CENTRO UNIVERSITÁRIO DA FEI

CARLOS ROBERTO CARDOSO DE SOUZA

INTERPRETAÇÃO LÓGICO-PROBABILÍSTICA DA FUNCIONALIDADE DE FAIXAS DE TRÂNSITO A PARTIR DE DADOS DE UMA CÂMERA ACOPLADA A UM VEÍCULO.

São Bernardo do Campo
2011

CARLOS ROBERTO CARDOSO DE SOUZA

INTERPRETAÇÃO LÓGICO-PROBABILÍSTICA DA FUNCIONALIDADE DE FAIXAS DE TRÂNSITO A PARTIR DE DADOS DE UMA CÂMERA ACOPLADA A UM VEÍCULO.

Dissertação de Mestrado apresentada ao Centro Universitário da FEI para obtenção do título de Mestre em Engenharia Elétrica, orientada pelo(a) Prof. Dr. Paulo Eduardo Santos.

São Bernardo do Campo
2011

Souza, Carlos Roberto Cardoso de.
Interpretação lógico-probabilística da funcionalidade de faixas de trânsito a partir de dados de uma câmera acoplada a um veículo / Carlos Roberto Cardoso de Souza. São Bernardo do Campo, 2011.
103 f.

Dissertação - Centro Universitário da FEI.
Orientador: Prof. Paulo Eduardo Santos

1. Raciocínio espacial qualitativo. 2. Lógica probabilística. 3. Visão cognitiva. I. Santos, Paulo Eduardo, orient. II. Título.

CDU 007.5

Centro Universitário da **FEI**

APRESENTAÇÃO DE DISSERTAÇÃO
ATA DA BANCA JULGADORA

PGE- 10

Programa de Mestrado de Engenharia Elétrica

Aluno: Carlos Roberto Cardoso de Souza Matrícula: 1091081

Título do Trabalho: **INTERPRETAÇÃO LÓGICO-PROBABILÍSTICA DA FUNCIONALIDADE DE FAIXAS DE TRÂNSITO A PARTIR DE DADOS DE UMA CÂMERA ACOPLADA A UM VEÍCULO.**

Área de Concentração: Inteligência Artificial Aplicada à Automação

Orientador: Prof. Dr. Paulo Eduardo Santos

Data da realização da defesa: 9 / setembro / 2011

ORIGINAL ASSINADA

A Banca Julgadora abaixo-assinada atribuiu ao candidato o seguinte:

APROVADO ☐ REPROVADO ☐

São Bernardo do Campo, 9 / setembro / 2011.

MEMBROS DA BANCA JULGADORA

Prof. Dr. Paulo Eduardo Santos

Ass.: _____

Prof. Dr. Reinaldo Augusto da Costa Bianchi

Ass.: _____

Prof. Dr. Denis Fernando Wolf

Ass.: _____

VERSÃO FINAL DA DISSERTAÇÃO

ENDOSSO DO ORIENTADOR APÓS A INCLUSÃO DAS RECOMENDAÇÕES DA BANCA EXAMINADORA

_____

Aprovação do Coordenador do Programa de Pós-graduação

_____

Prof. Dr. Carlos Eduardo Thomaz

À minha esposa Angélica e ao meu filho Denis.

AGRADECIMENTOS

Agradeço à minha esposa Angélica pelo incentivo e apoio nos momentos difíceis durante o desenvolvimento deste trabalho.

Agradeço ao meu filho Denis, que aguardou ansiosamente o término desta grande "lição de casa". Responder às suas perguntas me fazia relembrar o básico de cada questão, o que foi de grande valor nos momentos em que a solução parecia distante.

Aos meus pais que me ensinaram a ser persistente na realização dos meu objetivos.

Aos amigos do mestrado, pelo companheirismo e troca de experiências.

Aos professores da FEI, que sempre foram amigos e proporcionaram ótimas discussões.

Ao meu orientador professor Dr. Paulo Eduardo Santos, por estimular a reflexão e a autocrítica e pela persistência e paciência na revisão do meu artigo. Sempre à disposição para discutir qualquer dúvida ou sugestão.

A todos aqueles que direta ou indiretamente colaboraram para o desenvolvimento deste trabalho.

A Deus, por todos os desafios apresentados e superados nesta jornada.

Meu objetivo é simples. É uma completa compreensão do universo, porque ele é como é e porque ele existe enfim.

Stephen Hawking

RESUMO

Neste trabalho desenvolvemos um modelo de raciocínio lógico-probabilístico para cenas de tráfego rodoviário. O objetivo é prover uma interpretação de alto nível sobre a localização e comportamento de um veículo em uma estrada. Esta informação pode ser utilizada por sistemas de assistência ao motorista, tendo como foco o sistema assistente de faixas de trânsito. A percepção do ambiente de tráfego é proveniente de uma câmera acoplada a um veículo, portanto, o ponto de vista é egocêntrico, o qual também foi adotado para o modelo de raciocínio. Para tratar os dados da câmera, o algoritmo de visão classifica, com o auxílio da transformada de Hough, as posições discretas (por exemplo, esquerda, direita) das linhas divisórias e também o tipo de divisão (por exemplo, amarelo contínuo, branco tracejado). As informações de posição e tipo das linhas divisórias são utilizadas como evidências para a inferência da localização e comportamento do veículo. Esta inferência é feita numa base de conhecimento regida por regras de trânsito, sendo estas regras modeladas com lógica de primeira ordem. As regras do modelo lógico se baseiam em técnicas de raciocínio espacial qualitativo. As incertezas inerentes aos sensores e aos próprios fenômenos do mundo real são tratadas com lógica probabilística de primeira ordem, mais especificamente as redes lógicas de Markov, ferramenta relativamente nova que permite numa mesma representação, tratar incertezas e representar de forma compacta uma grande variedade de conhecimento. Isso nos permite estimar qual é a probabilidade de um evento ocorrer mesmo na presença de falhas ou imprecisão dos sensores, além da possibilidade de inserirmos informação contextual do ambiente, neste caso, as regras de trânsito. A avaliação de desempenho do algoritmo de inferência revelou elevados valores de sensitividade, acurácia, precisão e especificidade, confirmando o potencial das redes lógicas de Markov.

Palavras-chave: Raciocínio espacial qualitativo. Lógica probabilística. Visão cognitiva.

ABSTRACT

In this work we developed a probabilistic logical model of traffic scenes. The goal is to provide a high-level interpretation of a vehicle's location and behavior on a highway. This information can be used for driver assistance systems, focusing on the lane assistant system. The perception of the traffic environment is provided by a camera attached to a vehicle, so the viewpoint is egocentric. The same viewpoint was considered for the reasoning model. To process the camera's data, the vision algorithm classifies, based on the Hough transform, the discrete locations (e.g., left, right) of the lane dividers and also the type of divider (e.g., yellow continuous, white dashed). The divider's position and type information are used as evidence for the inference of vehicle's location and behavior. This inference is made on a knowledge base of traffic rules, where these rules are modeled with first-order logic. The rules of the logical model are based on qualitative spatial reasoning techniques. The uncertainties inherent to the sensors and to the real-world phenomena are treated with first-order probabilistic logic, more specifically the Markov logic networks, relatively new tool that allows in a single representation to deal with uncertainties and to compactly represent a wide range of knowledge. This allows us to estimate what is the probability of an event occurrence even in the presence of failure or inaccuracy of the sensors. It is also possible with Markov logic networks, to add contextual information of the environment in a representation, in this case the traffic rules. The performance evaluation of the inference algorithm revealed higher values of accuracy, sensitivity, precision and specificity. These results confirm the potential of Markov logic networks.

Keywords: Qualitative spatial reasoning. Probabilistic logics. Cognitive vision.

LISTA DE TABELAS

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A mudança de faixa de trânsito é reconhecida como uma das causas mais significativas de acidentes automotivos (HOLZMANN, 2008; DAIMLER, 2009a). Estes estudos serviram de base para os fabricantes de automóveis investirem no desenvolvimento de sistemas de segurança ativa para seus veículos.

Os sistemas de segurança ativa têm como objetivo prevenir acidentes por meio de avisos ao motorista (por meio de alarmes visuais ou sonoros) ou até controlando atuadores do veículo (por exemplo, força de reação no volante). Fazem parte desta categoria, sistemas de assistência ao motorista que compreendem entre outros, sistemas de estabilidade direcional, aviso de ponto cego e assistentes de faixa de trânsito (DAIMLER, 2009b; AMSEL et al., 2009). O presente trabalho contribui com módulos de percepção e raciocínio de um sistema assistente de faixa de trânsito (do inglês *lane assistant system* ou LAS).

Os sistemas assistentes de faixa de trânsito (LAS) possuem a função de detectar quando um veículo inicia uma mudança de faixa e de emitir em seguida um aviso ao motorista (alarme sonoro, sinalização no painel ou vibração no volante) no caso da manobra ser involuntária (sem acionar a seta). O objetivo deste trabalho é prover uma interpretação de alto nível às informações provenientes de um sistema de visão de um LAS, de forma que seja possível inferir além da mudança de faixa de trânsito, a posição discreta do veículo na estrada (por exemplo, esquerda, direita e centro) ou se a manobra é permitida. Estas informações adicionais possibilitam novos níveis de alerta que não se limitam ao LAS, mas teriam também aplicação em sistemas de comunicação carro-a-carro (do inglês *car-to-car communication*) (MANIFESTO, 2007) ou ainda numa supervisão mútua entre os sistemas (por exemplo, entre o LAS e o GPS por exemplo).

Os dados de percepção do modelo implementado são provenientes de uma câmera instalada no interior do veículo, assim como nos LAS comerciais. Desta forma, obtemos informações do ambiente e de objetos no ambiente por meio das imagens capturadas. O módulo de percepção implementado, que compreende o hardware e algoritmo de visão, apresenta desafios bem conhecidos da área de visão computacional como a trepidação, o rastreamento e a oclusão. Parte destes problemas tiveram de ser tratados durante a parametrização do algoritmo de visão para adaptá-lo aos dados obtidos, os quais são influenciados pela posição da câmera, velocidade variável do veículo e sinalização da estrada entre outros. O sistema proposto utiliza um algoritmo de visão de fácil acesso conhecido como “sistema de aviso de saída de faixa” (do inglês, *lane departure warning system*). Este algoritmo é uma demonstração do conjunto de ferramentas de visão (*vi-*

sion toolbox) do Matlab (MATHWORKS, 2010) e pode ser usado como referência para futuros trabalhos.

O uso da câmera no interior do veículo provê a percepção das localizações desses objetos em relação ao agente da percepção (BICAS, 2004; RUSSELL; NORVIG, 2004). Porém, diferente dos sistemas comerciais, a informação espacial é tratada qualitativamente, ou seja, a posição das linhas divisórias é discreta (esquerda, direita, sob). Desta forma, é possível otimizar o custo computacional de processamento, uma vez que saímos do domínio contínuo (por exemplo, a distância de uma linha em relação ao centro da imagem) e passamos para o domínio discreto (posição qualitativa das linhas). Outra vantagem da abordagem qualitativa é a redução de complexidade na modelagem do ambiente. Esta abordagem foi inspirada nas técnicas de raciocínio espacial qualitativo (REQ), as quais são utilizadas na formalização do conhecimento espacial por meio de sistemas discretos de símbolos (COHN; HAZARIKA, 2001).

O algoritmo de visão provê as posições das linhas divisórias e os tipos de linhas (amarelo contínuo, branco tracejado) como entrada do módulo de raciocínio ou inferência. Para tratar as incertezas inerentes ao sensor utilizado (câmera) e ao ambiente, utilizou-se a lógica probabilística de primeira ordem, neste caso, redes lógicas de Markov ou MLN (*Markov Logic Networks* em inglês) (DOMINGOS; LOWD, 2009). A inferência puramente probabilística, como no caso das redes de Markov, é baseada num modelo proposicional o que dificulta expressar e inferir relações entre os objetos de um domínio. Por sua vez, na inferência puramente lógica obtemos como resposta a uma consulta somente falso ou verdadeiro, não é permitido contradições nas sentenças. Além disso, ambas as inferências, lógica e probabilística, são intratáveis na maioria dos casos. As redes lógicas de Markov englobam ambos os modelos de inferência e são mais eficientes que as demais técnicas na maioria dos casos.

As redes lógicas de Markov representam fórmulas em lógica de primeira ordem que permitem a atribuição de pesos de uma rede de Markov, proporcionando uma suavização das restrições. Ou seja, se um mundo violar uma regra (isto é, se alguma cláusula de uma sentença for falsa), a implicação será menos provável mas não impossível, como no caso da lógica de primeira ordem. O uso de MLN, facilita a representação de conhecimento de alto nível, como as regras de trânsito, assim como as incertezas do domínio. Por exemplo, uma regra de trânsito indica que, numa via de mão dupla, se a linha que separa cada mão é da cor amarela e do tipo contínuo, a ultrapassagem por esta linha é proibida. Em MLN, uma das formas de representar esta regra seria por meio dos predicados *linhaEsquerda(tipo,status,tempo)* e *manobraProibida(tempo)*, onde os nomes entre parênteses são as variáveis que por sua vez podem ser instanciadas para alguns valores ou constantes como: 2,5 *linhaEsquerda(AmarelaContínua,Sob,10)* $\rightarrow$ *manobraProibida(10)*. Neste caso, temos condições de codificar várias características em

um único predicado, facilitando a representação do domínio. Em lógica proposicional não conseguimos expressar estas relações diretamente. Mesmo na tentativa de codificar várias características num único termo, devemos criar um termo para cada combinação possível, tornando a representação mais complexa.

Outra característica da MLN é a possibilidade de inferir estados em que uma relação causal não é clara entre os predicados. Isto é necessário na maioria dos fenômenos reais, onde não conseguimos prever todas as relações presentes entre as variáveis. Por exemplo, podemos consultar qual é a probabilidade de um veículo estar na faixa direita de uma via dado que estamos vendo somente linhas brancas tracejadas em ambos os lados do veículo.

Neste trabalho os algoritmos de visão e inferência são complementares, ou seja, como são esperados alguns erros provenientes do algoritmo de visão, devido a qualidade da imagem e alguns aspectos não modelados ou considerados (por exemplo, a iluminação do ambiente), esperamos que o algoritmo de inferência possa tratar estas deficiências de forma que haja a probabilidade de um evento ocorrer, mesmo quando as evidências dadas pelo algoritmo de visão estejam incorretas.

Algumas formalizações foram testadas e foi possível inferir qualitativamente a localização do veículo em relação à estrada e a função das linhas divisórias, provendo também informação sobre o sentido da via e manobras permitidas. O modelo lógico-probabilístico do ambiente de tráfego em rede lógica de Markov foi implementado com auxílio do programa de código aberto *Alchemy* (KOK et al., 2007), o qual possui diversos algoritmos de treinamento e inferência dedicados à MLN, o que significou economia de tempo e testes mais variados.

Na figura 1.1 apresentamos um diagrama do modelo implementado neste trabalho baseado na arquitetura de alguns sistemas comerciais (AMSEL et al., 2009).

Podemos descrever cada camada do LAS da seguinte forma:

a) *percepção*: captura informações do ambiente externo ao veículo por meio de um ou mais sensores. No modelo implementado foi instalada uma câmera monocular no interior do veículo;

b) *segmentação*: extrai as características (bordas) de cada quadro de vídeo obtido na camada anterior;

c) *classificação e rastreamento*: detecta as linhas divisórias da faixa de trânsito, classifica-as quanto ao tipo e cor e as rastreia;

d) *inferência em MLN*: compreende o modelo do domínio de tráfego rodoviário e o método

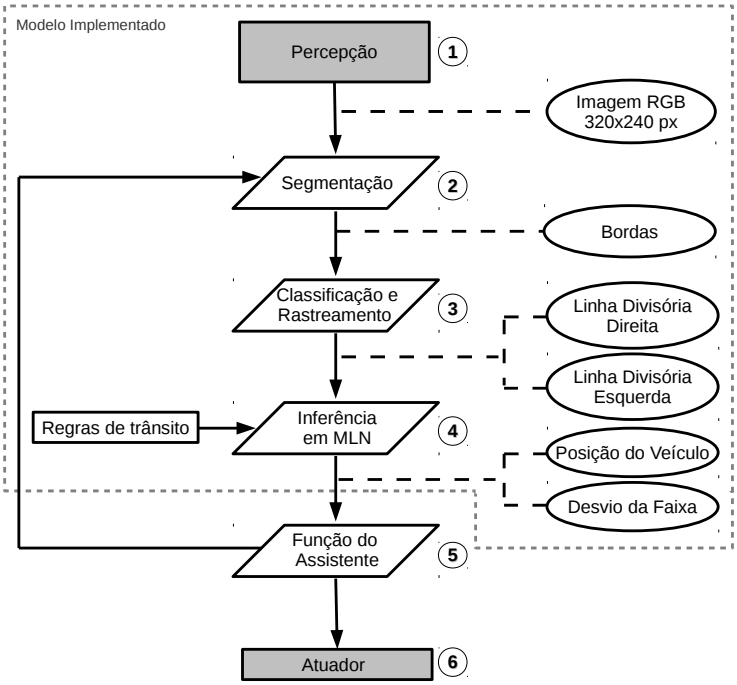

Figura 1.1: Sistema de assistência ao motorista (LAS) proposto.

Fonte: Autor "adaptado de" Amsel et al. (2009)

de inferência para consultar as condições de tráfego. As condições de tráfego consultadas foram: posição do veículo na estrada, mudança de faixa e se esta mudança é permitida;

e) *função do assistente*: baseado no resultado da camada anterior, decide qual tipo de sinal ou mensagem será enviada à próxima camada. A implementação desta camada está fora do escopo deste trabalho;

f) *atuador*: pode ser um alarme sonoro, uma mensagem ou luz no painel do veículo, ou ainda uma vibração ou reação no volante (se for do tipo *drive-by-wire*). Assim como a camada anterior, sua implementação está fora do escopo deste trabalho. Porém, em conjunto com a camada anterior, pode ser interessante para trabalhos futuros.

Em resumo, o objetivo deste trabalho é prover informações de posição e comportamento de um veículo em uma estrada. Estas informações serão a saída da camada 4 do sistema proposto (figura 1.1), tendo como entrada um vídeo de baixa resolução. A inferência destas informações será de modo probabilístico em um modelo lógico de primeira ordem do ambiente de tráfego rodoviário. As contribuições principais deste trabalho são a modelagem lógico-probabilística de um ambiente de tráfego rodoviário e uma nova aplicação de Redes Lógicas de Markov. As terminologias técnicas rodoviárias seguem a definição do Departamento Nacional de Estradas

de Rodagem (DNER) (DNER - DEPARTAMENTO NACIONAL DE ESTRADAS DE RODAGEM, 1997).

O presente trabalho está organizado da seguinte forma: no capítulo 2, apresentamos uma revisão da literatura relevante ao desenvolvimento deste trabalho, abordando temas como o raciocínio espacial qualitativo, análise de imagem, lógica probabilística e interpretação de cenas de tráfego. No capítulo 3 descrevemos os métodos utilizados para o desenvolvimento do modelo proposto. No capítulo 3.3 desenvolvemos o modelo do ambiente de tráfego. No capítulo 4 temos os resultados obtidos do modelo desenvolvido e a avaliação dos mesmos e então, no capítulo 5 concluímos o presente trabalho com sugestões para futuros estudos. Parte dos capítulos 3.3 e 4 foram publicados em (SOUZA; SANTOS, 2011).

2 REVISÃO BIBLIOGRÁFICA

Neste capítulo apresentamos uma revisão da literatura relacionada a quatro temas chave para a execução deste trabalho: percepção robótica, análise de imagens, raciocínio espacial qualitativo e lógica probabilística.

Na seção 2.1 abordamos o tema percepção robótica. Na seção 2.2 descrevemos as principais considerações em análise de imagens e os algoritmos mais comuns para classificação de objetos em uma cena. Na seção 2.3 vemos as técnicas de raciocínio espacial qualitativo, usadas para representar a informação espacial por meio de variáveis discretas e então na seção 2.4 apresentamos as principais técnicas de lógica probabilística de primeira ordem, visando o tratamento das incertezas inerentes aos fenômenos observados e aos métodos utilizados. A seção 2.5 cita alguns trabalhos relacionados à interpretação de cenas de tráfego e reconhecimento de faixas trânsito.

2.1 Percepção Robótica

A palavra percepção vem do latim *perceptio*, do verbo *percipere*, formado de *per*, "através de", "intensidade" e *capere*, "tomar", "captar". Significa então, obter uma informação externa através de um meio apropriado para representar tal informação (MACHADO, 2005).

Segundo Russell e Norvig (2004), a percepção robótica é o processo pelo qual os robôs mapeiam as medições de sensores em representações internas do ambiente em que se encontram. É um processo complexo, em geral, devido a sensores ruidosos e ambiente parcialmente observável, imprevisível e frequentemente dinâmico, o que dificulta uma fiel representação do ambiente nos estados internos do robô. Esta representação do conhecimento, no entanto, deve conter informações suficientes para o robô tomar as decisões corretas e devem ser estruturadas de forma que sejam atualizadas com eficiência e correspondam às variáveis do mundo físico. De modo similar, Shanahan (1996), acrescenta que a percepção robótica pode ser vista como um processo passivo onde os dados do sensor são assimilados no espaço de crença do robô por um processo de abdução, que consiste em criar hipóteses para a existência, forma e localização dos objetos.

A percepção robótica neste trabalho é baseada num sensor de visão, pois os dados obtidos com a visão são ricos em informações, ou seja, é possível extrair conhecimento sobre diversas características do ambiente à partir de técnicas de visão computacional. Entretanto, apesar das técnicas de visão computacional fornecerem informações complexas do ambiente, estas ainda

são limitadas. Uma alternativa para tratar estes dados de percepção é otimizar o sistema de raciocínio de forma que este lide com o máximo de informações provenientes de um único sensor, o que possibilita, dentre outras características, uma implementação mais econômica do projeto ou ainda, uma informação mais precisa que poderia ser usada numa fusão sensória, isto é, associando as informações já processadas do sensor de visão com os dados dos demais sensores do veículo servindo de supervisão ou redundância de alguma função. Portanto, o objetivo da percepção neste trabalho é prover todo o conhecimento do ambiente necessário ao LAS somente com os dados da câmera. Porém, para se obter dados condizentes com o estado do mundo real ainda é necessária uma interpretação dos sinais captados pelos sensores, e esta interpretação é também chamada de cognição por alguns autores (MACHADO, 2005).

No campo da cognição robótica, podemos citar um trabalho de longa duração (mais de 30 anos) apresentado por Nagel (2004), com o objetivo de construir um *Sistema de Visão Cognitiva* capaz de expressar em linguagem natural, cenas de tráfego de veículos. Este trabalho trouxe grandes contribuições no que se refere às experiências com rastreamento de objetos e percepção espacial.

O rastreamento de objetos é uma ampla área de pesquisa em visão computacional. Na próxima seção serão apresentados alguns processos de análise de imagem que proporcionam o rastreamento de um objeto de interesse em uma imagem.

2.2 Análise de imagem

A análise de imagem é uma importante tarefa no campo de visão computacional e pode ser utilizada, dentre outros fins, para se obter a localização de um objeto em cada quadro de um vídeo. Em geral, é um problema desafiador devido à movimentos bruscos de objetos, mudança na aparência da cena e objeto, estruturas não-rígidas, relações de oclusão e movimento da câmera (YILMAZ; JAVED; SHAH, 2006). Porém, assumem-se algumas restrições conforme cada caso, para evitar ou controlar estas questões.

Yilmaz, Javed e Shah (2006) apresentam uma pesquisa bem abrangente em relação ao estado-da-arte do rastreamento de objetos. Nesta seção faremos uma breve revisão da literatura a respeito, a fim de situar o processamento de imagem para situações de tráfego nestas técnicas.

Há diversas abordagens de análise que se diferenciam pela forma como representam os objetos, quais características (*features*) são importantes e como forma, aparência e movimento são modelados. Estas questões variam conforme o contexto e meio em que a análise será executada e como esta informação será usada.

Alguns pontos devem ser considerados para a detecção ou rastreamento de um objeto na imagem:

a) Representação adequada do objeto;

b) Características (*features*) da imagem usadas como entrada;

c) Estratégia para detecção do objeto na cena;

d) Método de classificação do objeto;

2.2.1 Representação do Objeto

Um objeto pode ser representado pela sua forma ou aparência. Em relação à forma, podemos representar um objeto por meio de pontos (ponto único ou múltiplos pontos) quando este ocupa pequenas regiões na imagem (VEENMAN; REINDERS; BACKER, 2001). Primitivas geométricas (retângulos e elipses) com o movimento modelado por translação, transformada afim (*affine*) ou projetiva (homografia) são indicadas para objetos rígidos simples, mas podem também representar os não-rígidos (COMANICIU; RAMESH; MEER, 2003). Outra forma é o contorno do objeto (sua fronteira), ou a silhueta, que é a região interna ao contorno. Ambos são ideais para representar formas complexas não-rígidas (YILMAZ; LI; SHAH, 2004). No presente trabalho utilizamos linhas como primitivas geométricas para representar o objeto de interesse no domínio do tráfego de automóveis (linhas divisórias da faixa de trânsito) e rastreamos o movimento de translação destas linhas.

A representação de objeto quanto à aparência, também pode ser combinada com representações de forma, temos a densidade de probabilidade da aparência, que pode ser paramétrica (Gaussiana) (PARAGIOS; DERICHE, 2002) ou não-paramétrica (histogramas) (COMANICIU; RAMESH; MEER, 2003), computando a cor ou textura do objeto a partir dos modelos de forma. Com os moldes (ou *templates*) a aparência do objeto é representada por simples formas geométricas ou silhuetas que carregam tanto informação espacial como de aparência, porém o objeto não pode sofrer variação considerável de pose no caso de rastreamento (FIEGUTH; TERZOPOULOS, 1997). Por sua vez, a aparência ativa é gerada pela modelagem simultânea de forma e aparência, onde a forma é representada por marcadores localizados na fronteira do objeto ou no interior de sua região (COOTES; EDWARDS; TAYLOR, 2001). Para cada marcador, um vetor de aparência é associado. Este modelo requer treinamento através de um conjunto de amostras usando o PCA (*Principal Component Analysis*) por exemplo. Em nossas cenas de tráfego a aparência do objeto é definida pela porcentagem de cor (amarela ou branca) na região das linhas divisórias;

2.2.2 Características da Imagem

Intimamente relacionada à representação do objeto, está a seleção das características, cuja propriedade esperada é a exclusividade, de forma que objetos possam ser facilmente distinguidos no espaço de características. Dentre as características visuais mais comuns temos a cor. A cor aparente de um objeto é influenciada principalmente pela distribuição espectral de iluminância e pela reflectância da superfície. O espaço de cor RGB (*red, blue, green*) é comumente usado porém, não é perceptualmente uniforme e suas dimensões são altamente correlacionadas, ou seja, a mudança em uma única dimensão no espaço de cor é percebida nas demais (PASCHOS, 2001). Já os espaços $YCbCr$, $L*u*v$ e $L*a*b$ ($Y/L*$ para luminância e $CbCr$, $u*v$ e $a*b$ para crominância) são perceptualmente uniformes, isto é, cores equidistantes são percebidas como equidistantes e alterar um dos parâmetros não afeta os outros dois, enquanto o HSV (*Hue, Saturation, Value*) é aproximadamente uniforme contudo, ambos são sensíveis à ruído e não há consenso em qual sistema é mais eficiente (SONG, 1996).

