Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Informática

Uma Abordagem Baseada em Regras para Detecção Automática de Eventos Pré-Definidos em Vídeos

Francisco Fabian de Macedo Almeida

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciências da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Modelos Computacionais e Cognitivos

Herman Martins Gomes
(Orientador)

Campina Grande, Paraíba, Brasil

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA CENTRAL DA UFCG

A447a Almeida, Francisco Fabian de Macedo

Uma abordagem baseada em regras para detecção automática de eventos pré-definidos em vídeos / Francisco Fabian de Macedo Almeida. — Campina Grande, 2010.

115 f : il. color.

Dissertação (Mestrado em Ciências da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática.

Referências.

Orientador: Prof. Dr. Herman Martins Gomes.

1. Inteligência Artificial 2. Sistemas Baseados em Regras 3. Visão Computacional 4. Vigilância Automática 5. Eventos em Vídeo I. Título.

CDU 004.891 (043)

"UMA ABORDAGEM BASEADA EM REGRAS PARA DETECÇÃO AUTOMÁTICA DE EVENTOS PRÉ-DEFINIDOS EM VÍDEOS"

FRANCISCO FABIAN DE MACEDO ALMEIDA

DISSERTAÇÃO APROVADA EM 07.05.2010

HERMAN MARTINS GOMES, Ph.D
Orientador(a)

JOSEANA MACÊDO FECHINE, D.Sc
Examinador(a)

ARISTÓFANES CORRÊA SILVA, Dr.
Examinador(a)

CAMPINA GRANDE - PB

Resumo

O objetivo central deste trabalho é desenvolver um sistema para detecção automática de eventos pré-definidos em vídeos. O monitoramento automático de vídeos é um tópico de pesquisa relevante tendo em vista o grande número de aplicações em que é inviável ter-se um monitoramento humano contínuo. Como exemplos, podem-se citar situações de tráfego viário, cenários de controle de acesso a pessoas ou veículos, desembarque em aeroportos, dentre outros. O sistema proposto é composto por uma arquitetura para detectar eventos que envolvam interações entre objetos móveis e entre esses objetos e elementos contextuais da cena, representados como formas 2D. Uma característica diferencial do sistema é a modelagem dos eventos e cenários a partir de uma especificação lógica utilizando regras e fatos em CLIPS - um ambiente para construção de sistemas especialistas. Módulos de baixo-nível da arquitetura utilizam algoritmos de visão computacional como subtração de fundo de vídeo e de rastreio para atualizar base de fatos de interesse, enquanto que a máquina de inferência de regras do CLIPS utiliza essas informações para detectar os eventos de alto-nível pré-definidos. O sistema é ilustrado com casos de cruzamento de trânsito, envolvendo situações de violações de regras de trânsito entre automóveis e pedestres, incluindo violações de sinais de semáforos. Foram realizados testes com vídeos *off-line* e os resultados da detecção de eventos do sistema são comparados com conjuntos-verdade de vídeos anotados por humanos, com resultados promissores.

Abstract

The main objective of this work is to develop a system for automatic detection of predefined events in videos. The automatic monitoring of videos is a relevant research topic when considering the large number of applications where it is impossible to have continuous human monitoring. As examples, it can be cited road traffic situations, scenarios involving access control to people or vehicles, arrival area at airports, among others. The proposed system is composed of an architecture designed to detect events that involve interactions between moving objects and between these objects and contextual elements in the scene, represented as 2D shapes. A distinctive feature of the system is the modeling of events and scenarios through a logic specification using rules and facts in CLIPS - an environment for building expert systems. Low-level modules of the architecture uses computer vision algorithms such as video background subtraction and tracking to update database of facts of interest, while the inference engine of CLIPS uses this information to detect predefined high-level events. The system is illustrated with cases of crossroad traffic situations involving violations of traffic rules between cars, pedestrians, including violations of signal lights of semaphores. Tests were conducted with videos processed off-line and the results of the event detection system are compared with ground truth manually annotated, producing promising results.

Agradecimentos

Agradeço a Ele por fornecer energia, chance, vida e alimento. Especiais agradecimentos aos meus pais, Gorete e Chico de Lola pelo apoio e entusiasmo e por acreditar nos estudos como forma de elevação cultural. Ao meu orientador Herman por me conceder oportunidades de crescimento profissional. Obrigado pela orientação e por sua paciência comigo. Ao meu padrinho Heleno pelo incentivo. Aos amigos do Laboratório de Visão Computacional: Bruno Alexandre, Saulo, FHC, Bosco, Cláudio, Xico e Eanes. A Elmano pelas conversas e exemplo de organização. À Aninha da Copin, sempre cuidadosa quando eu mais precisava. Aos amigos que convivi neste período, Djair, Glauber, Sonali, Cláudia, Anne, Elis, Emily, Walber, Walker e Tayrone. A minha tia Netinha pelas ajudas, me concedendo tempo para me concentrar nos estudos. Ao meu irmão Gian e sua esposa Karina pelas cobranças para terminar. E à minha irmã Gilianne e ao meu primo quase irmão Dário por sempre quebrar os galhos nos momentos certos.

Dedico a minha futura filha Ísis.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Códigos Fonte

Capítulo 1

Introdução

A análise automática de vídeos é um dos principais temas de pesquisa em Visão Computacional, considerando as suas relevantes aplicações e desafios envolvidos. Um dos conceitos da análise automática de vídeos é a identificação de eventos que ocorrem na cena, desde eventos simples como entrada de uma pessoa na cena, quanto um evento de alto nível envolvendo múltiplas pessoas. Como um subtema da análise automática de vídeos, o monitoramento automático de vídeos, em especial a detecção de eventos pré-definidos, é um tópico de pesquisa de destaque, tendo em vista o grande número de problemas em que é inviável ter-se um monitoramento humano contínuo.

A detecção de eventos pré-definidos em vídeos consiste na identificação de conteúdos semânticos que sejam de interesse. Esta é uma tarefa que envolve a interpretação de ações em alto nível da cena e, portanto, necessita de várias etapas intermediárias. De forma a preencher algumas lacunas de pesquisas que serão vistas em seções a seguir, esta dissertação tratará do estudo e desenvolvimento de técnicas para detecção de eventos pré-definidos em vídeo.

O restante desta introdução irá abordar exemplos de aplicações que envolvem análise automática de vídeos. Um detalhamento do problema de detecção de eventos e revisão bibliográfica dos principais artigos sobre detecção de eventos serão abordados no Capítulo 2.

1.1 Motivação

A interpretação semântica do conteúdo de vídeos é um tema que tem despertado interesse de vários grupos de pesquisa e desencadeado o desenvolvimento de grandes projetos. O projeto W4 [Haritaoglu et al., 2000] é uma implementação completa de um sistema de vigilância, o qual emprega técnicas de análise de forma e rastreio para identificação de ações de pessoas.

O DARPA (Defense Advanced Projects Agency, dos EUA) financiou o projeto *Visual Surveillance and Monitoring* (VSAM) [Davis et al., 1997], o qual se propunha a desenvolver técnicas automáticas para monitoramento de áreas de combate. A União Europeia patrocinou o projeto Advisor, que tinha como objetivo o monitoramento de estações de metrô. Como parte do projeto Advisor, se destaca o desenvolvimento do *Reading People Tracker* por Siebel [Siebel, 2003], consistindo num rastreador de pessoas, cujo código-fonte encontra-se livremente disponível na Web [1]. O projeto BESAFE (*Behavior Learning in Surveilled Areas with Feature Extraction*), fundado pelo programa da OTAN *Science for Peace and Security*, se propõe a desenvolver técnicas para identificar pessoas, detectar comportamentos anormais e prever ações perigosas em tempo real.

Em atividades que exijam segurança, como bancos, lojas comerciais, entrada em locais protegidos, locais públicos, estradas etc., as câmeras de vigilância normalmente captam as cenas e as gravam numa mídia, que pode ser analógica ou digital. Caso analógica, há a necessidade de codificação para meio digital antes que um processamento automático possa ser executado. Técnicas de vigilância automática podem ser utilizadas para identificar pessoas e veículos, detectar comportamentos anormais como congestionamento, agressões, furtos e colisões. As cenas podem ser captadas em ambientes externos, como praças ou vias públicas, ou internos, como escritórios e lojas. A análise de eventos pode também ser utilizada para detectar comportamentos anormais em multidões, como torcidas em conflito em estádios ou tumultos em shows.

Todos os projetos e aplicações citados anteriormente evidenciam o crescente interesse pelo desenvolvimento de pesquisas na área de análise automática de vídeos, mais especificamente na área de vigilância. Além desses projetos, diversos artigos foram publicados em jornais especializados como o IJCV (*International Journal of Computer Vision*), CVIU

[1] http://www.siebel-research.de/people_tracking/

(*Computer Vision and Image Understanding*), IVC (*Image and Vision Computing*) e também em congressos e conferências como o ICCV (*International Conference on Computer Vision*), ECCV (*European Conference on Computer Vision*) e IWVS (*IEE International Workshop on Visual Surveillance*). Discussões sobre uma seleção destes trabalhos são apresentadas nas próximas seções.

1.2 Descrição do problema

O presente trabalho abordará, mais especificamente, a detecção de eventos (ou atividades) pré-definidos por um especialista humano. Um especialista é aquele que conhece os eventos de interesse no contexto de um cenário e que sabe descrevê-los numa linguagem. Neste trabalho, a linguagem será baseada em regras lógicas. Uma avaliação da abordagem implementada foi desenvolvida envolvendo um cenário de cruzamento de trânsito.

1.3 Objetivos gerais

O objetivo principal deste trabalho é desenvolver um sistema para detecção de eventos pré-definidos em vídeos, onde um especialista possa definir em regras os eventos de interesse e em que o sistema gere alarmes quando da detecção dos eventos. Sistemas de vigilância automática serão os cenários alvo das situações em que o sistema se propõe a detectar, onde elementos móveis (*blobs*) interagem com elementos do cenário. As situações que o sistema poderá abordar incluem pessoas ou veículos em movimento, modeladas, por restrições de escopo, como formas 2D.

1.4 Objetivos específicos

Os objetivos específicos desta dissertação são a detecção de eventos de interesse num cenário de cruzamento de trânsito. Para tal, foi desenvolvida uma arquitetura em módulos que permite reutilizar soluções existentes para pré-processamento de vídeo. Nesta dissertação, foi proposta uma hierarquização de eventos em vídeo como atividades simples e atividades compostas. Um especialista humano deve facilmente especificar, utilizando uma linguagem

lógica, os eventos de interesse. Para isto, foi estendida uma linguagem baseada em regras para que um especialista possa especificar os eventos de interesse. Para efeito de restrição de escopo, considerou-se uma câmera de vídeo estática para captura das cenas, e os objetos de interesse são tratados como formas 2D.

1.5 Contribuições

A principal contribuição desta dissertação foi fornecer métodos para um especialista poder especificar os eventos de interesse em um vídeo, através de uma sugestão de representação e detecção de regras para eventos pré-definidos em vídeo. Uma das contribuições foi estender uma linguagem baseada em regras, adicionando raciocínio temporal para a construção de regras de eventos pré-definidos. Além da forma de utilização da linguagem baseada em regras do ambiente CLIPS para a tarefa de especificação e detecção de eventos, este trabalho sugeriu a decomposição de atividades compostas em atividades simples seguindo uma ordem temporal. Para a detecção das regras, este trabalho propôs a análise de regras simples prioritariamente em relação à análise de regras para atividades compostas, construindo um histórico de atividades simples que é posteriormente utilizado para a verificação temporal das relações contidas nas regras de atividades compostas.

Outra contribuição desta dissertação é a arquitetura modular para o problema de detecção de eventos pré-definidos em vídeo digital. Além disto, foi realizada uma avaliação da abordagem implementada no contexto de um cruzamento de trânsito, incluindo todo o conhecimento gerado na especificação das regras e análise do problema.

1.6 Estrutura da dissertação

- O Capítulo 2, Fundamentação, contém um detalhamento do problema, das teorias envolvidas e das abordagens adotadas por trabalhos relacionados.
- No Capítulo 3, Modelo Proposto, a solução proposta é descrita, o que inclui a arquitetura do sistema e as técnicas utilizadas na implementação.
- O Capítulo 4, Avaliação da Abordagem Implementada, objetiva validar o modelo e

obter conclusões a partir de testes sobre uma coleção de vídeos digitais.

- As conclusões e sugestões de trabalhos futuros estão descritas no Capítulo 5.

Capítulo 2

Fundamentação

A interpretação automática de vídeos pode ser vista como o processo de compreender automaticamente o que acontece nas cenas retratadas pelos quadros de vídeos, e de fornecer uma interpretação do que acontece nestas. Um sistema de interpretação automática de vídeos terá como entrada quadros de vídeos e como saída a interpretação das cenas.

Sistemas para análise de comportamento em vídeo, a exemplo dos sistemas de monitoramento automáticos, têm como objetivo geral a detecção de eventos que acontecem no ambiente monitorado. Evento em vídeo é um conceito de alto nível, que envolve a análise dos objetos em movimento e sua interação com demais elementos da cena. Estes sistemas têm em comum objetos de interesse que se movimentam no ambiente observado. Esses objetos precisam ser detectados, classificados e rastreados para que se possa, em seguida, interpretar os eventos que ocorrem no ambiente. Então, no campo da Visão Computacional, a tarefa de detectar eventos constitui um dos últimos objetivos no processamento de vídeo, precisando haver várias etapas anteriores para sua detecção. Desta maneira, diversas etapas são necessárias para o processamento como a detecção de movimento, a classificação dos objetos, o rastreamento e a detecção do evento propriamente dito. De uma maneira geral, esses sistemas são implementados seguindo uma estrutura de módulos encadeados, em que cada módulo objetiva solucionar uma etapa do processamento, com a saída de um módulo constituindo a entrada do próximo módulo, como ilustrado na Figura 2.1. Neste capítulo, são discutidos trabalhos relevantes sobre as diversas etapas envolvidas até se chegar à detecção de eventos de mais alto nível, como também são relacionados trabalhos na área de interpretação de eventos.

O presente trabalho de dissertação objetiva a detecção de eventos de interesse num cruzamento de trânsito, como exemplos a violação de sinal, entradas ilegais em avenidas, estacionamento ilegal, abordagens de pedestres em carros, etc. O trabalho se concentra na etapa de detecção de eventos, porém serão também revisadas as etapas anteriores de processamento.

A primeira etapa necessária na interpretação de alto nível de um vídeo é a detecção dos objetos que participam da cena. Na Seção 2.1, há uma descrição desta etapa inicial. Depois de detectados, opcionalmente poderá feita uma classificação destes objetos. Uma discussão sobre esta etapa encontra-se na Seção 2.2. De posse dos objetos detectados, o passo seguinte será rastreá-los ao longo do vídeo. Alguns métodos de rastreio são discutidos na Seção 2.3. A etapa final consiste em detectar os eventos com base na classificação e trajetória dos objetos, e na interação de objetos com os demais elementos da cena (como elementos estáticos ou outros objetos móveis). Esta etapa de mais alto nível de detecção de eventos é discutida na Seção 2.4.

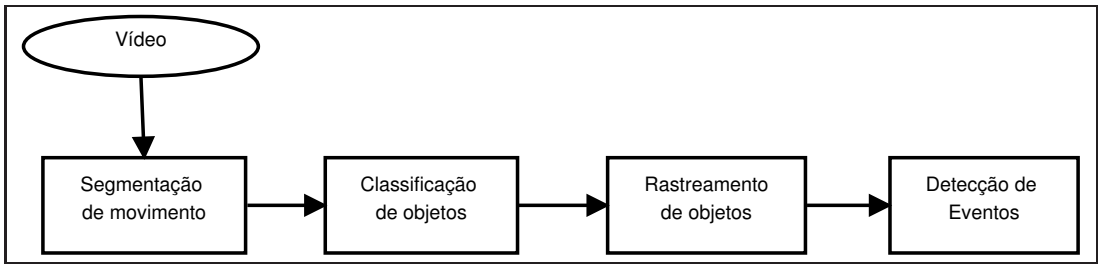


Figura 2.1: Sequência de etapas no processo de análise automática de vídeos.

2.1 Segmentação de movimento

A partir de uma fonte de vídeo, que poderá ser proveniente de uma câmera em tempo-real ou de um arquivo multimídia, normalmente são enviados quadros de vídeo para o módulo de segmentação de movimento. Este módulo objetiva classificar os *pixels* em objeto (*foreground*) ou em fundo da cena (*background*). Como resultado, são geradas imagens binárias destacando objetos (ou *blobs*) em movimento ou parte estática da cena. De acordo com da Costa [da Costa, 2008], alguns dos desafios deste módulo são problemas no vídeo como mudanças climáticas, variações na iluminação do ambiente, sombras, superfícies espelhadas, objetos com movimentos oscilatórios com baixa amplitude (como galhos e folhas de

vegetação sob efeito de ventos), dentre outros. Desta maneira, neste módulo poderão ser necessárias etapas auxiliares, como remoção de ruído e sombra.

Uma revisão de alguns métodos de segmentação de movimento pode ser encontrada no trabalho de Hu et al. [Hu et al., 2004]. Entre os métodos mais conhecidos estão a técnica de diferenciação temporal, técnica baseada em fluxo ótico e a técnica de subtração do fundo da cena. Uma das técnicas mais simples utilizadas é a técnica de diferenciação temporal. Esta utiliza o fato de que dois quadros de imagem subsequentes contêm regiões distintas onde se localizam os objetos móveis. Lipton et al. [Lipton et al., 1998], por exemplo, computa a diferença entre quadros sucessivos e detecta movimento, quando essa diferença ultrapassa um limiar predefinido.

Movimento em vídeos também pode ser detectado a partir do fluxo óptico. Nesta técnica, são computados os vetores de movimento, contendo direção e intensidade como, por exemplo, o trabalho de Giachetti et al. [Giachetti et al., 1998]. Smith e Brady [Smith and Brady, 1995] agrupam vetores em regiões para assim detectar o movimento.

Outra técnica comumente utilizada é a subtração de fundo da cena. Esta técnica busca manter um modelo de fundo de cena, representando o ambiente estático. Neste caso, os pixels poderão ser classificados como pixels de primeiro plano se a diferença entre o modelo de fundo de cena e o quadro atual em certas regiões da imagem excede um limiar predefinido. Os algoritmos que aplicam esta técnica têm precisão dependente da modelagem do fundo de cena. Haritaoglu et al. em [Haritaoglu et al., 1998] estima o modelo de fundo treinando quadros do vídeo onde não há objetos visíveis em movimento na cena. Neste caso, porém, o modelo de fundo se torna obsoleto na medida em que ocorrem mudanças na cena, não se adaptando por exemplo a uma possível mudança na iluminação da cena. Dessa forma, esta abordagem não é muito utilizada na prática. Tentando minimizar este problema, as abordagens de Stauffer e Grimson [Stauffer and Grimson, 2000] e do sistema de rastreio Pfinder descrito por Wren et al. [Wren et al., 1997] utilizam técnicas estatísticas para manter a variação do modelo de fundo da cena. Na abordagem de Wren, assume-se a distribuição de cor dos pixels do modelo de fundo de cena como uma distribuição Gaussiana. A cada quadro, a média de intensidade desses pixels e o desvio padrão da gaussiana são sempre atualizados. Já o método de Stauffer é mais complexo, utilizando uma mistura de funções gaussianas para modelar cada pixel. Já Li et al., no artigo [Li et al., 2003], propõem uma técnica que computa

um modelo estatístico das diferenças entre o quadro de vídeo atual e o quadro de referência do fundo da cena, a partir de um filtro chamado *Infinite Impulse Response*. O algoritmo é genérico, ou seja, não necessita de nenhuma especificação *a priori* do ambiente. Esta técnica foi utilizada na arquitetura proposta nesta dissertação.

Os resultados desta etapa de segmentação de objetos, os objetos em movimento (ou *blobs*), serão utilizados pela etapa opcional de classificação do tipo de objeto.

2.2 Classificação de objetos

Prosseguindo com as etapas de processamento, o módulo de classificação de objetos utiliza os objetos móveis, chamados de *blobs*, obtidos na etapa anterior de segmentação. Esta etapa objetiva determinar a qual tipo de objeto pertence o *blob*. Por exemplo, normalmente um sistema de monitoramento necessita determinar a classificação entre diferentes tipos de objeto, como pessoa, veículo, animal etc. a fim de interpretar melhor o que está acontecendo no ambiente. Este módulo é opcional pois é possível que existam aplicações em que não há interesse na diferenciação entre os diferentes tipos de objetos que podem ser encontrados numa cena.

No contexto de detecção de movimento em vídeo, a maioria dos trabalhos em classificação de objetos tem como objetivo separá-los em pessoas, grupos de pessoas, veículos ou pessoas carregando objetos ([Lipton et al., 1998], [Collins et al., 1999], [Dedeoglu et al., 2006] etc.). De uma forma geral, a classificação é feita pela extração de características dos objetos alvo. A escolha de tais características é fundamental para o sucesso da classificação. Devem-se escolher características representativas de cada classe de objetos.

De acordo com Hu et al. [Hu et al., 2004], existem dois grupos de características comumente utilizadas para a etapa de classificação: as baseadas na forma do objeto (*shape based features*) e aquelas que se baseiam no movimento dos objetos (*motion based features*). Métodos de classificação baseados em movimento fazem uso do fato de que o movimento humano é periódico em contraste com o movimento rígido de objetos como veículos.

Para métodos baseados na forma do objeto, muitos atributos do objeto podem ser utilizados. Lipton et al. [Lipton et al., 1998], por exemplo, utilizam dispersão dos blobs para distinguir entre humanos e veículos. O sistema de Lipton et al. extrai o atributo de dispersão

e utiliza regras simples para classificação de cada objeto. Collins et al. [Collins et al., 1999] utilizaram, além dos atributos de dispersão, a área e a relação de aspecto do blob, para treinar um classificador neural, cujo objetivo era distinguir entre as classes "humano simples", "múltiplos humanos" e "veículos".

Dedeoglu et al. [Dedeoglu et al., 2006] utilizam a silhueta do objeto como atributo, e com uma técnica de *template matching* (com exemplos de imagens das classes para treinamento) classifica entre diversas classes de objeto. A frequência de tempo é usada no trabalho de Cutler e Davis [Cutler and Davis, 2000] para analisar a repetição de movimento e então classificar o objeto. Neste trabalho, por sua simplicidade e resultados práticos, foi utilizada a técnica de Dedeoglu, com as classes humanos, grupo de humanos e veículos.

2.3 Rastreamento de objetos

A partir dos objetos detectados e classificados na etapa anterior, segue-se uma etapa de rastreamento. O rastreamento pode ser visto como o problema de encontrar as coordenadas dos objetos enquanto estes se movem na cena, atribuindo um identificador a cada *blob* em movimento. Segundo Hakeem [Hakeem, 2007], dentre os desafios encontrados para se realizar o rastreio estão oclusão de objetos, tamanho muito pequeno dos objetos, formas rígidas ou articuladas dos objetos, movimentos bruscos ou não-lineares.

Uma revisão detalhada de métodos de rastreio de objetos é descutida no artigo de Yilmaz et al. [Yilmaz et al., 2006]. O algoritmo *MeanShift* descrito em Comaniciu et al. [Comaniciu and Meer, 1999] é uma abordagem que começa com a posição anterior do objeto, e computa a distância entre o objeto e um candidato ser o objeto. Este algoritmo busca achar o melhor entre os candidatos a objeto. Uma abordagem probabilística, como filtro de partículas [Isard and Blake, 1998] é baseada na teoria bayesiana. Esta dissertação baseia-se na técnica Camshift [Allen et al., 2004] derivada da técnica MeanShift [Comaniciu and Meer, 1999], pois permite que o fundo da cena mude abrupdamente, sendo ideal em vídeos de ambientes externos. Além disto, segundo Qiu [Qiu and Lu, 2009], o Camshift permite uma computação eficiente sendo apropriado a vídeos em tempo-real.

De posse do rastreio e da classificação dos objetos, em seguida é realizada a etapa de detecção de eventos.

2.4 Detecção de eventos

A etapa final de um sistema típico de monitoramento automático a partir de vídeos é a detecção de eventos, consistindo em analisar e interpretar os resultados dos módulos de baixo nível, mais especificamente o rastreio e a classificação. O objetivo do módulo de detecção de eventos é interpretar o comportamento dos elementos observados da cena, podendo gerar uma descrição dos eventos que ocorrem no vídeo ou gerar alarmes quando uma situação específica é detectada.

Existe um grande número de pesquisas descritas na literatura especializada com diferentes abordagens para os problemas de detecção de eventos. Esta seção não tem por finalidade citar todos os trabalhos nesta área, mas apenas para dar uma visão geral nas pesquisas de alguns trabalhos relevantes. Revisões mais completas sobre os trabalhos nesta área foram realizadas nos trabalhos de Moeslund et al. [Moeslund et al., 2006], Wang et al. [Wang et al., 2003] e Hu et al. [Hu et al., 2004]. Segundo Moeslund at al., apesar de já existirem muitos trabalhos no campo de representação e reconhecimento de eventos em vídeo, este ainda é um campo de pesquisa relativamente imaturo. Ainda segundo o autor, ainda não há um consenso na literatura sobre a terminologia pois alguns termos são indiferentemente utilizados como ações, atividades, eventos e comportamentos.

Na literatura, de acordo com Hakeem [Hakeem, 2007], várias abordagens foram propostas para a detecção de eventos em vídeos. A maior parte pode ser agrupada em três categorias. A primeira delas é a abordagem de eventos pré-definidos, com modelos de eventos na forma de *templates*, regras ou restrições. Normalmente, estes modelos são definidos à mão por um especialista. Uma segunda categoria é a aprendizagem de eventos, usando dados para treinamento para criar modelos de eventos. A terceira categoria é a de agrupamento, em que se dispensa a modelagem de eventos, sendo utilizadas técnicas de agrupamento para a detecção dos eventos. A seguir são apresentados mais detalhes sobre cada uma destas categorias.

2.4.1 Aprendizagem de eventos

Na aprendizagem de eventos, também chamada de aprendizagem supervisionada, para realizar a interpretação de alto nível da cena, se faz a construção de um modelo de eventos a partir de um conjunto de treinamento [Hakeem, 2007]. A detecção de eventos de interesse

se dá a partir da comparação do evento em questão com o modelo, usando uma métrica de similaridade apropriada, e observando o grau de anomalia ou discrepância entre os dados da cena e o modelo.

Geralmente, nas aplicações de análise de movimento que utilizam a técnica de aprendizagem de eventos, a trajetória do movimento é considerada como o evento de interesse. Uma das características mais utilizadas na análise de eventos simples é a trajetória, como atestam os diversos trabalhos sobre agrupamento de trajetórias para detecção de anomalias [Johnson and Hogg, 1995; Stauffer and Grimson, 2000]. Normalmente, a trajetória é gerada por um módulo chamado rastreador que, em muitas aplicações, é utilizado para rastreio individual de objetos em movimento. Um dos primeiros trabalhos sobre modelagem de distribuição espacial de trajetórias usando quantização vetorial (QV) foi proposto por Johnson e Hogg [Johnson and Hogg, 1995]. Depois QV foi utilizada também por Stauffer e Grimson [Stauffer and Grimson, 2000] combinando-a com agrupamento hierárquico.

Alguns trabalhos, porém, não utilizam trajetórias dos objetos móveis para a interpretação da cena, ou detecção de anomalias. Na interpretação de cenas com multidões, algumas abordagens estão interessadas em detectar se existe multidão, ou como rastrear indivíduos na multidão. Panagiotakis et al. [Panagiotakis et al., 2008] utilizam deformações de silhuetas para detectar se vídeos esportivos contém grupos de indivíduos. Usam uma técnica que permite a detecção mesmo com a câmera em movimento, sendo utilizado para indexação e categorização de vídeos.

Alguns autores abordam a aprendizagem com características de baixo nível, descrevendo eventos discretos no nível de pixels. Nos trabalhos de Tao Xiang et al. [Xiang et al., 2002; Xiang and Gong, 2008], é apresentada uma abordagem utilizando como característica eventos discretos no nível de pixel, chamada de *Pixel Change History* para representar eventos (comportamentos) sem precisar do rastreio de objetos. Um evento neste caso é caracterizado por mudanças visuais acumuladas ao longo do tempo (um grupo de pixels que em certa área do quadro varia ao longo do tempo). Uma atividade seria caracterizada por uma ordenação entre vários eventos. Esta técnica se mostra bastante promissora, apesar de que os eventos ficam extremamente dependentes da posição onde ocorrem.

A abordagem estatística é uma das principais formas de aprendizagem, muito utilizada na construção de modelos de eventos, pois com ela pode-se determinar padrões de eventos

normais a partir do treinamento, e então pode-se identificar eventos que fogem ao padrão. O reconhecimento de eventos de curta duração, como gestos humanos, foi tema de diversos trabalhos usando abordagens estatísticas, como *Hidden Markov Models* (*HMMs*). No trabalho de Starner e Pentland [Starner and Pentland, 1995], HMMs foram utilizadas para detecção de linguagem de sinais. HMMs podem ser utilizadas para eventos sequenciais de diferentes durações, porém só em atividades que envolvam apenas um ator (objeto). Uma técnica chamada de *Coupled HMMs* foi sugerida no artigo de Oliver et al. [Oliver et al., 1999] tentando modelar a interação entre mais atores, fazendo um acoplamento de várias HMMs. Porém, em atividades envolvendo mais de dois atores, o problema se torna muito complexo e difícil de aprender a partir do conjunto de treinamento, já que a probabilidade de mudança de estado de um objeto precisa ser combinado com todos os outros objetos, levando a uma explosão combinatorial.

Algumas abordagens combinam técnicas estatísticas com estruturais, como no trabalho de Ivanov e Bobick [Ivanov and Bobick, 2000], no qual foram utilizadas gramáticas estocásticas livres de contexto. Naquele trabalho, os símbolos da gramática foram gerados por um módulo de detecção de eventos primitivos baseado em HMMs. Naquele trabalho, os símbolos da gramática foram gerados por um módulo de detecção de eventos primitivos baseado em HMMs.

2.4.2 Métodos de agrupamento

A aprendizagem não-supervisionada, como discutida anteriormente, é importante para a aplicações em que não se dispõe de um conjunto de treinamento rotulado. Desta maneira, técnicas de agrupamento são utilizadas objetivando o descobrimento automático de grupos de dados semelhantes. As técnicas de agrupamento podem ser divididas, de acordo com Han e Kamber [Han and Kamber, 2001], em cinco categorias: métodos de particionamento, hierárquico, baseado em densidade, baseados em *grid*, e baseados em modelos. Entre os algoritmos mais utilizados, destacam-se os mapas auto-organizáveis, *k-means*, *fuzzy k-means* e *Expectation-maximization*. Para mais informações, nos trabalhos de Xu et al. [Xu and Wunsch, 2005] e Jain et al. [Jain et al., 1999] são apresentadas as principais técnicas de agrupamento, aplicadas especificamente a problemas de detecção de anomalias em vídeos.

2.4.3 Eventos pré-definidos

O processo de reconhecimento de eventos pré-definidos no monitoramento automático de vídeos, analisando cada quadro de vídeo, pode ser definido como a análise de entidades (objetos rastreados, elementos estáticos da cena e eventos primitivos) relacionadas a propriedades temporais e não-temporais objetivando a interpretação das cenas.

Na abordagem de eventos pré-definidos, os eventos de alto nível da aplicação são conhecidos previamente por um especialista. Alguns autores utilizam máquinas de estados, gramáticas, sistemas baseados em regras etc. Algumas abordagens nesta categoria utilizam restrições e probabilidades para lidar com situações em que o evento não segue estritamente o modelo predefinido. Em todas estas técnicas, um especialista manualmente modela previamente os eventos ou elabora regras e gramáticas para detectar os eventos em vídeo.

