Universität Rostock

Traditio et Innovatio

Efficient human situation recognition using Sequential Monte Carlo in discrete state spaces

Dissertation

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

der Fakultät für Informatik und Elektrotechnik

der Universität Rostock

vorgelegt von

Martin Nyolt, geb. am 08.10.1987 in Rostock

https://doi.org/10.18453/rosdok_id00002533

Datum der Einreichung: 23. April 2019
Datum der Verteidigung: 13. September 2019

Gutacher:
Prof. Thomas Kirste, Institut für Informatik, Universität Rostock
Prof. Adelinde Uhrmacher, Institut für Informatik, Universität Rostock
Prof. Jesse Hoey, David R. Cheriton School of Computer Science, University of Waterloo

Abstract

Sequential Monte Carlo (SMC) methods, also known as particle filters, are a tool for sequential state estimation based on the framework of Bayesian filtering. The motivation of this dissertation is to infer human behaviour and situations of daily activities based on noisy sensor data. Using SMC methods allows for models with arbitrary large state spaces, transition models, duration models and observation models. This flexibility is needed for building *causal* models allowing accurate recognition of every-day situations.

The feasibility of this modelling and inference approach has previously been shown. However, applying these methods to real-world human behaviour models brings new algorithmic challenges. The categorical state spaces require an efficient method, as standard particle filters are only efficient for continuous state spaces. Additionally, the large degree of freedom of human behaviour results in very large state spaces, requiring many particles for an accurate approximation of the posterior distribution.

This dissertation analyses these challenges and provides solutions for SMC methods. The large, categorical and causal state-space is the largest factor for the inefficiency of current SMC methods. The marginal filter is analysed in detail for its advantages in categorical states over the particle filter. An optimal pruning strategy for the marginal filter is derived that limits the number of samples.

Zusammenfassung

Sequential Monte Carlo (SMC) Methoden, auch bekannt als Partikel-Filter, sind ein Verfahren zum rekursiven Schätzen des aktuellen Zustands und basieren auf dem Framework des Bayesschen Filterns. Die Motivation dieser Dissertation ist das Inferieren von menschlichen Verhaltens und von alltäglichen Situationen auf Basis von verrauschten Sensordaten. Das Benutzen von SMC-Methoden erlaubt Modelle mit beliebig großen Zustandsräumen, Transitions-, Zeit- und Beobachtungsmodellen.

Die Machbarkeit dieses Ansatzes zur Modellierung und Inferenz wurde schon in früheren Arbeiten gezeigt. Allerdings bringt das Anwenden dieser Methoden auf reale menschliche Verhaltensmodelle neue algorithmische Herausforderungen. Die kategorischen Zustandsräume verlangen effiziente Methoden, da der standard Partikelfilter nur effizient bei kontinuuierlichen Zustandsräumen ist. Zusätzlich resultieren die vielen Freiheitsgrade von menschlichem Verhalten in sehr großen Zustandsräumen, welche viele Partikel für eine genaue Approximierung der A-posteriori-Verteilung benötigen.

Diese Dissertation analysiert diese Herausforderungen und entwickelt Lösungen für SMC-Methoden. Der große, kategorische und kausale Zustandsraum ist der größte Faktor für die Ineffizienz von aktuellen SMC-Methoden. Die Vorteile des Marginalen Filters in kategorischen Zustandsräumen gegenüber dem Partikelfilter werden detailliert analysiert. Eine optimale Pruning-Strategie wird für den Marginal Filter entwickelt.

Acknowledgements

I would like to thank my supervisor Prof. Thomas Kirste who gave me the opportunity to work on my dissertation. He often contributed new ideas and was willing to give valuable feedback anytime. I would also like to thank my co-supervisor Prof. Adelinde Uhrmacher for her feedback on my draft, and I would like to thank her and Prof. Jesse Hoey for being willing to review this thesis.

I am very grateful to the German Research Foundation (DFG) and the Research Training Group GRK 1424 MuSAMA for funding large parts of my research. I am also thankful to my colleagues at the chair of MMIS, in particular to Sebastian Bader, Frank Krüger, Kristina Yordanova and Albert Hein. They all have been open to various discussions, supported my work and gave valuable feedback as well. Special thanks goes to Peter Eschholz for providing all the technical infrastructure, he was always willing to consider my requirements and provided continuous support.

Last but not least, I am very grateful to my family and friends for never stopping to support and motivate me.

Notational conventions

Throughout this work, we will use some notational abbreviations or use terms in a specific sense. These conventions are collected here.

Terminology We will refer to this work itself as the *dissertation*, in the sense of "A formal exposition of a subject, especially a research paper that students write in order to complete the requirements for a doctoral degree". In this dissertation, a few theses are formulated, with a *thesis* used in the sense of "A statement supported by arguments".

Usage of first person First person singular ('I') is used when the statement refers to a deliberate decision or choice made by the author of this dissertation (e. g. 'I argue that') as well as to indicate contributions of this thesis (e. g. 'I have developed a novel ...'). First person plural ('we') is used in the sense of 'the author and the reader' in contexts such as explanations (e. g. 'we see that ...') and proofs (e. g. 'we now proof that ...').

Mathematical notations Throughout this dissertation, the following conventions for typesetting mathematical texts are used. Names of sets are single, bold-faced upper-case letters $\mathbf{X}$. Names of variables are printed in italic lower-case letters i. Sequences of values $(x_1, x_2, \ldots, x_n)$ are denoted by $x_{1:n}$. Pairs and tuples of values $x_1, x_2, \ldots, x_n$ are enclosed in angle brackets $\langle x_1, x_2, \ldots, x_n \rangle$.

Random variables are printed in italic upper-case letters X. The set of possible values X can take (its domain) is denoted by $\mathbf{X}$. Instantiations of random variables are printed in italic lower-case letters x, using the same letter as the random variable. The distribution of a random variable X is denoted by $P(X)$, with $x \sim P(X)$ being a sample from $P(X)$. The probability density function (probability mass function) of a continuous (discrete) random variable X is denoted by $p(x)$. If x is an instantiation of X, the notion $p(x)$ is short-hand for $p(X = x)$, i. e. the density of X at the value x (the probability of X taking the value x).

For the most part of this work, we will prefer to characterise distributions $P(X)$ and conditional distributions $P(X \mid Y = y)$ by their densities $p(X)$ and $p(X \mid Y = y)$. When there is no ambiguity, we abbreviate $p(X \mid Y = y)$ as $p(X \mid y)$.

The Dirac delta function, written formally as $\mathit{Dirac}(x - c)$, will be written more conveniently as $\mathit{Dirac}(x = c)$, signalling that the random variable X will only take the value $x = c$.

Integrals of densities of probability distributions are written as $\int p(x)\ dx$ and are implicitly quantified over the domain X of x, i. e. are short-hand for $\int_{x \in \mathbf{X}} p(x)\ dx$.

Contents

1 Introduction

Summary: A long-standing research goal is the accurate recognition and thus automatic understanding of situations in everyday life of human users. In this dissertation, we follow a model-based approach of Bayesian sequential state estimation. The human behaviour models are characterised by large state spaces, complex and large transition models, arbitrary probabilistic duration models and noisy observation models.

Sequential Monte Carlo (SMC) methods, also known as particle filters, are a tool for approximate sequential state estimation based on the framework of Bayesian filtering. Although SMC methods have been successfully applied for a variety of use cases, it was unknown if they perform equally well for human behaviour models due to their complexity. The feasibility of this modelling and inference approach has been shown by Yordanova [217] and Krüger [111]. This work deals with the algorithmic challenges and the efficiency of the inference. I analyse the drawbacks of state-of-the-art inference methods, identify bottlenecks and develop improvements for SMC methods applied to human situation recognition.

This introductory chapter consists of three parts:

- The first two sections provide background knowledge on *human situation recognition* (Section 1.1) and *Bayesian filtering* (Section 1.2). The section on situation recognition introduces the application domain and motivates the need for situation recognition. Bayesian filtering is the foundation for the formal treatment of the models and algorithms.

  Understanding both subjects is an important precursor for reading this dissertation. Readers who are familiar with these subjects can skip the corresponding sections.

- Sections 1.3 and 1.4 introduce the goal and problem statement of this dissertation.

  The *requirements* (Section 1.3) describe necessary properties of the models and algorithms for inferring the current situation. These requirements are motivated by the application domain in Section 1.1. Additionally, this section also describes several premises and thus restrictions which are necessary for a distinct and succinct treatment of the subject. Together, these define the constraints on the subject and thus the focus of this work.

  Section 1.4 introduces the *challenges* for which this dissertation is providing novel solutions. These challenges are the basis for the contributions.

- Finally, Section 1.5 summarises the contributions of this work.

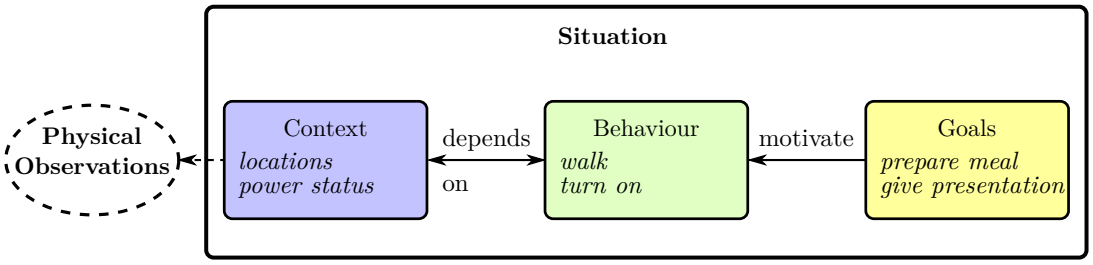


Figure 1.1: Overview of the different components of a *situation*, including typical examples of instances.

1.1 Human situation recognition

Recognising the situation of human users is the motivation of my work. Human situation recognition refers to estimating real world situations based on real sensor data. Although my contributions are applicable to all applications domains, they are motivated by and geared towards situation recognition of human behaviour.

Often, the rationale behind improvements and reasoning in this dissertation is based on the requirements and premises of human situation recognition. This section gives an overview of human situation recognition on an informal level. Its purpose is to give an understanding and intuition of the topic.

Defining *situation* A *situation*, as understood in this work, describes and characterises the environment, the users and the interactions between them. In the context of situation-aware computing, the term has been used at least since 1995 by Rekimoto and Nagao [168] and first definitions are available at least since 1999 by Chávez et al. [38]. In this dissertation, I divide a situation into three components *context*, *behaviour* and *goals*. An overview of the different components of a situation is shown in Fig. 1.1.

The *context* describes the current state of the environment, including users. For example, the locations of objects (e. g. kitchen utensils or users) and power status of devices (e. g. if a projector or oven is turned on). The *behaviour*, at different levels of abstractions, describes changes to the context. These might be from human users (e. g. walking to a different place or dimming a lamp) but also from autonomous devices (e. g. an alarm clock ringing). The *goals* of the human users represent the desires and needs that lead to the behaviour (i. e. we assume that human behaviour is goal-directed [9, p. 352]). A goal might be being not hungry (thus eating a meal) or finishing a presentation.

Applications Recognising the situation of users is not an application on its own, but is required for automatic *assistance* and support.

Automatically assisting humans in their tasks has become a primary goal in artificial intelligence, and much research has been performed in achieving this goal. For instance, context-aware computing aims at automatically adapting to the environment in which a user performs a task [13, 181]. Ubiquitous computing [209] and pervasive computing [14] envision computing devices that seamlessly integrate into everyday life and provide

information and assistance to users [78, 132]. Similarly, the vision of ambient intelligence [63, 165] emphasises on user-friendliness, services and supporting human interactions by intelligent devices that unobtrusively recognise and respond to the users' presence [63].

Assisting humans is motivated, for instance, by ageing population and healthcare [86, 166, 176, 198] (subsumed by the term ambient assisted living). One important aspect is the support of activities of daily living (ADLs) [97], which allow people maintaining an independent life and thus reducing caregiver costs. Other work is motivated by comfort and increased productivity [43, 122, 209] through smart environments or intelligent environments.

Recognising the situation is a requirement for assistance and can be found in many architectures of assistive systems [43, 50, 84, 206]. For example, the useful hint "you have to take the knife out of the drawer" can only be provided if the system is aware of the goal (cooking a meal), next action (cutting an ingredient) and context (location of the knife). As another example, understanding why a user deviates from a plan is important to estimate if interventions are plausible or detrimental, as argued by Hiatt et al. [85].

This leads to a central thesis that motivates situation recognition, and therefore this work:

Thesis 1. *There is increasing demand for assisting people in their everyday activities. Correctly recognising the situation is a precursor to useful assistance.*

Types of situation recognition A number of different technologies have been developed to recognise the user's situation. For instance, context-aware systems infer the *context* (such as locations of objects or users) to which the system adapts itself [13, 181]. Activity recognition classifies the behaviour and motions of users [35]. The problem of plan recognition, related to recognising the users' goals, has been described (at latest) in the 1970s [183, 184], with the seminal formal work in plan recognition by Kautz [98] in 1987. Research has mostly focussed on recognising parts (e. g. only behaviour or only plans) of the situation.

Sensor technology Situation recognition is always based on some kind of sensors from which the current situation is inferred. The sensor technology varies in aspects such as usability, data amount and data quality. Different sensor characteristics require different methods for recognition and can enable or disable certain applications.

Usually, the sensors can be categorised into *wearable sensors*, *dense sensors* and *vision-based sensors* [35]. *Wearable sensors* are worn by the human user and can thus directly record the user's activities and the environment surrounding the user [120]. Different types of wearable sensors can record the user's position, acceleration of the body or parts thereof (e. g. head, arms and legs), muscular activity and others. *Dense sensors* are usually installed at fixed locations within the environment (e. g. room or office). These sensors can either monitor environmental conditions, for example using temperature sensors or light sensors. Other types of dense sensors can indirectly observe activities of the users, for instance using RFID readers or passive infrared-based motion detectors.

Vision-based sensors, i. e. video cameras, capture images of the activities. Methods from computer vision research analyse, for instance, objects and body poses within the image [3]. One can also consider software as a fourth class of *logical sensor* [182], for instance which records key strokes or clicks within a graphical user interface.

1.2 Bayesian filtering

We now want to introduce a formal framework for recognising everyday situations. For this purpose, let the (random) variable X denote the current situation, and let y be the current observation, i. e. the sensor data collected from the environment. In a probabilistic framework, situation recognition can be achieved by probabilistic inference. Inference is the task of estimating a (random) variable X based on observations y, i. e. computing the posterior distribution $P(X \mid y)$. Bayesian inference is based on the Bayesian formula

$$P(X \mid y) \propto P(y \mid X)P(X),$$

which states that one can estimate X based on prior knowledge about X and the likelihood of the observations y depending on X.

Bayesian filtering However, we do not want to estimate a single situation X at a specific time based on a single set of observations y, but a *sequence* of situations $X_{1:i}$ based on a sequence of sensor data $y_{1:i}$. For this purpose, we model the situation as a dynamical system with a state space $\mathbf{X}$. Every state $x_i \in \mathbf{X}$ represents the situation at time i, the dynamical system models how the situations change over time.

To estimate the sequence of situations we use Bayesian filtering. Bayesian filtering is the extension of Bayesian inference to dynamical systems; in this case, inference is also called sequential state estimation [47].

This approach has been applied to a wide variety of applications where state sequences need to be inferred. Example applications include inertial vehicle localisation [47] (e. g. using the Global Positioning System (GPS) and inertial measurement units (IMUs)), spacecraft altitude and orbit estimation [123], weather forecasting [17], control theory [8], motion capturing [107], object tracking in computer vision [91], speech recognition [161] and activity recognition [214]. In this dissertation, we are only concerned with the latter, i. e. inferring the activities – or more generally, the situation – of human users.

Probabilistic models / state space models Bayesian filtering requires a *model*, i. e. a description, of the dynamical system. As the model describes uncertainties using probability distributions, it is a *probabilistic model*. This model provides the prior knowledge used by Bayesian inference. It describes how the system evolves and what observations it produces. To explicitly distinguish from the *model*, we occasionally use the terms *real world* and *real-life* to denote the actual physical environment in which the human user is situated and operates.

State space models are a probabilistic description of a dynamical system by a set of internal states $\mathbf{X}$. The dynamics are described using a transition distribution $P(X_i \mid x_{i-1})$

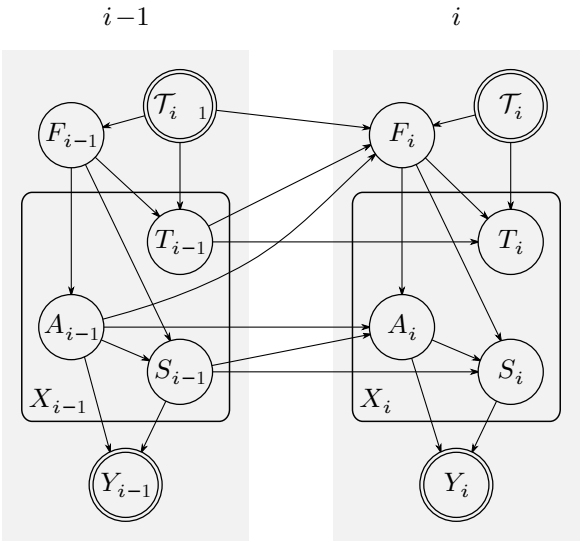


Figure 1.2: Slightly simplified DBN structure of the models used within this dissertation. The model consist, for every time $\mathcal{T}$, of a context state S and an executed action A, both generating possible observations Y. The variable T tracks the starting time of the current action, F is an auxiliary variable. Observable random variables are represented by double-circled nodes.

and a prior distribution of the initial state $P(X_1)$. An observation distribution $P(Y_i \mid x_i)$ models which observations $y_i \in \mathbf{Y}$ are generated by the system.

State space types Depending on the types of states, the state space can be classified into different types:

Continuous The states are continuously valued, e. g. in the domain of real numbers. States can be ordered in some sense, and it is always possible to define a state between to different states. The number of states is infinite. This implies that continuous states allow to converge to a value arbitrarily close. Example state spaces are locations on a continuous grid (e. g. geo-locations).

Discrete The states are *not* continuously valued, but are discrete, e. g. in the domain of integers. Thus states are still ordered, but it is not possible to converge to states arbitrarily close. Discrete states still allow to measure distance between two states. Example state spaces are locations on a discrete grid or counting populations.

Categorical (nominal) Categorical states neither allow converging arbitrarily close nor do they allow measuring distance. The values of categorical states are from finite, non-number valued sets. Example state spaces are locations in different named places (without modelling distance between these places) or places of household objects (knife, fork, ...).

Depending on the type of state space, different operations can either be allowed or forbidden. For this dissertation, categorical states spaces are of main interest.

Dynamic Bayesian Networks State space models of any type can be specified by Dynamic Bayesian Networks (DBNs). DBNs are a graphical representation of the model. They allow to represent independence between random variables and are thus a useful tool for efficient inference. For an introduction to state space models and DBNs, I refer to the dissertation of Murphy [138]; for an in-depth treatise of DBNs and graphical models in general, including common algorithms for inference and learning, I recommend the book by Koller and Friedman [108].

DBNs model a dynamical system by a set of random variables and their dependencies. This dissertation focuses on state space models that can be described by DBNs of the structure in Fig. 1.2 (the full DBN that is used in the implementation and a detailed description of the model structure is presented in Section 3.1). The model is characterised by a set of states $\mathbf{X}_i$ (for every timestep i), which is factored into a set of *context states* $\mathbf{S}_i$, a set of *actions* $\mathbf{A}_i$ representing transitions between context states and a set of action *starting times* $\mathbf{T}_i$ tracking the time when the action started.

Additionally, to correctly model the execution of actions over time, each action has an associated duration distribution. To keep track of the duration, T_i represents the current action's starting time ($\mathcal{T}$ is the current global "real" time). F_i is an auxiliary boolean random variable that indicates if the last action finished and a new action shall start its execution.

How the situations evolve over time (for instance how the next action is selected) is modelled by the dependencies between timesteps $i-1$ and i (the arrows from left to right in Fig. 1.2). The random variable Y_i models the observations generated by the system's execution.

Application to human situation recognition We use *Computational Causal Behaviour Models* (CCBMs) [111] and this DBN structure for our application domain of human situation recognition (see Section 3.1). Context states $s \in \mathbf{S}$ represent environment states like places of kitchen utensils, objects carried by the user and power states of devices. Actions $a \in \mathbf{A}$ are mostly (but not necessarily) the actions executed by the user, like moving between places and grabbing objects. In contrast to most other application domains of sequential state estimation, such as object tracking in computer vision, motion capturing or localisation, the state space is *categorical* and not continuously valued. In our domain models of daily activities, there are thousands to billions of context states $\mathbf{S}$ (cf. Section 3.4 and Table 3.1 for a overview of the models we use in this work). The general model structure also allows infinite state spaces. Because we want to recognise fine-grained actions as well as long-lasting actions, the set $\mathbf{T}$ of starting times can be potentially large; in our datasets, there are a few thousands timesteps. Therefore, the complete state space $\mathbf{X} \subseteq \mathbf{S} \times \mathbf{A} \times \mathbf{T}$ is very large.

Approximate inference Exact inference in the DBN of Fig. 1.2 has polynomial complexity of up to $\mathcal{O}(|\mathbf{X}|^2)$ *per timestep* ([138, p. 73], the tree-width is 2 for this DBN). Thus, for applications with large state spaces exact inference is often too time-consuming or even intractable, especially when inference must be done in real time. Approximate

solutions have therefore been developed. Our models also require approximate inference, of which Sequential Monte Carlo (SMC) methods seem to be the most viable approach. They are a sampled-based approach, i. e. they sample (or simulate) state sequences from the transition model, and weight samples according to the filtering equation. SMC methods are very flexible as they make no assumptions about the DBN structure and their inference complexity is independent of $|\mathbf{S}|$ and $|\mathbf{T}|$ (the two largest sets in human behaviour models).

To compare the quality of the approximations, we measure the *approximation error*:

Definition 1 (Approximation error). *Let $P(X_{1:i} \mid y_{1:i})$ be the true filtering distribution for all timesteps 1 to i, and let $\tilde{P}(X_{1:i} \mid y_{1:i})$ be an approximation of the true filtering distribution. The* approximation error *of the approximation $\tilde{P}$ is a distance measure from P to $\tilde{P}$.*

Suitable distance measures will be discussed in Chapter 5, where the first empirical evaluations are presented.

Limitations Although SMC methods can be applied to human behaviour models, state-of-the-art algorithms tend to perform worse for the type of human behaviour models we use in this dissertation than they perform for applications like object recognition, for which they have been initially developed. The traditional particle filter does not perform well due to the properties of human behaviour models (discussed later and outlined in Section 1.4). Given the increasing interest in and importance of recognising the situation of human users, tailoring inference algorithms towards these models seems desirable.

1.3 Requirements and premises

Krüger [111] has derived a list of five requirements for situation recognition. Four of these requirements are also the basis for the inference within this work. The last requirement, re-usability of the modelling approach, is also important but not related to the inference.

Situation recognition Krüger's first requirement is that the approach is able to recognise the users' *plans*, in particular the action sequence and the final goal. I extend and generalise this requirement to include the whole situation. To provide useful assistance on different levels of abstraction, the inference mechanism must be able to *infer any required aspect of the situation* (i. e. context, actions, tasks, goals).

Online inference The inference must also provide inference results *online*, given a potentially endless stream of observations. However, no hard real-time constraints are made.

Uncertainty To cope with the uncertainty in sensor data, the inference must *provide estimates* over the probability distributions of the current situation given the sensor data.

Latent infinity Finally, the inference must be able to handle potentially *infinite state spaces*, as the diversity of human behaviour and the number of different states if the environment is virtually unbounded (e. g. collecting any number of items at a place). Complementarily, the modelling approach must be able to construct infinite state spaces to correctly model the real world.

Besides the requirements, a few premises and assumptions are made for this dissertation.

Causal model We use *causal models*, this addresses the challenges described by Yordanova [217, Sec. 1.7] and also fulfils the modelling requirements of Krüger [111]. A causal model describes the state of a system (i. e. the user's environment, the context of the situation) in terms of actions (i. e. the user's behaviour) and describes actions in terms of the state [111]. Yordanova [217] has a good explanation and motivation for using causal models:

> '[...] Causal models do not specify a set of actions with which a goal can be achieved, but rather define the preconditions for reaching it, and the effects after the goal has been reached, thus creating a structure of causally related states that lead from the initial to the goal state. In difference with process-based models which answer the question *what*, causal models deal with the problem of *why a user is doing something*, thus investigating the cause and effects of a given action sequence.'
>
> – Yordanova [217], page 13

Categorical state space As a consequence of the causal model, the state space of the models used here is mostly *categorical* (i. e. discrete and nominal). Example categories are objects (e. g. kitchen utensils) and locations (e. g. rooms). We do not consider requirements or extensions to also support continuous states (e. g. three-dimensional location coordinates), as these do not model the question *why a user is doing something*.

Observation models We also assume that sensor models or *observation models* are available; this dissertation does not deal with low-level inference based on sensor data. How good observation models can be obtained, although an important question, is also not in the focus of this dissertation.

Sequential Monte Carlo inference This dissertation uses only the framework of *Bayesian filtering* for the probabilistic inference. In particular, we use *Sequential Monte Carlo* (SMC) methods, which can cope with infinite state spaces and are the focus of this work.

1.4 Challenges of human situation recognition

In principle, situation recognition is estimating $P(X_i \mid y_{1:i})$. However, I argue that recognising the situation in its entirety using models of human behaviour based on realistic sensor data is challenging due to several reasons. Here, the challenges will be

briefly summarised. A detailed analysis of the literature leading to these challenges is presented in Chapter 2.

Sensor data One challenge is the lack of high-quality observation data. Currently, sensor data is too noisy, too ambiguous or the sensors are very specific and cannot be used to monitor a variety of situations. For instance, dense sensors do not scale well to many objects, wearable sensors produce noisy and ambiguous data and video-based tracking has constraints such as required line of sight and privacy.

Complex human behaviour Human behaviour in every-day life is complex. Humans often behave non-deterministically and do different tasks in parallel. The environment they interact with is large and allows for a great variety of actions. Without discriminative sensor data, predicting human behaviour accurately is often infeasible.

Sensor data and the human behaviour complexity do not influence the *inference* per se. However, these are two underlying causes of challenges that do directly complicate probabilistic inference. The following challenges cause complex (or erroneous) models and thus increase the computational complexity of the inference task.

Modelling Due to the complex behaviour patterns, similar to a formalisation of common sense knowledge, modelling human behaviour is also a challenge [145]. This includes a collection of all possible activities (e. g. walking, taking objects, opening objects, mixing objects) with possible objects. But this process is also subject to include personal preferences and *causal* relations. From a practical perspective, behaviour models also need to be kept computationally tractable. Keeping a small tractable model which does explain all *required* behaviour (for specific applications) may also lead to bugs and errors in models.

Categorical states Modelling causal behaviour introduces important structured prior knowledge that helps to recognise human behaviour. The causal modelling approach used in this dissertation uses a symbolic description and leads to a categorical state space (e. g. discrete locations of objects instead of continuous three-dimensional coordinates). However, probabilistic inference (including approximations) works best with continuous state spaces, as these usually allow analytic solutions with fewer parameters. The states of human behaviour models are mostly categorical and thus discrete (without any inherent ordering or even distance measure) and thus methods developed for continuous states are less efficient for approximating the filtering distribution $P(X_i \mid y_{1:i})$.

Large state space Due to the complex human behaviour, and a huge variety of real-world situations that need to be distinguished by the models, the models generate a large state space $\mathbf{X}$. This state space can have billions (or even an infinite number) of distinct environmental states. When exactly computing the filtering distribution, the models used in this dissertation can only be filtered *at most* to timestep 35 (of more

than thousands) before running out of 512 GB of memory (more details are presented in Chapter 5 and Table 5.1). In conjunction with categorical states, a good sampling of states requires many computing resources.

Durative actions Human behaviour also does not only operate in space (on the environmental context), but also in time. Thus, the filtering distribution not only represents *what* action was executed, but also *when*. The duration of human behaviour can vary widely, even for specific actions. This leads to an even larger domain with larger support (of several orders of magnitude) for the filtering distribution.

Inference methods The state of the art inference methods are not suited for these characteristics of human situation recognition. Due to the large, categorical states and complex human behaviour, even approximate methods such as SMC struggle with keeping a good representation of the estimated posterior distribution.

These observations lead to the following thesis.

Thesis 2. *Inference of the situation of human behaviour in real-life requires many computing resources. Correctly estimating the true situation is very challenging.*

Given the vast amount of research in this area, this observation is not surprising. Later in Chapter 2, more specific theses will be formulated that cope with individual challenges.

1.5 Contributions and outline

The goal of this work is *not* to provide methods for increasing the recognition accuracy per se. Instead, my aim is to provide a basis for efficiently performing SMC inference in the large models imposed by situation recognition in everyday situations. For the purpose of this dissertation, I consider an algorithm A more efficient than an algorithm B if it is at the same time faster and has a smaller approximation error. As the motivation for this work are real-world applications, we define efficiency based on empirical results rather than theoretical complexity.

Definition 2 (Efficiency). *Let $P(X_{1:i} \mid y_{1:i})$ be the true filtering distribution for a specific model and sequence of observations. For an algorithm A and its approximation $\tilde{P}_A(X_{1:i}, y_{1:i})$, let e_A be the approximation error of $\tilde{P}_A$ wrt. P and let r_A be the runtime of A for computing $\tilde{P}_A$. Similarly, e_B and r_B are the approximation error and runtime of an algorithm B for the approximation $\tilde{P}_B$ of P.*

Algorithm A is more efficient than algorithm B (for the distribution P) if $e_A < e_B$ and $r_A < r_B$.

Importance of this work From a top-down perspective, situation recognition wants to compute the true state x_i for any given timestep i. Sensors are needed to make the real world accessible to computers. Not all modalities (e. g. location, temperature, ...) allow

error-free sensing, and it is impossible to sense the whole state of the world. Therefore it is impossible to be certain about the true state x_t and it is necessary to compute a probability distribution $p(X_t \mid y_{1:i})$. To correctly estimate the true state, probabilistic models are required that give a distribution $p(X_t \mid y_{1:i})$ which assigns as much weight as possible to the true state x_t. To make use of these probabilistic models, algorithms are needed that are able to efficiently and correctly compute the probability distributions.

Probabilistic models of the real world, for every-day situation recognition, tend to be very complex. Computing the values of the probability distributions is computationally hard, and the literature currently knows no such algorithm to correctly and efficiently (in real-time) compute the probability distribution $p(X_t \mid y_{1:i})$. Algorithms are either exact and unusable slow, or approximations with very large errors.

Instead of improving the algorithms, the literature (reviewed in Chapter 2) focuses on improving the models or sensor set-ups. Based on Thesis 2, the goal of this dissertation is to consistently improve the efficiency of SMC methods, specifically for the requirements and properties of human situation recognition. Only when the community has such algorithms to correctly compute the probability distributions, does it make sense to build and optimise probabilistic models for real-world applications, does it make sense to improve the sensor set up and does it make sense to perform assistive actions based on the estimated state. This goal is formulated in the central objective:

Central objective. *Improve the efficiency of Sequential Monte Carlo methods for online inference of human behaviour.*

Main contributions The three main contributions of this dissertation towards this central objective are:

- A thorough analysis of the state-of-the-art of approaches, evaluations and algorithms for recognising the situation of human users. The literature survey identifies open challenges and motivates this dissertation.

- Evaluating the approximation error and efficiency of the traditional particle filter for complex human behaviour models. Previously, the performance of inference algorithms has only been evaluated based on the accuracy wrt. the true state or manual annotations. I want to bring the focus back to minimising the approximation error and optimising the efficiency of the algorithms, which is a necessity for online inference in large state-spaces.

- A detailed analysis of the marginal filter algorithm, a new SMC method proposed by Krüger et al. [113]. This shows that the marginal filter algorithm is superior to the traditional particle filter for human behaviour models. This also includes a new derivation of an optimal pruning strategy and the development of an improvement for multi-user models.

This dissertation continues with the following 5 chapters.

Chapter 2 Towards the central objective, I first analyse the state of the art in Chapter 2. The first contribution in this chapter is a detailed review of different usage of terminology around *situation*. The second contribution is an analysis of open challenges, on which this dissertation is built. This survey focusses on the application domains, their complexity, inference methods and their capabilities. It can be shown that indeed either only very limited scenarios have been modelled and evaluated, or some of the requirements of Section 1.3 are not met. The results give more evidence for Thesis 2 and thus motivate the central objective.

Chapter 3 Chapter 3 presents modelling of human behaviour and Bayesian filtering, in particular SMC methods. This introduces the background in sufficient detail to understand the technical chapters. Here, we also introduce the models used for the evaluations in the following chapters. Some of these models are more complex than the models used in previous literature in terms of state space size, number of actions, use of actions durations and number of target classes used for the evaluations. This allows us to better test the algorithms with models that are closer to the requirements of human situation recognition.

The next two chapters deal with the identified open challenges from Section 1.4 and propose solutions. Two challenges are explicitly *not* in scope for this dissertation: (a) The *complexity of human behaviour* cannot be changed for every-day activities. Restricted applications may benefit from a constrained environment and a limited set of allowed behaviour, for instance in the application of assisted manufacturing [2]. (b) The lacking quality of *sensor data* may be coped with by improving the sensing technology; depending on the application, deploying more or (in some sense) better sensing hardware might also be a solution. However, both do not directly influence the operation of SMC algorithms and are thus not in the scope of this dissertation.

Chapters 4 and 5 deal with the individual challenges outlined in Section 1.4. These challenges emerge in the process from modelling over basic inference (without realistic sensor data) to realistic inference.

- Prior to any Bayesian inference is building a correct model. Thus, Chapter 4 first deals with the challenge of correctly building models, specifically for human behaviour. These models are a description of the state space, which is used for the inference (as described in Section 1.2).
- For the most part of the current research, only inference of the environmental state and actions (S and A) has been considered (this is discussed in Chapter 2). This is not very surprising, as knowing the behaviour of the user (e. g. cooking) and the state (e. g. if oven is turned on) is the most valuable information for further assistance. Thus, Chapter 5 deals with the properties of the state space and actions in the inference, and how these can be used to improve the efficiency of the inference.

In particular, the contributions of the chapters are as follows.

Chapter 4 Chapter 4 analyses errors in the *modelling* of human behaviour. Although the models are not directly part of the filtering algorithms, they influence the filtering operation and have a significant impact on the filtering performance and accuracy, and thus its efficiency. Therefore, a reliable process of building behaviour models is required; such process has been proposed by Yordanova and Kirste [218]. However, this process does not provide guidance for verifying the correctness of the model. In Chapter 4, I evaluated different models using model checking and indeed found several flaws (e. g. deadlocks or several types of inconsistencies).

Model checking is a methodology that is established in software and hardware design, which helps to find flaws in the design by verifying properties. When a property (e. g. free of deadlocks) is violated, the model can be corrected. However, as model checking is not an established technique in designing human behaviour models, it is unclear how the model engineer can define suitable properties. I analysed the errors presented in Chapter 4 and formulate different classes of properties that should hold in models of human behaviour. These classes help to formulate valuable properties and thus find flaws in the models, which is important for an accurate description and thus increase the inference accuracy.

Chapter 5 Chapter 5 presents the marginal filter, an SMC method designed for categorical state spaces. *Categorical state spaces* are common to human behaviour models, however, the common implementation of SMC is inefficient at inferring categorical state spaces. The marginal filter has been shown to outperform the traditional particle filter in these cases [111, 113]. However, there is currently no analysis of the reasons *why* the traditional particle filter fails and the marginal filter achieves better accuracy. I evaluate the differences in Chapter 5, along with a discussion of which model properties are responsible for this effect.

Additionally, Chapter 5 presents the *pruning* strategy for the marginal filter. For sampling-based approaches, a large number of particles are required to handle the *large state space* $\mathbf{X}$. The *pruning* step replaces resampling for the traditional particle filter and ensures that the computational complexity is bounded and not infinitely many particles are used. Resampling for the particle filter has been studied in depth. However, there is currently no study of pruning strategies for the marginal filter. I evaluate different pruning strategies and present and derivate an unbiased and linear-time pruning strategy in Chapter 5.

Chapter 6 Finally, Chapter 6 summarises this dissertation, discusses the contributions, and presents future research directions.

2 Literature review and problem analysis

Summary: This chapter reviews the state of the art in situation recognition, in particular the application domains, their complexity and capabilities of the inference. It shows that complex models lack support of efficient inference for real-life situations.

Contribution: This chapter provides a broad view on different definitions of terms related to situation *and provides own definitions based on the state-of-the-art. The main outcome of the chapter is an evaluation of the problem sizes of situation recognition in the literature. Based on this evaluation, I then derive current challenges of real-world situation recognition.*

Parts of this chapter are based on the article

> *[113] Frank Krüger, Martin Nyolt, Kristina Yordanova, Albert Hein, Thomas Kirste:* Computational State Space Models for Activity and Intention Recognition. A Feasibility Study. *PLoS One, 2014.*

We review the state-of-the-art in situation recognition. Objective of this chapter is to analyse how far current approaches can be applied to real-life recognition of the current situation. A comprehensive overview of the state-of-the art can be found in the introduction of the text book by Sukthankar et al. [197] and the survey article by Chen et al. [35]. In this chapter, we argue that current approaches cannot sufficiently handle the challenges of real-life.

First, we review the use of terminology in situation recognition in the literature in Section 2.1. We evaluate actual implementations and evaluations of situation recognition for human behaviour in Section 2.2. We show that the complexity of applications and the problem sizes are very small compared to what is expected from real-life applications. From the results, we identify the challenges of real-life human situation recognition in Section 2.3. This shows that not only the problem sizes are small, but also that current filter algorithms are not suited to handle significantly larger problem sizes. These results motivate the central objective (page 25) of this thesis and support Thesis 2:

Thesis 2. (repeated from page 24) *Inference of the situation of human behaviour in real-life requires many computing resources. Correctly estimating the true situation is very challenging.*

2.1 Defining *situation*

This dissertation provides improvements to inference for "situation recognition" in real-life of real users. However, "situation" and related terms are widely used in a non-standard way so that their meaning differs between authors and different fields of research. Having an understanding what is characteristic for *situation* and *activity* is important to correctly relate the state-of-the-art with the contributions of this dissertation.

Not all of the following terms are used in the remaining chapters of this dissertation. For instance, the formal model only describes "actions", but neither "activities" nor "tasks".

The outcome of this section is:

- It shows how broad and complex the area of situation recognition is, which gives an intuition for the complexity of this task.
- It gives a new, unifying view on the terminology, collected from different but related fields of research. This is a contribution in its own right.

Due to the broad topic, we cannot precisely define all terms. I merely want to give explanations of these terms with intuitions how they are used, backed up by usage and definitions in the literature. A formal model of real-life is introduced in Chapter 3, which has exact definitions for the model components.

The explanations here are chosen to reflect interpretations for the inference specialised to human situation recognition. Of course, the inference algorithms are also applicable to other application areas. Thus, terminology may differ; if appropriate or required by the use-case, the model engineer may of course deviate from these explanations.

Situation In this work, we use *situation* as the highest level of abstraction of describing and characterising users within an environment. Due to its nature of "being everything" and strong dependence on the application domain, we cannot give a precise definition. Many researchers from different fields have been concerned with finding appropriate definitions and descriptions for *situation* and the related *context* [42]. A circumscription of *situation* that conforms with the term as used in this dissertation is given by the following quote:

> 'an object or event is always a special part, phase, or aspect, of an environing experienced world—a situation'
>
> – Dewey [55], page 67

For example, one may describe the following situation:

Alice is in her kitchen and wants to prepare a meal. She is living alone and has invited three guests, she expects them to arrive in 90 minutes. Alice stands at the sink and is currently washing her hands, because she is going to prepare meat and wants to avoid contamination of the food. Alice has planned to cook steaks with mashed potatoes.

For the purpose of assistive systems, a situation helps in understanding what the users do; their location, time and circumstances of their behaviour; their motivation, what

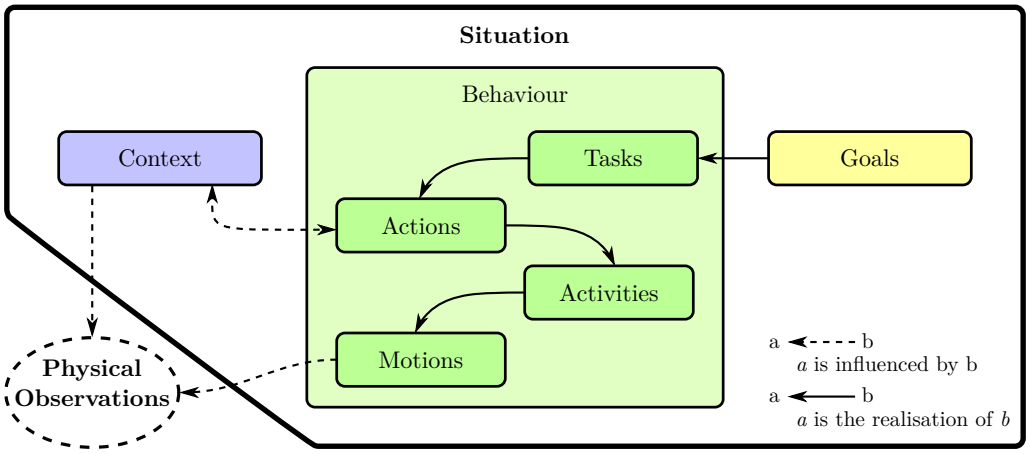


Figure 2.1: Overview of the different components of a *situation* and their relationships. Every realisation is usually a $1 : n$ relationship, i.e. a goal is realised by multiple tasks, a task requires execution of several actions, and so on.

they want to do and want to achieve. The idea that a situation helps in understanding and describes all circumstances of the users is also found in the following quote:

> 'In context-aware applications, situations are external semantic interpretations of low-level context [...], permitting a higher-level specification of human behaviour [...]. Situations inject meaning into the application [...].'
>
> – Bettini et al. [18], page 170

Note that a *situation* is not just the state of the environment - this is in analogy to the situation calculus, where *situation* is not just a state, but the complete history of actions. That is, a situation is anything that describes the users, their tasks, behaviours, goals and preferences, and all details from the environment that are relevant for the users. A similar characterisation was made by Chávez et al. [38]. In the sense of Coutaz et al. [46], the term *situation* resembles the concept of "context-as-process".

In view of these different definitions, I summarise *situation* as follows:

Situation: *the combination of* context*,* behaviour *and* goals*.*

How these terms are related to each other is shown in Fig. 2.1. The rest of this section now defines the remaining terms.

Context The context is the current state of the environment. Context plays a crucial rule in pervasive and ubiquitous computing, context-aware computing is one of the oldest research fields in this area. In the field of context-aware computing, authors define *context* differently. Sometimes, they use "situation" [56, 182] as a description, but never define situation precisely. Some authors also include the user's actions in the context [18, 34, 155, 182]. However, this is mostly due to the fact that *context* itself is regarded by many authors relative to the context-aware application (e. g. assistive system) [34, 56, 181, 208]:

> '[...] applications that are aware of the context in which they are run.'
> – Schilit et al. [181], page 85

> '[...] to enhance the behavior of any application by informing it of the *context* of its use. [...]
> **Context:** any information that can be used to characterize the situation of entities [...] that are considered relevant to the interaction between a user and an application [...].'
> – Dey et al. [56], page 97

I consider context always relative to the *users* within the environment, more specifically relative to the *actions*: context is what *directly* influences if actions can be executed, how they are executed and what is directly altered by an action. Thus, the behaviour of the users is in general *not* part of the context, this also conforms to the definition and usage of *context* by many authors [13, 25, 56, 181].

In this dissertation, context is the part of the situation that describes the current state (in the sense of a state machine) and condition of the environment, the state of being. Context is that which surrounds, i. e. a description of all entities in the environment, like location of objects and the power state of devices. Similar to Dey et al. [56], we summarise context as follows:

Context: *any information that can be used to characterise the states of entities at a given time, that is considered relevant to the interaction between users and the environment, including the users and the environment themselves.*

Note that we replaced "application" in the definition of Dey et al. by "environment", because in the sense of ubiquitous computing, there is not necessarily *the* application, but just an environment in which the users *may* be supported or monitored. With the terminology of Coutaz et al. [46], we define context as "context-as-state" as opposed to "context-as-process".

Because *context* is relative to the behaviour of users, the state of the mind of users, such as emotions, preferences, decisions, intentions or beliefs (the latter two in the sense of the Belief-Desire-Intention model Bratman [23]), can also be part of the context [13]. This has been used, for instance, in the modelling of behaviour of persons with dementia, where attitude [88] and ability [87] variables influence the user's behaviour.

What is considered context and needs to be modelled eventually depends on the model engineer and the application. Sometimes, even information like "the user is walking" or "the washing machine is washing" might be considered context (and not an action). Technically, context can be any object $s \in \mathbf{S}$, where $\mathbf{S}$ is an arbitrary set of contexts; context is mostly factored into context variables $\mathbf{S}_1 \times \mathbf{S}_2 \times \cdots \times \mathbf{S}_n$.

Behaviour Context itself is passive in the sense that what is described by context has no inherent executional semantics (e. g. "the light bulb is powered on" is part of the context). In contrast, the *behaviour* of a situation is active in the sense that what is described by behaviour has inherent dynamics (e. g. "cleaning the dishes"). Although

there is always some context, there might be situations with no behaviour; in this case, the context does not change.

Similar to Baxter et al. [16], we use *behaviour* to mean the overall process of acting which subsumes different levels of abstractions. There might be multiple, interacting (even competing) behaviours, in particular when multiple users are concerned. We summarise behaviour as follows:[1]

Behaviour: *the way a user behaves or acts and the way a device or system operates.*

Technically, a situation contains a set of behaviours.

Similar to context, what is considered *behaviour* depends on the model engineer and the application. In most cases, behaviour is any behaviour of the users, but any behaviour of the environment (like "automatic door is opening" or "printer is printing") might be considered behaviour as well.

Because classifying, recognising and characterising "human behaviour" as such is quite difficult, research has focused on different aspects and levels of abstraction of human behaviour. Typical aspects are *motion* [26, 32, 54], *activity* [3, 35, 120, 160, 202] and *action* [158, 190, 214]. These terms, however, are used differently among authors. While most authors consider an *activity* (high-level) consisting of *actions* (low-level) [33, 127, 176, 214], some define an *action* as based on *activities* [19, 160].

Based on the different concepts of behaviour found in the literature, I divide *behaviour* into *motion*, *activity*, *action* and *task*. This distinction is consistent with the definitions of Bobick [19]; furthermore, I distinguish *tasks* as an additional level of abstraction.

Motion The most primitive type of behaviour is physical *motion*, also called *movements* by Bobick [19]. Examples are moving the arm from left to right or to make a fist. According to this definition, *motions* do not depend on contextual knowledge. In general, the term *motion* is used to subsume any physical change in the real world.

Motion: *physical changes, e. g. of position (of parts) of the body of users or devices.*

Motions cause or influence sensor data from wearable sensors such as accelerometers. Recognising motions is not in the scope of this dissertation, but is a precursor to accurate classification of *activities*.

This definition also includes changes in brain activity, for instance. If one wants to support applications such as affective computing and attention management, the term 'motion' should be replaced by the more general 'physical changes'. However, in the context of this dissertation, 'motion' is a more natural term.

Activity In this dissertation, we use the term *activity* to mean more complex, possibly repeated, motions such as walking, running and riding bike. In this sense, *activity* is used in health care where it plays an important role for the assessment of the health and fitness of persons [31, 69, 90, 120]. Sleeping and laying as an empty motion in a specific body pose (namely, lying horizontally) is also considered an activity.

[1] Adapted from `https://en.wiktionary.org/w/index.php?title=behaviour&oldid=42320528`.

Table 2.1: Examples of tasks, actions, activities and motions. This table exemplifies the different levels of abstractions in modelling behaviour. Behaviour in columns to the right are a refinement of the behaviour to the left.

Task	Action	Activity	Motion
prepare meal	turn on stove	turn-on	rotate hand
give a talk	walk to stage	walk	move leg forward
set up illumination	press light switch	press	move hand forward
sleep	sleep in bed	sleep	lie
physical exercise	ride bike	ride bike	move legs

Activity: *a sequence or pattern of motions and body poses.*

According to this definition, activities do not have an effect on the *context* per se. For example, "walking" is by definition just a repeated motion of the legs and the body. I consider these physical activities only as an embodiment of behaviour of a greater purpose. For example, one may perform first the activity "walking" followed by "bicycling". This is what many activity recognition methods can detect [94, 180].

Sometimes, *activity* refers to more complex behaviour which require more cognitive skills [3, 127, 130]. This is used, for instance, in the assessment of activities of daily living (ADL), such as dressing, housekeeping, shopping and preparing meals [92, 195]. We will use the term *task* for this kind of behaviour (see below).

Action According to the previous definitions, motions and activities are simple movements with no inherent relation to the rest of the situation. In contrast, *actions* depend on the *context* and can also change the context. Due to their ability to modify context, *actions* are usually executed with some intention.

Action: *behaviour which atomically changes the context of the situation.*

The dependency of actions on the context is captured with the terms 'precondition' and 'effect'.

Precondition and effect: *How an action depends on context is called* precondition*, the way it modifies context is called* effect*.*

For example, consider the hypothetical action "get on bike". To get on the bike, the user must be at the same location as the bike (the precondition). As a result of this action, the bike is occupied and the user is on the bike (the effect). Here, the action depends on context (location of the user and bike) and changes context (status of the bike and location of the user).

Because *actions* depend on the context, the execution of actions also depends on each other – this is called *causality*. For example, one may not perform the *action* "go to bathroom" followed by "get on bike" (except if there is a bike in the bathroom ...). On the other hand, an *activity* may also be considered an *action* if it is executed in a specific

context with an intention, for instance "running" as a physical exercise or running away from a possible risk [33].

The property of atomicity of actions reflects that actions are the simplest causal behaviours, i. e. there is no specific order in which the effects can be divided, the effects apply simultaneously. Of course, the atomicity is only a property of the *model* and not of the real world, where the effects of an action do not necessarily take effect at the same instant.

Executing actions usually involves executing certain activities and motions. However, actions can also model decision making, problem solving or psychological processes which do not involve any physical activities. These actions still change the context, for example that the user is in a certain state of mind.

Task While *actions* are simple, atomic causal behaviours, we refer to higher-level actions as *tasks*. Executed tasks achieve some intention, i. e. cleaning the kitchen. (Again, whether an action is considered "low-level" or "high-level" is mostly a design decision made by the model engineer.) Completing tasks usually requires to execute several actions; the same task can usually be achieved by different sequences of actions. If one assumes goal-directed behaviour [9, p. 352][12], the sequence of actions of a task are determined by planning by the user.

Task: *any sequence of actions that achieves a specific purpose.*

Sometimes, *activity* is used to denote high-level actions [3, 130]. To distinguish these two meanings of *activity*, we use the term *task* for high-level actions. The confusion that activities are sometimes considered low-level *motions* [160] and sometimes high-level *tasks* [64] can be explained by the fact that often *tasks* have typical *motions*, such as the task of "washing". Depending on the authors, the recognition target is either the characteristic *motion* based on raw sensor data, or the *task* based on basic *actions*.

