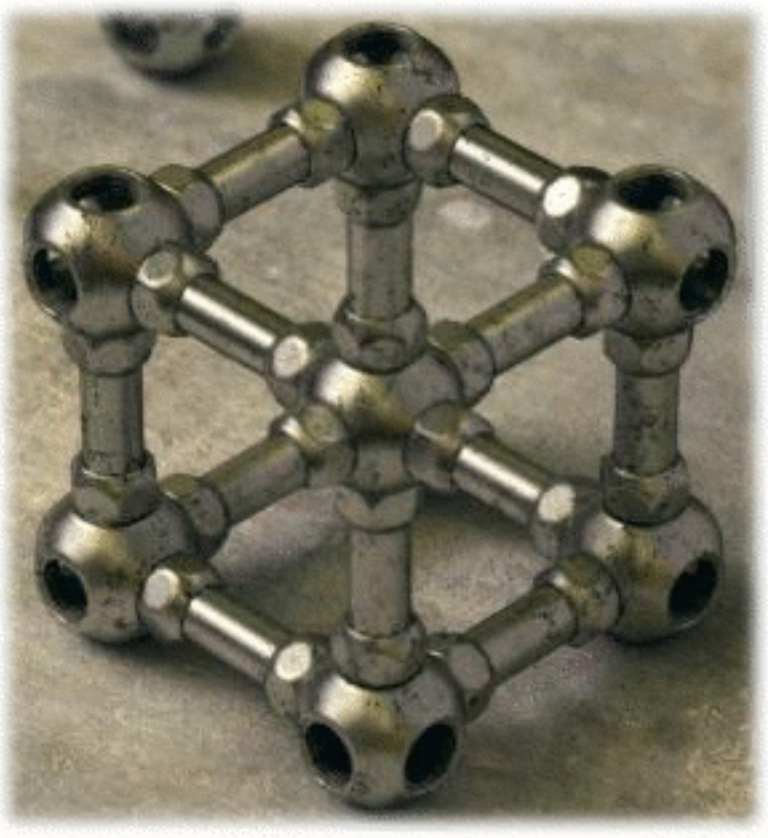

Probabilistic Logic Reasoning About Traffic Scenes [1]

Author(s):
Carlos R. C. Souza
Paulo E. Santos

[1]This work was supported by Fapesp Project LogProb, grant 2008/03995-5, São Paulo, Brazil.

Probabilistic Logic Reasoning About Traffic Scenes

Carlos R. C. Souza[1] and Paulo E. Santos[2]

[1] carlos.roberto@daimler.com
Mercedes-Benz do Brasil
Av. Alfred Jurzykowski, 562,
SBC, São Paulo, Brazil
[2] santosp@ieee.org
Centro Universitario da FEI,
Av. Humberto de A. Castelo Branco, 3972,
SBC, São Paulo, Brazil

Abstract. This paper describes a probabilistic logic reasoning system for traffic scenes based on Markov logic network, whose goal is to provide a high-level interpretation of localisation and behaviour of a vehicle on the road. This information can be used by a lane assistant agent within driver assistance systems. This work adopted an egocentric viewpoint for the vision and the reasoning tasks of the vehicle and a qualitative approach to spatial representation. Results with real data indicate good performance compared to the common sense interpretation of traffic situations.

1 Introduction

Lane changing is recognised as a significant cause of automotive accidents [4]. This fact justifies the development of active safety systems by car manufacturers. Active safety systems (e.g., *Lane Assistant System* or LAS) aim to prevent accidents by warning the driver through a visual or audio alarm, or even by controlling the vehicle's actuators. This paper contributes with a part of a LAS that is dedicated to a high-level interpretation scheme.

Commercial lane assistant systems are generally based on a monocular camera relying on quantitative measures of distance between the vehicle and the road divider [11]. In this paper, the perception of the traffic environment is sensed by a webcam attached to a vehicle. The dividers' discrete position (e.g., right, left) and type information (e.g., continuous, dashed) are extracted from an off-the-shelf vision system and used as evidence to infer localisation and behaviour of the vehicle. The uncertainties inherent to the sensors, actuators and to the real-world phenomena are treated with Markov logic networks (MLN) [1], which facilitates a complex representation of high-level knowledge (such as traffic rules) as well as of the domain uncertainties. MLN also permits us to estimate the probability of an event occurrence in the presence of faults or imprecision of the sensors from probabilistic inference.

