Radboud University Nijmegen

Master thesis Artificial Intelligence

Recognizing activities with the Kinect

A logic-based approach for the support room

Author:

Maaike Johanna Theresia Veltmaat
s0628972
m.j.t.veltmaat@student.ru.nl

Supervisors:

dr. ir. Martijn van Otterlo
Artificial Intelligence
Radboud University Nijmegen

Juergen Vogt, MSc
Brain, Body & Behavior
Philips Research, Eindhoven

July 3, 2013

Abstract

The new support and seclusion rooms at the High Care Unit of the mental healthcare services in Eindhoven are equipped with modern technologies, such as Ambient Experience and a large touch screen interaction wall. This setting allows for the use of other technologies as well, such as a monitoring system which can be used to get more information about the client to adjust the care to their needs.

We developed a prototype for such a monitoring system in which we use 2 Kinect sensors to record motion data of a person in a room. We will apply the event calculus on the recorded data to recognize simple activities.

The event calculus is a logic-based approach, that we can use to reason about events in time and their effects. A logic-based approach has the advantage that we can easily incorporate domain knowledge into the recognition process. We can also reason more easily about the recognized activities as we can define a more complex activity in terms of simpler activities. The approach can be extended by defining more complex activities or by employing the probabilistic event calculus.

As a proof of concept, we tested our recording and recognition system with healthy participants in an office environment. The participants were asked to perform easy tasks, for which we could recognize simple activities, such as walking to a location, or walking around. We were also able to recognize a bit more complex activities that built upon the simple activities. The results are promising and suggestions for further research are made in this thesis as well.

Contents

Acknowledgements

First of all, I want to thank my supervisors, Martijn van Otterlo and Juergen Vogt, for their guidance and encouragement during my research and writing.

I also want to thank Philips Research, for giving me the opportunity to work on this project, and providing me with the necessary hardware. And of course, the staff at the GGzE, for taking the time to show me around in their facility and providing the necessary information.

I want to thank everyone who took the time to read my thesis and provide me with feedback, and those who helped me to structure my mind. And last but not least, I want to thank my friends, family, and colleagues for their support and encouragement.

Part I

Problem Setting

Chapter 1

Introduction

Imagine a system that monitors you when you are walking in a room. The system is able to determine what you are doing and automatically adapts the room based on the detected behavior. When you are agitated or restless the system changes the room such that it becomes a place to calm down. When you are lazy, the atmosphere changes to make you feel more active.

Now try to imagine you are not in your own room, but in a seclusion room at a psychiatric facility. The staff on the ward has separated you from the rest of the group because you were becoming too aggressive. You have always been a fan of nature, as it calms you down. While you were walking towards the seclusion room, the walls in the hallway were lit with green light. The walls of the seclusion room you are in are green as well, while pictures of the forest are displayed on a screen. As you are calming down, you get access to other entertainment options, such as a video connection with your friends and family. But as soon as you are getting frustrated, and start hitting the wall, the atmosphere in the room changes again, to that same forest-like setting which relaxes and calms you.

A system that changes the atmosphere of a room based on your mood in order to induce another mood. For clients in a psychiatric wards who need to be secluded it could be a promising approach to create a relaxing and calming environment and make a stay in a seclusion room as short as possible. However, detecting someones mood is quite hard, even for the trained staff in a psychiatric facility. A monitoring system of the behavior of the client in a support room can help trained staff to assess the mood of the client, and recognize reoccurring patterns which might be indicative for a certain state of mind. With this information, staff might be able to adjust the care they are giving to the client, to facilitate shorter stays in the seclusion room.

An adaptive system consists of two parts; a *recognition/monitoring* part, and an *influencing* part. Both parts of the system influence each other. This thesis describes a first step towards the monitoring part of the system. In the remainder of this chapter, we will provide an introduction to the problem (section 1.1), an approach to solving it (section 1.2) and we will discuss the societal and scientific relevance of this project (section 1.3). In section 1.4 we will present an overview of the topics further discussed in this thesis.

1.1 Project introduction

An adaptive system that changes the environment based on the behavior of the person inside the environment needs a way to recognize the behavior before it can decide on the action to perform. This is called *activity recognition*. There are different levels of activity recognition which differ in time span and complexity (Chaaraoui et al., 2012). The lowest level is based on the *motion* of a person which has a short time span, for example moving your arm. The second level, *action*, is built upon one or more motions. An action consists of a few seconds when someone is reaching for a glass for example. The third level, *activity*, consists of multiple actions and can span minutes. The action `reaching for a glass`, combined with the actions `bringing glass to mouth`, `drinking from glass`, and `placing glass on table` can together

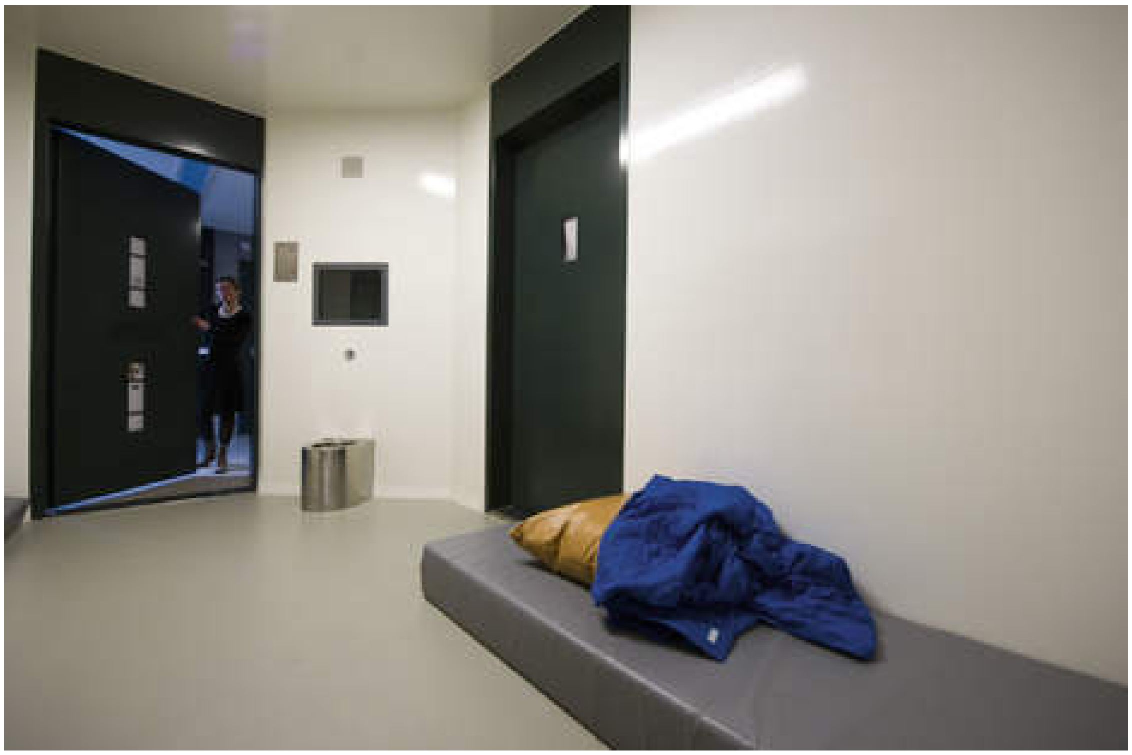


Figure 1.1: Seclusion room in a psychiatric facility. The seclusion room is used to seclude a client from the rest of the group to protect his safety. The room is furnished with a bed and a toilet and often has a board to draw on.

form the activity `drinking a glass of water`. The fourth level is *behavior* and can take up days or weeks. Analysis of behavior can be used to detect habits and routines and to detect abnormalities in known behavior. Research on the recognition of activities will be discussed in section 2.1.

In this thesis we will detect the activities of a client placed in a seclusion room at the mental healthcare services in Eindhoven (GGzE). A seclusion room is a room in a mental healthcare facility which is used to seclude clients from the rest of the group to protect their's and other clients' safety, for example when a client is experiencing a psychosis. A seclusion room is 'prikkel arm' (low stimulus) it is furnished with a bed, toilet, and often a board to draw on (see Figure 1.1 for an impression). The staff checks in on the client during his stay in the seclusion room. The seclusion is a radical intervention for both the client and the staff, and for the client it is often associated with shame. Therefore it is preferred to prevent seclusion or, when necessary, keep it as short as possible.

The contact between staff and client is limited to several contact moments during the day, which makes it hard for the staff to get a full assessment of the mental state of the client. It would be useful to have a monitoring system that detects specific behavior of the client to provide an indication to the staff about the clients mental state. Detecting the mental state of a client can be compared to detecting behavior; it can be indicated by the presence or absence of different activities and has a time span from minutes to hours or even days. As recognizing the mental state is hard, we will focus on the detection of activities of a client in the seclusion room. We assume that presenting the recognized activities to the staff can provide information to the staff that they can integrate with their expertise on and experience with crisis management.

The GGzE utilizes a high care vision which is comparable to the intensive care unit in a hospital. They provide 24-7 specialized care for the clients. The automatic detection of activities to support the staff in their tasks is an addition to this care. The new High Care Unit at GGzE "De Grote Beek" in Eindhoven facilitates the use of technology to support client and staff in the recovery process. We will base our research approach on the situation at the High Care Unit at the GGzE. More information about the High Care Unit can be found in section 2.3.

1.2 How to solve this problem?

We will describe an activity recognition system which can be used in the seclusion room of a mental healthcare facility. Activity recognition takes in data and tries to detect short term activities, or actions, from this data. We can recognize activities based on the recognition of actions. Different methods for recognizing actions and activity from data will be discussed in Section 2.1.1. We will use a logic-based approach to recognize activities in input data, which will be discussed in more detail in Section 6.1.3. In Section 6.2 we will discuss which activities we will detect.

Activities can be recognized from different types of input, for example video data, bio-physiological data, or location data. Related activity recognition research with other input data will be discussed in Section 2.1.2. We will work with 3D data recorded with the Microsoft Kinect, a consumer device that tracks people and provides a 3D location and 3D skeleton representation of a tracked user. More information about the Kinect and research with the Kinect will be given in Section 2.2.2.

The application of activity recognition at the mental healthcare facility comes with some practical constraints which influence our approach. These constraints will be discussed in Section 3.1. To the best of our knowledge, the application of activity recognition in a mental healthcare facility is unique. Related applications of activity recognition in other domains will be discussed in Section 2.2.

We will develop our system to be used in the High Care Unit at the GGzE. We might not be able to test at the High Care Unit, as we depend on the cooperation of the staff and clients. In the first months of the project we have the opportunity to test our system in a mock-up seclusion room.

1.3 Project relevance

This project impacts both the mental healthcare services and the research on activity recognition.

1.3.1 Relevance for mental healthcare services

For the GGzE it is important to provide the client with the best possible care that he needs. A seclusion is very radical and it is preferred that a client will not be secluded at all. When a client has to be secluded, the seclusion should be as short as possible. The staff will constantly monitor the client who is secluded, or placed in the support room, but they have to take care of the other clients on the ward as well. A monitoring system can provide the staff with more information about the behavior of the client and might detect behavior patterns of the client. In that way, it can support the staff when assessing the mental state of the client, and the staff can adjust the provided care if they find that necessary.

In contrast to human observers, a monitoring system can provide objective measurements; they are not colored by the interpretations of the different staff members. Therefore, it can also be used to compare the levels of activity of a secluded patient during the day, or over different days, even when the client was taken care of by different staff members. A monitoring system can also keep a history of the behavior of a client during a seclusion. We can compare activity levels or behavior of a client with the measurements of another day, but also compare it to a previous seclusion. When the client usually is very active between 11 and 12 AM, but suddenly is very slow, the system can inform the staff about this observation. The staff can check on the client to see if they should adjust the care. Besides the behavior comparison between different days, it is also possible to verify the expected influence of an intervention given to the client.

As the system we are developing is merely a monitoring system, the staff is and will remain responsible for the care of the client. The monitoring system can only be used to verify observations made by the staff. It cannot be used to replace the staff or check the exact influence of different interventions. Interventions must be made by the staff based on their expertise.

1.3.2 Relevance for activity recognition research

Our approach is to use an affordable consumer sensor, the Microsoft Kinect. This sensor allows us to extract the location and body position of a person, without requiring additional computer vision techniques. When we can perform activity recognition with data recorded with this sensor, we do not require expensive sensors. Researchers can focus on the intelligent recognition of activities, instead of having to deal with computer vision first. This will advance the field of activity recognition.

It will become easier to acquire data, and we can easily extend it to other environments as well. The sensor is not bound to specific lighting conditions. Even when the lighting conditions are continuously changing or when there is no light at all, the infrared technology ensures that we can always track people.

The use of a logic-based activity recognition approach enables us to reason about the activities of humans based on low-level observable actions. A more complex activity or eventually a behavior can be described in terms of simpler activities. It allows us to reason about those activities in a natural way, making them more understandable.

1.4 What to expect in this thesis

Related research and background information on the GGzE is presented in chapter 2. In chapter 3 we will discuss the practical constraints and motivation for our approach.

Chapters 4, 5 and 6 will discuss the implementation of the recognition system. Chapter 4 will focus on the recording of the data with the Kinect sensor. In Chapter 5 we will discuss the preprocessing of the data, which consists of the merging of the data from two Kinect sensors (Section 5.1.3), the visualization of the recorded data (Section 5.2). In Chapter 6 we will discuss the recognition of the activities. First we will elaborate on the theory of logic-based programming and the Event Calculus (Section 6.1), before discussing how the long-term activities are recognized from the actions (6.1.3). In Section 6.2 we will discuss the recognition of short-term activities (6.2.1) and the definitions of the long-term activities (6.2.2). In the last part of this thesis, we will present the results of our tests with recorded data (Section 7.2) in Chapter 7. Finally we will discuss our results and present the conclusions together with suggestions for future research in Chapter 8.

The appendices provide additional material which is not relevant for the understanding of this research. Appendix A provides a summary of the project in Dutch. Appendix B contains the poster which was made for the innovation market at the opening of the GGzE on October 3rd, 2012. In Appendices C, D and E we provide implementation manuals for the different programs.

Chapter 2

Research Context

This project involves two different aspects, the scientific field of activity recognition and the societal part of the mental healthcare. This chapter provides background information on both topics, starting with research on activity recognition in section 2.1. In this section we will elaborate on different types of data used for activity recognition, followed by a discussion of commonly used techniques. Finally we will specifically discuss research that uses the Kinect as sensor. In section 2.3 we will provide information on the application of the high care concept in the mental healthcare. Before discussing the use of the high care concept in the mental health-care, we will first describe the traditional seclusion room. We will end this chapter by sketching the implementation of the high care concept at the mental health-care services in Eindhoven (GGzE).

2.1 Activity recognition

The activity recognition process consists of multiple steps. First, something happens, an activity executed by a person which we can record with a sensor. This sensor gives us data, for example camera data, bio-physiological data like heart rate, motion data or 3D camera images. We can apply techniques to this data to recognize the activity that was executed by the person. In a monitoring system, this activity is presented to the user.

We will start by discussing common techniques used for activity recognition in Section 2.1.1. In Section 2.1.2 we will give examples of different sensors that are used in activity recognition. In section 2.2 we will discuss research that uses the Kinect as sensor for activity recognition.

2.1.1 Techniques

There are different approaches to recognize activities from recorded data, independently of the type of data. They can be broadly divided into approaches to recognize either actions or activities. Actions can be placed on a lower level than activities, where examples of the former are “walking”, “standing still”, and “reaching”, while activities can be “fighting”, “cooking”, or “reading the newspaper”. Approaches for recognizing activities are often hierarchical in nature; they use previously recognized actions as their input.

The low-level actions can be recognized with different approaches, see Turaga et al. (2008) for an elaborate discussion. Some approaches look at every single frame (2D templates, 3D object models), while others take the entire video into account (spatio-temporal filtering, sub-volume matching). These techniques extract features and match them to a template to recognize an action. Other techniques, such as hidden Markov models (HMMs), estimate a model on the temporal dynamics of an activity. The model parameters are learned from training data.

For the recognition of activities both Turaga et al. (2008) and Aggarwal and Ryoo (2011) discuss various techniques. We will discuss the commonly used approaches *probabilistic* and *logic-based* in more detail.

Probabilistic models Probabilistic models like (dynamic) Bayesian networks and hidden Markov models are often applied to sequential data. Based on the data, they give a probability for a sequence of observations. The underlying actual state, for example the performed activity, is not observable. We might be able to observe features of a certain activity, such as `moving an arm`, although we do not observe the activity itself, `drinking a cup of coffee`.

For each activity there is a probability distribution over the possible observable outcomes, like the `moving an arm` action. The probability distribution is determined by training the model. After training we can infer the performed activity based on the observations in each frame. A disadvantage, is that the model requires a large amount of training data to learn the conditional dependencies and transition probabilities between states.

Logic-based models Logic-based approaches describe activities in terms of sub-activities and the temporal, spatial, and logical relations between them. With a logic-based approach we can reason about the occurring sub-activities and in this way recognize higher-level activities. An activity is recognized when it's sub-activities occur and all the corresponding relations can be satisfied. Logic-based approaches are hierarchical in nature and able to recognize concurrent activities. Besides that, logic-based approaches enable the use of common-sense knowledge. A disadvantage of logic-based approaches lies in their inability to compensate for errors in the recognized input.

An example of a logic-based approach is the *Event Calculus* (EC), which is formulated by Kowalski and Sergot (1986). In classical first-order logic a statement can only have one truth value. We can formulate statements of which the truth value might change over time, for example `I am sleeping` which is `true` when I am sleeping, but `false` while I am awake. The Event Calculus allows the truth value of a statement (or 'fluent') to change over time. The occurrence of an event can initiate or terminate a period of time for which a fluent holds. For the statement `I am sleeping` this would mean that the event `falling asleep` initiates a period of time for which `I am sleeping = true`, while the event `waking up` will terminate `I am sleeping = true`; thereby initiating a period of time for which `I am sleeping = false`. The work of Artikis et al. (2010) describes the use of the Event Calculus for activity recognition.

In Skarlatidis et al. (2014) a probabilistic extension to the Event Calculus is presented. Instead of a Prolog implementation a ProbLog implementation is used, allowing the addition of probabilities to facts in the knowledge base. An activity is recognized when its probability is above a threshold. The attachment of probabilities allows the program to deal with uncertainties in the input data to overcome a common weakness of pure logic-based approaches.

We briefly discussed examples of probabilistic, syntactic and logic approaches for activity recognition. Both probabilistic and syntactic approaches can deal with noisy inputs but do not have the ability to recognize complex temporal structures such as concurrent activities. Logical approaches are able to recognize concurrent activities, but usually cannot handle errors in the input. An interesting combination between probabilistic and logic approaches can be found in the probabilistic Event Calculus. The probabilistic Event Calculus has the reasoning properties of logic-based approaches combined with the ability to deal with uncertainty in the input data as probabilistic approaches do.

2.1.2 Input

There are various types of data used for the recognition of activities, for example camera data (RGB-data), motion data, or location data from radio frequency identification (RFID) tags. Researchers are not bound to the use of one data type, but often employ a combination of data, acquired with different sensors. The combination of multiple sensors can improve a systems performance, as there is more data available for a recognition system(Chan et al., 2008).

Camera data RGB-camera data is often used in activity recognition, for example in surveillance applications. Before recognizing the activities in RGB data, the videos first have to be preprocessed to detect a person in a video frame and to track this person across multiple frames. The research by Kosmopoulos et al. (2008) uses the data from multiple cameras to project the location of a person onto a 2D map of the environment. From this projection, they can extract the trajectory of this person and use a Hidden Markov Model to classify it as either "normal" or "abnormal". Besides the trajectory classification they also extracted *short term activities* (STAs), actions such as "active", "inactive", "walking", and "running". These STAs were extracted by applying a decision tree on the trajectory points, the optical flow and the relative pose to the camera.

When using RGB-camera data in our application, we encounter two major issues. The first one concerns the privacy of the people we are monitoring. A person forced to stay in a seclusion room is vulnerable, and recording their stay in the seclusion room is not allowed. The second issue comes with the additional video processing which is required before we can recognize the activities.

Object data In order to recognize the activity someone is performing, we can also look at the type of objects they are using; when someone is using a frying pan it is likely that this person is cooking diner. One of the methods to determine which object someone is using employs radio frequency identification (RFID) tags. Saguna et al. (2011) use context information based on RFID tags to infer complex daily activities. In the research by Isoda et al. (2004) activity is described as a space-time relationship between the user and objects in the world. In the research by Park and Kautz (2008) RFID tags are used to learn temporal segmentation and object appearances.

The use of RFID tags is mainly important when we are interested in the use of objects for the recognition of activities. The clients in the seclusion room are likely not to use objects. We also want to detect when they are extremely active or inactive, which is not coupled to the use of objects. Therefore, RFID tags, might not be the best sensors for our monitoring system.

Motion data Other research uses the posture and motion of a person to determine the performed activity. Various methods exist to record posture and motion Usually the person being recorded wears 'markers' near the different joints that are tracked. A motion capture system then records the 3D position of the different markers. In the research by Zhu and Sheng (2012), motion data is combined with location data to recognize activities of daily living. The authors looked at the location of the participants, combined with body position and hand gestures to determine which daily activity, such as cooking, eating, or using a computer, was performed.

Another device that can be used for motion capture is the Microsoft Kinect (Microsoft, 2012). In contrast to common motion capture systems, the Kinect does not use markers; it runs software that provides a full-body 3D model consisting of 20 joints. Besides that, it also returns the location of the user. These features, along with the SDK and affordability of the technology have caused many research groups to use the Kinect for their applications. We can find examples in the field of robotics, assisted living, the recognition of daily activities, and behavior monitoring. Some examples will be discussed in Section 2.2.2.

2.2 Related research

Activity recognition is an emerging field and a lot of research can be discussed. In this section we will focus on research related to our application. We will start with research on mental state inference. Although we are not currently inferring the mental state of a client, this is an interesting application for the future. As we intend to use the Microsoft Kinect to record our data, we will focus on research with this sensor as well.

2.2.1 Mental state inference

We can never truly attribute a mental state to another person, but we can try to infer it based on observable features like the use of gestures or the tone of voice. In the interaction with another person we automatically infer the mental state of our conversational partner and possibly adapt our communication style along to it. Mental state inference can be applied in adaptive systems, to change the output based on the inferred mental state of the user. When the user is very agitated or in a hurry, an adaptive system can give to-the-point output to immediately provide the user with the requested information, while a relaxed or bored user might prefer a more extensive answer. In this section we will give some examples of mental state inference methods, based on different types of input and used for different applications.

In the work described by Sakr et al. (2010) bio-physiological measures are used to detect agitation and the transition phase towards agitation. They measure the heart rate, galvanic skin response and skin temperature of the person being monitored, because these can be measured non-invasive and can be understood by the subject. Although bio-physiological data showed to be successful for the detection of agitation it is less suitable in our application. It requires sensors to be attached to the body, and we prefer not to use sensors that need to be attached to the body as we expect the clients in the support room will not want to wear the sensors.

A less invasive input method can be found in the use of video data, for example by looking at facial expressions or gestures. Kaliouby and Robinson (2004) use facial expressions and head movements to determine a persons mental state. The data consisted of 164 video fragments, and with leave-one-out cross-validation they got an overall accuracy of 77.4%. People use facial expressions all the time to show their feelings and to infer the feelings of others which make them an interesting input source for automatic mental state inference. However, it requires data to be recorded from the persons face, which cannot be ensured in real-world situations like our support room. It is also possible to look at unintended gestures, as Abbasi et al. (2010) do. They trained a dynamic Bayesian network to infer mental states from unintentional gestures such as yawning and head scratching. This technique requires the specific gestures to be recognized first.

So far we have discussed examples of mental state inference, based on different types of input. Bio-physiological data such as heart-rate or galvanic skin response are reliable measures to determine whether the person is relaxed or stressed, but have the disadvantage that the client has to wear sensors on his body. As we are dealing with clients at a psychiatric ward who might become violent the use of wearable sensors is undesirable. Camera data is more promising, as it does not require the client to wear any sensors, but there might be privacy issues when the data is stored or watched by someone. Besides that, camera data requires preprocessing before features such as facial expressions or gestures can be extracted and used for mental state inference. For now, we will leave the inference of the mental state of a client in the support room aside and focus on the recognition of his behavior.

2.2.2 Behavior analysis with the Kinect

The recognition of behavior is used in different research areas, such as *surveillance* or *assisted living*. Where surveillance monitors different and unknown people, assisted living usually monitors one person in a home environment to detect when that person falls or is in need of help. In our application we are monitoring a single person to detect whether he is showing abnormal or unwanted behavior, or to detect whether he is calming down, by determining what he is doing. In section 2.1.2 we already presented research on behavior analysis research, based on the different input types. In this section we will focus on a specific sensor, the Microsoft Kinect. The Kinect is a motion-capture device that provides the developer with the 3D location and skeleton posture of the user. The skeleton representation is displayed in figure 2.1. The Kinect sensor is also equipped with an RGB camera and microphone array. Since the launch of the Kinect it is applied in different research projects in robotics, but also in activity recognition and assisted living.

Skeleton data Most research that uses the Kinect as sensor uses the skeleton representation as input for the recognition of activities. Sung et al. (2011) recognize everyday activities such as “brushing teeth” and “working on computer” from recorded skeleton data, even when the person was not seen before. Mastorakis and Makris (2012) use the skeleton information indirectly, by computing the 3D bounding box around a detected person to recognize a fall. Burba et al. (2012) estimate the respiratory rate based on the expansion and contraction of the chest area of the person. They also compute the amount of leg jiggling by determining how often the pixel right above a persons knee increases and decreases in depth. Leg jiggling can be an indication that the person being monitored is nervous. This method requires that the person is sitting in front of the Kinect sensor, something we cannot guarantee.

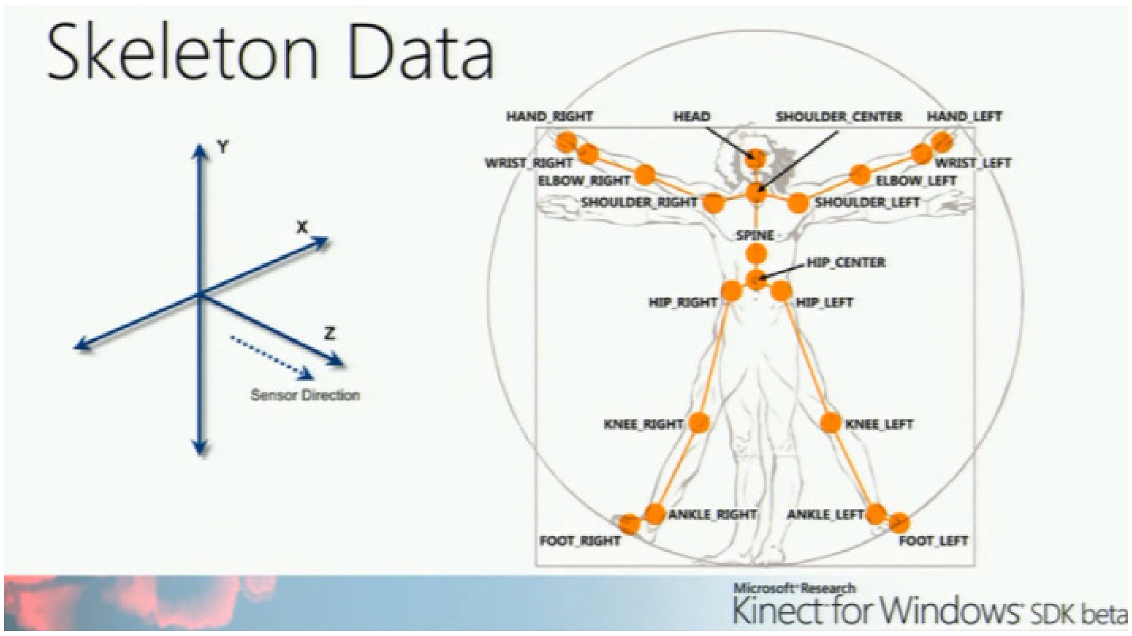


Figure 2.1: Representation of the skeleton inferred by the Kinect. The skeleton consists of 20 joints.

Tracking across multiple cameras Even though the Kinect is equipped with good tracking functionality, at the moment of writing we cannot yet track a person across multiple cameras. The research by Yu et al. (2011) and Sivalingam et al. (2012) monitors the behavior of children across multiple sensors. While Yu et al. try to detect anger and bullying behavior in children, Sivalingam et al. try to monitor children to find out who might be of risk of mental illnesses as autism and obsessive compulsory disorder as they display subtle behavior differences. Both projects are interesting for our application, but are still in an exploratory state.

Audio information Besides the skeleton, RGB- and depth-data, the Kinect can also be used to get audio data through the microphone array. This can be used for sound-source localization, but the input can also be passed to an automatic speech recognition module. This is done by Galatas et al. (2013) in an assisted living application, where they try to detect emergencies, such as a fall or a request for help. We expect to register loud noises in the support room when the client gets violent, or no sound at all. It might be possible to use the microphone array to pick up specific key words, or in general the level of sounds registered. This requires however that the Kinect is placed in the same room as the client, which might not be possible.

In this section we presented various research projects that use the Kinect as sensor. Most of them use the skeleton data to get the location and body position of a person that is tracked. When conditions are optimal, the Kinect is able to register small movements, as the expansion and contraction of the chest when someone is breathing, or to detect specific activities, as talking on the phone or drinking water. The recognition of the activities is usually not bound to specific environments, as we can choose features that are independent of environmental information, like the 3D bounding box of the skeleton. The Kinect is not only used to detect the behavior of healthy people, but also of elderly who are at risk of falling, or children who might be at risk of developing mental illnesses. This multi-employability of the Kinect, combined with it's affordability make it an interesting sensor in behavior research.

In the next section, we will focus on our research domain, the High Care Unit at the mental health-care services in Eindhoven (GGzE).

2.3 High Care Unit

In this section we will describe the context of our research at the mental health-care services in Eindhoven (GGzE). To give a full impression, we will first describe the traditional approach for a seclusion in the mental health-care, before describing new high-care approaches. Our focus will be on the mental healthcare in the Netherlands as our research is located in the Netherlands. We must note that the approach and attitudes towards seclusions in the mental healthcare might differ in other countries. Compared to other countries in Europe, there are a lot of seclusion in the Netherlands although there is less use of forced medication (Veilige zorg, ieders zorg, 2013; Van der Werf, 2009). The mental health-care services initiated a project in the Netherlands, 'Drang naar minder dwang' (urge for less force) to reduce the number of seclusions. More information on this project can be found on the website of the project (Rijksoverheid - Ministerie van VWS, Ministerie van Justitie, 2013; Veilige zorg, ieders zorg, 2013).

