Second Annual Conference on Advances in Cognitive Systems Poster Collection (2013) 77-92

An Expectations Framework for Domestic Robot Assistants

Michael Karg KARGM@IN.TUM.DE
Institute for Advanced Study, Technische Universität München, Lichtenbergstrasse 2a, D-85748 Garching, Germany, http://hcai.in.tum.de

Alexandra Kirsch ALEXANDRA.KIRSCH@UNI-TUEBINGEN.DE
Department of Computer Science, University of Tübingen, Sand 14, D-72076 Tübingen, Germany, http://www.hci.uni-tuebingen.de

Abstract

Robots that are supposed to work in everyday environments are confronted with a wide variety of situations. Not all such situations can be taken into account by a programmer when implementing the system. This is why robots often show strange behavior when they encounter situations that their programmer has not expected. We propose a knowledge-based approach to explicitly represent *expectations* in the robot program. Comparing those expectations to the current situation allows the robot itself to detect unusual situations and react appropriately. Our general framework can incorporate expectations from different knowledge sources and offers a flexible combination of different expectations. We demonstrate the feasibility of the approach in the context of a household robot in simulation and in a real environment. Finally we discuss the adequacy of our proposed solution and open questions for further research.

1. Introduction

Intelligent robots are supposed to work in unstructured environments and to collaborate with humans. Such domains naturally imply a wide variety of different situations. Failures are usually specified and tested for explicitly. This means that if a situation was not anticipated at design time, the robot will continue its standard behavior and possibly will recognize some failure when its actions do not succeed, but does not know the reason for the failure. This inflexibility usually leads to robot behavior that looks completely absurd to humans and severely limits the perceived reliability of autonomous robots.

Since it is hard to define what a failure is, we propose a general notion of *expectations*, which can be met to different degrees. We represent expectations explicitly in the robot program, using different knowledge sources. Figure 1 illustrates some categories of expectations: In the left picture there are objects floating in the air. This clearly violates the laws of physics and would contradict a naive physics reasoner. The human lying on the floor obeys the laws of physics, but is unusual behavior. The image on the right shows a less surprising situation, even though a pile of boxes on a table might not be a very normal thing in a household while they would be expected in a warehouse. In general, unexpected situations can have different causes including failures in the robot program

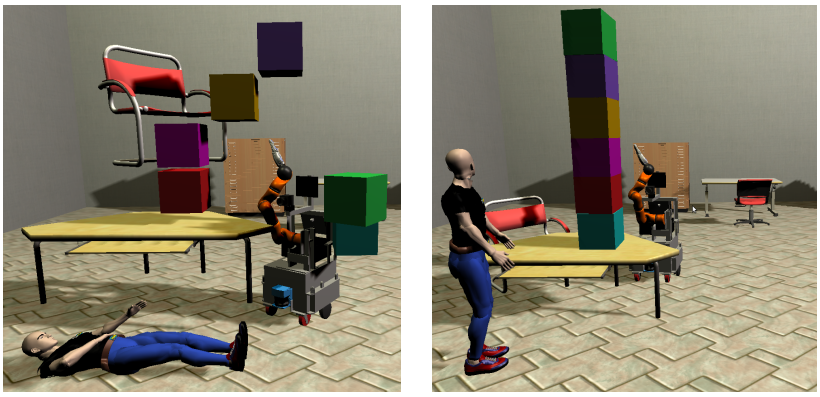

Figure 1. Left: An "abnormal" simulated scene in human robot interaction. Right: A "normal" simulated scene in human robot interaction.

(caused by inaccurate sensing or unreliable action execution); unexpected human behavior, such as dropping objects or forgetting things (the latter is especially relevant in the care of people with dementia); and events in the environment such as someone calling at the door or a storm coming up. We propose a framework that combines different kinds of knowledge to represent different classes of expectations that can differ according to the specific application. Promising candidates to provide such knowledge are naive physics reasoning (Akhtar & Kuestenmacher, 2011), simulation-based projection (Kunze et al., 2011), geometric reasoning (Mösenlechner & Beetz, 2013) or knowledge based inference (Kunze, Tenorth, & Beetz, 2010). Each of these methods can detect certain kinds of anomalies in the specific situation, but none of them covers all possible sources of surprise.

An unusual situation is not necessarily an error or a failure. Some events may not require any reaction of the robot at all (when the doorbell rings and an inhabitant of the household opens the door, the robot has no need for action). Others may require clarification actions by the robot such as asking the human lying on the floor what is wrong. In this paper, we do not deal with the reactions to unusual situations, but we offer a modular framework that helps to detect unusual situations and their degree of unusualness, and to give hints for possible solutions by explicitly representing which expectations are currently violated.

We first present related work of automatic detection of failures or surprise in general. Then we present our framework, in particular options for representing expectation models and how to combine different expectations. After that we demonstrate the feasibility of the approach for a household robot in a simulated world with up to 15 expectations and in a real-world situation using two expectations. The paper ends with a discussion of the benefits and limits of our approach, pointing out further research questions.

2. Related Work

General, formal approaches for failure detection in technical systems follow a similar aim as our proposed expectations framework and include research from model-based programming (Struss, 2008) and discrete control theory (Kelly et al., 2009). In both approaches, the system is described by a formal model like a state automaton with probabilistic transitions or Petri nets. By explicitly modeling failures, a system can avoid such states or diagnose the reasons for encountering a failure.