Within this conceptual framework, *motions* and *activities* do not change context, only *actions* do. However, *motions* may generate observable sensor data (e. g. accelerometer signals). Of course, this distinction is only at a conceptual level. For example, the *action* "press light switch" may be modelled to consist of the only *activity* "press", which may be modelled by a single "move hand forward" *motion*. In the real world, the movement of the arm does change the state of the button; however, the effect on the context is *modelled* within the *action*. Table 2.1 (page 34) gives some examples of the different levels of abstractions.

Goal When recognising the actions of a user, one may assume that the user is following some *goal* [98, 162]. This is in particular reasonable when the application is to assist users in doing their work. A goal is a description of how a desired situation should be.

Within this dissertation, a goal is a condition on the situation's *context*. Achieving a goal usually requires to achieve several tasks, and consequently to execute several actions. For example, the goal of eating may require to execute the tasks prepare meal, cook, prepare dish, eat and clean up.

Goals also play an important role in the Belief-Desire-Intention (BDI) model of Bratman [23]. The BDI model is a model of human thought which is also a popular foundation for software agent architectures [80]. However, within the BDI model, desires are not necessarily goals that the user currently and actively tries to achieve, but resemble preferences.

Goal: *a desired context motivating the agent to perform actions towards this context.*

Goals of users are set and unquestioned – a goal (e. g. to be not hungry and not thirsty) is its own purpose. In contrast, I consider an *intention* to be the desired effect of a task or action, i. e. an intention is the *reason* for executing a task or action. For example, the intention of eating is to be not hungry. A similar distinction between goals and intentions can also be found in the BDI model, where intentions are a plan that the user has committed to [80].

2.2 Related work in situation recognition

In this section, we review different approaches how situation recognition is achieved and evaluated. We analyse actual applications of situation recognition and their problem sizes. In particular, we evaluate how complex these applications are in order to discuss in Section 2.3 how far these evaluations can be considered real-life applications. I show that complex models (which try to model many aspects of real life) have not been evaluated under every-day conditions, and that evaluations under every-day conditions do not use complex models. I argue that complex models lack support of efficient inference.

I first describe my methods in searching for related work and present my evaluation criteria in Section 2.2.1. Section 2.2.2 and Section 2.2.3 present the findings and contain two tables Table 2.3 and Table 2.4 summarising the results.

2.2.1 Methods

Survey sources Due to the massive amount of research in this area, this survey can not review even a small fraction of the research. Therefore, I selected typical representative work; the list of papers is not meant to be complete, but shall only give an overview of related work. Potential *candidate papers* were selected mainly from survey articles (most importantly the survey from Chen et al. [35]) and web search. Candidate papers were then reviewed based on the following hard selection criteria:

Peer-review Only papers and articles from established peer-reviewed journals and conferences where included. Examples include the journals *Artificial Intelligence*, *Pervasive and Mobile Computing* and the *International Joint Conference on Artificial Intelligence* (IJCAI).

Experimental evaluation One of the main contributions of the paper must be a study that evaluates the approach presented in the paper. It is not considered sufficient if the paper solely describes an inference algorithm. The main purpose of this study is to evaluate the complexity of evaluation targeted to real-life applications.

Algorithmic description Nonetheless, the *inference* of the situation must be a central topic of that paper. This also includes a brief description of the algorithmic method used.

Contribution The paper must claim to provide some contribution in the area of situation recognition. Usually, this includes supporting a more realistic use-case, or dropping previous restrictions (e. g. that there is only a single user).

I consider publication bias not to be an issue for this review. I am mainly interested in the largest or most complex models that have been used for particular inference methods.

Evaluation criteria Selected papers were then evaluated according to a number of criteria. The evaluation criteria shall help to estimate how far the related work deals with the challenges of real-life situation recognition.

The evaluation criteria where guided by the following questions:

How complex is the inference task? Different works have focussed on different problems to be solved. Often, a specific issue is dealt with, and other issues have been set aside. This often leads to simpler solutions which are not applicable to real-life, and can therefore not be compared as-is to other solutions.

The following criteria have been chosen as a measure for the complexity of the inference task: the *recognition target*, if *causality of models* is supported and availability of *durative actions*.

How complex is the data and application scenario? Even if the inference task that is formulated can be theoretically complex, most work evaluate their approaches on very simple data sets. For instance, the application scenario is very confined, or very few different actions are actually allowed. This, of course, simplifies the inference in practice, and solutions to the theoretical complexity are not developed.

I decided to use the size of the *state space* and the number if *ground actions* as a measure for the complexity of the data sets. Additionally, the number of *target classes* describes how many different results are actually used for the evaluation of the performance.

What inference method is used? The inference method is selected based on the requirements of the inference task and the complexity of the data. Some inference methods have a higher time complexity of the inference and are thus not reasonably useful for complex real-life applications. Thus, the inference method is an additional indicator of the complexity of the inference task.

I collected the *inference method* used and whether the inference is exact or *approximate*.

This survey explicitly only covers *inference* of the situation. How the models are created – whether completely manually specified, learned through training samples, or a hybrid approach - is not in scope of this review.

The different evaluation criteria are explained now in more detail. These criteria correspond to columns in Table 2.3 and Table 2.4, which summarise the results. Based on the initial questions, the criteria are split into two groups: qualitative data of the intended application, and properties of the inference task.

Application criteria These criteria define the context in which the inference shall be executed. Thus, the application determines the requirements of the inference.

Application scenario The column "Application" denotes the application scenario that is used as a motivation for the paper and the source of the data set for the evaluation. This shows that recognising every-day human behaviour is indeed an active research topic. However, no work (including this dissertation) to date monitors the situation of a person in every environment and in every situation. All approaches are focussing on only a part of the user's life (for instance in the office).

Recognition target The column "Recognition" shows what parts of the *situation* is computed/inferred based on the sensor data. The values refer to the definitions of Section 2.1. As this dissertation is focussed on recognising the causal structure of the situations, this review is focussing on work that can recognise actions and/or goals. This criterion has been included, as I assume that recognising single parts of the situation (e. g. only the action, or only the goal) is conceptually easier to implement than a *joint* recognition. Recognising context and activities *alone* has to deal with other challenges, therefore these papers are a minority in this survey. There are other survey articles on context recognition [13, 18, 155] and activity recognition [3, 90, 120, 160]

Support of duration models Executing actions in the real world by a user always requires time. However, not all approaches *model* how long executing actions takes. Some models support a very restricted set of duration distributions, e. g. allow actions to have a geometric distribution or use a time out after a maximum time threshold has been reached. Other are not applicable to real-world applications at all and assume that all actions are completed within a single timestep. Without a good model of the actions' durations, the inference cannot use all available prior knowledge. The trade-off is usually more efficient inference instead of more expressive models.

This criterion states if the model supports a wide range of duration distributions. The formal criterion is that at least two different families of probability distributions must be supported (or can be approximated).

Context-based causal model This criterion is a combined criterion that is fulfilled when the model includes a specification of the *context* and is based on *causal* dependencies between situations. This criterion is motivated by the fact that human behaviour is causal, and that the *context* modelling is also central for execution actions. See Section 1.3 on page 22 for a definition of causal models.

Some approaches recognise, e. g., activities or actions as labels. Without an underlying causal model, there are two extremes possible. Either, these model allow *any* action to be executed after any previous action, without any causal relationship between actions. Or, these models have a *restricted set* of permissible action sequences (plan libraries). These plan libraries may be based on causal dependencies, but still do not include a model of the underlying *context* that gives rise to these causations.

Both extremes (either all action sequences are valid, or only a pre-defined set) cannot adequately cover the complex trajectories of human behaviour. This criterion has been introduced to evaluate if including a causal, context-based model leads to more complex models and inference in practice.

Use of real sensor data Not all works claiming to do situation recognition have been applied to real sensor data obtained from observing real users. Some approaches use synthetic or simulated data, partially simply due to the lack of real sensor data. While these approaches may certainly provide a contribution, they have not been shown to work with noisy, missing and ambiguous sensor data in practice.

Inference complexity The following criteria are partly quantitative data that influence the inference complexity, as well as the choice of the inference method itself. Both determine the accuracy and efficiency of the inference.

State space size The size of the state space of the *context* is one indicator of the complexity of the situations that are to be inferred. The larger the state space, the more aspects of the situation are usually modelled, and the more the model can be considered to be closer to real-life. A small state space size requires less computational power for inference, the size of the belief state is consequently smaller, and thus the probability of inferring the correct situation by chance is increased.

For this dissertation, we are mostly interested in categorical models of human behaviour. Therefore, the size of the state space only includes the *discrete* state space. If the state space is hybrid, then only the discrete state space is counted.

Number of actions As another indicator of the situations' complexity, we use the number of *actions* that *can* be performed in the experiment by the user. The number of actions represents, to some extent, the degree of freedom that is allowed in the evaluation.

Number of target classes For most evaluations of the reviewed literature, the performance of the inference is evaluated using accuracy and related measures (such as F_1-score) of the inference output (e. g. the current action or the user's goal) compared to the true value. However, what the inference method is inferring is often not completely used to actually evaluate the performance, and hence is not used for the final report of the performance. For example, actions might be grouped to action classes (e. g.'drinking' consist of both actions 'drink water' and 'drink

tea'). The *number of target classes* shows how many different classes the inference has to distinguish.

Inference method The inference method is collected to have an overview which methods are typically used in the literature. The list of inference methods is used to draw comparisons which classes of algorithms are used for which problem complexity (as defined by the other criteria).

We distinguish between probabilistic and non-probabilistic inference, where different approaches are grouped into popular groups. Probabilistic inference is further distinguished between non-sequential Bayesian inference, Bayesian filtering (requiring the model to be a Dynamic Bayesian Network), Sequential Monte Carlo methods (as a special case of Bayesian filtering that is in the focus of this dissertation) and discriminative inference. Discriminative inference differs from the Bayesian approach in that it uses training data instead of a system model. The training data contains samples of direct mappings from observations to the true situation.

Non-probabilistic inference is separated into logic-based inference, grammar-based inference, and inference based on Markov Logic Networks (an approach to unify probabilistic reasoning and logic rules). "Non-probabilistic" here refers to methods which use other approaches than directly computing probability distribution. Of course, probabilistic calculations are still allowed. For instance, grammar-based inference is based in grammar rules to replace non-terminals with other non-terminals and terminals, but many approaches add probabilities to these rules.

Joint recognition Recognising single parts of the situation (e. g. only the action or the goal) is conceptually easier to implement than a *joint* recognition. For this reason, some papers recognise multiple parts of a situation (e. g. actions and goals), but do this in a multi-layered approach: in a first layer the actions are recognised based on the sensor data, in a second layer the goals are recognised based on the inferred actions. While this approach is easier to implement and computationally more efficient (no joint probability distributions), it has the drawback that the prior knowledge about the situation cannot be used to infer the complete situation.

As an example, assume a layered action, context, goal (`A|C|G`) recognition where

- we inferred that action a_1 has probability of 0.4, and a_2 has probability of 0.6, based on sensor data,
- based on the current estimated state, we know that goal g_1 has probability of 1, and goal g_2 has probability of 0 and
- we know from prior knowledge that goal g_i is achieved by action a_i.

Consequently, a_1 has probability 1, and a_2 has probability 0. Using a two-layered approach, however, a_1 would be estimated with a probability of 0.4, which does not take the additional information from the upper layers (i. e. context recognition) into account. In contrast, a joint approach would correctly recognise a_1 as the only possible action (for instance using a inference in a single DBN that models the dependencies between actions and goals).

Table 2.2: Detailed explanations of all columns and factors of Table 2.3 and Table 2.4.

Column Level	Description	Level	Description
Application			
O	office environment or meeting scenario	K	a household kitchen environment (a special case of ADL)
A	activities of daily living (ADL)	o	other
Recognition			
a	activities	C	context
A	actions	T	tasks
G	goals		
Sensors			
D	dense sensors (e. g. RFID)	L	location data
V	video stream	W	wearable sensors (e. g. IMUs)
(none)	only synthetic or simulated data		
Method			
BF	Bayesian filtering (using a model based on DBNs)	BI	Bayesian inference (unless using a DBN)
SMC	Sequential Monte Carlo method	G	grammar or matching based inference
L	logic-based inference	D	discriminative inference
MLN	Markov-logic network inference		

In the column "Recognition", all recognition targets that are inferred are listed. A "|" indicates where there is no joint recognition; this is the case when the paper describes a multi-layered approach. For example, A|G is a two-layered approach (recognising first the actions, and based on the actions recognises the goals) and AG would denote a joint recognition (of actions and goals).

Exact inference Exact computations are more desirable, however these become challenging when the model sizes grow. Therefore, for this survey I collected whether the method does exact computations or approximates the inference.

2.2.2 Paper review

The selected papers are summarised in Table 2.3 and Table 2.4. The columns correspond to the evaluation criteria described in the previous section. Some columns have a fixed set of factor levels; these factor levels are summarised in Table 2.2.

In total, I selected 44 articles to include in my review. All articles satisfy the criteria defined in Section 2.2.1. The articles' publication years span from 2003 to 2016, see Fig. 2.2 for the detailed distribution.

Table 2.3: Qualitative data of the literature, columns are the criteria from Section 2.2.1.

Reference	Application	Recognition	Durations	Context/ Causal	Sensors
[6]	o	G			
[12]	o	G		✓	
[20]	K	T			
[26]	A	a			W
[27]	O	A			L
[36]	AK	A		✓	D
[37]	A	a			W
[49]	O	CA		✓	V
[51]	o	A		✓	
[64]	K	A	✓		L
[66]	o	T			
[73]	A	a		✓	D
[74]	o	T			
[81]	A	G			
[82]	AO	aCA		[a]	W
[85]	Oo	AG		✓	
[87]	K	CA		✓[b]	D
[88]	A	C | A		✓[b]	L
[89]	Ao	G			
[92]	A	A			D
[102]	o	a | C | A | G			L
[116]	O	A	✓	✓	L
[121]	A	A			D
[124]	K	T			
[126]	A	CAG	✓	✓	L
[127]	A	A			
[130]	A	A | T			D
[131]	o	AT		✓	
[140]	o	a	✓		V
[143]	K	CA		✓	L
[144]	O	CAT		✓	L
[150]	o	G		✓	
[153]	O	C			
[156]	AK	A | T			DW
[163]	Ko	G		✓	
[164]	KOo	G		✓	
[169]	K	A | T			D
[170]	o	A	✓		
[176]	AK	C | AT			D
[178]	o	CA		✓	L
[189]	o	AT			
[190]	o	A	✓		DV
[212]	AK	CA			DV
[216]	A	a			D

[a] The context is unknown in the initial state and the actions depend on the context. However, the context in the transition model itself is static, i. e. does not change over time.

[b] The action order is further restricted by a plan library.

In this section, I review a few representative and typical papers which give an overview over the different approaches. The results of the survey are summarised in the next Section 2.2.3.

Shi et al. [190] are one of the first who recognise sequential *actions* of a user following a plan and theoretically supporting arbitrary action durations. The action sequence has to follow a partially ordered plan of actions, the model does not allow a context-based modelling. For inferring the action sequence, the authors introduce Discrete Condensation as a Sequential Monte Carlo (SMC) method. Discrete Condensation has been designed for the discrete set of plan steps – this is one of the first works that proposes a discrete variant of SMC for situation recognition. The evaluation of the proposed approach is done by recognizing 14 actions during glucose monitor calibration. Observation data from a computer vision based tracker is used to classify the actual action. Their evaluation uses a model with a state space of approximately 20,000 states.

Table 2.4: Inference methods and complexity metrics of the literature, columns are the criteria from Section 2.2.1.

Reference	Method	Exact	State space	Actions	Targ. classes
[6]	BF	✓	912[f]	48	19
[12]	BF	✓	70,000[f]	9	3
[20]	L			40	6
[26]	D	✓		4[a]	4
[27]	D		720	6	1
[36]	L		?	30[f]	8
[37]	D	✓	238[f]	34[a]	34
[49]	BF	✓	250,000[f]	80[f]	5
[51]	BI		1,000[f]	10[f]	10
[64]	BF	✓	28	12	6
[66]	G	✓		7[f]	
[73]	MLN		?	9[a]	9
[74]	G	✓	4,000[f]	1296[fb]	
[81]	BI	✓		?	
[82]	SMC		253[f]	253[f]	23
[85]	BF	✓	5,000[f]	49[f]	
[87]	BF	✓	200,000	26	
[88]	BF	✓	70,000	24	6
[89]	D		512[f]	11	9
[92]	BF	✓	300[f]	12	15
[102]	G	✓		5	
[116]	SMC		70,000[f]	75[f]	10
[121]	BF	✓	8	8	8
[124]	G	✓	?	5	
[126]	SMC		1,000[f]	?	6
[127]	D	✓		30	5
[130]	L	✓		25[f]	6
[131]	G	✓	?	503	13
[140]	BF		2100[f]	14[a]	10
[143]	BF		96	13	3
[144]	BF		3500[f]	14	3
[150]	BF	✓		?	
[153]	G	✓	$< 1{,}000$[f]	?	
[156]	D | G			14	7
[163]	BF		10,000[f]	63[f]	3
[164]	BF		70,000[f]	29[f]	5
[169]	G	✓	$< 5{,}000$[f]	< 20[f]	3
[170]	G			14	14
[176]	G	✓	100[f]	10[f]	10
[178]	MLN		11,000[f]	200[f]	4
[189]	G		200[f]	9[f]	2
[190]	SMC		20,000	14[f]	14
[212]	BF	✓	528[f]	528[f]	33
[216]	G | D	✓		13[a]	13

[a] No actions in this evaluation; this figure is the number of activities instead.
[b] All actions occur exactly once and unambiguously identify the executed plan.
[f] This value is not explicitly mentioned in the paper, but has been estimated from the descriptions.

In contrast to Shi et al., Krüger et al. [116] evaluate action recognition using a causal, context-based model which also supports action durations. Their work is based on a symbolic model, where actions are described using preconditions and effects. The model also supports tracking the state and behaviour of multiple users simultaneously, making it applicable to a wider range of applications. They evaluate their approach in two different scenarios of the same meeting domain. Inference is done using traditional particle filtering, without adaptions to the discrete state space as Shi et al. [190] used. Despite this, they used a model with a larger state space of approximately 70,000 states.

Baker et al. [12] provide a framework for inferring the goals of users. They use a Markov Decision Process (MDP) to model user behaviour based on their goals. For recognising the goal, the MDP policy for each possible goal was computed. The goal is then determined by the policy that yields the most likely transition probabilities.

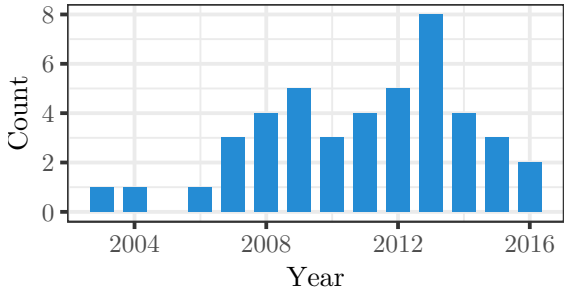


Figure 2.2: Histogram of the publication years of the articles covered in this survey.

The MDP models an agent moving on a 2-dimensional grid towards a goal, avoiding obstacles. Baker et al. created three different models, incorporating additionally dynamic goal changes and sub-goals. The state of the process contains the position of the agent and locations of obstacles. The model is thus context-based and causal. Transition probabilities are modelled according to the density $p(a \mid s, g) \propto \exp(\beta\, c_g(a, s))$, where $c_g(a, s)$ denotes the expected cost to reach goal g when performing action a in state s. Using this distribution, different levels of rationality (i. e. goal-directed action selection) can be modelled by adjusting the parameter β.

This approach is important towards recognising user's goals based on their behaviour. However, the work has some limitations that prevent it from being used for every-day situations. The model does not include actions with duration. The authors assume a fully observable world, where they observe the user's positions. Additionally, in their scenario exact inference is possible, and the authors state that for more complex problems approximate methods must be used.

Ramírez and Geffner [163] use a similar approach of modelling user behaviour and inferring their goals. For this, they synthesise a planning domain, where the goal-distances are computed by classical planners. The goal-distances are then used as a measure how likely the agent pursues a given goal. The authors assume to observe a subset of the executed actions and use traditional planning domains, one of which is based on a kitchen scenario. These domains are synthetic in the sense that they do not (necessarily) correspond to a real-world application domain, but are used as performance benchmark domains for planning algorithms. Real-world observations are also not included. Instead, the planning problem contains observations as goals (i. e. find a sequence of actions which generates these observations), and thus the planning problem must be solved for every new observation again. This leads to a complexity which is quadratic in the number of observations.

In their second work [164], Ramírez and Geffner use POMDPs for inferring goals of a POMDP agent. The approach assumes the POMDP to be known (except for the goals) to the inference. The inference gets an incomplete sequence of actions, the POMDP is used to compute the expected costs (similar to their previous work) from the current state to the goal.

The three works of Baker et al. and Ramírez and Geffner use models with up to 70,000 states. None of the approaches within this survey which recognise *only* the goal use real sensor data.

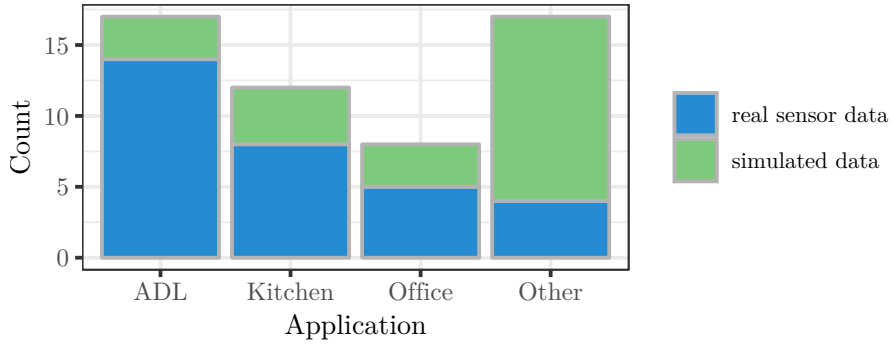


Figure 2.3: Number of different application scenarios in the surveyed literature. If a paper features different domains, each is counted individually. For real sensor data, activities of daily living (in particular the kitchen domain, counted separately) and office domains are predominant.

Kiefer and Stein [102] show a working application of goal recognition that integrates context, activity and action recognition. Their application is a user interface for an outdoor game, which shall show the appropriate screen for the current situation of the user (e. g. show map on correct zoom level or show hints for the current task). Instead of assuming perfect action observations (as other goal recognition approaches do), Kiefer and Stein use a classification of activities and context (locations only) based on motion tracks in a pre-processing stage. The authors use context free grammars (CFGs), where terminal nodes are *activities*. Context is integrated by adding spatial constraints on production rules in the grammar. *Actions* and *goals* are recognised by parsing the recognised *activities*. Context, activities, actions and goals are thus recognised all independently of each other, where the recognised results are passed to the next layer of recognition.

Liao et al. [126] present an approach for *jointly* inferring the context, the user's actions and goals using a unified model. the current transportation routine, with application to supporting cognitively impaired people. Their approach reads raw GPS data and is able to infer context (current location and speed of the person, location where the car has been parked), activity (if driving car or riding bus), and goal (the destination). In contrast to most other goal recognition approaches, the model has only 1,000 discrete states. This work is the only that infers the goal, uses real sensor data and also accounts for action durations within this survey. The authors employ a hierarchical Markov model and DBN-based inference with a Rao-Blackwellised particle filter. They also show how to learn (unsupervised) the user-specific typical destinations and routes, and how to recognise erroneous actions.

2.2.3 Results and discussion of the survey

In the following, I evaluate the results of the survey. This evaluation shows what models and algorithms have been considered and evaluated in the literature, and identifies gaps in the current work. These gaps lead to the open challenges discussed in the next section.

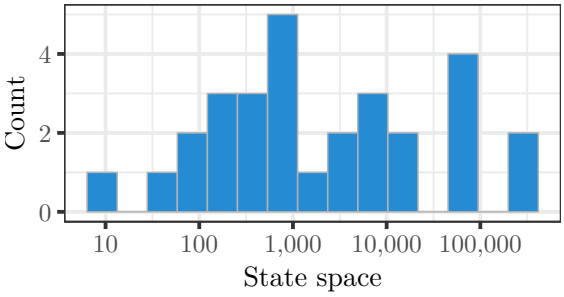


Figure 2.4: Histogram of the sizes of the discrete state spaces in this survey.

Application Scenario From the 44 papers of this survey, most (25) use a household scenario and recognise activities of daily living. Out of these, twelve papers have an evaluation focussed on the kitchen and food preparation domain. A set of eight papers uses an office or meeting scenario for their evaluation. For the other domains, no specific application scenarios are salient. In total there are 13 papers that have no application related to activities of daily living, for instance emergency response [150], out-door games [102, 178] and hospital monitoring/documentation [170].

From the papers in this survey, most research interest is in recognising activities of daily living. In particular the kitchen domain is a prominent example. One reason may be that the Kitchen Task Assessment [15] is an established test for cognitive performance, and health-care and ageing population are the most important motivations for most papers. Following activities of daily living, the office domain is the second most-used application scenario of the papers in this study. Other application domains are the minority, in particular when real sensor data is used (see Fig. 2.3).

State space size Although the state space size is an important feature to assess the complexity of the inference task, it is only explicitly reported in seven papers. The description was detailed enough to allow a rough estimate of the state space in 22 additional papers. For 15 papers, I could not reliably determine the state space size.

The state space size of discrete states in the literature was at most 250,000, with first, second (median) and third quartiles of 300, 1,000 and 11,000 states. That the state space is only reported in 16% of the papers and that state space sizes are relatively small, can be explained by the fact that efficient inference was not a target in most papers. They are often feasibility studies or show-cases for new modelling approaches. All work until now only models a small fraction of every-day life, thus no large state space sizes occurred.

The model with the largest state space in this survey models a group meeting [49]. The model contains hierarchical behaviour definitions of four users, for instance give presentation, discuss, group discussion, or two users talking to each other. The model also contains, among others, positions of users (in terms of one of 5 different places), talking direction (6 different directions) and body position (sit or stand).

The second largest state space (200,000 states) is generated by a model of making tea in a home kitchen of a single user. The authors modelled moving items between locations, opening the tea box, filling water to the kettle and others as actions (at this level of granularity). The context model contains locations of items (e. g. cup, tea bag

and spoon) and conditions of the kettle (e. g. powered and full) and tea box (open or closed).

Although these are the largest models, they only model a very specific scenario with the context variables and actions that are necessary for these scenarios.

The smallest model has eight states [121]. Each state is one of the eight household actions that the user executes (e. g. take out garbage or make breakfast), without additional environmental context.

In general, there is no obvious correlation with the application scenario or available sensors. The model sizes also do not significantly change with the publication years; the evaluation of the largest model was published in 2008. As can be seen from the histogram over the state space sizes (Fig. 2.4), small state spaces (less then 10,000 states) are predominant. I conclude from this survey, that situation recognition has only been evaluated in very limited scenarios, often modelling one specific task. Models with large state spaces (much more than millions of context states) have not been evaluated in the literature.

Number of actions Unlike state space sizes, the numbers of actions are reported more often (in 23 papers). The descriptions were in most cases sufficient enough to assess the possible actions of the users, four papers do not contain sufficient information to estimate the number of possible actions. The possible actions of users are always described as part of the experimental design, therefore reporting them is considered important for reproducible experiments.

This shows, to some extent, that capturing data is considered more important than describing the model of the data used for the inference. In fact, collecting high-quality sensor data of real users is expensive and time-consuming [95, 171].

The first, second (median) and third quartiles of the number of actions are 10, 14 and 42 actions. Evaluations with far more than 50 actions are the minority, although the largest number of actions is 1,296 (from a randomly generated benchmark library). However, the papers with most actions impose strong restrictions on the order of actions [74, 131] or effectively restrict the set of possible actions by additional machine learning [212].

Similar to the small state space sizes, I conclude that complex models with many actions have not yet been used for situation recognition. Models with large state spaces also do not have many actions (and vice versa), this becomes apparent in Fig. 2.5. Thus no model can be considered "complex" when using only number of actions and state space size as factors.

Support of duration models Interestingly, only six evaluations use models that support a flexible modelling of duration distributions. Other papers support a single type of duration distributions, or do not model durations at all. In some cases actions can last arbitrarily long, and the actual duration of the action's execution has no effect on the inference [102]. In other models, action durations are associated with constraints on their durations. These constraints, however, do not model the behaviour and they are used to

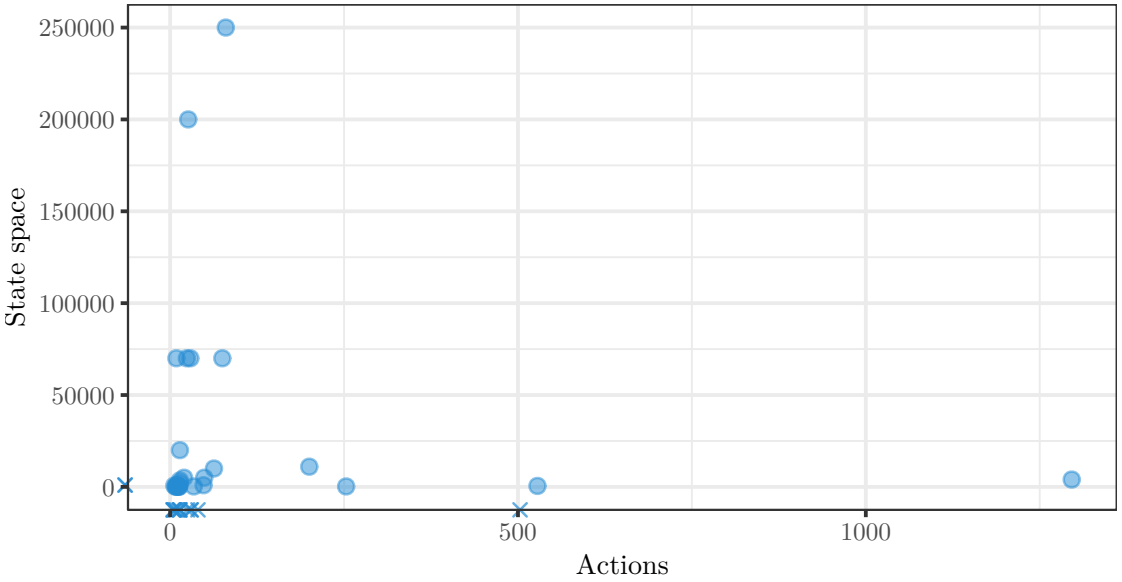


Figure 2.5: Correlation between the number of actions and the size of state space. Missing values are plotted on the axes lines. No model of the evaluations in the surveyed literature has both a comparatively large number of actions and at the same time a large state space.

rule-out certain actions after the classification [49, 88, 176].

None of the papers that focus on goal or task recognition support duration distributions. In all models in the literature I am aware of, action durations do not influence the reachable goals or the effect of actions (this is also true for the model in this dissertation). They assume that the action has been correctly identified and usually focus on the task structure [20, 102, 163, 169], and at most incorporate time *constraints* [124, 130, 150].

Many approaches of action recognition use models that allow efficient inference but have limited support for duration distributions. One of the reasons is that action and activity recognition, in particular using wearable sensors, is computationally intensive. For instance, Hidden Markov Models and extensions are often used [92, 143], but they only support geometric duration distributions (and their sum, the negative binomial distribution). Other approaches use template-matching to reconstruct actions [82, 127] based on time-features and therefore cannot model arbitrary duration distributions.

In comparison with models that do not support duration distributions, inference using models that use duration distributions seems to be more complex and less efficient. With one exception, all papers that support duration distributions use approximate inference methods. The approach with exact inference, however, has a state space of 28 states [64]. Similar, except for one model with 1,000 states [126], no joint recognition is supported, but solely either activities or actions are recognised. No evaluation with duration distribution uses wearable sensors.

My literature survey shows that action duration models are not widely used, and have only been used in small models.

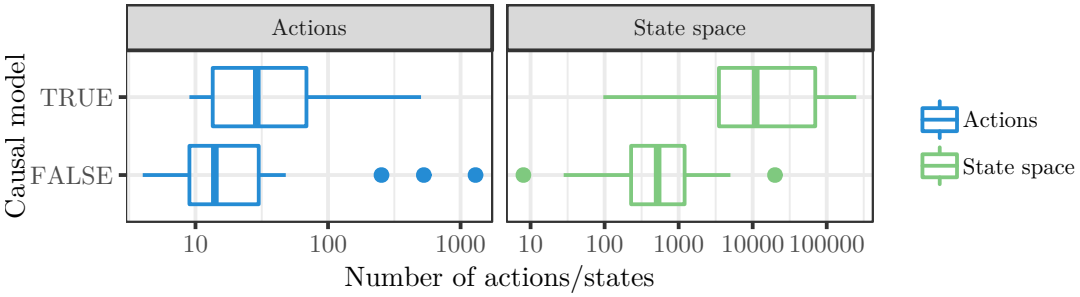


Figure 2.6: Boxplots showing the correlation between causal, context-based modelling and the complexity of models. While the number of actions is increased moderately, the state space is significantly larger for causal, context-based models.

Context-based causal model Out of the 44 papers, 17 support causal models. For example, both Dai et al. [49] and Krüger et al. [116] model the structure of meeting scenarios with presentations and discussions.

Figure 2.6 shows the state space size and number of ground states for both causal, context-based models and non-causal models. As hypothesised in the explanation of the criteria "context-based causal model" (Section 2.2.1), we can indeed observe a correlation between the use of causal models and an increased model complexity. There is a significant difference in the size of the state space between causal models and non-causal models ($U = 29.5$, $p = .001$). In particular, causal, context-based models tend to have a much larger state space (effect size[2] 0.85). There is also an effect that these models tend to have more actions; however, this effect is less strong ($U = 124.5$, $p = .08$, effect size 0.66).

This correlation can by explained by a reinforcing dependency between both. First, if a complex model has to be specified, a causal approach is usually easier to develop for human engineers because causality is inherent to human thinking [45] (see also the paragraphs on symbolic and causal models in Section 2.3.1). Second, when causal models are used, the models tend to grow due to the *computational* and *generative* nature of such models, including combinatorial state space explosion [219].

Inference in causal models also seems to be more complex. From all evaluations that use causal models, 59% use an approximate inference method. In contrast, of the evaluations without a causal model, 30% use an approximate inference. This shows that inference in these models (probably due to larger state spaces), is computationally more complex and exact solutions are often infeasible.

As a conclusion, causal models have larger state spaces, and therefore computational complexity of the inference is increased.

Use of real sensor data Real sensor data is used by 26 papers. All other evaluations use simulated or virtual sensor data as input. For instance, Baker et al. [12] simulate walking patterns, Ramírez and Geffner [163, 164] use artificial planning domains with

[2]Common language effect size: This is the ratio of how many causal models have a larger state space than non-causal models, for all pair-wise combinations.

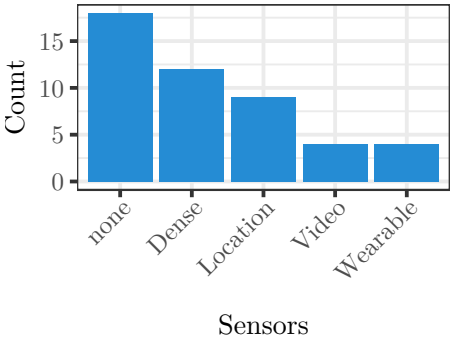


Figure 2.7: Histogram of the different sensor modalities used. The factor levels correspond to Table 2.2. If an evaluation uses multiple modalities, each is counted separately.

simulated data and Rogge-Solti et al. [170] use annotations of actions (with missing data) as input.

As can be seen from the histogram of the sensor modalities in Fig. 2.7, the least frequently used modalities are wearable sensors and video input. Evaluations with wearable sensors also use only models with small (less than a thousand) states. It also seems that public datasets with simple state-change sensors (e. g. tilt sensor at the kettle means water is poured [36], passive infra-red, RFID readers) are more popular than datasets of IMU data. For instance, the dataset of van Kasteren et al. [96] (published 2008) is cited 740 times, whereas the CMU multimodal Grand Challenge dataset by de la Torre et al. [201] (published 2009) is cited 130 times (according to Google Scholar as of late 2018). Indeed, Chen et al. argue that wearable sensors are 'obviously' not suitable for detecting interactions within an environment [35, p. 793]. In contrast, dense sensors and location sensing usually provide more precise and reliable observation data.

Inference method In this survey, 22 evaluations use Bayesian inference, discriminative inference is used in seven evaluations and grammar-based or logic-based approaches are used 17 evaluations. Approximate methods are used in 41% of the papers. This distribution does not represent popularity of certain approaches in general. Due to the main focus on Bayesian inference and more complex human behaviour models, this survey may be biased towards papers that use Bayesian inference.

Many approaches of discriminative classification pre-segment sensor data. For example, Bui et al. [27] cluster GPS locations and Ye et al. [216] use an adaptive segmentation based on sensor similarity. They are not used for joint recognition, mostly activities (e. g. using time-domain classifiers of accelerometers [26]) or actions (e. g. pattern mining [127]) are recognised. Discriminative approaches are also not used for inference in causal models (because there is no generative model), and no model has support for action durations. It can be seen that discriminative approaches are mostly used for motion recognition and time-series of fast-paced sensor streams such as accelerometers, but impractical for discrete labels and activity recognition where activities have varying durations [215].

Recognition Target Most papers in this survey recognise actions, followed by a combined goal/task recognition. Low-level recognition of context and activities are considered by the minority of the papers. This effect is mostly due to a selection bias of the

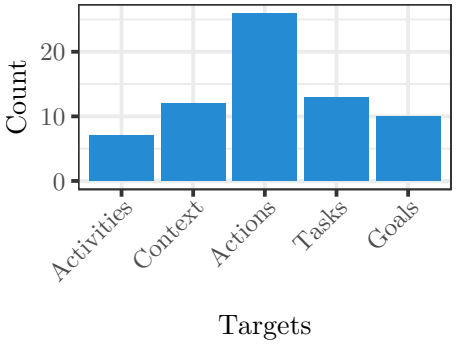


Figure 2.8: Histogram of the considered recognition targets in this survey. If a paper considered multiple tagets, these are counted individually.

papers: I am mostly interested in complex models of human behaviour, whereas low-level recognition uses mostly classification approaches without complex models. In particular, activity classification methods using machine learning and classification (e. g. recognise activities such as walking from accelerometer data using Support Vector Machines) were not considered for this survey.

Number of target classes The number of target classes is the number of different actions, goals, etc. that are the output of the inference and used for the evaluation, and thus directly influence the reported performance. The number of target classes is not always equal to the number of actions, goals, etc. defined in the model. In many cases, evaluating the complete (joint) output of the inference is not feasible simply because no ground truth is available (annotating user behaviour is very expensive and time-consuming). But this has a significant influence on the reported numbers: if, for example, the accuracy of 4 possible goals of a user are reported, any random inference method can achieve an accuracy of 25% just by chance.

The quartiles for the number of different target classes are 4.5, 7 and 11.5, respectively. The detailed histogram is shown in Fig. 2.9. This is much less than the number of all actions in the model (with quartiles of 10, 14 and 42). The data shows that this is often less than the number of actions in the model. According to these numbers, I do not consider the output of the inference used in the literature as very detailed for recognising daily situations of human behaviour.

Considering the target of the evaluation, almost all papers evaluate the accuracy of the inference wrt. the real human behaviour, mostly based on annotations. While this is very interesting and important for the application, only four papers also evaluated the runtime of their inference. This shows that algorithmic efficiency has not played an important role in human situation recognition.

Joint situation recognition Jointly recognising at least two different parts of the situation (activity, context, action or goal) is considered by twelve papers. A joint recognition of all context, actions and goals is supported by three papers. The majority of the papers (27) recognises a single component of the situation (action and goal recognition are the most common single targets done by nine and seven papers, respectively). From the eight papers which jointly recognise exactly two parts of the situation, none supports

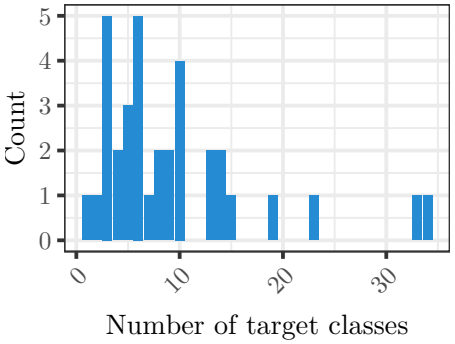


Figure 2.9: Histogram of the number of target classes used for evaluation. The number of target classes describe how many different classes have to be eventually distingiushed for the evaluation

action durations and no evaluation uses wearable sensors as sensor modality. Additionally, one work [126] does a joint recognition and at the same time supports action durations and a causal model. This work, however, uses only 1,000 discrete states.

The ratio of papers considering real-world sensor data increases with the numbers of jointly recognised targets (53% for no joint recognition, 62% for jointly recognising two targets and 100% for jointly recognising three targets). However, of all evaluations using real-world data, joint recognition is used in 35%.

Recognising goals and plans is particularly interesting for the application domains of situation recognition, e. g. in order to provide automated assistance. Goal and plan recognition approaches most often use actions as input (without a joint recognition approach), e. g. Ramírez and Geffner [164] get as input an (incomplete) sequence of observations and use a POMDP to infer the agent's goal; a similar approach was followed by Baker et al. [12]. A *joint* recognition of the goal and at least the context or actions is supported by 7 papers. Of these, the largest model has a state space of 5,000 states.

Whenever goal or context is recognised from real sensor data, it is always recognised in conjunction with actions or activities. In other words, context and goals are never recognised alone from real-world data, considering the papers of this survey. This shows that a joint recognition is used by most authors and thus an important property.

In summary, this shows that joint recognition is mostly used for the challenges of real sensor data, but the approaches need to be able to cope with larger models for wider usage.

Summary Almost all approaches in the state of the art, in particular those discussed in this survey, have at least one of the following limitations:

- They recognise only activities with user-object interactions and unambiguous sensors.
- They do not use a unifying model of context, activity and goal recognition.
- The evaluations do not recognise situations of every-day behaviour, but restrict themselves to small sub-problems.

There are two approaches which support both causal models and arbitrary action

durations [116, 126]. Therefore, I consider these two as complex models which provide a detailed modelling. Of these two approaches, no model has more than 100,000 states.

As a result, I observe that models which model aspects of real life (in particular context-based causal models and action durations) have not been evaluated under every-day conditions, which include a *joint* recognition, a sufficiently complex scenario with a large environmental state space (more than just a few thousand states) and possibly a large variety of noisy sensors.

2.3 Challenges for human situation recognition

In principle, situation recognition is estimating $P(X_i \mid y_{1:i})$. Based on the results of the previous section, I argue that recognising real-life situations as described in Section 2.1 is challenging (Thesis 2). This is due to two aspects: the *inference algorithms* that are employed and the *models* they are used with.

The contributions in this dissertation are tailored for the properties of real-life situations. In this section, we will first discuss some of the requirements of Section 1.3 on real-life situation recognition. Then, based on these requirements, we derive current challenges imposed by the human situation recognition and the models of human behaviour. This shows that the *models* are very complex (in different aspects), and that the current *inference algorithms* are not very efficient on these models. See Table 2.5 for an overview of the requirements and challenges.

2.3.1 Requirements for situation recognition

Inference algorithms applied in practice are often not used stand-alone. Instead, they are usually embedded into a larger system of control, monitoring or assistance. These systems define the required properties of the inference. As explained in Section 1.1, the predominant application for situation recognition in the literature is automatic assistance of human users. Hence, this is also the motivation for the requirements of this dissertation.

The requirements are mostly based on the requirements of Krüger [111] and presented in Section 1.3. Most of these requirements need no further analysis in this thesis. Here, I will briefly discuss the requirements of *online* inference and *symbolic* models as well as the choice of *causal* models. Some facts have not been discussed before or are important for further understanding.

Online inference Usually, assistive systems react on the recognised situation and select a suitable supporting action. For example, when the user is walking to the stage to give a talk, the smart meeting room can automatically turn on the projector. Thus in order to provide actual assistance to the user, the output of the inference must be delivered in real-time [191] to the rest of the assistance system. The real-time constraints (i. e. hard, firm or soft) depend on the application; in any case the output must be early enough so any subsequent supporting action based on that estimate is perceived by the user as helpful.

Table 2.5: Overview of requirements on and challenges of real-life inference of the *situation*.

Requirements:	Implicit challenges:	Challenges:
• recognise full situation • probabilistic inference • online inference • symbolic models • causal models	• realistic sensor data • complex human behaviour	• complex engineering process • categorical state space • large environmental state space • long-running, variable durations • numerous different situations

In particular, we require that the computation time per observation must not depend on the observation length $y_{1:i}$. For the rest of this dissertation, we will refer to this as *online* inference – this is in contrast to offline inference, where the full observation sequence is analysed after it has been recorded. Note that this is not a trivial requirement: the more observations are available, the more information can be analysed to determine the current situation.

Real-time constraints may also require that the inference must output the current estimate $P(X_i \mid y_{1:i})$ after the observation y_i is available and *before* the next observation y_{i+1} is available. That means the estimate of the situation must be output before an observation can invalidate the current situation. How fast a situation changes and how often observations occur depends on the application; ideally, observations should be made faster than situations can change. Therefore, the computation time per observation is constrained by the rate of observations.

I consider a time interval of 1 second to be sufficient. Solving simple problems and recognition tasks requires approximately a second for humans [192], *excluding* the motor skills to actually perform the *actions*. For instance, in the recordings of a kitchen domain [111], almost all actions (such as "take knife from counter") have a duration of at least 2 seconds. This means inference must be fast enough to process one observation per second, but this also means that it is not required that the inference processes more than, say, ten observations per second.

Symbolic models According to the Physical Symbol System Hypothesis by Newell and Simon [141], human reasoning involves symbolic manipulation. While it is not widely accepted that this hypothesis is true in general, it can be argued that conscious planning and reasoning about tasks (on the level of *actions*) indeed requires symbolic and rule-based manipulation [193]. A similar idea of *understanding* human thought is also present in the cognitive architecture ACT-R [5]. Rasmussen [167] divides behaviour into skill, rule-based and knowledge-based behaviour: skills are motor skills and emit

continuous signals, but cognitive task on the rule level and knowledge level make use of signs and symbols, which he describes as being discrete. Thus, cognitive *support* on a symbolical level requires also *recognition* of context, behaviour and goals on a symbolical level.

From another point of view, a *model* is a simplified description of the real world which models only the important properties. In the application of assistive systems, a change of discrete states reflects an important change in the real world, such that, for instance, assistance is required. This is one of the main tenets of qualitative reasoning [24], which aims to reason using only qualitative, non-numeric models similar to human reasoning.

Other work uses sub-symbolic models, for instance for motion modelling [26] and locomotion modelling [107]. However, all literature I am aware of that uses sub-symbolic models, does not discuss how the inference results can be used to provide actual assistance to the users.

Therefore, we require the model of human behaviour to be a symbolic model. For example, a symbolic model can be comprised of

- symbols for locations ("kitchen", "drawer", "in front of the projection screen") instead of sub-symbolic coordinates in a three-dimensional space,
- symbols for objects ("pot", "knife", "projector") and
- symbols for actions ("go to location", "take knife", "turn on projector") with symbolic preconditions and effects instead of real-valued functions, for example.

Additionally, if both the support and recognition are based on the same concepts (in particular, both are symbolic using the same set of symbols with equivalent semantics), then only a single model needs to be developed and maintained. This avoids the effort on maintaining separate models for recognition and support.

Causal models Causal models primarily restrict the sequence of possible actions to only causally correct sequences (see also Section 1.3 on page 22 for a definition of causal models). This is motivated by the fact that human behaviour and the real world obey causality. As described in Section 2.2.1, models without causality either allow any action sequence (even the impossible) or limit the set of recognisable action sequences. For example, the user cannot put an object into a cupboard if the cupboard is closed – the user has to open it first. Using causal models has also been discussed and proposed by Yordanova [217], Krüger [111] and others [49, 131, 190], and is also common in qualitative reasoning research [24].

2.3.2 Challenges imposed by the application

Studying the literature shows that *realistic sensor data* and *complex human behaviour* make situation recognition in real-life challenging.[3]

[3]This also conforms to personal experience while working on my dissertation.

Realistic sensor data Getting useful information from real-life sensor data is a challenge on its own. Depending on the sensor modality and sensor type, it can be very difficult to estimate a desired feature (such as *action*) from the raw sensor data.

Currently, sensor data is too noisy, too ambiguous or the sensors are very specific and cannot be used to monitor a variety of situations. For instance, dense sensors such as switches or reed sensors can be very reliable, but cannot practically be deployed to all objects and devices. In this survey, all approaches using such dense sensing recognise only a few (at most 30) different actions. RFID sensing with tags are more lightweight and can be attached to more objects, but are not reliable when deployed in large numbers and more intelligent designs must be followed [203]. An evaluation of Logan et al. [128] shows that RFID sensors are not very efficient, although deployed in large scale.

Thus, dense sensing does not scale to many users in all every-day locations (every object and environment must be equipped), and not every interaction or behaviour can be monitored by such sensors. On the other hand, wearable sensors such as accelerometers can observe every behaviour of a human, but are very noisy and ambiguous. While dense-sensing approaches very often rely on the sensors to directly indicate an action (e. g. an open/close sensor on a cupboard or drawer corresponding to the correct "open" action, as for half of these works [73, 87, 92, 156, 169, 176]), such a mapping is not available to wearable sensors in any of the papers studied. Recognising activities such as walking or sleeping is feasible, but recognising detailed interactions is very challenging [35, p. 793]. Similarly, tracking the environment with video-based sensing is challenging and has its own deficiencies (requires line of sight and sufficient lighting, computationally intensive and raises privacy concerns).

Real-life sensor data exhibits usually all of the following properties:

indirect Observations are not in a one-to-one relationship to a desired quantity. Instead, the quantity to estimate indirectly influences the measurements, and the measurements are influenced by a number of different factors. For example, the *action* "walk to office" may consist of a number of *activities* (walk, walk stairs) and each activity results in motions. Accelerometers worn by the user measure the acceleration due to those motions, but may also be influenced by other conditions (e. g. motions not related to the locomotion, such as sneezing or greeting a colleague).

partial Only a subset of observations is available to the system, that is, not the complete state of the world is measurable. The set of available sensors is limited by the application (e. g. if body-worn sensors are permissible), financial limits, space limits, computing resources and other constraints.

noisy Due to inherent measurement errors in the sensor hardware, the observations are not exact but can contain noise. Depending on the sensor hardware, the noise might be negligible (e. g. reliable simple button switches) or severe (e. g. microelectromechanical magnetometers).

missing As a special form of noise, some sensor modalities can also suffer from missing observations. Depending on the sensor communication infrastructure, the network may also lose measurements.

Although a possible attacker might also tamper with the data, we do not consider security issues here.

