Journal of Ambient Intelligence and Smart Environments 6 (2014) 623–649
DOI 10.3233/AIS-140290
IOS Press

Automatic understanding of group behavior using fuzzy temporal logic

Joris IJsselmuiden [a,*], David Münch [a], Ann-Kristin Grosselfinger [a], Michael Arens [a] and Rainer Stiefelhagen [b]

[a] *Fraunhofer IOSB, Fraunhoferstraße 1, 76131, Karlsruhe, Germany*
E-mail: joris.ijsselmuiden@wur.nl, {david.muench,ann-kristin.grosselfinger,michael.arens}@iosb.fraunhofer.de
Phone: 0031 317 481258, Fax: 0031 317 484819

[b] *Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Vincenz-Priessnitz-Straße 3, 76131 Karlsruhe, Germany*
E-mail: rainer.stiefelhagen@kit.edu

Abstract. Automatic behavior understanding refers to the generation of situation descriptions from machine perception. World models created through machine perception can be used by a reasoning engine to deduce knowledge about the observed scene. For this study, the required machine perception is annotated, allowing us to focus on the reasoning problem. The applied reasoning engine is based on fuzzy metric temporal logic and situation graph trees. It is evaluated in a case study on automatic behavior report generation for staff training purposes in crisis response control rooms. The idea is to use automatically generated reports for multimedia retrieval to increase the effectiveness of learning from recorded staff exercises. To achieve automatic report generation, various group situations are deduced from annotated person tracks, object information, and annotated information about gestures, body pose, and speech activity. The contribution of this paper consists of improvements to the existing knowledge base that models the group situations, and a quantitative evaluation using a substantial set of self-developed data and ground-truth. We also describe recent improvements to the self-developed software tools for annotating and visualizing data, ground-truth, and results.

Keywords: Automatic behavior understanding, group behavior, fuzzy metric temporal logic, situation graph trees

1. Introduction

Automatic behavior understanding, or the generation of situation descriptions from multimodal machine perception observing multiple objects is an unsolved problem, and the encompassing research field of high-level reasoning still contains many challenges. Reasoning methods suffer from problems stemming from the machine perception they depend on, but as machine perception progresses and the perceived world gets more complex, e.g. in research areas such as smart homes, smart work environments, smart cities, and the internet of things, the need for powerful reasoning methods becomes apparent.

This is the motivation of our work. Building on [29], we are taking steps toward general purpose, domain independent reasoning methods for automatic behavior understanding. In order to keep the focus on high-level reasoning, hypothetical machine perception outputs are annotated based on real audiovisual data, using a self-developed data annotation tool. This approach allows us to perform targeted evaluations of the reasoning methods only, as opposed to most other studies that combine perception and reasoning. It is often difficult to perform a targeted error analysis and compare results, when reasoning and perception are evaluated as a single system.

Audiovisual data was recorded during a fire brigade staff exercise. The data was annotated with hypothetical machine perception outputs, forming the input for

*Corresponding author.

fully automatic reasoning methods. These are based on fuzzy metric temporal logic (FMTL) and situation graph trees (SGTs) and model group situations that occur during staff exercises. Their output consists of situation descriptions which are compared to annotated ground-truth to evaluate performance. This work represents a step along the way toward a real-time system containing various machine perception components instead of annotated hypothetical machine perception. Ultimately, such a system should be applied to a variety of application domains, having embodiment and action generation as well as synchronous real-time visualizations of sensor data, machine perception, and situation descriptions.

Contribution In [29], fuzzy metric temporal logic and situation graph trees (FMTL and SGTs) were applied to automatic behavior understanding in smart work environments for the first time. That study completed several steps toward an integrated development toolkit for such purposes: a new dataset, new software tools for data analysis and annotation, and a unique case study with a large amount of perception modalities and objects. Furthermore, the study contributed a new FMTL/SGT knowledge base for this case study that is also applicable to other domains, as well as the first experimental results for the case study. It also contained a discussion on how to handle imperfect input data using FMTL and SGTs.

In this paper, we extend the work presented in [29] with the following contributions. First, the knowledge base that models the group situations is extended and improved: based on our previous work presented in [29,42], a variety of new and improved FMTL rules have been developed, and we introduce a new set of SGTs that use them. Second, a quantitative evaluation of the system and its current results is performed, using a considerably larger and more sophisticated set of self-developed data and ground-truth than in [29]. Furthermore, we describe the self-developed dataset and software tools that are used to perform the evaluation. New features as compared to [29] include bounding box visualization of ground-truth and results and more flexible ground-truth annotation functionality.

Case study scenario The application we focus on is automatic behavior report generation for staff training purposes in crisis response control rooms. The presented case study is situated at the State Fire Service Institute North Rhine-Westphalia, during one of their staff exercises for crisis response control room operations (Fig. 1). Such exercises consist of a role playing

Fig. 1. Staff exercise at the State Fire Service Institute North Rhine-Westphalia. Several situations we aim to recognize are visible here: conversation, analyzing a document together, editing a display.

setup where some participants take on the roles of control room staff while others are concerned with simulating field units, crisis dynamics, distress calls, and radio communications. For this particular exercise, the subject was a collision between a passenger train and a cargo train carrying hazardous material.

As described in more detail in [29], such control room operations follow strict procedures and the staff members have predefined roles. The typical workflow (corresponding to the recorded audiovisual data) consists of periodic cycles of briefings and dynamic control room operations. The briefings contain multiple phases, each lead by one of the first officers that is in charge of a specific functional area. During the dynamic operations between the briefings, the staff members scatter across the room, attending to their displays, documents, and messages. Groups are constantly forming and breaking, and there is a lot of discussion going on. The knowledge base presented in this paper aims to recognize the group behavior during dynamic control room operations.

In order to automatically generate behavior reports in this setting, the different types of group formations, person-person interactions and person-object interactions need to be modeled and recognized. First, the static and dynamic properties of the persons and objects in the room need to be perceived using machine perception. In our case, hypothetical machine perception outputs are annotated instead. Second, these properties need to be analyzed by a reasoning engine to generate the behavior reports. If our machine perception components can perceive (or annotations are provided for) the identity, position, orientation, and speech activity of the staff members over time, and the

state of the objects in the room, the reasoning engine can automatically generate corresponding descriptions and visualizations of group formations and interaction patterns, i.e. of who is doing what with whom, using which support tools. An example of such an automatically generated description could be: "First officers S1 and S2 are editing the overview map together."

In today's staff exercises, individual and task oriented feedback is hard to come by and is hence often neglected. This is mainly because it is impractical and time-consuming to obtain, given the available tools. Automatically generated behavior reports can improve this situation. In particular, they can be used to automatically or semi-automatically assess the performance of the individual participants: How close did they follow standard operating procedures? Who should have been part of which group? How long did it take them to complete specific tasks? Which bottlenecks have occurred? Which available resources were left idle? When combined with visualizations, audiovisual recordings, and trails of developments in the crisis response software that is being used (field unit status, crisis dynamics, and other context information), such a system could provide a rich information source for feedback and learning, conveniently searchable for specific situations. The presented case study is of interest because of its uniqueness and its large amount of perception modalities and objects.

Other application domains The applied reasoning methods are domain independent as they are only connected to underlying machine perception (or annotations) through a generic symbolic interface. The applied annotation process is also flexible and applicable to other application domains. Because the reasoning methods can use any combination of machine perception components, such as human pose estimation, speech recognition, vehicle tracking, and monitoring of activities in cyberspace, the range of possible application domains is wide: multimedia retrieval, robotics, smart homes, ambient assisted living, smart work environments, intelligent user interfaces, smart cities, the internet of things, indoor and outdoor surveillance, air traffic control, decision support for military and civil security, and more. Furthermore, in a complex system of systems, the applied reasoning methods can facilitate camera control, sensor deployment planning, prediction, information exchange between system components, and top-down knowledge for machine perception components to guide their search and improve their outputs.

Some particularly relevant applications can be found in the area of partially automated biological and psychological research. Machine perception and subsequent automated reasoning about the observed behavior of humans and animals can substantially improve the efficiency and reliability of experiments in behavioral science. On a related note, these methods can be used in sports analysis to automatically classify game situations from tracking data. This could allow semantic multimedia retrieval for improved training procedures and semantically guided media consumption.

Outline After Section 2, related work, Section 3 explains the applied methods, i.e. the developed annotation process as well as the automatic reasoning methods. The evaluation and experimental results are presented in Section 4. Finally, Section 5 provides a conclusion and ideas for further research.

2. Related work

The surveys [1,35,57,60,65] and the books [5,21,23] provide an overview over the research field around automatic behavior understanding. Single-layered approaches operate directly on sensor data, whereas hierarchical approaches divide the problem into multiple layers: one or more layers of machine perception with one or more layers of reasoning above them. At least three types of hierarchical approaches exist: statistical, syntactic, and description-based approaches.

Statistical, syntactic, or description-based Statistical approaches use probabilistic graphical models such as hidden Markov models or dynamic Bayesian networks to derive situation likelihoods [13,17,33,49,53]. In [66], a two-layered hidden Markov model is used to classify group situations in meetings. The first layer uses audio and video features to model person activities and the second layer combines them to group situations. In [20], group situations are detected using dynamic probabilistic networks.

Syntactic approaches combine atomic events into complex situations using formal (stochastic) grammars, mapping spatiotemporal changes in image sequences to events for instance [3,19,30,32]. Description-based methods employ knowledge about spatial, temporal, and abstract properties, in the form of various logic languages and related modeling techniques [16,25,46,59,63]. After the development of interval temporal logic [2] in which a successful axiomatization of time periods was introduced, it became feasible to model situations and actions in terms of tempo-

ral logic. The hierarchical logic approach for example [50], combines interval temporal logic with event logic to analyze football games. Limited first order logic is used in [61] to combine several sub-events that satisfy predefined temporal and spatial constraints constituting a situation. Statistical and description-based approaches are combined in Markov logic networks by [31,39,47,51,56]. Here, weighted logic rules form the input for Markov logic networks, allowing this approach to handle uncertainty in the input data appropriately. In [10], evidence is aggregated in Bayesian compositional hierarchies. The required rules are automatically generated from ontologies.

Ambient intelligence and smart environments From the community around ambient intelligence and smart environments, we draw inspiration from studies such as [11,24,38,44,52,54,58]. In [6], frequent relationships between actions are discovered from a collection of sensor data and the user is able to fine-tune the system. In [48], a framework is introduced for modeling intelligent environments. It is based on fuzzy transfer learning, allowing the transfer of learned models across different environments. This system predicts temperature development to allow for proactive control. A formal framework based on multi-valued temporal propositional logic is proposed in [37]. Similarly, bigraphs are introduced for the description, design, and analysis of intelligent environments in [27]. Context lattices are introduced in [64], allowing the inclusion of semantic information about the nature and relationships between sensor data and observed activities. The study presented in [45] uses a combination of first-order logic, fuzzy logic, and temporal logic exemplified in a military application. Markov logic networks are applied to the recognition of activities of daily living in [12], without the use of cameras or wearable sensors. In [34], a training-free method is introduced that generates probabilistic inference systems from causal models for human behavior. A goal-directed human activity computing model that captures the semantic relations between different atomic activities is presented in [62]. Studies about crisis management and training can be found in [7,36,55].

Fuzzy metric temporal logic, situation graph trees Our own hierarchical description-based approach to automatic behavior understanding uses fuzzy metric temporal logic (FMTL) combined with situation graph trees (SGTs). These methods were previously applied to the traffic domain in [4,18,43] and to surveillance/human behavior in [8,14,15,22,40–42]. In [9,26], similar methods are applied to robot control. Furthermore, this paper builds upon previous work presented in [29] as well as some earlier efforts presented in [28]. Over the years, the FMTL/SGT methods have been incrementally improved while adhering to their validity demonstrated in [4,18,22,40]. This paper's contribution (see Section 1) mainly consists of new FMTL/SGT models and their quantitative evaluation rather than new reasoning methods. The approach will be explained in some detail under Section 3.2.

Note that the models we use for representation and reasoning are based on expert knowledge rather than learned from training data. Compared to other approaches, expert-knowledge-based representation and reasoning in FMTL and SGTs is intuitive and flexible. The clear boundary between machine perception and reasoning makes it easier to improve one without the other. Furthermore, deductions are understandable by humans and completely provable, and existing FMTL rules and SGTs can be adapted to new settings with relatively little effort. The ability for humans to understand the reasoning process is essential to the presented case study. FMTL/SGT expert systems are suitable for knowledge intensive problems with heterogeneous search spaces such as the one presented here.

3. Methods

In Section 3.1, the three different phases of the annotation process are explained. Section 3.2 presents the reasoning engine, from the applied methods to the specific FMTL rules and SGTs that are used for the evaluation in Section 4.

3.1. Annotation process

The annotation process consists of three parts: recording audiovisual data, annotating the corresponding hypothetical machine perception outputs (data annotations), and annotating the ground-truth for the reasoning engine's evaluation (ground-truth annotations). The data annotations consist of person tracks, orientations, gestures, speech, and pose information as they might be generated by a hypothetical set of machine perception components. The ground-truth annotations are semantic situation descriptions that describe the observed scene.

3.1.1. Audiovisual recordings

The audiovisual data required for the annotation process was gathered at the State Fire Service Institute

Fig. 2. Images recorded at the fire brigade staff exercise and used in the annotation process (containing views from all five cameras). Such images are sampled from the raw video data at 1 *fps*.

North Rhine-Westphalia, during one of their staff exercises for crisis response control room operations (see Fig. 1). Five cameras were used to record the six hour long staff exercise, providing complete and redundant video coverage of the control room. Four microphones distributed across the room were used to provide audio coverage. After the recordings, the video footage was sampled at 1 *fps*, yielding the five-pane images exemplified by Fig. 2. These images, along with the recorded audio tracks, form the input for the data annotation process described below. Computer vision algorithms (especially ones with tracking) could benefit from higher sampling rates, but we use manual annotations. Providing images with 1 *fps* proved to be sufficient for human annotators, and the events we currently aim to recognize do not have fast dynamics. Hence, an image sampling rate of 1 *fps* is sufficient for our current purposes. Higher sampling rates can be obtained with corresponding annotation effort, or without much effort by interpolating the hypothetical machine perception outputs that were already annotated with 1 *fps*.

3.1.2. Data annotations

As input for the reasoning engine, hypothetical machine perception outputs are annotated using a dedicated annotation tool. This annotated data is referred to as hypothetical machine perception because it mimics a hypothetical set of machine perception components that could be installed in the environment to feed the reasoning engine.