In the field of robotics, such approaches are common for the diagnosis of internal faults of single components of a robot. Gerald Steinbauer (Steinbauer & Wotawa, 2005) introduces *observers* to perform model based diagnosis on robot components without affecting the control system, while Kuhn et al. (Kuhn et al., 2008) propose the paradigm of pervasive diagnosis to simultaneously enable active diagnosis and model based control. Wiliams et al. (Williams et al., 2003) use model based autonomy to enable autonomous systems to be aware of their states and possible errors in uncertain environments. They use models of the nominal behavior of a system as well as models of common failure modes to perform extensive reasoning and recognize and recover from failures. While the approaches mentioned so far consider only internal faults of technical systems, Akhtar and Kuestenmacher (Akhtar & Kuestenmacher, 2011) use qualitative reasoning on naive physics concepts for diagnosis for the prediction of external faults of autonomous systems.

However, these approaches are not suitable to failures in the high-level behavior of autonomous service robots. First, defining the interaction of a human and a robot in a finite state machine would mean a huge modeling effort. Whereas in discrete control theory the formal model can often be extracted directly from the system specification (which is usually given in a work-flow programming language), such an automaton would have to be hand-coded in the case of a robotic system interacting with humans. And this would mean that a designer would have to take all possible failure situations into account when modeling the behavior. Even more important, it is often impossible for a robot to decide whether it is in a failure state. A typical failure state would be "human has abandoned task". But the fact that a person has left the room can mean anything from fetching a necessary object to abandoning the joint task. And modeling on a direct observation-level would lead to extremely complex models, which would have to take into account the situational context.

Therefore, we propose to detect failures without an explicit model of failure states. Similar to the models of nominal behavior of technical systems of Wiliams et al. (Williams et al., 2003), we define a potential failure as all observations that are not compliant with the robot's experience and knowledge about how the world and in particular humans should behave. Combining different evidences and considering the degree of divergence from the robot's expectations prevents the robot from being overcautious. Work on such an explicit use of expectations has been done by Minnen et al. (Minnen, Essa, & Starner, 2003). They use extended stochastic grammars to introduce constraints into activity recognition to recognize a human playing "Towers of Hanoi" from video data by analyzing object interaction events. They find that humans have strong prior expectations about actions in activities and technical systems performing activity recognition can benefit from explicit models of expectations about high-level activities. Another example for the use of expectations in autonomous robots can be found in (Maier & Steinbach, 2010). Here Maier and Steinbach generate expectations to enable a mobile robot to detect unexpected scenes from video data. They use a dense map of images of the robot's environment and comparisons of luminance and chrominance values

of the images at different times. This enables them to detect changes in the robot's environment and make assumptions about the expectations and uncertainties in the environment.

These approaches work well for diagnosing faults in specific domains. But to our knowledge there is no approach that incorporates a combination of different models, including of humans, to enable robots performing diagnosis on cooperative everyday tasks.

3. Expectations Framework

We define an expectation as any piece of knowledge that describes normal mechanisms of the world, such as physical laws, robot behavior or habitual human behavior. The *normality* of a situation is assessed by comparing each expectation with the current situation and combining the single values of experience matching into one overall value.

Our framework offers a common interface for all expectations as illustrated by the class structure in Figure 2. From the implementation point of view this is done by a generic class "expectation" which is sub-classed by specific expectation classes to implement a concrete type of expectation. All expectations implement a validation method that returns a normality value between 0 and 1. A normality value of 1 means that the expectation is met perfectly in the current situation and a value of 0 means that the expectation is currently not at all fulfilled. Validation of expectations is triggered by a validation hook and can be different depending on the type of expectation. Imagine for example expectations about physical models in contrast to expectations about future locations of a person. While it makes sense to validate physical constraints continually, we can only validate the expectation that the human will go to the refrigerator next as soon as we know if he/she actually went to the refrigerator.

All expectations are stored in an *expectations pool* that can be modified dynamically. Depending on the context and the progression of the situation, expectations can be added, adapted and removed. For example, when a user enters the room, the robot should add expectations about human behavior, which can be removed when the person has left again.

Similar expectations can be grouped in categories, which are treated as composite objects with the same interface as single expectations just like the *Composite* pattern one might know from object oriented software design patterns (see e.g. (Erich et al., 1995)). The normality value of a category is computed recursively from the normality values of all the experiences in that category. One such category could be "Human Activity Expectations" to group several expectations about human activities and get an impression about how well the human actions observed so far generally fulfill the expectations of the robot. The introduction of expectation categories as well as the different types of expectations themselves strongly depend on the scenario of application.

3.1 Expectation Models

We already mentioned that the validation method of an expectation returns a value between 0 and 1 to express the degree to which expectation is fulfilled. For our household robot application we differentiate between logical, temporal and probabilistic expectation representations but one could as well think of generating a variety of other expectations like spatial expectations, exemplar-based expectations or similar.