These properties apply in particular to wearable data, but, as can be seen from the literature, the properties also apply in parts to dense sensing. Wearable sensors have been used in four papers of this review, most do not use any real sensors. Thus, we can conclude that realistic sensor data has not been dealt with in everyday situation recognition.

In summary, the sensor data quality leads to uninformative (uncertain data) or biased (inaccurate data) observation models $P(Y \mid x)$. This leads to a negative impact on the inference accuracy and efficiency. Remember, sequential state estimation may be simplified to the Bayes formula

$$\overbrace{p(x \mid y)}^{\text{inferred situation}} \propto \overbrace{p(y \mid x)}^{\text{observation likelihood}} \cdot \overbrace{p(x)}^{\text{prior knowledge}},$$

which states that the inferred probability of the situation can be computed by the observed sensor data and prior knowledge about the situation. Thus, poor sensor data quality can be compensated by prior knowledge about the human behaviour. As will be shown now, however, human behaviour (as part of a situation) is very complex, thus there is usually also very limited, uncertain prior knowledge.

Complex human behaviour The nature of human behaviour in general makes modelling and inference challenging. Kim et al. [103] lists four uncertainty sources in human activities, which are *concurrent*, *interleaved* and *ambiguous* activities as well as *multiple subjects*. I regard multiple subjects as a special case of concurrent behaviour and re-classify these sources as:

non-linear Multiple, possibly infinitely many, different paths of execution can lead to the same results. In combination with dynamic, open worlds, this makes future actions hard to predict.

non-deterministic behaviour Not only permits the real-world many different paths of execution, but users also tend to non-deterministically select one option or the other. Although they may select actions based on personal preferences, there is usually some variability within the behaviour of a single user. The variability between different users is usually even larger.

However, these properties have been dealt with in many studies and are also well represented in this literature survey. Instead, I found that all applications and datasets in the literature restrict the behaviour in several other aspects:

open world The user can execute actions of his own free will, with only limited restrictions (e. g. physical laws and causality). Constraints imposed by the user's task or by social norms may even not be applicable, in particular when recognising such improper or false behaviour is desired.

rich dynamics There are large numbers of possible actions, and the actions' effects can alter the environment in numerous ways. This leads to a high number of possible situations, making it necessary to track a large number of concurrent possible situations.

non-deterministic durations As another instance of non-deterministic behaviour, the execution speed of actions can show a large amount of variability. In our datasets, the shortest actions are a few seconds, while the longest are several minutes. One may also imagine behaviour such as sleeping, which can be modelled to last several hours.

In this survey, no approach did feature an open-world recognition, but is limited to a very special use-case. No approach allows real rich dynamics with more than 100 actions and at the same time more than 100,000 states. As analysed in Section 2.2.3, only six papers support probabilistic durations. In particular in combination with long-running recognition, non-deterministic durations are challenging.

Due to the open world, rich dynamics and long-running recognition, the distribution $P(X_i)$ has a large support. Additionally, the transition model $P(X_i \mid x_{i-1})$ is very flat due to rich dynamics and non-linear behaviour with a large variability in the duration of behaviour. Thus, Bayesian inference itself is challenging, following the same argumentation as the last section on realistic sensor data.

2.3.3 Challenges imposed by the human behaviour models

Realistic sensor data and complex human behaviour are challenging for *all* situation recognition approaches, and are not in the focus of this dissertation. However, these challenges have an effect on inference in causal, state-space based Bayesian *inference* as handled in this dissertation. In particular, these are the challenges of *modelling*, *categorical* state spaces and *large* state spaces and belief states.

In the following, more theses will be developed and formulated. These theses all support Thesis 2 and address individual challenges in more detail. They are the main motivation for the technical contributions in later chapters.

Modelling human behaviour Modelling the behaviour of the users is the first prerequisite to Bayesian inference. Due to the complexity and variety of human behaviour, building appropriate models of human behaviour is likewise complex and difficult.

Thesis 3. *Engineering useful and correct models of human behaviour is challenging and error-prone. Human-built models of human behaviour contain errors that influence the recognition. Many errors can be found by checking for a few classes of errors.*

Except for Hoey et al. [87], no work in this survey describes the process for building suitable models. The need for a systematic model development process has been described by Yordanova [217, 218]. As the model development is difficult, and model engineers tend to optimise the model to increase accuracy and/or inference speed, semantic errors are likely. This thesis is discussed in more detail in Chapter 4.

Large state space S One predominant factor in the probabilistic model that causes inefficient inference are the large sets of context states **S**.

Thesis 4. *Everyday human behaviour entails a large number of context states* **S**. *Handling the large set of context states* **S** *is one factor that limits efficiency of Sequential Monte Carlo (SMC) methods. Automatically reducing the set of relevant context states increases the efficiency of SMC.*

The large state space **S** is mainly caused by the open world properties of real-life in conjunction with the rich dynamics of human behaviour (see challenge on complex human behaviour on page 57). For example, the *Kitchen* model used in this dissertation (presented in Section 3.4.4) has approximately 146 million context states **S**. Trying to build a model of the activities in the *CMU* dataset (presented in Section 3.4.5) results in a state space of over 460 trillion states[4]. Despite these large numbers, the literature features comparatively smaller state space sizes. As analysed in Section 2.2.3, the largest state space has 250,000 states. The second largest state space (200,000 states) is also a single-task kitchen domain, but has much less than 146 million states.

Therefore, the numerous different context states are the key factor in the large number of situations.

Categorical state space Remember from Section 1.3 and Section 1.2 that the model describes the situation using a structured state space **X**. Due to the symbolic and causal models, the state space **X** is mostly categorical (and thus discrete).

Exact inference methods like variable elimination in Bayesian Networks can cope equally well with categorical, discrete and continuous state spaces, but have a prohibitive time complexity (linear in the number of states). Approximate inference algorithms like Generalised Belief Propagation can also handle categorical state spaces. However, these algorithms are too inefficient for dealing with the large state spaces of every-day human behaviour models, as they are approximate in the independence structure of the Bayesian Network, not in the number of states.

Sequential Monte Carlo (SMC) methods are a class of approximate inference algorithms the complexity of which is independent of the number of states. SMC methods draw samples from the target distribution and are thus more efficient at approximating $P(X_i \mid y_{1:i})$. However, SMC methods have been originally developed and analysed for continuous state spaces. Many improvements that make SMC applicable to real-world applications (e. g. computer vision) are only applicable to continuous state spaces. Only Shi et al. [190] of this survey provide an approach to handle categorical state spaces. Previous work [113] has also recognised this as a challenge for efficient inference.

Thesis 5. *The traditional particle filter, as the most popular instance of SMC, is inefficient for inference in categorical state spaces. An efficient management of categorical samples increases the quality of the estimate.*

[4]The state space of the CMU dataset could not be expanded completely using state-of-the-art model checkers with 128 GB of RAM, it may possibly be even much larger.

What is missing is a deeper understanding and evaluation of the reasons for the inefficiencies, as well as a thorough and uniform algorithmic solution. A detailed analysis and discussion of this topic is given in Chapter 5.

Durations Besides the large sets of context states, the number of possible starting times $\mathbf{T}$ is another important factor that makes inference inefficient.

Thesis 6. *Human behaviour has a large variability in the duration of behaviour. The results in a large set of possible starting times* $\mathbf{T}$*. Handling the large set of starting times* $\mathbf{T}$ *is one factor that limits efficiency of SMC methods.*

Although human behaviour has different durations, only few approaches support modelling duration distributions. As analysed in Section 2.2.3, these models are small and model simple settings. Of the 20 papers that use Bayesian filtering, four of them use SMC. The state space sizes are similar, however the works using SMC methods feature more consistently action durations. This is an indicator that inference using duration models is computationally more expensive and requires approximate methods.

The large set $\mathbf{T}$ is caused by the large variability of the durations in combination with the usually continuous monitoring. It must be noted here, although the transition model $P(S_i \mid s_{i-1})$ of the context states can be assumed to be Markovian, the duration models can be arbitrary. Thus, the model of human behaviour is semi-Markov, and more efficient inference algorithms (for instance using Hidden Markov Models), which have an inference complexity independent of $|\mathbf{T}|$, cannot be used. As a consequence, $|\mathbf{T}|$ is also a factor in the number of situations and thus contributes to the inference complexity. For the datasets we consider here, $|\mathbf{T}|$ is a few thousand timesteps, but can be arbitrarily large in general.

Inference in a large state space X As a consequence of the large set of context states $\mathbf{S}$ and probabilistic action durations, the overall set $\mathbf{X}$ of situations is very large and high dimensional (usually > 100 dimensions, i. e. state variables). Due to the large state space and noisy observation (see page 56), the distribution over states $P(X \mid y)$ usually has a large support. Managing a large number of high-dimensional situations requires more computation time to bound the variance of the estimate. This is necessarily true for exact inference algorithms (e. g. variable elimination for Bayesian Networks), the complexity of which also depends on the size of the discrete variable domains (i. e. $\mathbf{X}$). But this is also true for SMC methods – this will be discussed in Section 3.2.2.

Thesis 7. *For Sequential Monte Carlo inference of the situation using human behaviour models, the number of particles required to bound the variance to a certain level increases with the number and dimensions of situations. Reducing the belief state is important for efficient inference.*

Thus, efficient (in the sense of Definition 2, page 24) means to handle the large state-spaces are required.

2.4 Summary

Human situation recognition in real-life applications can be considered a challenging task. As outlined in Thesis 4 and Thesis 6, the state space $\mathbf{X}$ is very large. In conjunction with Thesis 7, SMC methods require a large number of particles to approximate the distribution $P(X_i \mid y_{1:i})$. Thus, even approximate inference requires many computing resources.

The inference of human behaviour in the real world is further challenging because of the uninformative observation models and rich system models (Section 2.3.2). In addition, building suitable models of human behaviour is error-prone (Thesis 3). Each report in the literature of recognising human situation lacks one of the requirements (Section 1.3), has a very limited application scope or makes other strong assumptions. In conclusion, Thesis 2 can be considered to be true.

In the surveyed literature, there have been selective improvements for SMC tailored for situation recognition (e. g. an adaptation for categorical states [190]). No approach developed techniques with a thorough analysis of all challenges I identified. Given that Sequential Monte Carlo methods suffer from the complexity inherent to models of situation recognition (Thesis 7) but are still one of the most popular and efficient methods (Chapter 2), it is important and reasonable to improve their efficiency.

3 Human behaviour modelling and inference

Summary: This chapter introduces the human behaviour model used in this dissertation and reviews Bayesian inference in general as well as the particle filter algorithm.

Before we can discuss the individual algorithmic contributions in detail, we need to review the background of human behaviour modelling as well as the basic inference methods. Recall from the introduction that we use a probabilistic model of human behaviour to facilitate Bayesian inference. This chapter first presents this probabilistic modelling in detail in Section 3.1. The inference is then presented in Section 3.2; the algorithmic contributions of this dissertation are based on the principles discussed there.

The algorithms are integrated into an existing implementation of state estimation, named the CCBM toolkit. The CCBM toolkit is a collection of tools for inference in human behaviour models based on real sensor data; it is briefly described in Section 3.3. A part of the CCBM toolkit is a modelling language that is used to describe the probabilistic model in a more accessible notation. Understanding the language is important for understanding the challenges and difficulties in developing these models, which lead to the modelling errors discussed in Chapter 4. This language is also briefly summarized in Section 3.3.

Finally, Section 3.4 presents the different models and datasets that are used for validation in this dissertation. Some of these models are much more complex in several aspects than the models used in previous literature:

State space The number of states exceeds several billion, compared to at most 250,000.

Actions We use models with nearly 400 actions for a single user. The few models in the literature with more actions have a considerably smaller state space.

Action durations Only few evaluations use a model of the duration of actions, again with more limitations.

Target classes For computing the approximation error, we take all actions into account. Other evaluations in the surveyed literature use much less actions, or cluster actions and compare only action classes.

3.1 Human behaviour models

Human situation recognition refers to determining the current situation x_i based on a sequence of observations $y_{1:i}$ which are generated by sensors deployed in the environment. In a probabilistic setting, human situation recognition means to determine the joint filtering distribution $P(X_{1:i} \mid y_{1:i})$.

Modelling principle One approach is to use *discriminative* models, for which one has to define (or learn) the distribution $P(X_{1:i} \mid Y_{1:i})$ directly. In this thesis, we adhere to the principle of *generative* modelling of human behaviour. In a generative model, one defines (or learns) the joint probability distribution $P(X_{1:i}, Y_{1:i})$.

For the rest of this work, we always characterise distributions $P(A)$ and conditional distributions $P(A \mid b)$ by their *densities* $p(A)$ and $p(A \mid b)$.

The advantages and disadvantages of generative and discriminative models shall not be dealt with here, they have been discussed numerous times in general [21, 142] [108, p. 709] and specifically for the application in human situation recognition [35, 94, 95, 111, 217]. One main argument in favour of generative models is: for every-day human behaviour, there is not enough data to have examples for every possible sequence $x_{1:i}$ to robustly learn or define $p(X_{1:i} \mid y_{1:i})$ (see also Section 2.3.2). However, it is possible to define a joint distribution by using *prior knowledge* of human behaviour and a suitable factorisation. In this dissertation, we use the modelling approach of Computational State Space Models (CSSMs) described by Krüger [111, 113]. This approach has been identified by Krüger to satisfy the requirements of Section 1.3; for details and more design rationales of the model, I refer to his dissertation [111].

Dynamic Bayesian Network interpretation and Markov property In the following, we discuss properties of the model and derive the probabilistic definitions. On a high-level view, the model of human behaviour can be represented by the DBN in Fig. 3.1.

In the context of situation recognition, one usually describes the situations X as a Markov process, i. e. the situation X_i at timestep i depends only on the previous situation X_{i-1} (this is called the Markov property). This can be justified by the fact that a *situation* describes everything important to characterise the circumstances of the user (as described in Section 2.1). The Markov property is also desired from the perspective of inference, because it simplifies inference algorithms and reduces their time and space complexity.[1] To aid the definition of the density, the model uses two additional sets of auxiliary random variables $F_{1:i}$ and $\mathcal{T}_{1:i}$. Both are related to modelling the durations of human behaviour and will be explained below.

[1] It simplifies inference algorithms, because only two situations X_{i-1} and X_i must be managed (constant space complexity) – otherwise, the full sequence of past sequences $X_{1:i}$ must be managed, or strategies for determining which previous situations $X_j (j < i)$ are not required any more must be developed (space linear in the number of timesteps). Inference in Bayesian Networks is exponential in the tree-width of the network [119]. Due to the Markov property, the tree-width is constant in the number of timesteps – otherwise, it would be linear.

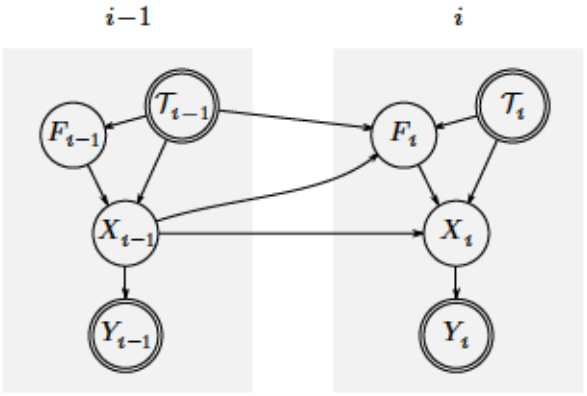


Figure 3.1: High-level DBN of the model structure for human behaviour. X is the current situation, i. e. state, Y is the observation generated by the current situation, F denotes if a new action starts and $\mathcal{T}$ is the current real-world time. Observable random variables are represented by double-circled nodes.

Simplifying assumptions Before diving deep into the definitions of the distributions and deriving the filtering algorithms, we first discuss some simplifying assumptions.

We assume deterministic effects, that is given a situation x_i, every applicable action a unambiguously defines the next situation x_{i+1}. However, all results can be extended to non-deterministic effects as well: as we are using a generative approach, for every non-deterministic action, we can define multiple synthetic deterministic actions.

We neither model *motions* nor *activities* as behaviour, but only *actions*. This dissertation is focused on the categorical, symbolic state space induced by the causal actions and symbolic context states. Extending the model and inference methods to also include continuous, sub-symbolic actions and motions is possible, but out of the scope of this dissertation. There is also already work on efficient inference for continuous, analytically tractable sub-models, using Rao-Blackwellised particle filters [59]. We also do not discuss and model *tasks*. The level of abstraction (distinguishing between what is considered a basic action and what a task) is eventually a design decision of the model engineer.

Joint density definition The DBN in Fig. 3.1 defines independence assumptions between the different model components. This DBN is based on the previous work of Krüger [111, Sect. 3.1], where he explains the rationale of this structure. Due to the Markov property and independence assumptions between the variables, the joint density can be defined as

$$\begin{aligned} p(x_{1:i}, y_{1:i}, f_{1:i}, \mathcal{T}_{1:i}) = {} & p(x_1, f_1, \mathcal{T}_1)\, p(y_1 \mid x_1) \\ & \cdot \prod_{i=2} \big(p(x_{i-1})\, p(\mathcal{T}_{i-1})\, p(\mathcal{T}_i) \\ & \quad \cdot p(f_i \mid \mathcal{T}_{i-1}, \mathcal{T}_i, x_{i-1})\, p(x_i \mid x_{i-1}, f_i, \mathcal{T}_i)\, p(y_i \mid x_i) \big), \end{aligned} \tag{3.1}$$

where $p(x_1, f_1, \mathcal{T}_1)$ is the density of the initial situations, $p(x_i \mid x_{i-1}, f_i, \mathcal{T}_i)$ the density of transitions between situations and $p(y_i \mid x_i)$ the observation model.

The densities $p(f_i \mid \mathcal{T}_{i-1}, \mathcal{T}_i)$ and $p(x_i \mid x_{i-1}, f_i, \mathcal{T}_i)$ are defined in terms of densities that are easier to specify for the model developer. For this purpose, every situation's

random variable X_i is factored into random variables S_i, A_i, T_i, I_i, G_i. The dependencies between all random variables are depicted in the DBN in Fig. 3.2 and will be explained below.

3.1.1 Model components

The DBN consists of the following random variables at timestep i:

- $\mathcal{T}_i$ – the real-world time
- X_i – the situation
- S_i – the *context* state
- A_i – the user's *action*
- T_i – the starting time of A_i
- F_i – auxiliary variable, denoting if a new action starts
- G_i – the *goal* of the user
- I_i – the initial context state
- Y_i – the observation

The user's *behaviour* is modelled to have a *duration*, i. e. executing this behaviour requires time. The "wall-clock" time observed in the real world for any particular timestep i is $\mathcal{T}_i$. Thus, $\mathcal{T}_i - \mathcal{T}_{i-1}$ is the time that passed between the two situations at timesteps $i-1$ and i. This is important for evaluating the duration of the action's execution.

The random variables S_i are the *context* state of the environment (see Section 2.1 for a definition). According to Section 1.3, the set of context states $\mathbf{S}$ is mostly symbolic and thus categorical – continuous sub-states are possible, though, as they might be important for certain applications. For fine-grained models and complex application domains, $\mathbf{S}$ may be very large or even infinite. To distinguish between a situation's state $x = (s, a, t, i, g)$ and a context's state s, we often use the terms "X state" and "S state", respectively.

Usually, the S state is itself factored into multiple variables, each denoting a specific property of the environment (e. g. location of the user, power status of a lamp). The factorisation of $\mathbf{S}$ is model-dependent and a choice of the model engineer. Due to the large number of different environment variables that influence the user's behaviour in every-day life, $\mathbf{S}$ has usually many dimensions (the models used in this dissertation have up to 130 dimensions).

The *actions* of the user for every timestep i are modelled in the random variables A_i. The action's *effects* are defined by the transition density $p(s_i \mid a_i, s_{i-1})$, its *precondition* is modelled as a predicate $pre_a(s)$ that states if a is applicable in s. As we assume deterministic effects, $a_i(s_{i_1}) = s_i$ is the transition function, and $p(s_i \mid a_i, s_{i-1}) = \mathit{Dirac}(s_i = a_i(s_{i-1}))$ is the Dirac distribution at $a_i(s_{i-1})$.

To model the duration of action executions, every action a has an associated *duration model* defined by the density $p(D_a)$, where $d_a > 0$ is the duration of the action. For example, the duration of the action "walk" might be distributed according to a log-normal distribution $d_a \sim \ln\mathcal{N}(2, 0.5)$. The random variable T tracks the *starting time* of the

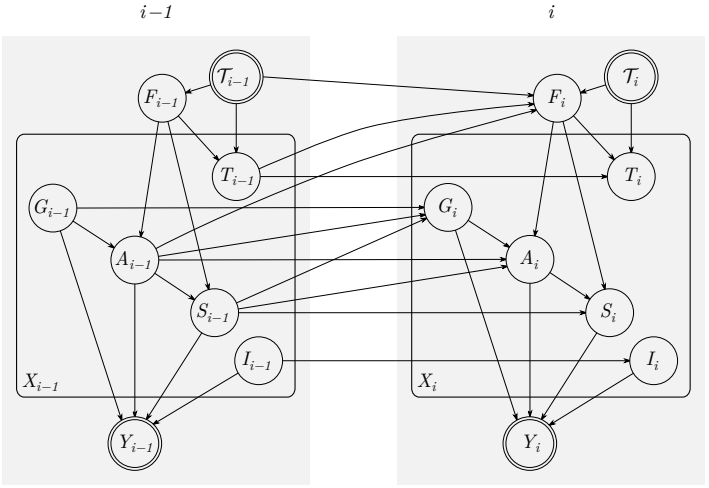


Figure 3.2: Complete Dynamic Bayesian Network of human behaviour models that is used in this work. The random variable X is factored into five other random variables.

action, thus for every new action, only one starting time has non-zero support in the filtering density. How long an action is currently running can be computed by $\mathcal{T}_i - T_i$, i. e. the current time subtracted by the time the action started. The duration's random variable D is not explicitly used in the DBN, but its distribution $P(D_a)$ is used in computing the auxiliary variable F_i. The boolean-valued random variable F_i determines if the action has finished executing, i. e. if the action's duration D_i is equal to its current running time $\mathcal{T}_i - T_i$.

As we assume *goal*-directed behaviour, the current goal G_i of the user is also part of the model. A goal g is modelled as a predicate on the state s, if $g(s)$ is true, the current goal is assumed to be achieved. During the course of actions, the user's goal may also change (either if the goal has been achieved, or due to other context changes).

Depending on the application, the initial situation of the user may be unknown. In these cases, the initial context state is modelled as another random variable I_i. In contrast to all other random variables, its value does not change during transition, but is constant for every timestep. The model engineer only defines the initial density $p(S_1)$, which is identical to $p(I_1)$. (Defining an extra random variable I_i makes it more convenient to forward-filter and reason about the distribution of initial states at a specific timestep I_i, instead of writing S_{1i}.)

Finally, Y is the current set of observations. The domain of Y is dependent on the application and determined by the available sensors. However, the observations are usually factored, i. e. there are different observation variables due to state (e. g. location), due to action (e. g. motions of the user) or goal (e. g. by querying user's calender).

For the sake of brevity, we did only describe the case of a single user operating in the environment. All these definitions can be easily extend to multiple users [111, p. 37]. For instance, A can be factored into actions $A^{(u)}$ for every user u, as can be F and G.

3.1.2 Factors and transition model

The transitions between different timesteps are modelled by $p(f_i \mid \mathcal{T}_{i-1}, \mathcal{T}_i, x_{i-1})$ and $p(x_i \mid x_{i-1}, f_i, \mathcal{T}_i)$ according to (3.1). Due to independence assumptions between the different variables within X, these transitions can be modelled by simpler densities:

$$p(f_i \mid \mathcal{T}_{i-1}, \mathcal{T}_i, x_{i-1}) = p(f_i \mid a_{i-1}, \mathcal{T}_{i-1}, \mathcal{T}_i, t_{i-1}) \tag{3.2}$$

$$\begin{aligned} p(x_i \mid x_{i-1}, f_i, \mathcal{T}_i) = {} & p(g_i \mid g_{i-1}, s_{i-1}, a_{i-1})\, p(a_i \mid f_i, s_{i-1}, a_{i-1}, g_i) \\ & p(s_i \mid f_i, s_{i-1}, a_i)\, p(t_i \mid f_i, t_{i-1}, \mathcal{T}_i)\, p(i_i \mid i_{i-1}) \end{aligned} \tag{3.3}$$

The underlying intuition can be explained by the operational semantics of the behaviour model: For every timestep, first the goal g_i is updated (if required, the goal may persist) based on the past situation. Then, based on the action's duration, it is determined if the action's execution has finished or continues (variable f_i). Then, a new action a_i is selected (if the previous action finished), based on the new goal and the past situation. Then, the new context state s_i is determined (if required) based on the new action. Finally, the action's starting time t_i is updated (if required). The initial state i_i is simply copied from the previous timestep.

The following densities are required to specify the full model:

- $p(F_i \mid a_{i-1}, \mathcal{T}_{i-1}, \mathcal{T}_i, t_{i-1})$ – the termination model, determines if the current action stops based on the duration model
- $p(A_i \mid f_i, s_{i-1}, a_{i-1}, g_i)$ – the action selection model, determines what action is executed
- $p(S_i \mid f_i, s_{i-1}, a_i)$ – the context state transition model
- $p(T_i \mid f_i, t_{i-1}, \mathcal{T}_i)$ – the update of the action's starting time
- $p^*(G_i \mid g_{i-1}, s_{i-1}, a_{i-1})$ – update of the user's goal
- $p^*(Y_i \mid s_i, a_i, g_i, i_i)$ – the observation model

Some of these densities are set by the underlying semantics of the behaviour model and will be defined below. Others must be defined by the model engineer and are marked as p^* in this section.

The boolean-valued random variable F_i determines if the action has finished executing, i. e. if the actions duration D_i is equal to its current running time $\mathcal{T}_i - T_i$. Because the model only considers discrete timesteps, this is approximated by testing if $T_i + D_i \in (\mathcal{T}_{i-1}, \mathcal{T}_i]$, i. e. the action has stopped within the half-open interval of the last two time steps. Accordingly, F_i in the DBN in Fig. 3.2 (i. e. the *probability* that an action a_{i-1} has stopped at timestep i) depends on the action of the previous timestep, the action's starting time, and the current and last real time, and is computed by

$$p(F_i = \mathit{true} \mid a_{i-1}, \mathcal{T}_{i-1}, \mathcal{T}_i, t_{i-1}) = p^*(D_{a_{i-1},i} \leq \mathcal{T}_i - t_i \mid D_{a_{i-1},i} > \mathcal{T}_{i-1} - t_i) \tag{3.4}$$

More details on computing F_i are described by Krüger [111, Sect. 3.1.2].

If the action has stopped (i. e. $F_i = \mathit{true}$), a new action A_i has to be executed by the user at timestep i. (We always assume the user is executing some action; if required a no-op "idle" action might be added.) As we assume *goal*-directed behaviour, the action selection also depends on the current goal G of the user. Thus,

$$p(a_i \mid F_i = \mathit{true}, s_{i-1}, g_i) = p(a_i \mid s_{i-1})\ \gamma(a_i, s_{i-1}, g_i), \tag{3.5}$$

where $p(a_i \mid s_{i-1})$ selects all applicable actions:

$$p(a_i \mid s_{i-1}) = \begin{cases} 0 & \text{if } \mathit{pre}^*_{a_i}(s_{i-1}) = \mathit{false} \\ 1/Z & \text{otherwise} \end{cases} \tag{3.6}$$

(where Z is a normalisation constant over all actions) and γ implements goal-directed behaviour, i. e. actions that work towards achieving the goal get a higher probability. For more details on computing γ see the dissertation of Krüger [111].

If a new action has been selected, the new state is computed by applying the action's effects, i. e.

$$p(s_i \mid F_i = \mathit{true}, s_{i-1}, a_i) = \mathit{Dirac}(s_i = a_i^*(s_{i-1})), \tag{3.7}$$

and the starting time of the new action is updated accordingly:

$$p(t_i \mid F_i = \mathit{true}, \mathcal{T}_i) = \mathit{Dirac}(t_i = \mathcal{T}_i). \tag{3.8}$$

If the current action continues ($F_i = \mathit{false}$), then A_i, S_i and T_i keep their old values from the previous timestep $i-1$. As a consequence, the effects of a new action are applied immediately to the S state, and the S state does not change during the execution of the action or at its end. This reflects the property that actions are atomic. Thus,

$$p(a_i \mid F_i = \mathit{false}, a_{i-1}) = \mathit{Dirac}(a_i = a_{i-1}) \tag{3.9}$$

$$p(s_i \mid F_i = \mathit{false}, s_{i-1}) = \mathit{Dirac}(s_i = s_{i-1}) \tag{3.10}$$

$$p(t_i \mid F_i = \mathit{false}, t_{i-1}) = \mathit{Dirac}(t_i = t_{i-1}). \tag{3.11}$$

Updating the user's goal can be modelled by defining $p^*(G_i \mid g_{i-1}, s_{i-1}, a_{i-1})$. However, for the sake of simplicity, we assume that goals do not change over time (but there may be multiple initial hypotheses about the user's goal).

The observation model $p^*(Y_i \mid s_i, a_i, g_i, i_i)$ must be defined by the model engineer and adapted to the sensor infrastructure. For the sake of brevity, we usually write $p(Y_i \mid x_i)$. However, we note that the observations do not depend on the action's starting time. This is an important assumption and simplification of the model. The underlying rationale is that actions are atomic in their effect of the context state, and that the action's effect is applied at the start of the execution. For the rest of the execution, the action is assumed to be constant. If two or more parts of an action with individual characteristic observations can be identified, then the action may as well be split into multiple actions.

Finally, the initial density $p(X_1)$ is defined based on the user-defined

- $p^*(G_1)$ – initial distribution of goals
- $p^*(S_1)$ – distribution of initial context state

by

$$p(x_1) = p(g_1)\, p(s_1)\, Dirac(a_1 = a_{init})\, Dirac(t_1 = \mathcal{T}_1)\, Dirac(f_i = true). \tag{3.12}$$

The action a_{init} is a synthetic action that is only valid in the initial state and has a duration of exactly one timestep (the first timestep can, of course, be modelled to take no real duration). This solves the problem that the initial action cannot depend on a previous state, the initial action has no preconditions and no effects.

3.2 Inference in human behaviour models

In the previous section, we described the human behaviour model and how it evolves. In fact, we described a *predictive* model that can generate new situation sequences by using the transition model.

However, the predictive model itself provides no means to *estimate* the current situation X_i based on sensor data $y_{1:i}$. That is, inference is the process of computing the *filtering density* $p(X_i \mid y_{1:i})$ of the current state given all past observations.

In this section, we first review the general equations for Bayesian inference in the human behaviour model described in Section 3.1. We give arguments why exact inference is intractable. We then review the SMC framework in Section 3.2.2 and explain why SMC algorithms suffer from high-dimensional state spaces.

This section discusses Bayesian inference on a high level of abstraction, independent of the DBN structure. In this section, we use X_i to denote the *complete* joint state of timestep i. Formally, it would be better to use a different name, such as Z_i (for Fig. 3.1, $Z_i = \langle \mathcal{T}, F, X, Y \rangle$). However, I believe introducing yet another name will cause more confusion than clarity.

3.2.1 Bayesian filter equations

According to Bayes' theorem, the filtering density can be computed by

$$\overbrace{p(X_i \mid y_{1:i})}^{\text{posterior}} \propto \overbrace{p(y_i \mid X_i)}^{\text{obs. likelihood}}\ \overbrace{p(X_i \mid y_{1:i-1})}^{\text{prior knowledge}}$$

with normalising the density to 1. That is, the density of a current state x_i can be computed by the observation likelihood multiplied by the *prediction* density $p(X_i \mid y_{1:i-1})$. The prediction density computes the densities of the states x_i that would be expected according to the system's transition model $p(X_i \mid x_{i-1})$.

Due to the Markov property, the prediction can be computed by marginalising over the possible previous states x_{i-1}:

$$p(X_i \mid y_{1:i-1}) = \int_{x_{i-1} \in \mathbf{X}} P(X_i \mid x_{i-1})\, p(x_{i-1} \mid y_{1:i-1})\, dx_{i-1}$$

That is, given the filtering density $p(X_{i-1} \mid y_{1:i-1})$ of the previous timestep, apply the transition model to every previous state, and sum the densities. As the model does not

exclude continuous states, we use an integral here; for discrete states, sums will be used accordingly.

Thus, the filtering density can be computed by the recursive Bayesian formula, which states that Bayesian filtering works by recursively predicting the next possible states, and correcting the prediction with the observations:

$$p(X_i \mid y_{1:i}) \propto p(y_i \mid X_i) \int_{x_{i-1} \in \mathbf{X}} P(X_i \mid x_{i-1})\, p(x_{i-1} \mid y_{1:i-1})\, dx_{i-1} \tag{3.13}$$

For an in-depth treatment of inference in Dynamic Bayesian Networks I refer to the comprehensive dissertation of Murphy [138].

Belief state In this dissertation, we will often refer to the *belief state*, which is another name for the filtering density $p(X_i \mid y_{1:i})$ at timestep i. It is called so, because this distribution represents the belief of the inference about the current state of the world; the belief is the central internal state that is managed by inference algorithms. The term belief state is particularly used in the context of control software and assistance systems [88, 174], where the system has to act according to its belief $p(X_i \mid y_{1:i})$ of the current state of the world.

The distinction between *belief state* and *filtering density* is important in cases where the inference algorithm approximates the filtering density. In this case, the *belief state* denotes the approximation of the *filtering density*, because this approximation is then the internal state of the inference.

Complexity Exact Bayesian inference in the DBN is intractable for our problem sizes (cf. Chapter 2). Although Bayesian inference is in general NP-hard [44] (even computing bounded approximations is NP-hard [48]) and #P-complete [175], it is *not* the exponential complexity that makes efficient computations infeasible. Almost all exact inference algorithms (such as variable elimination [108, Chapter 9]) are exponential in the treewidth of the Bayesian network [138, Section 3.5]. However, the treewidth for the DBN in Fig. 3.2 is 4, thus constant over all models and low. Inference (forward filtering) using variable elimination must first eliminate all variables from timestep i. By using variable elimination order I, G, F, T, A, S, we can show that the complexity of inference is at most

$$\mathcal{O}(|\mathbf{G}|^2\,|\mathbf{A}|\,|\mathbf{S}| + |\mathbf{G}|\,|\mathbf{A}|\,|\mathbf{S}|\,|\mathbf{T}|^2 + |\mathbf{G}|\,|\mathbf{A}|^2\,|\mathbf{S}|\,|\mathbf{T}| + |\mathbf{G}|\,|\mathbf{A}|\,|\mathbf{S}|^2\,|\mathbf{T}|).$$

In our applications, the largest component of $\mathbf{X}$ are the context states $\mathbf{S}$ (several thousands to billions) and the possible starting times $\mathbf{T}$ (one for every timestep, in our datasets a few thousands). Although inference complexity is not exponential in the size of the value domain $\mathbf{X}$, it is still quadratic in $|\mathbf{S}|$, $|\mathbf{A}|$ and $|\mathbf{T}|$; given the huge state spaces of our application domains this prevents efficient online inference.

Inference algorithms For some classes of models for stochastic processes, closed-form solutions with linear complexity exist; for instance, estimation of linear models with

Gaussian noise can be done using the Kalman filter. One example application is localisation using GPS, where normality of the errors is usually assumed. Another popular inference algorithm for inference in dynamic systems is the forward algorithm for Hidden Markov Models (HMMs) [161]. The HMMs operates in time quadratic in the number of states, i. e. the number of states X, which is very large for our models. This can be alleviated by interpreting the model as a Hidden semi-Markov Model, which is linear in the number of timesteps, but still quadratic in the number of context states S [161].

However, there is no known closed-form solution for our DBN structure that permits inference in linear time. Even linear-time would be too slow for models with trillions of states, and impossible for infinite state spaces. Therefore, approximate algorithms are required. Due to the complex model-structure and generative modelling approach, a sampling based approach seems to be promising. Therefore, this thesis focuses on inference using Sequential Monte Carlo methods. Sequential Monte Carlo methods also, to some degree, relate to a beam sampling approach proposed for infinite HMMs [204] and have also been used for inference in HMMs [68].

3.2.2 Sequential Monte Carlo

Sequential Monte Carlo (SMC) methods provide an approximate inference for arbitrary models. I recommend the introductions to SMC methods by Arulampalam et al. [7] and Doucet [58], the latter also has a good overview on the history of SMC methods. In this section, we briefly review the principles of SMC and give an algorithm for the DBN described in Section 3.1.

SMC is not a single algorithm, but a family of algorithms that are based on the same framework and underlying idea. Thus, the details of the operation are specific to individual implementations. We first describe the general framework of SMC methods, which is independent of any model structure as long as it can be described by a DBN with states X_i and observations Y_i. We then present a concrete implementation – the particle filter – adapted to our DBN of Fig. 3.2.

Basic framework Sequential Monte Carlo methods are a straightforward sampling-based implementation of the Bayesian filter equations (3.13). SMC methods use multiple samples $x_i^{(1)}, x_i^{(2)}, \dots x_i^{(N)}$ to approximate the filtering density $p(x_i \mid y_{1:i})$. For timestep i, the *belief state* is a set $\mathbf{p}_i$ of weighted samples $\langle x_i^{(n)}, w_i^{(n)} \rangle \in \mathbf{p}$. These weighted samples are called *particles*. The size of the belief state may vary, but is always finite; the weights are normalised, i. e. they sum to 1.

The filtering density can then be approximated by

$$\widetilde{p}(X_i \mid y_{1:i}) = \sum_{p_i^{(n)} = \langle x_i^{(n)}, w_i^{(n)} \rangle \in \mathbf{p}_i} w^{(n)} \cdot Dirac(x_i = x_i^{(n)}) \tag{3.14}$$

It can be shown that under weak assumptions $\widetilde{p}(X_i \mid y_{1:i})$ converges to the true density $p(X_i \mid y_{1:i})$ when N approaches infinity [58].

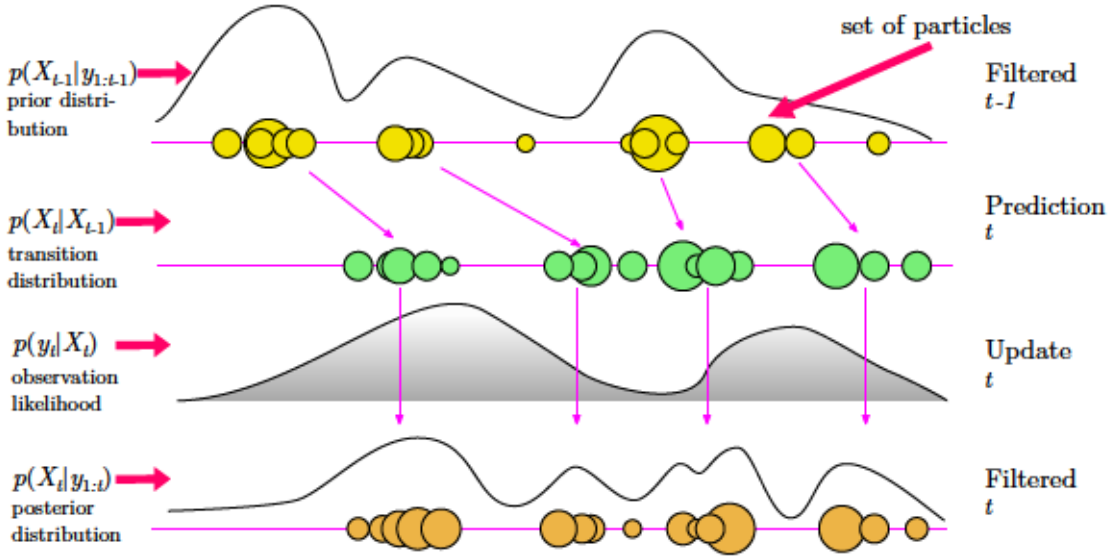


Figure 3.3: Visualisation of the idea underlying SMC methods, here depicted in a one-dimensional continuous state space (x-axis). The densities are approximated by a set of weighted particles (visualised by the size of the circles, not true to scale). Every particle is transformed by the transition model and updated according to the observations, resulting in an approximation of the posterior density. Based on a figure by Fleck et al. [70].

Intuitively, SMC methods sequentially simulate possible sequences $x_{1:i}$ based on the transition model, weighted according to the observations. The basic idea is: if we have a single particle x_{i-1} representing a sample of $p(X_{i-1} \mid y_{i-1})$, we can sample from the transition model $p(x_i \mid x_{i-1})$ of the DBN, and get the likelihood of that particle using the observation model $p(y_i \mid x_i)$; this can be repeated for every timestep i.[2] This idea is shown in Fig. 3.3.

The base algorithm of every SMC method is listed in Algorithm 1. First, INITIALIZE creates the initial set of particles based in the initial distribution $P(X_1)$ (3.12) and the first observation. (For the sake of brevity, we will assume in the following that there is no first observation.) Then, SMC approximation of the filtering density proceeds in three main steps:

1. Given the previous estimate $\tilde{p}(X_{i-1} \mid y_{1:i-1})$ represented by $\mathbf{p}_{i-1}$, *predict* the next state $\tilde{p}(X_i \mid y_{1:i-1})$ using the transition model $p(X_i \mid x_{i-1})$ by drawing samples. This results in a representation $\mathbf{p}_i$ of the integral in (3.13). (Note, however, that a marginalisation done by the integral is not necessarily part of this prediction step.) Depending on the implementation of PREDICT, the belief state's size $\mathbf{p}_i$ may be constant or may differ from $\mathbf{p}_{i-1}$.

[2] One may also generalise the approach by sampling from an arbitrary *importance distribution / proposal distribution* $q(X_i \mid y_{1:i})$ [7, 58, 106]. If $q(X_i \mid y_{1:i}) = p(X_i \mid y_{1:i})$, then this is called the optimal proposal distribution [177]. However, deriving better proposal distributions for the complex human behaviour and sensor models is even more challenging than modelling the transition distribution, as the proposal distribution also needs to take the observations into account.

Algorithm 1 The basic SMC algorithm. This framework is independent of the underlying DBN structure.

```
1: function SMC
2:     ▷ Initialisation.
3:     i ← 1
4:     p_1 ← INITIALIZE
5:     ▷ Sequential filtering.
6:     while y_i ← GETOBSERVATION do
7:         i ← i + 1
8:         p_i ← ∅                                   ▷ New set of particles.
9:         ▷ Compute predictions.
10:        p_i ← PREDICT(p_{i−1})
11:        ▷ Update weights.
12:        for p_i^(n) = ⟨x_i^(n), w_i^(n)⟩ ∈ p_i do
13:            w_i^(n) ← w_i^(n) · p(y_i | x_i^(n))
14:        end for
15:        ▷ Normalise weights.
16:        for p_i^(n) = ⟨x_i^(n), w_i^(n)⟩ ∈ p_i do
17:            w_i^(n) ← w_i^(n) / Σ_m w_i^(m)
18:        end for
19:        p_i ← IMPROVESAMPLES(p_i)                 ▷ optional
20:    end while
21: end function
```

2. Given the prediction, *update* the particles' weights in $\mathbf{p}_i$ according to the observation likelihood. This corresponds directly to the outer product in (3.13).

3. Normalise the particles' weights. This transforms the proportionality to equality in (3.13).

This procedure is repeated as long as observations are available (this loop corresponds to the recursion in (3.13)).

Optionally, several post-processing techniques to improve the sample quality may be employed in IMPROVESAMPLES. In particular, when the *prediction* step increases the size of the belief state, the post-processing step usually limits the size of the belief state. Other techniques include resampling [57], adding noise to the particles' states to increase the sample variance [54] or recombining states [62].

The implementation of the functions INITIALIZE, PREDICT and IMPROVESAMPLES is left to the specific algorithm. GETOBSERVATION must be implemented by the model engineer and provides the latest sensor data.

The particle filter The particle filter is the most common variant of SMC. Its belief state is a set of exactly N particles.

During the initialisation, the particle filter samples N states from the initial distribution (3.12) and assign each a weight of $1/N$. In the prediction step, the particle filter *samples* one possible successor state for each particle $p_{i-1}^{(n)}$. As only a single successor is sampled, the number of particles stays constant.

This repeated sampling of only a single successor state leads in most models to a degeneration of particle weights [60]. Particle degeneracy is the term to describe that most of the particles get negligible weight, and finally the approximation (3.14) degenerates to a single state. To overcome this problem, the particle filter performs a resampling step, i. e. it samples particles $p_i'^{(n)} \sim \widetilde{p}(X_i \mid y_{1:i})$ [7]. Resampling avoids the systematic accumulation of weight for only few particles; a number of resampling algorithms are available, a good overview and comparison is provided by Douc and Cappé [57].

Particle filters have been applied to numerous domains, including object tracking [62, 91, 99], motion capturing [71, 173] and gesture recognition [29], robotics [199] and vehicles [200], activity recognition [115, 152], surveillance [70, 144], estimating geophysical properties of rock formations [68] and atmospheric sciences [17].

Given the human behaviour model's DBN of Fig. 3.2, a particle filter would sample a successor state for a particle's state $x_i^{(n)}$ directly according to the densities of the filtering equation of Section 3.2.1. In summary, the procedural sampling would be done as follows (the details of the particle filter operations are shown in Algorithm 2): As an X state's action can either continue or stop (and a new action has to be selected), the particle first has to 'decide' whether the action stops or not (this is also reflected in the conditional dependency on F_i for A_i in the DBN). Thus, first evaluate F_i and sample $f_i \sim p(F_i \mid a_{i-1}, \mathcal{T}_{i-1}, \mathcal{T}_i, t_{i-1})$. If f_i is false, the particle stays unchanged. Otherwise, first an action is sampled, and then the new S state is computed (or sampled for non-deterministic action effects).

Performance of SMC The time complexity of SMC methods depends only on the number of particles, which is independent of $\mathbf{X}$, and on the complexity of drawing samples from the transition model (3.3). For the symbolic models of our applications, drawing samples from the action selection model (3.5) is linear in the number of actions $|\mathbf{A}|$, as every action's preconditions have to be evaluated. Evaluating the context's transition model (3.7) takes constant time, as the effects are deterministic (and applying the effect of an action is considered to take constant time). Computing the duration model of Eq. (3.4) is also usually a constant time operation for all duration distributions that are practically relevant. Thus, for our models, the complexity of SMC inference using particle filters is $\mathcal{O}(N\,|\mathbf{A}|)$ per timestep.

For the models we are concerned with, $|\mathbf{A}|$ is usually the smallest factor in the number of situations. In addition, all commonly used resampling methods are also linear in the number of particles. Therefore, the computation time of the particle filter (and similar most SMC methods) can be controlled by N. Using a large N increases computation time, but also decreases the error of the approximation of the filtering density (cf. Eq. (3.14)).

However, we can argue that the particle filter is unfit for high-dimensional, categorical state spaces due to two effects: fast particle degeneracy and poor sampling. Both will be

Algorithm 2 The particle filter algorithm, configured to use exactly N particles.

function INITIALIZE
  ▷ Sample N states from the initial distribution of Eq. (3.12).
  $\mathbf{p} \leftarrow \emptyset$
  for $n = 1, 2, \ldots, N$ **do**
    $g_1^{(n)} \sim P(G_1)$; $s_1^{(n)} \sim P(S_1)$
    $a_1^{(n)} \leftarrow a_{init}$; $t_1^{(n)} \leftarrow \mathcal{T}_1$ ▷ The action starts now.
    $x_1^{(n)} \leftarrow \langle s_1^{(n)}, a_1^{(n)}, t_1^{(n)}, g_1^{(n)} \rangle$
    $w_1^{(n)} \leftarrow 1/N$ ▷ Evenly distribute the weight to all particles.
    $p_1^{(n)} \leftarrow \langle x_1^{(n)}, w_1^{(n)} \rangle$ ▷ Create particle from state and weight.
    $\mathbf{p} \leftarrow \mathbf{p} \cup \{p_1^{(n)}\}$
  end for
  return $\mathbf{p}$
end function
function PREDICT($\mathbf{p}_{i-1}$)
  $\mathbf{p}_i \leftarrow \emptyset$
  for all $p_{i-1}^{(n)} = \langle x_{i-1}^{(n)}, w_{i-1}^{(n)} \rangle \in \mathbf{p}_{i-1}$ **do**
    $f_i^{(n)} \sim P(F_i \mid a_{i-1}^{(n)}, \mathcal{T}_{i-1}, \mathcal{T}_i, t_{i-1}^{(n)})$ ▷ Does the action stop?
    if $f_i^{(n)}$ **then**
      ▷ Execute a new action.
      $g_i^{(n)} \sim P(G_i \mid g_{i-1}^{(n)}, s_{i-1}^{(n)}, a_{i-1}^{(n)})$ ▷ Adjust goal.
      $a_i^{(n)} \sim P(A_i \mid f_i^{(n)} = true, s_{i-1}^{(n)}, g_i^{(n)})$ ▷ Sample a new action.
      $s_i^{(n)} \leftarrow a_i^{(n)}(s_{i-1}^{(n)})$ ▷ Set the new state.
      $t_i^{(n)} \leftarrow \mathcal{T}_i$ ▷ Set the new starting time.
    else
      ▷ Continue the current action.
      $x_i^{(n)} \leftarrow x_{i-1}^{(n)}$
    end if
    ▷ In any case, the weight of the particle does not change during prediction.
    $w_i^{(n)} \leftarrow w_{i-1}^{(n)}$
    ▷ Add the particle to the new set of particles.
    $\mathbf{p}_i \leftarrow \mathbf{p}_i \cup \{p_i^{(n)} = \langle x_i^{(n)}, w_i^{(n)} \rangle\}$
  end for
  return $\mathbf{p}_i$
end function
function IMPROVESAMPLES($\mathbf{p}_i$)
  ▷ Use any resampling method.
  return RESAMPLE($\mathbf{p}_i$)
end function

discussed in more detail in Section 5.1. Bengtsson et al. [17] show that number of particles needs to grow exponentially with the number of dimensions of the system for reliable estimates. As a result, particle degeneracy occurs faster for high-dimensional systems. The experimental results of Bengtsson et al. show that a few hundred dimensions already lead to particle degeneracy after a single observation update for even a million particles. For human behaviour models, each categorical state variable is a single dimension, easily leading to hundred dimensions for small models. This confirms Thesis 7.

3.3 The CCBM modelling language

The CCBM toolkit is designed for online situation recognition using either real or synthetic sensor data using models of human behaviour. It supports a language to define models of the DBN structure described in Section 3.1.

The CCBM toolkit with its language has been used for various analyses of situation recognition approaches [10, 111, 113–116, 146, 148, 188, 217–219]. This toolkit also provides the basis for the implementations of the algorithm and methods described in this dissertation. As the modelling language is used for the models used in this dissertation, it will be shortly reviewed in the following section. Understanding the principles of the modelling language allows the reader to understand the complexity of the generated state space defined by the models.