Outra característica frequentemente utilizada é a borda. A borda é usada para identificar fortes mudanças de intensidade na imagem sendo gerada pela fronteira do objeto. As bordas são menos sensíveis à iluminação do que a cor. Neste trabalho utilizamos o filtro de Sobel para detecção de bordas. Outro algoritmo popular de detecção de borda é o *Canny Edge* (CANNY, 1986) devido sua simplicidade e acurácia.

Em segmentação baseada em movimento e em algumas aplicações de rastreamento, utiliza-se o fluxo ótico (LUCAS; KANADE, 1981; BLACK; ANANDAN, 1996; SZELISKI; COUGHLAN, 1997). O fluxo ótico é um denso campo de vetores de deslocamento que define a translação de cada pixel em uma região. Utiliza uma restrição de brilho, onde considera-se o brilho constante dos pixels correspondentes em quadros consecutivos.

Uma característica pouco sensível à iluminação é a textura, que consiste na medida da variação de intensidade em uma superfície, quantificando propriedades como suavidade e regularidade. Há vários descritores de textura como as matrizes de co-ocorrência de níveis de cinza (do inglês GLCM) (HARALICK; SHANMUGAM; DINSTEIN, 1973), a medição de textura de Laws (LAWS, 1980), *wavelets* (MALLAT, 1989) e as pirâmides de revolução (GREENSPAN et al., 1994).

Uma característica é escolhida manualmente pelo usuário conforme o domínio da aplicação. Porém, houve um crescimento no uso de seleção automática de características, principalmente na área de reconhecimento de padrões. Esta seleção automática pode ser dividida entre métodos de filtro ou de envoltório (*wrapper*). Os métodos de filtro, selecionam características baseadas em critérios gerais como por exemplo, serem características não-correlatas (PCA). Já os métodos de envoltório, selecionam características discriminatórias para rastrear uma classe

específica de objeto, como é o caso do algoritmo *Adaboost* (TIEU; VIOLA, 2000).

De todas as características, a cor é a mais utilizada porém, devido a sua sensibilidade à variação de iluminação, outras características podem ser combinadas para modelar a aparência do objeto e melhorar o desempenho da análise. No ambiente de tráfego analisado, as características discriminantes utilizadas foram cor e borda, para classificar a cor da linha divisória e sua posição, respectivamente.

2.2.3 Detecção do Objeto

Todo método de classificação ou rastreamento requer um mecanismo de detecção de objetos, seja a cada quadro ou quando o objeto aparece pela primeira vez. A forma mais comum, é utilizar a informação em um único quadro. Porém, pode-se usar a informação temporal adquirida de uma sequência de quadros para reduzir falsas detecções.

A detecção de objeto é por si só um amplo campo de pesquisa, segue portanto somente uma breve descrição de alguns métodos mais utilizados.

Na subtração de fundo, um modelo de plano de fundo é criado para representar a cena e encontra desvios no modelo para cada quadro. Qualquer mudança significativa em uma região da imagem em relação ao plano de fundo, significa um objeto em movimento. Apesar de existirem métodos eficientes de subtração de fundo, estes têm como limitação a exigência de plano de fundo estático e câmeras fixas. Câmeras móveis distorcem o modelo do plano de fundo porém, Kanade et al. (1998) trataram tal problema regenerando os modelos de plano de fundo para pequenas janelas de tempo (três quadros), mas também assumindo cenas planas e pouco movimento entre os quadros.

O plano de fundo e a câmera estão em movimento no ambiente de tráfego e, em relação ao objeto de interesse (linhas divisórias), há mudanças significativas entre os quadros (por exemplo, oclusão de outros veículos), portanto a subtração de fundo não é recomendada nestas situações.

Uma alternativa de detecção do objeto é por meio da segmentação. A segmentação tem como objetivo particionar a imagem em regiões similares e dois pontos críticos são: o critério para um bom particionamento e o método para executá-lo eficientemente. Algumas das técnicas mais comuns são: *K-Means* (STEINHAUS, 1956), *Mean-Shift Clustering* (COMANICIU; MEER, 2002), *Graph-Cuts* (SHI; MALIK, 2000) e Contornos Ativos (PARAGIOS; DERICHE, 2000).

Neste trabalho, a visão computacional tem o objetivo de detectar e classificar linhas (limites da faixa de trânsito). Portanto, é desnecessário o uso da segmentação em regiões.

Outra alternativa seriam os detectores de pontos, que podem ser usados para encontrar pontos de interesse em imagens que possuem textura expressiva. Apresenta a qualidade de ser invariante à mudanças na iluminação e ponto de vista da câmera. Detectores bem conhecidos são: detector de Harris (HARRIS; STEPHENS, 1988), detector KLT (TOMASI; KANADE, 1991) e o detector SIFT (*Scale Invariant Feature Transform*) (LOWE, 2004). Dentre os detectores de objeto, o SIFT é o mais resiliente à deformações na imagem (MIKOLAJCZYK; SCHMID, 2005). Como os objetos de interesse deste trabalho são linhas, o detectores de pontos não são adequados para a tarefa de detecção destes objetos.

Neste trabalho portanto, o melhor método de detecção do objeto é o detector de curvas. Detectores de curvas são usados para encontrar qualquer tipo de curva com descrição parametrizada (linha reta, elipse, círculo). O algoritmo detector de curvas mais conhecido e utilizado é a Transformada de Hough (DUDA; HART, 1972), o qual foi escolhido neste trabalho para detectar as linhas divisórias que limitam a faixa de trânsito. No domínio de cenas de tráfego, outros métodos podem ser utilizados na detecção de linhas de trânsito como o gradiente de pixels brancos em linhas horizontais da imagem (AMSEL et al., 2009). A transformada de Hough foi adotada inicialmente por fazer parte do algoritmo de demonstração do Matlab. Porém, esta técnica se mostrou mais adequada para o objetivo deste trabalho, pois parece mais robusta em relação à técnica do gradiente citada anteriormente, considerando que é possível filtrar resultados baseados em janelas de ângulos e distância em relação à origem da imagem.

A transformada de Hough foi proposta e patenteada por Hough (1962) a fim de detectar linhas em imagens. Porém, foram Duda e Hart (1972) que proporam a forma como conhecemos atualmente, provendo meios para a detecção de outros tipos de curvas (círculos e elipses) e consiste na busca de instâncias dos objetos (por exemplo, bordas) por meio de um procedimento de votação no espaço de parâmetros.

No espaço da imagem, uma linha reta pode ser expressa pela equação $y = mx + c$, sendo necessário especificar dois pontos $(x_1, y_1; x_2, y_2)$ para definir uma reta. No espaço de parâmetros, a equação da reta pode ser reescrita como $c = -mx + y$, neste caso, somente um ponto especificado pelos parâmetros da reta (m,c) define diversas retas que passam por este ponto, sendo m um parâmetro de inclinação (*slope*) e c um parâmetro de interceptação (*intercept*). A proposta da transformada de Hough é agrupar pontos das bordas extraídas de uma imagem como hipóteses do objeto de interesse. Este agrupamento é feito por meio de um procedimento de votação sobre um conjunto de objetos parametrizados da imagem (SHAPIRO; STOCKMAN, 2001).

Os valores de m e c são ilimitados ($-\infty \leq (m,c) \leq \infty$), o que representa um alto custo computacional. Uma forma de representar as retas com parâmetros limitados é por meio de

coordenadas polares, neste caso, definidos como ρ e θ. O parâmetro ρ representa a distância perpendicular de uma linha à origem da imagem (x e y iguais a zero) e θ é o ângulo entre ρ e a origem da imagem (fig.2.1).

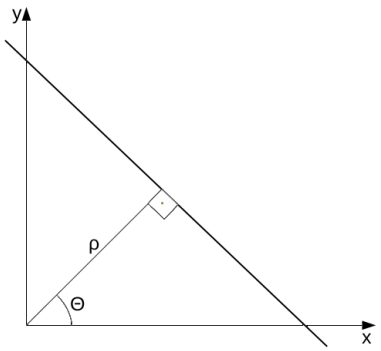

Figura 2.1: Parâmetros da transformada de Hough.

Fonte: Autor.

Desta forma, uma linha pode ser descrita por:

$$\rho = x\cos\theta + y\sin\theta, \tag{2.1}$$

onde o par (ρ, θ) é único para $\rho \geq 0$ e $\theta \in [0, 2\pi]$. A região delimitada por estes valores também é conhecida como espaço de Hough, o qual apresenta as seguintes propriedades ((DUDA; HART, 1972)):

Propriedade 2.2.1. *Um ponto no espaço da imagem corresponde a uma senoide no espaço de parâmetros.*

Propriedade 2.2.2. *Um ponto no espaço de parâmetros corresponde a uma reta no espaço da imagem.*

Propriedade 2.2.3. *Pontos que caem na mesma reta do espaço da imagem correspondem a curvas com o mesmo ponto em comum no espaço de parâmetros.*

Propriedade 2.2.4. *Pontos que caem na mesma curva no espaço de parâmetros correspondem às retas que passam por um ponto no espaço da imagem.*

Sendo assim, em um arranjo acumulador $A(\rho, \theta)$ incrementa-se o valor da célula cujos parâmetros satisfaçam a equação 2.1 para cada ponto (x, y) da imagem acima de um limiar (por exemplo, intensidade do pixel de borda). As células que possuem os maiores valores representam pontos colineares na imagem, definindo então uma reta por célula. Definido o limiar de decisão para o espaço de Hough, selecionam-se as células com a "votação" acima deste limiar.

No sistema de visão deste trabalho, a transformada de Hough é utilizada para detectar as linhas divisórias que limitam a faixa de trânsito. A seleção destas linhas é baseada nos máximos locais do espaço de Hough em uma região de interesse da imagem (metade inferior da imagem).

Após a detecção do objeto buscam-se rótulos de classe para associar a este objeto a fim de transformar os dados obtidos em informação, sendo este o objetivo de outro grande campo de pesquisa que é o aprendizado de máquina (BRADSKI; KAEHLER, 2008). Os classificadores de objeto fazem parte desta área.

2.2.4 Classificação do Objeto

Alguns dos classificadores mais conhecidos possuem em seu algoritmo um detector do objeto e podem ser discriminativos ou generativos, apresentando ainda um sistema de aprendizado supervisionado ou não-supervisionado.

Um algoritmo discriminativo apresenta a probabilidade de um rótulo (R) em relação aos dados (D) isto é, ($P(R \mid D)$), já o modelo generativo retorna uma distribuição dos dados dado um rótulo ($P(D \mid R)$).

Os algoritmos com aprendizado não-supervisionado são aqueles cujo vetor de dados não é rotulado e onde pretende-se verificar se os dados caem naturalmente em algum grupo. Estes algoritmos são comumente usados em segmentação.

Já em um sistema de aprendizado supervisionado, aprendem-se várias vistas do objeto gerando uma função que mapeia as entradas para as saídas desejadas. A formulação padrão de um aprendizado supervisionado é um problema de classificação onde o *aprendiz* aproxima o comportamento de uma função, gerando uma saída na forma de valor contínuo (*regressão*), ou em um rótulo de classe (*classificação*). No contexto de detecção de objeto, os exemplos de aprendizado são compostos de pares de características do objeto e uma classe de objeto associada onde ambos são definidos manualmente. É importante também selecionar características que sejam discriminantes entre as classes e, então, diferentes aparências do objeto podem ser aprendidas através de técnicas como redes neurais, árvores de decisão, *adaptive boosting* ou *support vector machines*. Estas técnicas calculam uma hipersuperfície que separa uma classe da outra em um alto espaço dimensional. Alguns dos algoritmos de aprendizado supervisionado mais comuns são: *K Vizinhos mais Próximos* (FIX; HODGES, 1951), *Árvores de Decisão* (BREIMAN et al., 1984) e *Redes Neurais* (RUMELHART; HINTON; WILLIAMS, 1988).

Geralmente o aprendizado supervisionado requer uma grande quantidade de amostras de cada classe do objeto, a qual é rotulada manualmente. Uma forma de reduzir esta condição

é acrescentar um *co-treinador*, com o intuito de treinar dois classificadores com um pequeno conjunto de dados rotulados, onde as características usadas para cada classificador são independentes. Após o treinamento, cada classificador é usado para associar dados não rotulados ao conjunto de treinamento de outro classificador. Esta técnica foi usada com sucesso nos algoritmos *Adaboost* (FREUND; SCHAPIRE, 1995; VIOLA; JONES; SNOW, 2003) e *Support Vector Machines* (BOSER; GUYON; VAPNIK, 1992).

Neste trabalho, a classificação das linhas detectadas é baseada na porcentagem de pixels coloridos na região de cada linha, não havendo a necessidade de um aprendizado para esta tarefa. O aprendizado supervisionado discriminativo foi utilizado na etapa lógico-probabilística para definir o peso de cada sentença lógica da formalização do domínio de tráfego, isto é, um conjunto de dados do domínio analisado é rotulado manualmente é utilizado como evidência para aproximar uma função de probabilidade condicional dos predicados consultados e então estimar os pesos de cada sentença. Isto será discutido na seção 2.4.1.

2.2.5 Considerações

Tanto o rastreamento (*tracking*) como a classificação de objetos representam um extenso campo de pesquisa, em constante evolução e ainda assim, com muitos desafios a serem superados como vídeos em ambientes não estruturados, comprimidos ou com ruído, com vários objetos em um campo de vista pequeno, câmera em movimento, etc. Nota-se que as considerações utilizadas para tratar alguns problemas como suavidade de movimento, constância de iluminação ou alto contraste, são geralmente violadas em cenários realistas limitando a aplicação prática das técnicas.

Em geral, um dos aspectos negligenciados no desenvolvimento de rastreadores ou classificadores é acrescentar informação contextual para colaborar no reconhecimento, pois o algoritmo não precisaria aplicar uma série de restrições a fim de se obter um desempenho médio para vários cenários (YILMAZ; JAVED; SHAH, 2006), ou seja, em um ambiente de tráfego por exemplo, a detecção das linhas divisórias de uma faixa de trânsito pode ser restrita à localização da estrada, ignorando construções nas laterais da via ou o céu.

Neste trabalho utilizaremos a informação contextual também na etapa de inferência lógico-probabilística, com o objetivo de associar o conhecimento de regras de trânsito na classificação da posição e comportamento do veículo. A formalização do ambiente de tráfego foi baseada em lógica probabilística de primeira ordem, a qual será apresentada no capítulo 3.3. Além da informação contextual, devemos tratar também a informação espacial na modelagem do ambiente de tráfego. Na seção seguinte, apresentamos as técnicas de raciocínio espacial qualitativo

que inspiraram a formalização dos aspectos espaciais do domínio considerado.

2.3 Raciocínio Espacial Qualitativo

Em nossa interação cotidiana com o mundo físico, o raciocínio espacial é feito na maioria dos casos através de abstrações qualitativas ao invés de um completo conhecimento quantitativo à priori (COHN; HAZARIKA, 2001). A partir desta constatação, buscou-se criar uma representação e raciocínio de aspectos espaciais do mundo, com cálculos mais eficientes, expressivos e úteis (COHN; RENZ, 2007), o que é de grande importância para a área de Inteligência Artificial (IA). O Raciocínio Espacial Qualitativo (REQ) foi escolhido para representar os aspectos espaciais neste trabalho visando esta expressividade e eficiência de cálculos, considerando o grande volume de informação proveniente das imagens de vídeo.

A essência do Raciocínio Qualitativo é encontrar meios de representar propriedades contínuas do mundo através de sistemas discretos de símbolos (COHN; RENZ, 2008). O Raciocínio Espacial Qualitativo (REQ) é a formalização do conhecimento espacial baseado em relações primitivas definidas entre entidades espaciais elementares (STOCK, 1997). Existem diversas abordagens com relação à estas entidades. Em (MORATZ; RENZ; WOLTER, 2000), se especifica uma relação qualitativa entre segmentos de linha orientados a fim de representar e raciocinar sobre informações de orientação. Por sua vez, em (LIGOZAT, 1998), uma rede binária de restrições (RBR) relaciona a direção cardeal (norte, sul, leste, oeste e intermediários) entre localizações em um espaço 2D. Em (RANDELL; COHN; CUI, 1992), especificam-se regiões espaciais na relação primitiva de conexão (C/2) entre duas regiões e em (COHN; RENZ, 2007), discute-se sobre as diferentes entidades espaciais primitivas e principalmente sobre as diversas relações que podem ser representadas, como topologia, tamanho, forma e distância entre outros.

Desenvolvimento posteriores inspirados nas técnicas citadas acima ou como extensão das mesmas, apresentam outras formalizações, como o raciocínio sobre movimento e perfis de profundidade (SANTOS; SHANAHAN, 2003; SANTOS, 2007; SOUTCHANSKI; SANTOS, 2008), considerando o ponto de vista do observador e o movimento de ambos objeto e observador.

A seguir descreveremos uma das principais teorias de REQ que serviram de base para a modelagem deste trabalho: o cálculo sobre conexão de regiões.

2.3.1 Cálculo sobre Conexão de Regiões

O Cálculo sobre Conexão de Regiões (RCC em inglês) (RANDELL; COHN; CUI, 1992; COHN; HAZARIKA, 2001), é uma axiomatização de regiões espaciais, que tem como base a

relação primitiva de conexão (C/2). Sejam duas regiões x e y, a relação C(x, y)(lê-se "x conecta com y"), é verdadeira se e somente se os limites topológicos das regiões x e y tem pelo menos um ponto em comum conforme os seguintes axiomas:

$\forall xC(x,x)$ (Relação reflexiva)

$\forall xyC(x,y) \rightarrow C(y,x)$ (Relação simétrica)

$\forall xy\forall z[C(z,x) \leftrightarrow C(z,y)] \rightarrow x = y$

Assumindo a relação binária C/2 e variáveis x, y e z para regiões espaciais, pode-se raciocinar sobre formas do tipo: "dado x em relação R_1 a y, e y em relação R_2 a z, quais relações são verdadeiras entre x e z". Definindo então um conjunto de relações mereotopológicas com seus significados como segue:

- *P(x,y)*, "x é parte de y";
- *O(x,y)*, "x sobrepõe y";
- *DR(x,y)*, "x é distinto de y";
- *PP(x,y)*, "x é parte própria de y";
- *Pi/2* e *PPi/2* são relações inversas de *P/2* e *PP/2* respectivamente;
- *DC(x,y)*, "x está desconectado de y";
- *EQ(x,y)*, "x é igual a y";
- *PO(x,y)*, "x sobrepõe y parcialmente";
- *EC(x,y)*, "x é externamente conectado a y";
- *TPP(x,y)*, "x parte própria tangencial de y";
- *NTPP(x,y)*, "x parte própria não-tangencial de y";
- *TPPi/2* e *NTPPi/2* são relações inversas de *TPP/2* e *NTPP/2* respectivamente.

O conjunto constituído pelas relações DC(x, y), EQ(x, y), PO(x, y), EC(x, y), TPP(x, y), NTPP(x, y), TPPi/2 e NTPPi/2 é mutuamente exaustivo e disjunto, isto é, pelo menos uma das relações deve ocorrer e não há elementos em comum no conjunto, agora denominado RCC-8.

A transição topológica entre relações RCC-8 é apresentada na figura 2.2 como um Diagrama Conceitual de Vizinhança (CND em inglês), que resumidamente é um grafo representando em seus vértices, relações de alguns objetos específicos, e em suas arestas, a transição contínua entre estas relações (RANDELL; COHN; CUI, 1992).

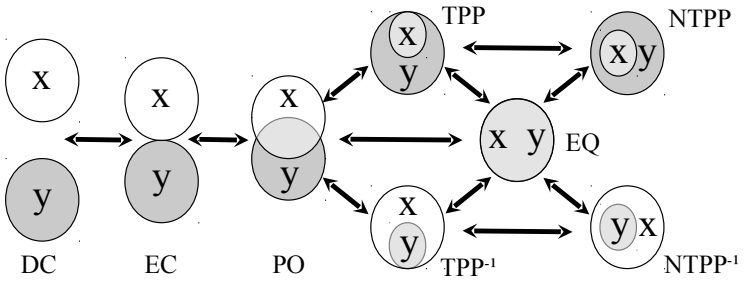

Figura 2.2: Diagrama conceitual de vizinhança (CND) das relações RCC-8.

Fonte: Randell, Cohn e Cui (1992).

Observa-se que entre vértices adjacentes não há possibilidade das respectivas regiões assumirem outras relações. A CND, portanto, é uma importante ferramenta de raciocínio espacial, considerando a continuidade de transição.

Apesar do RCC representar as relações qualitativas entre regiões espaciais, este não considera o ponto de vista do observador. Já a proposta *Linhas de Visão* (GALTON, 1994), apresenta uma teoria de oclusão de corpos convexos usando um conjunto discreto de 14 relações como "*parcialmente esconde*" ou "*escondido por*", o que levou Randell, Witkowski e Shanahan (2001) a desenvolverem o Cálculo sobre Oclusão de Regiões (*Region Occlusion Calculus* ou ROC), o qual permite raciocinar sobre relações espaciais conforme o movimento do objeto ou ponto de vista.

O ROC assume a ontologia do RCC-8 e estende o cálculo de linhas de visão de (GALTON, 1994) para raciocinar sobre distâncias relativas entre corpos (convexos e côncavos) em eventos de oclusão e transições entre eventos de oclusão, a fim de modelar os efeitos de paralaxe de movimento, seja do objeto ou do ponto de vista (fig.2.3).

O ROC-20 usa o conjunto de relações diádicas do RCC-8 para modelar a relação espacial entre corpos, volumes e imagens. Essa distinção se faz pela introdução de duas novas funções: "*região(x)*", lido como "região ocupada por x" e "imagem(x,v)", lido como "a imagem de x com relação ao ponto de vista v". A função região/1, mapeia um corpo no volume do espaço que ele ocupa, e imagem/2, mapeia um corpo e um ponto de vista para sua imagem (RANDELL; WITKOWSKI; SHANAHAN, 2001). Com esta abordagem, pode-se raciocinar sobre o estado dos objetos conforme eles (ou o observador) se movimentam pelo ambiente (SANTOS,

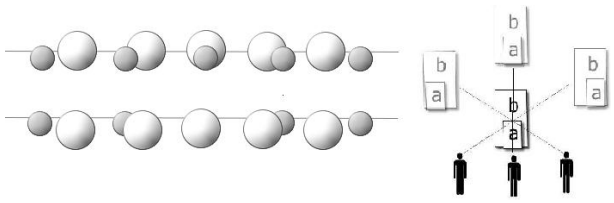

Figura 2.3: Oclusão em ação. Ponto de vista fixo na esquerda, ponto de vista móvel na direita.

Fonte: Randell, Witkowski e Shanahan (2001).

2003). Embora o ROC-20 seja um conceito estático, pode-se introduzir restrições dinâmicas sobre mudanças no tempo das relações de oclusão, a fim de criar um conceito dinâmico de oclusão.

Santos e Shanahan (2002) tratam a distância entre regiões como uma função primitiva a fim de classificar o grau de deslocamento entre dois objetos conforme há mudanças nos dados do sensor (visão estéreo), possibilitando a criação de predicados dinâmicos do tipo *aproximando(i(a,v,t), i(b,v,t))*, lido como " a imagem de *a* e de *b* estão se aproximando conforme notado pelo ponto de vista *v* no tempo *t*". Com esta proposta, tornou-se possível raciocinar sobre relações espaciais do ponto de vista do observador estando o mesmo ou o objeto em movimento.

O trabalho de Santos (2007) apresenta um formalismo lógico para representar o conhecimento sobre objetos no espaço e seus movimentos, construído através do ponto de vista de um observador inserido em um mundo dinâmico. Esta teoria de raciocínio sobre espaço e tempo utiliza os conceitos do RCC (RANDELL; COHN; CUI, 1992) e ROC (RANDELL; WITKOWSKI; SHANAHAN, 2001) associados ao processo de assimilação de dados sensoriais por abdução proposto por Shanahan (1996), utilizando como informação espacial elementar os perfis de profundidade obtidos da *fatia horizontal* da cena (fig.2.4).

Na figura 2.4a dois objetos *a* e *b* são observados do ponto de vista *v*, na figura 2.4b o perfil de profundidade (relativo ao ponto de vista v) representa os objetos *a* e *b* respectivamente pelos picos *p* e *q* (SANTOS; SHANAHAN, 2003).

O eixo disparidade de um perfil de profundidade é restringido pelo ponto mais distante que o sensor pode detectar, representado na figura 2.4 por *L*.

Com a informação dos perfis de profundidade foi possível criar predicados de relação entre os picos de dois corpos relativo ao ponto de vista do robô (ou veículo). A criação de hipóteses sobre a relação entre objetos utilizando profundidade e tamanho no mesmo formalismo é a proposta do cálculo de perfis de profundidade dinâmico (do inglês DDPC) (SANTOS, 2007)

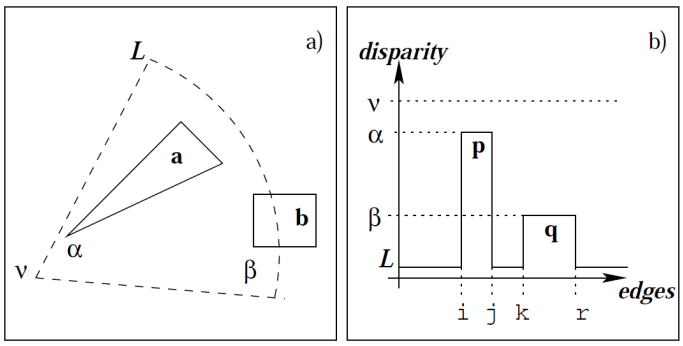

Figura 2.4: Objetos observados de um ponto de vista *v* e seus respectivos perfis de profundidade.

Fonte: Santos e Shanahan (2003).

para estender o exposto por Santos e Shanahan (2002), conectando os predicados que definem mudanças em uma cena à possíveis interpretações de movimento dos objetos ou observador.

O DDPC considera três restrições de domínios que atendem à maioria dos casos:

1. *Persistência do Objeto*: os objetos do domínio não podem aparecer ou desaparecer;
2. *Continuidade*: a mudança na localização do objeto é devido a um movimento contínuo;
3. *Substancialidade*: um objeto não passa através de outro;

Estas restrições estão ligadas a algumas classes particulares de relações espaço-temporais de corpos físicos à partir do ponto de vista do observador, denominadas de hipóteses de objeto-observador, as quais estão presentes em axiomas usados no processo de assimilação abdutivo de hipóteses do mundo. Isto tornou possível ligar as transições nos atributos dos objetos observados com conceitos de alto-nível sobre o movimento dos mesmos.