Alguns trabalhos utilizam redes de Petri como técnica de modelagem de eventos. Como exemplo, o trabalho apresentado por Castel et al. [Castel et al., 1996], propõe uma linguagem simbólica, expressando as propriedades e restrições dos elementos. Além disto, são propostos protótipos de atividade, que funcionam como regras e restrições para eventos primitivos, e protótipos de planos, utilizando redes de Petri para interpretar conjuntos de protótipos de atividades, detectando assim eventos de alto complexidade (compostos por vários eventos primitivos).

No trabalho de Albanese et al. [Albanese et al., 2008], os autores propõem a utilização de uma rede de Petri para a detecção de eventos de alto nível em vídeos. Um especialista é responsável por modelar manualmente todos os eventos possíveis numa rede de Petri, com probabilidades entre transições. A primeira etapa é a definição de um conjunto de eventos primitivos chamados de *actions symbols*, em que um algoritmo específico de processamento de imagem para cada *action* determina sua ocorrência em um determinado quadro do vídeo. Um módulo gera então rótulos para os quadros de vídeo para posterior utilização. Na rede de Petri, a ativação e o disparo de transições ocorrem quando o rótulo de entrada casa com a restrição desta transição, sendo então computada a probabilidade até chegar num estado terminal. Os estados terminais representam os eventos de alto nível, então no final do processamento é possível obter uma probabilidade para cada evento de alto nível e, a partir de um limiar, definir se o evento é válido. O autor apresenta dois algoritmos chamados *naivePPN-MPA* e *PPN-MPA* para determinar os eventos com maiores probabilidades em um trecho de

vídeo, como também um algoritmo chamado *PPN-MPS* para achar trechos mínimos em um vídeo em que ocorre um determinado evento.

Ghanem et al. [Ghanem et al., 2004] propõem um sistema para detectar eventos especificados numa linguagem de consultas de alto nível. Naquele trabalho, é proposta uma hierarquia de eventos primitivos e compostos, representados em rede de Petri colorida, com relações temporais, espaciais e lógicas entre os eventos. A rede é elaborada por um especialista para cada cenário, porém uma biblioteca de redes é utilizada para a detecção de eventos primitivos para vários cenários. Os *tokens* coloridos representam atores na cena, como carros ou pessoas. O reconhecimento dos eventos se dá com a dinâmica dos *tokens* na rede, e pela marcação da rede é possível saber o estado dos objetos na cena, e quando atingem o estado final significa que o evento de alto nível foi reconhecido.

Na pesquisa de Borzin et al. [Borzin et al., 2007], os autores propõem uma rede de Petri estocástica para detectar os eventos em vídeo. Na representação da rede, os *tokens* são associados com os objetos em cena, as posições representam os estados do objetos, e as transições são associadas a eventos primitivos. Após ser construída a rede, vídeos de exemplo para um cenário particular são utilizadas para treinar a rede, resultando em probabilidades e a definição de um tempo médio entre as transições. Dessa forma, é possível prever estados futuros com a utilização do gráfico de alcançabilidade. Como exemplo, os autores demonstraram uma aplicação de entrada em bancos com vistoria de clientes.

Uma série de modelos declarativos, propostos por Rota et al. em [Rota and Thonnat, 2000], é utilizada para descrever todo um sistema de monitoramento, como estados da cena, eventos e cenários. Os eventos são determinados pelas condições entre os objetos na cena. Os autores ainda sugerem um algoritmo de satisfação à restrição chamado de *Arc Consistency-4* ou *AC-4* para reduzir o tempo de processamento no reconhecimento dos eventos. Dessa forma, transformando o problema de reconhecimento de eventos em um problema de resolução de restrição.

Já no trabalho de Joo et al. [Joo and Chellappa, 2006], os autores propõem o reconhecimento de eventos anormais a partir de um *parser* de uma gramática probabilística. Os símbolos terminais da gramática são eventos primitivos da cena, que são detectados a partir de regras espaciais dos elementos da cena, enquanto que os não-terminais são eventos de alto nível. De acordo com a geração de eventos primitivos em tempo-real da cena e seu

casamento com a gramática, é possível identificar os eventos anormais, que são aqueles em que não obedecem a gramática. Já os eventos que satisfaçam à gramática, mas que tenham uma probabilidade abaixo de um nível pré-determinado, também são considerados eventos anormais. Caso contrário, são classificados como eventos normais.

Como uma abordagem diferente das anteriores, Shet et al. [Shet et al., 2005] propõem um sistema de regras em Prolog para representar e detectar eventos envolvendo pessoas, pacotes e o ambiente num vídeo. Um módulo de baixo nível faz rastreio dos elementos, um módulo intermediário gera eventos primitivos e adiciona-os no sistema como fatos em Prolog. Neste módulo, elementos do fundo da cena são rotulados a fim de serem identificados na interação com os elementos em movimento. O módulo de alto nível é a máquina de inferência Prolog que trata de reconhecer os eventos a partir dos fatos e das regras definidas previamente. As definições destas regras utilizam os fatos (eventos primitivos) gerados no módulo anterior. O sistema é utilizado tanto para monitorar continuamente um ambiente até que uma regra seja disparada, como também para possibilitar realização de consultas de alto nível em vídeos *offline*.

No trabalho de Vallejo et al. [Vallejo et al., 2009], é proposta a utilização da linguagem Prolog para detectar eventos anormais numa cena de cruzamento com semáforo. Um especialista define em regras de Prolog situações de normalidade num cruzamento de trânsito entre veículos e pedestres, como o respeito ao sinal do semáforo. Caso haja eventos que não satisfaçam as regras, portanto que não fazem parte do universo de regras, são considerados eventos anormais e um alerta é acionado.

Os autores Krausz e Herpers [Krausz and Herpers, 2007] propuseram a utilização da linguagem de sistemas especialistas chamada CLIPS para detecção de eventos pré-definidos numa estação de metrô. As regras são definidas baseadas nos objetos móveis e zonas de interesse, mas não há uma análise temporal dos eventos.

Uma técnica simbólica de representação temporal chamada de *Problema de Satisfação de Restrições Temporais* foi proposta por Dechter et al. [Dechter and Schwalb, 1991]. Um exemplo de uma situação que esta técnica pode resolver, de acordo com o autor, é descrita na Figura 2.2.

Dechter et al. [Dechter and Schwalb, 1991] propõem a utilização de um conjunto de variáveis X=$\{X_1, X_2, \ldots, X_n\}$ e um conjunto de restrições K. Cada variável apresenta um

João vai para o trabalho a carro (30 a 40 minutos) ou a ônibus (no mínimo 60 minutos). Frederico vai para o trabalho de carro (20 a 30 minutos) ou numa vã (40 a 50 minutos). Hoje João partiu de casaentre 7:10 e 7:20, e Frederico chegou ao trabalho entre 8:00 e 8:10. Também se sabe que João chegou ao trabalho entre 10 e 20 minutos depois que Frederico deixou a casa. Algumas perguntas que pode ser necessárias neste contexto são: "A informação da história está consistente?","É possível que João pegou um ônibus e Frederico usou a vã?","Quais são os tempos possíveis em que Frederico deixou a casa?", etc.

Figura 2.2: Um exemplo de problema de satisfação temporal [Dechter and Schwalb, 1991].

tempo. Cada restrição é apresentada como um conjunto de intervalos:

$\{I_1, \ldots, I_n\}$=$\{[a_1, \ldots, b_1], [a_n, \ldots, b_n]\}$

Uma restrição para a variável X_i limita sua existência em um conjunto de intervalos relacionados pela disjunção $(a_1 \leq X_i \leq b_1) \vee \ldots \vee (a_n \leq X_i \leq b_n)$.

Uma restrição binária entre duas variáveis X_i e X_j especifica os valores possíveis da distância X_j - X_i através do conjunto disjunto $(a_1 \leq X_j - X_i \leq b_1) \vee \ldots \vee (a_n \leq X_j - X_i \leq b_n)$.

A partir das variáveis e restrições, Dechter transforma estas relações em um grafo, onde os vértices correspondem às variáveis X e as arestas correspondem às restrições K. Na Figura 2.3 é ilustrado um grafo construído a partir do exemplo apresentado na Figura 2.2. Neste grafo, tem-se:

- O vértice 0 é o tempo inicial do problema;
- Os vértices 1 e 2 representam, respectivamente, o tempo que João deixou a casa e o tempo em que ele chegou no trabalho;
- Os vértices 3 e 4 representam, respectivamente, o tempo que Frederico deixou a casa e o tempo em que ele chegou no trabalho;
- As 5 arestas que envolvem as variáveis correspondem aos tempos que cada pessoa leva para ir ao trabalho.

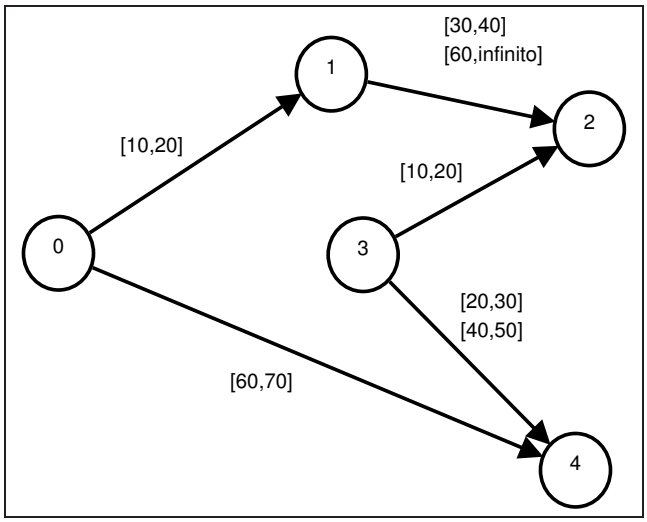


Figura 2.3: Um exemplo de um grafo para o problema de satifação temporal [Dechter and Schwalb, 1991].

A partir deste grafo apresentado na Figura 2.3, é possível resolver as questões elencadas no problema apresentado na Figura 2.2.

2.5 Modelagem do cenário

Uma representação dos elementos do vídeo em 3D foram discutidas nos trabalhos de Mahmood et al. [Syeda Mahmood et al., 2001] e Parameswaran et al. [Parameswaran and Chellappa, 2003]. Porém, a maioria dos trabalhos, como os trabalhos discutidos na Seção 2.4, detectam os eventos com uma modelagem 2D da cena. Uma das limitações da representação em 2D é que com ela a informação visual fica dependente do posicionamento da câmera. Já as técnicas de 3D têm dificuldades como a complexidade da modelagem, e dificuldades na reconstrução do cenário a partir de fonte de imagens 2D.

2.6 Considerações finais

Algumas das dificuldades da abordagem de aprendizagem de eventos residem em problemas de reconhecimento de eventos de longa duração, movimentos complexos e interação com outros objetos. Com estes requisitos, a compilação de um conjunto de treinamento representativo é impraticável, já que é difícil obter amostras suficientes dos possíveis eventos. A

dimensão do problema fica, dessa forma, de difícil tratamento. Além disso, normalmente estes sistemas não reconhecem o mesmo evento realizado de uma maneira diferente, e não lidam bem com concorrência e sincronização.

A qualidade de um sistema de classificação de padrões, em que uma base de treinamento é fornecida para, a partir dela, gerar grupos de padrões, poderá diminuir em aplicações em que os dados mudam constantemente ao longo do tempo. Em aplicações de monitoramento de vídeo, principalmente em vigilância, onde câmeras observam a cena durante longos períodos de tempo e com a possibilidade dos objetos em cena mudarem de configuração e interação, um conjunto inicial fixo de treinamento (*off-line*) pode não representar uma boa fonte de informação, pois os dados para treinamento podem ficar obsoletos.

Estes trabalhos evidenciam a importância da interpretação abstrata de alto nível da cena, onde principalmente eventos simples foram temas de pesquisas. Somados aos trabalhos sobre a abordagem de eventos pré-definidos, percebe-se um espaço para pesquisas sobre modelagem e detecção de eventos pré-definidos utilizando abordagens estruturais. Apesar de todas as referidas técnicas, é importante se ater ao problema que motivou este trabalho, que é a detecção de eventos pré-definidos de alto nível em vídeos. Dessa forma, objetivando a detecção de eventos automaticamente, a direção que foi tomada por este trabalho foi a modelagem de eventos complexos por abordagem estrutural, utilizando um sistema baseado em regras. Outro aspecto é a possibilidade de um especialista, com conhecimento específico de eventos para uma situação em particular, que envolvam múltiplos atores e elementos da cena, utilizar uma linguagem de fácil manipulação como é o caso da linguagem *CLIPS* [Kazarov and Ryabov, 1998].

Capítulo 3

Modelo proposto

3.1 Introdução

O principal problema que esta pesquisa busca resolver é a detecção de situações de interesses a partir da análise de vídeos digitais. Mais precisamente, este trabalho objetiva prover uma representação para especialistas predefinirem as atividades de interesse e um método para detectar essas atividades pré-definidas a partir dos resultados de rastreio e classificação de objetos da cena.

A partir do método para detecção de atividades, foi desenvolvido um sistema de *software* composto de vários módulos, os quais serão melhor explicados na Seção 3.6. Esse sistema recebe como entrada dados sobre o rastreio e a classificação dos objetos móveis, dados sobre os elementos contextuais de interesse da cena, como também regras de atividades de interesse pré-definidas.

O cenário alvo de aplicação do método proposto são seres humanos e carros interagindo, em um ambiente externo filmado por uma câmera fixa posicionada de forma a diferenciar humanos de carros. Mais especificamente, será um cenário de cruzamento de trânsito. Atividades pré-definidas de interesse são abordagens de indivíduos a carros parados, atropelamento, colisões entre automóveis e entre automóveis e pessoas, ultrapassagens ilegais, violação de semáforo. Outros cenários, que não serão alvos de teste, poderiam ser monitoramento de carga/descarga, identificação de pessoas com capacete entrando em estabelecimentos.

Por exemplo, um sistema final resultante desta pesquisa deverá ser capaz de detectar situações como:

- Carro que estaciona;
- Pessoa abordando carro;
- Veículo cruzando sinal vermelho;
- Ultrapassagem ilegal;
- Veículo em contra-mão; etc.

A análise dos fenômenos que ocorrem num cenário pode ser generalizada considerando veículos e pessoas como elementos dinâmicos (em movimento) da cena, porém mantendo suas particularidades, enquanto que as ruas, calçadas, meio-fio, faixas de cruzamento etc. são vistos como elementos contextuais do cenário. Dessa maneira, é razoável considerar que as atividades irão envolver a interação espaço-temporal entre os objetos móveis, como também a interação desses objetos móveis com outros elementos do cenário.

Devido à importância da utilização de sistemas especialistas neste trabalho, este tema será discutido na Seção 3.2. Na Seção 3.3 serão discutidos os conceitos para a representação das atividades, como também a importância da representação qualitativa para a definição de atividades por um especialista. A representação de atividades simples é detalhada na Seção 3.4. Já na Seção 3.5, é detalhada a representação e detecção de atividades compostas. Na Seção 3.6, é descrita uma arquitetura de software em módulos como um proposta inicial de implementação de uma solução para o problema de detecção de atividades pré-definidas em vídeo digital.

3.2 Sistemas especialistas

Um sistema especialista (ou um sistema baseado em conhecimento) é um software que busca resolver um problema de um domínio em particular, que requer um esforço de um especialista humano para a sua solução, emulando o processo de decisão de um especialista [Giarratano and Riley, 2005]. Um especialista é uma pessoa que tem *expertise* em uma determinada área, ou seja, tem o conhecimento e habilidades que a maioria das pessoas não tem. O conceito básico de um sistema especialista é ilustrado na Figura 3.1. Um usuário insere fatos

e recebe *expertise* em resposta. Internamente, a máquina de inferência utiliza a base de conhecimento para derivar conclusões e retorná-las ao usuário.

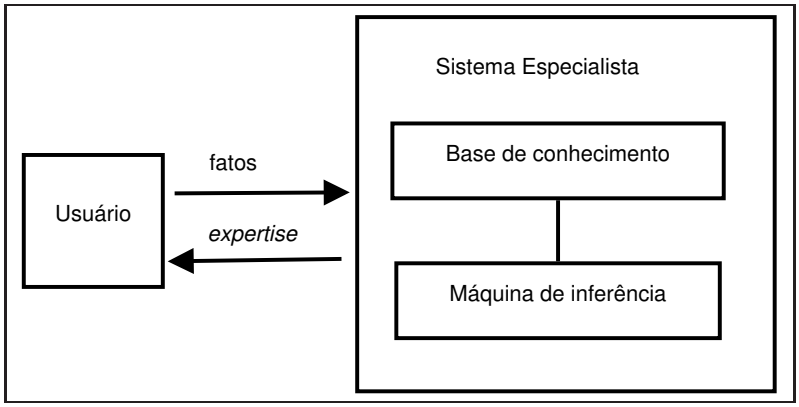


Figura 3.1: Visão geral de um sistema especialista. [Giarratano and Riley, 2005]

Uma forma comum de representar o conhecimento em sistemas especialistas se dá a partir de regras de inferência na forma **SE...ENTÃO...**. Por exemplo:

- **SE** carro estiver parado na rua **ENTÃO** carro está estacionado.

Este tipo de regra é o que permite os sistemas especialistas encontrarem soluções e derivar novos conhecimentos. Existem duas formas de raciocinar sobre as regras, chamadas de *backward chaining* e *forward chaining*. No primeiro caso, a análise das regras é realizada primeiramente pela checagem dos fatos da cláusula **ENTÃO**, para então verificar a cláusula **SE**. Assim, os objetivos (cláusulas **ENTÃO**) determinam quais regras serão analisadas primeiramente. Uma linguagem de programação relativamente popular que implementa este tipo de raciocínio é a PROLOG [Bergin and Gibson, 1996]. Em contraste, o método *forward chaining* analisa a cláusula **SE** para então aferir a cláusula **ENTÃO**. Uma das vantagens do método *forward chaining* em relação ao *backward chaining* é que a recepção de novos dados podem gerar novas inferências (conhecimentos), fazendo com que seja mais apropriado para situações dinâmicas, nas quais as condições referentes à cláusula **SE** mudam constantemente. Um exemplo de sistema especialista que implementa o método de raciocínio *forward chaining* é o CLIPS [Kazarov and Ryabov, 1998].

3.2.1 CLIPS

CLIPS (C Language Integrated Production System) é um software de domínio público para construção de sistemas especialistas. Apresenta um paradigma baseado em regras, permite orientação a objetos como também o paradigma procedural. A lista de fatos, a base de conhecimento e a máquina de inferência constituem os principais aspectos do CLIPS. CLIPS mantém uma lista de fatos nas quais as inferências são derivadas, a base de conhecimento contém regras previamente definidas e a máquina de inferência realiza o controle de execução.

A escolha de CLIPS para utilização como máquina de inferência para detecção de eventos pré-definidos deve-se a(o):

- capacidade de representação de conhecimento, fornecendo suporte a representações baseadas em regras e procedurais. Utilizando o paradigma procedural, pode-se agregar mais funções à linguagem, permitindo assim expandir a linguagem. Como exemplos, funções para detectar atividades simples e funções para raciocínio temporal;
- portabilidade, pois é escrito na linguagem C e compilado em várias plataformas de sistemas operacionais;
- integração com código de usuário, pois assim permite integrar com o código de todo o sistema. Assim, fatos são adicionados e alarmes são capturados através de chamadas a funções disponibilizadas pelo CLIP;
- possibilidade de priorizar a análise de regras. Desta forma, regras para detecção de atividades simples podem ser analisadas prioritariamente em relação às atividades compostas.

Fato em CLIPS é uma informação básica consistindo em um **nome** (*deftemplate-name*) e em zero ou mais atributos (*slots*) com seus valores associados. O Código 3.1 apresenta a sintaxe de definição de tipos de fatos em CLIPS. A palavra-chave *deftemplate* precede o nome do tipo do fato <*deftemplate-name*>, enquanto que *slot-definition* constituem as definições dos atributos associados ao tipo de fato.

Como exemplo, um tipo de fato chamado **pessoa** tem sua definição apresentada no Código 3.2. Um fato do tipo *pessoa* tem seus atributos relacionados na Tabela 3.1. O Código

Código Fonte 3.1: Sintaxe de definição de fato em CLIPS

```
1 (deftemplate <deftemplate-name> [<comment>]
2    <slot-definition>*)
```

3.3 apresenta uma instância de fato em CLIPS referente a esses atributos.

Tabela 3.1: Um fato com seus atributos.

Atributo	**Valor**
nome	Joaquim José da Silva Xavier
estado	Minas Gerais
ocupação	dentista
orientação	republicano

Código Fonte 3.2: Exemplo de definição de um tipo de fato chamado pessoa em CLIPS

```
1    (deftemplate pessoa (slot nome) (slot idade (type NUMBER)) (slot
        ocupacao) (slot orientacao))
```

Ao longo de uma execução de um sistema CLIPS, os fatos poderão ser adicionados, a partir da função *assert*, ou removidos, com o uso da função *retract*.

Para que um sistema especialista realize inferências, ele deve ser provido não apenas fatos, como também devem ter regras. As regras em CLIPS seguem a estrutura **SE ... ENTÃO...**. Assim, as regras são constituídas, além do seu nome, de duas partes, uma é o lado esquerdo (ou *Left-Hand Side, LHS*) e a outra é o lado direito (ou *Right-Hand Side, RHS*). O LHS é composto de uma série de elementos condicionais consistindo de padrões a serem casados quando forem avaliados. O RHS contém uma lista de ações a serem executadas quando o LHS da regra for satisfeito. O símbolo da seta (=>) separa o LHS do RHS. A sintaxe de definição de regras em CLIPS é apresentada no Código 3.4.

Um exemplo de uma regra que faz uso do fato apresentado no Código 3.3 é apresentado no Código 3.5. Nesta regra, será identificado o nome da pessoa que pertença ao estado de 'Minas Gerais' e é republicano.

Código Fonte 3.3: Exemplo de um fato em CLIPS

```
1   (pessoa (nome 'Joaquim Jose da Silva Xavier') (estado 'Minas Gerais')
        (ocupacao dentista) (orientacao republicano))
```

Código Fonte 3.4: Sintaxe de definição de regras em CLIPS

```
1 (defrule <rule-name> [<comment>]
2    [<declaration>]                ; Propriedades de regra
3    <conditional-element>*         ; Lado esquerdo, Left-Hand Side (LHS)
4    =>
5    <action>*)                     ; Lado direito, Right-Hand Side (RHS)
```

Quando a base de conhecimento (na forma de regras) e de fatos são construídas, CLIPS se prepara para executar as regras através da máquina de inferência. As regras que tem seu lado LHS satisfeito, são postas na *Agenda*. CLIPS ordena as regras da agenda em ordem decrescente de prioridade e dispara a regra de maior prioridade. O programador pode controlar a prioridade de uma regra a partir do uso de saliências (*salience*). Quando múltiplas regras tem a mesma prioridade, CLIPS automaticamente determina qual regra é mais apropriada para disparar.

3.3 Monitoramento em vídeo: conceitos

A aplicação alvo da detecção automática de atividades pré-definidas em vídeo é um sistema de monitoramento de vídeos numa área conhecida por um especialista. CLIPS (*C Language Integrated Production System*) [Kazarov and Ryabov, 1998] foi escolhido para implementação do sistema por ser uma ferramenta para desenvolvimento de sistemas especialistas lógicos com base numa memória residente de fatos e uma máquina de inferência de regras. Para um sistema de monitoramento de vídeo, concebido para detectar certos tipos de situações na cena, alguns conceitos são de fundamental importância, como o contexto da cena, a manutenção do estado da cena e a representação e detecção das atividades:

- *Contexto da cena*: um sistema de vigilância monitora um ambiente em particular. A partir dos elementos particulares ao cenário, como portões, muros, rua, semáforos,

Código Fonte 3.5: Definição de uma regra em CLIPS

```
1 (defrule detectaAgitador
2     (pessoa (nome ?nome) (estado 'Minas Gerais') (orientacao
        republicano))
3   =>
4     (inconfidente ?nome)
5 )
```

poderão ser realizadas inferências. O conhecimento sobre esses elementos do cenário precisa ser provido ao sistema de forma automática ou manual. Neste trabalho, a informação sobre o contexto do cenário é fornecida manualmente, na forma de zonas de interesse, especificadas como formas bidimensionais (2D). No contexto de um cruzamento de trânsito, como no exemplo ilustrado na Figura 3.2(a), os elementos contextuais da cena poderão ser: rua, calçada, faixa de cruzamento, semáforo. Esses elementos são representados por um conjunto estático de pontos, formando polígonos 2D, portanto são dependentes da posição da câmera no vídeo. Cada elemento estático tem propriedades bem definidas e são utilizadas como fatos no sistema especialista CLIPS. Um especialista humano deve fornecer ao sistema os elementos estáticos do cenário. Um exemplo de zonas demarcadas por um especialista num cruzamento de trânsito é ilustrado na Figura 3.2(b).

- *objetos móveis*: os agentes que se movimentam na cena têm uma importância fundamental nos sistemas de vigilância, pois são eles que interagem com o ambiente (contexto da cena), e são responsáveis pelas atividades que ocorrem. São considerados elementos dinâmicos, sendo representados normalmente por um retângulo delimitador (*bounding box*).

  Como exemplo, no ambiente de sistema especialista CLIPS, uma possível definição, por um especialista, de elemento móvel (*blob*) é ilustrada no Código 3.6. Os atributos *name*, *class*, *time*, *polygon*, *direction* e *speed* representam respectivamente o nome, o tipo de classificação, o tempo de atualização dos dados, o polígono, a direção e a velocidade do objeto móvel.

(a) (b)

Figura 3.2: (a) Exemplo de um cenário de cruzamento de trânsito. (b) Elementos contextuais de um cenário de cruzamento de trânsito especificados por um especialista humano.

Código Fonte 3.6: Definição de um objeto móvel em CLIPS

```
1 (deftemplate blob (slot name) (slot class (type SYMBOL))
2 (slot time (type NUMBER)) ;update time
3 (multislot polygon (type NUMBER)) (slot direction) (slot speed))
```

- *Manutenção do estado da cena*: um sistema de vigilância deverá manter o estado mais recente da cena. Deverá, por exemplo, atualizar as propriedades dos elementos móveis da cena, como suas posições, velocidades e direção do movimento. Como o sistema em CLIPS mantém uma memória de fatos, a manutenção do estado da cena se dará através de *asserts* e *retracts* dos fatos que se atualizam.

- *Representação e detecção de atividades*: como o objetivo de um sistema de vigilância é detectar algumas situações específicas que ocorrem, os tipos de atividades de interesse precisam ser conhecidos previamente para quando forem detectados, gerar algum alarme no sistema. As atividades pré-definidas são possíveis situações que podem ocorrer na cena, sendo conhecidas e modeladas manualmente por um especialista, de tal forma que ele deverá ser capaz de identificar um mesmo tipo de atividade em vídeos diferentes. Uma das características desejáveis ao elaborar as regras, é que elas sejam genéricas o suficiente para permitir sua utilização em diversos cenários. Porém,

algumas regras podem ser intrínsecas ao contexto do ambiente monitorado, quando alguma peculiaridade do cenário exige uma regra diferente.

3.3.1 Definições utilizadas

A definição dos termos utilizados varia entre as aplicações de análise de vídeo pois os desenvolvedores e especialistas têm suas próprias ideias e visões sobre como descrever as situações da cena. Dessa forma, para a modelagem dos objetos físicos da cena e suas ações, torna-se necessária a definição dos termos evitando erros conceituais. Primeiramente, alguns conceitos são introduzidos com uma descrição sobre eles. Em seguida, são descritas algumas relações entre os conceitos.

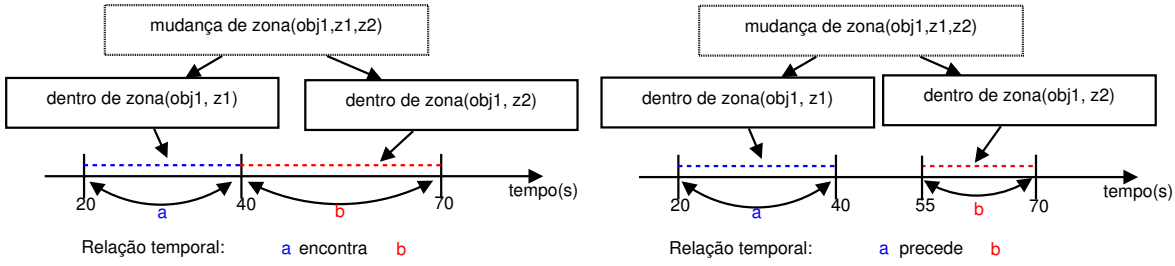
- Atributos de um objeto: são características de um objeto, podendo ser mutáveis ou imutáveis ao longo do tempo. Por exemplo, o identificador do objeto é imutável. Já os atributos de velocidade e direção do movimento são mutáveis, uma vez que podem ter seus valores alterados com o passar do tempo.

- Estado de um elemento: são os valores dos atributos de um certo objeto num certo instante ou intervalo de tempo. Usando como exemplo o Cálculo de Eventos [Shanahan, 1999], este conceito tem o princípio da inércia, ou seja, quando há uma modificação no seu valor, este permanece no tempo até ocorrer nova mudança. Por exemplo, os estados de um objeto móvel como um carro num instante t poderão ser o conjunto (se movendo, dentro de uma determinada zona de monitoramento).

- Evento simples: indica uma mudança instantânea de estado. Por exemplo, caso um objeto parado comece a se mover, isto pode ser interpretado como um evento de nome "objeto moveu-se".

- Atividade simples: é um conjunto de estados de um ou mais objetos em que os valores permanecem inalterados durante um intervalo de tempo. O conjunto de todas as atividades simples forma a base para se detectar as atividades pré-determinadas. Como exemplo podem-se citar "pessoa *p* está dentro de uma determinada zona", "veículo *v1* está parado", "veículo *v1* está parado numa zona rotulada de *z2*", "pessoa *p1* está rente ao veículo *v2*".

- Atividade composta: é um conjunto de relações temporais e lógicas entre as atividades simples. A definição de uma atividade composta compreende relações lógicas e temporais entre atividades simples. Por exemplo, a atividade composta "mudança de zona *z1* para *z2*" pode ser decomposta na sequência de sub-atividades "dentro de zona *z1*" e "dentro de zona *z2*". Como essas sub-atividades devem ter uma relação espacial, nas Figuras 3.3(a) e 3.3(b) são ilustrados, respectivamente, as situações onde a mudança de zona é imediata e quando há um intervalo de tempo entre a mudança.



(a) (b)

Figura 3.3: (a) Mudança de zona imediata, com o fim da primeira atividade coincidindo com o início da próxima atividade (**a** encontra **b**). (b) Mudança de zona onde existe uma separação no tempo entre as sub-atividades no intervalo [40;55] (**a** precede **b**).

- Atividade pré-definida: é um modelo de atividade composta, de interesse do especialista do sistema, para ser detectado. O especialista define relações e restrições qualitativas e quantitativas (por exemplo o tempo de parada de um carro) entre as sub-atividades, e quando o sistema observar a existência em tempo real da atividade composta descrita, é realizada alguma tarefa que pode atuar no ambiente, como soar um alarme ou fechar portões.

3.3.2 Etapas da detecção de atividades pré-definidas

O processo de detecção de atividades pré-definidas é realizado em tempo real, onde a dinâmica dos elementos do vídeo é constantemente analisa por módulos de visão de baixo nível, provendo o rastreio e a classificação dos objetos. Além disto, o processo de reconhecimento utiliza também informações predefinidas de elementos contextuais do ambiente e regras de

detecção de atividades. Neste trabalho de dissertação, o reconhecimento das atividades é baseado nas seguintes premissas:

1. *Classificação perfeita*: todos os objetos móveis são corretamente classificados. Na atual implementação do sistema, para o cenário de cruzamento de trânsito, o tipo de classificação de um objeto poderá ser humano, veículo ou grupo de pessoas.

2. *Rastreio perfeito*: os objetos são rastreados corretamente ao longo de sua existência no vídeo. Isto também significa que outros atributos como retângulo delimitador, silhueta, centroide, velocidade, e direção do movimento são corretamente detectados.