Traditionally, a seclusion room is a small room, furnished with a bed and toilet. It is called 'prikkel arm' (low-stimulus) as the client cannot have contact with the other clients and there are no other distractions. Usually there is a small window to enable people to look outside. The contact with the caretakers is limited to contact moments, although the client has the possibility to contact the staff by ringing a bell. The stay in the seclusion room can be requested, but usually it is enforced because the safety of the client, staff or other clients are in danger due to the behavior of the client. The forced seclusion can be a traumatic experience, which can evoke a lot of negative feelings (Rijksoverheid - Ministerie van VWS, Ministerie van Justitie, 2012).

New policy Since the start of 'Drang naar minder Dwang' in 2006 there have been multiple projects in the Netherlands that aim to reduce the number of seclusions and shorten the time clients have to spend in the seclusion room. An initiative we find in multiple projects is the use of a support or comfort room; a separate room in the facility with more comfortable furniture and for example a television and gaming set. A client can request to go into the support room to calm down. The comfort room was first introduced at Mediant (Mediant GGZ, 2010) in 2010 and can now be found in, for example, Clientenbelang Amsterdam and Parnassia as well. At the GGZ Friesland a media pillar is used in the seclusion room (Psy, 2011) to give more information to the client about their stay in the seclusion room or to provide distraction to the client by using it as a television. The GGZ Friesland also allow a mobile phone in the seclusion room to enable the client to call the nurse directly, instead of ringing the bell as used to be the case. To get more insight into a forced stay in a seclusion room, mental healthcare facilities request the help of *experience experts*, healthy ex-clients who had to stay in a seclusion room before. Experience experts can be consulted when developing new policies as they can provide inside information about their experience on the stay in a seclusion room. At the GGZ Oost Brabant, who joined the project in 2008, the number of seclusions decreased 70% in 2012 with respect to 2008 (GGZ Oost Brabant, 2012). Most of the projects envision a *high care* approach; 24-7 clinical care for the clients with intensive treatment and constant monitoring. We can find an implementation of this approach at the mental health-care services in Eindhoven (GGzE) (Kuijpers, 2012). They try to remain contact with the client, with respect to the different steps in the crisis management model. When there is more control on the clients' crisis, he can get more privileges. To facilitate the high care vision, the GGzE opened a special High Care Unit at the terrain of "De Grote Beek" in October 2012.

2.3.1 High Care Unit at GGzE

The High Care Unit in Eindhoven is a facility especially for psychosis patients. There are clients living on the ward, but there are also clients brought in by police or ambulance. The high care unit is a closed ward facility; no one can enter or leave the ward without staff approval. All clients have their own room and there are shared living rooms on the ground and first floor. The staff has no separate office anymore to encourage approachable care. A special part of the ward can be used to seclude clients from the rest of the group. Besides the traditional seclusion rooms, there are also two support rooms. The seclusion room is furnished with a bed, while the support room is a bit bigger and is, besides a bed, also furnished with a table, chairs and beanbag to create more comfort for the client. The difference between these seclusion and support rooms compared to those elsewhere is the use of additional technology; in each room we find a large touchscreen

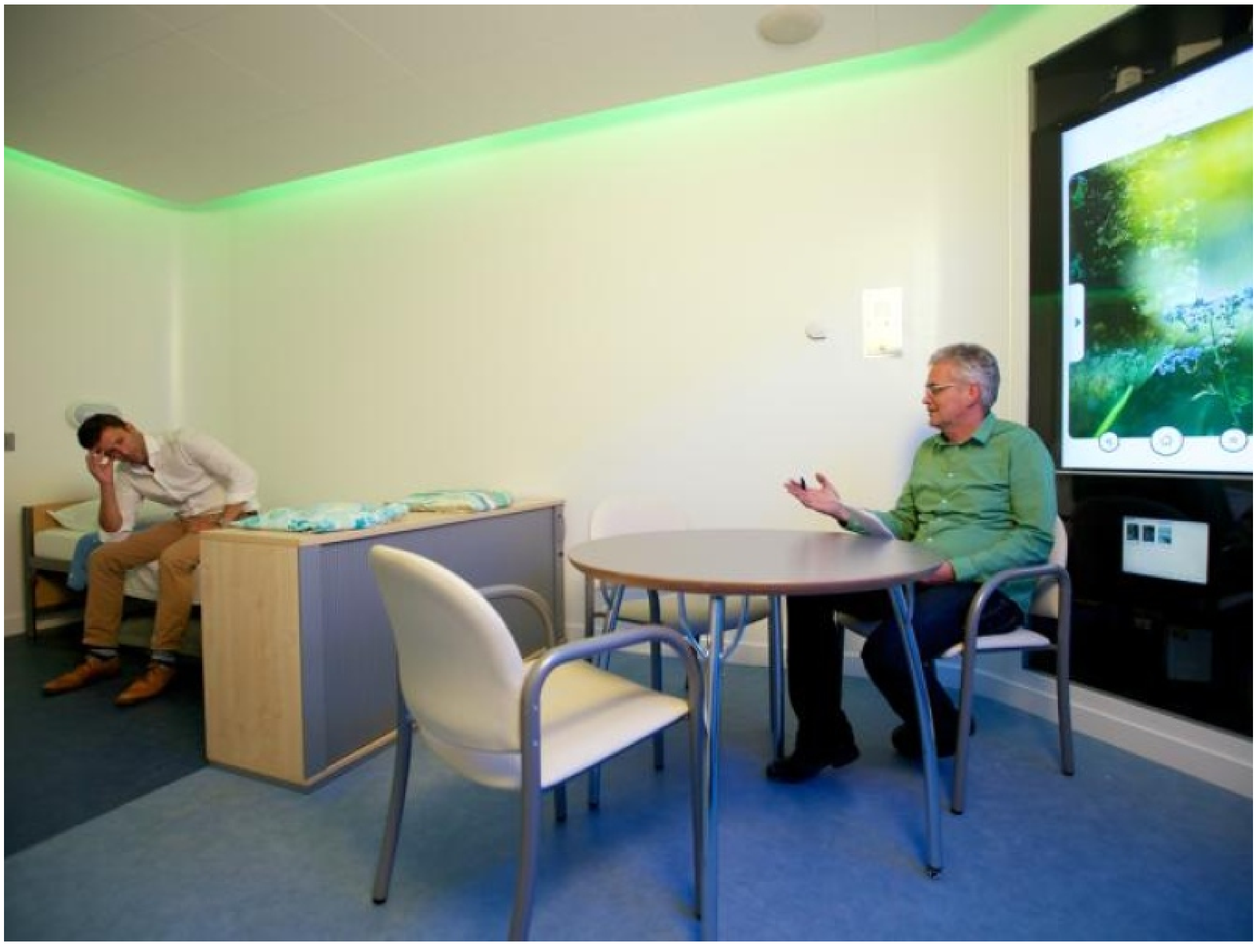


Figure 2.2: A picture of the support room. It includes the interaction wall, a table with chairs, and a bed. The bed is placed in the private zone of the support room which is indicated with a dark floor. The client has to give the care taker permission to enter this zone. The care taker can always enter the public zone of the room which is indicated with a lighter floor.

for entertainment and the rooms are fully equipped with *Ambient Experience* to create a different ambiance. A picture of one of the support rooms is shown in figure 2.2.

Ambient Experience is a product developed by Philips (Koninklijke Philips Electronics N.V., 2013). It combines light, animation, sound and spatial design to distract the client and give him a feeling of control. Based on different themes, the light, animation and sound are combined to create an ambiance in which the client can relax. For an impression of Ambient Experience see Figure 2.3. Ambient experience is originally used in medical applications, for example the MRI or PET/CT suite. On the Ambient Experience website, 360° tours of these rooms are available (Koninklijke Philips Electronics N.V., 2013). In the high care unit it is not only implemented in the support and seclusion rooms, but also in the hallway towards the rooms. When a client in brought into the client, the Ambient Experience theme can be chosen in the intake room, and the hallway and support room will be lit according to the theme. Once the client is in the support room, the theme can be changed by selecting the new theme either on the interaction wall or on a tablet PC outside the support room. By allowing the client to change the ambiance in the room he should feel more in control of the situation, hopefully reducing the length of the seclusion. First consults with the staff indicated that the clients appreciate the implementation of Ambient Experience in the support and seclusion rooms and that the 'Eindhoven theme' is chosen most frequently. In the future, it would be interesting to automatically change the Ambient Experience theme based on the mental state of client.

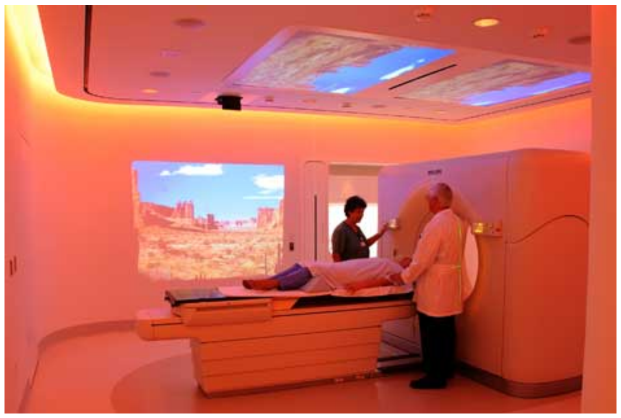


Figure 2.3: Ambient Experience solution in an imaging room. The pictures corresponding to the chosen theme is displayed on the wall and in the ceiling. The ambient lighting in the room is changed according to the chosen theme to create a relaxing atmosphere.

The touch screen or interaction wall (see figure 2.4 can be used to provide additional information or entertainment to the client. It can also be used to personalize the room, by displaying pictures or changing the Ambient Experience theme. When an Ambient Experience theme is chosen, pictures supporting this theme are displayed on the interaction wall. The rooms can be divided into two parts based on the color of the floor. The dark part is the 'private' zone, reserved for the client, while the lighter part is the 'public' zone of the room. A caretaker is always allowed to enter the public zone, but has to ask permission to enter the private zone. This should give the client more control of the situation and always leaves a safe spot in the room for the client.

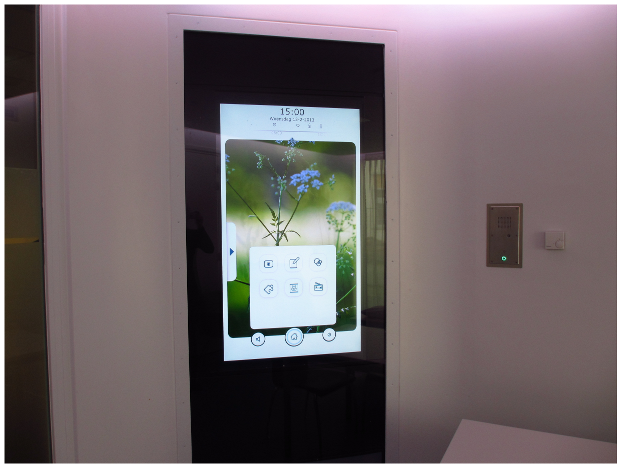


Figure 2.4: Interaction wall

Chapter 3

Approach

The first chapter presented a general overview of the problem we are trying to solve in this project. In the second chapter we discussed background literature. In this chapter we will discuss our project in terms of the practical constraints of the problem. Based on the practical constraints and the literature discussed in the previous chapter, we will present our approach for data acquisition and activity recognition in Sections 3.2 and 3.3.

3.1 Practical constraints

The construction and purpose of the seclusion and support rooms results in some practical constraints which we have to keep in mind when developing our monitoring system. We know for example, that there will be only one person present in the room, and when there are more people, at least one of them will be a member of the staff. In this section we will discuss these practical constraints and how they influence our development decisions. An overview of the support room is given in Figures 3.1 and 3.2.

One person in room As there is only one person in the support room, we know that either a member of the staff is present or a detection error has been made when the system detects more than 1 person. Although it is interesting to monitor the client when there is staff present, we mainly want to monitor him when there is nobody else present. When the staff is present they can observe the client themselves. We decided to initially only monitor one person, the client, and will only record his data. When the activity recognition works satisfactory we can extend data acquisition and activity recognition to more than one person, for example to recognize interactions between people.

Vandal proof We know that a person who is placed in the seclusion or support room might get

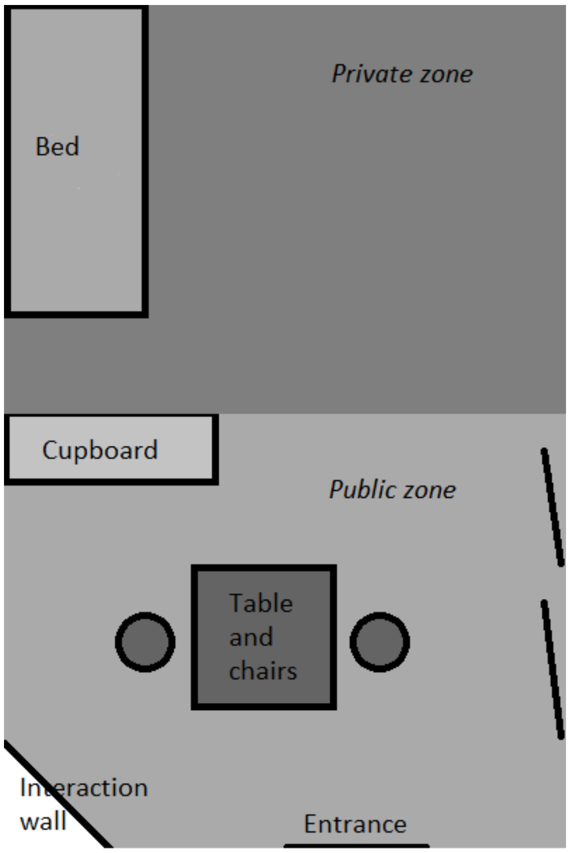


Figure 3.2: Floor plan of the support room.

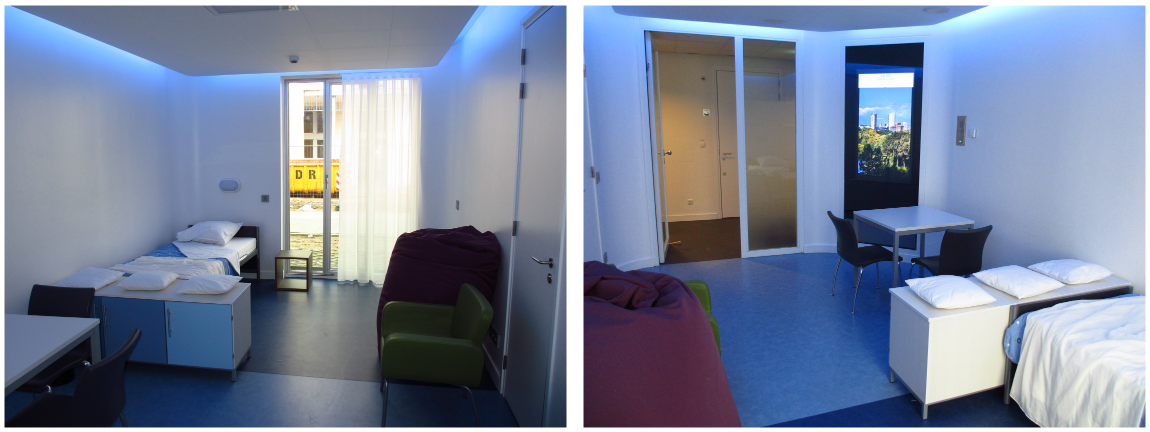

(a) Overview of the support room from the door. (b) Overview of the support room from the window.

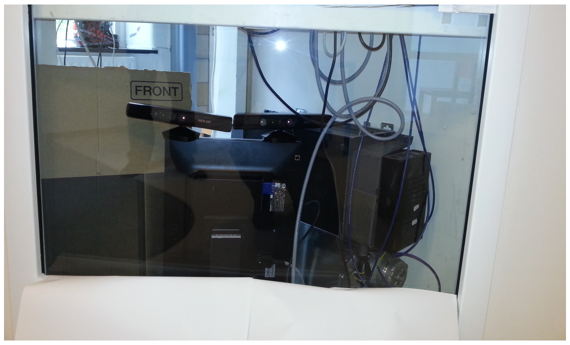
Figure 3.1: Overview of one of the support rooms at the High Care Unit of the GGzE. The support room is furnished with a bed, table with chairs, and beanbag. In one of the corners we see the interaction wall which can be used for information and entertainment purposes. In the picture, the Ambient Experience theme 'Eindhoven' is selected. This theme displays a slide show of pictures from Eindhoven and the lighting is changing depended on the picture.

violent. Therefore we have to make our system vandal-proof. The interaction wall in the room is made of safety glass. We decided to place the Kinect behind the interaction wall (see Figure 3.3); the client cannot access the Kinect when it is placed there, while the Kinect can still record data.

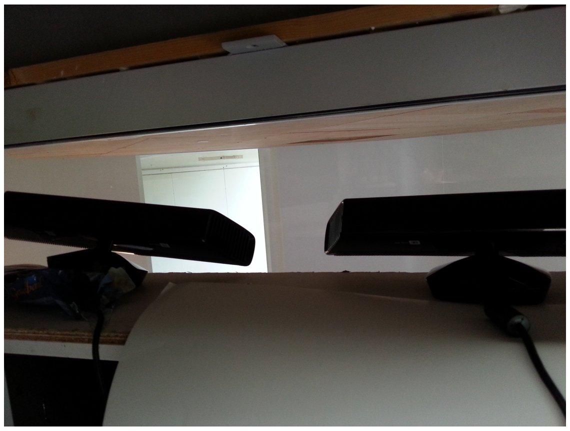
Overview of entire room When recording the data we want to capture the entire room in one view. The rooms are designed in such a way that the staff is able to get an overview in one look before entering the room. As the client is not able to hide from the staff, the staff always knows where the client is and can anticipate on that. We can use this construction to our advantage; the client will not be fully occluded by furniture in the room. We can always get information on the location of the client. When there is no location data available, there is either no person present or the camera has lost track of the client. We assume that the system will only monitor when there is a person present in the room, therefor the lack of data indicates the camera lost the user.

The arrangement of the furniture in the room is fixed; especially in the seclusion room. Clients and staff are able to move the furniture around in the support room, although the staff indicated that the furniture is left in its original location. The knowledge of the location of the furniture in the room can be used to understand the behavior of the client, as we can relate the position of the client to the location of, for example, the bed.

Test at location For the practical tests of our system and data acquisition we depend on the cooperation of the GGzE. During the first few months of our research the High Care Unit was not built, but we were able to test in a mock-up seclusion room. This mock-up was build at the terrain of the GGzE, and equipped with both the interaction wall and the Ambient Experience set-up. At the start of the project we intended to record data in the seclusion and support room, but we had to keep in mind that this might not be possible as the staff has to get acquainted with the technology first. As a back-up we might be able to record data in a lab-setting. Towards the end of the project we decided that it would not yet be possible to record data at the GGzE and therefore recorded several people in a lab-setting. The recording and evaluation of this data is described in Section 7.1. During the project, we focused on the support room instead of both seclusion and support room. Because of this, further writing will only mention the support room. Note that the activity recognition in the support room can be adapted to recognize activities in the seclusion room as well.



(a) View from inside the room.



(b) Behind the interaction wall.

Figure 3.3: The Kinect is placed behind the interaction wall in the support and seclusion room. Because the interaction wall is made of safety glass, the Kinect cannot be accessed from inside the room.

3.2 Data acquisition

We decided to record the data with the Kinect sensor, as it is an affordable sensor which provides an RGB- and depth-stream and it has built-in tracking software. The device determines the location of a tracked person and provides an estimation of the body position in 3D space. We decided to store the data in a file after recording it, to replay and reuse it. In this way, we can test multiple recognition approaches to find the best method for our application. We will process the data after recording it, but in a future application it is possible to process the data on-line.

In order to get a complete overview of the room, we require at least two Kinect sensors. This raised some issues, especially with the merging of the data. As noted before, the Kinect returns the location of the user in 3D space. The returned location is relative to the Kinect. While the y- and z-directions are straightforward, the x-direction is positive towards the right of the Kinect, when looking at the Kinect. When looking from the Kinect, the x-direction is positive towards the left of the device.

Since the Kinect returns coordinates relative to the sensor, we have to convert the measurements to know the actual location in the room. This is important when dealing with more than 1 Kinect sensor, as both sensors will return locations relative to itself. When we know the angle under which the Kinect records the position data, we can perform a matrix transformation to get the absolute position of the user in the room. This will be discussed in Section 4.2.

Another issue when recording with more than one sensor is in the timing information. The Kinect will record data with 30 frames per second, and there might be a slight difference in the exact times on which each sensor records. Even thought the offset is in milliseconds, it is good to keep this in mind when combining the data of multiple cameras. When both cameras return the location of a person, the system should check whether this is the same location. As mentioned before, we assume that there will be only one person present at each time. When we get a location for the person from each Kinect sensor we should merge these locations. In Section 5.1.3 we will discuss how we combine two data-streams and how we merge the two locations. While we were testing the recording software, we noticed that objects in the room, such as a window and a plant, were incorrectly recognized as a person. In Section 5.1.1 we will discuss how we handled this noise.

To verify the quality of our data, we also implemented several visualization options, among which the drawing a of detected trajectory and the generation of a heat map, to get an impression of the frequently visited locations in a room. The visualizations will be discussed in Section 5.2.

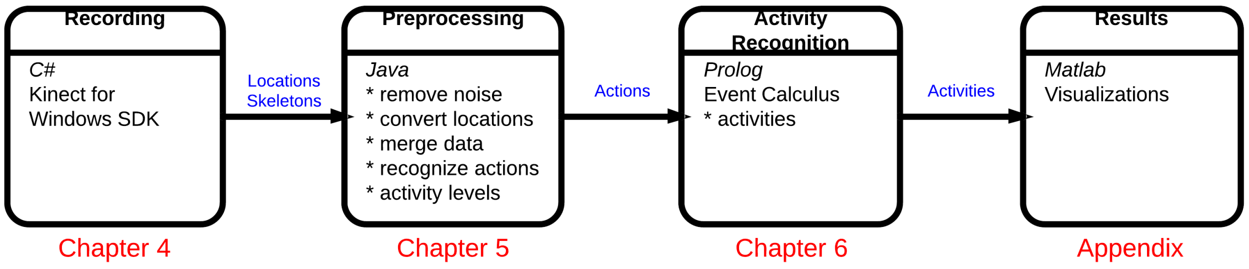


Figure 3.4: Overview of the different processing steps in the monitoring system.

.

3.3 Activity recognition

As presented in the literature (Section 2.1.1), probabilistic approaches as hidden Markov models are common in activity recognition. We choose to work with a logical approach. A logical approach allows us to reason more easily about the recognized activities. A complex activity can be defined as a combination of multiple shorter activities. These combinations are very intuitive; we can define that someone is reading a book when `P is a person`, `P is at the chair`, `P has a sitting body position` and `P is holding a book` are all `true`. When `P is holding a book` is `false`, the activity `P is reading a book` will also be `false`, as one of its sub-activities is false. The sub-activities can be a combination of different sub-activities as well, allowing us to recognize more complex activities.

In contrast to, for example, hidden Markov models where the number of states is fixed, we can easily extend the framework to recognize more activities, or recognize activities for more people. It is easier to model interactions between multiple people as well, without having to specify an interaction for each possible actor.

Another important advantage of a logic framework is that we can easily add more information to the recognition process. This additional information can be domain knowledge, but we can also add data from other sensors.

We choose to work with the Event Calculus (Kowalski and Sergot, 1986) as it comes with a lot of flexibility and allows us to reason about activities in a natural way. As was described in the paper by Skarlatidis et al. (2014), this approach can be extended to a probabilistic logical approach as well. In this way, we can still reason about the activities and events in a logical way, but also handle the uncertainty that arises when working with video data.

We will adapt the work of Skarlatidis et al. (2014) to our research context. The implementation of the Event Calculus as used by Artikis et al. (2010) and Skarlatidis et al. (2014) will be described in Section 6.1. The activities are described in Section 6.2.

Stages in recognition process Figure 3.4 shows the different steps in the recognition process. For each Kinect sensor we use a recording program in C#. Both programs store the recorded data, consisting of the locations and 'skeleton' of the recorded person, in a `.csv` file. The Java program will read these `.csv` files and process the data. Among the processing is the transformation from raw sensor readings to 'room coordinates' and the merging of both data streams. Additionally, the *actions* are determined based on the location data and these will be passed to the activity recognition program, implemented in Prolog. The implementation of the Event Calculus recognizes the defined *activities* from the *actions*. The recognized activities are visualized in Matlab.

Part II

Implementation

Chapter 4

Data Acquisition

Before we can analyze the activities of people in the support room, we first have to record a persons activities. We decided to record the data with the Microsoft Kinect for several reasons. First of all, it is an affordable sensor that also provides depth information of the scene. Besides that, it is equipped with on-board software to track a user and impose a skeleton representation on the user. The technical specifications of this sensor will be discussed in Section 4.1. Before we can record the data we need to know the angle under which the Kinect records the room; this will be discussed in Section 4.2. The recording of the data will be discussed in Section 4.3.

4.1 Microsoft Kinect sensor

The Microsoft Kinect (Figure 4.1) was released in November 2010 for the XBO 360 gaming console. Soon after it was released, developers were able to connect it to a computer and create their own applications with the Kinect.

Kinect sensor The Kinect sensor consists of an RGB camera, combined with a depth sensor and microphone array. The depth sensor consists of an infrared laser projector that projects a pattern of dots onto the scene and a CMOS sensor that captures the reflection of the dots. The software on the Kinect extracts the depth values for the reflected dots. As the depth sensor works with infrared technology, it does not require constant lighting conditions to capture data; it can even work in the dark (see Figure 4.2). Data capture can be distorted when the sensor is placed in direct sunlight, but in our tests we did not notice any distortions. The Kinect has on-board software to track up to six users in its field of view and it can perform motion tracking and gesture recognition of up to two users, based on a *skeleton* it imposes on a tracked user. In Figure 4.2 this skeleton is visualized as a green stick figure.

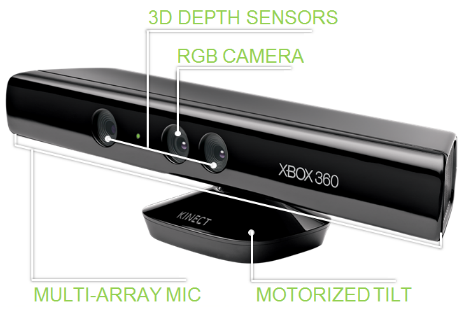
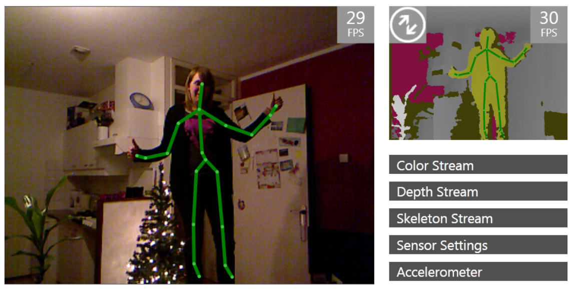
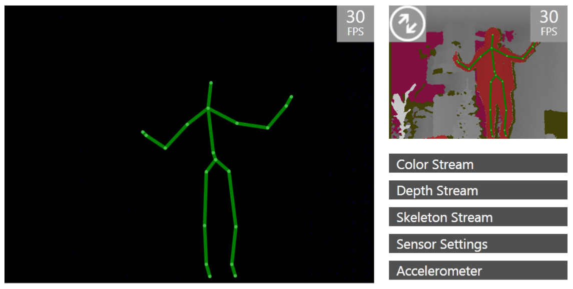


Figure 4.1: The Kinect sensor for XBOX 360. It consists of 3D depth sensors, an RGB camera, and Microphone array. The sensor can be tilted to change the view.

Range of the Kinect The horizontal field of view of the Kinect is 57° and the vertical field of view is 43°. The Kinect can also be tilted in the vertical axis, either a maximum of 27° up or down. Therefore we can place the Kinect at the floor or just below ceiling as well. The Kinect has an optimal range between 1.2 and 3.5 meters, though it can also track people between 0.8 and 6 meters. First tests in the mock-up seclusion room showed that this range was sufficient to capture a view of the entire room with two Kinect sensors. When the distance to the sensor increases, the depth measurements get less accurate. Research



(a) Lights on



(b) Lights off

Figure 4.2: Kinect output for recording a scene with the lights on and with the lights off.

on the accuracy of the sensor by Khoshelham and Elberink (2012) showed that the random *error* of depth measurements reached 4 centimeter at a range of 5 meter, and the depth *resolution* decreases to 7 centimeters at a range of 5 meters. Although we prefer the location of the user to be as accurate as possible, an error of 4 centimeters at the distance of 5 meters is still acceptable.

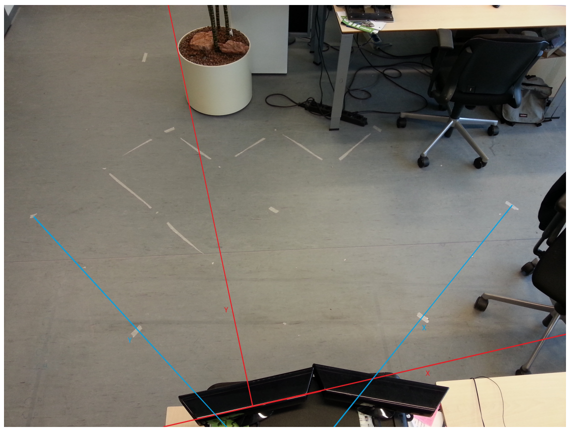
Software development In June 2011, Microsoft released the Kinect for Windows SDK on Windows 7, which allows developers to write applications for the Kinect in C++, C#, or Visual Basic. It provides access to the raw sensor streams, skeleton tracking, and advanced audio processing such as speech recognition. Other frameworks to develop for the Kinect are provided by *OpenNI* or *OpenKinect*. OpenNI (PrimeSense, 2013) is an organization that works on Natural Interfaces. One of it's members is PrimeSense, the company behind the depth sensing technology that is used in the Kinect. Besides drivers, they also provide middleware, NiTE, that can be used for motion tracking. A disadvantage of using NiTE, is that it requires the user to hold a calibration pose before he can be tracked, and that it cannot access the microphone array of the Kinect. The OpenKinect project (OpenKinect, 2012) is an open source project with main focus on the `libfreenect` library. Various wrappers are available to enable developers to program in different languages, and it allows both recording and playback of data. OpenKinect is supported on Windows, Linux and Mac. We use the Kinect for Windows SDK. An advantage of this SDK over the OpenNI framework, is that the user is not required to hold a calibration pose before he can be tracked. In our application in the support room we cannot ask the client to hold a specific pose before he has to go into the support room. Besides that, the SDK comes with sample code and documentation which we can use when developing our code.