From the audiovisual footage, two four minute sections and two ten minute sections were selected for annotation. The sections were selected to include data for each phase observed in the control room workflow and the transitions between them (see Section 1): briefings (consisting of multiple phases) as well as dynamic control room operations. The selected sections were taken from the first 1.5 hours of the exercise, because that part is the most eventful and it contains some unique situations. Currently, we focus on recognizing group behavior during the dynamic phases. For each second within these 28 minutes of audiovisual data, the reasoning engine needs a symbolic representation as input (hypothetical machine perception). To achieve this, a data annotation tool was developed using Python with Qt bindings. Figure 3 shows a screenshot of this tool, displaying a symbolic representation of the audiovisual data recorded during the staff exercise.

The video footage contains all the information we needed to annotate the symbolic representation, except for the participants' speech activity and some useful auditory context information about the operations in the control room, which are provided by the audio footage. The annotator analyzes the images and listens to the audio tracks while manipulating the modeled persons and objects in the bird's eye view of the data annotation tool. Using mouse interaction, each person can be moved and rotated, and their body pose, gesture activity, and speech activity can be set (see Fig. 3). Speech is indicated by a rim around the head, speech-supporting gesticulation (not visible in Fig. 3) by a rim around the right hand. An extended and optionally rotated arm indicates pointing or interaction with displays, notepads, and messages. An extended head indicates looking down and extended legs indicate sitting. Notepads and messages can only be moved around.

Fig. 3. Screenshot of the data annotation tool that was developed for the case study. It is used to annotate hypothetical machine perception outputs for each of the 1 *fps* images exemplified by Fig. 2. This annotated hypothetical machine perception is required as input by the reasoning engine. The annotator clicks various parts of the symbolic persons to manipulate their attributes and he uses the controls at the bottom to record frames, zoom, and navigate the time axis.

Of course, functionality for recording, playing back, zooming, and navigating through the data is included in the user interface, and data files can be saved to be reloaded later.

Using this approach, human annotators achieve between 10 and 20 seconds of data annotations per hour. This is alright for developing reasoning methods using manageable amounts of data, but a more automatic approach is required in the long run. This can range from e.g. wearable sensors to obtain tracking data automatically while manually annotating other attributes, to computer vision and speech detection, i.e. fully automatic unintrusive perception.

The resulting XML data, i.e. the input data for the reasoning engine, consists of symbolic object representations over time. The dynamic objects (with attributes that change over time) are: persons, notepads, and paper messages, and the static objects (with only constant attributes) are: tables, displays, doors, and devices. Their attributes are: name, type, position, horizontal orientation, width, and height (in the horizontal plane), and additionally for persons: gesture activity (two types: one binary for speech supporting gesticulation, one horizontal angle for pointing gestures), speech activity (yes/no), vertical head orientation (up/down), and body pose (sitting/standing). Hence, the data annotations for persons consist of state descriptors of the form [*name:el, type:person, x:774, y:412, w:54, h:20, orientation:144, speech:false, gesture:-18, looking_down:true, sitting:false*], meaning: person el is located at x = 774 cm and y = 412 cm, has width 54 cm and height 20 cm, and an orientation of 144°. He is not speaking, pointing 18° left of his orientation, looking down, and not sitting. The objects and persons in Fig. 3 and similar images throughout this paper are simply visualizations of such state descriptors.

3.1.3. Ground-truth annotations

In addition to input data, we need ground-truth to compare the reasoning output to. This is achieved through manual annotation in a separate Python/Qt software tool developed for the purpose of this case study (Fig. 4). The annotator analyzes the symbolic data (from Section 3.1.2/Fig. 3) in the bird's eye view to determine the correct situation descriptions, i.e. the ground-truth for the reasoning engine's output. The patterns for the situation descriptions that have to be annotated are supplied by the ground-truth annotation tool. Using the interaction panel on the right side of Fig. 4, the annotator has to repeatedly select the appropriate situation pattern as well as the involved objects. The resulting situation description (i.e. ground-truth result) is stored in the list in the top-right corner and the corresponding bounding box is drawn in the bird's eye view. With between 50 and 80 seconds of ground-truth per hour, the ground-truth annotations are less time-consuming than the data annotations (with 10–20 seconds per hour).

This way, the annotator can produce the required ground-truth: a set of situation descriptions for each frame of annotated data. The ground-truth annotation tool also offers functionality for recording, playback, and navigation, and the ground-truth can be saved to be reloaded later. The resulting XML ground-truth is compared to the reasoning outputs during evaluation (Section 4). Without knowledge of the reasoning process' inner workings, the annotator was instructed to annotate the ground-truth according to his own observations and common sense. For this paper, the second four minute section of annotated data was chosen for ground-truth annotation, because it contains dynamic control room operations only.

The following ground-truth situations were annotated, because they represent common group constellations and interaction patterns that occur during dy-

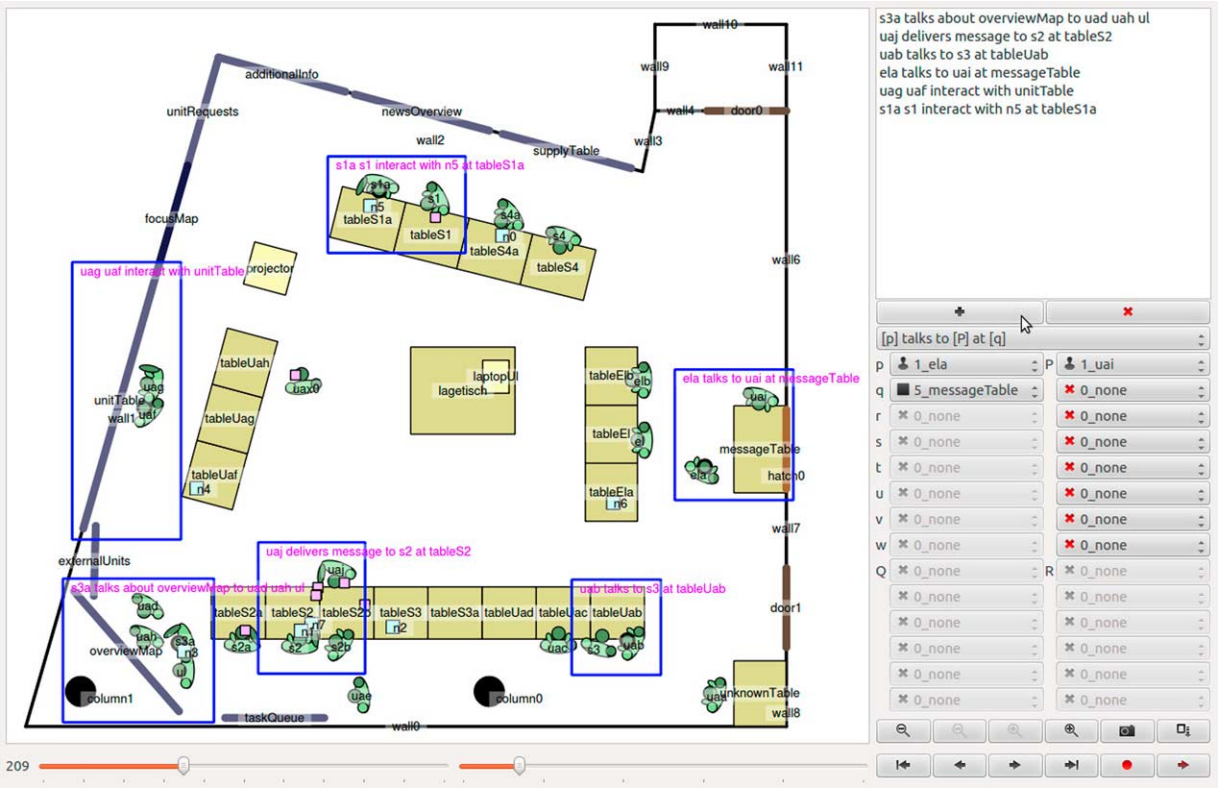


Fig. 4. Screenshot of the ground-truth annotation tool that was developed for the case study. It is used to annotate ground-truth that can be compared to the reasoning engine's output in order to evaluate its performance. The panel on the right is used select the appropriate situation pattern as well as the involved objects, yielding list entries (top right) and labeled bounding boxes for the ground-truth situation descriptions. The bottom of the interface contains functionality for recording, loading, saving, playing back, zooming, and navigating through the data.

namic control room operations. Italics signify variables that should be instantiated with a person or object, and bold-italics signify variables that should be instantiated with a list of persons.

- ***persons*** is-are listening to *person*,
- conversation between ***persons*** in group ***group***,
- silent group ***group***,
- ***persons*** in group ***group*** is-are sitting,
- ***persons*** in group ***group*** is-are standing,
- ***persons*** in group ***group*** is-are moving,
- ***persons*** is-are joining ***group***,
- ***persons*** is-are leaving ***group***,
- constant group ***group***,
- *person* is pointing at *object*,
- *person* is pointing at *object* and ***persons*** is-are looking at it,
- *person* is speaking and pointing at *object*,
- *person* is speaking and pointing at *object* and ***persons*** is-are looking at it.

3.2. Reasoning

Now that the methods for annotating data and ground-truth have been established, we turn to the methods for automatic reasoning. After introducing the applied methods, we present the part of our knowledge base that is the subject of the evaluation in Section 4: the situation graph trees (SGTs) as well as the fuzzy metric temporal logic (FMTL) rules they deploy.

The XML data that were annotated using the data annotation tool described in Section 3.1.2 are fed into a reasoning engine based on FMTL and SGTs, promising and widely applicable tools for automatic behavior understanding and related applications (see Section 1 for some examples) [4,14,29,42]. For implementation, we use F-Limette, an FMTL reasoning engine (similar to Prolog) written in C, and the SGT-Editor: a Java application for editing and traversing SGTs.

The SGTs constitute the top part of the reasoning process. FMTL rules are called from their nodes to perform the bottom part of the reasoning process.

FMTL rules are largely domain independent and typically about spatiotemporal relations, whereas SGTs are more domain specific as they usually constitute abstract relations between the FMTL rules they deploy. Once an FMTL rule base has been established, it stays relatively fixed and it can be used by different SGTs within the same application or even across different application domains. Over the years, the FMTL/SGT methods were incrementally improved while adhering to their validity demonstrated in [4,18,22,40].

3.2.1. Applied methods – Fuzzy metric temporal logic

The FMTL language is a first order logic extended with fuzzy evaluation and temporal modality. The language uses fuzzy instead of binary truth values, enabling the modeling of inherently vague concepts. Among other things, fuzzy evaluation can be used to model concepts such as "a person moving fast" or "two objects being close to each other". Such concepts should have truth values between 0.0 and 1.0 (rather than simply true or false), depending on the numeric speed of the person, the numeric distance between the objects, or some other value. Furthermore, fuzzy evaluation can be used to model uncertainty in the input data. How these methods can handle uncertainty in addition to vagueness was addressed on a theoretical level in [29], but for our current experiments we focus on modeling inherently vague concepts, without uncertainty in the input data.

The second extension, temporal modality, enables the modeling of developments along the time axis instead of just at a single point in time. Rule conditions can be grounded in past, current, and future states of the world. Furthermore, FMTL possesses a metric on time, allowing expressions about exact time differences in addition to categorical concepts such as "before" and "after". Temporal modality is essential for modeling the speed of an object for example, and for modeling situations consisting of multiple phases. It can also achieve a smoothing effect against noise, outliers, and brief changes that should be ignored.

3.2.2. Applied methods – Situation graph trees

Figure 5 shows an abstract SGT example that recognizes moving and stationary groups as well as meetings at tables and interaction with displays. An SGT is a hypergraph that consists of situation graphs, represented by transparent rounded rectangles. Additionally, each situation graph contains one or more situation schemes, represented by opaque rectangles. Each situation scheme possesses a name (top segments of opaque rectangles in Fig. 5), one or more conditions

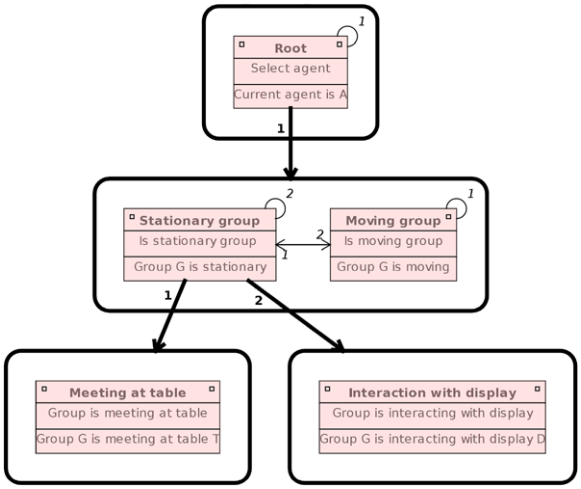


Fig. 5. An abstract example of a situation graph tree (SGT) that recognizes moving and stationary groups as well as meetings at tables and interaction with displays. During SGT traversal, the conditions (the middle segments of the rectangles) initiate deductions in FMTL (i.e. Prolog-like reasoning processes in F-Limette). The actions (bottom segments) can either be the generation of situation descriptions or actuator commands.

(middle segments), and zero or more actions (bottom segments). Typically, there is one SGT traversal per frame per observed person (or vehicle for example). Each traversal normally starts by selecting an agent for that traversal (in the *Root* situation scheme). The selected object becomes the center of the reasoning process during that SGT traversal. During each traversal, all possible paths between the root and the leafs of an SGT are traversed. If a condition along a path cannot be fulfilled, that path is done and the next one is started. The truth values of the outputs along each path provide feedback about which conditions cause the applicability of each situation to decrease. If traversal does not reach a leaf, but still fulfills part of the path to that leaf, the outputs on the fulfilled part are generated, but the ones below the point of failure are not, providing feedback about the cause of failure. More details about the traversal algorithm can be found in [40].

The situation schemes' conditions initiate deductions in FMTL (i.e. Prolog-like reasoning processes in F-Limette). Each condition returns a truth value between 0.0 and 1.0, depending on the rules that were directly or indirectly evaluated, and ultimately on the atomic facts from the input data. The next condition (either within the same situation scheme or in the next, conceptually refined situation scheme) uses the truth value returned by the previous one as base truth value. This means that the situations deduced along each path

of an SGT are always ordered from generic to specific, as their truth values cannot increase along the way. As soon as a condition returns the truth value 0.0, reasoning stops and the next path is started from the root situation scheme.