Isto significa que o módulo de representação e reconhecimento de atividades pré-definidas proposto espera que os dados dos módulos de baixo nível (classificação e rastreamento) sejam corretos, não tolerando assim falhas desses módulos. A tolerância à falhas pelo sistema é um aspecto a ser considerado num trabalho futuro. Em resumo, as etapas do processo de detecção de atividades pré-definidas, ilustradas na Figura 3.4, consistem em:

1. Definição dos elementos contextuais do ambiente;

2. Elaboração de regras para modelos de atividades pré-definidos;

3. Rastreio dos objetos móveis;

4. Reconhecimento de atividades simples;

5. Atualização do histórico de atividades de cada objeto;

6. Reconhecimento de atividades compostas;

7. Geração de alarme quando for detectada uma atividade composta.

Então, por definição, a detecção de uma atividade pré-definida é a tupla <ATIVIDADES, RELAÇÕES, TAREFA>. Baseando-se no exemplo de "mudança de zona", ilustrado na Figura 3.3, o especialista pode predefinir regras de atividades para detectar os casos (a) e (b) da figura. Usando relações qualitativas do cálculo temporal de Allen, que serão descritas na Seção 3.5, o especialista pode especificar a seguinte regra para detectar o caso (a):

1. ATIVIDADES: "*obj1* dentro de zona *z1*", "*obj1* dentro de zona *z2*";

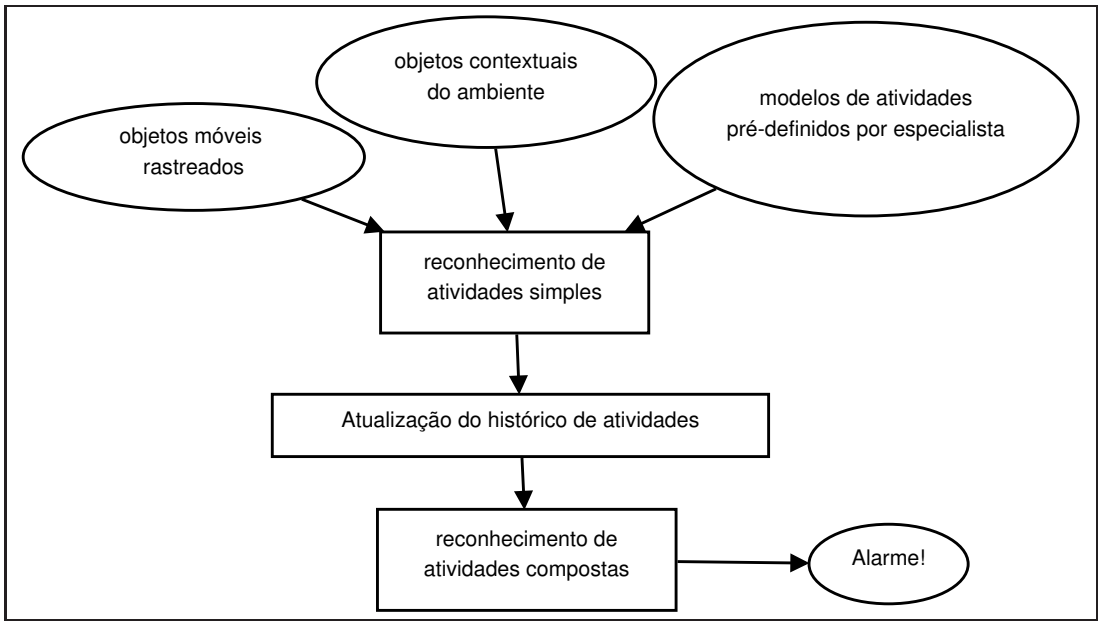


Figura 3.4: O processo de detecção de atividades pré-definidas.

2. RELAÇÕES: "*obj1* dentro de zona *z1*" *encontra* "*obj1* dentro de zona *z2*";

3. TAREFA: gerar alarme indicando mudança imediata de zona do *obj1* de *z1* para *z2*.

Para o caso (b) onde existe um intervalo de tempo entre as sub-atividades, pode-se utilizar uma relação qualitativa ou quantitativa entre elas. Caso o especialista opte por uma relação temporal qualitativa de Allen, a regra ficaria assim:

1. ATIVIDADES: "*obj1* dentro de zona *z1*", "*obj1* dentro de zona *z2*";

2. RELAÇÕES: "*obj1* dentro de zona *z1*" *precede* "*obj1* dentro de zona *z2*";

3. TAREFA: gerar alarme indicando mudança de zona do *obj1* de *z1* para *z2*.

Por outro lado, se o especialista optar por uma restrição mais exata, pode utilizar uma relação quantitativa, como por exemplo definir um intervalo exato (por exemplo 15 segundos) entre a mudança de zona. A seguinte regra pode ser especificada:

1. ATIVIDADES: "*obj1* dentro de zona *z1*", "*obj1* dentro de zona *z2*";

2. RELAÇÕES: diferença entre o tempo inicial de "*obj1* dentro de zona *z2*" e o tempo final de "*obj1* dentro de zona *z1*" igual a 15 segundos;

3. TAREFA: gerar alarme indicando mudança de zona do *obj1* de *z1* para *z2* com intervalo de 15 segundos.

3.4 Atividades simples

As atividades simples consistem nas unidades básicas no processo de reconhecimento de atividades pré-definidas. Nesta seção, serão apresentados alguns conceitos inerentes às atividades simples e sua forma de representação em CLIPS.

3.4.1 Informações quantitativas

O tipo de informação que o módulo de rastreio disponibiliza é considerada uma **informação quantitativa**, isto é, são dados brutos de posicionamento dos elementos no cenário. Informações como a silhueta de objeto e a localização espacial são disponibilizadas de acordo com a modelagem do cenário (duas ou três dimensões) no nível de *pixel*. De acordo com Goyal [Goyal et al., 2000], quando se trabalha com informação quantitativa, normalmente uma unidade de medida é pré-definida. Por exemplo, um objeto A poderá ser dito que está a 50 metros de distância de outro objeto B. Ou que um veículo X se move a 40 km/h. As unidades de medida pré-definidas nestes dois casos foram metros e km/h.

3.4.2 Informações qualitativas

Na **abordagem qualitativa da informação**, como anotado em [Bogaert et al., 2006], a informação no espaço contínuo é quantizada ou discretizada a partir de intervalos pré-definidos, resultando em um espaço de informação discreta. A ideia principal da abordagem qualitativa consiste em criar distinções apenas quando estas são importantes para o problema. No raciocínio qualitativo, apenas a essência da informação é analisada, representando a informação em um conjunto de símbolos predefinidos que fazem sentido para a aplicação. Como exemplo, considerar o espaço discreto {-, 0, +}, consistindo no valor de referência “0” e os intervalos abertos “-” e “+”. Este espaço poderá ser utilizado para descrever a velocidade relativa entre dois objetos: se não for possível ou não for necessário saber as velocidades exatas de um carro e uma motocicleta, mas se sabe que a velocidade do carro é maior que a da moto-

cicleta, então se poderá rotular esta informação com o valor qualitativo "+", significando que o carro está com uma velocidade superior a da motocicleta. Como também, pode-se rotular a relação inversa, na qual a motocicleta está com velocidade inferior ao carro, através do uso do símbolo "-" para esta relação. Já quando os dois objetos estão a uma mesma velocidade, o símbolo "0" é utilizado. Desta maneira, qualquer mudança na relação de maior para menor velocidade, ou o inverso, sempre atravessará o valor "0". Esta ideia ilustra a importância da continuidade no raciocínio qualitativo, como anota Forbus [Forbus, 1990]:

"A noção de continuidade é uma maneira formal da ideia intuitiva de que as coisas mudam suavemente. Uma simples consequência da continuidade, em relação a todos os sistemas qualitativos físicos, é que, na mudança, a quantidade deverá atravessar todos os valores intermediários. Ou seja, se A < B no tempo t_1, então não poderá acontecer de que num tempo futuro t_2, A > B, a menos que exista um tempo t_3 entre t_1 e t_2 tal que A = B".

Uma série de vantagens no raciocínio qualitativo faz com que esta abordagem seja utilizada, por se só ou como complemento ao raciocínio quantitativo, nas áreas de inteligência artificial e sistemas de informação geográfica. Entre algumas vantagens elencadas por alguns autores, estão:

- Existência de muitas situações em que a informação qualitativa é disponível e a quantitativa não é [Freksa, 1992] e [Freksa and Universität, 1992].

- O conhecimento qualitativo tende a ser menos custoso que o raciocínio quantitativo, já que contém menos informações [Freksa and Universität, 1992]. Num estágio inicial, muitas questões podem ser analisadas apenas com a informação qualitativa, quando a informação quantitativa requer uma análise mais profunda [Iwasaki, 1997].

- O principal objetivo de diversos sistemas de raciocínio é tomar uma decisão, sendo esta mais qualitativa do que quantitativa [Freksa and Universität, 1992].

- Nem sempre é o caso em que toda as informações precisam ser conhecidas para inferir algum conhecimento. Abstraindo detalhes métricos, a representação qualitativa é muito mais apropriada para lidar com informação incompleta do que a abordagem quantitativa [Cristani et al., 2000].

- Como atesta Sharma [Sharma, 1996], a abordagem qualitativa é mais apropriada na

representação espaço-temporal para a cognição humana em relação aos métodos quantitativos.

Desta forma, neste trabalho de dissertação serão utilizadas, além das informações quantitativas, as representações qualitativas da informação, objetivando facilitar a implementação de regras simbólicas de eventos por especialistas.

3.4.3 Conceitos visuais

Um dos objetivos dessa pesquisa de mestrado é permitir a um especialista especificar os eventos através de uma linguagem lógica. Para uma especificação de alto nível, uma representação qualitativa das informações pode auxiliar nesta tarefa. Esta subseção tem por objetivo descrever algumas técnicas conhecidas sobre representações espacial, temporal e espaço-temporal entre elementos numa cena. Elas descrevem relações qualitativas entre os elementos na cena, convertendo uma representação numérica em simbólica.

Relações espaciais entre elementos da cena

Devido à importância do espaço físico no contexto da detecção de eventos, faz-se necessário analisar as relações espaciais entre os elementos na cena. Uma ontologia sobre relações espaciais pode ser definida como sendo um conjunto de conceitos utilizados para descrever relações espaciais entre os elementos. Uma ontologia, proposta no trabalho de Kuipers [Kuipers, 1996], divide as relações espaciais entre relações topológicas, relações de distância e de orientação. Como a representação dos elementos nesta pesquisa é em formas 2D, a teoria será restrita a relações binárias em 2D.

1. **Relações topológicas:** Topologia se refere à noção de conexidade entre objetos. Um espaço topológico é dito desconexo quando contém a união de dois conjuntos abertos disjuntos não vazios. Caso contrário, é dito conexo. Segundo Knauff et al. [Knauff et al., 1997], estudos cognitivos empíricos mostraram que os humanos utilizam consideravelmente as relações topológicas no raciocínio visual. As relações topológicas são relações (binárias) entre duas regiões, e dentre as formalizações teóricas se destacam a RCC-8 (*Region Connection Calculus*) [Randell et al., 1992] e a *9-Intersection Model* [Egenhofer and Franzosa, 1991]. Essas duas teorias definem o mesmo conjunto de

relações. Em particular, a teoria de RCC-8 define oito relações topológicas básicas, ilustradas na Figura 3.5.

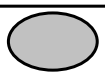
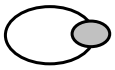
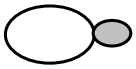
Relação	Significado	Representação
EQ(x,y)	x idêntico a y	
NTTP(x,y)	x é parte não tangente de y	
TTP(x,y)	x é parte tangente de y	
NTTP $^{-1}$(x,y)	x contém não tangencialmente y	
TTP $^{-1}$(x,y)	x contém tangencialmente y	
PO(x,y)	x parcialmente sobrepõe y	
EC(x,y)	x é externamente conectado com y	
DC(x,y)	x é desconectado de y	

Figura 3.5: Relações RCC-8

2. **Relações de orientação:** As relações relativas de orientação (ou de direção) descrevem onde um objeto é localizado em relação a outro. As relações de orientação podem ser descritas através de direções cardeais (*norte*, *sul*, *leste*, *oeste*) no contexto de um espaço geográfico, quando um ponto de referência como o Pólo Norte existe. Relações direcionais primárias como *esquerda*, *direita*, *acima* e *abaixo* foram sugeridas por Freeman [Freeman, 1975]. Além dessas relações de orientação descritas, é importante anotar que as relações de orientação não são binárias. Além dos dois objetos, é necessário especificar um quadro de referência para estabelecer a relação. Existem três tipos de quadro de referência: Alocêntrico (direção dos objetos através de uma perspectiva fixa imposta por fatores externos), Egocêntrico (direção dos objetos a depender da perspectiva do observador) ou Intrínseco (direção a depender das propriedades de orientação do objeto).

3. **Relações de distância:** As relações de distância envolvendo objetos normalmente são classificadas como próximo e distante. Como as relações de orientação, as relações de distância são estabelecidas por três conceitos básicos: objeto primário, objeto referido e quadro de referência. As distâncias são altamente dependentes da escala. Por exemplo, a relação de que A está próximo de B envolve não apenas as posições absolutas desses objetos, envolve também seus tamanhos relativos e a perspectiva do sistema de coordenadas (quadro de referência). Existem três tipos de quadro de referência do sistema de coordenadas:

   - Intrínsico: a distância é determinada a partir das propriedades intrínsecas dos objetos. Normalmente, o tamanho relativo do objeto primário é utilizado.
   - Extrínsico: a distância é determinada por fatores externos como por exemplo arranjo espacial dos objetos.
   - Dêitico: a distância é determinada pelo ponto de vista do observador.

   Uma visão geral das relações espaciais é ilustrada na Figura 3.6, que reproduz uma ilustração de Hudelot [Hudelot, 2005].

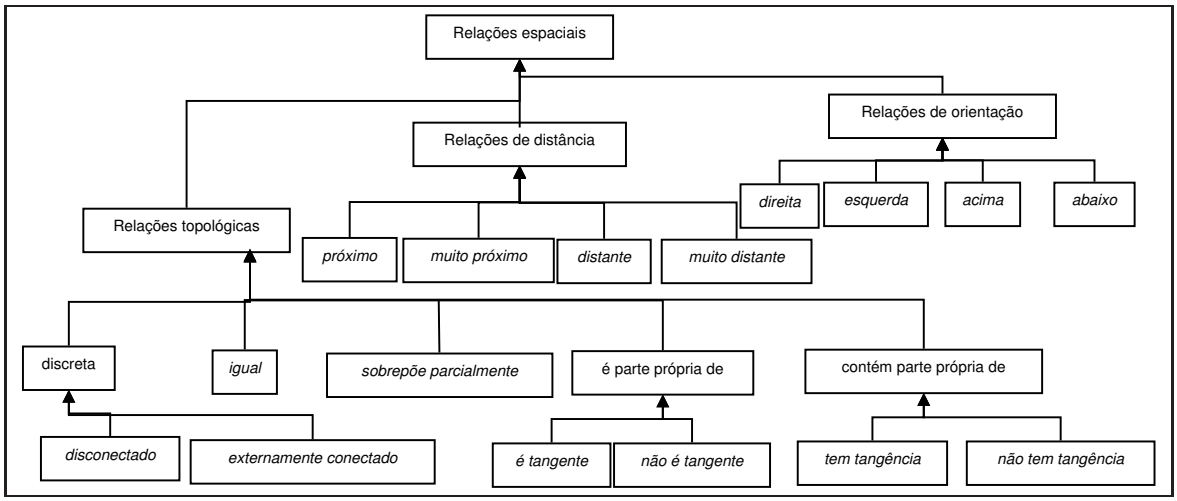


Figura 3.6: Visão geral das relações espaciais [Hudelot, 2005]

Relações espaço-temporais entre objetos

Os métodos formais de representação qualitativa de relações espaciais e temporais são formas úteis de abstrair o mundo real. A formalização das relações temporais foram alvo de

pesquisas de autores como Allen, que propôs o Cálculo de Intervalos [Allen, 1983], e do autor Freksa, que propôs o Cálculo de Semi-Intervalo [Freksa, 1992].

Uma noção intuitiva de movimento de um objeto acontece quando ele ocupa diferentes posições no espaço em tempos distintos. Pode-se pensar num movimento contínuo, quando se observa o percurso de uma bola lançada. Segundo Eschenbach [Eschenbach et al., 1999], geralmente um movimento é representado por uma trajetória, que é uma conexão de linhas tendo uma forma e uma direção.

Apesar da importância dos formalismos topológicos para caracterizar relações entre objetos no espaço, existe uma limitação para a representação de objetos móveis. Estes formalismos ignoram algumas características importantes de objetos móveis, como o conhecimento se dois objetos estão se afastando ou se aproximando. São situações envolvendo objetos desconexos, salvo os casos em que os objetos se tocam, como colisão de automóveis ou uma pessoa entrando num veículo. Na maioria das situações, portanto, as relações dos objetos móveis é caracterizada pela relação *disconnected from* (DC) no formalismo RCC-8 (vide Figura 3.5), ou seja, objetos desconexos.

Combinando as representações espacias e temporais, para representar os movimentos de objetos com relação DC do método formal RCC-8 referido na subseção anterior e dos cálculos temporais, Van de Weghe [de Weghe, 2004] propôs o Cálculo Qualitativo de Trajetória (*Qualitative Trajectory Calculus*), QTC.

De acordo com o autor, o QTC define relações qualitativas entre dois objetos móveis, comparando distâncias entre as posições dos objetos (pontuais) em diferentes momentos no tempo. Diferentes tipos de QTC foram propostos pelo autor, a depender do nível de detalhe e do número de dimensões espaciais necessárias. São dois tipos principais de QTC, dos quais os demais tipos derivam: QTC *Basic* (QTC_B) e QTC *Double-Cross* (QTC_C). Por exemplo, em QTC_{BL1} (QTC_B de nível 1), apenas a mudança da distância entre os dois objetos é de interesse, não importando a dimensão utilizada. Já no QTC_{BL2} (QTC_B de nível 2), existem o conhecimento sobre a distância e ainda a velocidade entre ambos objetos.

O princípio do cálculo QTC é a comparação entre posições dos objetos em diferentes pontos no tempo. O movimento de um objeto (k) com relação a um segundo objeto (l) é calculado comparando a distância entre l no tempo atual (t) e k durante o período imediatamente anterior ao tempo atual (t^-), com a distância entre l no tempo t e k no tempo imediatamente

após o tempo atual (t^+).

A comparação inversa do movimento, entre um objeto *l* e um segundo objeto de referência *k* é análoga. Compara-se a distância entre *k* no tempo *t* e *l* em t^-, com a distância entre *k* em *t* e *l* no tempo t^+. A partir desse resultado, é possível inferir se dois objetos estão afastando-se, se aproximando ou estão estáveis entre si, resultando respectivamente nos rótulos +, $-$ ou 0 para a relação entre ambos. Então a relação entre dois obejtos *k* e *l* é representada em QTC_{BL1} por uma dupla: a distância espaço-temporal relativa entre *k* e *l*, e entre *l* e *k*, com nove (3^2) possíveis combinações. Já o cálculo QTC_{BL2} (QTC_B de nível 2) incrementa o nível de possibilidades para vinte e sete (3^3), pois utiliza mais uma informação qualitativa, a velocidade relativa entre os dois objetos: a velocidade de *k* poderá ser mais lenta (-), a mesma (0) ou maior (+) que a velocidade de *l*. Dessa maneira, no QTC_{BL2} a relação entre um objeto *k* em relação a outro objeto *l* é representada por um rótulo de três caracteres representando as três seguintes relações qualitativas:

1. O movimento do objeto *k*, em relação à posição do objeto *l* no tempo *t*:

   - -: *k* está se aproximando de *l*, quando as distâncias estão diminuindo entre *k* (movendo no intervalo de tempo t^- até t^+) e a posição fixa de *l* no tempo *t*.
   - +: *k* está se afastando de *l*, em que "se afastando" significa um aumento da distância entre *k* (movendo no intervalo de tempo t^- até t^+) e a posição fixa de *l* no tempo *t*.
   - 0: não há movimento entre *k* e *l*, ou seja, não há nem aumento nem diminuição na distância entre *k* (movendo no intervalo de tempo t^- até t^+) e a posição fixa de *l* no tempo *t*.

2. O movimento do objeto *l*, em relação a posição do objeto *k* no tempo *t*:

   - -: *l* está se aproximando de *k*, quando as distâncias estão diminuindo entre *l* (movendo no intervalo de tempo t^- até t^+) e a posição fixa de *k* no tempo *t*.
   - +: *l* está se afastando de *k*, onde "se afastando" significa um aumento da distância entre *l* (movendo no intervalo de tempo t^- até t^+) e a posição fixa de *k* no tempo *t*.

- 0: não há movimento entre *l* e *k*, ou seja, não há nem aumento nem diminuição na distância entre *l* (movendo no intervalo de tempo t^- até t^+) e a posição fixa de *k* no tempo *t*.

3. A velocidade relativa entre *k* no tempo *t*, em relação à velocidade do objeto *l* no tempo *t*:

   - -: *k* está se movendo mais lento que *l*.
   - +: *k* está se movendo mais rápido que *l*.
   - 0: *k* e *l* tem a mesma velocidade.

Dessa forma, o cálculo QTC_{BL2} teoricamente deve conter 27 relações, porém na realidade só 17 são possíveis, como anota o próprio autor [de Weghe, 2004]. O motivo pelo qual 10 relações são inexistentes é que a velocidade impõe algumas restrições. Por exemplo, um objeto parado não pode se mover mais rápido que um objeto em movimento (–, 0, –). Como também não é possível haver diferença de velocidade quando ambos os objetos estão parados (relações '0, 0, +' ou '0, 0, –'). Na Figura 3.7, são exibidas as relações de QTC_B.

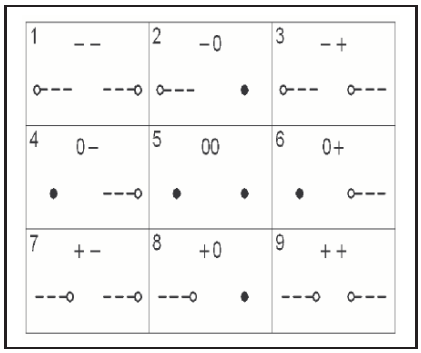


Figura 3.7: Relações do cálculo QTC_B [de Weghe, 2004]

Como discutido, no cálculo QTC Básico (QTC_B) apenas a distância e velocidade são de interesse. Já o cálculo QTC *Double-Cross* (QTC_C) considera adicionalmente a direção do movimento de um objeto em relação ao outro. Van de Weghe propôs examinar a relação do movimento de dois objetos *k* e *l* entre os tempos t_1e t_2. O movimento é representado por um vetor como sugerido na Figura 3.8 (a) ou por um ponto caso o objeto esteja estacionário, ilustrado na Figura 3.8 (b). Entre as origens dos dois vetores, que representam os movimento dos objetos, servem para traçar uma linha de referência chamada *RL*, e ainda mais duas linhas

perpendiculares a *RL*: uma passando por *k* em t_1 (*RL+1*) e outra passando por *l* em t_1 (*RL+2*). Essas três linhas formam então a cruz dupla, que será a referência para o cálculo QTC_C. Na Figura 3.8, é ilustrado um exemplo.

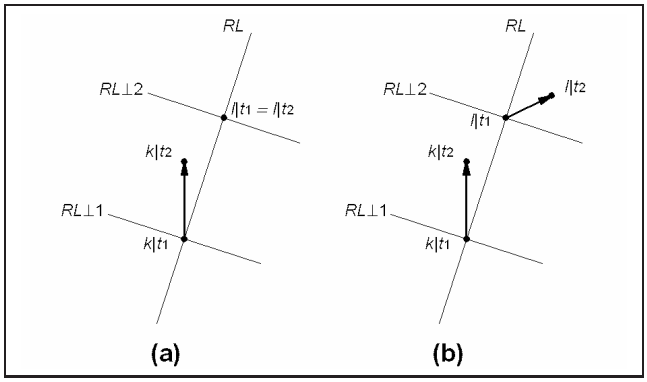


Figura 3.8: Referências para o cálculo QTC_C [de Weghe, 2004]

No cálculo QTC_C, a partir da cruz dupla de referência, as relações qualitativas entre os objetos compreendem as dicotomias aproximando-se/afastando-se e esquerda/direita. Dessa maneira, a relação qualitativa entre um objeto *k* em relação a outro objeto *l* é representada por um rótulo de quatro caracteres representando as quatro relações qualitativas:

1. Movimento de *k* em relação a primeira reta perpendicular (*RL+1*) no tempo *t*:
   - -: *k* está se aproximando de *l*.
   - +: *k* está se afastando *l*.
   - 0: *k* está parado em relação à *l*.
2. Movimento de *l* em relação a segunda reta perpendicular (*RL+2*) no tempo *t*:
   - -: *l* está se aproximando de *k*.
   - +: *l* está se afastando *k*.
   - 0: *l* está parado em relação à *k*.
3. Movimento de *k* em relação a linha de referência *RL* de *k* para *l* no tempo *t*:
   - -: *k* move-se à esquerda de *RL*.
   - +: *k* move-se à direita de *RL*.

- 0: *k* move-se sobre a reta *RL*.

4. Movimento de *l* em relação a linha de referência *RL* de *l* para *k* no tempo *t*:

   - -: *l* move-se à esquerda de *RL*.
   - +: *l* move-se à direita de *RL*.
   - 0: *l* move-se sobre a reta *RL*.

Na Figura 3.9 são ilustrados as 81 (3^4) possíveis relações do cálculo QTC_C. Na figura, pontos à esquerda representa o objeto *k* e à direita, o objeto *l*. Os pontos escuros representam objetos estacionários, os arcos tracejados representam a direção para onde o objeto está se movimentando e as linhas tracejadas da mesma forma.

Depois de discutir alguns métodos formais de representação qualitativa da informação espacial e temporal, será discutida, nas próximas seções, a utilização dessas informações no sistema especialista CLIPS. Mais detalhes dos cálculos QTC encontra-se na tese [de Weghe, 2004].

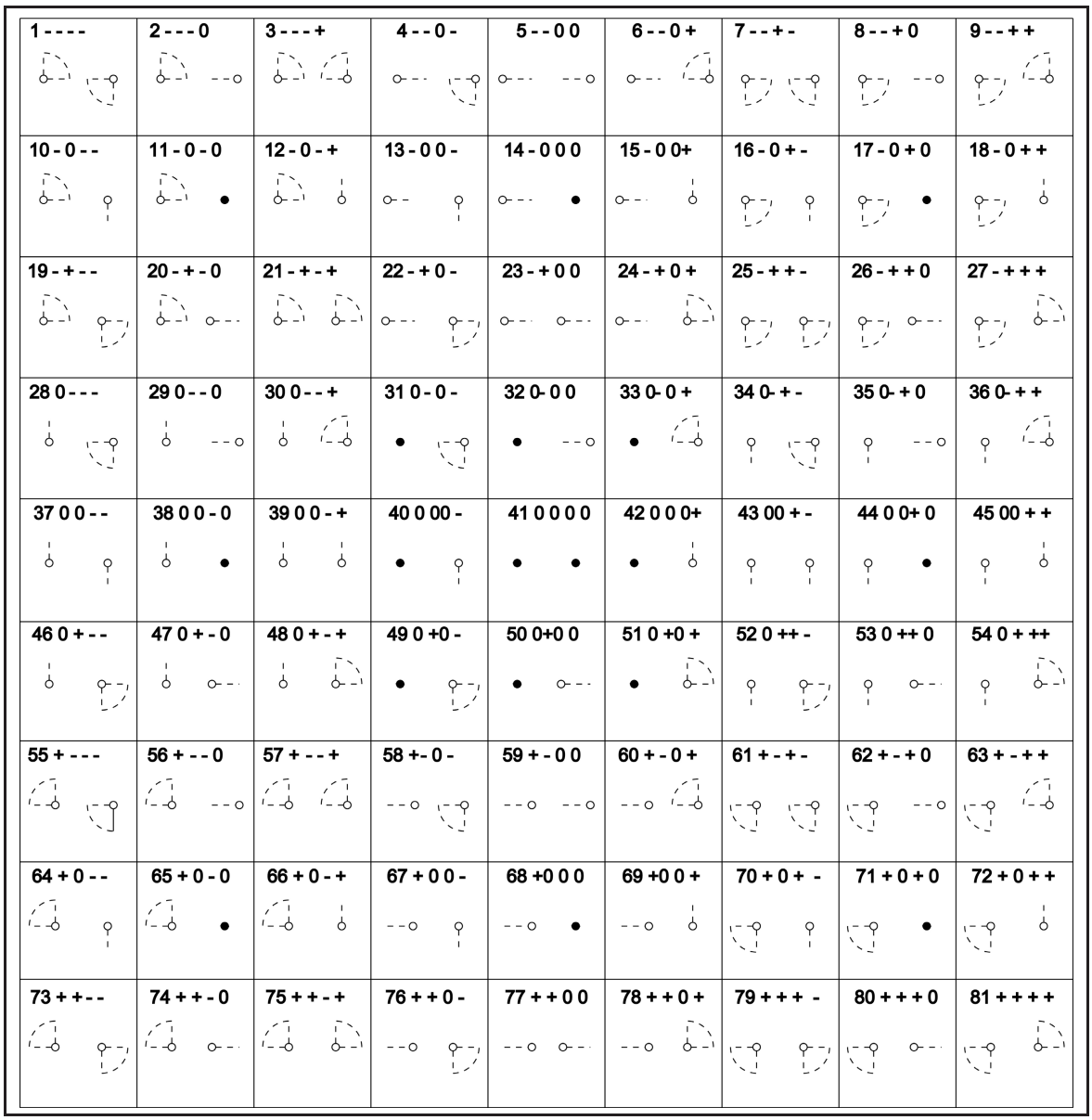


Figura 3.9: As 81 relações do cálculo QTC_C [de Weghe, 2004]

3.4.4 Exemplos de atividades simples

No início do processamento, os elementos contextuais do cenário são providos ao sistema por um especialista ou a partir de uma detecção automática. Para cada objeto móvel detectado na cena, as seguintes informações são disponíveis: identificador, centroide, retângulo delimitador (*bounding box*), silhueta, velocidade, direção do movimento e classificação (humano, automóvel, grupo de pessoas etc.). Se necessária, a trajetória poderá ser inferida a partir do histórico de pontos do centroide ao longo do tempo.

As atividades simples são estados de objetos ou relações entre objetos em cena que ocorrem durante um intervalo de tempo. Para a representação das atividades simples dos objetos, são utilizadas relações topológicas e espaciais entre objetos, nas formas quantitativa e qualitativa. Neste trabalho, por uma restrição de escopo, serão considerados apenas os eventos que envolvam no máximo duas entidades. Algumas das principais atividades simples são descritas a seguir e ilustradas na Figura 3.10.

- Dentro de zona: quando um objeto móvel se encontra fisicamente sobre uma zona de interesse. Esta atividade pode ser verificada a partir da intersecção entre os polígonos da zona e do objeto.

- Movimentação: quando um objeto móvel permanece estático ao longo de quadros sucessivos do vídeo é considerado parado. Para ser considerado parado, pode-se definir um tempo mínimo no qual o centroide do objeto permaneça estável para só a partir deste tempo o objeto ser considerado parado. Caso contrário, é assumido que o objeto está em movimento.

- Rente a um objeto: acontece quando um objeto móvel está fisicamente rente a um objeto estático. A distância entre eles poderá ser zero ou um valor abaixo de um limiar pré-definido.

- Se aproxima: quando a direção da trajetória de um objeto móvel indica que está se aproximando de outro objeto qualquer. O cálculo QTC fornece boas relações qualitativas sobre trajetórias.

- Se afasta: quando a direção da trajetória de um objeto móvel indica que está se afastando de outro objeto.