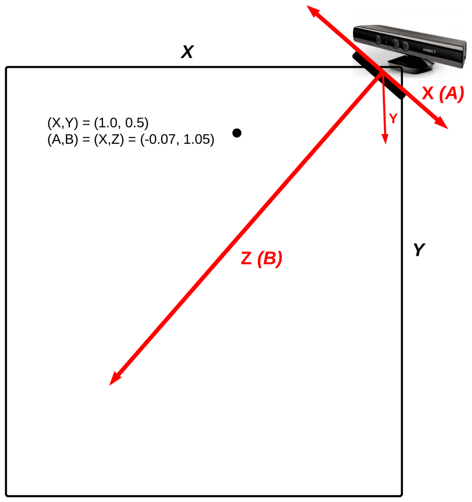
4.2 Calibration

As noted in section 3.2, the Kinect returns 3D coordinates relative to itself. We want to know the absolute location of the person in the room, therefore we have to perform a matrix transformation on the sensor readings, based on the angle under which the Kinect records the room. We will refer to the determination of this angle as *calibration*.

4.2.1 Calibration procedure

We use the absolute location of the user in the room instead of the location relative to the Kinect. This is computed by applying a matrix rotation to the raw sensor readings of the Kinect with the angle under which the Kinect records the room. In Figure 4.3a the blue lines indicate the coordinate system of the room while the red lines indicate the coordinate system of one of the Kinect sensors. The coordinate system of the Kinect (red) has to be transformed to map it onto the coordinate system of the room (blue). We require at least two Kinects to get an overview of the room. The Kinects must be placed such that they together get an overview of the entire room. After that, the calibration program can be used to get the raw sensor-readings


(a) The coordinate system of the Kinect (red) and the coordinate system in the room (blue). To map the coordinates of the Kinect to an absolute location in the room we have to apply a matrix transformation on the Kinect measurements.



(b) The coordinate system of the Kinect projected on a floor plan of the room. The x-coordinate indicates the location in the horizontal direction, parallel to the Kinect. The z-coordinate indicates the distance to the sensor or the depth, perpendicular to the sensor. The y-coordinate indicates the distance to the floor.

of a persons location. The angle is computed from the sensor readings combined with the known location in the room. We did this for multiple fixed locations in the room and averaged the resulting angles.

Fixed room coordinates We first mark the fixed locations in the room. We work with a 50 centimeter by 50 centimeter grid, but other grid sizes are possible as well. It is important that the x, y-coordinates in the room are known for the chosen locations. The raw-sensor readings for each fixed location in the room are used to compute the angle. Figure 4.3b shows a floor plan of the room; the location $(X, Y) = (1.00, 0.50)$ is fixed in the room and the sensors readings $(A, B) = (-0.07, 1.05)$ are returned by the Kinect.

Compute angle The angle will be computed based on the x- and y-coordinates in the room, and the x- and z- readings from the Kinect. For clarity, we will refer to the room coordinates in terms of x- and y-coordinates and to the returned x- and z- coordinates as a and b. Figure 4.3 shows a sketch of the situation. C is the Kinect on a known location in the room and L is the location of the user. Based on the readings from the Kinect we know the coordinate $L_{(a,b)}$, but we want to know the coordinate $L_{(x,y)}$. The helpline h is parallel to the X-axis in the room. We want to know the angle α (Figure 4.3) to map the coordinate system of the Kinect onto the coordinate system of the room. The angle α can be computed as the sum

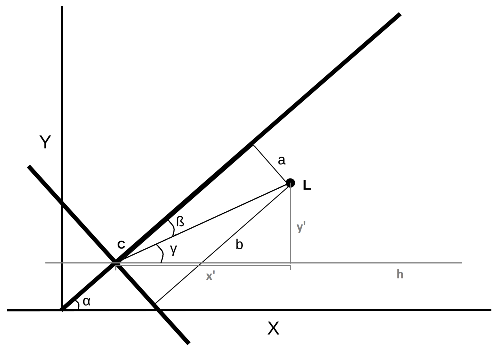


Figure 4.3: In order to know the angle α we need to compute angles β and γ. For the calibration point L we know coordinates x and y, and the Kinect sensor readings a and b. With $\beta = \tan^{-1}\frac{a}{b}$ and $\gamma = \tan^{-1}\frac{y'}{x'}$ we can compute $\alpha = \beta + \gamma$.

Recording	Time (s)	Time (minutes)	% seconds with 30 frames per second
1	324	5:24	81
2	193	3:13	71
3	209	3:29	87
4	304	5:04	66
5	800	13:20	32

Table 4.1: Overview of the performance of the recording program.

of angles β and γ. For β and γ we get:

$$\beta = \tan^{-1}(\frac{a}{b})$$
$$\gamma = \tan^{-1}(\frac{y'}{x'})$$

with a the x-reading of the Kinect, b the z-reading of the Kinect, x' the difference between the L_x and C_x, and y' the difference between L_y and C_y.

4.3 Recording

Our initial recording approach included one program to record the data, which will be discussed in Section 4.3.1. After testing, we noticed that we lost too much data and switched to an approach with two identical recording programs, one for each Kinect sensor. This program will be discussed in Section 4.3.2.

4.3.1 Recording 1.0

Initially we had 1 recording program that controlled both Kinect sensors. An issue is the fact that only 1 skeleton stream can be enabled at the same time, therefore we used a switching mechanism. When the recording Kinect does not detect the person within 75 frames, the program disables the skeleton stream of this Kinect and enables the skeleton stream of the other Kinect. This approach was tested in the mock-up of the seclusion room at the GGzE. The performance was analyzed based on the number of frames per second. The Kinect records with 30 frames per second, therefore we expect our data to contain 30 frames per second as well. During the switching, the number of frames per second can drop. A healthy participant performed 5 scenarios that can be expected in the support room. The number of frames per second are shown in Figure 4.4. There are multiple gaps in the data spanning several minutes. Although there are periods of time for which we get 30 frames per second, we still miss a lot of data. In Table 4.1 we show the time for all 5 recordings with the number of seconds for which we recorded 30 frames per second. While recording the data, we noticed that the program often lost track of the participant, and it was hard to retrieve tracking. The tracking was hard when the participant was walking around, or when the participant was under a blanket, on the bed, or on the floor. When the participant was not tracked, the program switched to the other Kinect. Due to the switching it was harder to track the participant again. Therefore we decided to implement a different recording approach.

4.3.2 Recording 2.0

Because a lot of data was lost due to the switching in our initial approach, we decided to use two recording programs, one for each Kinect. The difference between these programs is in the choice for the Kinect sensor; when there is more than one Kinect sensor attached to the computer, the `camera 2` recording program will choose the second attached Kinect while the `camera 1` recording program will always choose the first attached Kinect. The implementation details of the recording program are discussed in Appendix C.

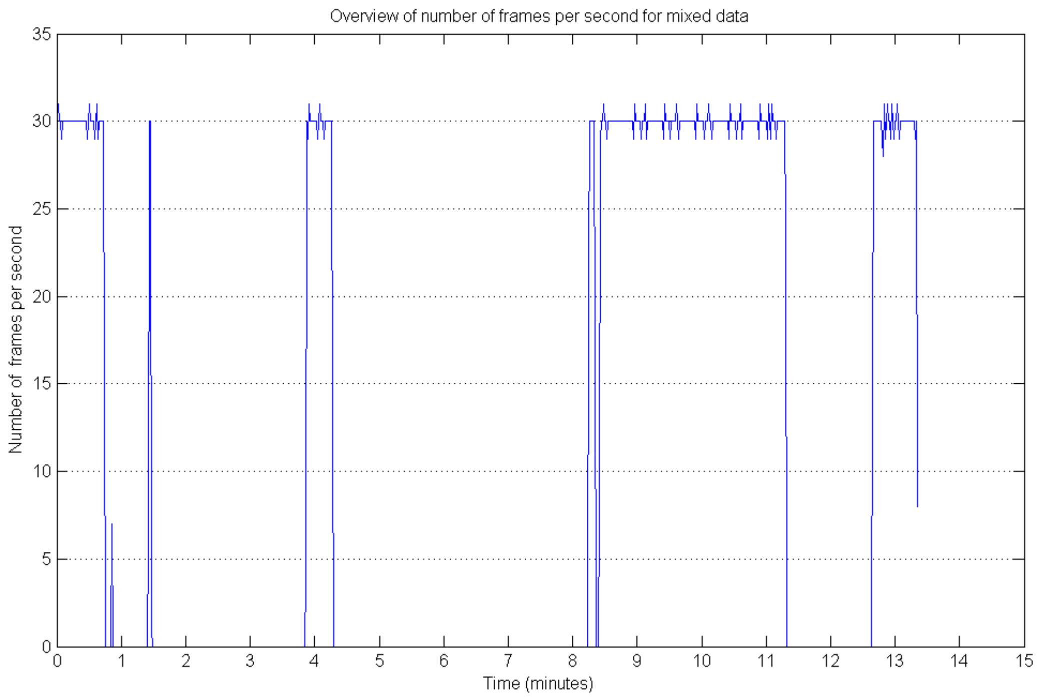


Figure 4.4: Overview of the number of frames per second for the data recorded in the mock-up with the initial recording program.

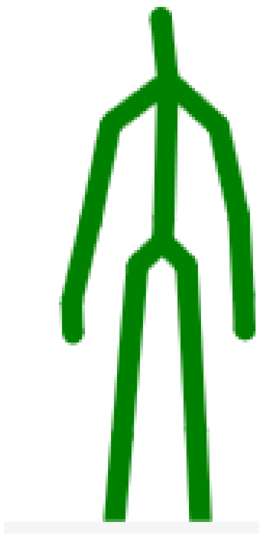


Figure 4.5: Stick figure representation of the Skeleton data.

Depth (mm)	Color
< 0	Black
0 - 900	Blue
901 - 3999	Green
> 4000	Red
Person	Yellow

Table 4.2: Color scheme for the depth-values

Starting the recording During the recording we display the depth-data from the Kinect sensor, together with the skeleton visualization (Figure 4.5) and the distance to the sensor. The depth-data is displayed according to the color scheme in Table 4.2. We only store the skeleton data of a tracked person. For our recognition approach we only require the location of the person. Besides that, it satisfies the privacy constraints, as we cannot identify a person based on the skeleton data. The stick figure of the skeleton is built up by drawing the different bones between the joints. The color of the bones and joints depends on the tracking state; when the Kinect tracked a joint, it will be drawn in green. When the Kinect had to infer the location of the joint it is drawn in lightgreen. When either of the joints is inferred and the other is tracked, the bone between them will be drawn in gray; when both joints are tracked the bone will be drawn in green.

4.3.3 Data storing

In order to reuse the data we store it, both as a serialized object and in a `.csv` (*c*omma-*s*eparated *v*alues) file. Serialization transforms an object into a format that can be stored. It can also be deserialized, allowing us to access the object as before. The `.csv` file can be read by other programs as well and stores the data in a human readable format. We store general information about the size of the room and the angle of the recording Kinect. For each frame we store a time stamp, the location of the skeleton, and the location and trackingstate of each joint.

Chapter 5

Data Processing

After the data has been recorded we need to preprocess it before we can use it to recognize the activities. The preprocessing consists of multiple steps that are implemented in Java. First we have to remove the noise, convert the raw sensor readings to room coordinates, and we have to merge the data from the two Kinect sensors. The different preprocessing steps will be discussed in Section 5.1. We have also implemented a few visualization options to explore the recorded data. These options will be discussed in Section 5.2.

5.1 Preprocessing

The preprocessing of the data consists of the removal of noise, converting the raw sensor readings to room coordinates and merging the data from both Kinect sensors. All these steps will be discussed in this section.

5.1.1 Noise removal

During the software testing we noticed that some objects in the room were incorrectly recognized as a person, for example a plant or wall behind the person. As this noise is at a specific location where the person cannot be, we removed all data entries at that specific location. Because the noise can differ for both Kinect sensors, the noise is removed for each Kinect separately and before merging the data streams.

5.1.2 Converting locations

As discussed in Section 3.2 the Kinect returns the location of a tracked person relative to itself. To get the absolute location of the person in the room we have to perform a matrix transformation of the raw sensor readings from the Kinect with the angle under which the Kinect records the room. In Section 4.2 we discussed how to compute this angle. This section explains how to get the location in the room. A matrix rotation rotates points in Euclidean space over an angle α. In our application, α is the angle under which the Kinect records the room. We can map a point in the Kinects coordinate system, $P(a, b)$ to its corresponding point in the rooms coordinate system, $P(x, y)$. A graphical overview is provided in Figure 4.3. The matrix rotation is given in Equation 5.1.

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \tag{5.1}$$

For each recorded point (a, b) from the Kinect, we can compute $x = a\cos\alpha - b\sin\alpha$ and $y = a\sin\alpha + b\cos\alpha$, with α the angle for that Kinect sensor.

5.1.3 Merging data streams

As we are using two Kinect sensors to record our data, we have to merge the data from both sensors. The data can be merged at several places in the recognition process, for example after the new locations are

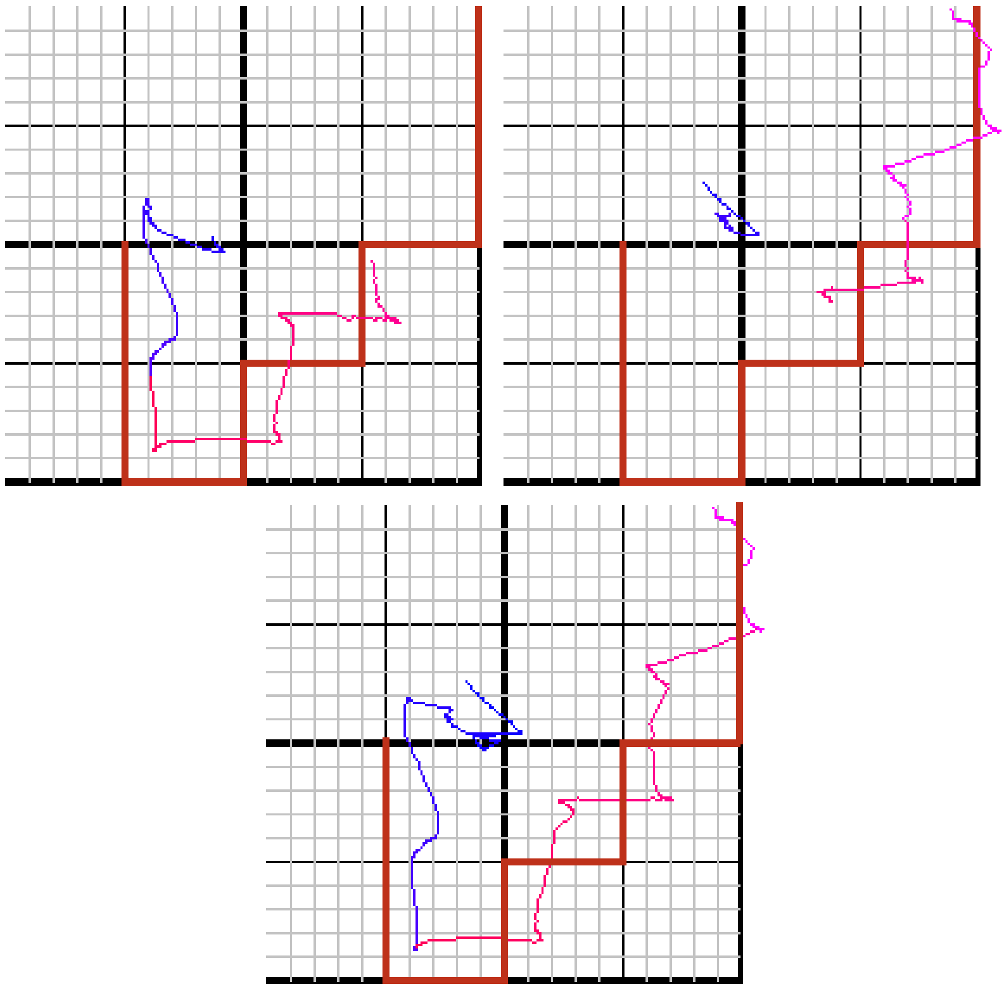


Figure 5.1: Result of the location merging. The top row displays the trajectory visualizations for both cameras independently. The bottom row displays the trajectory visualization for the merged data.

computed, after the short-term activities are recognized, or we can choose not to merge the data at all and feed two input streams into the activity recognition program. We choose to merge the data after the new locations are computed. An advantage is that we can visually explore both the data for each Kinect separately and for the merged data. All remaining processing steps are executed on merged data.

The merging of the data from two Kinect sensors is based on the timing information for each frame. We iterate through the lists containing the timestamps for each Kinect sensor. When the timestamps from both Kinects are within an interval of 30 milliseconds we merge the data, otherwise we store the data from the earlier timestamp. The location data and skeleton data are merged separately.

Location merging Before merging the location we check whether one of the locations is at the default point $(0, 0)$. Because the Kinect cannot record closer than 80 centimeters and $(0, 0)$ is the origin of the Kinect sensor this location cannot be accessed by a person. A location entry at this point indicates that the person was not tracked at that timepoint. Therefore we store the location of the other Kinect when the location of the current Kinect is $(0, 0)$. When the location in neither of the frames is $(0, 0)$ we merge the locations by averaging the x- and y-coordinates. In Figure 5.1 we show the result of the merging for a walked trajectory. The top row displays the trajectory visualizations for both cameras independently and the result

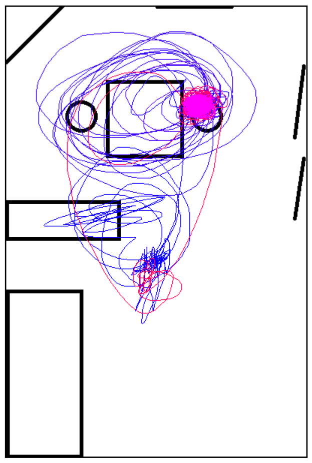


Figure 5.2: Visualization of simulated data. To integrate a timing element in the drawing, the first part of the trajectory is drawn in blue, the second part is drawn in red, and the last part is drawn in pink.

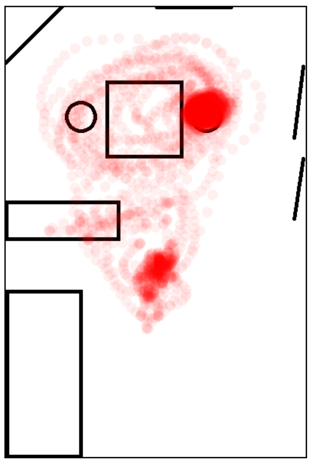


Figure 5.3: Example of a heat map of the room. Darker colors represent more visits to that location.

of the merging is showed in the bottom row. As you can see, the initial trajectories differed a bit at the point where they are merged, but the new locations are averaged. In this example we can also see the error caused by the angle conversion. The initial pattern walked by the participant is drawn in red. Each side of a gray square is 10 centimeters. At most points, the computed trajectory is about 10 to 20 centimeters away from the original trajectory, and at some points it is even 30 to 40 centimeters away from the original. In our application, we work with a global location in the room, and a displacement of 20 centimeters is still acceptable. We have to keep this error in mind when we are setting our recognition thresholds.

Skeleton merging The merging of the skeleton data is based on the timing information as well. Instead of averaging the x- and y-coordinates we average the location of the entire skeleton. Before we do this, the locations of all joints in the skeleton are normalized with respect to the HipCenter joint in that frame. The locations of all joints are in 3D space, relative to the Kinect. By normalizing the joints we make the HipCenter the origin of the coordinate system. After this, we merge the locations for each individual joint based on the data from both Kinects. When the joint was tracked by the Kinect its weight is 2, and when the Kinect had to infer the location of a joint its weight is 1. When one of the joints is not tracked by the Kinect, the location of the other joint is stored. The skeleton merging is not tested thoroughly enough to use in the recognition process.

5.2 Visualization

We implemented different visualization options to explore the recorded data. The visualizations can be divided into location and skeleton visualizations.

5.2.1 Visualization of location data

We looked at three different visualizations of the location; the walked trajectory, a heat map of the room, and the presence in the public and private zone.

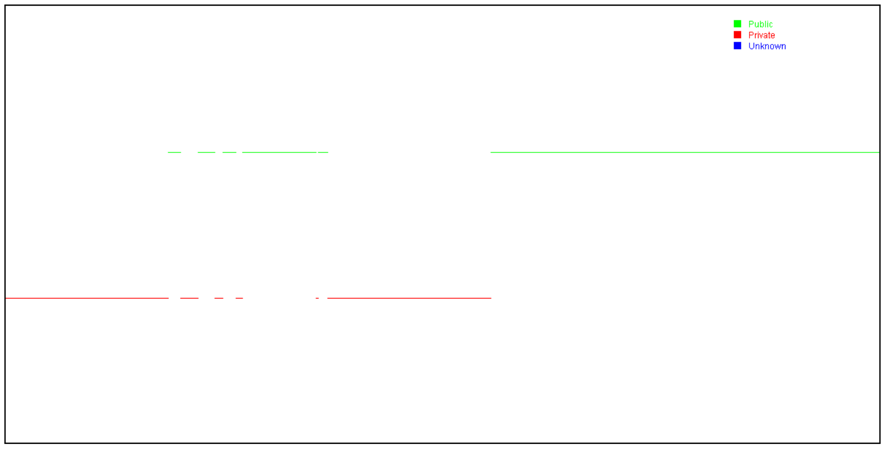


Figure 5.4: Visualization of the presence of the client in different zones.

TrackingState Joint 1	TrackingState Joint 2	Weight
Tracked	Tracked	2
Tracked	Inferred	1
Inferred	Tracked	1
Other		0

Table 5.1: Overview of the weights that are used when computing the activity level of a Skeleton. Joint 1 is a Joint of certain type, Joint 2 is a Joint of the same type, but recorded one frame later than Joint 1.

Trajectory The trajectory is based on the sequence of locations returned by the Kinect. To incorporate a global timing aspect, the first part of the trajectory is drawn in blue, the middle part in red, and the last part in pink, see Figure 5.2. Additionally we can visualize the trajectory in an animation which draws the walked trajectory incrementally over time. The trajectory can be used to get an idea about the order in which different parts of the room are visited.

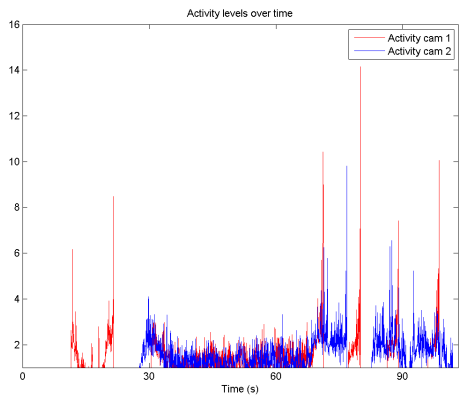
Heat map The heat map shows which parts of the room are visited more frequently, see Figure 5.3. The heat map is defined as a 600 by 400 zeros matrix as the room is 6 by 4 meters. It is computed by iterating over all locations and increasing the number at (heatmap[x][y]) with 1 for the location (x, y). The heat map can be used by the staff to see if the client is walking around the room, or more focused at one specific location.

Zones The support room can be divided in a public and private zone. For the staff it is interesting to see whether a client is more in the public or more in the private zone. Therefore we implemented a graph that shows whether the person is in the public or private zone, or whether this is not known because no data was recorded. An example is shown in Figure 5.4.

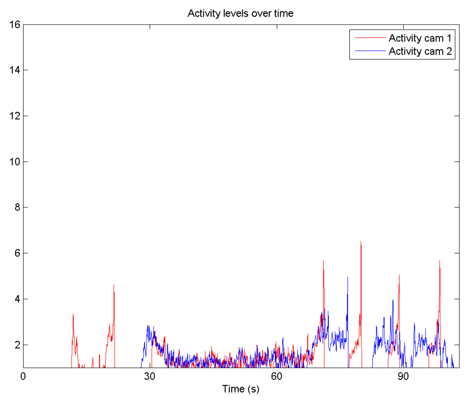
5.2.2 Visualization of skeleton data

The skeleton data is visualized by animating the movements of the person as a stickfigure (see Figure 4.5 on the screen. Besides that, we computed the displacement of each joint over time. This was plotted for every individual joint in each direction, summed over the three directions, and for each body part (head, arms, legs, core). As these plotting functions resulted in unclear images we decided to compute a single value to represent the movement of the entire skeleton for each time point. This value is called the *activity level*.

Activity level The activity level is the average displacement of all `Tracked` and `Inferred` joints between two consecutive frames. A joint is `Tracked` when the software on the Kinect detected that joint and could track it over multiple frames. A joint is `Inferred` when the Kinect had to infer its location using the position of the other joints. Joints that are not visible for the Kinect are not included in the activity level. The displacement is computed by iterating through the list with all skeletons, starting at the second frame. Each frame holds 1 skeleton, containing the location and tracking state of all 20 joints. For each joint we consider the location of that joint in the current and previous frame. The displacement is the Euclidean distance between the locations in the two frames. The activity level is the weighted average of all tracked and inferred joints. The weights are shown in Table 5.1. Initially we plotted the raw activity levels. Since the raw activity



(a) Raw activity levels.



(b) Median filtered activity levels.

Figure 5.5: Raw (left) and filtered (right) activity levels.

levels are changing rapidly over time, we smooth the data by applying a one-dimensional median filter to the data. Smoothing will give an overview of the general course of the activity level. The median filter computes a value $y(k)$ as the median of $x(k-(n-1)/2 : k+(n-1)/2)$. We use the default $n = 3$ which gives us $y(k)$ as the median of $x(k-(3-1)/2 : k+(3-1)/2) = x(k-1 : k+1)$ (The Mathworks, 2013). Figure 5.5 displays a plot of both raw and median filtered activity levels.

Chapter 6

Activity Recognition

The goal of activity recognition is to recognize higher-level semantic activities from lower-level data. We record the 2D-location of the client in a room and from this information we want to know *what* the client is doing. We need a way to represent the higher-level activities we want to recognize in terms of the lower-level data. We also need a way to recognize the higher-level activities from the lower-level data. We use a logic-based approach to recognize the higher-level activities. In our approach we put the lower-level activities in a knowledge base. One of the main advantages of a logic-based approach is that we can also incorporate background knowledge. Background knowledge is important as it can make the recognition process more focused. For example; we do not have to recognize that a person is walking towards the kitchen when we know there is no kitchen. Another advantage of a logic-based approach is that we can reason about the activities in an understandable manner.

6.1 Logic-based activity recognition

We use a logic-based approach to recognize the higher-level, or long-term activities (LTA). The lower-level, or short-term activities (STA) are determined during the processing of the Kinect data and depend on the difference in x, y-location of the client between two frames. This will be discussed in Section D. The long-term activities depend on a combination of the short-term activities with additional constraints. For a graphical overview see Figure 6.1.

6.1.1 Prolog

Prolog is a logic-based programming language. It does not specify how to get to a solution, but defines the constraints a solution has to satisfy. This can be done by storing *facts* in a *database* or *knowledge base*. With the use of *rules* we can reason about the facts in the knowledge base, and determine whether the provided solution satisfies all constraints. A term, or fact, is written in lowercase, for example `banana`. It can be used to substitute a variable `X` in `fruit(X)` to get `fruit(banana)`. When a variable is not substituted it is written in uppercase. We can also define *structures*, which represent relations between atoms. For example the relation `fruit(apple)`, which states that `apple` is a `fruit`. Facts and rules can be added to the knowledge base. Consider the following knowledge base:

```
fruit(apple).
fruit(banana).
fruit(coconut).
furniture(table).
edible(X) :- fruit(X).
```

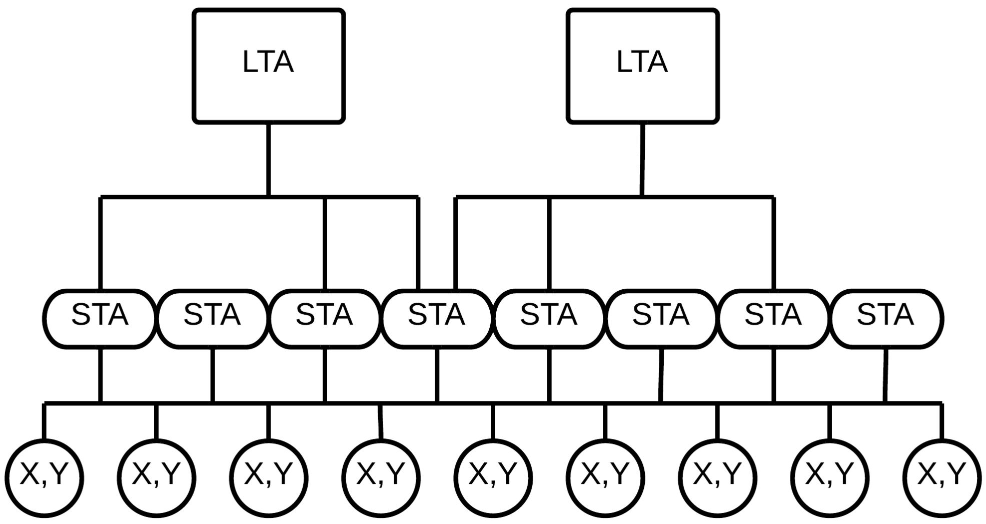


Figure 6.1: The short-term activities (STA) depend on the recorded location data x, y of the client. The long-term activities (LTA) are defined as a combination of different short-term activities with additional constraints.

In this knowledge base we can see that `apple`, `banana` and `coconut` are `fruit`, and that `table` is `furniture`. We also see a rule `edible(X):- fruit(X).`. `edible(X)` is the goal, which can be satisfied if the passed variable, `X`, is stored in the knowledge base as a `fruit`. We say that the variable `X` will be unified with an atom, for example `apple`, when this is present in the knowledge base, or can be derived from the facts in the knowledge base. When we query the knowledge base with `?- edible(X).`, this will return all possible unifications of `X` that satisfy the constraints for an atom being edible. In our definition something has to be `fruit` to be edible. Calling `edible(X)` will result in the solutions:

```
?- edible(X).
X = apple ? ;
X = banana ? ;
X = coconut
```

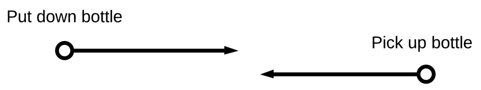
We can also query the knowledge base to see if a specific unification of `X` is edible, for example:

```
?- edible(apple).
yes
```

This results in the answer `yes`, which means that this unification of `edible(X)` can be matched to the facts and rules in the knowledge base. If we would ask `edible(table)` we get the answer `no`. According to our knowledge base only fruit is edible, and a table is not fruit; hence it is not edible. If we would ask another atom, for example `edible(cauliflower)`, we also get `no` for an answer, as this is not stated in the knowledge base. A goal, such as `edible(X)` is satisfied when all its subgoals, in this case `fruit(X)` can be matched to the knowledge base. Usually there is more than 1 subgoal. In that case, the goal is satisfied when all its subgoals are satisfied as well.