In [3] Markov logic networks are also used to infer object relations in traffic scenes but, although the agent is immersed in its environment, the reasoning is global, i.e., it is done from a bird's eyes view. In turn, [15] implements an egocentric reasoning using Bayesian networks and description logics to model the environment. However, the lack of expressiveness of description logics and the acyclicity constraint of Bayesian networks jeopardise scaling the model up to a practical application standard. Some of these issues are overcome in MLN, as we shall see in this paper.

This paper is organised as follows: in the next section we propose a framework for lane divider interpretation. On Section 3 we briefly describe the vision system. On Section 4 we present a brief overview of probabilistic logic and present the Markov logic network approach. Section 5 presents a formalisation of the traffic domain. On Section 6 we show some results with real data applied to our model of traffic environment. Section 7 concludes this work.

2 A Markov Logic Framework for a Lane Assistance System

A reasoning system was developed with Markov logic network (MLN) to infer the egocentric localisation of the vehicle relative to lane dividers and the vehicle's behaviour (e.g., whether it is driving on the wrong way, or crossing the lane). Data input to the reasoning system (identification, type and localisation of dividers) comes from a vision algorithm, which receives information from a webcam attached to a driving car. Both video processing and inference must be real time, with low resolution images from finite (reduced) domains this can be achieved. The proposed framework is designed as shown in Figure 1. This diagram is based on the work described in [11].

We can describe each layer of the diagram as follows:

1. *Perception*: a monocular camera that provides environment information. Here we use a Microsoft VX2000 webcam with 320x240 pixels of video resolution;
2. *Segmentation*: extracts features (edges) from the video obtained on Layer 1;
3. *Line Tracking*: identify lane dividers and track them;
4. *MLN Inference*: comprehends the model of the traffic domain and the inference methods for querying traffic conditions;
5. *Assistant Function*: based on the results of the previous layer, it decides what type of signal or message will be sent to the next layer. The implementation of this layer is outside the scope of this work;
6. *Actuator*: could be an audio alarm, a message or light in the panel or even an output from the main module to actuate on the vehicle itself.

Layers 2 and 3 are processed with Matlab and Layer 4 is an external program invoked by Matlab (as introduced in Section 4.1). A more complete description of each of these layers follows below.

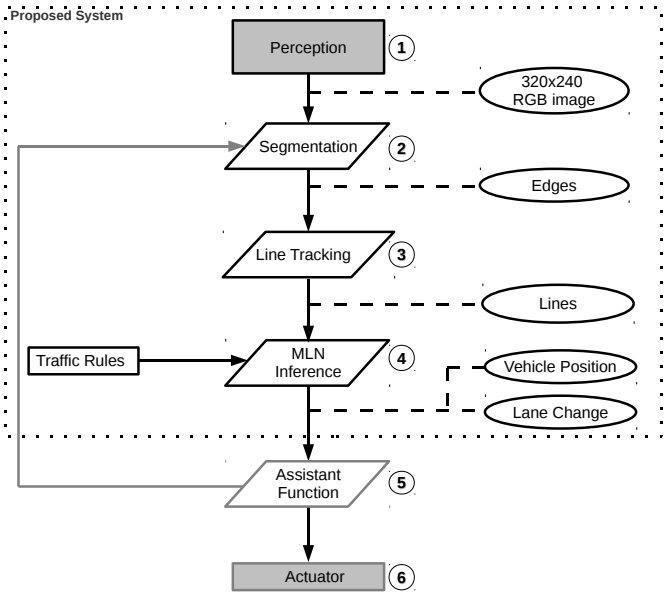

Fig. 1. LAS proposed based on the architecture of [11].

3 Vision System

The video stream used in this work has a resolution of 320x240 pixels and its frame rate is 30 Hz. Assuming the half-half methodology for training and testing [2], odd frames were used to train the MLN and even frames were used for test and inference. Thus, in practice, we had a frame rate of 15Hz to process.