3.3.1 Brief introduction to the modelling language

A key concept of the CCBM toolkit is the separation between implementation of the algorithms and specification of human behaviour. Therefore, a domain-specific language is provided for defining human behaviour models. The CCBM modelling language is based on a variant of PDDL (the Planning Domain Definition Language) [133]. PDDL is an action language that describes a set of *predicates* and *fluents*, which form the state space $\mathbf{S}$, and a set of *action schemas*, which are used to generate the set of *actions* $\mathbf{A}$. CCBM extends the syntax of PDDL to also define the duration model of actions δ_a and specify possible goals $\mathbf{G}$ and the initial state distribution $P(S_1)$.

Predicates assign a boolean value to tuples of objects. For example, the predicates `(hungry)`, `(cooked soup)` and `(printed alice job-a)` may be true in some state $s \in S$. Similarly, fluents can assign any value of a specific domain, not just boolean. For example, `(location soup)` might be `pot`, `bowl` or `eaten`. Within the domain, arbitrary constants can be defined and assigned a type. The CCBM toolkit also supports arbitrary numeric ranges as state variables. See Listing 3 how types and constants are defined and used. Yordanova [217, App. D] provides a definition of the modelling language.

An action schema is a parametrised action, each parameter can take values of a specific type. By instantiation, the set of actions $\mathbf{A}$ can be derived from the action schemas. To distinguish actions from action schemas, we occasionally use the term *grounded action*.

Consider the action schema `eat` of Listing 4. It takes an object to eat as parameter, and defines necessary preconditions and the effects of that action which also depend on the parameter. In this case, the precondition is that the user has cooked and actually is

Listing 3: Exemplary definitions of types, constants and fluents in PDDL.

```
(:types
    edible - entity ; every edible is also an entity
    location
)
(:constants
    carrot bread - edible
    table sink hand bowl eaten - location
    self spoon - entity
)
(:functions
    (location ?o - entity) - location
    (objects-in-hands) - (number 4) ; can hold up to 4 objects in both hands
)
```

Listing 4: Exemplary action `eat` defined in PDDL.

```
(:action eat
  :parameters (?what - edible)
  :duration (normal 300 60) ; duration in seconds follows normal distribution N(300, 60)

  :precondition (and
      (hungry)
      (cooked ?what)
      (= (location self) table)
      (= (location spoon) hand)
      (= (location ?what) bowl)
  )

  :effect (
      (not (hungry))
      (assign (location ?what) eaten)
  )

  :observation (set-action (eat))
)
```

hungry, is itself at the table with a spoon in the hand, and the object to eat is on the plate. Preconditions are first-order formulas over predicates - existential and universal quantification are supported as well as negation. `hungry` and `cooked` are simple binary predicates, the former is also set to false in the action's effect. In this example, the only fluent is `(location ?o)`, assigning any object o its current place.

The action schema has a parameter of type `edible`. For instance, if two constants `carrot` and `bread` of type `edible` have been defined, two grounded actions `(eat carrot)` and `(eat bread)` in $\mathbf{A}$ would be defined by this action schema.

Every predicate and fluent gets transformed into a state variable of S. In the case of the predicates and fluents of the action `eat`, a state $s \in \mathbf{S}$ would be factored into three boolean variables (one for `hungry` and two for `cooked`), and four variables with domain sufficiently large to represent all possible locations (one for each fluent `(location self)`, `(location spoon)`, `(location carrot)` and `(location bread)`).

Likewise, every grounded action defines the precondition function $pre_a(s)$ of Eq. (3.6) as well as the transition function $a(s)$ of Eq. (3.7), which takes the previous state as parameter, and returns the new state with the effects applied.

In addition to defining a bare transition model between states, every action can also be associated with a duration model and observation model, as can be seen in Listing 4. The duration model defines an arbitrary probability distribution. The observation model of an action refers to a user-supplied function that computes the observation likelihood $p(y_i \mid a_i)$. Similarly, the observation likelihood $p(y_i \mid s_i)$ can be defined by adding an `:observation` clause outside of actions.

Finally, state specifications for different initial states as well as goal states can be defined. A goal is defined by a first-order formula similar to preconditions.

3.4 Application domains and datasets

The algorithms and methods developed in this thesis are evaluated on different datasets and models of human behaviour. All these datasets have been used to evaluate the CCBM toolkit before, and all models satisfy the model-related requirements defined in Section 1.3. Target of the evaluations is

- to assess and discuss the influence of different models and model properties on the inference efficiency (Definition 2, page 24). In particular, I am interested in the state space size, number of actions and action duration models. Chapter 2 has shown that these are less studied for causal models of human behaviour that adhere to the requirements of Section 1.3.

- to show that my proposed improvements to the inference are not tailored for a specific model, but can be applied to different models of human behaviour.

Therefore, the models have been selected to represent not only different application domains, but also different properties and state space complexities. Additionally, these behaviour models have already been developed and real sensor data has been recorded, all by fellow researchers (cited in the corresponding sections). This allows this dissertation

Table 3.1: Summary of the key properties of the five models

Model	Users	Ground actions	Reachable state space
Office	3	124	1,957,158
Meeting	3	84	66,587
Indoor localisation	7	238	3.7×10^{19}
Kitchen	1	97	146,552,922
CMU kitchen	1	393	$> 10^{12}$

to concentrate on the evaluations of the algorithmic contributions. In the following, the datasets and corresponding behaviour models will be presented.

The models are from the application domains office or kitchen. Chapter 2 has shown that these are the most widely used domains for human behaviour recognition (see Fig. 2.3 on page 45). Both environments play an important role in our everyday life and thus the models represent possible application domains well. Previously, office environments were often the target of assistive technologies [49, 76, 100, 115, 208, 209, 220]. Assisting people in their everyday activities is an increasing research topic (see Thesis 1 in Section 1.1). Therefore, recently the assistance of activities of daily living has received growing attention [86, 88, 176], of which preparing and consuming a meal is a typical scenario [64, 113, 172, 201].

Table 3.1 summarizes the key properties of the models.

3.4.1 Office domain

The office domain models one of the simplest real-world scenario within this thesis. A room contains a printer and a coffee machine that people want to use concurrently – it is therefore a multi-user model. The model contains actions for entering and exiting the room and walking to different locations. Paper and ground coffee are resources that can be taken from different locations (only one at a time if the hands are free) and are used to refill the printer or coffee machine. A paper jam may occur, which must be repaired before printing is possible.

The model can be parametrised with the number of users and print jobs. Listing 5 lists the `print` action of that model. The model has in total 9 action schemas and 6 predicates. For three users and three print jobs, this results in 124 grounded actions and 34 state variables, with $|\mathbf{S}| = 1,957,158$.

This model is primarily used as an example of modelling and how difficult it may be to correctly model an apparently simple domain.

The dataset contains 6 recordings [104]; four recordings with a single user, one recording for two users and one recording for three parallel users. The recordings cover on average 89 seconds of data, with one observation per second. The observation data consists of anonymous presence sensors (specifically, pressure mats) at six key locations (for instance door and printer). How many users are at a location is unknown, the sensors only tell if there is at least one user.

Listing 5: Exemplary `print` action in the office domain.

```
1 (:action print
2   :parameters (?j - job)
3   :precondition (and
4       (not (printed ?j))
5       (not printer-jammed)
6       (has paper printer)
7   )
8   :effect (printed ?j)
9 )
```

3.4.2 Meeting domain

Three persons hold a presentation and discuss in a meeting room. The model's actions comprise walking to different locations, sitting on a chair, as well as actually presenting and discussing. A presentation can only start when the other persons are seated. To restrict the model, it was assumed that seats are not changed and that a person could not walk to different locations without doing anything there, so a person was only allowed to walk once before and between presentations. The model consists of 18 action schemas, resulting in 84 grounded actions, and 78 state variables, resulting in a state space of $|\mathbf{S}| = 66587$.

As sensor data, a tag-based localisation system was used, i. e. 2-dimensional coordinates for every user are available. Task of the inference is then to infer the executed situations (e. g. a discussion, or a presentation of a particular user) based on the position data. Over all, 20 different real life recordings are available [105]. On average, each recording lasted 4.1 minutes and has 3000 observations.

3.4.3 Indoor localisation

An office environment with several office rooms along a floor was equipped with passive infrared sensors. Five sensors were deployed along the floor, with one additional sensor in a public room. Up to seven users can move within the different rooms of the environment. Additionally, several actions such as getting a coffee or having a talk can be executed at particular locations. The objective was to identify which users were in which rooms, based only on the anonymous presence detectors.

The model is composed of 10 action schemas, with different numbers of grounded actions and state variables for different numbers of users. For a single user, the model consists of 34 actions and 7 state variables generating 625 states in $\mathbf{S}$. As the set of actions and states of one agent are independent of all others, there are in general $34n$ actions, $7n$ state variables and 625^n states. This makes 238 actions, 49 state variables and $|\mathbf{S}| \approx 3.7 \times 10^{19}$ for the maximum number of seven users in the dataset.

There are 36 recordings available [93], with one to seven users present in the environment. On average 470 sensor events were recorded.

3.4.4 Kitchen domain – single recipe

A single user prepares and eats a carrots soup in a kitchen environment. The experiment included the tasks setting up the table, eating the meal, cleaning up, and washing the dishes. The actions respect several causal relations, such as an object can only be taken if at least one hand is free, and cutting the carrot is only possible when a knife is in a hand and the carrot is on a cutting board.

The model is comprised of 25 action schemas resulting in 97 ground actions. The model has several predicates over eleven objects, there are five different locations. This results in 60 state variables and a state space of size $|\mathbf{S}| \approx 146$ million.

The dataset contains seven recordings [112], every recording was performed with a different user. Every user's body motions were recorded by six degree of freedom inertial measurement units (IMUs). The users were instructed to shorten long actions to decrease the influence of single actions to the overall recognition performance. On average, each user needed 4.2 minutes to achieve the goal. Every recording has on average 950 multi-dimensional pre-processed IMU observations that are used as input for the inference [113].

3.4.5 CMU kitchen domain – multiple recipes

The CMU dataset [201] uses a similar setting of preparation of a meal in a kitchen environment. However, this dataset is more complex and extensive. A user is cooking one of five different meals (brownies, eggs, pizza, salad or sandwich). This increases the complexity of the model in comparison to the single recipe kitchen domain. Our model, which is able to explain the annotated action sequence, consists of 14 action classes such as "put", "open", "clean" and "walk". These action classes are modelled by 26 action schemas (e. g. due to actions that can take different parameter numbers for slightly different semantics). The generated model consists of 393 grounded actions and 130 state variables. This makes it the most complex model.

Listing 6 shows a "fill" action schema of the model. This action schema is one of the more complex schemas and demonstrates how this action schema is fitted to the different possibilities of how some object can be transferred from one container to another. It demonstrates that the action schemas are restricted with many exceptions, handling the idiosyncrasies of daily life. This explains the large number of grounded actions (for many different object combinations) and state variables (many different context states to be aware of for correct causality).

The size of the reachable state space is not easily computable, but exceeds 10^{12}. This is not the largest state space compared to the localisation model (Section 3.4.3), but the largest state space for the behaviour of a *single* person. The state space of the localisation model is simply a combination of up to seven single user state spaces of size 625.

This dataset contains 86 data recordings of 39 distinct users (some users cooked different meals).

Listing 6: Excerpt of a `fill` action schema in the CMU domain, showing how complex individual actions are, and why the state space for modelling every-day behaviour can grow very fast (due to the need for many different interacting state variables). This schema models actions of filling some object from one container to another container. For example, `(fill jam jam_glass bread)` models "filling" (i. e. putting) jam from the glass to a slice of bread.

```
(:action fill
  :parameters (?what - fill1 ?from - fill2 ?to - fill3)
  :precondition (and
      (not (= ?from ?to))
      (or
          (= (is-at ?from) hands)
          (= ?from tap)
          (= ?from peanut_butter_glass)
          (= ?from jam_glass)
      )
      (or
          (= (is-at ?to) hands)
          (and (= ?to sink) (= ?what oil) (= ?from pan))
          (and (= ?to bowl) (= (is-at bowl) counter))
          (and (= ?to measuring_cup_s) (= (is-at measuring_cup_s) counter))
          (and
              (= ?to baking_pan)
              (or (= (is-at baking_pan) counter) (= (is-at baking_pan) board))
          )
          ...
      )
      (or
          (and
              (= ?to baking_pan) (= ?what dough) (= bowl ?from)
              (stired ?from) (in brownie_mix ?from)
              (in water ?from) (in oil ?from)
              (= (in-num egg ?from) 2)
          )
          ...
      )
  )
  :effect (and
      (in ?what ?to) (not (in ?what ?from))
      (not (stired ?to)) (not (is-clean hands))
      (when
          (and (= ?what dough) (= ?from bowl) (= ?to baking_pan))
          (not (is-clean bowl))
      )
  )
  :observation (set_action (actionId fill))
)
```

4 Testing human behaviour models for errors

Summary: Engineering human behaviour models is an error-prone process, which can degrade performance and accuracy of the inference. Using established methods and tools of model-checking, many errors can be found. I presents first results and experiences on model checking for activity recognition and provide arguments for Thesis 3.

Contribution: This chapter provides classes of modelling errors, guiding engineers to relevant properties to check. It extends and refines the verification approach of the model development process by Yordanova and Kirste [218].

Parts of this chapter are based on

> *[149] Martin Nyolt, Kristina Yordanova, Thomas Kirste:* Checking Models for Activity Recognition. *International Conference on Agents and Artificial Intelligence (ICAART), 2015.*

Only little work has been done on the model development process for behaviour models for activity recognition [218]. When inferring the current situation using Bayesian filtering (as described in Section 3.2), the behaviour models have to be causally correct and represent all desired details. Remember the filter equation which states that

$$\overbrace{p(X_i \mid y_{1:i})}^{\text{inferred situation}} \propto \overbrace{p(y_i \mid X_i)}^{\text{obs. likelihood}} \overbrace{p(X_i \mid y_{1:i-1})}^{\text{prediction}}.$$

Correct models are important The human behaviour model is responsible for the prediction. Thus any error in the model directly influences the recognition result: Any behaviour not present in the model can never be recognised (even if the sensor data support it). Any wrong behaviour in the model will re-enforce noise and misleading information from the sensor data.

This chapter will not formally or empirically analyse the effects of incorrect models on the estimated posterior distribution. From experience, the modelling errors discussed here are very difficult to notice with usual classifier performance metrics. Similar to bugs in other software, significant effects are visible only under certain conditions. For applications such as activity logging for self-monitoring this can be annoying. For health-care applications where the assistive system autonomously executes actions based on

the inferred situation this may be harmful. For such applications, developing a formally correct model is essential.

Complex development process The models have to satisfy different and partly opposing requirements. This leads to a complex development process and complex models that are difficult to keep free of errors.

On the one hand, models have to cover many causal dependencies between fine-grained activities in a detailed state space (challenge of open-world behaviour mentioned in Section 2.3.2). Without certain behaviour represented in the behaviour model (zero support in $p(X_i \mid y_{1:i-1})$), the estimate will also have zero support for these behaviours, regardless of the quality of the observation data. For instance, if a person has taken a knife from a counter, the model should always allow to put the knife down as long as the person is at the counter or a table.

On the other hand, a model that allows all unrelated interactions quickly becomes too complex, rendering (probabilistic) inference infeasible due to large state spaces (as stated in Thesis 4). Although we strive for recognising *all* everyday activities, not all sequences of activities are physically possible or causally correct. Furthermore, currently only models restricted to a specific application are practical instead of all imaginable everyday activities. Therefore, techniques have been developed to reduce these models on a symbolic level [219], reducing the number of required actions and states to distinguish.

Verifying models The result is a conflict between creating complex models and simplifying and fine-tuning the model. This tuning is error-prone and increases chances for models to become inconsistent. I experienced modelling errors and their negative effect myself while experimenting with the inference algorithms. Clearly the model should be free of errors, otherwise it could lead to undesired or even harmful actions of the system. Therefore, the need for verifying the correctness of models and checking consistency arises.

Model checking is an established tool for verifying the correctness of models by checking properties. For most digital or information processing systems, the set of correctness properties is usually limited, as all operations are specified prior to implementation. Model checking then assures that the implementation follows the specification. In contrast, such a specification is not available for everyday human behaviour. Therefore, the set of properties to check is not available from the start but have to be derived. However, it is unclear *how* these properties have to be derived.

Contribution Based on my experience, I identified a set of classes of modelling errors. For each error class, I define a template of properties which covers the error class. These property classes can guide the model engineer to derive properties and find more errors. These findings lead to Thesis 3.

Thesis 3. (repeated from page 58) *Engineering useful and correct models of human behaviour is challenging and error-prone. Human-built models of human behaviour contain*

errors that influence the recognition. Many errors can be found by checking for a few classes of errors.

In this chapter, three activity recognition models of Section 3.4 are evaluated, which all have been designed to work in and have been applied to a real smart environment [113, 114, 116]. This chapter first reviews three different approaches to model verification, including a brief introduction to model checking, in Section 4.1. The approach of model verification is derived and presented in Section 4.2, including a set of requirements. Section 4.3 presents different property classes that can be used as a guide to derive relevant properties. These classes are then used to evaluate existing models in Section 4.4. Section 4.5 concludes this chapter with a discussion.

4.1 Model verification review

4.1.1 Human behaviour model development process

Yordanova [217] has proposed the first systematic development process for causal human behaviour models [218] which targets the specific needs when developing a model such as the CCBM (Section 3.3). This process is giving structured guidelines and best practices. It is a two-layered process that iteratively develops

- the transition model (which corresponds in our model to the states $\mathbf{S}$, actions $\mathbf{A}$ as well as the precondition $pre_{a_i}(si-1)$ and effect functions $a_i(s_{i-1})$ for every action a_i),
- the probabilistic observation model ($p(Y_i \mid s_i, a_i, g_i)$),
- probabilistic hints (heuristics corresponding to the action selection model $p(A_i \mid f_i, s_{i-1}, a_{i-1}, g_i)$) and
- the actions' duration distributions $\tau_{a_i}(D_i)$.

Each iteration includes design, implementation and verification. A final evaluation measures the performance of the final model on real data (which has been acquired prior to any model development). Yordanova [217, 218] gives specific guidelines for data collection, domain analysis, implementation strategies [219], verification and evaluation (using the collected data and annotations).

One of the main prerequisites for this development process is the data collection. In particular, the development and verification phases utilise collected data from experiments with human subjects. The subjects perform the task and training data from the sensors for the observation model is collected. During or past the experiment, the action sequences that the user executed are collected into *annotated plans*.

Verification process The model verification consists of three steps, each verifying a different model parts by different test cases. Only when one part has been verified the next part will be developed.

First, the transition model is verified based on the annotated plans of the collected data. These plans are used as test cases to check if the transition model supports the plans. Second, the transition model, observation model and heuristics are verified using observations sampled from the annotated plans. Until now, actions are assumed to take no duration (a single timestep) for execution. The duration model is verified at last, sampling observations from the annotations by taking the durations into account.

4.1.2 Functional tests

Testing A number of verification tools and methods have been developed for software engineering [22]. The basic tools are software testing techniques [22], where test cases are designed to check specific aspects. Targets of software testing are single functional units (unit tests), interactions between different units (integration tests) and the entire system (system tests). Only unit testing is applicable for this chapter, because the transition model of human behaviour is a single component of the situation recognition system.

Functional tests Unit tests are functional tests, i. e. single functional units are tested separately. Often, individual functions or methods are tested, with different parameter combinations and checked, if they produced the desired outcome. Thus, the test designer focuses on individual functions for testing. Software is usually written to serve a specific purpose, which is documented and desired behaviour specified. Thus, the desired outcomes of the functions are often specified, and test cases can be designed based on the specifications.

Programming by contract A similar, but more formal and thorough approach as been promoted by Meyer [137]. Design or programming by contract [136] adds assertions to every method, namely preconditions and postconditions [137]. Preconditions define parameter and state combinations that are allowed, and postconditions are a guarantee to the caller defining the outcome.

As an advantage over functional tests, programming by contract does not rely on single test cases. Indeed, every test case can be seen as an instantiation of the assertions, where the effects are compared against the instantiated postcondition. Every method invocation may be seen as a test-case, making it easier to recognise faults faster. However, without functional tests, these faults are not recognised until executed in production, so tests are still necessary.

4.1.3 Model checking

Model checking is well established for verifying soundness and correctness of models [40], and this section will provide a brief introduction. Model checking has been successfully applied in circuit design and software verification, among others. Research in model checking has allowed to proof correctness of models with large state spaces of over 10^{200} [110] (of course highly depending on the model).

Basics Each model describes a dynamic system of evolving states $s_{1:t}$. For model checking, each state s is associated with a set of propositions that hold in that state. Note that model checking usually deals with discrete time systems, but real-time extensions are also possible.

For model checking, the engineer has to specify a property that the model must satisfy. The property is described as a formula in temporal logic, which in its basic form can be propositional logic not over a single state, but a sequence of states. A simple property may be of the form "state x can always be reached"; in a human situation recognition model, this might be "the user can always walk to the kitchen". The model checker is a separate tool which verifies if the model satisfies the properties. If the property is not satisfied, the model contains an error.

In contrast to functional tests, model checking verifies the outcome over all possible execution paths, instead of single cases. Similar to functional tests, in the usual applications of model checking the desired outcome of the system is specified and properties can be designed based on these specifications. The properties correspond to the assertions of programming by contract for single methods, but cover the behaviour of the whole system.

Temporal logic Temporal formulae are defined over sequences of states in a temporal logic. The most common temporal logics are linear temporal logic (LTL) and computation tree logic (CTL). LTL formulas describe a single path of model states, while CTL can also quantify over possible branches of execution (due to non-determinism, e. g. which action to execute).

A formula in temporal logic is composed of propositional formulas and temporal operators. Common temporal operators are

- `X` ϕ (ϕ must hold in the next state),
- `F` ϕ (ϕ must hold in at least one state of the sequence),
- `G` ϕ (ϕ must hold in all states of the sequence),
- `E` ϕ (ϕ must hold in at least one possible path, e. g. due to non-determinism),
- `A` ϕ (ϕ must hold in all possible paths), and
- ϕ `U` ψ (ϕ must hold until ψ holds),

where ϕ and ψ are formula in propositional logic describing a single state or a nested temporal formula. For instance, the formula `X (`$x \wedge$ `X G` y`)` describes that x must hold in the next state, and for every state after that, y must hold. The sequence $(\{x\}, \{y\}, \{x, y\}, \{y, z\}, \{y\}, \ldots)$ satisfies that formula, but $(\{x, y\}, \{x\}, \{y\}, \{y\}, \ldots)$ does not, as y does not hold in the second state.

Depending on special classes of temporal logic, not all combinations of temporal operators are allowed. This is used to reduce the complexity of the logic and thus complexity of the model checking problem. A short introduction to temporal logic is given by Venema [205].

Efficiency A simple model checker may do a depth-first traversal of all possible execution paths and check the path against the property. However, this is only feasible for very small models, as the computation time is linear in the state space size, and may be up to exponential in the size of the formula. Therefore, a number of techniques have been devised to alleviate the state space explosion problem [154]. Clarke et al. [40] provide an in-depth treatment of model checking techniques.

The two main classes of techniques to make model checking tractable are state space reductions and efficient storage [154].

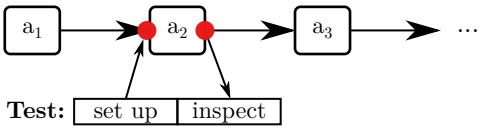
State space reduction techniques exploit the fact that different parts of the state space behave similar, and thus often either both satisfy the property, or none. Examples are searches for symmetries in state spaces (e. g. using Petri nets [185]) or partial order reductions, which account for independent actions that can occur in any order without changing the global behaviour [77].

Efficient storage techniques represent the large state spaces in compressed form, allowing to handle larger models with less memory, and at the same time reason over multiple compressed states at the same time. One example are decision diagrams [61], which allow compressing the state space, the transition function and efficient computations on these representations (e. g. checking if a proposition holds in some or all states or computing successor states of a large set of states), but other techniques exist as well [154, 187].

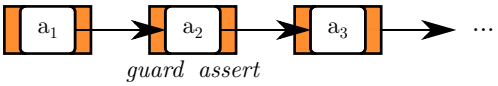
Approximations and heuristics for model checking have also been developed, but are not of relevance for this dissertation.

Model checking as situation recognition Due to its wide success, model checking has found application in AI and many researchers also discovered techniques from model checking as a utility for activity recognition. Magherini et al. [130] investigate the applicability of temporal logic and model checking to the problem of plan recognition (assuming actions observations). They present a linear temporal logic reasoning about past events and real-time constraints on actions. The application scenario is the recognition of activities of daily living, tracking the activities and calling assistance on potentially erroneous (e. g. dangerous) behaviour. Each activity is represented as one formula in the temporal logic, and at least one additional formula for erroneous executions of the actions. Inference works by finding that activity (i. e. formula) that is satisfied by the observed action sequence.

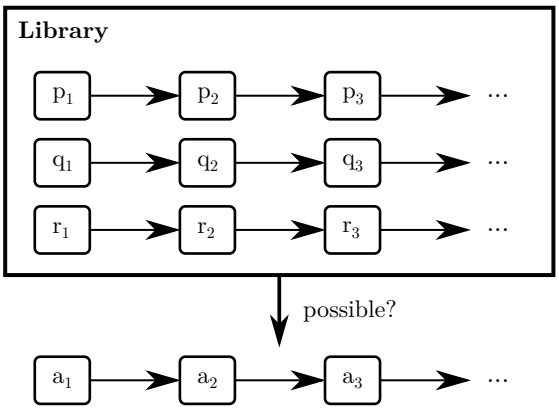
Cimatti et al. [39] design a planner in non-deterministic domain, i. e. domains with actions that have non-deterministic effects. They apply techniques from symbolic model checking and represent sets of states (possible outcomes of actions) as propositional formulae. Based on this model-based planner (MBP), Gromyko et al. [79] synthesise a controller for non-deterministic discrete event systems. Given required properties in temporal logic and a system, they generate a controller for this system such that it satisfies the properties.



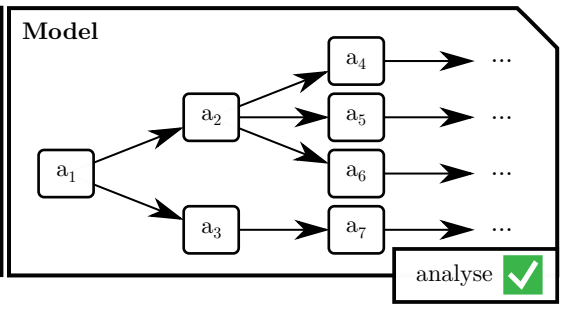
(a) Functional tests verify actions independently. The outcome of an action is compared with the expected result.



(b) Contracts can be used to monitor if each action is executed in the right contexts and if the desired outcome is produced.



(c) The model verification of the model development process uses a set of pre-existing plans. The model is verified if it can reproduce all of the plans.



(d) Model checking analyses the behaviour model as a whole, not plans or actions.

Figure 4.1: Comparison of the different verification approaches. a_i are the different actions that are executed during a behaviour sequence.

In many cases applying model checking techniques, the idea is to compactly represent states and efficiently generate a search tree. This is often done using efficient data structure such as Binary Decision Diagrams (BDDs). In contrast to Magherini et al., we do not recognise activities using model checking techniques in this chapter. Instead, we apply model checking in its original sense: proving that the models (used for activity recognition) fulfil all desired properties and contain no errors.

4.2 Model-verification approach

This section discusses the proposed approach to verifying the human behaviour models.

4.2.1 Requirements

The verification task is to find modelling errors that can result in recognising the situation incorrectly. As such, properties have to be tested that describe human behaviour.

One starting point of this effort was a modelling error in the meeting domain (Section 3.4.2). The model could come to a situation where multiple users are located at a seat, but only one can sit down. This is expected, but none of the other persons could

walk to a different (free) seat; once walking to a chair, the model only allows to sit down. If the chair is occupied, the user cannot do any action – and the model is stuck at this point, whereas in the real situation, the users naturally walk to a different seat.

Based on this example, a few requirements on model verification can be derived:

State analysis It must be possible to analyse the state space of the transition model and find states with certain properties. In the example, it is a state with no further possible actions.

Sequence analysis Sequences of human behaviour, not just single states, must be supported (e. g. after entering the room, every person must be able to get a seat). This includes temporal relations between situations, without an exhaustive specification of every situation (e. g. activities between entering and being seated may be possible).

Access context When formulating properties, the context state of every situation must be accessible, e. g. to specify that eventually every person is seated.

Multi-user Users interact, and thus the effects of these interactions must be analysed as well.

4.2.2 Discussion of related approaches

Figure 4.1 compares the approaches of the four different model verification techniques.

Human behaviour model development process The verification by Yordanova [217, 218] is based on annotated plans of the collected data. This has three main disadvantages.

- As producing annotations is a very costly task [83, 196], usually only a very limited number of sequences are available. These plans are thus samples of execution sequences. Due to the complexity of human behaviour (Section 2.3.2), they cover only very few situations. It is thus unlikely, that errors are found for uncommon (but possible) situations.
- Plans are sequential, thus all actions are only executed in the specific contexts of their plans. Behaviour in different contexts is not tested. Because human behaviour is complex (open-world, rich dynamics, see Section 2.3.2), only a few samples of behaviour sequences do not adequately represent all possible sequences.
- This verification does not detect cases where the model allows situations that should be impossible. Additionally, this not only reduces the accuracy, but also reduces the inference performance and thus efficiency, as more situations have to be handled.

This approach does not allow to analyse the state and access the context. For instance, after a given plan, this approach does not allow to check if more actions are executable. It also does not allow to check if a given state predicate (e. g. a person is seated) holds.

Functional tests Functional tests as well as programming by contract focus on individual functions. Either test cases or assertions implicitly or explicitly define properties of the functions. They ensure that functions or methods are only called in certain contexts and have a desired outcome. Essentially, the benefit of functional tests or contracts on methods are a declarative formulation of the imperative method. However, as the human behaviour model is already described using declarative preconditions and effects (Section 3.3), additional contracts or tests do not benefit the model design.

Using tests and assertions, the context state can be accessed, and the state may also be analysed. However, state sequences cannot be easily analysed; this may be implemented using complex test procedures, but is not the intended use-case of unit tests and assertions. Similarly, assertions and tests if single actions do not cover multi-user interactions.

Model checking Model checking has the advantage that it allows to verify properties over the complete execution sequences of the model, not just effects of single actions. Thus, it supports the analysis of state sequences. Because model checking verifies the properties over *all* possible execution sequences, it also equivalent to testing an exponential number of sequential plans. Temporal logic is expressive enough to describe behaviour sequences and relations, and to access context through propositional statements. Because it analysis the complete execution sequence, it can also deal with multi-user domains.

4.2.3 Model verification approach

Model checking offers the most features and satisfies all requirements. Usually, model checking is used as a *proof* that the model or implementation satisfies the specification, e. g. a leader-election algorithm terminates successfully [117]. These specifications are used to formulate the properties that the model checker verifies. However, as stated in the introduction, there is no complete set of specification for human behaviour.

Testing process Therefore, the verification of human behaviour is a *testing* process. Different properties have to be derived that cover potentially erroneous or important situation sequences. Then the model checker is used to prove if these properties are true. If a property does not hold, a potential error has been found and the model has to be revised. Of course, not all properties of human behaviour can be feasibly formulated

Checking for high-level errors Therefore, a relevant subset of properties has to be identified from which properties can be chosen. Although model verification is a testing process, I argue that the properties should *not* be inspired by the approach of unit testing individual functions (i. e. actions) or by assertions (by programming by contract). Properties based on assertions are likely to be re-statements of the actions' preconditions and effects (due to the declarative model description), possibly making the same conceptual errors. This would also contravene the purpose of analysing behaviour *sequences*.

Therefore I argue it is most useful to describe properties that cover long-term behaviour and situation sequences, and not simply define assertions to execute specific actions. This can detect errors by interactions of different actions and also multiple users.

Checking for causal errors We focus on the causal errors that can be found in the transition model and drop the duration models and the observation models. The duration model can be validated using the validation process of Yordanova [217, 218]. Similarly, the observation model is also covered by this process.

Tool selection For the subsequent evaluation, we employ the PRISM tool as a state-of-the-art, off-the-shelf model checker [118]. The main reasons for selecting PRISM were

Use of BDDs Previous experiments have shown that BDDs allow to represent much more states than an explicit representation. For instance, the CCBM toolkit's integrated state space analyser requires 17 GB of RAM and approx. 40 minutes to construct the state space graph of the Kitchen model (Section 3.4.4) on contemporary hardware. On the same machine, PRISM requires less than 2 GB of RAM and only 90 seconds to construct the BDD representation.

Modelling language The PRISM modelling language is a state-based language with actions defined based on preconditions and effects (in PRISM terminology these are commands, guards and updates). Thus, the modelling language can cover most of expressiveness of the CCBM language based on PDDL.

Counter examples PRISM can generate a counter example of a property if the model violates the property (for all relevant property classes). Counter examples give the model engineer an example state sequence which violates a property. This sequence can be helpful to find the cause of the error and fix the problem [41].

These advantages are not exclusive to PRISM, there are other tools with similar properties. PRISM also had a number of soft factors, which made me select this tool for the case study.

Documentation The tool, the modelling language and the property specifications are well documented, making it easy to generate own models.

Support of different models PRISM supports different types of model specifications (discrete and continuous time Markov chains, timed automata). That is, PRISM also supports non-deterministic and probabilistic models. Currently, the CCBM toolkit does not support non-deterministic actions, but these would be supported by PRISM if they were added later.

Support for different property specifications PRISM supports both LTL and most parts of CTL, which makes it convenient to try different property specifications. For probabilistic and non-deterministic models, PRISM also supports computing expected rewards or costs as well as the probability that a property will hold. This feature has also been used for unreported experiments using BDDs as data structure for other computations on the model.

Model transformation For the evaluation in Section 4.4 I extended the CCBM toolkit's compiler to transform the PDDL specification (see Section 3.3) into the PRISM modelling language. The PRISM language does not support all of the features of CCBM, therefore some adjustments are made. First, PRISM does not support action schemas (i. e. parametrised actions). Hence, the compiler instantiates them to grounded actions. Second, PRISM also does not support conditional effects. These are resolved by generating different actions for each possible combination of the outcomes of the conditional effects. These two changes do not change the semantics of the model but only increase the model's description length.

Third, the CCBM toolkit provides a mechanism of *non-repeating actions*, i. e. actions which are not allowed to be repeatedly executed directly after each other. These have been added to reduce the model complexity (in terms of branching factor) and are used accommodate prior knowledge on the action structure. This is useful when an action has not a direct effect (e. g. waiting), thus it may be executed consecutively, but it shall nevertheless have the effect of not being executable until a different action has been executed. Non-repeating actions are not supported by PRISM and ignored for the transformation. This may alter the state space, but it usually does not affect any other properties, as its purpose is mainly for actions without effect.

Last, PRISM does not support probabilistic action durations (it only supports timed automata with clock constraints). As has been discussed above, only the transition model is used for checking and actions are assumed to take no duration.

All goal formulas are transformed into labels in the PRISM models. These labels can be used in property formulas to conveniently check if a goal-state has been reached. The formula in PRISM can refer to state variables as well as labels, which are user-defined formulas on state variables. For example, the formula `F` *goal_1* states that it is always possible to reach a goal state referred to as "goal_1".

4.3 Property classes

This section proposes different classes of properties that check high-level causal errors in the behaviour model. These classes shall help the model designer to formulate reasonable properties. The challenge is to find non-trivial properties that cover high-level causal behaviour, i. e. are not restricted to single actions.

The first class contains basic soundness criteria that are expected to be satisfied by all models. The other two classes are based on experience, generalizing from known errors.

Soundness Soundness properties are imposed on other models as well, for instance models of business processes [1, 67]. These properties are not specific to human behaviour model, but can be considered as basic criteria that every model should satisfy. Although they are not specific for human behaviour models, these properties also apply to long-term causations and behaviour sequences.

Deadlock freeness Deadlocks are states in which no action can be executed. Deadlock freeness can be checked using the formula `G !"deadlock"` (PRISM automatically

provides a label "deadlock" for deadlock states). Deadlocks should be avoided in behaviour models. Either deadlock states are impossible in the underlying domain, in which case they should be unreachable, or they are possible, in which case actions are probably missing and the inference process cannot recognise subsequent activities. The underlying assumption is that humans almost always find a solution or continue acting in any case, so that deadlocks do not occur in the application (of course, domain-specific exceptions may exist).

Livelock freeness Livelocks, in our activity recognition case, are states in which actions are possible, but no goal state can be reached (checked with `F "goal"`). They indicate a possible problem in the domain or model, where the overall task is impossible to accomplish.

Domain-specific invariants I propose to check for domain-specific invariants. These are properties that are assumed to be true at all times, regardless of the behaviour or the behaviour sequence. These are usually properties of states. As these properties have to hold at any time, they are not specific to the execution of individual actions. These formulas are often of the form `G` ϕ (i. e. in all cases, ϕ must hold at any time), where ϕ is a propositional formula on the state.

Properties can be derived by taking different factors into account, for instance

Physical invariants These detect any state that is impossible by common sense and physical interactions. Examples are that any object or user can only be at on location (not regarding location hierarchies), and, depending on the model of location, only one object can be at one location.

Behaviour invariants and user abilities Other invariants include those that are related to users and their behaviour. These invariants are not determined solely by physical restrictions, but on the model of the application domain. For example, one invariant may be that a user can hold at most two objects at any time (one per hand).

However, invariants and modelling decisions must be extensively validated beforehand. For instance, experience from the single recipe kitchen experiment Section 3.4.4 has shown that users will find ways to defeat modelling assumptions anyway, for instance closing a shelf with their head because both hands were already holding an object.

Context affected by different actions Particular attention should be paid to the context states that are influenced by different action schemas. A context state that is influenced only by a single or very few action schemas (e. g. turn on a device or turn of a device) are "local" to such action schemas. Thus, the state can be modelled by their preconditions and effects, which are more unlikely to be erroneous as the interactions tend to be less complex. In contrast, states that are affected by many different actions can more likely result in invalid states. For example, the location of items can be affected by actions such as taking, putting, filling into other items, separating or combining items and so on.

Long-term causations While the former class captures invariants by multiple actions, this class captures the dynamics of (mostly different) actions executed in sequence. Thus, long-term causations ensure properties which can only be *implicitly* encoded within multiple actions. Making these assumptions and causations explicit is important to document and validate these properties. The (prior) knowledge of long-term causations can be made explicit using temporal logic.

Long-term causations include action sequences where one action is a prerequisite to a following action (e. g. first turning a device on, then using the device, then turning it off). Usually users can execute different sub-tasks in parallel, thus the action sequences may contain unrelated actions. Typical classes of long-term causations are:

Decision of users Sometimes models include decisions of users (e. g. where to sit, when to present) which do not immediately influence the behaviour, but alter the progress of the model when the decisions are realised later. Here properties typically are of the schema `G (`$\phi \Rightarrow F\,\psi$`)`, meaning only when ϕ holds, the model always behaves in a particular way according to ψ.

Repeatability In real-world domains, repeatability of certain actions are desired, like locomotion or basic interactions with the environment. These properties often follow the schema `G (F` ϕ`)`, stating that it must always be possible to reach a state where ϕ holds.

Irreversible effects Some effects may be irreversible, e. g. cutting ingredients. Usually this can be checked by static analysis of the model (there is no effect that reverses the previous effect), but sometimes a systematic verification can be advised to prevent future errors in the model.

4.4 Evaluation of activity recognition models

This section evaluates modelling errors in the activity recognition models of Section 3.4. I reviewed the models and derived properties for these models, which have then been verified.

I have considered all models of this thesis for verification. This evaluation shows that

- models can have modelling errors which restrict human behaviour, although the models are comparatively less complex and just a handful of properties have been checked, and
- the property classes proposed in Section 4.3 can be used to find modelling errors.

The models have been developed and reviewed by different sets of model designers, thus a bias of errors towards a single designer can be eliminated. However, all models have been developed in the same work group.

Out of the five models, three were eligible for deriving and checking properties: the office model, the meeting model and the single-recipe kitchen model. The other two models have not been used for this evaluation:

Table 4.1: Comparison of model complexity in terms of actions, boolean state variables, and state space size.

Model	Users	PDDL actions	Ground actions	State variables	States
Office	3	9	124	43	1,957,158
Meeting	3	10	84	78	66,587
Kitchen	1	25	97	60	146,552,922

- The indoor localisation model has an unrestricted transition model, i. e. at every state walking to every adjacent room is possible. Other context state variables do not exist in the model. It is therefore not possible to test any other properties, and deadlocks or livelocks do not exist by design.
- The multiple recipe kitchen model is too large to be handled by the PRISM model checker. To exclude bugs in the implementation as well as eliminate the fact that the techniques used by PRISM are not appropriate for this type of model, the model has also been tested with LoLA [211]. LoLA is a model checker based on a completely different set of techniques (Petri net analysis), but also fails to construct the internal representation of the model. As such, no further analysis could be done.

Each section briefly presents the properties and discusses the modelling errors found. Note that this is an empirical review of actual modelling errors, none of these errors has been artificially added to the model. Table 4.1 gives an overview of complexity metrics of each model. This shows that the models are not too complex (a few action schemas and at most 78 state variables) and can be handled by a single person.

4.4.1 Office domain

We checked the following properties for every single user u:

- no hands are free if and only if some object is held:
  `G ( !hands_free`$_u$ `<=> holds_water`$_u$ `| holds_paper`$_u$ `| holds_...`$_u$ `)`,
- when holding something it is possible to get the hands free:
  `G ( !hands_free`$_u$ `=> (F hands_free`$_u$`) )` and
- it is always possible to leave the room:
  `G ( !outside`$_u$ `=> (F outside`$_u$`) )`.

These properties are example of an invariant (covering different state variables), an intuitive assumption, and of ensuring repeatability of an action.

The second property was violated, no additional deadlocks or livelocks including multiple users have been found. Here, the agent could get some ground coffee for the coffee machine, refill the coffee machine, and get some additional ground coffee. After

getting additional ground coffee, the coffee machine still has resources and the agent is unable to release the ground coffee from his hand, allowing him only to walk between different locations without making any additional progress. A simple fix is to add an additional "drop" action, allowing to put resources back.

4.4.2 Meeting domain

We checked two domain-specific invariants and one unrepeatable action property for all three users a, b, c:

- at most one person is presenting:
  `G ( !(presenting`$_a$ `& presenting`$_b$`) & !(presenting`$_a$ `& presenting`$_c$`) & !(presenting`$_b$ `& presenting`$_c$`) )`,
- when presenting, all others are seated (for all *u1*, *u2*, *u3*):
  `G ( presenting`$_{u1}$ `=> (seated`$_{u2}$ `& seated`$_{u3}$`) )` and
- a person never presents twice (for all users u):
  `G ( presented`$_u$ `=> (G !presenting`$_u$`) )`.

All properties were satisfied, no livelock was present, but one deadlock could be found. The deadlock was caused by the way walking (e. g. to seats) has been implemented. To restrict the model, it was assumed uncommon to change seats and walk to different locations, so a person was only allowed to walk once before and between presentations. When the people enter the room they independently execute the action "walk to seat x". The deadlock happens if all persons execute the same action and go to the same seat, which is possible as the seat is still empty. When the persons arrive, only one can sit down, and the others could not move because walking was allowed only once.

Two options exist to remove the deadlock: reduce behaviour that leads to the deadlock or add behaviour to escape it. Reducing behaviour would require the persons to negotiate where to seat, which seems rather unnatural and complicates modelling even more. Allowing additional walking fixes the deadlock, but increases the state space to 49,765 states, making it rather costly.

4.4.3 Kitchen domain (single recipe)

The properties we checked included an invariant and two long-term causations:

- there is exactly one of the propositions true that zero, one, or two hands are free:
  `G ( hands-free-0 & !hands-free-1 & !hands-free-2 | !hands-free-0 & hands-free-1 & ... | ...)` ,
- when a person is not hungry he has cooked:
  `G ( !hungry => cooked )` and
- when eating, the person is seated and at the table:
  `G ( (hungry & X !hungry) => (seated & at-table) )`.

While the first two properties were true, the last indeed turned out to be false. The model allowed to eat without seating, it was sufficient to just stand at the table.

No deadlocks were present in the model, but a livelock could be found. As a precondition for the cook action, the spoon had to be in the hand. But in case the spoon was in the pot before cooking, it was impossible to remove the spoon, as the respective actions were too restrictive. In this case two fixes are possible. Either the actions could be relaxed to allow the spoon to be removed from the pot. Or the action could be restricted even more, preventing to put the spoon in the pot before finishing cooking. While the first option does not change the size of the state space, the second slightly decreases its size to 145 million states.

4.5 Discussion and Conclusion

Summary Model-based approaches to activity recognition require sound models. Model checking helps finding modelling errors and is therefore supposed to improve recognition rates. Improper models can not only have negative impact on recognition results, they can possibly lead to wrong, possibly harmful, decisions. This chapter presented a short case study of modelling errors found in activity recognition models. As a guide for practitioners and model developers, we identified important classes of relevant properties to identify modelling inconsistencies and errors, and how they may impact activity recognition.

Consequences of violated properties Some of these properties, when violated, may not always indicate a modelling error. Instead, they can provide useful insight to the problem domain and spot unexpected effects that may or may not be desirable. For instance, eating without sitting is possible, allowing such behaviour may therefore be desirable. Other violations may point out possible chances for reducing the model complexity and state space, such as limiting the model behaviour to the particular application domain.

It must be noted that unsatisfied properties, deadlocks, and livelocks not necessarily influence recognition results negatively. If some behaviour not present in the datasets is not modelled, recognition accuracy usually increases because less behaviour must be discriminated. Therefore these “errors” may sometimes be intended. Nonetheless, it must be kept in mind that this hinders re-usability of the models and can lead to unexpected problems, especially if they are not documented.

Thus, model checking can be used as a tool to find the trade-off between a small, restricted model and a large, unrestricted model.

Correcting the model Depending on the type of error, a few strategies can be used to correct the model, such that the properties are satisfied. Model checkers, including PRISM, can produce counterexamples for most violated properties. A counterexample is also called a witness path, i. e. a sequence of states which leads to a state where the property does not hold. The sequence of states (including the actions executed) can be analysed to pinpoint the fault in the model specification.

Resolving a deadlock depends on the model and the intended behaviour. Usually, action preconditions need to be relaxed such that they can be executed in deadlock states (e. g. walk to a different chair after realizing the first char is occupied). Sometimes, the path to deadlock states can be eliminated (e. g. avoiding early commits to a certain chair). Livelocks are more difficult to track, as usually no counterexamples can be given. One strategy is to split the goal formula and verify that parts of the goal can be reached.

Domain-specific invariants are usually easy to fix, as counterexamples are available and a certain action (or combination of actions) can be made responsible for producing an invalid state. Violated properties of long-term causations usually also produce a counterexample. If an action cannot be repeated, usually another action has to restore the precondition for this action; or the preconditions are too strict. Violated irreversible effects often have too much effects. However, when the irreversibility is context-dependent (i. e. sometimes, it is reversible), the reversing action's precondition may also be too relaxed for certain situations.

Applicability of model checking The kitchen domain model seems to be of a structure that cannot be handled (or: exploited) well by model checkers. If applications tend to support more complex use cases, models are expected to grow in the future. This can limit the applicability of model checking for larger real-life models. If these models arise more frequently, this may also open up a new research are for model checkers. If models grow in complexity, parts of the model might be re-used and such 'modules' might be verified independently.

Future work Future work includes an empirical analysis on the effects of violated properties. Also, more experiences and best practices can be collected to help model engineers with developing models, formulating properties and correcting models.

Currently, we employed an external model checking tool. Developing an integrated development environment for model specification, model simulation, model analysis and verification can ease and speed-up the development process.

5 Marginal filtering

Summary: This chapter presents the marginal filter algorithm. We analyse the traditional particle filter and discuss its limitations for categorical state spaces of human situation recognition (Thesis 5). The marginal filter algorithm is presented, including an analysis and discussion how it overcomes these limitations. Based on identified model properties, researchers and engineers can assess their models and choose an appropriate algorithm. As a measure to handle the large state spaces (Thesis 4), we also present a pruning strategy for the marginal filter.

Contribution: For the field of human situation recognition, this is the first evaluation that compares the approximation error and efficiency of the proposed algorithms. Previous evaluations focused on the accuracy wrt. annotations of the ground truth. I present a detailed analysis of the particle filter which model properties lead to a poor efficiency and favour the marginal filter. For the marginal filter, I also evaluate two different pruning strategies; previously, pruning (related to resampling for the particle filter) has never been considered before for the marginal filter. I also present a new, intuitive approach of deriving and proving an optimal pruning strategy. Additionally, this chapter introduces a novel sequential-prediction strategy for the marginal filter that improves its efficiency for multi-user models.

Parts of this chapter are based on

> *[148] Martin Nyolt, Frank Krüger, Kristina Yordanova, Albert Hein, Thomas Kirste:* Marginal filtering in large state spaces. *International Journal of Approximate Reasoning, 2015.*
>
> *[146] Martin Nyolt, Thomas Kirste:* On Resampling for Bayesian Filters in Discrete State Spaces. *International Conference on Tools with Artificial Intelligence (ICTAI), 2015. Best student paper award.*

Inference in state space models has been introduced in Section 3.2, in particular the particle filter has been presented in Section 3.2.2. As we will explore in this chapter, the particle filter does not perform well in terms of accuracy and efficiency for the human behaviour models of this thesis. For this reason, the marginal filter has been developed by Krüger [111, 113]. The marginal filter is another algorithm in the family of Sequential Monte Carlo (SMC) methods. It has been designed for the categorical state spaces of human behaviour models.

Recall from Section 3.2.2 that the particle filter maintains a sample-based belief state to approximate the filtering density $p(x_t \mid y_{1:t})$. The main operations of the particle filter are the three steps

- *prediction*, which computes the successor states for the particles from the belief state,
- *update*, which re-weights the particles according to the observation likelihood and
- *resample*, which resamples the belief state to mitigate particle degeneracy [60].

The first contribution of this chapter is an analysis of the particle filter for human behaviour models as described in Chapter 3. We show that the inefficiency is due to the way how samples are implicitly represented by particles. In degenerated cases, this representation leads to a representation of the sample weight by mere particle *counts*, not their weights. This is due to the way how prediction and resampling works. This analysis provides more evidence for Thesis 5:

Thesis 5. (repeated from page 59) *The traditional particle filter, as the most popular instance of SMC, is inefficient for inference in categorical state spaces. An efficient management of categorical samples increases the quality of the estimate.*

The marginal filter uses a different strategy for representing samples using particles. This is achieved by a different strategy for the *prediction* step, which is presented by Krüger et al. [113].

The second contribution of this chapter is an analysis of the marginal filter for human behaviour models. We discuss why the algorithm is more efficient for categorical state spaces than the particle filter, and is better suited for online-inference in human behaviour models. While the feasibility of the marginal filtering algorithm has been shown [111, 113], there is currently no detailed analysis *why* the marginal filter is more efficient in these state space models.