The situation schemes' actions generate the final outputs of the system: situation descriptions, or in embodied settings also actuator commands. The truth value returned by the last condition above each action is also associated with the action, representing vagueness, uncertainty, or both. Actions below unfulfilled conditions (returning the truth value 0.0) are not executed, and, as we will see in Section 4, one can ignore outputs with a truth value smaller than a certain threshold.

Trying to deduce a more specific situation after a more generic one is called conceptual refinement. This is demonstrated by the situation schemes *Stationary group*, *Meeting at table*, and *Interaction with display* in Fig. 5, connected through the corresponding bold numbered edges. To model temporal dynamics and situations consisting of multiple phases, situation schemes can be connected through temporal edges (thin numbered edges) as demonstrated by the situation schemes *Stationary group* and *Moving group* in Fig. 5. The numbers along these bold and thin edges do not have any special meaning for the presented experiments. The squares in the upper left and upper right corners of situation schemes indicate start situations and end situations in temporal chains, and the circles on their upper right corners represent reflexive temporal edges. For our current experiments, we focus on temporal modality implemented at rule level as opposed to using temporal SGT edges. Temporal modality at rule level was introduced in Section 3.2.1 and it will be demonstrated in Section 3.2.3.

3.2.3. *Knowledge base – Situation graph trees and fuzzy metric temporal logic rules*

Compared to [29], the SGTs and FMTL rules presented in this paper are considerably more advanced. We have developed a detailed taxonomic model of spatiotemporal concepts, interaction patterns, and group constellations, part of which is used and explained in this paper. Besides its use for the current case study and evaluation, this model (including the rules and trees not presented in this paper) is relevant to the field of high-level reasoning in general, and to many different application domains, such as the ones presented at the end of Section 1.

After analyzing the data, axioms for group models were formulated on a natural language level and then formalized into SGTs and FMTL rules using common sense knowledge and empirical trials. Which numeric distances, angles, and speeds correspond to which semantic concepts was also determined through common sense and empirical trials, and by comparing the data to other real-world examples (e.g. from surveillance data and average pedestrian speeds).

The four SGTs that are evaluated in Section 4 were developed to recognize common group constellations and interaction patterns that occur during dynamic control room operations. Following from the data analysis and knowledge formalization process these SGTs model groups in terms of the proximity of their members and the similarity between their orientations, their orientations toward each other, or orientations toward a mutual object or person. Two of the SGTs use this group model as their base, but they have different conceptual refinements. One of them models speech behavior across time within recognized groups and distinguishing between silent groups, monologues, and discussions. The other describes the sitting, standing, and moving members within groups. These refinements are again based on the observed data and on common sense knowledge. The third SGT uses the same group model as its base, but it extends it across time in order to recognize staff members that are joining and leaving groups. Following from the observed data and common sense knowledge, the fourth SGT applies a different strategy. It detects staff members that are referring to objects through pointing gestures and speech. Then, it selects the surrounding staff members that are oriented at the object or person that is being referred to.

Figures 6 through 9 display these SGTs, rendered by the SGT-Editor that also performed the experiments. All FMTL rules that are directly or indirectly used by the SGTs are listed and explained in Appendix A (interface specification) and Appendix B (rule specifications in alphabetical order). Furthermore, the rules in Appendix B that are considered non-trivial and not too verbose are included in Appendix C (Formulas 1 through 25) and Appendix D (graphs of trapezoid truth functions). These formulas are the logic notation equivalent of the actual source code, and the graphs are simply visualizations of further source code. Every FMTL rule that is directly or indirectly used by the presented SGTs also has a brief explanation in Appendix A or B. And the ones that have their formulas

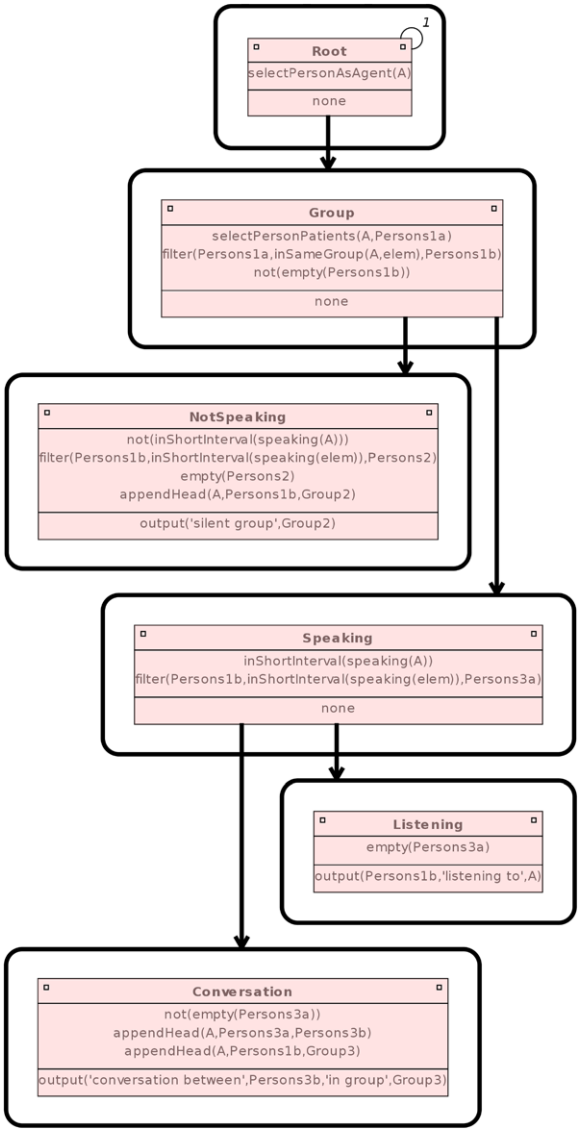


Fig. 6. An SGT that recognizes groups and their speaking and listening members.

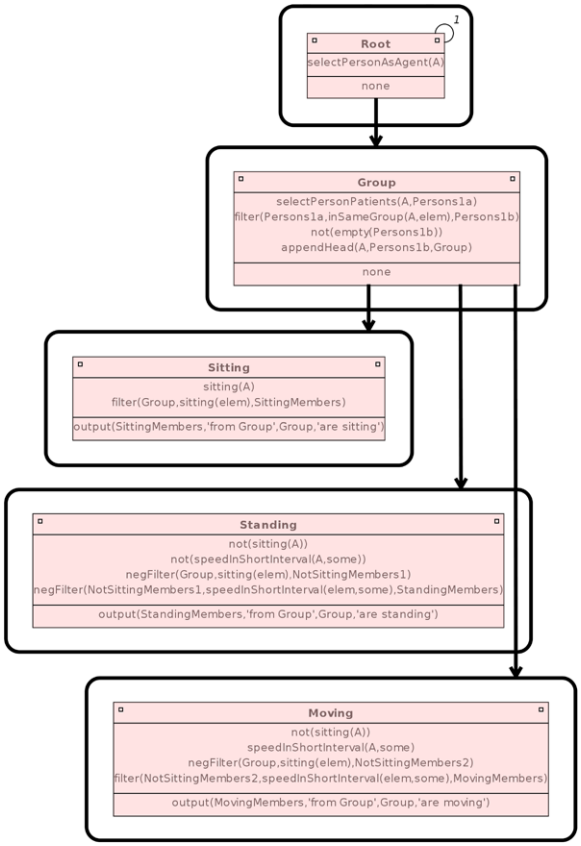


Fig. 7. An SGT that recognizes groups and their sitting, standing, and moving members.

listed in Appendix C are referenced accordingly from Appendix B and the text below.

Appendices A, B, and C use the following conventions. List variables are boldfaced, nested predicate variables are capitalized, and constant arguments are non-italic to distinguish them from variables, allowing us to omit quantifiers. Terms of the form $\Diamond_{dt}\phi$ mean: ϕ at time $t = t_{current} + dt$. Terms of the form $\Box^{a\%}_{dt_1,dt_2}\phi$ mean: ϕ in at least $a\%$ of time interval $[t_{current} + dt_1, t_{current} + dt_2]$. The symbol ! is the cut operator, preventing a rule from querying its remaining conditions. For practical reasons, the SGTs in Figs 6 through 9 use a different capitalization convention than Appendices A, B, and C.

Groups with speaking and listening members All four SGTs have the same *Root* node, selecting a person as the agent for the current traversal. The first SGT (Fig. 6) recognizes groups and their speaking and listening members. Its *Group* node selects all persons other than the agent as possible patients and filters them to only the ones that can be considered part of the same group as the agent, using *filter(Persons1a, inSameGroup(A, elem), Persons1b)* (Appendix C, Formula 5). Then, the node *NotSpeaking* checks whether neither the agent nor any of the other persons in the group are speaking during a 5 *s* interval, using *inShortInterval(speaking(P))* (Appendix C, Formula 25). If nobody is speaking (*not(inShortInterval(speaking(A)))* and *empty(Persons2)*), an output like "silent group *a p1 p2 p3*" is generated, where *a* is the agent for the current traversal and *p1, p2, p3* are three other persons that fulfill the described conditions. In the other branch,

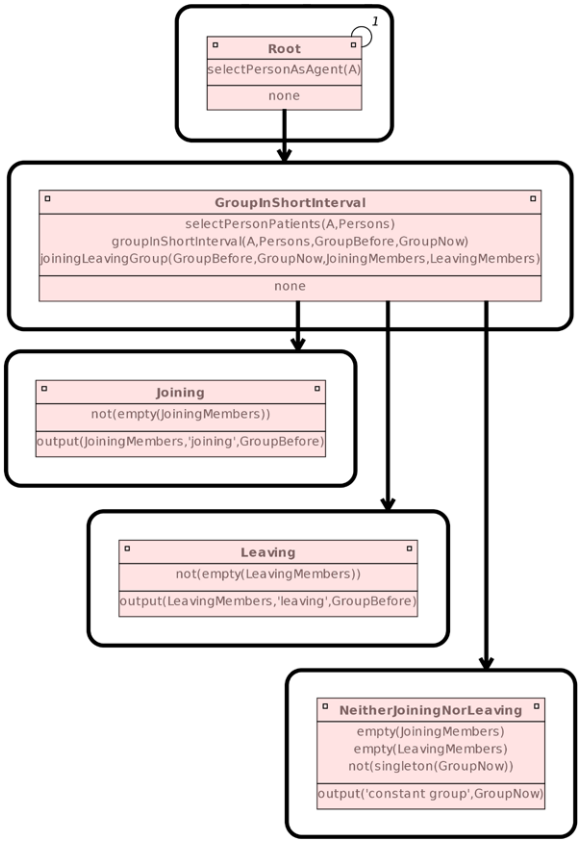


Fig. 8. An SGT that recognizes groups and their joining and leaving members.

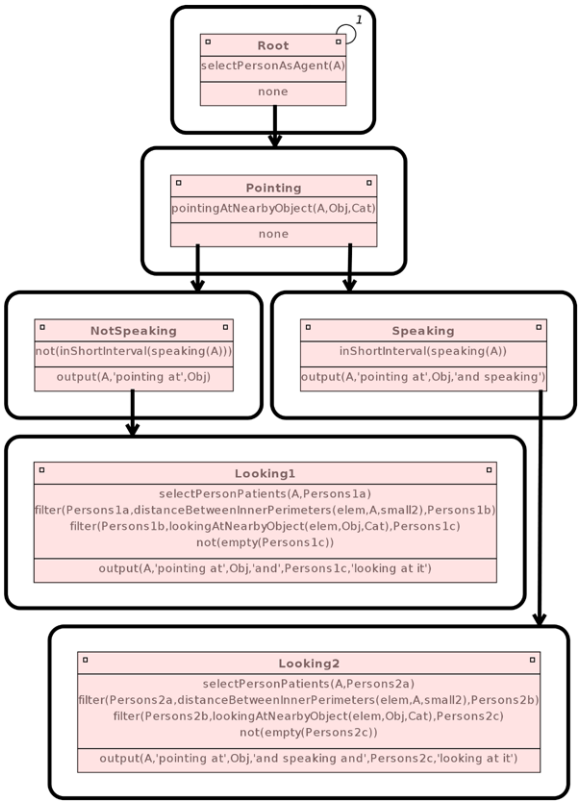


Fig. 9. An SGT that recognizes groups that are oriented at an object that is being referred to.

the node *Speaking* checks to what degree the agent and the other persons in the group are speaking during the same 5 *s* interval. Then, if the agent is speaking, the node *Conversation* checks whether at least one of the other persons in the group is also speaking: *filter(Persons1b, inShortInterval(speaking(elem)), Persons3a)* and *not(empty(Persons3a))*. If so, the SGT outputs something of the form "conversation between *a p1* in group *a p1 p2 p3*". If none of the other persons in the group are speaking (*filter(Persons1b, inShortInterval(speaking(elem)), Persons3a)* followed by *empty(Persons3a)*), the node *Listening* outputs something like "*p1 p2 p3* listening to *a*".

Groups with sitting, standing, and moving members The second SGT (Fig. 7) recognizes groups and their sitting, standing, and moving members. Its *Root* and *Group* node are the same as in Fig. 6, except for the additional condition *appendHead(A, Persons1b, Group)*. Then, the SGT splits into three branches. In the node *Sitting*, the agent's sitting property should hold, and the persons in the group are filtered to only those that are sitting. The corresponding output is of the form "*a p1* from group *a p1 p2 p3* are sitting" (*a* is the current agent, *p1, p2, p3* are other persons fulfilling the appropriate conditions). In the node *Standing*, the agent's sitting property should not hold, neither should he be moving (see Appendix C, Formula 15). The group is then filtered to only the members that are not sitting and not moving, i.e. that are standing. The corresponding output is of the form "*a p1* from group *a p1 p2 p3* are standing". Finally, the node *Moving* uses a similar method to generate output of the form "*a p1* from group *a p1 p2 p3* are moving".

Groups with joining and leaving members The third SGT (Fig. 8) starts with the usual *Root* node and a node called *GroupInShortInterval*. It uses *groupInShortInterval(...)* (Appendix C, Formula 3) to detect the group around the agent during the previous and current frame and it uses *joiningLeavingGroup(...)* (Appendix C, Formula 6) to determine which members join and leave the group during this time. If the list of joining members is not empty, the node *Joining* outputs something like "*p1* joining *a p2 p3*". The

node called *Leaving* does the same for leaving members, e.g. "*p1* leaving *a p2 p3*". Finally, in *NeitherJoiningNorLeaving*, output of the form "constant group *a p1 p2 p3*" is generated if *empty(JoiningMembers)* and *empty(LeavingMembers)*.

