Design and Evaluation of Description Logics based Recognition and Understanding of Situations and Activities for Safe Human-Robot Cooperation

Stephan Puls, Jürgen Graf and Heinz Wörn
Institute of Process Control and Robotics (IPR)
Karlsruhe Institute of Technology (KIT)
Karlsruhe, Germany
{stephan.puls, juergen.graf, woern}@kit.edu

***Abstract*—Recognition of human activities and situation awareness is a premise for advanced safe human-robot-cooperation. In this paper, a recognition module and its advancements based on previous work is presented and discussed. The usage of Description Logics allows for knowledge based representation of activities and situations. Furthermore, reasoning about context dependent actions enables conclusions about expectations for robot behavior. This work is extensively tested and benchmarked. The presented approach represents a significant step towards a full-fledged cognitive industrial robotic framework.**

Keywords – cognitive robotics, Description Logics, situation and action recognition, evaluation, human-robot cooperation.

I. INTRODUCTION

Industrial robotics is a challenging domain for cognitive systems, especially, when human intelligence meets solid machinery with certain degrees of freedom like most of today's industrial robots.

Hence, guaranteeing safety for human workers, safety fences are installed to separate humans and robots. As consequence no time and space sharing interaction or cooperation can be found in industrial robotics.

Some progress has gained in the past so that some modern working cells are equipped with laser scanners performing foreground detection. But with these systems one is not able to know what is going on in the scene and, therefore, could not contribute something meaningful for challenging tasks like safe human-robot cooperation.

We are conducting research on recognition of and reasoning about actions and situations in a human centered production environment, in order to enable interactive and cooperative scenarios.

In [1], we presented a first approach for using Description Logics (DLs) [9] as means for representation of knowledge and as reasoning facilities for inference about activities and situations. Furthermore, conclusions about user expectations about robotic behavior can be drawn. This paper focuses on presenting applied techniques and the advancements on previous work [1]. Also, there are further investigations taking into account effectiveness and runtime behavior of the presented recognition module.

In Section II, selected research work on reasoning about scenes and situations will be presented. In Section III, a framework is introduced, which enables the sensor data processing and subsequent knowledge based reasoning. In Section IV, DLs are briefly introduced and the module realizing the communication with a Description Logics reasoner, knowledge base management and reasoner result management is presented in detail. Also the modeled situations and activities are explained. Section V discusses experimental results which have been carried out for both, predetermined test cases and under real-life conditions. In Section VI, a summary is given. Finally, some hints for future work are mentioned.

II. RELATED WORKS

There are a lot of approaches for action recognition systems based on probabilistic methods, e.g., hidden Markov Models (HMMs) [17, 18, 19], as their theoretic foundation is well understood and applications in speech recognition and other domains have shown their capabilities.

Based on arguments, that HMMs are not suitable for recognition of parallel activities, propagation networks [20] have been introduced. The propagation network approach associates each node of the network with an action primitive, which incorporates a probabilistic duration model. Also conditional joint probabilities are used to enforce temporal and logical constraints. In analogy to HMMs, many propagation networks are evaluated, in order to approximate the observation probability.

In [21], Minnen et al. put forward arguments that recognition of prolonged activities is not feasible based on purely probabilistic methods. Thus, an approach is presented which uses parameterized stochastic grammars.

The application of knowledge based methods for action recognition tasks is scarce, but work on scene interpretation using DLs has been conducted.

In [10], Hummel et al. use DLs for reasoning about traffic situations and understanding of intersections. Deductive inference services are used to reduce the intersection hypotheses space and to retrieve useful information for the driver.

In [24], Tenorth and Beetz present a system, which uses Prolog in order to process knowledge in the context of robotic control. It is especially designed for use with personal robots. Knowledge representation is based on DLs and processed via a Web Ontology Language (OWL) Prolog plug-in. In contrast to our approach, the Prolog based reasoning system is not used to recognize activities or reason about situations. Instead, it is used to query on its environmental model. Actions and events are observed by

the processing framework and used as knowledge facts. The knowledge base can be extended by using embedded classifiers in order to search for groups of instances that have common properties.

In [11], Neumann and Möller establish scene interpretation using DLs. Table cover scenes are analyzed and interpreted based on temporal and spatial relations of visually aggregated concepts. The interpretation uses visual evidence and contextual information in order to guide the stepwise process. Additionally probabilistic information is integrated within the knowledge based framework in order to generate preferred interpretations. This work is widened to cope with general multimedia data in [12], in which a general interpretation framework based on DLs is presented.

In [13], Springer et al. introduce a comprehensive approach for situation-awareness, which incorporates context capturing, context abstraction and decision making into a generic framework. This framework manages sensing devices and reasoning components which allows for using different reasoning facilities. Thus, DLs can be used for high level decision making.

These last examples and our previous work show that the usage of DLs bears great potential. Hence its adoption in the situation and action recognition task incorporated into the human robot cooperation (MAROCO) framework.

To the best of our knowledge, this is the only work to incorporate description logics and recognition of situations and human activities in the domain of cognitive robotics. For reasons of this, it was not possible to directly compare the runtime analysis results to concurrent research groups.

There are investigations concerning runtime analysis of descriptions logic reasoners (see e.g., [22, 23]) but they are not directly related to the robotics community. Still, they show that the FaCT++ system, which was used in this publication, is one of the best with respect to the given constraints of the software architecture MAROCO.

The main motivation writing this paper is introducing the description logics approach to recognition of situations and activities into the domain of cognitive robotics. There are just a few other research groups which are dealing with description logics in a similar research domain and the most related ones were referenced in this paper. Most attention was spent on extending the cognitive robotic system MAROCO with description logics and building a knowledge base for action and gesture recognition.