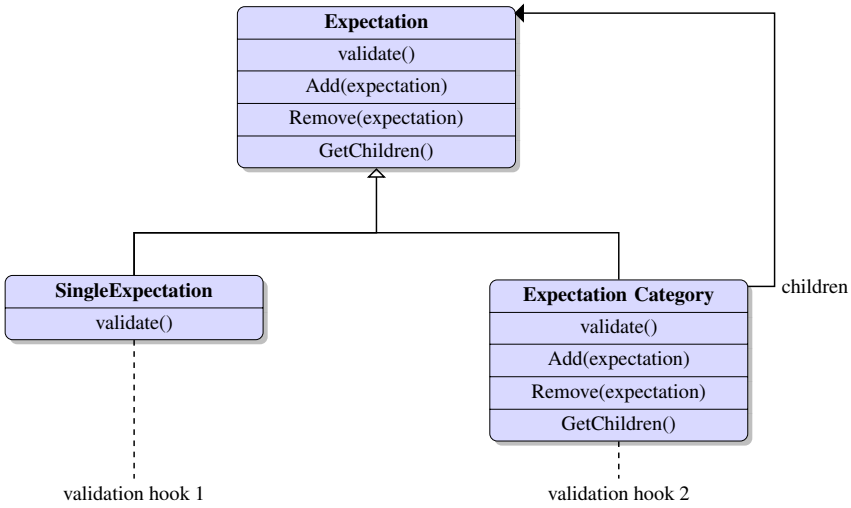

Figure 2. The implementation structure of expectations. The abstract class Expectation defines the interface that the concrete expectations implement. Every expectation consists of a validation method that returns a value between 0 and 1 that indicates to which degree the expectation has been fulfilled as soon as it is validated. Expectations can either be single expectations or compositions of several expectations grouped into categories. In addition to single expectations, an expectation category implements functions for adding and removing expectations and the validation is done by validating all expectations in the category that are ready to be validated.

3.1.1 Logical Expectations

We see logical expectations as expectations that are composed by propositions that either hold in the current situation or are violated. A validation of such an expectation will simply return 1 if the proposition holds and 0 otherwise. An example of such an expectation is: "Cup_1 on the table", which expresses that the robot would expect the object called "Cup_1" being located on the table. This can either be true or false. So generally, we can see logical expectations as binary logical propositions that return their truth-value when they are validated.

3.1.2 Temporal Expectations

We define temporal expectations as expectations that hold for a certain duration. They are an extension of logical expectations and include logical propositions as well as a duration during which the expectation is expected to hold. Validation of such an expectation returns 1 as long as the time during which the expectation holds so far is smaller than the expected duration. The normality decreases when the expected duration is exceeded. We define the decrease of the normality linearly by default, but one might as well think of other functions depending on the intended application. An example of such an expectation is the duration that a person is standing in front of a cupboard when picking up objects from it. In this case, he/she is expected to stay approximately 4 seconds in front of that cupboard. The longer a person is standing in front of such a cupboard, the more unlikely it is that he/she is picking up objects from it.

3.1.3 Probabilistic Expectations

In some cases, there might be different possibilities for events to happen and it cannot be clearly stated, which one is most likely to happen. Probabilistic expectations have probability distributions over random variables instead of binary propositions assigned to them. Such a distribution describes the probability that an event or a set of different possible events will occur. A robot might for example observe a person being at the table while its activity recognition tells it that the person is either preparing cereals or cleaning the table with associated probabilities. Thus the robot will expect the human to move to one of several locations next, each associated with a probability. Validation of a probabilistic expectation will return a value between 0 and 1 corresponding to the probability that was assigned to the expectation that was observed to be true. The implications of such a validation are discussed in section 5.

3.2 Expectation Validation

We assume that the overall normality of a situation results from the combination of all normalities observed for the single expectations. From single expectations and expectation classes we construct a *normality tree* of expectations values. Thus, the normality value of an expectation class is computed from the normality values of the single expectations contained in it and the overall normality is computed from the class values. The framework leaves the method for combining these values open. For our trials we have so far simply averaged the values from the child nodes. But other methods such as weighted sums, thresholds or maximum are also conceivable. Also the depth of the hierarchy is not restricted, since classes of expectation classes can be defined. In our scenarios we always used three layers as shown in the exemplary normality tree in Figure 3. Here, a situation is observed that mostly fulfills the robot's expectations and thus leads to the high overall normality of 0.95. In the example, the robot's expectations about the objects of interest are fulfilled since the table did not move and it has detected the cup on the table as expected, resulting in a normality value of 1 for the expectations about the objects. Also the validation of expectations about the human, consisting of two probabilistic expectations, return the values of 0.92 and 0.85, resulting in an average normality of 0.89 for the human normalities. The normality tree is updated as soon as one of the validated expectations changes its value so the robot will obtain an estimation of the normality of the current situation after every new event for which an expectation exists.

The normality tree allows a limited, but general way of finding the symptoms of surprising events. So when the overall normality value drops below a certain threshold, the robot should consider taking some action. By traversing the normality tree it can diagnose which of its expectations has been violated and act accordingly, for example by further clarifying the situation or taking immediate action.

4. Application: Expectations for a Household Robot

To evaluate the applicability of our expectations framework we set up our framework in a simulated apartment scenario using different types of expectations. Furthermore, we use a real world dataset of a human morning routine to equip a domestic robot helper with expectations about human routine

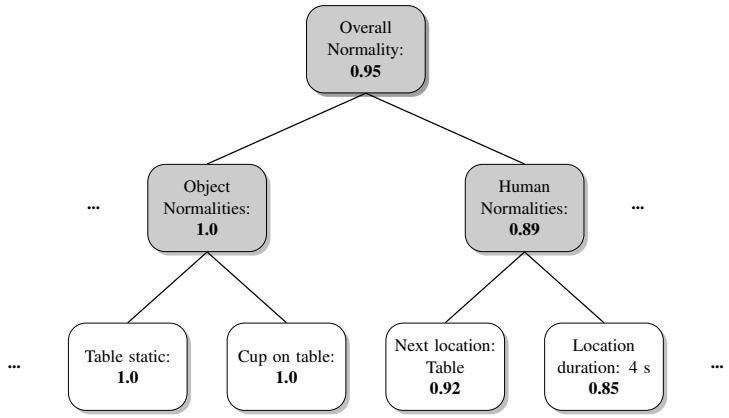

Figure 3. A normality tree generated by the validation of different types of expectations. The highest level of the tree describes the overall normality of the situation whereas the lower levels allow for a more fine-grained description of the normalities of specific expectations. Grey nodes represent normalities generated from composite expectations, white nodes correspond to normalities from single expectations.

behavior and showcase our framework in a real kitchen equipped with a Kinect sensor for human motion tracking.