6.1.2 Event Calculus

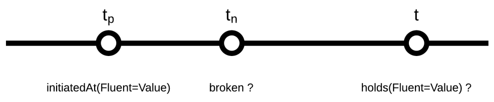
The Event Calculus is a logic-based language, first formalized in 1986 by Kowalski and Sergot. It can be used to reason about events in time and their effects. It is based on *actions* or events, and *fluents*. The value of a fluent can change over time. An example is the fact `the bottle is on the table`, this can be either true or false. This fluent is true when someone executes the action `put down bottle` and will be false when someone executes the action `pick up bottle`. We can say that the action `put down bottle` initiated a period of time for which `bottle on table` is true, and the action `pick up bottle` will terminate such a period of time. For the time in between, we assume that the bottle is on the table. More formally we can say:



```
initiates(put_down_bottle, bottle_on_table).
terminates(pick_up_bottle, bottle_on_table).
initiatedAt(bottle_on_table = true, T) :-
    initiates(put_down_bottle, bottle_on_table),
    happensAt(put_down_bottle, Tp),
    Tp < T.
initiatedAt(bottle_on_table = false, T):-
    terminates(pick_up_bottle, bottle_on_table),
    happensAt(pick_up_bottle, Tn),
    Tn < T.
```

To determine if the bottle is on the table at a certain time `T`, we can check if the fluent `bottle on table` "holds" at that time `T`:

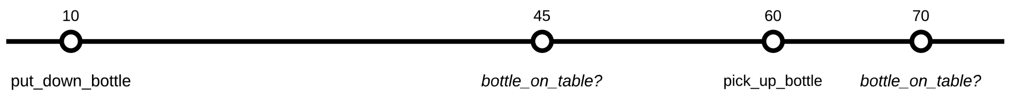
```
holdsAt(Fluent = Value, T).
holdsAt(bottle_on_table = true, T).
```



A fluent holds, or is true, at a time `T` if it was initiated at a time `Tp` before time `T`, and it was not broken at any time `Tn` between time `Tp` and `T`. A fluent is broken when it is initiated with another value or when it is terminated. We can formulate this in Prolog as:

```
holdsAt(Fluent = Value, T) :-
    initiated(Fluent = Value, Tp),
    not broken(Fluent = Value, Tn, T).
```

Our knowledge base consists of rules stating which actions initiate or terminate a fluent. Besides that, we also state which actions occur at which point in time. Then we can use the reasoning mechanism in Prolog to derive if a fluent holds at a certain time. As noted before, to determine whether `holdsAt(Fluent=Value, ... T)` can be derived from the facts in the knowledge base, we have to check if we can derive its subgoals `initiated(Fluent = Value, Tp)` and `not broken(Fluent = Value, Tn)` from the knowledge base. Consider the following example where we put down the bottle at 'time 10' and pick up the bottle at 'time 60'. We want to know whether the bottle is on the table at 'time 45' and at 'time 70'.



```
holdsAt(Fluent = Value, T) :-
    initiated(Fluent = Value, Tp), not broken(Fluent = Value, Tn, T).
broken(Fluent = Value, Tn, T) :-
    terminated(Fluent = Value, Tn), Tn < T.
initiates(put_down_bottle, bottle_on_table).
terminates(pick_up_bottle, bottle_on_table).
initiatedAt(bottle_on_table = true, T) :-
    initiates(put_down_bottle, bottle_on_table),
    happensAt(put_down_bottle, Tp),
    Tp < T.
initiatedAt(bottle_on_table = false, T):-
    terminates(pick_up_bottle, bottle_on_table),
    happensAt(pick_up_bottle, Tn),
    Tn < T.
happensAt(put_down_bottle, 10).
happensAt(pick_up_bottle, 60).

?- holdsAt(bottle_on_table, 45).
?- holdsAt(bottle_on_table, 70).
```

The first query, `?- holdsAt(bottle_on_table, 45)`, checks if `bottle_on_table` was initiated at a timepoint before the current timepoint, 45. In the knowledge base we see that the action `put_down_bottle` happens at time 10. With this action, the fluent `bottle_on_table` was initiated, and therefore the first subgoal can be matched to the knowledge base. Then Prolog checks the second subgoal; `not broken(bottle_on_table, ... Tn)`. A fluent is broken at a time `T` when it is terminated at a time `Tn` before time `T`. In our knowledge base we do not have a fact which states that `bottle_on_table` was terminated at at timepoint between 10 and 40. Therefore `bottle_on_table` is not broken, and the second subgoal can be matched to the knowledge base as well. As both subgoals of `holdsAt(bottle_on_table, 45)` can be matched to the knowledge base, we can conclude that the bottle was on the table at time 45.

To check if the bottle is on the table at 'time 70', we can match the first subgoal to the facts in knowledge base. When we want to match the second subgoal, `not broken(bottle_on_table, Tn, 70)`, we have to **not** match the following goals to the knowledge base:

```
broken(bottle_on_table = true, Tn, 70) :-
    terminated(bottle_on_table, Tn), Tn < 70 .
```

In our knowledge base we also have the action `pick_up_bottle` executed at time 60, this can substitute `Tn`. As `pick_up_bottle` terminates `bottle_on_table`, and this is occurring before the timepoint we are investigating, the fluent `bottle_on_table` is broken. Therefore we cannot derive that `bottle_on_table` holds at time 70.

6.1.3 Event Calculus for activity recognition

Because the Event Calculus can be used to reason about events in time and their effects, it can be used for activity recognition as well. An example of this can be found in the work by Artikis et al. (2010), who used the LTAR-EC dialect. Long-term activities are defined as a temporal combination of short-term activities, and are the goals we want to derive from the information in the knowledge base. The short-term activities are put into the knowledge base, such as the actions or events that were discussed before. The long-term activities can be recognized by applying the Event Calculus on the knowledge base containing the recognized short-term activities and the defined long-term activities. The main predicates that are used by Artikis et al. are given in Table 6.1. The Event Calculus was applied to the CAVIAR dataset (EC Funded CAVIAR

Predicate	Meaning
`happensAt(E, T)`	Event `E` occurs at time `T`
`happensFor(E, I)` *	`I` is the list of maximal intervals for which event `E` takes place
`initially(F=V)`	`F` has value `V` at time 0
`holdsAt(F=V, T)`	At time `T` `F` has value `V`
`holdsFor(F=V, I)` *	`I` is the list of maximal intervals for which `F=V` holds
`initiatedAt(F=V, T)`	At time `T` `F=V` is initiated
`terminatedAt(F=V, T)`	At time `T` `F=V` is terminated

Table 6.1: Main predicates for the LTAR-EC dialect as presented in (Artikis et al., 2010) and the Crisp-EC dialect as presented in (Skarlatidis et al., 2014). Predicates indicated with * are only present in the LTAR-EC dialect.

project/IST 2001 3750, 2001), which is often used for activity recognition research. The short-term activities were annotated in this dataset and consisted of *walking*, *running*, *inactive* and *abrupt motion*. The short-term activities are put into the knowledge base as an event with the `happensAt` predicate. The long-term activities consisted of *leaving an object*, *meeting*, *fighting*, and *moving*. The long-term activity *meeting* is recognized when two persons are within 25 centimeters of each other, while the first person is inactive and the second person is not running or moving abrupt. This is defined as follows:

```
initiatedAt(meeting(P1, P2)=true, T) :-
holdsAt(close(P1, P2, 25) = true, T),
holdsAt(person(P1), T),
happensAt(inactive(P1), T),
holdsAt(person(P2), T),
not happensAt(running(P2), T),
not happensAt(abrupt(P2), T).
```

The long-term activity is written as a fluent and defined in terms of the short-term activities and additional constraints, like the distance between two objects (`close` predicate). The recognition of this long-term activity is executed by querying the knowledge base with the `holdsAt(meeting(P1, P2)=true, T)` predicate. This can be done for each combination of people, `P1` and `P2`, and for all timepoints `T`. The same can be done for the other fluents. When the variables can be successfully substituted with the facts in the knowledge base, the long-term activity will be recognized. To make the recognition of long-term activities faster and less computationally expensive, it is possible to assert the recognized long-term activities to the knowledge base as well. When the program is substituting the fluent of a long-term activity for the next time point, we only have to check whether the long-term activity was not broken.

6.2 Definitions of activities

We will adapt the work by Artikis et al. (2010) to recognize activities from the recorded data in our own application. Their code can be found on the website of Artikis (2012) [1]. We consulted the staff about the activities to recognize. Staff members indicated that it would be interesting to know when the client was either continuously pacing around or not moving at all. These kinds of behavior can be indications that a client is going into a psychosis. As the support room is divided into a public and a private zone, it would also be interesting to see whether the clients are using this distinction. It would also be helpful to recognize specific behavior patterns. For an artificial intelligence perspective it is interesting to detect a clients activities in a more semantic sense, for example that he is constantly moving between the bathroom and the window. Based on these indications we decided to recognize the following activities:

[1] `http://users.iit.demokritos.gr/~a.artikis/EC.html`

- Continuously walking around
- Not moving at all
- Being at a specific location in the room
- Walking towards a location in the room
- Walking away from a location in the room

The specific locations depend on the floor plan of the room. As an extension to the recognized activities, we also defined more complex activities that built upon the recognition of the activities mentioned above.

- Move from one location to another
- Read the article for a longer time

These activities are defined in terms of previously recognized short-term activities.

6.2.1 Short-term activities

The Event Calculus uses events or short-term activities to recognize the activities. Artikis et al. (2010) used short-term activities such as `abrupt motion`, `walking`, `running`, `active` and `inactive`. We reduced the number of short-term activities and defined three, `walking`, `active` and `inactive`. We recognize the short-term activities based on the displacement, or traveled distance, of the client in the room. When the client is not moving at all, the short-term activity is `inactive`. When he is moving a bit, but remains in the same place, the short-term activity is `active`. When the client is moving around, the short-term activity is `walking`.

To compute the displacement we can consider different time spans. We implemented four different options.

- **Window-3** The `Window-3` option considers the frame before and after the current frame, to determine the traveled distance in the current frame.
- **Second** The `Second` option looks at the displacement in each second. We determine the traveled distance between every 30th frame.
- **FPS2** The `FPS2` option determines the traveled distance between every 15th frame. As we record with 30 frames per second, this results in 2 *f*rames *p*er *s*econd.
- **FPS3** `FPS3` determines the traveled distance between every 10th frame.

The `Window-3` option is the most computationally expensive, as it considers every frame 3 times. The `Second`, `FPS2` and `FPS3` options only consider every 30th, 15th, and 10th frame. For our "proof of concept" application we decided to look at either `FPS2` or `FPS3`, as they provide enough detail. An application in the support room can use the `Second` option as well, as we expect to look at hours or even days of data.

The short-term activity is based on the distance that was covered in the specified time span. When this distance is below the *inactive threshold*, the short-term activity will be `inactive`, when the distance is above the *walking threshold*, the short-term activity will be `walking`. The short-term activity will be `active` when it is between the *inactive* and *walking* threshold. These thresholds differ for the different time spans, as we assume that a person can cover a larger distance in one second, than in 1/30th of a second (`Window-3` option). The thresholds are determined by computing the average displacement in a separate recording in which each short-term activity was performed. Table 6.2 provides indications for the

Frame Processing	**Inactive**	**Walking**
Window-3	1	3
Second	5	20
FPS2	5	15
FPS3	2	7

Table 6.2: Proposed inactive and walking thresholds.

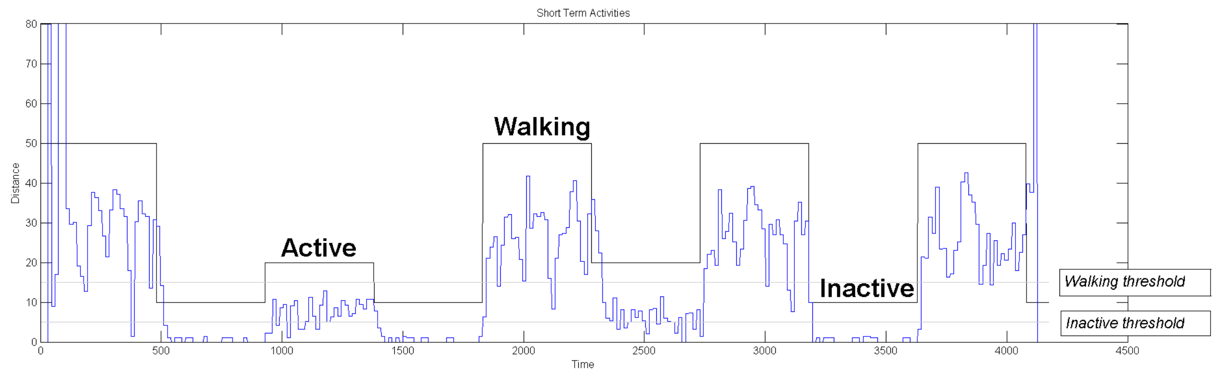


Figure 6.2: Distances computed with the FPS2 settings (blue) and the annotated short-term activities (black). The gray lines depict the *inactive*- and *walking*-thresholds. In the annotated data, a value of 10 indicates *inactive*, 20 is *active*, and 50 is *walking*.

thresholds given the different time spans. Initial estimates were based on an assumption about the distance that can be traveled in the number of frames. The final thresholds were determined by examining the computed distances for the performed short-term activities. An example is given in Figure 6.2.

6.2.2 Long-term activities

The long-term activities are composed of different short-term activities. In this section we discuss the different long-term activities and the corresponding Prolog code.

Long walking The *long walking* activity is defined to recognize a client pacing around in the room. It is initiated when the client has been walking for at least 10 (`walkingCount(Count)`) of the past 15 (`walkingTime(R)`) frames. It is terminated when the client was not walking in at least 10 of the past 15 frames.

```
initiatedAt(longWalking(ID)=true, T):-
    walkingTime(R),
    happens_n_of_r_frames(T, walking(ID), R, N),
    walkingCount(Count),
    N>Count.

initiatedAt(longWalking(ID)=false, T):-
    walkingTime(R),
    happens_n_of_r_frames(T, walking(ID), R, N),
    walkingCount(Count),
    \+N>Count. %less than or equal to walkingCount frames walking
```

Long immobile The *long immobile* activity is defined to recognize a client who is not moving at all. It is initiated when the client was inactive in at least 10 (`inactiveCount(Count)`)of the past 15 (`inactiveTime(R)`) frames. It is terminated when the client was inactive in less than 10 of the past 15 frames.

```
initiatedAt(longImmobile(ID)=true, T):-
    inactiveTime(R),
    happens_n_of_r_frames(T, inactive(ID), R, N),
    inactiveCount(Count),
    N>Count.

initiatedAt(longImmobile(ID)=false, T):-
    inactiveTime(R),
    happens_n_of_r_frames(T, inactive(ID), R, N),
    inactiveCount(Count),
    \+N>Count. %less than or equal to inactiveCount frames inactive
```

At location We can provide more understanding of the activities of a client if we know if the client is at a specific location in the room. This location is a variable part of the fluent. A client is considered to be at a location when his distance to that location is less than or equal to the specified distance `locDist`. This activity is terminated when the distance to the location is larger than the specified `locDist`.

```
    initiatedAt(atLoc(Id, Loc)=true, T):-
        locDist(Dist),
        holdsAt(close(Id, Loc, Dist)=true, T).

    initiatedAt(atLoc(Id, Loc)=false, T):-
        locDist(Dist),
        holdsAt(close(Id, Loc, Dist)=false, T).
```

Walking towards location To recognize that a client is walking towards a certain location he has to be close to this location, within `locCloseDist`, but not at that location. Besides that, the client has to be walking, and the distance to the location should be decreasing compared to the previous frame.

```
    initiatedAt(walkingTowardsLoc(Id, Loc)=true, T):-
        \+holdsAt(atLoc(Id, Loc)=true, T),
        locCloseDist(Dist),
        holdsAt(close(Id, Loc, Dist)=true, T),
        happensAt(walking(Id), T),
        holdsAt(distance(Id, Loc)=D1, T),
        consecutive(Tp, T),
        holdsAt(distance(Id, Loc)=D2, Tp),
        D2>D1. %distance decreasing
```

The activity will be terminated when either of these constraints is not satisfied.

```
    initiatedAt(walkingTowardsLoc(Id, Loc)=false, T):-
        holdsAt(atLoc(Id, Loc)=true, T).
    initiatedAt(walkingTowardsLoc(Id, Loc)=false, T):-
        \+happens(walking(Id), T).
    initiatedAt(walkingTowardsLoc(Id, Loc)=false, T):-
        holdsAt(distance(Id, Loc) = D1, T),
        consecutive(Tp, T),
        holdsAt(distance(Id, Loc)=D2, Tp),
        \+D2>D1. %D1 smaller than or equal to D2
    initiatedAt(walkingTowardsLoc(Id, Loc)=false, T):-
```

```
        locCloseDist(Dist),
        holdsAt(close(Id, Loc, Dist)=false, T).
```

Walking away from location To recognize that a client is walking away from one of the locations, he should have been at that location for at least 1 frame in the past 10 frames. Besides that, he has to be walking, close to the location, but not at the location, and the distance to the location has to be increasing compared to the previous frame.

```
    initiatedAt(walkingAwayFromLoc(Id, Loc)=true, T):-
        holds_n_of_r_frames(T, atLoc(Id, Loc)=true, 10, N),
        N>0, %at least N frames at loc
        happensAt(walking(Id), T),
        \+holdsAt(atLoc(Id, Loc)=true, T),
        locCloseDist(Dist),
        holdsAt(close(Id, Loc, Dist)=true, T),
        holdsAt(distance(Id, Loc)=D1, T),
        consecutive(Tp, T),
        holdsAt(distance(Id, Loc)=D2, Tp),
        D1>D2. %distance increasing
```

When either of these constraints is not satisfied, the activity will be terminated.

```
    initiatedAt(walkingAwayFromLoc(Id, Loc)=false, T):-
        holdsAt(atLoc(Id, Loc)=true, T).
    initiatedAt(walkingAwayFromLoc(Id, Loc)=false, T):-
        \+happensAt(walking(Id), T).
    initiatedAt(walkingAwayFromLoc(Id, Loc)=false, T):-
        locCloseDist(Dist),
        holdsAt(close(Id, Loc, Dist)=false, T).
    initiatedAt(walkingAwayFromLoc(Id, Loc)=false, T):-
        holdsAt(distance(Id, Loc)=D1, T),
        consecutive(Tp, T),
        holdsAt(distance(Id, Loc)=D2, Tp),
        \+D1>D2. %D2 smaller than or equal to D1
```

Move between two locations The recognition of a move between two locations depends on the recognition of the `walkingTowardsLoc` and `atLoc` activities. A client had to be at the -from- location in at least 1 of the 10 past frames. Besides that, he has to be walking towards the -to- location at this time. In the past 10 frames, we also require the client to be walking in 3 or more frames. When the client was active in the past frames this activity can still be recognized. For the `walkingTowardsLoc` activity the client has to be walking.

```
    initiatedAt(move(Id, X, Y)=true, T):-
        place(X),
        place(Y),
        \+ X=Y,
        holdsAt(walkingTowardsLoc(Id, Y)=true, T),
        holds_n_of_r_frames(T, atLoc(Id, X)=true, 10, N),
        happens_n_of_r_frames(T, walking(Id), 10, M),
        M>2, %at least M frames walking
        N>0. %at least N frames atLoc
```

A move will be terminated when the client is at the -to- location, was not at the -from- location during the past 10 frames, or is inactive.

```
   initiatedAt(move(Id, _, Y)=false, T):-
      \+holdsAt(atLoc(Id, Y)=true, T).
   initiatedAt(move(Id, X, _)=false, T):-
      holds_n_of_r_frames(T, atLoc(Id, X)=true, 10, N),
      N<10. %N is the number of frames client was atLoc(X); so N<R -> ...
         client was not atLoc(X)
   initiatedAt(move(Id, _, _)=false, T):-
      happensAt(inactive(Id), T).
```

Reading article This activity is applicable in our "proof of concept" tests but it can also be applied to the support room, for example when the client is staring out of the window. This activity is initiated when the client was at the article (or window) for at least 44 (`LocTime is R-1`) of the past 45 (`readingTime(R)`) frames. Besides that, the client can not be walking in any of those 45 frames.

```
   initiatedAt(readingArticle(Id)=true, T):-
      readingTime(R),
      holds_n_of_r_frames(T, atLoc(Id, article)=true, R, L),
      LocTime is R-1,
      L>LocTime,
      happens_n_of_r_frames(T, walking(Id), R, I),
      I<1.
```

This activity will be terminated when the client is not at the article or starts walking.

```
   initiatedAt(readingArticle(Id)=false, T):-
      \+holdsAt(atLoc(Id, article)=true, T).
   initiatedAt(readingArticle(Id)=false, T):-
      happensAt(walking(Id), T).
```

Part III

Results

Chapter 7

"Proof of Concept"

To test our system, we depend on the cooperation of the mental healthcare services in Eindhoven. During the project, staff had to get acquainted with the new technologies in the High Care Unit. Therefore we could not yet test the system at the support room in the High Care Unit, but we recorded data with healthy participants in an office room as a "proof of concept".

Before we tested the recognition of activities with the Event Calculus on recorded data, we tested the concept on manually generated data. We generated the data with a program in which we can simulate the movements of a person by dragging the mouse over the floor plan of the support room. The resulting data file contains the (x, y)-coordinates of the walked trajectory in the room. This is the same data we get when recording the location data of a real person. As we did not simulate the `Skeleton` data, we cannot compute the activity levels. The file containing the (x, y)-coordinates can be processed by the Java program to generate the visualizations and Prolog files. We simulated a client walking towards the bed, stay at the bed, and then walk away from the bed. This sequence was repeated once. We applied the Event Calculus on the data file and were able to recognize the performed activities. After this, we set up a proof of concept to recognize activities in recorded data.

7.1 Test setup

In our experiment we simulated an environment in which we let participants perform various activities. This section presents the setup of the environment and the processing steps. The sessions were recorded in an experiment room at Philips Research. An overview of the room is shown in Figure 7.1. The room is 2.5 by 3 meters and we furnished it with a chair, a note on the wall, and an article on the wall. The Kinect sensors were placed in a corner of the room (top-left in floor plan) on hip height. The placement of the Kinect sensors ensures that the union of the fields of view of both sensors spans the entire room. In one of the corners there was a table.

5 healthy participants, all intern or employee at Philips Research, were asked to perform the tasks in the room. Two of the participants were asked to perform the tasks again, as we noticed that we lost data during the first recording. Another participants returned later that day and performed the tasks again. This resulted in 8 recordings. In 5 of these recordings, the participants followed the schedule listed below. In the 3 other recordings, the participants performed all the activities in a self-chosen order.

```
Walk to chair
Walk to note
Walk to article
Read article
Walk 3 circles in the room
```

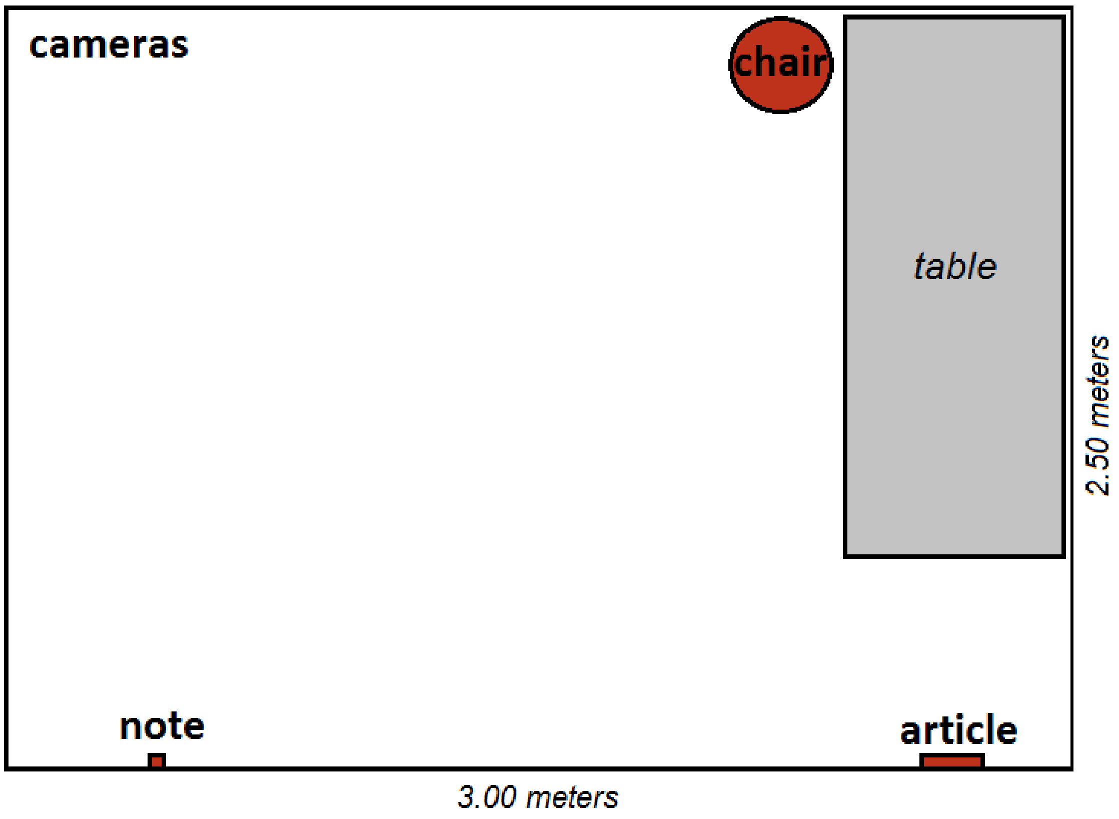


Figure 7.1: Floor plan of the room in which we recorded our experiments. It was furnished with a chair, a note on the wall and an article on the wall. The gray square in the corner is a table and participants could not access that location.

7.1.1 Preprocessing

Calibration We execute the calibration procedure (see Section 4.2) with one of the participants. She was asked to stand on the predefined locations in the room while the experimenter wrote down the sensor readings. The resulting angles are -0.5586 radians for the first Kinect and -1.1184 radians for the second Kinect. For participant `G2` the recording angles are -0.4655 radians for Kinect 1 and -1.0945 radians for Kinect 2.

Preprocessing For all recordings locations were transformed with the determined angles. In the `W2` data, the wall behind the table was recognized as a person. Participants could not access this location with a depth-reading of 3.17 meters, therefore we removed depth-readings larger than 3.17 meters as noise. The data from both Kinect sensors were merged based on the method described in Section 5.1.3. Trajectory, heap map and zones were visualized for each Kinect and for the merged data. The *activity levels* were computed for each Kinect sensor separately. The data was down-sampled to 2 frames per second (`FPS2` option in Section D), which provided a sufficient level of detail for our application. Short-term activities were recognized with an *inactive threshold* of 5 and a *walking threshold* of 15.

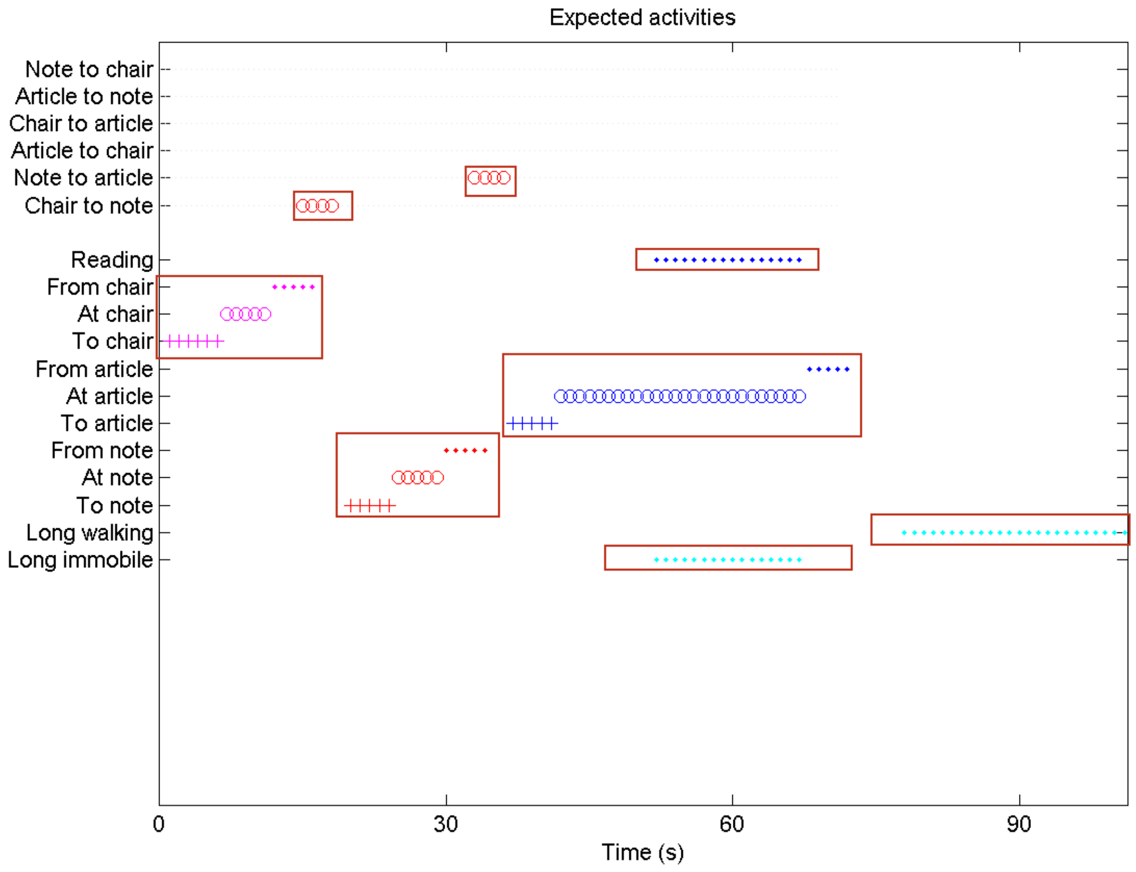


Figure 7.2: Expected results for participants who performed the activities in the fixed order.

7.1.2 Activity recognition

Based on the tasks given to the participants we can recognize the long-term activities listed in Table 7.1. For the participants who performed the tasks in the fixed order we expect to see a visualization as shown in Figure 7.2. For the participants who performed the tasks in self-chosen order we expect to recognize the listed activities in a different order. To determine if a participant was at a location, we used three different values of `locDist`; 30, 40 and 50 centimeters. The `locCloseDist` value, to determine if a participant was close to a location, was kept constant at 70 centimeters.