The markerless tracking of a human body in real time is not at the core of this paper. But this paper brings together markerless real time tracking of a human body, a safe robot path-planning module and the advanced description logic approach based on [1]. Thus, this paper intends to present novel results that are gathered from experimental investigations using description logics.

III. THE MAROCO FRAMEWORK

The MAROCO (human robot cooperation) framework [3, 4] is an implemented architecture that enables human centered computing realizing a safe human-robot interaction and cooperation due to advanced sensor technologies and fancy algorithms [7, 8].

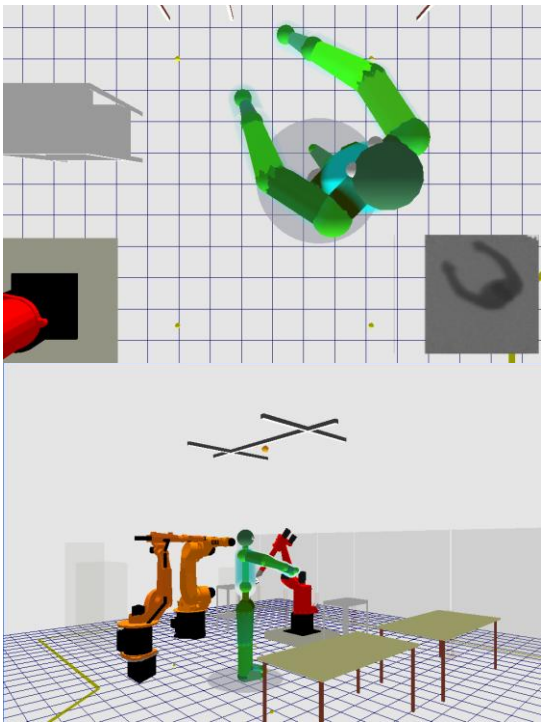

Figure 1. (Top) Reconstructed human model from depth images. (Bottom) Environmental scene model consisting of several kinematical chains. Three different industrial robots and a human model. All agents and robots have been reconstructed by MAROCO and are integrated into the virtual model in real-time including safety features extraction, risk estimation and path planning.

Every system implementing machine intelligence has to apply a sensor framework. The MAROCO system analyzes image sequences that are gathered from a 3D vision system [2] based on time-of-flight principle which is mounted to the top of the ceiling of the working cell (see Fig. 1). Modules dedicated to image sequence analysis make it possible to estimate more than a dozen of kinematical parameters, e.g., head orientation, upper body orientation, arm configuration, etc., of a human model without using any markers (Figure 1). The technical details of the methods realizing the real-time reconstruction of the kinematical model are not in the focus of this paper. Details can be found in [4, 7, 8].

As safety is one of the most demanding features when industrial robots get in contact with human workers, MAROCO is focused on estimating the risk for the human worker depending on the scene configuration. A variety of methods are integrated into the framework like pure functional evaluation, machine learning tools, e.g., support vector machines, and a two-threaded adaptive fuzzy logic approach, which at the moment makes the race [8].

Having estimated the risk, one is interested in finding a procedure minimizing the risk for both, the worker and machinery. Re-planning is an efficient tool minimizing the risk. A method for re-planning the path of the robot with respect to safety and real-time capability is presented in [5].

All these modules enable safe human-robot interaction and cooperation. Safety, in this context, is understood in general terms and from a scientific point of view.

The kinematical model also allows for recognition of human activities and situations inside the robot working area. Using DL reasoning facilities, conclusions about occurring situations, actions, their temporal relations and expectations about robot behavior can be drawn. This is presented in the following sections.

IV. The Recognition Module

This section is dedicated to discuss the recognition module including its components and modeled knowledge base after a very brief introduction to DLs.

A. Description Logics

In this paper, DLs [9] are used to formalize knowledge about situations, actions and expectations. DL is a 2-variable fragment of First Order Logic and most DLs are decidable. Thus, sound, complete and terminating reasoning algorithms exist. Due to this reason, different efficient algorithms have been engineered and implemented into diverse reasoning systems.

A DL knowledge base is divided distinctly into general knowledge and knowledge about individuals in a domain. The former defines the terminology of the domain and its axioms are declared in the terminology box, hence TBox. The latter defines assertions about individuals and, therefore, is declared in the assertion box, hence ABox. This allows for modular and reusable knowledge bases and thus for more efficient coding of knowledge [10].

Due to DL's open world assumption, it can deal naturally with incomplete information, which is essential in reasoning taking sensor data into account.

B. Reasoner Systems and Interfaces

By progress in the development of the semantic web, many reasoning systems were engineered in order to implement efficient algorithms, e.g., RacerPro [14], FaCT++ [15] or Pellet [16]. These systems can be interfaced by different means. In [1], we used the Pellet system and the DIG-interface. Pellet allows for efficient reasoning [22, 23]. The DIG-interface, on one side, has the advantage of its separation of application and reasoner by the means of programming language and execution place, because it implements communication via TCP and XML messages. On the other side, this interface is superseded by more recent developments which incorporate more features of recent web ontology languages, e.g., OWL API [25]. These new interfaces are based on Java implementations.

Because the MAROCO framework is implemented in C++, a Java based implementation of an interface was not feasible. The FaCT++ reasoning system, though, is written in C++. Furthermore, as shown in [15, 22, 23], FaCT++ uses very efficient algorithms. Thus, it is used as reasoning facility for the recognition module.

FaCT++ uses a tableaux decision procedure which includes optimization techniques that exploits structural features of typical ontologies [15]. Due to its architecture, it allows for a wide range of heuristic optimizations.

In this work, version 1.5.1 of FaCT++ was used. Besides the introduction of a new volatile axiom type, described in Section IV C, there are no further optimizations implemented into the reasoner system. Furthermore, FaCT++ does not allow for incremental reasoning after assertions have been retracted, added, or changed.

