Available online at www.sciencedirect.com
ScienceDirect
Artificial Intelligence 172 (2008) 351–391
Artificial Intelligence
www.elsevier.com/locate/artint

Representation of occurrences for road vehicle traffic

R. Gerber, H.-H. Nagel *

Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe (TH), 76128 Karlsruhe, Germany

Received 28 June 2004; received in revised form 17 July 2007; accepted 19 July 2007

Available online 27 July 2007

Abstract

Our 3D-model-based Computer Vision subsystem extracts vehicle trajectories from monocular digitized videos recording road vehicles in inner-city traffic. Steps are documented which import these quantitative geometrical results into a conceptual representation based on a Fuzzy Metric-Temporal Horn Logic (FMTHL, see [K.H. Schäfer, Unscharfe zeitlogische Modellierung von Situationen und Handlungen in Bildfolgenauswertung und Robotik, Dissertation, 1996]). The *facts* created by this *import step* can be understood as verb phrases which describe elementary actions of vehicles in image sequences of road traffic scenes. The current contribution suggests a *complete conceptual* representation of *elementary vehicle actions* and reports results obtained by an implementation of this approach from real-world traffic videos.

Keywords: Computer vision; Knowledge representation; Temporal reasoning; Reasoning about actions and change; Fuzzy metric-temporal logic

1. Introduction

An adult is expected to be able to write down—not necessarily with style and precision—what he sees. Conceding similar, but appropriately adapted reservations, what is required to have a computer perform an analogous task? Obviously, an analogue to human seeing could be Computer Vision. The notion of an automatic report generator, too, is no longer considered as science fiction. It most likely turns into a challenge, however, to imagine the detailed communication between a computer vision (sub)system and an algorithmic report generator. What looks like the mere definition of an interface will turn out to require the design of a system-internal *logic-based* conceptual representation of a text in combination with the design of an entire set of processes operating on this representation.

Investigations to be discussed in the sequel address an important step towards the algorithmic transformation of video signals into a natural language text which describes the recorded scene, in particular its temporal development. The presentation will first sketch an overall system concept in order to provide a framework for the subsequent discussion which will then concentrate on the conversion of *geometric tracking results* into *elementary conceptual representations* of relevant aspects of the (short-term) development in the recorded scene. A preliminary version of this approach has been partially outlined in [10].

* Corresponding author.
E-mail address: nagel@iaks.uni-karlsruhe.de (H.-H. Nagel).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2007.07.001

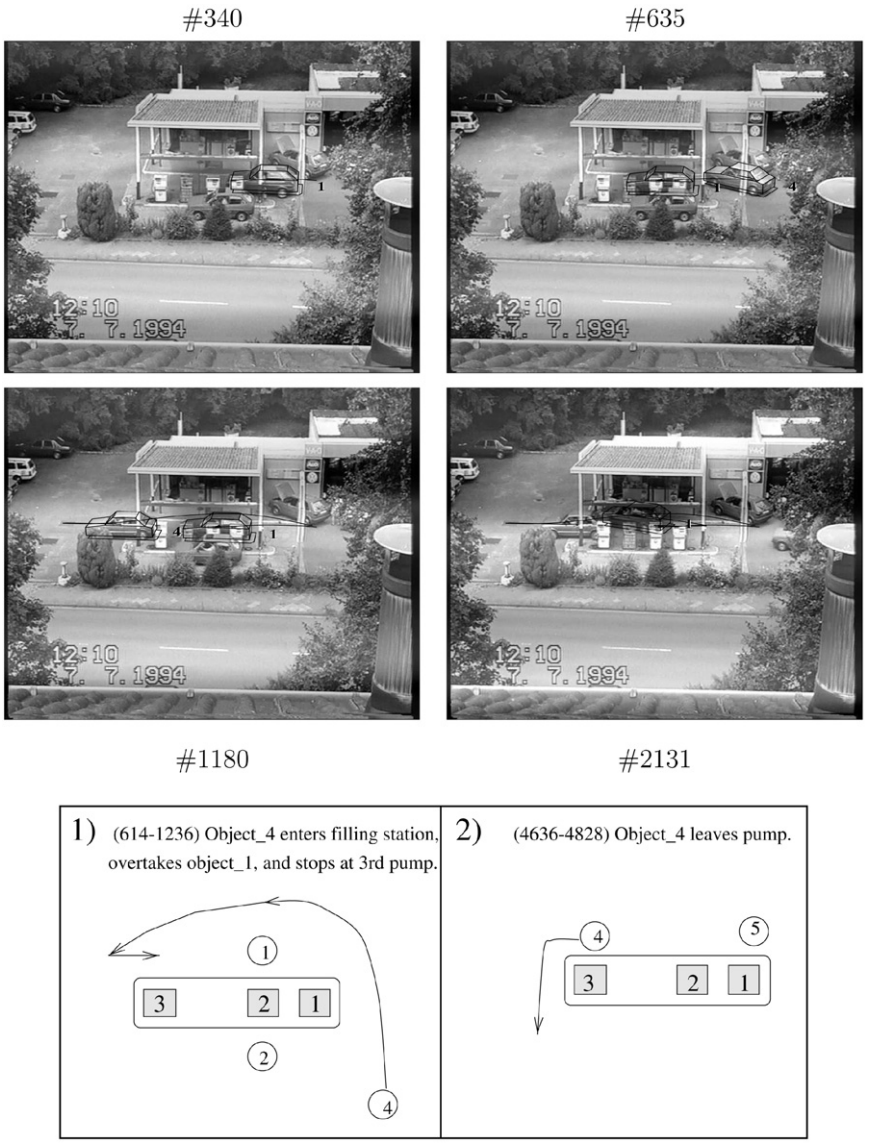


Fig. 1. In the upper left panel, the image plane projection of a polyhedral model for a fastback has been overlaid to frame number 340 from an image sequence recorded at a gas station. In addition, one can see the trajectory segment obtained by automatic model-based tracking of this vehicle which will be referred to as *object_1*. Frames 635, 1180, and 2131 show snapshots of various maneuvers of another vehicle (*object_4*), with analogous overlays of a projected polyhedral model and the trajectory for this vehicle (see Section 1 for more explanations). The sketch in the bottom row illustrates the maneuvers of object_4 in this sequence.

Fig. 1 illustrates a coherent source of examples for different stages of such a transformation. The first frame[1] #340 in the upper left panel shows a snapshot where a fastback has already stopped at the second petrol pump on the filling lane (see Fig. 2) closer to the observer (subsequently referred to as the 'lower filling lane'). A second fastback (subsequently referred to as 'object_1') had just entered the gas station and selected the filling lane on the other side

[1] Based on special derivative operators which suitably interpolate between digitizations in even and odd scanlines of interlaced video (see, e.g., [32]), actually each half-frame—or field in video-coding terminology—is evaluated in its own right, resulting in a temporal sampling rate of 20 msec. This aspect reduces the approximation errors by the Extended Kalman-Filter used and thus improves the tracking quality. Beyond this fact, however, it does not influence the conversion of geometric results to natural language concepts. In order to simplify the presentation, we shall use the term frame henceforth without further qualifications.

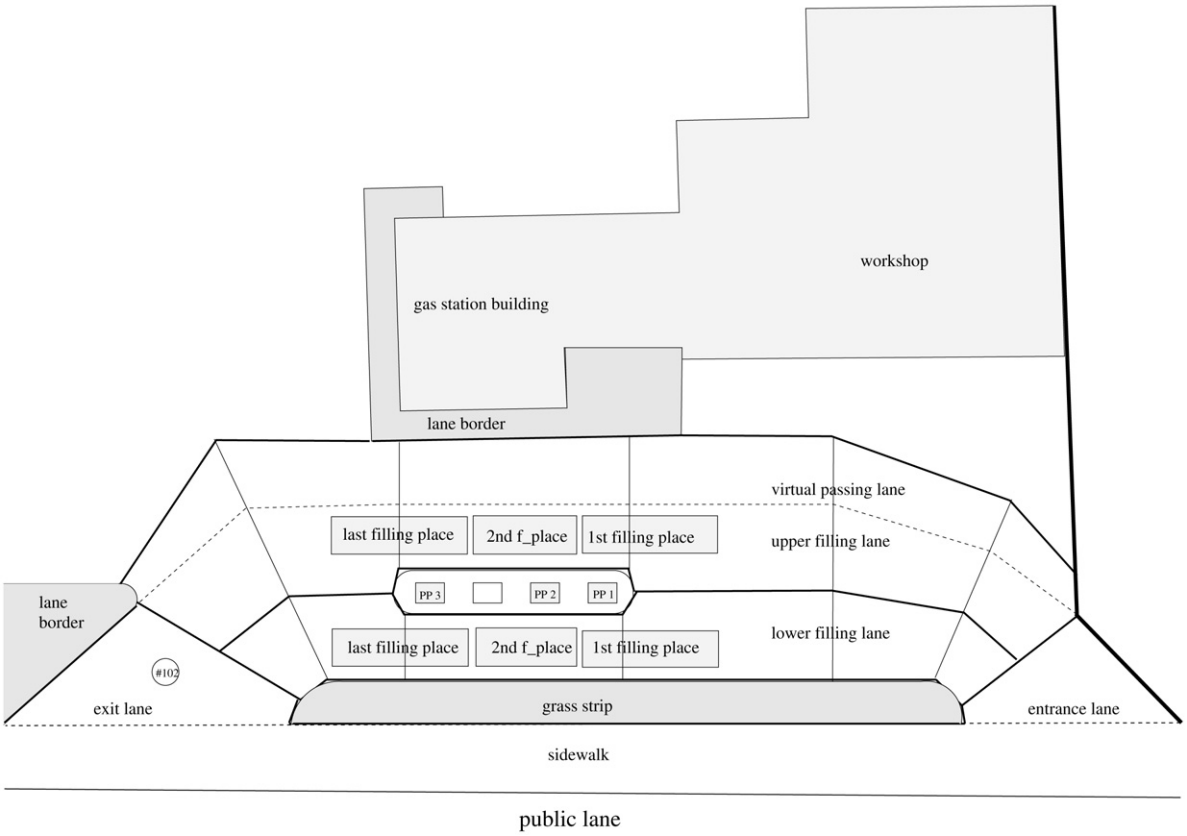


Fig. 2. Groundplan sketch of the gas station.

of the petrol pumps ('upper filling lane') in order to stop next to the second petrol pump, too. About 300 frames—i.e. 6 seconds—later, a third vehicle, a sedan ('object_4'), entered the gas station and headed towards a passing lane between the upper filling lane and the gas station building. It passed object_1, changed back to the upper filling lane, stopped there and backed up slowly until it eventually stood next to the third petrol pump, immediately in front of object_1, around frame-time 1180. About 1000 frames (20 secs) later, after the fastback on the lower filling lane had already left the gas station, object_1 started to move backwards to gain space in order to change to the passing lane. Object_1 then passed object_4 and headed towards the exit of the gas station. About three quarters of a minute later around frame-time 4600, object_4 started to move forward and headed towards the exit, too. Examples will refer mostly to the sequence of maneuvers performed by object_4 and object_1 during the period while this image sequence had been recorded.

The derivation of conceptual representations and textual descriptions of agent behavior from visual input has become a research topic of constantly growing interest in the last few years. Such research has to deal with the uncertainties related to the geometric results estimated from video sequences and with *bridging the semantic gap* between (mainly geometric) computer vision results and (mainly conceptual) action descriptions. This contribution addresses a basic topic related to the second aspect, namely *isolatable agent activities* or *occurrences* for *non-human* agents, in particular rigid vehicles in videos recorded from road traffic. A discussion of relevant prior publications will be postponed to the concluding sections because similarities and differences can be stated there more succinctly with respect to what will be reported in the sequel.

2. System outline

Algorithmic text generation based on a recorded video sequence has to be concerned with at least two disciplines, namely computer vision and computational linguistics. Each of these two disciplines already covers several subdis-

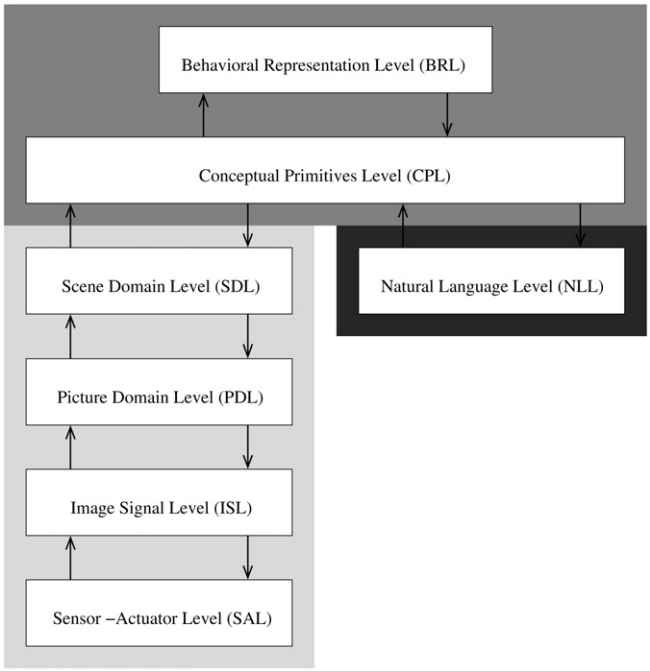


Fig. 3. Coarse layer structure of the overall system (from [28]; ©2000 IEEE, by permission). The layers with light gray background constitute the *core Computer Vision* subsystem for the extraction of a (mostly geometric) 3-D scene representation. The *Conceptual Representation* subsystem has a medium gray background, the *text generation* is incorporated into the *Natural Language Level* with background in dark gray (see, too, the text or [29]).

ciplines. Any system for video-to-text transformation will thus be complex and, therefore, difficult to present and to analyse. The following subsection provides an overview of our entire system approach, thereby setting the frame for a more detailed outline of steps in video-based text generation proper. More information about the development of this system concept during past decades can be found in [29], with recent developments being discussed in [2].

2.1. Overall system structure

The transformation of video signals into a text describing the recorded temporal development within the depicted scene can be subdivided into three groups of processes—see Fig. 3:

(1) The subsystem which controls the video recording and the subsequent processing steps up to and including the extraction of 3-D time-dependent geometric descriptions of the scene and, in particular, of visibly *moving* bodies. This subsystem comprises the layers devoted to the following subtasks:
   (a) Control of the recording equipment including actuators required, for example, to change pan and tilt of video camera heads, zoom of camera lenses, etc.—the *Sensor-Actuator-Level* (*SAL*).
   (b) The *Image-Signal-Level* (*ISL*) devoted to image processing operations on the recorded video signal.
   (c) The *Picture-Domain-Level* (*PDL*) where information extracted from the image signal is aggregated into Picture-Domain-Descriptors in the 2-D image plane.
   (d) The *Scene-Domain-Level* (*SDL*) which combines Picture-Domain-Descriptors with knowledge about the camera and about the scene in order to obtain a *three-dimensional* representation of (at least) the geometry of temporal developments in the recorded scene.

   In the particular example illustrated by Fig. 1, this information comprises the 3-D vehicle status together with the 3-D model of those vehicles which have been detected, initialized, and tracked. The vehicle status comprises the scene ground plane coordinates (x, y) of the model reference point, the vehicle orientation θ, speed v, and

steering angle[2] ψ. The vehicle status is updated at each frame time point, i.e. every 20 msec, by a Kalman-Filter incorporated into a model-based tracking process—see [14,19,23].[3]

(2) The quantitative 3-D spatio-temporal information provided by the model-based vehicle tracking subsystem is converted into an elementary conceptual representation at the interface between the Scene-Domain-Level and the *Conceptual-Primitives-Level* (*CPL*). Information about the spatio-temporal developments in the scene represented in form of conceptual primitives is aggregated by abstraction processes into information about, for example, the behavior of agents in the depicted scene at the *Behavior-Representation-Level* (*BRL*). These two layers constitute the interface between the core Computer Vision subsystem (comprising the SAL, the ISL, the PDL, and the SDL; light gray background in Fig. 3) on the one hand and the text generation subsystem incorporated into the remaining subsystem, namely

(3) the *Natural-Language-Level* (*NLL*) (dark gray background in the same figure). This latter layer could comprise in principle—in addition to a natural language text generation component—also a natural language question-answering component although such a component has not been studied yet in this context.

The remainder of this exposition will concentrate on the subsystem for the representation and use of *Conceptual Primitives*.

2.2. Principal steps for text generation

The system outlined in Fig. 4 still constitutes an exploratory stage of text generation from video recordings. Design and implementation of this system version were based on the following considerations:

- It appeared more important at this stage of the overall investigation to conceive and implement an *entire* system which attempts to *cover a carefully delimited discourse domain completely* rather than to construct isolated subsystems devoted to an in-depth study of special problems.
- In particular, the design and exploitation of *conceptual representations* should be emphasized, based on the hypothesis that the interface between the extraction of geometric information and the formulation of this information as a natural language text constitutes the real challenge at this stage of our investigations.
- In order to base such a system on a reliable methodology, treatment of intermediate results at the conceptual level should *use formal logic* to the extent possible.
- The Computer Vision subsystem `Xtrack` evaluates monocular image sequences of road traffic scenes recorded by a stationary video camera. The detection and tracking of road vehicles by `Xtrack` exploits their motion relative to the (assumed) static background and foreground. Apart from this more technical aspect, vehicular motion is of prime interest because it provides the basis for
  – a (short-time) prediction of vehicle appearance in the next image frame and
  – some (longer-time) prediction during entire image subsequences.
  The latter aspect should allow to characterize vehicle *behavior* conditioned on a representation of the current status of the depicted scene.

As will be explained in more detail below, geometric results of the core Computer Vision subsystem are imported into the conceptual representation subsystem—see Fig. 4—together with knowledge about the geometry of the static part of the depicted scene. A two-step approach converts this input from a quantitative, numerical representation into a qualitative, conceptual one: the first step converts the input into discrete values compatible with predefined attribute schemes, the second step then combines the resulting attributes to assert *occurrences*, i.e. conceptual representations for a recognizable significant change (or its absence!) of a vehicle's status, in particular its motion status. The time

[2] Some of the experimental results discussed in the sequel have been obtained with an older version of the road vehicle tracking system `Xtrack` (see [14,19,21]) where the angular velocity around the vertical axis of the vehicle (here identical with the ground plane normal) constituted the fifth component of the vehicle state vector.

[3] A framework called `MOTRIS` (**Mo**del-based **Tr**acking in **I**mage **S**equences) comprising the `Xtrack` model-based tracking approach has been re-designed and re-implemented in Java. It has been released under GNU GPL, see [26].