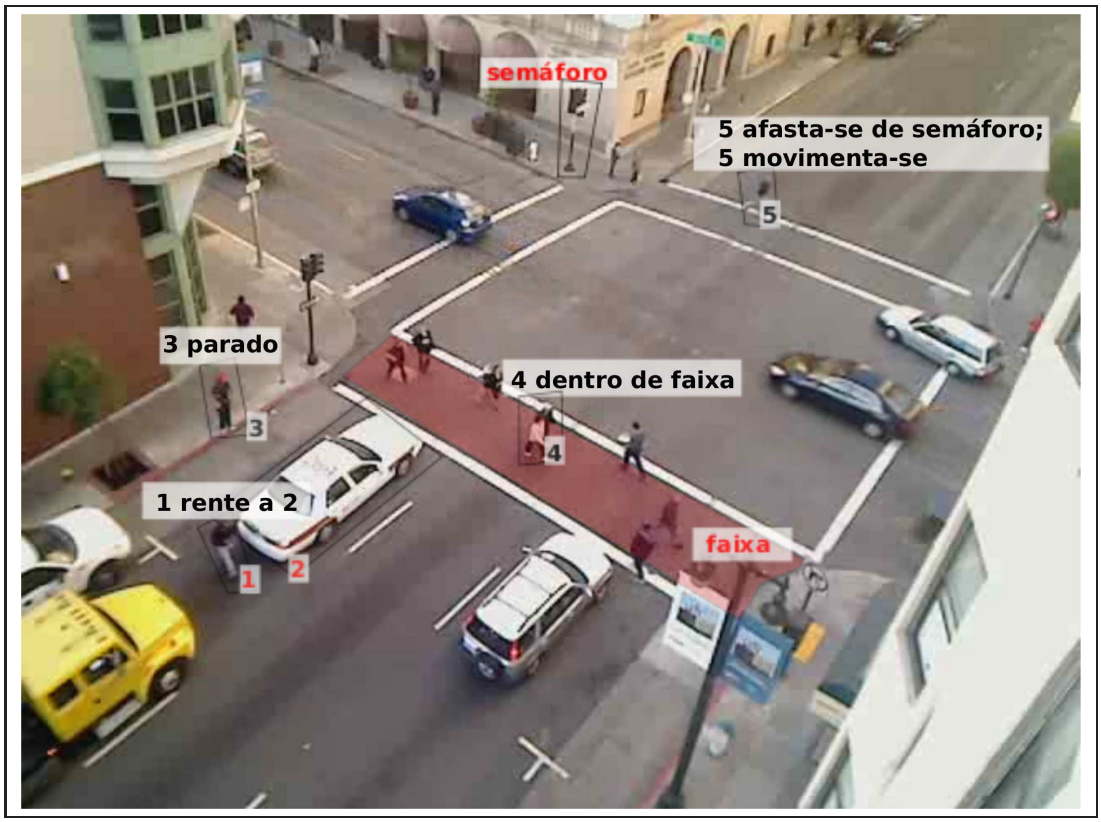


Figura 3.10: Exemplos de atividades simples que podem ocorrer em certo instante numa cena de vídeo.

Cada atividade simples exemplificada anteriormente está associada a um objeto (movimento) ou dois objetos (dentro de zona, rente a um objeto, se aproxima, se afasta) e acontece num certo intervalo de tempo, podendo essas atividades se repetirem ao longo do tempo. Como será discutida na Seção 3.5.4, a manutenção de um histórico dessas atividades simples é importante para detectar atividades compostas. Assim, as atividades que forem reconhecidas podem serem vistas como variáveis temporais. Por exemplo, a variável é temporal quando a atividade "Pessoa *p* está dentro de zona *z1*" é reconhecida durante o intervalo de tempo [5;30]. Dessa forma, a variável tem um tempo de duração (ou seja, início e fim), em que seu valor permanece constante (princípio da inércia). Em um determinado instante de tempo, pode-se identificar os valores de cada variável e a partir daí fazer novas inferências. A manutenção do histórico de atividades e a realização de inferências a partir dele são discutidas na Subseção 3.5.1.

3.4.5 Representação em CLIPS de atividades simples

A representação de atividade simples em CLIPS é composta de uma regra e ainda, a depender do tipo de atividade, de uma função associada à atividade. Por exemplo, se a atividade simples que o sistema deve detectar utiliza apenas as propriedades de objetos, como é o caso da atividade simples chamada "objeto está parado", não há necessidade de chamar uma função, e sim apenas testar o valor da propriedade. A regra então é composta, do lado esquerdo (LHS), por:

- cabeçalho definindo o nome da regra da atividade simples;
- definição de ordem de prioridade (saliência igual a zero) para disparo da regra;
- recuperação dos elementos (do contexto do cenário e/ou objetos móveis);
- teste se a função e/ou propriedades dos objetos relacionada à atividade simples específica possui(em) valor(es) específico(s).

O lado direito (RHS) de uma regra para atividade simples é composta apenas por:

- registro da atividade simples no histórico.

O registro da atividade simples no histórico é necessário para a representação de atividades compostas, que utiliza-o para acessar as atividades simples anteriores do objeto. O restante desta subseção discute alguns exemplos de atividades simples representadas em CLIPS.

Um exemplo de regra para a detecção de uma atividade simples chamado *"está na zona z1"*, é apresentado no Código 3.7. Este exemplo é utilizado para detectar quando uma pessoa está na zona identificada por *z1*. A regra em CLIPS para detecção desta atividades simples é composta de:

- Linha 1: cabeçalho definindo o nome da regra;
- Linha 2: definição de ordem de prioridade alta (igual a zero) para disparo de regra;
- Linha 3: recupera os *blobs* de pessoas;
- Linha 4: recupera todas as zonas;

- Linha 5: compara os polígonos da pessoa e das zonas, através da função *inside*, verificando a relação topológica entre eles. Esta função utiliza uma biblioteca chamada *General Polygon Clipper* (*GPC*), em que verifica se um polígono está dentro de outro;
- Linha 7: realiza o registro desta atividade simples no histórico.

Código Fonte 3.7: Regra para detectar objeto dentro de uma zona

```
1 (defrule dentro_zona
2 (declare (salience 0))
3 (blob (name ?name1) (class person) (polygon $?pol1) (time ?time))
4 (zone (name ?name2) (polygon $?pol2))
5 (test (> (inside $?pol1 $?pol2) 0))
6 =>
7 (registra ?name1 (implode$ (create$ dentro_zona ?name2)) ?time)
8 )
```

A regra faz uma chamada a uma função de nome *inside(o,z,t)*, para verificar se *o* está geometricamente dentro da zona *z* no tempo atual. Esta verificação na atual implementação verifica se o polígono 2D que representa *o* possui a relação *inside* (de acordo com o cálculo RCC-8) em relação ao polígono *z*. Caso a atividade simples seja detectada (presença do objeto *o* na zona *z*), é importante o registro dessa atividade como uma variável no sistema para posterior utilização na detecção de atividades compostas. O registro de eventos será abordado numa seção mais adiante, quando for tratada a detecção de atividades compostas.

Outro caso similar de atividade simples é o caso chamado *"está rente zona z1"*, apresentado no Código 3.8. Esta regra também utiliza propriedades geométricas para estabelecer a relação entre dois objetos, que podem ser objetos estáticos ou móveis. A função *rente(o1,o2)* verifica se *o1* está colado ou muito próximo a *o2*. Especificamente no código de exemplo, o alarme é ativado quando uma pessoa está rente à zona *zona1*.

Um exemplo de atividade simples para identificar quando um objeto está parado é apresentado no Código 3.9. Esta regra verifica para todos os objetos móveis, quais têm o vetor de velocidade igual a zero. Esta propriedade de velocidade é inferida diretamente dos módulos de rastreio de vídeo.

Código Fonte 3.8: Regra para detectar objeto rente a zona específica

```
1 (defrule rente_a_zona
2 (declare (salience 0))
3 (blob (name ?p_id) (class person))
4 (test (rente(?p_id, z1)))
5 =>
6 (printout t "Alarme: " ?p_id " rente a zona " z1)
7 (registra (?p_id, rente_z1))
8 )
```

Código Fonte 3.9: Regra para detectar objeto parado

```
1 (defrule esta_parado
2 (declare (salience 0))
3 (blob (name ?p_id) (speed ?v))
4 (test = ?v 0)
5 =>
6 (printout t "Alarme: " ?p_id " parado")
7 (registra (?p_id, parado))
8 )
```

Para verificar se um objeto móvel está em movimento, apenas o teste da velocidade diferencia a regra de movimento de uma regra de "parado"anteriormente discutida. Uma regra para este caso é apresentada no Código 3.10. Esta regra verifica para todos os objetos móveis, quais têm o vetor de velocidade maior que zero. Da mesma maneira, esta propriedade de velocidade é inferida diretamente dos módulos de rastreio de vídeo.

Uma regra para uma atividade simples quando um objeto se afasta de outro é apresentada no Código 3.11. Na linha 3 e 4, *blob* retorna todos os objetos móveis do sistema, enquanto que na linha 5 garante que os objetos sendo avaliados não são os mesmos. A função *QTC* testa os vetores de velocidade e retorna um identificador da relação espaço-temporal do cálculo QTC_B (vide Fig. 3.7), permitindo assim inferir se os dois objetos estão se afastando.

Um exemplo de interesse num cenário de cruzamento de trânsito é o evento “violação de um semáforo”, ilustrado na Figura 3.11(a) e de colisão, ilustrado na Figura 3.11(b). Caso o

Código Fonte 3.10: Objeto em movimento

```
1 (defrule esta_movimentando
2 (declare (salience 0))
3 (blob (name ?p_id) (speed ?v))
4 (test > ?v 0)
5 =>
6 (printout t "Alarme: " $p_id " movimenta-se")
7 (registra ($p_id, movimenta))
8 )
```

Código Fonte 3.11: Regra para detectar a atividade objeto obj1 se afasta de objeto obj2

```
1 (defrule afasta_se
2 (declare (salience 0))
3 (blob (name ?obj1) (speed ?s1) (direction ?d1))
4 (blob( name ?obj2) (speed ?s2) (direction ?d2))
5 (test (neq ?obj1 ?obj2))
6 (test (= (QTC ?s1 ?d1 ?s2 ?d2) 1))
7 =>
8 (registra ($obj1, afasta de $obj2))
9 )
```

especialista opte por representar estes eventos de interesse como atividade simples, a regra em CLIPS sugerida para a violação do semáforo é apresentada no Código 3.12, onde testa se um veículo está em cima da faixa de trânsito no momento que o sinal está vermelho e ainda gera um alarme no sistema. A informação sobre o estado do semáforo é obtida pelo módulo de Sensor de Semáforo, descrito na Seção 3.6.

Da mesma maneira, caso a colisão seja modelada como uma atividade simples, uma sugestão de regra é apresentada no Código 3.13, testando se os *blobs* se interceptam. Nota-se nestes dois casos que o poder de representação de regras de atividades simples, para detectar situações mais complexas, é pequena. Desta forma, uma representação de atividades mais complexas é necessária.

Todas essas atividades simples são detectadas a partir de informações geradas pelo módulo de rastreio a respeito dos elementos que compõem o cenário. É tarefa do módulo de

(a) (b)

Figura 3.11: (a) Exemplo de uma situação de violação de semáforo. (b) Exemplo de uma colisão entre carros.

Código Fonte 3.12: Exemplo de regra para detectar violação de sinal de trânsito

```
1 (defrule moveRedTrafficLight
2 (declare (salience 0))
3 (blob (name ?name1) (class car) (time ?time) (polygon $?pol1))
4 (crossingtrack (name ?name2) (semaname ?semaname) (polygon $?pol2))
5 (semaphore (name ?semaname) (status red))
6 (test (inside $?pol1 $?pol2))
7 =>
8 (printout t "Alarme: avancando de semaforo vermelho" ?name1 crlf)
```

detecção de atividades simples registrar no histórico o tempo inicial e o tempo final em que a atividade simples acontece. A partir das atividades simples reconhecidas por esse módulo, o sistema poderá detectar atividades mais complexas de alto nível.

3.5 Atividades compostas

A cada quadro de imagem, é enviada a imagem para processamento e detecção de atividades simples. A partir das atividades simples dos objetos que foram reconhecidas, pode-se inferir eventos de maior complexidade, envolvendo relações lógicas e também temporais entre as atividades simples.

Código Fonte 3.13: Exemplo de regra para detectar colisão entre veículos

```
1 (defrule colision
2 (declare (salience 0))
3 (blob (name ?name1) (class car) (time ?time) (polygon $?pol1))
4 (blob (name ?name2) (class car) (time ?time) (polygon $?pol2))
5 (test (neq (?name1 ?name2))
6 (test (inside $?pol1 $?pol2))
7 =>
8 (printout t "Alarme: colisao entre carros " ?name1 " e " ?name2 crlf)
```

Uma atividade pré-definida composta se diferencia de uma atividade simples pois contém um conjunto > 1 de sub-cenários onde estão especificadas relações lógicas e temporais entre essas sub-atividades. São exemplos de **relações lógicas** os operadores *e*, *ou*, *não*. As **relações temporais** incluem os operadores da álgebra de Allen e relações quantitativas sobre duração, início e fim de eventos. Como exemplo, um conjunto encadeado de atividades é considerado uma relação de sequência.

Em resumo, a definição de uma regra em alto nível num sistema especialista para detectar atividades compostas, envolve a definição das atividades simples juntamente com relações temporais e operadores lógicos.

Depois de analisar os objetos independentemente (a partir das atividades simples), e suas relações com o contexto do cenário (atividades simples com elementos de contexto) e entre outros objetos (atividades simples com múltiplos objetos), os eventos que um especialista possa estar interessado envolverá essas atividades detectadas, representadas como estado do objeto, com relações lógicas e temporais entre os objetos. É justamente a possibilidade de utilização de elementos lógicos e temporais, como também o acesso a estados anteriores dos objetos, que distinguem as atividades simples das atividades compostas.

Os operadores lógicos são representados diretamente em CLIPS. A álgebra de Allen [Allen, 1983] é utilizada na forma de predicados temporais em CLIPS. Na Figura 3.12 são ilustrados as sete relações temporais (**precede**, **encontra**, **sobrepõe**, **inicia**, **finaliza**, **durante** e **igual**) que podem existir entre os intervalos I_1 e I_2. A implementação da álgebra de intervalo temporal de Allen em CLIPS é feita a partir de funções em CLIPS.

As atividades simples que ocorrem ao longo do tempo são representadas como uma va-

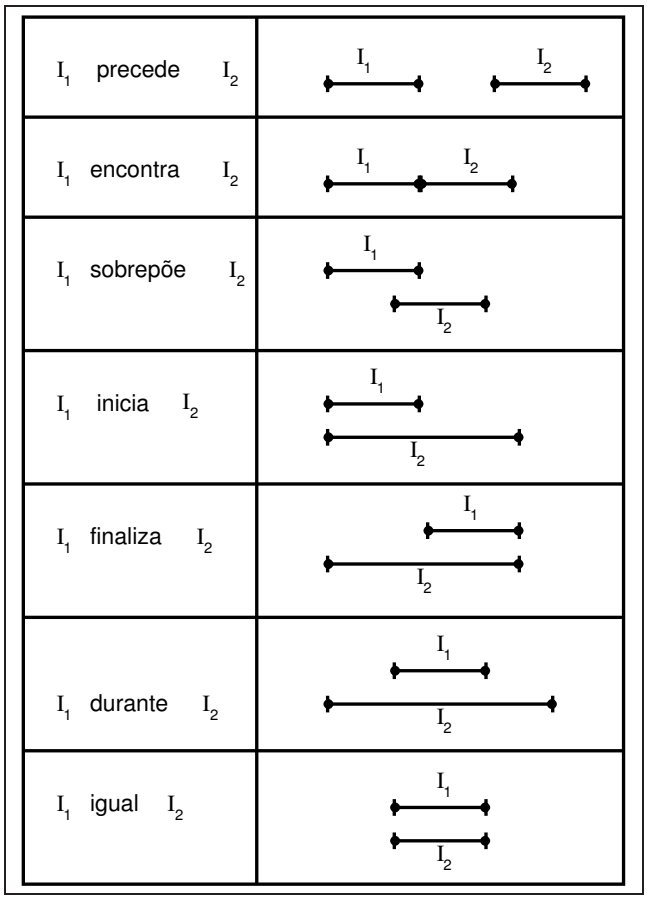


Figura 3.12: Relações temporais entre os intervalos I_1 e I_2. [Allen, 1983]

riável possuindo um conjunto de intervalos temporais, que são registradas no histórico de atividades. Neste trabalho de dissertação, foram definidas as atividades referentes a um objeto como sendo uma dupla da forma <O, V)>, em que:

- O: o identificador do objeto referido;
- V: uma variável temporal, representando um tipo de atividade simples associada ao objeto O. O conjunto V=$\{v_n, v_{n-1}, \ldots, v_1\}$, em que *n* é o índice do intervalo, representa o conjunto de intervalos onde ocorre esta atividade. Já para o intervalo $v_i = [a; b]$, **a** é o início e **b** o fim deste intervalo.

Sejam *a* e *b* os extremos de um intervalo de uma determinada atividade associada a um objeto. Dessa maneira, ficam assim definidas em CLIPS as relações temporais da álgebra de Allen:

- *precede(a,b)*: quando o tempo final de *a* < tempo inicial de *b*.

- *encontra(a,b)*: quando o tempo final de a = tempo inicial de b.
- *sobrepoe(a,b)*: quando o tempo final de a > tempo inicial de b.
- *inicia(a,b)*: quando o tempo inicial de a = tempo inicial de b.
- *finaliza(a,b)*: quando o tempo final de a = tempo final de b.
- *durante(a,b)*: quando o tempo inicial de a > tempo inicial de b e o tempo final de a < tempo final de b.
- *igual(a,b)*: quando o tempo inicial de a = tempo inicial de b e o tempo final de a = tempo final de b.

As especificações das regras temporais, como já discutidas, envolverão duas entidades temporais, que neste trabalho serão as atividades simples. A sintaxe de uma função temporal, que é utilizada na definição das regras de atividades pré-definidas, é apresentada no Código 3.14. O nome da relação temporal é o próprio nome da função <*funcao temporal*>. Na atual implementação, os nomes das funções são *precede_index, encontra_index, sobrepoe_index, inicia_index, finaliza_index, durante_index* e *igual_index*. Esses nomes têm a terminação *_index* apenas para indicar que é uma função que tem como parâmetros os índices. Este índice indica qual intervalo da atividade simples que o especialista está interessado, sendo útil quando se está interessado numa regra composta que envolva várias instâncias de uma atividade em intervalos distintos. Para os parâmetros da função, se observa:

- <obj1>: nome do objeto associado à primeira atividade simples;
- <atv1>: nome da primeira atividade simples;
- <i1>: índice da ocorrência da atividade simples do primeiro objeto. Este índice especifica o intervalo da atividade simples associada ao objeto. Quando o índice é igual a um, se considera o mais recente intervalo.
- <obj2>: nome do segundo objeto referente à segunda atividade simples;
- <atv2>: nome da segunda atividade simples;
- <i2>: índice referente à segunda atividade realizada pelo segundo objeto.

Código Fonte 3.14: Sintaxe de uma função de relação temporal.

```
1 (<funcao temporal> <obj1> <atv1> <i1> <obj2> <atv2> <i2>)
```

Na seção posterior sobre reconhecimento de atividades compostas, é discutida uma forma de representação e análise das relações entre as atividades. Caso se utilize um gráfico de tempo com todas as variáveis associadas a um objeto, o gráfico poderá ser consultado e modificado pelas atividades que forem sendo identificadas ao longo do tempo. Além de ser útil para inferir relações temporárias, seria útil também pela possibilidade de se fazer inferências de eventos pontuais como também atividades com um intervalo de duração.

Para exemplificar, seja o exemplo de contagem de carros estacionados exemplificado no trabalho de Ghanem et al. [Ghanem et al., 2004]. Foi elaborada uma rede de Petri para tal objetivo, como ilustra a Figura 3.13. Como simplificação, mas de maneira análoga, o objetivo é descrever em regras em CLIPS para detectar uma situação de estacionamento, descartando a contagem de carros estacionados num intervalo de tempo.

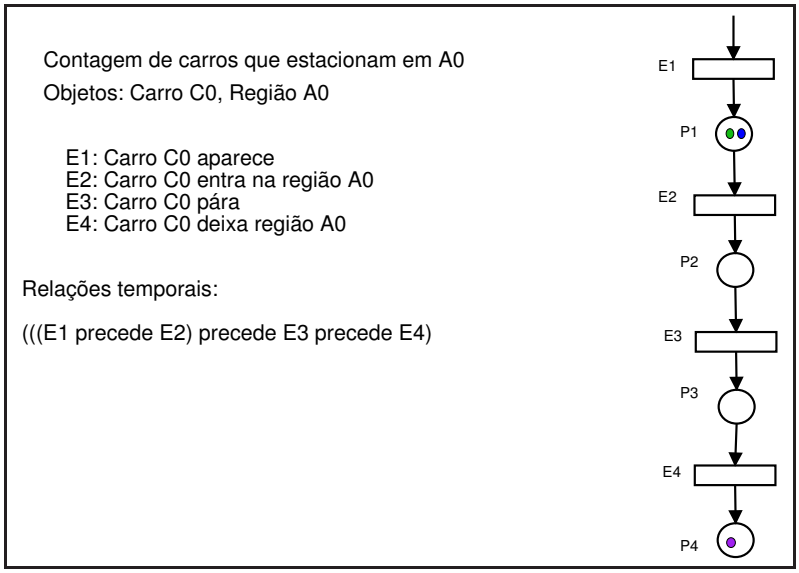


Figura 3.13: Uma rede de Petri para contar o número de carros que estacionaram [Ghanem et al., 2004].

A regra em CLIPS para a detecção de uma situação de estacionamento é apresentada no Código 3.15.

Um exemplo de uma atividade composta que utiliza estas relações é o caso de uma abor-

Código Fonte 3.15: Regra em CLIPS para detecção da atividade pré-definida estacionou

```
1  (defrule estacionou
2  (define (salience -1))
3  (tempo ?now)
4  (blob (name ?car_id) (class car))
5  (historic (nameOBJ ?car_id) (nameACT "dentro_zona z0") (timepoints $?
       times1))
6  (test (eq (- ?now (nth$ 1 $?times1)) 1))
7  (test (durante_index ?car_id "dentro_zona z0" 1 ?car_id parado 1))
8  (test (precede_index ?car_id "dentro_zona z0" 1 ?car_id "afasta z0" 1))
9  =>
10 (printout t "Alarme: " ?car_id " estacionou" crlf)
11 )
```

dagem a um carro, ilustrado na Figura 3.14. Neste exemplo, quatro atividades simples constituem a abordagem a um carro: pessoa p1 está na calçada calcada1 no instante t_1; carro c1 entra na rua zona1 no instante t_2; p1 fica rente a c1 na rua no instante t_3; p1 se afasta de c1 voltando à calçada em t_4; e carro sai da rua no instante t_5.

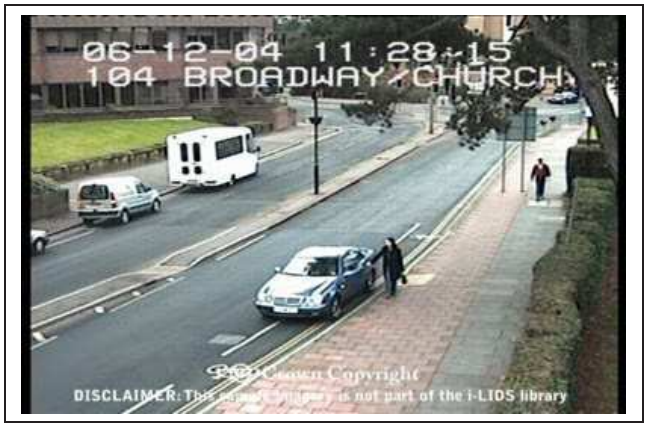


Figura 3.14: Abordagem em um carro.

Os objetos físicos envolvidos no exemplo de abordagem a carro são “carro c1” e “pessoa p1”. As sub-atividades são as seguintes:

- **a**. p1 está dentro de calcada1;
- **b**. c1 está dentro de rua zona1;

- **c**. p1 está rente a c1;
- **d**. p1 se afasta de c1 e entra em calcada1;
- **e**. c1 sai da rua;

E as restrições temporais entre as atividades são:

- **a** antes de **c**.
- **c** durante **b**.
- **d** depois de **c**.
- **e** depois de **d**.

Na Figura 3.15, é ilustrada a decomposição da atividade composta “abordagem a um carro” em sub-atividades e em regras.

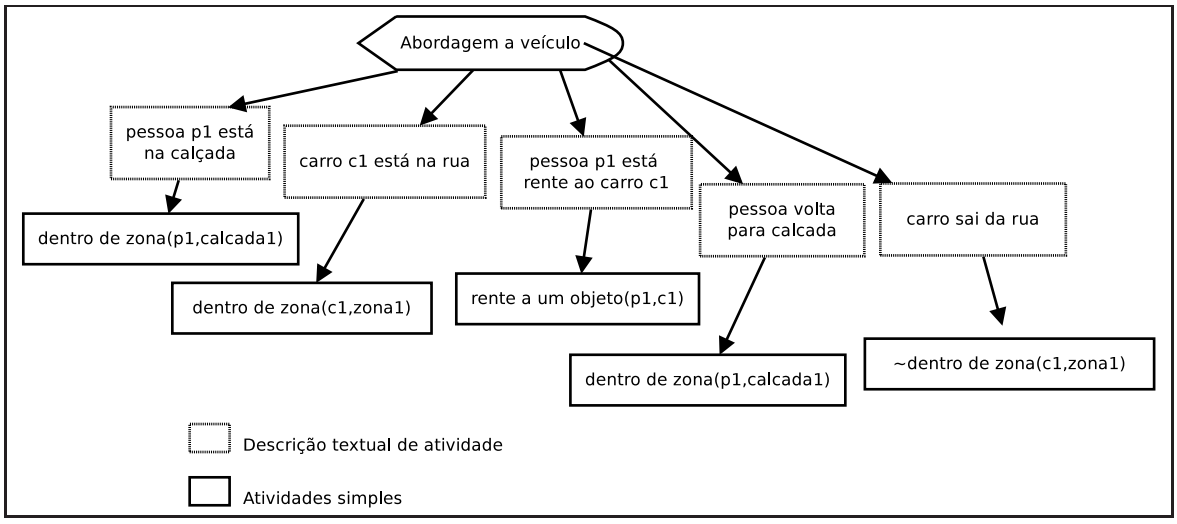


Figura 3.15: A atividade abordagem em um carro decomposta em atividades simples, representadas por retângulos pontilhados, e essas decompostas nas atividades representadas por um retângulo de bordas pretas.

Uma linha de tempo dos eventos em p1 e em c1 permite uma melhor visualização dos eventos, como ilustra a Figura 3.16.

As relações temporais entre as atividades são ilustradas na Figura 3.17.

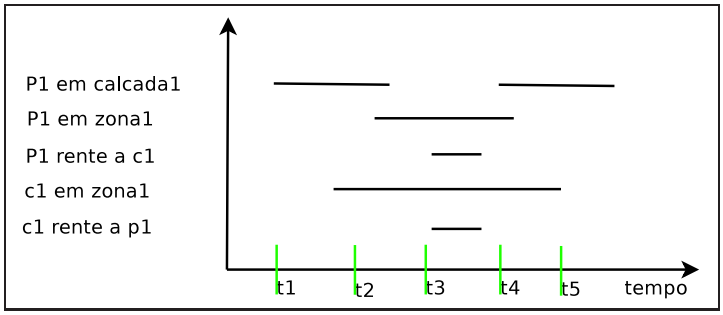


Figura 3.16: Linha de tempo de eventos em p1

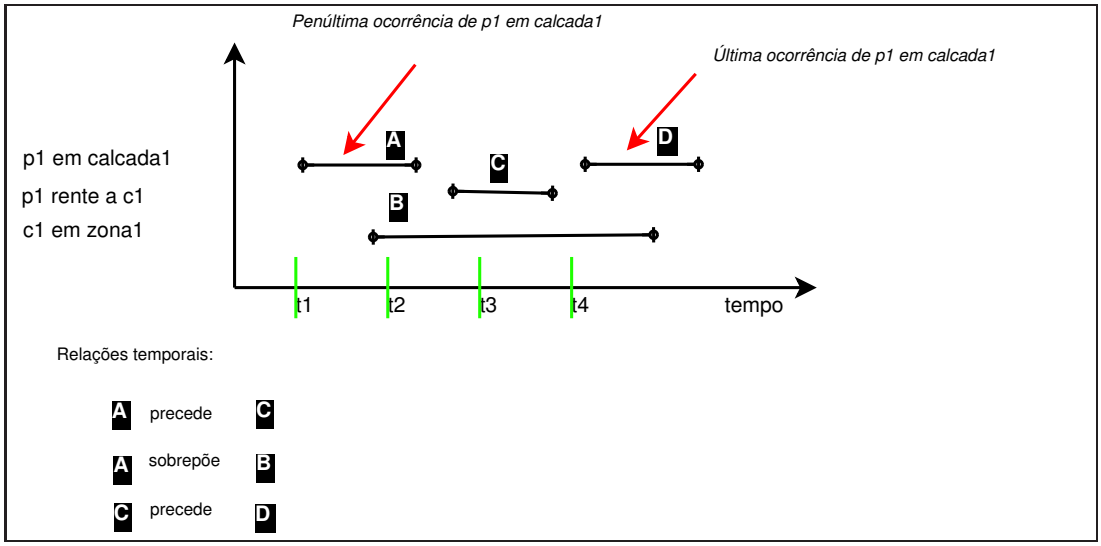


Figura 3.17: Linha de tempo exibindo as relações temporais entre as quatro atividades simples (A, B, C e D)

No exemplo de uma abordagem ao automóvel, existem várias relações envolvendo atividades simples, como mudança de zona. Já a regra para inferir a abordagem no carro é uma atividade composta, podendo ser descrito em CLIPS com a regra descrita no Código 3.16. Neste exemplo, tem-se na linha:

- 1: definição do nome da regra;
- 2: definição de prioridade (menor para regras compostas);
- 3: captura do tempo atual;
- 4: associação de blob do tipo car a variável ?v;
- 5: associação de blob do tipo human a variável ?p;
- 6: captura do tempo em que ocorreu a última atividade (veículo dentro de zona1);
- 7: verificação do tempo para que a regra continue a ser analisada apenas em um instante em que o veículo sair de zona1;
- 8: verifica se temporalmente a pessoa dentro da calçada (penúltima vez na calçada, índice 2) sobrepõe veículo dentro da rua;
- 9: verifica se temporalmente a pessoa dentro da calçada (penúltima vez na calçada) precede a pessoa rente ao veículo;
- 10: verifica se temporalmente o veículo na rua sobrepõe pessoa dentro da calçada (última vez na calçada, índice 1);
- 12: geração de alarme indicando uma abordagem de uma pessoa em um carro.

Código Fonte 3.16: Regra para detecção do evento pré-definido abordagem a carro

```
1  (defrule abordagem
2  (declare (salience -1))
3  (tempo ?now)
4  (blob (name ?v) (class car))
5  (blob (name ?p) (class human))
6  (historic (nameOBJ ?v) (nameACT "dentro_zona zona1") (timepoints $?
       timesv1))
7  (test (eq (- ?now (nth$ 1 $?timesv1)) 1))
8  (test (sobrepoe_index ?p "dentro_zona calcada1" 2 ?v "dentro_zona zona1"
       1))
9  (test (precede_index ?p "dentro_zona calcada1" 2 ?p (str-cat "rente " ?v)
        1))
10 (test (sobrepoe_index ?v "dentro_zona zona1" 1 ?p "dentro_zona calcada1"
       1))
11 =>
12 (assert (alarm (name "Abordagem de pessoa num carro") (time ?now)))
13 )
```

3.5.1 Reconhecimento de atividades compostas

O detalhe específico do sistema é que ele deverá detectar em tempo-real as atividades pré-definidas. A todo instante, a informação do presente da cena é disponibilizada em forma de rastreio e classificação de objetos e dos elementos contextuais do cenário. Apenas o presente e o passado das atividades é disponível, portanto futuro não é conhecido nem se objetiva neste trabalho sua previsão. Quando se detecta alguma atividade simples a partir dos dados do presente através de regras, isto é colocado num histórico. O passado fica então armazenado na forma de histórico de atividades simples. Regras de atividades compostas também são testadas após a avaliação das regras de atividades simples, a cada momento. Então o reconhecimento de atividades compostas se dá na última etapa.