C. The Module Design

The recognition module needs to fulfill at least the tasks of instantiating a Description Logics reasoner, managing the knowledge base and managing the reasoner results.

The recognition module is embedded in the MAROCO Framework and is executed in parallel to the sensor data analysis module. This allows for fast computations for the safety relevant robot control and, with less priority, computations for higher cognitive processes, e.g., situation and activity recognition.

The recognition module consists of different subcomponents (see Figure 2).

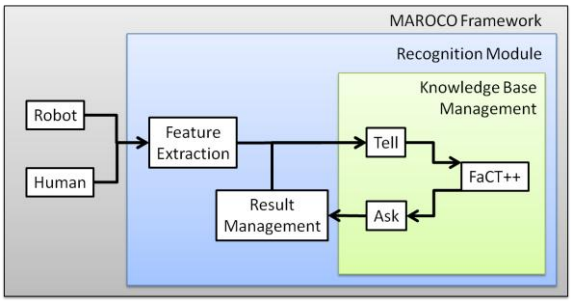

Figure 2. Components of the recognition module.

The knowledge base management follows a functional approach called *Tell&Ask* [9]. After defining a knowledge base – the *tell* operation – reasoner results and information can be retrieved – the *ask* operation. The modification of an existing knowledge base after using an *ask* operation can be achieved by using the *retract*-functionality of FaCT++. It allows single axioms to be marked as unused and flags the knowledge base as changed. In a subsequent processing cycle new axioms can be added. The retraction of axioms in FaCT++ does not actually delete these axioms due to its inner data management. Thus, by repeated retraction and addition of axioms, memory requirements increase.

In the realm of sensor data processing, it is advisable having an update functionality rather than retraction and addition. Thus, we augmented the FaCT++ reasoning system with a new *volatile* axiom. This allows updating the DL knowledge base without increasing its memory usage (see Figure 3).

As a consequence the recognition module needs to manage an up-to-date model of the knowledge base, which consists of domain specific knowledge and assertions dependent on the current kinematical human model and robot specific parameters. This distinction corresponds in Description Logics with TBoxes and ABoxes. The domain specific knowledge is modeled a priori; the assertional knowledge is updated in each runtime cycle. The modeled

knowledge base will be explained in more detail in Section IV D.

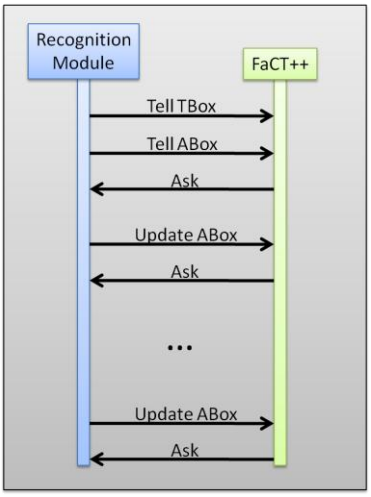

Figure 3. Interaction between recognition module and FaCT++.

As the assertional knowledge depends on kinematical parameters a feature extraction component is applied in order to fill the attribute values of the assertions. The following features are important w.r.t. the component *Human*:

- Angles of both elbows,
- Angles of both shoulder joints,
- Angle difference between head orientation and robot,
- Walking velocity, and
- Used tool.

The feature *used tool* is not supported by existing sensors at the moment and is therefore simulated. It can have one of the following values: *none*, *measurement tool* or *working tool*. The simulation of this parameter can be influenced directly by user input using standard human machine interfaces. As a result, complex working scenarios can be modeled and analyzed.

The component *Robot* provides the parameters for: gripper status, which can be *empty* or *full*, and movement status, which can be one of

- Stopped,
- Following predefined path, or
- Follow user given task.

During feature vector creation, extracted values are mapped onto sharp sets. The knowledge base is then populated with corresponding set strings which can be used for comparative operations during reasoning.

One major aspect of understanding human activity is modeling temporal relations between different actions. In this work, these relations are introduced by defining an *after*-role. Hence a certain action can only be recognized if certain other actions occurred prior. This *after*-role can be regarded as defining preconditions onto actions. Previously recognized actions need to be included in the knowledge base in order to allow for correct recognition of current actions. All recognized actions are stored by the reasoner result management component and are retrieved during updating of the knowledge base. Each occurred action is included in the DL knowledge base as an ABox instance.

The after-role is defined as a transitive role. Thus, in order to relate a new action instance to all past ones, only the relation to the previous action needs to be defined in the ABox update step.

D. The Knowledge Base

In Figure 4, the ontology about situations which is modeled by the knowledge base is presented. The concept *Situation* has the attribute *Number Humans* to distinguish between the concepts *Robot alone* and *Human present*.

In addition to the described situation ontology in [1], a new sub-concept *Partially Attentive* is introduced. It allows for a more detailed differentiation if observed actions and instructions need to be complied by the robot.

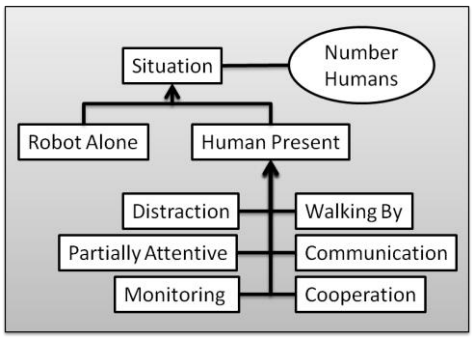

Figure 4. ER model of the situation ontology.

Depending on the *Activity*, which is *done by* the *Human*, different sub-concepts can be distinguished. In order to relate the concepts *Situation*, *Activity* and *Human*, the roles *done by* and *takes place* are defined.

In Figure 5, the concept *Activity* with its sub-concepts is depicted. In the line of the extension of the situation ontology, the concept *Paying Partially Attention* is introduced to the activity ontology. The concept *Human*, its properties and the defined roles are not shown for clarity reasons.