Walking to chair
Being at chair
Walking away from chair
Move from chair to note
Walking to note
Being at note
Walking away from note
Move from note to article
Walking to article
Being at article
Long immobile
Reading article
Walking away from article
Long walking

Table 7.1: Activities performed by participantss

7.2 Results

7.2.1 Activities

Figure 7.3 shows the results for participant R, who performed the activities in the fixed order. He had to walk to the chair, to the note, read the article, and walk around the room. The blocks in the graph indicate where to expect the specified activities. In the graph we see that the participant

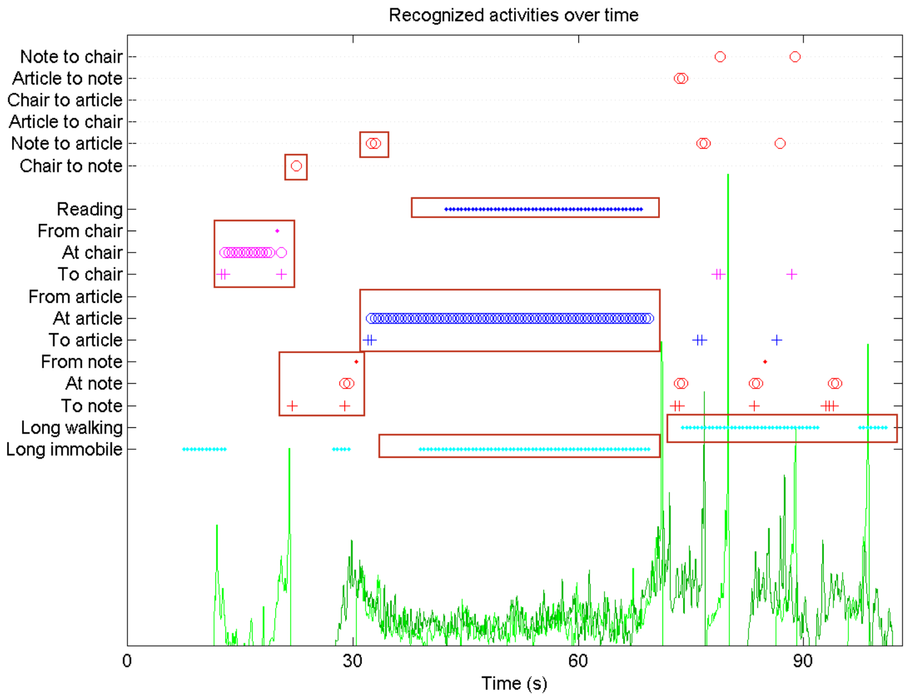


Figure 7.3: Recognized activities for one of the participants who performed the activities in fixed order.

walked to the chair, was at the chair and walked away from it. We see that the participant moved from the chair to the note and that he walked to the note. There are only two recognitions of the participant walking towards the note. The activity levels show that there was no activity recorded between those points. They also show that the first recognition of walking to note recorded by the first Kinect (light green), while the second recognition was recorded by the other Kinect (dark green). This suggests that the participant walked out of the view of the first Kinect and into the view of the second Kinect while he was walking towards the note. We can also see that the participant was at the note, walked away from the note and walked towards the article. The move between note and article was recognized as well. When the participant is reading the article, we also recognize the long immobile activity and we see that the activity levels are lower in comparison with the activity levels of the rest of the recording. We did not recognize that the participant walked away from the article. When the participant is walking around the room we recognize the long walking activity. The activity levels show a larger variation, and the source of the activity levels changes when the participant is walking around. Additionally we see a recognition of walking to note, article and chair with the corresponding moves. We did not expect to see these results. We assume that the participant was getting too close to the different locations while he was walking around the room.

Fixed order activities The left part of Table 7.2 shows the recognition results for the participants who performed the tasks in the fixed order. For all 5 participants we recognized that they were walking to the note and article, and that they were at the article. For 4 of the 5 participants we recognized that they ware walking around, walking to the chair, or at the note. For 2 of the 5 participants we recognized that they were at the chair, moving from chair to note and from note to article. In 1 of the recordings we recognized that the participant was walking away from chair or note, and that the participant was reading the article. For none of the participants we recognized that they were walking away from the article.

Participants →	Fixed order						Self-chosen order				
Activities ↓	G	M	R	W1	M2		J	W2	G2		
To chair		x	x	x	x	**4**	x	x	x	**3**	**7**
At chair			x		x	**2**				**0**	**2**
From chair			x			**1**				**0**	**1**
To note	x	x	x	x	x	**5**	x	x	x	**3**	**8**
At note		x	x	x	x	**4**		x	x	**2**	**6**
From note			x			**1**			x	**1**	**2**
To article	x	x	x	x	x	**5**	x	x		**2**	**7**
At article	x	x	x	x	x	**5**	x	x		**2**	**7**
From article						**0**				**0**	**0**
Reading article			x			**1**	(x)			**0**	**1**
Walking around	x	x	x	x		**4**	x	x	x	**3**	**7**
Chair to note			x		x	**2**	-		-	**0**	**2**
Note to article			x		x	**2**	-	x	-	**1**	**3**
	4	6	12	6	8	36	5	7	5	17	53

Table 7.2: Recognized activities for all participants for a locDist of 30 centimeters.

Activities performed in self-chosen order The self-chosen order recordings are discussed in more detail. The recognition results are given in the right side of Table 7.2. The visualized results for these recordings are given in Appendix G.

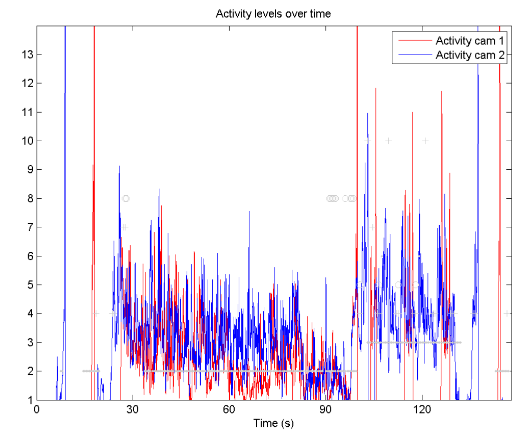
Participant `W2` first walked around, before he walked to the chair, the note, and the article. The results are shown in Figure G.1. For this participant we recognized 7 of the 13 activities.

Participant `J` first read the article, then walked to the note, after that he walked around, then he moved to the chair, and walked around again. The results are shown in Figure G.2. We recognized 6 of the 12 possible activities. As this participant did not move from chair to note or from note to article we can not recognize these activities. The participant did move from article to note, which we did recognize.

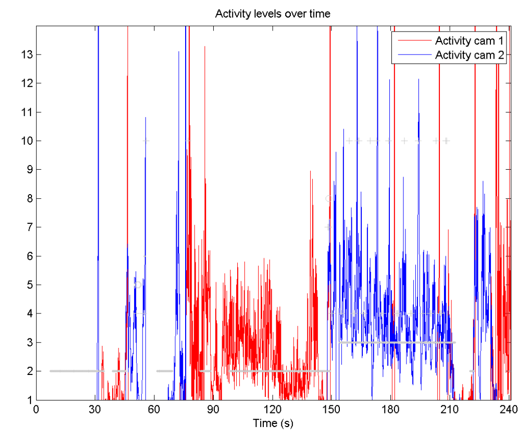
Participant `G2` first walked to the note, then to the chair before going to the article. After he had read the article he walked around the room. While he was walking around the room he was heavily shaking his arms and legs, as if he was getting aggressive and wanted to attack someone. The recognition results are shown in Figure G.3. We recognized 6 of the 13 activities. The data recorded with participant `G2` were also used to investigate the influence of movement of arms and legs on the activity levels. Figure 7.4 shows the activity levels for the first and second recording of this participant. Note that both recordings differ in time span. The line at $y = 2$ indicates the *long immobile* activity, and the line at $y = 3$ indicates the *long walking* activity. There is a difference in activity level between walking and standing still. Unfortunately there is not a higher activity level when the participant is waving his arms and legs in comparison to the activity levels when he is walking around. This suggests that we cannot use the raw values of the activity levels to indicate more movement compared to walking. We can use the activity levels to indicate whether data was recorded, and which Kinect was recording the data.

7.2.2 Threshold values

To investigate the influence of `locDist`, the maximum allowed distance between the client and a location, we also recognized the activities with a `locDist` of 40 and 50 centimeters. Table 7.3 shows the summarized recognition results for different values of `locDist`. With an increased `locDist` the recognition of at location increased as well. The recognition of reading article increased as well, as the participants were recognized at article more often and for a longer period of time. The recognition of walking to and walking away from a location decreased. In the `W2` data for example, we recognized to note and at note for a `locDist` of 30 and 40 centimeters, but not for a `locDist` of 50 centimeters. We recognized at note instead of to note. An overview of the recognized activities for the different values of `locDist` is given in Table 7.2 in Appendix G.



(a) First recording.



(b) Second recording.

Figure 7.4: Raw activity for first and second run of participant G.

	30	40	50	*Sum*	*Average*
To chair	7	6	6	19	6,33
At chair	2	2	6	10	3,33
From chair	1	0	1	2	0,67
To note	8	7	6	21	7
At note	6	8	8	22	7,33
From note	2	1	1	4	1,33
To article	7	6	6	19	6,33
At article	7	7	7	21	7
From article	0	0	0	0	0
Reading article	1	4	6	11	3,67
Walking around	7	7	7	21	7
Chair to note	2	1	2	5	1,67
Note to article	3	3	4	10	6,33

Table 7.3: Number of participants for which an activity was recognized for the different values of locDist.

7.3 Discussion of the results

In this section we will discuss the recognition of the different activities in more detail.

To note For all participants we were able to recognize that they were walking *to note*.

To article, at article We recognized *to article* and *at article* of 7 of the 8 participants. We were not able to recognize these activities for participant `M2`, as we did not record data when she performed those activities.

Walking around The same holds for the *walking around* activity; we did not record data for participant `M2` when she was walking around the room. For all other participants we were able to recognize that they were *walking around* the room. We assessed this activity based on the recognition of *long walking*.

At note The activity *at note* was recognized for 6 of the 8 participants; we did not recognize it for participants `G` and `J`. We were able to recognize that these participants were at note when we used `locDist` of 50 or 50 centimeters. This suggests that they were not close enough to the note to be recognized at note with a `locDist` of 30 centimeters.

To chair For 7 participants recognized the walking *to chair* activity. For the fixed-order sessions, this was the first task to perform. Participant `G` was not immediately recognized when we started recording and we did not record data of him walking to the chair. Therefore we did not recognize this activity. With a `locdist` of 40 or 50 centimeters *to chair* was recognized in 6 recordings. For `W1` we recognized *at chair* instead of *to chair*. This can be caused by the larger area for which we can recognize *at chair*. A participant who is 35 centimeters away from the location will be recognized close to that location for a `locDist` of 30 centimeters, but at that location for a `locDist` of 40 or 50 centimeters.

Move from note to article Participants `J` and `G2` did not move from note to article, therefore there are only 6 recordings in which we can recognize this activity. The recognition of a move from *note to article* depends on the recognition of *at note* and *to article*. In all 6 remaining recordings we recognized *to article* and *at note* was recognized in 5 of them. The move from *note to article* was recognized in 3 recordings. During the recordings for participants `W1` and `M` we lost track of the participants when they were at the note. The participants were not tracked before they were at article again. We did not record and could therefore not recognize the move from note to article. As *at note* was not recognized for participant `G` the move from note to article can not be recognized either.

From note The recognition of *from note* depends on the recognition of *at note*. Additionally, the participant has to be walking and the distance to the location has to increase. In 6 of the recordings the participants were recognized *at note*. From note was only recognized in 2 of these recordings, participants `R` and `G2`. For `G2` *from note* was only recognized for `locDist` 30, for larger `locDist` *at note* was recognized instead of *from note*.

We expected to recognize *from note* for participants `M2` and `W2`. When looking at the data we see that the participants were moving more than 40 centimeters per frame; resulting in a speed of approximately 1 meter per second. Therefore the participants were either at the note or too far away. *from note* can only be recognized when the participant is less than 70 centimeters (`locCloseDist`) away, but farther away than 30 centimeters (`locDist`). With the smallest value for `locDist`, 30 centimeters, the difference between the `locDist` and `locCloseDist` is 40 centimeters. As the participants were walking more than 40 centimeters per frame, they 'skipped' a timepoint at which they were not at the location, but still close enough to the location. They were walking too quickly.

A solution for this is to define a larger `locCloseDist`, making the difference between `locDist` and `locCloseDist` larger. It might also be solved by choosing a different frame processing value. For this recording we used `FPS2`, but we can also use `FPS3`. In that case, we still have a reasonable number of frames, a small number is preferred for the processing time, and we still have enough datapoints to recognize a person being close to a location, but not at that location.

At chair The *at chair* activity was recognized in 2 recordings. When the `locDist` increased to 40 centimeters we recognized *at chair* in 3 recordings. With a `locDist` of 50 centimeters it was recognized in 6 of the recordings. This suggests that participants were more than 30 centimeters away from the chair. One of the participants was not tracked before he was walking towards the note. Therefore we did not record data when he was walking towards the chair or at the chair. For another participant we see a drop in activity levels when he was at the chair.

Move from chair to note The recognition of a move from *chair to note* depends on the recognition of *at chair* and walking *to note*. *at chair* was recognized in 2 of the recordings. In these recordings we recognized the move from chair to note as well. For larger values of `locDist` we did recognize *at note*. This did not influence the recognition of the move chair to note.

From chair The *from chair* activity can only be recognized for a participant if the *at chair* activity has been recognized for that participant as well. As this activity was only recognized for 2 of the 8 participants, `R` and `M2`, we can only to recognize *from chair* for those two participants. *from chair* was only recognized for one of these recordings, but only for a `locDist` of 30 centimeters. It is possible that the participant was walking too quickly to recognize *from chair* for larger `locDist` values. We also recognized *from chair* for participant `W1` with a `locDist` of 50 centimeters.

Reading article The recognition of *reading article* requires the participant to be *at article* for at least 20 consecutive frames; which is equal to 10 seconds. Additionally, the participant is not allowed to be walking. In 7 recordings we recognized the participant *at article*. *reading article* was recognized in 1 recording with a `locDist` of 30 centimeters. When the `locDist` increased to 40 centimeters, *reading article* was recognized in 4 recordings. Increasing the `locDist` to 50 centimeters resulted in a recognition of *reading article* in 6 recordings. With an increased `locDist` value *at article* was recognized for more consecutive frames. As *reading article* requires *at article* to be recognized for at least 20 consecutive frames, the recognition of *reading article* increases when *at article* is recognized more often.

From article The recognition of *from article* depends on the recognition of *at article*, which was recognized in all recordings. In none of the recordings we recognized *from article*. When we changed the `locDist` values we still did not recognize *from article*.

It might be that the participants were walking away from the location too quickly, which was explained in the discussion of *from note* as well. In that case, *from article* might be recognized when the `locCoseDist` is increased, or when there are more frames per second, for example with the `FPS3` processing option.

Chapter 8

Discussion

8.1 Discussion

8.1.1 Summary

In this project we developed a monitoring system to recognize activities performed by clients in a support room. As we were not yet able to test the system in the support room, we performed a “proof of concept” with 5 healthy participants in an office environment. The participants had to perform 5 different tasks, resulting in 13 activities. The activities were recorded with two Kinect sensors in a corner of the room. For each participant we stored the location and the skeleton data. The skeleton data was used to compute the activity level; the average displacement of all tracked joints. The location data from both Kinect sensors was converted from Kinect coordinates to room coordinates and merged based on the timing information. The location data was used to determine the short-term activities, active, inactive or walking. Additionally we determined if the client is appearing to the scene, visible, or disappearing from the scene. The short-term activities are stored in a knowledge base, used by the Event Calculus to recognize the activities.

The participants had to walk to a chair, note and article in the room. Additionally, they had to read the article and walk around the room. The resulting long-term activities consist of walking to a location, being at a location, or walking away from it, walking from one location to another, reading the article, pacing around and standing still for a longer time. Some of those activities were easy to recognize, for example walking towards a location or walking around. The recognition of a person being at a location depended on the value of `locDist`, the maximum allowed distance to be recognized at a location. A distance of 30 centimeters was not sufficient in most recordings, while a distance of 40 or 50 centimeters was sufficient for most recordings. The recognition of walking away from a location depends on the recognition of a person at that location. Even when the recognition of a participant at a location increased, it remained hard to recognize a person walking away from a location. A possible explanation is that participants were walking too quickly to be recognized walking away from a location; they were either detected at the location or too far away from it. This might be solved by changing the number of frames per second, for example to 3 frames (`FPS3`) instead of 2 frames per second (`FPS2`). Another solution is to increase the `locCloseDist`, the maximum allowed distance to be close to a location. We also showed that we can recognize long-term activities that are based on the recognition of other long-term activities, such as a person walking from one location to another. An unexpected and unwanted result is the recognition of the walking to, being at, and walking away from location activities when the participants were walking around. A solution might be to constrain the time a person has to be at a location, or to require the participant to be active or inactive before he can be recognized at that location.

8.1.2 Application in support room

The activities we defined for our “proof of concept” can be translated into activities that can be observed in the seclusion room. Instead of a client reading the article we can recognize a client staring out of the window. The recognition of walking to the note is similar to recognizing a client walking to the bathroom. The definitions of the long-term activities can be easily changed into definitions for activities in the seclusion room. The activities that were indicated by the staff, pacing around the room and entirely standing still, were included in the “proof of concept” as well. Pacing around the room was defined as long walking, which was recognized for almost all participants. Standing still was part of the reading article activity. This was not recognized for all participants as they were not standing close enough to the article. We did however recognize the long immobile activity; which is recognized when someone is standing still for at least 5 out of the past 7.5 seconds. When we want to recognize this in the seclusion room we can use a larger time span.

When developing an adaptive system, it is important to keep the wishes of the client and staff in mind. When the system is changing the environment in the room, the client and staff should always have the possibility to overrule the system. It might be possible that a system automatically changing the environment in order to calm down the client has the reversed effect, making the client more agitated.

8.1.3 Implications of the approach

We can see an influence of the chosen approach in two parts; the recording with the Kinect and the activity recognition with the Event Calculus. As the Kinect returns the location of a tracked person relative to itself, we have to convert the sensors readings to a location in the room. The location in the room can be computed using a matrix rotation with the angle under which the Kinect records the room. When this angle is not accurate enough, it will result in errors in the new locations. As the rest of the processing is based on the locations in the room, this will affect the merging, short-term activity recognition, and with that, the long-term activity recognition. When the angle is determined more accurately, the activity recognition can be improved. Another issue with the recording is that the Kinect occasionally loses track of the person in the room. When the client is only in the field of view of only one Kinect, we can not record data at that time. When the Kinect loses track of the client this shows up in the activity level, as this is 0 when no data is recorded.

The logic-based approach for the activity recognition shows promising results. We can reason about the recognized activities in an understandable way. The long-term activities can be built up by combining the short-term activities with additional constraints, or by combining other long-term activities. Besides that, we can easily change the definitions of the activities to recognize activities in the seclusion room as well. The logic-based approach also allows us to incorporate background knowledge into the knowledge base. It is also easy to incorporate data from other sensors. Another advantage of the Event Calculus is the extension to the probabilistic Event Calculus. With this approach we can deal with uncertainties in the recognized short-term activities.

8.2 Future research

We presented a prototype for a support room monitoring system which was tested in a “proof of concept”. To get from the “proof of concept” to a real experiment some additional steps have to be taken. Additionally, we present improvements to make the recognition system more robust, and we discuss some additions.

8.2.1 Next steps

Test at mental healthcare services The most important next step, is to test the system at the mental healthcare services. We can start by recording healthy participants who perform the activities in a fixed order. After that, we can test the recognition with clients from the mental healthcare services. Some adjustments have to be made in the code to recognize activities specific for the support room.

More activities Besides the adjustment of the already defined long-term activities, we have to define more activities as well. The currently defined activities were all quite simple. For the staff it is interesting to detect unexpected behavior patterns. It can be expected that a client walks towards the table, sits down and stays there for a while, before walking towards the bathroom. It is unexpected when the client is moving between the table and bathroom three times in a minute. These kinds of patterns are interesting to recognize, and could be defined in terms of multiple occurring *move* activities.

Probabilistic Event Calculus Various extensions to the Event Calculus are introduced, but one of the most interesting is the probabilistic Event Calculus, prob-EC by Skarlatidis et al. (2014). In their paper, Skarlatidis et al. present a system that recognizes human activities from video data using a probabilistic logic programming version of the Event Calculus, Prob-EC. This can be executed in ProbLog, a probabilistic version of Prolog (Raedt et al., 2007). The facts in ProbEC have a probability attached. ProbLog assumes that all facts are independent, so when a goal consists of multiple facts, the overall probability can be computed as the product of all facts. A goal, in this application a long-term activity, is recognized when the probability is above a certain threshold. When there are multiple ways to initiate an activity it is said to be *weakly* initiated. When there is only 1 way to initiate an activity it is said to be *strongly* initiated. The same holds for weak and strong termination of activities. When the initiation condition for an activity is continuously present, the probability of that activity will increase. This can be compared to the way humans recognize others activities; when there is more input or proof that someone is executing a certain activity, we are more certain that this activity is executed. Skarlatidis et al. compared the performance of the Crisp-EC dialect (similar to the LTAR-EC dialect discussed in Section 6.1.3) to the performance of Prob-EC on the CAVIAR dataset (EC Funded CAVIAR project/IST 2001 3750, 2001). They found that Prob-EC outperforms Crisp-EC when an activity is weakly initiated (multiple initiation conditions) and consists of a small number of facts. When there are high noise-levels in the recognized short-term activities, Prob-EC is at least as accurate as Crisp-EC.

This last finding is interesting for our application. As we are using a rule-of-thumb to determine the short-term activities, these short-term activities might be noisy. A reasoning approach that can handle the noise might resolve problems that arise with the recognition of the short-term activities. The probability of the short-term activity can be based on the distance to the inactive and walking thresholds; when the distance is close to these, the probability of the recognized short-term is lower as we are less certain this short-term activity is correct. When the traveled distance is further away from the inactive and walking thresholds the probability for that short-term activity is higher, as we are more certain we recognized the correct short-term activity. We can also add a probability to the recognition of the `at location` activity, as the recognition of this activity is influenced by the `locDist` threshold. When someone is standing closer to the location, we can add the `at location` activity to the knowledge base with a higher probability. When someone is farther away from the location, we can add `at location` to the knowledge base with a lower probability.

On-line processing of data The processing and activity recognition are executed offline. We can also choose to process the data while recording it. There would be a small delay as some of the processing functions depend on the differences between two frames. When we process the data online, we have to consider whether to merge the data while recording, or to process the data independently.

8.2.2 Improvements

In our program we mainly implemented quick solutions for the problems we encountered, for example in the determination of the angle under which the Kinect records the room, or for the merging of the data from two Kinect sensors. There are several suggestions for improving these methods.

Computing angle We determine the angle under which the Kinect records the room based on single recordings for multiple locations in the room. While writing down the sensor readings, we round this number. If we would write down the entire number, or take the average of the number over multiple frames, the angle can get more accurate. Besides that, we can also take more measurements for each fixed location recorded with multiple people.

Currently we map the coordinate system of both Kinect sensors onto the coordinate system of the room. It is also possible to map the coordinate system of one of the Kinect sensors onto the coordinate system of the other Kinect. Additionally we have to map the coordinate system of the room onto that of that Kinects. We only have to convert the sensor readings of one of the Kinect sensors. The locations in the room can be converted to "Kinect coordinates" beforehand. As we only have to convert the sensor readings of one of the Kinect sensors, this will reduce the computation time. A disadvantage is that the locations are less understandable for a human, as they are all relative to the Kinect.

Noise removal The noise we remove is caused by incorrectly recognized objects on a known location in the room. All sensor readings at that location are removed. It is preferred to create a smarter noise removal function. For example by examining the change in location of a tracked person, or by looking at the tracking ID of a tracked person. When a new person enters the scene, he or she will be assigned a different tracking ID.

Smart merging Two data points are merged when they are recorded at the same time by averaging the $x-$ and $y-$ coordinates for both data points. When there is an error in the computed location, for example because the angle is not accurate enough, this will influence the merged location as well. When the data points of the two Kinect sensors are not merged, they are stored separately. This can lead to jumps in the locations as well. This will influence the recognition of the short-term activities, as these are determined with the traveled distance.

A solution is to merge the data after the short-term activities have been determined. In the current implementation, the short-term activities are determined with down-sampled data. We have to keep the down-sampled timing information in mind when merging the data after the short-term activities have been determined. Additionally, the short-term activities have to be merged instead of the locations. In the probabilistic Event Calculus we can add probabilities to the short-term activities based on the two short-term activities being merged; when both are the same the probability is higher than for two different short-term activities. Another option is not merging the data at all and provide the activity recognition program with two data streams. For both data streams we need to use the same start time to ensure the Prolog facts are using the same timing information.

Recognition of short-term activities The short-term activities are recognized with a rule-of-thumb; when the traveled distance is below the *inactive* threshold, the short-term activity is *inactive*, when the distance is larger than the *walking* threshold, the short-term activity is *walking.* When the traveled distance is in between the *inactive* and *walking* thresholds, the short-term activity is *active.* The thresholds are determined by observing the computed traveled distance when we recorded the three short-term activities. The thresholds can be improved by computing the distances when other people are executing the different short-term activities. Another possibility is to add certainty values to the recognized short-term activities. We can also use a probabilistic model for the recognition of the short-term activities, for example a Hidden Markov model. The short-term activities can also be computed at a different point in the processing cycle; for example before the merging, or even before the transformation to room coordinates.

Multiple people We assume that there will be only 1 person in the support room. Therefore the data from both Kinect sensors are merged when they both track a person at the same time, as it must be the same person. In reality there might be more than 1 person in the room, for example a care taker. Although the need for a monitoring system is less urgent when a care taker is present, it is desirable that the program can handle more than one person. The activity recognition in the Event Calculus is based on a variable ID, therefore it can handle the recognition of activities for more than one person. The Kinect assigns a tracking ID to each person in its field of view. This tracking ID can be used to distinguish between two people. We have to keep this in mind when we are merging the data, as we do not want to merge the locations of two different people.

8.2.3 Additions

Besides the above mentioned next steps and improvements, we can also make some additions to the monitoring system.

Use more sensors The Event Calculus allows us to easily add more information from different sensors to the recognition process. We can use more Kinect sensors to get a more accurate location of the person, but we can also add the data from other sensors. Because of the placement of the Kinect behind the interaction wall, we cannot use the audio data from the microphone. A microphone placed inside the support room can provide additional audio information for example to detect loud noises. Another possibility is to use skin conductance data in the recognition process. Skin conductance can be used as an indication of psychological or physical arousal and can improve the detection of stress.

Other version of Kinect sensor Microsoft is still developing new versions of the Kinect sensor. Besides the XBOX Kinect sensor we used, there is a Kinect for Windows sensor. The newer versions of the Kinect sensor are likely to have different specifications, for example a bigger field of view or a larger range. It is good to keep in mind which version is currently used and whether there is a newer version available with better hardware specifications.

8.3 Conclusion

We showed that we can use a logic-based approach to recognize activities performed by different healthy participants recorded with two Kinect sensors. The long-term activities (walking to location, being at location, walking away from location, reading article, standing still for a long time, walking around for a long time, move from one location to another) are defined in terms of recognized short-term activities (active, inactive, walking). It is easy to recognize a person walking towards a location. The recognition of a person at a location is influenced by the allowed distance to that location, and the error caused by the transformation of the raw sensor readings to room coordinates. It is hard to recognize that a client is walking away from a location, probably because clients are walking too quickly. The recognition of this activity might be improved by increasing the specified close distance. There is not yet a solution for the timepoints on which the Kinect loses track of the client.

We presented a proof of concept for a monitoring system that recognizes a persons behavior in data recorded with two Kinect sensors. We showed that we can recognize activities with an affordable consumer recording device (the Kinect) and that we can recognize more complex activities that are built upon simpler activities. Even though a lot of improvements and additions can be made, the results are promising for further use of this approach.

Appendix A

Nederlandse samenvatting

In oktober 2012 is de nieuwe High Care Unit geopend bij de GGz in Eindhoven, gespecialiseerd in de opvang van psychose patiënten. Naast de bekende separeercel is er een zogenoemde support-room aanwezig; een kamer waar cliënten zich kunnen afzonderen, maar de afzondering is minder strikt dan in de separeercel. De separeercel en support-room zijn uitgerust met moderne technologie, waaronder een interactiescherm en Ambient Experience van Philips. Hiermee is het mogelijk de sfeer in de kamer te veranderen door de verlichting in het plafond van kleur te veranderen. Deze kleur is afhankelijk van een bepaald thema en op het scherm worden beelden getoond die bij dit thema passen, bijvoorbeeld de natuur. Dit alles heeft tot doel de cliënt te kalmeren. De medewerkers van de GGzE kunnen de cliënt niet constant in de gaten houden. Daarom is een monitoringssysteem een goede aanvulling op de zorg. De medewerkers kunnen de informatie gebruiken om de zorg voor een cliënt af te stemmen op diens behoeften. In de toekomst zou een monitoringssysteem uitgebreid kunnen worden, om automatisch het thema in de kamer aan te passen wanneer de cliënt onrustig wordt.

We beschrijven het begin van de ontwikkeling van een dergelijk monitoringssysteem. We gebruiken 2 Kinect sensoren om de locatie en bewegingen van een persoon in een kamer op te nemen. De Kinect sensor is een camera die tevens beschikt over diepte informatie, en software heeft om een persoon te herkennen, te volgen, en een representatie van het 'skelet' te maken. Wij gebruiken de locatie van een persoon om op een hoger niveau te kunnen zeggen wat iemand aan het doen is in de kamer, bijvoorbeeld naar het bed lopen, of uit het raam kijken.

Omdat de medewerkers van de GGzE nog moeten wennen aan de nieuwe technologieën, hebben we het systeem getest met gezonde deelnemers in een kamer bij Philips. De deelnemers zijn gevraagd naar verschillende locaties in de kamer te lopen, een artikel aan de muur te lezen en vervolgens 3 rondjes door de kamer te lopen.