Groups oriented at referred object The fourth and last SGT (Fig. 9) recognizes groups that are gathered around an object that is being referred to. After the *Root* node, it uses *pointingAtNearbyObject(...)* (Appendix C, Formula 2) in its *Pointing* node to determine whether the agent is pointing at a nearby object. Then, the nodes *NotSpeaking* and *Speaking* use *inShortInterval(...)* (Appendix C, Formula 25) to determine whether the agent is speaking in a 5 *s* interval, generating the corresponding output "*a* pointing at *obj*" or "*a* pointing at *obj* and speaking". Note that one could also combine the conditions for pointing and speaking into one rule (using the agent's orientation toward the object in the absence of pointing), so that pointing would not be a necessary condition. Alternatively, two independent SGTs could be used to recognize groups gathered around an object that is being pointed at and groups gathered around an object that is being talked about. Such SGTs would operate separately, but their outputs would be gathered and potentially merged. The nodes *Looking1* and *Looking2* are identical, except for their outputs. They use *distBetweenInnerPerimeters(...)* (Appendix C, Formula 12) to find the persons close to the agent and *lookingAtNearbyObject(...)* (Appendix C, Formula 1) to filter them to only the ones that are looking at the object that the agent is pointing at. Their outputs are of the form "*a* pointing at *obj* and *p1 p2 p3* looking at it" and "*a* pointing at *obj* and speaking and *p1 p2 p3* looking at it" respectively.

4. Evaluation

The system was evaluated in four experiments, where the goal was to have the situation graph trees (SGTs) produce the same situation descriptions as a human annotator. Input data for these experiments was annotated using the method described in Section 3.1.2. These input data are fed into each of the four SGTs displayed in Figs 6 through 9, and the results are compared to appropriate ground-truth that was annotated using the method described in Section 3.1.3.

From the 28 minutes of available input data, 4 minutes were selected for ground-truth annotation. A motivation for selecting these data and ground-truth is given in Sections 3.1.2 and 3.1.3. Each of the four experiments described below uses more than 1000 ground-truth results. On average, each experiment has six ground-truth results per second. All situation descriptions generated by the system were compared to the ones produced by a human annotator in a performance evaluation, counting false negatives, false positives, and true positives in order to calculate precision, recall, and F-score (harmonic mean of precision and recall). This was repeated for truth value thresholds between 0.1 and 1.0 (with step size 0.1). Reasoning results with truth values smaller than the applied truth value threshold are rejected, so increasing the truth value threshold leads to higher precision but lower recall.

Note that the presented problem is not a classification between a few classes. Instead, the situation template (e.g. "conversation between ***persons*** in group ***group***") must be classified correctly, *and* all involved persons and objects (e.g. instantiating the list variables ***persons*** and ***group***) must be consistent with the annotated ground-truth. After presenting the conducted experiments in Section 4.1, errors are discussed in Section 4.2 and runtimes in Section 4.3.

To introduce the evaluation procedure, Appendix E shows an example frame with ground-truth annotations (top) as well as corresponding reasoning results and truth values (bottom), rendered by the tool described in Section 3.1.3. The displayed frame is from the first experiment described below. It contains the situation descriptions "***persons*** is-are listening to *person*", "conversation between ***persons*** in group ***group***", and "silent group ***group***".

4.1. Experiments

Groups with speaking and listening members In this experiment, groups are classified with respect to which members are speaking, distinguishing between groups listening to a single speaker, conversations, and silent groups (SGT in Fig. 6). This SGT generates situation descriptions of the form "***group*** listening to *person*", "conversation between ***persons*** in group ***group***", and"silent group ***group***". Example ground-truth (top) and the corresponding reasoning result (bottom) are visualized in Fig. 10. *Similarly, Figs 13, 16, and 19 provide examples for the other three experiments.*

In this example, three persons are close together with two of them oriented at the third one (s3), sat-

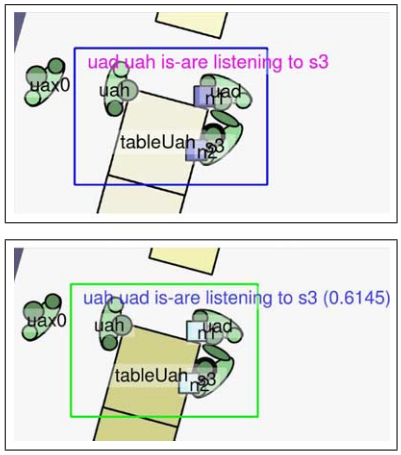


Fig. 10. Example ground-truth (top) and corresponding reasoning results with their truth values (bottom) from the experiment "groups with speaking and listening members".

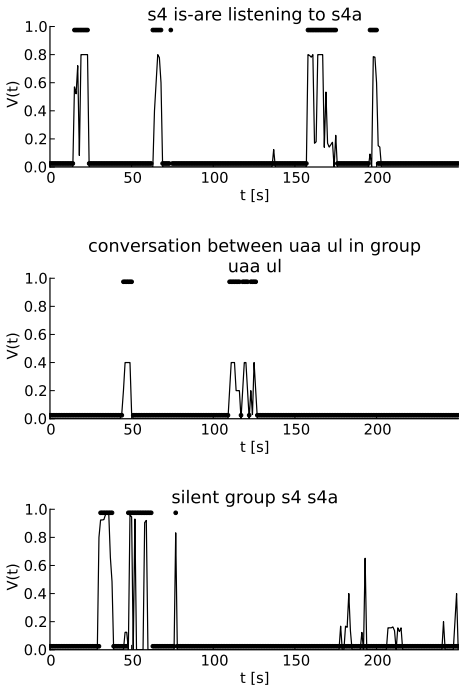


Fig. 11. Truth values V over time for some example ground-truth and corresponding reasoning results, from the experiment "groups with speaking and listening members". The bold line shows the annotated ground-truth and the thin line shows the reasoning results.

isfying the conditions for *InSameGroup(p, q)* (Appendix C, Formula 5). Because only s3 is speaking between $t = -2$ and $t = 2$ (*InShortInterval(Speaking(p))*, Appendix C, Formula 25), the output "uad uah is-are listening to s3" is generated. Further examples from this experiment can be found in Appendix E.

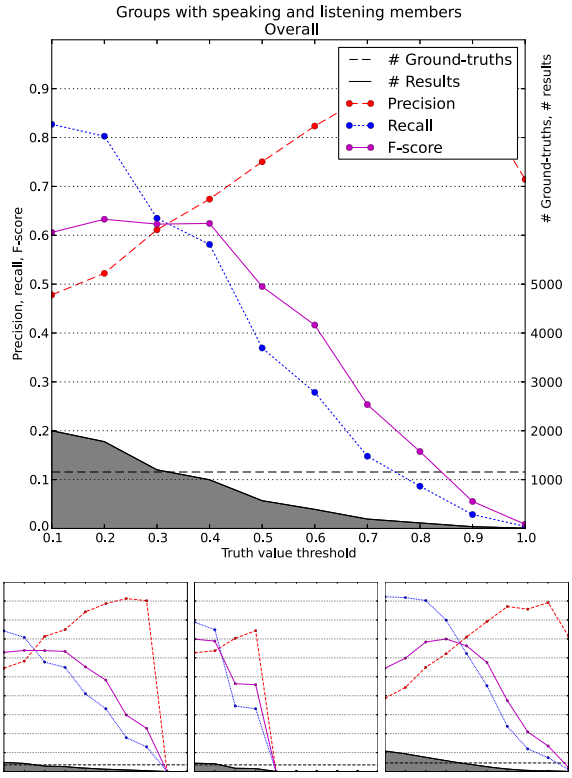


Fig. 12. System performance for the experiment "groups with speaking and listening members" (SGT in Fig. 6): precision, recall, and F-score over varying truth value thresholds, and the number of ground-truth results and reasoning results involved (the latter depending on the truth value threshold). The bottom graphs, from left to right, show the separated performance for each type of situation description generated by the SGT: "***persons*** is-are listening to *person*", "conversation between ***persons*** in group ***group***", and "silent group ***group***".

The following also applies to Figs 14, 17, and 20. Each of the three situation descriptions generated by this SGT is further exemplified by the graphs in Fig. 11, where the truth values of specific situations are plotted over time. As the positions, orientations, and amounts of speech of the involved persons vary, the truth values of the recognized situations in Fig. 11 vary with them. The goal is to maximize the correspondence between the bold line (annotated ground-truth) and the thin line (reasoning results).

The following also applies to Figs 15, 18, and 21. The system performance for this experiment is displayed in Fig. 12. Precision, recall, and F-score are plotted as a function of the applied truth value threshold (below which results are rejected, higher truth

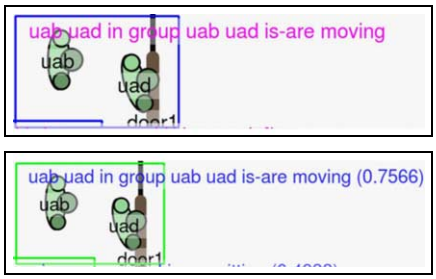


Fig. 13. Example ground-truth (top) and corresponding reasoning results with their truth values (bottom) from the experiment "groups with sitting, standing, and moving members".

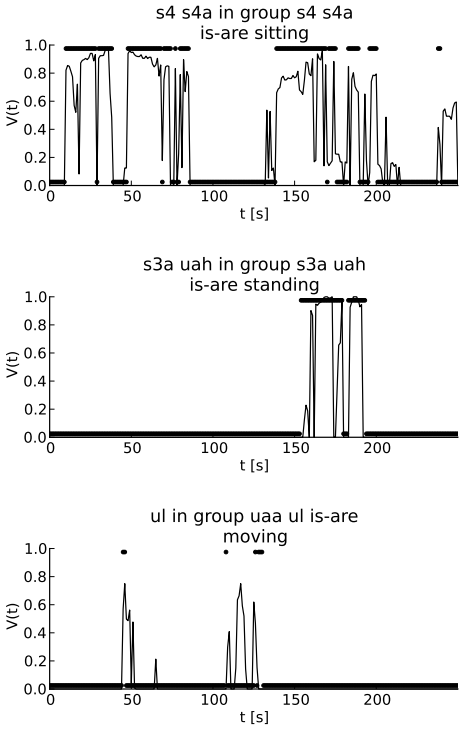


Fig. 14. Truth values V over time for some example ground-truth and corresponding reasoning results from the experiment "groups with sitting, standing, and moving members". The bold line shows the annotated ground-truth and the thin line shows the reasoning results.

value thresholds leading to higher precision but lower recall). The bottom graphs show the separated performance for each type of situation description generated by the SGT, in this case: "***persons*** is-are listening to *person*", "conversation between ***persons*** in group ***group***", and "silent group ***group***" (from left to right).

Groups with sitting, standing, and moving members Here, groups are assorted with respect to which members are sitting, standing, and moving (SGT in Fig. 7).

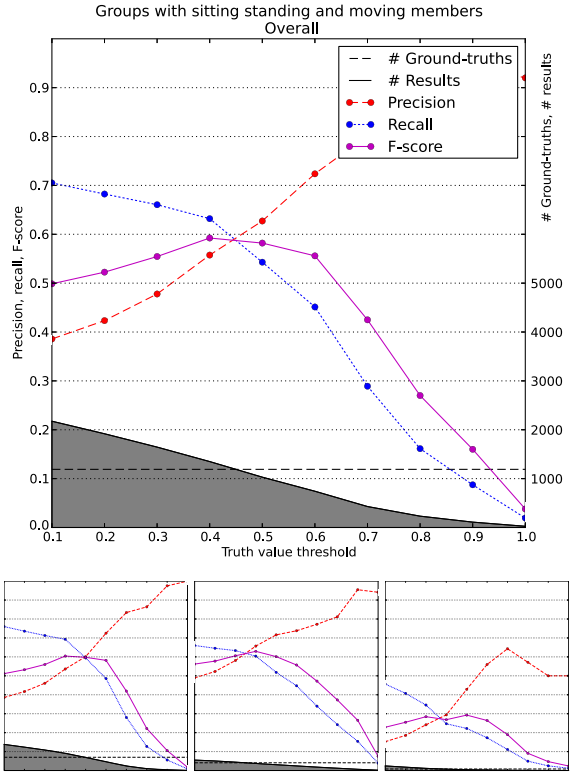


Fig. 15. System performance for the experiment "groups with sitting, standing, and moving members" (SGT in Fig. 7): precision, recall, and F-score over varying truth value thresholds, and the number of ground-truth results and reasoning results involved (the latter depending on the truth value threshold). The bottom graphs, from left to right, show the separated performance for each type of situation description generated by the SGT: "***persons*** in group ***group*** is-are sitting", "***persons*** in group ***group*** is-are standing", and "***persons*** in group ***group*** is-are moving".

This SGT generates situation descriptions of the form "***persons*** in group ***group*** is-are sitting", "***persons*** in group ***group*** is-are standing", and "***persons*** in group ***group*** is-are moving". Some examples are visualized in Fig. 13 and plotted over time in Fig. 14. In Fig. 13, uab and uad belong to the same group, because they are close together and they have appropriate orientations; *InSameGroup(p, q)*, (Appendix C, Formula 5). Their positions accross three frames change enough for the system to output "uab uad in group uab uad is-are moving"; *SpeedInShortInterval(p, c)* (Appendix C, Formula 15) The system performance for this experiment is displayed in Fig. 15.

Groups with joining and leaving members This experiment deals with groups and persons that are joining and leaving them (SGT in Fig. 8). The situation descriptions are: "***persons*** is-are joining ***group***",

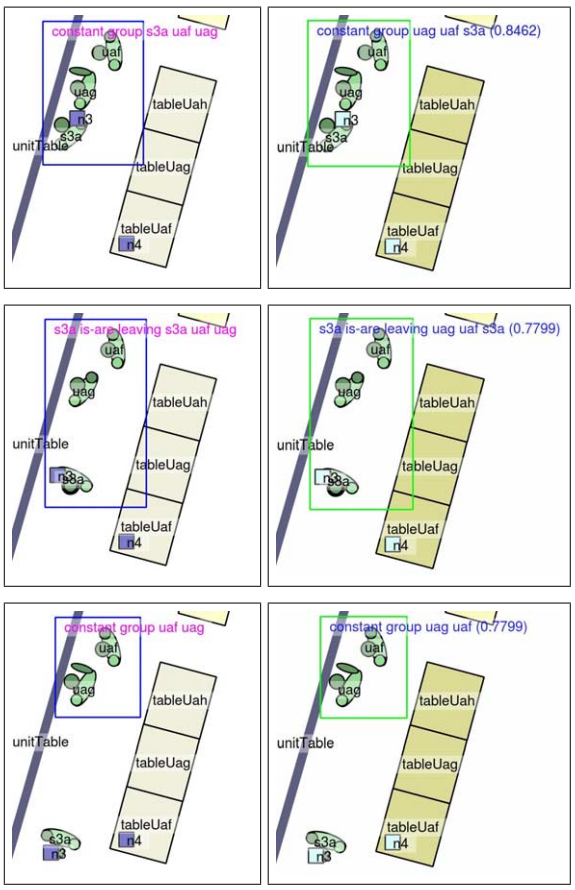


Fig. 16. Example ground-truth (left) and corresponding reasoning results with their truth values (right) from the experiment "groups with joining and leaving members", with three consecutive frames from top to bottom.