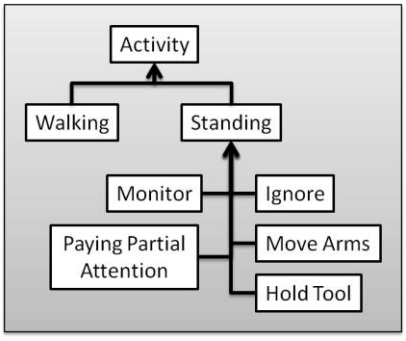

Figure 5. ER model of the activity ontology.

In Figure 6, the ontology concerning *Actions* and *complex Actions* is shown. As pointed out above, actions can

have a temporal relation expressed as *after*-role. The action *Put Tool Away* can only happen after occurrence of the action *Take Tool*. This role is also exploited in complex actions, e.g., *Continue Robot Motion* can only be signaled after *Stop Robot* was recognized.

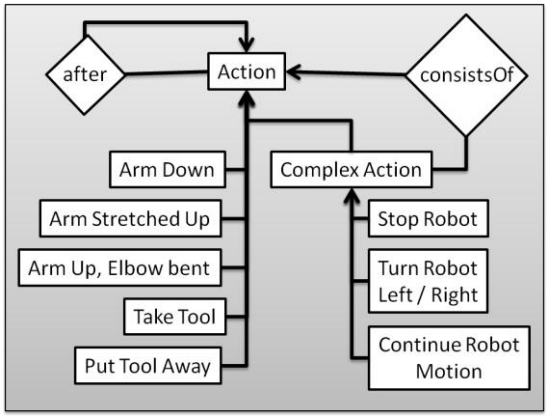

Figure 6. ER model of the action ontology.

Actions can be regarded as atomic concepts, whereas complex actions consist of other actions, regardless of atomicity. The concepts *Take Tool* and *Put Tool Away* are considered atomic, because they are defined by and based on a single attribute *Used Tool*. This attribute is directly altered by user input, therefore, does not result from sensor data analysis. The role *doneBy* which is defined for activities is also modeled for actions. For reasons of readability this relation is not depicted.

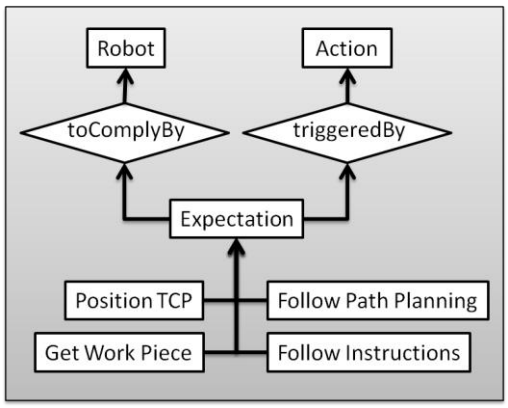

Figure 7. ER model of the expectation ontology.

The occurrence of the situation *Cooperation* implies that there are *expectations* towards the robot behavior. Moreover, an expectation can be *triggered by* an action (see Figure 7). This allows for reasoning about expectations without necessarily recognizing a triggering action. This implicit relation is also exploited between the activities *Monitor*, *Hold Tool* and *Actions*.

The resulting expectations can be used as input to a task planning module. The scope of each possible expectation is variable. *Position TCP* and *Get Work Piece* are concrete commands. Complying *Follow Instructions*, on the other side, needs also the information about recognized actions.

V. Experimental Results

For reasons of experimental analysis of the implemented activity and situation recognition different courses of action were executed and the recognition results were recorded.

In order to analyze different scenarios efficiently, means of automated feature value presetting have been implemented. The overall analysis is based on these presets and on actual sensor data processing. Hence natural movements and transitions between actions can be tested and special use cases can be investigated.

In this section, recorded recognition results will be illustrated and discussed. Due to the advancements and changes to the recognition system compared to the presented work in [1], all experimental investigations were repeated and have to stand up to comparison. During result analysis special emphasis was put on efficiency and elapsed processing time.

A. Exemplary Result Records

The recorded experimental results contain a timestamp which indicates the starting time of the recognition cycle in milliseconds since program start. This timestamp is then followed by the extracted feature values if there is a human worker in the supervised area. The components of the feature vector are listed in following order: Angle arm left, angle arm right, angle elbow left, angle elbow right, walking velocity, angle difference between head orientation and robot, holding tool, gripper status and robot movement status.

The next number is the timestamp of the final result message from the DL reasoner (see Table I). Results will be recorded whenever there are new insights. Thus, the last two lines of Table I have no special entries past the last return timestamp.

TABLE I. Example Record Based on Sensor Data

```
34942 34980 RobotAlone FollowPathPlanning
34980 0 0 0 0 3 105 0 0 0 35082 Distraction Ignore
35082 0 0 0 0 1 125 0 0 0 35240
35240 0 7 0 9 2 117 0 0 0 35408
```

Table II demonstrates the recognition of different situations and activities. Furthermore, an additional action and expectation are reasoned and recognized.

TABLE II. Example Record Based on Presets

```
75041 90 0 0 0 20 0 0 0 1 75141 WalkingBy Walking
. . .
79949 90 0 0 0 20 0 0 0 1 80109
80109  0 0 0 0  0 0 1 0 1 80164 Cooperation
                      HoldTool TakeTool getWorkPiece
```

During a recognition cycle all recognized concepts are returned from the DL reasoner in a single flush, therefore,

the number of lines in the records represents the number of returned responses.

Table I and II also depict the different feature values achieved by either using processed sensor data – Table I – or presets – Table II respectively. Assuming the recognition is fast enough, natural movement and action transitions can be observed. In the next section, this will be investigated.

B. Results

Tables I and II already indicate that the processing time of a recognition cycle varies between 100 ms and 200 ms. By analysis of a large amount of processing cycles, this indication needs to be corrected only slightly upwards.