In addition, the previous evaluations of the marginal filter compared its performance based on the accuracy wrt. annotations of user behaviour. While this is eventually important for the applications, it is equally important to evaluate the actual approximation error (Definition 1) and efficiency (Definition 2) of the algorithm. This is particularly important for human behaviour models, which can have a very large state space, affecting the performance of the inference.

The third contribution of this chapter is the selection of an optimal pruning strategy to replace the *resampling* step of the particle filter. Previously, the selection of the pruning has neither been discussed nor evaluated for the marginal filter. We prove the efficiency of the algorithm and show that it is unbiased and optimal (in terms of errors in the weights of the particles). A pruning strategy is necessary, since the marginal filter explores all context states. However, for large state spaces, this becomes infeasible, which is stated in Thesis 4:

Thesis 4. (repeated from page 59) *Everyday human behaviour entails a large number of context states* $\mathbf{S}$*. Handling the large set of context states* $\mathbf{S}$ *is one factor that limits*

efficiency of Sequential Monte Carlo (SMC) methods. Automatically reducing the set of relevant context states increases the efficiency of SMC.

The rest of this chapter is structured as follows. First, we will give more evidence for the challenges presumed by Thesis 5 and Thesis 4 in Section 5.1. The theses are backed up by analysing the operation of the particle filter for human behaviour models. Second, Section 5.2 explores related work for improvements to the particle filter and how these techniques are applicable for this dissertation. Third, the marginal filter algorithm will be presented and discussed in Section 5.3. Then, the pruning strategy will be derived in Section 5.4, including proofs for its efficiency. Finally, Section 5.5 evaluates the marginal filter for the data-sets using real sensor data.

The results in this chapter include a total of 111,821 individual filtering runs over all models, filter algorithm variants, parametrisations (number of particles) and repetitions for randomised algorithms (50 repetitions each). These runs amount to a total of 2,160 CPU-hours for their computation. These computations have been carried out on a heterogeneous compute cluster.

5.1 Particle filter analysis

Particle filters have been successfully applied to numerous domains, including object tracking [62, 91, 99] and motion capturing [71, 173], robotics [199] and vehicles [200], surveillance [70, 144], estimating geophysical properties of rock formations [68] and atmospheric sciences [17]. All the models that are employed in these domains have a continuous state space in common: the location of objects, robots or persons on a map, the joint angles of humans or robots, nuclear magnetic responses and wheather conditions.

However, models for human situation recognition primarily have categorical state spaces, and to the best of my knowledge, there is currently no evaluation of a complex human behaviour model with categorical state spaces that uses a particle filter with great success. Indeed, Krüger et al. [113] have shown cases where the particle filter performs poorly, and present the marginal filter as an alternative. In this section, we want to extend this analysis of the particle filter and explain *why* the particle filter performs poorly for these kinds of models.

5.1.1 Methods

Approximation error of the actions Krüger et al. [113], as many other reports, evaluate only the *accuracy* of the particle filter wrt. the actual user actions. Throughout all following evaluations, we will use the *approximation error* wrt. the true filtering distribution (see Definition 1).

The annotation represents only the one action that is executing, but not the true filtering distribution. Indeed, the true filtering distribution does not even need to correctly predict the true action. Thus, using annotations is more suited for evaluating the model and sensors, but not the algorithm – in this thesis, we are concerned about the algorithmic efficiency. Furthermore, the annotations usually have only a few classes, as obtaining

correct annotations is very expensive (in the case of the single-recipe kitchen, there are 16 action classes), while there are many more actions and states in the model (kitchen: 97 actions). Thus, using all actions (as opposed to grouping them) gives a more exact picture of the error.

Unfortunately, it was not feasible to use the complete state space for comparison. This would lead to very large result files of at least several terabytes (in particular for each individual run of the exact filter, which has to be compared to the distributions of the particle filter) and weeks or months of computing the error measures. Therefore, the evaluations are restricted to the filtering distributions of the actions $p(A_i \mid y_{1:i})$. But an error measure of the action sequence is also a good indicator of the error of the full filtering distribution, as one can reconstruct the full distribution knowing only the initial state and the action sequence; conversely, knowing only the sequence of states and starting times allows one to reconstruct the action sequence (since all actions are deterministic).

Obtaining the true filtering distribution In order to compute the approximation error, we need to obtain the true filtering distribution $p(A_i \mid y_{1:i})$. Computing the true distribution is of course not viable in practice, as the models are very large (thus need much more memory as is usually available) and the computation time is too long and not suited for online inference (see Section 3.2).

Conceptually, the true filtering distribution can be obtained by a particle filter with infinitely many particles. This was implemented by a modified SMC algorithm which deterministically expanded all successor states (prediction step) and updated the weights according to the observations as usual (update step). No resampling step was performed, as this would introduce error.

The distributions were computed on a single computer with 512 GB of RAM. It is not to be expected that any model can completely fit into the available memory. However, as the initial state is fixed for all models, the required memory is very low for the first timestep and increases (exponentially) with every timestep (i. e. observation). The true distribution is saved up to the timestep where the model fitted into memory.

Thus, the approximation error is only computed for all timesteps that the exact filter could compute with the available memory. If different datasets resulted in different lengths of the distributions, we only use the common set of timesteps to ensure comparability between different runs of the same model.

Error measure We use the summed absolute error between the approximate filtering distribution $\widetilde{p}(A_i \mid y_{1:i})$ and the true filtering distribution $p(A_i \mid y_{1:i})$, which is defined as the sum of the absolute differences between the individual densities of all actions, for a specific timestep i:

$$e_i = \sum_{a \in \mathbf{A}} |\widetilde{p}(A_i = a \mid y_{1:i}) - p(A_i = a \mid y_{1:i})|$$

For example, assume the model has three actions a_1, a_2 and a_3, the exact filtering

distribution at time i has the densities (omitting the condition on the observations for brevity) $p(a_{1_i}) = 0.1$, $p(a_{2_i}) = 0.6$, $p(a_{3_i}) = 0.3$, and the approximation is $\widetilde{p}(a_{1_i}) = 0.2$, $\widetilde{p}(a_{2_i}) = 0.7$, $\widetilde{p}(a_{3_i}) = 0.1$, then the error $e_{i,u} = 0.4$. Note that this definition is for the case of a single user. If the model has multiple users, the error is computed for each user independently, as each user has its own distribution over actions he is executing.

In the evaluations, we are not interested in the individual errors per timestep and user, but we compare the total errors over the complete filtering run (up to the last timestep of the exact filter). To ensure that the errors are comparable between different models, we show the average error over all timesteps and users, i. e. total sum of errors divided by number of timesteps and users.

Alternative error measures The total summed error is a very generic measure, not particularly designed for probability distributions. The literature knows many different error measures, also measures which are specialised for comparing distributions. However, all other measures I considered are either not applicable or not well suited for the target of this evaluation.

Prominent measures for the difference of distributions are the *Kullback-Leibler divergence* (KLD) and the *Jensen-Shannon distance* (JSD), both measure how one distribution diverges from a second. When computing the divergence of $\widetilde{p}$ from p, they require that whenever $p(a) = 0$, $\widetilde{p}(a) = 0$, too. This condition is violated in practice due to a restricted precision of floating point numbers. It can happen that the true filtering distribution has a weight smaller than the smallest number which can be represented in a standard floating point number and thus effectively gets zero weight, whereas the resampling step of the particle filter can increase the weight of the same state. Using substitute numbers (i. e. a very small number which can be represented whenever $p(a) = 0$) is also not feasible, as this would simply assume wrong numbers and produce wrong results. Using arbitrary precision numbers for both filtering and evaluation was considered to be too much effort. This restriction indeed applies to all f-divergences, of which KLD and JSD are only two instances.

Another error measure is the *RMSE* (root-mean-square error), which is often used in favour of absolute error measures. However, it is sensitive to outliers and variability within the errors, therefore some authors suggest the use of absolute error measures [210]. I also follow this approach, as RMSE weights large errors (> 1) more than smaller errors, and the errors are expected to be less then 1 in most cases.

Having a large variability in the errors would favour the use of the *relative error* $1 - \widetilde{p}(a)/p(a)$. However, $p(a)$ can and will often be 0, and the relative error is not applicable, too.

Interpretation of the error The total summed error is always in the range from 0 to 2 (both for individual timesteps as well as the average error of the complete filtering distribution). A value of 0 means that the approximate distribution is equal to the exact distribution, and a value of 2 indicates a maximum error (there may be many distributions with a maximum error).

The following example shall give an intuition of how a small error can still have a large impact on the result. Assume there are 100 possible, similar ground actions which are to be distinguished in the filtering distribution. This number of actions can occur easily in our models of everyday human behaviour, for example think of 2 actions with similar observations (take or put), using 2 out of 6 similar objects (knife, fork, plate, ...) at two similar locations (sink, drawer). If all 100 actions are equally likely, each has a weight of 0.01. Assume that one action is slightly favoured by the model (e. g. more likely observations or slightly different duration) with weight 0.011, the other action have thus on average a weight of $(1 - 0.011)/99 \approx 0.00999$. If the approximation of the distribution assigns all 100 actions a weight of 0.01, then the error $e = (0.001 + 0.001/99 * 99) = 0.002$. In this example an error above 0.002 can result in a shift of the most likely action.

Note that all of the models used in this dissertation have a total number of actions less then 400, with not much more than 200 actions that can be confused as in the example above. Thus, I consider an approximation only as sufficiently good – for the use-case of everyday human behaviour recognition using the models of this dissertation – if its approximation error is significantly below 0.001.

Configurations For the evaluation and analysis of the particle filter, I varied three different factors, which compose the individual configurations that are to be compared.

Model The model is one of the models presented in Section 3.4, except for the office model. The office model has a small state space and is similar to the meeting model, and thus left out for brevity. Analysing the results in different models can help in identifying which model properties have a particular impact on the error.

Number of particles The number of particles is the main parametrisation of the particle filter and can be used as a trade-off between faster computation time or better approximations. Hence, the number of particles directly influences the efficiency of the particle filter. For the models, we use 10, 32, 100, 320, ..., 32,000 and 100,000 particles, where 32,000 particles is the maximum for the meeting model (no improvements could be observed in preliminary studies for more particles) and CMU (long computation time for more particles exceeding online inference). The localisation domain for 7 users used only up to 10,000 particles due to a long computation time (exceeding online inference) for more particles.

Each configuration was used to compute the filtering distributions for all available datasets.

Additionally, due to the various samplings in the PF (duration model, action selection, resampling), the particle filter is a randomised algorithm and we used 50 repetitions for each dataset and parametrisation.

5.1.2 Results

Exact filter distribution Table 5.1 shows the number of reached timesteps per model, including the maximum number of states that could be expanded and the median time

Table 5.1: Number of computed timesteps and reached states of the exact filtering results. 'Steps' shows for how many timesteps the exact filter could compute the filtering distribution. 'States' is the number of states X in the last complete timestep, i. e. how many states could be expanded before the filter ran out of memory. 'Time' is the average time minutes over all datasets.

		States		
Model	Steps	min	max	Time (minutes)
Kitchen	34	51,313,853	258,138,600	223
CMU kitchen	6	186,086,814	756,544,614	92
Meeting	35	34,905,958	34,905,958	91
Localisation (2 users)	33	28,645,060	99,592,656	128
Localisation (7 users)	3	230,496	40,986,000	85

required per dataset. The single-recipe kitchen model, meeting model and localisation model for 2 users can all be filtered for over 30 timesteps. I consider these all as sufficiently many timesteps for computing the approximation error. The CMU kitchen model and localisation model for 7 users are both much larger (they both allow many more actions per timestep) and can only be filtered for 6 and 3 timesteps, respectively. That the number of reached states is comparatively low for these two models is due to the high branching factor: During the prediction of the next timestep, the filter ran out of memory and probably reached far more than 500 million states. As in just 6 and 3 timesteps there are not so many actions executing, the errors have to be taken with a grain of salt and cannot be considered representative for a complete filtering run. However, as there are 86 datasets for the CMU kitchen model, I consider the overall error over all datasets as sufficiently representative. We will report the errors of the 7 user localisation model for completeness.

Particle filter results In most cases, the approximation error of the particle filter is above 0.01 and thus far above the target of 0.001. The error is below 0.01 only for runs with the most particles in the localisation (7 users) and meeting domain. Figure 5.1 shows the approximation error of the various filter runs for all models and configurations. The figure shows the time required for the filter runs on the x-axis.

As one can see from the figure, the approximation error does not scale well when increasing the number of particles for both Kitchen domains and the localisation (2 users) domain. That is: even though the number of particles (and thus computation time) is drastically increased, the approximation error improves only slightly. The particle filter is thus not very efficient for these models.

The better scaling wrt. more available particles for the meeting and localisation model (7 users) can be explained by the following:

The meeting model is the simplest of the models, with a small state space and few possible actions per user. The positions of the user are very informative of the rest of the state and the observations are mildly noisy position data for all users. Since particle

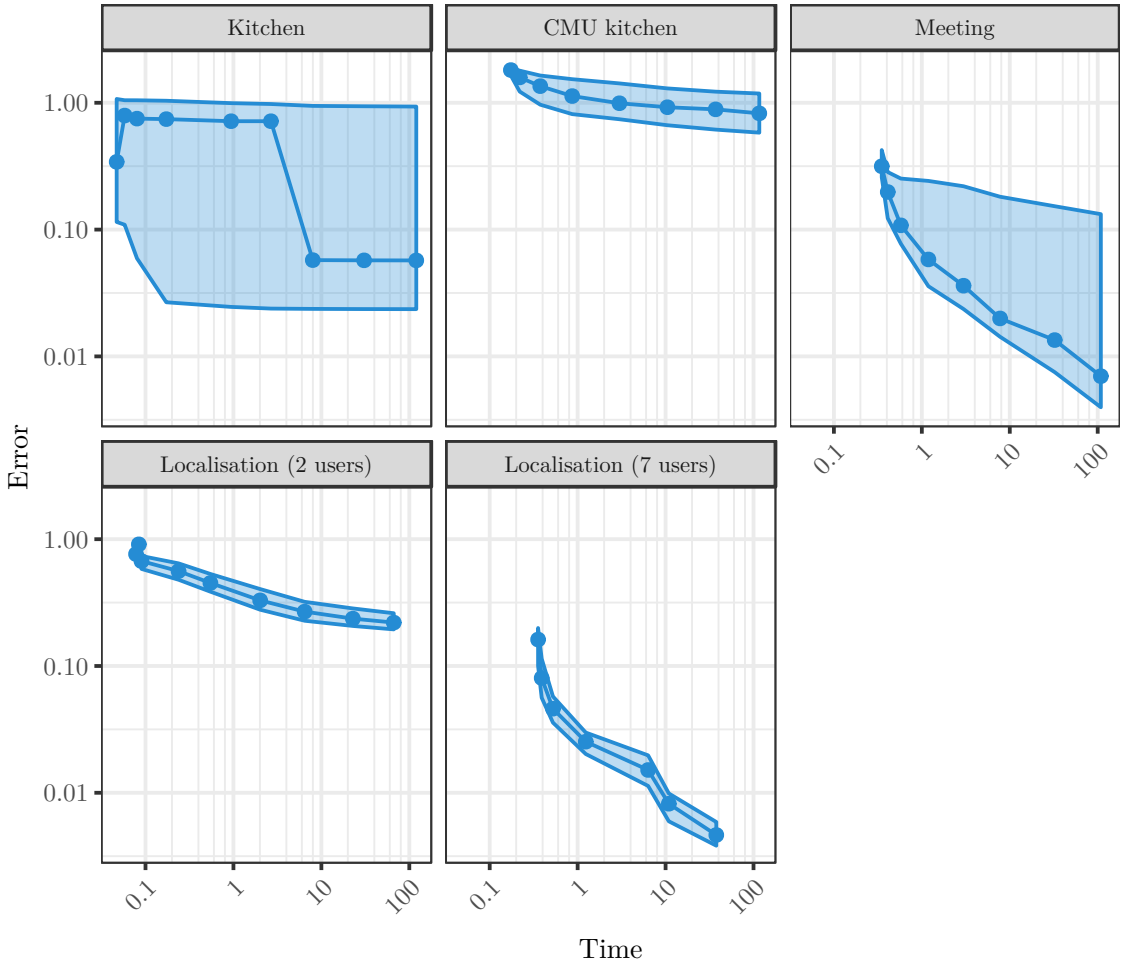


Figure 5.1: Overview of the individual particle filter results for the models. The x-axis denotes the computation time required per run, the y-axis represents the approximation error wrt. the true filtering distribution. The runs are grouped by the number of particles, each group represents all datasets and random repetitions for a given parametrisation. The dots are located at the median of the time and error, the ribbon visualises the 25% and 75% quantiles of the error. Note the logarithmic scale on both axes.

filters are known to be good estimators for motions and positions [53, 71, 91, 173], this efficiency is expected. (In contrast, the localisation model (2 users) has a much larger state space, more actions per user, and a less informative observation model, with no direct position data as observations.)

For the localisation model (7 users), the ground truth from the exact filter covers only 3 timesteps, where the state of the first timestep is known in the model. Thus, potential error can only be made in 2 timesteps, and the number of likely different states in the first few steps is relatively low.

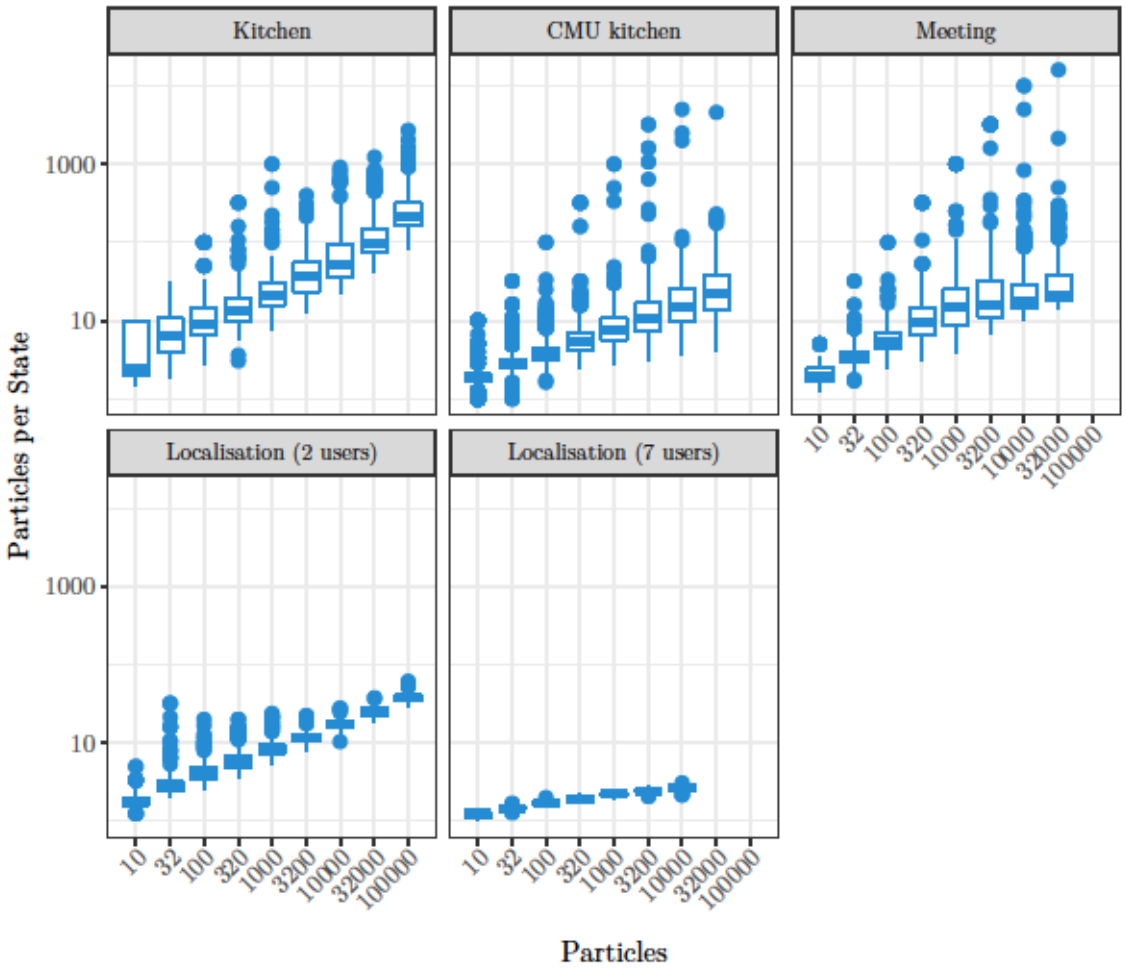


Figure 5.2: Number of particles representing the same state for the particle filter, plotted against different models and number of available particles. Both axes are logarithmic. The particle filter does not scale well with increased number of particles, as on average more particles represent identical states.

Particle utilisation From the data, we can also observe another deficiency of the particle filter: It does not efficiently utilise the particles for representing different states, and hence does not efficiently utilise the particles for the belief state $\widetilde{p}(x_i|y_{1:i})$. Figure 5.2 shows how many different particles represent identical states x. Not only is one state on average represented by multiple particles, but the number of particles per state is *increasing* with the number of available particles. This means that although more particles are used (increasing the computation time), the number of states represented does not increase by the same proportion.

Following the observations When evaluating the results of the runs, one can observe that not all particle filter runs could follow all observations until the end of the dataset.

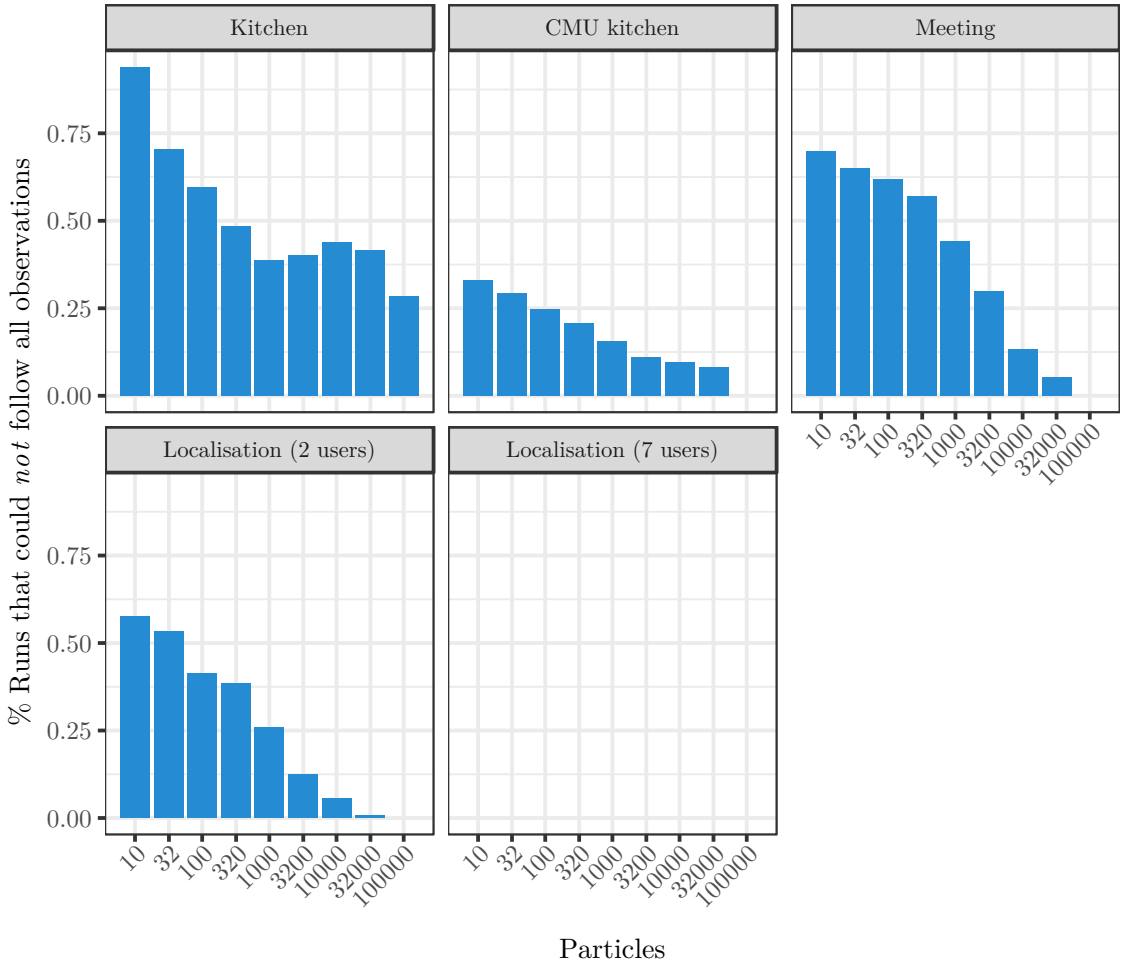


Figure 5.3: Percentage of runs that can follow all observation data. If a run could not follow the observation data, then at one point in time, all particles for this run had zero weight.

That is, at some point during the prediction or update steps, all particles got zero weight. This can happen if the observations are unlikely according to the transition model, and the prediction step does not sample any state that can be explained by the observations (i. e. $p(y_i \mid x_i) = 0$, at least to the precision of double floating point numbers). Figure 5.3 shows how many individual runs could not follow all observations.

With two exceptions for the kitchen model, the more particle are available, the more observations can be followed. The localisation model (7 users) is not experiencing any case where the observations could not be followed. This is because the observation model for the 7 users allows for many different states per observation.

There is no causality between the approximation error and this measure: the approximation error is only based on the first few timesteps where the exact distribution could be computed. Here, we show how many runs of the particle filter could not follow the *complete* observation sequence, irrespective whether the filtering distribution is exact or

not for these timesteps. There are, however, a few runs for 10-320 particles for both kitchen models where the particle filter could not even follow the observations of the exact filter.

5.1.3 Discussion

The case for the tails One might argue that it is in practice not necessary to keep an exact filtering distribution, especially because the distribution is peaked and most states have very low weight. It might be sufficient that only states with larger weight are represented, and errors in the tails are acceptable.

Experience shows that this is unfortunately not true for our models of human behaviour. There are two reasons for this:

Variability of human behaviour Keeping the tails, i. e. seemingly unlikely states, is important for human situation recognition. Humans can always behave unpredictable, which may make a now unlikely state a likely state in the future. Loosing these states early may not have a large impact on the approximation error (e. g. the state is only likely for a short time). But it can have a large impact on the filtering result when comparing to the ground truth (annotations of human behaviour) and potential decisions based on these results, e. g. automated assistance.

Unable to follow the observations This is a consequence of the variability of human behaviour. As is visible in Fig. 5.3, runs with less particles are prone to loosing the connection to the observations. Human behaviour models do have a high branching factor, i. e. there are many different actions possible for every state. Due to sampling in the prediction step, it is possible that no successor state is sampled that is able to explain the observations.

Having this case happen in practice should be avoided by all means. If at any point in time all particles have zero weight, the filter run has ended and cannot produce any more results. Except for re-starting the filter again (possibly with an uninformed initial state, which is likely to suffer from the same problems), there are no known universal options for recovery.

Sample impoverishment The results show that the particle filter provides no good approximation (even with 100,000 particles) and that the approximation does not scale well when increasing particles. This indicates that the particle filter is inherently inefficient for models with categorical state spaces, such as human behaviour models.

The data of Fig. 5.2 is the result of a phenomenon called sample impoverishment, which describes a loss of diversity of the particle states [7, 58]. Sample impoverishment occurs after the resampling step of the particle filter (which itself is necessary to avoid the particle degeneracy problem, see Section 3.2.2). Resampling duplicates particles, thus multiple particles represent the same state, resulting in a less utilisation of the particles. According to Bengtsson et al. [17], this effect increases with high-dimensional state spaces, as the number of particles must grow exponentially wrt. the number of

dimensions. In our case, each state variable is a single dimension. For instance, the Kitchen model has 60 variables and the CMU model has 130 variables. This is much more than usual filtering of a few objects in 3D-space.

It is noteworthy that the observed effect of sample impoverishment increases when more particles are available (as can be seen in Fig. 5.2). This is due to a combination of two factors:

Peaked filtering distribution We can observe that the filtering distribution for our models is usually peaked and has long tails with low density. The reasons are peaked (but still noisy) observation models and a large model which allows many transitions.

Sampling-based approximation The particle filter uses sampling for both the prediction step as well as the resampling step. This largely favours particles with a high weight.

In fact, for the particle filter to correctly work in categorical state spaces, it *needs* duplicated particles. Due to the sampling of a single successor state per particle, the particle filter needs multiple particles of the same state to correctly approximate the distribution of successor states. This effect increases with higher-dimensional state spaces, as possibly more successor states need to be sampled. Sampling multiple successor states instead of one, generating multiple particles and resampling down, only reduces this problem but does not solve it.

Categorical state spaces The results indicate that the particle filter is not well suited for state spaces with categorical states, as they appear in models of human behaviour. The operation of the particle filter, in particular sampling and resampling, works best with continuous states.

This shall be demonstrated using a simple example of a continuous state space: a one-dimensional motion. Let the state $x \in$, the transition model advances x by one with some noise: $p(x_i \mid x_{i-1})\ \mathcal{N}(x_{i-1} + 1; 1)$. Let the initial state $x_0 = 0$. If the filter samples $x_1 = 1.1$ instead of the 'true' $x_1 = 1$, then it can still likely sample $x_2 \approx 2$ in the next timestep. Thus, errors do not add up and it can be said that small approximation errors will be self-corrected easily. This is because:

- The noise in the transition model has some expected value, to which the particle filter can converge over time.
- The variance of the noise is large enough to allow the filter to converge to this expected value with a finite amount of particles. Arulampalam et al. [7, p. 180] note that a low noise of the transition model leads to severe sample impoverishment, thus having a high variance is important.

On the other hand, it is usually not possible to recover from approximation errors in categorical models. This is because:

- The noise of a categorical transition model does not have an expected value (as categorical distributions do not have one).

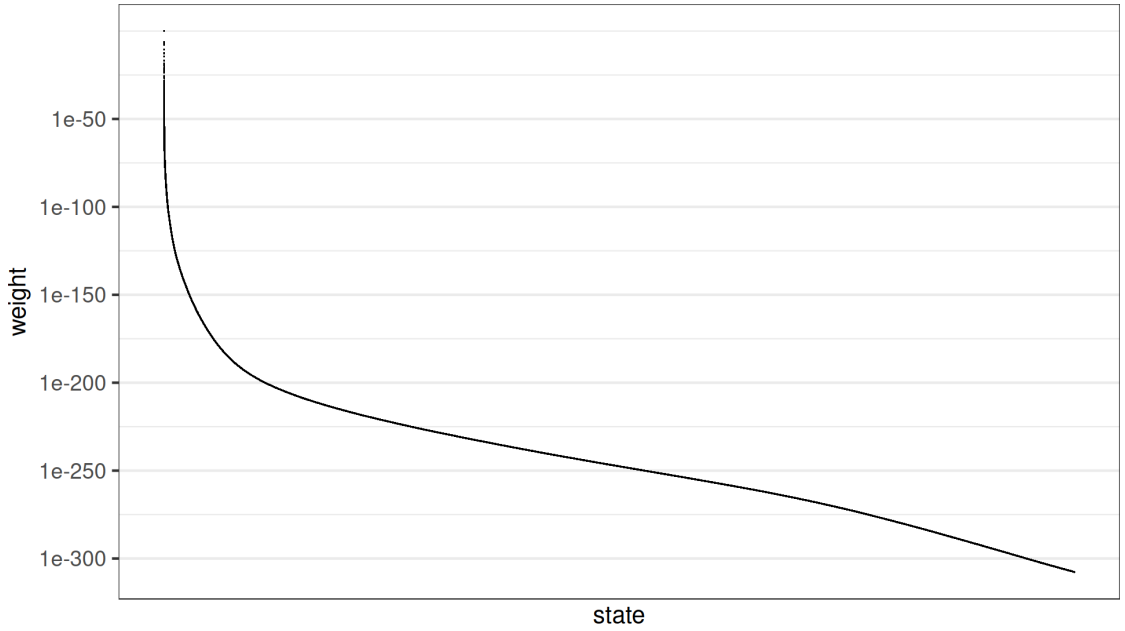


Figure 5.4: Exact density of the filter distribution for the single-recipe kitchen model, first dataset, at timestep $i = 23$. This distribution features 13,052,219 distinct states. The states are ordered in decreasing weight, showing that this is a very peaked distribution with a long tail.

- Although the variability in behaviour is usually large, it is not high in the sense that it allows the filter to recover from errors. The 'noise' in these models is usually *randomness* in behaviour, not noise due to physical processes. Categorical states of human behaviour models have a complex causal structure. Two states with just a single differing predicate can have no common successor states. Likewise, it is usually not possible to add additional true noise to the states, as this would invalidate causality and lead to impossible or even nonsensical states.

Thus, categorical models do not have two important properties that are required for optimal operation of the particle filter. This results in a set of very distinct particles, eventually leading to the effect of sample impoverishment, where only very few states (and eventually only one) are represented by the particles.

Sampling in categorical states In summary, sample impoverishment and the high-dimensional state spaces lead to the effect that only very few states are represented by particles. Thus, large portions of the state space are essentially unknown to the particle filter. In fact, the following two observations indicate that sampling, the core mechanics of the particle filter, leads to poor approximations of categorical distributions.

Sampling from peaked distributions Non-flat distributions show a high variance in the probability of the weights. Figure 5.4 shows the exact filtering distribution over all categorical states from one of the examples. This distribution is very peaked.

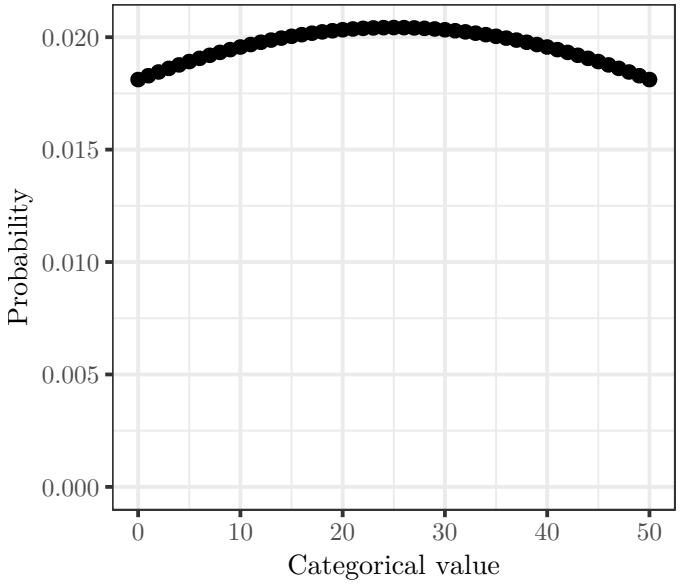


Figure 5.5: Probability function for the exemplary categorical distribution, here with 51 different values. The probabilities are similar to the densities of a Normal distribution in the range from -0.5 to 0.5.

Approximating the full distribution by sampling requires particles in the order of magnitude of the ratio of the probabilities, even if sampling from the true distribution. In the example, the quotient of the most probable state to the second-most probable state is approximately 10^6, which means that sampling 10^6 times from this distribution is expected to result only once in sampling the second state, and $10^6 - 1$ times the most probable state. For the example, approximating the full distribution would require approximately 10^{300} samples.

The particle filter samples at two main steps: sampling a successor state, and resampling all particles. Both steps suffer from this poor sampling, leading to the two main problems particle degeneracy and sample impoverishment of the particle filter.

Sampling from flat distributions When the distribution is flat, the samples are also more evenly distributed and the shape of the distribution can be approximated more closely (compare Fig. 5.5 and Fig. 5.6). In these cases, however, the mode of the distribution cannot be approximated correctly, even if sampling from the true distribution.

Figure 5.5 shows an exemplary flat distribution with 51 different states; Figure 5.6 shows a distribution obtained from sampling from this distribution 10,000 times. As we can see, although the shape is approximated quite well (every state got sampled many times), the mode is not correctly estimated. In fact, the mode is only correctly estimated 60 out of 1,000 times. The performance deteriorates exponentially with increasing number of states, see Fig. 5.7.

In summary, the results support Thesis 5, motivating the need for a more efficient SMC implementation for human behaviour models:

Thesis 5. (repeated from page 59) *The traditional particle filter, as the most popular instance of SMC, is inefficient for inference in categorical state spaces. An efficient management of categorical samples increases the quality of the estimate.*

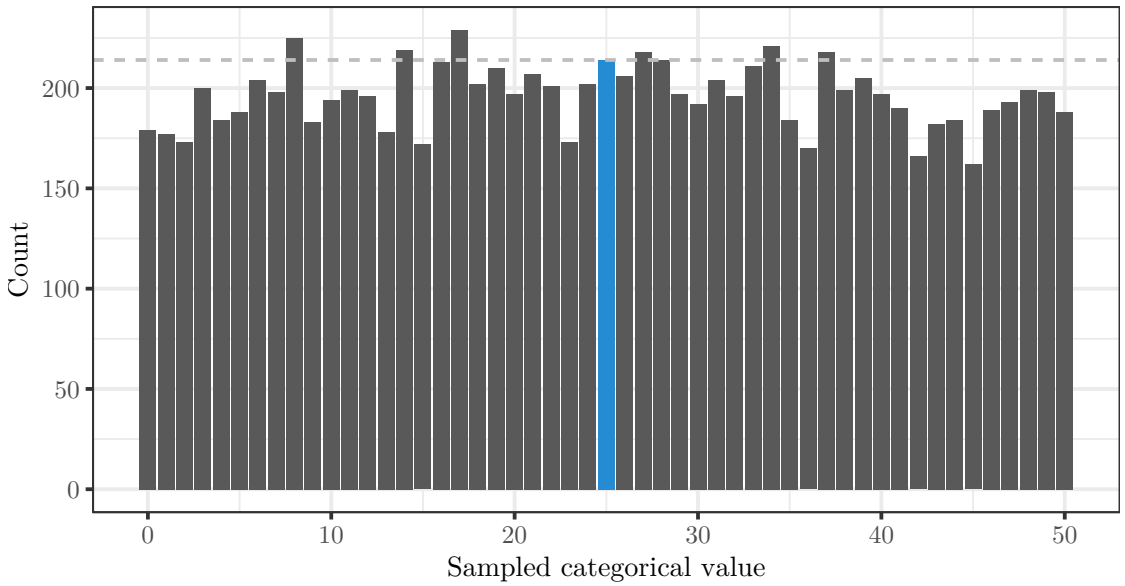


Figure 5.6: Histogram of 10,000 samples of the exemplary categorical distribution with 51 different values. The blue bar indicates the true mode.

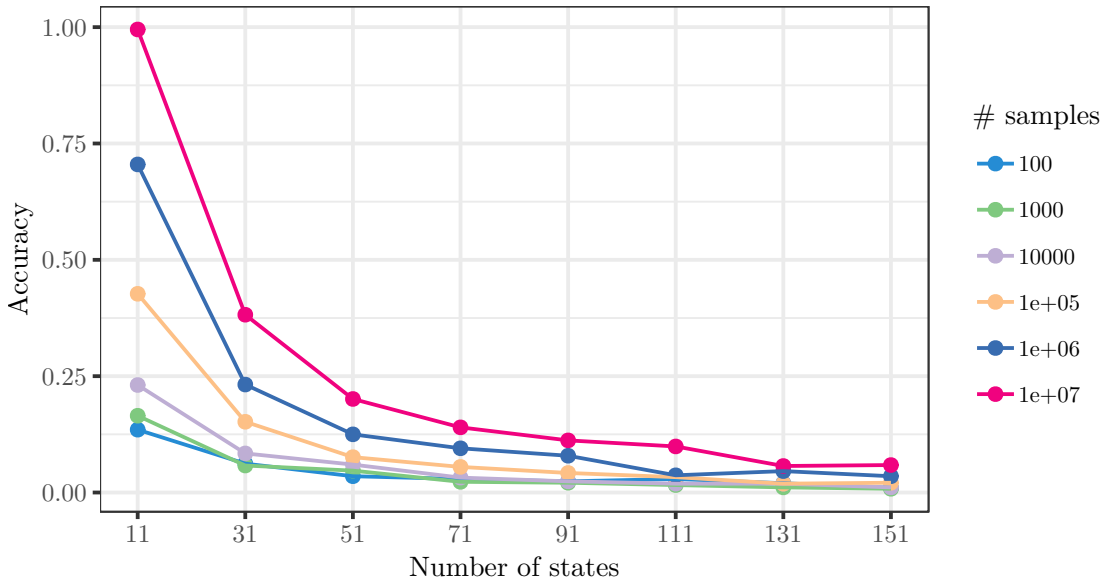


Figure 5.7: Accuracy of estimating the mode by sampling from a flat categorical distribution, with increasing number of states. For each configuration, 1000 samples have been drawn; the accuracy is the ratio of times the mode has been correctly estimated. The number of intervals increases linearly, the number of samples exponentially, yet the accuracy degrades exponentially.

5.2 Related work

Due to the history of the Particle Filter, the majority of the work on improvements is only applicable to continuous state spaces. This includes, for example, the annealed particle filter [54] (that adds Gaussian noise to states), combinatorial resampling [62] (requires that sub-states may be freely exchanged), Sequential quasi Monte-Carlo [75] (defined only on continuous state spaces), and deterministic resampling [125] (requires a distance measure to be defined on the states).

There are also techniques directly target at circumventing the problem of sample impoverishment (see Arulampalam et al. [7] and Doucet [58]), but these are not applicable to categorical state spaces. The most common idea is to add some form of noise to the state, which again is not possible to categorical states. Daum and Huang [52] propose a new filtering scheme that is not based on sampling (thus avoiding sampling issues be design), but on particle flow; this work also depends on continuous state spaces.

Klaas et al. [106] propose the marginal particle filter (MPF). The underlying idea is that the particle filter actually computes the joint distribution $p(x_{1:i} \mid y_{1:i})$ by sampling state sequences, but usually only the marginal filter distribution $p(x_i \mid y_{1:i})$ is needed. This is also true in our case. Klaas et al. thus propose an algorithm that marginalises over all past states $x_{1:i-1}$ before sampling successor states. This avoids the particle degeneracy problem, which in turn avoids the need for resampling and thus prevents sample impoverishment. Their approach is based on the case where marginalisation can be done analytically for continuous state spaces.

A similar idea has been used by Shi et al. [190], who propose an algorithm called D-Condensation which marginalises over past states. In their setting, they are explicitly using a categorical state space. Marginalisation is achieved by merging identical states, and instead of *sampling* from the successor states, the *expand* all successor states. After the update step, they use a beam-search pruning and keep only the N particles with most weight to prevent an unlimited grow of the particle count. Their evaluation is based on a single real-world application (glucose monitor calibration). The state space is relatively small (20,000), and they do not evaluate the approximation error but only the recognition accuracy wrt. annotated human behaviour.

In summary, there is not much work on SMC methods for categorical states. As Chapter 2 has shown, most literature in the situation recognition community is focussing on applications and recognition results, less on algorithms, efficiency and approximation issues. Those literature that focuses on algorithmic results are using continuous state space models.

5.3 Marginal filter algorithm

This section presents the marginal filter algorithm (MF) in detail, which uses a similar idea as the D-Condensation algorithm [190]. The MF has been specifically developed for discrete state spaces that occur in human behaviour models [113]. It fits into the general framework of SMC methods in that it represents states by weighted particles, and does

prediction and updates of the weights according to Algorithm 1 (page 74).

The basic idea of the marginal filter is to prevent sample impoverishment by avoiding the need for sampling at all, and computing the weights *exactly*, but still maintaining a limited belief state. This is motivated by the observations from Section 5.1.3, where we have shown that sampling from (categorical) distributions leads to poor estimates and requires a large amount of particles. Computing the weights exactly is achieved by marginalising over all past states (in the same vein as Klaas et al. [106]). Thus, every state x is represented by exactly one particle. Similar to Shi et al. [190], this marginalisation is done by merging identical states. This is possible since we have discrete, categorical states where integration can be done by summation.

The marginal filter uses a set of at most N particles $\langle x_i^{(n)}, w_i^{(n)} \rangle$ as its belief state. As one state is represented by exactly on particle, the belief state can contain less than N particles if there are only so many different states (this happens most likely in the first few timesteps). In contrast, the particle filter uses a belief state of *exactly* N particles.

After comparing the marginal filter to the related work, we will present the three functions INITIALIZE, PREDICT, and IMPROVESAMPLES of the SMC framework from Algorithm 1.

Related work The marginal filter has been first introduced by Krüger [111, 113]. Krüger also compares the marginal filter to the marginal particle filter (MPF) by Klaas et al. [106] and D-Condensation by Shi et al. [190]. Here, we briefly compare the marginal filter to the MPF and D-Condensation with respect to the focus of this dissertation.

The idea of the MPF is to not use a proposal distribution $q(x_i^{(n)} \mid x_{i-1}^{(n)})$ for every single particle $x_{i-1}^{(n)}$ of the previous distribution, but to sample from a marginalised proposal distribution by summing over all previous particles [106]. This avoids sample impoverishment, as no particle duplications are required to sample a diverse set of successor state. The MPF, however, is described only for continuous state spaces and requires that the marginalisation can be computed analytically. The marginal filter as presented in this dissertation is the application of the same idea to discrete state spaces.

The core of the D-Condensation algorithm is the same as the marginal filter or the MPF: successor states are sampled from the marginalisation over all past states [190]. The D-Condensation differs from the particle filter only by how it is used and embedded with the model. Shi et al. do not use a causal transition model $p(s_i \mid s_{i-1})$, but use a model called Propagation Networks. These P-Nets represent every action as a random variable; context states are not modelled. Instead, multiple actions can be active at the same time. Thus, the key difference between the marginal filter and D-Condensation is the underlying model structure and how the semantics of the underlying model need to be represented in the filter algorithm. The algorithmic ideas on a higher level of abstraction are identical.

Algorithm 7 The initialisation step of the marginal filter algorithm.

```
1: function INITIALIZE
2:     ▷ Enumerate all states from the initial distribution of Eq. (3.12).
3:     p ← ∅
4:     for all x_1 ∈ X : p(x_1) > 0 do        ▷ Loop over all possible initial states.
5:         x_1^(n) ← x                        ▷ Use this state as the particle's state.
6:         w_1^(n) ← p(x)                     ▷ Particle weight equals the density of the state.
7:         p_1^(n) ← ⟨x_1^(n), w_1^(n)⟩       ▷ Create particle from state and weight.
8:         p ← p ∪ {p_1^(n)}
9:     end for
10:    return p
11: end function
```

5.3.1 Initialize

The INITIALIZE function (Algorithm 7) creates one particle for each initial state x_1 (with non-zero support $p(x_1) > 0$) of the initial state distribution. In contrast to the particle filter, which creates exactly N particles and samples from the initial distribution (Algorithm 2), the MF creates only as many particles as there are initial states. Usually, there are only a few different possible initial states (except for the CMU kitchen model with 18 states, the models used here have only one initial state). By not sampling N times from the distribution, the MF ensures that each initial state is represented by exactly one particle. If the number of initial states is larger than N, the MF guarantees that the number of particles does not exceed N after the first timestep (this will be discussed shortly).

Implementation note The initial state only influences the particles' state s_0 at timestep 0, but never influences the transitions $p(s_i \mid x_{i-1})$ of the particles given the previous state. Once particles are in the same $\langle S, A, T, G \rangle$ state (even if the had different initial states I at an earlier timestep), they only differ in the weights by a constant factor for different initial states. Thus, weights of the different initial states can be saved in a single particle of the same $\langle S, A, T, G \rangle$ state. For most models, where the initial state has only little influence on the long-term state trajectories (e. g. the initial position of the user, but the user can walk to all locations from all initial positions), this makes inferring the initial state very cheap. Similarly, the goal G only influences the sample probability of the actions and thus the *weights* of the particles, but not their state. Thus, each particle can store a set of weighted goals, instead of one particle for each state/goal combination.

For the sake of brevity and clarity, we will thus consider the initial state and goals as static throughout the rest of this chapter.

5.3.2 Prediction

The PREDICTION function is the core of the marginal filter and presented in Algorithm 8. It computes the prediction density $p(x_i \mid y_{1:i-1})$ and ensures that during prediction no states are represented by multiple particles. In the following, we will first describe the principles of the function before explaining it in detail.

One particle per state Representing every state by exactly one particle prevents the problem sample impoverishment. Note that having two particles representing the same state x_i can occur only when two different states $x_{i-1}^{(1)}, x_{i-1}^{(2)}$ of the previous timestep lead to the same state x_i. This can be prevented by marginalising over all past states. Since we are in a discrete state space, marginalising is achieved in practice by adding the weights of two particles with identical states, leading to a single particle. In the rest of this chapter, we call this process *merging* of particles.

Expand all states Remember that the particle filter *samples* successor states – only one successor state for every particle. This was only possible since each state was represented by multiple particles, and thus different possible successor states could be sampled. In contrast, the marginal filter has only one particle for every state. If it would sample one successor state, the number of distinct states and particles would decrease over time (the number of distinct states can never increase, but two successor states can be merged).

Therefore, the prediction of the MF works by expanding *all* possible successor states (by executing all possible actions) for all current particles. Figure 5.8 visualises the approach in comparison to the particle filter. This expansion will usually increase the number of particles during prediction, increasing it to $M > N$ particles. Thus, the IMPROVESAMPLES function needs to reduce the belief state to contain only N particles (this will be discussed in the next section). We will also denote new particles at timestep i with $p_i^{(m)}$, i. e. with index (m), to indicate that the index of the new particles generally do not coincide with the index (n) of their predecessor particle $p_{i-1}^{(n)}$.

Algorithm in detail The prediction in Algorithm 8 is a bit more complex than the prediction of the particle filter (Algorithm 2, page 76), as more cases have to be considered.

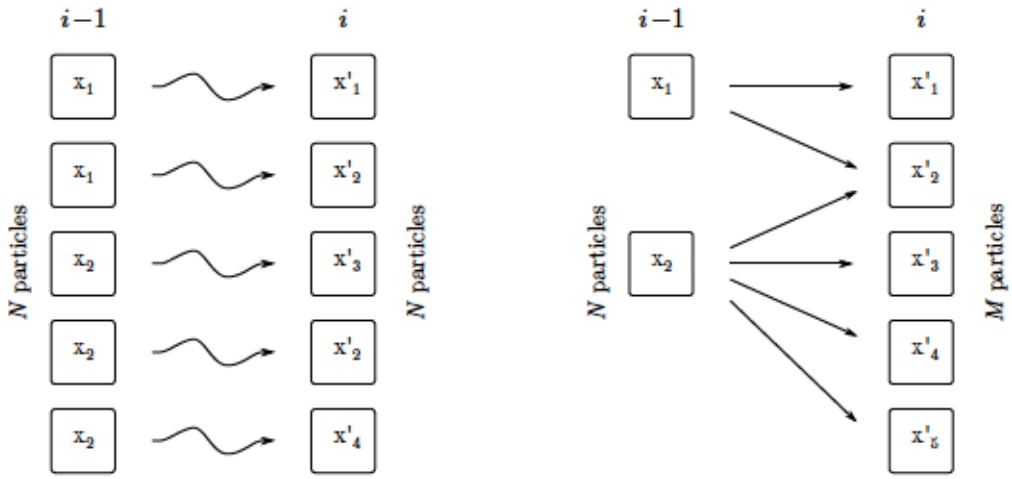
The result of the prediction step is a new set of particles with all successor states of the current particles. Line 2 thus starts with creating a new, empty set $\mathbf{p}_i$ of particles; these are returned in line 30.