"***persons*** is-are leaving ***group***", and "constant group ***group***". Example screenshots and truth value over time plots are provided in Figs 16 and 17. The top row in Fig. 16 shows a group of three persons. In the second row, the next frame, s3a is not part of the group anymore, due to his changed position and orientation. This is achieved using *GroupInShortInterval(p,* ***q****,* ***r****,* ***s****)* and *JoiningLeavingGroup(****p****,* ***q****,* ***r****,* ***s****)*, (Appendix C, Formulas 3 and 6), yielding the output "s3a is-are leaving s3a uaf uag". In the next frame (bottom row), the system outputs "constant group uaf uag", because s3a has left. System performance is displayed in Fig. 18.

Groups oriented at referred object The last experiment detects groups in which one of the members is referring to an object, and others are oriented at it (SGT in Fig. 9). The situation descriptions generated by this

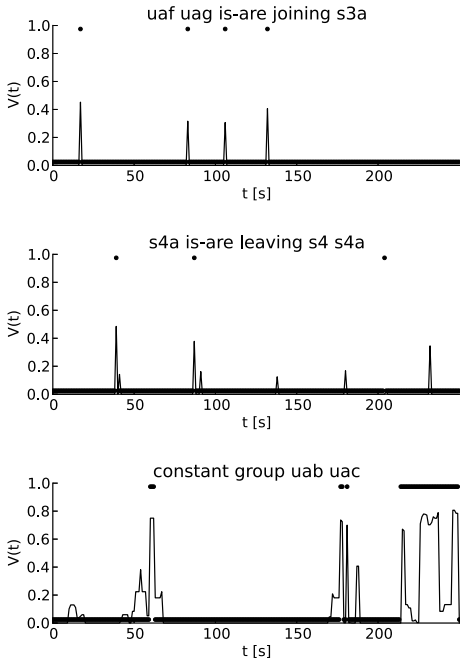


Fig. 17. Truth values *V* over time for some example ground-truth and corresponding reasoning results from the experiment "groups with joining and leaving members". The bold line shows the annotated ground-truth and the thin line shows the reasoning results.

SGT are: "*person* is pointing at *object*", "*person* is pointing at *object* and ***persons*** is-are looking at it", "*person* is speaking and pointing at *object*", and "*person* is speaking and pointing at *object* and ***persons*** is-are looking at it". The corresponding example screenshots, truth value over time plots, and system performance are provided in Figs 19, 20, and 21. For Fig. 19, the reasoning engine checks whether the current agent (uag) is pointing at an object (unitTable); *PointingAtNearbyObject(p, q, c)*, Appendix C, Formula 2. Because s3a is in the same group as uag (*InSameGroup(p, q)*, Appendix C, Formula 5), the output "uag is speaking and pointing at unitTable and s3a is-are looking at it" is generated.

4.2. Error analysis

Performance graphs The following is an overview of the most notable facts deduced from the performance graphs in Figs 12, 15, 18, and 21.

- Groups with speaking and listening members (Fig. 12):
  - ∗ best overall performance (F-score $>$ 0.6) for truth value thresholds 0.2, 0.3, and 0.4,

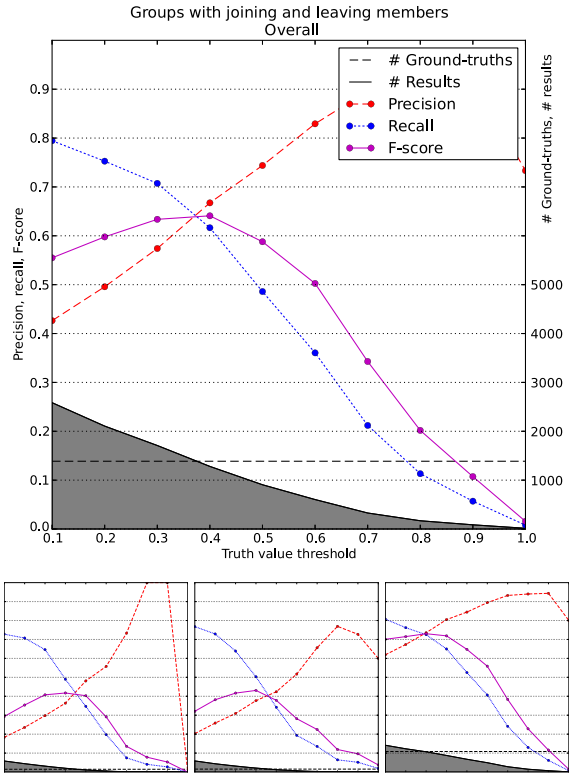


Fig. 18. System performance for the experiment "groups with joining and leaving members" (SGT in Fig. 8): precision, recall, and F-score over varying truth value thresholds, and the number of ground-truth results and reasoning results involved (the latter depending on the truth value threshold). The bottom graphs, from left to right, show the separated performance for each type of situation description generated by the SGT: "***persons*** is-are joining ***group***", "***persons*** is-are leaving ***group***", and "constant group ***group***".

- ∗ overall number of results (gray area in top graph) in good proportion to number of ground-truth situations (dashed horizontal line),
- ∗ "***persons*** is-are listening to *person*" (bottom left): number of results (gray area) small compared to number of ground-truths (dashed horizontal line), causing low recall,
- ∗ "conversation between ***persons*** in group ***group***" (bottom center): only low truth value results (proportional to amount of speech in group in 3*s* interval, causing a preference for low truth value thresholds), too little reasoning results compared to ground-truth (causing low recall).

– Groups with sitting, standing, and moving members (Fig. 15):

- ∗ best overall performance (F-score ≈ 0.6) for truth value thresholds 0.4 and 0.5,
- ∗ overall number of results in good proportion to number of ground-truths,
- ∗ "***persons*** in group ***group*** is-are moving" (bottom right): relatively poor performance, small number of situations present.

– Groups with joining and leaving members (Fig. 18):

- ∗ best overall performance (F-score > 0.6) for truth value thresholds 0.3 and 0.4,
- ∗ overall number of results in good proportion to number of ground-truths,
- ∗ "constant group ***group***" (bottom right): relatively good performance and large number of situations present.

– Groups oriented at referred object (Fig. 21):

- ∗ best overall performance (F-score ≈ 0.6) for truth value thresholds 0.7 and 0.8,
- ∗ overall nr. of results large compared to number of ground-truths, causing low precision,

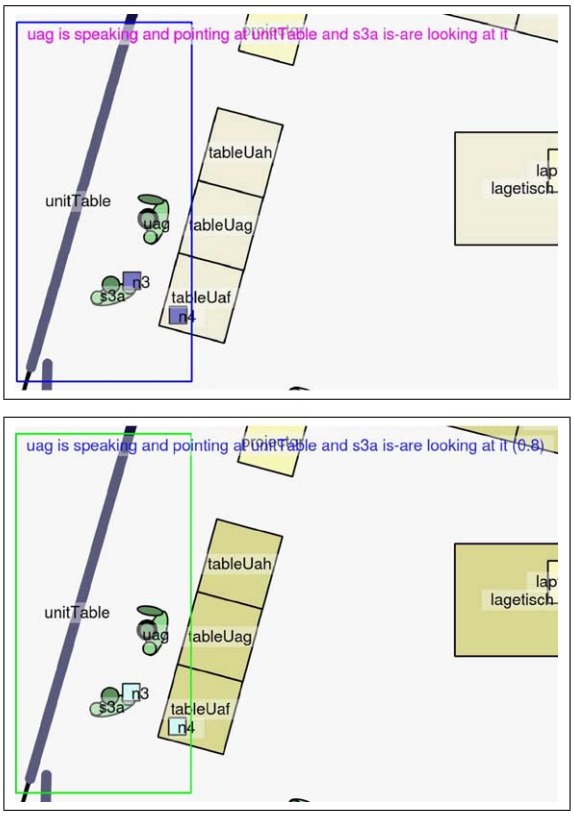


Fig. 19. Example ground-truth (top) and corresponding reasoning results with their truth values (bottom) from the experiment "groups oriented at referred object".

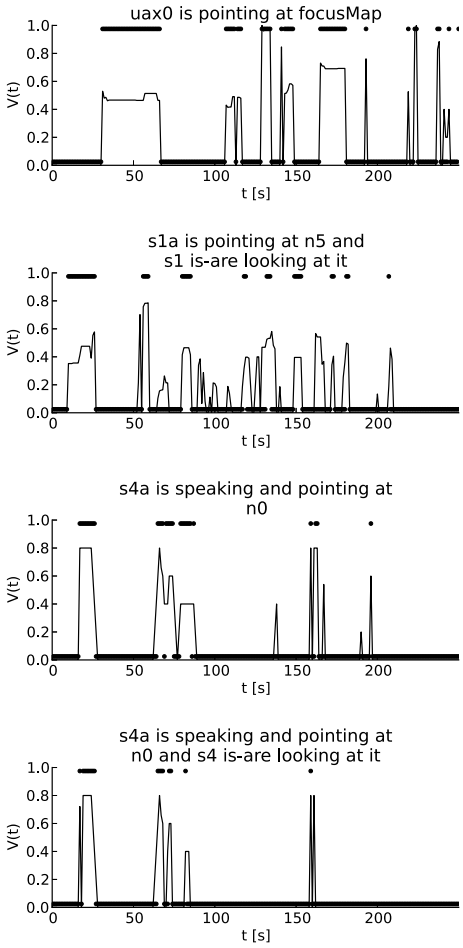


Fig. 20. Truth values V over time for some example ground-truth and corresponding reasoning results from the experiment "groups oriented at referred object". The bold line shows the annotated ground-truth and the thin line shows the reasoning results.

- best performance and most situations present for "*person* is pointing at *object*".

Possible optimizations One of the conditions for the recognition of a group is the proximity of its members. False negatives occur if this condition is not met. Sometimes, one of the group members is standing at a distance from the rest of the group. Human annotators can usually deduce from the context that he is still part of the group, whereas our current knowledge base applies the proximity between group members as a necessary condition. Optimizing the parameters of the trapezoid truth functions in Appendix D could solve some of these cases, effectively increasing the system's separating power by changing the proximity conditions. Other cases cannot be solved like this. They would require more complex models instead.

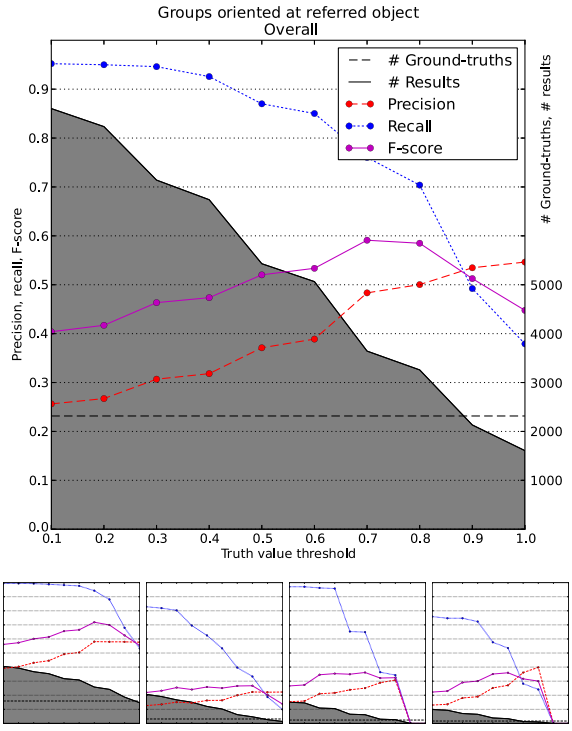


Fig. 21. System performance for the experiment "groups oriented at referred object" (SGT in Fig. 9): precision, recall, and F-score over varying truth value thresholds, and the number of ground-truth results and reasoning results involved (the latter depending on the truth value threshold). The bottom graphs, from left to right, show the separated performance for each type of situation description generated by the SGT: "*person* is pointing at *object*", "*person* is pointing at *object* and ***persons*** is-are looking at it", "*person* is speaking and pointing at *object*", and "*person* is speaking and pointing at *object* and ***persons*** is-are looking at it".

Some false negatives are caused by correct results with low truth values. In such cases, correct results are ignored because of the applied truth value threshold. An example thereof with a truth value of 0.2 is visible in the top-right of the images in Appendix E. Detecting this result using a truth value threshold $\leqslant 0.2$ is not feasible, because it would lead to many false positives elsewhere in the data. The opposite effect also occurs: wrong results with truth values higher than the applied truth value threshold. Wrong results with low truth values on the other hand, can be filtered out using an appropriate truth value threshold.

Further errors are caused by either too many or too few group members. More specifically, annotators tend to select the largest possible groups, whereas the reasoning engine sometimes prefers a subgroup thereof, due to failing conditions for some of the persons involved. Such errors might be solved by ap-

plying more subtle (fuzzy and temporal) conditions, or by optimizing the trapezoid truth functions in Appendix D.

In other cases, two groups are interpreted as one because of their close proximity and harmonic orientations. This could again be partially solved by optimizing the trapezoids, and by modeling more subtle fuzzy temporal conditions, paying more attention to the members' attributes in neighboring frames. Furthermore, single groups are sometimes interpreted as two groups when the members are not close enough together. The applied proximity condition could be combined with fuzzy temporal interaction conditions to cover such cases. There are also cases where persons are recognized as members of two groups simultaneously. Annotators seem to apply a constraint against this that our knowledge base does not yet have. Finally, a staff member passing through or passing by a group is interpreted as joining, belonging to, and leaving the group, leading to further errors. Additional conditions should check how long the staff member stays and whether he is interacting with the group.