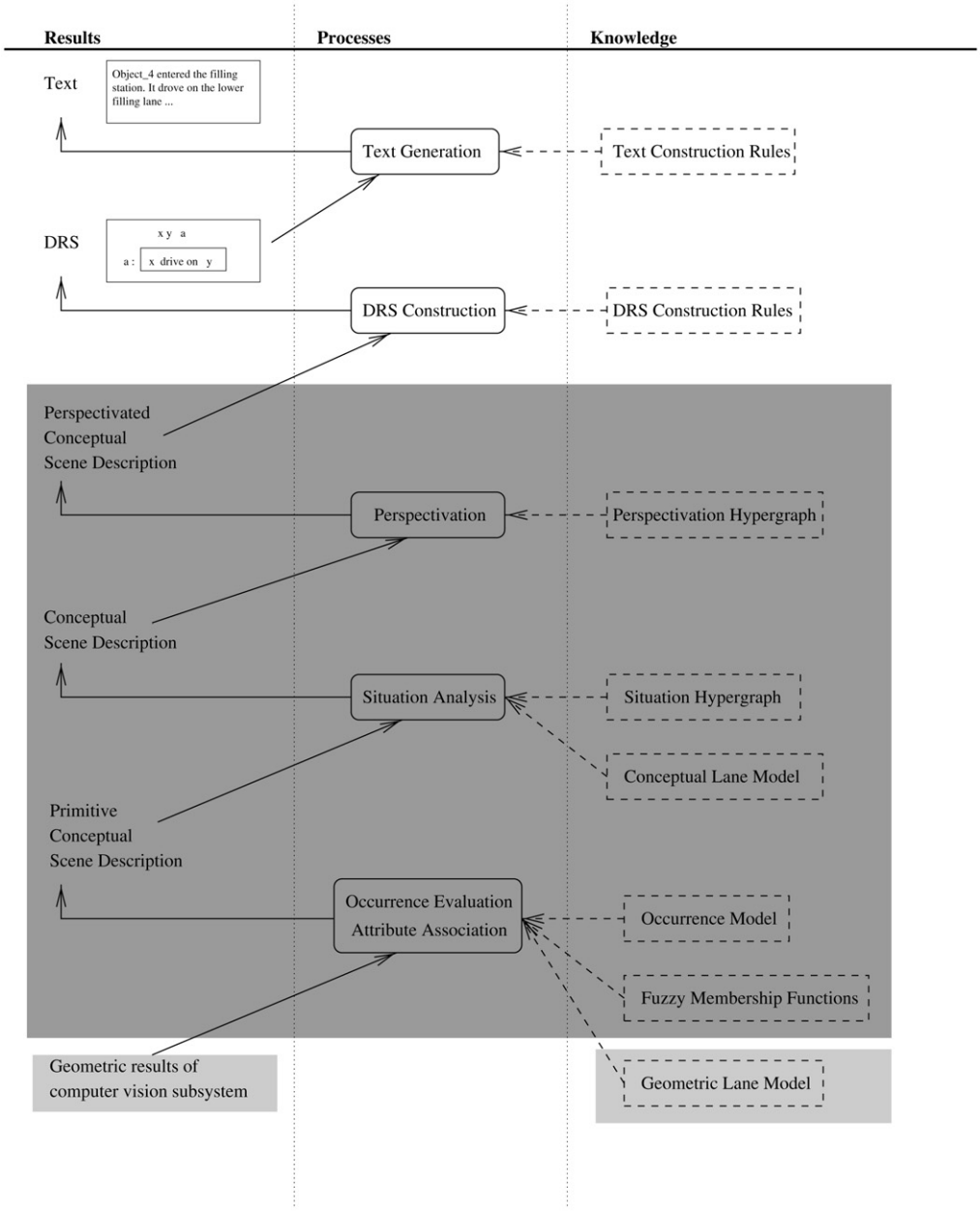


Fig. 4. A feed-forward system version designed to convert geometric results obtained by a model-based tracking (core Computer Vision) subsystem into a coherent natural language text. At each layer, an entry into the left column corresponds to the representation of intermediate results at that layer. Entries into the right column explicate the kind of knowledge exploited by the transformation subprocess at that particular layer. The left entry at the bottom layer refers to results obtained by the Computer Vision subsystem. The entry into the right column of the bottom layer (Geometric Lane Model) refers to a-priori knowledge about the geometry of the depicted scene. The layers with dark gray background use a *Fuzzy Metric-Temporal Horn Logic* (FMTHL) in order to represent and manipulate a-priori knowledge together with the results provided by the next lower layer. The topmost two layers rely on *Discourse Representation Structures* in order to convert this conceptual representation into a coherent natural language text.

scale relating to such *primitive* conceptual representations of vehicle motion extends from a fraction of a second upwards to several seconds.

The next step (Situation Analysis) combines primitive conceptual representations with knowledge about conditions in the scene which may influence the switch between particular occurrences as being the most appropriate description

Table 1
Resulting Degree of Validity (DoV) of a fuzzy logic operation δ combining two formulae $\mathcal{F}_1, \mathcal{F}_2$ with related DoV $\wp_1, \wp_2$, respectively, for weak, medium, and strong semantics (from [35])

δ	**w**eak	**m**edium	**s**trong
$\leftarrow$	$\min(1, 1 - \wp_2 + \wp_1)$	$1 - \wp_2 + \wp_1 * \wp_2$	$\max(\wp_1, 1 - \wp_2)$
$\wedge$	$\min(\wp_1, \wp_2)$	$\wp_1 * \wp_2$	$\max(0, \wp_1 + \wp_2 - 1)$
$\vee$	$\min(1, \wp_1 + \wp_2)$	$\wp_1 + \wp_2 - \wp_1 * \wp_2$	$\max(\wp_1, \wp_2)$

Table 2
General structure of facts, rules, and queries in F-Limette and FMTHL (from [35])

	Expression in F-Limette	Corresponding FMTHL-formula
Fact	$\lambda \mid t_1 : t_2 \,!\, Rel.$	$\downarrow_\lambda \circ_{[t_1,t_2]} Rel$
Rule	$\lambda \mid t_1 : t_2 \,!\, (Rel_1 \text{:-} Rel_2).$	$\downarrow_\lambda \circ_{[t_1,t_2]} (Rel_1 \leftarrow Rel_2)$
Query	$\lambda \mid t_1 : t_2 \,?\, Rel.$	$\downarrow_\lambda \diamond_{[t_1,t_2]} Rel$

of (short-term) vehicular behavior. The dominant scale for temporal intervals of interest thus changes by 1–2 orders of magnitude, i.e. to between several seconds and a minute or more.

The *Conceptual Scene Description* obtained at this level is then prepared for presentation to a human user who is anticipated in our case so far as a reader of the textual descriptions to be generated. The conceptual representation of preferences expressed by this reader—his *perspective* on the temporal development of the recorded scene—is exploited in order to create a *Perspectivated Conceptual Scene Description* PCSD.

This PCDS is transferred to processes at the Natural Language Level in order to be converted into natural language text. As a first step, the *Perspectivated Conceptual Scene Description* obtained from a particular image (sub)sequence will be converted into a *Discourse Representation Structure* (see [18]) which in turn is converted into the output text, see [11]. In the remainder of this contribution, we shall concentrate on the interface—see Fig. 3—between the core Computer Vision subsystem and the *Conceptual Primitives Level*, i.e. the bottom two levels indicated in Fig. 4.

3. Notation

In order to facilitate a compact presentation, a short introduction of the *Logical* Vocabulary of the Fuzzy Metric Temporal Horn Logic notation is provided which will be used in the sequel.

Fuzzy Metric Temporal Logic (*FMTL*) extends the First Order Predicate Calculus (FOPC) with metric temporal and fuzzy reasoning (see [35]). *Temporal* Logic extends FOPC by adding sets of time instants and their ordering. As a consequence, the *Degree of Validity* (DoV) of formulae can vary with time. *Metric* temporal logic is based on a linear, discrete time structure corresponding to integer values, here representing the frame number of video image frames. FMTHL comprises two temporal logic operations ($\circ_{s,e}$ and $\diamond_{s,e}$) for denotation of universal and existential validity of formulae, respectively, within given sets of time instants. *Fuzzy* Logic generalizes FOPC by *real number* truth values $\mu \in [0, 1]$. Fuzzy degrees of validity are generated for logic subjunction, conjunction, disjunction, and negation. Table 1 comprises three possible semantics—*weak (w), medium (m)* and *strong (s)*—related to the first three logic operations. The two monadic logic operations $\downarrow_\lambda$ and $\uparrow_\lambda$ represent *weakening* and *intensivation*, respectively, of validity.

FMTHL is the Horn-Logic fragment of FMTL. FMTHL-formulae can be subdivided into *rules, facts*, and *queries*. The *head* of a rule consists of a certain predicate p, whereas its body comprises specific conditions whose validities determine the validity of p. Stating a query to F-Limette[4] involves finding a substitution on the basis of given rules which verifies the validity of the query: for each predicate p currently required, the bodies of all rules are evaluated recursively whose head comprises p.

Table 2 conveys the general structure of expressions in F-Limette and corresponding FMTHL-formulae. Facts and rules are, in general, universally quantified whereas queries are quantified existentially.

[4] This system has been made available as a package under GNU Public License, in connection with the *Situation Graph Editor* of M. Arens, see http://cogvisys.iaks.uni-karlsruhe.de/Vid-Text/.

F-Limette follows widespread conventions for formal logic in that identifier symbols for variables start with capital letters whereas function and predicate symbols start with a lower case letter. Constants are considered as 0-ary function symbols and thus begin, too, with a lower case letter. Temporal constants $-i$ and $+i$ allow to use *infinity* as a boundary value for the temporal validity of formulae. The term `always` in front of a formula stands for '1 | -i : +i !' which means unrestricted validity of the formula (expressed by the number 1 for *true* in front of the vertical separation bar) from minus *infinity* until plus *infinity*, i.e. for all time instants. The term `next` within a formula stands for '+1 !' which means that the predicate following this term has to be valid at the *next* time instant.

The two-letter symbol `:-` denotes a (re-)implication operator. Predicates in the body of an FMTHL-expression which are separated by a *comma* are *joined conjunctively*. The *semicolon* denotes the *disjunction* operator. The operator `is` assigns the result for the expression on its right-hand side to the variable on its left-hand side, whereas `>=, =, <, +, -, *, /` correspond to conforming relational and arithmetic operators, respectively.

4. Results and a-priori knowledge imported from the computer vision subsystem

The Computer Vision subsystem provides two different types of information to the conceptual representation subsystem: *tracking results*—i.e. estimates for the time-dependent state of vehicles detected and tracked in an input video sequence by a model-based approach—and (time-independent) *geometric* lane data exploited in this context. The latter data have to be converted into a *conceptual* representation of the lane structure in the recorded scene for further use by the FMTHL inference engine F-Limette activated by the conceptual representation subsystem.

4.1. Import of tracking results

Tracking results provided by the Computer Vision subsystem comprise geometric values for the (x, y)-position of vehicles in the recorded scene, their orientation θ, their velocity v, and their steering angle ψ for each *agent* (vehicle) and frame time point. Each result, after conversion to an FMTHL fact in the form of a predicate named `has_status` connected with individuals for the variable `Agent` (for example, 'object_4'), is imported into the conceptual representation subsystem. In the following, a small part of the trajectory data derived for 'object_4' (see Fig. 1) is shown below:

time [frame #]	!		agent	x [m]	y [m]	θ [°]	v [m/s]	ψ [°]
614	!	has_status(	object_4,	8.941049,	1.849823,	146.9672,	2.934614,	−2.064247).
615	!	has_status(	object_4,	8.884917,	1.873256,	147.0714,	2.873042,	−2.462217).
616	!	has_status(	object_4,	8.830486,	1.899629,	147.1823,	2.809821,	−2.830728).
617	!	has_status(	object_4,	8.780960,	1.926439,	147.2313,	2.803807,	−3.210003).
618	!	has_status(	object_4,	8.733459,	1.954635,	147.2106,	2.793914,	−3.581725).
619	!	has_status(	object_4,	8.686386,	1.986568,	147.1799,	2.737727,	−3.541704).
620	!	has_status(	object_4,	8.639212,	2.014085,	147.2301,	2.675781,	−3.131224).

Since the Computer Vision subsystem currently does not associate yet a Degree of Validity (DoV) with its geometric results, each imported fact is treated as absolutely valid (i.e. $\mu = 1$). For details concerning the notation of FMTHL facts see Section 3, Table 2.

4.2. Lane geometry

The lane geometry of a traffic scene needs to be known to our Computer Vision subsystem `Xtrack`, for example, in order to present the lane structure of selected image frames for interactive inspection of intermediate tracking results.

Fig. 5 shows a bird's eye view of the lane model corresponding to the gas station where the gas station sequence illustrated in Fig. 1 has been recorded.

The geometric lane model consists of points, lines and lane segments (for example, points p347–p350 in Fig. 6). This data is used in the conceptual representation subsystem in order to associate agent positions with lanes and to derive conceptual descriptions of vehicle behavior. Analogously to the trajectory data in Section 4.1, these geometric

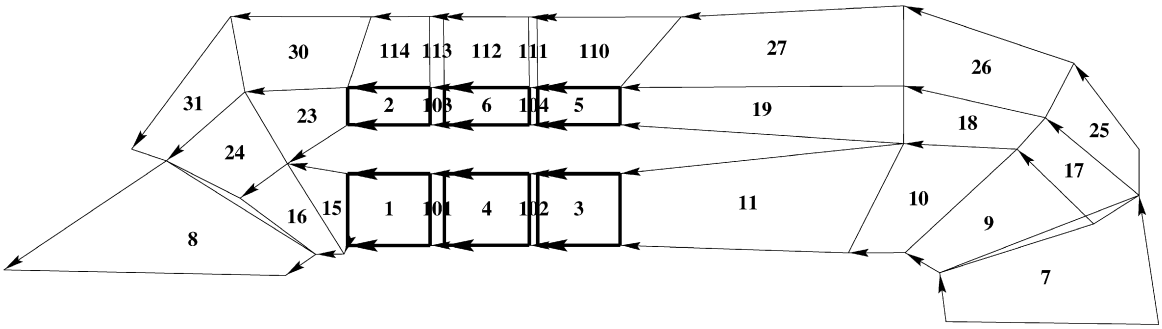


Fig. 5. Bird's eye view of the quantitatively known lane model for the gas station. The alphanumeric identifiers shown here for *lane segments* establish a correspondence between the *geometric* description given by the line segments on the one hand and the *conceptual* representation of the lane structure required for text generation.

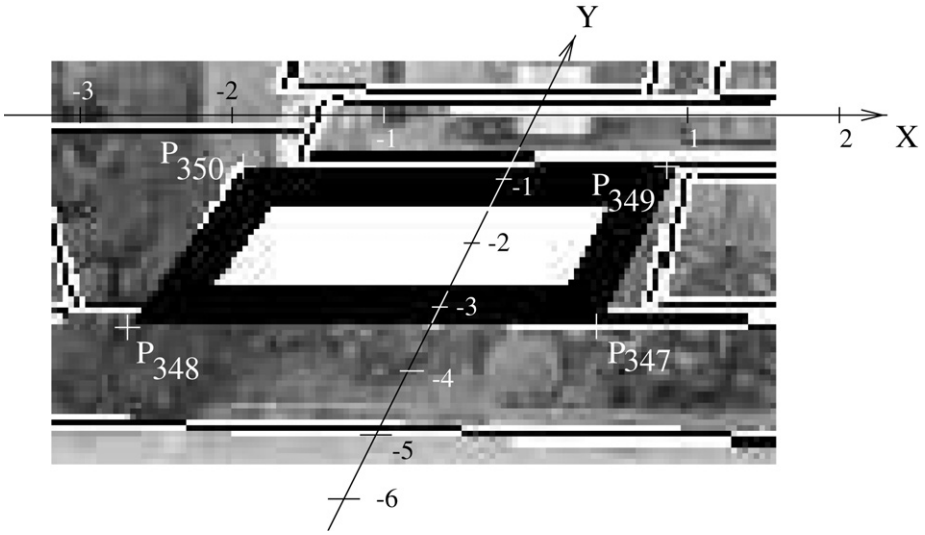


Fig. 6. Overlay of a suitable projection of the lane model from Fig. 5 onto one image of the gas station sequence illustrated in Fig. 1. Here: Enlargement of the area around the last filling place on the lower filling lane of the gas station (lane segment '1' in Fig. 5). In addition, the points p347–p350 and the scene coordinate system are shown.

lane data are imported into the conceptual representation subsystem by a set of *facts* and by FMTHL-rules making additional elementary properties about the lane geometry available in conceptual rather quantitative geometric terms. Points are referenced by their x/y-position in the scene and an unique identifier:

```
always point( 1.030, -3.352, p347).
always point(-1.870, -3.353, p348).
always point( 1.031, -0.852, p349).
always point(-1.871, -0.853, p350).
```

Again, all these data are assumed by the interpretation process to be absolutely valid (i.e. $\mu = 1$). Lines are described by the identifiers of the corresponding endpoints and their own identifier:

```
always line(p347, p348, l420).
always line(p349, p350, l421).
```

Lines are implicitly oriented from their first endpoint (e.g., 'p347') to their second ('p348'). In consequence, lines are enlisted as directed edges in Fig. 5.

Neighboring (more or less parallel) lines are combined into lane segments (e.g., lane segment '1' in Figs. 5 and 6):

```
always segment_of_lane(l421, l420, lseg_1).
```

Thus, each lane segment can be conceived as a convex polygon connecting the overall four endpoints of their two explicitly implemented lines. Neighboring lane segments can be combined to *lane objects* or, for short, *lanes*. For instance, lane segments '11, 3, 102, 4, 101, 1, and 15' are combined to a lane named 'lobj_230' by the *part_of*-Relation:

```
always lane_segment(lseg_11).
always lane_segment(lseg_3).
always lane_segment(lseg_102).
always lane_segment(lseg_4).
always lane_segment(lseg_101).
always lane_segment(lseg_1).
always lane_segment(lseg_15).

always part_of(lseg_11,  lobj_230).
always part_of(lseg_3,   lobj_230).
always part_of(lseg_102, lobj_230).
always part_of(lseg_4,   lobj_230).
always part_of(lseg_101, lobj_230).
always part_of(lseg_1,   lobj_230).
always part_of(lseg_15,  lobj_230).

always lane(lobj_230).
```

5. Generation of a primitive conceptual representation for time-dependent agent properties

The facts obtained from the data provided by the Computer Vision subsystem constitute the link between the Computer Vision and the Conceptual Representation subsystems. Some of these facts comprise quantitative numerical values for, e.g., velocity and positions of agents as in

614 ! has_status(object_4, 8.941049, 1.849823, 146.9672, 2.934614, −2.064247).

(see Section 4.1). This status information about the tracked vehicle 'object_4' provides the value range of an interpretation function for the following logical formula

```
has_speed(Agent, small)
```

which is a logical predicate relating the speed of an agent (e.g., 'object_4') to a discrete conceptual value `small`. To do so, one has to derive the Degree of Validity (DoV) with which the geometric value of the agent's velocity V is mapped to the discrete value `small`.

degreeOfValidity 5-ary predicate symbol. Its DoV corresponds to the function value of the trapezoidal function described by the last four arguments related to the first argument. The meta-predicate `sp` overwrites the DoV by the arithmetic expression of its argument, see [35].

```
always (degreeOfValidity(X,P1,P2,P3,P4) :-
        X >= P1 , X < P2 , Wert is (X - P1) / (P2 - P1) , sp(Wert) ;
        X >= P2 , X < P3 , sp(1.0) ;
        X >= P3 , X < P4 , Wert is (P4 - X) / (P4 - P3) , sp(Wert)).
```

Fig. 7 shows one example relating the geometric value for V to the discrete value `small` (and to others). The trapezoidal function μ_{small} can be conceived as logical interpretation function

$\mathcal{I}\{\text{associate\_speed(V, small)}\} = \mu_{small}(V)$.