4.1 Expectations in a Simulated Human Apartment

In this scenario, we demonstrate the applicability of the expectations framework in different situations using 5 to 15 expectations grouped in different categories in a simulated apartment environment including a person and a PR2 robot as shown in Figure 4. We use the realistic physical simulator MORSE [1] that includes a human avatar that can be controlled like in modern 3D computer games (Lemaignan et al., 2012) and enables a user to perform pick- and place actions, open doors, cupboards and drawers and operate switches in the simulated scenario. The robot is equipped with a simulated object detection sensor that returns the name of objects that are within the field of view of the robot as well as their positions. Furthermore, the robot is able to detect if doors are open or closed and the positions of humans, while it is unable to distinguish between different persons. The robot moves using the ROS-based 2D navigation "move_base" [2]. A video of both simulated scenarios is available online [3].

4.1.1 Kitchen robot

For this scenario, we assume that our robot is located in the kitchen where the table has been set for breakfast as illustrated in Figure 5 on the left. It expects furniture and objects not to move around while no human is close to them and it expects typical kitchen objects not to lie on the ground. Specifically, for each object, we generate one expectation to express that the object is not

1. http://www.openrobots.org/wiki/morse
2. http://wiki.ros.org/move_base
3. http://vimeo.com/hcai/expectations

Figure 4. The simulated apartment scenario with a human avatar and a PR2 robot using the MORSE simulator.

Figure 5. The kitchen robot scenario in the simulated apartment. Left: A simulated PR2 robot starts navigating away from the table. Middle: The robot inadvertently hits the table causing objects to fall to the ground. Right: The robot looks back, detecting more objects lying on the ground.

expected to move and one expectation that the object is not expected to lie on the ground. We include the following objects into this scenario: fork, knife, jam, hazelnut spread, cereals, bowl and plate. Furthermore we generate one expectation that the kitchen table is not expected to move. So we have a total of 15 expectations that we continually validate every two seconds. When the robot starts moving towards the door of the kitchen, it inadvertently hits the table with its side causing some objects on the table to fall onto the floor as shown in Figure 5 in the middle. Having detected that some objects do not fulfill their expectations, the robot turns around at a later point in time and detects more objects lying on the ground as shown in Figure 5 on the right.

The average normality plotted over time is shown in Figure 6. The normality drops as soon as the robot hits the table since it detects objects within its field of view moving unexpectedly, falling onto the floor. However, not all objects that fell down, are detected since the robot is moving away from the table and not all objects are in the field of view of its object detection sensor. When all objects stop moving, the normality is around 0.7. When the robot later turns around and detects that more objects have moved since their last detection and lie on the ground, the normality drops even more. At the end, the normality slightly increases since the objects that have moved since their last detection (when they were still on the table) are now stationary again and only the expectations that those objects are expected to be on the table are not fulfilled. The normalities in this scenario are a valuable indicator to the robot that an action of the robot caused a change in the normality of the

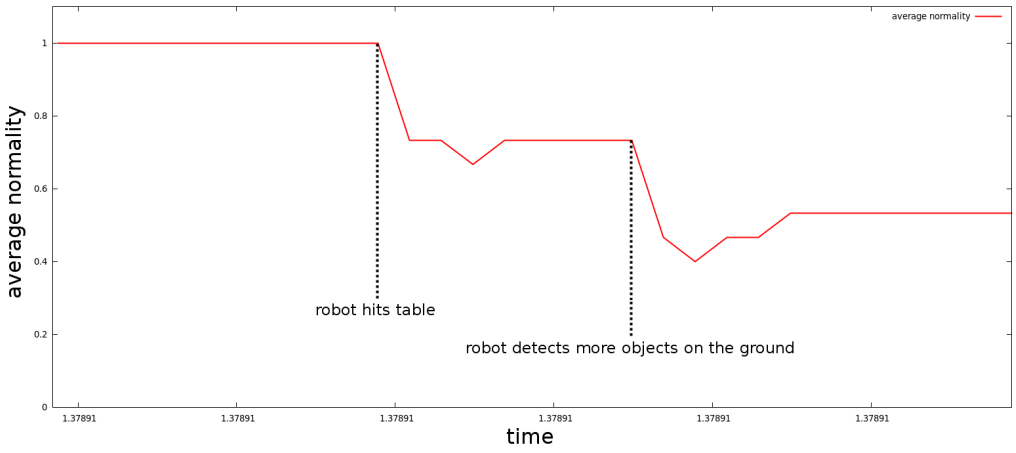

Figure 6. Average normality of the kitchen robot scenario plotted over time.

situation and could serve as input for an Explanation Generator in Goal Directed Autonomy (GDA) which could use such discrepancies to create a new goal and ultimately react appropriately as e.g. in the work of Klenk et al. (Klenk, Molineaux, & Aha, 2012).