Basically the vision algorithm includes the following steps:

1. Take the lower half of RGB image (the region of interest (ROI)) convert it to intensity and apply a 2D filter with a vertical mask. For our video stream we changed the original mask to a Sobel vertical mask, which increased intensity of the edges of the image;
2. Binarise the filtered image and apply the Hough transform;
3. Clear the Hough matrix based in a window of ρ and θ and locate two local maxima; where ρ is the perpendicular distance from the lines detected to the image's origin and θ is the angle between ρ and the top of the image;
4. Track a set of lane dividers, matching the current detection with those lines of the previous frame;
5. Detect if there is a lane departure (a divider) under the vehicle. For the reasoning system it was adopted that every time a line passes through the bottom of the image, it is considered to be under the vehicle;
6. Take the RGB image and convert it to YCbCr. Detect the type and the colour of lane dividers by counting the percentage of white or yellow pixels on the lines;

7. Display the type and colour text of the lines and output a lane crossing warning text when it is true.

A more detailed description of this vision algorithm can be found at [7]. Figure 2 presents some frames obtained with this algorithm.

Fig. 2. Some frames obtained with the vision algorithm.

The next section makes a brief overview of probabilistic logics and introduces the Markov logic framework.

4 Probabilistic Logics

For a long time in Artificial Intelligence, logic and probabilistic reasoning approaches were treated separately [14]. However, many tasks of the real-world require probabilistic reasoning about relational data representing multiply related objects with large sets of variables. This model of the world needs to be compactly represented in order to minimise representational and inferential complexities. Propositional probabilistic approaches such as Bayesian networks are insufficient to cope with these requirements, because they describe a fixed set of random variables, and specify dependencies and probability distributions for each variable individually. Thus, a significant number of *first-order probabilistic languages* (FOPL) have been proposed over the last decades, since they are able to compactly represent large sets of random variables by abstracting over objects [9].

In [9] these approaches are organised into a taxonomy of the outcome spaces to which the languages assign probabilities. In another work [10], these approaches were divided into two groups: *extensional* and *intensional* systems. The first one propagates truth values generalised from true or false to a scale of varying degrees of certainty. The second group places restrictions on a probability distribution on possible worlds.

There are further divisions within the intensional approaches, but the most predominant subgroup is the family of *Knowledge Base Model Construction* (KBMC), which constructs a propositional graphical model from a first-order language specification that answers a query [14]. Some KBMC approaches are based on Bayesian networks such as *Bayesian Logic* (BLOG) [8] or *Probabilistic Relational Models* (PRM) [6]. Other approaches have Markov networks as

probabilistic model, such as *Relational Markov Networks* (RMN) [16] or *Markov Logic Networks* (MLN) [13].

Different from Bayesian networks, Markov networks are undirected models that use weights to define the relative probability of instantiations. In relational domains, some random variables can occasionally depend on one another without a clear causal relation. In this case, Markov network models have an advantage over Bayesian network because they have no acyclicity constraint, which simplifies the modelling of the problem. However, the disadvantage is that learning can be hard in undirected graphical models when using, for example, existentially quantified formulae.

Markov Logic Networks (MLNs) [13] have evolved rapidly in recent years, in general, due to its simple semantics while retaining the expressiveness of first-order logic. Accompanied by a well supported software (*Alchemy* [5]) and its large spectrum of real domain applications, MLNs are in an advanced stage of development in relation to any other FOPL approaches [14].

4.1 Markov Logic and Alchemy

Markov Logic Network (MLN) was developed aiming an unified language to be an interface layer for Artificial Intelligence. This interface should provide the basic infrastructure to the "foundational" areas of AI such as knowledge representation, automated reasoning, probabilistic models and machine learning. [1].

The basic idea of MLNs is to soften the restrictions imposed by a first-order knowledge base (KB). Each first-order formula has an associated weight that reflects how strong is the formula constraint: the higher the weight, the greater the difference in probability between a world that satisfies the formula and one that does not satisfy [1].