TABLE III. RESULTS FROM EVALUATION (PRESETS)

# Recognition cycles	2830		# > 500 ms	441 (15.58%)
Ø Response time [ms]	263.19		# > 1000 ms	100 (3.53%)
Min [ms]	54		# > 1200 ms	100 (3.53%)
Max [ms]	1221		# > 1220 ms	1 (0.03%)
Standard Deviation [ms]	284.17			

In Table III, the results of 2830 recognition cycles are summarized. Feature value presets were used and it shows that the average processing time is approximately 263 ms. The lower bound is 54 ms. The casual outliers take up to 1.2 seconds in worst case scenarios. The number of cycles taking more than 1 second reaches 3.53% of all cycles. Almost all of these outliers are situated between 1.2 and 1.22 seconds.

In Table IV, corresponding results are shown using actual processed sensor data during recognition. Recorded were 2680 cycles with an average processing time of approximately 237 ms. This seems faster than using value presets. Interestingly, the maximal outliers and the standard deviation are worse.

TABLE IV. RESULTS FROM EVALUATION (SENSOR DATA)

# Recognition cycles	2680		# > 500 ms	381 (13.46%)
Ø Response time [ms]	236.68		# > 1000 ms	112 (3.96%)
Min [ms]	37		# > 1200 ms	79 (2.79%)
Max [ms]	1543		# > 1500 ms	41 (1.45%)
Standard Deviation [ms]	320.24			

In Figure 8, the processing time of some cycles using presets are shown. The reoccurring nature of the feature value presets can clearly be recognized. It can also be seen, that the outliers are systematic and presumably dependent on feature values.

In Figure 9, the cycle processing time and the corresponding returned number of recognized concepts are depicted. It can be seen, that the number of resulting concepts is not directly related to the cycle time.

In order to investigate the difference in runtime behavior further, the evident change in cycle times, marked by a red rectangle in Figure 9, is examined in Table V. The first number in each row marks the cycle index. The first two rows enumerate the used feature values during those cycles. The bottom rows present the recognized concepts. The only difference is the occurrence of the concepts *Monitor* (*Monitoring*) and *Ignore* (*Distraction*) respectively. This change is triggered solely by the change of the angle difference between viewing angle and robot, namely changing from 0 to 60.

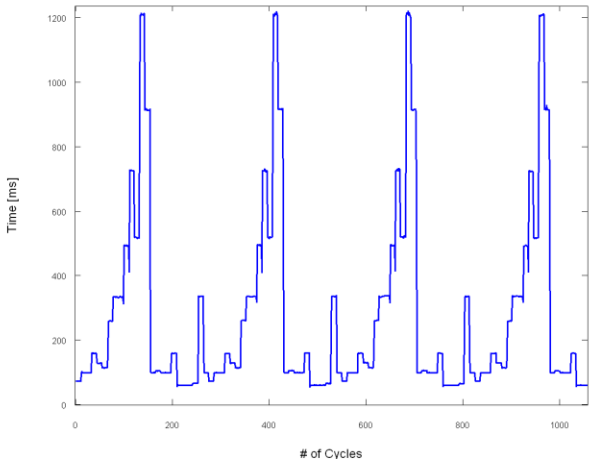

Figure 8. Runtime analysis with reoccurring feature value presets.

In the knowledge base the definition of these concepts only differs in the evaluation of this angle difference. Thus, the change in runtime duration cannot be directly related to the character of declaration of these concepts.

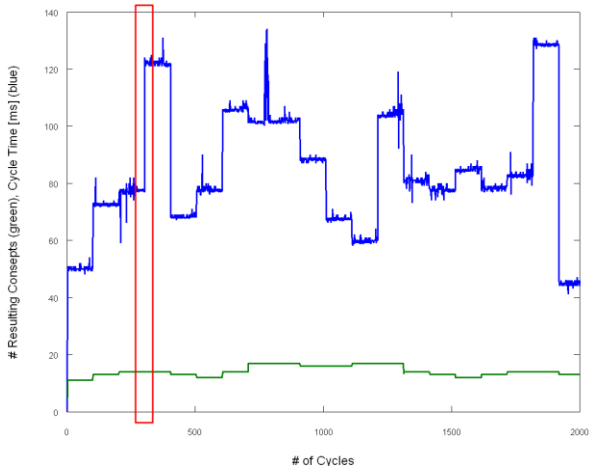

Figure 9. The bottom line (green) shows the number of returned concepts. The upper line (blue) shows the corresponding cycle processing time. For recognition, feature value presets were used. The red rectangle highlights the examined feature value change.

It is noticeable, that the increase in cycle times occurs with the recognition of a *Distraction*. This is counterintuitive, as the possibilities of interaction decrease with distraction and increase with an attentive human worker.

The repeatable and counterintuitive observation will need further investigation in order to optimize the DL knowledge base and achieve better performance.

TABLE V. EXAMINATION OF FEATURE VALUE CHANGE AND CYCLE TIME

```
303 0 0 0 0 0  0 0 0 1                  used feature
304 0 0 0 0 0 60 0 0 0                        values

303 TOP HumanPresent Monitoring TOP Standing
    Monitor ArmsDown FollowPathPlanning
304 TOP HumanPresent Distraction TOP Standing
    Ignore ArmsDown FollowPathPlanning
```

In Figure 10, the cycle processing time is shown, when using processed sensor data. As expected, the repeatability of the preset feature values cannot be achieved. Peaks of more than 1000 ms are not a rare coincidence. Thus, further investigations about runtime durations are necessary. Nevertheless, most processing cycles have shorter durations than 800 ms, and as Table IV shows, the average processing time is below 237 ms. This allows for recognition frame rates of about 4.2 Hz on average.