Na Seção 3.3, foram definidos termos relativos à representação dos eventos. Já na Seção 3.5, foram discutidas as representações para atividades compostas, utilizando atividades simples com relações temporais e espaciais. Dessa maneira, constata-se a necessidade de um gerenciamento de um histórico de atividades, a fim de inferir atividades compostas que tenham relações temporais. Nesta seção, primeiramente será apresentado um conceito de variáveis temporais. Logo em seguida, discute-se uma forma intuitiva de visualização, atualização e a realização de inferências referentes às atividades num gráfico temporal.

3.5.2 Variáveis temporais

Nesta seção será introduzido o conceito de variáveis temporais, o qual será util no entendimento do processo de detecção de eventos. Seja um conjunto de variáveis V que representa todos os cenários simples de uma aplicação. E seja V_i o conjunto de variáveis associadas a um objeto *i*. Seja $v_{i,j}$ uma variável temporal j particular desse vetor V_i, ou seja V_i=$\{v_{i,1}, v_{i,2}, \ldots, v_{i,j}, \ldots, v_{i,n}\}$. Tem-se que para a representação de uma atividade composta, as relações a seguir são aplicadas: $v_{i,j}$=$[tempoInicial, tempoFinal(1)]$, significa que a variável assume o valor verdade **true** apenas durante o intervalo de tempo especificado.

Em outras palavras, considere-se um objeto i que está associado a uma regra r de atividade simples que disparou. Para esta atividade, haverá uma variável temporal V_i^r que indica os intervalos de tempo que a regra está disparada com respeito ao objeto i. Desta forma, variáveis temporárias tem a peculiaridade de ser um encadeamento de valores com a

noção temporal de início e fim. Em cada instante de tempo, tem-se um conjunto de variáveis referente a um tipo particular de atividade associadas a um objeto, como por exemplo: V_i^r={$[1, 5], [7, 90], [98, 359]$}. Desta forma, nos intervalos especificados anteriormente, a variável V tem valor verdadeiro, fora do intervalo tem valor falso, indicando que aquele tipo de atividade especificado pela regra R não ocorre.

É justamente realizando inferências sobre as variáveis temporais que se consegue a detecção de eventos compostos.

Uma forma de visualizar a evolução de uma variável ao longo do tempo é traçar um gráfico de linha de tempo desta variável, como no exemplo da abordagem a um carro, ilustrado na Figura 3.16.

Como discutida na seção anterior, a modelagem das atividades compostas é o principal objetivo no sistema de detecção de eventos. As atividades compostas são constituídas de atividades simples com relações lógicas e temporais. Sendo as variáveis representando cada sub-atividade, uma forma intuitiva de analisar os relacionamentos foi ilustrada na Figura 3.17. Traçando-se um gráfico, para cada objeto, com todas as variáveis a ele associadas, tem-se uma noção das relações temporais envolvendo todas as atividades associadas ao objeto.

Uma regra em CLIPS, como comentada na seção anterior, quando detecta uma atividade, faz o registro através da chamada da função *registrar*. O registro de atividades é utilizado para o reaproveitamento desta atividade para detectar atividades compostas. Nesta função, associará a um objeto, um valor a variável que representa a atividade. Enquanto esta atividade for verdadeira para este objeto, o valor desta variável não se altera. Quando não for mais válida a atividade, o valor da variável assume um valor vazio. Portanto, o sistema manterá um histórico de variáveis temporais representando as atividades. Cada variável está associada a uma atividade, e então pode-se consultar no histórico o valor da variável em determinado intervalo de tempo.

Atualização de variáveis temporais

As primeiras regras que são avaliadas no sistema especialista CLIPS são aquelas que representam as atividades simples. Isto é necessário para detectar as atividades inerentes aos objetos, que serão utilizadas posteriormente na detecção de atividades compostas. Então, quando ocorre a identificação de uma regra *r* de atividade simples no quadro de vídeo nú-

mero *qn*, os seguintes passos são realizados:

1. A função *registra(atividade, objetos)*, especificada em CLIPS, consulta a existência de uma variável que representa a regra simples **r** associada a um objeto **O**. Caso não exista, então associa-se uma variável ao objeto, criando **rO**.

2. Verificar se a mesma atividade estava sendo detectada no quadro anterior, e atualizar seu fim caso afirmativo. Seja **rOx** o último intervalo da variável **rO** ao longo do tempo, então se fim de rOx = (qn -1), fim de rOx recebe qn. Caso contrário, início de rOx recebe qn.

Ao final da atualização de todas as variáveis temporais, a máquina de inferência de CLIPS continua o processamento para detectar as atividades compostas. Então, para cada relação temporal entre as atividades simples contidas na regra de atividades compostas, são analisadas os conjuntos de variáveis temporais.

3.5.3 Índices das variáveis

A detecção de atividades compostas consiste na análise das sequências de atividades simples ao longo do tempo. Cada atividade simples poderá ser realizada em vários intervalos de tempo. Uma atividade simples *V* associada a um objeto *o* é a variável temporal Vo=$\{v_n, v_{n-1}, \ldots, v_1\}$, contendo um conjunto de intervalos, onde v_i é o intervalo de índice *i* da variável *V*. O índice *i* mais recente tem o valor 1, e é incrementado quando se referir a um índice anterior. O especialista que descreve a regra para detecção de atividade composta precisa especificar, além das relações temporais e lógicas, qual o índice em que a atividade simples ocorre. A mais recente atividade simples considera-se como o primeiro índice (igual a considerar o índice igual a 1). O índice 2 de uma atividade considera-se como o penúltimo intervalo de existência e assim por diante. Dessa forma, permite-se detectar situações além do último intervalo de uma atividade.

3.5.4 Visualização a partir de gráfico de histórico

Objetivando uma melhor visualização e reconhecimento dos eventos, este trabalho sugere um Gráfico de Histórico Temporal (GHT), que relaciona as atividades simples com o tempo.

Foi discutida anteriormente a necessidade de atividades complexas unir informações temporais de atividades simples. Dessa forma, as atividades simples são aqueles que representam ações relativas a um único objeto como também sua relação com o ambiente, sem haver uma relação temporal. Essas atividades podem ocorrer várias vezes ao longo da existência do objeto, e também são caracterizadas por ocorrerem num ponto no tempo (um evento instantâneo) ou também ocorrerem durante intervalos de tempo. Alguns exemplos de variáveis (atividades simples):

- carro está se movendo ou está parado;
- trajetória do objeto em relação a um ponto referencial usando o cálculo QTC_C, contendo 81 possíveis valores;
- o estado de um semáforo.

O objetivo do GHT é ilustrar a representação das mudanças temporais das variáveis (atividades simples detectadas) de uma entidade no tempo. Alguns fatos são de interesse geral de todos os objetos sendo monitorados. Por exemplo, o estado de um semáforo se relaciona com vários objetos móveis numa cena de cruzamento de trânsito. Quando um objeto móvel passa a ser observado, essa informação pode constar no histórico, associando a este objeto, uma atividade que representa objeto presente na cena, rotulado pelo tempo no qual começou a ser observado.

O GHT tem as variáveis na reta vertical como sendo variáveis temporais que representam atividades simples. A quantidade de variáveis será igual à soma das atividades simples associadas aos atores. Por exemplo, se houver as atividades simples “dentro de zona”, “rente a área”, “parado” e “movimentando-se” associadas ao tipo de objeto “automóvel”, e houver 5 automóveis analisados, então a quantidade de variáveis será de 5 vezes 4 (quantidade de atividades simples). O exemplo de monitoramento de um cruzamento de trânsito, contém uma variável para cada estado de um semáforo (verde, amarelo, vermelho).

3.5.5 Raciocinando com os gráficos

Nesta subseção, serão demonstrados exemplos de como um especialista pode visualizar as atividades no gráfico, como também preencher as informações e como acessá-las.

Para a construção de um gráfico de históricos, na análise de uma atividade simples, constrói-se um gráfico de função tempo/variáveis (ou atividades simples).

Outro atributo relativo ao cenário em particular é o estado do semáforo. Supondo a existência de apenas um semáforo com identificador *sema1*, podendo assumir os valores (Verde, Amarelo, Vermelho, Indefinido), haverá quatro variáveis: (sema1 verde, sema1 amarelo, sema1 vermelho, sema1 indefinido).

O semáforo começa a ser monitorado normalmente desde o início do processamento, no tempo inicial zero, pois é uma entidade conhecida que existe no cenário definido. A partir do valor de cor (Verde, Amarelo, Vermelho, Indefinido) detectado pelo sensor de semáforo, um ponto no gráfico na linha (sema1 <valor_do_sinal>) é feito, formando uma tripla (Ator, Ação, Tempo), que o identifica univocamente. Este valor da cor perdura até haver uma mudança. Portanto, existe um intervalo de existência para cada dupla (Ator, Ação). Na Figura 3.18, é ilustrado esse exemplo. Pode-se perceber, nesta figura, na linha sema1 vermelho, que entre o intervalo [t1,t2] o estado da variável vermelho é verdadeiro, indicando a luz do semáforo. Entre [t2;t3], o sinal é verde. Assim, sucessivamente ao longo da reta, percebe-se a variação.

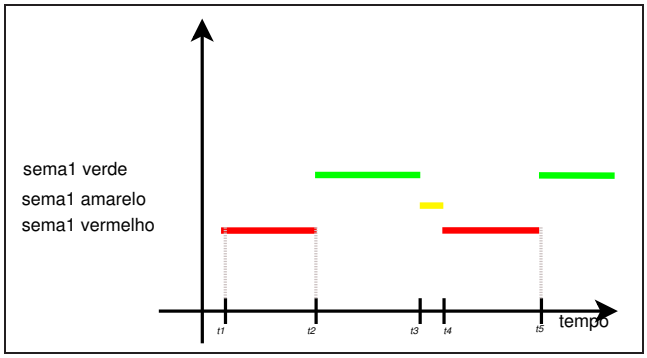


Figura 3.18: Exemplo de um Gráfico de Histórico. O semáforo sema1 está sendo monitorado como uma variável representando o cenário sinal do semáforo sema1 ao longo do tempo. Entre {[t1;t2], [t4;t5]}, o sinal é vermelho. Entre {[t2;t3], [t5;∞]}, o sinal é verde. Entre {[t3;t4]}, o sinal é amarelo.

No cenário de monitoramento de um cruzamento de trânsito, é importante monitorar os objetos móveis. Por exemplo, todas as atividades simples, representadas pelas variáveis, associadas a um automóvel, devem ser especificados pelo especialista, como os seguintes:

- aparecimento: quando o automóvel aparecer na cena, é positivo. Caso contrário, é

negativo;

- movimentação: um estado binário podendo ser está em movimento ou parado;
- dentro de zona z_x: quando o automóvel se encontra dentro de uma zona x em particular;
- trajetória: a trajetória do automóvel no intervalo especificado. Caso seja implementado o cálculo QTC_C, poderá ter 81 possibilidades.

Na Figura 3.19, é ilustrado esse exemplo. Na figura, a atividade simples que representa a variável movimento no intervalo [t0;t1], não é detectada. Porém, nos intervalos [t1;t4] e [t5;t8], a variável é verdadeira, significando que a atividade movimento do objeto está presente durante estes intervalos. A variável presença é verdadeira durante todo o tempo, pois ela representa o evento objeto presente na cena. Já dentro_zona1 ocorre no intervalo [t3;t7] quando o objeto está dentro da zona de nome zona1. A variável dentro_zona2 não é detectada neste exemplo. A variável trajetória representa as trajetórias predefinidas que foram detectadas. Por exemplo, o valor 12 indica que o objeto está tendo uma relação de trajetória predefinida segundo o QTC_c de número 12 no instante t2.

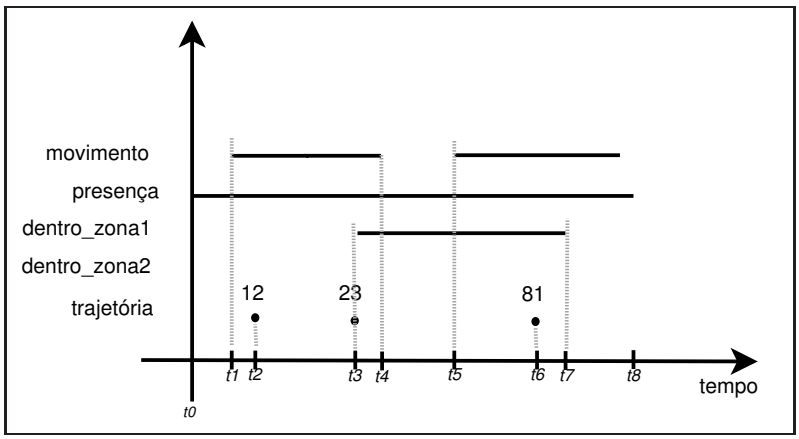


Figura 3.19: Exemplo de um Gráfico de Histórico relacionando atividades à um ator.

A cada quadro de vídeo, o sistema especialista verifica as regras das atividades simples, e, quando as condições e restrições são avaliadas, pode-se inferir a ocorrência da atividade em certo instante do tempo. Então, quando o sistema detecta uma atividade simples ao longo do tempo, ela é registrada no gráfico de histórico, preenchendo-se os valores da respectiva

variável. Quando a atividade simples deixa de existir, o valor da variável torna-se indefinida. Portanto, a variável terá ao longo do tempo vários valores que permanecem em intervalos de tempo. No exemplo de automóvel, esse módulo receberá dados a cada quadro do vídeo como identificador, centroide, retângulo delimitador (*bounding box*), silhueta, velocidade, direção do movimento e classificação (automóvel, em particular). A partir dessas informações, este módulo poderá detectar para o cenário em particular:

- Atividades simples de um simples objeto: são atividades que dependem apenas do próprio objeto. Exemplos: apareceu, desapareceu, movendo, parado etc;
- Atividades simples com múltiplos objetos: são atividades geradas a partir da relação do objeto com os elementos contextuais do cenário ou com outros objetos móveis. Exemplos: dentro de uma zona, movendo em direção a uma zona específica, objeto se move em direção a outro objeto, objeto colide com outro objeto.

Por exemplo, quando o módulo detecta inicialmente que um objeto está dentro de uma zona *x*, pode-se pensar numa linha que começa a ser desenhada no gráfico na variável *dentro de zona x*. Quando o objeto ao longo de seu movimento sai da zona, a linha desenhada é interrompida.

Em resumo, para criar o Gráfico de Histórico, é necessário ter conhecimento das entidades monitoradas criando uma variável para esta entidade e as respectivas atividades associadas. Então, o especialista precisa ter conhecimento das atividade simples de interesse e os valores que assumem ao longo do tempo. Para o preenchimento dos valores, é necessário o sistema especialista detectar as atividades simples, inferindo então uma mudança na variável e inserir esta mudança nos gráficos. Para acessar a informação dos gráficos, pode-se navegar pelo tempo ou pelas variáveis.

3.6 Arquitetura de software

As seções anteriores se concentraram em discutir conceitos de representação e detecção de atividades. Nesta seção, é discutida uma arquitetura de software que objetiva utilizar os módulos de baixo nível de processamento em vídeo juntamente com módulo de interpretação

de eventos de alto nível para a detecção de atividades pré-definidas em vídeos. Na Figura 3.20, é ilustrada a arquitetura do sistema proposto.

Os módulos da arquitetura proposta são os seguintes:

1. *Leitor de fonte de dado*: esta entidade abstrai os leitores de dados de baixo nível. *Fonte de Vídeo do Ambiente*, por exemplo, se conecta à fonte de vídeo e disponibiliza os quadros do vídeo para os sensores. Isto facilita a mudança na fonte de vídeo, que pode ser provida de uma câmera em tempo real, ou de um vídeo *offline*. Já se a informação de áudio for de interesse para a detecção de eventos, o *Fonte de Audio do Ambiente* pode ser utilizado, pois ele se conecta a uma fonte de áudio e fornece fluxos de áudio para análise.

2. *Sensores do ambiente*: estes sensores têm por função tratar dados de baixo nível, extraindo informações úteis e adicionando fatos ao ambiente CLIPS, a partir dos dados fornecidos pelos leitores de fonte de dados. Por exemplo, o sensor de estado de semáforo monitora os *pixels* de uma área pré-definida do vídeo onde se localiza o semáforo. Quando o sensor detecta uma mudança no histograma de cores na área do semáforo que corresponda a um dos estados possíveis (verde, amarelo ou vermelho), esta informação é atualizada na base de fatos. O sensor de elementos móveis em cena engloba todos os módulos discutidos no Capítulo 2, como subtração do fundo de cena, rastreio e classificação de objetos. Este sensor atribui um identificador e uma classificação para cada objeto rastreado e mantém a base de fatos atualizada. Foi implementado um módulo de distinção entre pessoas, carros e grupos de pessoas baseados no trabalho de Dedeoglu [Dedeoglu et al., 2006].

3. *Ambiente CLIPS*. Os fatos representam o estado atual do cenário (posições dos elementos móveis e dos elementos contextuais, além dos eventos que ocorreram) enquanto que a máquina de inferência do CLIPS trata de gerar os eventos que ocorrem a depender da dinâmica do cenário.

4. *Gerador de alarmes*. Este módulo captura os eventos gerados pelo CLIPS gerando ações. Por exemplo, uma ação pode ser descrever em vídeo os eventos que ocorrem, ou poderá acionar um alarme sonoro para um humano.

O sistema foi implementado com os módulos integrados de tal forma que a passagem de parâmetros entre eles se dá através de chamadas de funções. Os leitores de fonte de dados e os sensores são implementados como *threads* sincronizadas por semáforos, a fim de assegurar uma independência entre os módulos e evitar problemas quando o processamento de um módulo levar mais tempo que o de outro. O ambiente CLIPS é acessado através de uma API (*Application Programming Interface*) própria, possibilitando a adição ou remoção de fatos através de chamadas de funções. Embora a implementação de funções para a detecção de atividades simples pudesse ser realizada utilizando funções em CLIPS, decidiu-se por implementar essas funções como código C++, por questões de eficiência e de um maior acervo de bibliotecas para processamento de imagens e vídeo na linguagem C++.

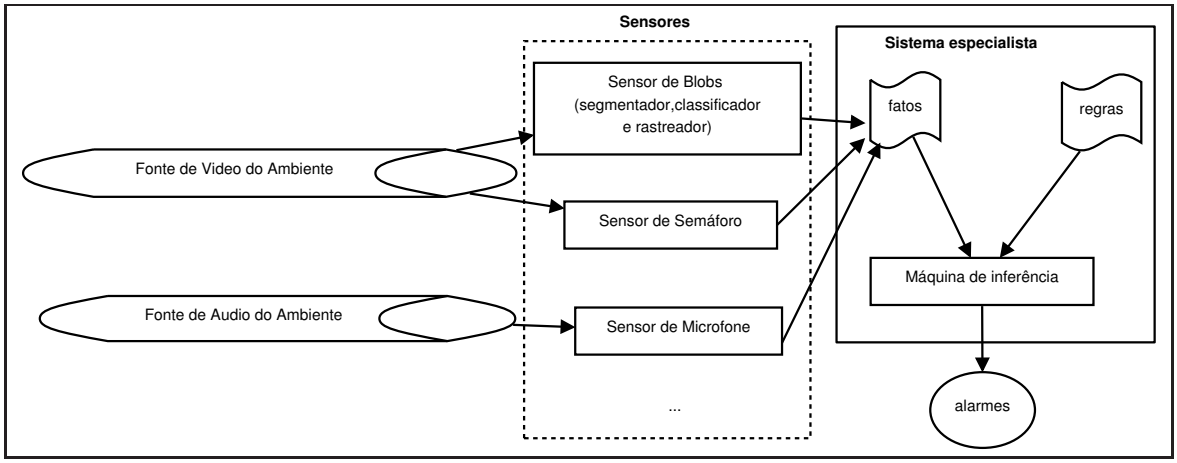


Figura 3.20: Fluxograma da arquitetura do sistema.

Essa arquitetura recebe como entrada dados sobre o rastreio e a classificação dos objetos móveis, dados sobre os elementos contextuais de interesse da cena, como também regras de atividades de interesse pré-definidas. Ao modelar um problema de uma instância em particular de um cenário, um especialista humano necessita seguir um conjunto de passos para a detecção de atividades pré-definidas. O fluxo das etapas é ilustrada na Figura 3.21 e consiste nas seguintes tarefas:

1. Identificação de elementos contextuais e atividades simples do cenário de interesse (tráfego de automóveis, sinal de trânsito, corredor de shopping, estacionamentos, monitoramento de carga/descarga);

2. Identificação de atividades compostas do respectivo cenário a serem detectadas.

3. Elaboração manual de regras em CLIPS para detecção de atividades compostas pré-definidas. Caso a atividade composta já tenha sido definida para outro cenário, usar a regra da biblioteca de regras. Se não, a regra criada pode ser mantida numa biblioteca de regras.

4. Definição de fatos em CLIPS dos elementos contextuais estáticos do cenário, a depender do posicionamento da câmera de cada vídeo;

5. Implementação de funções de detecção de atividades simples. Caso haja uma função na biblioteca de funções, para detecção da atividade simples, não há necessidade de criar uma nova função.

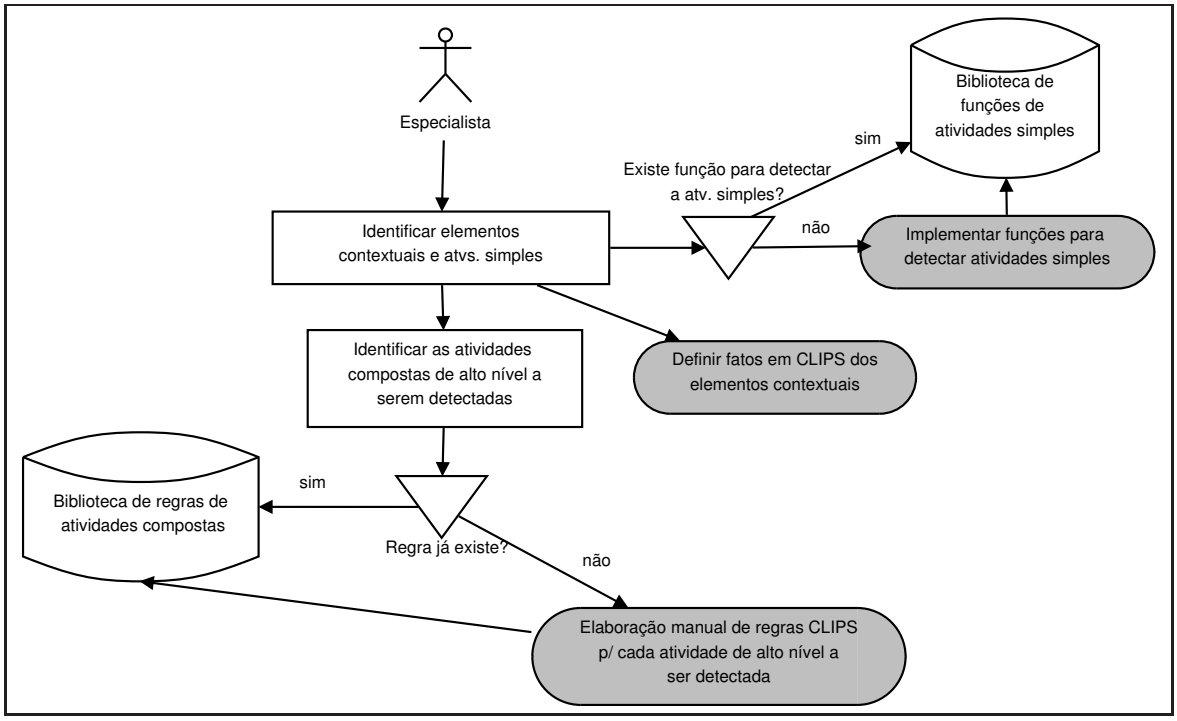


Figura 3.21: Fluxograma da fase de modelagem de atividades.

3.6.1 Detalhes de implementação

Como ambiente de desenvolvimento, foi utilizado o sistema operacional Ubuntu 9.10 (baseado no Linux Kernel versão 2.6.31), a plataforma Eclipse (versão Galileo), juntamente com o pacote (plugin) CDT (C++ Develpment Tool), para desenvolvimento de aplicações

em linguagem C++. A linguagem C++ foi escolhida para as atividades de implementação da arquitetura por ser uma linguagem compatível com as bibliotecas que foram utilizadas e por ser mais flexível que a linguagem C.

A biblioteca GNU C Library [1] foi utilizada provendo operações básicas como operações com *strings* e impressão dos resultados na saída padrão. Os sensores de **blobs** e **semáforo** foram implementados como *threads*, pois compartilham o mesmo recurso que é a fonte de vídeo. Assim, a biblioteca POSIX Threads [2] foi utilizada pois provê operações com threads.

O segmentador de movimento, rastreador e classificador de objetos na arquitetura proposta são encapsulados como um sensor de *blobs*. O segmentador e o rastreador realizam chamadas à biblioteca Opencv (versão 1.0) [3], que fornece implementações de rastreio (uma versão do algoritmo *Mean-shift*) e segmentador de movimento (baseado no trabalho [Li et al., 2003]). Para o classificador de objetos, por sua simplicidade e resultados práticos, foi implementada a técnica de Dedeoglu [Dedeoglu et al., 2006], para detectar as classes humanos, grupo de humanos e veículos.

A partir dos resultados dos sensores, como blobs em movimento e o estado do semáforo, a cada quadro do vídeo, são realizadas chamadas à uma interface que utiliza CLIPS. A ligação entre os sensores e o sistema CLIPS é realizada através de um módulo que utiliza a interface provida pelo código CLIPS [4]. Neste módulo, são realizadas operações de escrita e leitura dos fatos da base de conhecimento, como também é neste módulo que são implementadas as funções do usuário incrementado assim as funções existentes em CLIPS. Uma das funções adicionais que foi implementada é a detecção de intersecção entre polígonos, implementada pela função *inside*. Esta função realiza chamadas à biblioteca *General Polygon Clipper library* (GPC) [5], que provê uma gama de operações entre polígonos.

As regras, funções e definições de elementos, de atividades simples, as funções para as relações temporais e os fatos iniciais sobre o posicionamento dos elementos do contexto do cenário, definidas ao longo deste capítulo, foram implementadas diretamente em CLIPS (exceto a função *inside*). Para isto, foi elaborado um arquivo de texto com as definições de

[1] http://www.gnu.org/software/libc/

[2] https://computing.llnl.gov/tutorials/pthreads/

[3] http://sourceforge.net/projects/opencvlibrary/

[4] http://clipsrules.sourceforge.net/

[5] http://www.cs.man.ac.uk/ toby/alan/software/

regras e funções, e para cada vídeo de teste, foi elaborado um arquivo com os fatos sobre o posicionamento dos elementos contextuais. Esses arquivos são acessados pelo módulo de interface com CLIPS discutido no parágrafo anterior.

Para a geração de vídeos sintéticos, foi utilizado um software de criação de animações vetoriais em 2D, chamado Synfig Studio (versão 0.61).

3.6.2 Execução do sistema

A execução do sistema se dá através da linha de comando descrita no Código 3.17.

Código Fonte 3.17: Linha de comando de execução do sistema

```
1 obj_det <engine> <rules> <facts> <video>
```

Os parâmetros que o programa recebe são os seguintes:

- <engine>: nome de arquivo de regras em CLIPS onde está implementado o raciocínio temporal;
- <rules>: nome de arquivo de regras em CLIPS onde estão implementadas as regras de alarmes para o vídeo a ser analisado;
- <facts>: nome de arquivo de fatos em CLIPS onde estão descritos os elementos do cenário no vídeo;
- <video>: nome de um arquivo de vídeo a ser processado.

De posse dos parâmetros, o sistema é executado e em uma janela é exibido o vídeo com possíveis alarmes gerados. Os alarmes também são registrados na saída padrão (console) do ambiente de execução.

3.7 Conclusões

Nesta seção foi apresentada uma proposta de representação e de detecção de eventos pré-definidos em vídeo. Quando o objetivo da aplicação e o cenário não explicitam a forma dos

eventos, uma abordagem não-supervisionada em que os eventos devem ser encontrados automaticamente é necessária. Neste caso, não há um especialista que definirá os eventos: o próprio sistema deverá se encarregar de criar os tipos de eventos quando julgar necessário. Necessitaria de uma considerável quantidade de amostras de vídeos com diversas situações possíveis, se tornando impraticável, além do que os eventos sendo pré-definidos haverá um melhor controle das situações possíveis descritas anteriormente. Um dos objetivos desta seção foi apresentar um método para um especialista descrever de maneira clara e intuitiva as ações de interesse num monitoramento de vídeo. Outro objetivo foi propôr um método para reconhecer os eventos simples e o compostos, descritos por um especialista. Num primeiro momento, justificou-se a utilização em paralelo de informações quantitativas e qualitativas como uma forma de facilitar a descrição dos eventos. Logo após, foram descritos os conceitos utilizados e a forma de representação no sistema especialista CLIPS. Para a detecção, foi sugerido o mapeamento temporal dos eventos reconhecidos num gráfico. E com a análise temporal deste gráfico é possível inferir as relações temporais entre os eventos de uma maneira simples.

Capítulo 4

Avaliação da Abordagem Implementada

O alvo deste trabalho é desenvolver um ferramental para um especialista poder descrever regras e a partir delas, detectar atividades para um cenário específico, como o de um cruzamento de trânsito. A técnica proposta envolve primeiramente a detecção de atividades simples para, em seguida, serem detectadas as atividades compostas. No capítulo anterior, foram estudadas diversas formas de representação qualitativa e quantitativa que podem ser utilizadas como informações para a representação dos elementos móveis na cena e suas respectivas atividades no cenário. Como sugestão, este trabalho propôs implementar as atividades simples *dentro de zona de interesse* e *rente a objeto*. Outras atividades como trajetória, movimentação, relações topológicas não foram utilizadas pois apresentaram problemas principalmente em relação à falta de robustez do rastreio e à complexidade de implementação. Assume-se neste trabalho uma segmentação e rastreio perfeitos, isto é, não foi sugerido algoritmo de correção de erros de alto nível. Porém, as atividades simples implementadas são o suficiente para a demonstração da técnica proposta e a execução de alguns exemplos práticos. Além das atividades simples *dentro de zona* e *rente a objeto*, foi implementada também a atividade simples relativa aos semáforos de trânsito: o estado do sinal verde, amarelo ou vermelho.

Num primeiro momento, foi gerado um vídeo sintético em duas dimensões, em que dois círculos móveis percorrem um cenário com quatro zonas de interesse. A confecção desse vídeo objetivou exemplificar importantes relações temporais envolvendo as zonas e os objetos móveis, como também foi tomado o cuidado para que os objetos móveis fossem corretamente rastreados.

Para os experimentos, um humano rotulou as atividades envolvidas para posterior comparação com a detecção do sistema. Em seguida, foram testadas as regras temporais que envolvem duas atividades. Posteriormente, algumas regras mais complexas foram desenvolvidas e, logo após, as regras para atividades compostas foram analisadas. Na última seção, a técnica proposta é testada num experimento envolvendo um cenário semi-sintético, o qual também permitiu avaliar regras envolvendo abordagem de pessoa a veículos e infrações em semáforos. Em função das dificuldades mencionadas acima em relação à falta de robustez do algoritmo de segmentação de movimento em vídeos reais, os experimentos envolvendo cenários reais foram deixados como trabalho futuro.

