF}}$ and $\hat{\dot{z}}_{max_{KF}}$. Consequently inactivity, height map and $\hat{\dot{h}}$ will be investigated for each fall.

The reason absence duration analysis is omitted is because this is intended for detecting falls in rooms which are out of Kinect coverage, and would presumably work as long as the entry/exit zones are segmented correctly, which is analyzed in the section above.

To reduce the amount of plots, graphs will be omitted if the reponse is similar to another plot; the reponse may not necessarily be equal, but similar and the same conclusions can be drawn. Sample frames, fall tracks etc. can be found for all falls on the enclosed cd.

Two sample depth image frames from each fall can be seen in sec. B.3.

4.6.1. Scenario 1: Living Room

Fall 1

Fall 1 is a forward fall after walking towards the camera, with the Kinect having a full view of the fall. The track generated by the fall is shown in Fig. 4.26 along with the placement of sample cells. Furthermore, a sample cell is shown for both inactivity detection and height map in Fig. 4.26. It is clear that there are significant differences both in inactivity and height.

The estimated person height and height derivative is also shown in Fig. 4.26. The fall happens around $t = 1\,\mathrm{s}$ and is easily recognized as a sudden decrease in height and a large, negative velocity in z-direction.

Fall 2

Fall 2 is a forward fall to the knees after walking away from the camera, with the Kinect having a full view of the fall. When it comes to sample frames and generated height map, the response is similar to fall 1, and one can easily see the difference between the fall and the normal behaviour.

Because the person first falls to the knees, $\hat{\dot{h}}$ is far smaller than in fall 1, with a maximum of around $-0.85\,\mathrm{m/s}$ compared to a maximum of around $-1.5\,\mathrm{m/s}$ for fall 1. This is shown in Fig. 4.24. The fall is still recognizable, but the velocity spike is smaller.

Fall 3

Fall 3 is a backwards fall ending sitting after walking perpendicular relative to the camera, with the Kinect having a full view of the fall. When it comes to sample frames and generated height map, once again, the response is similar to fall 1, and one can easily see the difference between the fall and the normal behaviour. When it comes to fall 3, $\hat{\dot{h}}$ is large, similar to fall 1.

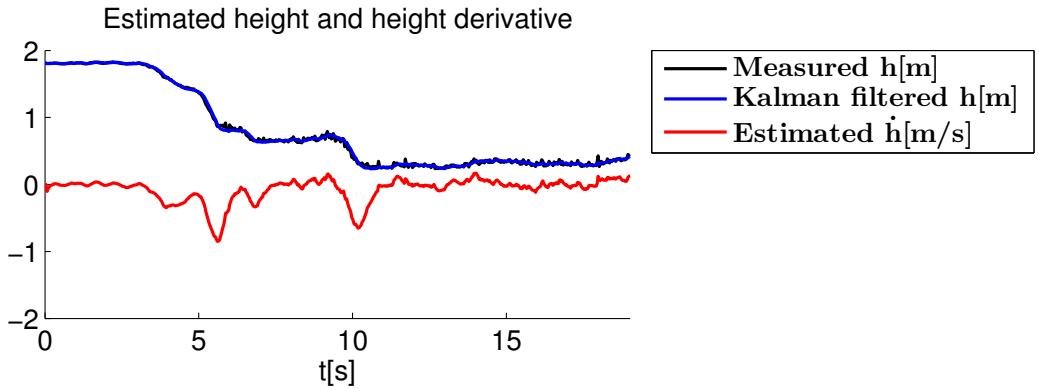


Figure 4.24.: Scenario 1, fall 2 estimated height and height derivative

Fall 4

Fall 4 is a more complicated scenario; the person is first sitting in a chair for a while, and when the person is getting up, the legs fail, and the person falls to his knees. The person is facing the Kinect when sitting in the chair. The track generated by the fall is shown in Fig. 4.27. Two sample frames are chosen for both inactivity and height, one for the chair area and one on the floor where the person is located after the fall. When the person is sitting, both height and inactivity duration is within the normal behaviour, but when the person has fallen, both properties suggest abnormal behaviour.

$\hat{\dot{h}}$ is shown in Fig. 4.25. It has two clear spikes, one when sitting down and one when the person is falling. This suggests that also normal behaviour can create a large negative $\hat{\dot{h}}$, which in turn suggests that $\hat{\dot{h}}$ alone runs the risk of false positives.

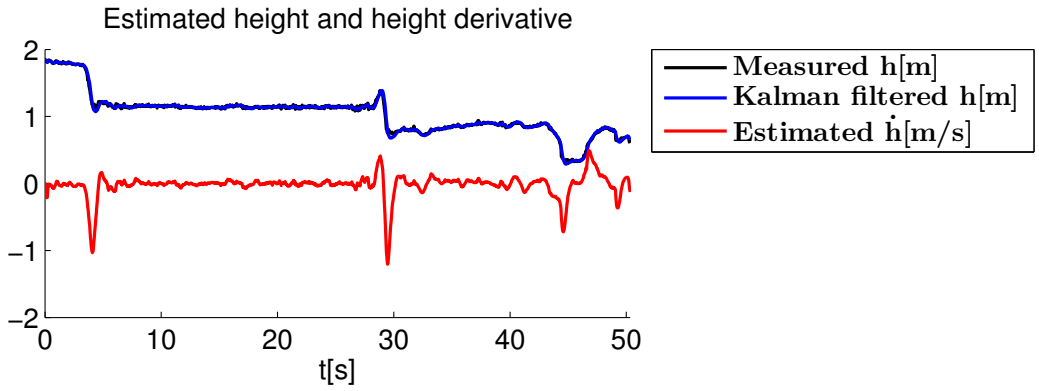


Figure 4.25.: Scenario 1, fall 4 estimated height and height derivative

Fall 5

Fall 5 is analyzed to show one of the weaknesses of the suggested features. The person is walking perpendicular to the camera, and falls behind a chair, resulting in a lost track. The track generated by the fall is shown in Fig. 4.28. When the person is lost during the fall, inactivity detection will clearly fail. Rougier et al.[18] suggests using $\hat{\dot{h}}$ before the occlusion in order to classify a fall, and Kepski and Kwolek[16] suggest using an additional accelerometer to better handle occlusions. An additional source for information in the case of this thesis is the height map; depending on the height of the occlusion, the person will have a height lower than the height map suggests, right before the whole person is occluded. To illustrate this, a sample frame from the height map is also shown in Fig. 4.28. It is clear that a few frames right before the person is lost are far below the normal height.

By looking at Fig. 4.28, it is also clear that there is a spike in $\hat{\dot{h}}$ right before the track is lost. This is unusual activity, and also a sign that a person has fallen. Furthermore, by looking at the track generated, the point where the track is lost is some distance (around 0.7 m) away from the exit zone. From eq. 4.11 we have that $k = 17.8$, which is rather large, and therefore losing the person at this point should be considered unusual. This can be seen from the ellipse describing covariance of the nearest exit zone, also shown in Fig. 4.28.

There are now three parameters that suggests a person has fallen right before the total occlusion occurs; the height is far below normal height indicated by the height map, $\hat{\dot{h}}$ spiked right before losing the person and the 2D position where the person is lost seems unlikely based on previous exits.