A Markov logic network L is a set of formulae F_i in first-order logic (e.g., $divider(YDashed, Right, t) \Rightarrow wrongWay(t)$) with a weight w_i (real number) attached to each formula. This can be viewed as a template for constructing Markov networks $M_{L,C}$, where C is a set of constants (e.g., for position: Left, Right, Centre, Under) and the probability distribution over possible worlds x is given by:

$$P(X=x)=\frac{1}{Z}exp\left(\sum_i w_i f_i(x)\right)=\frac{1}{Z}\prod_i \phi_i(x_{\{i\}})^{n_i(x)},$$

where $n_i(x)$ is the number of true groundings of F_i in x, $x_{\{i\}}$ is the state of the i-th clique which has a corresponding feature $f_i(x) \in \{0.1\}$ and an associated weight $w_i = log\phi_i(x_{\{i\}})$. Z is a normalisation factor also known as the partition function and is given by $Z = \sum_{x\in\mathcal{X}} \prod_i \phi_i(x_{\{i\}})$.

Inference in MLN can be probabilistic or logical, but are #P-Complete and NP-Complete respectively. However, a MLN allows encoding of knowledge, including context-specific independencies which makes inference more efficient. Moreover, it is also possible to use approximate inference methods in Markov

networks such as the *Markov Chain Monte Carlo* (MCMC), with Gibbs sampling, to sample each variable in turn given its Markov blanket [13]. These and other algorithms for inference and learning in MLN are implemented in the open-source software *Alchemy* [5] which is used in this work.

Learning in MLN can be discriminative or generative and the algorithm more commonly used for both learning and inference is the MC-SAT. MC-SAT is a slice sampling MCMC which uses a combination of satisfiability testing and simulated annealing to sample from the slice of a distribution [1]. MC-SAT treats deterministic or near-deterministic dependencies by efficiently finding isolated modes in the distribution, resulting in a fast mixing of Markov chain. Alchemy has MC-SAT as default inference, but there are other algorithms available, such as belief propagation, Gibbs sampling and simulated tempering.

In the next section we describe our formalisation of the problem and the contextual information.

5 Contextual Information

Contextual information (in the case of traffic scenes) is the relevant information to understand the environment and it could be related to traffic rules or to attributes of things in the environment. One important characteristic of a Lane Assistance System (LAS) is the fact that the agent of perception is inside the context, e.g., the point of view of the agent is the same of the driver.

In order to model the traffic environment, we first have to answer some questions about normal driving:

- *What is important to know about the environment to keep the car on the lane?*
  Taking into account only the road, without obstacles, we visually follow the lane dividers on both sides of the vehicle, controlling it to stay at the centre of the lane.
- *What sort of identification gives us sense of direction or localisation on the road?*
  Shape and colour of a divider give us the sense of direction and position. We know that because there are traffic rules which give us conditions to reason about the lane dividers we are seeing. The rules concerning our problem domain are described in Table 2 below.

The following variables were defined to identify the environment:

- Type of lane divider: *type*={ *YContinuous, WContinuous, YDashed, WDashed, Merge*}. Where *W* means white and *Y* means yellow. *Merge* is the line representing the access to enter the road and to depart from it, it is a dashed line with shorter intervals than the dashed lane dividers;
- Road direction: *way*={*One, Two*}, one or two-way road;

– Vehicle position: *lanepos*={*Left, Centre, Right*}. Vehicle position is relative to the road (e.g., "the vehicle is on the centre lane"). Centre is considered every position that is not in the extreme left or right lanes. Divider position is relative to the vehicle (e.g., "the divider is on the left side of the vehicle");
– Divider status: *status*={*Ok,Under*}. Indicates if a divider is under the vehicle (e.g., during a crossing) or at the vehicle sides (*Ok*).

Using these variables the following predicates were defined. Notice that they all represent facts about the ego-vehicle, thus the variable for the agent was omitted. In what follows the variable *time* represents the frame number at which the predicates hold.

– *dividerL/R(type,status,time)*: divider identification with type (*Dashed, Continuous*), its status (*ok, under*), its frame number (time) and its relative position to the car (*Left or Right: L/R*). Ground facts of *dividerL*/3 and *dividerR*/3 are output by the vision algorithm;
– *carRelPos(lanepos,time)*: car's relative position wrt the road (*lanepos*);
– *crossingLeft(time)*: vehicle is crossing to the left lane;
– *crossingRight(time)*: vehicle is crossing to the right lane;
– *emergencyLane(time)*: vehicle is on the emergency stop lane;
– *prohibitedManoeuvre(time)*: vehicle is doing a prohibited manoeuvre (e.g., crossing a yellow continuous divider);
– *wrongWay(time)*: vehicle is on the wrong way of the road.