To achieve a more powerful knowledge base with more fine-grained control, more FMTL rules should be wrapped in rules like *InShortInterval(C)* (Appendix C, Formula 25), combining the power of temporal modality and fuzzy evaluation. Furthermore, the slopes of the trapezoid truth functions could be widened to facilitate fuzziness, and the rule *RayHitsObject(p, o, q)* in particular (Appendix B) could benefit from more fuzzy conditions. For some temporal modeling (e.g. of speed), a higher temporal resolution than 1 *fps* would be beneficial, which can be obtained or at least approximated by interpolating the current data annotations.

Performance criteria One could reconsider the applied performance criteria. For some applications, a hard truth value threshold would not be necessary. One could instead report and visualize all results along with their truth values, and have a human operator decide which results are the most interesting. Under such criteria (truth value threshold $\leqslant 0.01$) one would require high recall with reasonable precision. Figures 12, 15, 18, and 21 show that this can be achieved.

One can also perform the presented evaluations while allowing for small temporal offsets to show that some correct situations are detected too early or too late, but within a few seconds from the annotated ground-truth. For some applications, these results would still be valuable. Similarly, the evaluations can be performed while allowing for partial group member match. This means that the list variables in the results and ground-truth only have to match by a fraction between 0.0 and 1.0. Results with partial group member match would still be valuable for some applications. Performance increases if such concessions are made.

Table 1

Runtime on a desktop computer with a six core Intel Xeon W3690 CPU: total runtime for each experiment (with 240 *s* of 1 *fps* input data) and average runtime per frame of input data

Experiment	Runtime in *s*	Runtime in *s/frame*
Groups with speaking and listening members	1131	4.7
Groups with sitting, standing, and moving members	959	4.0
Groups with joining and leaving members	2602	10.8
Groups oriented at referred object	563	2.3

4.3. Runtime

For many applications, real-time performance is essential. In this case study, the system achieves near-real-time performance and actual real-time is within reach. Table 1 lists the runtimes for the four experiments explained above, i.e. how long it took to process the four minutes of 1 *fps* input data that were used for each experiment. Runtime depends on many factors that are determined by the application domain at hand. In the current case study, the frame rate is only 1 *fps*, leading to faster runtimes. However, as many as 25 persons and 35 objects are involved, which leads to slower runtimes. The applied hardware and parallelization strategy also have a profound effect on runtime, in this case a desktop computer with a six core Intel Xeon W3690 using CPU multithreading. Finally, runtimes can be improved by optimizing the SGT traversal algorithm, preserving reasoning results that can be used again in other parts of the traversal.

5. Conclusion

This paper is concerned with automatic behavior understanding in smart work environments. The presented system automatically generates situation descriptions from annotated machine perception using fuzzy metric temporal logic (FMTL) and situation

graph trees (SGTs). It is evaluated in a case study on automatic behavior report generation for staff training purposes in crisis response control rooms. The motivation for this case study is the ability to automatically generate reports for multimedia retrieval, in order to increase the learning effect of recorded staff exercises. The required machine perception is annotated based on a real fire brigade exercise, using a specially developed data annotation tool. This hypothetical machine perception consists of person tracks, object information, and information about gestures, body pose, and speech activity, used as input by an FMTL/SGT reasoning engine to deduce situation descriptions: various group constellations and interaction patterns that can be used for automatically generated behavior reports.

Compared to [29], the contribution of this paper consists of significant improvements to the knowledge base that models group constellations and interaction patterns, some useful improvements to the software tools for annotating and visualizing data, ground-truth, and results, and a detailed quantitative evaluation of the system and its current results, using a substantial set of self-developed data and ground-truth. Four experiments were conducted in which the situation descriptions generated by the reasoning engine are compared to annotated ground-truth, produced by a self-developed ground-truth annotation tool. This is not a classification problem between a few classes. Situations must be classified correctly *and* all involved persons and objects must be consistent with ground-truth.

Future work There are plenty of opportunities to build upon the work presented in this paper. We will keep improving the FMTL rules and SGTs and perform new experiments with them, using the same data combined with a new set of ground-truth. Some ideas for improvements have been presented in Section 4.2. We will expand the knowledge base to recognize more sophisticated situations from the presented case study data, exploiting the full power of fuzzy evaluation and temporal modality. For example, we plan to model briefings (a recurring situation in control room operations) and their distinct phases, as well as the delivery and processing of messages, and persons underway between specific locations. Furthermore, we are analyzing inter-annotator agreement, adding DBSCAN (clustering) as preprocessing, and performing parameter learning on the applied trapezoid truth functions.

Another interesting opportunity is to perform experiments on imperfect data to evaluate the system's robustness. In previous studies [29,42], we took a first step in this direction, but some challenges remain. Imperfect data can contain noise, outliers, uncertainty, and gaps, which can be created artificially, as shown for data gaps in [42]. We plan to perform repeated random experiments, adding different amounts of noise, outliers, uncertainty, and data gaps, evaluating their influence on system performance. In [29], we described on a theoretical level how the applied methods can handle each type of imperfection, and how uncertainty values associated with the input data can be combined with truth values expressing vagueness. But these theories need to be evaluated, and the heuristics for combining uncertainty and vagueness need to be chosen carefully. Additionally, under certain circumstances, it will be benificial to interpolate the annotated data to obtain higher frame rates than 1 *fps*.

Furthermore, end-users, human science experts, and developers of crisis response software should get involved. The current case study is focused on the physical attributes of the people and objects in the room, but the system can be improved by taking into account more domain specific attributes, i.e. context information: field unit status, crisis dynamics, staff roles, more sophisticated object information, which can be largely obtained by monitoring developments in the crisis response software that is being used. This would allow us to model more sophisticated expert knowledge in FMTL and SGTs to deduce a richer set of situation descriptions that is of greater use to potential end-users.

Finally, the presented system can be applied to other application domains, such as the ones described in Section 1. Our research is situated in an environment that focuses on computer vision, other forms of machine perception, and human-machine interaction, which facilitates the progress toward an online system. The ultimate goal is a domain independent real-time system containing various machine perception components instead of annotated hypothetical machine perception, that has embodiment and action generation as well as synchronous real-time visualizations of sensor data, machine perception, and situation descriptions.

Acknowledgements

This work is supported by Fraunhofer-Gesellschaft Internal Programs Grant 692026.

Appendix

A. Interface specification

Rules for selecting objects as agents and patients, rules for obtaining atomic facts from the input data, and a rule for generating output.

Rule head	Explanation
SelectPersonAsAgent(p)	Select person p as center of reasoning (i.e. agent) for current traversal $-$ *Type(p,*person*)*.
SelectPatient(p, q)	Select person/object q as reasoning subject (i.e. patient) with agent $p - q \neq p$.
SelectPatients(p, $\boldsymbol{q}$*)*	Select list $\boldsymbol{q}$ containing all possible patients for agent $p - q \in \boldsymbol{q} \Leftrightarrow q \neq p$.
SelectPersonPatients(p, $\boldsymbol{q}$*)*	Select list $\boldsymbol{q}$ containing all possible patients of type person for agent $p - q \in \boldsymbol{q} \Leftrightarrow q \neq p \wedge$ *Type(q,* person*)*.
Type(p, t)	Query type t for person/object p.
Position(p, x, y)	Query position x, y for person/object p.
Size(p, w, h)	Query size w, h (in the horizontal plane) for person/object p.
Orientation(p, o)	Query horizontal orientation o for person/object p.
Geometry(p, x, y, w, h, o)	Query geometry x, y, w, h, o for person/object p.
Speaking(p)	Query if person p is speaking.
Gesticulating(p)	Query if person p is gesticulating.
ExtendingArm(p, o)	Query horizontal orientation o for extended arm of person p.
LookingDown(p)	Query if person p is looking down.
Sitting(p)	Query if person p is sitting.
Output(a, b, ...)	Output a string consisting of the arguments a, b, ... (arbitrary number of arguments).

B. Rule specifications

All rules that are directly or indirectly used by the SGTs in Figs 6 through 9, in alphabetical order.

Rule head	Explanation
AbsOrientationOfExtendedArm(p, o)	Orientation o (in °) is the absolute orientation of person p's extended arm (Appendix C: Formula 9).
AppendHead(p, $\boldsymbol{q}$*,* $\boldsymbol{r}$*)*	Append element p to the head of list $\boldsymbol{q}$, yielding list $\boldsymbol{r}$ ($\boldsymbol{r} = \{p\} \cup \boldsymbol{q}$).
AngleBetweenPoints(x1, y1, x2, y2, a)	Angle a is the angle of the line between points *(x1, y1)* and *(x2, y2)* (Appendix C: Formula 23).
AssocDiffBtwnOrient&AngToCenter(d, c)	Associate difference d (in °) with difference category c, where d is the difference between the orientation of a person/object and the angle of the line between its center and another person's/object's center. Category c can either be instantiated (e.g. with "small") which returns an appropriate truth value, or uninstantiated, allowing the rule to return a truth value > 0 for each defined and appropriate category (e.g. more than one of "no, small, medium, large") (Appendix D).
AssocDiffBetweenOrientations(d, c)	Associate difference d (in °) with difference category c, where d is the difference between the orientations of two persons/objects. Category c can either be instantiated or uninstantiated (Appendix D).
AssocDistBetweenCenters(d, c)	Associate distance d (in *cm*) with distance category c, where d is the distance between the centers of two persons/objects (Appendix D).

Continues on next page.

Continued from previous page.

AssocDistBetweenInnerPerimeters(d, c)	Associate distance d (in cm) with distance category c, where d is the distance between the inner perimeters of two persons/objects (Appendix D).
AssocSpeedInShortInterval(v, c)	Associate speed v (in cm/s) with speed category c, where v is the speed of a person/object over 3 s (Appendix D).
Atanoid(dx, dy, a)	Angle a is the arctangent for two perpendicular lines dx and dy (like *atan2*, but in a different coordinate system).
*BuildGroup(p, **q**, **r**)*	List $\boldsymbol{r}$ contains person p and all persons $q \in \boldsymbol{q}$ that fulfill *InSameGroup(p, q)* ($\boldsymbol{r} = \{\mathrm{p}\} \cup \{q \in \boldsymbol{q} : \mathit{InSameGroup(p, q)}\}$, Appendix C: Formula 4).
CalcDiffBtwnOrient&AngToCenter(p, q, d)	Calculate difference d (in °) between the orientation of person/object p and the angle of the line between p's center and another person/object q's center (Appendix C: Formula 19).
CalcDiffBetweenOrientations(p, q, d)	Calculate difference d (in °) between the orientations of two persons/objects p and q (Appendix C: Formula 18).
CalcDistBetweenCenters(p, q, d)	Calculate distance d (in cm) between the centers of persons/objects p and q (Appendix C: Formula 16).
CalcDistBetweenInnerPerimeters(p, q, d)	Calculate distance d (in cm) between the inner perimeters of persons/objects p and q (Appendix C: Formula 17).
CalcSpeedInShortInterval(p, v)	Calculate speed v of person/object p, using a 3 s interval (Appendix C: Formula 20).
Call(C)	Call rule head C and return its truth value. This is used to call rules dynamically at runtime.
DiffBetweenAngles(a, b, d)	Calculate difference d (in °) between angles a and b (Appendix C: Formula 24).
DiffBetweenAngles(x1, y1, x2, y2, b, d)	Calculate difference d (in °) between the angle of the line spanning points *(x1, y1)* and *(x2, y2)*, and angle b (Appendix C: Formula 22).
DiffBtwnOrient&AngToCenter(p, q, c)	Calculate the difference between the orientation of person/object p and the angle of the line between p's center and another person/object q's center. Then associate this difference with difference category c (Appendix C: Formula 14).
DiffBetweenOrientations(p, q, c)	Calculate the difference between the orientations of persons/objects p and q and associate this difference with difference category c (Appendix C: Formula 13).
*Difference(**p**, **q**, **r**)*	List $\boldsymbol{r}$ is the set difference of lists $\boldsymbol{p}$ and $\boldsymbol{q}$ ($\boldsymbol{r} = \boldsymbol{p} \setminus \boldsymbol{q}$); $\boldsymbol{r}$ contains all members of $\boldsymbol{p}$ that are not members of $\boldsymbol{q}$.
DistBetweenCenters(p, q, c)	Calculate the distance between the centers of persons/objects p and q and associate this distance with distance category c (Appendix C: Formula 11).
DistBetweenInnerPerimeters(p, q, c)	Calculate the distance between the inner perimeters of persons/objects p and q and associate this distance with distance category c (Appendix C: Formula 12).
DistBetweenPoints(x1, y1, x2, y2, d)	Calculate distance d (in cm) between points *(x1, y1)* and *(x2, y2)* (Appendix C: Formula 21).
*Empty(**p**)*	Determine whether $\boldsymbol{p}$ is an empty list ($\boldsymbol{p} = \emptyset$).
*Filter(**p**, C, **q**)*	Filter list $\boldsymbol{p}$, applying rule head C to its elements. List $\boldsymbol{q}$ contains the elements of $\boldsymbol{p}$ that fulfill rule head C ($\boldsymbol{q} = \{p \in \boldsymbol{p} : p \text{ fulfills } C\}$). The truth value returned by *Filter(**p**, C, **q**)* is the average of the truth values returned by rule head C applied to the elements of $\boldsymbol{p}$. If none fulfill C, $\boldsymbol{q} = \emptyset$, and *Filter(**p**, C, **q**)* returns truth value 1.0. Rule head C has the constant "elem" as one of its arguments, which is a placeholder for the elements of $\boldsymbol{p}$.
*GroupInShortInterval(p, **q**, **r**, **s**)*	List $\boldsymbol{r}$ contains person p and all persons in list $\boldsymbol{q}$ that fulfill *BuildGroup(p, **q**, **r**)* at $t = t_{current} - 1$. List $\boldsymbol{s}$ contains person p and all persons in list $\boldsymbol{q}$ that fulfill *BuildGroup(p, **q**, **s**)* at $t = t_{current}$ (Appendix C: Formula 3).

Continues on next page.

Continued from previous page.