Posteriormente, Soutchanski e Santos (2008) propõem o uso do Cálculo de Situações (SC em inglês) para criar uma Teoria de Profundidade e Movimento (do inglês TDM) utilizando os conceitos do DPC.

O cálculo de situações (SC) é uma linguagem lógica de segunda ordem usada para representar dinamicamente um mundo em mudança (REITER, 2001). Toda alteração no mundo é resultado de *ações*. Uma possível história do mundo, correspondente à uma sequência de ações, é um termo de primeira ordem denominado *situação*. Funções e predicados que variam conforme a situação, são chamados de *fluentes*.

O formalismo resultante da TDM permite através dos dados do sensor, tanto a assimilação como o raciocínio sobre o mundo em movimento e pode fornecer consequência lógica de

fenômenos puramente qualitativos, como por exemplo, objetos se aproximando mutuamente.

Neste trabalho, a assimilação das hipóteses do mundo é baseada na transição espacial e de atributos das linhas divisórias de uma faixa de trânsito. As ações inferidas são referentes ao observador (não ao objeto). Como exemplo de ação do observador, temos a mudança de faixa à esquerda ou à direita, cujos predicados foram definidos como *crossingLeft(time)* e *crossingRight(time)*, respectivamente. Nota-se que estes predicados omitem o observador pois o ponto de vista (egocêntrico) é implícito.

Quanto à representação espacial do nosso objeto de interesse, temos como exemplo o predicado *dividerL(WDashed,Under,100)*, que define uma linha divisória branca tracejada (*WDashed*) ao lado esquerdo (*dividerL*) e sob (*Under*) o observador num instante de tempo (100 - centésimo quadro).

Definida a forma de representação espacial do domínio, temos de criar as regras para a formali-zação do ambiente de tráfego rodoviário. Como citado na introdução (Cap.1), utilizamos a lógica probabilística para modelar o nosso problema. Na próxima seção apresentamos uma revisão das técnicas de lógica probabilística.

2.4 Lógica Probabilística

Entre as décadas de 1970 e 1980 houve um grande interesse por sistemas especialistas, nos quais regras são aplicadas a fatos existentes, produzindo novos fatos como conclusão (RUSSELL; NORVIG, 2004). A fim de tratar regras e fatos incertos, criou-se a abordagem dos *fatores de certeza*, cujas regras carregam números representando *graus de certeza*, que são propagados para as conclusões durante a inferência.

Fatores de certeza porém, não conseguem tratar de forma correta a inferência bidirecional. Por exemplo, com as duas regras $(A \leftarrow B) : c_1$ e $B : c_2$ podemos concluir A com grau de certeza $c_1 \times c_2$ se B for verdadeiro com grau de certeza c_2, mas não podemos concluir nada sobre B. Por sua vez, as probabilidades $P(A \mid B)$ e $P(B)$ são restrições para $P(B \mid A)$ como discutido por Pearl (1988). Outra desvantagem dos fatores de certeza é que não há uma semântica clara e algumas vezes são apresentados resultados inesperados e não-intuitivos.

A procura por semântica mais clara para regras com certeza variável, dentre outras questões, levou à criação de abordagens como as Redes Bayesianas (PEARL, 1988). Redes Bayesianas, são modelos orientados que especificam uma distribuição de probabilidade condicional para cada variável, dado alguma variável pai, expressando uma relação de causalidade entre elas. Porém, é uma técnica essencialmente proposicional com muito menos expressividade do

que sistemas lógicos, pois não consegue representar relações entre objetos ou funções.

Esta limitação motivou a definição de uma semântica mais precisa para probabilidades em sistemas lógicos, o que levou à lógica probabilística, sendo um dos trabalhos mais influentes neste assunto o de Nilsson (1986). Neste trabalho, é estabelecido de forma sistemática a determinação de intervalos de probabilidades para conjuntos de questões desde que, em princípio, o conjunto de evidências seja consistente com toda a faixa de probabilidade determinada. Porém, o problema torna-se intratável mesmo com um número modesto de sentenças, uma vez que todos os mundos possíveis devem ser enumerados. Ainda assim, esta proposta tornou mais fácil expressar probabilidades subjetivas como "*um pássaro voa com probabilidade 0.9*", mas não permitia a inserção de fatos estatísticos como "*90% dos pássaros voam*". Isso foi incluído por Bacchus (1990) e discutido formal e filosoficamente quando é correto usar o fato estatístico: "90% dos pássaros voam" (isto é, em cada mundo possível 90% dos pássaros voam), em vez de assumir a incerteza "um pássaro escolhido aleatoriamente voa com probabilidade de 0.9" (ou seja, a soma das probabilidades de todos os mundos possíveis onde um pássaro voa é de 0,9).

Estes trabalhos não tinham, até então, algoritmos de inferência eficientes, os quais foram alvos da Lógica Probabilística de Primeira Ordem (do inglês *First-Order Probabilistic Logic* ou FOPL). Os trabalhos em FOPL podem, segundo Pearl (1988), ser divididos em sistemas *extensionais* e *intensionais*. No primeiro grupo, as declarações são mais procedimentais, representando licenças para propagar fatos de verdadeiros ou falsos para graus variáveis de certeza. No segundo grupo, as declarações são restrições na distribuição de probabilidade dos mundos possíveis.

Milch e Russell (2006) apresentam uma taxonomia das abordagens referentes aos sistemas intensionais onde as abordagens de FOPL se diferenciam basicamente pelo espaço de estados, ou seja, os conjuntos de estados aos quais se associa uma probabilidade. Por sua vez, para Braz, Amir e Roth (2008), os sistemas intensionais são classificados em cinco famílias baseadas no tipo de inferência: regras de dedução, computação exaustiva de derivações, amostragem, *lifted inference* e Construção de Modelo de Base de Conhecimento (do inglês *Knowledge Base Model Construction* ou KBMC), sendo esta última (KBMC), a família mais proeminente dos modelos de FOPL.

As abordagens de KBMC geram um modelo gráfico proposicional a partir de uma especificação de linguagem de primeira ordem que responde a uma consulta. Como esta construção geralmente é feita especificamente para a consulta atual, partes irrelevantes do grafo podem ser eliminadas aumentando assim a eficiência. Algumas abordagens constroem redes Bayesianas, como a Lógica Bayesiana (*Bayesian Logic* ou BLOG) (MILCH et al., 2005) e os Modelos Re-

lacionais Probabilísticos (*Probabilistic Relational Models* ou PRM) (KOLLER; PFEFFER, 1998), dentre outros. Outras abordagens constroem redes de Markov, como as Redes de Markov Relacionais (*Relational Markov Networks* ou RMN) (TASKAR; ABBEEL; KOLLER, 2002) ou as Redes Lógicas de Markov (*Markov Logic Networks* ou MLN) (RICHARDSON; DOMINGOS, 2006).

Diferente das redes Bayesianas, Redes de Markov são modelos não orientados que usam pesos para definir a probabilidade relativa das instanciações. Em domínios relacionais, algumas variáveis aleatórias podem ocasionalmente depender umas das outras sem possuírem uma relação clara de causalidade entre elas. Neste caso, os modelos que usam rede de Markov têm vantagem sobre os de rede Bayesiana, pois não possuem restrição de aciclicidade, i.e, podem haver ciclos no grafo, o que facilita a modelagem do problema. Porém, há a desvantagem do aprendizado ser mais difícil em modelos gráficos não-orientados do que nos modelos orientados. Já a restrição do mundo ser finito é aplicada tanto às redes de Markov quanto nas redes Bayesianas.

Os trabalhos em Redes Lógicas de Markov ou MLN (RICHARDSON; DOMINGOS, 2006) vêm evoluindo rapidamente nos últimos anos, tendo como principal diferencial uma semântica simples mantendo a expressividade da lógica de primeira ordem. Sendo esta abordagem acompanhada de um software (*Alchemy*) com um bom suporte e já aplicado em domínios reais atualmente, as MLNs estão em um estágio de desenvolvimento mais avançado em relação a qualquer outra técnica de FOPL proposta (BRAZ; AMIR; ROTH, 2008). Além disso, Campos, Cozman e Luna (2009) apontam que apesar de serem necessários mais estudos em relação às condições de Markov em redes Bayesianas, tudo indica que os modelos gráficos não-orientados serão as ferramentas escolhidas para a lógica probabilística devido à forte contribuição da Lógica de Markov.

Um histórico e comparativo mais profundos em lógica probabilística são apresentados por Milch e Russell (2006), Braz, Amir e Roth (2008), Campos, Cozman e Luna (2009). Na próxima seção

2.4.1 Rede Lógica de Markov

Redes Lógicas de Markov (do inglês *Markov Logic Networks* ou MLN) foram desenvolvidas visando uma linguagem unificada que servisse como camada de interface para a área de Inteligência Artificial (IA). Esta interface serve de linguagem base para áreas como: representação de conhecimento, raciocínio automático, modelos probabilísticos e aprendizado de máquina (DOMINGOS; LOWD, 2009).

A ideia básica das MLNs é suavizar as restrições impostas por uma base de conhecimento

BC de primeira ordem, ou seja, se um mundo violar alguma fórmula da BC este mundo será menos provável, ou seja, se uma cláusula de uma sentença ou fórmula for falsa, esta sentença terá menor probabilidade, mas não será impossível como na lógica de primeira ordem. Quanto menos fórmulas um mundo violar, mais provável ele será. Cada fórmula tem um peso associado que reflete quão forte é a sua restrição: quanto maior for o peso, maior será a diferença de probabilidade entre um mundo que satisfaz a fórmula e o que não satisfaz, estando o restante igual (DOMINGOS; LOWD, 2009). Por exemplo, em lógica de primeira ordem se dissermos que "*fumar causa câncer*" e que "*amigos possuem os mesmos hábitos*", um mundo onde um amigo de um fumante não tiver câncer ou não fumar é um mundo impossível. Já em redes lógicas de Markov, o mundo do exemplo citado passa a ser possível com uma determinada probabilidade. A tabela 2.1 apresenta um exemplo de lógica de primeira ordem e de MLN. Am/2 é abreviação para Amigos/2, Fu/1 para Fuma/1 e Ca/1 para Câncer/1.

Tabela 2.1: Exemplo de uma BC de primeira ordem e MLN.

	MLN	
Português e Lógica de Primeira Ordem	**Forma Clausal**	**Peso**
"Amigos de amigos também são amigos." $\forall x \forall y \forall z Am(x,y) \wedge Am(y,z) \Rightarrow Am(x,z)$	$\neg Am(x,y) \vee \neg Am(y,z) \vee Am(x,z)$	0.7
"Fumar causa câncer." $\forall x Fu(x) \Rightarrow Ca(x)$	$\neg Fu(x) \vee Ca(x)$	1.5
"Se duas pessoas são amigas e uma fuma, então a outra também fuma." $\forall x \forall y Am(x,y) \wedge Fu(x) \Rightarrow Fu(y)$	$\neg Am(x,y) \vee \neg Fu(x) \vee Fu(y)$	1.1

Fonte: Autor "adaptado de" Richardson e Domingos (2006)

A sintaxe das fórmulas de MLN é a mesma da lógica de primeira ordem (LPO) com o diferencial de se atribuir um peso às fórmulas. A LPO pode ser vista como um caso particular da MLN se considerarmos o caso de atribuir pesos infinitos às fórmulas da MLN.

Uma MLN é um arcabouço para a construção de redes de Markov e consiste de um conjunto de fórmulas ponderadas de primeira ordem e um universo de objetos, vide exemplo na figura 2.5. Sua semântica é a mesma da rede de Markov. Em uma rede de Markov, temos um grafo não orientado para um modelo proposicional, onde os elementos de uma sentença são os nós e a distribuição de probabilidade sobre estes elementos são representados pelas arestas. Em uma MLN, seus elementos são instanciações de todas as fórmulas dado o universo de objetos. O potencial de um elemento é definido como o exponencial do seu peso caso o elemento seja verdadeiro. Seus nós representam variáveis e as arestas correspondem a alguma noção de interação probabilística direta entre variáveis vizinhas, sendo esta interação parametrizada por funções potenciais. Há uma função potencial para cada subgrafo completamente conectado (*clique*) dentro

do grafo, onde esta função potencial é uma função não-negativa com valor real do estado do *clique* correspondente. A distribuição conjunta de um conjunto de dados $X = (X_1, X_2, ..., X_n) \in \mathscr{X}$ representada por uma rede de Markov é dada por:

$$P(X = x) = \frac{1}{Z} \prod_i \phi_i(x_{\{i\}}), \tag{2.2}$$

em que $x_{\{i\}}$ é o estado do $i-ésimo\ clique$ e Z, conhecido como *função de partição*, é um fator de normalização dado por $Z = \sum_{x \in \mathscr{X}} \prod_i \phi_i(x_{\{i\}})$ para garantir que $\sum_{x \in \mathscr{X}} P(X = x) = 1$. Por sua vez, ϕ_i é a função potencial de valor real e não-negativo para o estado de um *clique*. Uma rede de Markov pode ser convenientemente representada por *modelos log-lineares*:

$$P(X = x) = \frac{1}{Z} exp\left(\sum_i w_i f_i(x)\right), \tag{2.3}$$

onde um elemento (*feature*) $f_i(x)$, em uma relação direta com a função potencial da equação 2.2, corresponde a cada estado possível de $x_{\{i\}}$ de cada *clique*, com peso $w_i = log\phi_i(x_{\{i\}})$. Esta representação é exponencial no tamanho dos *cliques* porém, pode-se especificar um número bem menor de elementos (por exemplo, funções lógicas do estado do *clique*), permitindo uma representação mais compacta. Na figura 2.5 apresentamos uma rede instanciada de Markov obtida pela aplicação das duas últimas fórmulas da tabela 2.1 para as constantes Anna(A) e Bob(B).

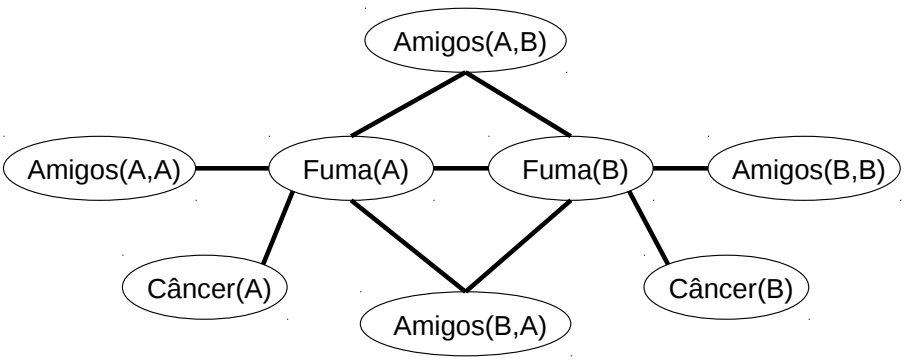

Figura 2.5: Rede instanciada de Markov.

Fonte: Autor "adaptado de" Domingos e Lowd (2009).

Segue abaixo a definição de rede lógica de Markov:

Definição 2.4.1. *Uma rede lógica de Markov L é um conjunto de pares de* (F_i, w_i) *onde* F_i *é uma formula em lógica de primeira ordem e* w_i *é um número real. Um conjunto finito de constantes*

$C = \{c_1, c_2, ..., c_C\}$ define uma rede de Markov $M_{L,C}$ como segue:

1. *$M_{L,C}$ contém um nó binário para cada instanciação possível de cada predicado em L. O valor do nó é 1 se o predicado instanciado for verdadeiro e 0 caso contrário;*

2. *$M_{L,C}$ contém um elemento para cada instanciação possível de cada fórmula F_i em L. O valor deste elemento é 1 se a fórmula básica for verdadeira e 0 caso contrário. O peso do elemento é w_i associado a F_i em L.*

Da definição 2.4.1 e equações 2.2 e 2.3, a distribuição de probabilidade sobre os mundos possíveis x na rede básica de Markov $M_{L,C}$ é dada por:

$$P(X = x) = \frac{1}{Z} exp\left(\sum_{i=1}^{F} w_i f_i(x)\right) = \frac{1}{Z} \prod_i \phi_i(x_{\{i\}})^{n_i(x)}, \tag{2.4}$$

onde $n_i(x)$ é o número de instanciações verdadeiras de F_i em um mundo x, $x_{\{i\}}$ é o estado (valores verdade) dos predicados em F_i e $\phi_i(x_{\{i\}}) = e^{w_i}$. Embora inicialmente definida como um modelo *log-linear*, a segunda igualdade da equação 2.4 mostra a definição de MLN por produto de funções potenciais, onde é mais conveniente representar domínios com mistura de restrições rígidas e suaves (onde fórmulas lidam com certeza, levando à probabilidade zero para alguns mundos).

O grafo de $M_{L,C}$ segue também a definição 2.4.1: há um arco entre dois nós de $M_{L,C}$, se e somente se os predicados instanciados correspondentes aparecerem em ao menos uma instanciação de uma fórmula em L. Portanto, os átomos em cada fórmula instanciada formam um subgrafo completamente conectado (*clique*) em $M_{L,C}$. As suposições a seguir garantem que o conjunto de mundos possíveis para (L,C) seja finito e que $M_{L,C}$ represente uma distribuição de probabilidade bem definida e única para estes mundos independente da interpretação e domínio:

Suposição 2.4.1. *Nomes únicos: Constantes diferentes referem-se a objetos diferentes (GENESERETH; NILSSON, 1987)*

Suposição 2.4.2. *Domínios fechados: Os únicos objetos do domínio são aqueles cuja representação é possível usando as constantes e funções símbolos em (L,C) (GENESERETH; NILSSON, 1987)*

Suposição 2.4.3. *Variáveis conhecidas: Para cada função em L, é conhecido o valor desta função aplicada a toda lista possível de argumentos e este valor é um elemento de C*

Estas considerações são bastante razoáveis para a maioria dos casos práticos e simplificam muito o uso das redes lógicas de Markov. Nos demais casos há situações em que estas considerações podem ser relaxadas desde que o domínio seja finito (DOMINGOS; LOWD, 2009).

Quanto à inferência, a MLN permite tanto inferência probabilística como lógica, porém estas são #P-completa[1] e NP-completa respectivamente, portanto intratáveis. Porém, como a MLN também permite codificar conhecimento incluindo independências contexto-específicas, a inferência pode em alguns casos ser mais eficiente do que em modelos gráficos ordinários. Na parte lógica, a semântica probabilística da MLN facilita a inferência aproximada, representando um potencial de ganho em eficiência. Quando a MLN está em forma clausal, o algoritmo mais utilizado é o *MaxWalkSat* (KAUTZ; SELMAN; JIANG, 1996), encontrando um valor verdade que maximiza a soma de pesos das cláusulas satisfeitas. Se houver restrições rígidas (cláusulas com peso infinito), o *MaxWalkSat* encontra regiões que as satisfazem e então um *amostrador de Gibbs* obtém estimativas de probabilidade para estas regiões.

Para computar probabilidades condicionais como $P(F_1 \mid F_2, L, C)$ pode-se utilizar o algoritmo Cadeia de Markov Monte-Carlo (do inglês *Markov Chain Monte-Carlo* ou MCMC) (GELFAND; SMITH, 1990) para aproximar a função rejeitando todos os movimentos para estados que não satisfazem F_2 e contando o número de amostras onde F_1 é satisfeito. Como isto pode ser bem lento ainda, é possível focar os casos em que F_2 é uma conjunção de literais instanciados (vide figura 2.6).

Quando há dependências determinísticas porém, onde o conjunto de mundos possíveis é separado em regiões isoladas umas das outras, a *amostragem de Gibbs* (GEMAN; GEMAN, 1984) falha, pois esta tende a ficar presa em uma das regiões. Para lidar com este tipo de problema utiliza-se então um MCMC *amostrador de fatias*, conhecido como *MC-SAT* por combinar o MCMC com *teste de satisfazibilidade* (POON; DOMINGOS, 2006).

Uma forma de se obter melhores modelos ou refinar redes lógicas de Markov, com menos trabalho do que especificar manualmente, é utilizar algoritmos de aprendizado. Neste trabalho utilizamos o algoritmo MC-SAT (algoritmo 2.1) para o aprendizado e inferência no modelo desenvolvido.

[1] Uma complexidade P-Completa refere-se a problemas de decisão do tipo: "há uma correspondência perfeita em um dado grafo bipartido?". Por sua vez problemas #P-Completos são provenientes da contagem de soluções: "quantas correspondências perfeitas há em um grafo bipartido?" (VALIANT, 1979).

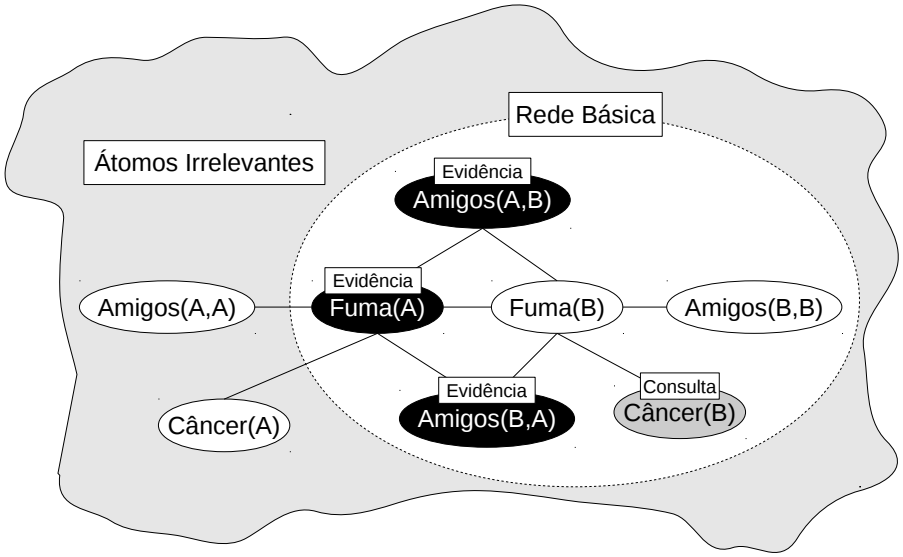

Figura 2.6: Rede instanciada para computar $P(Ca(B)|Fu(A),Am(A,B),Am(B,A))$.

Fonte: Autor "adaptado de" Domingos e Lowd (2009).

Função : MC-SAT(L,n)
entrada: L, um conjunto de cláusulas com peso (w_j,c_j)
n, número de amostras
saída : $\{x^{(1)},...,x^{(n)}\}$, conjunto de n amostras
$x^{(0)} \leftarrow$ satisfaz(cláusulas rígidas em L);
for $i \leftarrow 1$ **to** n **do**
  $M \leftarrow \emptyset$
  foreach $(w_k,c_k) \in L$ *satisfeitos por* $x^{(i-1)}$ **do**
  com probabilidade $1-e^{-w_k}$ adicione c_k em M;
  amostre $x^{(i)} \sim \mathscr{U}_{SAT(M)}$
end

Algoritmo 2.1: Inferência com MC-SAT (DOMINGOS; LOWD, 2009).

No algoritmo 2.1, $\mathscr{U}_{SAT(M)}$ é uma distribuição uniforme sobre o conjunto $SAT(M)$ de estados que satisfazem M. M é um subconjunto que contém as cláusulas instanciadas satisfeitas pela amostra do estado atual do mundo e que devem ser satisfeitas na próxima amostra de estados do mundo. Cada cláusula instanciada satisfeita é incluída em M com probabilidade $1-e^{-w}$, onde w é o peso da cláusula. Apesar do MC-SAT executar inferência aproximada, este algoritmo produz estimativas bem acuradas de probabilidades e seu desempenho é pouco sensível à parametrização do estágio de amostragem (DOMINGOS; LOWD, 2009).

Outro algoritmo de inferência importante para este trabalho é a propagação de crença

(*belief propagation*) (PEARL, 1982). Este algoritmo se beneficia do fato de uma MLN poder ser representada também com grafos de fator (*factor graphs*) (KSCHISCHANG; FREY; LOELIGER., 2001), que são grafos bipartidos com um nó para cada variável ou fator do modelo conectados por arestas não orientadas. Um fator é considerado $f_i(x) = exp(w_i g_i(x))$ para cada elemento $g_i(x)$. A propagação de crença não tem garantia de convergência ou de apresentar resultados acurados, porém na prática este algoritmo converge, é acurado e pode ser mais eficiente que outros métodos principalmente se os nós puderem ser combinados até obtermos uma árvore, onde a inferência seria exata. A propagação de crença consiste de procedimentos de passagem de mensagem da variável para um fator e vice-versa para calcular as probabilidades marginais. A mensagem da variável x para um fator f é dada por:

$$\mu_{x \to f}(x) = \prod_{h \in nb(x) \setminus \{f\}} \mu_{h \to x}(x), \tag{2.5}$$

onde $nb(x) \setminus \{f\}$ é o conjunto de valores vizinhos de x não presentes em f. A mensagem de um fator para uma variável é dada por:

$$\mu_{f \to x}(x) = \sum_{\sim \{x\}} \left(f(x) \prod_{y \in nb(f) \setminus \{x\}} \mu_{y \to f}(y) \right), \tag{2.6}$$

onde $nb(f) \setminus \{x\}$ são argumentos de f não presentes em x e a soma é sobre todos os argumentos exceto x. As mensagens das variáveis folhas são inicializadas em 1 e passadas das folhas até a raiz e de volta da raiz até as folhas. A distribuição marginal de cada variável x é dada por $\prod_{y \in nb(f)} \mu_{y \to f}(y)$. A evidência é especificada por $f(x) = 0$ quando o estado x for incompatível a mesma. O processo de passar a mensagem entre variáveis e fatores é executado até atingir um critério de convergência. Este algoritmo foi utilizado com a intenção de otimizar o tempo de processamento na inferência sobre o modelo. Discutiremos esta decisão na conclusão do trabalho (Cap.5).

Durante o processo de treinamento, um algoritmo de inferência é utilizado nos passos intermediários para definir a direção da correção do peso. Na próxima seção apresentamos o método de treinamento usado no modelo proposto para o aprendizado dos pesos da MLN.

2.4.2 Treinamento da MLN

O treinamento do nosso modelo utiliza a forma discriminativa, onde conhecemos *a priori* quais átomos serão evidências (conjunto X) e quais serão consultados (conjunto Y). Desejamos estimar a probabilidade condicional de Y dado X:

$$P(Y = y)(X = x) = \frac{1}{Z_x} exp\left(\sum_i w_i n_i(x,y)\right), \tag{2.7}$$

onde Z_x é normalização sobre o possíveis mundos condizentes com a evidência x e $n_i(x,y)$ é o número de instanciações verdadeiras da i-ésima fórmula nos dados. As cláusulas satisfeitas pela evidência ou que não possuem os átomos de consulta podem ser ignoradas.