De activiteiten zijn herkend met een formele logica, de Event Calculus. Deze logica laat ons redeneren over gebeurtenissen in de tijd en de gevolgen hiervan. Een activiteit is gedefinieerd als een combinatie van korte-termijn acties, zoals lopen, stilstaan, of bewegen op de plaats, met extra beperkingen aan bijvoorbeeld de locatie of tijdsduur. Wanneer iemand langere tijd stilstaat en zich vlakbij het artikel aan de muur bevindt, zeggen we dat hij het artikel aan het lezen is. Met de Event Calculus kunnen we niet alleen simpele activiteiten herkennen, zoals een persoon op een locatie, maar ook moeilijkere activiteiten die voortbouwen op de herkenning van deze simpele activiteiten, zoals van de ene naar de andere locatie lopen. Het voordeel van deze aanpak is dat we op een begrijpelijke manier kunnen redeneren over de verschillende activiteiten. Daarnaast kunnen we makkelijk nieuwe informatie toevoegen om de herkenning van activiteiten uit te breiden en te verbeteren. Omdat de herkenning nog lang niet altijd perfect is, beschrijven we tevens verbeteringen, uitbreidingen en toevoegingen aan het monitoringssysteem.

Appendix B

Poster presentation opening High Care Unit GGzE - October 3rd 2012

On Wednesday October 3rd 2012 the GGzE (Mental Healthcare services Eindhoven) opend their new High Care Unit. In this High Care Unit, we can find the seclusion room and support room, equiped with Ambient Experience. Because of this opening, an innovation market was organized where companies presented their technological innovations for stakeholders. On behalf of Philips, I was asked to give a poster presentation about my work with the Kinect, together with a short demonstration of the Kinect. We demonstrated the `HandsDemo` program, which displays the speed of each hand.

Activity Observation in Support Room

Maaike Veltmaat, Juergen Vogt⁺, Martijn van Otterlo*
*⁺Philips Research, Eindhoven, The Netherlands, *Radboud University, Nijmegen, The Netherlands*
maaike.veltmaat@philips.com

Current status

Now in support room

- Only information on behavior during contact moments or real-time video data
- No possibilities to look back at video data

Goal:

- Provide staff with more insight in the clients behavior on moments that staff could not monitor
- Look back at recorded data to compare
- Get an objective measure

Set up

- 2 Kinect cameras
- Recording of skeleton data

Looking into

- How to recognize which activities
- How to provide useful feedback to the staff

Kinect camera:

- Released in 2010 for Microsoft Xbox 360
- Control the game console with your body
- Developpers can easily write software for Kinect
  - Low cost and easy development lead to use in research projects

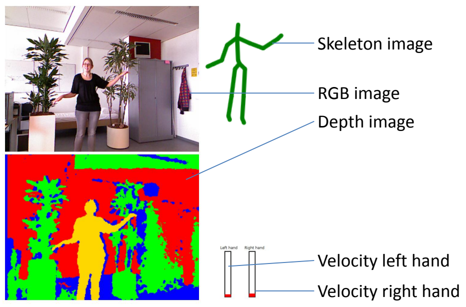
- RGB camera
- Infra red depth camera
- Microphone array

Applications of Kinect

- Games
- Fall detection [1]
- Behavior analysis [2]

[1] *Mastorakis, G., & Makris, D. (2012). Fall detection system using Kinect's infrared sensor. Journal of RealTime Image Processing. Springer.*

[2] *Walczak, N., Fasching, J., Toczyski, W., Sivalingam, R., Bird, N., Cullen, K., Morellas, V., et al. (2012). A nonintrusive system for behavioral analysis of children using multiple RGB+depth sensors. 2012 IEEE Workshop on the Applications of Computer Vision WACV. IEEE.*



Future possibilities

Working towards application to monitor client in the support room

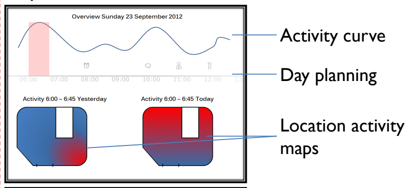
- Activity monitoring
  - How much/which activity
- Monitoring the influence of treatment
- Provide an objective measurement

Usage of system:

- Assist the staff
- Provide staff with information
  - Level of activity
  - Specific activities
  - …

Patients always have to give consent to be monitored.

Options for feedback to staff


Overview Sunday 23 September 2012
Activity 6:00 – 6:45 Yesterday
Activity 6:00 – 6:45 Today
Activity curve
Day planning
Location activity maps

?

Open for suggestions

Radboud University Nijmegen

PHILIPS

A0-format (84,1 x 118,9 cm)

Appendix C

Data Acquisition Manual - C#

When starting up the c# program, you will see a screen as in Figure C.1. To continue from here, you can click `Calibration`, `Record`, or `Open`. The `Open` button was used to convert previously stored serialized `DataClass`es to a `.csv` format. In the current version, the recording will be stored to both a serialized `DataClass` file and a `.csv` format at the same time. The program for calibrating the system can be started

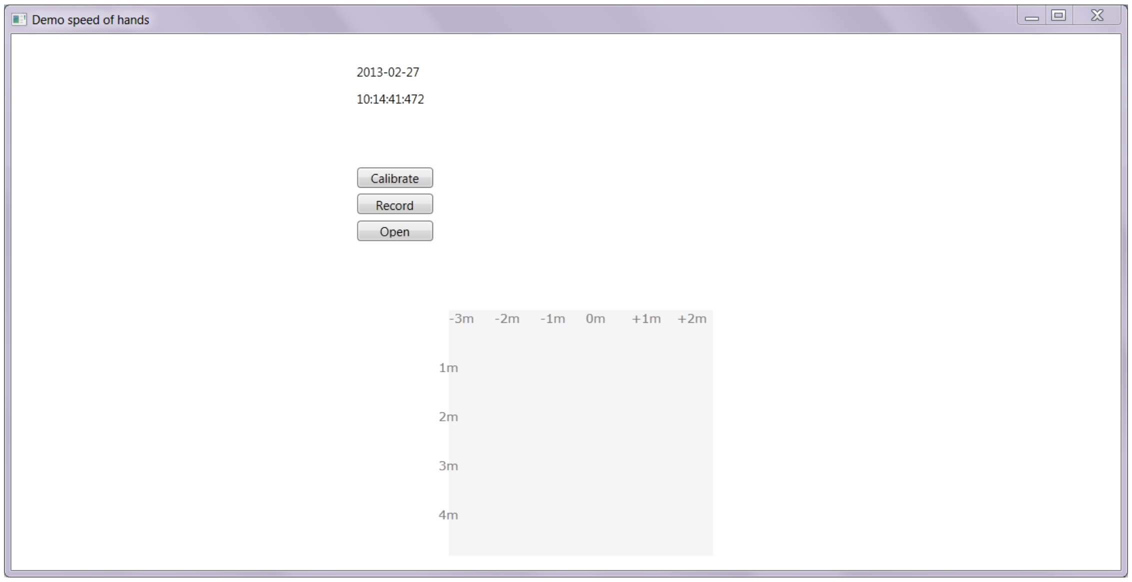


Figure C.1: Start-up screen C# program `SingleKinectRecording - Calibration`.

by clicking `Calibration`. This will be explained in Section C.1. When clicking the `Record` button, you are able to record data. This part of the program will be discussed in Section C.2.

C.1 Calibration

In order to read out the sensor reading for executing the calibration steps, you have to start up the C# program, and click the `Calibration` button. This program shows the basic RGB and Depth pictures, as provided by the Kinect for Windows SDK. Besides that, it also shows the X-, Y-, and Z-coordinates of the Skeleton position. An example is shown in figure C.2.

On the left, you see the basic RGB and depth display. In the depth display, the user is colored yellow, objects that are too far away from the camera are colored red, objects that are too close or not correctly

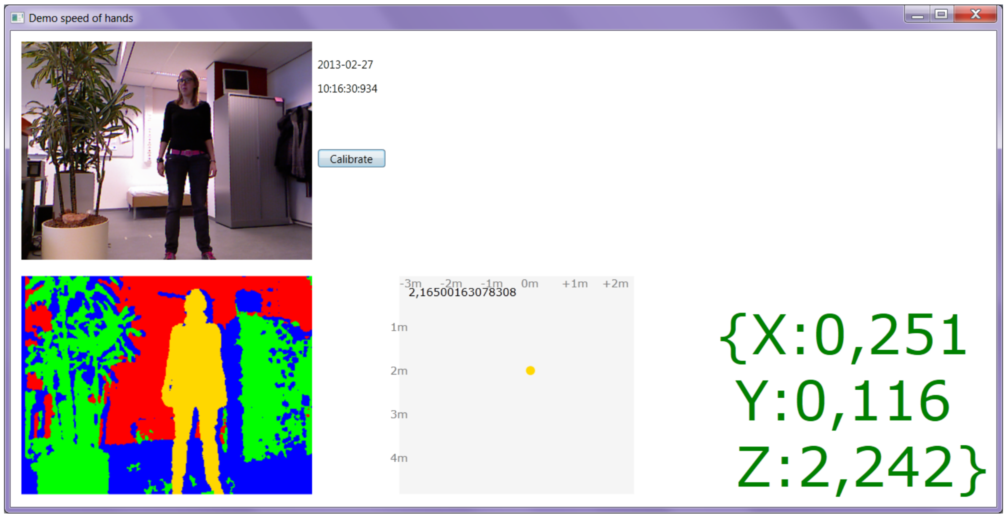


Figure C.2: Screen output for the `Calibration` mode. On the left, you see the basic RGB- and Depth visualizations that are provided in the Kinect for Windows SDK. In the middle, the program shows the position of the user, with respect to the camera. On the right, you see the X-, Y-, and Z-coordinates of the position of the Skeleton, with respect to the camera.

measureable are blue, and objects within range are colored green. In the middle you see the visualization of the position of the tracked Skeleton with respect to the Kinect sensor. On the X-axis, the x-location with respect to the Kinect is shown, on the Y-axis, the depth with respect to the Kinect is shown. In the right part of the screen, you see the X-, Y-, and Z-coordinates of the Skeleton displayed on screen. These are necessary for the Calibration procedure.

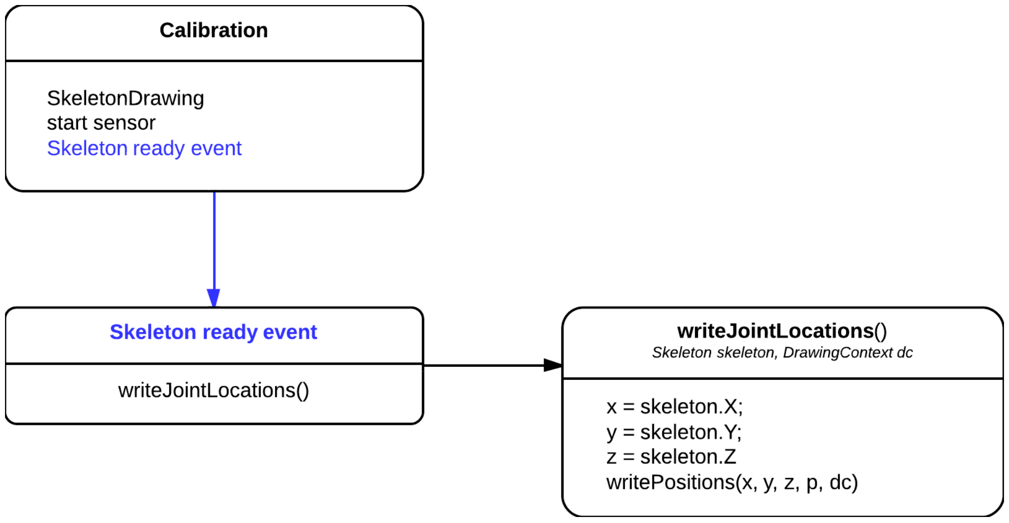


Figure C.3: Code overview for `Calibration`.

The code overview for the calibration is shown in Figure C.3. When the `Calibration` button is clicked, a `SkeletonDrawing` object will be instantiated. After that the Kinect sensor will be started. Everytime a frame is ready, in our implementation 30 times per second, the SkeletonReadyEvent will fire. In the `Calibration` mode, this causes the `SkeletonDrawing` objects to call `writeJointLocations()`, with the current recognized `Skeleton` and the `DrawingContext` as input. This functions calls the function `writePositions(x, y, ...`

`z, p, dc)`, which draws the given `x`, `y`, and `z` on the point `p` of the `DrawingContext dc`. `x`, `y`, and `z` are the corresponding X-, Y-, and Z-locations of the received `Skeleton`.The rest of the Calibration procedure consists of entering the observed readings for known locations in the room into a `.csv` file, and calculate the resulting averaged angle in Matlab.

C.2 Recording

To record data, you have to click the `Record` button in the C# start up screen. This will result in the screen as shown in figure C.4. On the left, the RGB and Depth visualizations are displayed, which are provided by

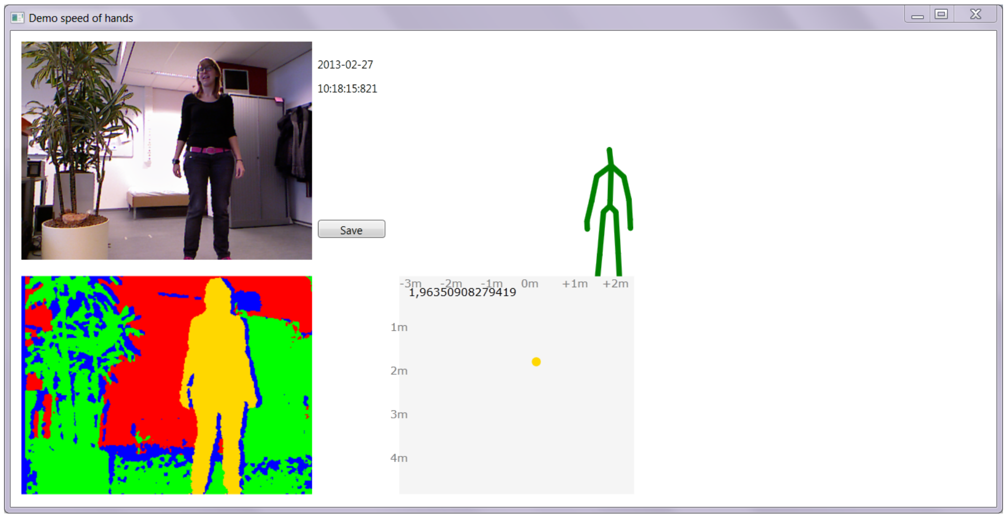


Figure C.4: Screen output for the `Record` mode. On the left, you see the basic RGB- and Depth visualizations that are provided in the Kinect for Windows SDK. In the middle, the program shows the position of the user, with respect to the camera. Above the position of the user in the room, the recognized Skeleton of the user is drawn. When the bone between two joints is `tracked`, the bone is drawn in green. When one of the joints is `inferred`, the bone is drawn in gray.

the Kinect for Windows SDK. In the middle, the location of the tracked person with respect to the Kinect is shown on the bottom of the screen, and the visualization of the tracked `Skeleton` is shown above that. When the two `Joints` that form a bone are both `tracked`, the bone is painted in green; when either of the two `Joints` is `inferred`, the bone will be painted in gray. When a `Joint` is `inferred`, it will be displayed in light green, while it will be displayed in green when it is `tracked`. When a `Joint` is neither `tracked` nor `inferred`, it will not be displayed, and no bone will be drawn to such a joint.

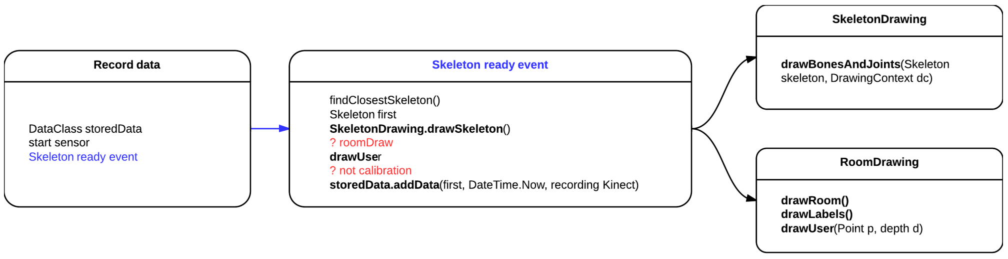
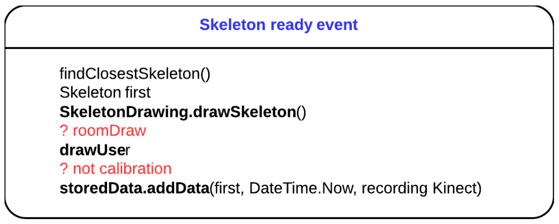


Figure C.5: Code overview for the `Record` option in `SingleKinectRecording`.

The code overview for recording the data is shown in Figure C.5. When the `Record` button is clicked, the program first instantiates a new `DataClass` object. This is initialized with the `KinectInfo` that is stored in the `Settings` of the program. After that, the sensor is started. When a frame is ready the `SkeletonReadyEvent` is fired.



```
findClosestSkeleton()
Skeleton first
SkeletonDrawing.drawSkeleton()
? roomDraw
drawUser
? not calibration
storedData.addData(first, DateTime.Now, recording Kinect)
```

Figure C.6: `SkeletonReadyEvent` for the Record mode.

`SkeletonReadyEvent` When the `SkeletonReadyEvent` is fired, it first tries to find the closest skeleton in the frame. It is also possible to let it find the first tracked skeleton. Assuming that there is only 1 person in the room, it does not matter which of the two options you pick, but when there are more people in the room, it is preferred to let the program select the skeleton that was first tracked.

`SkeletonDrawing` When the `Skeleton` is selected, it is sent to the `SkeletonDrawing` object, to be drawn on the screen.

`RoomDrawing` When the `bool roomDraw` is `true`, the location of the user with respect to the sensor is drawn in the bottom middle of the screen. When the `bool calibration` is `false`, the data is added to the `DataClass` object.

`addData` The `addData` function of a `DataClass` object takes a `Skeleton skel`, a `DateTime time` object set to `DateTime.Now`, and an `int recordingKinect` as it's input. It initializes a `new TimedData(skel, ... time, recordingKinect)` object, with the passed arguments. The number of data points is also increased with 1.

C.2.1 Saving the recorded data

When the `Record` button is clicked, the `Calibration` and `Record` buttons disappear, and the `Save` button is shown on the screen. When this button is clicked, the recorded data will call the `Save` function. An overview of the code is shown in figure C.7. The `Save` function, first creates a file name in the form of `"..\\..\\Data\\DataClass(0:yyyyMMdd_hhmmss).data"` based on the current time. The same filename is generated for storing the `.csv` data, with `"Conv"` instead of `"DataClass"`. After that, a new `DataClassIO` object is instatiated, with the current `DataClass` object and the generated file names. This object handles the storage of the serialized `DataClass` object to a `.data` file, and the conversion of the `DataClass` object to a `.csv` file.

Serialization The objects in a `DataClass` object are serialized and stored to file to enable us to later read in the stored data and access it as if we are recording it. There are serialized versions of the `DataClass`, and it's objects `KinectInfo`, `TimedData`, and `Skeleton`.

`convertToCSV()` The recorded data is stored to `.csv` format as well, to enable portability to other programs, and fast visual inspection of the data. The first row of the resulting `.csv` file contains information on the `KinectInfo` and the `width` and `height` of the room. The information stored in each `TimedData` object in the `DataClass` is printed in the data rows. The first part of this row holds the `DateTime` in milliseconds, the number of the recording Kinect, and the X-, Y-, and Z-coordinates of the location of the `Skeleton`. After that, all the `Joints` in the `Skeleton` are printed as `String JointType`, `X`, `Y`, `Z`, and `TrackingState`.

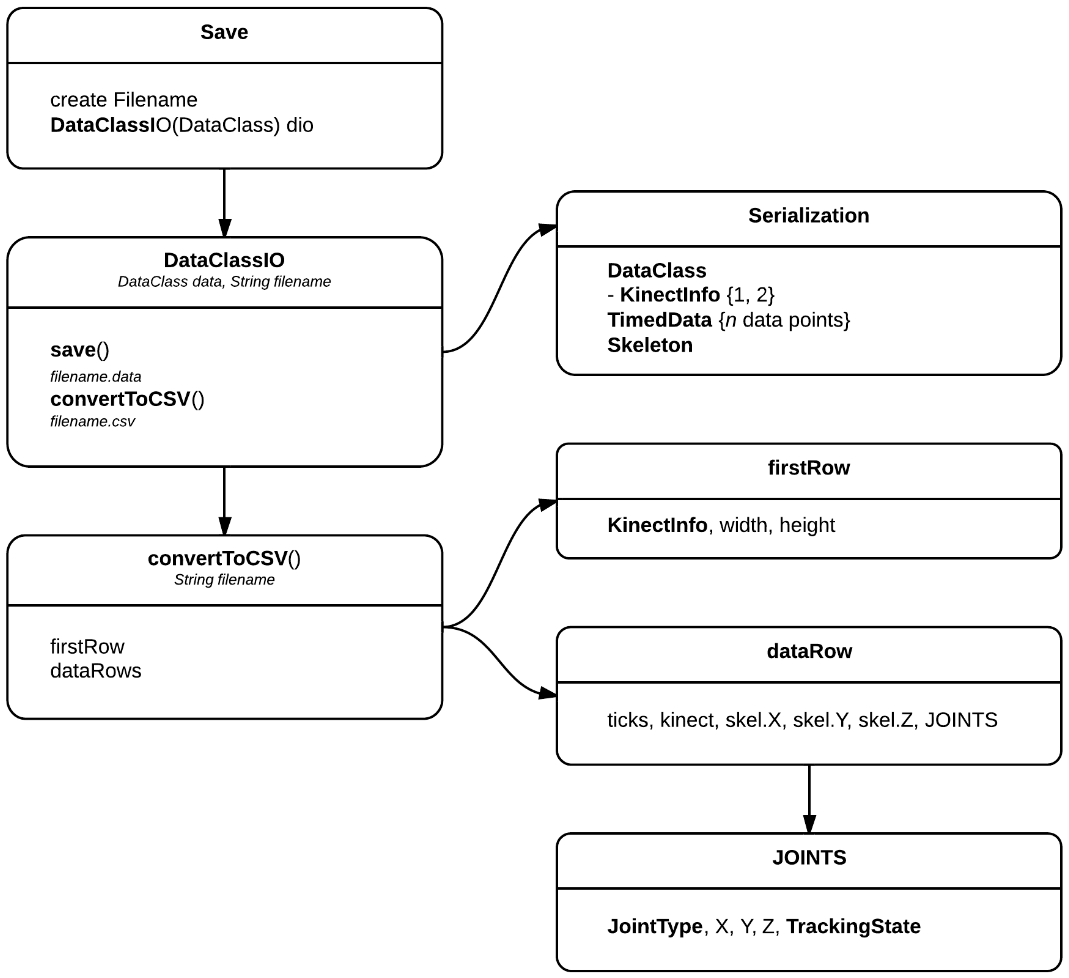
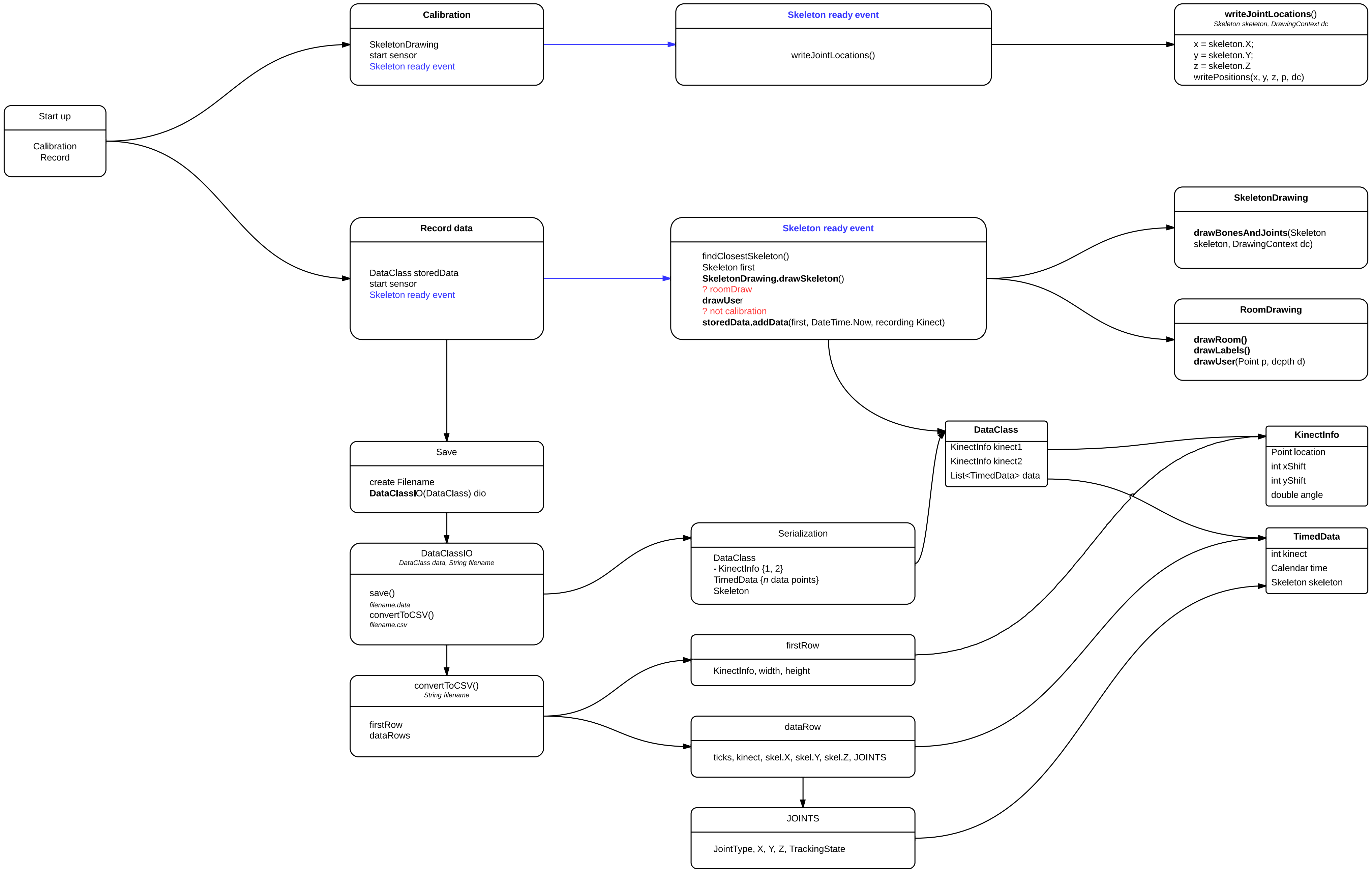


Figure C.7: Overview of executed functions for saving the recorded `DataClass` to file.


Start up
Calibration
Record
Calibration
SkeletonDrawing
start sensor
Skeleton ready event
Skeleton ready event
writeJointLocations()
writeJointLocations()
Skeleton skeleton, DrawingContext dc
x = skeleton.X;
y = skeleton.Y;
z = skeleton.Z
writePositions(x, y, z, p, dc)
Record data
DataClass storedData
start sensor
Skeleton ready event
Skeleton ready event
findClosestSkeleton()
Skeleton first
SkeletonDrawing.drawSkeleton()
? roomDraw
drawUser
? not calibration
storedData.addData(first, DateTime.Now, recording Kinect)
SkeletonDrawing
drawBonesAndJoints(Skeleton skeleton, DrawingContext dc)
RoomDrawing
drawRoom()
drawLabels()
drawUser(Point p, depth d)
Save
create Filename
DataClassIO(DataClass) dio
DataClassIO
DataClass data, String filename
save()
filename.data
convertToCSV()
filename.csv
convertToCSV()
String filename
firstRow
dataRows
Serialization
DataClass
- KinectInfo {1, 2}
TimedData {n data points}
Skeleton
firstRow
KinectInfo, width, height
dataRow
ticks, kinect, skel.X, skel.Y, skel.Z, JOINTS
JOINTS
JointType, X, Y, Z, TrackingState
DataClass
KinectInfo kinect1
KinectInfo kinect2
List<TimedData> data
KinectInfo
Point location
int xShift
int yShift
double angle
TimedData
int kinect
Calendar time
Skeleton skeleton

Appendix D

Processing Manual - Java

The data is processed in Java. A graphical overview of the most important functions is shown at the end of this manual. I will discuss the steps this function takes. First, we need to set the Settings object, and retrieve the filenames for the data

```
Settings set = new Settings(5,15, FRAMEPROC.FPS2, CAMERA.BOTH, noise, convert, ...
    RECLOCATIONS.confRoom, 350,250);

String M1 = "..\\..\\Data\\Recordings 20130130\\Mel2\\Conv20130130_011948_1.csv";
String M2 = "..\\..\\Data\\Recordings 20130130\\Mel2\\Conv20130130_011946_2.csv";
String M = "..\\..\\Data\\Recordings 20130130\\Mel2\\20130130_M_FPS2";
```

With this Settings object, the String filenames, and processing settings (**boolean** prolog, **boolean** visualization, ... **boolean** skeletons) we can call the processRecData() function.

processRecData()

Reading data This function takes String F1, String ... F2, String F3, Settings set, **boolean** prolog, **boolean** visualization, **boolean** ... skeletons as it's input. It calls functions necessary to process the data. The first call is to readDataClassAndLocationsFromCSV(), to read in both recordings in F1 and F2. For each file it is instantiated with, it returns a Pair<DataClass, LocationFile> , which will be used for further processing.

Merging data After extracting the DataClass and LocationFile for both datasets, we create the DataMergeNew dmn to merge the data. From the dmn we can get the merged locations, times, xs, ys, and Skeleton. It must be noted that the merged Skeleton is not yet at a satisfactory level and was not used in the recognition proces.

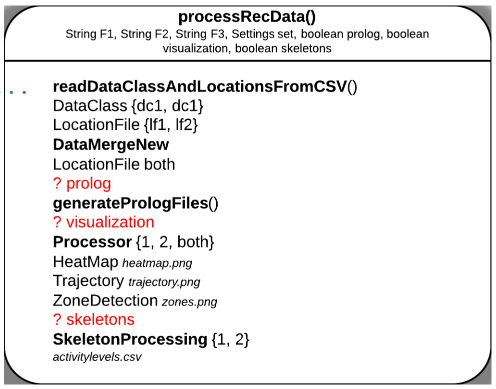
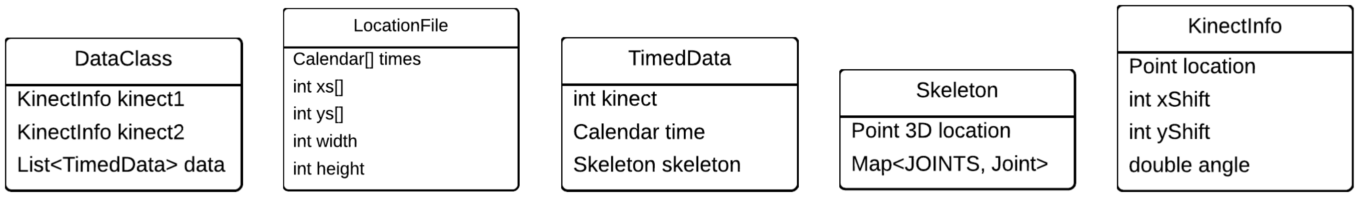


Figure D.1: Overview of the function processRecData.