InSameGroup(p, q)	Persons/objects p and q are in the same group if the distance between their centers is small and, either q is oriented at p, or p and q have the same orientation, or they are oriented at the same person/object r (Appendix C: Formula 5).
InShortInterval(C)	Call rule head C for every second in interval $[t_{current} - 2, t_{current} + 2]$ and return a truth value that is proportional to the number of seconds that C is fulfilled (Appendix C: Formula 25).
*Intersection(**p**, **q**, **r**)*	List $\boldsymbol{r}$ is the set intersection of lists $\boldsymbol{p}$ and $\boldsymbol{q}$ ($\boldsymbol{r} = \boldsymbol{p} \cap \boldsymbol{q}$); $\boldsymbol{r}$ contains all elements that are in $\boldsymbol{p}$ as well as in $\boldsymbol{q}$.
*JoiningLeavingGroup(**p**, **q**, **r**, **s**)*	Given a group at $t = t_1$ ($\boldsymbol{p}$) and at $t = t_2$ ($\boldsymbol{q}$), lists $\boldsymbol{r}$ and $\boldsymbol{s}$ respectively contain the members that joined and left the group between t_1 and t_2 (Appendix C: Formula 6).
LookingAt(p, q)	Determine whether person p is looking at person/object q (Appendix C: Formula 7).
LookingAtNearbyObject(p, q, c)	Determine whether person p is looking at person/object q and associate the distance between p and q with distance category c (Appendix C: Formula 1).
MaxAndMin(a, b, max, min)	Given two numbers a and b, determine their maximum and minimum.
*NegFilter(**p**, C, **q**)*	Filter list $\boldsymbol{p}$, applying rule head C to its elements. List $\boldsymbol{q}$ contains the elements of $\boldsymbol{p}$ that *do not* fulfill rule head C ($\boldsymbol{q} = \{p \in \boldsymbol{p} : \neg(p \text{ fulfills } C)\}$). *NegFilter(**p**, C, **q**)* returns truth value 1.0. Rule head C has the constant "elem" as one of its arguments, which is a placeholder for the elements of $\boldsymbol{p}$.
NormalizeAngle(a, b)	Angle b is the normalized equivalent of angle a.
Not(C)	Invert the truth value returned by rule head C (the F-Limette equivalent of "¬").
OrientedAtSame(p, q, r)	Determine to what degree persons p and q are oriented at the same person/object r, depending on their distances to r as well as their orientations relative to r (Appendix C: Formula 10).
PointingAt(p, q)	Determine whether person p is pointing at person/object q (Appendix C: Formula 8).
PointingAtNearbyObject(p, q, c)	Determine whether person p is pointing at person/object q and associate the distance between p and q with distance category c (Appendix C: Formula 2).
RayHitsObject(p, o, q)	Determine whether the ray emanating from person p's center, and following his orientation o, hits object q.
ReturnTruthValue(v)	When used inside another rule, that rule returns truth value v.
*Singleton(**p**)*	Determine whether $\boldsymbol{p}$ is a list containing only one element.
SpeedInShortInterval(p, c)	Calculate speed of person/object p in a 3 s interval and associate this speed with speed category c (Appendix C: Formula 15).

C. Formulas

Source code in logic notation for a substantial part of the rule specifications in Appendix B.

$$LookingAtNearbyObject(p,q,c) \leftarrow LookingAt(p,q) \wedge DistBetweenInnerPerimeters(p,q,c) \quad (1)$$

$$PointingAtNearbyObject(p,q,c) \leftarrow PointingAt(p,q) \wedge CalcDistBetweenInnerPerimeters(p,q,d) \wedge [c = \text{small} \vee (\neg LookingDown(p) \wedge c = \text{notLarge}) \vee (\neg LookingDown(p) \wedge Type(q, \text{display}) \wedge c = \text{any})] \wedge AssocDistBetweenInnerPerimeters(d,c) \quad (2)$$

$$GroupInShortInterval(p, \boldsymbol{q}, \boldsymbol{r}, \boldsymbol{s}) \leftarrow \Diamond_{-1} BuildGroup(p, \boldsymbol{q}, \boldsymbol{r}) \wedge BuildGroup(p, \boldsymbol{q}, \boldsymbol{s}) \quad (3)$$

$$BuildGroup(p, \boldsymbol{q}, \boldsymbol{r}) \leftarrow Filter(\boldsymbol{q}, InSameGroup(p, \text{elem}), \boldsymbol{r'}) \wedge AppendHead(p, \boldsymbol{r'}, \boldsymbol{r}) \quad (4)$$

$$InSameGroup(p,q) \leftarrow DistBetweenCenters(p,q,\text{small}) \wedge [DiffBetweenOrientationAndAngleToCenter(q,p,\text{small}) \vee DiffBetweenOrientations(q,p,\text{small}) \vee OrientedAtSame(p,q,r)] \quad (5)$$

$$JoiningLeavingGroup(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{r}, \boldsymbol{s}) \leftarrow Intersection(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{t}) \wedge Difference(\boldsymbol{q}, \boldsymbol{t}, \boldsymbol{r}) \wedge Difference(\boldsymbol{p}, \boldsymbol{t}, \boldsymbol{s}) \quad (6)$$

$$LookingAt(p,q) \leftarrow p \neq q \wedge Orientation(p, o_p) \wedge RayHitsObject(p, o_p, q) \quad (7)$$

$$PointingAt(p,q) \leftarrow p \neq q \wedge AbsOrientationOfExtendedArm(p, o_{arm_p}) \wedge RayHitsObject(p, o_{arm_p}, q) \quad (8)$$

$$AbsOrientationOfExtendedArm(p, o_{arm_p}) \leftarrow Orientation(p, o_p) \wedge ExtendingArm(p, o_{arm}) \wedge o'_{arm_p} = o_p + o_{arm} \wedge NormalizeAngle(o'_{arm_p}, o_{arm_p}) \quad (9)$$

$$OrientedAtSame(p,q,r) \leftarrow SelectPatient(p,r) \wedge SelectPatient(q,r) \wedge DistBetweenCenters(p,r,\text{notLarge}) \wedge DistBetweenCenters(q,r,\text{notLarge}) \wedge LookingAt(p,r) \wedge LookingAt(q,r) \quad (10)$$

$$DistBetweenCenters(p,q,c) \leftarrow CalcDistBetweenCenters(p,q,d) \wedge AssocDistBetweenCenters(d,c) \quad (11)$$

$$DistBetweenInnerPerimeters(p,q,c) \leftarrow CalcDistBetweenInnerPerimeters(p,q,d) \wedge AssocDistBetweenInnerPerimeters(d,c) \quad (12)$$

$$DiffBetweenOrientations(p,q,c) \leftarrow CalcDiffBetweenOrientations(p,q,d) \wedge AssocDiffBetweenOrientations(d,c) \quad (13)$$

$$DiffBetweenOrientationAndAngleToCenter(p,q,c) \leftarrow CalcDiffBetweenOrientationAndAngleToCenter(p,q,d) \wedge AssocDiffBetweenOrientationAndAngleToCenter(d,c) \quad (14)$$

$$SpeedInShortInterval(p,c) \leftarrow CalcSpeedInShortInterval(p,v) \wedge AssocSpeedInShortInterval(v,c) \quad (15)$$

$$CalcDistBetweenCenters(p,q,d) \leftarrow Position(p,x_p,y_p) \wedge Position(q,x_q,y_q) \wedge DistBetweenPoints(x_p,y_p,x_q,y_q,d) \quad (16)$$

$$CalcDistBetweenInnerPerimeters(p,q,d) \leftarrow CalcDistBetweenCenters(p,q,d_{pq}) \wedge Size(p,w_p,h_p) \wedge Size(q,w_q,h_q) \wedge MaxAndMin(w_p,h_p,d_{p_{out}},d_{p_{in}}) \wedge MaxAndMin(w_q,h_q,d_{q_{out}},d_{q_{in}}) \wedge d = d_{pq} - 0.5\,d_{p_{in}} - 0.5\,d_{q_{in}} \quad (17)$$

$$CalcDiffBetweenOrientations(p,q,d) \leftarrow Orientation(p,o_p) \wedge Orientation(q,o_q) \wedge DiffBetweenAngles(o_p,o_q,d) \quad (18)$$

$$CalcDiffBetweenOrientationAndAngleToCenter(p,q,d) \leftarrow Position(p,x_p,y_p) \wedge Position(q,x_q,y_q) \wedge Orientation(p,o_p) \wedge DiffBetweenAngles(x_p,y_p,x_q,y_q,o_p,d) \quad (19)$$

$$CalcSpeedInShortInterval(p,v) \leftarrow \Diamond_{-1} Position(p,x_{-1},y_{-1}) \wedge Position(p,x_0,y_0) \wedge \Diamond_1 Position(p,x_1,y_1) \wedge DistBetweenPoints(x_{-1},y_{-1},x_0,y_0,d_{-1,0}) \wedge DistBetweenPoints(x_0,y_0,x_1,y_1,d_{0,1}) \wedge v = 0.5\,(d_{-1,0} + d_{0,1}) \quad (20)$$

$AssocDistBetweenCenters(d,c)$, $AssocDistBetweenInnerPerimeters(d,c)$, $AssocDiffBetweenOrientations(d,c)$, $AssocDiffBetweenOrientationAndAngleToCenter(d,c)$, and $AssocSpeedInShortInterval(v,c)$ are represented graphically in Appendix D.

$$DistBetweenPoints(x_1,y_1,x_2,y_2,d) \leftarrow d_x = x_2 - x_1 \wedge d_y = y_2 - y_1 \wedge d = \sqrt{d_x^2 + d_y^2} \quad (21)$$

$$DiffBetweenAngles(x_1,y_1,x_2,y_2,a_1,d) \leftarrow AngleBetweenPoints(x_1,y_1,x_2,y_2,a_2) \wedge DiffBetweenAngles(a_1,a_2,d) \quad (22)$$

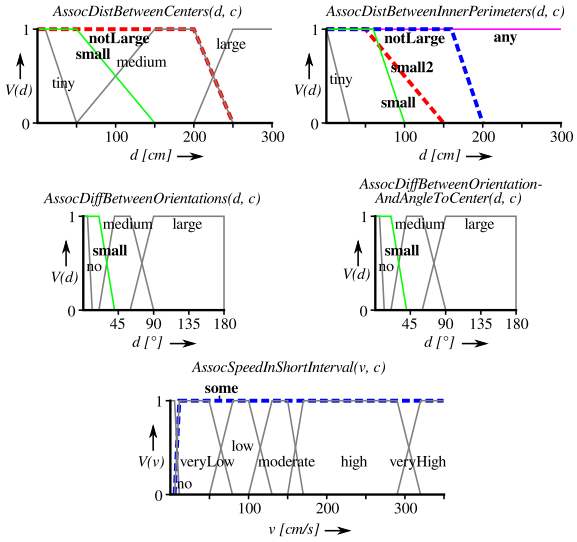
$$AngleBetweenPoints(x_1,y_1,x_2,y_2,a) \leftarrow d_x = x_2 - x_1 \wedge d_y = y_2 - y_1 \wedge Atanoid(d_x,d_y,a) \quad (23)$$

$$DiffBetweenAngles(a_1,a_2,d) \leftarrow MaxAndMin(a_1,a_2,a_{max},a_{min}) \wedge d' = a_{max} - a_{min} \wedge NormalizeAngle(d',d) \quad (24)$$

$$InShortInterval(C) \leftarrow [\Box^{100\%}_{-2,2} Call(C) \wedge ReturnTruthValue(1.0) \wedge !] \vee [\Box^{80\%}_{-2,2} Call(C) \wedge ReturnTruthValue(0.8) \wedge !] \vee [\Box^{60\%}_{-2,2} Call(C) \wedge ReturnTruthValue(0.6) \wedge !] \vee [\Box^{40\%}_{-2,2} Call(C) \wedge ReturnTruthValue(0.4) \wedge !] \vee [\Box^{20\%}_{-2,2} Call(C) \wedge ReturnTruthValue(0.2)] \quad (25)$$

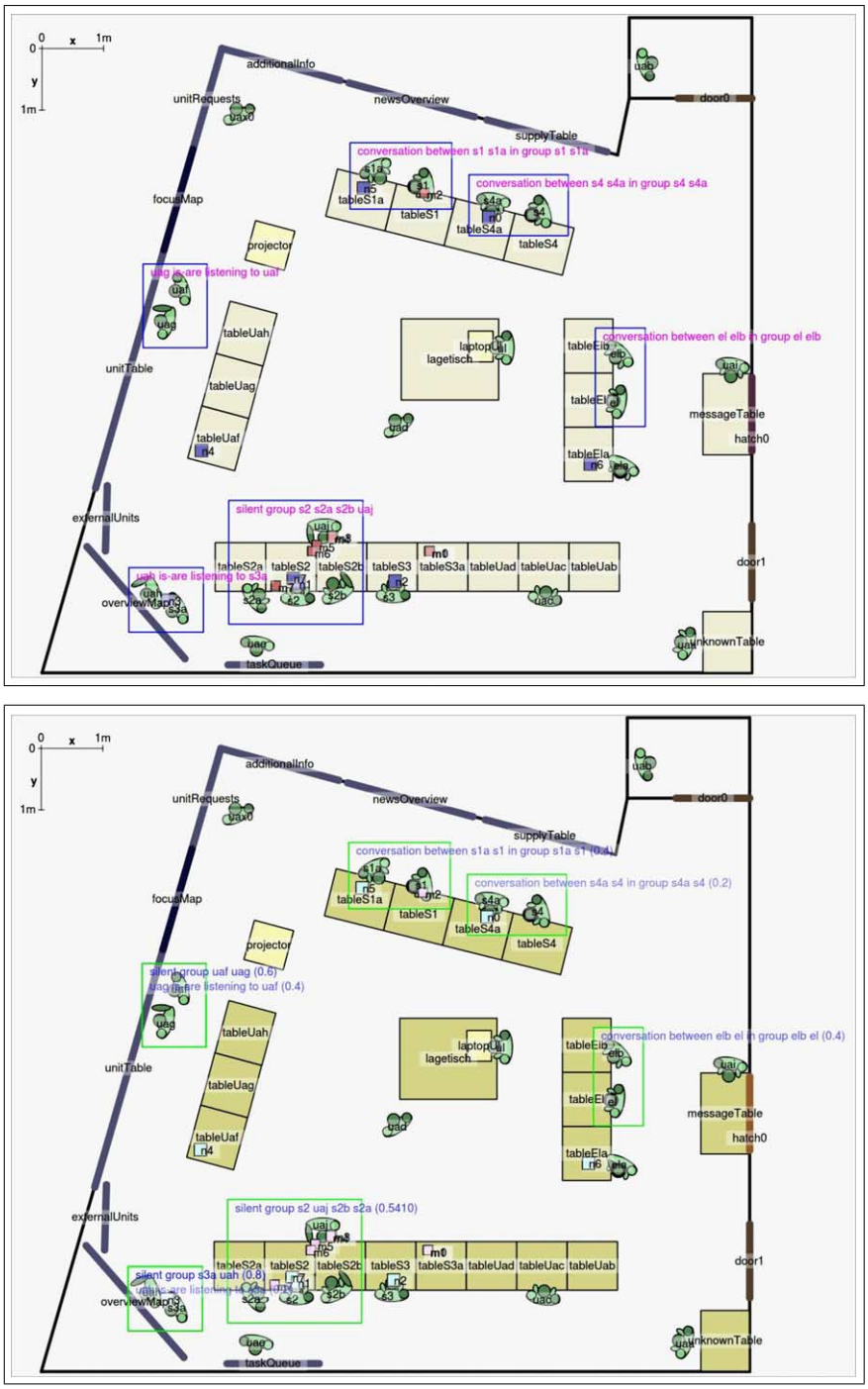
D. Trapezoid truth functions

The trapezoid truth functions (visualizations of the corresponding source code) used by Formulas 11 through 15 in Appendix C. They associate distances d in cm, differences d in °, and speeds v in cm/s to appropriate categories. As values on the x-axes increase, the truth values V on the y-axes (for the corresponding categories) increase from 0.0 to 1.0, stay constant at 1.0, and then decrease back to 0.0. The categories that are applied in this paper have boldfaced labels and colored curves.