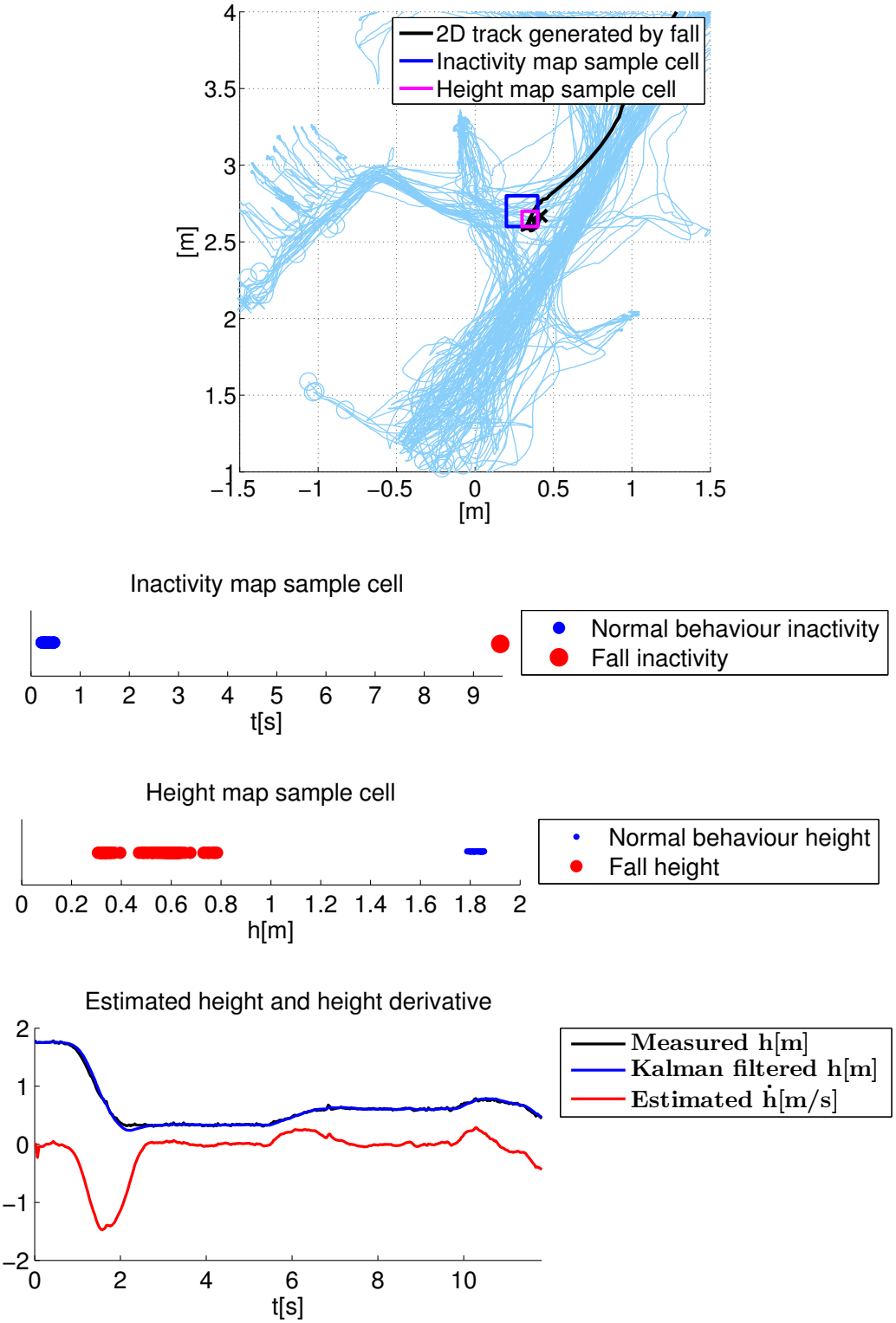


Figure 4.26.: Scenario 1, fall 1 track, sample map frames and height estimate

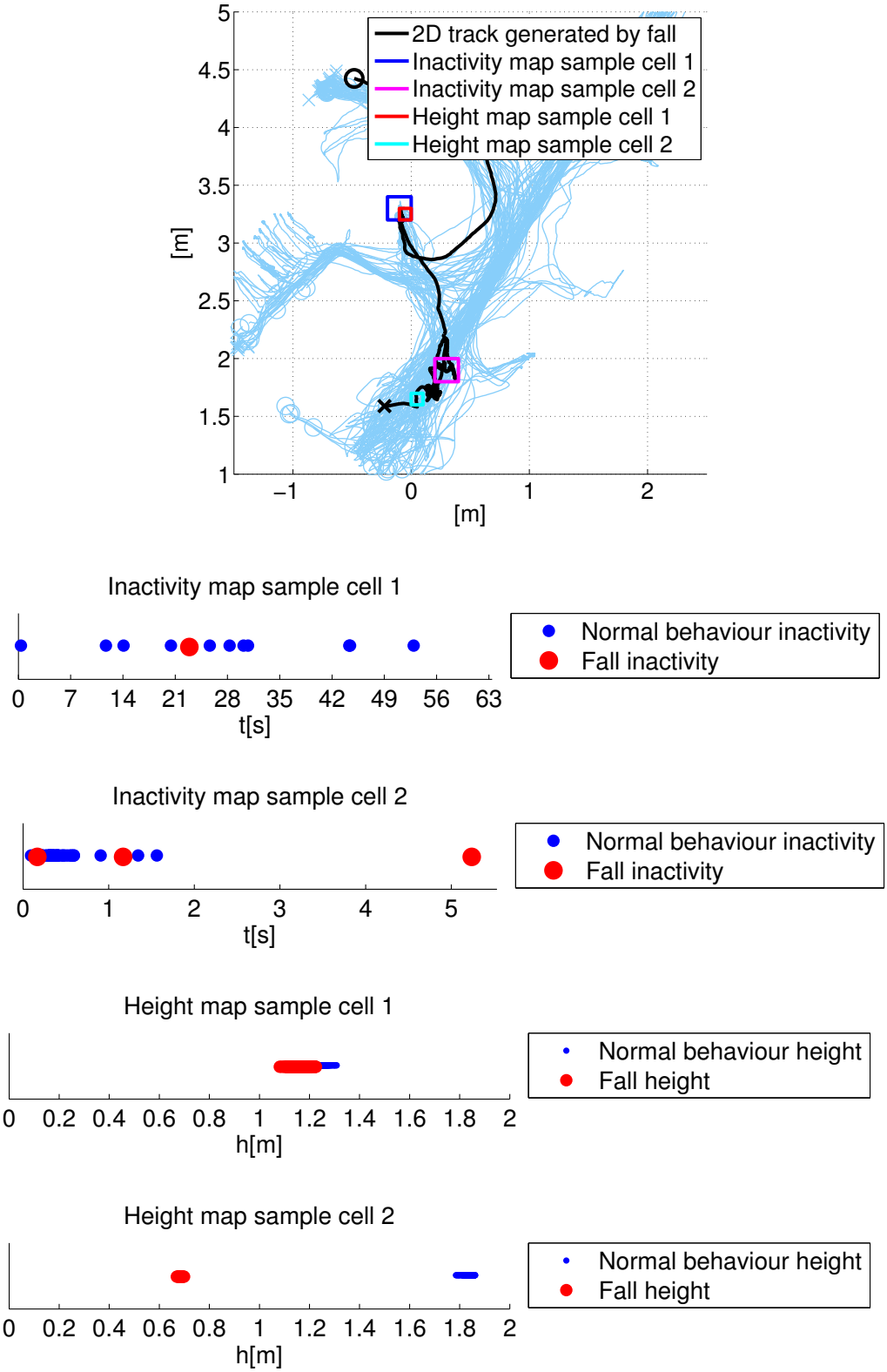


Figure 4.27.: Scenario 1, fall 4 track and sample map frames

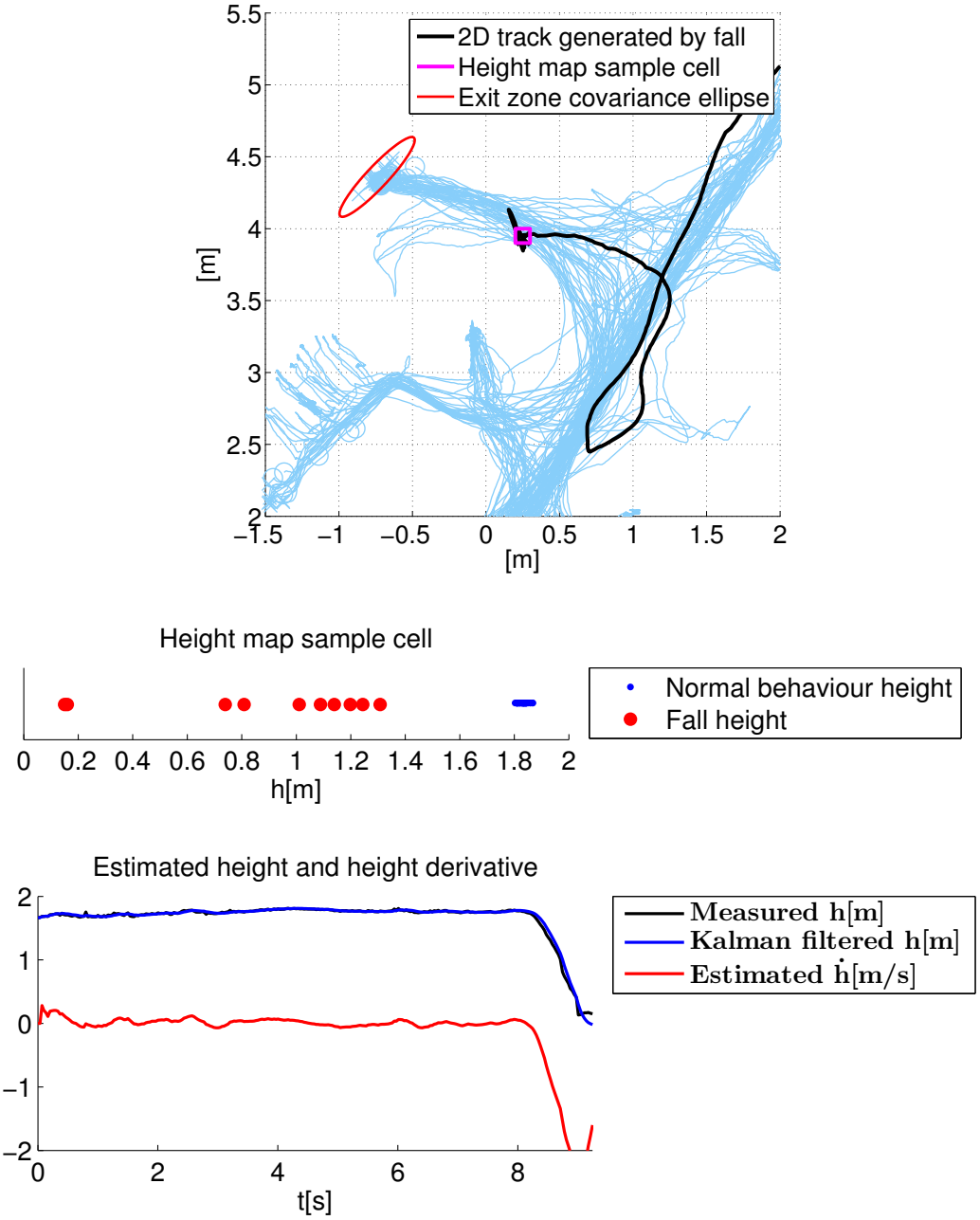


Figure 4.28.: Scenario 1, fall 5 track,sample map frame and height estimate

4.6.2. Scenario 2: Hallway

As mentioned, the area of the hallway is far smaller than in the living room. As a result of this, none of the falls occur with the Kinect having a full view of the fall.

Fall 1,2,3

As can be seen from the sample depth frames in sec. B.3, the whole person is not in the view of the Kinect at all times, especially after the fall has occured. However, because neither the 2D position nor the person height are dependent on viewing the whole person, the response is still similar to fall 1,2 and 3 in scenario 1. More details can be found on the enclosed cd.

Fall 4,5

Fall 4 and 5 are more difficult to detect. In both cases the person is falling out of the Kinect view, and through one of the doors in the hall. A sample frame from each height map is shown in Fig. 4.29 and Fig. 4.30, along with $\hat{\dot{h}}$ for both falls. The same indications as in fall 5, scenario 1 are present; velocity spike, lower height than normal. However, the velocity spike is smaller in magnitude, the height difference is smaller and the exit point is far closer to trained exit points than in scenario 1; $k = 14.9$ and $k = 4.45$. Fall 4 and 5 can be considered difficult cases. The signs are still there, but they are less clear, and one must consider robustness issues and the desire for as few false positives as possible. However, as will be suggested in further sections, some fall detection criterions can be employed only when the person is lost out of Kinect view.

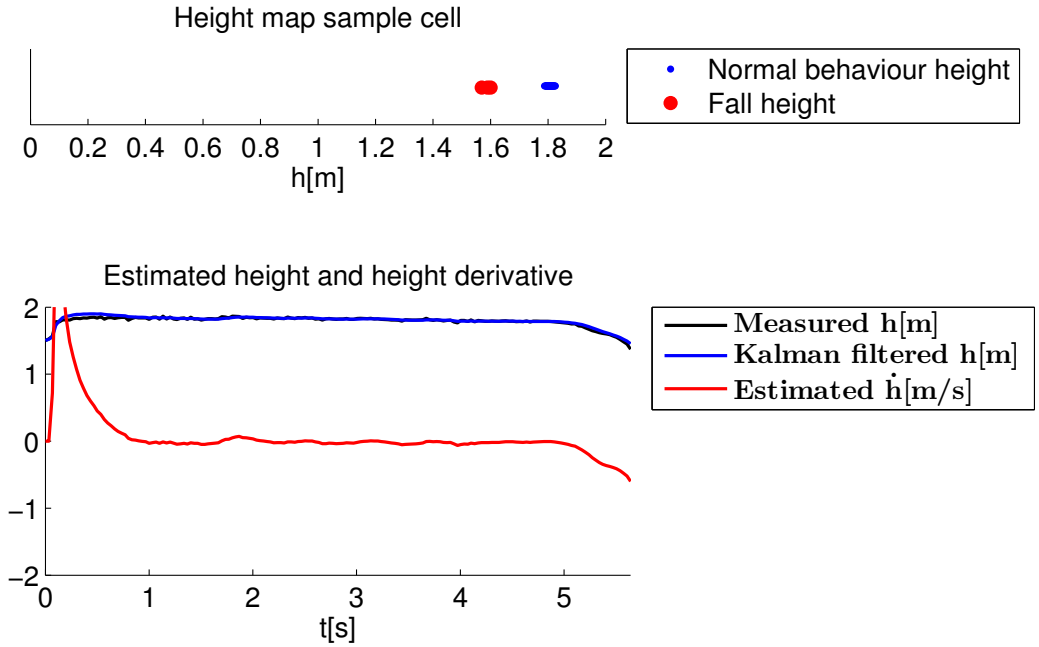


Figure 4.29.: Scenario 2, fall 4 sample map frame and height estimate

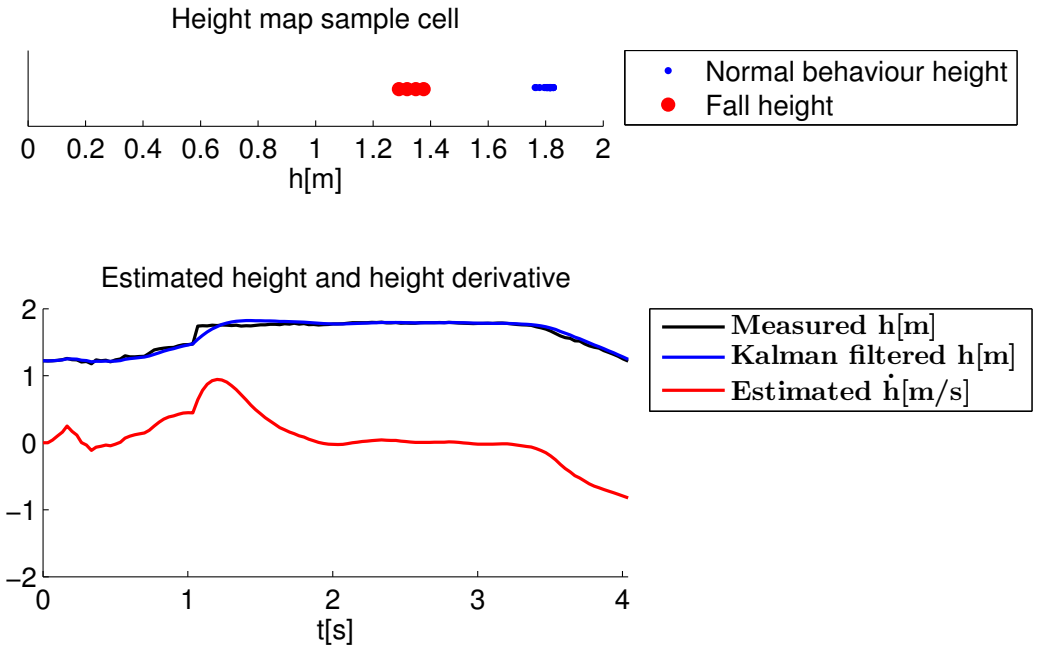


Figure 4.30.: Scenario 2, fall 5 sample map frame and height estimate

4.6.3. Scenario 3: Bedroom

There are several challenges for a small space with a lot of occlusions such as the bedroom in scenario 3. Both falling out of Kinect view, and the problem of segmenting a person from the bed discussed in sec. 4.5.1 are situations which makes fall detection more difficult.