O aprendizado discriminativo do peso w_i é obtido pela maximização da probabilidade logarítmica condicional da equação 2.7, onde a evidência tipicamente restringe a distribuição de probabilidade a um conjunto menor de estados possíveis. Portanto, o aprendizado é um problema de otimização, sendo assim, buscamos minimizar a probabilidade logarítmica condicional negativa da função:

$$f(w) = log(P(Y = y)(X = x)) = \sum_i w_i n_i(x,y) - log(Z_x). \tag{2.8}$$

Um método de primeira ordem para otimizar a equação 2.8 consiste em direcionar a busca por meio do gradiente descendente desta função:

$$\frac{\delta}{\delta w_i} f(w) = E_{w,y}[n_i(x,y)] - n_i(x,y), \tag{2.9}$$

onde y é o estado dos átomos não-evidência, x é o estado da evidência e $E_{w,y}$ é a expectativa sobre os átomos não-evidência Y. O algoritmo MC-SAT (algoritmo 2.1) é usado então na aproximação de $E_{w,y}$ definindo uma direção do gradiente g escalada por uma taxa de aprendizado η ($w_{(i+1)} = w_i + \eta g_i$) (DOMINGOS; LOWD, 2009). Este método é uma adaptação da *Divergência Contrastiva* de (HINTON, 2002) cuja etapa de aproximação de $E_{w,y}$ é executada com o algoritmo MCMC em vez de MC-SAT.

Os algoritmos de inferência e aprendizado como o MC-SAT e uma variedade de outros algoritmos relacionados à MLN estão inclusos no programa de código aberto chamado *Alchemy*, criado por pesquisadores da universidade de Washington para tornar os modelos e algoritmos discutidos em (DOMINGOS; LOWD, 2009) facilmente acessíveis para pesquisadores e alunos. Na próxima seção apresenta-se o *Alchemy* e suas funcionalidades.

2.4.3 Alchemy

O *Alchemy* (KOK et al., 2007) é um software livre, de código aberto e contém algoritmos para aprendizado e inferência em redes lógicas e híbridas de Markov incluindo aprendizado

de pesos discriminativos e generativos, inferência do estado mais provável (MAP/MPE) e inferência probabilística.

Funções, predicados e fórmulas de primeira ordem são especificados em arquivos .mln. A primeira aparição de um predicado é considerada sua declaração. Uma fórmula pode ser precedida por um peso ou terminada por um ponto. O ponto indica que a fórmula é rígida, ou seja, os mundos que a violarem devem ter probabilidade zero.

A sintaxe do *Alchemy* é similar à sintaxe da lógica de primeira ordem e permite também algumas instruções da linguagem *C/C++*. Alguns diferenciais em relação à lógica de primeira ordem seguem abaixo:

a) o ponto (.) no final de uma sentença, como citado anteriormente, indica que esta cláusula é rígida (*hard clause*), ou seja, é o mesmo que atribuir peso infinito para a mesma, tornando-a equivalente a uma cláusula comum de lógica de primeira ordem;

b) o sinal de soma (+) antecedendo uma variável, indica que durante o treinamento deve-se aprender um peso individual da sentença para cada constante instanciada. Como exemplo, supondo que em um jogo de dado cada face tem uma probabilidade diferente, utiliza-se a cláusula `Resultado(jogada,+face)`. Sem o sinal de +, seria aprendido um peso para a sentença (ou predicado) de modo que este fosse igualmente distribuído para as seis faces. Com o sinal de +, cada instância de face (Face1, Face2, ..., Face6) teria um peso diferente, supondo que nos dados de treinamento a quantidade de ocorrências de cada face fosse diferente;

c) o ponto de exclamação (!) antecedendo um predicado é a negação do mesmo (equivale ao símbolo lógico $\neg$). Logo após a variável, o ponto de exclamação indica que somente um valor é assumido pela variável a cada instanciação. Por exemplo, no mesmo jogo de dado onde cada jogada apresenta somente um e no máximo um resultado, podemos utilizar a cláusula `Resultado(jogada,face!)`. Desta forma, para cada jogada teremos como resultado somente uma face;

d) os símbolos lógicos são representados com os seguintes caracteres ASCII: v $(\vee)$, ^ $(\wedge)$, => $(\Rightarrow)$ e <=> $(\Leftrightarrow)$.

Seguindo a MLN da rede social da tabela 2.1, a tabela 2.2 apresenta um exemplo do arquivo de entrada.

Se os pesos não forem conhecidos ou especificados, estes podem ser obtidos através de um treinamento. Para tal utiliza-se um arquivo com evidências .db, como mostra a tabela 2.3,

Tabela 2.2: *fuma.mln*, um exemplo de MLN para o domínio de amigos e fumantes.

```
//Declaração dos predicados
Amigos(pessoa , pessoa)
Fuma(pessoa)
Câncer(pessoa)

// Se você fuma, você adquire câncer
1.5 Fuma(x) => Câncer(x)

// Amigos tem hábitos similares de fumar
0.8 Amigos(x, y) => (Fuma(x) <=> Fuma(y))
```

Fonte: Autor “adaptado de” Domingos e Lowd (2009)

cuja estrutura é a mesma para a inferência quando já se conhecem os pesos das fórmulas.

Tabela 2.3: *fuma-treino.db*, um exemplo de banco de dados para treinamento de *fuma.mln*.

```
Amigos(Anna, Bob)
Amigos(Bob, Anna)
Amigos(Anna, Edward)
Amigos(Edward, Anna)
Amigos(Bob, Chris)
Amigos(Chris, Bob)
Amigos(Chris, Daniel)
Amigos(Daniel, Chris)
Fuma(Anna)
Fuma(Bob)
Fuma(Edward)
Câncer(Anna)
Câncer(Edward)
```

Fonte: Autor “adaptado de” Domingos e Lowd (2009)

Para executar a inferência, utiliza-se o comando *infer*. A inferência padrão é do tipo MC-SAT. O resultado terá a seguinte aparência:

```
Fuma(Chris) 0.238926

Fuma(Daniel) 0.141286
```

A probabilidade de Anna, Bob e Edward fumarem não é computada, pois seus status de fumantes já são conhecidos na base de dados.

No caso do aprendizado dos pesos das fórmulas, o comando utilizado é o *learnwts* (*learn weights*).

A tabela 2.4 apresenta o resultado do aprendizado de pesos para a MLN da tabela 2.1 com as evidências da tabela 2.3

Tabela 2.4: *fuma-out.db*, um exemplo de aprendizado de pesos para uma MLN.

```
//Declaração dos predicados

Câncer(pessoa)
Amigos(pessoa,pessoa)
Fuma(pessoa)

//Declaração das funções

// 1.51903 Fuma(x) => Câncer(x)
1.51903 !Fuma(a1) ∨ Câncer(a1)

// 0.994841 Amigos(x,y) => (Fuma(x) <=> Fuma(y))
0.49742 !Amigos(a1,a2) ∨ Fuma(a1) ∨ !Fuma(a2)
0.49742 !Amigos(a1,a2) ∨ Fuma(a2) ∨ !Fuma(a1)

// 0 Amigos(a1,a2)
0 Amigos(a1,a2)

// 0.86298 Fuma(a1)
0.86298 Fuma(a1)

// -1.0495 Câncer(a1)
-1.0495 Câncer(a1)
```

Fonte: Autor “adaptado de” Domingos e Lowd (2009)

O *Alchemy* proporciona uma economia de tempo em pesquisa e desenvolvimento de sistemas lógico-probabilísticos devido a implementação de vários algoritmos de inferência e aprendizado conhecidos e, por ter código aberto, ainda permite criar extensões ao software, o que garante permanente evolução da ferramenta. Toda documentação e arquivos relacionados ao *Alchemy* encontram-se disponíveis em *http://alchemy.cs.washington.edu*.

Na próxima seção apresentamos alguns trabalhos de interpretação de cenas de tráfego rodoviário, incluindo os que utilizam técnicas de lógica probabilística.

2.5 Interpretação de Cenas de Tráfego Rodoviário

Com base em estatísticas de acidentes automotivos, vários fabricantes de automóveis passaram a investir mais em soluções de segurança ativa, ou seja, sistemas que auxiliam na prevenção de acidente. Estatísticas de acidentes automotivos na Alemanha do ano de 2007 [1] indicam que a mudança de faixa de trânsito foi a causa de 19,2% dos acidentes (DAIMLER, 2009a), outro dado do ano de 2004 para a Europa mostra que este índice passa de 35% (HOLZMANN, 2008), o que justifica o investimento em pesquisa e desenvolvimento em sistemas assistentes de faixa de trânsito (do inglês *Lane Assistant System* ou LAS). A função de um LAS é informar ao motorista se uma possível mudança involuntária de faixa está ocorrendo e em alguns casos até corrigir a trajetória do veículo para retornar à faixa de origem (AMSEL et al., 2009; DAIMLER, 2009b).

Alguns assistentes de faixa (LAS), inclusive os já comercializados em veículos de série, apresentam em geral uma implementação com visão monocular e processamento quantitativo para a classificação das linhas divisórias da faixa (BERTOZZI et al., 2002; AMSEL et al., 2009). Porém, a visão monocular limita a percepção de profundidade impondo restrições na modelagem do ambiente, apesar de alguns estudos tratarem deste tema (WEDEL et al., 2006).

Nos últimos anos no entanto, com o avanço tecnológico dos computadores e câmeras, muito se desenvolveu no reconhecimento de faixas de trânsito em ambiente 3D (BERTOZZI et al., 2002; NEDEVSCHI et al., 2008). O uso de câmera estéreo proporciona a informação de profundidade do ambiente, porém o processamento em tempo real no domínio do tráfego ainda é um desafio. Muitos trabalhos têm abordado a otimização de processamento da visão estéreo tanto no reconhecimento de faixa de trânsito como no reconhecimento de obstáculos (STEINGRUBE; GEHRIG; FRANKE, 2009; KLETTE, 2008), inclusive em parceria com os fabricantes de automóveis (FRANKE et al., 2008; ASAI et al., 2008).

Iniciativas como os laboratórios de navegação (*NAVLAB*)(Thorpe ; KANADE, 1985) também receberam maior atenção nas universidades (principalmente as norte-americanas) desde a década de 90. Nestes laboratórios tem sido desenvolvidos sistemas de navegação (terrestre, aéreo e marítimo) com enfoque na condução não tripulada do veículo. Com o apoio da agência americana de projetos avançados de pesquisa em defesa (*DARPA*), as universidades e empresas de tecnologia dos Estados Unidos desenvolvem continuamente diversas abordagens para guiar os veículos de forma não tripulada (THRUN et al., 2006; MONTEMERLO et al., 2008). Em geral, há muito desenvolvimento e pesquisa em detecção de faixas trânsito e serviços de assistência ao motorista. McCall e Trivedi (2006) fazem uma revisão e análise do estado da arte das técnicas de

[1] Não é do conhecimento do autor até então, estudo similar no Brasil

detecção de faixas de trânsito, estado do veículo (intenção do motorista) e estimativa de posição do veículo. McCall e Trivedi (2006) apresentam também, um sistema de assistência ao motorista denominado *VioLET* (*video-based lane estimation and tracking*) no qual as tarefas de extração e rastreamento das linhas de trânsito são auxiliadas por meio de filtros orientáveis (*steerable filter*) e modelos matemáticos de estrada e do veículo. Porém, em relação às cenas de tráfego, estes trabalhos apresentam uma interpretação quantitativa (por meio da fusão sensória de câmeras de vídeo, sensores a laser, GPS e sonares), não há uma interpretação semântica da localização do veículo.

Uma exceção a estas abordagens é o "veículo terrestre autônomo em uma rede neural", o ALVINN (do inglês *Autonomous Land Vehicle in a Neural Network*) (JOCHEM; POMERLEAU; THORPE, 1993), onde uma rede neural incorpora diretamente a detecção de características do ambiente no algoritmo de controle do veículo sem haver uma retroalimentação do rastreamento. Apesar da robustez das redes neurais, no caso do ambiente de tráfego, estas não fornecem o conhecimento de problemas como oclusão parcial, sombras e iluminação (BERTOZZI et al., 2002).

No presente trabalho, apresentamos uma interpretação de alto nível (ou semântica) sobre as cenas de tráfego. Alguns trabalhos de interpretação qualitativa de cenas por meio da avaliação de sequência de imagens de tráfego datam da década de 70 (NAGEL, 1977). Alguns trabalhos utilizam o fluxo ótico para raciocinar sobre o movimento (BOUTHEMY; FRANCOIS, 1993; NAGEL, 2000; GERBER; NAGEL; SCHREIBER, 2002). Num contexto mais geral sobre interpretação de cenas dinâmicas, Chella, Frixione e Gaglio (2000) propõem uma extensão dos espaços conceituais (GäRDENFORS, 2000) de forma que a tarefa de interpretação seja feita em três níveis. No primeiro nível, o *sub-conceitual*, ocorre o processamento dos dados dos sensores de modo a gerar uma descrição da cena dinâmica observada. No segundo nível, o *conceitual*, geram-se descrições de alto-nível da cena de acordo com os dados do nível anterior e utilizando o cálculo de situações (REITER, 2001). Com as descrições de alto-nível, o terceiro nível (*linguístico*) efetua a interpretação das cenas. Em geral, é necessário um processamento visual mais preciso quando a modelagem do problema é puramente lógica. Buscamos tratar possíveis falhas de um processamento visual mais simples com a lógica probabilística. Desta forma podemos nos beneficiar de uma implementação de baixo custo computacional, proporcionando tempo real de processamento.

Um campo de pesquisa mais recente, tem acompanhado os trabalhos de reconhecimento de faixa de trânsito e obstáculos e o de interpretação das cenas de tráfego. O objetivo destes estudos é prover um conhecimento de alto nível para interpretar probabilisticamente os dados percebidos pelos sensores.

Dentre diversas abordagens para interpretação de cenas de tráfego, podemos citar a teoria probabilística GO (PGO) (PARKER et al., 2007), que baseia-se em métodos de otimização, como programação linear, para definir a probabilidade de localização de um veículo em movimento em um dado instante de tempo. Embora apresente uma abordagem dinâmica e probabilística na predição de posição de veículos, o observador é estático e está fora do contexto analisado, ou seja, há uma visão global do ambiente e não egocêntrica. Uma aplicação disso seria o rastreamento de objetos a partir de um radar.

Por sua vez, a incerteza em lógica de descrição (SANTOS et al., 2010) usa uma abordagem probabilística da lógica de descrição para tratar incertezas e representar o domínio espacial a fim de responder algumas perguntas como: "qual faixa da estrada é a ego-faixa?" e "qual é o sentido permitido de cada faixa?". Este trabalho parece ser pioneiro na modelagem dos sensores em uma linguagem lógica. Outra característica importante é que se adota a visão egocêntrica do veículo, item necessário aos sistemas de assistência ao motorista, leva em consideração também outros sensores além da câmera, como o GPS e um mapa digital.

A informação contextual do ambiente de tráfego é considerada na construção das regras para a estrada e na modelagem dos sensores. Utiliza-se como conceitos a faixa de trânsito com primitivas de sentido de direção, a linha divisória podendo ser sólida ou tracejada, o veículo estando em uma estrada de mão única ou mão dupla e também, o que foi detectado pelos sensores. A percepção do sentido da via é proveniente do GPS, por sua vez as linhas divisórias são percebidas pela câmera e para indicar se a estrada é de mão dupla ou não, utiliza-se o mapa digital.

Apesar de mostrar com sucesso que as consultas efetuadas eram consequências da formalização assumida para o domínio, um grande desafio enfrentado foi evitar a recursividade na rede Bayesiana que associada à expressividade limitada da lógica de descrição (por ser um subconjunto da lógica de primeira ordem) dificulta a expansão do modelo.

Por sua vez, Hensel et al. (2010) utilizam a lógica de Markov para representar e inferir a relação entre pares de objetos em uma cena de tráfego. Por ser uma lógica probabilística de primeira ordem, apresenta um expressividade melhor que a da lógica de descrição. Outra vantagem é o uso de redes de Markov para a inferência, o que elimina a restrição de aciclicidade facilitando a modelagem do domínio. O trabalho de Hensel et al. (2010) leva em consideração o contexto da cena, como sugerido por Yilmaz, Javed e Shah (2006) porém, apesar dos dados serem egocêntricos, o raciocínio executado é global (vendo a estrada do alto) e a inferência é em relação aos obstáculos, não havendo inferência sobre a faixa de trânsito.

Com a aplicação de técnicas de lógica probabilística podemos nos beneficiar de uma

implementação mais simples e de baixo custo do sistema de visão, de modo que, mesmo com detecções equivocadas no sistema visão, ainda haverá a possibilidade de um evento esperado ocorrer durante a etapa inferência. Por exemplo, se o sistema de visão detectar uma linha branca contínua à esquerda do veículo como uma linha branca tracejada, ainda será possível inferir qual é a probabilidade do veículo estar na faixa esquerda, o que seria impossível somente com inferência lógica.

A próxima seção é dedicada aos métodos utilizados no desenvolvimento e implementação deste trabalho.

3 IMPLEMENTAÇÃO DA PROPOSTA

Os métodos utilizados para implementar este trabalho foram escolhidos conforme os seguintes critérios:

a) facilidade de instalação da câmera e captura de vídeo, possibilitando gravar os vídeos em diversos momentos com posição e ângulos similares da câmera;

b) baixa resolução de vídeo a fim de otimizar o custo computacional do sistema de visão;

c) algoritmo de visão bem difundido e de fácil acesso, proporcionando um ponto de partida para trabalhos futuros;

d) linguagem de lógica probabilística que apresentasse boa expressividade e semântica simples;

e) algoritmos de aprendizado de máquina e inferência que fossem eficientes e atendessem à linguagem escolhida.

Os equipamentos e programas utilizados neste trabalho foram os seguintes:

a) computador com processador AMD Turion X2 de 64 bits e velocidade de 2200MHz, 4GB de memória RAM DDR2 e sistema operacional Linux Ubuntu versão 10.04 LTS;

b) câmera de vídeo Microsoft, tipo *webcam*, modelo VX-2000;

c) software de engenharia Matlab, versão R2010b (MATHWORKS, 2010). Utilizado o *toolbox* de visão para detecção e classificação das linhas divisórias;

d) software de código aberto *Alchemy* (KOK et al., 2007), para a modelagem do ambiente de tráfego, treinamento e inferência em MLN.

Retomando o diagrama proposto na seção 1, figura 1.1 (p.14), as camadas implementadas foram as seguintes:

a) *percepção*: captura das imagens à frente do veículo por meio da webcam;

b) *segmentação*: extração das bordas verticais de cada quadro de vídeo obtido na camada anterior. Esta operação é a primeira parte do algoritmo "*videoldws*" (*lane departure warning system*) do Matlab;

c) *classificação e rastreamento*: detecção das linhas divisórias por meio da transformada de Hough, classificação quanto ao tipo e cor e rastreamento. Estas operações são efetuadas na segunda parte do mesmo algoritmo da camada anterior (*videoldws*);

d) *inferência em MLN*: principal contribuição deste trabalho. Compreende a modelagem do domínio de tráfego rodoviário, o treinamento da rede lógica de Markov e a inferência das condições de tráfego. As condições de tráfego consultadas foram a posição do veículo na estrada, a mudança de faixa e se esta mudança é permitida;

Desta forma, descrevemos nas seções seguintes cada método ou ferramenta utilizados: na seção 3.1 detalhamos a camada de percepção. Na seção 3.2 descrevemos o algoritmo de visão do Matlab utilizado nas camadas de extração, classificação e rastreamento. Na seção 3.3 modelamos o ambiente de tráfego rodoviário em Redes Lógica de Markov utilizando a sintaxe do *Alchemy*.

3.1 Módulo de Percepção

Neste trabalho assumimos um sistema de segurança automotiva que recebe e processa as informações do ambiente em tempo real. Com imagens de baixa resolução de domínios finitos e reduzidos, podemos atingir este objetivo. O arcabouço introduzido na figura 1.1 (p.14) foi projetado com base no trabalho de (AMSEL et al., 2009), onde os autores apresentam conceitos de sistemas de assistência ao motorista que são comuns entre algumas montadoras europeias.

A camada de percepção da figura 1.1 foi implementada com uma *webcam* Microsoft modelo VX2000, configurada para captar imagens no formato AVI não comprimido (exigência da versão R2010b do Matlab para Linux), no espaço de cor RGB de 24 bits, com resolução de 320x240 pixels e taxa de quadro (*frame rate*) de 30Hz.

Esta câmera foi instalada logo abaixo do retrovisor interno do veículo (fig.3.1), de modo que as linhas divisórias de cada lado do veículo ficassem aproximadamente à mesma distância do centro do quadro no momento em que o veículo estivesse no centro da faixa.

O vídeo capturado é armazenado para uso posterior no algoritmo de visão. É possível obter os dados da câmera em tempo real para o Matlab. Porém, isto está fora do escopo deste trabalho. As camadas de segmentação, classificação e rastreamento (camadas 2 e 3 da figura 1.1 p.14) fazem parte do algoritmo de visão implementado em Matlab, que será apresentado na seção a seguir.

Figura 3.1: Detalhe de instalação da câmera no veículo.

Fonte: Autor.

3.2 Algoritmo de Visão para Detecção das Faixas de Trânsito

Os vídeos capturados neste trabalho tem uma taxa de atualização de 30Hz. Assumindo a metodologia meio-a-meio para aprendizado e teste (FITZPATRICK; SONKA, 2000), os quadros ímpares foram usados para o aprendizado das regras e os quadros pares para inferência. Então, na prática, temos uma taxa de atualização de 15Hz. A princípio pode parecer tendencioso utilizar quadros próximos para teste e treinamento, mas as características do domínio não mudam significativamente se as condições climáticas forem as mesmas durante o trajeto. Até as mudanças do tipo de faixa ocorrem após vários quadros (ou segundos) durante uma manobra. Uma mudança mais significativa ocorre com o aumento de velocidade sobre as linhas tracejadas, onde a imagem tende a borrar alongando as linhas brancas. Este fenômeno ocorre devido às características da câmera como resolução e sensor CMOS.

Foi utilizado o algoritmo de demonstração *videoldws* do Matlab (MATHWORKS, 2010), integrante do *toolbox* de visão, o qual inclui os seguintes passos:

1. Converter a metade inferior de uma imagem colorida em intensidade, atribuindo intensidade nula para a metade superior da imagem;

   A metade inferior da imagem é a região de interesse (*region of interest* ou ROI), pois na metade superior em geral, se encontram o céu e algumas construções que podem ser ignoradas. O resultado deste passo está representado na figura 3.2.

2. Aplicar um filtro bidimensional (2D) com máscara vertical;

   O filtro bidimensional (2D) utilizado é um detector de bordas de Sobel com máscara ver-

Figura 3.2: Conversão de RGB para intensidade.

Fonte: Autor.

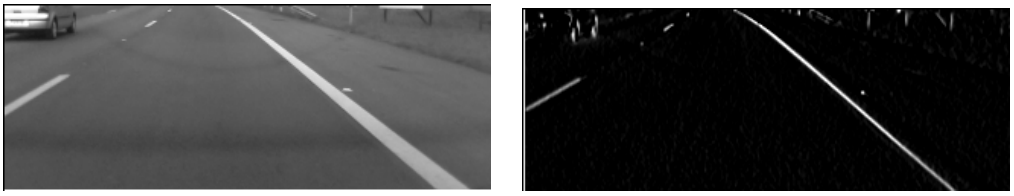

Figura 3.3: Aplicação do filtro de Sobel.

Fonte: Autor.

tical $S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ a fim de realçar as componentes verticais da imagem. Não foi utilizado o algoritmo de Canny pois este gera bordas de espessura mínima, o que prejudica a seleção de máximos locais da transformada de Hough nas etapas seguintes. Na figura 3.3 apresentamos o resultado deste passo;

3. Binarizar a imagem resultante;

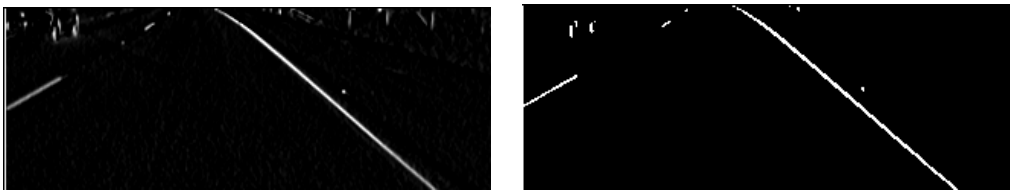

Figura 3.4: Binarização da imagem.

Fonte: Autor.

Após a aplicação do filtro 2D, a imagem ainda apresenta valores intermediários de intensidade (entre 0 e 1) que devem ser eliminados para a posterior transformada de Hough.

Este passo é representado na figura 3.4.

4. Aplicar a transformada de Hough;

Figura 3.5: Transformada de Hough.

Fonte: Autor.

Neste passo é gerada a matriz de Hough, onde os valores máximos da matriz representam as hipóteses de linhas divisórias (fig.3.5).

5. Zerar a matriz de Hough na região definida por uma janela baseada em ρ e θ (introduzidos na Seção 2.2.3, p.22) e localizar dois máximos locais. Os máximos locais, neste domínio, fornecem as coordenadas das linhas divisórias;

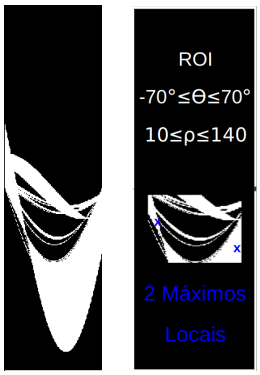

Figura 3.6: Filtragem da matriz de Hough.

Fonte: Autor.

Os objetos de interesse (linhas divisórias) se encontram numa região onde $-70° \leq \theta \leq 70°$ e $10 \leq \rho \leq 140$. Estes valores foram definidos empiricamente medindo o ângulo (ρ) e a distância (θ) das linhas em alguns quadros com auxílio de uma ferramenta de edição de imagens (Gimp 2.6). Cada máximo local fornece as coordenadas de uma linha no espaço de Hough (ρ, θ). O resultado é apresentado na figura 3.6.

6. Rastrear as linhas divisórias que compõem uma faixa de trânsito;

   O rastreamento consiste também de alguns passos:

   a) armazenar os máximos locais obtidos no passo anterior em um repositório de rastreamento (no máximo 20 linhas);

   b) calcular a distância entre as linhas do quadro atual e as linhas do repositório, atualizando o apontador da lista;

   c) se houver correspondência entre as linhas atuais e as linhas do repositório (baseada num limiar), substituir as linhas do repositório pelas linhas atuais na posição da lista em que apresentaram a menor distância;

   d) se não houver correspondência, indicar uma posição não utilizada da lista;

   e) converter os valores apontados na lista, de coordenada polar (ρ, θ) para coordenada cartesiana (x,y) utilizando a equação 2.1 (p.22);

   f) inserir na imagem original as linhas com as coordenadas obtidas.

   O resultado do rastreamento é apresentado na figura 3.7.

Figura 3.7: Rastreamento das linhas divisórias.

Fonte: Autor.