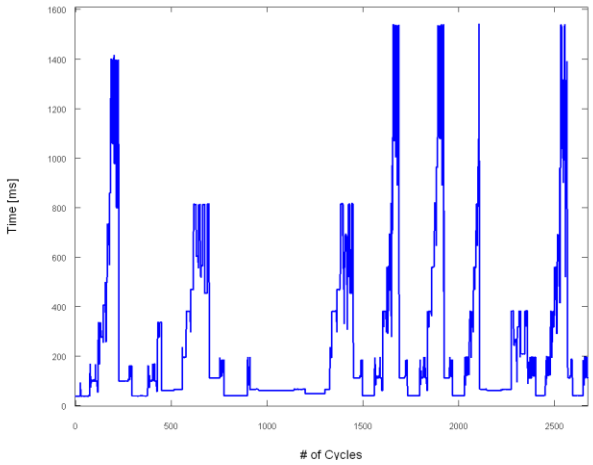

Figure 10. Runtime analysis with processed sensor data.

In Figure 11, the frame rates of the MAROCO framework are shown. In each frame sensor data is processed, risk is evaluated and the robot motion and path planning are adapted accordingly. The frame rates reach occasional lows with 15.9 Hz and average out around 33.8 Hz. Recorded sensor streams were used for playback during these tests in order to be independent on a possible bottleneck due to sensor restrictions. The repeating sensor input can clearly be recognized in Figure 11.

The peaks in performance are reached when there is no human worker in the supervised working area of the robot. These peaks reach up to 126 Hz. When the human reenters this area, the data shows noticeable performance decrease. The circles in Figure 11 mark obvious examples. In these cases, the performance drops to a low and recovers afterwards to converge with the average frame rate. This indicates the adaption of the path planning [5] to the human presents. Though, this data is not sufficient to allow profound analysis. Thus, the recognition module might cause these performance decreases.

By using the kinematical human model, recognition of gestures and human motion can be analyzed. In Figure 12, different examples of recognized situations and actions are depicted. The topmost picture shows a human watching the robot. The icons to the right symbolize the recognition results. Thus, the identified situation is *Monitoring*. No specified action is recognized. The robot is expected to carry on with its task of following its preplanned path.

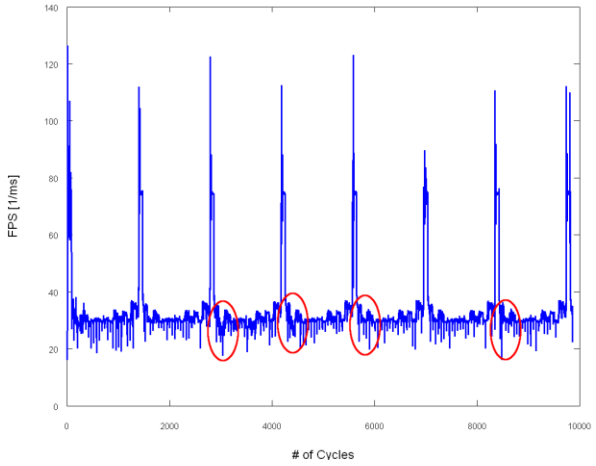

Figure 11. Frame rates of the MAROCO sensor data processing and robot control cycle running in parallel to the recognition module. The circles mark the reentrance of the human into the work area of the robot with noticeable decrease of performance.

In the second image of Figure 12, a human is communicating with the robot. The complex action to signal a stop of robot movement is recognized. Thus, the resulting situation is identified as *Communication*. The robot is expected to comply with the users instructions.

The bottommost picture also shows a human communicating. The complex action to signal a right turning movement is recognized. The robot is expected to comply accordingly.

TABLE VI. EXAMPLE RECORD FOR NATURAL MOVEMENT

```
342000 0 0 1 0 7 4 0 1 1 342111 Monitoring Monitor
                                followPathPlanning
. . .
342779 0 0 1 0 7 3 0 1 1 342890
342890 14 13 10 11 3 4 0 1 1 342952
342952 20 26 13 16 3 4 0 1 1 343012
343012 35 39 16 22 4 4 0 1 1 343073
343073 47 46 19 28 0 5 0 1 1 343134
343134 47 46 19 28 0 5 0 1 1 343195
343195 54 51 21 31 0 5 0 1 1 343428 Comm. MoveArms
                                 StopRobot followInstructions
```

Table VI shows an example in which a human first watches the robot. This concludes the expectation, that the robot shell follow a planned path. After some time the

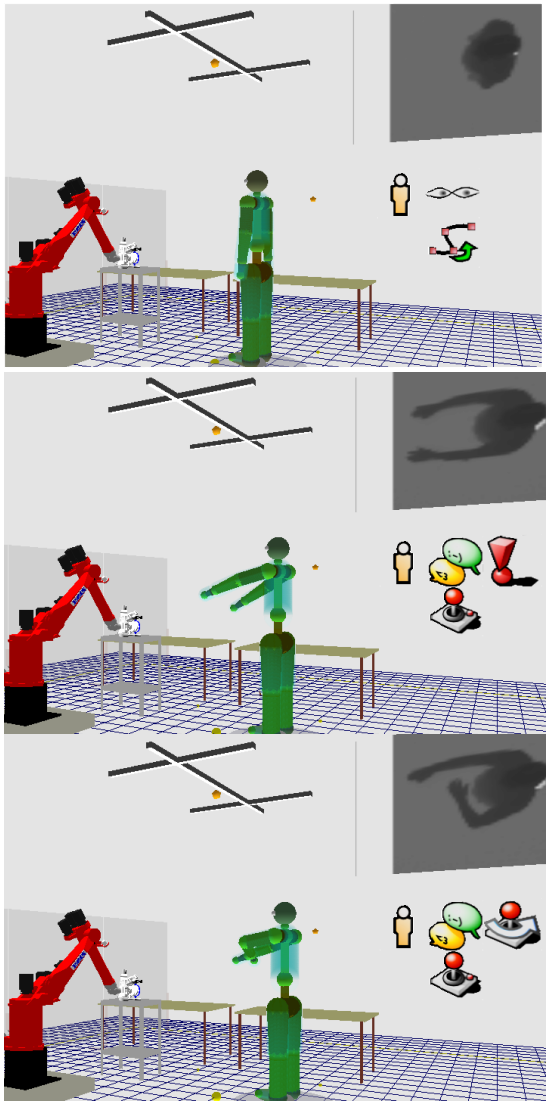

Figure 12. Different examples of recognized situations and actions. The icons on the right in each image symbolize the recognition results.