4.1 Testes de regras

Alguns testes foram conduzidos a fim de aferir os acertos (exatidão) das regras. Como as atividades envolveram objetos móveis em duas dimensões (2D) e sua interação com o ambiente, foram realizados experimentos com vídeos nos quais objetos se locomovem entre elementos do cenário.

Na primeira subseção, foram testadas as regras de atividades simples. O seu teste é importante pois as regras para atividades compostas as utilizam. Em seguida, foram testadas regras com relações temporais e na última subseção foram testadas as atividades compostas.

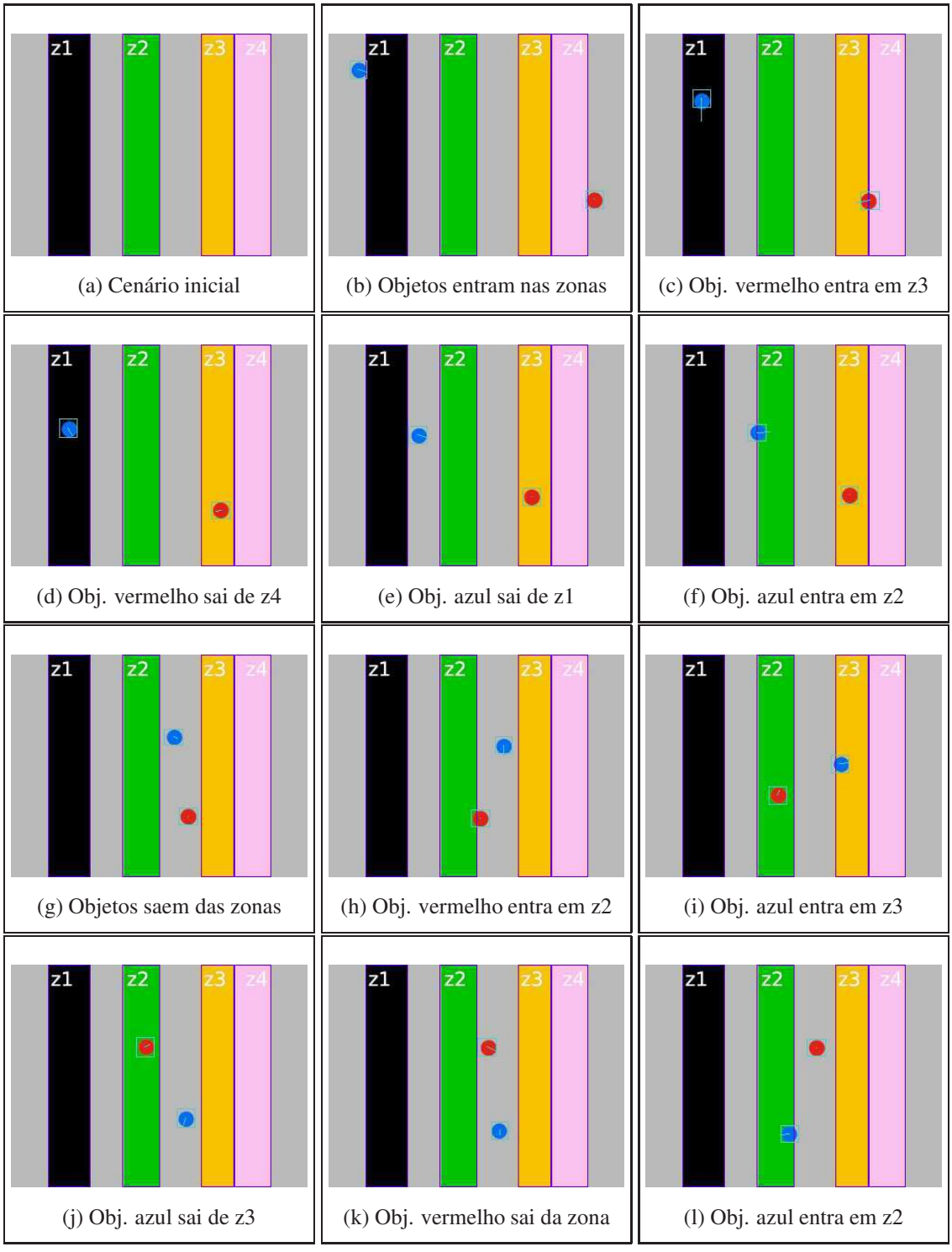
4.1.1 Análise da detecção de atividades simples

Como uma forma de exemplificar o funcionamento de regras, foi implementada em CLIPS a regra de atividade simples *dentro de zona*. A álgebra de Allen é uma teoria para a representação das relações entre duas entidades temporais. Então, as especificações de regras temporais envolveram relações binárias entre duas atividades simples.

Nas Figuras 4.1 e 4.2, são apresentados quadros-chave de vídeo de um cenário hipotético no qual dois objetos móveis se locomovem entre quatro zonas de interesse.

Na Figura 4.3, são ilustradas as atividades simples ocorridas ao longo do tempo rotuladas por um humano, enquanto que na Tabela 4.1, são descritos os dados da Figura 4.3.

Na Figura 4.4, é ilustrado o gráfico de histórico das atividades simples ocorridas ao longo do tempo, as quais foram detectadas pelo sistema.



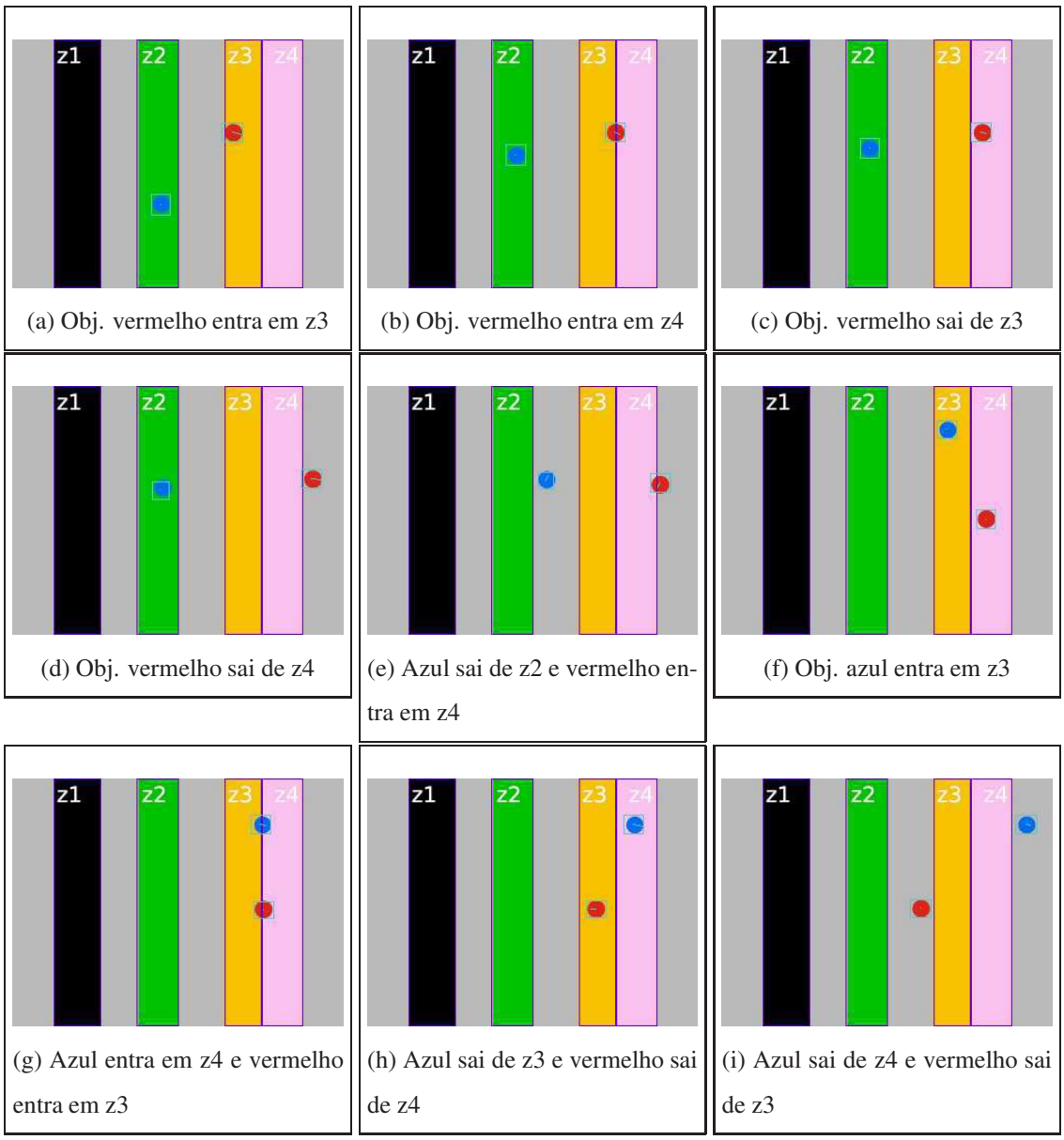
(a) Cenário inicial (b) Objetos entram nas zonas (c) Obj. vermelho entra em z3

(d) Obj. vermelho sai de z4 (e) Obj. azul sai de z1 (f) Obj. azul entra em z2

(g) Objetos saem das zonas (h) Obj. vermelho entra em z2 (i) Obj. azul entra em z3

(j) Obj. azul sai de z3 (k) Obj. vermelho sai da zona (l) Obj. azul entra em z2

Figura 4.1: Quadros-chave das mudanças de zonas de dois círculos coloridos num vídeo sintético. São 7 entradas e 7 saídas de zonas do círculo vermelho e 6 entradas e 6 saídas de zonas do círculo azul, o que corresponde ao total de 26 entradas e saídas.



(a) Obj. vermelho entra em z3

(b) Obj. vermelho entra em z4

(c) Obj. vermelho sai de z3

(d) Obj. vermelho sai de z4

(e) Azul sai de z2 e vermelho entra em z4

(f) Obj. azul entra em z3

(g) Azul entra em z4 e vermelho entra em z3

(h) Azul sai de z3 e vermelho sai de z4

(i) Azul sai de z4 e vermelho sai de z3

Figura 4.2: Continuação dos quadros-chave da Figura 4.1.

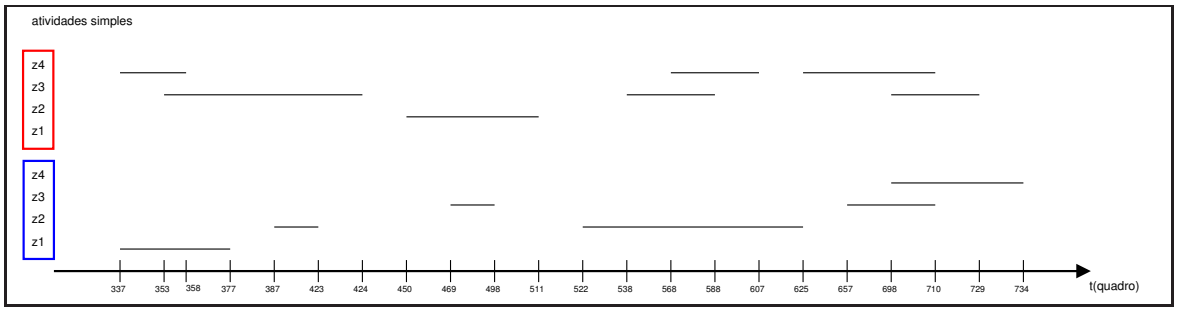


Figura 4.3: Resultado da análise visual humana das atividades simples em vídeo.

Tabela 4.1: Resultado dos intervalos na análise visual humana das atividades simples em vídeo.

Objeto	**Atividade**	**Intervalos de Tempo**
azul	dentro de z1	[(337; 377)]
azul	dentro de z2	[(387; 423), (522; 625)]
azul	dentro de z3	[(469; 498), (657; 710)]
azul	dentro de z4	[(698; 734)]
vermelho	dentro de z2	[(450; 511)]
vermelho	dentro de z3	[(353; 424), (538; 588), (698; 729)]
vermelho	dentro de z4	[(337; 358), (568; 607), (625; 710)]

A execução do sistema para o vídeo do exemplo envolve chamadas de funções das atividades simples que utilizam as informações do rastreio e da classificação dos objetos. A detecção da atividade simples “dentro de zona” envolve a análise da intersecção dos polígonos que representam os objetos estáticos do cenário com o polígono que representa os objetos móveis. Então, assumindo que a função de interseção entre polígonos está corretamente implementada, a detecção das atividades simples e seu registro no histórico de atividades simples dependerão apenas do rastreio e da classificação dos objetos. Na Tabela 4.2, está o resultado da detecção das atividades simples pelo sistema.

Comparando os dados fornecidos por um humano na Tabela 4.1 com as atividades identificadas pelo sistema na Tabela 4.1, constata-se que as atividades foram detectadas pelo sistema, exceto pela exatidão dos intervalos de tempo. Porém, a diferença máxima, no que se refere aos extremos dos intervalos dada a observação humana e a detecção pelo sistema,

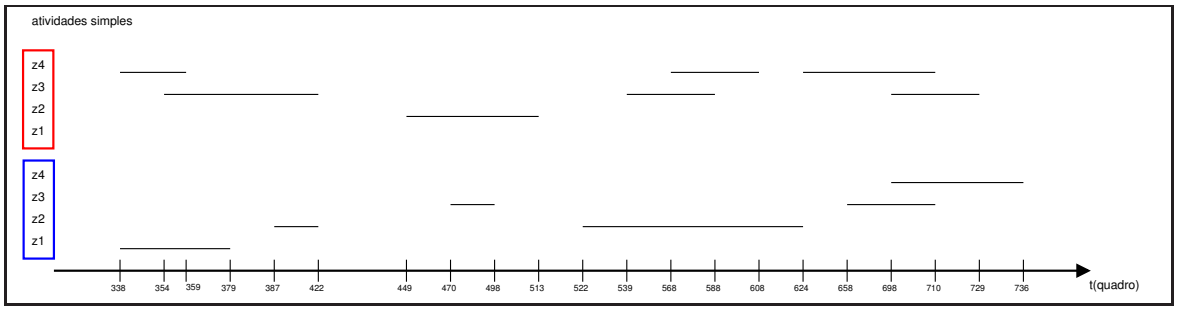


Figura 4.4: Resultado da detecção pelo sistema das atividades simples em vídeo.

Tabela 4.2: Resultado dos intervalos na detecção das atividades simples em vídeo pelo sistema.

Objeto	**Atividade**	**Intervalos de Tempo**
azul	dentro de z1	[(338; 379)]
azul	dentro de z2	[(387; 422), (522; 625)]
azul	dentro de z3	[(470; 498), (658; 710)]
azul	dentro de z4	[(698; 736)]
vermelho	dentro de z2	[(449; 513)]
vermelho	dentro de z3	[(354; 422), (539; 588), (699; 729)]
vermelho	dentro de z4	[(338; 359), (568; 608), (624; 710)]

foi de apenas dois quadros, como é o caso do tempo final da atividade “dentro de z1” do objeto azul (377 contra 379, respectivamente). Isto se deve ao pré-processamento e o rastreio do objeto móvel, que na atual implementação do sistema pode detectar uma extensão do objeto que se move, como um ligeiro rastro. A exatidão dos intervalos de tempo é importante na análise da álgebra temporal de Allen, já que é um cálculo sensível em relação aos extremos dos intervalos temporais. Já em respeito às relações temporais, percebe-se que apenas uma relação foi definitivamente afetada pela imprecisão do rastreio, que foi a relação das primeiras ocorrências das atividades “objeto vermelho na zona z3” e “objeto azul na zona z2”, em que o humano identificou uma relação de “durante” enquanto o sistema identificou uma relação de “finaliza”. Na Figura 4.5, é ilustrada a única relação temporal afetada pelo rastreio, comparando a obervação humana e as atividades detectadas pelo sistema.

Duas regras foram criadas para alarmar quando um objeto entrar numa zona ou quando

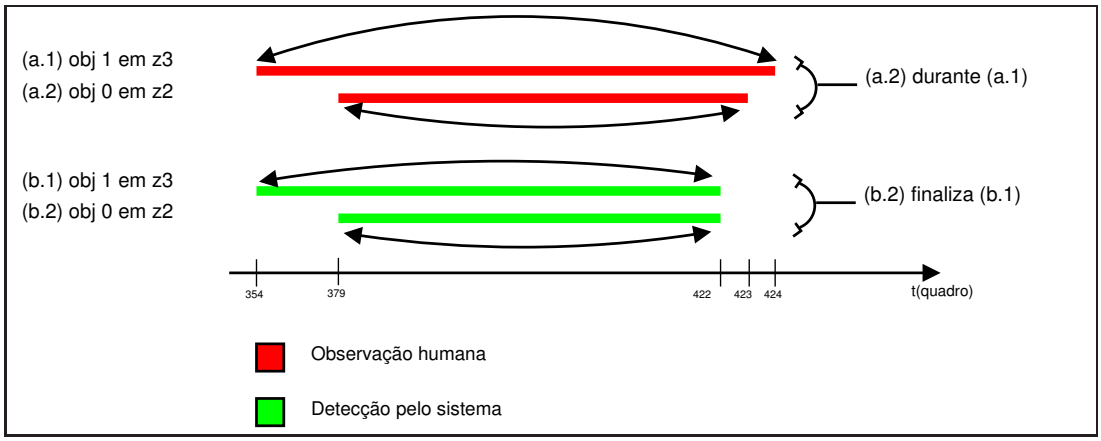


Figura 4.5: Comparação entre a observação humana e a detecção do sistema. Por causa de inexatidão no módulo de rastreio, enquanto que para a visão humana a relação é objeto 0 em zona z2 durante objeto 1 em zona z3, o sistema detectou objeto 0 em zona z0 finaliza objeto 1 em zona z3.

ele sair da zona. O código em CLIPS da regra para detectar quando um objeto entrar numa zona é apresentada no Código 4.1. Nesta regra, tem-se:

- a expressão *(declare (salience -1))* significa que a regra só é analisada pela máquina de inferência CLIPS após as regras de atividades simples serem analisadas previamente;
- *(tempo ?now)* captura o tempo atual em ?now;
- *(historic (nameOBJ ?obj) (nameACT ?act) (timepoints $?times1))* irá associar as atividades simples referentes ao objeto ?obj;
- *(test (eq (inicia ?now $?times1) 1))* irá testar se no tempo atual (lembrando que o sistema é executado em tempo-real) a atividade simples é iniciada.

O código em CLIPS da regra para detectar quando um objeto sai de uma zona é apresentado no Código 4.2. A diferença para a regra de detectar a entrada em zonas descrita no Código 4.1 reside na linha 5 dos dois códigos. Na regra *entrar em zona*, *(eq (inicia $?times1 ?now) 1)* testa se no tempo atual a atividade iniciou. Na regra *sair de zona*, *(eq (encontra $?times1 ?now) 1)* testa se no tempo atual (lembrando que o sistema é executado em tempo-real) a atividade simples deixou de existir.

Código Fonte 4.1: Regra para identificar quando um objeto entrar em zonas

```
1 (defrule regraEntrouZonas
2 (declare (salience -1))
3 (tempo ?now)
4 (historic (nameOBJ ?obj) (nameACT ?act) (timepoints $?times1))
5 (test (eq (inicia ?now $?times1) 1))
6 =>
7 (printout t "Alarme: obj" ?obj " entrou em " ?act " no quadro " ?now crlf
      )
8 )
```

Código Fonte 4.2: Regra para identificar quando um objeto sair de zonas.

```
1 (defrule regraSaiuZonas
2 (declare (salience -1))
3 (tempo ?now)
4 (historic (nameOBJ ?obj) (nameACT ?act) (timepoints $?times1))
5 (test (eq (encontra $?times1 ?now) 1))
6 =>
7 (printout t "Alarme: obj " ?obj " saiu de " ?act " no quadro " ?now crlf)
8 )
```

O *dump* apresentado no Código 4.3 mostra o resultado das gerações de alarmes para as regras de entrar em zona e sair de zona, respectivamente apresentadas no Códigos 4.1 e 4.2. Pode-se perceber que a entrada em zonas coincide com o início de intervalo de tempo analisado na Tabela 4.2, enquanto que a saida de zona acontece exatamente um quadro após o fim do intervalo de tempo, pois no instante anterior o objeto ainda estava dentro de zona. Ao todo são 26 alarmes, o que coincide com a quantidade de mudanças de zonas dos círculos vermelho e azul ilustrados anteriormente nas Figuras 4.1 e 4.2.

Código Fonte 4.3: Resultado dos disparos das regras de entrar e sair de zonas.

```
1  Alarme: obj blob_1 entrou em dentro_zona z4 no quadro 338
2  Alarme: obj blob_0 entrou em dentro_zona z1 no quadro 338
3  Alarme: obj blob_1 entrou em dentro_zona z3 no quadro 354
4  Alarme: obj blob_1 saiu de dentro_zona z4 no quadro 360
5  Alarme: obj blob_0 saiu de dentro_zona z1 no quadro 380
6  Alarme: obj blob_0 entrou em dentro_zona z2 no quadro 387
7  Alarme: obj blob_1 saiu de dentro_zona z3 no quadro 423
8  Alarme: obj blob_0 saiu de dentro_zona z2 no quadro 423
9  Alarme: obj blob_1 entrou em dentro_zona z2 no quadro 449
10 Alarme: obj blob_0 entrou em dentro_zona z3 no quadro 470
11 Alarme: obj blob_0 saiu de dentro_zona z3 no quadro 499
12 Alarme: obj blob_1 saiu de dentro_zona z2 no quadro 514
13 Alarme: obj blob_0 entrou em dentro_zona z2 no quadro 522
14 Alarme: obj blob_1 entrou em dentro_zona z3 no quadro 539
15 Alarme: obj blob_1 entrou em dentro_zona z4 no quadro 568
16 Alarme: obj blob_1 saiu de dentro_zona z3 no quadro 589
17 Alarme: obj blob_1 saiu de dentro_zona z4 no quadro 609
18 Alarme: obj blob_1 entrou em dentro_zona z4 no quadro 624
19 Alarme: obj blob_0 saiu de dentro_zona z2 no quadro 624
20 Alarme: obj blob_0 entrou em dentro_zona z3 no quadro 658
21 Alarme: obj blob_0 entrou em dentro_zona z4 no quadro 698
22 Alarme: obj blob_1 entrou em dentro_zona z3 no quadro 699
23 Alarme: obj blob_1 saiu de dentro_zona z4 no quadro 711
24 Alarme: obj blob_0 saiu de dentro_zona z3 no quadro 711
25 Alarme: obj blob_1 saiu de dentro_zona z3 no quadro 730
26 Alarme: obj blob_0 saiu de dentro_zona z4 no quadro 737
```

4.1.2 Análise de regras temporais

As especificações das regras temporais, como já discutidas, envolverão duas entidades temporais, que neste trabalho serão as atividades simples. As análises das regras temporais nesta subseção utiliza como base de dados o exemplo descrito na seção anterior, com o gráfico de atividades ilustrado na Figura 4.4.

Sobrepõe Um exemplo de regra para *A sobrepõe B* é apresentada no Código 4.4. Este exemplo é para identificar ocorrências da relação **sobrepõe** entre atividades realizadas por um mesmo objeto. Nesta regra, a função *sobrepoe_index* recebe os parâmetros dos objetos, das atividades simples e dos índices para as atividades.

Código Fonte 4.4: Regra para detecção da relação sobrepoe.

```
1  (defrule sobrepoe1
2  (declare (salience -1))
3  (tempo ?now)
4  (blob (name ?obj))
5  (historic (nameOBJ ?obj) (nameACT ?atv1) (timepoints $?times1))
6  (historic (nameOBJ ?obj) (nameACT ?atv2) (timepoints $?times2))
7  (test (eq (- ?now (nth$ 1 $?times2)) 1))
8  (test (sobrepoe_index ?obj ?atv1 1 ?obj ?atv2 1))
9  =>
10 (printout t "Alarme: Para obj " ?obj ", " ?atv1 " sobrepoe " ?atv2 " em t
       =" ?now crlf)
11 )
```

A partir da execução do sistema com a regra apresentada no Código 4.4, foram gerados os quatro alarmes listados no Código 4.5. Percebe-se que na listagem 4.5, apenas uma relação de sobrepõe entre atividades foi detectada para o objeto blob_0 (círculo azul), e três foram detectadas para o objeto blob_1 (círculo vermelho), o que pode ser constatado no gráfico de atividades temporais da Figura 4.6.

Inicia Outro exemplo testado é a regra para *A inicia B*, apresentada no Código 4.6. Este exemplo é para identificar ocorrências da relação **inicia** entre atividades realizadas por um

Código Fonte 4.5: Resultado para detecção das relações sobrepõe de um objeto.

```
1 Alarme: Para obj blob_1, dentro_zona z4 sobrepoe dentro_zona z3 em t=423
2 Alarme: Para obj blob_1, dentro_zona z3 sobrepoe dentro_zona z4 em t=609
3 Alarme: Para obj blob_1, dentro_zona z4 sobrepoe dentro_zona z3 em t=730
4 Alarme: Para obj blob_0, dentro_zona z3 sobrepoe dentro_zona z4 em t=737
```

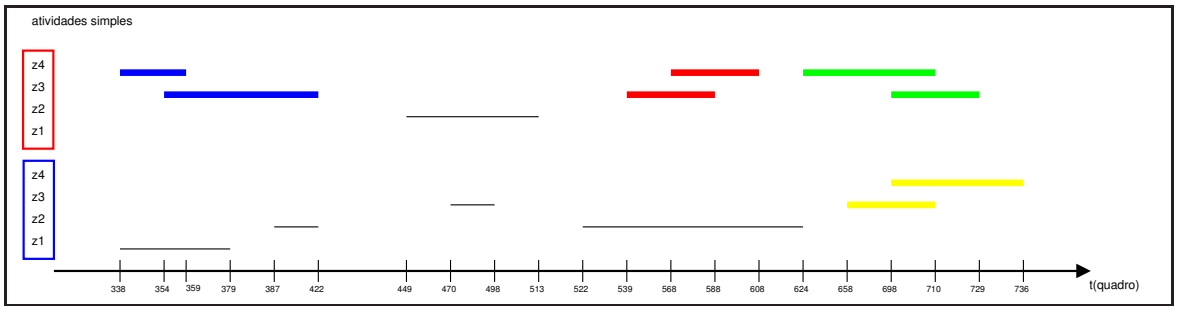


Figura 4.6: Resultado da detecção, com os intervalos espessos de mesma cor correspondendo às atividades que têm uma relação sobrepõe entre si.

ou dois objetos. A partir da execução do sistema desta regra, foi gerado o alarme listado em 4.7, a partir do qual pode ser constatado a coerência com no gráfico de atividades temporais da Figura 4.7.

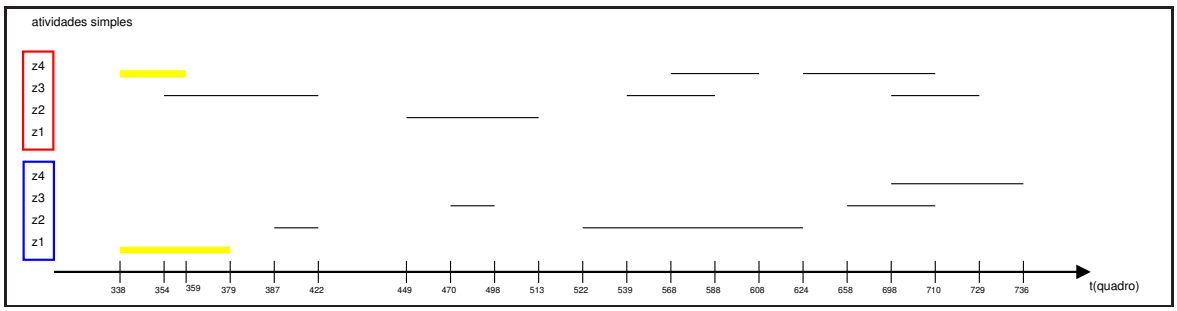


Figura 4.7: Resultado da detecção, com os intervalos espessos da cor amarela correspondendo às atividades que têm uma relação inicia entre si.

Código Fonte 4.6: Regra para detecção da relação inicia entre atvs. de um ou dois objetos.

```
1  (defrule inicia1
2  (declare (salience -1))
3  (tempo ?now)
4  (historic (nameOBJ ?obj1) (nameACT ?atv1) (timepoints $?times1))
5  (historic (nameOBJ ?obj2) (nameACT ?atv2) (timepoints $?times2))
6  (test (eq (- ?now (nth$ 1 $?times2)) 1))
7  (test (inicia_index ?obj1 ?atv1 1 ?obj2 ?atv2 1))
8  =>
9  (printout t "Alarme: " ?obj1 " em " ?atv1 " inicia " ?obj2 " em " ?atv2
       ", no tempo=" ?now crlf)
10 )
```

Código Fonte 4.7: Resultado para detecção da relação inicia de um ou dois objetos.

```
1  Alarme: blob_1 em dentro_zona z4 inicia blob_0 em dentro_zona z1, no
       tempo=380
```

Finaliza A partir da execução do sistema com a regra para *A finaliza B* para identificar ocorrências da relação **finaliza** entre atividades realizadas por um ou dois objetos, foram gerados os alarmes listados em 4.8 e ilustrados na Figura 4.8.

Código Fonte 4.8: Resultado para detecção da relação finaliza de um ou dois objetos.

```
1 Alarme: blob_0 em dentro_zona z2 finaliza blob_1 em dentro_zona z3, no
      tempo=423
2 Alarme: blob_0 em dentro_zona z3 finaliza blob_1 em dentro_zona z4, no
      tempo=711
```

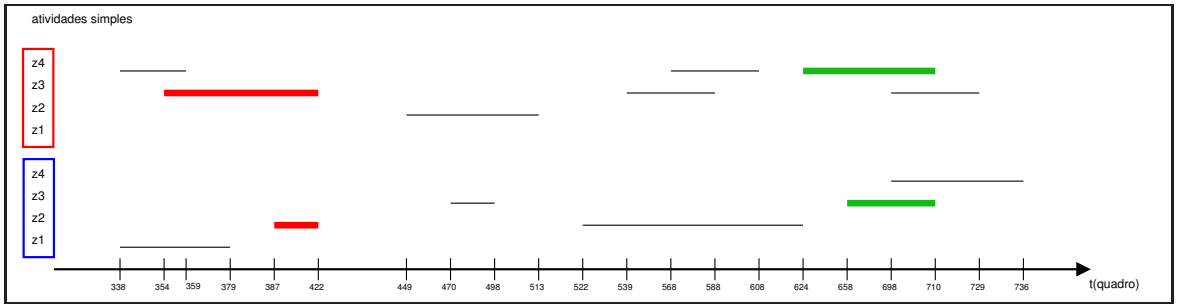


Figura 4.8: Os intervalos espessos de mesma cor (vermelhos e verdes) correspondem às atividades que têm uma relação finaliza entre si.

4.1.3 Teste de regras de atividades compostas

A análise da detecção de atividades compostas consistirá em duas etapas. Na primeira etapa 4.1.3(A), será considerado o mais recente índice do intervalo das atividades simples (igual a considerar o índice igual a 1). Na segunda etapa 4.1.3(B), serão conduzidos testes com utilização de índices recentes e anteriores, permitindo detectar situações além do último intervalo de uma atividade (será possível utilizar intervalos anteriores).

4.1.3(A) Análise de detecção de atividades compostas utilizando o mais recente intervalo de tempo das atividades simples

A primeira etapa, sem considerar os índices das atividades, consiste nos seguintes experimentos: 4.1.3(A1) detecção de uma atividade composta em particular, 4.1.3(A2) detecção de todas as atividades compostas que obedeçam a certa restrição.