E. Example ground-truth with corresponding reasoning results

Example frame with ground-truth annotations (top) and corresponding reasoning results with truth values (bottom), from the experiment "groups with speaking and listening members", rendered by the tool described in Section 3.1.3.



References

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

[2] J.F. Allen and G. Ferguson, Actions and events in interval temporal logic, *Logic and Computation* **4** (1994), 531–579.

[3] Y. Aloimonos, G. Guerra-Filho and A. Ogale, The language of action: A new tool for human-centric interfaces, in: *Human Centric Interfaces for Ambient Intelligence*, 2009, pp. 95–131.

[4] M. Arens, R. Gerber and H.H. Nagel, Conceptual representations between video signals and natural language descriptions, *Image and Vision Computing* **26**(1) (2008), 53–66.

[5] J.C. Augusto and C.D. Nugent, eds, Designing smart homes, in: *The Role of Artificial Intelligence*, Lecture Notes in Artificial Intelligence, Vol. 4008, Springer, 2006.

[6] A. Aztiria, J.C. Augusto, R. Basagoiti, A. Izaguirre and D.J. Cook, Discovering frequent user-environment interactions in intelligent environments, *Personal and Ubiquitous Computing* **16** (2012), 91–103.

[7] L. Bacon, L. MacKinnon, A. Cesta and G. Cortellessa, Developing a smart environment for crisis management training, *Journal of Ambient Intelligence and Humanized Comp.* **4** (2013), 581–590.

[8] N. Bellotto, B. Benfold, H. Harland, H.H. Nagel, N. Pirlo, I. Reid, E. Sommerlade and C. Zhao, Cognitive visual tracking and camera control, *Computer Vision and Image Understanding* **116**(3) (2012), 457–471.

[9] N. Bellotto, Robot control based on qualitative representation of human trajectories, in: *AAAI Symposium on Designing Intelligent Robots: Reintegrating AI*, AAAI Technical Report SS-12-02, 2012.

[10] W. Bohlken and B. Neumann, Generation of rules from ontologies for high-level scene interpretation, in: *Symposium on Rule Interchange and Applications (RuleML)*, 2009, pp. 93–107.

[11] O. Brdiczka, Integral framework for acquiring and evolving situations in smart environments, *Ambient Intelligence and Smart Environments (JAISE)* **2**(2) (2010), 91–108.

[12] P. Chahuara, A. Fleury, F. Portet and M. Vacher, Using markov logic network for on-line activity recognition from non-visual home automation sensors, in: *Conference on Ambient Intelligence (AmI)*, 2012, pp. 177–192.

[13] P. Dai, H. Di, L. Dong, L. Tao and G. Xu, Group interaction analysis in dynamic context, *IEEE Transactions on Systems, Man, and Cybernetics* **38**(1) (2008), 275–282.

[14] C. Fernández, P. Baiget, F.X. Roca and J. González, Determining the best suited semantic events for cognitive surveillance, *Expert Systems with Applications* **38** (2011), 4068–4079.

[15] C. Fernández, P. Baiget, F.X. Roca and J. González, Augmenting video surveillance footage with virtual agents for incremental event evaluation, *Pattern Recog. Lett.* **32**(6) (2011), 878–889.

[16] C. Filippaki, G. Antoniou and I. Tsamardinos, Using constraint optimization for conflict resolution and detail control in activity recognition, in: *Conf. on Ambient Intelligence (AmI)*, 2011, pp. 51–60.

[17] Y. Fischer and J. Beyerer, Defining dynamic bayesian networks for probabilistic situation assessment, in: *Conference on Information Fusion (FUSION)*, 2012, pp. 888–895.

[18] R. Gerber and H.H. Nagel, Representation of occurrences for road vehicle traffic, *Artif. Intelligence* **172**(4–5) (2008), 351–391.

[19] C. Geib, Delaying commitment in plan recognition using combinatory categorial grammars, in: *International Joint Conference on Artificial Intelligence (IJCAI)*, 2009, pp. 1702–1707.

[20] S. Gong and T. Xiang, Recognition of group activities using dynamic probabilistic networks, in: *International Conference on Computer Vision (ICCV)*, 2003, pp. 742–749.

[21] S. Gong and T. Xiang, *Visual Analysis of Behaviour, From Pixels to Semantics*, Springer, 2011.

[22] J. González, D. Rowe, J. Varona and F.X. Roca, Understanding dynamic scenes based on human sequence evaluation, *Image and Vision Computing* **27**(10) (2009), 1433–1444.

[23] B. Gottfried and H. Aghajan, eds, *Behaviour Monitoring and Interpretation, Smart Environments*, Ambient Intelligence and Smart Environments, Vol. 3, IOS Press, 2009.

[24] C. Gouin-Vallerand, B. Abdulrazak, S. Giroux and A.K. Dey, A context-aware service provision system for smart environments based on the user interaction modalities, *Ambient Intelligence and Smart Environments (JAISE)* **5**(1) (2013), 47–64.

[25] A. Gupta, P. Srinivasan, J. Shi and L. Davis, Understanding videos, constructing plots, learning a visually grounded storyline model from annotated videos, in: *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 2012–2019.

[26] M. Hanheide, A. Peters and N. Bellotto, Analysis of human-robot spatial behaviour applying a qualitative trajectory calculus, in: *Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2012, pp. 689–694.

[27] M. Henson, J. Dooley, A. Al Malaise Al Ghamdi and L. Whittington, Towards simple and effective formal methods for intelligent environments, in: *Conference on Intelligent Environments (IE)*, 2012, pp. 251–258.

[28] J. IJsselmuiden and R. Stiefelhagen, Towards high-level human activity recognition through computer vision and temporal logic, in: *German Conf. on Artificial Intelligence (KI)*, 2010, pp. 426–435.

[29] J. IJsselmuiden, A.K. Grosselfinger, D. Münch, M. Arens and R. Stiefelhagen, Automatic behavior understanding in crisis response control rooms, in: *Conference on Ambient Intelligence (AmI)*, 2012, pp. 97–112.

[30] Y. Ivanov and A. Bobick, Recognition of visual activities and interactions by stochastic parsing, *Pattern Analysis and Machine Intelligence* **22**(8) (2000), 852–872.

[31] A. Kembhavi, T. Yeh and L. Davis, Why did the person cross the road (there)? Scene understanding using probabilistic logic models and common sense reasoning, in: *European Conference on Computer Vision (ECCV)*, 2010, pp. 693–706.

[32] K.M. Kitani, Y. Sato and A. Sugimoto, Recovering the basic structure of human activities from noisy video-based symbol strings, *Pattern Recognition and Artificial Intelligence* **22** (2008), 1621–1646.

[33] D.I. Kosmopoulos, N.D. Doulamis and A.S. Voulodimos, Bayesian filter based behavior recognition in workflows allowing for user feedback, *Computer Vision and Image Understanding* **116**(3) (2012), 422–434.

[34] F. Krüger, K. Yordanova, C. Burghardt and T. Kirste, Towards creating assistive software by employing human behavior models, *Ambient Intelligence and Smart Environments (JAISE)* **4**(3) (2012), 209–226.

[35] G. Lavee, E. Rivlin and M. Rudzsky, Understanding video events: A survey of methods for automatic interpretation of semantic occurrences in video, *IEEE Transactions on Systems, Man, and Cybernetics* **39**(5) (2009), 489–504.

[36] B. Ley, V. Pipek, C. Reuter and T. Wiedenhoefer, Supporting improvisation work in inter-organizational crisis management, in: *Conference on Human Factors in Computing Systems (CHI)*, 2012, pp. 1529–1538.

[37] Z. Lu, J.C. Augusto, J. Liu, H. Wang and A. Aztiria, A system to reason about uncertain and dynamic environments, *Artificial Intelligence Tools* **21**(5) (2012).

[38] S. McKeever, J. Ye, L. Coyle, C. Bleakley and S. Dobson, Activity recognition using temporal evidence theory, *Ambient Intelligence and Smart Env. (JAISE)* **2**(3) (2010), 253–269.

[39] V. Morariu and L. Davis, Multi-agent event recognition in structured scenarios, in: *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 3289–3296.

[40] D. Münch, K. Jüngling and M. Arens, Towards a multi-purpose monocular vision-based high-level situation awareness system, in: *Workshop on Behaviour Analysis and Video Understanding @ ICVS*, 2011.

[41] D. Münch, J. IJsselmuiden, M. Arens and R. Stiefelhagen, High-level situation recognition using fuzzy metric temporal logic, case studies in surveillance and smart environments, in: *Workshop on Analysis and Retrieval of Tracked Events and Motion in Imagery Streams (ARTEMIS) @ ICCV*, 2011.

[42] D. Münch, J. IJsselmuiden, A.K. Grosselfinger, M. Arens and R. Stiefelhagen, Rule-based high-level situation recognition from incomplete tracking data, in: *Symposium on Rules, Research Based and Industry Focused (RuleML)*, 2012.

[43] H.H. Nagel, Steps toward a cognitive vision system, *AI Magazine* **25**(2) (2004), 31–50.

[44] F. Pecora, M. Cirillo, F. Dell'Osa, J. Ullberg and A. Saffiotti, A constraint-based approach for proactive, context-aware human support, *Ambient Intelligence and Smart Environments (JAISE)* **4**(5) (2012), 347–367.

[45] P. Rangel, J.G. Carvalho Junior, M.R. Ramirez and J.M. de Souza, in: *Context Reasoning Through a Multiple Logic Framework Conference on Intelligent Environments (IE)*, 2010, pp. 116–121.

[46] M. Ryoo and J. Aggarwal, Semantic representation and recognition of continued and recursive human activities, *Computer Vision* **82** (2009), 1–24.

[47] A. Sadilek and H. Kautz, Location-based reasoning about complex multi-agent behavior, *Artificial Intelligence Research* **43** (2012), 87–133.

[48] J. Shell and S. Coupland, Towards fuzzy transfer learning for intelligent environments, in: *Conference on Ambient Intelligence (AmI)*, 2012, pp. 145–160.

[49] Y. Shi, Y. Huang, D. Minnen, A. Bobick and I. Essa, Propagation networks for recognition of partially ordered sequential action, in: *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004, pp. 862–869.

[50] J.M. Siskind, Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic, *Artificial Intelligence Research* **15**(1) (2001), 31–90.

[51] A. Skarlatidis, G. Paliouras, G.A. Vouros and A. Artikis, Probabilistic event calculus based on markov logic networks, in: *Symp. on Rules, Research Based & Industry Focused (RuleML)*, 2011.

[52] T. Springer and A.Y. Turhan, Employing description logics in Ambient Intelligence for modeling and reasoning about complex situations, *Ambient Intelligence and Smart Environments (JAISE)* **1**(3) (2009), 235–259.

[53] K.A. Tahboub, Intelligent human-machine interaction based on dynamic bayesian networks probabilistic intention recognition, *Intelligent and Robotic Systems* **45** (2006), 31–52.

[54] H.J. ter Horst and A. Sinitsyn, Structuring reasoning for interpretation of sensor data in home-based health and well-being monitoring applications, *Ambient Intelligence and Smart Environments (JAISE)* **4**(5) (2012), 461–476.

[55] Z.O. Toups, A. Kerne and W.A. Hamilton, The team coordination game: Zero-fidelity simulation abstracted from fire emergency response practice, *ACM Transactions on Computer-Human Interaction* **18**(4) (2011), 23:1–23:37.

[56] S. Tran and L. Davis, Event modeling and recognition using markov logic networks, in: *European Conference on Computer Vision (ECCV)*, 2008, pp. 610–623.

[57] P. Turaga, R. Chellappa, V. Subrahmanian and O. Udrea, Machine recognition of human activities: A survey, *Circuits and Systems for Video Technology* **18**(11) (2008), 1473–1488.

[58] T.L.M. van Kasteren, G. Englebienne and B.J.A. Kröse, Activity recognition using semi-markov models on real world smart home datasets, *Ambient Intelligence and Smart Environments (JAISE)* **2**(3) (2010), 311–325.

[59] T.L.M. van Kasteren, G. Englebienne and B.J.A. Kröse, Hierarchical activity recognition using automatically clustered actions, in: *Conference on Ambient Intelligence (AmI)*, 2011, pp. 82–91.

[60] S. Vishwakarma and A. Agrawal, A survey on activity recognition and behavior understanding in video surveillance, in: *The Visual Computer*, 2012, pp. 1–27.

[61] V.T. Vu, F. Bremond and M. Thonnat, Automatic video interpretation: a novel algorithm for temporal scenario recognition, in: *Conf. on Artificial intelligence (IJCAI)*, 2003, pp. 1295–1300.

[62] J. Xiang, J. Tian and A. Mori, Goal-directed human activity computing, *Ambient Intelligence and Smart Environments (JAISE)* **3**(2) (2011), 127–145.

[63] B.Z. Yao, X. Yang, L. Lin, M.W. Lee and S.C. Zhu, I2T: Image parsing to text description, *Proceedings of the IEEE* **98**(8) (2010), 1485–1508.

[64] J. Ye and S. Dobson, Exploring semantics in activity recognition using context lattices, *Ambient Intelligence and Smart Environments (JAISE)* **2**(4) (2010), 389–407.

[65] J. Ye, S. Dobson and S. McKeever, Situation identification techniques in pervasive computing: A review, *Pervasive and Mobile Computing* **8**(1) (2011), 36–66.

[66] D. Zhang, D. Gatica-Perez, S. Bengio and I. McCowan, Modeling individual and group actions in meetings with layered HMMs, *Transactions on Multimedia* **8**(3) (2006), 509–520.