Fall 1

Fall 1 is a regular fall to the knees with the person walking away from the camera heading out of the door after sitting a while on the bed. Both height and inactivity deviates from the maps, and $\hat{\dot{h}}$ has a clear spike, as in previous scenarios. This fall is deviating significantly from normal behaviour.

Fall 2

In fall 2 the person is sitting on the bed, and as the person is getting up, the legs are failing and a fall occurs. The track generated by the fall is shown in Fig. 4.31 along with the sample frames chosen from inactivity- and height maps. Estimated height and $\hat{\dot{h}}$ is also shown in Fig. 4.31, and it seems that normal behaviour such as sitting down on the bed create spikes of the same magnitude as the fall, happening around $t = 14\,\mathrm{s}$. The two positive spikes are because the person is lifting the upper

body using the arms, as might happen in a real situation. Once again, the fall is deviating from normal behaviour.

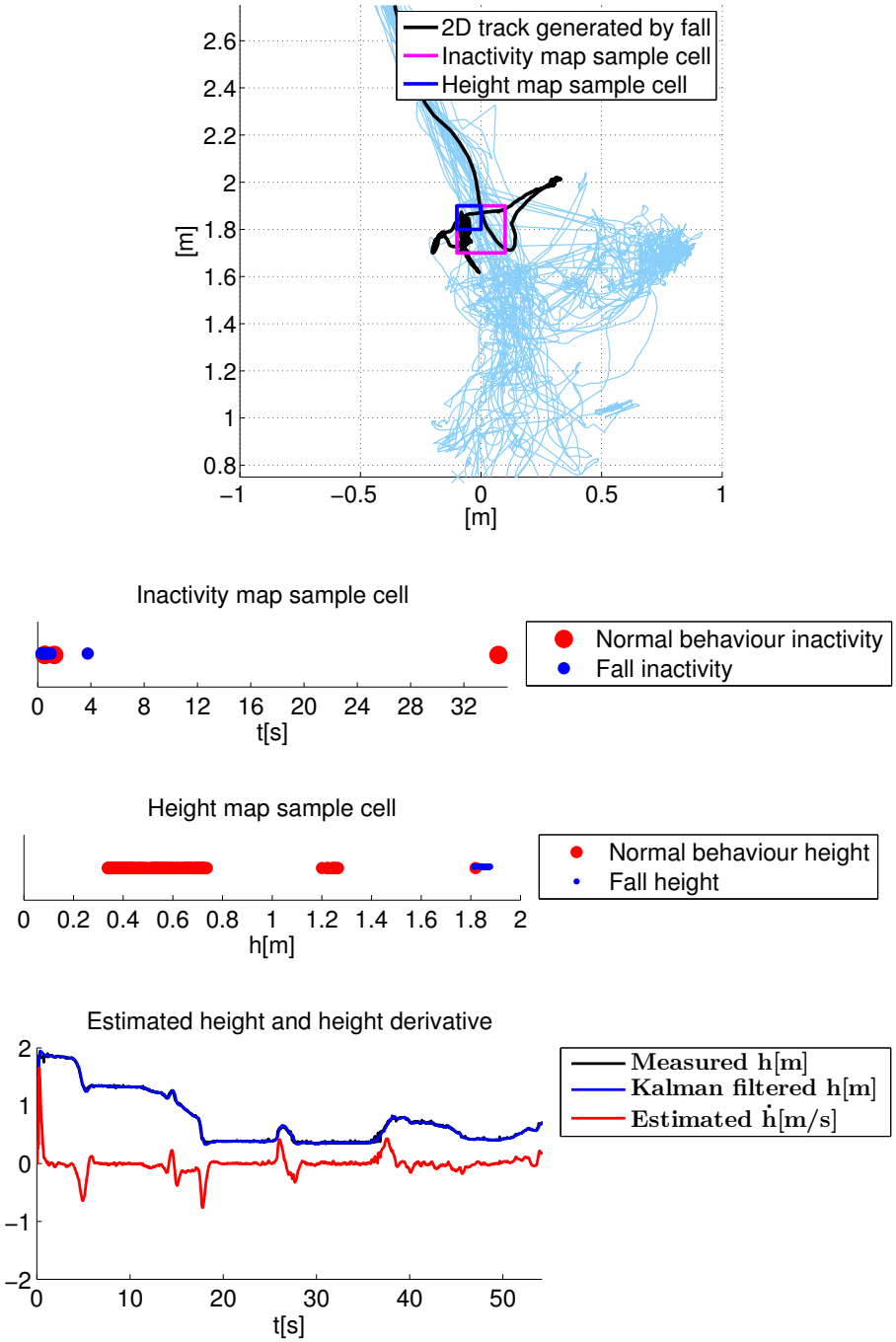


Figure 4.31.: Scenario 3, fall 2 track, sample map frames and height estimate

Fall 3

Fall 3 investigates the problem of incorrectly segmenting the person and the bed as one person. The track generated by the fall is shown in Fig. 4.32 along with the sample frames chosen from inactivity- and height maps. The estimated height and $\hat{\dot{h}}$ is also shown in Fig. 4.32. It is clear that because of the segmenting error, this fall generates data much closer to normal behaviour. The fall occurs around $t = 46\,\mathrm{s}$, and although there is a spike in $\hat{\dot{h}}$, this does not deviate much from the spikes generated from lying down in the bed and noisy segmentation. The registered heights are lower than the normal behaviour, but the difference is not as large as in more clear falls. In contrast, the inactivity duration is much larger than normal behaviour. This is because the person "draws" the 2D position away from the bed.

Whether or not all falls of this kind would be detected is unclear. When a person exits the bed, the segmentation is rather unpredictable at the moment, and even though it seems that the example fall deviates enough to detect it, this might not happen consistently.

Fall 4

Fall 4 is similar to fall 2; a person is sitting on the bed, and falls while getting up. However, the person is falling partially out of Kinect coverage resulting in a track loss a few seconds after the fall has occured. Because the person track is lost, inactivity detection will not yield abnormal responses. A sample height map cell, the estimated height and $\hat{\dot{h}}$ is shown in Fig. 4.33. Both of these parameters suggest a fall has occured, and $k = 6.75$. In other words, as long as the fall is captured, losing the track after will only cause inactivity detection to malfunction. A conclusion can be drawn that extra functionality must be added when a track is lost or that inactivity detection must not be a parameter overriding the other two, seeing as fall 4 would not be detected if this is the case.

Fall 5

The case of a person falling out of view is once again analyzed in fall 5. The person is walking towards the Kinect, falling backwards out of Kinect view and ending sitting. The track generated by the fall is shown in Fig. 4.34 along with an ellipse describing the variance of the nearest entry/exit zone. Inactivity detection will not be useful in this case. The heights registered at the sample cell are shown in Fig. 4.34 along with $\hat{h}$ and $\hat{\dot{h}}$. We can see that the registered height values are lower for the fall, but not a lot lower. The same goes for $\hat{\dot{h}}$; it is negative, but the spike has not reached the minimum when the person falls out of view. $k = 5.82$. Once again, it seems that a special fall detection criterion must be employed when the person is lost during the fall.

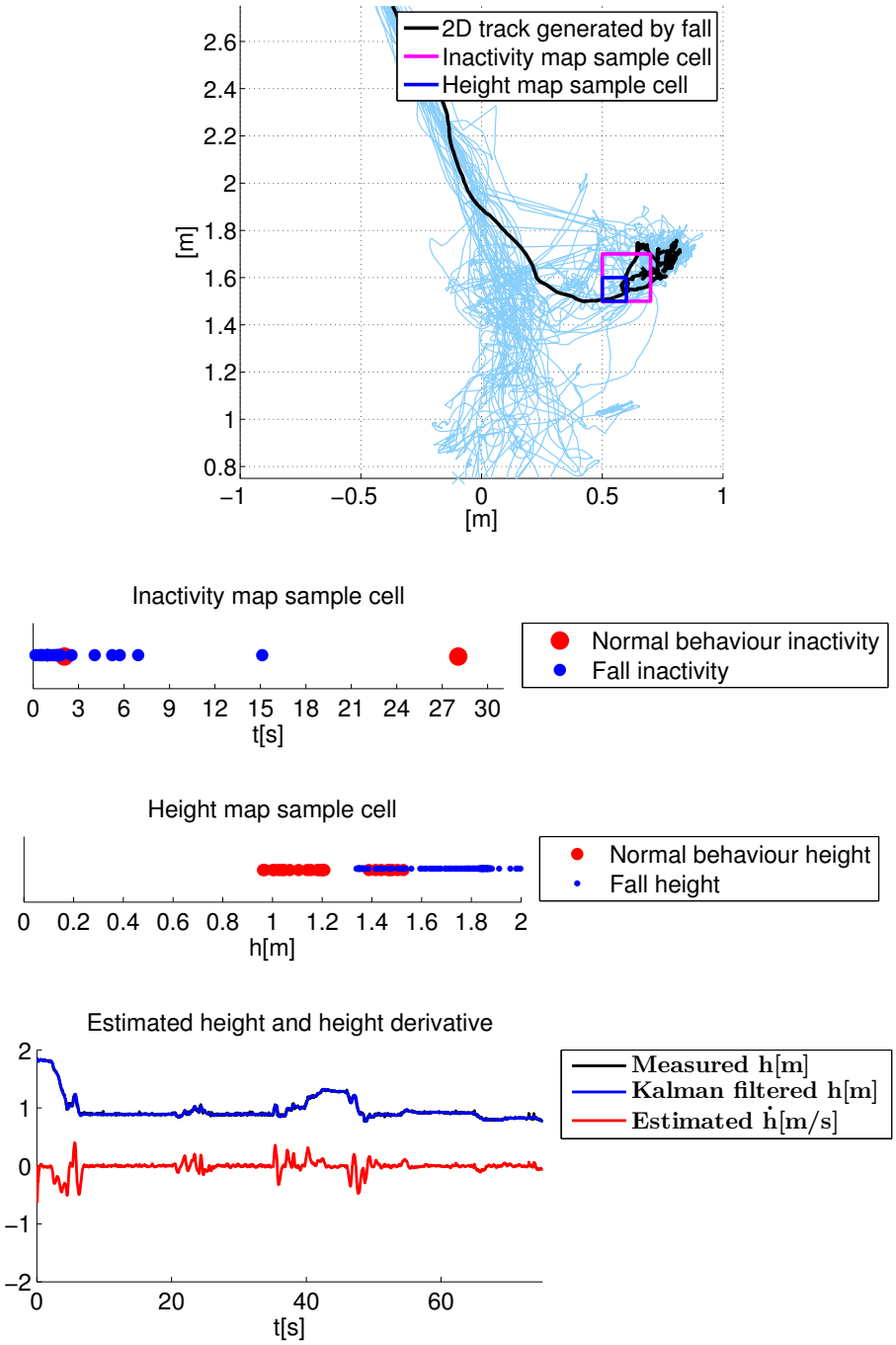


Figure 4.32.: Scenario 3, fall 3 track, sample map frames and height estimate

4.6.4. Summary and Conclusion

It seems that "regular falls", in which a person is walking in full view of the Kinect followed by a high speed acceleration towards the ground are easily detected. Inac-

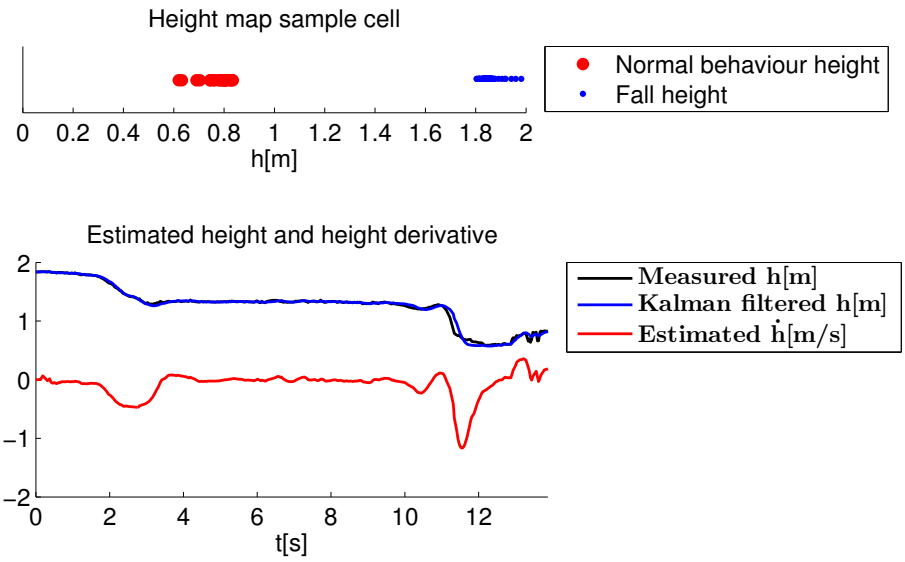


Figure 4.33.: Scenario 3, fall 4 sample map frame and height estimate

tivity detection, height map and $\hat{\dot{h}}$ all show clear deviations from the normal.