Line 4 computes the probability $f_i^{(n)}$ that the current action stops, based on the termination model (3.4). There are two non-exclusive cases, explained in more detail below:

- $f_i^{(n)} < 1$, i. e. the current action can continue and the state does not change. This adds the current particle to the set $\mathbf{p}_i$.
- $f_i^{(n)} > 0$, i. e. the current action may stop and successor states have to be computed. This adds k particles (where k is the number of successor states) to the set $\mathbf{p}_i$.

Algorithm 8 The prediction step and core of the marginal filter algorithm. We always assume that $x_i^{(n)}$ is defined to be $\langle g_i^{(n)}, a_i^{(n)}, s_i^{(n)}, t_i^{(n)} \rangle$.

function PREDICT($\mathbf{p}_{i-1}$)
  $\mathbf{p}_i \leftarrow \emptyset$
  for all $p_{i-1}^{(n)} = \langle x_{i-1}^{(n)}, w_{i-1}^{(n)} \rangle \in \mathbf{p}_{i-1}$ **do**
    $f_i^{(n)} \leftarrow p(f_i \mid a_{i-1}^{(n)}, \mathcal{T}_{i-1}, \mathcal{T}_i, t_{i-1}^{(n)})$ ▷ probability that the action stops
    if $f_i^{(n)} < 1$ **then**
      $x_i^{(m)} \leftarrow x_{i-1}^{(n)}$ ▷ Action may continue.
      $w_i^{(m)} \leftarrow w_{i-1}^{(n)} \cdot (1 - f_i^{(n)})$ ▷ probability that this action continues
      $\mathbf{p}_i \leftarrow \mathbf{p}_i \cup \{\langle x_i^{(m)}, w_i^{(m)} \rangle\}$
    end if
    if $f_i^{(n)} > 0$ **then** ▷ Action may stop, new states are possible.
      ▷ Assume $g_i^{(m)}$ has already been determined.
      ▷ Create all possible successor states for all applicable actions.
      for all $a_i^{(m)} \in \mathbf{A} : p(a_i^{(m)} \mid f_i = true, s_{i-1}^{(n)}, g_i^{(m)}) > 0$ **do**
        ▷ Execute a new action.
        $s_i^{(m)} \leftarrow a_i^{(m)}(s_{i-1}^{(n)})$ ▷ Set the new state.
        $t_i^{(m)} \leftarrow \mathcal{T}_i$ ▷ Set the new starting time.
        ▷ Compute the new weight based on the product
        ▷ inside the integral of Eq. (3.13).
        $w_i^{(m)} \leftarrow w_{i-1}^{(n)} \cdot p(x_i^{(m)} \mid x_{i-1}^{(n)}, f_i = true, \mathcal{T}) \cdot f_i^{(n)}$
        if $w_i^{(m')} \leftarrow$ LOOKUP($\mathbf{p}_i$, $x_i^{(m)}$) **then**
          ▷ Merge particles with identical states by adding their weights.
          $w'^{(m')}_i \leftarrow w_i^{(m)} + w_i^{(m')}$
          $\mathbf{p}_i \leftarrow \mathbf{p}_i \setminus \{\langle x_i^{(m)}, w_i^{(m')} \rangle\} \cup \{\langle x_i^{(m)}, w'^{(m')}_i \rangle\}$ ▷ Update the set.
        else
          $\mathbf{p}_i \leftarrow \mathbf{p}_i \cup \{\langle x_i^{(m)}, w_i^{(m)} \rangle\}$ ▷ Add new particle to set.
        end if
      end for
    end if
  end for
  return $\mathbf{p}_i$
end function



(a) The particle filter samples one successor state for every particle. Note that five particles represent the two distinct states at timestep $i-1$, and four distinct states at timestep i (state x'_2 has been sampled twice).

(b) The marginal filter expands all possible successor states for every particle. Note how the successor state x'_2 occurs only once in the resulting timestep i.

Figure 5.8: Comparison of the prediction steps between the particle filter and the marginal filter. In this example, the timestep $i-1$ has two states. State x_1 has two successor states and x_2 has four successor states, where x'_2 is a common successor state. The maximum number of particles is set to $N = 5$ for this example. The marginal filter does not need all N particles to represent both states. After prediction, the marginal filter has merged the successor state x'_2 originating from both particles. Note that the marginal filter is also able to represent one additional state at timestep i in comparison to the particle filter.

Consequentially, if $0 < f_i^{(n)} < 1$ then $k+1$ particles are added to $\mathbf{p}_i$.

Case $f_i^{(n)} < 1$ The block starting at line 5 handles the case that the action may continue, i. e. the probability that the action finishes is less then 1. In this case, the state of the particle stays unchanged (line 6) and only the weight needs to be updated (line 7). The old weight is multiplied by the probability that the action may continue, i. e. $1 - f_i^{(n)}$ (this implements the condition $F_i = \mathit{false}$ of Equations (3.9)–(3.11)). Note that if $f_i^{(n)} = 0$, then the particle's state *and* weight is left unchanged and no successor states are computed, which is the correct behaviour.

Case $f_i^{(n)} > 0$ The block starting at line 10 generates new particles for all successor states of $x_{i-1}^{(n)}$. It starts by computing all applicable actions in line 13 (see Equations (3.5) and (3.6)) and iterating over them. For each applicable actions, the successor state is computed by applying the action to the old state (line 15). As the state has now changed and a new action has started executing, the starting time for this particle is set to the current timestep in line 16. Finally, the weight of the new particle is computed in line 19 according to the product inside the integral of (3.13). This includes the

Algorithm 9 The pruning step of the marginal filter algorithm (simple beam-search). This implements the ImproveSamples function of the SMC method and ensures that after every timestep, no more than N particles are used.

```
1: function ImproveSamples(p_i)
2:     ▷ Prune the particles to keep at most N particles.
3:     if |p_i| > N then
4:         return BeamSearch(p_i, N)
5:     end if
6: end function
```

probability of selecting the action ((3.5)) as well as finishing the previous action ($f_i^{(n)}$, which corresponds to the condition $F_i = \mathit{true}$ of Equations (3.5)–(3.8)).

In principle, this finishes the creation of new particles and computing their weight. Basically, the particles are computed similar as in the particle filter (Algorithm 2), but instead of sampling, the weight are computed by probability that this state would be sampled. However, some particles might have identical states, as two states can have common successor states. These particles need to be merged; this is handled in the block starting form line 20. First, the current set of particles is search for another particle with the same state (this is done by the Lookup function)[1]. If such a particle exists (there can be at most one, by induction), then the weights are added in line 22; this correspond to a marginalisation over past states. Line 23 then updates the weight of the particle that is already in the set; the new particle is discarded. If no other particle exists in the set with the same state, then the new particle is simply added to this set in line 25.

5.3.3 Pruning

The prediction step usually produces a set of $M > N$ particles, as every particle can have multiple successor states. In order to limit the computational complexity and memory requirements, it must be ensured that the belief state contains at most N particles after every timestep. The marginal filter does not dictate a specific strategy to use – two algorithms will be presented here. The first straightforward algorithm is beam-search, which is also used by Shi et al. [190]. Beam-search selects the N particles with the most weight, and discards all particles with lower weight. This is simple to implement and works well in most situations. An improved algorithm will be presented in the next section.

5.3.4 Discussion

Expanding all states Models of human behaviour usually have a large number of applicable actions per state. Thus, the temporary number of particles can be drastically increased wrt. N, which can have a large impact on the performance. One could also

[1] In practice, this is efficiently implemented by managing a hash map with states as keys and corresponding particles as values.

(adaptively) sample multiple successor states for every particle (until N reached). However, this was not considered for the evaluation for the following reasons:

- in practice, this will likely lead to bad sampling of successor states, often still only one sample per particle (if the belief state contains N distinct states, then on average only one successor state can be sampled to create N new states)
- a multi-sampling approach can be regarded as the middle ground between the marginal filter and particle filter; expanding all states was chosen to explore how efficient this approach is at this end of the spectrum

Exact filtering The marginal filter is computing the true filtering distribution as long as the number of states within the distribution is $\leq N$. This is because the weights of the particles are exactly their densities, and a change of the distribution occurs only during pruning. This exact filtering up to N particles demonstrates the superior efficiency of the MF compared to the particle filter: Suppose a model with three initial states of equal probability. The particle filter would sample N times from the initial distribution. If N is not divisible by three (e. g. a power of ten, which is a common particle count), then the initial distribution will never be represented exactly. In contrast, the belief state of the MF will contain exactly three particles, each with a weight of $^1/_3$.

Pruning after the update step Note that the pruning operation is implemented in the ImproveSamples function, which is executed *after* the update step. It would also be possible to prune the particles to N immediately at the end of the prediction step, thus keeping the high memory requirements to a minimum. However, computing the observation likelihood is usually fast, in particular in comparison to the prediction step. Thus, the marginal filter applies the update step to all predicted particles. This results in a much more accurate belief state with only a slight overhead.

Algorithmic complexity The algorithmic complexity of the marginal filter is $\mathcal{O}(N\,|\mathbf{A}|)$, i. e. identical to the particle filter (see Section 3.2.2). The most complex operation is the inner loop at line 13, where for every of the N particles all possible actions are evaluated. The lookup operation and weight update can be done in constant time using a hash map. All other operations are also constant time.

The pruning strategy should also be chosen such that it is linear in N. This is true for the beam search strategy, which can be implemented by a selection algorithm in linear time [139].

Multi-user models So far, the marginal filter has only been discussed with a single user executing the actions sequentially. However, models may contain a set $\mathbf{U}$ of independent users, each executing actions parallel to the other users. The adaptations to the initialisation step is straightforward. No changes need to be made to the update and pruning steps, as these only make computations on the particle weights, but not their states. However, there are (at least) two different strategies, how concurrent users can be

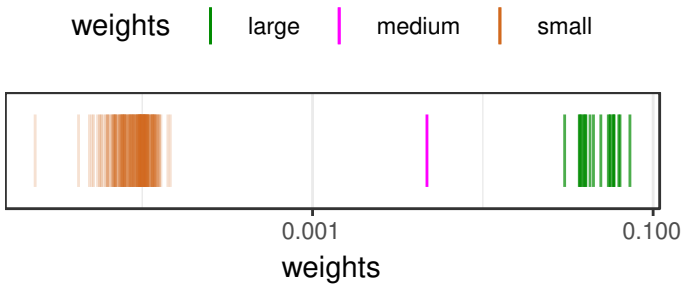


Figure 5.9: Distribution of the particle weights in the running example. One can clearly distinguish three different "populations" of the particles.

handled in the prediction. For this discussion, let $\mathbf{A}_u$ denote the set of all actions that can be executed for user $u \in U$.

Expand all possible action combinations – single prediction When seeing the model transition as an opaque process, the set of actions A is the compound set of all possible concurrent user actions $A = A_{u_1} \times A_{u_2} \times \cdot \times A_{u_{|\mathbf{U}|}}$. That is, we can use the prediction step unchanged. This means, however, that the complexity is now exponential in the number of users, i. e. $\mathcal{O}(N\,|\mathbf{A}|^{|\mathbf{U}|})$. The space complexity grows similarly, so that even in practice a very large number of particles can be generated during the prediction.

Prune between individual user predictions – sequential prediction To counter the time and space complexity, it is possible to limit the number of particles by executing prediction-update-prune cycles after processing each user. The update step within each user's prediction can only be applied if the observation model allows to relate certain observations to particular users. Any observations that cannot be assigned to a specific user must be handled after the prediction within the usual update step. This idea is shown Algorithm 10. Pruning after every user reduces the complexity back to $\mathcal{O}(N\,|\mathbf{A}|)$, but at the cost of additional approximations made during the prediction.

Note that the same problem occurs also in the particle filter when sampling the set of actions for all users. Our implementation of the particle filter always uses the second strategy and sequentially samples the actions for the users.

5.4 The pruning step

This section will discuss three different pruning strategies: classical resampling known from the particle filter, beam search and the resampling strategy proposed by Fearnhead and Clifford [68]. We will call the latter resampling *Fearnhead-Clifford resampling* by the name of its authors.

Algorithm 10 Deviation of the SMC framework for sequential prediction. The particles are pruned after expanding the states for every user $u \in U$. This strategy avoids exponential complexity in the number of concurrent users. If and how the state and observations can be restricted to a particular user depends highly on the model.

function MARGINALFILTERSEQUENTIALPRUNING
  ▷ Initialisation.
  $i \leftarrow 1$
  $\mathbf{p}_1 \leftarrow$ INITIALIZE
  while $y_i \leftarrow$ GETOBSERVATION **do**
    $i \leftarrow i + 1$
    $\mathbf{p}_i \leftarrow \mathbf{p}_{i-1}$ ▷ Start with previous set of particles.
    for all $u \in U$ **do** ▷ Predict each user sequentially.
      ▷ Only expand all successor states wrt. u, keep other states the same.
      $\mathbf{p}_i \leftarrow$ PREDICT$(\mathbf{p}_i, u)$
      ▷ Update weights, only if observation model allows separation of users.
      for $p_i^{(m)} = \langle x_i^{(m)}, w_i^{(m)} \rangle \in \mathbf{p}_i$ **do**
        $w_i^{(m)} \leftarrow w_i^{(m)} \cdot p(y_i \mid x_{i[u]}^{(m)})$ ▷ Restrict $x_i^{(m)}$ to user u.
      end for
      if not last user **then**
        ▷ To make use of all observations, prune after the final update state.
        $\mathbf{p}_i \leftarrow$ PRUNE$(\mathbf{p}_i)$
      end if
    end for
    ▷ Update weights, only observations not yet used.
    for $p_i^{(m)} = \langle x_i^{(m)}, w_i^{(m)} \rangle \in \mathbf{p}_i$ **do**
      $w_i^{(m)} \leftarrow w_i^{(m)} \cdot p(y_i \mid x_i^{(m)})$
    end for
    ▷ Final pruning after the last user.
    $\mathbf{p}_i \leftarrow$ PRUNE$(\mathbf{p}_i)$
  end while
end function

Running example Let $\mathbf{w} = (w^{(m)})_{m=1:M}$ denote the weights of the particles to be resampled. Let $N \geq 2$ be the target number of particles. The task of pruning is then to reduce the number of particles from M to N (Fig. 5.10).

As a running example, we use $N = 100, M = 200$, and weights that are distributed according to Fig. 5.9. In the particle set, we have 20 particles with relatively large weight $w > 0.03$, one single particle with weight $w \approx 0.005$, and 179 particles with low weight $w \approx 10^{-4}$. Figure 5.10 shows the pruning task applied to this running example.

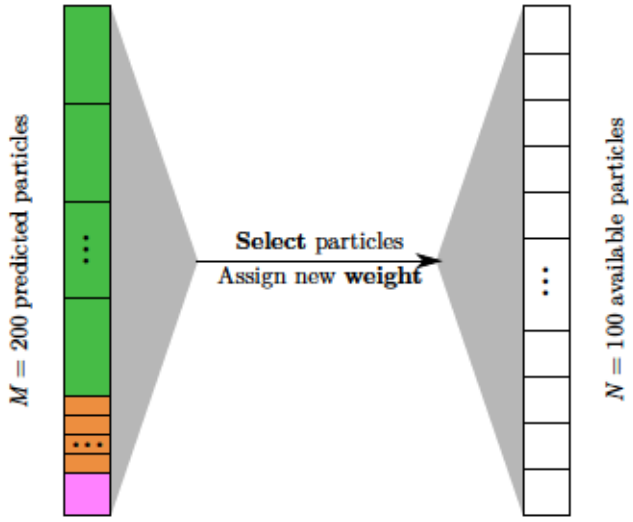


Figure 5.10: Visualisation of the pruning task. On the left hand side, the M particles are shown, where the height correlates with the weight (the weights are taken from the running example in Fig. 5.9: the large particles come first, the medium-weight particle is the last in the sequence). The task of pruning is then to select N particles (with possibly different weights).

5.4.1 Resampling

Resampling algorithms for the particle filter are usually used to resample N new particles from N particles. However, the resampling algorithms are generally also applicable to sample from $M \neq N$ particles [57] and can thus be used as a pruning strategy.

However, as resampling independently samples particles according to their weight, it is not a good choice as a pruning strategy. Independent sampling results again in particle duplications, as can be seen in Fig. 5.11. Most particles are duplicates of the few states with large weight. This is contrary to the idea of the marginal filter.

One alternative would be to resample every particle at most once, simply not using the duplicates, or to merge duplicated particles again. However, then only a small subset of the N available particles are used, decreasing the particle utilisation and the accuracy of the approximation.

5.4.2 Beam search

While the beam search is fast and easy to implement, it can be shown that it is biased. Assume a different example with $N = 2$ and $M = 4$, with four particles $p^{(1:4)}$ before pruning. All particles have different states, but execute one of two actions. Particles $p^{(1)}$ and $p^{(2)}$ have a weight of $w^{(1)} = w^{(2)} = 0.3$, both execute action a_1. Particles $p^{(3)}$ and $p^{(4)}$ have a weight of $w^{(3)} = w^{(4)} = 0.2$, both execute action a_2.

In this example, the estimate of the current action $p(A)$ gives a probability of 0.6 to a_1 and 0.4 to a_2 before pruning. Beam search will now select the particles $p^{(1)}$ and $p^{(2)}$, as they have the largest weight. This in turn results now in an estimate for the

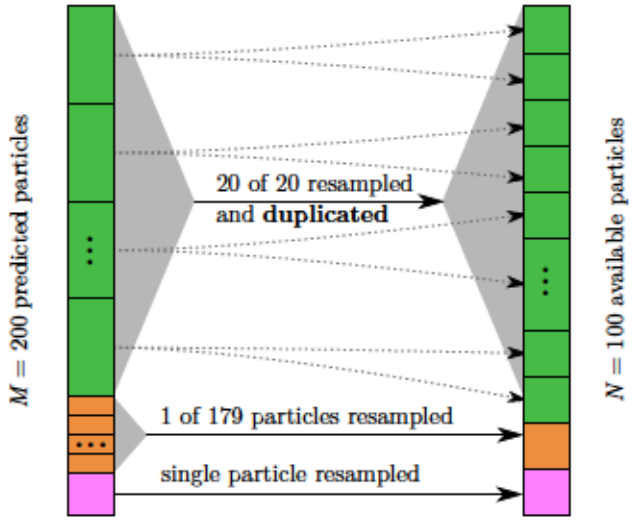


Figure 5.11: Likely behaviour of resampling algorithms for the running example. The few particles with large weights are all resampled multiple times. Out of the many particles with small weights only one is resampled. The single particle with intermediate weight is also resampled.

current action $p(A)$ where action a_1 has probability 1. This clearly diverges from the true estimate and can introduce a large approximation error. In other words: although the mass of particles (esp. if $M \gg N$) represent a particular state, this mass is simply discarded in beam search.

5.4.3 Fearnhead-Clifford resampling

Fearnhead and Clifford [68] have proposed a resampling algorithm for the particle filter. Their particle filter uses a hybrid model of an HMM with a discrete and a continuous state. We now show a novel approach and proof of their algorithm and show how this resampling is also applicable to the marginal filter. This pruning strategy combines both unbiased samples and avoids duplication of particles. It can be regarded as a combination of classical resampling and beam-search.

Approach: first prune, then resample In traditional resampling techniques, duplications are only likely for particles with weight $w^{(m)} > 1/N$. This might give rise to a first idea for pruning: set a threshold $k = 1/N$ and directly *accept* all particles with weight greater or equal this threshold ($w^{(m)} \geq k = 1/N$) exactly once, and *resample* to select particles among the rest. All *accepted* particles get their original weight. Thus, all particles which occupy the majority of the probability mass are guaranteed to "survive" pruning and duplications are avoided. The remaining particles have all weight less than $1/N$. If stratified resampling [57] is used, then it is guaranteed that particles with weight $< 1/N$ are never duplicated.

Resampling on the subset When resampling from M to N particles in the particle filter, then all particles get the constant weight $1/N$ after resampling. This ensures that the new sample of particles is unbiased. If we first directly accept some particles, then every *resampled* particle from the rest gets the constant weight $w'(k)$:

$$w'(k) = \frac{\widehat{w}(k)}{N - A(k)}, \tag{5.1}$$

where $A(k)$ denotes the number of accepted particles, and $\widehat{w}(k)$ is the total weight of the not-accepted particles:

$$\widehat{w}(k) = \sum_{m=1}^{M} \begin{cases} 0 & \text{if } w^{(m)} \geq k \\ w^{(m)} & \text{if } w^{(m)} < k \end{cases} \tag{5.2}$$

That is, we resample $N - A(k)$ particles from the remaining $M - A(k)$ particles. If we would skip the first pruning step (i. e. $k > 1$), then $A(k) = 0$ and thus $N - A(k) = N$, $M - A(k) = M$, and $w'(k) = 1/N$; i. e. this is standard resampling performed on the subset of not-accepted particles.

Constraint: accept less then N particles In general, if we *accept* all particles the weights of which are greater or equal some threshold k, we have

$$\widehat{w}(k) \leq 1 - A(k) * k. \tag{5.3}$$

One necessary condition is that we do not *accept* more than N particles, as this is the maximum number of particles, i. e. $A(k) \leq N$. Note that we also require $A(k) \neq N$: *accepting* all N particles with largest weight is the beam-search pruning strategy. Both conditions result in the constraint

$$N - A(k) > 0. \tag{5.4}$$

Constraint: preserve weight ordering Accepted particles have, by definition, a weight $w^{(m)} \geq k$; this is also the weight they get after the pruning step. We *resample* only among the particles which have all strictly less weight than the *accepted* particles, i. e. $w^{(m)} < k$. Resampled particles eventually all get a weight of $w'(k)$. However, it is not guaranteed that $w'(k) \leq k$ and thus *resampled* particles may actually get more weight than *accepted* particles after resampling. This would change the 'order' of the particles in that low-weight particles would suddenly get much higher weight. This bias should be avoided and we require that the new weight of the resampled particles is less then the weight of accepted particles:

$$k \geq w'(k). \tag{5.5}$$

The criterion that combines (5.4) and (5.5) is

$$\widehat{w}(k)/k + A(k) \leq N. \tag{5.6}$$

Lemma 1. *k satisfies (5.6) if and only if k satisfies (5.4) and (5.5).*

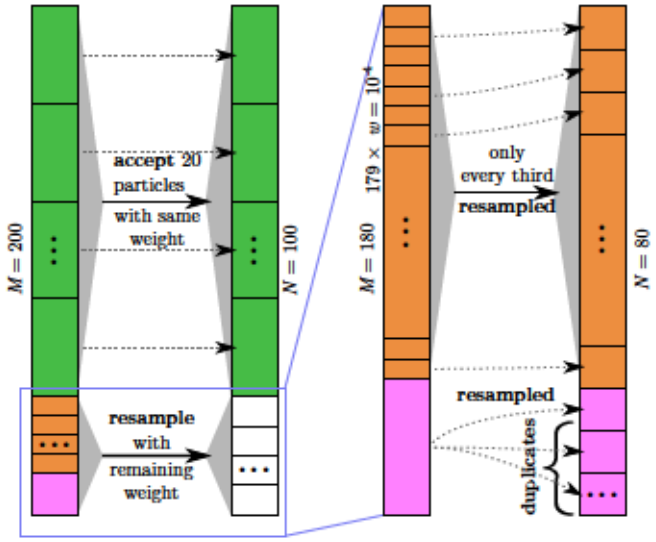


Figure 5.12: Visualisation of pruning in our running example with the intuitive choice $k = 1/N$ and stratified resampling. Left: the 20 large particles with weight $w > 0.03$ are *accepted*, and 180 remaining particles have to be *resampled*. Right: magnification of the particles to resample. Only 62 of the small particles get *resampled*, the medium-weight particle will be sampled 18 times.

Proof. Both implications can be easily verified. □

If we turn back to our first intuitive idea and set the threshold $k = 1/N$ and *accept* all particles with $w \geq k$, we have

$$\begin{aligned} \frac{\widehat{w}(k)}{k} + A(k) &\leq \frac{1 - A(k) * k}{k} + A(k) && \text{by (5.3)} \\ &\leq N \cdot (1 - A(k)/N) + A(k) && \text{by } k = 1/N \\ &\leq N \end{aligned}$$

and therefore it is guaranteed that condition (5.6) is satisfied.

Resampling can still duplicate particles In our example, we have $k = 1/N = 0.01$, and would *accept* only the 20 large particles with $w^{(m)} > 0.03$. Then the remaining weight $\widehat{w}(k) \approx 0.02$, and the new weight of the *resampled* particles $w'(k) \approx 3 \cdot 10^{-4}$, which is clearly less than k. However, resampling is still very likely to duplicate particles. Stratified resampling would *resample* only 62 of the small weights $w^{(m)} \approx 10^{-4}$. Then it reaches the medium-weight particle with $w^{(m)} \approx 0.005 > w'$, and it gets *resampled* with 17 additional duplicates. This situation is shown in Fig. 5.12.

Recursive pruning If we resample $N - A(k)$ particles with total weight $\widehat{w}(k)$, we may have *resampled* particles with weight $w^{(m)} < k$ but $w^{(m)} > w'(k)$ (this does not violate (5.5)). If using stratified resampling, any weight $w^{(m)} > w'(k)$ might get duplicated. This

is because stratified resampling duplicates particles the weight of which is larger than the 'target' (final) weight[2]. Intuitively, this duplication makes sense as this ensures that the sample is unbiased.

In fact, the underlying problem is that we are facing a new resampling problem on the subset of not-accepted particles, with the same problems described at the start of Section 5.4. Intuitively, the solution is therefore to start the same pruning algorithm recursively until no particles have weight $w^{(m)} > w'(k)$, and then resample among the rest. In the running example, we would additionally *accept* the medium-weight particle with $w^{(m)} \approx 0.005$, and *resample* only among the low-weight particles with $w^{(m)} \approx 10^{-4}$.

Constraint: avoiding all duplicates More formally, we have to avoid that the weight of any particle that is a candidate for *resampling* is greater than the newly-assigned weight $w'(k)$ of the *resampled* particles. This is reflected in the following necessary condition for k:

$$\forall_m : w^{(m)} < k \Rightarrow w^{(m)} < w'(k). \tag{5.7}$$

Finding the optimal threshold In summary, when pruning we need to find a threshold k such that both the basic constraint (5.6) and the condition preventing duplications (5.7) hold. So instead of setting $k = 1/N$, we choose any valid k such that $w'(k)$ is sufficiently low and hence (5.7) holds. Any valid k can also be limited to the range $[k_{min}, k_{max}]$, where k_{min} is the smallest k that fulfils (5.4), i. e. $N - A(k_{min}) = 1$, and $k_{max} = 1$ is the largest sensible value. Intuitively, k_{min} is the threshold that leads to accepting the $N - 1$ largest particles, and k_{max} is the threshold that accepts no particles. Note that decreasing k to fulfil (5.7) might violate (5.5). One solution is any fixed point $\widehat{k} = w'(\widehat{k})$.

Lemma 2. *For all $k \in [k_{min}, k_{max}]$ any fixed point $\widehat{k}$, i. e. any solution of the equation*

$$k = w'(k) \tag{5.8}$$

satisfies (5.6) and (5.7).

Proof. (5.5) and (5.7) hold trivially by their definition. Because of $\widehat{k} \geq k_{min}$ condition (5.4) holds, and using Lemma 1 (5.6) holds as well. □

Alternative solutions See Fig. 5.13 for the relation between k, $w'(k)$, the accepted particles as well as the two conditions (5.5) and (5.7). The existence of a fixed point $\widehat{k}$ is not guaranteed, as $w'(k)$ is discontinuous at every $k \in \mathbf{w}$ (where $\mathbf{w}$ is the set of all weights $w^{(m)}$). However, as k only serves as threshold which particles are *accepted*, we only need to consider thresholds that correspond to existing particle weights, i. e. $k \in \mathbf{w}$. Let $\mathbf{W}$ be the set of all eligible particle weights:

$$\mathbf{W} = \{w^{(m)} \mid w^{(m)} \in \mathbf{w}\} \cap [k_{min}, k_{max}] \tag{5.9}$$

For the rest of this section and without loss of generality, let the weights $w^{(m)}$ be sorted in decreasing order, i. e. $w^{(1)} \geq w^{(2)} \geq \cdots \geq w^{(N)} \geq \cdots \geq w^{(M)}$.

[2] In other resampling strategies, duplicates are also more likely to occur for particles with large weight

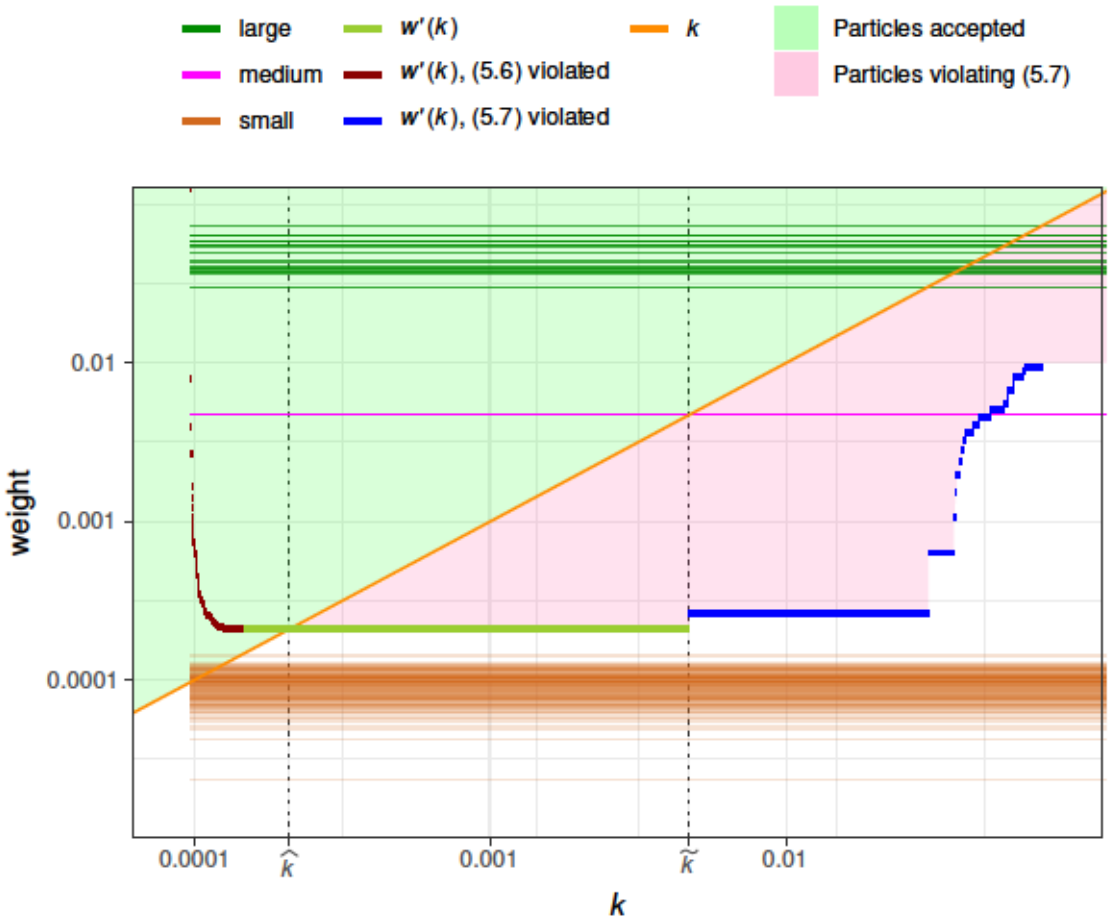


Figure 5.13: Visualisation of the *accepted* particles and w' in dependence of k (x-axis). The horizontal lines are the individual particles. For a specific k, all particles with $w \geq k$ are accepted (green area). The dashed line is the new weight $w'(k)$ that is associated to all particles that would be *resampled*. The brown part of $w'(k)$ is where (5.6) is violated, the blue part where (5.7) is violated. All particles where (5.7) is violated ($w > w'(k)$) are in the red area. For example for $k = 1/N = 0.01$, the large particles (green) get *accepted*, but the medium (magenta) and small particles (brown) would subject to resampling. In this case, the medium particle has weight $w > w'(k)$ and violates (5.7), therefore it may get resampled multiple times. If we choose $k = \widehat{k}$ (the solution of $w'(\widehat{k}) = \widehat{k}$), the large and medium particles get *accepted*, and none of the *resampled* particles can be resampled multiple times. As $\widehat{k}$ can be difficult to determine, it suffices to determine $\widetilde{k}$, the smallest weight among the particles such that (5.6) holds.

Finding the solution $\widetilde{k}$ Our goal is now to show that it suffices to find $\widetilde{k}$ as the smallest $k \in \mathbf{W}$ that fulfils (5.6), which also ensures proper pruning without duplicated particles (5.7). More specific, it suffices to find two successive $k_1 < k_2 \in \mathbf{W}$ (i. e. there is some j with $k_2 = w^{(j)}$ and $k_1 = w^{(j+1)}$) where (5.6) does *not* hold for k_1 but for k_2. First we prove that such a pair (k_1, k_2) exists, is uniquely determined and $\widetilde{k} = k_2$ also fulfils (5.7). Later we present an algorithm to find the smallest k such that (5.6) holds.

If we have such a pair (k_1, k_2), where (5.6) holds for k_2 but not for k_1, then a direct consequence is that $w'(k_1) > k_1$ and $w'(k_2) < k_2$. Intuitively, if there is a fixed point according to Lemma 2 $\widehat{k} = w'(\widehat{k})$ (as in the case of Fig. 5.13), then k_2 is the particle weight which *accepts* the same particles as the threshold $\widehat{k}$, and hence $\widetilde{k} = k_2$ satisfies (5.6) and (5.7) by the same reason as $\widehat{k}$ does. If there is no fixed point $\widehat{k}$, on the other hand, then k_2 is the discontinuity of $w'(k)$ which 'jumps' over the solution $w'(k) = k$.

In the following, we first show that there is exactly one such pair $k_1 < k_2$. Then we show that $\widetilde{k} = k_2$ satisfies not only (5.6) but also (5.7).

Theorem 1. *There is exactly one pair (k_1, k_2), where $k_1 \in \mathbf{W}$ and $k_2 \in \mathbf{W}$ are two particle weights with the following properties:*

$$k_1 < k_2,$$
$$\text{there is no } w^{(m)} \text{ such that } k_1 < w^{(m)} < k_2, \text{ and}$$
$$(5.6) \text{ does not hold for } k_1 \text{ but } (5.6) \text{ holds for } k_2. \quad (5.10)$$

Proof. Recall that k_{min} is the smallest k that fulfils (5.4), i. e. $N - A(k_{min}) = 1$, and $k_{max} = 1$ is the largest sensible value for any k. By definition, k_{min} approaches, but is greater than $w^{(N)}$ (otherwise $A(k_{min}) = N$). Without loss of generality, let $k_{min} = w^{(N)} + \epsilon$, where $\epsilon < w^{(M)}$ is smaller than any weight. Hence the remaining weight of the not-accepted particles $\widehat{w}(k_{min})$ is greater than k_{min}:

$$\widehat{w}(k_{min}) \geq w^{(N)} + w^{(M)} \geq w^{(N)} + \epsilon > k_{min}$$

Further, by (5.1)

$$w'(k_{max}) = 1/N < k_{max}.$$

Because w' is defined everywhere in the interval $[k_{min}, k_{max}]$, $w'(k_{min}) > k_{min}$ and $w'(k_{max}) < k_{max}$ (i. e. both fulfil (5.5)), both fulfil (5.4) and hence (5.6), there must also be two successive weights $k_1, k_2 \in \mathbf{W}$ for which (5.10) holds.

Sketch of the proof (by contradiction) that this pair (k_1, k_2) is unique. Assume there as an additional pair (k_1', k_2') for which (5.10) holds. Than there must be also $\bar{k}_1 < \bar{k}_2$ for which $w'(k)$ jumps over k in the other direction, i. e. $w'(\bar{k}_1) < \bar{k}_1$ and $w'(\bar{k}_2) > \bar{k}_2$. (Without loss of generality, assume $k_1 < k_2 \leq \bar{k}_1 < \bar{k}_2 \leq k_1' < k_2'$.)

By applying the definitions of $w'(k)$ and $\widehat{w}(k)$, one can show that

$$\widehat{w}(\bar{k}_1) = \widehat{w}(\bar{k}_2) \cdot c_1 \quad (5.11)$$
$$N - A(\bar{k}_1) = (N - A(\bar{k}_2)) \cdot c_2 \quad (5.12)$$
$$w'(\bar{k}_1) = w'(\bar{k}_2) \cdot \frac{c_1}{c_2} \quad (5.13)$$

where

$$c_1 = 1 - \frac{\bar{k}_2}{w'(\bar{k}_2)} \cdot \frac{1}{N - A(\bar{k}_2)} \tag{5.14}$$

$$c_2 = 1 - \frac{1}{N - A(\bar{k}_2)}. \tag{5.15}$$

The definition of $\bar{k}_2$ states that $w'(\bar{k}_2) > \bar{k}_2$ which implies $c_1 > c_2$. We can now show that $w'(\bar{k}_1) > \bar{k}_1$, which contradicts its definition $w'(\bar{k}_1) < \bar{k}_1$.

$$\begin{aligned} w'(\bar{k}_1) &> w'(\bar{k}_2) && \text{by } c_1 > c_2 \text{ and (5.13)} \\ &> \bar{k}_2 && \text{by definition of } \bar{k}_2 \\ &> \bar{k}_1 \end{aligned}$$

Therefore the assumption is false and (k_1, k_2) is unique. □

With respect to our example in Fig. 5.13, $\tilde{k}$ is the smallest k that fulfils (5.6). The following theorem states the central property that it suffices to search for the smallest $\tilde{k}$ such that (5.6) holds.

Theorem 2. *Let $\tilde{k} \in \mathbf{W}$ be the smallest k such that (5.6) holds, i. e. $\widehat{w}(\tilde{k})/\tilde{k} + A(\tilde{k}) \leq N$. Then (5.7) holds for $\tilde{k}$.*

Proof. Let $w^{(j)}$ be the particle weight with $w^{(j)} = k_2$. As $\tilde{k}$ is the smallest $k \in \mathbf{W}$ such that (5.6) holds, $w^{(j+1)}$ does not satisfy (5.6). Thus $w^{(j+1)}, w^{(j)}$ are k_1, k_2 of Theorem 1, and $k_2 = \tilde{k}$.

Case 1: $\widehat{k} = w'(\widehat{k})$ exists Then $k_1 < \widehat{k} < k_2$. The smallest particle weight accepted with $\widehat{k}$ is $w^{(j)}$, which is also the smallest particle accepted by k_2 (because $w^{(j)} = k_2 > \widehat{k}$). Therefore using $\tilde{k} = k_2$ as threshold accepts the same set of particles as using $\widehat{k}$, and thus $w'(\tilde{k}) = w'(\widehat{k})$ and consequently (5.7) holds for $\tilde{k}$.

Case 2: $\widehat{k}$ does not exist To show (5.7) holds, let $w^{(m)} < k_2$. As k_1 is the largest weight less than k_2, it follows that $w^{(m)} \leq k_1$. By using (5.13), (5.14), (5.15) and $k_2 > w'(k_2)$, one can easily show that $w'(k_1) < w'(k_2)$. Accordingly

$$w^{(m)} \leq k_1 < w'(k_1) < w'(k_2)$$

Hence, for every $w^{(m)} < k_2$ it follows that $w^{(m)} < w'(k_2)$ and (5.7) holds for $\tilde{k} = k_2$. □

Algorithm for finding $\tilde{k}$ Based on Theorem 2, $\tilde{k}$ can be found using Algorithm 11. The basic idea is to only test (5.6) for $k \in \mathbf{w}$. If that test fails, we have to increase k. If that test succeeds, there may be smaller k. Note that this algorithm only shows the basic idea of selecting k based on (5.6). In this implementation it has quadratic run-time, as the remaining weights are repeatedly computed for every k. It further only finds $\tilde{k}$ and does not implement the pruning itself. Fearnhead and Clifford [68] present an efficient algorithm based on QUICKSELECT that finds $\tilde{k}$ and resamples in linear time. The same strategy can be applied to Algorithm 11 and is not presented here for brevity and clarity.

Algorithm 11 The core of the Fearnhead-Clifford pruning algorithm for the marginal filter. This algorithm computes $\widetilde{k}$, which is the threshold for accepting all particles having at least this weight.

function COMPUTE $\widetilde{k}$
  $K \leftarrow \mathbf{w}$ ▷ The list of all candidate ks.
  $\widetilde{k} = 1$
  while $K \neq \emptyset$ **do**
    select some $k \in K$
    $\widehat{w} \leftarrow \sum_i \begin{cases} 0 & \text{if } w^{(m)} \geq k \\ w^{(m)} & \text{otherwise} \end{cases}$ ▷ remaining weight
    if $\widehat{w}/k + A(k) > N$ **then**
      ▷ (5.6) does not hold.
      $K \leftarrow \{w \mid w \in K, w > k\}$ ▷ Increase k.
    else
      ▷ There may be smaller k satisfying (5.6)
      $K \leftarrow \{w \mid w \in K, w < k\}$ ▷ Decrease k.
      $\widetilde{k} \leftarrow k$ ▷ Save last known valid k.
    end if
  end while
  return $\widetilde{k}$
end function

Relation to the original approach In their work, Fearnhead and Clifford [68] presented a different criterion for finding the 'optimal' $\widehat{k}' = 1/c$. They require $\widehat{k}'$ to be the solution of

$$\sum_{i=1}^{M} \min(w^{(m)}/\widehat{k}', 1) = N.$$

To better understand this formula, we show that this can easily be cast to our criterion $\widehat{k} = w'(\widehat{k})$ (5.8):

Lemma 3. *Let $\widehat{k}'$ be the solution of $\sum_{i=1}^{M} \min(w^{(m)}/\widehat{k}, 1) = N$, and let $\widehat{k}$ be the solution of $\widehat{k} = w'(\widehat{k})$. Then $\widehat{k}' = \widehat{k}$.*

Proof.

$$
\begin{aligned}
N &= \sum_{i=1}^{M} \min(w^{(m)}/\widehat{k}', 1) \\
&= \sum_{i=1}^{M} \begin{cases} 1 & \text{if } w^{(m)} \geq \widehat{k}' \\ w^{(m)}/\widehat{k}' & \text{if } w^{(m)} < \widehat{k}' \end{cases} \\
&= A(k) + \frac{1}{\widehat{k}'} \sum_{i=1}^{M} \begin{cases} 0 & \text{if } w^{(m)} \geq \widehat{k}' \\ w^{(m)} & \text{if } w^{(m)} < \widehat{k}' \end{cases} \\
N &= A(k) + \frac{\widehat{w}(\widehat{k}')}{\widehat{k}'} && \text{according to (5.2)} \\
\widehat{k}' &= \frac{\widehat{w}(\widehat{k}')}{N - A(k)} \\
\widehat{k}' &= w'(\widehat{k}') && \text{by (5.1)} \\
\widehat{k}' &= \widehat{k} && \text{by (5.8)}
\end{aligned}
$$

□

5.5 Evaluation

After theoretically presenting the marginal filter (MF), the objective is this section is to evaluate the efficiency of the marginal filter. We will use the same models and datasets as for the analysis of the particle filter (PF) in Section 5.1. In particular, we will focus on evaluating the following hypotheses, based on the three algorithm described in the previous section (each is to be considered in the context of human situation recognition):

- The MF avoids sample impoverishment and is able to follow all observations.
- The MF is more efficient than the PF, as it avoids sample impoverishment.
- Fearnhead-Clifford pruning is more efficient than beam search pruning, as it is unbiased and thus statistically more sound.
- Sequential prediction (as discussed in Section 5.3.4) is more efficient than single prediction for multi-user models, as the exponential growth in the number of users is avoided.

The rest of this section will first detail the methods for these evaluations (Section 5.5.1). Section 5.5.2 will present the results, which are then discussed in more detail in Section 5.5.3. Finally, Section 5.5.4 will summarise the evaluation and conclude this chapter.

5.5.1 Methods

Efficiency comparisons Efficiency is tested according to Definition 2 (page 24), i. e. one result is considered more efficient if it requires less time *and* has a smaller approximation

error. The approximation error of the marginal filter is also computed wrt. the exact filtering distribution, using the same error measure as described in Section 5.1.1. This means that the results only include the estimates for the first few timesteps, just as they did for the particle filter analysis in Section 5.1 (in particular for the localisation model with 7 users). However, the relative tests of the marginal filter compared to the particle filter are valid for these timesteps.

The evaluation of the sequential prediction for multi-user models can be considered as an additional approximation on top of the marginal filter with single prediction. Thus, we can and will also report the approximation error and efficiency wrt. to marginal filter with single prediction, including all timesteps of the estimate.

Implementation As the evaluation of the efficiency includes the wall-clock run-time of the algorithms, a 'developer bias' towards the marginal filter may occur (I want to propose the marginal filter as a novel algorithm and have thus invested much more time in tweaking and fine-tuning the execution on the hardware). To minimise this bias, the MF and PF share a large part of the code base. This includes the complete model definition as well as input and output routines. Only the actual algorithms are implemented differently, but still using the same concepts, architecture and interfaces. It is also worth noting that the implementation is based on and extending the CCBM toolkit [217], which originally only featured a particle filter and was thus primarily optimised for the particle filter.

The Fearnhead-Clifford pruning was implemented based on the QuickSelect algorithm, not using the (simpler to understand) quadratic-time implementation shown in Algorithm 11.

Configurations To evaluate all hypotheses, the following configurations were varied for the different runs:

Model The same models as for the particle filter analysis in Section 5.1.1 were used: the single-recipe kitchen model, the CMU kitchen model, the meeting model and the localisation model (2 and 7 users).

Number of particles The same number of particles were used for the marginal filter, with the exception for the localisation model (7 users) where the single-prediction MF was limited to 1,000 particles due to long computation time for more particles.

Filter algorithm We run both the particle filter and the marginal filter on all models with the corresponding number of particles. Technically, the results of the particle filter from the analysis of Section 5.1 are re-used.

Pruning strategy For the marginal filter, we use both beam search and Fearnhead-Clifford pruning to compare their influence on the efficiency.

Multi-user prediction We also use the single prediction as well as the sequential prediction for the marginal filter and the multi-user models. The single-user models use only the single prediction, as sequential prediction is identical to single prediction in this case.

Table 5.2: Overview over the different configurations used for the evaluation of the marginal filter (MF) and comparison to the particle filter (PF). The column 'Pruning' shows the pruning algorithm used for the MF; Beam-search is denoted by 'BS' and Fearnhead-Clifford pruning by 'FC'. The column 'Prediction' shows the prediction strategy used for the MF; 'Single' refers to single prediction and 'Seq.' refers to sequential prediction as in Algorithm 10. The maximum number of particles used for the corresponding filter is shown in the column 'Max. particles'. The column 'Number runs' shows how many different, individual runs were actually computed. The number of runs depends on the different number of particles used, the number of datasets and random repetitions for the particle filter and Fearnead-Clifford configurations.

Model	Filter	Pruning	Prediction	Max. particles	Number runs
Kitchen	PF			100,000	3,150
	MF	BS, FC	Single	100,000	3,213
CMU kitchen	PF			32,000	34,400
	MF	BS, FC	Single	32,000	35,088
Meeting	PF			32,000	8,000
	MF	BS, FC	Single, Seq.	32,000	16,320
Localisation (2 u.)	PF			100,000	2,250
	MF	BS, FC	Single, Seq.	100,000	4,590
Localisation (7 u.)	PF			10,000	1,750
	MF	BS, FC	Single	1,000	1,275
	MF	BS, FC	Sequential	10,000	1,785

All configurations using the particle filter as algorithm or using the marginal filter with Fearnhead-Clifford pruning use random sampling or resampling. Thus, all these configurations are repeated 50 times to minimise random artefacts. In total, this results in all configurations shown in Table 5.2.

Common language effect size As we have no efficiency measure for a single run, we cannot compare the 'efficiencies' of two individual runs (i. e. a specific datasets for a MF configuration with the same dataset for a PF configuration). Instead, we can only *test if* an individual run for one filter configuration (we will call it the *target filter*) is more efficient than for another filter configuration (the *baseline filter*). As stated above, this test is based on Definition 2.

However, we will not report individual test results (there are too many individual runs). Instead, we employ the *paired common language effect size* when comparing a target filter with a baseline filter. The common language effect size is the ratio of how many target filter runs are more efficient than baseline filter runs, comparing all possible combinations of target filter runs and baseline filter runs.[134] We will make it a paired test by comparing only runs for the same dataset.

For this evaluation, we have three target filters, each corresponding to the second, third, and fourth hypotheses (see Section 5.5):

- marginal filter with beam-search pruning (second hypothesis),

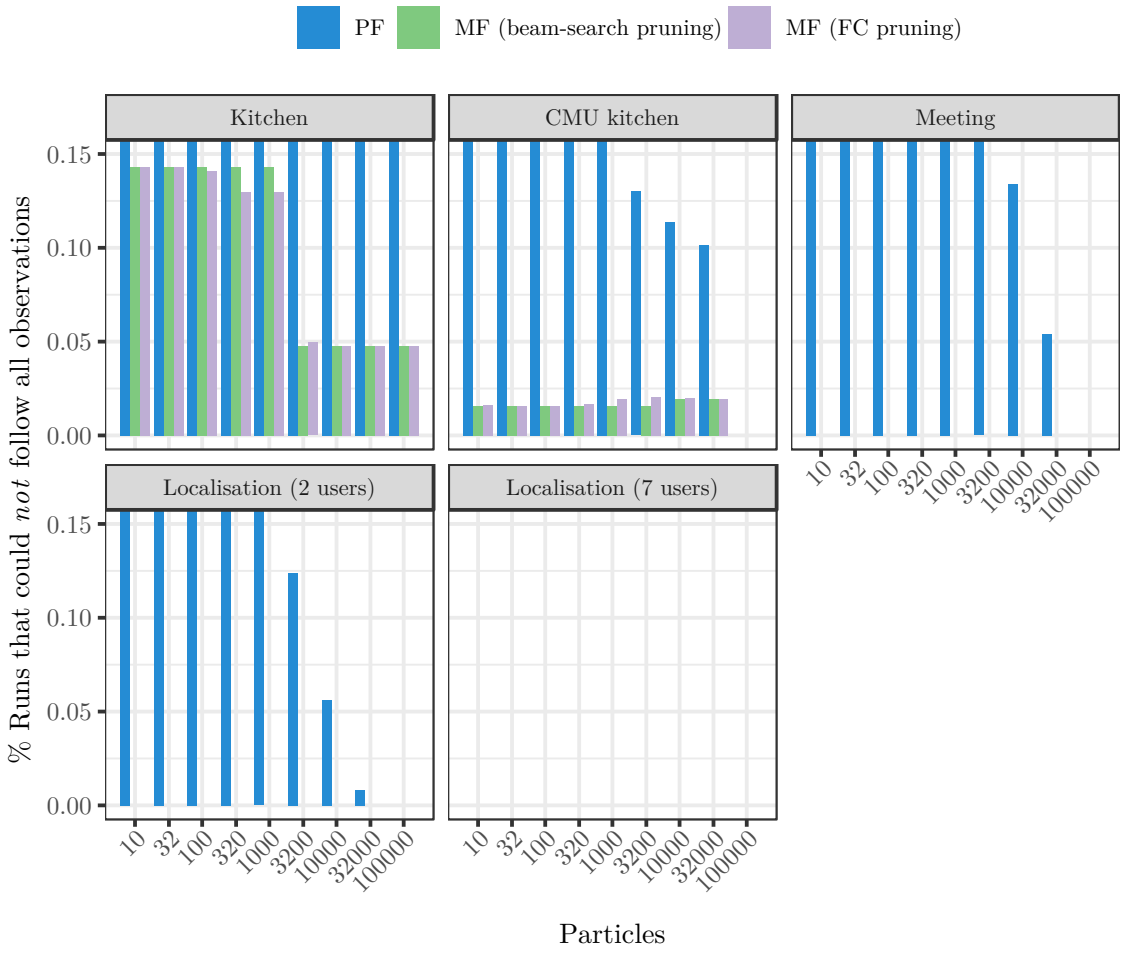


Figure 5.14: Percentage of runs that could not follow all observation data. Lower is better. If a run could not follow the observation data, then at one point in time, all particles for this run had zero weight. The marginal filter can always follow the observations better than the particle filter. To focus on the results of the marginal filter, the plot has been clipped. The full plot for the particle filter only is shown in Fig. 5.3.