Processing data Based on the parameters passed to processRecData() (**boolean** prolog, **boolean** visualization, ... **boolean** skeletons), the program calls the functions that handle the generation of prolog files, the visualization of the location data, and the processing of the skeleton data. If **boolean** prolog is **true**, the function generatePrologFiles() is called. If **boolean** visualization is **true**, the program will create three different Processor objects; for the both individual recordings, and for the merged recorder. For each Processor object the function processLocations(**boolean** zones, **boolean** trajectory, **boolean** heatmap) will be called, with all variables set to **true**. If **boolean** skeletons is **true**, the program creates two SkeletonProcessing objects, 1 for each data recording. For each



DataClass LocationFile TimedData Skeleton KinectInfo

Table D.1: Class attributes for DataClass, LocationFile, TimedData, Skeleton, and KinectInfo

SkeletonProcessing object, the activity levels are extracted and written to file. When the skeleton merging is optimized, it is possible to perform additional operations on the merged skeletons as well.

readDataClassAndLocationsFromCSV() The function CSV.readDataClassAndLocationsFromCSV takes the filename of the `.csv` file as it's input, together with the CAMERA cam attribute which holds the camera source of the recording, either `CAMERA.A` or `CAMERA.B`. It returns a Pair holding a DataClass and LocationFile object.

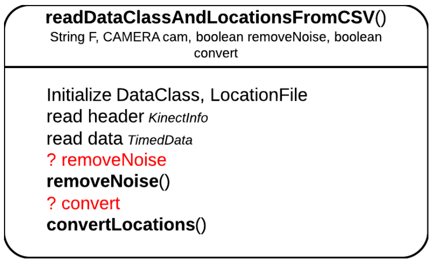

readDataClassAndLocationsFromCSV()
String F, CAMERA cam, boolean removeNoise, boolean convert
Initialize DataClass, LocationFile
read header KinectInfo
read data TimedData
? removeNoise
removeNoise()
? convert
convertLocations()

Figure D.2: Overview of the function readDataClassAndLocationsFromCSV().

DataClass The DataClass object consists of two KinectInfo objects, one for each Kinect sensor, and a List holding TimedData objects. When the recording is made with only one Kinect sensor, the KinectInfo kinect2 object is **null**. The List<TimedData> contains the data points for the recording.

LocationFile The LocationFile object holds information about the location of the person for the given timestamps. These are all stored in arrays, Calendar[] times, **int**[] xs, in[] ys. It also holds information about the width and height of the room in which the data was recorded, **int** width and **int** height. If **boolean** removeNoise is **true**, a function will be called to remove the static noise from the recording. Static noise is noise caused by known artifacts in the environment, such as a plant or door, that are recognized as a person. We know the locations of these artifacts, and have the ability to remove the noise they cause. The location of such noise has to be manually set per camera in the function Main.CSV.removeNoise(). If **boolean** convert is **true**, the function Main.CSV.convertLocations is called, with the List<Double> Xs, List<Double> Ys and **double** angle as it's arguments. It converts the given locations based on a matrix rotation with angle angle.

Reading header As said, the input to the function must be a `.csv` file. The first contains the basic information about the recording and it has either one of the following two forms:

```
int x1, int y1, int xShift1, int yShift1, double angle1, int x2, int y2, int xShift2, int ...
    yShift2, double angle2, int width, int height
```

or

```
int x1, int y1, int xShift1, int yShift1, double angle1, int width, int height
```

The length of the first row must be either 12 or 7. If this is not the case, the program will print an error message. The first header, with length 12, reflects a situation with two Kinect sensors. This is applicable for old data recordings. The new datarecordings have only one Kinect sensor, which corresponds to a header with length 5. width and height are the width and height of the recording room. The **int** x, **int** y, **int** xShift, ... **int** yShift, **double** angle belonging to one Kinect, will all be stored in a KinectInfo object.

value of a	value of b	times tA and tB	Action
$\geq aL$	$\geq bL$		$\rightarrow$ do nothing
$< aL$	¡ bL	within interval	$\rightarrow$ merge
$< aL$		tA before tB	$\rightarrow$ store A
	$< bL$	tB before tA	$\rightarrow$ store B
$< aL$	$\geq bL$		$\rightarrow$ store A
$\geq aL$	$< bL$		$\rightarrow$ store B
else			$\rightarrow$ merge

Table D.2: Logic used for merging the data for recording A and B in the class DataMergeNew. a and b are both indices of the arrays in DataMergeNew, holding object arrays: `Calendar[] times, Skeleton [] skels , int [] xs, int [] ys`. aL is the length of all object arrays for data A, bL the length of all object arrays for data B. tA is the a^{th} element in the Calendar[] of A, the same holds for tB

Reading data The next rows contain the recorded data. The first part of each data row consists of the following items:

```
    long time , int kinect , double posX , double posY , double posZ ,
```

The following entries in the row hold the Skeleton attributes. For each of the 20 Joints we have:

```
    String joint , double x , double y , double z , JOINTSTATUS state
```

Both the location data and the joint data will be added to a TimedData object which is generated for each data row. The class attributes for TimedData are shown in Table D.1. Inside the TimedData object, the joints will be added to a Skeleton object. The attributes for a Skeleton object are shown in Figure Table D.1.

DataMergeNew The merging of data is based on the timing information. The DataMergeNew constructor takes as input two LocationFile objects, lf1 and lf2 one for each data recording which we will call `A` and `B`, and two DataClass objects, dc1 and dc2 for `A` and `B` respectively. The program first checks whether the LocationFile and DataClass objects of 1 recording are of equal length; when this is not the case, it prints an error message.

When the LocationFile and DataClass objects for each recording are of equal length, the program continues with the merging procedure. Consider index `a` for data `A` which has length `aL`, and index `b` for data `B` with length `bL`. We initialize `a,b=0`. Merging is based on the logic in table D.2:

The merging will happen for the Skeleton objects, the Calendar objects, and the Point location.

Calendar For the Calendar object, we will look at the times in milliseconds for both points, and average these. With the resulting value, we will instantiate a new Calendar object.

Location For the location merging we will first check if either of the points is at location $(0,0)$, because this is the default point for which we have no data. If this is the case, we store the other. Otherwise, we will compute the new x-location by averaging both

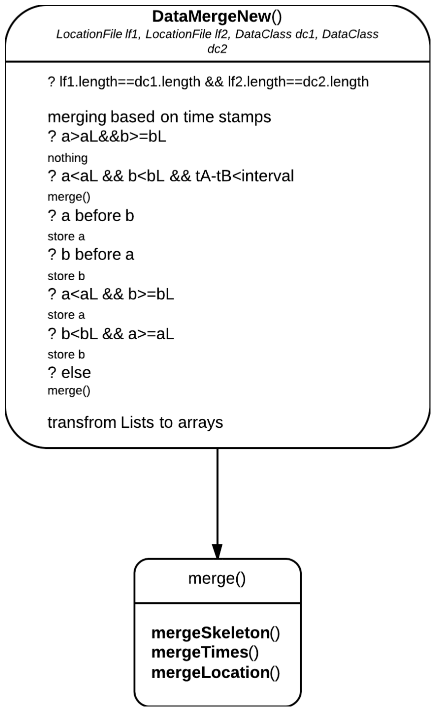

DataMergeNew()
LocationFile lf1, LocationFile lf2, DataClass dc1, DataClass dc2
? lf1.length==dc1.length && lf2.length==dc2.length
merging based on time stamps
? a>aL&&b>=bL
nothing
? a<aL && b<bL && tA-tB<interval
merge()
? a before b
store a
? b before a
store b
? a<aL && b>=bL
store a
? b<bL && a>=aL
store b
? else
merge()
transfrom Lists to arrays
merge()
mergeSkeleton()
mergeTimes()
mergeLocation()

Figure D.3: Overview of the class DataMergeNew.

Joint 1	Joint 2		w1	w2	new TRACKINGSTATE
equal TRACKINGSTATE		→	1	1	trackingstate Joint 1 & 2
NOTTRACKED		→	0	1	trackingstate Joint 2
	NOTTRACKED	→	1	0	trackingstate Joint 1
TRACKED		→	2	1	INFERRED
	TRACKED	→	1	2	INFERRED
else		→	1	1	INFERRED

Table D.3: Weights and new TRACKINGSTATE for merging the Point3D of a Joint of a Skeleton, based on the TRACKINGSTATE of both Joints.

x-values from the data points, and the new y-location by merging the two y-values. When both points are at $(0, 0)$ we store $(0, 0)$ as the location.

Skeleton To merge the Skeleton, we first normalize all the Joint locations with respect to the `HipCenter` joint, making `HipCenter` $(0, 0, 0)$. After that, each Joint will be merged. This is a weighted average of the x-, y-, and z- coordinates of the Joint Point3D locations. The weight and new TRACKINGSTATE is based on the TRACKINGSTATE of the two Skeletons, and they are shown in table D.3. It is possible to adjust the weights for the different TRACKINGSTATEs, or not to weigh them at all.

The merging of two Skeleton objects has only been tested superficial. More work is needed to optimize the merging of the Skeletons, and to handle, for example missing Joints.

generatePrologFiles The function generatePrologFiles() takes as input a LocationFile object, holding the locations, a Settings object, a String filename with the base filename for the result files (String MovFile = filename+"_movement.pl" and String ... AppFile = filename+"_appearance.pl"), and a String origFileName, containing the original filename (or filenames) of the recording that was used to generate the LocationFile object. Based on the LocationFile locs and Settings set we will first create a STARecognizer staRecog, which will be further discussed in section D. From this STARecognition object, we get the arrays holding the recognized short term activities (STA[] stas), appearances (APPEARANCE[] apps), and the corresponding time stamps (Calendar[] times) and locations (**int** [] xs and **int** [] ys). These will be passed to the function convertToECFiles(stas, xs, ys, times, apps, MovFile, AppFile, origFileName), to get all data in the `Prolog` format.

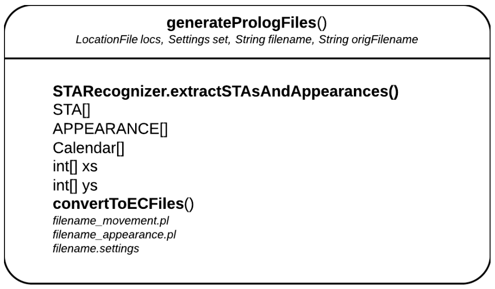


Figure D.4: Overview of the function generatePrologFiles.

STARecognition The recognition of short-term activities is done with the STARecognition class. It processes the data, based on the FRAMEPROC setting. Currently FPS2, FPS3, SECOND and WINDOW3 are implemented. For all settings, we determine the STA with getSTAFromDistance(**double** distance) based on the distance between two points, which is computed with computeDistance(**double** xStart, **double** xEnd, **double** ... dStart, **double** dEnd).

The APPEARANCE is determined based on the previous, current, and next Point. When the current location is $(0, 0)$, the APPEARANCE is set to NONE. When the start location is $(0, 0)$ and the current location is not equal to $(0, 0)$, the APPEARANCE is set to APPEAR. When the current location is not $(0, 0)$ and

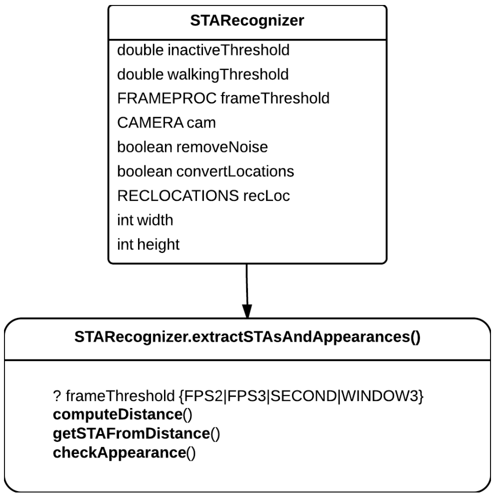


Figure D.5: Overview of the class attributes and class logic of STARecognizer and extractSTAsAndAppearances().

the next location is $(0,0)$, the APPEARANCE is set to DISAPPEAR. When the previous, current, and next location are all not at $(0,0)$, the APPEARANCE is set to VISIBLE.

FPS2 When the FRAMEPROCESSING is set to FPS2, the data is downsampled to 2 frames per second. Since the recording is done with 30 frames per second, this means that we will create new data points for every 15 frames.

FPS3 When working with FPS3 we will look at every 10 frames to create 1 data point, resulting in 3 frames per second.

SECOND A FRAMEPROCESSING of SECOND looks at every 30 frames to create a new data point. This results in a framerate of 1 frame per second.

WINDOW3 WINDOW3 looks at all available data, so with a framerate of 30 frames per second, and determines an STA for all of them.

Processor The Processor class is used to process the location data, and create visualizations for it. It takes as it's input an array with the locations, either as `Point[] locations` or `int[] xs` and `int[] ys`, the `int width` of the room, the `int height` of the room, the RECLOCATIONS `recloc` of the data, the CAMERA `cam` used for the recording, and it has a `String filename` that is used to store the generated visualizations.

The function `processLocations(boolean zones, boolean trajectory, ... boolean heatmap)` is used to create the actual visualizations when the given `boolean`s are `true`.

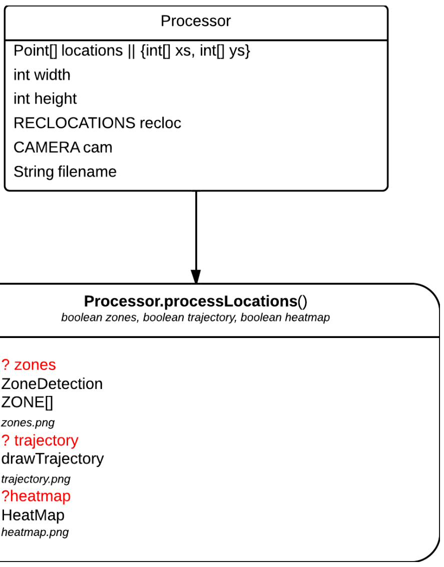


Figure D.6: Overview of the class attributes and class logic of Processor and processLocations.

Zones The room can be divided into two different zones, the `public` and the `private` zone. When the parameter `zones` is `true`, a `ZoneDetection` object is created, which returns an array with the ZONE[] classifications over time, and it draws this classification to a file with name `filename + ... "_zones_"+cam.toString()+ ".png"`.

Trajectories When the parameter `boolean trajectory` is `true`, the function `drawTrajectory(String filename, boolean original)` is called. This function draws the walked trajectory based on the locations. The background of the image is set according to the recording location `recloc`. When the `boolean original` is `true`, it will draw the original testpath into the image. The image will be stored to a file with name `filename + "_trajectoryTime_"+ cam.toString()+ ".png"`.

Heatmap The parameter `boolean heatmap` is `true` when a `HeatMap` object has to be created and drawn to file. We instantiate a `new HeatMap(xs, ys, width, height, recloc)`, and the `HeatMap` object will create the corresponding heat map. A heat map shows for every location in the room how many times the tracked user has been there. It can give a visualization of places that were visited a lot. The created heat map will be stored to a file `filename + "_heatmaptime_"+ cam.toString()+ ".png"`.

SkeletonProcessing The SkeletonProcessing currently consists of computing the activity levels of all Joints and different body parts. Activity is computed as the amount of movement in 3D space. The different body parts that are taken into account are `Head`, verb—Core— (`ShoulderCenter`, `Spine`, `HipCenter`), `Left Arm` (`ShoulderLeft`, `ElbowLeft`, `HandLeft`, `WristLeft`), `Left Right` (`ShoulderRight`, `ElbowRight`, `HandRight`, `WristRight`), `Leg Left` (`HipLeft`, `KneeLeft`, `AnkleLeft`, `FootLeft`), and `Leg Right` (`HipRight`, `KneeRight`, `AnkleRight`, `FootRight`).

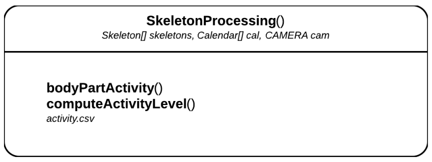
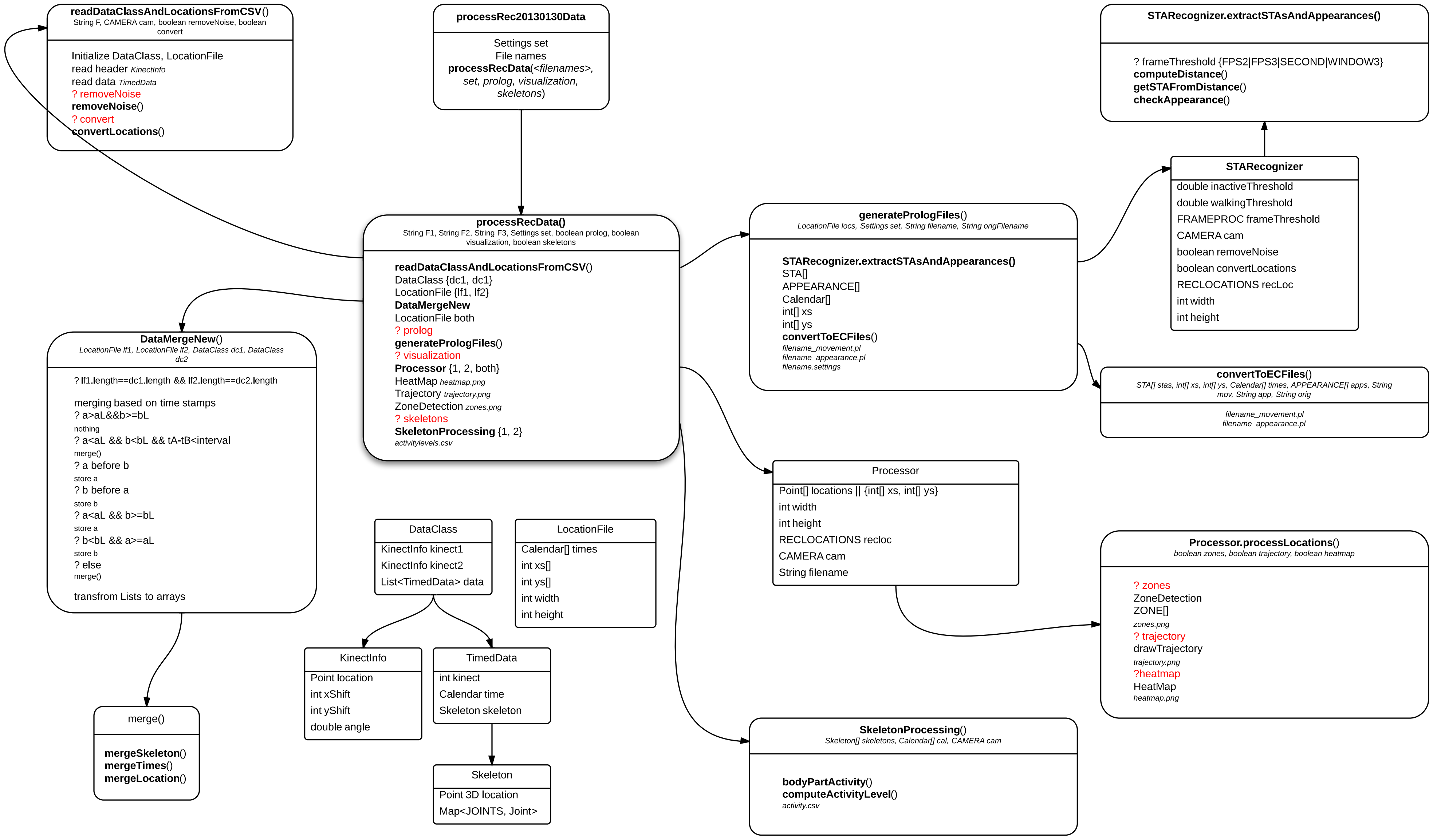


Figure D.7: Overview of the class logic of SkeletonProcessing.

The activity level is computed as the average of movement in 3D space for all TRACKED Joints. This is written to a file with name filename+"_activity1.csv".


readDataClassAndLocationsFromCSV()
String F, CAMERA cam, boolean removeNoise, boolean convert
Initialize DataClass, LocationFile
read header KinectInfo
read data TimedData
? removeNoise
removeNoise()
? convert
convertLocations()
processRec20130130Data
Settings set
File names
processRecData(<filenames>, set, prolog, visualization, skeletons)
STARecognizer.extractSTAsAndAppearances()
? frameThreshold {FPS2|FPS3|SECOND|WINDOW3}
computeDistance()
getSTAFromDistance()
checkAppearance()
STARecognizer
double inactiveThreshold
double walkingThreshold
FRAMEPROC frameThreshold
CAMERA cam
boolean removeNoise
boolean convertLocations
RECLOCATIONS recLoc
int width
int height
processRecData()
String F1, String F2, String F3, Settings set, boolean prolog, boolean visualization, boolean skeletons
readDataClassAndLocationsFromCSV()
DataClass {dc1, dc1}
LocationFile {lf1, lf2}
DataMergeNew
LocationFile both
? prolog
generatePrologFiles()
? visualization
Processor {1, 2, both}
HeatMap heatmap.png
Trajectory trajectory.png
ZoneDetection zones.png
? skeletons
SkeletonProcessing {1, 2}
activitylevels.csv
generatePrologFiles()
LocationFile locs, Settings set, String filename, String origFilename
STARecognizer.extractSTAsAndAppearances()
STA[]
APPEARANCE[]
Calendar[]
int[] xs
int[] ys
convertToECFiles()
filename_movement.pl
filename_appearance.pl
filename.settings
convertToECFiles()
STA[] stas, int[] xs, int[] ys, Calendar[] times, APPEARANCE[] apps, String mov, String app, String orig
filename_movement.pl
filename_appearance.pl
DataMergeNew()
LocationFile lf1, LocationFile lf2, DataClass dc1, DataClass dc2
? lf1.length==dc1.length && lf2.length==dc2.length
merging based on time stamps
? a>aL&&b>=bL
nothing
? a<aL && b<bL && tA-tB<interval
merge()
? a before b
store a
? b before a
store b
? a<aL && b>=bL
store a
? b<bL && a>=aL
store b
? else
merge()
transfrom Lists to arrays
Processor
Point[] locations || {int[] xs, int[] ys}
int width
int height
RECLOCATIONS recloc
CAMERA cam
String filename
DataClass
KinectInfo kinect1
KinectInfo kinect2
List<TimedData> data
LocationFile
Calendar[] times
int xs[]
int ys[]
int width
int height
Processor.processLocations()
boolean zones, boolean trajectory, boolean heatmap
? zones
ZoneDetection
ZONE[]
zones.png
? trajectory
drawTrajectory
trajectory.png
?heatmap
HeatMap
heatmap.png
KinectInfo
Point location
int xShift
int yShift
double angle
TimedData
int kinect
Calendar time
Skeleton skeleton
merge()
mergeSkeleton()
mergeTimes()
mergeLocation()
Skeleton
Point 3D location
Map<JOINTS, Joint>
SkeletonProcessing()
Skeleton[] skeletons, Calendar[] cal, CAMERA cam
bodyPartActivity()
computeActivityLevel()
activity.csv

Appendix E

Activity Recognition Manual - Prolog

The recognition of activities is written in Prolog. We created a testfile that loads all the necessary files, and then calls the performFullER clause, with the name of the result file as argument. This manual discusses the different files and clauses needed for the recognition of activities. An overview is given in Figure E.1.

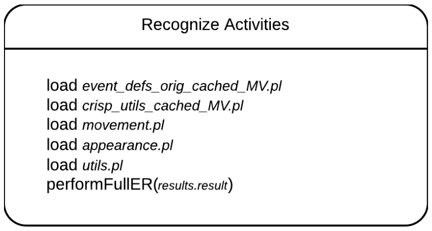

Recognize Activities
load *event_defs_orig_cached_MV.pl*
load *crisp_utils_cached_MV.pl*
load *movement.pl*
load *appearance.pl*
load *utils.pl*
performFullER(*results.result*)

Figure E.1: Overview of calls in a Prolog test file to recognize the activities. It loads the files with the event definitions (event_defs_orig_cached_MV), utilities (crisp_utiles_orig_cached_MV, utils), and the data (movement and appearance). Then it the performFullER(Filename) clause, with Filename the name of the results file.

Event definitions The long term activities that we want to recognize are defined in the file event_defs_orig_cached_MV, Figure E.2. This file first reads in ec_cached_MV.pl, which contains clauses for the Event Calculus, and locationsRec.pl. This last file contains definitions about objects in the environment that are important for the detection of our activities. It also contains the clauses that define how close a person must be to a location, in order to be detected at that location.

Data files The datafiles that are used for the event recognition are movement.pl and appearance.pl. The first file contains information about the movements of a person in the form `happensAt(STA(Id), T).` and `holdsAt(coord(Id)=(X,Y), T).` The first term contains information about the short term activity `STA` for a person `Id` at time `T`. At this same time `T`, person `Id` was at location `(X,Y)`. The appearance.pl file contains information about the visibility of the person in the form `holdsAT(appearance(Id)=Appearance, T)`. The

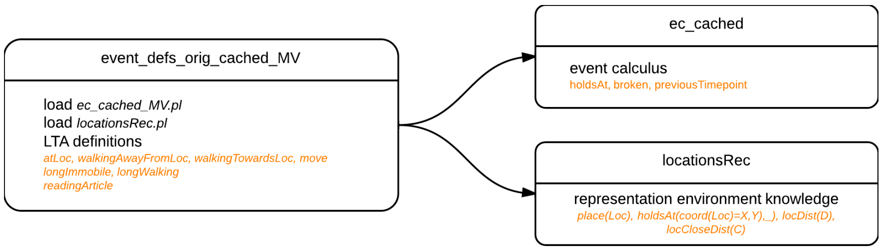

event_defs_orig_cached_MV
load *ec_cached_MV.pl*
load *locationsRec.pl*
LTA definitions
atLoc, walkingAwayFromLoc, walkingTowardsLoc, move
longImmobile, longWalking
readingArticle
ec_cached
event calculus
holdsAt, broken, previousTimepoint
locationsRec
representation environment knowledge
place(Loc), holdsAt(coord(Loc)=X,Y),_), locDist(D),
locCloseDist(C)

Figure E.2: Overview of the event_defs_orig_cached_MV file. It loads two other Prolog files, ec_cached_MV.pl and locationsRec.pl and it contains the definition clauses for the Long Term Activities that we want to recognize.

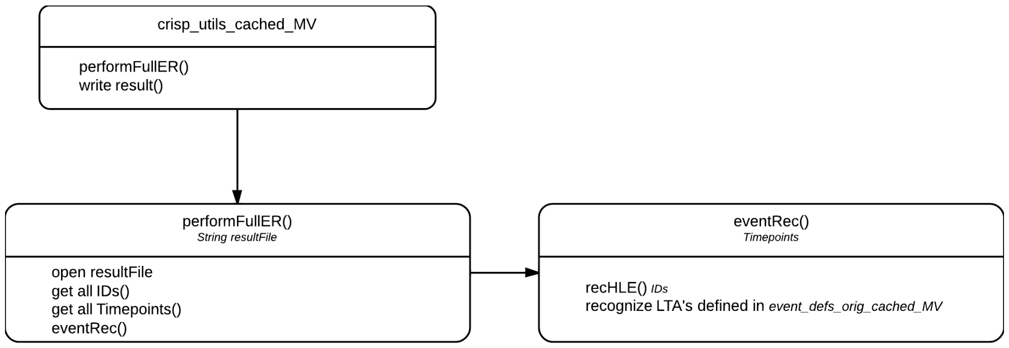


Figure E.3: Overview of the crisp_utils_orig_cached_MV file. This file contains the definition for the performFullER clause, which performs the activity recognition for the activities we specified in event_defs_orig_cached_MV. The actual recognition of these activities is called in the recHLE clause.

Appearance can be visible, appear, disappear or none.

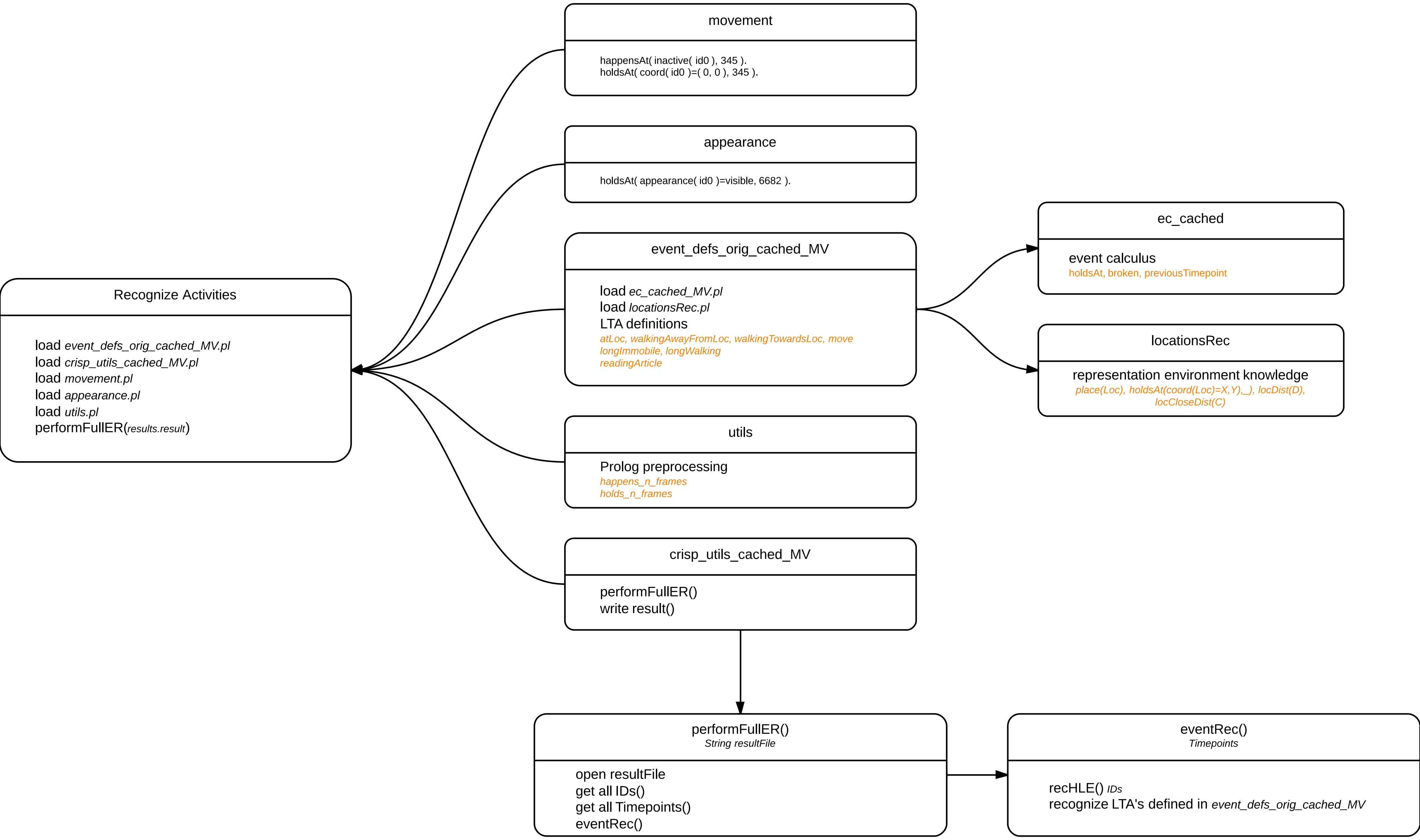
Utilities There are two files that contain additional utility information, crisp_utils_cached_MV.pl and utils.pl. The first file was given by the Event Calculus implementation by `http://users.iit.demokritos.gr/~a.artikis/EC.html`. It contains the clause performFullER, that executes the recognition, and a clause to write the results to the given filename.