Falls where a person is sitting and falling while getting up are also detected relatively easy; also here all three parameters show clear deviations from the normal, especially the height map and inactivity detection.

When the person falls out of Kinect view, it seems that specific detection criterions must be employed. Inactivity detection cannot be used, but the placement of the exit point can be added in the decision if a fall has occured. Height map and $\hat{\dot{h}}$ can be evaluated before the track is lost.

The bedroom is a more complicated scenario, especially the process of getting out of bed seeing as there is a weakness in the segmentation algorithm, which should be improved or accounted for in software. However, this weakness might not necessarily cause the fall detection algorithm to miss a fall, seeing as the parameters still seem to show somewhat abnormal behaviour.

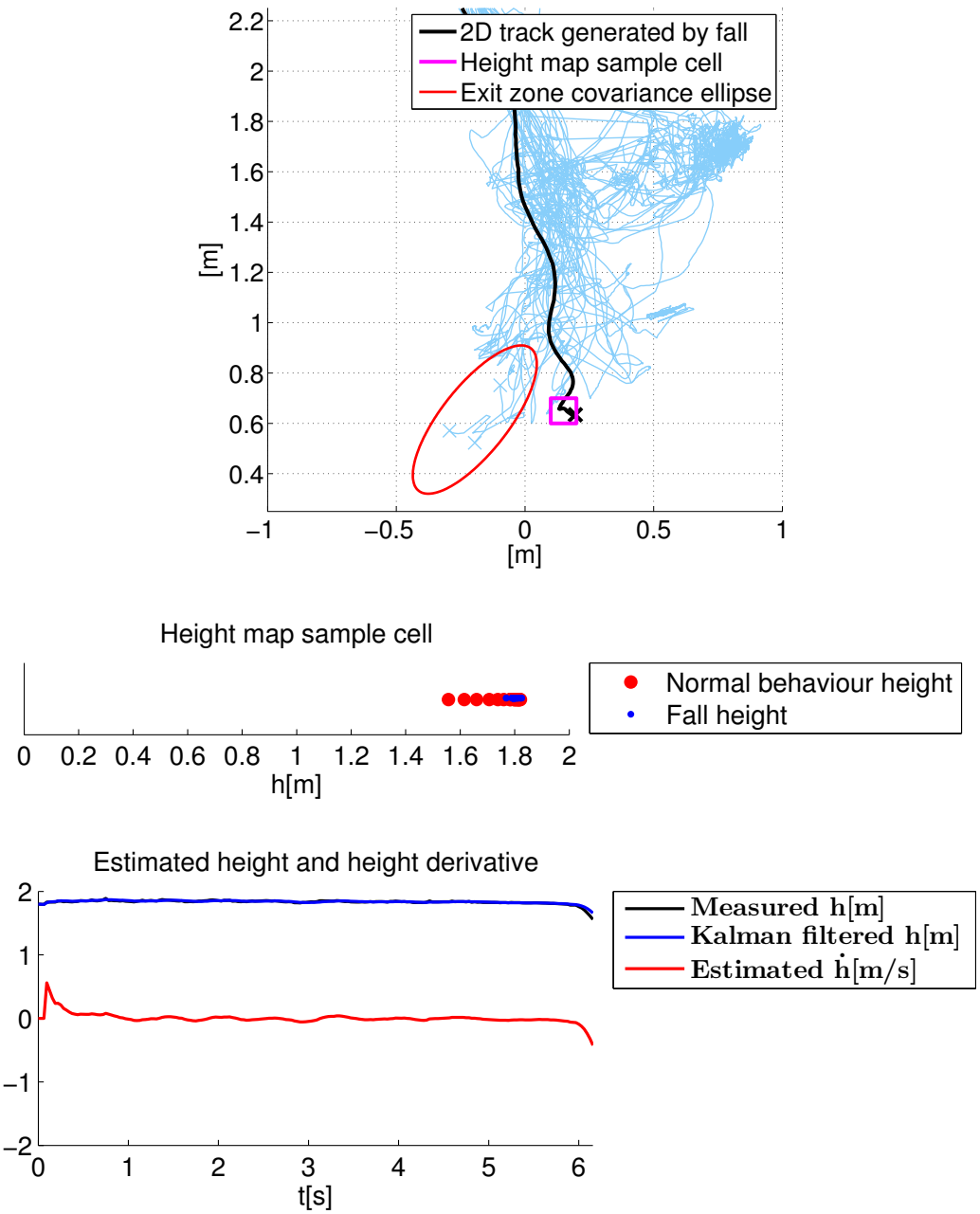


Figure 4.34.: Scenario 3, fall 5 track,sample map frame and height estimate

5. Suggested Solution

In the two previous chapters some simple and more advanced parameters are analyzed. The parameters that seemed most robust and most promising was inactivity detection, height map and $\hat{h}$, plus entry/exit analysis for fall detection partially or fully out of Kinect view. The challenge is combining these parameters to create a system which minimizes both false negatives and false positives. Each parameters' strengths and weaknesses should be evaluated regarding false positives and negatives.

The possible approaches vary from very simple such as a simple voting system among the three parameters to more complex approaches such as using Hidden Markov Models. The drawback of using classifiers is the need for a training set, which is hard to generate with realistic falls, especially because the three parameters are dependent of the generated maps which are updated dynamically. A simple voting system based on the proposed properties would be more understandable and be based on more obvious parts of a fall. It makes sense to keep the detection scheme as simple as possible, so if a voting principle is sufficient, this would be better. It could also be the case that some weaknesses are of such a nature that a more advanced classifier would not increase performance.

5.1. False Positives and Negatives

It is clear that all parameters except $\hat{\hat{h}}$ are dependent on a (to a varying degree) large and covering training period. For evaluation of how prone a parameter is to false positives and negatives it is assumed that the maps have been trained appropriately and tuned such that they give an accurate map of the room. It is also assumed that a person is segmented correctly.

5.1.1. Inactivity Analysis

Inactivity analysis by itself will most likely detect falls in view seeing as a critical fall will be followed by an unusual period of inactivity. If the person is able to move after the fall has occured, the problem is far less critical, seeing as the person will most likely be able to get/call for help by themselves. However, inactivity analysis by itself is also prone to false positives. If a person stands still on a floor cell which

has previously been an activity zone for a longer period of time, inactivity analysis will detect unusual behaviour. The same goes for a person sleeping for an unusually long amount of time. Furthermore, if the person track is lost, inactivity analysis will not work.

5.1.2. Height Analysis

The height map might be prone to false positives. If a person is for example picking up something from the floor, the height will be far lower than the usual height and the height analysis will detect unusual behaviour.

5.1.3. $\hat{\dot{\mathbf{h}}}$

It has been shown in previous chapters that $\hat{\dot{h}}$ is prone to both false positives and negatives. It seems that normal activity also gives negative spikes in $\hat{\dot{h}}$; sitting down in chairs, bending down to pick up an item on the floor and lying down in bed. The false negatives occur if a person falls slowly by getting dizzy and moving more controlled towards the ground. It can be argued that without a period of uncontrolled acceleration towards the ground the probability of a critical fall occuring is small, and it has been mentioned in literature that this is in fact what constitutes a fall. However, the tracker combined with the Kalman Filter may not register this short, uncontrolled acceleration, even though it is likely seeing as the framerate is 30 fps, which is rather large for the purpose of tracking people movement. Also, falling while getting out of a chair will give a shorter distance down to the floor, resulting in a smaller spike in $\hat{\dot{h}}$.

5.1.4. Entry/exit Analysis

As previously mentioned, absence duration will be very prone to false positives if an upper variance is not set. Taking a vacation would result in unusual behaviour. Other than this, because the information is very limited, absence duration is in general prone to false positives. Long showers etc. would constitute as abnormal behaviour, but this must be accounted for by introducing some form of slack if the calculated variance is not enough.

When it comes to using entry/exit zones when a person falls out of the Kinect view, this approach will be prone to both false negatives and positives. For example if a person falls through a door, this will be close to common exit points resulting in a false negative. In addition, if a person exits the view, a sudden drop in height similar to a fall will be observed. This will be discussed further in sec. 5.4.2.

5.1.5. Opening and Closing Doors

Because the process of opening and closing doors introduce movement in the scenery which is not the person directly, this can disturb the tracker. From running tests, it seems there are three different ways the tracker will react to closing and opening doors depending on different factors such as placement of the door in view, orientation of the door and the way the door is moved by the person.

The first reaction is the desired segmenting; the person is tracked and the door is avoided, even though it is moving. This will clearly not cause problems seeing as this is exactly how one wishes the tracker to react.

The second reaction will not cause problems, even though it is somewhat inaccurate. In this case the person and the door is segmented as two persons during the process of opening and closing. On the other hand, the door track will stay in the state "calibrating" until the track is lost, which will not disturb the map creation and fall detection.

The third reaction may cause problems. Because the person might disappear out of view as the door is in motion, the tracker just "transfers" the person track over to the door. This means that the door will be associated with an already confirmed track, and the track will not be lost when the person is exiting. This will naturally cause problems when it comes to entry/exit analysis, inactivity detection and also possibly the height map. There should be some way of taking this into account, especially given the fact that the walls are detected, and the 2D position will be stationary and close to the wall when the door is tracked. Depending on how similar the characteristics are every time a door is tracked, there might also be other ways to detect this, for example using the height, or the projected area, which is quite different for a door compared to a human.

The reason problems regarding doors has not been emphasized in this thesis is that this might not be a problem with the new Kinect 2. The accuracy is greatly improved, along with the tracking algorithm, which suggests this will not cause problems using the Kinect 2 and its tracking algorithm. The difference between a tracked door and a tracked person will also be larger with better accuracy, and one of the techniques suggested in the previous paragraph might work well. If not, other ways of detecting a tracked door should be available.

5.1.6. Refurnishing

An obvious problem of using the maps suggested for describing normal behaviour is if the room is refurnished; chairs, tables and beds are moved around. A way of dealing with this in the long run is by forgetting observations after a while. In other words using a constant number of observations to generate the maps, forgetting the older observations. Nevertheless, the probability of generating false positives shortly

after the refurnishing is significant, seeing as putting a chair in a former activity zone for example, will give unusual values for both height and inactivity.

A way of dealing with refurnishing is defining some global, map-independent parameters for use when the map is not sufficiently large, or the map is yielding unusual values which are not in general unusual. When it comes to the height map a way of doing this is, for example, defining a fall-threshold for height, for example at $h_T = 1\,\text{m}$ in which a fall will not be assumed regardless of values registered in the height map. This will lead to the height map only being used if the normal height is below 1 m. Unfortunately, false positives will still be registered if the item moved is below h_T, but it seems more likely that chairs are moved around compared to beds.

When it comes to the inactivity map it seems harder to define a global, map-independent feature. A minimum inactivity duration could be defined to increase robustness, but this would not help if a chair is moved for example.

Seeing as drastic refurnishing happens rather infrequently, it is possible to simply order the system to reset if this happens.