7. Detectar se uma das linhas divisórias está sob o veículo;

   Sempre que a extremidade de uma linha divisória passa pela parte inferior da imagem, considera-se que esta linha está sob o veículo, vide figura 3.7;

8. Converter o espaço de cor da imagem de RGB para YCbCr para garantir uniformidade na detecção (conforme discutido na Seção 2.2.2, p.19). Classificar o tipo e cor de cada linha divisória, analisando a porcentagem de pixels brancos e amarelos numa região de 10 pixels em torno de cada linha detectada. Outros espaços de cores uniformes (Lab, Luv

ou HSV) poderiam ser usados, mas como este processo também faz parte do algoritmo de demonstração, este ponto não foi avaliado. Vemos na figura 3.8 o resultado deste passo;

Figura 3.8: Mudança do espaço de cor de RGB para YCbCr.

Fonte: Autor

9. Mostrar texto com tipo e cor junto às linhas e sinalizar a mudança de faixa em caso positivo. O resultado deste último passo pode ser observado na figura 3.9;

Figura 3.9: Exibição das mensagens na tela.

Fonte: Autor

Uma descrição mais detalhada deste algoritmo é apresentado em (MATHWORKS, 2010). Na figura 3.10 apresentam-se alguns quadros obtidos com este algoritmo. Podemos observar também nestes quadros alguns dos desafios encontrados ao lidar com cenas reais como: limites do acostamento (fig.3.10a), iluminação do ambiente e de outros veículos (fig.3.10b) e oclusão de outros veículos (fig.3.10c).

Este algoritmo de visão faz parte do módulo 2 e 3 da figura 1.1 (p.14) e foi utilizado para gerar as evidências que servirão de entrada para o sistema de regras sobre o ambiente de

a) 

b) 

c) 

Figura 3.10: Alguns quadros obtidos com o algoritmo de visão.

Fonte: Autor.

tráfego rodoviário. Estas regras foram modeladas em redes lógicas de Markov e serão descritas em detalhes na seção a seguir, principal contribuição deste trabalho.

3.3 Modelo Lógico do Ambiente de Tráfego Rodoviário

Uma das vantagens da MLN é a possibilidade de codificar informação contextual do ambiente de modo a proporcionar uma modelagem de conhecimentos de alto nível, tornando a inferência mais eficiente. A informação contextual neste trabalho é a informação relevante para entendermos o ambiente e está relacionada às regras de trânsito e aos atributos dos objetos deste ambiente (por exemplo, características das faixas). Um sistema de assistência ao motorista (*lane assistance system* ou LAS) tem como característica que o agente da percepção está inserido no contexto, isto é, o ponto de vista do agente é o mesmo do motorista e ambos se movimentam no ambiente analisado. O domínio considerado neste trabalho é o ambiente de tráfego real de uma rodovia. Embora os obstáculos (outros veículos ou imperfeições da estrada) não tenham sido modelados, os vídeos analisados apresentam os diversos tipos de obstáculos e imperfeições de uma rodovia (por exemplo, oclusão de outros veículos, ondulações na pista, curvas). Seguindo estas premissas, respondemos à duas questões chave sobre condução rodoviária a fim modelar o ambiente de tráfego:

a) “o que é importante saber sobre o ambiente para manter o veículo na faixa de trânsito?”

   Considerando somente a estrada, sem obstáculos, visualmente seguimos as linhas divisórias da faixa de trânsito em ambos os lados do veículo, controlando-o para permanecer no centro da faixa.

b) “que tipo de identificação nos dá senso de direção ou localização na estrada?”

   A cor e a forma das linhas divisórias nos dá senso de direção e posição na estrada. Temos este conhecimento à partir das regras de trânsito, que nos permite raciocinar sobre as linhas divisórias que vemos durante a condução de um veículo.

Neste capítulo são apresentadas as regras desenvolvidas neste trabalho para gerar a MLN que será utilizada na interpretação da funcionalidade das faixas de trânsito. As regras que regem o nosso domínio são descritas na tabela 3.1. Porém, antes de definir as regras foi necessário especificar variáveis para identificar algumas características do ambiente:

a) tipo de linha divisória: *type*={*YContinuous, WContinuous, YDashed, WDashed, Merge*}. Onde *W* significa a cor branca (*white*) e *Y* significa a cor amarela (*yellow*). *Continuous* refere-se à linha contínua, *Dashed* é referente à linha tracejada, *Merge* é a linha descontínua que representa o acesso de entrada ou saída de uma rodovia e possui intervalos mais curtos que os da linha tracejada;

b) mão da rodovia: *way*={*One, Two*}, mão única (*One*) ou mão dupla (*Two*);

c) posição do veículo e das linhas divisórias: *lanepos*={*Left, Centre, Right*}. A posição do veículo é relativa à rodovia e ao ponto de vista do motorista, considerando a mão correta de trânsito (por exemplo, "o veículo está na faixa esquerda da rodovia"). Centro é considerado toda posição que não seja a extrema faixa direita ou esquerda. Por sua vez, a posição da linha divisória é em relação ao veículo (por exemplo, "a linha tracejada está à direita do veículo");

d) condição da linha divisória: *status*={*Ok,Under*}. *Under* indica que a linha está sob o veículo (durante uma mudança de faixa) e *Ok* indica que a linha está ao lado do veículo (não está sob).

As variáveis foram definidas de forma a abranger todas as características desejadas no modelo mantendo um número reduzido de símbolos. Por este motivo não tratamos as linhas axiais ou centrais (linha divisória amarela dupla) e suas variantes (com permissão ou não para ultrapassagem). O raciocínio sobre as linhas amarelas contínuas e tracejadas no modelo implementado, nos permite inferir as mesmas condições de trânsito das linhas axiais ou centrais. Por exemplo, em uma linha central com permissão de ultrapassagem para o egoveículo (veículo agente da percepção), a linha mais próxima ao egoveículo será do tipo amarelo tracejado, caso contrário, esta linha será do tipo amarelo contínuo. Portanto, são os dois tipos de linhas já considerados na modelagem do problema.

Após a definição das variáveis acima, especificamos os seguintes predicados:

a) *dividerL/R(type,status,time)*: identificação da linha divisória com o seu tipo (*Dashed, Continuous*), sua condição (*ok, under*), o número do quadro atual (time) e sua posição relativa ao veículo, por exemplo, esquerda (*Left*: L) ou direita (*Right*: R);

a) *carRelPos(lanepos,time)*: posição do veículo relativa à mão em correta de trânsito (*lanepos*);

b) *crossingLeft(time)*: veículo cruzando à esquerda (mudando para a faixa à esquerda);

c) *crossingRight(time)*: veículo cruzando à direita (mudando para a faixa à direita);

d) *emergencyLane(time)*: veículo está na faixa de emergência (acostamento);

e) *prohibitedManoeuvre(time)*: veículo executando uma manobra proibida (por exemplo, cruzando uma linha amarela contínua);

f) *wrongWay(time)*: veículo na mão errada da rodovia (contramão);

g) *roadWay(way,time)*: rodovia de mão única ou mão dupla.

Podemos notar que todos os predicados representam fatos sobre o egoveículo, portanto a variável para o agente foi omitida. A variável *time* representa o número do quadro onde o predicado é instanciado. O uso do termo *time* se deve ao fato de se tratar de uma informação temporal. Por exemplo, na taxa de quadro utilizada de 30Hz o intervalo de tempo é de 33,3ms entre cada quadro, portanto no quadro 100 teríamos as instâncias dos predicados no instante de tempo de 3,33s.

Quando o veículo está na contramão, não necessariamente estará executando uma manobra proibida, por este motivo definimos os predicados: *prohibitedManoeuvre/1* e *wrongWay/1*. Por exemplo, se o veículo estiver na contramão mas a linha divisória for amarela tracejada, a manobra é permitida durante uma ultrapassagem. Ainda em relação à posição do veículo na estrada, a posição esquerda é definida como a extrema esquerda da mão correta de trânsito. Portanto, se o veículo entrar na contramão, será considerado que ele esteja na faixa esquerda. Raciocínio análogo é feito sobre os acostamentos (direito e esquerdo).

As instâncias de *dividerL/3* e *dividerR/3* são provenientes do algoritmo de visão e serão utilizadas como evidências no processo de inferência. Este predicado é fornecido sempre em pares (linha direita e linha esquerda) para cada quadro (*time*).

Definidos os predicados, as fómulas em MLN foram construídas de modo a codificar as regras de trânsito para cirulação à direita e o conhecimento do ambiente de tráfego (tab.3.1). Podemos perceber que inicialmente as fórmulas são somente sentenças em lógica de primeira ordem (sem os pesos).

Na tabela 3.1, apresentamos a descrição de cada regra do nosso domínio em linguagem natural seguida de sua formalização lógica. Novamente, algumas considerações foram

necessárias visando conter a complexidade do modelo, pois cada variável ou predicado criado aumenta exponencialmente o número de nós (e consequentemente de *cliques*) da rede de Markov.

Tabela 3.1: Formalização lógica para o LAS.

1. Se o veículo está sobre uma linha amarela contínua ou se houver uma linha do mesmo atributo à direita, o veículo está executando uma manobra proibida na contramão. A rodovia é de mão dupla e o veículo está na faixa esquerda.
$dividerL(YContinuous, Under, t) \vee dividerR(YContinuous, Ok, t) \Rightarrow prohibitedManoeuvre(t) \wedge wrongWay(t) \wedge roadWay(Two, t) \wedge carRelPos(Left, t) \wedge \neg carRelPos(Centre, t) \wedge \neg carRelPos(Right, t) \wedge \neg emergencyLane(t)$

2. Se não há evidência de mão dupla, considere a rodovia de mão única.
$\neg dividerL(YContinuous, s, t) \vee \neg dividerL(YDashed, s, t) \vee \neg dividerR(YContinuous, s, t) \vee \neg dividerR(YDashed, s, t) \Rightarrow roadWay(One, t) \wedge \neg roadWay(Two, t)$

3. Se há uma linha amarela contínua ou tracejada em qualquer posição, a rodovia é de mão dupla.
$dividerL(YContinuous, s, t) \vee dividerL(YDashed, s, t) \vee dividerR(YContinuous, s, t) \vee dividerR(YDashed, s, t) \Rightarrow roadWay(Two, t) \wedge \neg roadWay(One, t) \wedge carRelPos(Left, t) \wedge \neg carRelPos(Centre, t) \wedge \neg carRelPos(Right, t) \wedge \neg emergencyLane(t)$

4. Se a linha esquerda é branca contínua e a linha direita é branca tracejada, a rodovia é de mão única e o veículo está na faixa esquerda.
$dividerL(WContinuous, Ok, t) \wedge (dividerR(WDashed, Ok, t) \Rightarrow roadWay(One, t) \wedge \neg roadWay(Two, t) \wedge carRelPos(Left, t) \wedge \neg carRelPos(Centre, t) \wedge \neg carRelPos(Right, t) \wedge \neg prohibitedManoeuvre(t) \wedge \neg wrongWay(t)$

5. Se as linhas direita e esquerda forem brancas tracejadas, o veículo está na faixa central.
$dividerL(WDashed, Ok, t) \wedge dividerR(WDashed, Ok, t) \Rightarrow carRelPos(Centre, t) \wedge \neg carRelPos(Left, t) \wedge \neg carRelPos(Right, t) \wedge \neg wrongWay(t) \wedge \neg crossingLeft(t) \wedge \neg crossingRight(t) \wedge \neg prohibitedManoeuvre(t) \wedge \neg emergencyLane(t)$

6. Se a linha esquerda não é branca tracejada e a linha direita é branca tracejada, o veículo está na faixa esquerda.
$\neg dividerL(WDashed, Ok, t) \wedge dividerR(WDashed, Ok, t) \Rightarrow carRelPos(Left, t) \wedge \neg carRelPos(Centre, t) \wedge \neg carRelPos(Right, t) \wedge \neg crossingLeft(t) \wedge \neg crossingRight(t) \wedge \neg prohibitedManoeuvre(t) \wedge \neg emergencyLane(t)$

Continua

Conclusão

7. Se a linha direita é amarela tracejada, o veículo está na contra-mão e na faixa esquerda.
$dividerR(YDashed, Ok, t) \wedge dividerL(ty, Ok, t) \Rightarrow wrongWay(t) \wedge carRelPos(Left, t) \wedge \neg carRelPos(Centre, t) \wedge \neg carRelPos(Right, t) \wedge \neg crossingLeft(t) \wedge \neg crossingRight(t) \wedge \neg prohibitedManoeuvre(t) \wedge \neg emergencyLane(t)$

8. Se a linha esquerda é branca tracejada e a linha direita é branca contínua ou de acesso, o veículo está na faixa direita.
$(dividerR(WContinuous, Ok, t) \vee dividerR(Merge, Ok, t)) \wedge dividerL(WDashed, Ok, t) \Rightarrow carRelPos(Right, t) \wedge \neg carRelPos(Centre, t) \wedge \neg carRelPos(Left, t) \wedge \neg wrongWay(t) \wedge \neg crossingLeft(t) \wedge \neg crossingRight(t) \wedge \neg emergencyLane(t)$

9. Se uma linha é branca contínua e a outra não é branca tracejada, o veículo possivelmente está na faixa de emergência.
$dividerL(WContinuous, Ok, t) \wedge \neg dividerR(WDashed, s, t) \Rightarrow emergencyLane(t) \wedge carRelPos(Right, t) \wedge \neg carRelPos(Left, t) \wedge \neg carRelPos(Centre, t) \wedge \neg prohibitedManoeuvre(t) \wedge \neg wrongWay(t)$

$dividerR(WContinuous, Ok, t) \wedge \neg dividerL(WDashed, s, t) \Rightarrow emergencyLane(t) \wedge carRelPos(Left, t) \wedge \neg carRelPos(Right, t) \wedge \neg carRelPos(Centre, t) \wedge \neg prohibitedManoeuvre(t) \wedge \neg wrongWay(t)$

10. Se o veículo está sobre a linha direita ou sobre a linha esquerda com a linha direita *Ok*, o veículo está cruzando à direita. É análogo para o cruzamento à esquerda.
$dividerR(ty1, Under, t) \vee (dividerL(ty2, Under, t) \wedge dividerR(ty2, Ok, t)) \Rightarrow crossingRight(t) \wedge \neg crossingLeft(t)$

$dividerL(ty1, Under, t) \vee (dividerR(ty2, Under, t) \wedge dividerL(ty2, Ok, t)) \Rightarrow crossingLeft(t) \wedge \neg crossingRight(t)$

Fonte: Autor.

Os predicados sem constantes específicas (s, ty, t) serão instanciados para todas as constantes de cada variável. A negação de alguns predicados na implicação de cada fórmula (conhecida também como axioma de quadro) foi necessária para especificar características inalteradas do domínio nas condições dadas. Axiomas de quadro são axiomas usados para especificar os *não-efeitos* de uma ação (MORGENSTERN; MCILRAITH, 2011). Por exemplo, na regra 5 da tabela 3.1, que especifica as condições para o veículo estar na faixa central, os demais predicados diferentes de *carRelPos(Centre,t)* e que estão negados são os axiomas de quadro usados para indicar, por exemplo, que se o veículo estiver na faixa central, este não estará na faixa esquerda ou direita, nem cruzando para qualquer lado da via. Apesar de aumentar o custo computacional, esta ação foi necessária para otimizar a resposta à consulta. Os axiomas de quadro indicam ao algoritmo de inferência que esta condição tem probabilidade zero de ocorrência, reduzindo o espalhamento da distribuição de probabilidade sobre os predicados que não possuem relação causal com as evidências atuais. Esta é uma forma simplificada de tratar o problema da persistência (*frame problem*) num domínio finito de tamanho reduzido. Em qualquer domínio há condições para um grande número destes axiomas de quadro. Sendo n o número de ações (ou evidências) e m o número de fluentes (ou predicados de consulta), teríamos aproximadamente nm axiomas (MORGENSTERN; MCILRAITH, 2011).

Por outro lado, a fim de evitar o uso de axiomas de quadro em todas as sentenças, aplicamos também uma estratégia conhecida como *sleeping dog strategy* (MCDERMOTT, 1987), que consiste em especificar somente o que é relevante para a consulta considerando o restante inalterado. Esta estratégia entra em conflito com a primeira (axiomas de quadro), sendo necessário definir uma relação de custo-benefício entre ambas. Por exemplo, na regra 2 da tabela 3.1, somente especificamos os predicados *roadWay*/2 na implicação pois o ganho não foi significativo utilizando os axiomas de quadro para as demais condições.

Na regra 2 da tabela 3.1, formalizamos a ideia de que, se não estamos visualizando qualquer linha amarela, não precisamos nos preocupar se a rodovia é de mão dupla, ou seja, não há o risco imediato de executar uma manobra proibida ou entrar na contramão. Esta regra otimiza a distribuição de probabilidade para o predicado $roadWay\{One,time\}$. Em testes preliminares sem esta regra, o resultado da consulta para $roadWay\{One,time\}$ era prejudicado devido ao espalhamento da distribuição de probabilidade nas situações onde o veículo se encontra na faixa direita ou central, ou seja, nas situações onde de fato não é possível inferir se a via é de mão única ou dupla.

Por sua vez, a regra 9 foi baseada em alguns experimentos no algoritmo de visão, onde percebemos que no acostamento raramente ocorre a detecção de linhas tracejadas brancas no lado sem demarcação de linha. Portanto, apesar de não ser um fato (por isto a expressão

"possivelmente está") esperamos inferir satisfatoriamente esta condição do acostamento (sem demarcação em um dos lados).

A regra 10 apresentou alguns desafios na modelagem do ambiente, apesar da aparente simplicidade. O conceito adotado no algoritmo de visão considera a posição de cada linha a partir do centro da imagem. Portanto no centro da imagem uma linha divisória pode oscilar entre as posições esquerda e direita. Esta situação é crítica durante uma mudança de faixa. Por exemplo, no início de uma mudança de faixa à esquerda o veículo está sobre a linha esquerda e permanecerá assim até a linha passar do centro da tela. Após passar pelo centro da tela, a linha que era esquerda passa a ser classificada como direita, porém o veículo ainda está mudando de faixa à esquerda. O mesmo acontece na mudança de faixa à direita. Na figura 3.11a, vemos o quadro 156 onde a linha amarela representa a detecção de uma linha à esquerda. Na figura 3.11b, vemos o quadro 160 onde a linha lilás representa a linha detectada à direita. Podemos notar que a mensagem de mudança de faixa à esquerda permanece nos dois quadros. A forma encontrada para prevenir que um predicado de cruzamento sobreponha-se ao outro indevidamente, foi negar o predicado que seria falso positivo com relação à mudança de faixa. Acrescentamos também, nas sentenças de cruzamento, a codificação da passagem da linha pelo centro da tela. Mesmo com estas ações, a inferência ainda apresenta uma resposta que necessita de uma avaliação posterior, esta condição será discutida no próximo capítulo.

a) 

b) 

Figura 3.11: Posição das linhas na mudança de faixa.

Fonte: Autor.

3.3.1 Treinamento do Modelo

Após a formalização lógica do domínio, o aprendizado dos pesos para cada fórmula foi executado com o algoritmo MC-SAT (DOMINGOS; LOWD, 2009) (conforme apresentado na Seção 2.4.1, p.38) em um total de 2700 quadros ímpares (segundo metodologia "meio-a-meio" (FITZPATRICK; SONKA, 2000)). Estes quadros são provenientes de 2 sequências distintas de

vídeo, isto é, de vídeos gravados em dias, horários e rodovias diferentes. Isto possibilita o treinamento e inferência sobre condições diferentes captadas em cada vídeo, por exemplo, iluminação natural em horários diferentes do dia, conservação da sinalização das estradas e até deslocamento da câmera. Portanto, o objetivo de usar vídeos distintos é tornar o modelo mais robusto de forma que este mantenha o desempenho de detecção nas mudanças ambientais que são comuns no domínio do tráfego rodoviário. O *ground truth* foi rotulado manualmente para os 5400 quadros de vídeo (quadros pares e ímpares) seguindo os mesmos critérios definidos para a modelagem do ambiente. O *ground truth* dos quadros ímpares foi usado na criação do arquivo de treinamento do *Alchemy*. O *ground truth* dos quadros pares foi usado na avaliação de desempenho do algoritmo (será discutida na Seção 4). Na tabela 3.2 apresentamos parte do arquivo de treinamento. Na tabela 3.3, temos o nosso modelo descrito na sintaxe do *Alchemy* após o aprendizado dos pesos. O peso w_i obtido significa que um mundo onde n cláusulas de uma fórmula são verdadeiras é $e^{w_i n}$ menos provável que um mundo onde todas as cláusulas são verdadeiras (vide Seção 2.4.1, p.32). Esta característica é importante para um LAS pois o modelo pode prover respostas plausíveis mesmo com evidências classificadas erroneamente. No apêndice A mostramos o conteúdo completo da modelagem no *Alchemy* antes e depois do treinamento.

Tabela 3.2: Parte do arquivo de treinamento do *Alchemy*.

```
//Frame 355
dividerL(WContinuous,Ok,355)        // evidência do sistema de visão para a linha esquerda
dividerR(WContinuous,Under,355)     // evidência do sistema de visão para a linha direita
carRelPos(Right,355)                // ground truth para posição
!crossingRight(355)                 // ground truth para cruzamento à direita
!crossingLeft(355)                  // ground truth para cruzamento à esquerda
!emergencyLane(355)                 // ground truth para acostamento
!prohibitedManoeuvre(355)           // ground truth para manobra proibida
!wrongWay(355)                      // ground truth para contramão
roadWay(One,355)                    // ground truth para mão dupla ou única

//Frame 357
dividerL(WDashed,Ok,357)
dividerR(WContinuous,Under,357)
carRelPos(Right,357)
!crossingRight(357)                 // idem quadro anterior
!crossingLeft(357)
!emergencyLane(357)
!prohibitedManoeuvre(357)
!wrongWay(357)
roadWay(One,357)
```

Fonte: Autor.

Na tabela 3.2 vemos que, embora as evidências indiquem um cruzamento à direita (*divi-*

derR(WContinuous,Under,355) e *dividerR(WContinuous,Under,357)*), o *ground truth* indica que não há cruzamento (*!crossingRight(355)* e *!crossingRight(357)*). Isto é devido ao problema da regra 10 (tab.3.1) apresentado na figura 3.11.

Tabela 3.3: Modelo na sintaxe do Alchemy.

```
//predicate declarations
carRelPos(lanepos!,time)
prohibitedManoeuvre(time)
wrongWay(time)
crossingLeft(time)
dividerL(type,status,time)
dividerR(type,status,time)
emergencyLane(time)
roadWay(way!,time)
crossingRight(time)

// 15.5852 dividerL(YContinuous,Under,t) v dividerR(YContinuous,s,t) => prohibitedManoeuvre(t) ^ wrongWay(t)
^ roadWay(Two,t) ^ carRelPos(Left,t) ^ !carRelPos(Centre,t) ^ !carRelPos(Right,t) ^ !emergencyLane(t)

// 2.17189 !dividerL(YContinuous,s,t) v !dividerL(YDashed,s,t) v !dividerR(YContinuous,s,t) v
!dividerR(YDashed,s,t)
=> roadWay(One,t) ^ !roadWay(Two,t)

// 61.4571 dividerL(YContinuous,Ok,t) v dividerL(YDashed,Ok,t) => roadWay(Two,t) ^ !roadWay(One,t) ^
carRelPos(Left,t) ^ !carRelPos(Centre,t) ^ !carRelPos(Right,t) ^ !emergencyLane(t) ^ !crossingLeft(t) ^
!crossingRight(t) ^ !prohibitedManoeuvre(t) ^ !wrongWay(t)

// 20.4789 dividerL(WContinuous,Ok,t) ^ dividerR(WDashed,Ok,t) => roadWay(One,t) ^ !roadWay(Two,t) ^
carRelPos(Left,t) ^ !carRelPos(Centre,t) ^ !carRelPos(Right,t) ^ !crossingLeft(t) ^ !crossingRight(t) ^
!prohibitedManoeuvre(t) ^ !wrongWay(t)

// 32.3628 dividerL(WDashed,Ok,t) ^ dividerR(WDashed,Ok,t) => carRelPos(Centre,t) ^ !carRelPos(Left,t) ^
!carRelPos(Right,t) ^ !emergencyLane(t) ^ !crossingLeft(t) ^ !crossingRight(t) ^ !prohibitedManoeuvre(t) ^
!wrongWay(t)

// 4.56344 !dividerL(WDashed,Ok,t) ^ dividerR(WDashed,Ok,t) => carRelPos(Left,t) ^ !carRelPos(Centre,t) ^
!carRelPos(Right,t) ^ !emergencyLane(t) ^ !crossingLeft(t) ^ !crossingRight(t)

// 17.638 dividerL(YDashed,Under,t) v (dividerR(YDashed,Ok,t) ^ dividerL(ty,s,t)) => wrongWay(t) ^
carRelPos(Left,t) ^ !carRelPos(Centre,t) ^ !carRelPos(Right,t) ^ !prohibitedManoeuvre(t) ^ !emergencyLane(t)

// 40.1161 (dividerR(WContinuous,Ok,t) v dividerR(Merge,Ok,t)) ^ dividerL(WDashed,s,t) => carRelPos(Right,t)
^ !carRelPos(Centre,t) ^ !carRelPos(Left,t) ^ !crossingLeft(t) ^ !crossingRight(t) ^ !prohibitedManoeuvre(t)
^ !wrongWay(t)

// 15.813 dividerL(WContinuous,Ok,t) ^ !dividerR(WDashed,Ok,t) => emergencyLane(t) ^ carRelPos(Right,t) ^
!carRelPos(Left,t) ^ !carRelPos(Centre,t) ^ !crossingLeft(t) ^ !crossingRight(t) ^ !prohibitedManoeuvre(t) ^
!wrongWay(t)

// -0.000385247 dividerR(ty1,Under,t1) v (dividerL(ty1,Under,t2) ^ dividerR(ty2,Ok,t2)) => crossingRight(t)
^ !crossingLeft(t)

// 0.000385247 dividerL(ty1,Under,t1) v (dividerR(ty1,Under,t2) ^ dividerL(ty2,Ok,t2)) => crossingLeft(t) ^
!crossingRight(t)
```

Fonte: Autor.

Na próxima seção mostramos os resultados da inferência efetuada no modelo apresentado.

4 RESULTADOS

Na seção anterior, modelamos o ambiente de tráfego e apresentamos o aprendizado dos pesos para o modelo desenvolvido. No modelo treinado, temos condições de efetuar inferências lógico-probabilísticas sobre os diversos predicados do problema. O algoritmo MC-SAT (Seção 2.4.1, p.32) foi usado na inferência sobre os 2700 quadros pares dos mesmos vídeos usados no treinamento (metodologia "meio-a-meio" (FITZPATRICK; SONKA, 2000)).