human moves his arms which results in a communicative situation. The reasoning results in the expectation that the robot shell comply with the instructions. It can be seen, that both arms are moved upwards at the same time. The value changes are observable over some cycles.

Consequently natural movements and actions can be recognized despite the average cycle processing time of approx. 250 ms.

Tables II and VI demonstrate that depending on situation and actions expectations are generated. The generation of expectation is also dependent on the robot movement status. Table VII shows that at first a cooperative situation is recognized and a generated expectation *get Work Piece*. At this moment the robot was following a planned path, which is signaled as `1` in the feature vector. In the simulation incorporated in MAROCO, this generated expectation leads to a change of the robot movement status which sets the corresponding feature value to `2`, meaning the robot is obeying instructions. This change allows the reasoning to conclude the new expectation to position the robot's tool center point in order to ease the work that the user is about to do with the work piece.

TABLE VII. EXAMPLE FOR DYNAMIC EXPECTATION REASONING

```
96795 75 0 21 0 0 3 1 0 1 97287 Coop. HoldTool
                                TakeTool getWorkPiece
97289 75 0 22 0 0 0 1 1 2 97799 positionTCP
```

This process of interaction between reasoner results and robotic behavior demonstrates the dynamic abilities of the presented approach to recognize and understand situations and actions.

C. Evaluation of Results

The results demonstrate that the capabilities of the presented approach reach beyond sole activity and situation recognition. By generating expectations towards robot behavior, an understanding of the situation can be achieved. This induction of relations between concepts can hardly be realized by purely probabilistic methods.

The achieved processing cycle time of approx. 250 ms does not allow for safe cooperation based only on the recognition module. Thus, the MAROCO framework uses its implemented techniques and algorithms to enforce safety and real-time capabilities during robot motion. Nevertheless, the measured results will be used to quantify improvements of later developments.

In comparison with presented results in [1], an increase of performance was achieved. The recognition module executes its processing cycle more than two times faster on average. Due to the incorporation of the DLs reasoning system into the MAROCO framework, these speedups are gained. It avoids the overhead of the DIG-interface and allows for a more thorough investigation concerning runtime and cycle duration.

Having a closer look at the results, there are still outliers that take more than a second. All cycle times were faster than 5 seconds, which was observed in [1]. Investigation in concept dependent runtime is needed in order to optimize the dependencies between concepts and the overall recognition performance.

In Table VI, it is demonstrated, that the rates of changes of actions can be captured. Still, to the best of our knowledge, there are no investigations concerning the rate of change of human actions in human-robot cooperative scenarios. During the experiments conducted for this publication, the subjective impression of the MAROCO system was responsive and accurate. Nevertheless, it can be assumed that repetitive work can be carried out faster by an experienced human worker than the current module is capable of recognizing. More effort has to be spent to be able to evaluate the real-time capabilities of the recognition module accurately.

To the best of our knowledge, there are no other such time related results made available in the field of industrial

human-robot cooperation or another related field close to it so far.

VI. Summary and Future Work

In this paper, a situation and action recognition module was presented, which is capable of generating expectations towards robotic behavior.

A knowledge base containing domain and assertional knowledge is modeled. It defines concepts about situations, activities, actions and expectations. These concepts are linked and related by role definitions. Temporal associations of actions are modeled by an *after*-role, which allows preconditioning the recognition of certain actions.

Description Logics are used to define the knowledge base. The Description Logics reasoner FaCT++ was incorporated into the MAROCO framework. A *volatile axiom* definition was introduced to the reasoner to avoid increasing memory requirements due to repeated updates to the assertional knowledge.

In order to express value constraints on concept attributes, the feature extraction process maps feature values onto sets, which can be represented as strings in the knowledge base. This allows additionally for support of future development in regards to symbol based classifiers. This might ease the load on DL reasoning and achieve further increase of performance.

During evaluation the effectiveness was shown. Situations, activities and naturally conducted actions are recognized. Expectations are generated and can influence dynamically subsequent processing cycles.

Compared to previous work, an increase of recognition performance was achieved. The recognition cycle requires less than half the processing time on average. Extreme outliers of over 1.5 seconds duration do not occur.

The here presented experimental results are promising for further research in the field of cognitive industrial robotics.

The next steps will be modeling a broader knowledge base in order to incorporate multi-robot setups and more complex cooperation scenarios. Also, the implementation of action plan recognition will deepen the understanding of situations and enable the analysis of complex cooperation scenarios.

Optimizations to the reasoning system FaCT++ matched to the structure of the implemented ontology might increase recognition performance. Thus, further investigation of concept dependent cycle time durations is needed. Also, implementation of incremental reasoning can avoid processing of unchanged knowledge.

Moreover, investigations concerning rate of changes of human actions will allow better evaluation of real-time constraints and capabilities of the recognition module.

Using generated expectations towards robotic behavior as input for subsequent task planning will augment the cooperative experience and will allow research on system responsiveness and accuracy. It will also enable investigations concerning interaction of reasoner results and robotic behavior.