5.1.7. Summary

Similar to the analysis in sec. 3.4, it is of interest to sum up how each parameter will behave in different scenarios to investigate the need for a more advanced detection algorithm, and to evaluate the importance of each parameter. The scenarios are shown in Tab. 5.1, and are a mix of the scenarios given in sec. 3.4, and other scenarios which, from experience will cause problems. Similar to sec. 3.4, the parameters are

Fall	a)	Fall from standing upright, ending in Kinect view
	b)	Fall from standing upright, ending out of Kinect view
	c)	Fall, out of Kinect view
	d)	Fall to the knees from standing upright
	e)	Fall to the knees from sitting in chair
	f)	Fall when getting out of bed
	g)	Slowly collapsing

Neutral	h)	Picking something up from floor
	i)	Standing still on floor for a while
	j)	Lying down on floor for a while
	k)	Sitting in chair
	l)	Lying in bed
	m)	Refurnished, sitting in chair

Table 5.1.: Scenarios used for evaluating parameter importance

	Inactivity Analysis	Height Analysis	$\hat{\dot{\mathbf{h}}}$	Entry/exit Analysis
a)	2	2	2	-
b)	-	1	1	1
c)	-	-	-	1
d)	2	2	1	-
e)	2	2	1	-
f)	2	2	1	-
g)	2	2	1	-
h)	2	0	1	-
i)	0	2	2	-
j)	0	0	1	-
k)	2	2	1	-
l)	2	2	1	-
m)	0	1	1	-

Table 5.2.: Apropriateness of parameters

rated from 0 to 2, depending on ability to generate true positives/negatives.

The result is shown in Tab. 5.2. It seems that as long as the fall happens fully in Kinect view, both inactivity- and height analysis will give true positives. $\hat{\dot{h}}$ is more uncertain due to the fact that the height difference will not be large from for example lying in bed to falling to the knees out of bed thus generating smaller peaks in $\hat{\dot{h}}$.

The most apparent problem with false positives is definitely scenario j) - lying down on floor for a while. This makes sense seeing as this might even be hard to distinguish for a person actually viewing the depth frames; it seems the only thing separating a person falling, from a person lying down on the floor is $\dot{h}$. However, avoiding a false positive for this scenario based on the three parameters would require having $\hat{\dot{h}} < T_{\dot{h}}$ as a necessary parameter. Seeing as $\hat{\dot{h}}$ is rather uncertain, and a person lying down might have a larger spike in $\hat{\dot{h}}$ than a person falling from a chair to the knees it seems that the choice is between accepting that a person lying down will give a false positive, and an increased number of false negatives.

It seems the important part is as few false negatives as possible, which leads to the choice of simply accepting a person lying down on the floor as a false positive. Furthermore, the system is focusing on fall detection for elders, and it seems rather unlikely that elders will lie down on the floor on purpose.

5.2. Combining Parameters

5.2.1. Normal Falls

It seems from the results in the previous section that as long as the height- and inactivity maps are correct, a fall detection scheme could be based on simply requiring both maps to conclude with abnormal behaviour. These assumptions might be rather optimistic, but it could be argued that the maps must be very inaccurate to give a wrong result, seeing as the differences in both height and inactivity are large when comparing a fall and normal behaviour. It is therefore suggested that for cases where the amount of information in the map for the given position is sufficient, unusual behaviour in both height and inactivity constitutes a fall.

5.2.2. Falls With No Map Information

If one or more of the maps lack information at the 2D position where a fall occurs, the problem is more complicated and other criteria must be defined. This could happen early in the learning process, after refurnishing or simply if the fall happens in an unusual place. Seeing as $\hat{\dot{h}}$ is more uncertain, but map-independent, this seems like a natural addition to the fall detection scheme when required information is not available. A way of making the height map "map-independent" is defining an independent fall threshold, as discussed in sec. 5.1.6. This could be used if no height map information is available. Consequently, the suggested approach for this case is $\hat{\dot{h}} < T_{\dot{h}}$ and unusual behaviour in the height map if the inactivity map yields no information.

5.2.3. Falls Ending Out of Kinect View

This case has been discussed in previous sections. Inactivity detection cannot be used, but entry/exit analysis provide a third parameter in this case. The segmentation is rather unpredictable when a person is exiting the view, due to the fact that in the process of exiting only some parts of the person might be visible. For example when walking through a door, even if the height is constant, the segmentation algorithm may wrongly segment the person as having a lower height. An example of this is in scenario 1, in which a person exits through the door described by exit zone 1 and the registered height is around 0.4 m even though the person is walking upright. This sudden change in height will naturally also affect $\hat{\dot{h}}$.

Due to the fact that one should run the detection algorithm every time a person moves out of Kinect view, this should be rather strict to account for unstable segmentation in the moments of exiting. The suggested approach for this is $\hat{\dot{h}} < \bar{T}_{\dot{h}}$, and unusual height and exit point. In other words, all three parameters should show

unusual behaviour. $\bar{T}_{\dot{h}}$ should be smaller than $T_{\dot{h}}$ due to the fact that the view is interrupted during the fall thus leading to a $\hat{\dot{h}}$-value smaller than it would have been if the fall was fully in view.

5.3. Fall Detection Algorithm

The full algorithm is given in pseudocode in Algorithm 5.1. Seeing as fall detection via absence duration is rather straight forward (and very similar to inactivity detection), and must be analyzed and validated in more realistic settings, this is omitted.

Algorithm 5.1 Full Fall Detection Algorithm

Tuning parameters:
h_T, $\bar{P}_h$, $\bar{T}_{\dot{h}}$, $T_{\dot{h}}$, k_e, P_h, $\bar{P}_i$

```
for current frame
   if track is lost
      if h_k < h_{P_h} and ĥ_k < T̄_ḣ and min_n(k_{exit_n})>k_e
         fall detected
      end
   else
      if sufficient inactivity map is available
         if h_k < min(h_T, h_{P̄_h}) and t_inact > t_{P̄_i}
            fall detected
         end
      else
         if h_k < min(h_T, h_{P̄_h}) and ĥ_k < T_ḣ
            fall detected
         end
      end
   end

   if no fall detected and inactivity cell is changed
      add values from current inactivity cell to height map and inactivity map
   end
end
```

h_T, $\bar{T}_{\dot{h}}$ and $T_{\dot{h}}$ are numbers which must be found when testing. Suggested value for h_T is $h_T = 1\,\mathrm{m}$. When it comes to $\bar{T}_{\dot{h}}$ and $T_{\dot{h}}$, they should be low enough to detect falls, but large enough to not cause unneccesary false positives. To detect all falls in the scenarios in sec. 4.5, we have $\bar{T}_{\dot{h}} > -0.42\,\mathrm{m/s}$.

$\bar{P}_h$, P_h and $\bar{P}_i$ are probabilities between 0 and 1, typically around 0.05. $h_{\bar{P}_h}$ are the values of h in which $P(h_k < h_{\bar{P}_h}) = \bar{P}_h$ and $t_{\bar{P}_i}$ are the values of t_{inact} in which $P(t_{inact_k} > t_{\bar{P}_i}) = \bar{P}_i$. The probabilities are calculated as described earlier by assuming the previously observed values are independent and drawn from a certain probability distribution. In tests the values are assumed to be GMMs, but other probability distributions might yield better results. k_{exit_n} is the k given in eq. 4.11 and discussed in sec. 4.3.3 for exit zone n. To detect all falls in the scenarios, $k_e < 4.45$.

If noisy height measurements are a problem, maybe a series of unusual heights in a row can be required, instead of using only h_k.

5.4. Testing and Results

During testing the fall detection algorithm described above is run at all times. For each scenario, at first the training tracks are run, and the last 5 tracks are the fall tracks. After some small modifications all falls are detected. Still, it is of interest to look at the number of false positives with these threshold values during the learning period. Some problems and possible solutions to these problems are discussed in this section. With the small additions explained below, all falls were detected, and one false positive was present. On the other hand, if the criterion for number of exit points creating a valid exit zone would have been a bit more strict, this one false positive would also have been avoided.

5.4.1. Two Height Map Thresholds

It seems that two height map thresholds may be necessary. In the bedroom scenario, as discussed in sec. 4.5.1, the person segmenting algorithm has a weakness in which a person and the bed is viewed as one person when the person is getting out of bed. This makes the height inaccurate, and if h_T is too small, the height will not be unusually small if a person falls getting out of bed. However, if h_T is too large, a fall might be detected as the system is trained and also during unusual but not fall-related acitivty.

It inutitively makes sense to be more strict about the height treshold after the system has been trained. When no information about the normal height is available, one should be less strict to account for beds etc, but when enough height observations are present to have an impression of the normal height at a certain position it seems natural to be more strict regarding height.

It could be argued that inaccurate segmenting should be taken care of in the segmenting algorithm, and not be accounted for in the fall detection algorithm. This might not be very complicated seeing as this probably has not been considered at

all when the segmenting algorithm has been written, and therefore might be simply fixed by tuning. Regardless, two height tresholds seems reasonable, and with $h_{T_{invalid}} = 0.7\,\text{m}$ and $h_{T_{valid}} = 1\,\text{m}$ all falls are detected whilst maintaining robustness.

5.4.2. Falls When Exiting

Because the case of a lost track in Algorithm 5.1 requires a valid height map (number of observations above a certain threshold) at the point of exit, some false negatives when it comes to falling out of view has been experienced. To account for this, if $\min_n(k_{exit_n}) > k_e$ but the height map is invalid, fall detection for invalid inactivity map and valid height map is run for $t_{end} - 3\,\text{s} \leq t \leq t_{end} - 0.33\,\text{s}$. This is to account for the fact that the person might have crossed cells with a valid height map, but the exit point might not necessarily have one. There are other ways to deal with this, for example lowering the threshold for a valid height map cell.

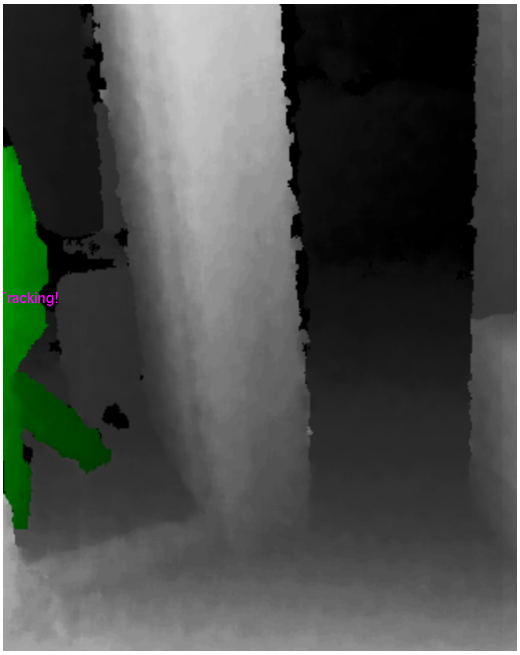


Figure 5.1.: Illustration of height decrease when exiting view

Fall detection when a person is exiting the Kinect view is a significant source for false positives. This is due to the fact that if a person walks out of view where the view is occluded, or at the border of the Kinect view, only some parts of the person might be visible; for example if the view is skewed, parts of the lower body might be the only visible segment, resulting in a large drop in height as the person is exiting. An illustration of this is shown in Fig. 5.1, where a person is exiting through a door in scenario 2. This sudden drop in height is hard to distinguish from a person falling based on height and height velocity alone. Both looking at exit position and normal height at this position can help in determining if a fall or an exit has occured. Seeing

as this partial segmentation will most likely be recurring, requiring a valid height map before a fall can be detected helps, but might not fix all cases.

Another way of handling this, which has been employed during testing is simply not detecting falls when exiting if a person has crossed a "wall", in other words walked through a door. This seems natural seeing as the person is expected to exit the view after this has occured. This will of course result in a missed fall if the person falls out of view after walking through a door. However, the idea is to have one sensor in each room, and one can assume that the sensor in the other room will detect this fall. If no sensor is present, the absence duration scheme can also detect this fall, even though the response will be delayed depending on which room the fall occurs in.

The approach described in the previous paragraph will not help when falling out of Kinect view inside the room. On the other hand, the Kinect 2.0 will, as mentioned have a larger view angle, both horizontally and vertically. It will also be more accurate, which (hopefully) will lead to better segmentation. According to Microsoft, the segmentation has been greatly improved. If the skeleton tracker is also greatly improved, an approach where head height is used instead of z_{max} might yield better results. Furthermore, the larger view angles will make a person falling out of Kinect view inside the room less likely, and this can possibly be left to absence duration analysis.