4.1.3(A1) Detecção de uma atividade composta em particular. Um exemplo de atividade composta específica é a situação onde um objeto 0 está dentro de uma zona z2, enquanto outro objeto 1 atravessa outras duas zonas z3 e z4. A Figura 4.9 ilustra esta situação hipotética. Para a detecção desta situação, foi definida a regra apresentada no Código 4.9.

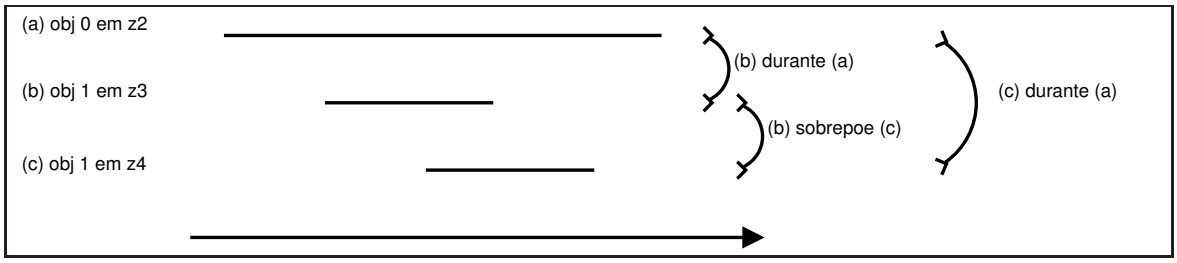


Figura 4.9: Atividade composta específica a ser detectada.

A execução do sistema resultou em apenas uma detecção no quadro de número 610: “enquanto obj 0 estava em z2, obj 1 entrou em z3 e atravessou para z4! t: 610”. A Figura 4.10 ilustra as atividades envolvidas neste alarme detectado. Comparando as relações entre os traços laranja desta figura com a Figura 4.9, conclui-se que o sistema detectou corretamente a situação desejada.

Código Fonte 4.9: Regra para detectar uma atividade composta específica.

```
1 (defrule sequencia_0z2_1z3_1z4
2 (declare (salience -1))
3 (tempo ?now)
4 (test (durante_index 1 "dentro_zona z3" 1 0 "dentro_zona z2" 1))
5 (test (sobrepoe_index 1 "dentro_zona z3" 1 1 "dentro_zona z4" 1))
6 (test (durante_index 1 "dentro_zona z4" 1 0 "dentro_zona z2" 1))
7 =>
8 (printout t "Alarme: enquanto obj 0 estava em z2, obj 1 entrou em z3 e
    atravessou para z4! t: " ?now crlf)
9 )
```

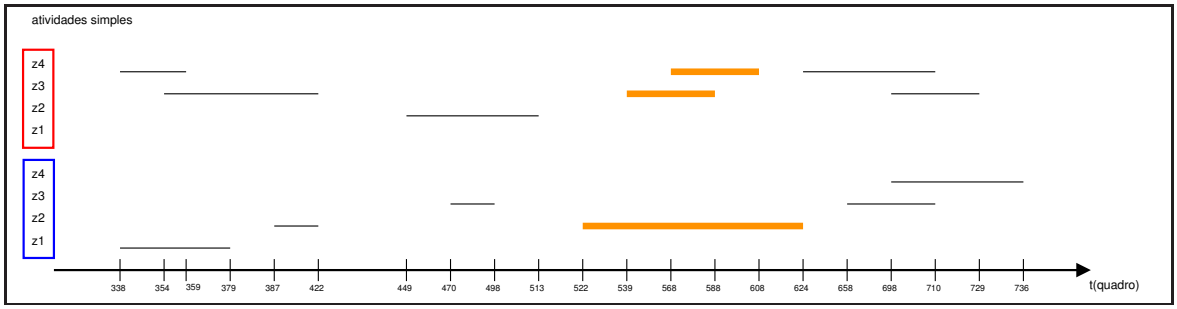


Figura 4.10: Os traçados em laranja representam as atividades relacionadas ao alarme.

4.1.3(A2) Detecção de todas as atividades compostas que obedeçam a certa restrição

Uma situação hipotética de interesse é saber enquanto uma atividade simples é executada, se pelo menos mais duas atividades simples ocorrem paralelamente. Esta atividade composta é ilustrada na Figura 4.11 e a regra correspondente é apresentada no Código 4.10. A diferença para a atividade composta descrita na Figura 4.9 é que os atores e as atividades simples não são especificados.

A execução do vídeo de exemplo com esta regra soou o alarme uma vez, ou seja, houve apenas uma situação onde a regra foi satisfeita. O alarme é descrito na Figura 4.12. Este resultado de um alarme genérico está de acordo com o que foi detectado na ilustração da Figura 4.10, ou seja, apenas uma ocorrência.

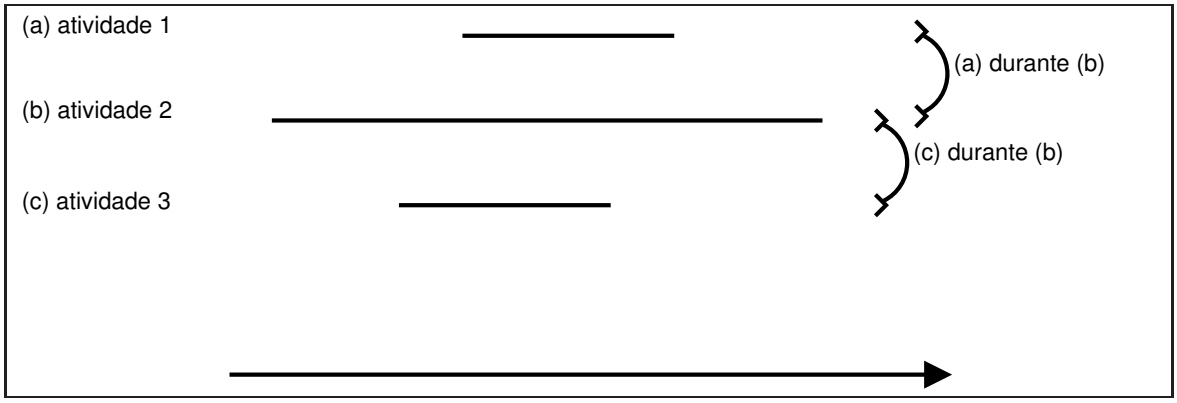


Figura 4.11: Atividade composta genérica a ser detectada.

```
Alarme: obj 1 em dentro_zona z4 durante obj 0 em dentro_zona z2
 e obj 1 em dentro_zona z3 durante obj 0 em dentro_zona z2. t: 609
```

Figura 4.12: Resultado da detecção de uma atividade composta genérica.

Código Fonte 4.10: Regra composta genérica para detectar 3 atividades.

```
1  (defrule durante_durante_teste
2  (declare (salience -1))
3  (tempo ?now)
4  (historic (nameOBJ ?obj1) (nameACT ?act_a) (timepoints $?times1))
5  (historic (nameOBJ ?obj2) (nameACT ?act_b) (timepoints $?times2))
6  (historic (nameOBJ ?obj3) (nameACT ?act_c) (timepoints $?times3))
7  (test (eq (- ?now (nth$ 1 $?times2)) 1))
8  (test (and (durante_index ?obj2 ?act_b 1 ?obj1 ?act_a 1) (durante_index ?
       obj3 ?act_c 1 ?obj1 ?act_a 1)))
9  =>
10 (printout t "Alarme: (durante, durante) obj " ?obj2 " em " ?act_b "
       durante obj " ?obj1 " em " ?act_a " e ")
11 (printout t "obj " ?obj3 " em " ?act_c " durante obj " ?obj1 " em " ?
       act_a ". t: " ?now crlf)
12 )
```

4.1.3(B)Análise de detecção de atividades compostas utilizando variados intervalos de tempo

A segunda etapa para os experimentos serão os testes sobre atividades compostas que envolvam índices. Isto significa que as regras terão maior poder de expressão, visto que uma mesma atividade simples em que houve sua ocorrência em vários intervalos de tempos passados (em relação ao tempo atual, já que o sistema é em tempo real) poderá ser usada não apenas com o seu mais recente intervalo, como também será possível usar regras que usam intervalos passados. Os experimentos serão realizados para a detecção de atividades compostas com variação no índice.

Sequência de ocorrêncas de uma atividade Uma situação em que o especialista pode estar interessado é saber quando uma mesma atividade é realizada duas vezes seguidas por um objeto. A regra é apresentada no Código 4.11 e o resultado encontra-se ilustrado da forma de alarmes textuais na Figura 4.13 e explicitado na Figura 4.14. Nesta Figura 4.14, estão explicitadas seis conjuntos (1 e 2), (2 e 3), (4 e 5), (5 e 6), (7 e 8) e (9 e 10) onde cada número representa uma atividade simples que foi rotulada para visualização. Por exemplo, os intervalos *1* e *2* representam a atividade simples *objeto vermelho dentro da zona z4* em tempos diferentes. Assim, a sequência de intervalos (1 e 2) nesta figura foi detectada pois a atividade simples *objeto vermelho dentro da zona z4* ocorreu duas vezes.

Código Fonte 4.11: Regra para identificar uma mesma atividade de um objeto sendo repetida.

```
1 (defrule seq_de_dois
2 (declare (salience -1))
3 (tempo ?now)
4 (historic (nameOBJ ?obj) (nameACT ?act_a) (timepoints $?times1))
5 (test (eq (- ?now (nth$ 1 $?times1)) 1))
6 (test (precede_index ?obj ?act_a 2 ?obj ?act_a 1))
7 =>
8 (printout t "Alarme: Sequencia de 2: obj: " ?obj "; atividade: " ?act_a
      crlf)
9 )
```

```
Alarme: Sequencia de 2: obj: 1; atividade: dentro_zona z3; t: 589
Alarme: Sequencia de 2: obj: 1; atividade: dentro_zona z4; t: 609
Alarme: Sequencia de 2: obj: 0; atividade: dentro_zona z2; t: 624
Alarme: Sequencia de 2: obj: 1; atividade: dentro_zona z4; t: 711
Alarme: Sequencia de 2: obj: 0; atividade: dentro_zona z3; t: 711
Alarme: Sequencia de 2: obj: 1; atividade: dentro_zona z3; t: 730
```

Figura 4.13: Resultado para regra composta genérica para identificar uma mesma atividade sendo realizada duas vezes por um objeto.

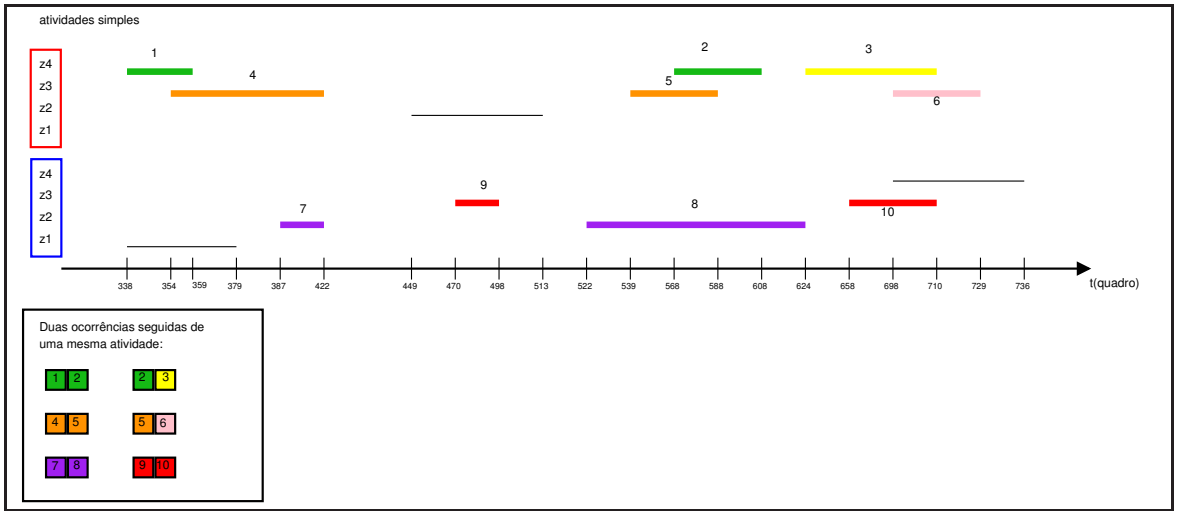


Figura 4.14: Resultado da regra para detectar quando há uma atividade simples sendo executada duas vezes seguidas por um mesmo objeto. Os conjuntos de intervalos (1 e 2), (2 e 3), (4 e 5), (5 e 6), (7 e 8), (9 e 10) foram detectados pois são atividades simples que se repetiram.

Sequência ABA Um exemplo também interessante é reconhecer a sequência de atividades [A B A]. Isto significa uma atividade simples A é intercalada por outra B. A regra para detectar esta situação é descrita no Código 4.12 e o resultado na execução do vídeo pelo sistema é apresentado no Código 4.13 e ilustrado na Figura 4.15. Nesta figura, as atividades simples foram rotuladas com números de 1 a 10, e foram exibidos 4 detecções de sequências ABA: (1, 5, 2), (3, 5, 4), (7, 9, 8) e (9, 8, 10). Por exemplo, a sequência (7, 9, 8) é a sequência de atividades simples *objeto azul dentro de z2*, *objeto azul dentro de z3*, *objeto azul dentro de z2*. Ou seja, a atividade simples *objeto azul dentro de z2* foi intercalada pela atividade *objeto azul dentro de z3*.

Código Fonte 4.12: Regra para identificar a sequência de atividade (ABA)

```
1  (defrule seq_aba
2  (declare (salience -1))
3  (tempo ?now)
4  (historic (nameOBJ ?obj) (nameACT ?act_a) (timepoints $?times1))
5  (historic (nameOBJ ?obj) (nameACT ?act_b) (timepoints $?times2))
6  (test (eq (- ?now (nth$ 1 $?times1)) 1))
7  (test (precede_index ?obj ?act_a 2 ?obj ?act_b 1))
8  (test (precede_index ?obj ?act_b 1 ?obj ?act_a 1))
9  =>
10 (printout t "Alarme: Sequencia [" ?act_a "," ?act_b "," ?act_a "]
       identificada. obj: " ?obj "; t: " ?now crlf)
11 )
```

Código Fonte 4.13: Resultado da identificação de sequência (ABA)

```
1 Alarme: Sequencia [dentro_zona z3,dentro_zona z2,dentro_zona z3]
      identificada. obj: 1; t: 589
2 Alarme: Sequencia [dentro_zona z4,dentro_zona z2,dentro_zona z4]
      identificada. obj: 1; t: 609
3 Alarme: Sequencia [dentro_zona z2,dentro_zona z3,dentro_zona z2]
      identificada. obj: 0; t: 624
4 Alarme: Sequencia [dentro_zona z3,dentro_zona z2,dentro_zona z3]
      identificada. obj: 0; t: 711
5 )
```

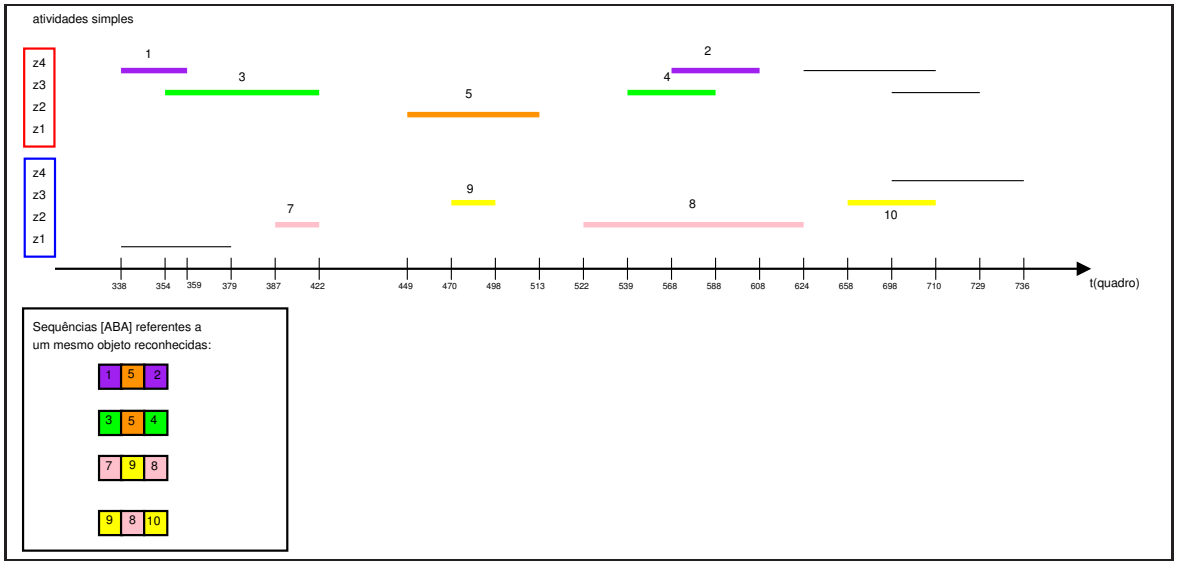


Figura 4.15: Resultado da regra de atividade composta numa sequência ABA, em um gráfico temporal. Os conjuntos de intervalos (1, 5, 2), (3, 5, 4), (7, 9, 8), (9, 8, 10) foram detectados.

4.2 Experimentos envolvendo um cruzamento de trânsito

Os experimentos com vídeos de cruzamento de trânsito foram realizados com um vídeo semi-sintético de um cenário onde algumas regras são testadas. A geração do vídeo semi-sintético foi realizada atraves de um software de animação em 2D, chamado *Synfig Studio*. Uma figura de um cruzamento de trânsito foi utilizada como fundo estático de cena. Foram utilizadas figuras de veículos como camadas que se movem ao longo da cena.

4.2.1 Violação de semáforo

Uma situação que pode ocorrer num cruzamento de trânsito é a violação de sinal de semáforo, ocorrendo quando o sinal está "fechado" e um automóvel trafega na direção proibida. As figuras 4.18 e 4.18 ilustram quadros-chave de um vídeo semi-sintético gerado para testar uma situação de violação de sinal. Se as zonas delimitadas forem rotuladas, o cenário de cruzamento deste vídeo sintético fica rotulado como ilustrado na Figura 4.16. O percurso de um automóvel neste exemplo envolve a travessia das zonas *zona1*, *zonaCentral* e *zona3*, respectivamente.

Figura 4.16: Um quadro de vídeo sintético delimitado por zonas.

De posse das informações das zonas e de posicionamento de automóvel, uma forma intuitiva de descrever uma violação de semáforo é: "quando o sinal está fechado e automóvel atravessa de zona1 para a zonaCentral". A Figura 4.17 ilustra graficamente as vinte e sete possíveis situações de violações de semáforo, considerando-se a álgebra de Allen. Por exemplo, na Figura 4.17(1), tem-se que o sinal vermelho acontece antes do veículo entrar na zona1

e termina antes do veículo deixar a zonaCentral, porém existe um intervalo que o automóvel não está em zona1 e nem em zonaCentral (este seria o caso em que o automóvel está inserido totalmente dentro da faixa de trânsito). Em termos das relações algébricas temporais, a violação do sinal de semáforo neste exemplo envolve as relações:

- automóvel dentro de zona1 *durante* sinal no vermelho e
- automóvel dentro de zona1 *precede* automóvel dentro de zonaCentral e
- sinal no vermelho *sobrepõe* automóvel dentro de zonaCentral.

Já na Figura 4.17(11), o sinal passa a ser vermelho ao mesmo tempo em que o automóvel entra em zona1, e o sinal deixa de ser vermelho ao mesmo tempo que o automóvel sai de zonaCentral. Além disso, no instante seguinte à saída do automóvel de zona1, o automóvel entra na zonaCentral. Neste caso, não há um intervalo entre existência do automóvel entre zonas, porém é uma situação factível já que, a depender da velocidade do automóvel e frequência com que os quadros do vídeo são capturados, poderá ocorrer do automóvel cruzar a faixa de trânsito "instantaneamente". As relações temporais desse exemplo de violação do sinal de semáforo são as seguintes:

- automóvel dentro de zona1 *inicia* sinal no vermelho e
- automóvel dentro de zona1 *encontra* automóvel dentro de zonaCentral e
- automóvel dentro de zonaCentral *finaliza* sinal no vermelho.

Assim a elaboração de uma regra para detectar "violação de sinal de trânsito" para o cruzamento de vídeo do exemplo, deve contemplar as 27 possíveis situações como ilustradas na Figura 4.17. Cada situação é definida por três relações temporais entre as atividades:

1. automóvel dentro de zona1 e automóvel dentro de zonaCentral;
2. automóvel dentro de zona1 e sinal vermelho;
3. automóvel dentro de zonaCentral e sinal vermelho.

Uma regra para detectar a situação (1) da Figura 4.17 é apresentada no Código 4.14. O código em CLIPS desta regra pode ser analisado através das linhas:

Código Fonte 4.14: Regra para detectar uma violação de sinal vermelho.

```
1  (defrule violaSinal1
2  (declare (salience -1))
3  (tempo ?now)
4  (historic (nameOBJ sema1) (nameACT red) (timepoints $?times1))
5  (test (eq (- ?now (nth$ 1 $?times1)) 1))
6  (blob (name ?obj1) (class car))
7  (test (durante_index ?obj1 "dentro_zona zona1" 1 sema1 red 1))
8  (test (precede_index ?obj1 "dentro_zona zona1" 1 ?obj1 "dentro_zona
       zonaCentral" 1))
9  (test (sobrepoe_index sema1 red 1 ?obj1 "dentro_zona zonaCentral" 1))
10 =>
11 (printout t "Alarme: Veiculo " ?obj1 " viola sinal vermelho em t=" ?now
       crlf)
12 )
```

- (1.) Cabeçalho da declaração de regras em CLIPS.
- (2.) Prioridade mais baixa para o disparo da regra, já que as regras para atividades compostas devem ter prioridade menor que as regras para atividades simples.
- (3. até 5.) O tempo atual de processamento em tempo-real será utilizado para decidir se a regra deve ser analisada neste tempo: quando o semáforo deixar de ser vermelho, a análise da regra prossegue.
- (6.) Considera para a regra todos os *blobs* que representam um carro.
- (7.) Testa se a última ocorrência da atividade "objeto dentro de zona1" ocorre *durante* a última ocorrência da atividade "semáforo vermelho".
- (8.) Testa se a última ocorrência da atividade "objeto dentro de zona1" *precede* a última ocorrência da atividade "objeto dentro de zonaCentral vermelho".
- (9.) Testa se a última ocorrência da atividade "semáforo vermelho" *sobrepoe* a última ocorrência da atividade "objeto dentro de zonaCentral".

- (11.) Imprima na saída padrão do sistema um alerta caso ocorra a violação do sinal vermelho.

Para o sistema ser capaz de detectar todas as 27 situações do exemplo, pode-se criar novas definições de regras, ou de uma forma mais apropriada, especificar as demais relações temporais nesta regra e implementá-las através de conjunções (OU).

Os quadro-chaves de um exemplo sintético de vídeo de cruzamento de trânsito são apresentados nas figuras 4.18 e 4.19. Neste exemplo, existem como elementos do cenário um semáforo e um cruzamento entre duas ruas. No decorrer do vídeo, três automóveis atravessam o cruzamento, em tempos distintos. O primeiro automóvel atravessa no sinal verde, o segundo atravessa no sinal vermelho e o terceiro atravessa no verde e o sinal muda após atingir a intersecção das ruas. Ou seja, apenas o segundo veículo comete uma infração, violando o sinal vermelho do semárofo. Nestas figuras, o sinal do semáforo foi detectado erroneamente como um objeto em movimento pelo módulo de baixo nível de segmentação de movimento, porém isto não interferiu na execução das regras.

A dinâmica das atividades simples que são reconhecidas também pode ser representada pelo gráfico de históricos ilustrado na Figura 4.20.

A Figura 4.21 ilustra a detecção pelo sistema da violação do sinal de trânsito, possuindo as mesmas relações da Figura 4.17(3), ou seja:

- automóvel “dentro de zona1” durante sinal vermelho
- automóvel “dentro de zona 1” sobrepõe automóvel “dentro de zonaCentral”
- sinal vermelho sobrepõe automóvel “dentro de ZonaCentral”

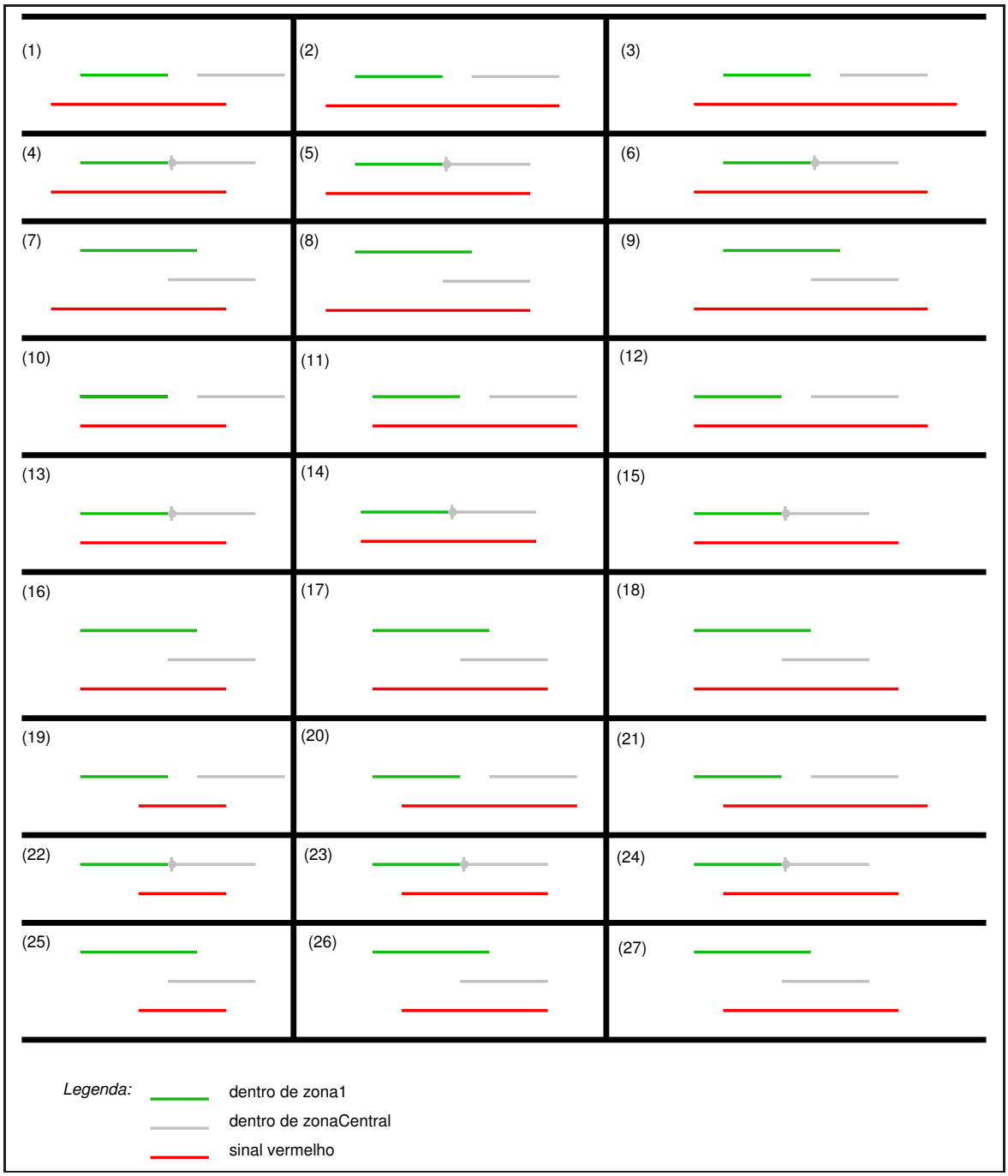
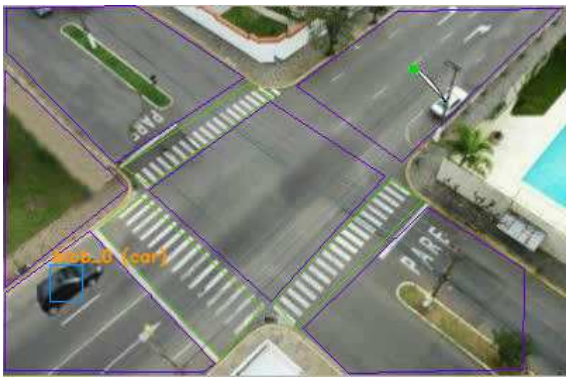
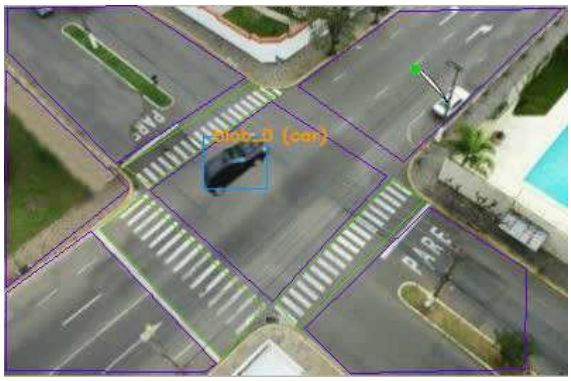


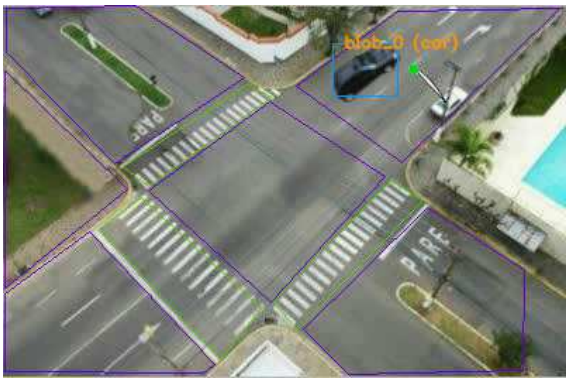
Figura 4.17: 27 possíveis relações de violação de sinal de semáforo.