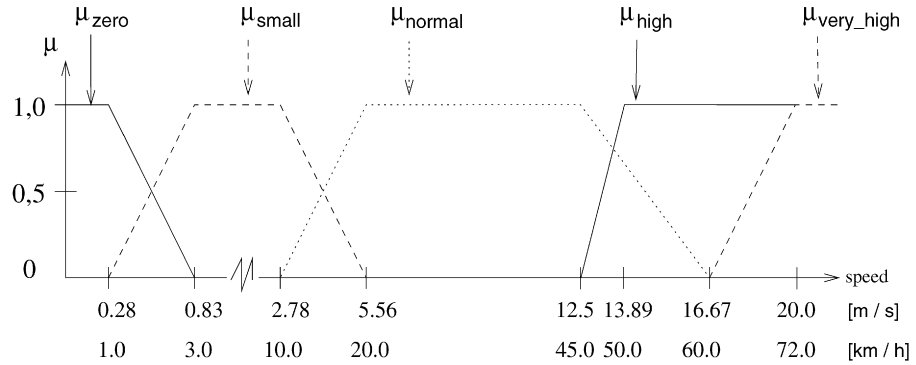


Fig. 7. Discretization of continuous speed values into a set of intervals. The upper part shows the fuzzy membership functions μ_{speed_value} for the subset { `zero, small, normal, high, very_high` } of discrete conceptual speed values.

To be more general, $\mu_{DISCRETE\_VALUE}(V)$ can be written as $\mu_{A,B,C,D}(V)$ where $A < B < C < D$ represent the arguments for which the slope of the trapezoidal function exhibits a discontinuity. According to Fig. 7, one obtains, for example,

$$\mu_{small}(V) = \mu_{0.28,0.83,2.78,5.56}(V)$$

or

$$\mu_{normal}(V) = \mu_{2.78,5.56,12.5,16.67}(V).$$

In the following, one possible implementation of this predicate in terms of an FMTHL-rule during the import of information from the Computer Vision subsystem into the Conceptual Representation subsystem is given:

```
always (has_speed(Agent, small) :- has_status(Agent,X,Y,Theta,V,Psi),
                                   associate_speed(V, small)).

always (associate_speed(V, small) :- degreeOfValidity(V,0.28,0.83,2.78,5.56)).
```

This corresponds to the following logical interpretation:

$$\mathcal{I}\{\texttt{has\_speed(Agent, small)}\}$$
$$== \mathcal{I}\{\texttt{has\_status(Agent,X,Y,Theta,V,Psi)} \wedge \texttt{associate\_speed(V,small)}\}$$
$$== \mathcal{I}\{\texttt{has\_status(Agent,X,Y,Theta,V,Psi)} \wedge \texttt{degreeOfValidity(V,0.28,0.83,2.78,5.56)}\}$$
$$== \min(\mathcal{I}\{\texttt{has\_status(Agent,X,Y,Theta,V,Psi)}\},$$
$$\mathcal{I}\{\texttt{degreeOfValidity(V,0.28,0.83,2.78,5.56)}\})$$

where the symbol == has been used to indicate that this operation refers to a numeric equality, i.e. neither to a logical equality operator nor to an assignment operator as it is used in many programming languages. The minimum function has been used as the fuzzy version of the conjunction as it is usually done.

Since `has_status` is not a fuzzy predicate (see Section 4.1), one obtains

$$\mathcal{I}\{\texttt{has\_speed(Agent, small)}\} == 0$$

provided

$$\mathcal{I}\{\texttt{has\_status(Agent,X,Y,Theta,V,Psi)}\} == 0,$$

which will occur at time points where no trajectory data for the agent is available (for example because the agent is currently not in the field of view of the recording camera). As a consequence, no geometric velocity value V for the agent will be available and thus no association to a discrete value can be performed. Otherwise, one obtains

$$\mathcal{I}\{\texttt{has\_speed(Agent, small)}\}$$
$$== \min(1, \mathcal{I}\{\texttt{degreeOfValidity(V,0.28,0.83,2.78,5.56)}\})$$
$$== \mathcal{I}\{\texttt{degreeOfValidity(V,0.28,0.83,2.78,5.56)}\}$$
$$== \mu_{0.28,0.83,2.78,5.56}(V).$$

As a result, geometric input data can be associated with discrete conceptual values by defining FMTHL predicates such as `has_speed(Agent, small)`. These predicates constitute the head of simple FMTHL rules whose body consists of the non-fuzzy `has_status`-predicate in order to access the geometric value and of the `degreeOfValidity`-predicate. The latter represents the compatibility of the value imported from the Computer Vision subsystem with the vagueness of the discrete concept `small` as expressed numerically by the trapezoidal function.

6. Occurrences

In general, already a short observation of a road vehicle allows a fairly good prediction regarding its subsequent motion during a few seconds. Such a prediction will become even more reliable if the movements of other traffic participants in its environment, in particular, of other road vehicles, and properties of the road can be taken into account. Since a reliable prediction of the behavior of other traffic participants is important for safe and smooth road traffic, it is no surprise that a highly specific vocabulary has been developed in order to communicate such behavior.

Already a short introspection will reveal that such communication can be characterized as either pertaining to the short-term movement of a single vehicle—possibly supplemented by reference to the road or to some other immediately relevant object—or to some abstraction referring to an entire, usually goal-directed, sequence of such short-term movements. We consider the first type as elementary and denote a recognizable movement primitive as an *occurrence*: this notion appears as sufficiently neutral to allow its potential extension to other types of movement primitives beyond those relating to road vehicle motion.

A systematic search for all verbs in a standard dictionary of the German language yielded a subset of about sixty verbs (from among about 9200) which relate to road vehicle movements (see [20,27]). Occurrences related to this subset can be categorized

(1) as *perpetuative* if they tend to retain the dominant aspect of a movement without major change,
(2) as *mutative* if they characterize the *systematic change of some aspect*, or
(3) as *terminative* if they relate to the *beginning* or *ending* of a dominant movement characteristic.

The algorithmic recognition of a particular occurrence will be presented here as based on logical inference although the approach exhibits obvious *analogies* to pattern recognition—*provided* the essential *time-dependencies are neglected*. Each occurrence can be characterized uniquely by a conjunction of predicates. These in turn consist of a conjunction of up to three (sub)predicates, namely

(1) a *PRE-Condition* (*PREC*) which has to be satisfied *before* the occurrence in question could be considered to represent a valid description of the temporal development in which the agent is involved;
(2) a *MONotonicity-Condition* (*MONC* or *MC*) indicating the type of admissible monotonous change which may take place while the occurrence represents a valid description;
(3) a *POST-Condition* (*POSTC*) which becomes true once the occurrence in question will no longer constitute an adequate description of the temporal development in which the agent is involved.

As will be seen shortly, evaluation of temporal relations between the validity of these three (sub)predicates are essential for a proper characterization of occurrences. In some cases, the monotonicity condition will be irrelevant and can be omitted, in other cases only the monotonicity condition will be relevant such that pre- and post-conditions could be omitted. In a fourth variant, the pre-condition remains true while an occurrence constitutes a valid description of temporal developments although attribute values may vary during this period, thereby preventing the satisfiability of a monotonicity condition. If the pre-condition is identical with the post-condition, pre- and post-condition are unified into a **COND**ition.

Table 3
Time-dependent (!) predicates defining occurrences which refer only to the agent. The definition of the predicate `has_speed` can be found in Appendix A.1.1, that of the predicate `has_direction` in Appendix A.1.3, and correspondingly in Appendix A.1.2 for `has_mode`

Occurrence	Type	`has_speed`			`has_direction`	`has_mode`
		PREC	MONC	POSTC	MONC	COND
accelerate	mut	moving	higher	moving	–	–
brake	mut	moving	smaller	moving	–	–
drive at constant speed	mut	moving	constant	moving	–	–
drive at regular speed	perp	normal	–	normal	–	–
drive fast	perp	high	–	high	–	–
drive forward	perp	–	–	–	–	forward
drive off	term	zero	higher	small	–	–
drive slowly	perp	small	–	small	–	–
drive straight ahead	mut	moving	–	moving	straight	–
drive very fast	perp	very_high	–	very_high	–	–
reverse	perp	–	–	–	–	backwards
stand	perp	zero	–	zero	–	–
stop	term	small	smaller	zero	–	–
turn left	mut	moving	–	moving	left	–
turn right	mut	moving	–	moving	right	–

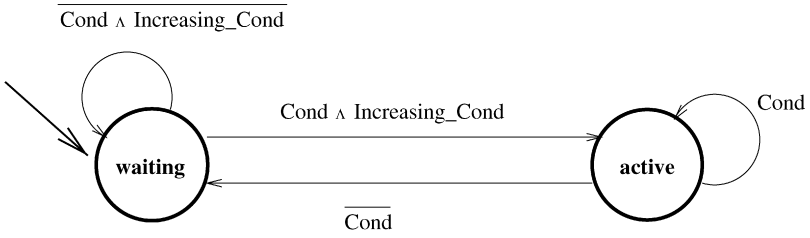


Fig. 8. Transductor for the recognition of *perpetuative* occurrences.

6.1. Occurrences which refer only to the agent

These are the simplest occurrences and will be treated first. They can be characterized by (conjunctions of) three *time-dependent* predicates which evaluate whether an attribute function takes on a particular (discrete) value or not. One such predicate relates to the agent's *speed*, one to the agent's *direction*, and one to the *mode* of agent motion—see Table 3. It has been discussed in Section 5 how predicates will be evaluated with respect to the validity of an attribute-value binding in case of a particular agent at a particular point in time: based on results provided by the core Computer Vision subsystem, logic formulae—which express schematic (a-priori) knowledge about what may happen in a road traffic scene—are *interpreted*.

6.2. Taking temporal dependencies into account: transductors *for occurrence recognition*

The a-priori knowledge about temporal relations between the satisfaction of equality attributes has been coded into the *type* of an occurrence (perpetuative, mutative, terminative) and into the way attribute values are required for the PREC, the MONC, and the POSTC. A finite state acceptance automaton (*transductor*) has been designed for each type. These transductors determine whether or not the required conditions are satisfied in the prescribed temporal order. Supplementary predicates which appear in FMTHL formulae specifying a transductor—like, e.g., `increasing_condition(Agent,Verb)`—are given in Appendix A.2.

6.2.1. Transductor for perpetuative *occurrences*

The transition diagram for this transductor is given in Fig. 8. This transductor is realized by the following fuzzy metric-temporal logic inferences (see Section 3 for an explanation of the logical vocabulary of FMTHL):

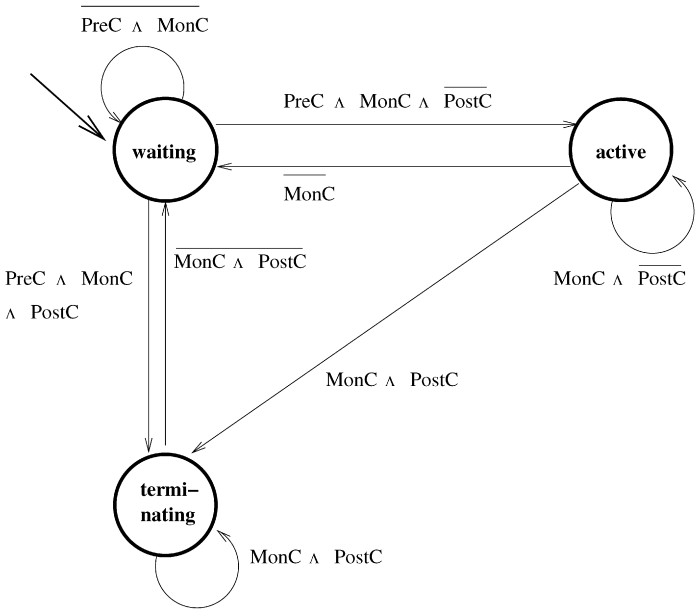


Fig. 9. Transductor for the recognition of *mutative* occurrences.

```
R1: always (perpetuative(Agent,Verb)  :- waiting_perp(Agent,Verb)).
R2: always (waiting_perp(Agent,Verb)  :- condition(Agent,Verb) ,
                                         increasing_condition(Agent,Verb), ! ,
                                         active_perp(Agent,Verb)).
R3: always (waiting_perp(Agent,Verb)  :- has_status(Agent,X,Y,Theta,V,Psi) ,
                                         1 ? waiting_perp(Agent,Verb)).
R4: always (active_perp(Agent,Verb)   :- condition(Agent,Verb) , ! ,
                                         output(DoV, Verb, Agent) ,
                                         1 ? active_perp(Agent,Verb)).
R5: always (active_perp(Agent,Verb)   :- has_status(Agent,X,Y,Theta,V,Psi) ,
                                         1 ? waiting_perp(Agent,Verb)).
```

The heads of the rules (implications) correspond to the states of the automaton. Their bodies comprise—in addition to conditions according to each state—the name of the (possible) successor state. Evaluation starts in rule `R1`. Rule `R2` tests (according to the state `waiting` of the automaton), whether a certain *condition* is currently valid and whether its Degree of Validity (DoV) increases during five consecutive time points. In this case, the automaton switches into the `active` state (`R4`–`R5`). Otherwise, provided additional trajectory data are available, the automaton remains `waiting` (`R3`).

Perpetuative occurrences which refer only to the agent can be evaluated just by using attributes with agent reference (speed, direction, and mode). The entry for `drive_slowly` in Table 3 can be transformed into an FMTHL implication according to the following example:

```
always (condition(Agent,drive_slowly) :-  has_speed(Agent,small)).
```

In this case, the formulation of a single condition can be extracted directly from Table 3 in analogy to explanations in Section 5. Evaluation is started by a simple logical query, for example, in the case of `drive_slowly`:

```
?- perpetuative(Agent, drive_slowly).
```

6.2.2. Transductor for mutative *occurrences*

See Section 3 for an explanation of the logical vocabulary of FMTHL.

```
always (mutative(Agent,Verb)    :- waiting_mut(Agent,Verb)).
always (waiting_mut(Agent,Verb) :- precondition(Agent,Verb) ,
                                   mon_condition(Agent,Verb) ,
                                   postcondition(Agent,Verb) , ! ,
                                   terminating_mut(Agent,Verb)).
always (waiting_mut(Agent,Verb) :- precondition(Agent,Verb) ,
                                   mon_condition(Agent,Verb) ,
                                   active_mut(Agent,Verb)).
always (waiting_mut(Agent,Verb) :- has_status(Agent,X,Y,Theta,V,Psi) ,
                                   1 ? waiting_mut(Agent,Verb)).
always (active_mut(Agent,Verb)  :- mon_condition(Agent,Verb) ,
                                   postcondition(Agent,Verb) ,
                                   terminating_mut(Agent,Verb)).
always (active_mut(Agent,Verb)  :- mon_condition(Agent,Verb) ,
                                   1 ? active_mut(Agent,Verb)).
always (active_mut(Agent,Verb)  :- has_status(Agent,X,Y,Theta,V,Psi) ,
                                   1 ? waiting_mut(Agent,Verb)).
always (terminating_mut(Agent,Verb)
                                :- postcondition(Agent,Verb) ,
                                   mon_condition(Agent,Verb) ,
                                   output(DoV, Verb, Agent) ,
                                   1 ? terminating_mut(Agent,Verb)).
always (terminating_mut(Agent,Verb)
                                :- has_status(Agent,X,Y,Theta,V,Psi) ,
                                   1 ? waiting_mut(Agent,Verb)).
```

Pre-, monotonicity- and postcondition have to be evaluated in order to determine the validity of `driving straight ahead`. These conditions can be derived directly from Table 3:

```
always (precondition(Agent,drive_straight_ahead)  :-  has_speed(Agent,moving)).

always (mon_condition(Agent,drive_straight_ahead) :-  has_direction(Agent,straight)).

always (postcondition(Agent,drive_straight_ahead) :-  has_speed(Agent,moving)).
```

6.2.3. Transductor for terminative *occurrences*

See Section 3 for an explanation of the logical vocabulary of FMTHL.

```
always (terminative(Agent,Verb)  :- waiting_term(Agent,Verb)).
always (waiting_term(Agent,Verb) :- precondition(Agent,Verb) , ! ,
                                    ready_term(Agent,Verb)).
always (waiting_term(Agent,Verb) :- has_status(Agent,X,Y,Theta,V,Psi) , 1 ?
                                    waiting_term(Agent,Verb)).
always (ready_term(Agent,Verb)   :- mon_condition(Agent,Verb) ,
                                    1 ? active_term(Agent,Verb)).
always (ready_term(Agent,Verb)   :- precondition(Agent,Verb) , ! ,
                                    1 ? ready_term(Agent,Verb)).
always (ready_term(Agent,Verb)   :- has_status(Agent,X,Y,Theta,V,Psi) ,
                                    1 ? waiting_term(Agent,Verb)).
always (active_term(Agent,Verb)  :- postcondition(Agent,Verb) , ! ,
                                    1 ? terminating_term(Agent,Verb)).
always (active_term(Agent,Verb)  :- mon_condition(Agent,Verb) , ! ,
                                    has_status(Agent,X,Y,Theta,V,Psi) ,
                                    1 ? active_term(Agent,Verb)).
always (active_term(Agent,Verb)  :- waiting_term(Agent,Verb)).
always (terminating_term(Agent,Verb)
```

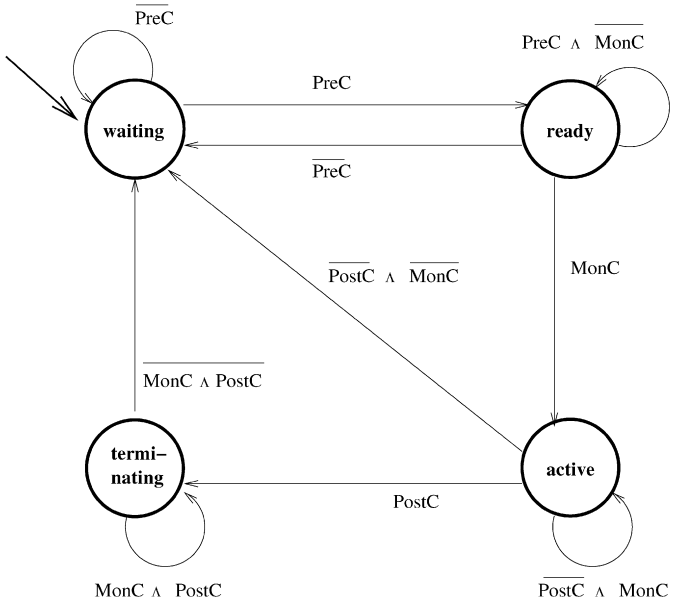


Fig. 10. Transductor for the recognition of *terminative* occurrences.

```
                                   :- postcondition(Agent,Verb) ,
                                      mon_condition(Agent,Verb) , ! ,
                                      output(DoV, Verb, Agent) ,
                                      1 ? terminating_term(Agent,Verb)).
always (terminating_term(Agent,Verb)
                                   :- has_status(Agent,X,Y,Theta,V,Psi) , 1 ?
                                      waiting_term(Agent,Verb)).
```

6.3. Occurrences which refer to agent and location

Table 4 presents—in analogy to Table 3—the definition of predicates which together characterize occurrences referencing both the agent and a specified location in the road traffic scene.