The same set of expectations would enable the robot to detect damages caused by an earthquake or a child throwing objects on the floor. Even though the causes are different and the robot may not even have the appropriate sensors to detect the specific cause, it will at least notice that the situation is unusual and something has to be done.

4.1.2 Patrol Robot

In this scenario, our household robot — having nothing else to do — is guarding the apartment while the human is sleeping. It constantly patrols several locations to check if everything is as expected and it stays at each location for two seconds. We use five expectations in this scenario: The TV is expected to be in the living room and it is expected not to move. Humans are not expected to be anywhere else but in the bedroom or outside of the house. The entrance door of the apartment is locked and the robot navigation works normally. For the robot navigation, we generate a temporal expectation each time the robot starts moving to another waypoint and queries the navigation path planner for a new path plan. It then estimates the time to the location using the length of the path plan and its average speed and generates a temporal expectation. The expectation gets removed when a goal point is reached, meaning that during the two seconds where the robot is standing still at each location, only the four expectations mentioned before are active.

In this scenario, we use the human avatar to simulate a burglar that enters the apartment during the robot's patrol. He opens the entrance door, passes the hallway to get into the living room, picks up the TV and passes the hallway again, carrying the TV out of the apartment. The simulated burglary is illustrated in Figure 7. The average normality of the robot during this burglary plotted over time is shown in Figure 8. The normality drops as soon as the robot detects the open entrance

Figure 7. The patrol robot scenario. Left: The robot detects the open door and the human in the hallway. Middle: The robot detects the TV moving and not beeing in the living room. Right: The burglar has left the apartment with the TV.

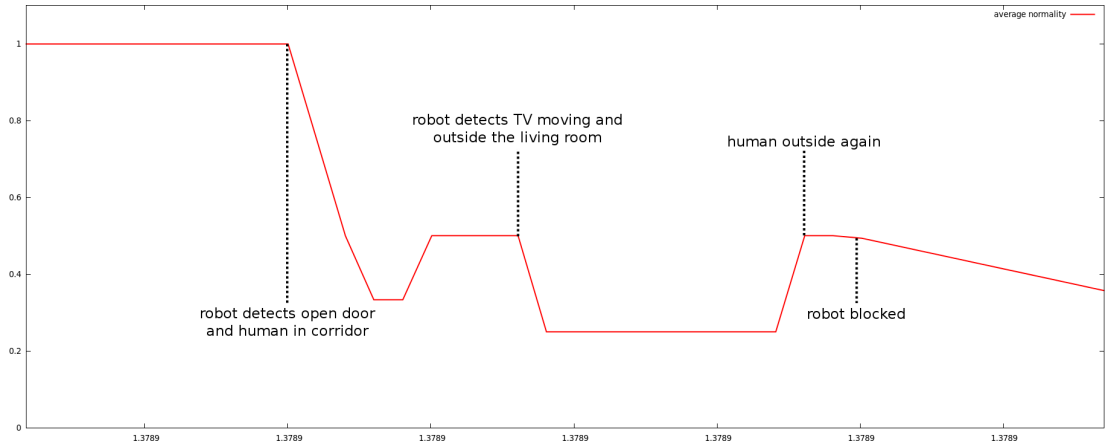

Figure 8. Average normality of the patrol robot scenario plotted over time.

door and the human in the hallway. It drops even further when it detects that the TV is not in the living room any more and has moved since the last detection. However, the normality never drops to 0 since the navigation of the robot is still working correctly. When leaving the apartment with the TV, the burglar had to pass the robot really closely, getting into the sensor range of the simulated laser scanner thus the robot's navigation is blocked due to security reasons. This caused the navigation to take longer than expected and the temporal expectation starts decreasing linearly towards the end.

4.2 Expectations About a Human Morning Routine

To demonstrate the applicability of our approach in a real-world scenario, we use a dataset that contains motion tracking data of a typical human morning routine. The dataset features a male participant who had no knowledge about our system, acting out his morning routine from 14 weekdays, based on notes in a specifically kept diary. The activities performed during the morning routines of all days consist of preparing a drink, drinking a glass of water, preparing cereals, eating cereals, preparing curd-cheese, eating curd-cheese, preparing bread, eating bread, cleaning the table and

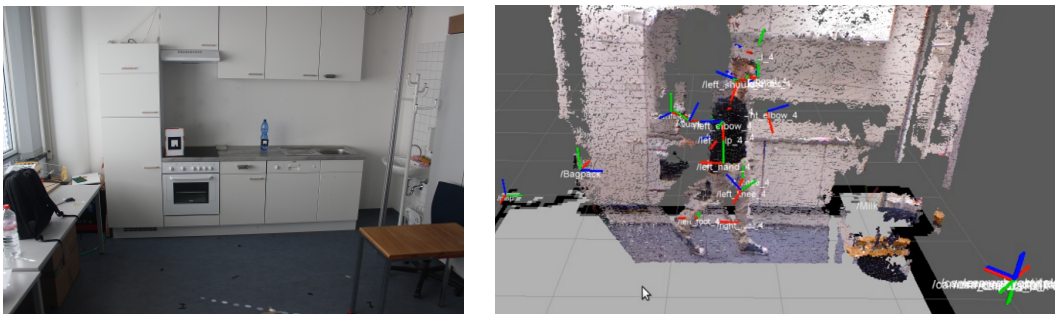

Figure 9. The left image shows the sensor equipped kitchen environment while in the right image illustrates the sensor input of the real scenario. The motion tracker returns coordinate transformations for each joint of the human while the visual marker detection returns the approximate position of the markers that we attached to the objects.

preparing for work. The available sensor data includes motion tracking data recorded with a Kinect using the Openni tracker[4] and partial object detections based on visual markers using the ROS AR Kinect toolbox[5] (Karg & Kirsch, 2013). An overview of the sensor equipped kitchen is shown in Figure 9.