- marginal filter with Fearnhead-Clifford pruning (third hypothesis), and
- marginal filter with sequential prediction for multi-user models.

The baseline filter is always the particle filter, and also the marginal filter (beam-search pruning) for the last two hypotheses / target filters.

5.5.2 Results

Avoiding sample impoverishment (first hypothesis) First, we want to empirically confirm that the MF indeed avoids sample impoverishment. This should also lead to a much better ability to follow the observations. The results indeed show that at every

single timestep, the marginal filter uses one particle per state. This confirms that particle merging is working and thus results in a much better particle utilisation. (This is not shown in a plot: the plot would be not very informative, as it would essentially be identical to Fig. 5.2, where the marginal filter is just a thin line at 'Particles per State' = 'Particles'.)

Figure 5.3 of Section 5.1.2 has shown that the particle filter could not follow the observations for all models. That is, it occurred relatively often that the approximation of the particle filter had not a single particle with weight > 0, i. e. the approximation of the filtering distribution has too few states from the exact distribution. In this respect, the marginal filter has a much better ability to follow the observations, as shown in Fig. 5.14. While the PF could not finish all runs of the meeting and localisation model (2 users), the MF had not a single run where it could not follow the observations. Similar, it has always significantly less runs not following the observations for both kitchen models. The MF thus clearly outperforms the PF, as it suffers less from cases where the approximation contains too few states.

Neither beam-search nor Fearnhead-Clifford pruning has clear advantages wrt. the ability to follow the observations. For the kitchen model, Fearnhead-Clifford pruning can follow the observations longer in 15 cases. Beam-search pruning can only follow the observations longer in one case. For the CMU kitchen model, beam-search pruning performs better in 83 cases, but worse only in 15 cases. As the MF could follow all observations in all multi-user models, no differences between single prediction and sequential prediction could be observed.

Efficiency overview Figure 5.15 shows the approximation error by the different marginal filter configurations compared with the particle filter, plotted over the run-time required. As can be apparently seen from the plots, the marginal filter can achieve always smaller approximation errors with sufficiently many particles. It is also only considerably slower for the multi-user models when using single prediction.

Note that we cannot compare the efficiency paired by the number of available particles. The run-time of each filter is different for the same number of particles. In particular, the marginal filter is almost always slower than the particle filter for the same number of particles, but achieves much lower errors. Likewise, the same run-time can be achieved by selecting different number of particles.

Figure 5.16 shows for how many particles of the target filter the effect size is maximised compared to the baseline filter for a particular number of particles. In other words, how many particles the target filter requires to achieve the best efficiency.

This figure should be read as follows: If one currently uses the particle filter for the Kitchen model (top row) using 32,000 particles (blue line), and wants to replace it by the marginal filter (beam-search pruning, left column), then one should use 100 particles for the marginal filter. This ensures that the marginal filter is as efficient as possible compared to that PF configuration, i. e. it uses less time and has a less error for most runs.

The size and shape of the points denote the effect size, i. e. how many individual runs are

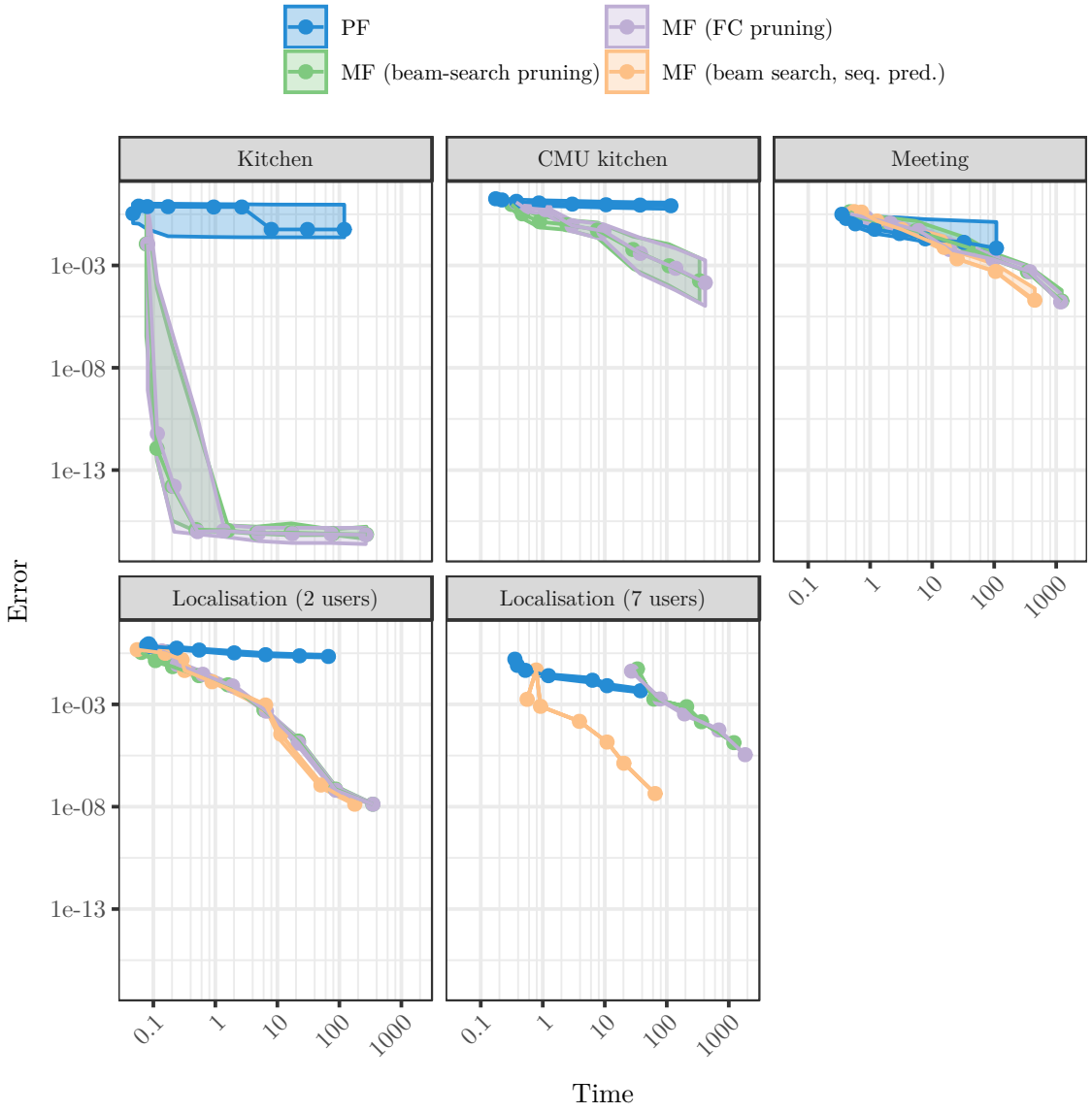


Figure 5.15: Overview of the individual marginal filter results for the models, compared to the particle filter results. The x-axis denotes the computation time required per run, the y-axis represents the approximation error wrt. the true filtering distribution. The runs are grouped by the number of particles, each group represents all datasets and random repetitions for a given parametrisation. The dots are located at the median of the time and error, the ribbon visualises the 25% and 75% quantiles of the error. Note the logarithmic scale on both axes.

more efficient. For instance, the large circle for the above example (MF with 100 particles compared to PF with 32,000 particles, kitchen model) shows that the marginal filter is more efficient for *all* of individual runs. In contrast, when comparing the MF to the PF with 1000 particles for the localisation model (7 users), the MF (target filter) is more efficient for *none* of the individual runs. In this case, the marginal filter requires much more computation time (7 users with single prediction) for even 10 particles (see also Fig. 5.15).

Efficiency of the marginal filter (second hypothesis) For now, we want to compare the efficiency of the marginal filter (beam-search pruning) to the particle filter (this corresponds to the second hypotheses). We will thus focus first on the blue/green lines of Fig. 5.15 and the left column of Fig. 5.16.

For all models, the MF achieves much lower approximation errors than the PF when using the same number of particles. This effect is significant[3] for all models (except the meeting model) with at least 32 particles ($p < 0.001$) and largest for the kitchen model and localisation model (2 users). For the meeting model, the MF can achieve significantly lower approximation errors only for 1,000 particles or more ($p < 0.001$).

When comparing the efficiency (i. e. combination of both approximation error and time), the MF is generally more efficient only for the two kitchen models (single-recipe and multi-recipe) and localisation model with 2 users. From Fig. 5.15, we see that the MF almost always has results which are below (less error) and to the left (less time) of the PF for these models, and is thus more efficient.

This can be seen in more detail in Fig. 5.16 (left column). One can always find an MF configuration that is more efficient for most runs (effect size > 0.75) than the PF for both Kitchen models and the localisation model (2 users), as long as the particle filter uses at least 320 particles. The MF is not generally more efficient for less than 320 particles in these models, as the PF has less run-time. Likewise, there is no MF configuration that is more efficient for the meeting model and localisation model (7 users) than the PF, irrespective of how many particles the PF uses.

Figure 5.16 also shows that the marginal filter is particularly more efficient than the particle filter in the single-recipe kitchen model (top-left plot). Even when the PF uses 10,000 particles, the MF is more efficient in all runs (effect size is 1) with just 10 particles. It similarly outperforms the PF for the localisation model with 2 users, where it only needs 32 particles for better efficiency for all runs.

From Fig. 5.15 it is apparent that the marginal filter (green line) requires much more computation time than the particle filter for the localisation model with 7 users. The cause is that the computational complexity is exponential in the number of users.

Efficiency of Fearnhead-Clifford pruning (third hypothesis) From Fig. 5.15 we see that the results of Fearnhead-Clifford pruning (purple line) are very similar to the results of beam-search pruning. A significantly different error than the beam-search pruning

[3] Using Wilcoxon significance test with confidence level = 0.99 where the alternative hypothesis is that the MF error is less than the PF error.

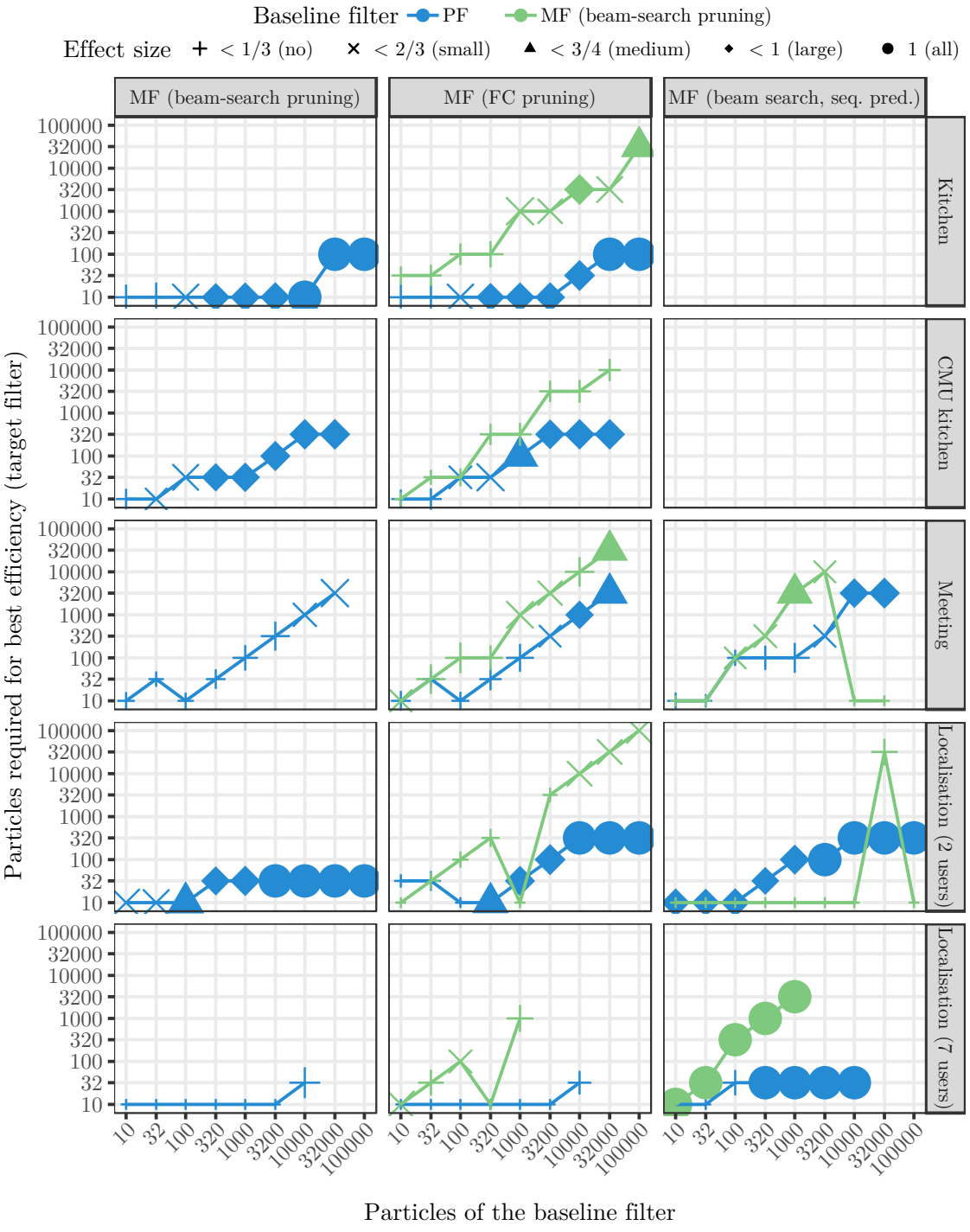


Figure 5.16: Paired common language effect sizes for the efficiency tests between the marginal filter and particle filter for all models. Columns represent target filters, one row for every model. The larger the point, the larger the effect size (i. e. better efficiency for the target filter). The lower the line, the less particles are required for the efficiency.

cannot be consistently be shown, the majority of the Wilcoxon tests assign a p-value $>$ 0.1.

The efficiency effect size for this comparison can be seen in the middle column of Fig. 5.16, where the comparison to the beam-search pruning is shown using the green line. The effect size compared to the particle filter in the blue line in shown for comparison. Only for three configurations, the largest effect size is greater than 2/3, i. e. the Fearnhead-Clifford pruning is more efficient for 2/3 of the runs.

The green line in the middle column is consistent with the result, that there are no significant differences between Fearnhead-Clifford pruning and beam-search pruning. The best efficiency of FC-pruning compared to beam-search pruning is achieved when using (in most cases) the same number of particles. This is indicated by the fact that the line scales roughly with the number of particles for the beam-search pruning (i. e. the baseline filter). Note that the best efficiency of FC-pruning leads to effect sizes in the range between 1/3 and 2/3 for most results, i. e. approximately half of the runs are more efficient for FC-pruning, while the other half is more efficient for beam-search pruning.

Efficiency of sequential prediction (fourth hypothesis) Sequential prediction aims at reducing the computational complexity for multi-user models, such that the run-time is not exponential in the number of users. As Fearnhead-Clifford pruning shows no strong effect, we will only show the effect of sequential prediction using beam-search pruning, both as target filter and as the baseline filter (with single prediction).

The results of the sequential prediction are shown using the orange line in Fig. 5.15. This figure shows that sequential prediction outperforms the MF with single prediction for the localisation model with 7 users. It requires much less time with similar approximation errors. However, sequential prediction generally incurs an additional approximation error. Over all models, the approximation error of sequential prediction is larger than single prediction, this result is significant ($W = 4302$, $p < 0.001$). The median of the additional approximation error is 0.01.

Therefore, the sequential prediction for the localisation model (7 users) is much faster with only a slight additional approximation error. This error can be compensated by using more particles – this is reflected in Fig. 5.16 (bottom-right plot). There we can see that there is a configuration of the MF with sequential prediction which is always (effect size = 1) more efficient than the MF with single prediction. For instance, when compared to a marginal filter with single prediction (green line) using 1000 particles, the best efficiency is achieved by using the MF with sequential prediction and 3,200 particles.

In contrast to the localisation model with 7 users, the sequential prediction is not more efficient for the other multi-user models (with 3 and 2 users). There are no large effect sizes for these models in Fig. 5.16 (right column) compared to single prediction (green line). This is because the additional approximation error is not outweighed by a slightly faster computation.

5.5.3 Discussion

The results of the previous section did not identify a single 'best' configuration or filter algorithm. As it is, the right choice depends on the model and its properties. We will thus now discuss a few properties and which configuration achieves the best efficiency.

Simple models For the meeting model, the PF can also achieve good results. In the case of this model, the action can be estimated quite good based on the current position alone (a continuous property), which favours the particle filter. This shows that for simple models, or models which are for a large part continuous, the particle filter might be more suitable.

Complex human behaviour models In contrast, the MF is (in some configuration) generally more efficient than the PF. This is because, as shown before, the MF avoids sample impoverishment and better utilises the available particles. This then results in a lower approximation error using only slightly more time, over-all resulting in a better efficiency. As Fig. 5.16 shows, the MF is always more efficient than the PF (blue line) using less particles than the PF; as long as the PF uses more than 100 particles.

Particle filter with few particles For the configurations of the particle filter that use at most 100 particles, the marginal filter is usually not more efficient (the effect size in Fig. 5.16 is smaller than 3/4). This is because the PF with few particles has very low run-time, which cannot be beaten by the marginal filter (although the marginal filter usually has lower approximation error in these cases). It follows that the MF is not suited to replace the PF with very few particles. But in real applications with complex models, the approximation error with less than 1000 particles is usually too high, so more particles are required anyway for a sufficiently good estimate.

Multi-user models Due to the exponential complexity in the number of users, single prediction is much slower for multi-user models. This has also been confirmed empirically by the results of the localisation model with 7 users. Thus for models with a large number of users, sequential prediction is required for limited run-time.

It is worth noting that sequential prediction is an additional approximation on top of the marginal filter. Although the additional approximation error was very small, there might be cases where the error can be larger. In particular, the number of particles must be sufficient to represent all possible actions of at least a few users. If the number of particles is very small, but each user can execute a large number of actions, the sequential prediction can discard many states too early.

Fearnhead-Clifford pruning Although statistically more sound, Fearnhead-Clifford pruning does not show better results than beam-search pruning. In these complex human-behaviour models, it is probably very unlikely that corner-cases as presented in Section 5.4.2 occur. Additionally, the difference between FC pruning and beam-search

pruning is only for some particles (remember that FC pruning is a hybrid between beam-search and resampling). And for the resampling-phase of FC pruning, particles with higher weight are more likely to get resampled – these particles with higher weight are also selected in beam-search pruning. Thus, the difference in approximating the filter distribution is not very large and only affects states with low weight.

In a rigorous filter implementation, Fearnhead-Clifford pruning should probably be selected due to its statistical properties. It is not slower than beam-search. However, it is a randomised pruning strategy. For simple tasks or better reproducibility, beam-search pruning might be the better choice.

5.5.4 Summary

Based on these results, we can confirm three of the four hypotheses (one only partially) from the introduction of Section 5.5:

- *Confirmed*: The MF avoids sample impoverishment and is able to follow all observations.
- *Confirmed partially*: The MF is more efficient than the PF, as it avoids sample impoverishment.
- *Not confirmed*: Fearnhead-Clifford pruning is more efficient than beam search pruning, as it is unbiased and thus statistically more sound.
- *Confirmed*: Sequential prediction (as discussed in Section 5.3.4) is more efficient than single prediction for multi-user models, as the exponential growth in the number of users is avoided.

We could not confirm that the marginal filter is generally more efficient for simple models with continuous state spaces (i. e. the meeting model).

As a general advice, the particle filter should be used for simpler models, i. e. models with few states and informative observation models. For complex models of human behaviour, the marginal filter should be used. The efficiency of one algorithm over an other should be tested in preliminary studies and tests, as should be the number of particles. Generally, sequential prediction is favoured (which is identical to single prediction for single-user models), beam-search pruning is simpler to implement and sufficient for most cases. If the particle filter has been used before, the number of particles for the marginal filter can usually be reduced by an order of magnitude for increased efficiency (see the plots of Fig. 5.16).

6 Conclusion and future work

6.1 Summary

As stated in Chapter 1 (page 25), the central objective of this dissertation is to improve the efficiency of Sequential Monte Carlo methods for human behaviour models. The motivation for more efficiency is to enable online inference for realistic and complex behaviour models.

To justify this need, Chapter 2 has analysed the state of the art in situation recognition. It shows that past research has not considered realistic and thus very complex behaviour models. Experiments and analyses have been performed in either virtual settings or very constrained domains (e. g. distinguishing only a few activities of daily living). Only few research is interested in recognising the full situation of the user, that is build a model of the activities and understanding the context and environment where the user performs his activities. As a result, the algorithmic foundation of inferring an estimate of the current situation has not seen much research in this direction. The experiments mostly rely on existing frameworks (such as the particle filter).

The main contribution of this dissertation is the analysis and improvement of the marginal filter as presented in Chapter 5. Existing approaches using SMC methods are based on the particle filter. Chapter 5 analysis the behaviour of the particle filter for human behaviour models. It identifies the causal and symbolic state space as a key factor for a low efficiency of the particle filter compared to continuous state space models. Based on these observations, it is shown how the marginal filter performs much better for human behaviour models than the particle filter.

Another aspect of efficient inference are error-free models, in particular for the causal modelling approach followed in this dissertation. If the models contain error such as deadlocks or livelocks, valuable resources are used for non-sensible or actually impossible states. To mitigate these problems, Chapter 4 proposes techniques based on model checking. Analysis of exemplary models used in previous publications has indeed shown avoidable modelling errors.

However, in practice a working situation recognition needs a lot more than the results provided in this dissertation. Accurate recognition of the true situation needs first and foremost accurate sensors, a wide variety of sensing modalities and a good coverage of the environment with sensors. Only this can ensure that sufficient information is available to the situation recognition. The literature review of existing installations has also shown that the deployment of sensors and installation of the recognition system overall needs a lot of engineering to be adapted to the specific requirements and recognition targets. For these reasons, Hoey et al. [87] propose two engineers for their method of building an assistive systems: a ‘ubiquitous sensing technician’ who is responsible for selecting

appropriate sensors, and a 'human factors annotator' who is responsible for observing and de-structuring the behaviour of the user. For approaches using causal models based on state space systems, a third modelling engineer following a modelling process (e. g. as proposed by Yordanova [217]) might also be necessary.

My work provides theoretical foundation to deal with models and the *computational* challenges for such a setup. I have shown that realistic models tend to be very complex (in terms of developing and maintaining) and also very large (in terms of state space size in probabilistic reasoning). The marginal filtering algorithm was shown to be able to more efficiently (wrt. computation time and approximation error) handle such models than other state-of-the-art inference methods. In the following section, I sketch further possible improvements to the marginal filter.

6.2 Discussion

It could be shown that the Marginal filter is more efficient for most of the domains used in Chapter 5. Often, the marginal filter's efficiency is considerably better. This shows that

- it is worth improving the inference algorithms when complex models are used. Before optimising the sensor set-up and models, it is important to ensure that the inference algorithm produces meaningful (see Fig. 5.3 where the particle filter could not finish the inference) and correct results.
- when a model-based approach with Bayesian filtering is used, the Marginal filter should be favoured over the Particle filter as the inference algorithm for causal models with categorical state spaces.

Applicability of the results The results presented here only cover probabilistic state space models with large categorical state spaces and causal relations between states and actions. It is expected that the Marginal filter will not be similarly efficient for models with other characteristics. In particular, the Particle filter is expected to be more efficient for models with continuous state spaces and actions (e. g. tracking real-valued locations). Continuous state spaces and actions have meaningful definitions of measures such as the mean, allowing for more accurate estimates. They also allow for all the improvements to the Particle filter algorithms presented in Section 5.2.

If the model is a causal yet simple model, other approaches might be viable as well. For small state spaces without long-running actions, exact approaches entirely omit any approximation errors. The Marginal Filter also operates exactly as long as the support for the filtering distribution does not exceed the maximum number of particles. If the action durations can be modelled as exponential distributions, Hidden Markov Models are a natural and efficient representation, with the Forward algorithm [138, p. 58] as an efficient and exact inference method.

The Marginal filter relies on good observations models, i. e. a probabilistic model of the sensor output given a state. From a sensor perspective, the behaviour model is used

to fill the gaps between sensors, and add prior knowledge how the situation evolves. If good *generative* observations models are not available, but the sensors cover all of the required states with no need for prior knowledge about the future behaviour, then a discriminative approach might give better results [94].

Approximation error versus accuracy This work compared the efficiency of the different filters. Efficiency was defined (Definition 2) in terms of approximation error and run-time. Use of the approximation error is a key factor in this dissertation. The literature in situation recognition only evaluates the complete set-up in most cases (ranging from algorithm and models to sensors) by measuring the achieved accuracy wrt. the actual situation based on annotations (e. g. documented by filming the experiments). In contrast, this work focusses on only the algorithmic challenges for situation recognition. Thus, the output of the algorithms (probability distributions over the states) is evaluated wrt. the exact distribution (computed by exact, but very slow, algorithms).

Using the approximation error as performance measure also excludes errors introduced by the model engineer in the behaviour model, observation model and errors in the annotations. These are all factors that can invalidate the recognition accuracy wrt. true action, but do not systematically affect the approximation error.

6.3 Future work

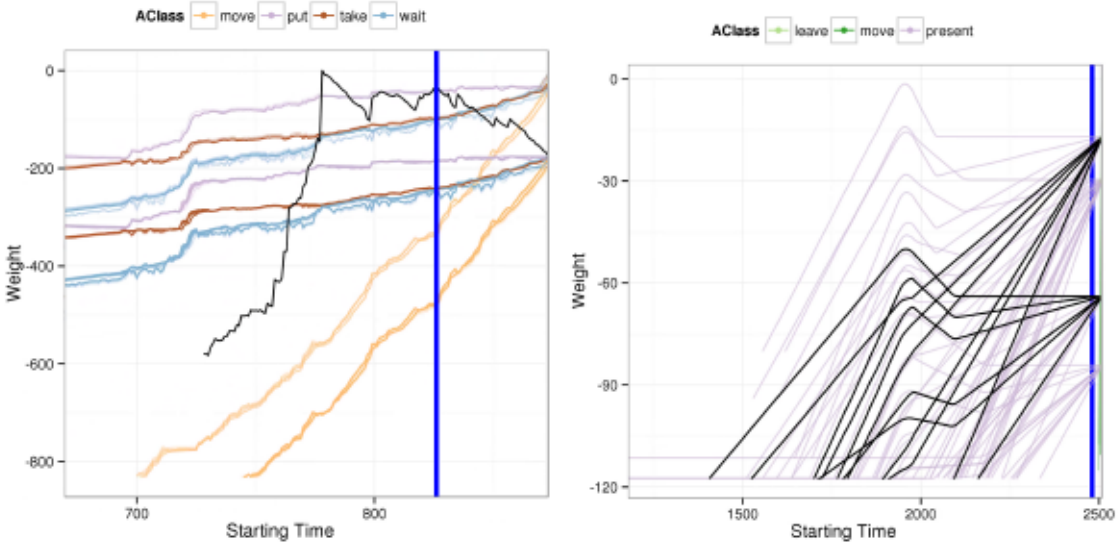
6.3.1 Efficiently handling action durations

Problem statement The literature analysis in Chapter 2 has shown that there is currently no work that handles the problem of long-running actions. SMC methods use a representation and factorisation of the state as discussed in Section 3.1.1, where one particle consists (among others) of the environmental state S, the current action A and the action's starting time T. The starting time is necessary to keep track of the duration of the current action, as this duration information can also help distinguishing between actions by different execution durations (e. g. washing a pot might be modelled to take considerably longer than washing a single spoon). As the inference progresses through the events, the starting points for specific actions might become very uncertain. As a result, a large number of different starting times have to be considered.

For the models used in this work, the marginal filter tends to consider only very few different starting times (for instance, using single-user kitchen model only on average 2-5 different starting times per $\langle S, A \rangle$ state). Other starting times are pruned and thus removed from the approximation of the filtering density. This causes two issues:

- When only very few different starting times are considered per $\langle S, A \rangle$ state, the correct estimation of the action's duration and thus recognising executing a new action is degraded

- When the filter does consider a large number of possible starting times, the filter allocates a large number of particles for only estimating the starting time, increasing



(a) Single-recipe kitchen model

(b) Meeting model

Figure 6.1: Distribution of the starting times for every $\langle S, A \rangle$ state at an exemplary timestep during the inference. Every line is a different $\langle S, A \rangle$ state, the x axis corresponds to a specific starting time, the y axis the corresponding particle's weight (log-domain). Different action classes can be distinguished by colours (black is the true action), the blue bar indicates the true starting time.

the error in recognising the true $\langle S, A \rangle$ state (i. e. the true environmental state and action).

Therefore, the marginal filter needs a more efficient handling of action durations and efficiently represent the action's starting times. Potential improvements can exploit the fact that the starting times of an action are one-dimensional distributions (in contrast to the high-dimensional, causal transition model between environmental states and actions).

State of the art It is known that modelling action durations can increase the accuracy of the recognition [94, 135]. Past approaches of modelling action durations, however, are restricted to special classes of duration distributions like exponential [180, 212], Gaussian [170] or Coxian [64]. These restrictions (among other simplifications in these specific models) allow the use of exact filtering algorithms, which is not applicable for the setting of this dissertation. To my knowledge, there is yet no work on efficient duration inference using SMC methods.

Problem analysis In a preliminary study, I have analysed the actual distribution of starting times for the different $\langle S, A \rangle$ states. In order to get meaningful distributions of starting times, I modified the filter such that more starting times are tracked by limiting the number of different $\langle S, A \rangle$ states. The results then show how the starting times would be distributed if much more starting times would be present in the belief state of the

filter. Two samples for the single-recipe kitchen model and the meeting model are shown in Fig. 6.1.

By visual inspection, the distribution of the starting times are approximately line-segments in the log-domain. This is true over all timesteps in these models and can be explained by the observation models and duration models: The duration model is one-dimensional and thus very flat compared to the high-dimensional observation model of our domains. When the $\langle S, A \rangle$ state conforms to the observation model, the weight stays approximately the same (with only small changes by the duration model), resulting in a flat line. When the $\langle S, A \rangle$ state does not conform to the observation model, the weight is multiplied by a very small factor. As the observations usually only slightly change in time, the factor from the observation model is approximately constant over many timesteps, resulting in a straight line in the log-domain. What is more, our DBN from Section 3.1 states that there is no dependency between the starting time and the observation model, hence lines in the log-domain of the starting times distribution are preserved over all timesteps.

Proposed solution I propose a piecewise log-linear model to approximate the distribution of starting times for every particle. This allows a single particle to represent multiple starting times and thus reduce the number of particles by a few orders of magnitude. Conversely, using the same number of particles, more states can be supported during inference, increasing the efficiency. Additionally, this compact representation reduces the number of state transitions (Section 3.1.2), as the applicability of actions and transitions of the environmental state does not depend on the exact value of the duration distribution (cf. (3.5)–(3.11)).

I developed an efficient (at most $\mathcal{O}(\log |\mathbf{T}|)$) model update using a novel constant-time change-point detection algorithm that has been published in [147]. When using this particle representation, we can observe a reduced approximation error using the same number of particles (Fig. 6.2) for the single-recipe kitchen domain. The CMU kitchen domain has no consistent results (the algorithm does not work with the other models, as they are multi-user models).

As a downside, the marginal filter runs using log-linear approximation take 5 times longer, so that the filter execution is *not* more efficient than the standard marginal filter. The Valgrind profiler[1] on a specific run (Kitchen model with 1000 particles) shows that

- the approximation spents 50% of the time computing exponential and logarithmic functions; the standard MF only 20%
- one of the most expensive function is computing the total weight over all line segements, which is used in different places and takes 20% of the time (85% of which takes place in the math library)
- excluding the run time of the math library, the approximated MF is still 3 times slower

[1] `http://valgrind.org`

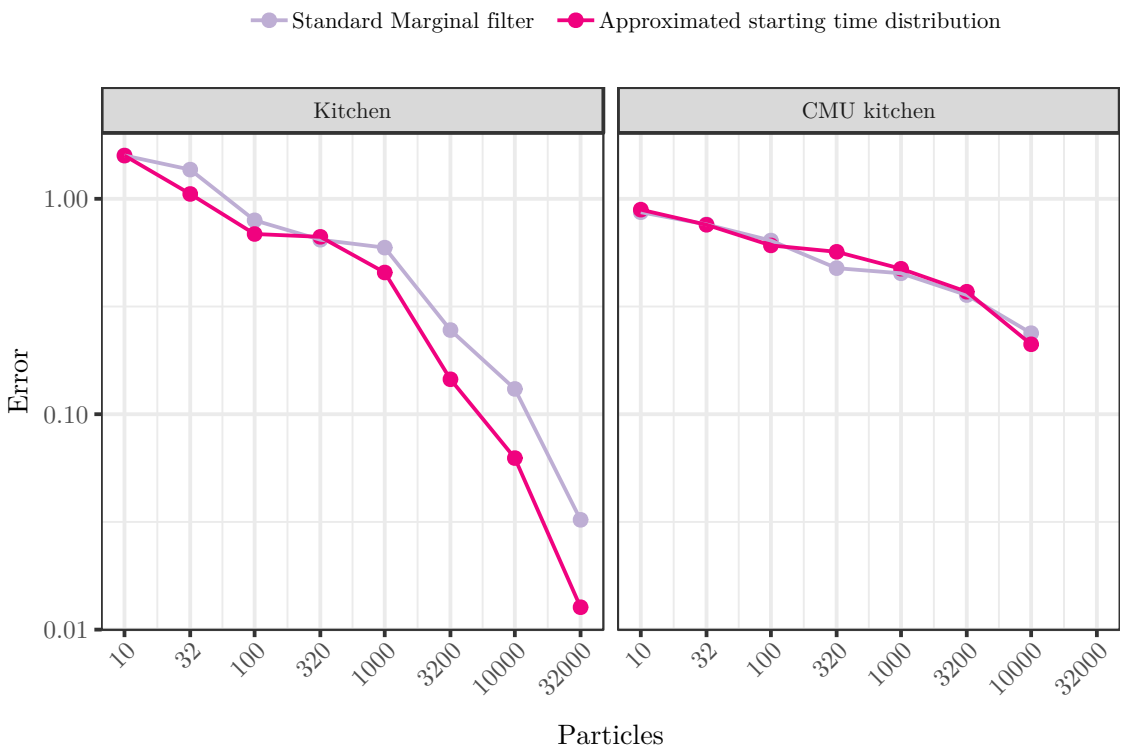


Figure 6.2: Comparison of the approximation errors between the standard marginal filter and the marginal filter with approximated starting time distribution (using log-linear models). The error is computed wrt. to standard marginal filter with 100,000 particles, as the real distribution is not available.

It seems that the overhead of managing the line segments does not outweigh the benefit, and approximating more starting times does not drastically decrease the error to compensate for longer running times.

Discussion The main future work is to improve the efficiency of the log-linear models, so that the running times are comparable to the standard Marginal filter. This might include working on a new representation of the linear model so that the operations required by the marginal filter can be executed in less time.

The current approximation using log-linear models relies on the fact that the observation model "dominates" the duration model. It must be investigated if and how approximating the starting times distribution using log-linear models removes the influence of the duration model on the particle's weights. The duration model might then only serve as a constraint on the action duration, i. e. specifying minimum and maximum durations for the actions. There is a trade-off to be made between the accuracy of the duration model (increasing the recognition accuracy) and representing more possible states (also potentially increasing the recognition accuracy).

Furthermore, the algorithm is currently only applicable to single-user models and must be extended to multi-user models. How the linear-models can be extended to a multi-user setting is currently unclear.

6.3.2 State-space reduction

Problem statement If detailed states (e. g. location of specific kitchen utensils) and actions (e. g. what ingredient is currently cut) are to be recognised, the model needs to be likewise very detailed. This results in a very large state space of the state-action transition system (ignoring action durations for now). Due to the freedom of actions, the branching factor can be very high (e. g. the user always has a plethora of different actions to select from) leading to the state space explosion problem. As the sensors cannot sense the complete state, it is often possible that many states lead to identical sensor observations. For the rest of this section, these are called *observation-equivalent* states.

If states are observation-equivalent, they cannot be distinguished by the inference. If they cannot be distinguished, it might be feasible to not use multiple particles, but use a single particle to represent each equivalence class. This might lead to a considerable reduction in the need of particles, thus potentially reducing the run-time, without increasing the approximation error.

Related work Lifted inference [101, 157, 179] is an approach to reason over the probabilities of quantified states (e. g. *take* X). This is done by exploiting symmetries in the probabilistic model. A disadvantage is that only existing structural symmetries based on the model can be used (e. g. the transition and observation distributions are identical for all objects x of the action *take* X). Observation-equivalence between unrelated actions (e. g. *take* X and *put* Y) cannot be found.

Lüdtke et al. [129] have recently applied the idea of lifted inference and applied it to the filtering task. Their solution is an exact inference algorithm.

In the models we used for this dissertation, the transition distribution is based on a state-transition system (the state s_t depends on the previous states s_{t-1} and the action a_t). In a generative model-based approach, this transition system is described by generative actions, i. e. as functions of the previous state. If such a description is available and can be expressed using Petri Nets, a wider set of symmetries can potentially be found than by just looking at the probabilistic model [185].

Decision diagrams [11] are a tool to compactly assign values to sets of states. They are particular compelling as also the transition function can be compactly represented using decision diagrams, with algorithms for applying the transition function at the same time for all states of a decision diagram. Although decision diagrams have been successfully applied to POMDPs (which are a tool for selection best actions based on noisy sensor observations), it has been noted by Poupart [159, p. 101] that only reward functions and conditional probability tables (similarly to state transitions) benefit from a compact representation. It could also be observed that the belief state generates a large ADD due to very different values.

Discussion It is important to evaluate how the ideas of lifted inference and compact state representation can be leveraged to recognising everyday situations. It is open if the exact approach of Lüdtke et al. [129] scales to large and complex models, where symmetries are potentially rare.

In the models I have used for this dissertation, only few symmetries could be found using LoLA, a state-of-the-art tool for model-checking [109, 186].

A main open question is to what kind of states should be considered equal. The mostly available approach is based on lifted inference and only considers states equal that are modelled to be equal (e. g. the definition of all actions do not distinguish between particular instances of objects). These symmetries are probably rare and provide only local optimisations. The most benefit can probably be achieved from models with a large number of users, e. g. in public surveillance applications.

Another idea is to include the observation model and collect all observation-equivalent states. This probably required additional information from the model designer to exactly specify when observations and states can be considered equivalent. How this can be automated, how the states can be compactly represented and if this actually increases the efficiency of the inference are open questions.

6.3.3 More future work

Continuous sub-models The marginal filter, the model described in Section 3.1 and the modelling language presented in Section 3.3 describe situations in terms of *symbolic* states and actions operating on these states. Remember from Section 2.1 that actions model the behaviour of users wrt. the context states. Also remember from Fig. 3.2 that actions can be inferred from observations, i. e. sensor measurements. The model allows to specify an observation distribution $p(Y_i \mid a_i)$ which describes the distribution of possible measurements given the current actions.

However, actions can be comprised of a series of physical motions and activities. Thus, a simple probability distribution $p(Y_i \mid a_i)$ might not be sufficient to correctly describe the possible measurements: during the course of executing the action, the observations can change. This is particularly important when observing physical motions, e. g. using accelerometers and gyroscopes (as has been done in both kitchen models used in this dissertation). For example, consider the action of taking some object out of a drawer: this can generally not been described by a uniform distribution of accelerometer values, as the user's hand will most likely perform several motions with turns and straight lines in different directions.

These motions might also be symbolically described using a large set of detailed actions with many preconditions and effects. But this will lead to very complex models and even larger state space sizes. Instead, these motions are best described using *continuous* models that have proven well in the literature [28, 194, 207].

It is currently unclear if and how these models can be integrated into the SMC framework and the Marginal filter in particular. There are three main challenges:

Modelling The probabilistic model must be extended to allow sub-models and unify them into a sound framework for Bayesian inference.

Modelling language The modelling language must be extended to allow to textually describe the models. Possibly training data must also be provided to enable automatic learning of the models.

Inference The inference of the SMC framework must be extended to allow inference in the sub-models for every particle. If the sub-models allow efficient inference, a Rao-Blackwellisation scheme can be applied for improved efficiency [30, 59].

Task structures The abstraction level of actions can be too low when there is the need to model many different complex behaviours of users. For this purpose, a hierarchical modelling can be used. Hierarchical models have been successfully applied in the field of Human Computer Interaction, with ConcurTaskTrees (CTTs) as the most important one [151]. This notation has also been extended to specifically model user and team behaviour in smart environments [72, 213]. On the other hand, Hierarchical Task Networks (HTNs) have been successfully applied to planning domains and are more expressive than PDDL [65], the foundation of the modelling language used for the models in this dissertation (Section 3.3). A restricted set of HTNs can also be automatically translated to PDDL [4]. Future work needs to evaluate if these approaches can introduce more structured knowledge into the model, and how the knowledge can be used by the inference for improved efficiency. High-level task models may also alleviate the need for checking long-term causations and other high-level properties as discussed in Chapter 4 for model checking.

Efficient data structures The prediction step of the Marginal filter maintains a set of all particles. Whenever a new particle is found, it is checked if a particle with the same state has already been predicted and added to the set. If yes, the particles are merged. This is the core of the marginal filter and described in lines 20–25 in Algorithm 8 (page 122).

The implementation uses a standard hash map implementation (from the C++ STL) for this purpose. Using a profiler revealed that the implementation spends over a third of the total run time for these lines. It is thus important to investigate this performance bottle neck. The issue might be a poorly performing hash function with either many collisions or just expensive to evaluate.

Exploiting independences in the state For complex models with multiple users and parallel actions, not all context state variables depend on each other. For instance, users might be in different rooms not interacting with each other. For this case, it is inefficient to estimate a joint distribution $p(s_1, s_2, \ldots, s_n \mid y)$ when estimating the individual distributions $p(s_1 \mid y), \ldots p(s_n \mid y)$ is also possible. Estimating a joint distribution requires exponentially more particles for SMC methods. Furthermore, these individual distributions can also be estimated in parallel.

This idea leads to a couple of research questions:

- How can such independences be detected or modelled? Is an automatic approach viable? Model-checking techniques as discussed in Chapter 4 might be viable.

- Can independences be assumed as an approximation? How good can such an approximation be?
- Can these independences be dynamic? For instance, if the users eventually meet in a single room, the individual distributions need to be transformed to a joint distribution. How can such situations be detected?

List of Figures

List of Tables

List of Listings

Bibliography

[1] van der Aalst, W. M. P. (1998). The application of Petri nets to workflow management. *Journal of Circuits, Systems and Computers*, 08(01):21–66. DOI: 10.1142/s0218126698000043.
(cited on page 95)

[2] Aehnelt, M. and Bader, S. (2015). Information assistance for smart assembly stations. In *Proceedings of the International Conference on Agents and Artificial Intelligence*, pages 143–150, Lisbon, Portugal. Scitepress. DOI: 10.5220/0005216501430150.
(cited on page 26)

[3] Aggarwal, J. K. and Ryoo, M. S. (2011). Human activity analysis: A review. *ACM Computing Surveys*, 43(3):16:1–16:43. DOI: 10.1145/1922649.1922653.
(cited on pages 18, 33–35, 38)

[4] Alford, R., Kuter, U., and Nau, D. S. (2009). Translating HTNs to PDDL: A small amount of domain knowledge can go a long way. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1629–1634.
(cited on page 157)

[5] Anderson, J. R. (1996). ACT: A simple theory of complex cognition. *American Psychologist*, 51(4):355–365. DOI: 10.1037/0003-066x.51.4.355.
(cited on page 54)

[6] Armentano, M. G. and Amandi, A. (2009). Goal recognition with variable-order markov models. In *Proceedings of the 21st International Joint Conference on Artifical intelligence*, volume 9 of *IJCAI'09*, pages 1635–1640, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
(cited on page 42f.)

[7] Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *Signal Processing, IEEE Transactions on*, 50(2):174–188. DOI: 10.1109/78.978374.
(cited on pages 72f., 75, 113f., 118)

[8] Aström, K. J. and Murray, R. M. (2012). *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press. online version.
(cited on page 18)

[9] Austin, J. T. and Vancouver, J. B. (1996). Goal constructs in psychology: Structure, process, and content. *Psychological Bulletin*, 120(3):338–375. DOI: 10.1037/0033-2909.120.3.338.
(cited on pages 16, 35)

[10] Bader, S., Krüger, F., and Kirste, T. (2015). Computational causal behaviour models for assisted manufacturing. In *Proceedings of the 2nd International Workshop on Sensor-based Activity Recognition and Interaction (WOAR)*, Rostock, Germany. Association for Computing Machinery (ACM). DOI: 10.1145/2790044.2790058.
(cited on page 77)

[11] Bahar, R. I., Frohm, E. A., Gaona, C. M., Hachtel, G. D., Macii, E., Pardo, A., and Somenzi, F. (1997). Algebric decision diagrams and their applications. *Formal methods in system design*, 10(2-3):171–206.
(cited on page 155)

[12] Baker, C. L., Saxe, R., and Tenenbaum, J. B. (2009). Action understanding as inverse planning. *Cognition*, 113(3):329–349. DOI: 10.1016/j.cognition.2009.07.005.
(cited on pages 35, 42–44, 49, 52)

[13] Baldauf, M., Dustdar, S., and Rosenberg, F. (2007). A survey on context-aware systems. *Int. J. Ad Hoc and Ubiquitous Computing*, 2(4):263.
(cited on pages 16f., 32, 38)

[14] Banavar, G., Beck, J., Gluzberg, E., Munson, J., Sussman, J., and Zukowski, D. (2000). Challenges: An application model for pervasive computing. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, MobiCom '00, pages 266–274, New York, NY, USA. ACM. DOI: 10.1145/345910.345957.
(cited on page 16)

[15] Baum, C. and Edwards, D. F. (1993). Cognitive performance in senile dementia of the alzheimer's type: The kitchen task assessment. *American Journal of Occupational Therapy*, 47(5):431–436. DOI: 10.5014/ajot.47.5.431.
(cited on page 46)

[16] Baxter, R., Lane, D., and Petillot, Y. (2010). Recognising agent behaviour during variable length activities. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 803–808, Amsterdam, The Netherlands, The Netherlands. IOS Press.
(cited on page 33)

[17] Bengtsson, T., Bickel, P., and Li, B. (2008). Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. *Probability and Statistics: Essays in Honor of David A. Freedman*, page 316–334. DOI: 10.1214/193940307000000518.
(cited on pages 18, 75, 77, 105, 113)

[18] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., and Riboni, D. (2010). A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2):161–180.
(cited on pages 31, 38)

[19] Bobick, A. F. (1997). Movement, activity and action: the role of knowledge in the perception of motion. *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, 352(1358):1257–1265. DOI: 10.1098/rstb.1997.0108.
(cited on page 33)

[20] Bouchard, B., Bouzouane, A., and Giroux, S. (2007). A keyhole plan recognition model for alzheimer's patients: first results. *Applied Artificial Intelligence*, 21(7):623–658.
(cited on pages 42f., 48)

[21] Bouchard, G. and Triggs, B. (2004). The tradeoff between generative and discriminative classifiers. In *16th IASC International Symposium on Computational Statistics (COMPSTAT)*, pages 721–728, Prague, Czech Republic.
(cited on page 64)

[22] Bourque, P. and Fairley, R. E., editors (2014). *Guide to the Software Engineering Body of Knowledge (SWEBOK®)*. IEEE Computer Society, 3 edition.
(cited on page 88)

[23] Bratman, M. E. (1999). *Intention, Plans and Practical Reason*. The Center for the Study of Language and Information Publications.
(cited on pages 32, 36)

[24] Bredeweg, B. and Struss, P. (2003). Current topics in qualitative reasoning. *AI Magazine*, 24(4):13–16.
(cited on page 55)

[25] Brown, P. J., Bovey, J. D., and Chen, X. (1997). Context-aware applications: from the laboratory to the marketplace. *IEEE Personal Communications*, 4(5):58–64. DOI: 10.1109/98.626984.
(cited on page 32)

[26] Bruno, B., Mastrogiovanni, F., Sgorbissa, A., Vernazza, T., and Zaccaria, R. (2012). Human motion modelling and recognition: A computational approach. In *Automation Science and Engineering (CASE), 2012 IEEE International Conference on*, pages 156–161. IEEE.
(cited on pages 33, 42f., 50, 55)

[27] Bui, H. H., Phung, D., Venkatesh, S., and Phan, H. (2008). The hidden permutation model and location-based activity recognition. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 3*, AAAI'08, pages 1345–1350. AAAI Press.
(cited on pages 42f., 50)

[28] Bulling, A., Blanke, U., and Schiele, B. (2014). A tutorial on human activity recognition using body-worn inertial sensors. *ACM Comput. Surv.*, 46(3):33:1–33:33. DOI: 10.1145/2499621.
(cited on page 156)

[29] Caramiaux, B., Montecchio, N., Tanaka, A., and Bevilacqua, F. (2014). Adaptive gesture recognition with variation estimation for interactive systems. *ACM Trans. Interact. Intell. Syst.*, 4(4):18:1–18:34. DOI: 10.1145/2643204.
(cited on page 75)

[30] Casella, G. and Robert, C. P. (1996). Rao-blackwellisation of sampling schemes. *Biometrika*, 83(1):81–94.
(cited on page 157)

[31] Caspersen, C. J., Powell, K. E., and Christenson, G. M. (1985). Physical activity, exercise, and physical fitness: definitions and distinctions for health-related research. *Public health reports*, 100(2):126–131.
(cited on page 33)

[32] Chang, I.-C. and Lin, S.-Y. (2010). 3D human motion tracking based on a progressive particle filter. *Pattern Recognition*, 43(10):3621 – 3635. DOI: 10.1016/j.patcog.2010.05.003.
(cited on page 33)

[33] Chaquet, J. M., Carmona, E. J., and Fernández-Caballero, A. (2013). A survey of video datasets for human action and activity recognition. *Computer Vision and Image Understanding*, 117(6):633 – 659. DOI: 10.1016/j.cviu.2013.01.013.
(cited on pages 33, 35)

[34] Chen, G. and Kotz, D. (2000). A survey of context-aware mobile computing research. Technical Report TR2000-381, Department of Computer Science, Dartmouth College, Hanover, NH, USA.
(cited on page 31)

[35] Chen, L., Hoey, J., Nugent, C. D., Cook, D. J., and Yu, Z. (2012a). Sensor-based activity recognition. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(6):790–808. DOI: 10.1109/TSMCC.2012.2198883.
(cited on pages 17, 29, 33, 36, 50, 56, 64)

[36] Chen, L., Nugent, C. D., and Wang, H. (2012b). A knowledge-driven approach to activity recognition in smart homes. *IEEE Trans. on Knowl. and Data Eng.*, 24(6):961–974. DOI: 10.1109/TKDE.2011.51.
(cited on pages 42f., 50)

[37] Cheng, H.-T., Griss, M., Davis, P., Li, J., and You, D. (2013). Towards zero-shot learning for human activity recognition using semantic attribute sequence model. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 355–358, New York, NY, USA. ACM. DOI: 10.1145/2493432.2493511.
(cited on page 42f.)