In order to compute occurrences with reference to a location, the particular supplementary argument has to be integrated into the inference rules for an occurrence. For example, in the case of a perpetuative occurrence with location reference, the rule `R2` mentioned in Section 6.2.1

```
R2: always (waiting_perp(Agent,Verb) :- condition(Agent,Verb) ,
                                        increasing_condition(Agent,Verb), ! ,
                                        active_perp(Agent,Verb)).
```

has to be changed into

```
R2_L: always (waiting_perp(Agent,Location,Verb)
                                     :- condition(Agent,Location,Verb) ,
                                        increasing_condition(Agent,Location,Verb), ! ,
                                        active_perp(Agent,Location,Verb)).
```

6.4. Occurrences which refer to the agent and an additional object

Table 5 presents—in analogy to Table 3—the definition of predicates which together characterize occurrences referencing both the agent and a *stationary* object in the road traffic scene. The definition of predicates for the case of an additional *moving* object can be found in Table 6.

Table 4
Time-dependent (!) predicates defining occurrences which refer to both the agent and a location. The symbol '>' indicates a *decreasing* slope for the value subject to the monotonicity condition MONC, the symbol '<' correspondingly an *increasing* slope. `has_course` denotes the abbreviation of the predicate `has_course_towards_loc`, see Appendix A.1.4. Similarly, `has_distance` stands for the predicate `has_distance_to_loc`, see Appendix A.1.5

Occurrence	Type	has_speed(Agent)			has_course(Agent,Location)			has_distance(Agent,Location)		
		PREC	MONC	POSTC	PREC	MONC	POSTC	PREC	MONC	POSTC
arrive at loc	term	moving	–	moving	–	–	–	small	>	zero
depart from loc	term	moving	–	moving	–	–	–	zero	<	small
drive to loc	mut	moving	–	moving	appr.	–	appr.	not_zero	>	small
leave loc	term	zero	<	moving	–	–	–	zero	–	–
leave loc behind	mut	moving	–	moving	leaving	–	leaving	small	<	not_zero
park at loc	perp	zero	–	zero	–	–	–	zero	–	zero
pass loc	term	moving	–	moving	appr.	changing	leaving	not_zero	–	not_zero
run over loc	perp	moving	–	moving	–	–	–	zero	–	zero
stop at loc	term	moving	>	zero	–	–	–	–	–	zero

Table 5
Time-dependent (!) predicates defining occurrences which refer to both the agent and another *stationary* object. The predicate referred to by the column heading `have_course` is treated in Appendix A.1.7. In addition to the predicates enumerated in this table, `has_speed(Object, zero)` always has to be true. See caption of Table 4 regarding the symbols '<' and '>'

Occurrences	Type	has_speed(Agent)			have_course(Agent,Object)			have_distance(Agent,Object)		
		PREC	MC	POSTC	PREC	MC	POSTC	PREC	MC	POSTC
be standing near	perp	zero	–	zero	–	–	–	small	–	small
collide with	term	moving	–	zero	–	–	–	small	>	zero
drive past	mut	moving	–	moving	passing	constant	passing	small	–	small
merge in front of	term	moving	–	moving	passing	changing	leaving	small	–	small
move away from	mut	moving	–	moving	leaving	–	leaving	small	<	–
move towards	mut	moving	–	moving	appr.	–	appr.	small	>	–
pull out behind	term	moving	–	moving	appr.	changing	pass.	small	–	small
start in front of	term	zero	<	moving	–	–	–	small	–	small
stop behind	term	small	>	zero	–	–	–	small	–	small

In order to compute occurrences with additional reference to an object, the particular supplementary argument has to be integrated into the inference rules for an occurrence. For example, in the case of a perpetuative occurrence with object reference, the rule `R2` mentioned in Section 6.2.1

```
R2: always (waiting_perp(Agent,Verb) :- condition(Agent,Verb) ,
                                        increasing_condition(Agent,Verb), ! ,
                                        active_perp(Agent,Verb)).
```

has to be changed into

```
R2_O: always (waiting_perp(Agent,Object,Verb)
                                    :- condition(Agent,Object,Verb) ,
                                       increasing_condition(Agent,Object,Verb), ! ,
                                       active_perp(Agent,Object,Verb)).
```

In this case, the characterization of an occurrence depends on whether the additional object remains stationary or moves itself.

6.5. Occurrences which refer to agent and lane

Table 7 presents—in analogy to Table 3—the definition of predicates which together characterize occurrences referencing both the agent and a lane.

Table 6
Time-dependent (!) predicates defining occurrences which refer to both the agent and another *moving* object. The predicate `have_difference_in_orientation` is treated in Appendix A.1.8 and the predicate `have_distance` in Appendix A.1.6. The spatial relation predicate appearing in the last two columns headed by 'relative_position' and 'configuration' are treated in Appendix A.1.9. In addition to the predicates enumerated in this table, `has_speed(Agent, moving)` and `has_speed(Object, moving)` always have to be true

Occurrence	Type	have_difference_in_orientation			have_distance			relative_position		Configuration	
		PREC	MC	POSTC	PREC	MC	POSTC	PREC	POSTC	PREC	POSTC
approach crossing	mut	crossing	–	crossing	normal	>	small	–	–	–	–
approach oncoming	mut	opposite	constant	opposite	normal	>	small	–	–	–	–
catch up with	mut	equal	constant	equal	normal	>	small	behind	behind	straight	straight
close up to	mut	crossing	–	equal	–	<	–	–	behind	–	–
cut in front of	mut	crossing	–	–	small	>	zero	front	front	–	–
draw ahead of	mut	equal	constant	equal	–	<	–	front	front	–	–
drive in front of	perp	equal	–	equal	–	–	–	front	front	straight	straight
fall behind	mut	equal	constant	equal	small	<	normal	behind	behind	straight	straight
flank	mut	equal	constant	equal	–	–	–	beside	beside	–	–
follow	perp	equal	–	equal	–	–	–	behind	behind	straight	straight
get out of the way of	mut	equal	constant	equal	small	–	–	behind	beside	–	–
leave crossing	mut	crossing	–	crossing	small	<	–	–	–	–	–
leave oncoming	mut	opposite	constant	opposite	small	<	–	–	–	–	–
let run into	mut	equal	constant	equal	small	>	zero	front	front	straight	straight
lose a lead on	mut	equal	constant	equal	normal	>	small	front	front	straight	straight
make way	mut	opposite	constant	opposite	small	–	–	behind	beside	–	–
merge in front of	mut	crossing	changing	equal	small	–	–	–	front	–	straight
move past	mut	equal	constant	equal	–	–	–	beside	front	–	–
pass	mut	equal	constant	equal	–	<	–	front	front	straight	straight
pull up to	mut	equal	constant	equal	small	–	small	beside	beside	half-left	left
run into	mut	equal	constant	equal	small	>	zero	behind	behind	straight	straight
slip in front of	mut	equal	constant	equal	small	<	–	–	front	half-left	straight
swing out	mut	equal	constant	equal	small	>	–	behind	–	straight	half-left

In order to compute occurrences with additional reference to a lane, the particular supplementary argument has to be integrated into the inference rules for an occurrence. For example, in the case of a perpetuative occurrence with lane reference, the rule `R2` mentioned in Section 6.2.1

```
R2: always (waiting_perp(Agent,Verb) :- condition(Agent,Verb) ,
                                        increasing_condition(Agent,Verb), ! ,
                                        active_perp(Agent,Verb)).
```

has to be changed into

```
R2_L: always (waiting_perp(Agent,Lane,Verb)
                                     :- condition(Agent,Lane,Verb) ,
                                        increasing_condition(Agent,Lane,Verb), ! ,
                                        active_perp(Agent,Lane,Verb)).
```

These occurrences—see Table 7—are specified in analogy to the ones treated in preceding subsections.

7. Results

Results will be illustrated for two different kinds of road traffic, namely for vehicles crossing an inner-city intersection and for vehicles maneuvering at a gas station. The results will be presented as graphs of Degree of Validity (DoV) (see Section 3) versus frame-number, i.e. (discretized) time, for selected occurrences.

Please note that the DoV of an occurrence at a particular frame-number can imply a small temporal non-locality due to temporal derivatives which may have been incorporated into the definition of an occurrence. In any case, the DoV-value obtained for each frame-number should be looked at as an individual experiment which can be assessed

Table 7
Time-dependent (!) predicates defining occurrences which refer to both agent and *lane*. The predicates `agent_residence, l_element` and `lane_ref` are treated in Appendix A.1.10

Occurrence	Type	has_speed		has_direction	l_element	agent_residence		lane_ref
		PREC	POSTC	MC	MC	PREC	POSTC	MC
change lane	mut	moving	moving	straight	changing	on	on	–
cross a lane	perp	moving	moving	–	equal	–	–	across
drive on lane	perp	moving	moving	–	equal	–	–	along
travel on lane	perp	moving	moving	–	–	–	–	along
turn	mut	moving	moving	not_straight	changing	on	on	–

Fig. 11. Representative image frames from the sequence dtneu05 (left: frame 20; right: frame 400). The trajectory including a model projection of vehicle1 is superimposed to the image frame in the left panel. The right panel comprises trajectories and model projections for vehicle2 through vehicle5 (from [22]).

as being acceptable or not. In this manner, the reader may form his own assessment rather than being confronted with potentially highly aggregated assessments derived by test persons. As a byproduct of this way of presentation, the sensitivity of DoV-results for a selected occurrence and correlations between consecutive DoV-values in a subsequence of frames can be taken into account. In addition, the relation between certain problems in vehicle tracking (like partial occlusion of a vehicle) and the resulting DoV-results can be detected and assessed.

7.1. Selected occurrences for vehicles crossing an intersection

Two representative frames from the image sequence dtneu05 recorded at the Durlacher-Tor-Platz in Karlsruhe are shown in Fig. 11.

The left panel of Fig. 12 plots the DoV for two potential occurrence associations with the visible trajectory segment of vehicle1 in Fig. 11(left). Most of the time, the DoV is practically 1 for the occurrence `drive_at_constant_speed`, i.e. the system considers this occurrence as an appropriate characterization of the behavior of this vehicle. The right panel of Fig. 12 shows plots of the DoV for the occurrences `drive_straight_ahead`, `turn_left` and `turn_right` associated with the same trajectory segment. Apart from the initial phase when model-based vehicle tracking had not yet stabilized sufficiently, the correct alternative `drive_straight_ahead` has the highest DoV, justifying the use of this occurrence in order to describe the behavior of this vehicle during the tracked period. Either one or both of these high-DoV occurrences could be chosen, depending on the aspect of vehicle behavior to be emphasized.

The analogous plots in Fig. 13 for vehicle2 from Fig. 11(right) cover a larger temporal interval during which this vehicle turns left before continuing straight ahead again. The characterization `drive_at_constant_speed` dominates most of the time although the alternatives occasionally exhibit a higher DoV during the turning phase. The directional aspect of the maneuver performed by this vehicle during the tracking period is shown in the right panel of

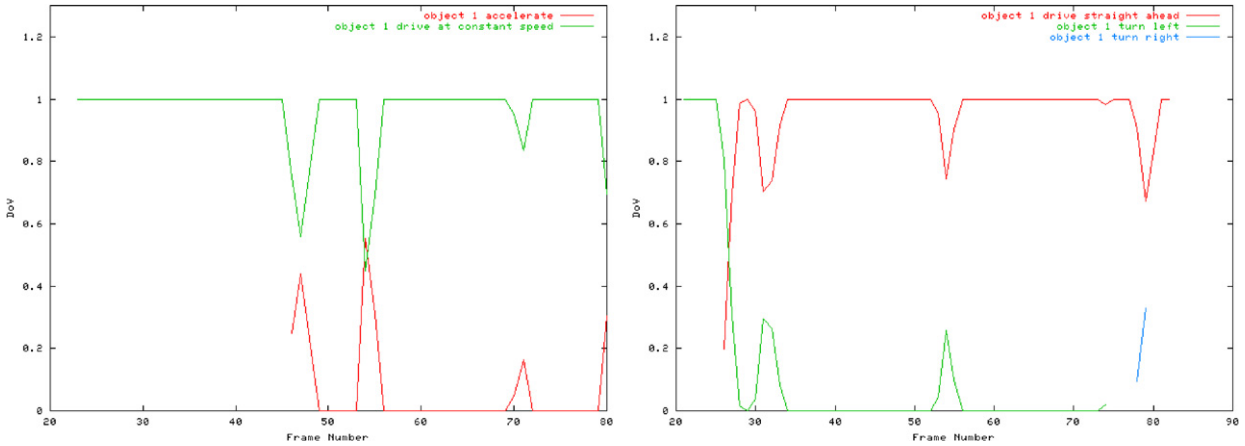


Fig. 12. (Left panel) The `Degree of Validity (DoV)` for the occurrences `accelerate` and `drive_at_constant_speed` plotted as a function of frame-number for vehicle1 in Fig. 11(left). (Right panel) Analogously for the occurrences `drive_straight_ahead`, `turn_left` and `turn_right`.

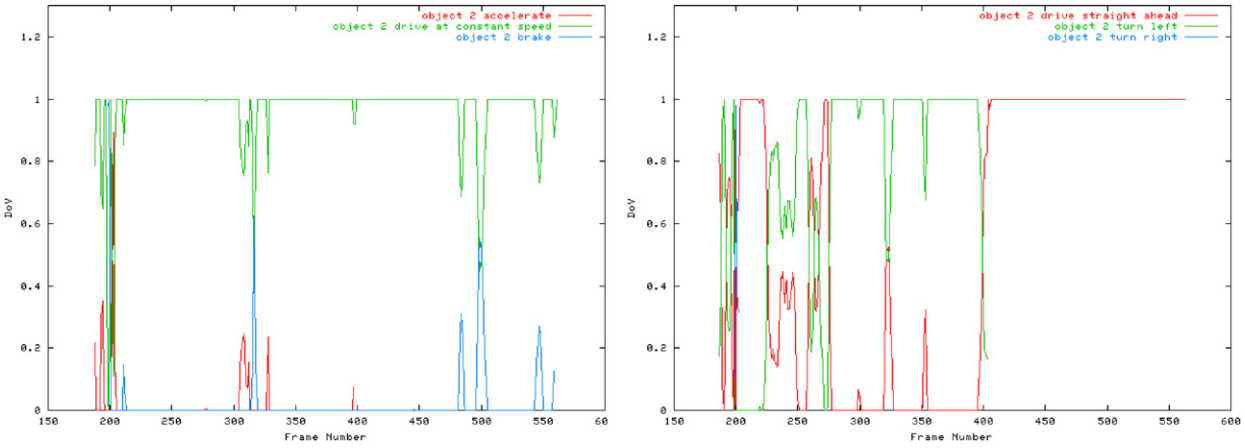


Fig. 13. (Left panel) The `Degree of Validity (DoV)` for the occurrences `accelerate`, `drive_at_constant_speed`, and `brake` plotted as a function of frame-number for vehicle2 in Fig. 11(right). (Right panel) Analogously for the occurrences `drive_straight_ahead`, `turn_left` and `turn_right`.

Fig. 13. Following some initial oscillations, `turn_left` dominates between frame-numbers 280 and 400 after which `drive_straight_ahead` clearly takes over as the most appropriate characterization of directional behavior for this vehicle.

Results mostly analogous to those obtained for vehicle2 are shown in Fig. 14 for vehicle3. It should be noted that this vehicle is occluded quite severely by the advertisement column during the first part of its trajectory. This fact shows up by noticeable oscillations of the DoV associated with alternative occurrence associations during this period although the DoV-value stabilizes again once the vehicle can be tracked without significant occlusions.

Results for vehicle5 are similar to those for vehicle1, with occasional oscillations due to occlusion by a mast, the advertisement column, and another mast in the center of the image frame. Tracking of vehicle5 had been initialized already prior to its (short-time partial) occlusion by the mast in the right upper quadrant of Fig. 11(right). It drove faster than the turning vehicle3 and thus its partial occlusion by the top of the advertisement column did not affect the occurrence associations to the same degree as in the case of vehicle3. Results for vehicle4 are similar to those shown for vehicle5 and thus have been omitted.

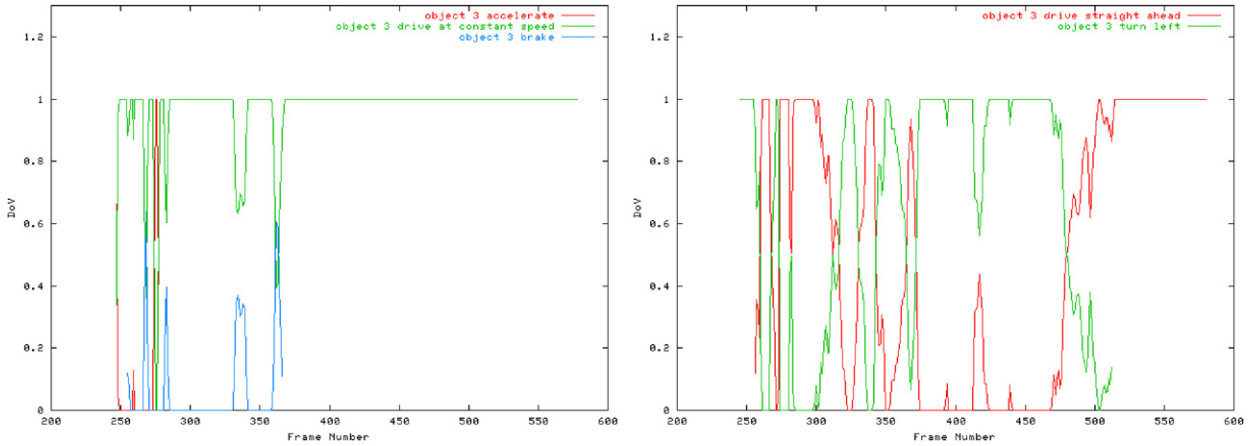


Fig. 14. Analogous to Fig. 13, but for vehicle3. Note that the occurrence `turn_right` never acquired a DoV greater than zero and thus has not been plotted in the right panel.

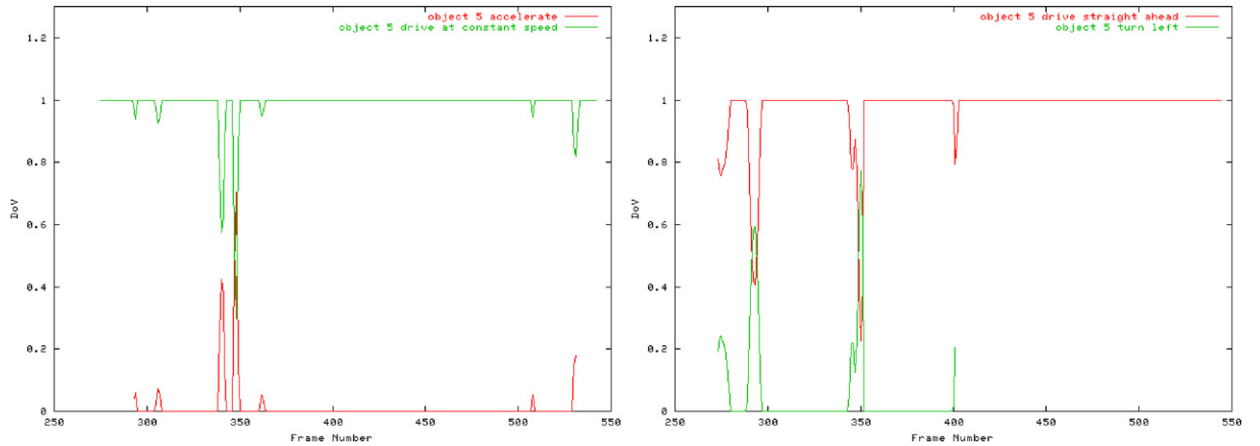


Fig. 15. Analogous to Fig. 12, but for vehicle5 (without DoV-values for `turn_right`).