The recognition process is initiated by calling the function perforumFUllER('filename') (Listing E and Figure E.3, which is specified in the file crisp_utils_cached_MV.pl. This clause opens a Stream with filename 'filename' and then creates lists containing all IDs, all Timepoints, and all consecutive timepoints. The recognition is started by calling the clause eventRec(Timepoints, IDs, Stream). This function will call the recHLE(Timepoint, [ID | ... RestIDs], Stream) clause for each Timepoint in the Timepoints list, while iterating over all the IDs in the ID list. The recognition of each long term activity is called by recHLE in the form of recognize(readingArticle(ID)=**true**, ... Timepoint, Stream). recognize() checks if the Fluent has the given Value at the passed Timepoint. When this is the case, the result will be written in the file with 'filename'.

```
performFullER(Filename):-
  open(Filename, write, Stream),
  allIDs(IDs),
  allTimePoints(Timepoints),
  createConsecList(Timepoints),
  eventRec(Timepoints,IDs,Stream),
  close(Stream),!.
```

An overview of the different files and functions for the activity recognition in Prolog is shown on the last page of this manual.

Using other locations To use the implemented event calculus with different coordinates for the locations, we need to change the specified coordinates in the locationsRec.pl file. All locations are defined as a place(X), with holdsAt(coord(X)=(Xcoord, Ycoord),_). to indicate the coordinates. When we want to recognize the defined activities for other locations, for example Y, we need to add place(Y). to the locationsRec.pl file. Additionally, we have to add the recognition of these activities in the recHLE clause in the crisp_utils_cached_MV.pl file. Instead of, for example recognize(atLoc(ID, article)=**true**, Timepoint, Stream), we have to fill in Y for article. For the visualizations to work with the new locations, we have to change the names of the places to the names of the new defined places.


movement
happensAt(inactive(id0), 345).
holdsAt(coord(id0)=(0, 0), 345).
appearance
holdsAt(appearance(id0)=visible, 6682).
event_defs_orig_cached_MV
load *ec_cached_MV.pl*
load *locationsRec.pl*
LTA definitions
atLoc, walkingAwayFromLoc, walkingTowardsLoc, move
longImmobile, longWalking
readingArticle
ec_cached
event calculus
holdsAt, broken, previousTimepoint
locationsRec
representation environment knowledge
place(Loc), holdsAt(coord(Loc)=X,Y),_), locDist(D), locCloseDist(C)
Recognize Activities
load *event_defs_orig_cached_MV.pl*
load *crisp_utils_cached_MV.pl*
load *movement.pl*
load *appearance.pl*
load *utils.pl*
performFullER(*results.result*)
utils
Prolog preprocessing
happens_n_frames
holds_n_frames
crisp_utils_cached_MV
performFullER()
write result()
performFullER()
String resultFile
open resultFile
get all IDs()
get all Timepoints()
eventRec()
eventRec()
Timepoints
recHLE() *IDs*
recognize LTA's defined in *event_defs_orig_cached_MV*

Appendix F

Visualization recognized activities

The following script is used to create the visualizations of the results in Matlab. `filenameIn` is the name of the result file from the activity recognition. `act1In` and `act2In` are the files containing the activity levels for Kinect 1 and 2. `filenameOut` is the filename to store the resulting plot.

```
 function [Data] = processData(filenameIn, act1In, act2In, filenameOut)
 fileID = fopen(filenameIn);
D = textscan(fileID, '%s %s %d');
 fclose(fileID);
 activities = D{1};
 times = D{3};
 subject = [];
 for i=1:length(activities)
     subject = [subject str2num(activities{i}(end-1))];
end
%for each activity; create a data vector; 0:absent 1:present
longImm = not(cellfun(@isempty, strfind(activities, 'longImmobile(id')));
longWalk = not(cellfun(@isempty, strfind(activities, 'longWalking(id')));
toNote = not(cellfun(@isempty, regexp(activities, 'walkingTowardsLoc(id[0-9],note')));
 toArticle = not(cellfun(@isempty, regexp(activities, 'walkingTowardsLoc(id[0-9],article')));
 toChair = not(cellfun(@isempty, regexp(activities, 'walkingTowardsLoc(id[0-9],chair')));
 fromNote = not(cellfun(@isempty, regexp(activities, 'walkingAwayFromLoc(id[0-9],note')));
 fromArticle = not(cellfun(@isempty, regexp(activities, ...
     'walkingAwayFromLoc(id[0-9],article')));
 fromChair = not(cellfun(@isempty, regexp(activities, 'walkingAwayFromLoc(id[0-9],chair')));
atNote =not(cellfun(@isempty, regexp(activities, 'atLoc(id[0-9],note')));
 atArticle =not(cellfun(@isempty, regexp(activities, 'atLoc(id[0-9],article')));
 atChair =not(cellfun(@isempty, regexp(activities, 'atLoc(id[0-9],chair')));
moveNA = not(cellfun(@isempty, regexp(activities, 'move(id[0-9],note,article')));
moveNC = not(cellfun(@isempty, regexp(activities, 'move(id[0-9],note,chair')));
moveCA = not(cellfun(@isempty, regexp(activities, 'move(id[0-9],chair,article')));
moveCN = not(cellfun(@isempty, regexp(activities, 'move(id[0-9],chair,note')));
moveAC = not(cellfun(@isempty, regexp(activities, 'move(id[0-9],article,chair')));
moveAN = not(cellfun(@isempty, regexp(activities, 'move(id[0-9],article,note')));
reading = not(cellfun(@isempty, regexp(activities, 'readingArticle(id[0-9]')));

%get activity data
[act1, t1, act2, t2] = processActivityData(act1In, act2In);
%compute max length
m = max(t1(end-1), max(t2(end-1), times(end-1)));

%plot recognized activities
scrsz = get(0,'ScreenSize');
%multiply all featurevectors with subject-value+1?
f = figure('Position', [10 50 3*scrsz(3)/4 4*scrsz(4)/5],'Name', 'Activities over time', ...
    'NumberTitle', 'Off');
longImm = longImm .*2;
longWalk = longWalk.*3;
toNote = toNote.*4;
atNote = atNote.*5;
fromNote = fromNote.*6;
toArticle = toArticle.*7;
atArticle = atArticle.*8;
fromArticle = fromArticle.*9;
toChair = toChair.*10;
atChair = atChair.*11;
fromChair = fromChair.*12;
reading = reading.*13;
%plot all the datavectors to time

%Plot the activity levels;
filt1 = medfilt1(act1, 3); %applying the median filter
filt2 = medfilt1(act2, 3);

%plot activities; either grey or color.
plot(t1, filt1, 'color', [0.8, 0.8, 0.8]); %'r' [0.8, 0.8, 0.8]
hold on
plot(t2, filt2, 'color', [0.9,0.9,0.9]); %'b' [0.9, 0.9, 0.9]
%add activities to legend, in the same order as you plotted them
% hleg = legend('Activity cam 1', 'Activity cam 2');

%plot activities (in color)
plot(times, longImm, 'c.')
hold on
plot(times, longWalk, 'c.')
plot(times, toNote, 'r+')
plot(times, atNote, 'ro')
plot(times, fromNote, 'r.')
plot(times, toArticle, 'b+')
plot(times, atArticle, 'bo')
plot(times, fromArticle, 'b.')
plot(times, toChair, 'm+')
plot(times, atChair, 'mo')
plot(times, fromChair, 'm.')
plot(times, reading, 'b.')

%uncomment to plot activities in gray
% plot(times, longImm, 'Color', [0.8, 0.8, 0.8], 'LineStyle', '.');
% hold on
% plot(times, longWalk, 'Color', [0.8, 0.8, 0.8], 'LineStyle','.')
% plot(times, toNote,'Color', [0.8, 0.8, 0.8], 'LineStyle','+')
% plot(times, atNote, 'Color', [0.8, 0.8, 0.8], 'LineStyle','o')
% plot(times, fromNote, 'Color', [0.8, 0.8, 0.8], 'LineStyle', '.')
% plot(times, toArticle, 'Color', [0.8, 0.8, 0.8], 'LineStyle','+')
% plot(times, atArticle, 'Color', [0.8, 0.8, 0.8], 'LineStyle','o')
% plot(times, fromArticle, 'Color', [0.8, 0.8, 0.8], 'LineStyle', '.')
% plot(times, toChair, 'Color', [0.8, 0.8, 0.8], 'LineStyle','+')
% plot(times, atChair, 'Color', [0.8, 0.8, 0.8], 'LineStyle','o')
% plot(times, fromChair, 'Color', [0.8, 0.8, 0.8], 'LineStyle', '.')
% plot(times, reading, 'Color', [0.8, 0.8, 0.8], 'LineStyle', '.')
%end uncomment activities in grey

axis([0 m 1 13.5])
xlabel('Time (s)')
set(gca, 'XTick', [0:30000:m])
set(gca, 'XTickLabel', [0:30:m])

%add activities to legend, in the same order as you plotted them
set(gca, 'YTick', 0:13)
set(gca, 'YTickLabel', {'', ' ', 'Long immobile', 'Long walking', 'Walking towards note', ...
    'At note', 'Walking away from note', 'Walking towards article', 'At article', 'Walking ...
    away from article', 'Walking towards chair', 'At chair', 'Walking away from chair', ...
    'Reading'});%'Note to article', 'Note to chair', 'Chair to article', 'Chair to note', ...
    'Article to chair', 'Article to note', 'Reading article'})
title('Recognized activities over time')

print(f, '-dpng', filenameOut)

Data = horzcat(times, subject', longImm, longWalk, towards, atbed, away, toNote, atNote, ...
    fromNote, toArticle, atArticle, fromArticle, toChair, atChair, fromChair, moveNA, ...
    moveNC, moveCA, moveCN, moveAC, moveAN, reading);
```

Appendix G

Overview Results

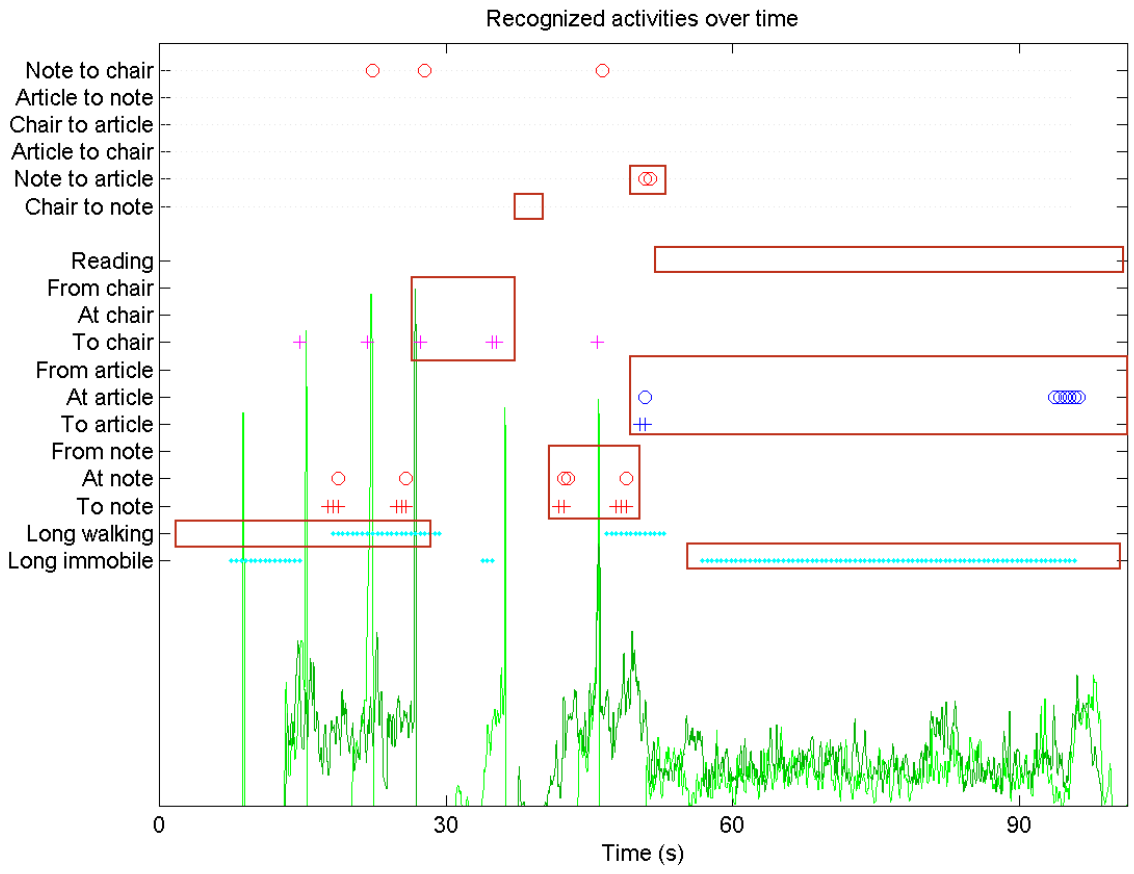


Figure G.1: Results for participant W2, the blocks indicate which cluster of activities has to be recognized at each time. This participant first walked around before he walked to the chair, the note and the article. We recognized 7 of the 13 activities.

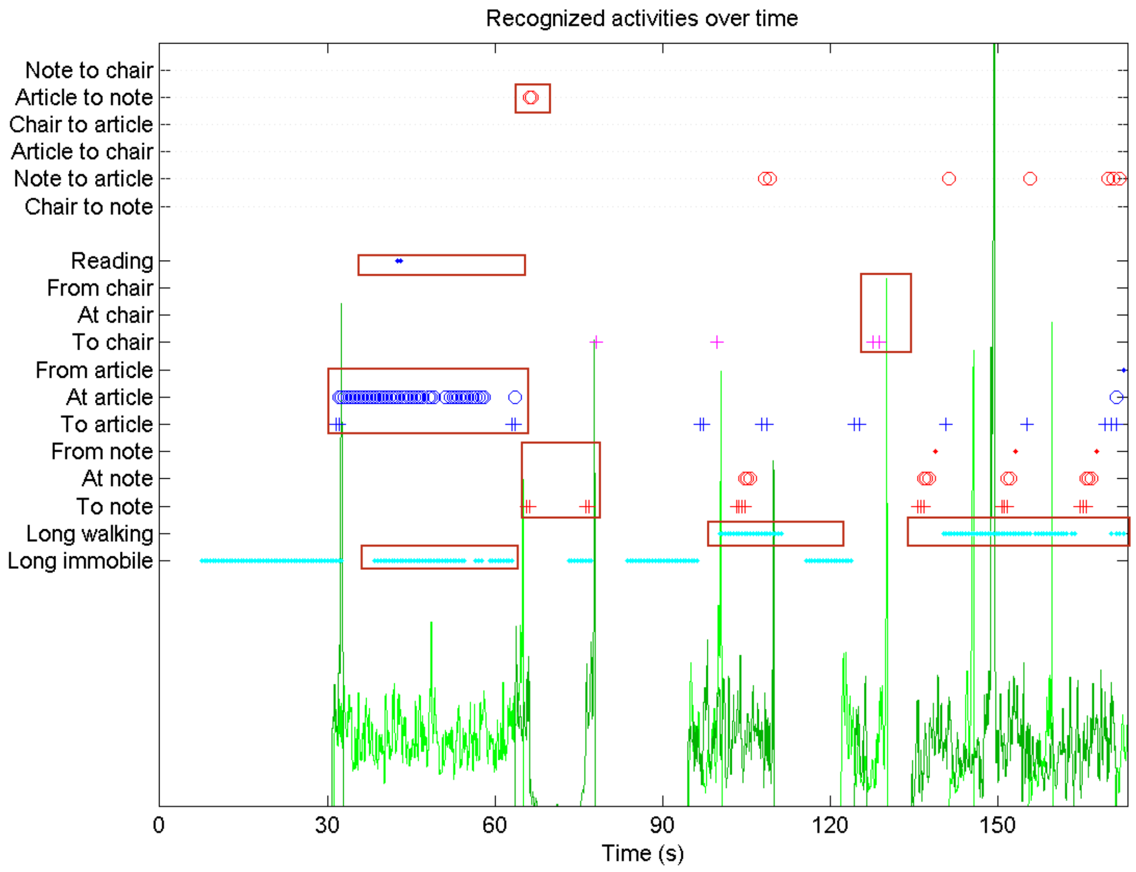


Figure G.2: Results for participant J, the blocks indicate which activity to expect. This participant first read the article, than walked to the note, walked around, moved to the chair, and walked around again. We recognized 6 of the 12 activities.

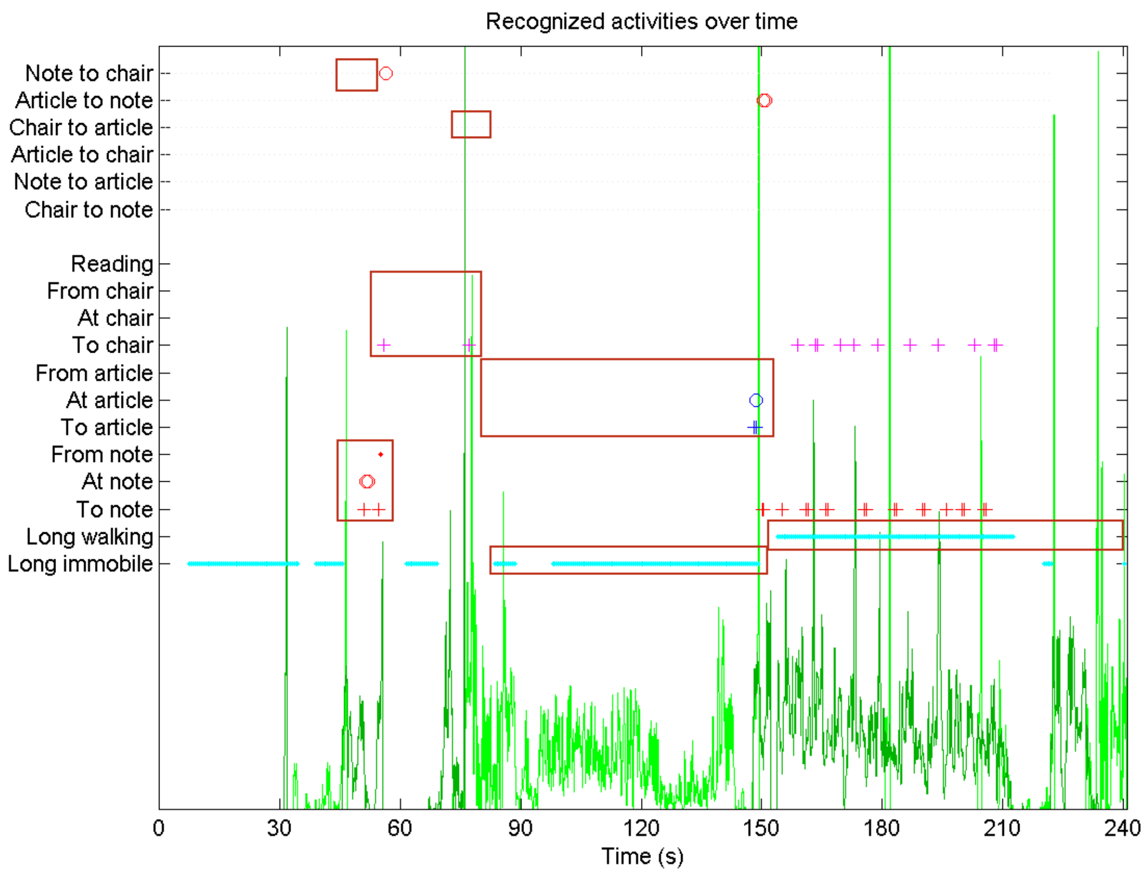


Figure G.3: Results for participant G2, the blocks indicate which activity to expect. This participant walked to the note and to the chair before going to the article. After he read the article he walked around the room. We recognized 6 of the 13 activities.

	G			M			R			J			W1			W2			M2			G2		
	30	40	50	30	40	50	30	40	50	30	40	50	30	40	50	30	40	50	30	40	50	30	40	50
To chair				x	x	x	x	x	x	x	x	x	x			x	x	x	x	x	x	x	x	x
At chair						x	x	x	x			x		x	x			x	x	x	x			
From chair							x								x									
To note	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x		x	x	x	x	x	x
At note		x	x	x	x	x	x	x	x		x	x	x	x	x	x	x	x	x	x	x	x	x	x
From note		x	x				x															x		
To article	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x	x	x	x	x			
At article	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x	x	x	x	x		x	x
From article																								
Reading article		x	x		x	x	x	x	x	(x)	x	x						x						x
Walking around	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x				x	x	x
Chair to note							x	x	x										x	x	x			
Note to article		x	x				x	x	x							x	x	x	x	x	x			
	4	8	8	6	7	8	12	10	10	5	7	8	6	3	4	7	7	8	8	8	8	5	5	6

Table G.1: Recognized activities for all participants.

Bibliography

Abbasi, A., Dailey, M., Afzulpurkar, N., and Uno, T. (2010). Student mental state inference from unintentional body gestures using dynamic bayesian networks. *Journal on Multimodal User Interfaces*, 3(1):21–31.

Aggarwal, J. and Ryoo, M. (2011). Human activity analysis: A review. *ACM Computing Surveys*, 43(3):16:1–16:43.

Artikis, A. (2012). Event calculus. `http://users.iit.demokritos.gr/~a.artikis/EC.html`. Date visited: December 2012.

Artikis, A., Sergot, M., and Paliouras, G. (2010). A logic programming approach to activity recognition. In *Proceedings of the 2nd ACM international workshop on Events in multimedia*, EiMM '10, pages 3–8, New York, NY, USA. ACM.

Burba, N., Bolas, M., Krum, D. M., and Suma, E. (2012). Unobtrusive measurement of subtle nonverbal behaviors with the Microsoft Kinect. In *IEEE VR Workshop on Ambient Information Technologies*, pages 10–13, Orange County, CA.

Chaaraoui, A. A., Climent-Pérez, P., and Flórez-Revuelta, F. (2012). A review on vision techniques applied to human behaviour analysis for ambient-assisted living. *Expert Systems with Applications*, 39(12):10873 – 10888.

Chan, E., Wang, D., and Pasquier, M. (2008). Towards intelligent self-care: Multi-sensor monitoring and neuro-fuzzy behavior modelling. In *Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on*, pages 3083 –3088.

Clientenbelang Amsterdam (2012). Eigen regie in crisis? `http://www.psy.nl//fileadmin/files/Psyarchief/Files_2012/Eigen_regie_in_crisis_definitief__2_.pdf` via `http://www.dwangindezorg.nl/nieuws/zelf-de-regie-houden-tijdens-een-crisis`. Date visited: 2013-02-18.

EC Funded CAVIAR project/IST 2001 3750 (2001). CAVIAR dataset. `http://homepages.inf.ed.ac.uk/rbf/CAVIAR/`.

Galatas, G., Ferdous, S., and Makedon, F. (2013). Multi-modal person localization and emergency detection using the Kinect. *International Journal of Advanced Research in Artifical Intelligence (IJARAI)*, 2:41–46.

GGZ Oost Brabant (2012). Niet meer in de separeer. `http://www.ggzoostbrabant.nl/over-ons/kwaliteit-van-zorg/drang-naar-minder-dwang/`. Date visited: 2013-02-18.

Isoda, Y., Kurakake, S., and Nakano, H. (2004). Ubiquitous sensors based human behavior modeling and recognition using a spatio-temporal representation of user states. *Advanced Information Networking and Applications, International Conference on*, 1:512.

Kaliouby, R. E. and Robinson, P. (2004). Real-time inference of complex mental states from facial expressions and head gestures. In *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 10 - Volume 10*, CVPRW '04, pages 154–, Washington, DC, USA. IEEE Computer Society.

Khoshelham, K. and Elberink, S. O. (2012). Accuracy and resolution of Kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454.

Koninklijke Philips Electronics N.V. (2004-2013). Ambient Experience. `http://www8.healthcare.philips.com/ae/`. Date visited: 2013-03-12.

Kosmopoulos, D. I., Antonakaki, P., Valasoulis, K., Kesidis, A., and Perantonis, S. (2008). Human behavior classification using multiple views. In *Proceedings of the 5th Hellenic conference on Artificial Intelligence: Theories, Models and Applications*, SETN '08, pages 123–134, Berlin, Heidelberg. Springer-Verlag.

Kowalski, R. and Sergot, M. (1986). A logic-based calculus of events. *New Generation Computing*, 4:67–95.

Kuijpers, E. (2012). Het roer om (GGz Eindhoven). `http://www.dwangindezorg.nl/inspiratie/de-praktijk/best-practices/projecten-in-zuid-nederland/het-roer-om`. Date visited: 2013-02-18.

Mastorakis, G. and Makris, D. (2012). Fall detection system using Kinects infrared sensor. *Journal of Real-Time Image Processing*, pages 1–12.

Mediant GGZ (2010). Jaardocument 2010. `http://www.mediant.nl/ufc/file2/mediant_sites/unknown/ecbcf6354124805ce0f621a8a542b8a8/pu/Jaardocument_2010.pdf` on `http://www.mediant.nl/`. Date visited: 2013-02-18.

Microsoft (2012). Kinect for Windows. `http://www.microsoft.com/en-us/kinectforwindows/`. Date visited: 2013-04-11.

OpenKinect (2012). OpenKinect project. `http://openkinect.org/wiki/Main_Page`. Date visited: 2013-04-11.

Park, S. and Kautz, H. (2008). Hierarchical recognition of activities of daily living using multi-scale, multi-perspective vision and RFID. In *Intelligent Environments, 2008 IET 4th International Conference on*, pages 1–4.

Parnassia (2012). Minder dwang en drang bij Gastvrij Parnassia (Parnassia). `http://www.dwangindezorg.nl/inspiratie/de-praktijk/best-practices/projecten-in-west-nederland/minder-dwang-en-drang-bij-gastvrij-parnassia-2009-2010`. Date visited: 2013-02-18.

PrimeSense (2013). NiTE Middleware. `ww.primesense.com/solutions/nite-middleware/`. Date visited: 2013-04-11.

Psy (2011). Mediazuil en telefoon in de separeer. `http://www.psy.nl/meer-nieuws/dossier/Artikel/mediazuil-en-telefoon-in-de-separeer/?L=&cHash=518e2955e2c12ec96610ff174e5545f6`. Date visited: 2013-02-19.

Raedt, L. D., Kimmig, A., and Toivonen, H. (2007). Problog: a probabilistic prolog and its application in link discovery. In *In Proceedings of 20th International Joint Conference on Artificial Intelligence*, pages 2468–2473. AAAI Press.

Rijksoverheid - Ministerie van VWS, Ministerie van Justitie (2012). Welke gevolgen heeft dwang? `http://www.dwangindezorg.nl/vraag-en-antwoord/8-1-8-dwang-gevolgen/pp352-welke-gevolgen-heeft-dwang-voor`. Date visited: 2013-02-18.

Rijksoverheid - Ministerie van VWS, Ministerie van Justitie (2012-2013). Dwang in de zorg bij psychische problemen. `http://www.dwangindezorg.nl/psychiatrische-problemen`. Date visited: 2013-02-18.

Saguna, S., Zaslavsky, A., and Chakraborty, D. (2011). Complex activity recognition using context driven activity theory in home environments. In *Proceedings of the 11th international conference and 4th international conference on Smart spaces and next generation wired/wireless networking*, NEW2AN'11/ruSMART'11, pages 38–50, Berlin, Heidelberg. Springer-Verlag.

Sakr, G. E., Elhajj, I. H., and Huijer, H. A.-S. (2010). Support vector machines to define and detect agitation transition. *IEEE Transactions on Affective Computing*, 1:98–108.

Sivalingam, R., Cherian, A., Fasching, J., Walczak, N., Bird, N., Morellas, V., Murphy, B., Cullen, K., Lim, K., Sapiro, G., and Papanikolopoulos, N. (2012). A multi-sensor visual tracking system for behavior monitoring of at-risk children. In *International Conference on Robotics and Automation (ICRA) 2012*. IEEE.

Skarlatidis, A., Artikis, A., Filippou, J., and Paliouras, G. (2014). A probabilistic logic programming event calculus. *Theory and Practice of Logic Programming: Special Issue on Probability, Logic and Learning*. Forthcoming.

Sung, J., Ponce, C., Selman, B., and Saxena, A. (2011). Human activity detection from rgbd images. In *Plan, Activity, and Intent Recognition*, volume WS-11-16 of *AAAI Workshops*. AAAI.

The Mathworks, I. (2013). medfilt1. `http://www.mathworks.nl/help/signal/ref/medfilt1.html`. Date visited: 2013-05-18.

Turaga, P., Chellappa, R., Subrahmanian, V. S., and Udrea, O. (2008). Machine Recognition of Human Activities: A Survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(11):1473–1488.

Van der Werf, L. (2009). Dwangtoepassing in Europa; is Nederland voortrekker of achterloper". *Signal Symposia*.

Veilige zorg, ieders zorg (2013). Patiëntveiligheid in de GGZ - Dwang en Drang. `http://www.veiligezorgiederszorg.nl/speerpunten/dwang-en-drang.html`. Date visited: 2013-02-18.

Yu, X., Wu, L., Liu, Q., and Zhou, H. (2011). Children tantrum behaviour analysis based on Kinect sensor. In *Intelligent Visual Surveillance (IVS), 2011 Third Chinese Conference on*, pages 49–52.

Zhu, C. and Sheng, W. (2012). Realtime recognition of complex human daily activities using human motion and location data. *IEEE Transactions on Biomedical Engineering*, 59(9):2422–2430.