With these predicates, MLN formulae were constructed to encode traffic rules (about right-handed traffic) and the knowledge about the environment as shown on Table 2. Note that initially these formulae are only first-order logical sentences without weight, the weights will be learnt from the data using MC-SAT on the related Markov network. Due lack of space we abbreviate the predicates in the formulae as shown in Table 1.

dL	*dividerL*	pM	*prohibitedManoeuvre*	Le	*Left*
dR	*dividerR*	emg	*emergencyLane*	Ri	*Right*
cRP	*carRelPos*	t	*time*	WC	*WContinuous*
xL	*crossingLeft*	ty	*type*	YC	*YContinuous*
xR	*crossingRight*	s	*status*	WD	*WDashed*
rW	*roadWay*	Un	*Under*	YD	*YDashed*
wW	*wrongWay*	Ce	*Centre*	M	*Merge*

Table 1. Abbreviation of predicates and constants.

Predicates without specific constants (e.g., s, ty, t) will be grounded for all constants. The negation of some predicates on each formula was necessary to specify unchanged characteristics of the domain given the evidences.

In the next section we show some results of training and inference applying this model on real data.

1. If vehicle is on a yellow continuous divider or there is one at the right side, it is doing a prohibited manoeuvre.
$dL(YC, Un, t) \vee dR(YC, Ok, t) \Rightarrow pM(t) \wedge wW(t) \wedge rW(Two, t) \wedge cRP(Le, t) \wedge \neg cRP(Ce, t) \wedge \neg cRP(Ri, t) \wedge \neg emg(t)$

2. If there isn't evidence of a two way road, consider it to be a one way road.
$\neg dL(YC, s, t) \vee \neg dL(YD, s, t) \vee \neg dR(YC, s, t) \vee \neg dR(YD, s, t) \Rightarrow rW(One, t) \wedge \neg rW(Two, t)$

3. If there is a yellow continuous or yellow dashed divider at any position, the road has two ways.
$dL(YC, s, t) \vee dL(YD, s, t) \vee dR(YC, s, t) \vee dR(YD, s, t) \Rightarrow rW(Two, t) \wedge \neg rW(One, t) \wedge cRP(Le, t) \wedge \neg cRP(Ce, t) \wedge \neg cRP(Ri, t) \wedge \neg emg(t)$

4. If the left divider is white continuous and the right divider is white dashed, the road has one way and the vehicle is on the left lane.
$dL(WC, Ok, t) \wedge (dR(WD, Ok, t) \Rightarrow rW(One, t) \wedge \neg rW(Two, t) \wedge cRP(Le, t) \wedge \neg cRP(Ce, t) \wedge \neg cRP(Ri, t) \wedge \neg pM(t) \wedge \neg wW(t)$

5. If the divider is dashed on both sides of the view, the vehicle is on the central lane. Centre is considered any position that isn't the extreme right or left lanes.
$dL(WD, Ok, t) \wedge dR(WD, Ok, t) \Rightarrow cRP(Ce, t) \wedge \neg cRP(Le, t) \wedge \neg cRP(Ri, t) \wedge \neg wW(t) \wedge \neg xL(t) \wedge \neg xR(t) \wedge \neg pM(t) \wedge \neg emg(t)$

6. If the left divider is not white dashed and the right divider is white dashed, the vehicle is on the left lane.
$\neg dL(WD, Ok, t) \wedge dR(WD, Ok, t) \Rightarrow cRP(Le, t) \wedge \neg cRP(Ce, t) \wedge \neg cRP(Ri, t) \wedge \neg xL(t) \wedge \neg xR(t) \wedge \neg pM(t) \wedge \neg emg(t)$