7.2. Graphical illustrations of selected occurrences for the gas station sequence

Fig. 1 shows four images of a scene recorded at a gas station. The whole video sequence comprises about 8.000 frames or about 160 seconds (this corresponds to a scan rate of 50 frames per second). Seven vehicles were tracked successfully which perform complex maneuvers during the recording period. Here, we concentrate on the maneuvers performed by 'object_4' during this sequence (see Fig. 1). A sample of geometric results obtained by the computer vision system has already been presented in Section 4.1.

The names of lanes and filling places in the gas station sequence are explained in the groundplan map of the gas station (see Fig. 2).

The two bottom panels of Fig. 1 sketch the maneuvers of 'object_4' during the recording period illustrated by representative image frames in the top four panels. Trajectory data obtained by the model-based tracking system `Xtrack` have been associated with occurrences as described in the preceding sections. Fig. 16 plots the Degree of Validity DoV for the occurrences

1. `be standing near`,
2. `drive past`,
3. `move away from`, and
4. `move towards`

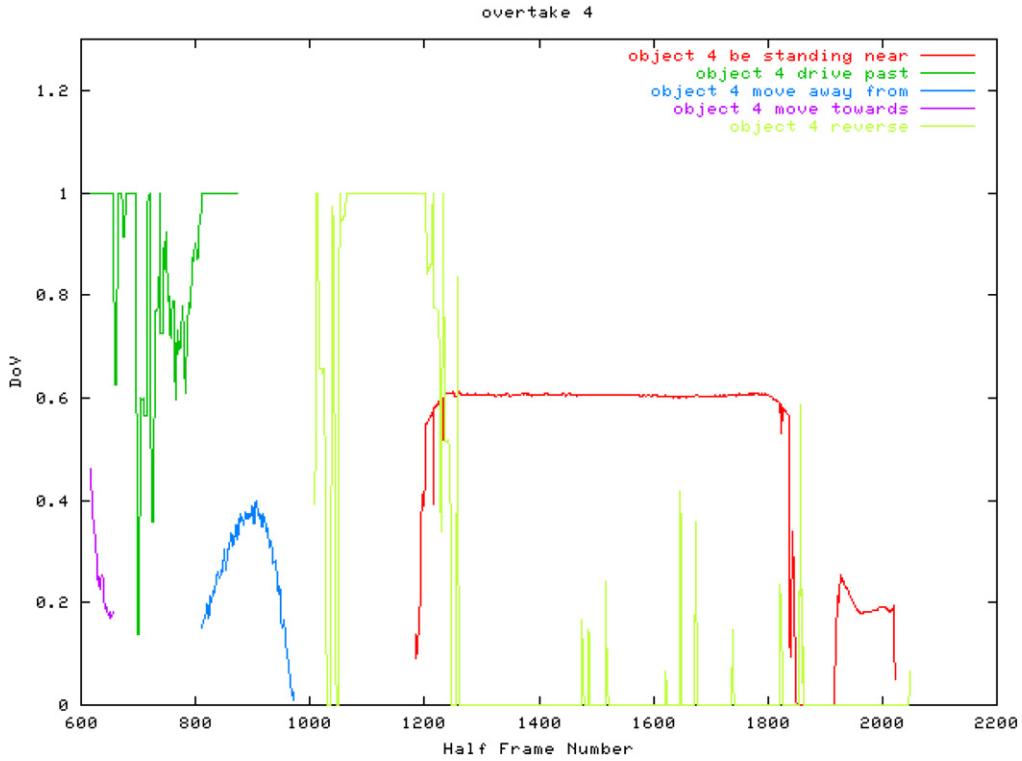


Fig. 16. The (green) curve starting at frame 600 with DoV = 1.0 represents the degree of validity for the occurrence 'object_4 `drive past` object_1'. The (violet) curve in the lower left corner of this graph indicates the diminishing DoV for the occurrence 'object_4 `move towards` object_1'. Between frames 800 and about 970, the occurrence 'object_4 `move away from` object_1' (represented by the blue curve) reaches a local maximum with DoV≈ 0.4 which nicely supplements the two other occurrences for a *detailed* description of 'object_4 overtakes object_1'. The `backup-maneuver` of object_4 between frames 970 and about 1200 is indicated by the subsequent (light-green) curve. The subsequent occurrence 'object_4 `be standing near` object_1' (red curve) shows up rather clearly between frames 1200 and about 1850. Then, object_1 starts to move backwards (see Fig. 17). Therefore, `be standing near` is no longer valid (since `be standing near` is only defined when both vehicles are standing) and its DoV decreases to zero. Later, around frame 1950, object_1 stops which means that from now on `be standing near` becomes valid again, but now with a lower DoV, since now the distance between object_4 and object_1 is significantly higher.

(see Table 5) as a function of frame number, i.e. of time. The initial maneuvers where object_4 passes object_1 can be recognized by the complementary variations of the DoV for the occurrences `move towards`, `drive past`, and `move away from` between frames 600 and about 970. After a short backward motion the subsequent occurrence `be standing near` shows up unmistakably through the approximately constant value of ≈ 0.6 for the associated DoV between frames 1200 and about 1850.

Analogous results obtained for object_1 are illustrated in Figs. 18 and 19. The maneuvers of object_1 are sketched in Fig. 17.

8. Related publications and concluding discussions

A *systems approach* has been outlined for the algorithmic transformation of video signals into a natural language text describing recorded vehicle maneuvers in road traffic scenes. Elaborating a part of this framework, the discussion focuses on a detailed presentation of steps which transform the geometric results for 3D-model-based vehicle tracking obtained by a computer vision subsystem into *conceptual* representations for movements of a single road vehicle (*occurrences*). An occurrence representation is associated with the 'Conceptual Primitives Level' of Fig. 3. The system-internal representation of about sixty occurrences for the description of vehicular road traffic has been specified. This is an attempt to cover all occurrences relevant for the description of motor vehicles on roads explicitly, thus facilitating the treatment of a broad range of road traffic image sequences. In addition, it should ease the discrimination between cases where a performance insufficiency may be attributed to the lack of a suitable occurrence definition, to

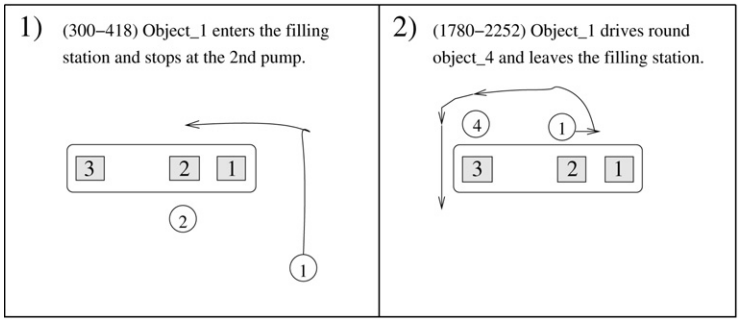


Fig. 17. Maneuver of object_1 from the gas station sequence.

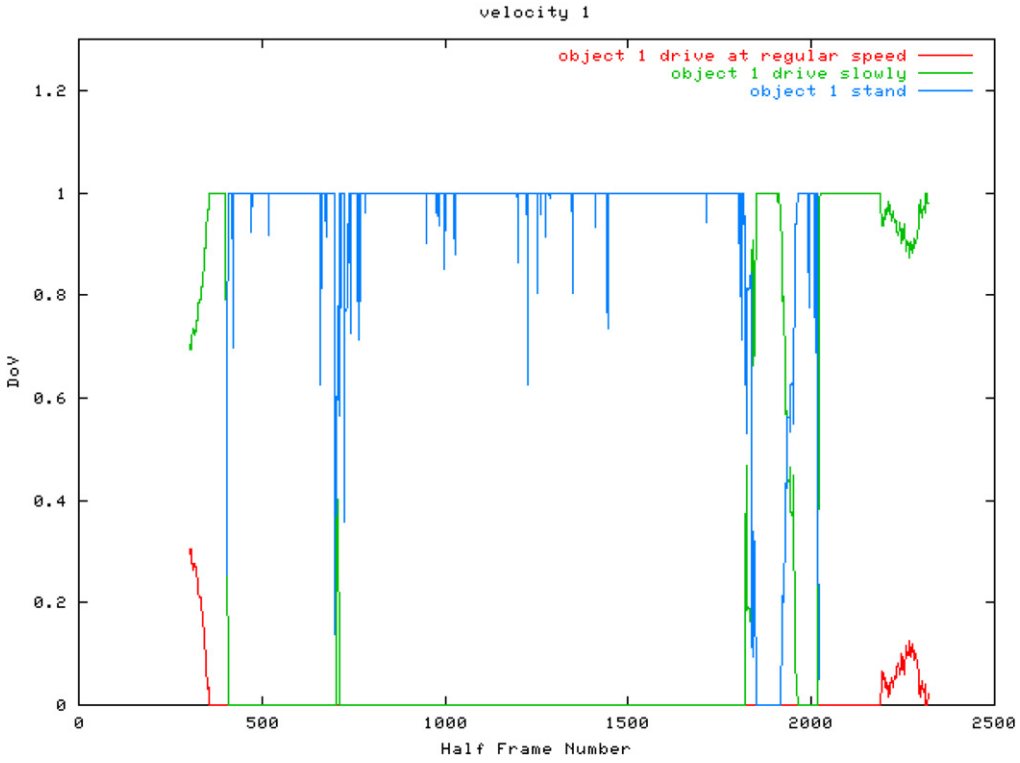


Fig. 18. While driving towards the second pump of the gas station (see Fig. 17), object_1 permanently reduces its velocity. Starting at frame 300, the DoV of the occurrence `drive at regular speed` thus continuously decreases from approximately 0.3 to zero (red curve) whereas the DoV of `drive slowly` increases from approximately 0.7 to 1.0 (green curve). Then, object_1 remains standing (blue curve) until frame 1780. Between frames 1800 and 1950 the vehicle starts to back up which is indicated by the decrease of the DoV of `stand` and the increase of `drive slowly`. Between frames 1950 and 2050, the opposite performance can be observed. Here, the vehicle `stands` again in order to change mode. Then it `drives slowly` again, now in forward direction. It permanently accelerates whereby `drives at regular speed` becomes valid again. See Table 3 for the definition of the occurrences displayed here.

an inappropriate parameterization, or to an implementation error. The price to be paid for such advantages consists in the attention and space to be devoted to the required details.

A fuzzy metric-temporal logic has been chosen as the basis for the creation and manipulation of a system-internal representation of the temporal developments within the recorded scene. This choice obviates the necessity to learn, approximate, or postulate a large number of probability distributions which are required for a probabilistic inference approach providing the same coverage. So far, we did not encounter experimental evidence that the fuzzy membership functions explicated here caused difficulties. These membership functions are intuitively acceptable and can be easily adapted to special requirements if that should become necessary. In addition, the same fuzzy metric-temporal inference

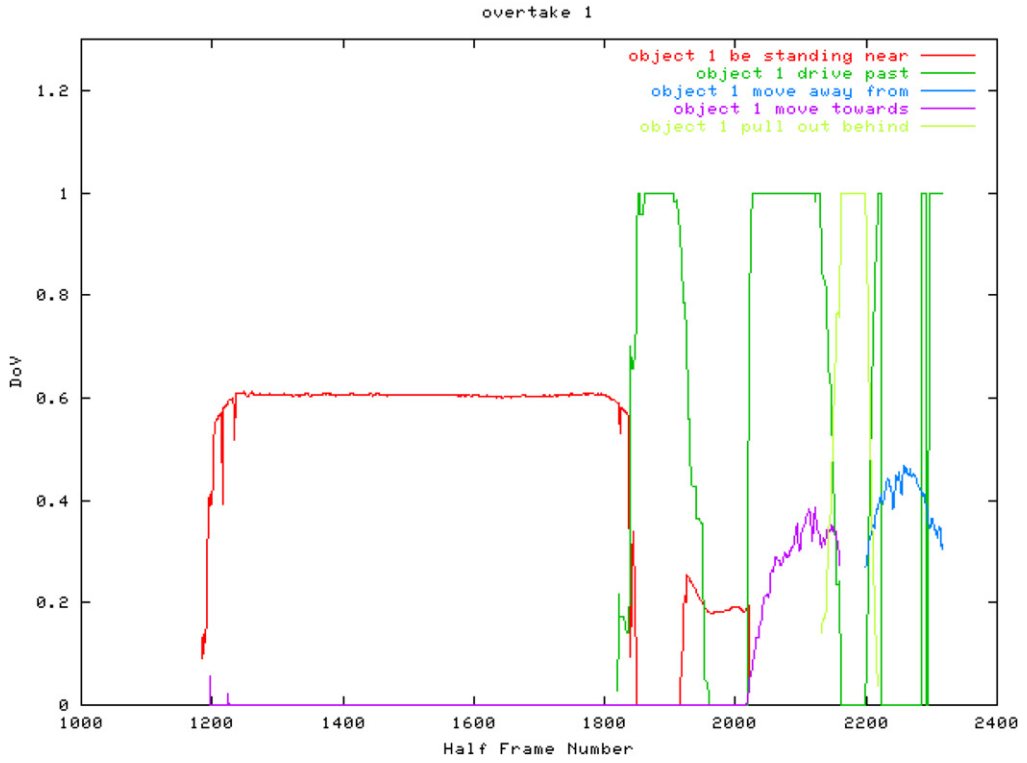


Fig. 19. This figure illustrates object_1 overtaking object_4 (see, as well, Fig. 17). The illustration starts at frame 1200 where object_1 is `standing near` object_4 (red curve) until frame 1800. Then object_1 starts to back up and `drive past` object_4 (green curve). Then, object_1 comes to a stop, which means that the DoV of `drive past` is reduced to zero and `standing near` is becoming valid again, with a smaller value for DoV because of the now greater distance between both vehicles. Object_1 then changes mode and begins to `move towards` object_4 (violet curve). The (light green) curve starting at frame 2150 represents the degree of validity for the occurrence 'object_1 `pulls out behind` object_4' which decreases at frame 2200, where object_1 starts to `move away from` object_4 (blue curve). For a detailed description of the occurrences mentioned here, see Table 5.

mechanism has been used uniformly throughout the Conceptual Representation Level. This fact together with the overall system approach facilitates to trace down insufficiencies in system performance and to provide remedies in the most appropriate component.

Parameters have been separated from the algorithmic steps proper by definition and instantiation of occurrences *based on formal logic*. This should facilitate an analysis of the approach, in particular regarding its potential extension to various natural languages. So far, it can be used equally well for occurrences formulated in English and German.

The required processing steps have been illustrated by results obtained for vehicles crossing an inner-city intersection and for vehicle maneuvers at a gas station. Altogether, thousands of frames have been analyzed in this manner with the conclusion that our algorithmic approach yields results generally compatible with our judgment. So far, remaining discrepancies can be attributed to difficulties (detection, initialization, and tracking of vehicle images) in the Computer Vision subsystem.

It may have been noted that an occurrence can be mapped to a *verb phrase* in the linguistic sense. Normally, the subject for such verb phrases will be the vehicle of interest which we denote as *agent*. It thus is possible to formulate a simple 'single-sentence text' which comprises the agent as subject and a suitably conjugated verb phrase derived from an occurrence representation which characterizes a movement primitive in isolation. The analogy of such an approach to case-based single sentence text understanding [8] should be obvious. The ability to generate simple one-sentence texts which describe particular aspects of a vehicle's motion is expected to help tracking down system shortcomings in a focused manner.

We refrain from claims that this formalism can be extended to cover other occurrences because we have not yet accumulated sufficient experience in this direction. It appears more important for us at the moment to improve the

detection and tracking of road vehicles. Such tracking results can then be exploited to subject this approach to more encompassing tests.

Given the preceding expositions, several remarks concerning earlier publications by other authors may clarify similarities of and differences between approaches. Publications which appeared prior to the year 2000 will be considered to be accessible via introductory sections of more recent publications, in particular of recent surveys or special issues [3,4,12,34]. Such earlier publications thus will not be discussed generally in what follows. Similarly, we shall refrain in general from discussing the treatment of *human movements*.

8.1. Publications linking traffic videos to conceptual representations

Research by Caelli and coworkers [5,6] should be mentioned here as an early combination of information extraction from real-world intersection traffic images and knowledge-based approaches. A rectangular representation for vehicle images was extracted by background subtraction and clustering of neighboring change pixels. Feature-based matching (centroid coordinates and orientation of enclosing rectangles) were used to establish correspondences between three image frames recorded at 400 msec time intervals in order to obtain estimates for velocity and orientation changes. The vehicle image descriptors were then exploited to interpret a-priori knowledge about typical movements and behavior of vehicles at inner-city road intersections. This knowledge relied on a frame-based representation in a kind of semantic network in combination with rule-based evaluation of predicates.

Remagnino et al. [33] report on a hybrid approach towards the generation of textual descriptions based on the evaluation of videos recorded at parking lots. A data-driven technique is used to track (isolated) persons in the image plane. Knowledge about the pose of the recording stationary camera with respect to the ground plane then enables the authors to estimate the 3D-scene position of the tracked person in order to relate this to vehicle positions in the scene. No details are given about how vehicle positions in the scene have been determined. A Bayesian Belief Network representation of a small number of behaviors associates pedestrian position and movements to textual formulations of a kind to be expected from an automated video-based parking-lot surveillance system.

Howarth and Buxton [17] derive a small set of both primitive and abstracted events recorded at a roundabout in the street traffic domain by means of a model-based tracking approach. Fernyhough et al. [7] study the derivation of event models from object movements using a data-driven *learning* approach—see, too, [9]. Simple events like overtaking processes can be identified by evaluation of neighborhood relations between image segments.

Liu et al. [24] derive textual descriptions from recorded traffic scenes. Fuzzy membership functions are used to derive conceptual primitives from the trajectories which describe type, speed, or association of moving objects with particular lanes. The primitives are combined into a small set of complex movements comprising four situations of giving way to other objects and to an alarm situation.

The change detection approach reported by Stauffer and Grimson [36] has been extended in [25] to extract and track image regions corresponding to moving bodies even from image sequences recorded by a *non-stationary* camera (as opposed to the case of a stationary camera [16]). The resulting trajectories are used in turn to infer the behavior of road vehicles or humans in the recorded scene. For this purpose, a hierarchy of entities is proposed consisting of image features, mobile object properties, and scenarios. This representation explicates links between (schematic) high-level event descriptions and low-level image features. In addition, temporal relations occurring more frequently in this application domain are explicated as well.