In this work, we use the motion tracking data of the dataset and a semantically annotated map of the environment in a module that performs activity recognition. This module enables us to detect typical human activities and predict future locations of the observed person. The semantically annotated environment map is combined with learned models of locations where humans typically are located when picking up objects from cupboards, drawers etc. (Karg & Kirsch, 2012). The models of the activities used for recognition were hand-coded in this scenario and include the locations a person visits during task performance as well as the durations that he/she commonly spends at those locations. However, such models of activities of daily living can also be generated automatically by observing motion tracking data as we showed in previous work (Karg & Kirsch, 2012). The activity recognition module uses the activity models in the form of Hierarchical Hidden Markov Models (HHMMs) (Fine, Singer, & Tishby, 1998) and provides our robot with a probability distribution about which activity the human is likely to be executing as well as probabilities about which locations the human is likely to go to next. The activity recognition module lies beyond the focus of this paper; for a detailed description we refer to our previous work (Karg & Kirsch, 2013).

Given the limited data available from Kinect sensors, we used the following two expectations about the human activities, grouped in a composite expectation we call *human activity expectations*:

- *Location expectation:* a probabilistic expectation about the locations that the person is likely to visit next;
- *Duration expectation:* a temporal expectation about the duration that a human usually stays at certain locations.

4. http://ros.org/wiki/openni$_t racker$
5. http://www.ros.org/wiki/ar$_k inect$

Table 1. Average normalities while the participant performed his morning routine using expectations about the next expected location and the duration the person stays maximally at each location. Averaging over all days, the overall normality of the observed situations is **0.80**.

Day1	Day2	Day3	Day4	Day 5	Day 6	Day 7
0.76	**0.75**	**0.84**	**0.82**	**0.76**	**0.81**	**0.80**
Day8	Day9	Day10	Day11	Day12	Day13	Day14
0.75	**0.84**	**0.80**	**0.81**	**0.83**	**0.82**	**0.83**

Table 2. Average normalities while the participants performed a cleaning task which is not expected during the morning routine. Averaging over all days, the overall normality of the observed situations is **0.51**.

Person 1	Person 2	Person 3	Person 4	Person 5	Person 6	Person 7	Person 8	Person 9
0.40	**0.45**	**0.53**	**0.49**	**0.48**	**0.57**	**0.71**	**0.51**	**0.46**

Each time our system detects that the human is standing still for a short time, we query our annotated environment model to which location the current position corresponds, add an observation to the HHMM of the activity recognition module and create or update the location expectation and the duration expectation. We consider the human "standing still" at locations where the center of mass of the human is not moving more than 25 cm within 0.5 seconds (0.5 m/s). The activity recognition then returns a new probability distribution that estimates probabilities for each location being visited next by the human. This probability distribution is used to create a probabilistic expectation over likely next locations of the human. Since our activity models also include the durations that a human typically spends at certain locations, we can also generate a temporal expectation about these durations each time the human is standing still. We define the validation function to linearly decrease its normality value as soon as the expected time at a location is exceeded. The two expectations generated so far are dynamically updated each time the human is standing at a new location.

Figure 10 shows the estimated overall normality over time for three different situations: The green line represents a typical morning routine of the participant, which mostly corresponds to the robot's expectations. Thus, the average normality is quite high (0.83 on average). In some cases, the normality drops to almost 0.6 which is due to wrong location detections and uncertainties in the activity recognition. But mostly our expectations framework classifies the situation as quite normal. The blue dotted line shows an example where the participant starts performing his typical morning routine but then just stays sitting at the table instead of continuing his task. In this case, the situation is classified as quite normal as long as the duration expectation does not exceed its expected duration and the normality decreases as time advances. A possible reason for the human to not move any more could be that he/she is sick or even unconscious or maybe he/she just needs more time than usual due to other reasons. The red line shows the normality values when the robot observed a cleaning task that is not part of the morning routine. The data for the cleaning task was taken from a data set of 9 participants — staff members of the Computer Vision group who had no knowledge of our system — who were asked to clean the same experimental kitchen in which the morning routine was recorded.

The normality values averaged over time for all 14 days in the morning routine data set are shown in Table 1 and the overall normalities of the cleaning task data (when expecting the morning