7. If the divider is yellow dashed at the right side, the car is on the wrong way. It is not a prohibited manoeuvre because this condition is permitted for overtaking.
$dR(YD, Ok, t) \wedge dL(ty, Ok, t) \Rightarrow wW(t) \wedge cRP(Le, t) \wedge \neg cRP(Ce, t) \wedge \neg cRP(Ri, t) \wedge \neg xL(t) \wedge \neg xR(t) \wedge \neg pM(t) \wedge \neg emg(t)$

8. If the left divider is white dashed and the right divider is white continuous or merge, the vehicle is on the right lane.
$(dR(WC, Ok, t) \vee dR(M, Ok, t)) \wedge dL(WD, Ok, t) \Rightarrow cRP(Ri, t) \wedge \neg cRP(Ce, t) \wedge \neg cRP(Le, t) \wedge \neg wW(t) \wedge \neg xL(t) \wedge \neg xR(t) \wedge \neg emg(t)$

9. If a divider is white continuous and another is not white dashed, the vehicle can be on the emergency lane.
$dL(WC, Ok, t) \wedge \neg dR(WD, s, t) \Rightarrow emg(t) \wedge cRP(Ri, t) \wedge \neg cRP(Le, t) \wedge \neg cRP(Ce, t) \wedge \neg pM(t) \wedge \neg wW(t)$
$dR(WC, Ok, t) \wedge \neg dL(WD, s, t) \Rightarrow emg(t) \wedge cRP(Le, t) \wedge \neg cRP(Ri, t) \wedge \neg cRP(Ce, t) \wedge \neg pM(t) \wedge \neg wW(t)$

10. If the vehicle is over a right divider or over a left divider with a right divider *Ok*, the car is crossing to the right (and analogously for crossing to the left). This prevents one predicate overlapping with another when, as given by the vision system, the divider changes from left to right and vice-versa at the middle of the screen.
$dR(ty1, Un, t) \vee (dL(ty2, Un, t) \wedge dR(ty2, Ok, t)) \Rightarrow xR(t) \wedge \neg xL(t)$
$dL(ty1, Un, t) \vee (dR(ty2, Un, t) \wedge dL(ty2, Ok, t)) \Rightarrow xL(t) \wedge \neg xR(t)$

Table 2. MLN formulae for LAS.

6 Results

The weight learning for each formula was executed with MC-SAT algorithm [1] for the 1800 odd frames. The weight w_i learnt means that a world where n clauses of a formula are true is $e^{w_i n}$ less probable than a world where all clauses are true. This is specially important to a lane assistance system, since it allows the system to provide plausible answers even with misclassified evidences.

During inference MC-SAT is also used with evidences generated by the vision algorithm on the 1800 even frames as we previously described on Section 2. The query was made on the predicates: carRelPos/2, crossingLeft/1, crossingRight/1, emergencyLane/1, prohibitedManoeuvre/1, wrongWay/1 and roadWay/2. The inference results were plotted on Figure 3 along with their relative ground truth.

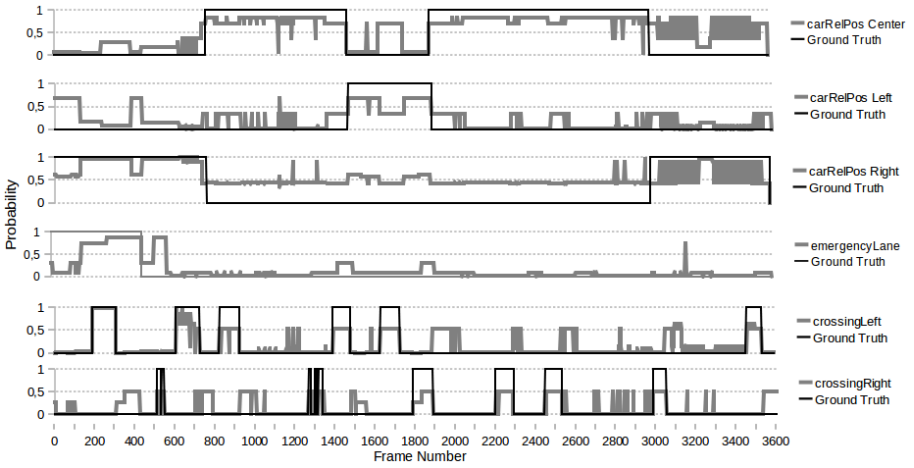

Fig. 3. Position and crossing probabilities querying the system at the even frames and the relative ground truth.