8.2. On conceptualizations of temporal changes

The discussion in the preceding section did not differentiate the treatment of publications according to whether they addressed *events* or *isolatable activities*, let alone the problem how to concatenate (principally isolatable) activities into a representation of agent *behavior*(*s*). Space limitations do not permit to enter into an in-depth discussion of such terminological differences. This is the reason, too, why we do not discuss [15] which exploits instantiations of occurrence schemata (rather similar to the ones discussed here) for a study of more complex vehicle behavior.

Neumann reported on a similar systems approach [30]: it, too, uses a system-internal representation for events which is based on *formal logic*. This publication introduces the notion of a (time-dependent) Geometric Scene Description (GSD) which is supposed to be provided by what we call here a core Computer Vision subsystem. Time-stamped components of this GSD provide *individuals* in the sense used for the *interpretation of predicate sets in*

formal logic—see, e.g., [31]. These predicates represent the schematic (a-priori) system-knowledge required to understand and describe temporal developments in the recorded video at a conceptual level. Details about the relations between this approach reported in [30] and the historical development of the approach reported here can be found in [27,29]. When Neumann formulated his approach, computing power available at current costs was smaller by about a factor of 10 000 compared to 2004/2005 with the consequence that the extraction of a suitable GSD was simply not yet feasible at that time. In addition to now having an operational 3D-model-based vehicle tracking system available, our system differs from Neumann's approach by relying on a *fuzzy* logic which provides much more flexibility to accommodate to stochastic errors as well as to the vagueness of the conceptual terms introduced. In addition, we use a *metric*-temporal logic which allows to *quantify* temporal relations if desirable. As illustrated in the preceding sections, the entire systems approach has been implemented and tested to the extent that non-trivial experiments on real-world videos are feasible. It remains a question for further study, though, to investigate the conceptual differences implied by the concepts 'event' vs. 'occurrence', the latter being oriented towards an *activity* aspect of temporal developments in the scene as opposed to the *result* character of such a development implicitly given by the former conceptualization.

Similar questions are raised by the more recent research reported in [7]: these authors study boundary conditions under which system-internal representations for *events* in road traffic need not be *designed*—as it has been done both by [30] and in this contribution—but can be *learned* from observations. Many of the research problems related to these different approaches have been mentioned in the concluding sections of the publication by [7]. The reader may profit from comparing their conclusions to the results reported in the preceding sections. In particular, the problem whether the set of occurrences attributed to Cahn von Seelen in [7] does indeed exhaust the conceptual space of road vehicle motions now becomes amenable to an experimental test along the lines suggested in [7]: the 'Diplomarbeit' by Cahn von Seelen mentioned there had been stimulated by considerations which occurred during the preparation of an invited talk at the Alvey Vision Conference 1987 [27]. The set of elementary road vehicle movements mentioned in [7] is thus a direct predecessor of the set of occurrences documented in this contribution.

8.3. Concluding remarks

Looked at from a different point of view, published literature related to our subject can be subdivided roughly into two categories. On the one hand, more theoretically oriented approaches investigate action and event modelling using various representation formalisms (see, e.g., [1] and [37]). These approaches are not yet applied to real data. On the other hand, there are approaches pursued by groups from the vision community which rely on signal-near image processing in order to detect and describe observations at a higher level of abstraction. Most publications mentioned above can be assigned to this second category. Due to their data-driven origin, these approaches are often limited by peculiarities of the application domain or by the capabilities of the vision system. In comparison with the more theoretically oriented approaches, examples from the latter category appear somewhat brittle at the conceptual layer. Approaches which try to connect a robust vision subsystem operating in the 3D scene domain with a comprehensive and theoretically well-founded representation formalism on a conceptual layer (as we prefer to see our approach) are still hard to find.

Anticipated improvements could comprise system modifications which estimate any temporal derivatives required by occurrence definitions already in the Computer Vision subsystem, for example by extending the status vector for the Kalman-Filter-based vehicle tracking with appropriate additional components. A suitable choice of the system covariance for these derivative components should provide smoother estimates than the local 5-point derivatives used now. It remains to be seen, however, how the Kalman-Filter will react if the dimension of the status vector is increased substantially, quite apart from the increase in computing time associated with such a modification. This consideration nicely illustrates the options which become available if the entire transformation from video to natural language concepts takes place within a single integrated systems approach.

Another path for future research consists in changing the discourse domain in order to study how this would enforce modifications to the techniques used so far. Although many other interesting topics are likely to come to one's mind upon additional reflection, it might be worth to consider the effort required to turn the presently available experimental system approach into a *reliable* tool. Experience suggests that efforts along this direction should be given priority, for example by enlarging the size and variety of video sequences to be evaluated. Such experiments will facilitate to determine the really important system parameters and to tune the overall performance by systematically varying these parameters until the system exhibits a generally satisfactory performance. Only then will one be able to distinguish

between insufficiencies due to suboptimal parameter choices and more serious shortcomings of the entire approach. Such insights should form the basis for more fundamental research into alternatives which promise to retain the advantages of the current approach and nevertheless allow to remove some of the insufficiencies which are likely to become more evident with an increasing number of experiments.

It appears, though, that the system in its current form already allows to formulate much more specific research topics than it was possible at a time where no system with the capabilities described here was available (see [29]). In addition, the fuzzy metric-temporal logic formalism used here for the import of geometric results from the core Computer Vision subsystem into the Conceptual Primitives Level can be used without modifications to aggregate such conceptual primitives into higher levels of conceptual abstractions and to instantiate related schematic representations. This representational homogeneity facilitates the transfer of the methodological approach towards other application domains as illustrated by treatment of human behavior in [13].

Acknowledgement

The authors gratefully acknowledge the carefully prepared recommendations by all anonymous reviewers.

The investigations reported here have been partial supported by the European Union FP5-project `CogViSys` (IST-2000-29404).

Appendix A. *Non*-logic vocabulary

The following list comprises the definition of all relevant predicates, including the predicates which are conceived as facts provided by the core Computer Vision subsystem (see Section 4).

A.1. Attributes for occurrence analysis

As discussed in Sections 5 and 6, the notion *occurrence* refers to an *elementary* or *primitive* movement. Each occurrence is characterized by the requirement that specific spatio-temporal relations between particular attribute values have to be satisfied.

The following subsections enumerate the attributes which have been defined for the purpose of characterizing occurrences related to *elementary or primitive road vehicle movements*. Forward references to Appendix A.2 regarding supplementary predicates like `derivative` can be found repeatedly since it is anticipated that the subsections of this appendix need not be studied serially.

A.1.1. Attributes related to 'speed'

See Fig. 7.

```
always (has_speed(Agent,Value) :-
        has_status(Agent,X,Y,Theta,V,Psi) ,
        associate_speed(V,Value)).

always (associate_speed(V,Value) :-
        degreeOfValidity(V,0,0,0.28,0.83) , Value = zero ;
        degreeOfValidity(V,0.28,0.83,2.78,5.56) , Value = small ;
        degreeOfValidity(V,2.78,5.56,12.5,16.67) , Value = normal ;
        degreeOfValidity(V,12.5,13.89,100.0,100.0) , Value = high ;
        degreeOfValidity(V,16.67,20.0,100.0,100.0) , Value = very_high ;
        degreeOfValidity(V,0.28,0.83,100.0,100.0) , Value = moving).

always (has_speed_change(Agent,Value) :-
        -2 ! has_status(Agent,X_2,Y_2,Theta_2,V_2,Psi_2) ,
        -1 ! has_status(Agent,X_1,Y_1,Theta_1,V_1,Psi_1) ,
             has_status(Agent,X,Y,Theta,V,Psi) ,
         1 ! has_status(Agent,X1,Y1,Theta1,V1,Psi1) ,
```

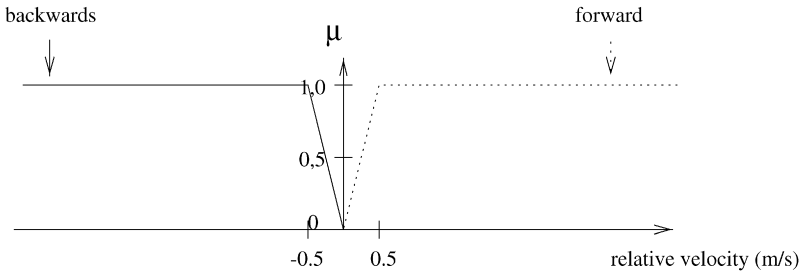


Fig. 20. Discretization of signed relative velocity values into the 'modes' `backwards` and `forward`.

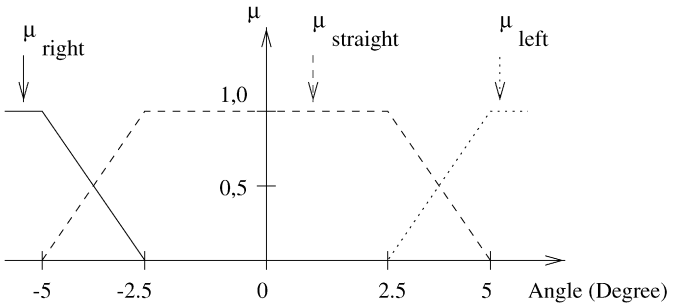


Fig. 21. Mapping of steering angle estimates to conceptual `direction` values.

```
         2 ! has_status(Agent,X2,Y2,Theta2,V2,Psi2) ,
        derivative(V_2,V_1,V,V1,V2,Deriv) ,
        associate_speed_change(Deriv,Value)).

always (associate_speed_change(Deriv,Value) :-
        degreeOfValidity(Deriv,-100,-100,-1.11,-0.56) , Value = smaller ;
        degreeOfValidity(Deriv,-1.11,-0.56,0.56,1.11) , Value = constant ;
        degreeOfValidity(Deriv,0.56,1.11,100,100) , Value = higher).
```

A.1.2. Attributes related to 'mode'

This attribute characterizes a movement as `standing`, `backwards`, or `forward`, depending on the sign and magnitude of the velocity estimate `V` obtained by model-based tracking—see Fig. 20. The attribute value `standing` corresponds essentially to the velocity interval where the degree of validity for `backwards` and `forward` drops below 1.

```
always (has_mode(Agent,Value) :-
        has_status(Agent,X,Y,Theta,V,Psi) ,
        associate_mode(V,Value)).

always (associate_mode(V,Value) :-
        degreeOfValidity(V,-100,-100,-0.5,0) , Value = backwards ;
        degreeOfValidity(V,0,0.5,100,100) , Value = forward ;
        degreeOfValidity(V,-0.83,-0.28,0.28,0.83) , Value = standing).
```

A.1.3. Attributes related to 'direction'

```
always (has_direction(Agent,Value) :-
        has_status(Agent,X,Y,Theta,V,Psi) ,
        has_speed(Agent,moving) ,
        associate_direction(Psi,Value)).
```

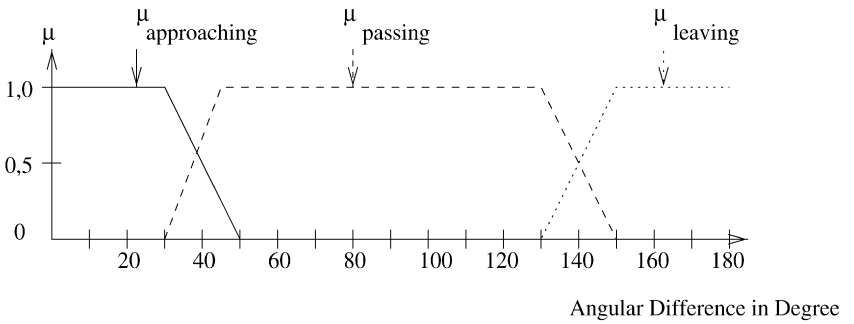


Fig. 22. Mapping the angular difference between the vehicle orientation and the line connecting the current vehicle position with a specified location in the environment into the conceptual 'course descriptions' `approaching`, `passing`, and `leaving` the location.

```
always (associate_direction(Psi,Value) :-
        degreeOfValidity(Psi,-100,-100,-5,-2.5) , Value = right ;
        degreeOfValidity(Psi,-5,-2.5,2.5,5) , Value = straight ;
        degreeOfValidity(Psi,2.5,5,100,100) , Value = left ;
        (degreeOfValidity(Psi,-100,-100,-5,-2.5) ;
         degreeOfValidity(Psi,2.5,5,100,100)) , Value = not_straight).
```

A.1.4. Attributes related to 'course_towards_location'

The supplementary predicates `ang_direction`, `ang_diff`, `ang_norm`, and `derivative` are treated in Appendix A.2.

```
always (has_course_towards_loc(Agent,Loc,Value) :-
        course_of_agent_towards_loc(Agent,Loc,Course) ,
        associate_course_towards_loc(Course,Value)).

always (course_of_agent_towards_loc(Agent,Loc,Course) :-
        has_status(Agent,X,Y,Theta,V,Psi) , location(Loc,XO,YO) ,
        ang_direction(XO-X,YO-Y,R) , ang_diff(Theta,R,Diff) ,
        ang_norm(Diff,Course)).

always (associate_course_towards_loc(Course,Value) :-
        degreeOfValidity(Course,-50,-30,30,50) , Value = approaching ;
        degreeOfValidity(Course,30,50,130,150) , Value = passing ;
        degreeOfValidity(Course,130,150,180,200) , Value = leaving ;
        degreeOfValidity(Course,-150,-130,30,50) , Value = passing ;
        degreeOfValidity(Course,-200,-200,-150,-130) , Value = leaving).

always (has_course_change_towards_loc(Agent,Loc,Value) :-
        -2 ! course_of_agent_towards_loc(Agent,Loc,Course_2) ,
        -1 ! course_of_agent_towards_loc(Agent,Loc,Course_1) ,
             course_of_agent_towards_loc(Agent,Loc,Course) ,
         1 ! course_of_agent_towards_loc(Agent,Loc,Course1) ,
         2 ! course_of_agent_towards_loc(Agent,Loc,Course2) ,
        derivative(Course_2,Course_1,Course,Course1,Course2,Deriv) ,
        associate_course_change_towards_loc(Deriv,Value)).

always (associate_course_change_towards_loc(Deriv,Value) :-
        degreeOfValidity(Deriv,-15,-5,5,15) , Value = constant ;
        degreeOfValidity(Deriv,-100,-100,-15,-5) , Value = changing ;
        degreeOfValidity(Deriv,5,15,100,100) , Value = changing).
```

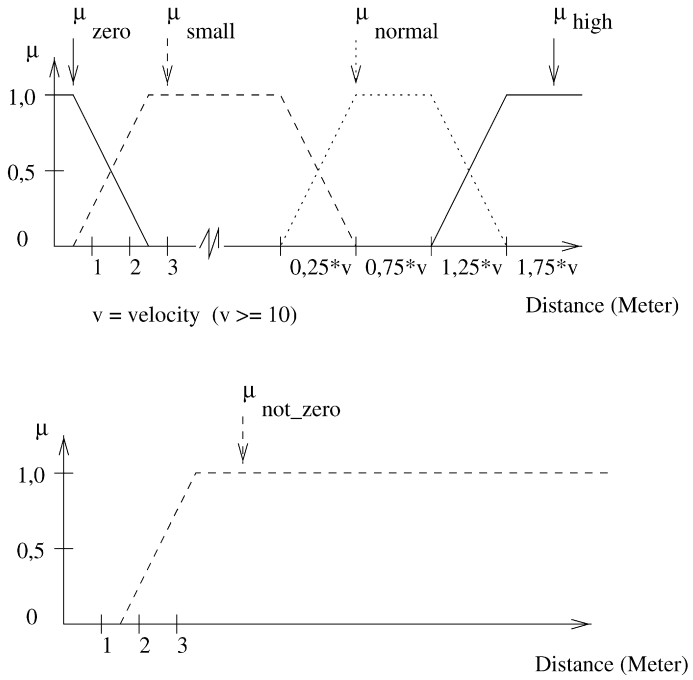


Fig. 23. Mapping the distance between the current estimate for the vehicle position and a specified location in the environment into a set of conceptual values. Note that the conceptual values for the attribute `distance_to_location` depends, too, on the estimated velocity: the threshold for the assignment of, e.g., the conceptual value `normal` increases with increasing speed.

A.1.5. Attributes related to 'distance_to_location'

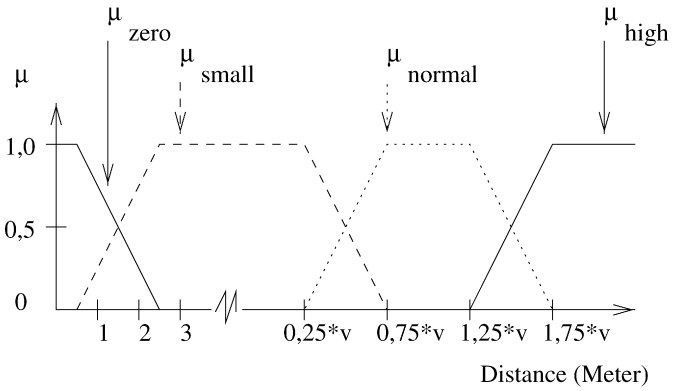
An explanation of the supplementary predicates `length`, `maximum`, and `derivative` can be found in Appendix A.2.

```
always (has_distance_to_loc(Agent,Loc,Value) :-
        distance_of_agent_to_loc(Agent,Loc,Distance,Offset) ,
        associate_distance_to_loc(Distance,Offset,Value)).

always (distance_of_agent_to_loc(Agent,Loc,Distance,Offset) :-
        has_status(Agent,X,Y,Theta,V,Psi) , location(Loc,XO,YO) ,
        length(X-XO,Y-YO,Distance) , maximum(V,10,Offset)).

always (associate_distance_to_loc(Distance,OS,Value) :-
        degreeOfValidity(Distance,-5,-5,0.5,2.5) , Value = zero ;
        degreeOfValidity(Distance,0.5,2.5,0.25*OS,0.75*OS) , Value = small ;
        degreeOfValidity(Distance,0.25*OS,0.75*OS,1.25*OS,1.75*OS) ,
                                                          Value = normal ;
        degreeOfValidity(Distance,1.25*OS,1.75*OS,100,100) , Value = high ;
        degreeOfValidity(Distance,1.5,3.5,100,100) , Value = not_zero).

always (has_distance_change_to_loc(Agent,Loc,Value) :-
        -2 ! distance_of_agent_to_loc(Agent,Loc,Distance_2,Offset_2) ,
        -1 ! distance_of_agent_to_loc(Agent,Loc,Distance_1,Offset_1) ,
             distance_of_agent_to_loc(Agent,Loc,Distance,Offset) ,
         1 ! distance_of_agent_to_loc(Agent,Loc,Distance1,Offset1) ,
         2 ! distance_of_agent_to_loc(Agent,Loc,Distance2,Offset2) ,
        derivative(Distance_2,Distance_1,Distance,Distance1,Distance2,Deriv) ,
        associate_distance_change_to_loc(Deriv,Value)).
```



v = max((v1 - v2) / 2), 10)

v1 : Velocity of the Agent
v2 : Velocity of the Object

Fig. 24. Mapping the distance between the current estimates for the position of the agent vehicle and that of another (patient) vehicle into a set of conceptual values. Note that the conceptual values for the attribute `distance` depends, too, on the estimated *relative* velocity between agent and patient, in analogy to the case of `distance_to_location` treated in Section A.1.5.

```
always (associate_distance_change_to_loc(Deriv,Value) :-
        degreeOfValidity(Deriv,-1,-0.1,0.1,1) , Value = constant ;
        degreeOfValidity(Deriv,-100,-100,-1,-0.1) , Value = smaller ;
        degreeOfValidity(Deriv,0.1,1,100,100) , Value = higher).
```