As evidências usadas na inferência foram geradas pelo algoritmo de visão a partir dos mesmos 2700 quadros pares. As evidências são a saída do algoritmo de visão, isto é, são as duas linhas detectadas e representadas pelos predicados *dividerL*/3 e *dividerR*/3. Dadas as evidências a cada quadro, a consulta foi feita para os seguintes predicados: *carRelPos*/2, *crossingLeft*/1, *crossingRight*/1, *emergencyLane*/1, *prohibitedManoeuvre*/1, *wrongWay*/1 e *roadWay*/2. Em uma consulta inferimos a probabilidade de cada instanciação de um predicado dadas as evidências. Por exemplo, em uma consulta sobre o predicado *roadWay*/2 no quadro de vídeo número 700 temos o seguinte: $P(roadWay(One,700) \mid dividerL(WDashed,Under,700), dividerR(Merge,Ok,700)) = 0,898$ e $P(roadWay(Two,700) \mid dividerL(WDashed,Under,700), dividerR(Merge,Ok,700)) = 0,102$.

Os resultados são apresentados na figura 4.1 com seus respectivos *ground truth*. Os valores de probabilidade de cada predicado estão representados com a cor cinza e os valores do *ground truth* estão em vermelho. No eixo X estão os números dos quadros e no eixo Y estão os valores de probabilidade.

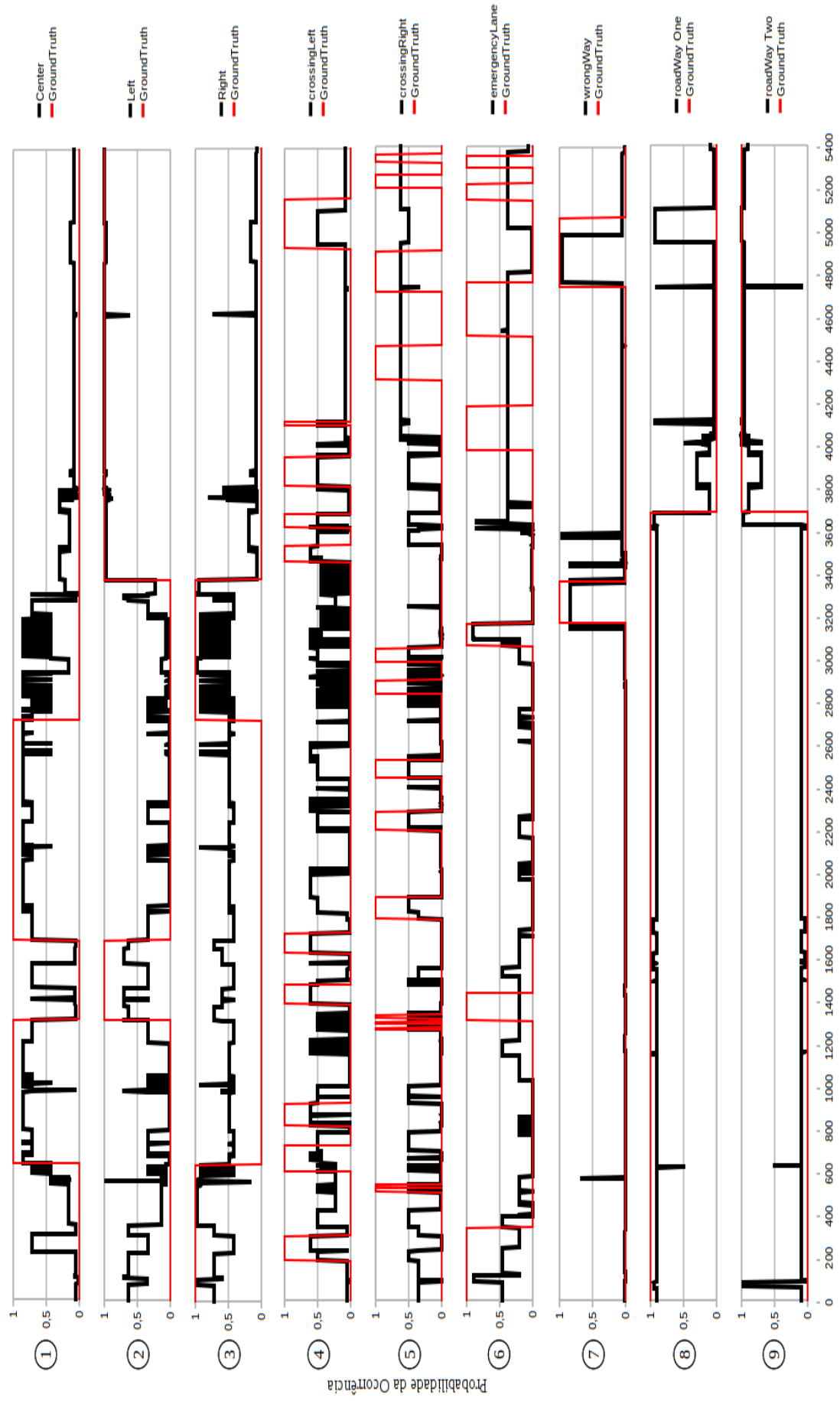

Figura 4.1: Probabilidades dos predicados consultados e seus respectivos *ground truths*.

Fonte: Autor.

No primeiro gráfico da figura 4.1 temos a probabilidade do predicado *carRelPos(Centre,t)* em função dos quadros. Vemos uma alta correspondência do predicado com o *ground truth*. Podemos observar também falsos positivos em torno do quadro 250 e do quadro 1500 e um ruído entre os quadros 2700 e 3100. Estes falsos positivos estão relacionados com a detecção da faixa de emergência (vide *ground truth* do gráfico 6 da figura 4.1) devido ao espaçamento do tracejado oscilar (conforme velocidade do veículo associada à amostragem da câmera).

No gráfico 2 da figura 4.1 apresentamos as probabilidades do predicado *carRelPos(Left,t)*. Observamos alguns falsos positivos nos quadros iniciais e oscilações abaixo de 0,5 na região onde o *ground truth* é zero. A probabilidade abaixo de 0,5 em torno do quadro 1500 tem alguma relação com o falso positivo do primeiro gráfico na mesma região. A partir do quadro 3600 a correspondência é quase total entre predicado e *ground truth*. Neste caso, o resultado a partir do quadro 3600 foi favorecido pela estrada não ter a pista central e consequentemente nenhuma ocorrência de tracejados brancos em ambos os lados do veículo.

No gráfico 3 da figura 4.1 as probabilidades do predicado *carRelPos(Right,t)* apresentam boa correspondência com o *ground truth*. Na região do quadro 250, o predicado apresenta uma probabilidade abaixo de 0,5 que parece ter relação com o falso positivo da mesma região no primeiro gráfico. Na região entre os quadros 600 e 2700, o predicado apresenta valores próximos a 0,5, porém a distinção entre o que é falso ou verdadeiro é perceptível. Após o quadro 3600 os valores do predicado passam a ser próximos do *ground truth* pelo mesmo motivo apresentado para o gráfico 2 nesta região.

Nos predicados *crossingLeft*/1 e *crossingRight*/1, respectivamente gráficos 4 e 5 da figura 4.1, percebemos que os valores que representam falso e verdadeiro são distinguíveis (ou próximos de zero ou próximos de 0,5). Na região entre os quadros 2700 e 3100, ambos apresentam ruídos, assim como nos gráficos 1 e 3, ou seja, no gráfico 5 vemos que em seu *ground truth* ocorrem 2 cruzamentos (à direita) no momento em que provavelmente ocorrem falsas detecções da linha do tipo *Merge*, por isto as posições oscilam entre central e direita. O *ground truth* foi definido, considerando um cruzamento sempre que a extremidade inferior de uma faixa aparecer na linha inferior de um quadro. O cruzamento permanecerá até que a faixa mude de lado (logo após o centro do quadro), o mesmo vale para a mudança de posição. Neste caso, o chaveamento excessivo dos gráficos 4 e 5 ocorre quando o veículo está muito próximo de uma das linhas divisórias e a detecção desta linha na parte inferior do quadro oscila. Esta oscilação tem origem no deslocamento do máximo local da transformada de Hough conforme varia a condição da faixa (cor, sujeira, falhas). Outra causa deste chaveamento é a própria concepção do *ground truth* de cruzamento, pois ao mudar de lado, a linha divisória ainda permanece com sua extremidade no limite inferior do quadro ocasionando uma falsa detecção de cruzamento para o lado oposto. Na

seção 4.2 será discutida esta condição e uma proposta para solucioná-la.

No modelo declaramos que, se um veículo está cruzando uma faixa, sua posição na estrada é indefinida pois verificamos, como esperado, que especificar a posição do veículo durante uma mudança de faixa degrada a predição dos predicados de cruzamento. Isto é devido à rede de Markov, onde os predicados de posição neste caso, passariam a fazer parte do mesmo clique dos predicados de cruzamento, compartilhando o mesmo peso e portanto, espalhando a distribuição de probabilidade.

No gráfico 6 da figura 4.1, o predicado *emergencyLane*/1 apresenta na região dos quadros 50 e 3150 uma alta correspondência com o *ground truth*. Vemos valores do predicado próximos a 0,5 nos quadros iniciais e finais. Isto também está relacionado com as falsas detecções de linha tracejada enquanto o veículo está na faixa de emergência. Podemos verificar também oscilações de aproximadamente 0,2 relacionadas com o término de algum dos cruzamentos, pois se o veículo está na pista direita ou esquerda há uma possibilidade de mudar para a faixa de emergência após um cruzamento.

Nos gráficos 7, 8 e 9 da figura 4.1, os predicados *wrongWay*/1, *roadWay(One,t)* e *roadWay(Two, t)* apresentam ótima correspondência em relação ao *ground truth*. Em geral, apresentam poucos falsos positivos de curta duração com exceção do predicado *roadWay(One,t)*, onde na região contínua entre os quadros 4950 e 5100 apresenta falsos positivos associados à detecção da linha divisória amarela como branca (devido iluminação do sol) .

Na figura 4.1, podemos ter uma boa noção visual de como o modelo proposto no capítulo 3.3 detecta cada situação de tráfego. Porém, a informação gráfica das probabilidades individuais de cada predicado é insuficiente para avaliar o desempenho do modelo. Portanto, na seção 4.1 apresentamos o critério utilizado para avaliação de desempenho do modelo e os valores obtidos na avaliação.

4.1 Avaliação

Para avaliar o modelo desenvolvido usamos a matriz de confusão conforme a tabela 4.1:

Com uma matriz de confusão para cada uma de nossas consultas nós podemos medir a acurácia, sensitividade, precisão e especificidade de uma predição. A acurácia é dada por:

$$acurácia = \frac{pv + nv}{(pv + nv + fp + fn)}. \tag{4.1}$$

A acurácia é a medida de confiabilidade de detecção, isto é, do número total de predições

Tabela 4.1: Matriz de confusão para avaliar o modelo.

		Decisão do Algoritmo	
		Predicado verdadeiro	Predicado falso
Ground Truth	Predicados verdadeiros	positivo verdadeiro (**pv**)	falso negativo (**fn**)
	Predicados falsos	falso positivo (**fp**)	negativo verdadeiro (**nv**)

Fonte: Fitzpatrick e Sonka (2000).

quantas são corretas.

A sensitividade ou recuperação é a fração das instanciações verdadeiras detectadas como tais e é dada por:

$$sensitividade = \frac{pv}{(pv+fn)}. \tag{4.2}$$

A precisão é a medida de acurácia para uma classe específica, em nosso modelo estamos interessados na precisão de detecção das instanciações verdadeiras, isto é, o algoritmo reporta posição e comportamento do veículo sem falsas detecções. A precisão é dada como:

$$precisão = \frac{pv}{(pv+fp)}. \tag{4.3}$$

A especificidade é a medida de frequência com que o algoritmo reporta instanciações falsas como tais e é dada por:

$$especificidade = \frac{nv}{(nv+fp)}. \tag{4.4}$$

Com os limiares de decisão individuais da tabela 4.2 executamos a inferência para os 2700 quadros pares. Na tabela 4.3 apresentamos os resultados obtidos na inferência. Apresentamos também na tabela 4.4 os resultados obtidos somente com o algoritmo de visão. Lembramos que os algoritmos de inferência e de visão possuem em comum somente a detecção da mudança de faixa, as demais instâncias ou características detectadas são específicas para cada algoritmo (conforme apresentado nos capítulos 3 e 3.3).

Os limiares individuais para cada predicado da tabela 4.3 foram definidos empiricamente baseados nos itens de 1 a 9 da figura 4.1. Este procedimento é plausível, considerando que o comportamento do modelo é conhecido e os valores entre falso e positivo são perfeitamente

Tabela 4.2: Limiares de decisão por predicado.

Predicado	**Limiar de Decisão**
carRelPos(Centre)	0,70
carRelPos(Left)	0,60
carRelPos(Right)	0,70
crossingLeft	0,50
crossingRight	0,45
emergencyLane	0,40
prohibitedManoeuvre	0,50
wrongWay	0,70
roadWay(One)	0,70
roadWay(Two)	0,70

Fonte: Autor.

distinguíveis. A implementação destes limiares individuais seria trivial na camada cinco da figura 1.1. Na seção 4.2 discutiremos esta decisão. Todos os predicados da tabela 4.3 podem ser utilizados em um sistema de assistência ao motorista. Os predicados de posição e sentido da via (*carRelPos/2, roadWay/2*) podem ser auxiliares ao GPS e os predicados de comportamento (*crossingLeft/1, crossingRight/1, prohibitedManoeuvre/1, wrongWay/1, emergencyLane/1*) podem ser usados para gerar os alertas ao motorista.

Tabela 4.3: Resultados obtidos na inferência. O símbolo * indica que não houve positivos verdadeiros para o predicado.

Predicado	**Acurácia (%)**	**Sensitividade (%)**	**Precisão (%)**	**Especificidade (%)**
carRelPos(Centre)	91,0	98,9	75,2	88,1
carRelPos(Left)	93,9	93,3	89,5	93,3
carRelPos(Right)	90,8	64,4	87,2	97,6
crossingLeft	87,3	74,6	65,3	90,4
crossingRight	76,4	89,2	41,3	73,8
emergencyLane	80,9	78,8	45,9	81,4
prohibitedManoeuvre	99,7	*	*	100
wrongWay	98,0	79,3	89,6	99,3
roadWay(One)	96,7	99,9	95,4	89,6
roadWay(Two)	98,4	99,8	95,3	97,7
Média	*91,3*	*86,5*	*76,1*	*91,1*
Desvio Padrão σ	*7,2*	*12,1*	*19,6*	*7,9*

Fonte: Autor.

Na tabela 4.3, os predicados *crossingLeft*/1 e *crossingRight*/1 apresentam um ganho em sensitividade se comparados com os mesmos predicados da tabela 4.4. Do ponto de vista prático,

Tabela 4.4: Resultados obtidos com o algoritmo de visão.

Característica	**Acurácia (%)**	**Sensitividade (%)**	**Precisão (%)**	**Especificidade (%)**
crossingLeft	87,8	54,5	74,9	95,7
crossingRight	87,3	51,1	66,9	94,8
Merge	96,2	40,2	62,2	98,9
WContinuous	98,1	93,3	98,9	99,7
WDashed	95,3	99,2	91,0	92,3
YContinuous	95,4	100	64,1	95,0
YDashed	95,5	42,2	95,0	99,8
Média	*93,6*	*68,6*	*79,0*	*96,6*
Desvio Padrão σ	*4,0*	*25,5*	*14,5*	*2,7*

Fonte: Autor.

pode-se dizer que um sistema é mais seguro quanto maior for sua sensitividade, pois as situações reais de risco serão reportadas com maior frequência. Nota-se também na tabela 4.3 que, na mesma comparação com a tabela 4.4 para os predicados *crossingLeft*/1 e *crossingRight*/1, há uma redução em especificidade, precisão e, no caso de *crossingRight*/1, há também redução de acurácia. Em geral, a otimização da sensitividade ocorre em detrimento principalmente da especificidade (FITZPATRICK; SONKA, 2000). No caso de um LAS, a preferência é por sinalizar algo ao motorista mesmo que não haja perigo (baixa especificidade) ao invés de não alertar uma situação real de perigo (baixa sensitividade).

Os menores valores obtidos na tabela 4.3 foram: 76,4% de acurácia (*crossingRight*/1), 64,4% de sensitividade (*carRelPos(Right,t)*), 41,3% de precisão (*crossingRight*/1) e 73,8% de especificidade (*crossingRight*/1). Com exceção dos 41,3% de precisão para o predicado *crossingRight*/1, os demais valores se encontram bem acima da escolha aleatória ou chance (50%), o que indica um bom desempenho do algoritmo de inferência no pior caso. Esta baixa precisão do predicado *crossingRight*/1 tem origem no elevado número de falsos positivos que podemos ver na figura 4.1 item 5.

Por sua vez, na tabela 4.4 os menores valores foram: 87,3% de acurácia (*crossingRight*), 40,2% de sensitividade (*Merge*), 62,2% de precisão (*Merge*) e 92,3% de especificidade (*WDashed*). Neste caso, o algoritmo de visão também apresenta bom desempenho no pior caso com exceção da sensitividade para a linha do tipo *Merge*. Uma baixa sensitividade pode ter origem em um número elevado de falsos negativos. Vemos como consequência desta falha na visão, os falsos negativos no item 6 da figura 4.1.

Os maiores valores de desempenho na tabela 4.3 foram: 99,7% de acurácia (*prohibitedManoeuvre*/1), 99,9% de sensitividade (*roadWay(One,t)*), 95,4% de precisão (*roadWay(One,t)*)

e 100% de especificidade (*prohibitedManoeuvre*/1). Na tabela 4.4 os maiores valores foram: 98,1% de acurácia (*WContinuous*), 100% de sensitividade (*YContinuous*), 98,9% de precisão (*WContinuous*) e 99,83% de . Isto demonstraria um excelente desempenho no melhor caso de ambos os algoritmos. Porém, o predicado *prohibitedManoeuvre*/1 apresenta estes valores devido a ausência de positivos verdadeiros, isto é, não foram detectadas as condições de manobra proibida conforme o *ground truth*. Este problema é devido à falha na detecção da linha tracejada amarela pelo algoritmo de visão, razão pela qual *YDashed* apresenta alta especificidade. Esta condição e uma possível solução serão discutidos na próxima seção.

Os valores imediatamente anteriores à acurácia e especificidade de *prohibitedManoeuvre*/1 são: 98,4% de acurácia para *roadWay(Two,t)* e 99,3% de sensitividade para *wrongWay*/1. No caso da especificidade de *YDashed* o valor imediatamente anterior é de 99,7% de especificidade para *WContinuous*. Portanto, ainda assim demonstra-se um ótimo desempenho no melhor caso.

A média de performance é uma métrica para avaliar o algoritmo como um todo (FITZPATRICK; SONKA, 2000). Por sua vez, o desvio padrão apresenta a dispersão das múltiplas saídas do algoritmo em relação à média. Avaliando o desvio padrão na tabela 4.3, vemos que a maior dispersão ocorre para a precisão ($\sigma = 19,6\%$). Nota-se que os predicados *crossingRight*/1 e *emergencyLane*/1 são responsáveis por este valor de dispersão devido seus valores de precisão reduzirem a média significativamente. Por sua vez, na tabela 4.4 vemos a maior dispersão em sensitividade ($\sigma = 25,5\%$), gerada pela baixa sensitividade das variáveis *crossingLeft* (54,5%), *crossingRight* (51,1%), *Merge* (40,2%) e *YDashed* (42,2%).

Com base nestes dados de sensitividade em ambas as tabelas, podemos notar que o algoritmo de inferência conseguiu manter uma boa sensitividade (acima de 64%) apesar da baixa sensitividade apresentada pelo algoritmo de visão em diversas variáveis. Vemos também em ambas as tabelas (4.3 e 4.4) um baixo desvio padrão ($\sigma \leq 7,9$) para a acurácia e especificidade, o que indica que todos os predicados e variáveis apresentam um valor próximo ao da média, isto é, para todos os predicados e variáveis temos uma boa confiabilidade de detecção (acurácia em torno de 91,3% e 93,6% para a inferência e a visão respectivamente) favorecida principalmente pela taxa de negativos verdadeiros corretamente detectados (especificidade em torno de 91,1% e 96,6% para a inferência e a visão respectivamente).

Na próxima seção discutiremos os resultados obtidos e suas respectivas avaliações.

4.2 Discussão

O limiar de decisão foi definido inicialmente para as probabilidades maiores que 50%. Podemos notar na figura 4.1 que este limiar poderia ser maior para os predicados de posição (*carRelPos*/2), porém os predicados de comportamento (por exemplo, *crossingLeft*/1 e *crossingRight*/1) tem probabilidades menores que os demais predicados. No entanto, a distinção entre estes predicados é bem clara, isto é, a diferença entre os valores de probabilidade que definem os patamares falso e verdadeiro é facilmente percebida. A causa da baixa probabilidade dos predicados de comportamento ocorre durante a mudança de faixa, quando a linha divisória direita se transforma em esquerda (e vice-versa) após passar pelo centro da imagem (vide Cap.3.3, fig.3.11, p.61). Portanto, até o veículo terminar a mudança de faixa, precisamos evitar que *crossingLeft*/1 tenha probabilidade maior que *crossingRight*/1 se o veículo estiver cruzando à direita. Mesmo apresentando uma probabilidade baixa vemos que os valores de falso e verdadeiro são perfeitamente distinguíveis. Porém, ainda não há condições de uma interpretação correta considerando somente as probabilidades obtidas, pois em determinados momentos as probabilidades de ambos os predicados de cruzamento são iguais e ocorrem simultaneamente como podemos ver entre os quadros 2200 e 2600. No diagrama proposto (fig.1.1), poderíamos tratar este problema em particular (*crossingLeft*/1 versus *crossingRight*/1) na camada cinco, analisando qual predicado iniciou a mudança de faixa e ignorando a ocorrência do outro predicado se esta ocorrência se tornar concorrente. Na figura 4.2 descrevemos graficamente como seria a solução proposta.

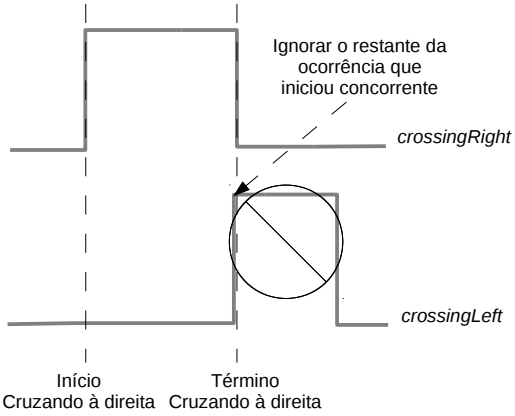

Figura 4.2: Otimização proposta para raciocinar sobre o cruzamento.

Fonte: Autor.

Sugerimos deixar o tratamento do cruzamento na camada seguinte, pois modificar o mo-

delo para esta condição tornou-se inviável. Na primeira tentativa de modificação, adequamos o modelo para usar 2 pares de evidências: o par atual e o do quadro anterior. Especificamos então, que se a evidência anterior indicava um cruzamento à direita, a evidência atual não poderia indicar um cruzamento à esquerda (análogo para o outro lado). O resultado obtido foi muito próximo do resultado da figura 4.1, pois só resolvia o instante da transição. Concluímos que uma solução melhor seria verificar o estado anterior, não somente as evidências, mas esta ação tornaria o modelo muito complexo e de alto custo computacional. Seria necessário criar predicados que codificassem os diversos estados que o modelo poderia assumir, armazenando o estado do quadro anterior. Porém, nos métodos de inferência probabilística em geral, não podemos usar um predicado de consulta como evidência, ou seja, não podemos usar o estado consultado anteriormente como evidência na próxima inferência, a não ser que a informação do estado anterior seja externa (do algoritmo de visão) ou retroalimentada (fig.1.1, p.14). Esta solução porém, demanda um estudo mais abrangente em trabalhos futuros.

Outra questão é em relação ao predicado *prohibitedManoeuvre/1*, o qual não foi plotado na figura 4.1 pois suas instâncias positivas não foram detectadas. Esta falha teve origem no algoritmo de visão e posteriormente no treinamento. Na segunda sequência de vídeo (a partir do quadro 3600), a estrada é de mão dupla com linha divisória amarela dupla em algumas regiões. No algoritmo de visão, a detecção das linhas é baseada em máximos locais respeitando uma distância mínima entre as linhas esquerda e direita. No momento em que houve uma linha dupla com permissão de ultrapassagem para o egoveículo, o algoritmo detectou a linha amarela contínua em vez da tracejada. Portanto, no treinamento foi aprendido com várias ocorrências que se a linha esquerda for amarela contínua, *prohibitedManoeuvre/1* será falso. Esta falha tornou o peso negativo desta condição na primeira regra da tabela 3.1 (p.58). Uma sugestão para resolver este problema é modificar o algoritmo de visão, mantendo o conceito atual, para detectar três linhas em vez de duas. Consequentemente seria necessário modificar o modelo em MLN para considerar mais três tipos de linhas divisórias.

O bom desempenho na detecção do predicado *roadWay(One)*, dada pelos valores: 96,7% de acurácia, 99,9% de sensitividade, 95,4% de precisão e 89,6% de especificidade, pode ser atribuída, em geral, à asserção da regra 2 na tabela 3.1 (p.58): “se não há evidência para uma estrada de mão dupla, considere-a como mão única”. De fato, é razoável dizermos que se não estamos vendo uma faixa amarela, não precisamos nos preocupar se a estrada tem mão dupla. No caso do predicado *roadWay(Two)*, podemos atribuir a qualidade de detecção (98,4% de acurácia, 99,8% de sensitividade, 95,3% de precisão e 97,7% de especificidade) à duas características: a boa acurácia na detecção das cores amarelas pelo algoritmo de visão (95,4% para *YDashed* e 95,5% para *YContinuous*) e ao fato de poucos predicados terem relação causal direta com as

linhas divisórias amarelas (regras de 1 a 3, tab.3.1, p.58).

A reduzida precisão dos predicados de cruzamento (65,3% para *crossingLeft*/1 e 41,3% para *crossingRight*/1) é devida à mudança de lado da linhas no centro da tela como explanado anteriormente (Cap.3.3, fig.3.11, p.61). Teoricamente, se tivéssemos o mesmo número de ocorrência de ambos os predicados de cruzamento, a precisão de cada um seria próxima de 50 %. Por sua vez, a baixa precisão (45,9%) do predicado *emergencyLane/1* está diretamente relacionada ao algoritmo de visão. Podemos ver na tabela 4.4 que a sensitividade (40,2%) e precisão (62,2%) para o predicado *Merge* foram prejudicadas. Neste caso, o limiar de decisão da visão para a linha do tipo *Merge* é muito sensível à velocidade do veículo. I.e, com velocidade baixa a linha *Merge* é detectada como tracejada, e em alta velocidade é detectada como contínua.

O limiar de decisão utilizado inicialmente (50%), ainda apresentava resultados insatisfatórios nos predicados de cruzamento. Portanto, foi necessário estudar uma forma de incrementar o desempenho do algoritmo utilizando a curva característica de operação do receptor (*Receiver Operating Characteristic* ou ROC) (FITZPATRICK; SONKA, 2000). O ROC é uma ferramenta padrão para definir um espaço de custo-benefício entre a sensitividade e a especificidade. Sendo assim, temos condições de decidir se o aumento em sensitividade compensa a redução em especificidade em nossa aplicação. Numa curva ROC, tipicamente plotamos a taxa ou fração de positivo verdadeiro (FPV ou sensitividade) no eixo Y e a taxa ou fração de falso positivo (FFP ou 1-especificidade) no eixo X. O ponto ideal de operação se encontra onde $FPV = 1$ e $FFP = 0$ (FITZPATRICK; SONKA, 2000).