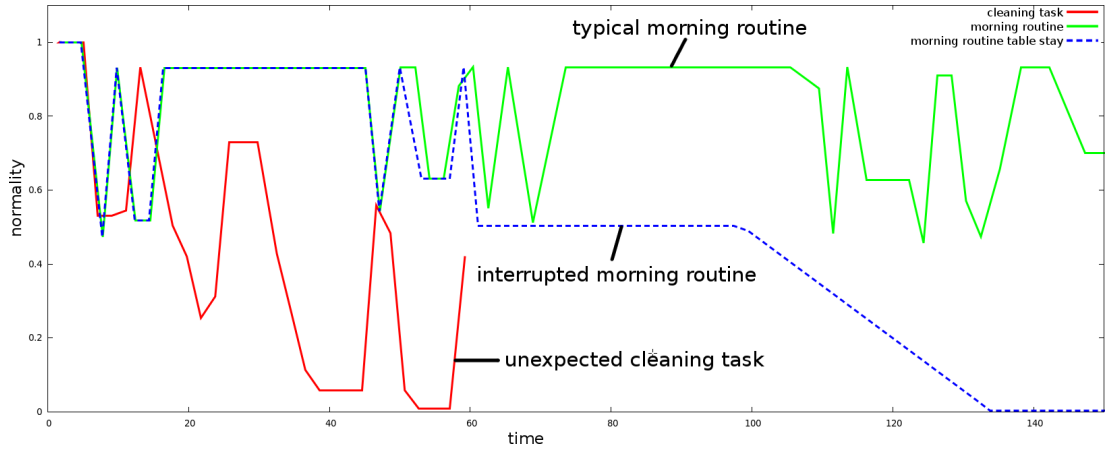

Figure 10. Estimated average normality of three different situations over time. Green line: the normality value during a typical morning routine; dotted blue line: normality for an interrupted morning routine; red line: normalities for a non-morning-routine task.

routine) averaged over time is illustrated in Table 2. The tables show that in most cases just the two simple expectations we used can clearly distinguish between expected and unexpected behavior.

5. Discussion

The proposed framework of expectations along with their validation is designed to offer a general, modular and knowledge-based way to monitor and react to unusual situations. We make no assumptions about the knowledge used for representing expectations except that it can be used to quantify the normality of a situation. With the modular definition of expectations the framework can be used and extended for arbitrary applications. The combination of normality estimates in the normality tree provides some guidance towards suitable reactions. By exploiting knowledge that the robot may also need for its decision making or state estimation, the recognition of unusual situations can be done by the robot itself, thus freeing engineers from the burden of considering every single situation the robot might encounter.

One open question is how to get the necessary expectations. The number and types of expectations a robot needs to identify abnormal situations largely depends on the type of tasks it is designed to perform. While for example a vacuuming robot can benefit from expectations about the human's future locations in order to avoid vacuuming in close proximity to the human, expectations about the location of a cup will be of no interest for such a robot. In contrast, a more elaborate household robot that is designed for household activities like setting the table, expectations about the location of a cup can prove useful. In our current application scenarios, we partly manually define expectations that seem useful in specific situations. We also use learned models about the expected behavior and future locations of humans performing different activities and give an example about how we can use learned models about human activities in combination with activity recognition to generate and validate expectations about human task performance. Another way of generating

expectations is by inference using common sense knowledge. Tenorth et al. (Tenorth et al., 2010), for example, use Knowledge-Linked Semantic Object Maps in the knowledge processing system KnowRob to infer likely storage locations of objects in a kitchen which can be modeled as expectations. When the robot uses an AI planner, also the effects of the planning operators can directly be used as expectations about the outcome of the action.

Another difficulty lies in the weighting of different pieces of information. Even though we have used the average of different expectation values, probabilistic expectations can implicitly have a lower weight if the possible outcomes are almost equally distributed. For example, let us consider a robot observing a human activity and trying to predict future locations of the person. Given its observations $o_{1:n}$ so far, the robot generated the following discrete probability distribution $P(l_i^{t+1}|o_{1:n})$ about the next location l_i^{t+1} the human will visit:

$$P(l_i^{t+1}|o_{1:n}) = \{\text{refrigerator: } 0.75,\ \text{oven: } 0.15,\ \text{drawer: } 0.05,\ \text{table: } 0.05\}$$

The robot expects the human to most likely go to the refrigerator next (with probability 0.75) and if the person will actually go to the refrigerator, the validation will return 0.75 as normality value for the next location expectation, which means that the probabilistic expectation about future locations has been fulfilled with 75 %. In another case where the robot might not be quite certain about the future location of the human e.g. due to uncertainty in its activity recognition or noisy sensor readings, the expectation might be based on the following probability distribution:

$$P(l_i^{t+1}|o_{1:n}) = \{\text{refrigerator: } 0.4,\ \text{oven: } 0.4,\ \text{drawer: } 0.15,\ \text{table: } 0.05\}$$

Here the robot is quite certain that the person will got to either the refrigerator or the oven next, with equal probabilities assigned to both locations. Validation will return a normality of 0.4 if the person will go to one of the two most likely locations. Basically the robot's expectation has been fulfilled, but due to the uncertainty in the predictions, the normality still has a rather low value of 0.4. Generally this means we make the normality of a situation also dependent on the uncertainty of our predictions when we use probabilistic expectations, which can be problematic. A possible solution to this problem could be inclusion of uncertainty into a weighting of the expectations.

The choice of the combination function for normality values in general is an open question and it is not clear whether our proposed representation in the normality tree is enough to cover all necessary interactions of normality values. We chose the average as the combination function, which can be justified by the tallying heuristic of humans, where weights are ignored for decision making (Gigerenzer & Gaissmaier, 2011). However, in some cases weights may be necessary and some kind of non-linear combination may be appropriate. For example when expectations are violated that imply a state of danger, just averaging may mean that the overall normality value is not affected strongly enough. So when the robot detects smoke it should not wait until its overall averaged normality factor drops below a threshold, but act immediately. One way of achieving such a shortcut is by not only monitoring the overall normality value, but also single critical values.