A.1.6. Attributes related to the 'distance between the agent and another moving *object'*

```
always (have_distance(Agent,Patiens,Value) :-
        distance_is(Agent,Patiens,Distance,Offset) ,
        associate_distance(Distance,Offset,Value)).

always (distance_is(Agent,Patiens,Distance,Offset) :-
        has_status(Agent,X,Y,Theta,V,Psi) ,
        has_status(Patiens,XO,YO,ThetaO,VO,PsiO) ,
        length(X-XO,Y-YO,Distance) , maximum(V,10,Offset)).

always (associate_distance(Distance,OS,Value) :-
        degreeOfValidity(Distance,-5,-5,0.5,2.5), Value = zero ;
        degreeOfValidity(Distance,0.5,2.5,0.25*OS,0.75*OS) , Value = small ;
        degreeOfValidity(Distance,0.25*OS,0.75*OS,1.25*OS,1.75*OS) ,
                                                          Value = normal ;
        degreeOfValidity(Distance,1.25*OS,1.75*OS,100,100) , Value = high ;
        degreeOfValidity(Distance,1.5,3.5,100,100) , Value = not_zero).

always (have_distance_change(Agent,Patiens,Value) :-
        -2 ! distance_is(Agent,Patiens,Distance_2,Offset_2) ,
        -1 ! distance_is(Agent,Patiens,Distance_1,Offset_1) ,
             distance_is(Agent,Patiens,Distance,Offset) ,
         1 ! distance_is(Agent,Patiens,Distance1,Offset1) ,
         2 ! distance_is(Agent,Patiens,Distance2,Offset2) ,
        derivative(Distance_2,Distance_1,Distance,Distance1,Distance2,Deriv) ,
        associate_distance_change(Deriv,Value)).
```

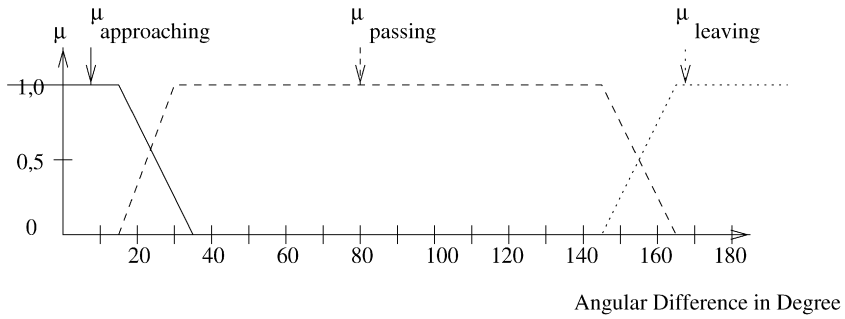


Fig. 25. Mapping the angular difference between the vehicle orientation and the line connecting the agent and patient vehicle positions into the conceptual 'course descriptions' `approaching`, `passing`, and `leaving` the patient vehicle.

```
always (associate_distance_change(Deriv,Value) :-
        degreeOfValidity(Deriv,-1,-0.1,0.1,1) , Value = constant ;
        degreeOfValidity(Deriv,-100,-100,-1,-0.1) , Value = smaller ;
        degreeOfValidity(Deriv,0.1,1,100,100) , Value = higher).
```

A.1.7. Attributes related to 'course'

This attribute evaluates the 'course' of an agent vehicle with respect to another—*standing*—patient vehicle in analogy to `course_towards_location` (see Appendix A.1.4). The supplementary predicates `ang_direction`, `ang_diff`, `ang_norm`, and `derivative` are treated in Appendix A.2.

```
always (have_course(Agent,Patiens,Value) :-
        course_is(Agent,Patiens,Course) ,
        associate_course(Course,Value)).

always (course_is(Agent,Patiens,Course) :-
        has_status(Agent,X,Y,Theta,V,Psi) ,
        has_status(Patiens,XO,YO,ThetaO,VO,PsiO) ,
        has_speed(Agent,moving) ,
        has_speed(Patiens,zero) ,
        ang_direction(XO-X,YO-Y,R) , ang_diff(Theta,R,Diff) ,
        ang_norm(Diff,Course)).

always (associate_course(Course,Value) :-
        degreeOfValidity(Course,-50,-30,30,50) , Value = approaching ;
        degreeOfValidity(Course,30,50,130,150) , Value = passing ;
        degreeOfValidity(Course,130,150,180,200) , Value = leaving ;
        degreeOfValidity(Course,-150,-130,30,50) , Value = passing ;
        degreeOfValidity(Course,-200,-200,-150,-130) , Value = leaving).

always (have_course_change(Agent,Patiens,Value) :-
        -2 ! course_is(Agent,Patiens,Course_2) ,
        -1 ! course_is(Agent,Patiens,Course_1) ,
             course_is(Agent,Patiens,Course) ,
         1 ! course_is(Agent,Patiens,Course1) ,
         2 ! course_is(Agent,Patiens,Course2) ,
        derivative(Course_2,Course_1,Course,Course1,Course2,Deriv) ,
        associate_course_change(Deriv,Value)).

always (associate_course_change(Deriv,Value) :-
        degreeOfValidity(Deriv,-15,-5,5,15) , Value = constant ;
        degreeOfValidity(Deriv,-100,-100,-15,-5) , Value = changing ;
        degreeOfValidity(Deriv,5,15,100,100) , Value = changing).
```

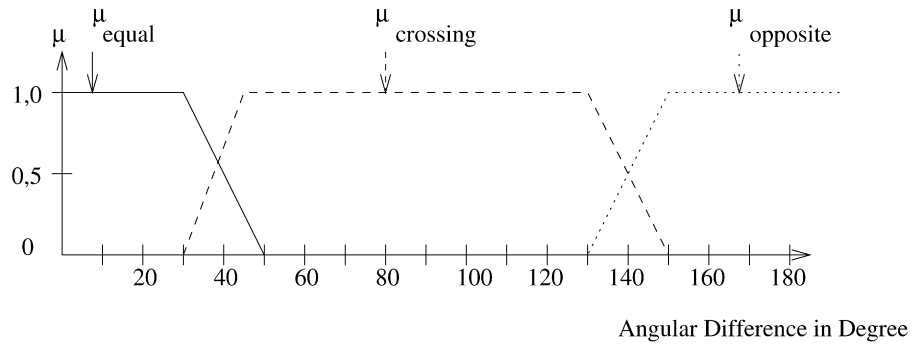


Fig. 26. Mapping the angular difference between the orientation of the agent and the patient vehicle into the conceptual values `equal`, `crossing`, and `opposite`.

A.1.8. Attributes related to the 'difference_in_orientation'

This attribute evaluates the orientation difference between the agent and a patient vehicle. The supplementary predicates `ang_diff`, `ang_norm`, and `derivative` are treated in Appendix A.2.

```
always (have_difference_in_orientation(Agent,Patiens,Value) :-
        difference_is(Agent,Patiens,Difference) ,
        associate_difference_in_orientation(Difference,Value)).

always (difference_is(Agent,Patiens,Difference) :-
        has_status(Agent,X,Y,Theta,V,Psi) ,
        has_status(Patiens,XO,YO,ThetaO,VO,PsiO) ,
        ang_diff(ThetaO,Theta,Diff) ,
        ang_norm(Diff,Difference)).

always (associate_difference_in_orientation(Difference,Value) :-
        degreeOfValidity(Difference,30,50,130,150) , Value = crossing ;
        degreeOfValidity(Difference,-50,-30,30,50) , Value = equal ;
        degreeOfValidity(Difference,130,150,400,400) , Value = opposite).

always (have_change_in_difference(Agent,Patiens,Value) :-
        -2 ! difference_is(Agent,Patiens,Diff_2) ,
        -1 ! difference_is(Agent,Patiens,Diff_1) ,
             difference_is(Agent,Patiens,Diff) ,
         1 ! difference_is(Agent,Patiens,Diff1) ,
         2 ! difference_is(Agent,Patiens,Diff2) ,
        derivative(Diff_2,Diff_1,Diff,Diff1,Diff2,Deriv) ,
        associate_change_in_difference(Deriv,Value)).

always (associate_change_in_difference(Deriv,Value) :-
        degreeOfValidity(Deriv,-15,-5,5,15) , Value = constant ;
        degreeOfValidity(Deriv,-100,-100,-15,-5) , Value = changing ;
        degreeOfValidity(Deriv,5,15,100,100) , Value = changing).
```

A.1.9. Attributes related to the position of agent relative to another vehicle

The `agent-patient-position` characterizes the position of an agent vehicle either with respect to another vehicle alone or with respect to another vehicle and the (same or neighboring) lanes on which both vehicles currently drive. In the former case, the conceptual descriptions `left_of`, `half_left_of`, `half_right_of`, or `right_of` will be used, see Fig. 27. In the latter case, the conceptual descriptions `in_front_of`, `beside_of`, or `behind` will be used—see Fig. 28—in order to emphasize that the position between agent and *patient* vehicle is given with respect to the *road spine*, i.e. by evaluation of the position difference *along the lane structure*.

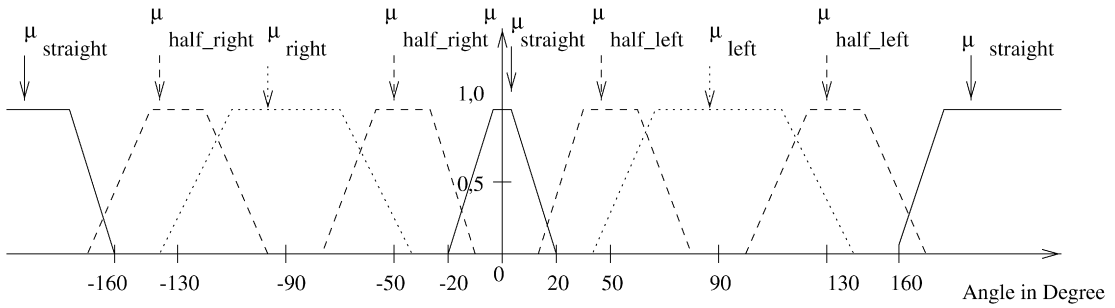


Fig. 27. The position of an *agent* vehicle relative to another (*patient*) vehicle is mapped into the conceptual values `left_of`, `half_left_of`, `half_right_of`, or `right_of`.

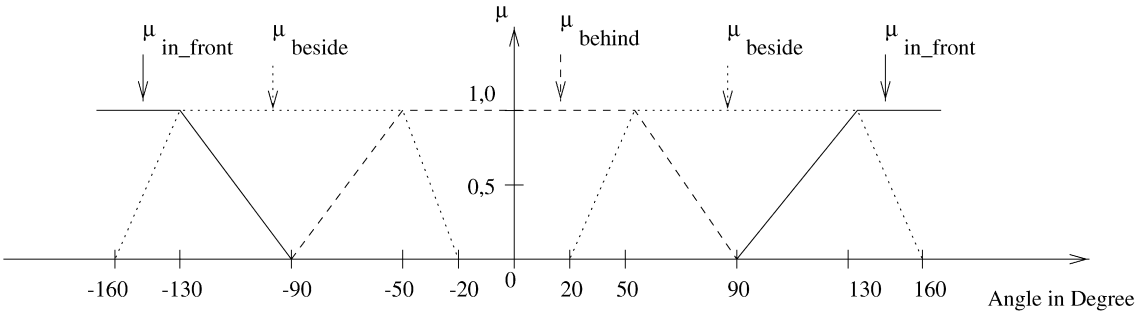


Fig. 28. The conceptual descriptions `in_front_of`, `beside_of`, or `behind` emphasize that the position between agent and patient vehicle is given with respect to the *road spine*, i.e. by evaluation of the position difference *along the lane structure*.

The representation of a lane has been presented in Appendix 4.2. The supplementary predicates `ang_direction`, `ang_diff`, and `ang_norm` are treated in Appendix A.2.

```
always (relative_position(Agent,Patiens,Value) :-
        left_of(Agent,Patiens) , Value = left ;
        half_left_of(Agent,Patiens) , Value = half_left ;
        straight_of(Agent,Patiens) , Value = straight ;
        right_of(Agent,Patiens) , Value = right ;
        half_right_of(Agent,Patiens) , Value = half_right).

always (left_of(Agent,Patiens) :-
        have_relative_position(Agent,Patiens,Angle) ,
        degreeOfValidity(Angle,40,70,110,140)).

always (half_left_of(Agent,Patiens) :-
        have_relative_position(Agent,Patiens,Angle) ,
        (degreeOfValidity(Angle,10,30,60,80) ;
         degreeOfValidity(Angle,100,120,150,170))).

always (straight_of(Agent,Patiens) :-
        have_relative_position(Agent,Patiens,Angle) ,
        (degreeOfValidity(Angle,-400,-400,-175,-160) ;
         degreeOfValidity(Angle,-20,-5,5,20) ;
         degreeOfValidity(Angle,160,175,400,400))).

always (half_right_of(Agent,Patiens) :-
        have_relative_position(Agent,Patiens,Angle) ,
        (degreeOfValidity(Angle,-170,-150,-120,-100) ;
```

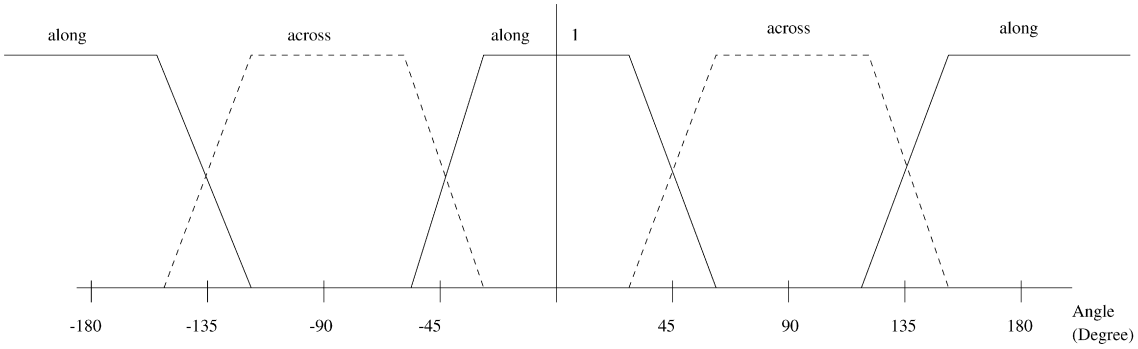


Fig. 29. Mapping the angular difference between the orientation of an `agent` vehicle and the orientation of the lane on which it drives into the conceptual values `along` and `across`.

```
          degreeOfValidity(Angle,-80,-60,-30,-10))).

always (right_of(Agent,Patiens) :-
        have_relative_position(Agent,Patiens,Angle) ,
        degreeOfValidity(Angle,-140,-110,-70,-40)).

always (have_relative_position(Agent,Patiens,Angle) :-
        has_status(Agent,X,Y,Theta,V,Psi) ,
        has_status(Patiens,XO,YO,ThetaO,VO,PsiO) ,
        Agent <> Patiens ,
        ang_direction(XO-X,YO-Y,R) , ang_norm(Theta,Theta2) ,
        ang_diff(R,Theta2,Diff) , ang_norm(Diff,Angle)).

always (configuration(Agent,Patiens,Value) :-
        in_front_of(Agent,Patiens) , Value = front ;
        beside_of(Agent,Patiens) , Value = beside ;
        behind(Agent,Patiens) , Value = behind).

always (behind(Agent,Patiens) :-
        have_relative_position(Agent,Patiens,Angle) ,
        degreeOfValidity(Angle,-90,-50,50,90)).

always (in_front_of(Agent,Patiens) :-
        have_relative_position(Agent,Patiens,Angle) ,
        (degreeOfValidity(Angle,90,130,200,200) ;
         degreeOfValidity(Angle,-200,-200,-130,-90))).

always (beside_of(Agent,Patiens) :-
        have_relative_position(Agent,Patiens,Angle) ,
        (degreeOfValidity(Angle,-160,-130,-50,-20) ;
         degreeOfValidity(Angle,20,50,130,160))).
```

A.1.10. Attributes related to agent orientation relative to lane

The orientation of an agent vehicle is described by the conceptual values `along` or `across` relative to the lane segment on which it drives—see Fig. 29. The representation of a lane has been discussed in Section 4.2. The supplementary predicates `ang_direction`, `ang_diff`, and `ang_norm` are treated in Appendix A.2.

```
always (agent_residence(Agent,Lane,on) :- on(Agent,Lane)).

always (on(Agent,Lane) :- on_lobj(Agent,Lane)).
```

```
always (lane_ref(Agent,Lane,Value) :-
        along(Agent,Lane) , Value = along ;
        across(Agent,Lane) , Value = across).

always (along(Agent,Lane) :-
        have_relative_orientation(Agent,Lane,Value) ,
        (degreeOfValidity(Value,-400,-400,-150,-120) ;
         degreeOfValidity(Value,-60,-30,30,60) ;
         degreeOfValidity(Value,120,150,400,400))).

always (across(Agent,Lane) :-
        have_relative_orientation(Agent,Lane,Value) ,
        (degreeOfValidity(Value,-150,-120,-60,-30) ;
         degreeOfValidity(Value,30,60,120,150))).

always (have_relative_orientation(Agent,Lane,Value) :-
        has_status(Agent,X,Y,Theta,V,Psi) ,
        on_lseg(Agent,Lseg) ,
        part_of(Lseg,Lane) ,
        segment_of_lane(L1,L2,Lseg) ,
        line(P1,P2,L1) , line(P3,P4,L2) ,
        point(P11,P12,P1) , point(P21,P22,P2) ,
        point(P31,P32,P3) , point(P41,P42,P4) ,
        ang_direction(P21-P11,P22-P12,Ang) ,
        ang_diff(Theta,Ang,Angdiff) ,
        ang_norm(Angdiff,Value)).

always (l_element(Agent,Value) :-
        along(Agent,Lane1) , next along(Agent,Lane2) ,
        (Lane1 <> Lane2 , Value = changing) ;
         Lane1 == Lane2 , Value = equal)).
```