[38] Chávez, E., Ide, R., and Kirste, T. (1999). Interactive applications of personal situation-aware assistants. *Computers & Graphics*, 23(6):903–915. DOI: 10.1016/S0097-8493(99)00121-1.
(cited on pages 16, 31)

[39] Cimatti, A., Pistore, M., Roveri, M., and Traverso, P. (2003). Weak, strong, and strong cyclic planning via symbolic model checking. *Artificial Intelligence*, 147(1-2):35–84. DOI: 10.1016/S0004-3702(02)00374-0.
(cited on page 90)

[40] Clarke, Jr., E. M., Grumberg, O., and Peled, D. A. (1999). *Model Checking*. MIT Press.
(cited on pages 88, 90)

[41] Clarke, Jr., E. M. and Veith, H. (2003). *Verification: Theory and Practice*, volume 2772 of *Lecture Notes in Computer Science*, chapter Counterexamples Revisited: Principles, Algorithms, Applications, pages 208–224. Springer Berlin Heidelberg, Berlin, Heidelberg. DOI: 10.1007/978-3-540-39910-0_9.
(cited on page 94)

[42] Cole, M. (1995). The supra-individual envelope of development activity and practice, situation and context. *New Directions for Child and Adolescent Development*, 1995(67):105–118. DOI: 10.1002/cd.23219956712.
(cited on page 30)

[43] Cook, D. J. and Das, S. K. (2007). How smart are our environments? An updated look at the state of the art. *Pervasive and mobile computing*, 3(2):53–73.
(cited on page 17)

[44] Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2):393–405. DOI: 10.1016/0004-3702(90)90060-D.
(cited on page 71)

[45] Corrigan, R. and Denton, P. (1996). Causal understanding as a developmental primitive. *Developmental Review*, 16(2):162–202. DOI: 10.1006/drev.1996.0007.
(cited on page 49)

[46] Coutaz, J., Crowley, J. L., Dobson, S., and Garlan, D. (2005). Context is key. *Commun. ACM*, 48(3):49–53. DOI: 10.1145/1047671.1047703.
(cited on page 31f.)

[47] Crassidis, J. L. and Junkins, J. L. (2011). *Optimal Estimation of Dynamic Systems*, volume 24 of *Applied Mathematics & Nonlinear Science*. Chapman and Hall/CRC press, 2nd edition.
(cited on page 18)

[48] Dagum, P. and Luby, M. (1993). Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141 – 153. DOI: 10.1016/0004-3702(93)90036-B.
(cited on page 71)

[49] Dai, P., Di, H., Dong, L., Tao, L., and Xu, G. (2008). Group interaction analysis in dynamic context. *Trans. Sys. Man Cyber. Part B*, 38(1):275–282. DOI: 10.1109/TSMCB.2007.909939.
(cited on pages 42f., 46, 48f., 55, 80)

[50] Das, S. K., Cook, D. J., Battacharya, A., Heierman, III, E. O., and Lin, T.-Y. (2002). The role of prediction algorithms in the mavhome smart home architecture. *Wireless Commun.*, 9(6):77–84. DOI: 10.1109/MWC.2002.1160085.
(cited on page 17)

[51] Dash, D., Voortman, M., and de Jongh, M. (2013). Sequences of mechanisms for causal reasoning in artificial intelligence. In *Proc. 23rd International Joint Conference on Artificial Intelligence*.
(cited on page 42f.)

[52] Daum, F. and Huang, J. (2011). Particle degeneracy: root cause and solution. In Kadar, I., editor, *Signal Processing, Sensor Fusion, and Target Recognition XX*. SPIE. DOI: 10.1117/12.877167.
(cited on page 118)

[53] Dellaert, F., Fox, D., Burgard, W., and Thrun, S. (1999). Monte carlo localization for mobile robots. In *Proceedings 1999 IEEE International Conference on Robotics and Automation*, pages 1322–1328. IEEE. DOI: 10.1109/robot.1999.772544.
(cited on page 110)

[54] Deutscher, J., Blake, A., and Reid, I. (2000). Articulated body motion capture by annealed particle filtering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '00)*, volume 2, pages 126–133. DOI: 10.1109/CVPR.2000.854758.
(cited on pages 33, 74, 118)

[55] Dewey, J. (1938). *Logic: The Theory of Inquiry*. Henry Holt.
(cited on page 30)

[56] Dey, A. K., Abowd, G. D., and Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16(2):97–166. DOI: 10.1207/S15327051HCI16234_02.
(cited on page 31f.)

[57] Douc, R. and Cappé, O. (2005). Comparison of resampling schemes for particle filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis (ISPA 2005)*, pages 64–69. DOI: 10.1109/ISPA.2005.195385.
(cited on pages 74f., 128f.)

[58] Doucet, A. (1998). On sequential simulation-based methods for bayesian filtering. Technical Report CUED/F-INFENG/TR 310, Department of Engineering, Cambridge University.
(cited on pages 72f., 113, 118)

[59] Doucet, A., de Freitas, N., Murphy, K., and Russell, S. (2000a). Rao-Blackwellised particle filtering for dynamic bayesian networks. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI 2000)*, pages 176–183, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
(cited on pages 65, 157)

[60] Doucet, A., Godsill, S., and Andrieu, C. (2000b). On sequential monte carlo sampling methods for bayesian filtering. *Statistics and Computing*, 10(3):197–208. DOI: 10.1023/A:1008935410038.
(cited on pages 75, 104)

[61] Drechsler, R. and Becker, B. (1998). *Binary Decision Diagrams: Theory and Implementation*. Springer. DOI: 10.1007/978-1-4757-2892-7_3.
(cited on page 90)

[62] Dubuisson, S., Gonzales, C., and Nguyen, X. S. (2012). DBN-based combinatorial resampling for articulated object tracking. In *Proceedings of the 28th Conference on Uncertainty in Artificial Intelligence (UAI 2012)*, pages 237–246.
(cited on pages 74f., 105, 118)

[63] Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., and Burgelman, J.-C. (2001). Scenarios for ambient intelligence in 2010. IST Advisory Group, European Commision.
(cited on page 17)

[64] Duong, T., Phung, D., Bui, H., and Venkatesh, S. (2009). Efficient duration and hierarchical modeling for human activity recognition. *Artificial Intelligence*, 173(7-8):830–856. DOI: 10.1016/j.artint.2008.12.005.
(cited on pages 35, 42f., 48, 80, 152)

[65] Erol, K., Hendler, J., and Nau, D. S. (1994). Htn planning: complexity and expressivity. In *Proceedings of the twelfth national conference on Artificial intelligence*, volume 2 of *AAAI'94*, pages 1123–1128, Menlo Park, CA, USA. American Association for Artificial Intelligence.
(cited on page 157)

[66] Fagundes, M. S., Meneguzzi, F., Bordini, R. H., and Vieira, R. (2014). Dealing with ambiguity in plan recognition under time constraints. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, pages 389–396, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.
(cited on page 42f.)

[67] Fahland, D., Favre, C., Jobstmann, B., Koehler, J., Lohmann, N., Völzer, H., and Wolf, K. (2009). Instantaneous soundness checking of industrial business process models. In Dayal, U., Eder, J., Koehler, J., and Reijers, H., editors, *Business Process Management (BPM)*, volume 5701 of *Lecture Notes in Computer Science (LNCS)*, pages 278–293. Springer Nature. DOI: 10.1007/978-3-642-03848-8_19.
(cited on page 95)

[68] Fearnhead, P. and Clifford, P. (2003). On-line inference for Hidden Markov Models via particle filters. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 65(4):887–899. DOI: 10.1111/1467-9868.00421.
(cited on pages 72, 75, 105, 126, 129, 135f.)

[69] Feldhege, F., Mau-Möller, A., Lindner, T., Hein, A., Markschies, A., Zettl, U., and Bader, R. (2015). Accuracy of a custom physical activity and knee angle measurement sensor system for patients with neuromuscular disorders and gait abnormalities. *Sensors*, 15(5):10734–10752. DOI: 10.3390/s150510734.
(cited on page 33)

[70] Fleck, S., Busch, F., Biber, P., and Straber, W. (2006). 3D surveillance – a distributed network of smart cameras for real-time tracking and its visualization in 3D. In *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*. IEEE. DOI: 10.1109/cvprw.2006.6.
(cited on pages 73, 75, 105)

[71] Fontmarty, M., Lerasle, F., and Danès, P. (2007). Data fusion within a modified annealed particle filter dedicated to human motion capture. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 3391–3396. DOI: 10.1109/IROS.2007.4399521.
(cited on pages 75, 105, 110)

[72] Forbrig, P. and Buchholz, G. (2017). Subject-oriented specification of smart environments. In *Proceedings of the 9th Conference on Subject-oriented Business Process Management, S-BPM ONE*, page 8. DOI: 10.1145/3040565.3040570.
(cited on page 157)

[73] Gayathri, K. S., Elias, S., and Ravindran, B. (2014). Hierarchical activity recognition for dementia care using Markov logic network. *Pers Ubiquit Comput*, 19(2):271–285. DOI: 10.1007/s00779-014-0827-7.
(cited on pages 42f., 56)

[74] Geib, C. W. and Goldman, R. P. (2009). A probabilistic plan recognition algorithm based on plan tree grammars. *Artificial Intelligence*, 173(11):1101–1132. DOI: 10.1016/j.artint.2009.01.003.
(cited on pages 42f., 47)

[75] Gerber, M. and Chopin, N. (2015). Sequential quasi Monte Carlo. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77(3):509–579. DOI: 10.1111/rssb.12104.
(cited on page 118)

[76] Giersich, M. (2009). *Concept of a Robust & Training-free Probabilistic System for Real-time Intention Analysis in Teams*. PhD thesis, University of Rostock, Rostock, Germany.
(cited on page 80)

[77] Godefroid, P. (1996). *Partial-Order Methods for the Verification of Concurrent Systems: An Approach to the State-Explosion Problem*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.
(cited on page 90)

[78] Grimm, R., Anderson, T., Bershad, B., and Wetherall, D. (2000). A system architecture for pervasive computing. In *Proceedings of the 9th Workshop on ACM SIGOPS European Workshop: Beyond the PC: New Challenges for the Operating System*, EW 9, pages 177–182, New York, NY, USA. ACM. DOI: 10.1145/566726.566763.
(cited on page 17)

[79] Gromyko, A., Pistore, M., and Traverso, P. (2006). A tool for controller synthesis via symbolic model checking. In *Discrete Event Systems, 2006 8th International Workshop on*, pages 475–476. DOI: 10.1109/WODES.2006.382523.
(cited on page 90)

[80] Haddadi, A. and Sundermeyer, K. (1996). Belief-desire-intention agent architecture. In *Foundations of Distributed Artificial Intelligence*. John Wiley & Sons Inc.
(cited on page 36)

[81] Han, T. A. and Pereira, L. M. (2010). Anytime intention recognition via incremental bayesian network reconstruction. In *2010 AAAI Fall Symposium Series*.
(cited on page 42f.)

[82] Hardegger, M., Roggen, D., Calatroni, A., and Tröster, G. (2016). S-SMART: A unified bayesian framework for simultaneous semantic mapping, activity recognition, and tracking. *ACM Trans. Intell. Syst. Technol.*, 7(3):34:1–34:28. DOI: 10.1145/2824286.
(cited on pages 42f., 48)

[83] Hein, A., Krüger, F., Bader, S., Eschholz, P., and Kirste, T. (2017). Challenges of collecting empirical sensor data from people with dementia in a field study. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*. accepted.
(cited on page 92)

[84] Henricksen, K. and Indulska, J. (2006). Developing context-aware pervasive computing applications: Models and approach. *Pervasive and Mobile Computing*, 2(1):37–64. DOI: 10.1016/j.pmcj.2005.07.003.
(cited on page 17)

[85] Hiatt, L. M., Harrison, A. M., and Trafton, J. G. (2011). Accommodating human variability in human-robot teams through theory of mind. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence*, IJCAI'11, pages 2066–2071. AAAI Press. DOI: 10.5591/978-1-57735-516-8/IJCAI11-345.
(cited on pages 17, 42f.)

[86] Hoey, J., Boutilier, C., Poupart, P., Olivier, P., Monk, A., and Mihailidis, A. (2013). People, sensors, decisions: Customizable and adaptive technologies for assistance in healthcare. *ACM Trans. Interact. Intell. Syst.*, 2(4):20:1–20:36. DOI: 10.1145/2395123.2395125.
(cited on pages 17, 80)

[87] Hoey, J., Plötz, T., Jackson, D., Monk, A., Pham, C., and Olivier, P. (2011). Rapid specification and automated generation of prompting systems to assist people with dementia. *Pervasive and Mobile Computing*, 7(3):299–318. DOI: 10.1016/j.pmcj.2010.11.007.
(cited on pages 32, 42f., 56, 58, 149)

[88] Hoey, J., Poupart, P., von Bertoldi, A., Craig, T., Boutilier, C., and Mihailidis, A. (2010). Automated handwashing assistance for persons with dementia using video and a partially observable markov decision process. *Computer Vision and Image Understanding*, 114(5):503–519. DOI: 10.1016/j.cviu.2009.06.008.
(cited on pages 32, 42f., 48, 71, 80)

[89] Hu, D. H. and Yang, Q. (2008). Cigar: Concurrent and interleaving goal and activity recognition. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 3*, volume 8 of *AAAI'08*, pages 1363–1368, Chicago, Illinois. AAAI Press.
(cited on page 42f.)

[90] Incel, O. D., Kose, M., and Ersoy, C. (2013). A review and taxonomy of activity recognition on mobile phones. *BioNanoScience*, 3(2):145–171. DOI: 10.1007/s12668-013-0088-3.
(cited on pages 33, 38)

[91] Isard, M. and Blake, A. (1998). Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28. DOI: 10.1023/A:1008078328650.
(cited on pages 18, 75, 105, 110)

[92] Kalra, L., Zhao, X., Soto, A. J., and Milios, E. (2013). Detection of daily living activities using a two-stage markov model. *Journal of Ambient Intelligence and Smart Environments (JAISE)*, 5(3):273–285. DOI: 10.3233/AIS-130208.
(cited on pages 34, 42f., 48, 56)

[93] Kasparick, M. and Krüger, F. (2015). Probabilistic action selection - tracking multiple persons in indoor environments. data set, University of Rostock. DOI: 10.18453/rosdok_id00000114.
(cited on page 81)

[94] van Kasteren, T. L. M., Englebienne, G., and Kröse, B. J. A. (2010a). Activity recognition using semi-Markov models on real world smart home datasets. *Journal of Ambient Intelligence and Smart Environments (JAISE)*, 2(3):311–325. DOI: 10.3233/AIS-2010-0070.
(cited on pages 34, 64, 151f.)

[95] van Kasteren, T. L. M., Englebienne, G., and Kröse, B. J. A. (2010b). Transferring knowledge of activity recognition across sensor networks. In *Proceedings of the 8th international conference on Pervasive Computing*, Pervasive'10, pages 283–300, Berlin, Heidelberg. Springer-Verlag. DOI: 10.1007/978-3-642-12654-3_17.
(cited on pages 47, 64)

[96] van Kasteren, T. L. M., Noulas, A., Englebienne, G., and Kröse, B. J. A. (2008). Accurate activity recognition in a home setting. In *Proceedings of the 10th international conference on Ubiquitous computing*, UbiComp '08, pages 1–9, New York, NY, USA. ACM, ACM. DOI: 10.1145/1409635.1409637.
(cited on page 50)

[97] Katz, S. (1983). Assessing self-maintenance: Activities of daily living, mobility, and instrumental activities of daily living. *Journal of the American Geriatrics Society*, 31(12):721–727. DOI: 10.1111/j.1532-5415.1983.tb03391.x.
(cited on page 17)

[98] Kautz, H. A. (1987). *A formal theory of plan recognition*. PhD thesis, Bell Laboratories.
(cited on pages 17, 35)

[99] Kembhavi, A., Schwartz, W. R., and Davis, L. S. (2008). Resource allocation for tracking multiple targets using particle filters. In Jones, G., Tan, T., Maybank, S., and Makris, D., editors, *The Eighth International Workshop on Visual Surveillance - VS2008*, Marseille, France.
(cited on pages 75, 105)

[100] Kern, N., Schiele, B., Junker, H., Lukowicz, P., and Tröster, G. (2003). Wearable sensing to annotate meeting recordings. *Personal Ubiquitous Comput.*, 7(5):263–274. DOI: 10.1007/s00779-003-0242-y.
(cited on page 80)

[101] Kersting, K. (2012). Lifted probabilistic inference. In De Raedt, L., Bessiere, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., and Lucas, P., editors, *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI '12)*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 33–38. IOS Press. DOI: 10.3233/978-1-61499-098-7-33.
(cited on page 155)

[102] Kiefer, P. and Stein, K. (2008). A framework for mobile intention recognition in spatially structured environments. In *Workshop on Behaviour Monitoring and Interpretation*, pages 28–41.
(cited on pages 42f., 45–48)

[103] Kim, E., Helal, S., Nugent, C., and Beattie, M. (2015). Analyzing activity recognition uncertainties in smart home environments. *ACM Trans. Intell. Syst. Technol.*, 6(4):52:1–52:28. DOI: 10.1145/2651445.
(cited on page 57)

[104] Kirste, T. (2015a). Anonymous activity recognition in an office environment (office tasks dataset). data set, University of Rostock. DOI: 10.18453/rosdok_id00000113.
(cited on page 80)

[105] Kirste, T. (2015b). Detecting high-level team intentions (three person meeting dataset). data set, University of Rostock. DOI: 10.18453/rosdok_id00000112.
(cited on page 81)

[106] Klaas, M., de Freitas, N., and Doucet, A. (2005). Toward practical N^2 Monte Carlo: the marginal particle filter. In *Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence (UAI 2005)*, pages 308–315, Edinburgh, Scotland. AUAI Press.
(cited on pages 73, 118f.)

[107] Kok, M., Hol, J. D., and Schön, T. B. (2015). Indoor positioning using ultrawideband and inertial measurements. *IEEE Transactions on Vehicular Technology*, 64(4):1293–1303. DOI: 10.1109/TVT.2015.2396640.
(cited on pages 18, 55)

[108] Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. Adaptive Computation and Machine Learning series. The MIT Press.
(cited on pages 20, 64, 71)

[109] Kordon, F., Garavel, H., Hillah, L. M., Hulin-Hubard, F., Amparore, E., Beccuti, M., Berthomieu, B., Ciardo, G., Dal Zilio, S., Liebke, T., Linard, A., Meijer, J., Miner, A., Srba, J., Thierry-Mieg, Y., van de Pol, J., and Wolf, K. (2018). Complete Results for the 2018 Edition of the Model Checking Contest. http://mcc.lip6.fr/2018/results.php.
(cited on page 156)

[110] Kordon, F., Linard, A., Beccuti, M., Buchs, D., Fronc, Ł., Hillah, L.-M., Hulin-Hubard, F., Legond-Aubry, F., Lohmann, N., Marechal, A., Paviot-Adet, E., Pommereau, F., Rodríguez, C., Rohr, C., Thierry-Mieg, Y., Wimmel, H., and Wolf, K. (2013). Model checking contest @ petri nets, report on the 2013 edition. arXiv.
(cited on page 88)

[111] Krüger, F. (2016). *Activity, Context, and Plan Recognition with Computational Causal Behavior Models*. PhD thesis, University of Rostock.
(cited on pages 15, 20–22, 27, 53–55, 64f., 67–69, 77, 103f., 119)

[112] Krüger, F., Hein, A., Yordanova, K., and Kirste, T. (2015). Recognising the actions during cooking task (cooking task dataset). data set, University of Rostock. DOI: 10.18453/rosdok_id00000116.
(cited on page 82)

[113] Krüger, F., Nyolt, M., Yordanova, K., Hein, A., and Kirste, T. (2014). Computational state space models for activity and intention recognition. A feasibility study. *PLoS One*. DOI: 10.1371/journal.pone.0109381.
(cited on pages 25, 27, 29, 59, 64, 77, 80, 82, 87, 103–105, 118f.)

[114] Krüger, F., Steiniger, A., Bader, S., and Kirste, T. (2012a). Evaluating the robustness of activity recognition using computational causal behavior models. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 1066–1074, New York, NY, USA. ACM. DOI: 10.1145/2370216.2370443.
(cited on page 87)

[115] Krüger, F., Yordanova, K., Burghardt, C., and Kirste, T. (2012b). Towards creating assistive software by employing human behavior models. *Journal of Ambient Intelligence and Smart Environments (JAISE)*, 4(3):209–226. DOI: 10.3233/AIS-2012-0148.
(cited on pages 75, 80)

[116] Krüger, F., Yordanova, K., Hein, A., and Kirste, T. (2013). Plan synthesis for probabilistic activity recognition. In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence (ICAART)*, pages 283 – 288, Barcelona, Spain. DOI: 10.5220/0004256002830288.
(cited on pages 42f., 49, 53, 77, 87)

[117] Kwiatkowska, M., Norman, G., and Parker, D. (2005). Probabilistic model checking in practice. *ACM SIGMETRICS Performance Evaluation Review*, 32(4):16–21. DOI: 10.1145/1059816.1059820.
(cited on page 93)

[118] Kwiatkowska, M., Norman, G., and Parker, D. (2011). PRISM 4.0: Verification of probabilistic real-time systems. In Gopalakrishnan, G. and Qadeer, S., editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer.
(cited on page 94)

[119] Kwisthout, J. H. P., Bodlaender, H. L., and van der Gaag, L. C. (2010). The necessity of bounded treewidth for efficient inference in bayesian networks. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 237–242, Amsterdam, The Netherlands. IOS Press.
(cited on page 64)

[120] Lara, Ó. D. and Labrador, M. A. (2013). A survey on human activity recognition using wearable sensors. *Communications Surveys Tutorials, IEEE*, 15(3):1192–1209. DOI: 10.1109/SURV.2012.110112.00192.
(cited on pages 17, 33, 38)

[121] Lasecki, W. S., Song, Y. C., Kautz, H., and Bigham, J. P. (2013). Real-time crowd labeling for deployable activity recognition. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 1203–1212, New York, NY, USA. ACM. DOI: 10.1145/2441776.2441912.
(cited on pages 42f., 47)

[122] Le Gal, C., Martin, J., Lux, A., and Crowley, J. L. (2001). Smartoffice: design of an intelligent environment. *Intelligent Systems, IEEE*, 16(4):60–66. DOI: 10.1109/5254.941359.
(cited on page 17)

[123] Lefferts, E., Markley, F., and Shuster, M. (1982). Kalman filtering for spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics*, 5(5):417–429. DOI: 10.2514/3.56190.
(cited on page 18)

[124] Levine, S. J. and Williams, B. C. (2014). Concurrent plan recognition and execution for human-robot teams. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*.
(cited on pages 42f., 48)

[125] Li, T., Sattar, T. P., and Sun, S. (2012). Deterministic resampling: Unbiased sampling to avoid sample impoverishment in particle filters. *Signal Processing*, 92(7):1637 – 1645. DOI: 10.1016/j.sigpro.2011.12.019.
(cited on page 118)

[126] Liao, L., Patterson, D. J., Fox, D., and Kautz, H. (2007). Learning and inferring transportation routines. *Artificial Intelligence*, 171(5-6):311–331. DOI: 10.1016/j.artint.2007.01.006.
(cited on pages 42f., 45, 48, 52f.)

[127] Liu, Y., Nie, L., Han, L., Zhang, L., and Rosenblum, D. S. (2015). Action2Activity: Recognizing complex activities from sensor data. In *Artificial Intelligence (IJCAI), 2015 International Joint Conference on*, pages 1617–1623.
(cited on pages 33f., 42f., 48, 50)

[128] Logan, B., Healey, J., Philipose, M., Tapia, E. M., and Intille, S. (2007). A long-term evaluation of sensing modalities for activity recognition. In *Proceedings of the 9th international conference on Ubiquitous computing*, UbiComp '07, pages 483–500, Berlin, Heidelberg. Springer-Verlag.
(cited on page 56)

[129] Lüdtke, S., Schröder, M., Bader, S., Kersting, K., and Kirste, T. (2018). Lifted filtering via exchangeable decomposition. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*.
(cited on page 155f.)

[130] Magherini, T., Fantechi, A., Nugent, C. D., and Vicario, E. (2013). Using temporal logic and model checking in automated recognition of human activities for ambient-assisted living. *Human-Machine Systems, IEEE Transactions on*, 43(6):509–521. DOI: 10.1109/TSMC.2013.2283661.
(cited on pages 34f., 42f., 48, 90)

[131] Mao, W., Gratch, J., and Li, X. (2012). Probabilistic plan inference for group behavior prediction. *Intelligent Systems, IEEE*, 27(4):27–36. DOI: 10.1109/MIS.2010.133.
(cited on pages 42f., 47, 55)

[132] Mark, W. (1999). Turning pervasive computing into mediated spaces. *IBM Systems Journal*, 38(4):677–692. DOI: 10.1147/sj.384.0677.
(cited on page 17)

[133] McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., and Wilkins, D. (1998). PDDL—the planning domain definition language. Technical Report CVC TR-98-003 / DCS TR-1165, Yale Center for Computational Vision and Control.
(cited on page 77)

[134] McGraw, K. O. and Wong, S. P. (1992). A common language effect size statistic. *Psychological Bulletin*, 111(2):361–365. DOI: 10.1037/0033-2909.111.2.361.
(cited on page 139)

[135] McKeever, S., Ye, J., Coyle, L., Bleakley, C., and Dobson, S. (2010). Activity recognition using temporal evidence theory. *Journal of Ambient Intelligence and Smart Environments (JAISE)*, 2(3):253–269. DOI: 10.3233/AIS-2010-0071.
(cited on page 152)

[136] McKim, J. C. (1996). Programming by contract. *Computer, IEEE*, 29(3):109–111. DOI: 10.1109/2.485902.
(cited on page 88)

[137] Meyer, B. (1992). Applying "Design by Contract". *Computer, IEEE*, 25(10):40–51. DOI: 10.1109/2.161279.
(cited on page 88)

[138] Murphy, K. P. (2002). *Dynamic Bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley.
(cited on pages 20, 71, 150)

[139] Musser, D. R. (1997). Introspective sorting and selection algorithms. *Software: Practice and Experience*, 27(8):983–993. DOI: 10.1002/(sici)1097-024x(199708)27:8<983::aid-spe117>3.0.co;2-.
(cited on page 125)

[140] Natarajan, P. and Nevatia, R. (2013). Hierarchical multi-channel hidden semi markov graphical models for activity recognition. *Computer Vision and Image Understanding*, 117(10):1329 – 1344. DOI: 10.1016/j.cviu.2012.08.011.
(cited on page 42f.)

[141] Newell, A. and Simon, H. A. (1976). Computer science as empirical inquiry: symbols and search. *Communications of the ACM*, 19(3):113–126. DOI: 10.1145/360018.360022.
(cited on page 54)

[142] Ng, A. Y. and Jordan, M. I. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In Dietterich, T. G., Becker, S., and Ghahramani, Z., editors, *Advances in Neural Information Processing Systems 14*, pages 841–848. MIT Press.
(cited on page 64)

[143] Nguyen, N., Venkatesh, S., and Bui, H. (2006). Recognising behaviours of multiple people with hierarchical probabilistic model and statistical data association. In *BMVC 2006: Proceedings of the 17th British Machine Vision Conference*, pages 1239–1248. British Machine Vision Association.
(cited on pages 42f., 48)

[144] Nguyen, N. T., Bui, H. H., Venkatesh, S., and West, G. (2003). Recognizing and monitoring high-level behaviors in complex spatial environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '03)*, volume 2, pages II–620–5. DOI: 10.1109/CVPR.2003.1211524.
(cited on pages 42f., 75, 105)

[145] Nguyen, T. A., Kambhampati, S., and Do, M. (2013). Synthesizing robust plans under incomplete domain models. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 2472–2480. Curran Associates, Inc.
(cited on page 23)

[146] Nyolt, M. and Kirste, T. (2015). On resampling for bayesian filters in discrete state spaces. In *Proc. International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE Computer Society. accepted.
(cited on pages 77, 103)

[147] Nyolt, M. and Kirste, T. (2017). Efficient SMC inference in Semi-Markov models by compressing the belief state. In *Proc. 3rd International Conference on Control, Automation and Robotics*.
(cited on page 153)

[148] Nyolt, M., Krüger, F., Yordanova, K., Hein, A., and Kirste, T. (2015a). Marginal filtering in large state spaces. *International Journal of Approximate Reasoning*, 61:16–32. DOI: 10.1016/j.ijar.2015.04.003.
(cited on pages 77, 103)

[149] Nyolt, M., Yordanova, K., and Kirste, T. (2015b). Checking models for activity recognition. In Loiseau, S., Filipe, J., Duval, B., and von den Herik, J., editors, *Proceedings of the International Conference on Agents and Artificial Intelligence*, Lisbon, Portugal. SCITEPRESS. DOI: 10.5220/0005275204970502.
(cited on page 85)

[150] Oh, J., Meneguzzi, F., and Sycara, K. (2014). Probabilistic plan recognition for proactive assistant agents. In *Plan, Activity, and Intent Recognition: Theory and Practice*. Newnes.
(cited on pages 42f., 46, 48)

[151] Paternò, F. (2000). *Model-based design and evaluation of interactive applications*. Springer Science & Business Media.
(cited on page 157)

[152] Patterson, D. J., Liao, L., Fox, D., and Kautz, H. (2003). Inferring high-level behavior from low-level sensors. In Dey, A., Schmidt, A., and McCarthy, J., editors, *UbiComp 2003: Ubiquitous Computing*, volume 2864 of *Lecture Notes in Computer Science*, pages 73–89. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-39653-6_6.
(cited on page 75)

[153] Pearson, R., Donnelly, M. P., Liu, J., and Galway, L. (2016). Generic application driven situation awareness via ontological situation recognition. In *2016 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*. Institute of Electrical & Electronics Engineers (IEEE). DOI: 10.1109/cogsima.2016.7497800.
(cited on page 42f.)

[154] Pelánek, R. (2009). Fighting state space explosion: Review and evaluation. In Cofer, D. and Fantechi, A., editors, *Formal Methods for Industrial Critical Systems*, volume 5596 of *Lecture Notes in Computer Science*, pages 37–52. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-03240-0_7.
(cited on page 90)

[155] Perera, C., Zaslavsky, A., Christen, P., and Georgakopoulos, D. (2014). Context aware computing for the internet of things: A survey. *Communications Surveys Tutorials, IEEE*, 16(1):414–454. DOI: 10.1109/SURV.2013.042313.00197.
(cited on pages 31, 38)

[156] Phua, C., Foo, V. S.-F., Biswas, J., Tolstikov, A., Aung, A.-P.-W., Maniyeri, J., Huang, W., That, M.-H., Xu, D., and Chu, A. K.-W. (2009). 2-layer erroneous-plan recognition for dementia patients in smart homes. In *Proceedings of the 11th international conference on e-Health networking, applications and services*, Healthcom'09, pages 21–28, Piscataway, NJ, USA. IEEE, IEEE Press.
(cited on pages 42f., 56)

[157] Poole, D. (2003). First order probabilistic inference. In *Proceedings of the 18th International Joint Conference on Artificial intelligence (IJCAI '03)*.
(cited on page 155)

[158] Poppe, R. (2010). A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990. DOI: 10.1016/j.imavis.2009.11.014.
(cited on page 33)

[159] Poupart, P. (2005). *Exploiting structure to efficiently solve large scale partially observable Markov decision processes*. PhD thesis, Citeseer.
(cited on page 155)

[160] Preece, S. J., Goulermas, J. Y., Kenney, L. P. J., Howard, D., Meijer, K., and Crompton, R. (2009). Activity identification using body-mounted sensors—a review of classification techniques. *Physiological Measurement*, 30(4):R1. DOI: 10.1088/0967-3334/30/4/R01.
(cited on pages 33, 35, 38)

[161] Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286. DOI: 10.1109/5.18626.
(cited on pages 18, 72)

[162] Ramírez, M. and Geffner, H. (2009). Plan recognition as planning. In *Proceedings of the 21st International Jont Conference on Artifical Intelligence*, IJCAI'09, pages 1778–1783, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
(cited on page 35)

[163] Ramírez, M. and Geffner, H. (2010). Probabilistic plan recognition using off-the-shelf classical planners. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI-10)*, Atlanta, Georgia, USA.
(cited on pages 42–44, 48f.)

[164] Ramírez, M. and Geffner, H. (2011). Goal recognition over POMDPs: inferring the intention of a POMDP agent. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, volume 3 of *IJCAI'11*, pages 2009–2014. AAAI Press. DOI: 10.5591/978-1-57735-516-8/IJCAI11-335.
(cited on pages 42–44, 49, 52)

[165] Ramos, C., Augusto, J. C., and Shapiro, D. (2008). Ambient intelligence—the next step for artificial intelligence. *Intelligent Systems, IEEE*, 23(2):15–18. DOI: 10.1109/MIS.2008.19.
(cited on page 17)

[166] Rashidi, P. and Mihailidis, A. (2013). A survey on ambient-assisted living tools for older adults. *Biomedical and Health Informatics, IEEE Journal of*, 17(3):579–590. DOI: 10.1109/JBHI.2012.2234129.
(cited on page 17)

[167] Rasmussen, J. (1983). Skills, rules, and knowledge; signals, signs, and symbols, and other distinctions in human performance models. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(3):257–266. DOI: 10.1109/TSMC.1983.6313160.
(cited on page 54)

[168] Rekimoto, J. and Nagao, K. (1995). The world through the computer. In *Proceedings of the 8th annual ACM symposium on User interface and software technology (UIST)*, pages 29–36. ACM. DOI: 10.1145/215585.215639.
(cited on page 16)

[169] Riboni, D., Bettini, C., Civitarese, G., Janjua, Z. H., and Helaoui, R. (2015). Fine-grained recognition of abnormal behaviors for early detection of mild cognitive impairment. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*. Institute of Electrical & Electronics Engineers (IEEE). DOI: 10.1109/percom.2015.7146521.
(cited on pages 42f., 48, 56)

[170] Rogge-Solti, A., Mans, R. S., van der Aalst, W. M., and Weske, M. (2013). Improving documentation by repairing event logs. In *Proceedings of the 6th IFIP WG 8.1 Working Conference, PoEM*, volume 165 of *Lecture Notes in Business Information Processing*, pages 129–144, Riga, Latvia.
(cited on pages 42f., 46, 50, 152)

[171] Roggen, D., Calatroni, A., Rossi, M., Holleczek, T., Forster, K., Troster, G., Lukowicz, P., Bannach, D., Pirkl, G., Ferscha, A., Doppler, J., Holzmann, C., Kurz, M., Holl, G., Chavarriaga, R., Sagha, H., Bayati, H., Creatura, M., and del R. Millán, J. (2010). Collecting complex activity datasets in highly rich networked sensor environments. In *Networked Sensing Systems (INSS), Seventh International Conference on*, pages 233–240. IEEE. DOI: 10.1109/INSS.2010.5573462.
(cited on page 47)

[172] Rohrbach, M., Amin, S., Andriluka, M., and Schiele, B. (2012). A database for fine grained activity detection of cooking activities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '12)*, pages 1194–1201. DOI: 10.1109/CVPR.2012.6247801.
(cited on page 80)

[173] Rose, C., Saboune, J., and Charpillet, F. (2008). Reducing particle filtering complexity for 3D motion capture using dynamic bayesian networks. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 3*, AAAI'08, pages 1396–1401. AAAI Press.
(cited on pages 75, 105, 110)

[174] Ross, S., Pineau, J., Paquet, S., and Chaib-Draa, B. (2008). Online planning algorithms for pomdps. *Journal of Artificial Intelligence Research (JAIR)*, 32:663–704.
(cited on page 71)

[175] Roth, D. (1996). On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1–2):273–302. DOI: 10.1016/0004-3702(94)00092-1.
(cited on page 71)

[176] Roy, P. C., Giroux, S., Bouchard, B., Bouzouane, A., Phua, C., Tolstikov, A., and Biswas, J. (2011). A possibilistic approach for activity recognition in smart homes for cognitive assistance to alzheimer's patients. In Chen, L., Nugent, C. D., Biswas, J., and Hoey, J., editors, *Activity Recognition in Pervasive Intelligent Environments*, volume 4 of *Atlantis Ambient and Pervasive Intelligence*, pages 33–58. Atlantis Press. DOI: 10.2991/978-94-91216-05-3_2.
(cited on pages 17, 33, 42f., 48, 56, 80)

[177] Rui, Y. and Chen, Y. (2001). Better proposal distributions: object tracking using unscented particle filter. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. (CVPR)*, volume 2, pages 786–793. Institute of Electrical & Electronics Engineers (IEEE). DOI: 10.1109/cvpr.2001.991045.
(cited on page 73)

[178] Sadilek, A. and Kautz, H. (2012). Location-based reasoning about complex multi-agent behavior. *Journal of Artificial Intelligence Research (JAIR)*, 43(1):87–133. DOI: 10.1613/jair.342.
(cited on pages 42f., 46)

[179] de Salvo Braz, R., Amir, E., and Roth, D. (2005). Lifted first-order probabilistic inference. In *Proceedings of the 19th International Joint Conference on Artificial intelligence (IJCAI '05)*.
(cited on page 155)

[180] Sánchez, D., Tentori, M., and Favela, J. (2008). Activity recognition for the smart hospital. *Intelligent Systems, IEEE*, 23(2):50–57. DOI: 10.1109/MIS.2008.18.
(cited on pages 34, 152)

[181] Schilit, B., Adams, N., and Want, R. (1994). Context-aware computing applications. In *Proceedings of the 1st Workshop on Mobile Computing Systems and Applications (WMCSA '94)*, pages 85–90, Washington, DC, USA. IEEE Computer Society. DOI: 10.1109/WMCSA.1994.16.
(cited on pages 16f., 31f.)

[182] Schmidt, A., Aidoo, K. A., Takaluoma, A., Tuomela, U., van Laerhoven, K., and van de Velde, W. (1999). Advanced interaction in context. In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, HUC '99, pages 89–101, London, UK, UK. Springer-Verlag. DOI: 10.1007/3-540-48157-5_10.
(cited on pages 18, 31)

[183] Schmidt, C. F. (1975). Understanding human action. In *Proceedings of the 1975 Workshop on Theoretical Issues in Natural Language Processing*, TINLAP '75, pages 196–200, Stroudsburg, PA, USA. Association for Computational Linguistics. DOI: 10.3115/980190.980240.
(cited on page 17)

[184] Schmidt, C. F., Sridharan, N. S., and Goodson, J. L. (1978). The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence*, 11(1):45–83. DOI: 10.1016/0004-3702(78)90012-7.
(cited on page 17)

[185] Schmidt, K. (2000a). How to calculate symmetries of Petri nets. *Acta Informatica*, 36(7):545. DOI: 10.1007/s002360050002.
(cited on pages 90, 155)

[186] Schmidt, K. (2000b). Lola: a low level analyser. In *Proceedings of the 21st international conference on Application and theory of petri nets*, pages 465–474. Springer-Verlag.
(cited on page 156)

[187] Schmidt, K. (2006). Automated generation of a progress measure for the sweep-line method. *International Journal on Software Tools for Technology Transfer*, 8(3):195. DOI: 10.1007/s10009-005-0201-1.
(cited on page 90)

[188] Schröder, M., Bader, S., Krüger, F., and Kirste, T. (2016). Reconstruction of everyday life behaviour based on noisy sensor data. In *Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART)*, Rome, Italy. Scitepress. DOI: 10.5220/0005756804300437.
(cited on page 77)

[189] Schwering, C., Beck, D., Schiffer, S., and Lakemeyer, G. (2012). Plan recognition by program execution in continuous temporal domains. In *KI 2012: Advances in Artificial Intelligence*, pages 156–167. Springer. DOI: 10.1007/978-3-642-33347-7_14.
(cited on page 42f.)

[190] Shi, Y., Huang, Y., Minnen, D., Bobick, A., and Essa, I. (2004). Propagation networks for recognition of partially ordered sequential action. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '04)*, pages 862–869, Washington, DC, USA. IEEE Computer Society. DOI: 10.1109/CVPR.2004.1315255.
(cited on pages 33, 42f., 55, 59, 61, 118f., 124)

[191] Shin, K. G. and Ramanathan, P. (1994). Real-time computing: a new discipline of computer science and engineering. *Proceedings of the IEEE*, 82(1):6–24. DOI: 10.1109/5.259423.
(cited on page 53)

[192] Simon, H. A. (1990). Invariants of human behavior. *Annual Review of Psychology*, 41(1):1–19. DOI: 10.1146/annurev.ps.41.020190.000245.
(cited on page 54)

[193] Sloman, S. A. (1996). The empirical case for two systems of reasoning. *Psychological Bulletin*, 119(1):3–22. DOI: 10.1037/0033-2909.119.1.3.
(cited on page 54)

[194] Sminchisescu, C., Kanaujia, A., and Metaxas, D. (2006). Conditional models for contextual human motion recognition. *Computer Vision and Image Understanding*, 104(2):210 – 220. DOI: https://doi.org/10.1016/j.cviu.2006.07.014. Special Issue on Modeling People: Vision-based understanding of a person's shape, appearance, movement and behaviour.
(cited on page 156)

[195] Spector, W. D., Katz, S., Murphy, J. B., and Fulton, J. P. (1987). The hierarchical relationship between activities of daily living and instrumental activities of daily living. *Journal of Chronic Diseases*, 40(6):481–489. DOI: 10.1016/0021-9681(87)90004-X.
(cited on page 34)

[196] Stein, S. and McKenna, S. J. (2013). Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, UbiComp '13, pages 729–738, New York, NY, USA. ACM. DOI: 10.1145/2493432.2493482.
(cited on page 92)

[197] Sukthankar, G., Goldman, R. P., Geib, C., Pynadath, D. V., and Bui, H. H., editors (2014). *Plan, Activity, and Intent Recognition: Theory and Practice*. Newnes.
(cited on page 29)

[198] Teipel, S., Babiloni, C., Hoey, J., Kaye, J., Kirste, T., and Burmeister, O. K. (2016). Information and communication technology solutions for outdoor navigation in dementia. *Alzheimer's & Dementia*, 12(6):695–707. DOI: 10.1016/j.jalz.2015.11.003.
(cited on page 17)

[199] Thrun, S. (2002). Particle filters in robotics. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, UAI'02, pages 511–518, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
(cited on pages 75, 105)

[200] Törnqvist, D., Schön, T. B., Karlsson, R., and Gustafsson, F. (2009). Particle filter SLAM with high dimensional vehicle model. *Journal of Intelligent and Robotic Systems*, 55(4-5):249–266. DOI: 10.1007/s10846-008-9301-y.
(cited on pages 75, 105)

[201] de la Torre, F., Hodgins, J., Bargteil, A., Martin, X., Macey, J., Collado, A., and Beltran, P. (2009). Guide to the Carnegie Mellon University multimodal activity (CMU-MMAC) database. Technical report, Robotics Institute, Carnegie Mellon University.
(cited on pages 50, 80, 82)

[202] Turaga, P., Chellappa, R., Subrahmanian, V. S., and Udrea, O. (2008). Machine recognition of human activities: A survey. *IEEE Trans. Cir. and Sys. for Video Technol.*, 18(11):1473–1488. DOI: 10.1109/TCSVT.2008.2005594.
(cited on page 33)

[203] Vales-Alonso, J., Parrado-García, F., and Alcaraz, J. (2016). OSL: An optimization-based scheduler for RFID dense-reader environments. *Ad Hoc Networks*, 37(2):512–525. DOI: 10.1016/j.adhoc.2015.10.004.
(cited on page 56)

[204] van Gael, J., Saatci, Y., Teh, Y. W., and Ghahramani, Z. (2008). Beam sampling for the infinite hidden markov model. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1088–1095, New York, NY, USA. ACM. DOI: 10.1145/1390156.1390293.
(cited on page 72)

[205] Venema, Y. (2001). Temporal logic. In Goble, L., editor, *The Blackwell Guide to Philosophical Logic*, pages 203–223. Wiley-Blackwell.
(cited on page 89)

[206] Vergnès, D., Giroux, S., and Chamberland, D. (2005). Interactive assistant for activities of daily living. In *Proc. of the 3rd International Conference on Smart homes and health Telematic, ICOST*, volume 5, pages 229–236.
(cited on page 17)

[207] Vital, J. P. M., Faria, D. R., Dias, G., Couceiro, M. S., Coutinho, F., and Ferreira, N. M. F. (2017). Combining discriminative spatiotemporal features for daily life activity recognition using wearable motion sensing suit. *Pattern Analysis and Applications*, 20(4):1179–1194. DOI: 10.1007/s10044-016-0558-7.
(cited on page 156)

[208] Ward, A., Jones, A., and Hopper, A. (1997). A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47. DOI: 10.1109/98.626982.
(cited on pages 31, 80)

[209] Weiser, M. (1991). The computer for the 21st century. *Scientific american*, 265(3):94–104. DOI: 10.1038/scientificamerican0991-94.
(cited on pages 16f., 80)

[210] Willmott, C. J. and Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, 30(1):79–82.
(cited on page 107)

[211] Wolf, K. (2007). Generating petri net state spaces. In Kleijn, J. and Yakovlev, A., editors, *Petri Nets and Other Models of Concurrency – ICATPN 2007*, volume 4546 of *Lecture Notes in Computer Science*, pages 29–42. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-73094-1_5.
(cited on page 98)

[212] Wu, J., Osuntogun, A., Choudhury, T., Philipose, M., and Rehg, J. M. (2007). A scalable approach to activity recognition based on object use. In *IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE. DOI: 10.1109/ICCV.2007.4408865.
(cited on pages 42f., 47, 152)

[213] Wurdel, M., Sinnig, D., and Forbrig, P. (2008). CTML: Domain and task modeling for collaborative environments. *Journal of Universal Computer Science*, 14(19):3188–3201.
(cited on page 157)

[214] Yamato, J., Ohya, J., and Ishii, K. (1992). Recognizing human action in time-sequential images using hidden markov model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '92)*, pages 379–385. DOI: 10.1109/CVPR.1992.223161.
(cited on pages 18, 33)

[215] Ye, J., Stevenson, G., and Dobson, S. (2014). USMART: An unsupervised semantic mining activity recognition technique. *ACM Trans. Interact. Intell. Syst.*, 4(4):16:1–16:27. DOI: 10.1145/2662870.
(cited on page 50)

[216] Ye, J., Stevenson, G., and Dobson, S. (2015). KCAR: A knowledge-driven approach for concurrent activity recognition. *Pervasive and Mobile Computing*, 19:47–70. DOI: 10.1016/j.pmcj.2014.02.003.
(cited on pages 42f., 50)

[217] Yordanova, K. (2014). *Engineering Human Behaviour Models for Activity Recognition*. PhD thesis, University of Rostock.
(cited on pages 15, 22, 55, 58, 64, 77, 87, 92, 94, 138, 150)

[218] Yordanova, K. and Kirste, T. (2015). A process for systematic development of symbolic models for activity recognition. *ACM Transactions on Interactive Intelligent Systems*, 5(4):20:1–20:35. DOI: 10.1145/2806893.
(cited on pages 27, 58, 85, 87, 92, 94)

[219] Yordanova, K., Nyolt, M., and Kirste, T. (2014). Strategies for reducing the complexity of symbolic models for activity recognition. In Agre, G., Hitzler, P., Krisnadhi, A. A., and Kuznetsov, S. O., editors, *16th International Conference on Artificial Intelligence: Methodology, Systems, Applications*, volume 8722 of *Lecture Notes in Computer Science*, pages 295–300. Springer International Publishing. DOI: 10.1007/978-3-319-10554-3_31.
(cited on pages 49, 77, 86f.)

[220] Yu, Z. and Nakamura, Y. (2010). Smart meeting systems: A survey of state-of-the-art and open issues. *ACM Computing Surveys*, 42(2):8. DOI: 10.1145/1667062.1667065.
(cited on page 80)

Theses

Thesis 1. There is increasing demand for assisting people in their everyday activities. Correctly recognising the situation is a precursor to useful assistance.

Thesis 2. Inference of the situation of human behaviour in real-life requires many computing resources. Correctly estimating the true situation is very challenging.

Thesis 3. Engineering useful and correct models of human behaviour is challenging and error-prone. Human-built models of human behaviour contain errors that influence the recognition. Many errors can be found by checking for a few classes of errors.

Thesis 4. Everyday human behaviour entails a large number of context states $\mathbf{S}$. Handling the large set of context states $\mathbf{S}$ is one factor that limits efficiency of Sequential Monte Carlo (SMC) methods. Automatically reducing the set of relevant context states increases the efficiency of SMC.

Thesis 5. The traditional particle filter, as the most popular instance of SMC, is inefficient for inference in categorical state spaces. An efficient management of categorical samples increases the quality of the estimate.

Thesis 6. Human behaviour has a large variability in the duration of behaviour. The results in a large set of possible starting times $\mathbf{T}$. Handling the large set of starting times $\mathbf{T}$ is one factor that limits efficiency of SMC methods.

Thesis 7. For Sequential Monte Carlo inference of the situation using human behaviour models, the number of particles required to bound the variance to a certain level increases with the number and dimensions of situations. Reducing the belief state is important for efficient inference.