We have used as decision threshold the probability greater than 50%. As we can note on Figure 3, this threshold would be higher for position predicates ($CarRelPos$), but the behaviour predicates (*crossingLeft/Right*) had lower probabilities than the others. However, the distinction between them were clearly defined (as shown on the two bottom graphs of Figure 3). The low probabilities of *CrossingLeft/Right* predicates is due to the fact that the vision system only sees the nearest lane dividers, thus in the middle of a *CrossingLeft* event (for instance) there is also evidence that this might be a *CrossingRight* event.

In the framework presented on Figure 1, we could treat this specific problem on layer five, analysing which predicate started the lane change and ignoring the other predicates right after its next occurrence. This, however, is an issue for further research.

The ground truth was manually labelled, considering a lane departure every time the inferior extremity of a divider appears at the bottom line of the frame. Crossing a divider was adopted until the divider changes the side wrt the vehicle (i.e., when it crosses the centre of the frame), analogously for lane position changes. We verified (as expected) that a specification of position during a lane change in the model degraded the prediction of *crossing* predicates. This is due to the nature of the Markov network, where the position predicates become part of the same clique of crossing predicates, thus contributing to the same probability distribution.

6.1 Evaluation

In order to evaluate our model we use a confusion matrix as defined in [2]: $\left(\begin{smallmatrix} tp & fn \\ fp & tn \end{smallmatrix}\right)$, where *tp* stands for true positive, *fp* is false positive, *tn* is true negative and *fn* is false negative. With a confusion matrix, for each of our queries, we measured the accuracy, sensitivity, precision and specificity of prediction.

Accuracy is given as: $accuracy = \frac{tp+tn}{(tp+tn+fp+fn)}$. The accuracy is the measure of reliability of the detection, i.e., from the total number of predictions how much were correct.

The sensitivity (or recall) is the fraction of the existing true predicates that were detected as such and is given by: $sensitivity = \frac{tp}{(tp+fn)}$.

The precision is a measure of accuracy of a specific class, in our case we are interested in the detection of true positive predicates, i.e., the algorithm reports position and behaviour of the vehicle without false detection. The precision is given as: $precision = \frac{tp}{(tp+fp)}$.

The specificity is a measure of how frequently the algorithm reports false predicates as true negative and is given by: $specificity = \frac{tn}{(tn+fp)}$.

On Table 3 we present the results obtained for the inferences using the 1800 even frames.

Although our video stream does not present situations of prohibited manoeuvre, wrong way driving or two way road, their predicates were 100% true negative. It is worth pointing out that there were misclassified evidences of yellow dividers in a few frames (in 10 consecutive frames of the dataset), and that even with this misclassified data the true negative rate was 100%.

For the *roadWay(One)* ground predicate, its precise detection can be attributed, in general, to the assertion: "if there isn't evidence for a two-way road, consider it as a one-way road" (Statement 2 in Table 2). It is indeed reasonable to say that if we are not seeing a yellow divider, we do not need to worry if the road has two ways.

In the next section, we discuss the results obtained and conclude this paper.

7 Concluding Remarks

In this paper, we proposed a formalisation in probabilistic logic of a lane assistance system. The results showed that this model presents coherent values given

Ground Predicate	Accuracy (%)	Sensitivity (%)	Precision (%)	Specificity (%)
carRelPos(Centre)	75.5	77.8	74.9	73.1
carRelPos(Left)	91.0	56.2	61.0	95.4
carRelPos(Right)	84.2	55.1	96.2	98.9
crossingLeft	90.5	97.3	64.2	89.2
crossingRight	89.8	86.4	53.5	90.3
emergencyLane	96.0	66.7	82.9	98.7
prohibitedManoeuvre	100	*	*	100
wrongWay	100	*	*	100
roadWay(One)	100	100	100	#
roadWay(Two)	100	*	*	100
Mean	*89.6*	*77.1*	*76.1*	*90.9*

Table 3. Results obtained for the inference. The symbol * means that there was no true grounded predicates on the data and # means the same for false grounded predicates.

the evidences and that the distinction between cases were clearly defined. Low precision values for crossing predicates were due the change of divider's side during the crossing action. Low sensitivity for lateral lane positions (Left / Right) and emergency lane, can be associated to the misclassification of dividers by the vision algorithm.