It was taken a stand against the probabilistic way of estimating actions from image sequences in the beginning of the related work section. But it is suggested to evaluate different approaches in the near future which also take probabilistic methods into account or maybe apply different methods in a boosting like manner bringing together the best of both worlds.

Acknowledgment

We would like to thank Dmitry Tsarkov for his support during development regarding the incorporation of FaCT++ into the MAROCO framework.

References

[1] J. Graf, S. Puls, and H. Wörn, “Recognition and Understanding Situations and Activities with Description Logics for Safe Human-Robot Cooperation”, in Proc. of Cognitive 2010, pp.90-96, 2010.

[2] http://www.pmdtec.com/products-services/pmdvisionr-cameras/pmdvisionr-camcube-30/ [Last visited on 2012-01-19]

[3] J. Graf and H. Wörn, “An Image Sequence Analysis System with Focus on Human-Robot-Cooperation using PMD-Camera”, in VDI Proc. of Robotik 2008, June 2008, pp. 223-226.

[4] J. Graf and H. Wörn, “Safe Human-Robot Interaction using 3D Sensor”, in Proc. of VDI Automation 2009, June 2009, pp. 445-456.

[5] J. Graf, S. Puls, and H. Wörn, “Incorporating Novel Path Planning Method into Cognitive Vision System for Safe Human-Robot Interaction”, in Proc. of Computation World, pp. 443-447, 2009.

[6] S. Bechhofer, “The DIG Description Logic Interface: DIG/1.1.”, in Proc. of the 2003 Description Logic Workshop, 2003.

[7] J. Graf, F. Dittrich, and H. Wörn, „High Performance Optical Flow Serves Bayesian Filtering for SafeHuman-Robot Cooperation”, in Proc. of the Joint 41th Int. Symp. on Robotics and 6th German Conf. on Robotics, pp. 325-332, Munich, 2010.

[8] J. Graf, P. Czapiewski, and H. Wörn, “Evaluating Risk Estimation Methods and Path Planning for Safe Human-Robot Cooperation”, in Proc. of the Joint 41th Int. Symp. on Robotics and 6th German Conf. on Robotics, pp. 579-585, Munich, 2010

[9] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, “The Description Logic Handbook”, 2nd Edition, Cambridge University Press, 2010.

[10] B. Hummel, W. Thiemann, and I. Lulcheva, “Description Logic for Vision-Based Intersection Understanding”, in Proc. of Cognitive Systems with Interactive Sensors (COGIS), Stanford University, CA, 2007.

[11] B. Neumann and R. Möller, “On Scene Interpretation with Description Logics”, in Image and Vision Computing, vol. 26, pp. 81-101, 2008.

[12] R. Möller and B. Neumann, “Ontology-Based Reasoning Techniques for Multimedia Interpretation and Retrieval”, in Semantic Multimedia and Ontologies, part 2, pp. 55-98, Springer London, 2008.

[13] T. Springer, P. Wustmann, I. Braun, W. Dargie, and M. Berger, „A Comprehensive Approach for Situation-Awareness Based on Sensing and Reasoning about Context“, in Lecture Notes in Computer Science, vol. 5061, pp. 143-157, Springer, Berlin, 2010.

[14] V. Haarslev, R. Möller, and M. Wessel, “RacerPro User’s Guide and Reference Manual”, Version 1.9.1, May 2007.

[15] D. Tsarkov and I. Horrocks, “FaCT++ Description Logic Reasoner: System Description”, in Lecture Notes in Computer Science (LNCS), vol. 4273, pp. 654-667, 2006.

[16] E. Sirin, B. Parsia, B. Grau, A. Kalyanpur, and Y. Katz, “Pellet: A practical OWL-DL reasoner”, in Web Semantics: Science, Services and Agents on the World Wide Web, vol.5 (2), pp. 51-53, 2007.

[17] V. Krüger, D. Kragic, A. Ude, and C. Geib, „The Meaning of Action: A Review on action recognition and mapping“, in Proc. of Advanced Robotics, Vol. 21, pp. 1473-1501, 2007.

[18] P. Raamana, D. Grest, and V. Krueger „Human Action Recognition in Table-Top Scenarios : An HMM-Based Analysis to Optimize the Performance", in Lecture Notes in Computer Science (LNCS), Vol. 4673, pp. 101-108, 2007.

[19] Y. Wu, H. Chen, W. Tsai, S. Lee, and J. Yu, „Human action recognition based on layered-HMM", in IEEE Inter. Conf. on Multimedia and Expo (ICME), pp.1453-1456, 2008.

[20] Y. Shi, Y. Huang, D. Minnen, A. Bobick, and I. Essa, „Propagation Networks for Recognition of Partially Ordered Sequential Action", in Proc. of Computer Vision and Pattern Recognition (CVPR), Vol. 2, pp. 862-869, 2004.

[21] D. Minnen, I. Essa, and T. Starner, „Expectation Grammars: Leveraging High-Level Expectations for Activity Recognition", in Proc. of Computer Vision and Pattern Recognition (CVPR), Vol. 2, pp. 626-632, 2003.

[22] T. Gardiner, I. Horrocks, and D. Tsarkov, "Automated Benchmarking of Description Logic Reasoners", in Proc. of the 2006 Intern. Workshop on Description Logics (DL2006), Windermere, Lake Districrt, UK, 8 pages, June 2006.

[23] Z. Pan, "Benchmarking DL Reasoners Using Realistic Ontologies", in Proc. of the First OWL Experiences and Directions Workshop, 2005.

[24] M. Tenorth and M. Beetz, "KNOWROB – Knowledge processing for Autonomous Personal Robots", in IEEE Inter. Conf. on Intelligent Robots and Systems (IROS), 2009.

[25] M. Horridge and S. Bechhofer, "The OWL API: A Java API for Working with OWL 2 Ontologies", in Proc. of OWL: Experiences and Directions, 2009.