5.5. Further Remarks

Even though the scenarios are realistic when it comes to layout and occlusions; tables, chairs bed etc., the person tracks are recorded with creating a training set in mind. Furthermore, falls are also recorded with fall detection in mind. It has been suggested that when elders fall, the fall happens much slower than the falls used in this thesis. This suggests tuning of $\hat{\dot{h}}$ thresholds is necessary for the system to function properly. In general, some functionality might prove inaccurate or unnecessary, such as the additional "fall when exiting" detection mentioned in the previous section. This should be reviewed in further testing steps.

It is also mentioned that it is possible to have different thresholds for different times of the day. For example, the slack should be larger during daytime than at night; it is not likely that a person is inactive outside of the bed when for example going to the bathroom at night. This should also be investigated when realistic testing data is available.

One could also put all data from a track in a "buffer", and add these data only if no falls are detected. It should also be considered removing "outliers" resulting from abnormal activity, which may disturb normal thresholds. However, if a limited amount of observations are kept for each cell, these abnormal values will be deleted

after a while, and will most likely not be dominating when they are in the buffer either.

As mentioned above, if the new Kinect 2.0 shows improved skeleton tracking, and have added features, these might be well suited for fall detection as long as they are robust enough. The approach suggested above will also work with the Kinect 2.0, seeing as only segmentation is required, and will also be more accurate with more accurate segmentation.

It has also been mentioned that abnormal inactivity could be detected isolated, not as a fall, but as abnormal behaviour in general. In this way, elders for example fainiting in their chair could also be detected. This can easily be added to the algorithm above, simply adding a criterion for abnormally long inactivity periods.

It is apparent that the tuning parameters can be set to be quite aggressive, or one can choose to have a larger slack. If a system is created in which the fall detection algorithm sends a snapshot of the scene to a person making the final call when a potential fall is detected, this will decrease the inconvenience of false positives significantly. As a result, the algorithm can be tuned more aggressively making it less likely that a real fall will be overlooked. With this approach the fall detection scheme functions as a filter, only extracting abnormal activity which could make fall detection more accurate.

6. Further Work

The most apparent part of further work is, as mentioned a number of times, more realistic testing. Elders have different activity levels, and move more carefully. It seems that height- and inactivity maps will take this into account automatically, but it is still something that must be investigated. Furthermore, the amount of time before a satisfactory map is available is also currently unknown; an hour of intensive activity with the intention of creating a map seemed to give descriptive maps, but the case is very different with normal activity. The time spent inactive in chairs etc. is a lot longer, and time spent moving around is a lot shorter. In addition, a larger number of different falls should be investigated, and each fall should be repeated a number of times. The reason this has not been done is that a realistic scenario with for example an elder in an apartment has not been available. It seems like this is the next step, and even though using a younger, more active person which does not have the creation of a training set in mind could yield more realistic results, it would still be somewhat different from a completely realistic scenario.

A more advanced fall detection scheme based on the same parameters is also possible. Using advanced classfiers and a training set may give better results. However, this is not certain seeing as some false positives are very similar to a real fall. Also, because the maps are dynamic, it complicates being able to use a classifier.

Some of the same analysis as done in this thesis should also be done with the Kinect 2.0. Especially problems regarding inaccurate segmentation should be investigated, and if the results are similar to the Kinect 1 ways of handling this should be proposed.

An important step to create an actual sensor is getting the Kinect 2.0 to work with a smaller computer, such as a computer-on-module system. In this way a stand-alone sensor can be created, which is necessary if one wants to make a commercial system.

A. Enclosed CD

The MATLAB folder on the enclosed CD contains the most important MATLAB-scripts used in this thesis. All files named 'sec... .m' are files related to a specific section, and the rest of the files are help-functions.

If the file does not say scenario in the name, the scenario desired to view must be specified in the file, as a variable 'scenario' almost at the top of the script.

For the fall detection tests in the file "sec_4_6_.... .m" both scenario and fall desired to view must be specified at the top of the script.

"sec_5_4_run_alg.m" is the final algorithm, and also here the scenario must be specified. This file iterates through the falls in the specified scenario comparing the response to the generated maps using the training period. When this file is run and a fall is detected, the relevant sample height- and inactivity map frames are shown, along with observed current value and training set values.

The reason the modified UserViewer program is not included is that this is mostly modified to simply save basic values from the tracker to a file, which is not that relevant in the scope of this thesis. Furthermore, the program will not be runnable without a Kinect and the recorded data, which is too extensive to be included on the CD.

B. Sample Frames

B.1. Basic Property Analysis - sec. 3.4.3

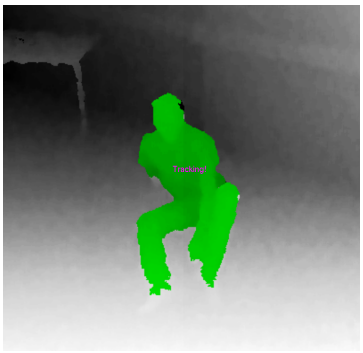


Figure B.1.: a)

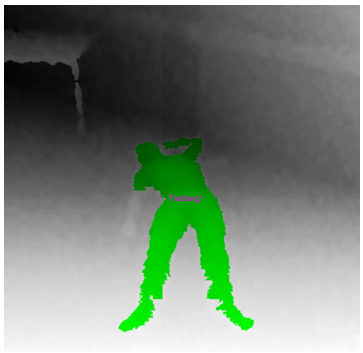


Figure B.2.: b)

Figure B.3.: c)

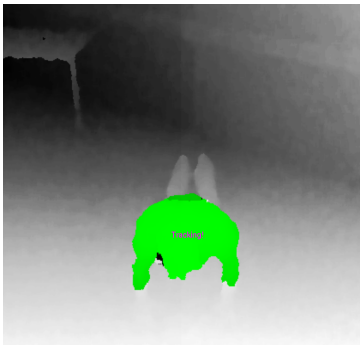


Figure B.4.: d)

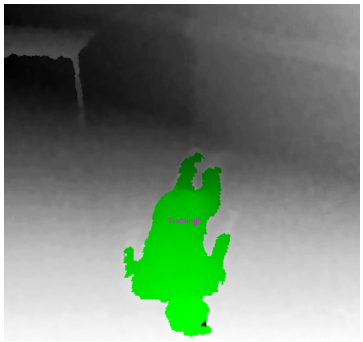


Figure B.5.: e)

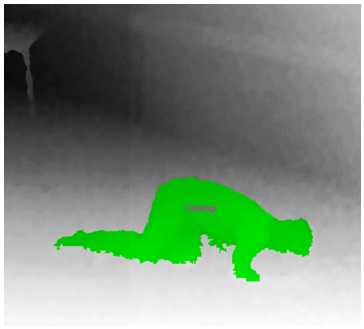


Figure B.6.: f)

Figure B.7.: g)

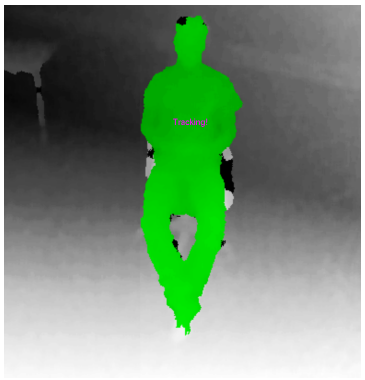


Figure B.8.: h)

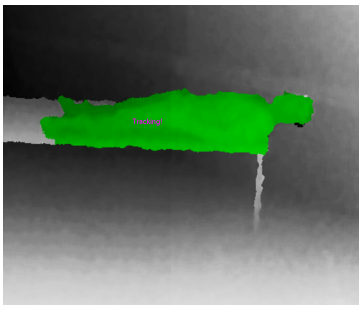


Figure B.9.: i)

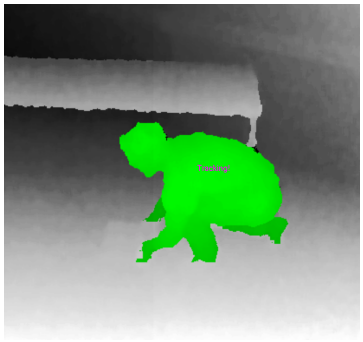


Figure B.10.: j)

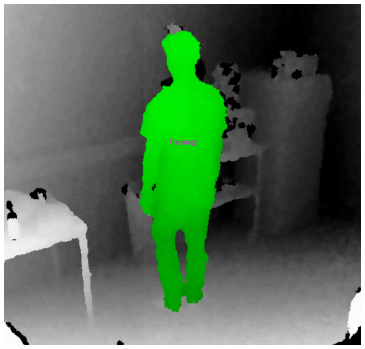


Figure B.11.: k)