Na figura 4.3, apresentamos a curva ROC para o nosso modelo em MLN sem considerar o predicado *prohibitedManoeuvre*/1 (MLN), tendo em vista que a causa raiz da falha está no algoritmo de visão. Apresentamos também a curva ROC para o modelo considerando o predicado *prohibitedManoeuvre*/1 (pM) e também, a título de comparação, a curva da escolha aleatória. As curvas foram geradas à partir de cinco pontos, cada qual definido pela média de desempenho para um limiar de decisão (30%, 40%, 50%, 60% e 70%).

Podemos notar que mesmo considerando o predicado *prohibitedManoeuvre*/1 o algoritmo ainda apresenta um bom desempenho, ou seja, a curva do algoritmo está bem distante da curva da escolha aleatória.

Na figura 4.3, há também um ponto que representa o limiar individual. Este ponto representa a média de desempenho do algoritmo para os limiares de decisão individualizados na tabela 4.2. A localização deste ponto acima da curva ROC caracteriza um melhor desempenho do algoritmo, portanto comprova a eficácia do uso de limiares individuais por predicado.

Considerando os resultados expostos e discutidos neste capítulo, concluímos o presente

trabalho no próximo capítulo.

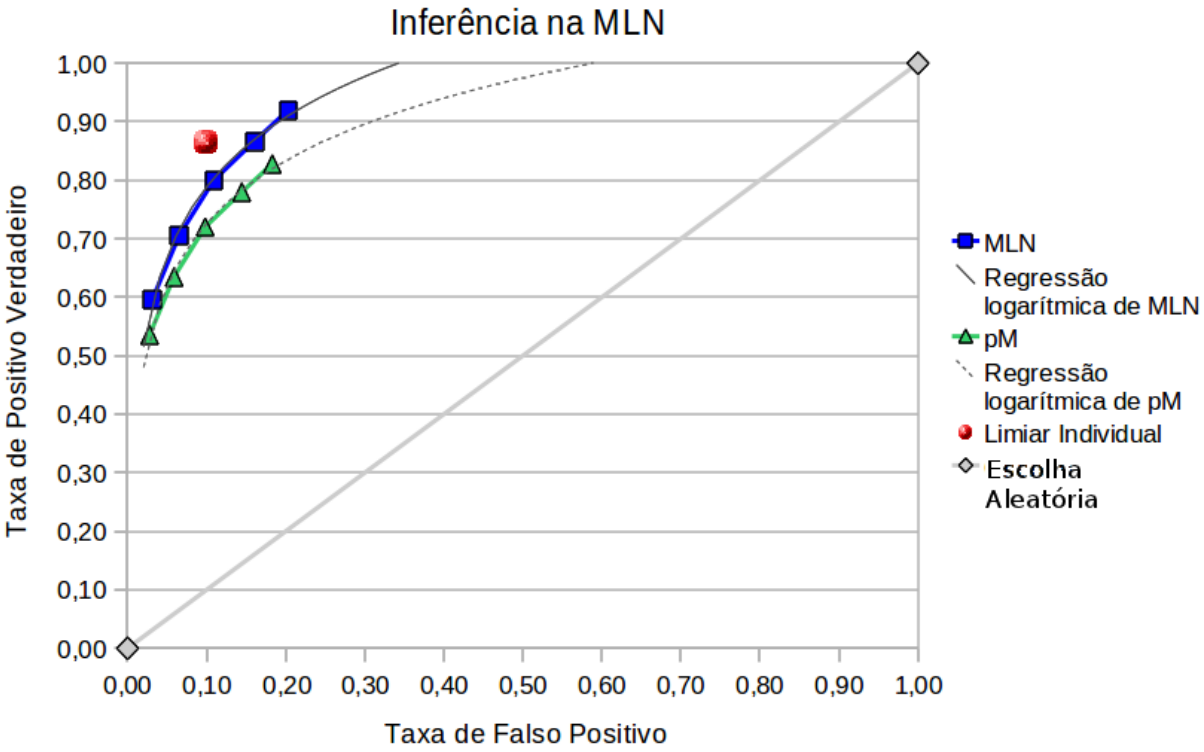

Figura 4.3: Espaço ROC para o algoritmo de inferência.

Fonte: Autor.

5 CONCLUSÃO

Neste trabalho, apresentamos um modelo de raciocínio lógico-probabilístico que faz parte de um arcabouço orientado à um sistema assistente de faixa de trânsito (do inglês *Lane Assistant System* ou LAS). O desenvolvimento de soluções em segurança para sistemas automotivos tem aumentado nas últimas décadas e não está fundamentado somente nas estatísticas de acidentes. A indústria automotiva em geral, acompanha e contribui com a evolução tecnológica propondo meios para aumentar tanto a segurança do motorista quanto o seu conforto e prazer ao dirigir. Este trabalho compartilha também desta paixão por automóveis e tecnologia.

Um desafio enfrentado em sistemas automotivos nos últimos anos tem relação com a informação visual captada por câmeras. Devido ao grande volume de informação proveniente até das mais simples das câmeras, foi necessário desenvolver hardwares e softwares mais eficientes para processar estes dados. Neste ponto, alguns sistemas comerciais apresentam uma binarização da região de interesse e detecção das linhas divisórias por meio do gradiente sobre linhas transversais da imagem. No trabalho apresentado, usamos uma solução que também executa a binarização da região de interesse, mas detecta as linhas divisórias por meio da transformada de Hough. Esta solução aparenta ser menos sensível à imperfeições nas linhas divisórias (sujeira ou falhas), porém ainda sofre com variação de iluminação, velocidade do veículo e grandes oclusões. Soluções desenvolvidas com processamento de imagens em estéreo resolvem parte destes problemas, mas necessitam de hardware dedicado (GPU) para possibilitar um processamento em tempo real. Neste trabalho atingimos tempo real de processamento (incluindo o algoritmo de inferência) com o hardware descrito no capítulo 3. Este hardware é um computador portátil (*notebook*) cuja configuração já está ultrapassada por uma geração de processadores neste momento. Portanto, com o rápido avanço tecnológico em geral, a capacidade de processamento deixa de ser um problema para o modelo apresentado. Porém, devemos tomar o cuidado de criar algoritmos cada vez mais eficientes para possibilitar a modelagem e o processamento de problemas mais complexos no futuro.

Após tratar os dados de visão conforme as premissas apresentadas nos capítulos 3 e 3.3, modelamos o ambiente de tráfego rodoviário com redes lógicas de Markov (MLN). Esta modelagem pode ser considerada a principal contribuição deste trabalho, pois utiliza uma ferramenta relativamente nova (MLN) para representar um domínio em lógica probabilística. Além disso, nesta modelagem utilizamos o conceito de raciocínio espacial qualitativo para tratar a informação espacial do ambiente (posição das linhas e do veículo) de forma discreta. Tivemos de lidar também com alguns aspectos do problema da persistência (*frame problem*), ou

seja, como especificar o que permanece inalterado após uma ação sem tornar o modelo muito complexo. Neste caso, tivemos de incluir alguns axiomas de quadro (negação de um predicado inalterado) para os estados que compartilhavam o mesmo clique. Nesta modelagem portanto, levamos em consideração uma relação de custo-benefício entre a complexidade do modelo e o desempenho na inferência.

Outra característica deste trabalho foi a aplicação do arcabouço proposto (camadas de 1 a 4 da figura 1.1) em cenas reais de tráfego, captando situações de oclusão, oscilação da linha do horizonte, vibração da câmera e iluminação de outros veículos (farol e sinalização). Apesar do algoritmo de visão efetuar falsas detecções em alguns destes casos, obtivemos boa reposta a partir do modelo em MLN. Portanto, a modelagem em MLN proporcionou uma forma de lidar com tais problemas indiretamente por meio da abordagem lógico-probabilística, o que permitiu inferir a probabilidade de ocorrência de cada predicado mesmo na presença de eventos inesperados e, consequentemente, de classificação errônea das linhas.

Para inferir estas condições de trânsito usamos o programa de código aberto *Alchemy* (Seção 2.4.3). O uso do *Alchemy* proporcionou uma economia de tempo na obtenção dos resultados, pois como este programa apresenta diversos algoritmos de treinamento e inferência implementados, com poucas parametrizações pudemos verificar que o algoritmo de propagação de crença (*belief propagation*) foi mais eficiente que o algoritmo MC-SAT.

A inferência com o algoritmo MC-SAT neste modelo foi eficiente mas não em tempo real no equipamento utilizado (Cap.3, p.47). Obtivemos aproximadamente 0,15 segundos de inferência por quadro (com o MC-SAT), mas em alguns casos este tempo atingiu 42 segundos, pois o MC-SAT instancia todas as consultas e em algum momento leva mais tempo para sair de uma região isolada da rede de Markov (seção 2.4.1). Obtivemos tempo real de processamento usando o algoritmo de propagação de crença (Seção 2.4.1, p.38), obtendo aproximadamente 0,02 segundos por quadro. Resultados similares foram obtidos com ambos os algoritmos, a diferença da propagação de crença é que alguns predicados não aparecem nos resultados. Verificamos então, que os predicados consultados com MC-SAT para os mesmos quadros de vídeo onde a propagação de crença os omitiu, eram negativos verdadeiros (probabilidade próxima de zero). Portanto, consideramos estes predicados omitidos como negativos verdadeiros. Confirma-se então, que embora não haja garantia de convergência e acurácia com a propagação de crença, na prática obtemos resultados satisfatórios.

Com os resultados obtidos na inferência lógico-probabilística, avaliamos o desempenho do modelo desenvolvido por meio de uma matriz de confusão (Seção 4.1, p.68) que permitiu definir a acurácia, a sensitividade, a precisão e a especificidade da inferência neste modelo. Efe-

tuamos então uma análise no espaço ROC (Seção 4.2, p.75) a fim de determinar o melhor limiar de decisão para o algoritmo de inferência e avaliar graficamente seu desempenho. Comprovamos por meio da curva ROC, que a definição de limiares de decisão individuais para cada predicado melhorou o desempenho na inferência. O bom desempenho e o processamento em tempo real são características primordiais para a aplicação prática deste modelo em projetos futuros.

5.1 Trabalhos Futuros

O presente trabalho tem condições de evoluir em diversos pontos conforme segue:

a) o algoritmo de visão pode ser otimizado com uma variável de velocidade na rotina de contagem dos pixels brancos para melhorar a detecção das linhas do tipo *Merge*. Outro ponto seria a detecção da terceira faixa e criação de variáveis para as possibilidades de linha central (dupla com tracejada à esquerda ou à direita e dupla contínua) a fim de melhorar a detecção das linhas tracejadas amarelas e situações de manobra proibida;

b) a detecção das linhas divisórias pode ser por meio da análise da mudança de um pixel preto para um pixel branco e vice-versa na região de uma linha horizontal, em vez de usar a transformada de Hough. É a forma que alguns modelos comerciais de LAS usam e pode ser mais eficiente se for bem parametrizada;

c) a modelagem do problema pode ser refinada, mudando o conceito de raciocínio espacial qualitativo para raciocínio sobre ações. Com o cálculo de situação (REITER, 2001) ou o cálculo de eventos (SHANAHAN, 1996) poderíamos modelar as ações do egoveículo de modo a evitar o problema espacial do cruzamento das linhas. Por exemplo, poderíamos criar fluentes para o início e o final de um cruzamento de forma a englobar as posições das linhas (ações) nos diversos estados possíveis (situações). A modelagem nestas linguagens é mais complexa e até o momento não há algoritmo no *Alchemy* para tal inferência. Porém, o conceito de MLN pode ser estendido ao modelo;

d) em uma evolução para detectar outros veículos e obstáculos, torna-se necessário o uso de um par de câmeras em estéreo. Isto proporciona informação de profundidade do ambiente, porém o processamento das imagens é complexo e de alto custo computacional para os computadores atuais. Este custo computacional pode ser otimizado com o uso de placas de processamento gráfico. O modelo lógico-probabilístico também precisaria ser adequado às novas informações do ambiente;

e) implementar as camadas 5 e 6 do diagrama proposto (fig.1.1, p.14). A camada 6 seria a implementação dos atuadores no veículo (aviso sonoro, visual ou controle de direção) de

modo que respondam conforme a saída da camada 5 (limiares de decisão e hierarquia de avisos);

f) obter vídeos em outras condições ambientais como: dias ensolarados, chuvosos e à noite entre outros. Estas condições representam um grande desafio para o sistema de visão. Resolvendo estes problemas, teríamos um grande volume de informação que tornaria o treinamento extremamente demorado, pois toda a rede de Markov é instanciada durante o treinamento. Esta tarefa porém, pode ser executada em um *cluster* de computadores para agilizar o processamento. Estes estudos confirmariam a escalabilidade do modelo implementado.

Em geral, mantendo-se o conceito do trabalho ora apresentado, os pontos que apresentam maiores possibilidades de otimização tem relação com o sistema de visão. Em relação à inferência, outros algoritmos e opções de parametrização do *Alchemy* podem ser testados, associados a algumas adequações do modelo em MLN a fim de estabelecer comparação de desempenho entre as possíveis abordagens.

REFERÊNCIAS

AMSEL, C. et al. **Automobilelektronik**. 3. ed. Wiesbaden: Vieweg Teubner, 2009. ISBN 978-3-8348-0446-4.

ASAI, T. et al. 3D line reconstruction of a road environment using an in-vehicle camera. In: BEBIS, G. et al. (Ed.). **ISVC - International Symposium on Visual Computing**. [S.l.]: Springer, 2008. (Lecture Notes in Computer Science, v. 5359), p. 897–904. ISBN 978-3-540-89645-6.

BACCHUS, F. **Representing and reasoning with probabilistic knowledge: a logical approach to probabilities**. Cambridge: MIT press, 1990. ISBN 0-262-02317-2.

BERTOZZI, M. et al. Artificial vision in road vehicles. In: IEEE. **Proceedings of the IEEE**. [S.l.], 2002. v. 90, n. 7.

BICAS, H. E. A. Fisiologia da visão binocular. **Arquivos brasileiros de oftalmologia**, v. 67, n. 1, p. 172–180, 2004.

BLACK, M. J.; ANANDAN, P. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. **Computer vision and image understanding**, Elsevier Science, New York, v. 63, n. 1, p. 75–104, 1996. ISSN 1077-3142.

BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. In: **COLT '92: Proceedings of the fifth annual workshop on computational learning theory**. New York: ACM, 1992. p. 144–152. ISBN 0-89791-497-X.

BOUTHEMY, P.; FRANCOIS, E. Motion segmentation and qualitative dynamic scene analysis from an image sequence. **International journal of computer vision**, Springer, Netherlands, v. 10, p. 157–182, 1993. ISSN 0920-5691.

BRADSKI, G.; KAEHLER, A. **Learning OpenCV**. 1. ed. [S.l.]: O'Reilly Media, 2008. ISBN 978-0-596-51613-0.

BRAZ, R. de S.; AMIR, E.; ROTH, D. A survey of first-order probabilistic models. In: HOLMES, D. E.; JAIN, L. C. (Ed.). **Innovations in Bayesian networks**. [S.l.]: Springer, 2008, (Studies in computational intelligence, v. 156). p. 289–317. ISBN 978-3-540-85065-6.

BREIMAN, L. et al. **Classification and regression trees**. Monterey, CA: Wadsworth and Brooks, 1984.

CAMPOS, C. P. de; COZMAN, F. G.; LUNA, J. E. O. Assembling a consistent set of sentences in relational probabilistic logic with stochastic independence. **Journal of applied logic**, Amsterdam, v. 7, n. 2, p. 137–154, 2009.

CANNY, J. A computational approach to edge detection. **IEEE transactions on pattern analysis and machine intelligence**, IEEE computer society, Washington, DC, v. 8, n. 6, p. 679–698, 1986. ISSN 0162-8828.

CHELLA, A.; FRIXIONE, M.; GAGLIO, S. Understanding dynamic scenes. **Artificial intelligence**, Elsevier Science, Essex, v. 123, p. 89–132, 2000. ISSN 0004-3702.

COHN, A. G.; HAZARIKA, S. M. Qualitative spatial representation and reasoning: an overview. **Fundamenta informaticae**, Netherlands, v. 46, n. 1-2, p. 1–29, 2001.

COHN, A. G.; RENZ, J. Qualitative spatial representation and reasoning. In: ____. [S.l.]: Elsevier, 2007. cap. 1.

____. Qualitative spatial representation and reasoning. **Qualitative reasoning and decision technologies**, Elsevier, v. 6526, n. 07, p. 551–596, 2008.

COMANICIU, D.; MEER, P. Mean shift: a robust approach toward feature space analysis. **IEEE transactions on pattern analysis and machine intelligence**, v. 24, n. 5, p. 603–619, 2002. ISSN 0162-8828.

COMANICIU, D.; RAMESH, V.; MEER, P. Kernel-based object tracking. **IEEE transactions on pattern analysis and machine intelligence**, IEEE computer society, Washington, DC, v. 25, n. 5, p. 564–575, 2003. ISSN 0162-8828.

COOTES, T. F.; EDWARDS, G. J.; TAYLOR, C. J. Active appearance models. **IEEE transactions on pattern analysis and machine intelligence**, Springer, v. 23, n. 6, p. 681–685, 2001.

DAIMLER. **Milestones in vehicle safety. The vision of accident-free driving.** Stuttgart, 2009.

____. **Shaping future transportation. Safe drive technologies.** Stuttgart, 2009.

DNER - DEPARTAMENTO NACIONAL DE ESTRADAS DE RODAGEM. **Glossário de Termos Técnicos Rodoviários**. Rio de Janeiro, 1997.

DOMINGOS, P.; LOWD, D. **Markov Logic: an interface layer for artificial intelligence**. [S.l.]: Morgan & Claypool, 2009.

DUDA, R. O.; HART, P. E. Use of the hough transformation to detect lines and curves in pictures. **Communications of the ACM**, ACM, v. 15, n. 1, p. 11–15, 1972.

FIEGUTH, P.; TERZOPOULOS, D. Color-based tracking of heads and other mobile objects at video frame rates. **Computer vision and pattern recognition**, IEEE Comput. Soc, p. 21–27, 1997.

FITZPATRICK, J. M.; SONKA, M. Handbook of medical imaging. medical image processing and analysis. In: ____. 1. ed. [S.l.]: SPIE press, 2000. v. 2, cap. 10, p. 567–605. ISBN 0819436224.

FIX, E.; HODGES, J. L. **Discriminatory analysis: nonparametric discrimination: consistency properties**. Randolf Field, Texas, 1951. 261–279 p.

FRANKE, U. et al. Towards optimal stereo analysis of image sequences. In: SOMMER, G.; KLETTE, R. (Ed.). **Robot vision**. [S.l.]: Springer, 2008. (Lecture Notes in Computer Science, v. 4931), p. 43–58. ISBN 978-3-540-78156-1.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. In: **EuroCOLT '95: Proceedings of the european conference on computational learning theory**. London: Springer-Verlag, 1995. p. 23–37. ISBN 3-540-59119-2.

GALTON, A. P. Lines of sight. In: AISB - ARTIFICIAL INTELLIGENCE AND THE SIMULATION OF BEHAVIOUR. **Workshop on spatial and spatio-temporal reasoning.** [S.l.], 1994.

GELFAND, A. . E. .; SMITH, A. F. M. Sampling-based approaches to calculating marginal densities. **American statistics association**, v. 85, p. 398–409, 1990.

GEMAN, S.; GEMAN, D. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. **IEEE transactions on pattern analysis and machine intelligence**, Taylor & Francis, v. 6, n. 6, p. 721–741, nov. 1984.

GENESERETH, M.; NILSSON, N. **Logical foundations of artificial intelligence**. San Mateo, CA: Morgan Kaufmann, 1987.

GERBER, R.; NAGEL, H.-H.; SCHREIBER, H. Deriving textual descriptions of road traffic queues from video sequences. In: **ECAI**. [S.l.: s.n.], 2002. p. 736–740.

GREENSPAN, H. et al. Overcomplete steerable pyramid filters and rotation invariance. In: **Computer vision and pattern recognition**. Seattle: [s.n.], 1994. p. 222–228.

GäRDENFORS, P. **Conceptual Spaces: the geometry of thought**. [S.l.]: MIT Pres, 2000. (Bradford Book).

HARALICK, R.; SHANMUGAM, K.; DINSTEIN, I. Texture features for image classification. **IEEE transactions on systems, man, and cybernetics**, v. 3, n. 6, p. 610–621, 1973.

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. In: **Alvey vision conference**. Manchester: [s.n.], 1988. p. 147–151.

HENSEL, I. et al. Understanding object relations in traffic scenes. **VISAPP - International conference on computer vision theory and applications**, Angers, 2010.

HINTON, G. E. Training products of experts by minimizing contrastive divergence. **Neural computation**, Massachusetts, v. 14, n. 8, p. 1771–1800, 2002.

HOLZMANN, F. Needs of improved assistant systems. In: **Adaptive Cooperation between Driver and Assistant System**. [S.l.]: Springer Berlin Heidelberg, 2008. p. 3–10. ISBN 978-3-540-74474-0.

HOUGH, P. V. C. Methods and means for recognizing complex patterns. **US Patent 3069654**, 1962.

JOCHEM, T. M.; POMERLEAU, D. A.; THORPE, C. E. Maniac: A next generation neurally based autonomous road follower. In: **International conference on intelligent autonomous systems**. Pittsburgh: [s.n.], 1993.

KANADE, T. et al. Advances in cooperative multi-sensor video surveillance. In: **Proceedings of DARPA image understanding workshop**. [S.l.: s.n.], 1998.

KAUTZ, H.; SELMAN, B.; JIANG, Y. A general stochastic approach to solving problems with hard and soft constraints. In: **The satisfiability problem: theory and applications**. [S.l.]: American mathematical society, 1996. p. 573–586.

KLETTE, R. Stereo-vision-support for intelligent vehicles - the need for quantified evidence. In: **Proceedings of the 21st australasian joint conference on artificial intelligence**. Berlin: Springer-Verlag, 2008. p. 1–17. ISBN 978-3-540-89377-6.

KOK, S. et al. **The Alchemy system for statistical relational AI.** Washington, DC, 2007. Disponível em: <http://alchemy.cs.washington.edu.>.

KOLLER, D.; PFEFFER, A. Probabilistic frame-based systems. In: **Proceedings of AAAI**. [S.l.]: AAAI press, 1998. p. 580–587.

KSCHISCHANG, F. R.; FREY, B. J.; LOELIGER., H. A. Factor graphs and the sum-product algorithm. **IEEE transactions on information theory**, Zurich, v. 47, p. 498–519, 2001.

LAWS, K. **Textured image segmentation**. Tese (Doutorado) — University of southern California, 1980.

LIGOZAT, G. Reasoning about cardinal directions. **Journal of visual language computing**, [S.l.], v. 9, n. 1, p. 23–44, 1998.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. **International journal of computer vision**, v. 60, n. 2, p. 91, 2004.

LUCAS, B. D.; KANADE, T. An iterative image registration technique with an application to stereo vision. In: HAYES, P. J. (Ed.). **International joint conference on artificial intelligence**. [S.l.]: William Kaufmann, 1981. p. 674–679.

MACHADO, L. **SuperInteligência**. 1. ed. Rio de Janeiro: Quality mark, 2005.

MALLAT, S. G. A theory for multiresolution signal decomposition: the wavelet representation. **IEEE transactions on pattern analysis and machine intelligence**, IEEE computer society, Washington, DC, USA, v. 11, n. 7, p. 674–693, 1989. ISSN 0162-8828.

MANIFESTO, C.-C. Car 2 car communication consortium manifesto. **System**, CAR 2 CAR® Communication Consortium, p. 1–94, 2007.

MATHWORKS. **Lane departure warning system.** 05 2010. Revision: 1.1.6.3. Disponível em: <http://www.mathworks.com/ products/ viprocessing/ demos.html?file=/ products/ demos/ shipping/ vipblks/ vipldws.html#5>.

MCCALL, J. C.; TRIVEDI, M. M. Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. **IEEE transactions on intelligent transportation systems**, v. 7, n. 1, p. 20–37, 2006.

MCDERMOTT. We've been framed: or, why AI is innocent of the frame problem. In: PYLYSHYN, Z. W. (Ed.). **The Robot's Dilemma**. Norwood, NJ.: Ablex, 1987. p. 113–122.

MIKOLAJCZYK, K.; SCHMID, C. A performance evaluation of local descriptors. **IEEE transactions on pattern analysis and machine intelligence**, [S.l], v. 27, n. 10, p. 1615–1630, 2005.

MILCH, B. et al. Blog: probabilistic models with unknown objects. In: **International joint conference on artificial intelligence**. [S.l.: s.n.], 2005. p. 1352–1359.

MILCH, B.; RUSSELL, S. J. First-order probabilistic languages: into the unknown. In: MUGGLETON, S.; OTERO, R. P.; TAMADDONI-NEZHAD, A. (Ed.). **Inductive logic programming**. Berlin: Springer, 2006. (Lecture notes in computer science, v. 4455), p. 10–24. ISBN 978-3-540-73846-6.

MONTEMERLO, M. et al. Junior: the stanford entry in the urban challenge. **Journal of field robotics**, v. 25, n. 9, p. 569–597, 2008.

MORATZ, R.; RENZ, J.; WOLTER, D. Qualitative spatial reasoning about line segments. In: **Proceedings of the 14th european conference on artifical intelligence**. [S.l.]: IOS press, 2000. p. 234–238.

MORGENSTERN, L.; MCILRAITH, S. A. John mccarthy's legacy. **Artificial intelligence**, [S.l.], v. 175, n. 1, p. 1–24, 2011.

NAGEL, H.-H. Analysing sequences of tv-frames. In: **Proceedings of the 5th international joint conference on Artificial intelligence**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1977. v. 2, p. 626–626.

____. Image sequence evaluation: 30 years and still going strong. **International conference on pattern recognition**, IEEE Computer Society, Los Alamitos, CA, USA, v. 1, p. 1149, 2000.

____. Steps toward a cognitive vision system. **AI magazine**, American association for artificial intelligence, Menlo Park, CA, USA, v. 25, n. 2, p. 31–50, 2004. ISSN 0738-4602.

NEDEVSCHI, S. et al. A sensor for urban driving assistance systems based on dense stereovision. In: ____. Vienna: Stereo vision, 2008. (I-Tech), cap. 14, p. 235–258. ISBN 978-953-7619-22-0.

NILSSON, N. J. Probabilistic logic. **Artificial intelligence**, [S.l.], v. 28, n. 1, p. 71–87, 1986.

PARAGIOS, N.; DERICHE, R. Geodesic active contours and level sets for the detection and tracking of moving objects. **IEEE transactions on pattern analysis and machine intelligence**, [S.l.], v. 22, p. 266–280, 2000.