The inference in our model considers only the current evidences, for that reason, some influence of misclassified dividers is observable primarily on lateral lanes where continuous dividers are sometimes detected as dashed.

Future work shall refine the model by using the event calculus [12] to reason about actions, in this work we focused on spatial reasoning (mainly on reasoning about the divider's positions). Further study can be done with Markov conditions on inference, i.e, with current and past evidences. In parallel, the vision algorithm has space for improvement. For example, type detection and tracking could have as parameters the vehicle's speed in order to automatically adjust values of filters or of decision thresholds.

Inference on Alchemy with this model was efficient but not real-time with MC-SAT algorithm (approximately 0.15 seconds per frame). We achieved real time processing changing the inference algorithm to belief propagation (0.02 seconds per frame) whereas analogous true (false) positive rates were obtained with both algorithms on this domain.

Acknowledgements

Paulo Santos acknowledges support from FAPESP project LogProb, 2008/03995-5, São Paulo, and also travel support from CNPq, Brazil.

References

1. Domingos, P., Lowd, D.: Markov Logic: an interface layer for artificial intelligence. Morgan & Claypool (2009)

2. Fitzpatrick, J.M., Sonka, M.: Handbook of Medical Imaging. Medical Image Processing and Analysis, vol. 2, chap. 10, pp. 567–605. SPIE press, 1 edn. (Jun 2000)
3. Hensel, I., Bachmann, A., Hummel, B., Tran, Q.: Understanding object relations in traffic scenes. VISAPP (2010)
4. Holzmann, F.: Needs of improved assistant systems. In: Adaptive Cooperation between Driver and Assistant System, pp. 3–10. Springer Berlin Heidelberg (2008)
5. Kok, S., Sumner, M., Richardson, M., Singla, P., Poon, H., D. Lowd, Domingos., P.: The alchemy system for statistical relational ai. Technical report, Department of computer science and engineering, university of Washington (2007), http://alchemy.cs.washington.edu.
6. Koller, D., Pfeffer, A.: Probabilistic frame-based systems. In: In proceedings of AAAI. pp. 580–587. AAAI press (1998)
7. MathWorks: Lane departure warning system. (05 2010), http://www.mathworks.com/ products/ viprocessing/ demos.html?file=/ products/ demos/ shipping/ vipblks/ vipldws.html#5, revision: 1.1.6.3
8. Milch, B., Marthi, B., Russell, S., Sontag, D., Ong, D.L., Kolobov, A.: Blog: probabilistic models with unknown objects. In: In IJCAI. pp. 1352–1359 (2005)
9. Milch, B., Russell, S.J.: First-order probabilistic languages: into the unknown. In: Muggleton, S., Otero, R.P., Tamaddoni-Nezhad, A. (eds.) ILP. Lecture notes in computer science, vol. 4455, pp. 10–24. Springer (2006)
10. Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann (September 1988)
11. Reif, K. (ed.): Automobilelektronik. Vieweg Teubner, Wiesbaden, 3 edn. (2009)
12. Reiter, R.: Knowledge in action: logical foundations for specifying and implementing dynamical systems. The MIT press, Massachusetts, MA, illustrated edition edn. (2001)
13. Richardson, M., Domingos, P.: Markov logic networks. Machine learning 62(1-2), 107–136 (2006)
14. de Salvo Braz, R., Amir, E., Roth, D.: A survey of first-order probabilistic models. In: Holmes, D.E., Jain, L.C. (eds.) Innovations in Bayesian networks, Studies in computational intelligence, vol. 156, pp. 289–317. Springer (2008)
15. Santos, P., Cozman, F., Pereira, V.F., Hummel, B.: Probabilistic logic encoding of spatial domains. UniDL (2010)
16. Taskar, B., Abbeel, P., Koller, D.: Discriminative probabilistic models for relational data. In: Darwiche, A., Friedman, N. (eds.) UAI. pp. 485–492. Morgan Kaufmann (2002)