B.2. Scenarios Analyzed in sec. 4.5 and sec. 4.6

Figure B.12.: Scenario 1, kinect in view

Figure B.13.: Scenario 1, view from Kinect

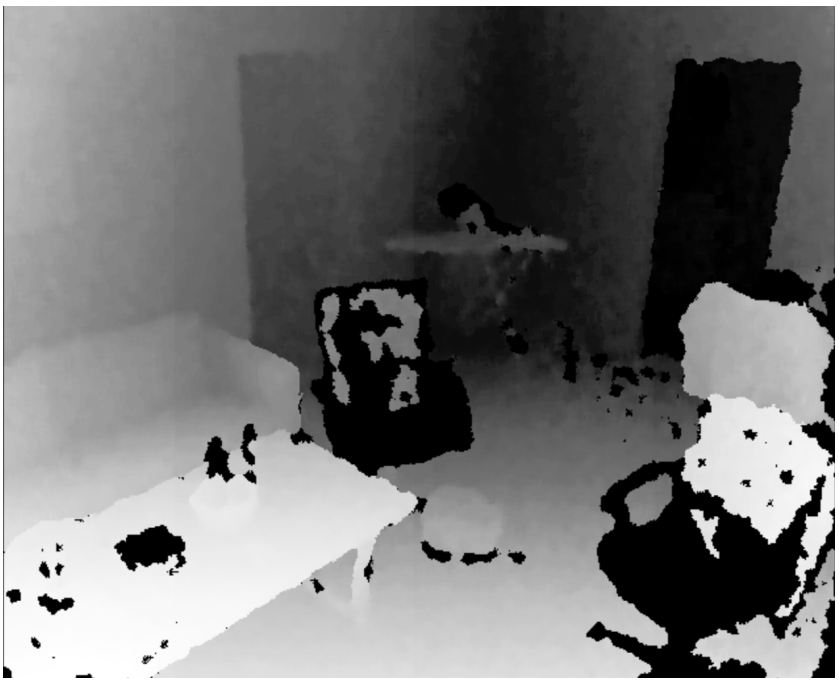


Figure B.14.: Scenario 1, sample Kinect depth frame

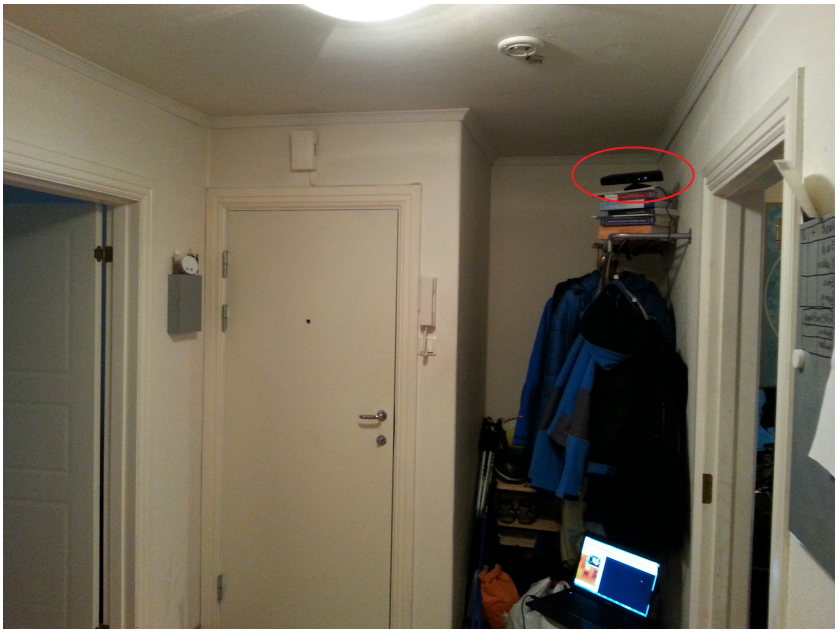


Figure B.15.: Scenario 2, kinect in view

Figure B.16.: Scenario 2, view from Kinect

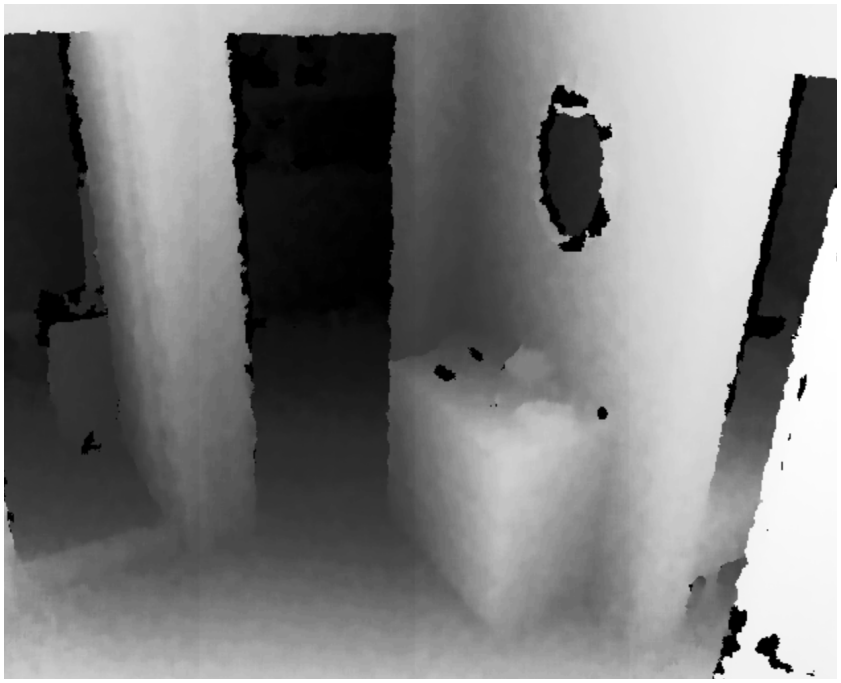


Figure B.17.: Scenario 2, sample Kinect depth frame

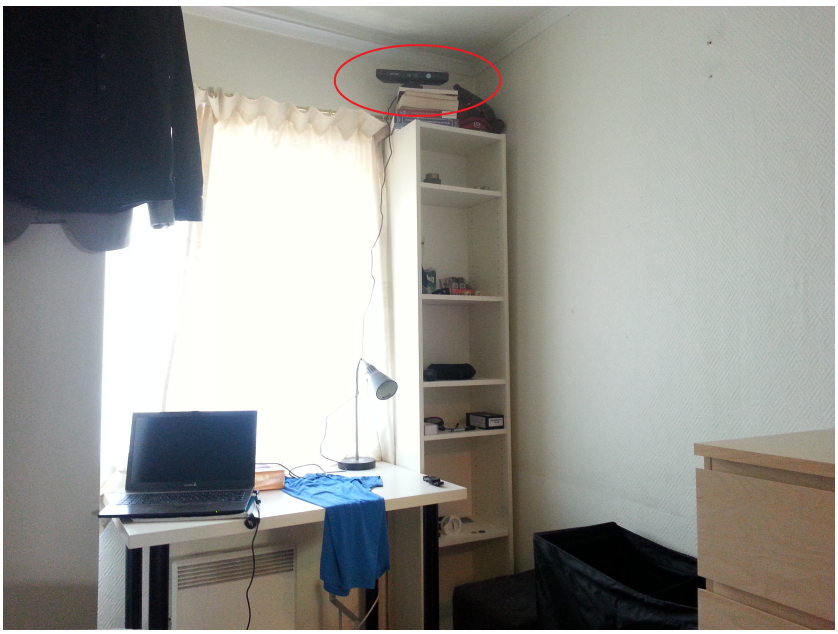


Figure B.18.: Scenario 3, kinect in view

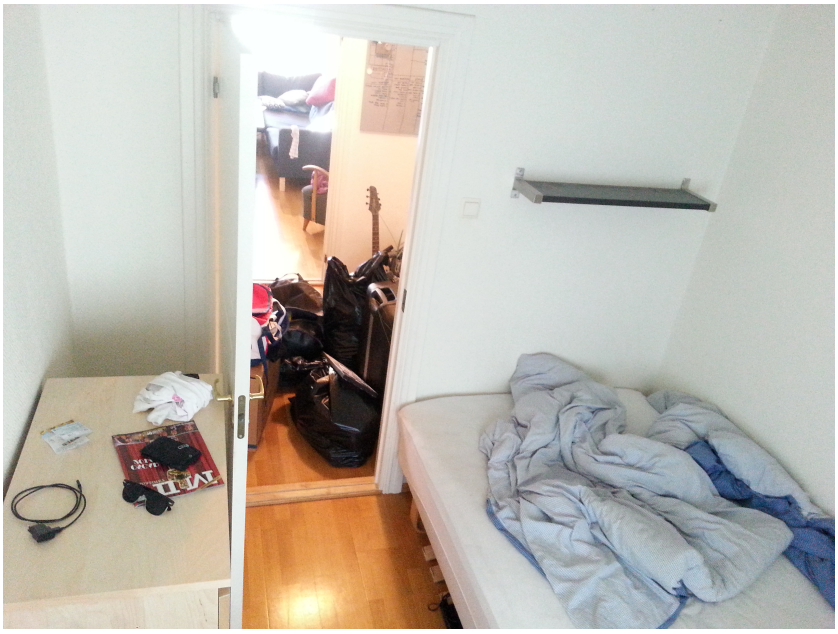


Figure B.19.: Scenario 3, view from Kinect

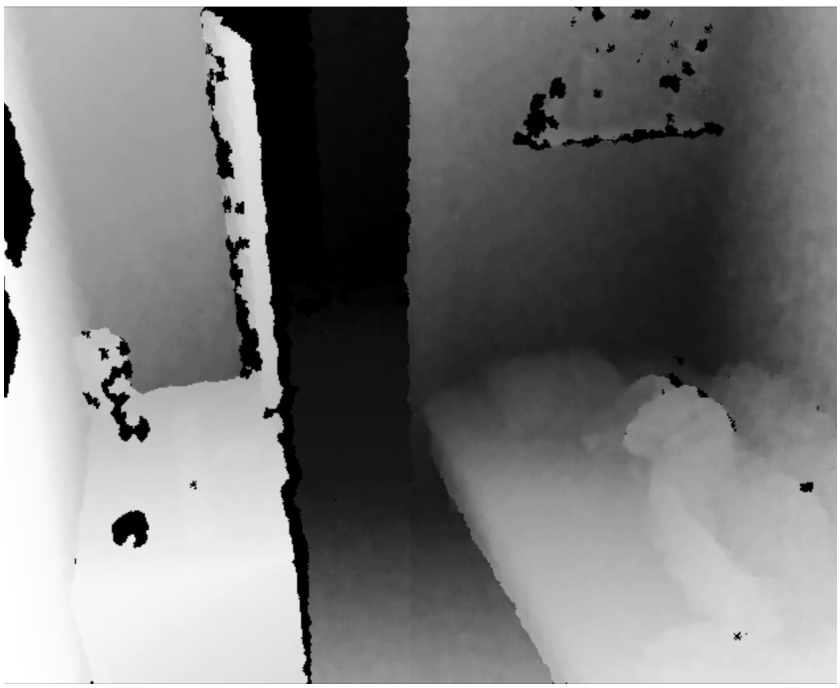


Figure B.20.: Scenario 3, sample Kinect depth frame

B.3. Sample Frames for Falls Analyzed in sec. 4.6

The figures are marked with s..f.., for example s1f2 means scenario 1, fall 2.

There are two sample frames from each fall, one right before the fall and one after the fall has occured.