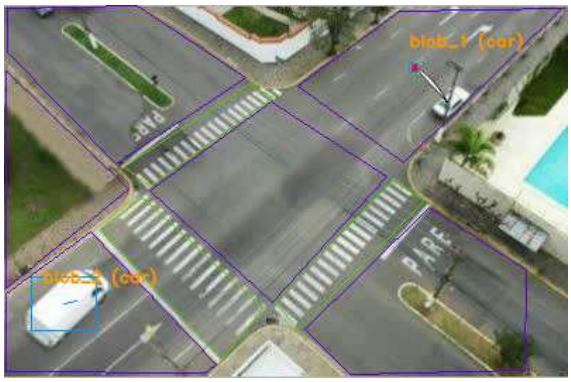
(a) Primeiro carro (blob_0) entra em zona1 e sinal está verde



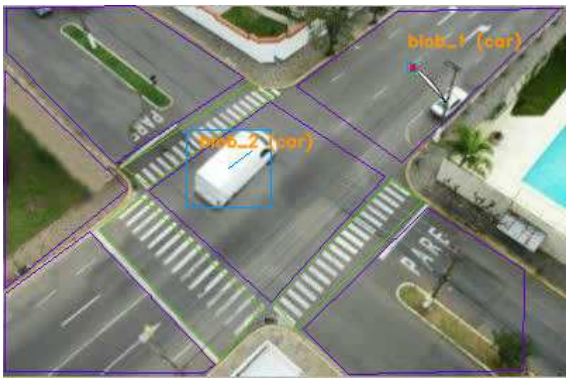
(b) Carro na zonaCentral e sinal continua verde



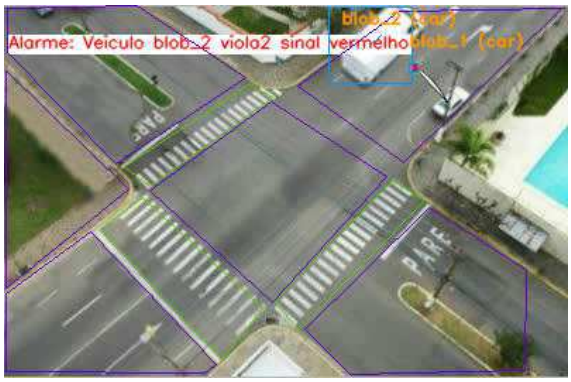
(c) Carro em zona3 e sinal continua verde



(d) Segundo carro (blob_2) na zona1 e sinal continua verde

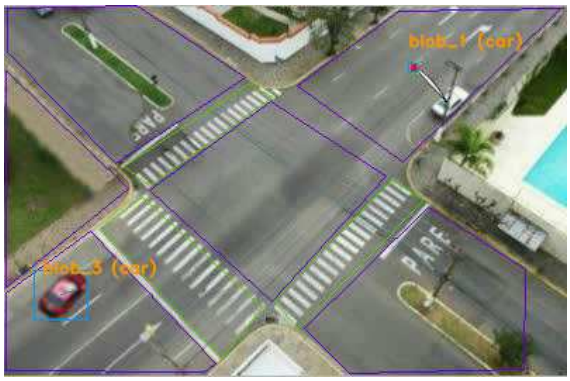


(e) Sinal muda para vermelho enquanto carro está na zonaCentral

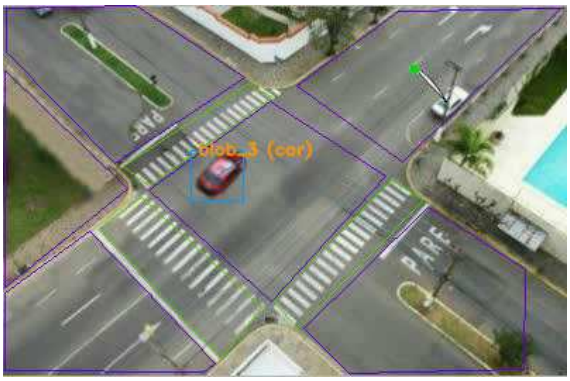


(f) Carro muda para zona3 e um alarme de violação de semáforo é exibido

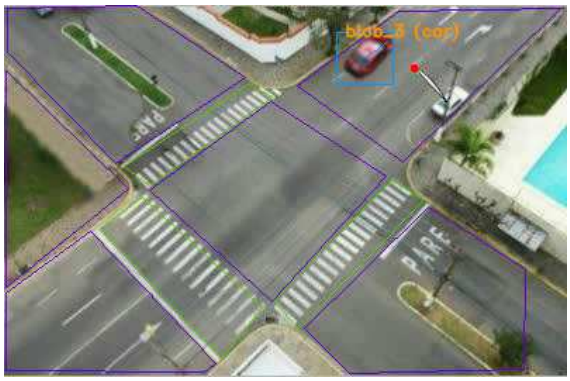
Figura 4.18: Quadros-chave de um vídeo onde ocorre uma violação do semáforo.



(a) Terceiro carro (blob_3) na zona1 e sinal no vermelho



(b) Sinal mudou para verde e carro na zonaCentral



(c) Carro na zona3 e sinal mudou para vermelho

Figura 4.19: Continuação dos quadros-chave da FIgura 4.1 de um vídeo onde ocorre uma violação do semáforo.

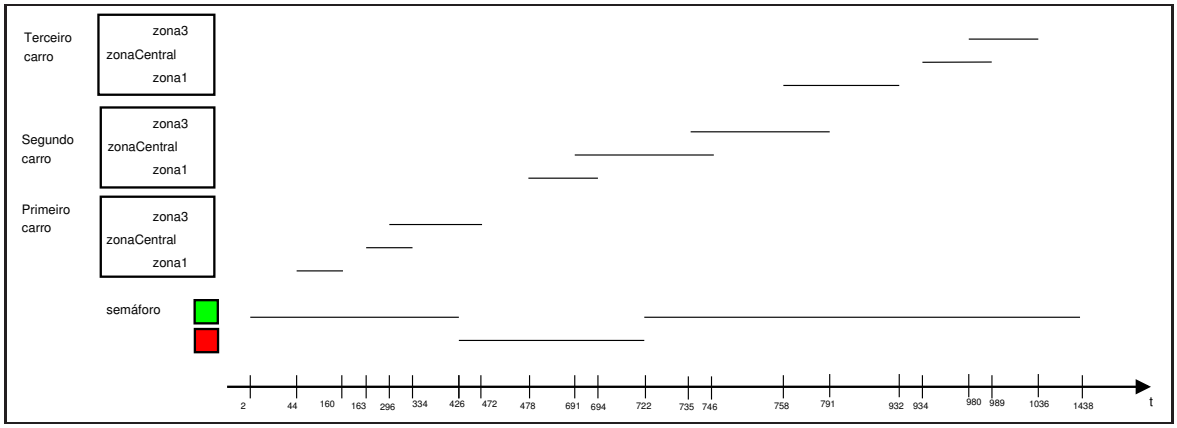


Figura 4.20: Histórico de atividades simples detectadas pelo sistema.

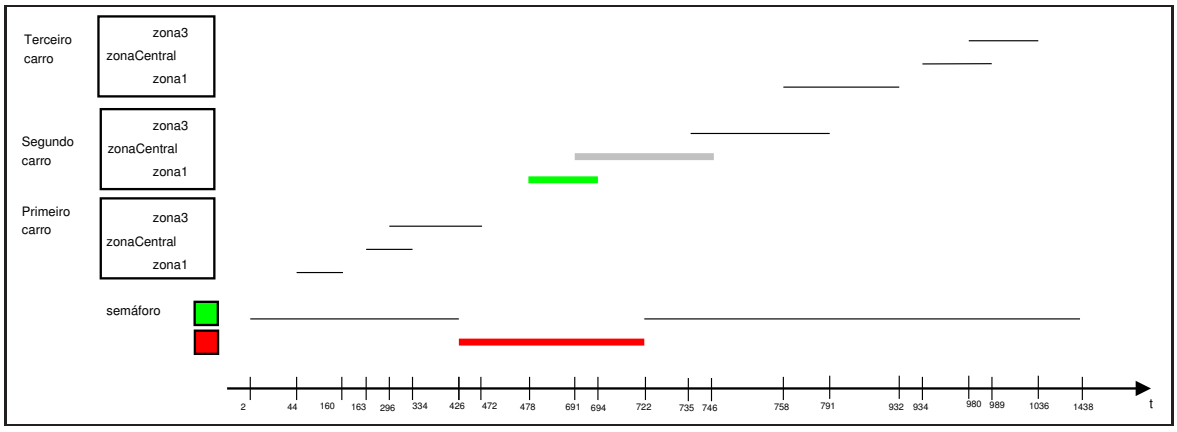


Figura 4.21: Violação detectada pelo sistema.

4.2.2 Abordagem a um veículo

Um caso de interesse num sistema de vigilância é a detecção de casos de furto a veículos parados num semáforo. A abordagem a veículos foi descrita no Capítulo 3 e nesta seção foram realizados três testes para aferir se casos de abordagens são detectados corretamente pelo sistema. Considera-se a ocorrência a uma abordagem a veículo quando uma pessoa que estava numa calçada fica rente a um carro que entrou na rua, e depois ela retorna à calçada enquanto que o carro sai da rua (ver Figura 3.14 na Seção 3.5 para maiores detalhes). Os vídeos para testes foram gerados a partir do cenário de cruzamento como zoneado na Figura 4.16. Esses testes foram descritos a seguir.

Teste 1 O primeiro teste utilizou um vídeo em que foi gerada sinteticamente uma abordagem a um carro, sem qualquer outro elemento em movimento na cena. Alguns quadros-chave da abordagem a um carro estão ilustrados na Figura 4.22. O sistema detectou corretamente a abordagem e lançou um alarme que foi exibido na janela, como se observa na Figura 4.22e.

Teste 2 Neste segundo teste, foram adicionados mais alguns carros no momento da abordagem a fim de que o sistema continuasse a detectar a abordagem. Alguns quadros-chave da abordagem a um carro estão ilustrados na Figura 4.23. O sistema detectou corretamente a abordagem e lançou um alarme que foi exibido na janela, como se observa na Figura 4.23d.

Teste 3 No terceiro experimento, foi testada uma situação semelhante de abordagem onde a pessoa ao invés de voltar para a calçada, vai para outra calçada enquanto o carro se distancia. Alguns quadros-chave deste vídeo estão ilustrados na Figura 4.24. O sistema não detectou abordagem, ou seja, apresentou um comportamento esperado.

(a) Pessoa (blob_0) entra na cena.

(b) Carro (blob_1) entra na cena.

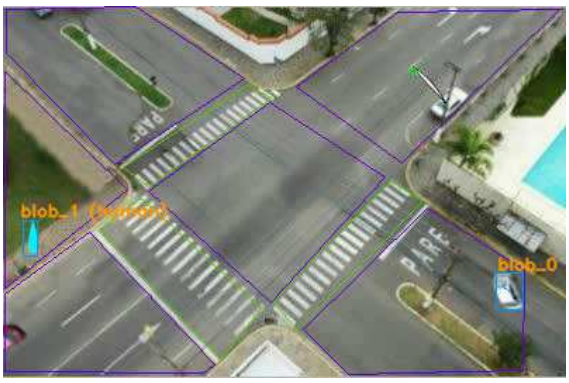
(c) Pessoa fica rente a carro.

(d) Pessoa se afasta do carro e retorna a calçada.

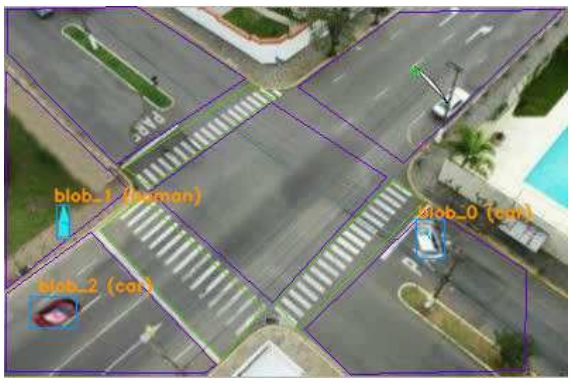
(e) Alarme gerado quando carro sai da zona da rua.

(f) Carro continua seu deslocamento.

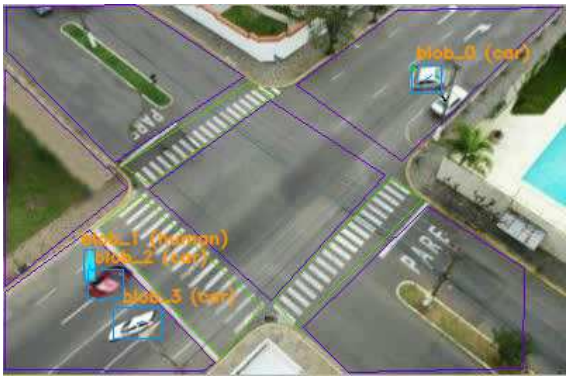
Figura 4.22: Quadros-chave de um vídeo onde ocorre uma abordagem.



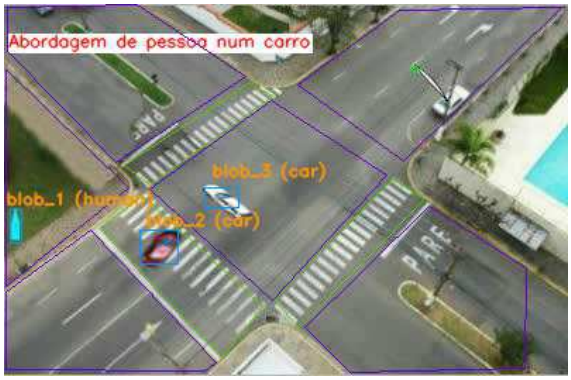
(a) Pessoa (blob_1) e um carro (blob_0) detectados.



(b) Carro vermelho detectado (blob_2).

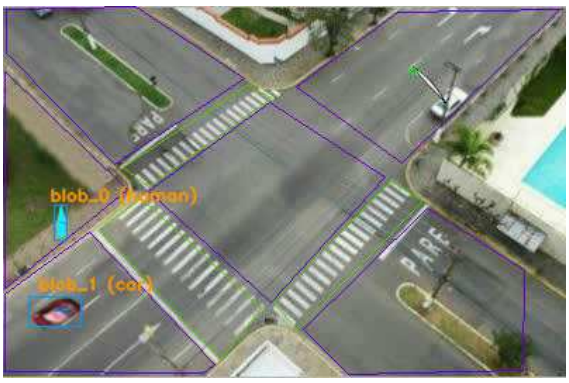


(c) Novo carro detectado (blob_3) e pessoa fica rente a carro.

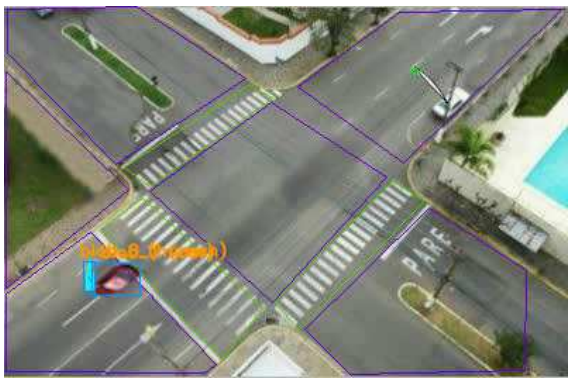


(d) Alarme gerado quando o carro abordado (blob_2) sai da zona de rua.

Figura 4.23: Quadros-chave de um vídeo onde ocorre uma abordagem, com múltiplos objetos que não fazem parte da regra de detecção de abordagem.

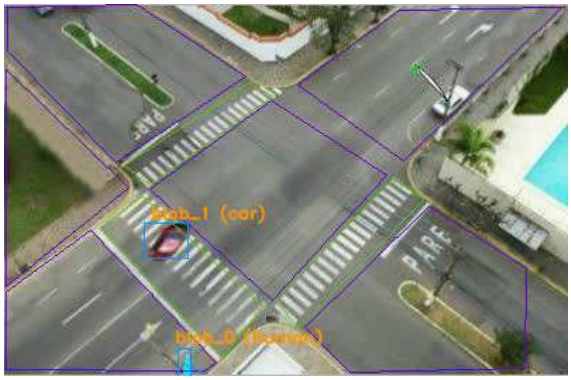


(a) Pessoa (blob_0) e carro (blob_1) detectados.



(b) Pessoa fica rente ao carro.

(c) Pessoa se afasta do carro e vai em direção a outra calçada.



(d) Pessoa entra na outra calçada e carro sai da zona da rua, sem ocorrer disparo de alarme.

Figura 4.24: Quadros-chave de um vídeo onde não ocorre uma abordagem de acordo com a regra, pois a pessoa não volta à calçada original.

4.3 Conclusões

A partir dos testes realizados neste capítulo, pôde-se constatar o grande poder de expressão de atividades e a exatidão do método proposto. Os únicos erros foram devido ao rastreio. Ainda assim, assumindo que os dados do rastreio foram corretos, o método continuou apresentando resultados coerentes.

Capítulo 5

Conclusão

Este trabalho de dissertação objetivou criar um sistema de detecção de eventos pré-definidos em vídeo, de forma a facilitar um especialista especificar eventos em sistemas de vigilância automática. A principal contribuição desta dissertação foi fornecer métodos para um especialista poder especificar os eventos de interesse em um vídeo, através de uma sugestão de representação e detecção de regras para eventos pré-definidos em vídeo. Uma das contribuições foi estender uma linguagem baseada em regras, adicionando raciocínio temporal para a construção de regras de eventos pré-definidos. Além da forma de utilização da linguagem baseada em regras do ambiente CLIPS para a tarefa de especificação e detecção de eventos, este trabalho sugeriu a decomposição de atividades compostas em atividades simples seguindo uma ordem temporal. Para a detecção das regras, este trabalho propôs a análise de regras simples prioritariamente em relação à análise de regras para atividades compostas, construindo um histórico de atividades simples que é posteriormente utilizado para a verificação temporal das relações contidas nas regras de atividades compostas.

Além disto, foi definida uma arquitetura modular para o problema de detecção de eventos pré-definidos em vídeo digital. Para os módulos de baixo nível, foram utilizadas técnicas tradicionais desenvolvidas por outros autores, a exemplo da segmentação de movimento, do rastreio e da classificação. O desenvolvimento do estudo de caso foi realizado objetivando testar as regras de especificação de atividades, a fim de automatizar a detecção de atividades em um cenário de cruzamento de trânsito.

Foram discutidos no Capítulo 3 métodos de representação de conhecimento envolvendo objetos em movimento (cálculo de trajetória, relações topológicas etc.) e de ações temporais

(álgebra de Allen) a fim de detectar atividades pré-definidas mais complexas. A partir da combinação destes métodos foi possível aumentar o poder de expressão da linguagem CLIPS permitindo um especialista elaborar regras complexas. A utilização do sistema proposto não se restringe a sistemas de vigilância automática, já que os métodos de representação propostos podem ser utilizados em outros sistemas onde exista a necessidade de detectar atividades que possam ser decompostas em atividades simples com relacionamento temporal. Em resumo, foram implementadas regras e funções em CLIPS que representam a teoria de atividades simples e compostas. O experimento alvo foi a detecção de atividades ilegais em um cruzamento, demonstrando sua possibilidade de uso em situações reais, envolvendo várias atividades paralelas.

Arquitetura do sistema A arquitetura do sistema apresentada na Seção 3.6 do Capítulo 3 demonstrou ser uma arquitetura flexível onde os algorítmos de baixo nível podem ser alterados sem modificar a arquitetura. A implementação do sensor do estado do semáforo foi realizada de uma forma simples, apresentando bons resultados em vídeos sintéticos. Porém, em situações reais, uma implementação mais robusta deste sensor se faz necessária.

5.1 Trabalhos futuros

Diante dos resultados apresentados no Capítulo 4 e de acordo com a arquitetura proposta para um sistema de vigilância automática apresentada na Seção 3.6 do Capítulo 3 surgem, naturalmente, novas ideias para melhoramento dos experimentos, desenvolvimentos de novos modelos e linhas de pesquisa. Neste sentido, estão listados a seguir alguns dos trabalhos futuros sugeridos para a continuidade da pesquisa em sistemas de vigilância automática:

- Experimentar novos modelos de segmentação de movimento, de rastreio e de classificação;
- Pesquisar outras formas de relações temporárias mais apropriadas à análise de objetos móveis em sistemas de vigilância automática;
- Pesquisar a representação de informações quantitativas como, por exemplo, a definição de restrições temporárias de atividades simples. Exemplo: Se carro ficou 10 minutos

parado, soar alarme;

- Refinar as alterações na linguagem CLIPS, elaborando mais funções que facilite o especialista na elaboração de regras de atividades compostas. Como exemplo, função para detectar uma sequência de eventos, sem a necessidade de explicitar os índices das atividades;

- Realizar experimentos de desempenho de detecção quando os módulos de rastreio ou segmentação falham;

- Verificar a possibilidade de atividades compostas possuírem outra atividades compostas como sub-atividades;

- Implementar um leque maior de atividades simples;

- Pesquisar formas de detecção automática dos elementos contextuais do cenário;

- Pesquisar formas de representação e de detecção de atividades onde múltiplas câmeras são utilizadas;

- Implementar formas de representação 3D da cena;

- De posse de um rastreio mais robusto, realizar mais experimentos com vídeos em tempo-real, bem como realizar experimentos com base de dados públicos [1], a fim de elaborar uma comparação com outros sistemas desenvolvidos.

[1] Como exemplo, o Workshop de testes de vigilância automática PETS, http://winterpets09.net/

Bibliografia

[Albanese et al., 2008] Albanese, M., Chellappa, R., Moscato, V., Picariello, A., Subrahmanian, V. S., Turaga, P., and Udrea, O. (2008). A constrained probabilistic petri net framework for human activity detection in video. In *Multimedia, IEEE Transactions on*, volume 10, pages 982–996.

[Allen, 1983] Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843.

[Allen et al., 2004] Allen, J. G., Xu, R. Y. D., and Jin, J. S. (2004). Object tracking using camshift algorithm and multiple quantized feature spaces. In *VIP '05: Proceedings of the Pan-Sydney area workshop on Visual information processing*, pages 3–7, Darlinghurst, Australia, Australia. Australian Computer Society, Inc.

[Bergin and Gibson, 1996] Bergin, T. J. and Gibson, R. G. (1996). *History of Programming Languages II*. ACM Press, Addison-Wesley, New York.

[Bogaert et al., 2006] Bogaert, P., de Weghe, N. V., Cohn, A. G., Witlox, F., and Maeyer, P. D. (2006). The qualitative trajectory calculus on networks. In *Spatial Cognition*, pages 20–38.

[Borzin et al., 2007] Borzin, A., Rivlin, E., and Rudzsky, M. (2007). Surveillance event interpretation using generalized stochastic petri nets. In *WIAMIS '07: Proceedings of the Eight International Workshop on Image Analysis for Multimedia Interactive Services*, page 4, Washington, DC, USA. IEEE Computer Society.

[Castel et al., 1996] Castel, C., Chaudron, L., and Tessier, C. (1996). What is going on? a high level interpretation of sequences of images.

[Collins et al., 1999] Collins, R. T., Lipton, A. J., and Kanade, T. (1999). A system for video surveillance and monitoring.

[Comaniciu and Meer, 1999] Comaniciu, D. and Meer, P. (1999). Mean shift analysis and applications. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, page 1197, Washington, DC, USA. IEEE Computer Society.

[Cristani et al., 2000] Cristani, M., Cohn, A. G., and Bennett, B. (2000). Spatial locations via morpho-mereology. In *in Proceedings of KR'2000*, pages 15–25. Morgan Kaufmann.

[Cutler and Davis, 2000] Cutler, R. and Davis, L. S. (2000). Robust real-time periodic motion detection, analysis, and applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):781–796.

[da Costa, 2008] da Costa, B. A. D. (2008). Segmentação, Rastreamento de Objetos e Detecção de Eventos Primitivos com Aplicação no Monitoramento Automático de Ações Humanas em Vídeo. Master's thesis, Universidade Federal de Campina Grande, Campina Grande.

[Davis et al., 1997] Davis, L., Chellappa, R., Yacoob, Y., and Zheng, Q. (1997). Visual surveillance and monitoring of human and vehicular activity. In *DARPA97*, pages 19–2.

[de Weghe, 2004] de Weghe, N. V. (2004). *Representing and Reasoning about Moving Objects: A Qualitative Approach*. PhD thesis, Ghent University, Belgium. Adviser-Mubarak Shah.

[Dechter and Schwalb, 1991] Dechter, R. and Schwalb, E. (1991). Temporal constraint networks. *Artificial Intelligence*, 49:61–95.

[Dedeoglu et al., 2006] Dedeoglu, Y., Toreyin, B., Gudukbay, U., and Cetin, A. (2006). Silhouette-based method for object classification and human action recognition in video. In *CVHCI06*, pages 64–77.

[Egenhofer and Franzosa, 1991] Egenhofer, M. and Franzosa, R. (1991). Point-set topological spatial relations. In *International Journal of Geographical Information Systems*, pages 161–174.

[Eschenbach et al., 1999] Eschenbach, C., Habel, C., and Kulik, L. (1999). Representing simple trajectories as oriented curves. In *Proceedings of the Twelfth International Florida Artificial Intelligence Research Society Conference*, pages 431–436. AAAI Press.

[Forbus, 1990] Forbus, K. D. (1990). Qualitative physics: past, present, and future. *Readings in Qualitative Reasoning about Physical Systems*, pages 11–39.

[Freeman, 1975] Freeman, J. (1975). The modelling of spatial relations. In *Computer Graphics and Image Processing*, pages 156–171.

[Freksa, 1992] Freksa, C. (1992). Temporal reasoning based on semi-intervals.

[Freksa and Universität, 1992] Freksa, C. and Universität, F. I. (1992). Using orientation information for qualitative spatial reasoning.

[Ghanem et al., 2004] Ghanem, N., DeMenthon, D., Doermann, D., and Davis, L. (2004). Representation and recognition of events in surveillance video using petri nets. In *CVPRW '04: Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 7*, page 112, Washington, DC, USA. IEEE Computer Society.

[Giachetti et al., 1998] Giachetti, A., Campani, M., and Torre, V. (1998). The use of optical flow for road navigation. *IEEE Transactions on Robotics and Automation*, 14:34–48.

[Giarratano and Riley, 2005] Giarratano, J. C. and Riley, G. (2005). *Expert systems: principles and programming*. Thomson Course Technology, c2005.

[Goyal et al., 2000] Goyal, R., Goyal, R. K., and Goyal, K. (2000). Similarity assessment for cardinal directions between extended spatial objects.

[Hakeem, 2007] Hakeem, A. (2007). *Learning, detection, representation, indexing and retrieval of multi-agent events in videos*. PhD thesis, University of Central Florida, Orlando, FL, USA. Adviser-Mubarak Shah.

[Han and Kamber, 2001] Han, J. and Kamber, M. (2001). Data mining: Concepts and techniques.

[Haritaoglu et al., 1998] Haritaoglu, I., Harwood, D., and Davis, L. S. (1998). W4: Who? when? where? what? a real time system for detecting and tracking people.

[Haritaoglu et al., 2000] Haritaoglu, I., Harwood, D., and Davis, L. S. (2000). W4: Real-time surveillance of people and their activities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):809–830.

[Hu et al., 2004] Hu, W., Tan, T., Wang, L., and Maybank, S. (2004). A survey on visual surveillance of object motion and behaviors. *IEEE Transactions on Systems, Man and Cybernetics*, 34:334–352.

[Hudelot, 2005] Hudelot, C. (2005). Towards a cognitive vision platform for semantic image interpretation; application to the recognition of biological organisms.

[Isard and Blake, 1998] Isard, M. and Blake, A. (1998). Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28.

[Ivanov and Bobick, 2000] Ivanov, Y. A. and Bobick, A. F. (2000). Recognition of visual activities and interactions by stochastic parsing. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):852–872.

[Iwasaki, 1997] Iwasaki, Y. (1997). Real world applications of qualitative reasoning: Introduction to the special issue.

[Jain et al., 1999] Jain, A. K., Murty, M. N., and Flynn, J. P. (1999). Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323.

[Johnson and Hogg, 1995] Johnson, N. and Hogg, D. (1995). Learning the distribution of object trajectories for event recognition. In *BMVC '95: Proceedings of the 6th British conference on Machine vision (Vol. 2)*, pages 583–592, Surrey, UK, UK. BMVA Press.

[Joo and Chellappa, 2006] Joo, S.-W. and Chellappa, R. (2006). Attribute grammar-based event recognition and anomaly detection. In *CVPRW '06: Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, page 107, Washington, DC, USA. IEEE Computer Society.

[Kazarov and Ryabov, 1998] Kazarov, A. and Ryabov, Y. (1998). Clips expert system tool: a candidate for the diagnostic system engine.

[Knauff et al., 1997] Knauff, M., Rauh, R., and Renz, J. (1997). A cognitive assessment of topological spatial relations: Results from an empirical investigation. In *In Proceedings of the 3rd International Conference on Spatial Information Theory (COSIT'97), volume 1329 of Lecture Notes in Computer Science*, pages 193–206. Springer.

[Krausz and Herpers, 2007] Krausz, B. and Herpers, R. (2007). Metrosurv: detecting events in subway stations. *Multimedia Tools and Applications*, 2.

[Kuipers, 1996] Kuipers, B. (1996). A hierarchy of qualitative representations for space. In *In Working papers of the Tenth International Workshop on Qualitative Reasoning about Physical Systems (QR-96*, pages 113–120. AAAI Press.

[Li et al., 2003] Li, L., Huang, W., Gu, I. Y. H., and Tian, Q. (2003). Foreground object detection from videos containing complex background. In *In MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 2–10. ACM Press.

[Lipton et al., 1998] Lipton, A. J., Fujiyoshi, H., and Patil, R. S. (1998). Moving target classification and tracking from real-time video. *Applications of Computer Vision, IEEE Workshop on*, 0:8.

[Moeslund et al., 2006] Moeslund, T. B., Hilton, A., and Kroger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3):90–126.

[Oliver et al., 1999] Oliver, N., Rosario, B., and Pentland, A. (1999). A bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:831–843.

[Panagiotakis et al., 2008] Panagiotakis, C., Ramasso, E., Tziritas, G., Rombaut, M., and Pellerin, D. (2008). Shape-based individual/group detection for sport videos categorization. *IJPRAI*, 22(6):1187–1213.

[Parameswaran and Chellappa, 2003] Parameswaran, V. and Chellappa, R. (2003). View invariants for human action recognition. In *CVPR03*, pages II: 613–619.

[Qiu and Lu, 2009] Qiu, X. and Lu, Q. (2009). Target tracking and localization of binocular mobile robot using camshift and sift. In *GEC '09: Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation*, pages 483–488, New York, NY, USA. ACM.

[Randell et al., 1992] Randell, D. A., Cui, Z., and Cohn, A. G. (1992). A spatial logic based on regions and connection. In *KR*, pages 165–176.

[Rota and Thonnat, 2000] Rota, N. and Thonnat, M. (2000). Activity recognition from video sequences using declarative models. In *ECAI*, pages 673–680.

[Shanahan, 1999] Shanahan, M. P. (1999). The event calculus explained. In Woolridge, M. J. and Veloso, M., editors, *Artificial Intelligence Today, Lecture Notes in AI no. 1600*, pages 409–430. Springer.

[Sharma, 1996] Sharma, J. (1996). Integrated spatial reasoning in geographic information systems: Combining topology and direction. Technical report, USA, University of Maine.

[Shet et al., 2005] Shet, V., Harwood, D., and Davis, L. (2005). Vidmap: video monitoring of activity with prolog. *Advanced Video and Signal Based Surveillance, IEEE Conference on*, 0:224–229.

[Siebel, 2003] Siebel, N. T. (2003). *Design and Implementation of People Tracking Algorithms for Visual Surveillance Applications*. PhD thesis, Department of Computer Science, The University of Reading, Reading, UK.

[Smith and Brady, 1995] Smith, M. S. and Brady, J. M. (1995). Asset-2: real-time motion segmentation and shape tracking. *Computer Vision, IEEE International Conference on*, 0:237.

[Starner and Pentland, 1995] Starner, T. and Pentland, A. (1995). Real-time american sign language recognition from video using hidden markov models. In *ISCV '95: Proceedings of the International Symposium on Computer Vision*, page 265, Washington, DC, USA. IEEE Computer Society.

[Stauffer and Grimson, 2000] Stauffer, C. and Grimson, W. E. L. (2000). Learning patterns of activity using real-time tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):747–757.

[Syeda Mahmood et al., 2001] Syeda Mahmood, T., Vasilescu, M., and Sethi, S. (2001). Recognizing action events from multiple viewpoints. In *EventVideo01*, pages 64–72.

[Vallejo et al., 2009] Vallejo, D., Albusac, J., Jimenez, L., Gonzalez, C., and Moreno, J. (2009). A cognitive surveillance system for detecting incorrect traffic behaviors. *Expert Syst. Appl.*, 36(7):10503–10511.

[Wang et al., 2003] Wang, L., Hu, W., and Tan, T. (2003). Recent developments in human motion analysis. *Pattern Recognition*, 36:585–601.

[Wren et al., 1997] Wren, C., Azarbayejani, A., Darrell, T., and Pentland, A. (1997). Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:780–785.

[Xiang and Gong, 2008] Xiang, T. and Gong, S. (2008). Video behavior profiling for anomaly detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(5):893–908.

[Xiang et al., 2002] Xiang, T., Gong, S., and Parkinson, D. (2002). Autonomous visual events detection and classification without explicit object-centred segmentation and tracking. In *In British Machine Vision Conference*, pages 233–242.

[Xu and Wunsch, 2005] Xu, R. and Wunsch, D. (2005). Survey of clustering algorithms. *Neural Networks, IEEE Transactions on*, 16(3):645–678.

[Yilmaz et al., 2006] Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *ACM Comput. Surv.*, 38(4):13.