Finally, the integration of our expectation framework into a complete cognitive system is a crucial question that we have only mentioned briefly so far. On the one hand, the evaluation of single expectations must be integrated with the robot perception. In our case we used position data from Kinect sensors to formulate and verify expectations. But other expectation types need respective

perceptual input, for example for monitoring naive physics rules. The interaction with perception could also be useful in the other direction. When an expectation is not met, the reason can simply be a wrong interpretation of the world. So the expectation framework could build a bridge between perception and reasoning by requiring the perception system to re-evaluate the situation. On the other hand, appropriate action has to be taken when the situation strays too far from normality. Ideally, the knowledge source that indicated the deviation from normality could also provide information about the severity of the situation and possible actions. Also action selection and planning methods can be invoked to generate a new course of action for the changed situation. The action to take can also interact with the perception: in many cases the best reaction is to clarify the situation further by moving the robot to a position where it can get more detailed sensor information or by communicating with the user.

In all, our framework is a first step towards a general treatment of failures and unusual situations by autonomous robots. It can be easily integrated with traditional methods of failure recognition and handling, which is still the most robust way to handle frequent failures or dangerous situations. But our knowledge-based approach can cover the large variety of situations that an engineer would not think of or for which specific coding would be too costly.

6. Conclusion

This paper presented a promising first step towards a general framework that enables robot assistants to measure the normality of situations by validating a combination of different expectations. We showed how different types of expectations can be created, grouped and validated, giving a robot an impression of the overall normality of the situation as well as the normality of different categories or single expectations. In a simulated apartment, we showed how a robot can use a variety of expectations to detect situational anomalies and we illustrated how it can get a continuous impression of the normality of human behavior by observing motion tracking data in a sensor equipped kitchen. Future work includes other combinations of normality values as well as improved validation techniques for probabilistic expectations.

Acknowledgements

With the support of the Technische Universität München - Institute for Advanced Study, funded by the German Excellence Initiative, and the Bavarian Academy of Sciences and Humanities.

References

Akhtar, N., & Kuestenmacher, A. (2011). Using naive physics for unknown external faults in robotics. *22nd International Workshop on Principles of Diagnosis (DX-2011)* (p. 23).

Erich, G., Richard, H., Ralph, J., & John, V. (1995). Design patterns: elements of reusable object-oriented software. *Reading: Addison Wesley Publishing Company*.

Fine, S., Singer, Y., & Tishby, N. (1998). The hierarchical hidden markov model: Analysis and applications. *Machine learning*, *32*, 41–62.

Gigerenzer, G., & Gaissmaier, W. (2011). Heuristic decision making. *Annual Review of Psychology*, *62*, 451–482.

Karg, M., & Kirsch, A. (2012). Acquisition and Use of Transferable, Spatio-Temporal Plan Representations for Human-Robot Interaction. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Karg, M., & Kirsch, A. (2013). Simultaneous Plan Recognition and Monitoring (SPRAM) for Robot Assistants. *Proceedings of Human Robot Collaboration Workshop at Robotics Science and Systems Conference (RSS) 2013*.

Kelly, T., Wang, Y., Lafortune, S., & Welsh, M. (2009). A formal foundation for failure avoidance and diagnosis.

Klenk, M., Molineaux, M., & Aha, D. W. (2012). Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence*.

Kuhn, L., Price, B., de Kleer, J., Do, M., & Zhou, R. (2008). Pervasive diagnosis: Integration of active diagnosis into production plans. *proceedings of AAAI*.

Kunze, L., Dolha, M. E., Guzman, E., & Beetz, M. (2011). Simulation-based temporal projection of everyday robot object manipulation. *Proc. of the 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*. Taipei, Taiwan: IFAAMAS.

Kunze, L., Tenorth, M., & Beetz, M. (2010). Putting People's Common Sense into Knowledge Bases of Household Robots. *33rd Annual German Conference on Artificial Intelligence (KI 2010)* (pp. 151–159). Karlsruhe, Germany: Springer.

Lemaignan, S., G., E., Karg, M., Mainprice, M., Kirsch, A., & Alami, R. (2012). Human-robot interaction in the morse simulator. *Proceedings of the 2012 Human-Robot Interaction Conference (late breaking report)*.

Maier, W., & Steinbach, E. (2010). A probabilistic appearance representation and its application to surprise detection in cognitive robots. *IEEE Transactions on Autonomous Mental Development*, *Vol. 2, No. 4, pp. 267 - 281*.

Minnen, D., Essa, I., & Starner, T. (2003). Expectation grammars: Leveraging high-level expectations for activity recognition. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on* (pp. II–626).

Mösenlechner, L., & Beetz, M. (2013). Fast temporal projection using accurate physics-based geometric reasoning. *IEEE International Conference on Robotics and Automation (ICRA)*. Karlsruhe, Germany.

Steinbauer, G., & Wotawa, F. (2005). Detecting and locating faults in the control software of autonomous mobile robots. *19th International Joint Conference on Artificial Intelligence (IJCAI-05)* (pp. 1742–1743).

Struss, P. (2008). Model-based problem solving. 395–465.

Tenorth, M., Kunze, L., Jain, D., & Beetz, M. (2010). KNOWROB-MAP – Knowledge-Linked Semantic Object Maps. *Proceedings of 2010 IEEE-RAS International Conference on Humanoid Robots*. Nashville, TN, USA.

Williams, B., Ingham, M., Chung, S., Elliott, P., Hofbaur, M., & Sullivan, G. (2003). Model-based programming of fault-aware systems. *AI Magazine*, *24*, 61.