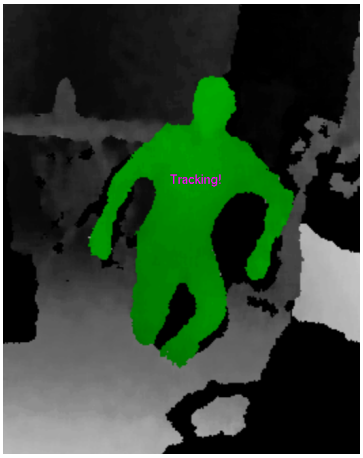


Figure B.21.: s1f1

Figure B.22.: s1f1

Figure B.23.: s1f2

Figure B.24.: s1f2

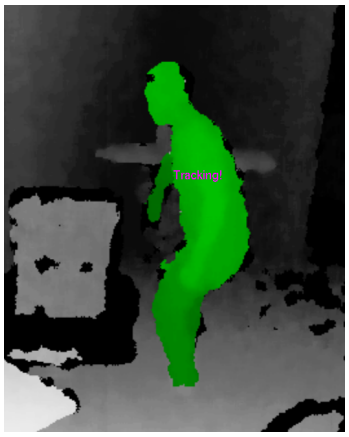


Figure B.25.: s1f3

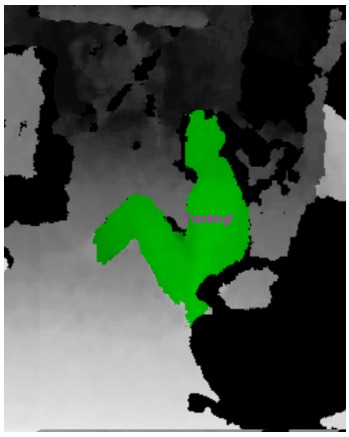


Figure B.26.: s1f3

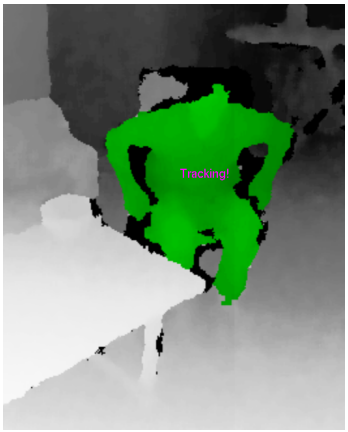


Figure B.27.: s1f4

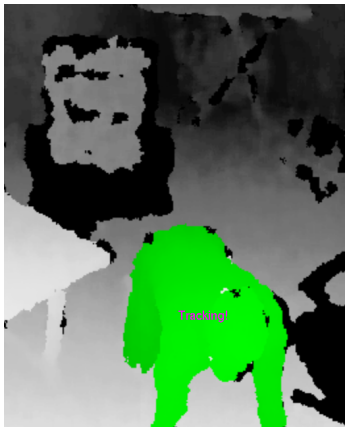


Figure B.28.: s1f4

Figure B.29.: s1f5

Figure B.30.: s1f5

Figure B.31.: s2f1

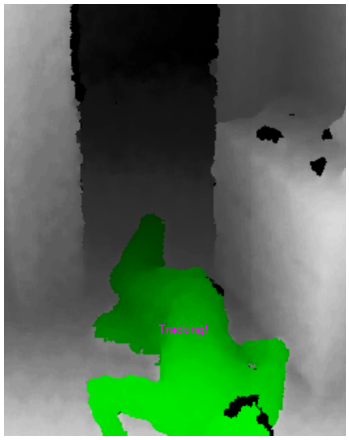


Figure B.32.: s2f1

Figure B.33.: s2f2

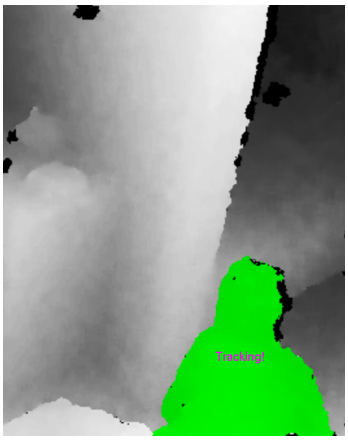


Figure B.34.: s2f2

Figure B.35.: s2f3

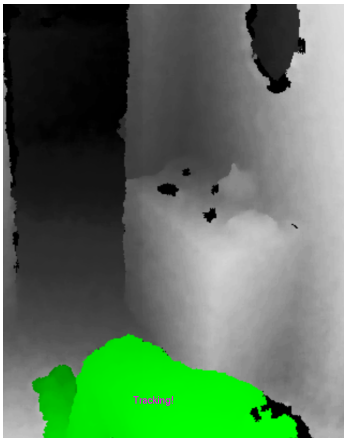


Figure B.36.: s2f3

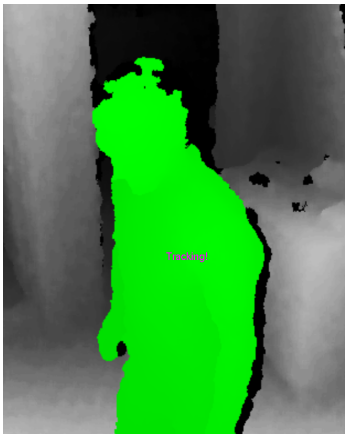


Figure B.37.: s2f4

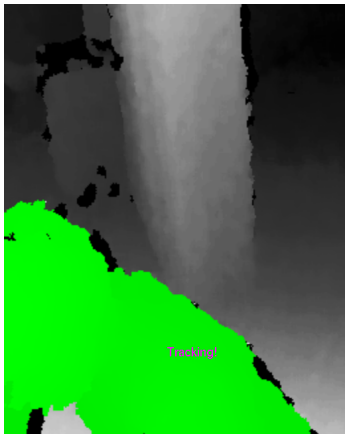


Figure B.38.: s2f4

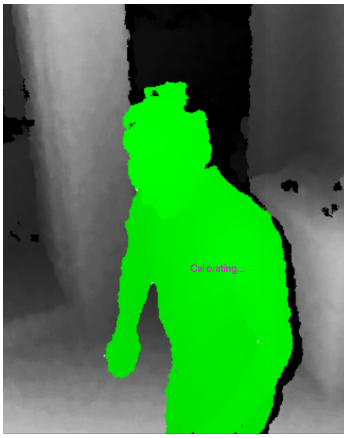


Figure B.39.: s2f5

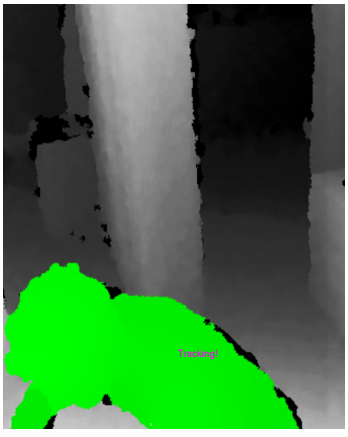


Figure B.40.: s2f5

Figure B.41.: s3f1

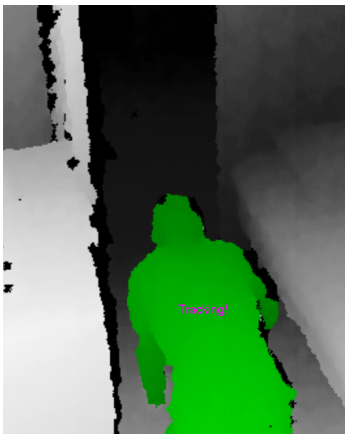


Figure B.42.: s3f1

Figure B.43.: s3f2

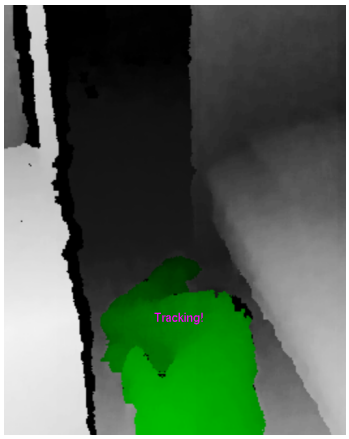


Figure B.44.: s3f2

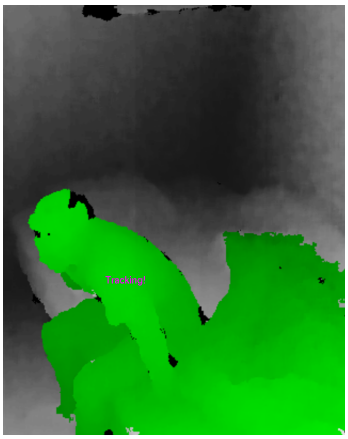


Figure B.45.: s3f3

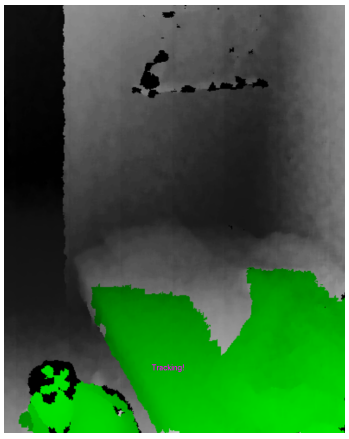


Figure B.46.: s3f3

Figure B.47.: s3f4

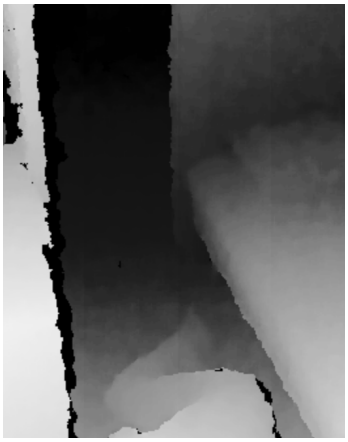


Figure B.48.: s3f4

Figure B.49.: s3f5

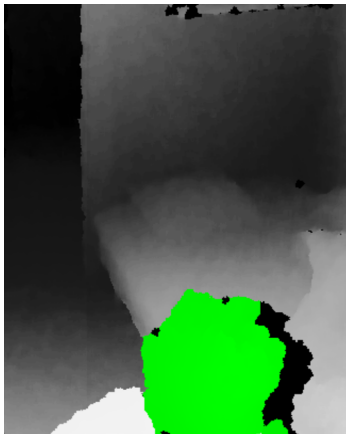


Figure B.50.: s3f5

Bibliography

[1] Microsoft Developer Network - Coordinate Spaces, http://msdn.microsoft.com/en-us/library/hh973078.aspx. Accessed: xx.xx.xx.

[2] J. A. Stevens, P. S. Corso, E. A. Finkelstein, and T. R. Miller, "The costs of fatal and non-fatal falls among older adults," *Injury prevention*, vol. 12, no. 5, pp. 290–295, 2006.

[3] SSB, "Et aldrende samfunn," *Dette er Norge*, 2009.

[4] R. Planinc and M. Kampel, "Introducing the use of depth data for fall detection," *Personal and ubiquitous computing*, vol. 17, no. 6, pp. 1063–1072, 2013.

[5] X. Yu, "Approaches and principles of fall detection for elderly and patient," in *e-health Networking, Applications and Services, 2008. HealthCom 2008. 10th International Conference on*. IEEE, 2008, pp. 42–47.

[6] C. Doukas, I. Maglogiannis, P. Tragas, D. Liapis, and G. Yovanof, "Patient fall detection using support vector machines," in *Artificial Intelligence and Innovations 2007: from Theory to Applications*. Springer, 2007, pp. 147–156.

[7] M. Nyan, F. E. Tay, M. Manimaran, and K. Seah, "Garment-based detection of falls and activities of daily living using 3-axis mems accelerometer," in *Journal of Physics: Conference Series*, vol. 34, no. 1. IOP Publishing, 2006, p. 1059.

[8] M. Alwan, P. J. Rajendran, S. Kell, D. Mack, S. Dalal, M. Wolfe, and R. Felder, "A smart and passive floor-vibration based fall detector for elderly," in *Information and Communication Technologies, 2006. ICTTA'06. 2nd*, vol. 1. IEEE, 2006, pp. 1003–1007.

[9] A. Sixsmith and N. Johnson, "A smart sensor to detect the falls of the elderly," *Pervasive Computing, IEEE*, vol. 3, no. 2, pp. 42–47, 2004.

[10] C. Doukas and I. Maglogiannis, "Advanced patient or elder fall detection based on movement and sound data," in *Pervasive Computing Technologies for Healthcare, 2008. PervasiveHealth 2008. Second International Conference on*. IEEE, 2008, pp. 103–107.

[11] C. N. Doukas and I. Maglogiannis, "Emergency fall incidents detection in assisted living environments utilizing motion, sound, and visual perceptual components," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 15, no. 2, pp. 277–289, 2011.

[12] B. U. Töreyin, Y. Dedeoğlu, and A. E. Çetin, "Hmm based falling person detection using both audio and video," in *Computer Vision in Human-Computer Interaction*. Springer, 2005, pp. 211–220.

[13] D. Anderson, J. M. Keller, M. Skubic, X. Chen, and Z. He, "Recognizing falls from silhouettes," in *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*. IEEE, 2006, pp. 6388–6391.

[14] H. Nait-Charif and S. J. McKenna, "Activity summarisation and fall detection in a supportive home environment," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, vol. 4. IEEE, 2004, pp. 323–326.

[15] B. Jansen and R. Deklerck, "Context aware inactivity recognition for visual fall detection," in *Pervasive Health Conference and Workshops, 2006*. IEEE, 2006, pp. 1–4.

[16] M. Kepski and B. Kwolek, "Human fall detection using kinect sensor," in *Proceedings of the 8th International Conference on Computer Recognition Systems CORES 2013*. Springer, 2013, pp. 743–752.

[17] X. Dai, M. Wu, B. Davidson, M. Mahoor, and J. Zhang, "Image-based fall detection with human posture sequence modeling," in *Healthcare Informatics (ICHI), 2013 IEEE International Conference on*. IEEE, 2013, pp. 376–381.

[18] C. Rougier, E. Auvinet, J. Rousseau, M. Mignotte, and J. Meunier, "Fall detection from depth map video sequences," in *Toward Useful Services for Elderly and People with Disabilities*. Springer, 2011, pp. 121–128.

[19] G. Mastorakis and D. Makris, "Fall detection system using kinect's infrared sensor," *Journal of Real-Time Image Processing*, pp. 1–12, 2012.

[20] OpenNI Home Page, http://www.openni.org/. Accessed: xx.xx.xx.

[21] PrimeSense Home Page, http://www.primesense.com/. Accessed: xx.xx.xx.

[22] M. Kepski and B. Kwolek, "Unobtrusive fall detection at home using kinect sensor," in *Computer Analysis of Images and Patterns*. Springer, 2013, pp. 457–464.

[23] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[24] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, "Fall detection from human shape and motion history using video surveillance," in *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, vol. 2. IEEE, 2007, pp. 875–880.

[25] B. Efron, "The convex hull of a random set of points," *Biometrika*, vol. 52, no. 3-4, pp. 331–343, 1965.

[26] Mathworks function download, fit_ellipse, http://www.mathworks.com /matlabcentral/fileexchange/3215-fitellipse. Accessed: 27.03.14.

[27] B. Chazelle, "An optimal convex hull algorithm in any fixed dimension," *Discrete & Computational Geometry*, vol. 10, no. 1, pp. 377–409, 1993.

[28] N. Noury, A. Fleury, P. Rumeau, A. Bourke, G. Laighin, V. Rialle, and J. Lundy, "Fall detection-principles and methods," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*. IEEE, 2007, pp. 1663–1666.

[29] J. Zhao, J. Katupitiya, and J. Ward, "Global correlation based ground plane estimation using v-disparity image," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 529–534.

[30] P. Cuddihy, J. Weisenberg, C. Graichen, and M. Ganesh, "Algorithm to automatically detect abnormally long periods of inactivity in a home," in *Proceedings of the 1st ACM SIGMOBILE international workshop on Systems and networking support for healthcare and assisted living environments*. ACM, 2007, pp. 89–94.

[31] S. J. McKenna and H. Nait-Charif, "Learning spatial context from tracking using penalised likelihoods." in *ICPR (4)*, 2004, pp. 138–141.

[32] A. P. Dempster, N. M. Laird, D. B. Rubin *et al.*, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal statistical Society*, vol. 39, no. 1, pp. 1–38, 1977.

[33] S. J. Roberts, D. Husmeier, I. Rezek, and W. Penny, "Bayesian approaches to gaussian mixture modeling," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 20, no. 11, pp. 1133–1142, 1998.

[34] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[35] M. Steinbach, G. Karypis, V. Kumar *et al.*, "A comparison of document clustering techniques," in *KDD workshop on text mining*, vol. 400, no. 1. Boston, 2000, pp. 525–526.

[36] J. Illingworth and J. Kittler, "A survey of the hough transform," *Computer vision, graphics, and image processing*, vol. 44, no. 1, pp. 87–116, 1988.

[37] Wikipedia Page - Hough Transform, http://en.wikipedia.org/wiki/Hough_transform. Accessed: xx.xx.xx.