____. Geodesic active regions and level set methods for supervised texture segmentation. **International journal of computer vision**, [S.l.], v. 46, p. 223–247, 2002.

PARKER, A. et al. Probabilistic go theories. In: VELOSO, M. M. (Ed.). **International joint conference on artificial intelligence**. [S.l.: s.n.], 2007. p. 501–506.

PASCHOS, G. Perceptually uniform color spaces for color texture analysis: an empirical evaluation. **IEEE transactions on image processing**, [S.l.], v. 10, n. 6, p. 932–937, 2001.

PEARL, J. Reverend Bayes on inference engines: A distributed hierarchical approach. In: **Proceedings of the american association of artificial intelligence conference**. Pittsburgh, PA: [s.n.], 1982. p. 133–136.

____. **Probabilistic reasoning in intelligent systems: networks of plausible inference**. [S.l.]: Morgan Kaufmann, 1988. ISBN 1558604790.

POON, H.; DOMINGOS, P. Sound and efficient inference with probabilistic and deterministic dependencies. In: **21th National conference on artificial intelligence**. [S.l.]: AAAI press, 2006. p. 458–463.

RANDELL, D.; COHN, A.; CUI, Z. A spatial logic based on regions and connections. In: **3rd International conference on knowledge representation and reasoning**. [S.l.]: Morgan Kaufmann, 1992. p. 165–176.

RANDELL, D. A.; WITKOWSKI, M.; SHANAHAN, M. From images to bodies: modelling and exploiting spatial occlusion and motion parallax. In: NEBEL, B. (Ed.). **International joint conference on artificial intelligence**. [S.l.]: Morgan Kaufmann, 2001. p. 57–66. ISBN 1-55860-777-3.

REITER, R. **Knowledge in action: logical foundations for specifying and implementing dynamical systems**. Massachusetts: The MIT press, 2001. ISBN 0262182181.

RICHARDSON, M.; DOMINGOS, P. Markov logic networks. **Machine learning**, [S.l.], v. 62, n. 1-2, p. 107–136, 2006.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning internal representations by error propagation. **Neurocomputing: foundations of research**, MIT Press, Cambridge, MA, USA, v. 1, p. 673–695, 1988.

RUSSELL, S.; NORVIG, P. **Inteligência Artificial**. 2. ed. [S.l.]: Campus, 2004.

SANTOS, P. **Spatial reasoning and abductive interpretation of sensor data obtained by a mobile robot in a dynamic environment**. Tese (Doutorado) — Imperial college of London, 2003.

SANTOS, P. Reasoning about depth and motion from an observer's viewpoint. **Spatial cognition and computation: an interdisciplinary journal**, v. 7, n. 1542-7633, p. 133–178, 2007.

SANTOS, P. et al. Probabilistic logic encoding of spatial domains. **UniDL - International workshop on uncertainty in description logics**, United Kingdom, v. 1, 2010.

SANTOS, P.; SHANAHAN, M. Hypothesising object relations from image transitions. In: HARMELEN, F. van (Ed.). **European conference on artificial intelligence**. [S.l.]: IOS Press, 2002. p. 292–296.

____. A logic-based algorithm for image sequence interpretation and anchoring. In: GOTTLOB, G.; WALSH, T. (Ed.). **International joint conference on artificial intelligence**. [S.l.]: Morgan Kaufmann, 2003. p. 1408–1411.

SHANAHAN, M. Robotics and the common sense informatic situation. In: WAHLSTER, W. (Ed.). **European conference on artificial intelligence**. [S.l.]: John Wiley and Sons, Chichester, 1996. p. 684–688.

SHAPIRO, L.; STOCKMAN, G. **Computer vision**. [S.l.]: Prentice Hall, 2001. 608 p.

SHI, J.; MALIK, J. Normalized cuts and image segmentation. **IEEE transactions on pattern analysis and machine intelligence**, v. 22, n. 8, p. 888–905, 2000.

SONG, K. Defect detection in random colour textures. **Image and vision computing**, Elsevier, v. 14, n. 9, p. 667–683, 1996.

SOUTCHANSKI, M.; SANTOS, P. Reasoning about dynamic depth profiles. In: GHALLAB, M. et al. (Ed.). **European conference on artificial intelligence**. [S.l.]: IOS press, 2008. (Frontiers in artificial intelligence and applications, v. 178), p. 30–34. ISBN 978-1-58603-891-5.

SOUZA, C. R. C.; SANTOS, P. E. Probabilistic logic reasoning about traffic scenes. In: GROß, R. et al. (Ed.). **Towards autonomous robotic systems**. Berlin: Springer-Verlag, 2011. (Lecture notes in computer science, v. 6856), p. 219–230. ISBN 978-3-642-23231-2.

STEINGRUBE, P.; GEHRIG, S. K.; FRANKE, U. Performance evaluation of stereo algorithms for automotive applications. In: FRITZ, M.; SCHIELE, B.; PIATER, J. H. (Ed.). **International conference in computer vision systems**. [S.l.]: Springer, 2009. (Lecture notes in computer science, v. 5815), p. 285–294. ISBN 978-3-642-04666-7.

STEINHAUS, H. Sur la division des corp materiels en parties. **Bulletin of the polish academy of sciences and mathematics**, v. 1, p. 801–804, 1956.

STOCK, O. **Spatial and temporal reasoning**. Norwell: Kluwer academic publishers, 1997. ISBN 0792346440.

SZELISKI, R.; COUGHLAN, J. Spline-based image registration. **International journal of computer vision**, Kluwer academic publishers, Hingham, MA, USA, v. 22, n. 3, p. 199–218, 1997. ISSN 0920-5691.

TASKAR, B.; ABBEEL, P.; KOLLER, D. Discriminative probabilistic models for relational data. In: DARWICHE, A.; FRIEDMAN, N. (Ed.). **Uncertainty in artificial intelligence**. [S.l.]: Morgan Kaufmann, 2002. p. 485–492. ISBN 1-55860-897-4.

Thorpe , C.; KANADE, T. Vision and navigation for the carnegie mellon navlab. In: **Proceedings of the 1985 DARPA Image Understanding Workshop**. [S.l.: s.n.], 1985. p. 143–152.

THRUN, S. et al. Stanley: The robot that won the darpa grand challenge. **Journal of field robotics**, v. 23, n. 9, p. 661–692, 2006.

TIEU, K.; VIOLA, P. Boosting image retrieval. In: **International journal of computer vision**. [S.l.: s.n.], 2000. p. 228–235.

TOMASI, C.; KANADE, T. **Detection and tracking of point features**. [S.l.: s.n.], 1991.

VALIANT, L. G. The complexity of computing the permanent. **Theoretical computer science**, [S.l.], v. 8, p. 189–201, 1979.

VEENMAN, C. J.; REINDERS, M. J. T.; BACKER, E. Resolving motion correspondence for densely moving points. **IEEE transactions on pattern analysis and machine intelligence**, IEEE computer society, Washington, DC, v. 23, n. 1, p. 54–72, 2001. ISSN 0162-8828.

VIOLA, P.; JONES, M. J.; SNOW, D. Detecting pedestrians using patterns of motion and appearance. In: **International conference on computer vision**. [S.l.: s.n.], 2003. p. 734–741.

WEDEL, A. et al. Realtime depth estimation and obstacle detection from monocular video. In: FRANKE, K. et al. (Ed.). **German association for pattern recognition - symposium**. [S.l.]: Springer, 2006. (Lecture notes in computer science, v. 4174), p. 475–484. ISBN 3-540-44412-2.

YILMAZ, A.; JAVED, O.; SHAH, M. Object tracking: a survey. **ACM computing survey**, ACM Press, New York, v. 38, n. 4, p. 13, 2006. ISSN 360-300.

YILMAZ, A.; LI, X.; SHAH, M. Contour based object tracking with occlusion handling in video acquired using mobile cameras. **IEEE transactions on pattern analysis and machine intelligence**, [S.l.], v. 26, p. 1531–1536, 2004.

APÊNDICE A – MODELO IMPLEMENTADO NO ALCHEMY

Apresentamos a seguir o modelo proposto implementado no software Alchemy antes e após o aprendizado dos pesos (tabelas A.1 e A.2). Podemos observar que no aprendizado, assim como na inferência, cada fórmula é proposicionalizada de modo a gerar uma rede de Markov com as instâncias obtidas e o peso ser aprendido para cada sentença. Vemos também que, após o aprendizado, as fórmulas em lógica de primeira ordem são comentadas (símbolo "//" antes de cada regra), preparando o arquivo para inferência, isto é, eliminando a necessidade de proposicionalizar o modelo novamente.

Tabela A.1: Modelo implementado no Alchemy sem o treinamento.

```
//---> Variables Declaration

status={Ok,Under}
lanepos={Left,Center,Right}
type={WDashed,WContinuous,YDashed,YContinuous,Merge}
way={One,Two}


//---> Predicates Declaration

dividerR(type,status,time)
dividerL(type,status,time)
wrongWay(time)
crossingLeft(time)
crossingRight(time)
roadWay(way!,time)
prohibitedManeuver(time)
emergencyLane(time)


//---> Formulae Declaration

dividerL(YContinuous,Under,t) v dividerR(YContinuous,s,t) => prohibitedManeuver(t) ^ wrongWay(t) ^
roadWay(Two,t) ^ carRelPos(Left,t) ^ !carRelPos(Center,t) ^ !carRelPos(Right,t) ^ !emergencyLane(t)

!dividerL(YContinuous,s,t) v !dividerL(YDashed,s,t) v !dividerR(YContinuous,s,t) v !dividerR(YDashed,s,t) =>
roadWay(One,t) ^ !roadWay(Two,t)

dividerL(YContinuous,Ok,t) v dividerL(YDashed,Ok,t) => roadWay(Two,t) ^ !roadWay(One,t) ^ carRelPos(Left,t)
^ !carRelPos(Center,t) ^ !carRelPos(Right,t) ^ !emergencyLane(t) ^ !crossingLeft(t) ^ !crossingRight(t) ^
!prohibitedManeuver(t) ^ !wrongWay(t)

dividerL(WContinuous,Ok,t) ^ dividerR(WDashed,Ok,t) => roadWay(One,t) ^ !roadWay(Two,t) ^ carRelPos(Left,t)
^ !carRelPos(Center,t) ^ !carRelPos(Right,t) ^ !crossingLeft(t) ^ !crossingRight(t) ^ !prohibitedManeuver(t)
^ !wrongWay(t)

dividerL(WDashed,Ok,t) ^ dividerR(WDashed,Ok,t) => carRelPos(Center,t) ^ !carRelPos(Left,t) ^
!carRelPos(Right,t) ^ !emergencyLane(t) ^ !crossingLeft(t) ^ !crossingRight(t) ^ !prohibitedManeuver(t) ^
!wrongWay(t)

!dividerL(WDashed,Ok,t) ^ dividerR(WDashed,Ok,t) => carRelPos(Left,t) ^ !carRelPos(Center,t) ^
!carRelPos(Right,t) ^ !emergencyLane(t) ^ !crossingLeft(t) ^ !crossingRight(t)

dividerL(YDashed,Under,t) v (dividerR(YDashed,Ok,t) ^ dividerL(ty,s,t)) => wrongWay(t) ^ carRelPos(Left,t) ^
!carRelPos(Center,t) ^ !carRelPos(Right,t) ^ !prohibitedManeuver(t) ^ !emergencyLane(t)

(dividerR(WContinuous,Ok,t) v dividerR(Merge,Ok,t)) ^ dividerL(WDashed,s,t) => carRelPos(Right,t) ^
!carRelPos(Center,t) ^ !carRelPos(Left,t) ^ !crossingLeft(t) ^ !crossingRight(t) ^ !prohibitedManeuver(t) ^
!wrongWay(t)

dividerL(WContinuous,Ok,t) ^ !dividerR(WDashed,Ok,t) => emergencyLane(t) ^ carRelPos(Right,t) ^
!carRelPos(Left,t) ^ !carRelPos(Center,t) ^ !crossingLeft(t) ^ !crossingRight(t) ^ !prohibitedManeuver(t) ^
!wrongWay(t)

dividerR(ty1,Under,t1) v (dividerL(ty1,Under,t2) ^ dividerR(ty2,Ok,t2)) => crossingRight(t) ^ !crossingLeft(t)

dividerL(ty1,Under,t1) v (dividerR(ty1,Under,t2) ^ dividerL(ty2,Ok,t2)) => crossingLeft(t) ^ !crossingRight(t)
```

Fonte: Autor

Tabela A.2: Modelo implementado no Alchemy após o treinamento.

```
//predicate declarations
carRelPos(lanepos!,time)
prohibitedManeuver(time)
wrongWay(time)
crossingLeft(time)
dividerL(type,status,time)
dividerR(type,status,time)
emergencyLane(time)
roadWay(way!,time)
crossingRight(time)

// 15.5852  dividerL(YContinuous,Under,t) v dividerR(YContinuous,s,t) => prohibitedManeuver(t) ^ wrongWay(t)
^ roadWay(Two,t) ^ carRelPos(Left,t) ^ !carRelPos(Center,t) ^ !carRelPos(Right,t) ^ !emergencyLane(t)

-2.34468  !dividerL(YContinuous,Under,a1) v prohibitedManeuver(a1)
3.0999  !dividerL(YContinuous,Under,a1) v wrongWay(a1)
8.77969  !dividerL(YContinuous,Under,a1) v roadWay(Two,a1)
3.49554  !dividerL(YContinuous,Under,a1) v carRelPos(Left,a1)
1.85412  !dividerL(YContinuous,Under,a1) v !carRelPos(Center,a1)
1.62638  !dividerL(YContinuous,Under,a1) v !carRelPos(Right,a1)
4.49477  !dividerL(YContinuous,Under,a1) v !emergencyLane(a1)
-3.41654  !dividerR(YContinuous,a1,a2) v prohibitedManeuver(a2)
-3.76603  !dividerR(YContinuous,a1,a2) v wrongWay(a2)
6.6337  !dividerR(YContinuous,a1,a2) v roadWay(Two,a2)
-1.16932  !dividerR(YContinuous,a1,a2) v carRelPos(Left,a2)
1.24698  !dividerR(YContinuous,a1,a2) v !carRelPos(Center,a2)
-2.72595  !dividerR(YContinuous,a1,a2) v !carRelPos(Right,a2)
-2.2233  !dividerR(YContinuous,a1,a2) v !emergencyLane(a2)

// 2.17189  !dividerL(YContinuous,s,t) v !dividerL(YDashed,s,t) v !dividerR(YContinuous,s,t) v
!dividerR(YDashed,s,t) => roadWay(One,t) ^ !roadWay(Two,t)

-0.335825  dividerL(YContinuous,a1,a2) v roadWay(One,a2)
-0.335825  dividerL(YContinuous,a1,a2) v !roadWay(Two,a2)
3.03069  dividerL(YDashed,a1,a2) v roadWay(One,a2)
3.03069  dividerL(YDashed,a1,a2) v !roadWay(Two,a2)
-0.739161  dividerR(YContinuous,a1,a2) v roadWay(One,a2)
-0.739161  dividerR(YContinuous,a1,a2) v !roadWay(Two,a2)
-0.869761  dividerR(YDashed,a1,a2) v roadWay(One,a2)
-0.869761  dividerR(YDashed,a1,a2) v !roadWay(Two,a2)

// 61.4571  dividerL(YContinuous,Ok,t) v dividerL(YDashed,Ok,t) => roadWay(Two,t) ^ !roadWay(One,t) ^
carRelPos(Left,t) ^ !carRelPos(Center,t) ^ !carRelPos(Right,t) ^ !emergencyLane(t) ^ !crossingLeft(t) ^
!crossingRight(t) ^ !prohibitedManeuver(t) ^ !wrongWay(t)

5.59246  !dividerL(YContinuous,Ok,a1) v roadWay(Two,a1)
5.59246  !dividerL(YContinuous,Ok,a1) v !roadWay(One,a1)
4.71847  !dividerL(YContinuous,Ok,a1) v carRelPos(Left,a1)
2.30295  !dividerL(YContinuous,Ok,a1) v !carRelPos(Center,a1)
2.47454  !dividerL(YContinuous,Ok,a1) v !carRelPos(Right,a1)
0.494641  !dividerL(YContinuous,Ok,a1) v !emergencyLane(a1)
2.30527  !dividerL(YContinuous,Ok,a1) v !crossingLeft(a1)
-0.481789  !dividerL(YContinuous,Ok,a1) v !crossingRight(a1)
5.94404  !dividerL(YContinuous,Ok,a1) v !prohibitedManeuver(a1)
2.74346  !dividerL(YContinuous,Ok,a1) v !wrongWay(a1)
1.21582  !dividerL(YDashed,Ok,a1) v roadWay(Two,a1)
1.21582  !dividerL(YDashed,Ok,a1) v !roadWay(One,a1)
3.53746  !dividerL(YDashed,Ok,a1) v carRelPos(Left,a1)
0.883024  !dividerL(YDashed,Ok,a1) v !carRelPos(Center,a1)
2.64948  !dividerL(YDashed,Ok,a1) v !carRelPos(Right,a1)
4.92289  !dividerL(YDashed,Ok,a1) v !emergencyLane(a1)
3.48811  !dividerL(YDashed,Ok,a1) v !crossingLeft(a1)
3.6667  !dividerL(YDashed,Ok,a1) v !crossingRight(a1)
4.35588  !dividerL(YDashed,Ok,a1) v !prohibitedManeuver(a1)
3.83542  !dividerL(YDashed,Ok,a1) v !wrongWay(a1)

// 20.4789  dividerL(WContinuous,Ok,t) ^ dividerR(WDashed,Ok,t) => roadWay(One,t) ^ !roadWay(Two,t) ^
carRelPos(Left,t) ^ !carRelPos(Center,t) ^ !carRelPos(Right,t) ^ !crossingLeft(t) ^ !crossingRight(t) ^
!prohibitedManeuver(t) ^ !wrongWay(t)
```

Continua

Continuação

```
0.854419  !dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v roadWay(One,a1)
0.854419  !dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v !roadWay(Two,a1)
1.48242  !dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v carRelPos(Left,a1)
3.52041  !dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v !carRelPos(Center,a1)
-0.701269  !dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v !carRelPos(Right,a1)
4.58165  !dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v !crossingLeft(a1)
0.85916  !dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v !crossingRight(a1)
4.36847  !dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v !prohibitedManeuver(a1)
4.65921  !dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v !wrongWay(a1)

// 32.3628  dividerL(WDashed,Ok,t) ^ dividerR(WDashed,Ok,t) => carRelPos(Center,t) ^ !carRelPos(Left,t) ^
!carRelPos(Right,t) ^ !emergencyLane(t) ^ !crossingLeft(t) ^ !crossingRight(t) ^ !prohibitedManeuver(t) ^
!wrongWay(t)

1.78113  !dividerR(WDashed,Ok,a1) v !dividerL(WDashed,Ok,a1) v carRelPos(Center,a1)
4.45742  !dividerR(WDashed,Ok,a1) v !dividerL(WDashed,Ok,a1) v !carRelPos(Left,a1)
0.0776864  !dividerR(WDashed,Ok,a1) v !dividerL(WDashed,Ok,a1) v !carRelPos(Right,a1)
5.88789  !dividerR(WDashed,Ok,a1) v !dividerL(WDashed,Ok,a1) v !emergencyLane(a1)
4.45427  !dividerR(WDashed,Ok,a1) v !dividerL(WDashed,Ok,a1) v !crossingLeft(a1)
3.87695  !dividerR(WDashed,Ok,a1) v !dividerL(WDashed,Ok,a1) v !crossingRight(a1)
6.0046  !dividerR(WDashed,Ok,a1) v !dividerL(WDashed,Ok,a1) v !prohibitedManeuver(a1)
5.82291  !dividerR(WDashed,Ok,a1) v !dividerL(WDashed,Ok,a1) v !wrongWay(a1)

// 4.56344  !dividerL(WDashed,Ok,t) ^ dividerR(WDashed,Ok,t) => carRelPos(Left,t) ^ !carRelPos(Center,t) ^
!carRelPos(Right,t) ^ !emergencyLane(t) ^ !crossingLeft(t) ^ !crossingRight(t)

-0.649773  !dividerR(WDashed,Ok,a1) v dividerL(WDashed,Ok,a1) v carRelPos(Left,a1)
-0.914103  !dividerR(WDashed,Ok,a1) v dividerL(WDashed,Ok,a1) v !carRelPos(Center,a1)
0.319406  !dividerR(WDashed,Ok,a1) v dividerL(WDashed,Ok,a1) v !carRelPos(Right,a1)
1.3715  !dividerR(WDashed,Ok,a1) v dividerL(WDashed,Ok,a1) v !emergencyLane(a1)
-0.45486  !dividerR(WDashed,Ok,a1) v dividerL(WDashed,Ok,a1) v !crossingLeft(a1)
4.89127  !dividerR(WDashed,Ok,a1) v dividerL(WDashed,Ok,a1) v !crossingRight(a1)

// 17.638  dividerL(YDashed,Under,t) v (dividerR(YDashed,Ok,t) ^ dividerL(ty,s,t)) => wrongWay(t) ^
carRelPos(Left,t) ^ !carRelPos(Center,t) ^ !carRelPos(Right,t) ^ !prohibitedManeuver(t) ^ !emergencyLane(t)

1.61986  !dividerL(YDashed,Under,a1) v wrongWay(a1)
3.28833  !dividerL(YDashed,Under,a1) v carRelPos(Left,a1)
1.67281  !dividerL(YDashed,Under,a1) v !carRelPos(Center,a1)
1.47225  !dividerL(YDashed,Under,a1) v !carRelPos(Right,a1)
4.37722  !dividerL(YDashed,Under,a1) v !prohibitedManeuver(a1)
3.82754  !dividerL(YDashed,Under,a1) v !emergencyLane(a1)
-0.948354  !dividerR(YDashed,Ok,a1) v !dividerL(a2,a3,a1) v wrongWay(a1)
0.0402027  !dividerR(YDashed,Ok,a1) v !dividerL(a2,a3,a1) v carRelPos(Left,a1)
0.0199024  !dividerR(YDashed,Ok,a1) v !dividerL(a2,a3,a1) v !carRelPos(Center,a1)
0.0200331  !dividerR(YDashed,Ok,a1) v !dividerL(a2,a3,a1) v !carRelPos(Right,a1)
0.208473  !dividerR(YDashed,Ok,a1) v !dividerL(a2,a3,a1) v !prohibitedManeuver(a1)
2.03972  !dividerR(YDashed,Ok,a1) v !dividerL(a2,a3,a1) v !emergencyLane(a1)

// 40.1161  (dividerR(WContinuous,Ok,t) v dividerR(Merge,Ok,t)) ^ dividerL(WDashed,s,t) => carRelPos(Right,t)
^ !carRelPos(Center,t) ^ !carRelPos(Left,t) ^ !crossingLeft(t) ^ !crossingRight(t) ^ !prohibitedManeuver(t) ^
!wrongWay(t)

3.54371  !dividerR(WContinuous,Ok,a1) v !dividerL(WDashed,a2,a1) v carRelPos(Right,a1)
1.62326  !dividerR(WContinuous,Ok,a1) v !dividerL(WDashed,a2,a1) v !carRelPos(Center,a1)
1.80597  !dividerR(WContinuous,Ok,a1) v !dividerL(WDashed,a2,a1) v !carRelPos(Left,a1)
1.18784  !dividerR(WContinuous,Ok,a1) v !dividerL(WDashed,a2,a1) v !crossingLeft(a1)
3.93892  !dividerR(WContinuous,Ok,a1) v !dividerL(WDashed,a2,a1) v !crossingRight(a1)
4.77959  !dividerR(WContinuous,Ok,a1) v !dividerL(WDashed,a2,a1) v !prohibitedManeuver(a1)
4.6523  !dividerR(WContinuous,Ok,a1) v !dividerL(WDashed,a2,a1) v !wrongWay(a1)
2.466  !dividerR(Merge,Ok,a1) v !dividerL(WDashed,a2,a1) v carRelPos(Right,a1)
0.284307  !dividerR(Merge,Ok,a1) v !dividerL(WDashed,a2,a1) v !carRelPos(Center,a1)
2.61666  !dividerR(Merge,Ok,a1) v !dividerL(WDashed,a2,a1) v !carRelPos(Left,a1)
0.237704  !dividerR(Merge,Ok,a1) v !dividerL(WDashed,a2,a1) v !crossingLeft(a1)
3.92913  !dividerR(Merge,Ok,a1) v !dividerL(WDashed,a2,a1) v !crossingRight(a1)
4.48398  !dividerR(Merge,Ok,a1) v !dividerL(WDashed,a2,a1) v !prohibitedManeuver(a1)
4.56672  !dividerR(Merge,Ok,a1) v !dividerL(WDashed,a2,a1) v !wrongWay(a1)

// 15.813  dividerL(WContinuous,Ok,t) ^ !dividerR(WDashed,Ok,t) => emergencyLane(t) ^ carRelPos(Right,t) ^
!carRelPos(Left,t) ^ !carRelPos(Center,t) ^ !crossingLeft(t) ^ !crossingRight(t) ^ !prohibitedManeuver(t) ^
!wrongWay(t)
```

Continua

Conclusão

```
-0.174849  dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v emergencyLane(a1)
0.949927  dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v carRelPos(Right,a1)
-0.581028  dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v !carRelPos(Left,a1)
2.80827  dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v !carRelPos(Center,a1)
2.89888  dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v !crossingLeft(a1)
0.622253  dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v !crossingRight(a1)
4.80898  dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v !prohibitedManeuver(a1)
4.4806  dividerR(WDashed,Ok,a1) v !dividerL(WContinuous,Ok,a1) v !wrongWay(a1)

// -0.000385247  dividerR(ty1,Under,t1) v (dividerL(ty1,Under,t2) ^ dividerR(ty2,Ok,t2)) => crossingRight(t)
^ !crossingLeft(t)

-0.000346279  !dividerR(a1,Under,a2) v crossingRight(a3)
0.000162533  !dividerR(a1,Under,a2) v !crossingLeft(a3)
-0.000379739  !dividerR(a1,Ok,a2) v !dividerL(a3,Under,a2) v crossingRight(a4)
0.000178238  !dividerR(a1,Ok,a2) v !dividerL(a3,Under,a2) v !crossingLeft(a4)

// 0.000385247  dividerL(ty1,Under,t1) v (dividerR(ty1,Under,t2) ^ dividerL(ty2,Ok,t2)) => crossingLeft(t)
^ !crossingRight(t)

-0.000178238  !dividerL(a1,Under,a2) v crossingLeft(a3)
0.000379739  !dividerL(a1,Under,a2) v !crossingRight(a3)
-0.000162533  !dividerR(a1,Under,a2) v !dividerL(a3,Ok,a2) v crossingLeft(a4)
0.000346279  !dividerR(a1,Under,a2) v !dividerL(a3,Ok,a2) v !crossingRight(a4)
```

Fonte: Autor

APÊNDICE B – ARTIGO PUBLICADO

Durante este trabalho, publicamos o artigo (SOUZA; SANTOS, 2011) o qual anexamos neste apêndice.