A.2. Supplementary rules

ang_diff Ternary predicate symbol. Returns a substitution for variable *Diff*, where $Diff = R - T$ becomes true.

```
always (ang_diff(R,T,Diff) :- Diff is R - T).
```

ang_direction Ternary predicate symbol. Returns a substitution for variable *Ang* (representing the value of an angle between a straight line $\overline{(0,0)(X,Y)}$ and the positive x-coordinate of the 2D standard coordinate system).

```
always (ang_direction(X,Y,Ang) :-
        X==0 , Y < 0 , Ang is -90 ;
        X==0 , Y >= 0 , Ang is 90 ;
        X>0 , Y>=0 , Ang is atan(Y/X) * 360 / 6.2831853 ;
        X>0 , Y<0 , Ang is atan(Y/X) * 360 / 6.2831853 + 360 ;
        X<0 , Ang is atan(Y/X) * 360 / 6.2831853 + 180).
```

ang_norm Binary predicate symbol. Returns a substitution for variable *Norm* where the given mathematical constraint becomes true.

```
always (ang_norm(Ang,Norm) :-
        Ang >= 180 , Norm is 360 - Ang ;
        Ang < -180 , Norm is 360 + Ang ;
        Ang >= -180 , Ang < 180 , Norm is 1 * Ang).
```

degreeOfValidity 5-ary predicate symbol. Its DoV corresponds to the function value of the trapezoidal function described by the last four arguments related to the first argument, see Section 5. The meta-predicate `sp`—see Appendix B—overwrites the DoV by the arithmetic expression of its argument, see [35].

```
always (degreeOfValidity(X,P1,P2,P3,P4) :-
        X >= P1 , X < P2 , Wert is (X - P1) / (P2 - P1) , sp(Wert) ;
        X >= P2 , X < P3 , sp(1.0) ;
        X >= P3 , X < P4 , Wert is (P4 - X) / (P4 - P3) , sp(Wert)).
```

derivative 6-ary predicate symbol. Returns a substitution for variable *Deriv*, where the given mathematical constraint becomes true. Used for 5-point-derivation.

```
always (derivative(A,B,C,D,E,Deriv) :- Deriv is (-2.0*A)-B+D+(2.0*E)).
```

increasing_condition Binary predicate symbol. True, if the DoV of *condition(Agent,Verb)* is increasing at five consecutive time instants; analogously for *increasing_moncondition* and *increasing_postcondition*.

```
always (increasing_condition(Agent,Verb) :-
        -2 ? (A1 {condition(Agent,Verb)} B1) ,
        -1 ? (A2 {condition(Agent,Verb)} B2) ,
             (A3 {condition(Agent,Verb)} B3) ,
         1 ? (A4 {condition(Agent,Verb)} B4) ,
         2 ? (A5 {condition(Agent,Verb)} B5) ,
        positive_derivative(B1,B2,B3,B4,B5)).
```

inside_segment Ternary predicate symbol. True, if 2D-point (X,Y) is inside lane segment `L_Seg`.

```
always (inside_segment(X,Y,L_Seg) :- (lane_segment(L_Seg),
    segment_of_lane(L1,L2,L_Seg), line(P1,P2,L1), line(P3,P4,L2),
    test_l(X,Y,P2,P1) , test_l(X,Y,P1,P3),
    test_l(X,Y,P3,P4) , test_l(X,Y,P4,P2))).
```

length Ternary predicate symbol. Returns a substitution for variable L, where the given mathematical constraint representing the euclidean distance of two points in 2D-space becomes true.

```
always (length(A,B,L) :- L is sqrt((A * A) + (B * B))).
```

maximum Ternary predicate symbol. Returns a substitution for variable M, where the given mathematical constraint becomes true.

```
always (maximum(A,B,M) :- A < B , M is B ; A >= B , M is A).
```

negative_derivative 5-ary predicate symbol. True, if the derivation of 5 points given is negative.

```
always (negative_derivative(A,B,C,D,E) :-
        Derivative is (-2.0*A)-B+D+(2.0*E) ,
        Derivative < 0).
```

on_lobj Binary predicate symbol. Derives a DoV for an `Agent` being on a lane object `Lobj`.

```
always (on_lobj(Agens,Lobj) :- on_lseg(Agens,Lseg) , part_of(Lseg,Lobj)).
```

on_lseg Binary predicate symbol. Derives a DoV for an `Agent` being on a lane segment `Lseg`.

In order to relate vehicles to lanes, spatial reasoning is needed which, however, is not yet provided by FMTHL. Nevertheless, a binary non-fuzzy predicate `on_lane(Agent,Lane)` has been implemented which becomes absolutely true if the vehicle `Agent` is on lane `Lane`. Otherwise this predicate becomes absolutely false. In order to compute the DoV $\in \{0, 1\}$ for this predicate, the current position of the agent (obtained from corresponding results imported via the `has_status` predicate) is related to the endpoints of each segment of the lane (given by the geometric lane model, see Section 4).

```
always (on_lseg(Agens,Lseg) :- has_status(Agens,X,Y,_,_,_),
                                inside_segment(X,Y,Lseg)).
```

output 3-ary and 4-ary predicate symbol. Generates output of its arguments in the notation of metric temporal logic facts.

```
always (output(DoV, Verb, Agens) :- write(DoV) , write(' | ') ,
                    showtime, write(Verb), write('(') ,
                    write(Agens) , writeln(').')).

always (output(DoV, Verb, Agens, Object) :- write(DoV) , write(' | ') ,
                        showtime, write(Verb), write('(') ,
                        write(Agens) , write(',') ,
                        write(Object) , writeln(').')).
```

positive_derivative 5-ary predicate symbol. True, if the derivation of 5 points given is positive.

```
always (positive_derivative(A,B,C,D,E) :-
        Derivative is (-2.0*A)-B+D+(2.0*E) ,
        Derivative > 0).
```

showtime predicate symbol for time output.

```
always (showtime :-
        ci(Low,High),
        write({Low:High}),
        write(' ! ')).
```

The meta-predicate `ci` returns the current time interval, see [35].

test_l 6-ary predicate symbols. Derives a non-fuzzy DoV whether a location (X,Y) is on the left side of a directed line from location (X1,Y1) to location (X2,Y2).

```
always (test_l(X,Y,P1,P2) :- point(X1,Y1,P1), point(X2,Y2,P2), ! ,
        (S is (Y2-Y1) * (X-X1)  +  (X1-X2) * (Y-Y1)), (S >= 0.0)).
```

Appendix B. Alphabetical list of constants, variables, and predicates

across	Used as Constant and Predicate. See Appendix A.1.10.
Agent	Variable.
agent_residence	Predicate. See Appendix A.1.10.
along	Used as Constant and Predicate. See Appendix A.1.10.
always	Shortform of '1 | -i : +i !'. See Section 3.
ang_diff	Predicate. See Appendix A.2.
ang_direction	Predicate. See Appendix A.2.
ang_norm	Predicate. See Appendix A.2.
approaching	Constant. See Appendices A.1.4, A.1.7.
associate_speed	Predicate. See Appendix A.1.1.

atan	Function (Arcustangens). See Appendix A.2.
backwards	Constant. See Appendix A.1.2.
behind	Constant. See Appendix A.1.9.
beside	Constant. See Appendix A.1.9.
changing	Constant. See Appendices A.1.4, A.1.7, A.1.8, A.1.10.
ci	Meta-Predicate. Returns current time interval; see [35].
configuration	Predicate. See Appendix A.1.9.
constant	Constant. App. A.1.1, A.1.4, A.1.5, A.1.6, A.1.7, A.1.8.
crossing	Constant. See Appendix A.1.8.
degreeOfValidity	Predicate. See Section 5 and Appendix A.2.
derivative	Predicate. See Appendix A.2.
equal	Constant. See Appendices A.1.8, A.1.10.
forward	Constant. See Appendix A.1.2.
front	Constant. See Appendix A.1.9.
half_left	Constant. See Appendix A.1.9.
half_right	Constant. See Appendix A.1.9.
has_course_towards_loc	Predicate. See Appendix A.1.4.
has_distance_to_loc	Predicate. See Appendix A.1.5.
has_mode	Predicate. See Appendix A.1.2.
has_speed	Predicate. See Appendix A.1.1.
has_status	Predicate. See Section 4.1.
have_course	Predicate. See Appendix A.1.7.
have_difference_in_orientation	Predicate. See Appendix A.1.8.
have_distance	Predicate. See Appendix A.1.6.
high	Constant. See Appendices A.1.1, A.1.5, A.1.6.
higher	Constant. See Appendices A.1.1, A.1.5, A.1.6.
-i, +i	Temporal constants. See Section 3.
increasing_condition	Predicate. See Appendix A.2.
inside_segment	Predicate. See Appendix A.2.
is	Operator. See Section 3.
l1, l2, ...	Proper names for lines.
Lane, Lane1, ...	Variables (for a lane)
lane_ref	Predicate. See Appendix A.1.10.
lane_segment	Predicate. See Section 4.2.
leaving	Constant. See Appendices A.1.4, A.1.7.
left	Constant. See Appendices A.1.3, A.1.9.
l_element	Predicate. See Appendix A.1.10.
length	Predicate. See Appendix A.2.
line	Predicate. See Section 4.2.
lobj_1, lobj_2, ...	Proper names for lanes and lane segments.
maximum	Predicate. See Appendix A.2.
moving	Constant. See Appendix A.1.1.
negative_derivative	Predicate. See Appendix A.2.
next	Shortform of '+1 !'. See Section 3.
normal	Constant. See Appendices A.1.1, A.1.5, A.1.6.
not_straight	Constant. See Appendix A.1.3.
not_zero	Constant. See Appendices A.1.5, A.1.6.
obj_1, obj_2, ...	Proper names for Vehicles.
on	Predicate. See Appendix A.1.10.
on_lobj	Predicate. See Appendix A.2.
on_lseg	Predicate. See Appendix A.2.
opposite	Constant. See Appendix A.1.8.

output	Predicate. See Appendix A.2.
P1, P2, ...	Variables (for point coordinates).
p1, p2, ...	Proper names for Points.
part_of	Predicate. See Section 4.2.
passing	Constant. See Appendices A.1.4, A.1.7.
point	Predicate. See Section 4.2.
positive_derivative	Predicate. See Appendix A.2.
Psi	Variable (for steering angle of a vehicle). See Section 4.1.
relative_position	Predicate. See Appendix A.1.9.
right	Constant. See Appendices A.1.3, A.1.9.
segment_of_lane	Predicate. See Section 4.2.
showtime	Predicate. See Appendix A.2.
small	Constant. See Appendices A.1.1, A.1.5, A.1.6.
smaller	Constant. See Appendices A.1.1, A.1.5, A.1.6.
sp	Meta-Predicate. Overwrites the Degree of Validity (DoV) by the arithmetic expression of its argument; see [35].
standing	Constant. See Appendix A.1.2.
straight	Constant. See Appendices A.1.3, A.1.9.
test_l	Predicate. See Appendix A.2.
Theta	Variable (for orientation of a vehicle). See Section 4.1.
V	Variable (for velocity).
Value	Variable (for conceptual or arithmetic constants).
very_high	Constant. See Appendix A.1.1.
X	Variable (for x-coordinate of points).
Y	Variable (for y-coordinate of points).
zero	Constant. See Appendices A.1.1, A.1.5, A.1.6.
+, -, *, /, <, =, >=	Arithmetic and Relational Operators. See Section 3.

References

[1] V. Akman, S.T. Erdogan, J. Lee, V. Lifschitz, H. Turner, Representing the zoo world and the traffic world in the language of the causal calculator, Artificial Intelligence Journal 153 (2004) 105–140.

[2] M. Arens, R. Gerber, and H.-H. Nagel, Conceptual representations between video signals and natural language descriptions, Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe (TH), 76128 Karlsruhe, Germany, January 2004.

[3] H. Buxton, Learning and understanding dynamic scene activity: A review, Image and Vision Computing 21 (1) (2003) 125–136.

[4] H. Buxton, A. Mukerjee, Conceptualizing images, Image and Vision Computing 18 (2) (2000) 79.

[5] T. Caelli, W.F. Bischof, Machine Learning and Image Interpretation, Plenum Press, New York, London, 1997.

[6] S. Dance, T. Caelli, Z.-Q. Liu, Picture Interpretation—A Symbolic Approach, World Scientific Publishing Co., Singapore, 1995.

[7] J. Fernyhough, A.G. Cohn, D.C. Hogg, Constructing qualitative event models automatically from video input, Image and Vision Computing 18 (2) (2000) 81–103.

[8] C.J. Fillmore, The case for case, in: E. Bach, R.T. Harms (Eds.), Universals in Linguistic Theory, Holt, Rinehart & Winston, New York, 1968, pp. 1–90.

[9] A. Galata, A. Cohn, D. Magee, D. Hogg, Modeling interaction using learnt qualitative spatio-temporal relations and variable length Markov models, in: F. van Harmelen (Ed.), Proc. 15th European Conference on Artificial Intelligence (ECAI-2002), 21–26 July 2002, Lyon, France, IOS Press, Amsterdam, 2002, pp. 741–745.

[10] R. Gerber, H.-H. Nagel, 'Occurrence' extraction from image sequences of road traffic scenes, in: L. van Gool, B. Schiele (Eds.), Proceedings Workshop on Cognitive Vision, 19–20 September 2002, ETH, Zurich, Switzerland, 2002, pp. 1–8, http://www.vision.ethz.ch/cogvis02/finalpapers/gerber.pdf.

[11] R. Gerber, H.-H. Nagel, Algorithmic conversion of road traffic videos into natural language descriptions, Technical Report, Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe (TH), 76128 Karlsruhe, Germany, January 2005.

[12] S. Gong, H. Buxton, Understanding visual behaviour, Image and Vision Computing 20 (12) (2002) 825–826.

[13] J. Gonzàlez i Sabató, Human sequence evaluation: the key-frame approach, Doctoral Dissertation, Universitat Autònoma de Barcelona, Bellaterra, Spain, May 2004. Ediciones Gráficas Rey, S.L., 2004, ISBN 84-933652-2-X. http://www.cvc.uab.es/poal/hse/hse.htm.

[14] M. Haag, H.-H. Nagel, Combination of edge element and optical flow estimates for 3D-model-based vehicle tracking in traffic image sequences, International Journal of Computer Vision 35 (3) (1999) 295–319.

[15] M. Haag, H.-H. Nagel, Incremental recognition of traffic situations from video image sequences, Image and Vision Computing 18 (2) (2000) 137–153.

[16] S. Hongeng, R. Nevatia, Multi-agent event recognition, in: Proc. 8th Int. Conference on Computer Vision (ICCV 2001), vol. II, 9–12 July 2001, Vancouver, BC, Canada, IEEE Computer Society, Los Alamitos, CA, 2001, pp. 82–91.
[17] R.J. Howarth, H. Buxton, Conceptual descriptions from monitoring and watching image sequences, Image and Vision Computing 18 (2) (2000) 105–135.
[18] H. Kamp, U. Reyle, From Discourse to Logic, Kluwer Academic Publishers, Dordrecht, Boston, London, 1993.
[19] D. Koller, K. Daniilidis, H.-H. Nagel, Model-based object tracking in monocular image sequences of road traffic scenes, International Journal of Computer Vision 10 (1993) 257–281.
[20] H. Kollnig, H.-H. Nagel, Ermittlung von begrifflichen Beschreibungen von Geschehen in Straßenverkehrsszenen mit Hilfe unscharfer Mengen, Informatik—Forschung und Entwicklung 8 (1993) 186–196 (in German).
[21] H. Kollnig, H.-H. Nagel, 3D pose estimation by directly matching polyhedral models to gray value gradients, International Journal of Computer Vision 23 (3) (1997) 283–302.
[22] S. Krieger, M. Arens, R. Gerber, H.-H. Nagel, Estimation of vehicle color and its use for text generation from traffic videos, CogViSys–Final Report, Institut für Algorithmen und Kognitive Systeme, Universität Karlsruhe (TH), Karlsruhe, Germany, February 2005.
[23] H. Leuck, H.-H. Nagel, Automatic differentiation facilitates OF-integration into steering-angle-based road vehicle tracking, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition CVPR'99, vol. 2, 23–25 June 1999, Fort Collins, CO, pp. 360–365.
[24] Z.-Q. Liu, L.T. Bruton, J.C. Bezdek, J.M. Keller, S. Dance, N.R. Bartley, C. Zhang, Dynamic image sequence analysis using fuzzy measures, IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics 31 (4) (2001) 557–572.
[25] G. Medioni, I. Cohen, F. Brémond, S. Hongeng, R. Nevatia, Event detection and analysis from video streams, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 23 (8) (2001) 873–889.
[26] http://kogs.iaks.uni-karlsruhe.de/motris/.
[27] H.-H. Nagel, From image sequences towards conceptual descriptions, Image and Vision Computing 6 (2) (1988) 59–74, Invited Lecture presented at the Third Alvey Vision Conference, 15–17 September 1987, Cambridge, UK.
[28] H.-H. Nagel, Image sequence evaluation: 30 years and still going strong, in: A. Sanfeliu, J.J. Villanueva, M. Vanrell, R. Alquézar, J.-O. Eklundh, Y. Aloimonos (Eds.), Proc. 15th International Conference on Pattern Recognition ICPR-2000, vol. 1, 3–7 September 2000, Barcelona, Spain, IEEE Computer Society, Los Alamitos, CA, 2000, pp. 149–158.
[29] H.-H. Nagel, Steps toward a cognitive vision system, AI-Magazine 25 (2) (2004) 31–50.
[30] B. Neumann, Natural language description of time-varying scenes, in: D. Waltz (Ed.), Semantic Structures—Advances in Natural Language Processing, Lawrence Erlbaum Associates, Publ., Hillsdale, NJ, 1989, pp. 167–206 (Chapter 5, Report FBI-HH-B-105/84, Fachbereich Informatik der Universität Hamburg, Hamburg, Germany).
[31] B. Neumann, Th. Weiss, Navigating through logic-based scene-models for high-level scene interpretations, in: J.L. Crowley, J.H. Piater, M. Vincze, K. Paletta (Eds.), Proc. 3rd International Conference on Computer Vision Systems (ICVS 2003), 1–3 April 2003, Graz, Austria, in: Lecture Notes in Computer Science, vol. 2626, Springer-Verlag, Berlin, Heidelberg, New York, 2003, pp. 212–222.
[32] M. Otte, H.-H. Nagel, Estimation of optical flow based on higher order spatiotemporal derivatives in interlaced and non-interlaced image sequences, Artificial Intelligence Journal 78 (1995) 5–43.
[33] P. Remagnino, T. Tan, K. Baker, Agent oriented annotation in model based visual surveillance, in: Proc. Sixth International Conference on Computer Vision (ICCV 1998), 4–7 January 1998, Bombay, India, Narosa Publishing House, New Delhi, India, 1998, pp. 857–862.
[34] M. Shah, Guest introduction: the changing shape of computer vision in the twenty-first century, International Journal of Computer Vision 50 (2) (2002) 103–110.
[35] K.H. Schäfer, Unscharfe zeitlogische Modellierung von Situationen und Handlungen in Bildfolgenauswertung und Robotik, Dissertation, Fakultät für Informatik der Universität Karlsruhe (TH), Juli 1996 (in German). Erschienen in der Reihe 'Dissertationen zur Künstlichen Intelligenz (DISKI)', Band 135, Sankt Augustin: infix 1996, ISBN 3-89838-135-8. This book is available from Akademische Verlagsgesellschaft GmbH, Berlin, Germany; it can be ordered via their internet website http://www.aka-verlag.de.
[36] C. Stauffer, W.E.L. Grimson, Learning patterns of activity using real-time tracking, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 22 (8) (2000) 747–757.
[37] M. Ying, H. Wang, Lattice-theoretic models of conjectures, hypothesis and consequences, Artificial Intelligence 139 (2002) 253–267.