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

ARCABOUÇO PARA ANÁLISE DE EVENTOS EM VÍDEO

Adson Diego Dionisio da Silva

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 Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Visão Computacional

HERMAN MARTINS GOMES,
EANES TORRES PEREIRA
(Orientadores)

Campina Grande, Paraíba, Brasil
Adson Diego Dionisio da Silva, 31/08/2015

Resumo

O reconhecimento automático de eventos de interesse em vídeos envolvendo conjuntos de ações ou de interações entre objetos. Pode agregar valor a sistemas de vigilância, aplicações de cidades inteligentes, monitoramento de pessoas com incapacidades físicas ou mentais, dentre outros. Entretanto, conceber um arcabouço que possa ser adaptado a diversas situações sem a necessidade de um especialista nas tecnologias envolvidas, continua sendo um desafio para a área. Neste contexto, a pesquisa realizada tem como base a criação de um arcabouço genérico para detecção de eventos em vídeo com base em regras. Para criação das regras, os usuários formam expressões lógicas utilizando Lógica de Primeira Ordem e relacionam os termos com a álgebra de intervalos de Allen, adicionando assim um contexto temporal às regras. Por ser um arcabouço, ele é extensível, podendo receber módulos adicionais para realização de novas detecções e inferências. Foi realizada uma avaliação experimental utilizando vídeos de teste disponíveis no site Youtube envolvendo um cenário de trânsito, com eventos de ultrapassagem do sinal vermelho e vídeos obtidos de uma câmera ao vivo do site Camerite, contendo eventos de carros estacionando. O foco do trabalho não foi criar detectores de objetos (e.g. carros ou pessoas) melhores do que aqueles existentes no estado da arte, mas propor e desenvolver uma estrutura genérica e reutilizável que integra diferentes técnicas de visão computacional. A acurácia na detecção dos eventos ficou no intervalo de 83,82% a 90,08% com 95% de confiança. Obteve acurácia máxima (100%) na detecção dos eventos, quando substituído os detectores de objetos por rótulos atribuídos manualmente, o que indicou a eficácia do motor de inferência desenvolvido para o arcabouço.

Abstract

Automatic recognition of relevant events in videos involving sets of actions or interactions between objects can improve surveillance systems, smart cities applications, monitoring of people with physical or mental disabilities, among others. However, designing a framework that can be adapted to several situations without an expert in the involved technologies remains a challenge. In this context, this work is based on the creation of a rule-based generic framework for event detection in video. To create the rules, users form logical expressions using first-order logic (FOL) and relate the terms with the Allen's interval algebra, adding a temporal context to the rules. Once it is a framework, it is extensible, and may receive additional modules for performing new detections and inferences. Experimental evaluation was performed using test videos available on Youtube, involving a scenario of traffic with red light crossing events and videos from Camerite website containing parking car events. The focus of the work was not to create object detectors (e.g. cars or people) better than those existing in the state-of-the-art, but, propose and develop a generic and reusable framework that integrates differents computer vision techniques. The accuracy in the detection of the events was within the range of 83.82% and 90.08% with 95% confidence. Obtained maximum accuracy (100 %) in the detection of the events, when replacing the objects detectors by labels manually assigned, what indicated the effectiveness of the inference engine developed for this framework.

Agradecimentos

Agradeço a Deus por todas as oportunidades que ele me tem concedido, a meus orientadores Eanes e Herman que me apoiaram, mesmo nas adversidades, pelas correções do meu português ruim, pela paciência, por cada elogio e crítica recebida, por cada orientação, por todo o conhecimento que me passaram e por terem me ajudado a chegar ao final desse mestrado.

Agradeço a minha mãe e ao meu irmão que me apoiaram e sempre estiveram ao meu lado. A minha namorada, que me apoiou e me ajudou nas revisões finais, aos meus amigos Yuri e José que me ajudaram a chegar até aqui.

Agradeço também a todos os que me ajudaram nessa caminhada: familiares, amigos, companheiros de trabalho, professores e funcionários da UFCG.

Conteúdo

Lista de Símbolos

BPM - *Business Process Modeling*
DARPA - *Defense Advanced Research Projects Agency*
EDL - *Event Definition Language*
HMM - *Hidden Markov Model (Modelos Ocultos de Markov)*
IEC - *The International Electrotechnical Commission*
IEEE - *Instituto de Engenheiros Eletricistas e Eletrônicos*
ISO - *The International Organization for Standardization*
LPO - *Lógica de Primeira Ordem*
SVDD - *Support Vector Data Description*
SVM - *Support Vector Machine (Máquina de vetor de suporte)*
TPM - *Temporal Pyramid Model*
XML - *Extensible Markup Language*
CCM - *Coeficiente de Correlação de Matthews*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Vigilância (PICIARELLI; FORESTI, 2011), monitoramento automático de trânsito no contexto de cidades inteligentes (YUNG; LAI, 2001; ALMEIDA, 2010), monitoramento automático de pacientes, idosos ou crianças (NA; QIN; WRIGHT, 2011), são alguns dos problemas abordados em várias áreas da computação. Pesquisas no ramo da visão computacional, principalmente envolvendo o processamento de alto nível (GONZALEZ; WOODS, 2006), no qual a análise de uma cena retorna uma informação relevante (e.g. ação, evento, indivíduo), têm se direcionado também para os problemas citados anteriormente, por meio da análise automática de vídeo.

Dentre os conceitos de análise automática de vídeo, está a detecção de eventos em vídeo, desde eventos primitivos, como a detecção da entrada de uma pessoa em um ambiente monitorado, até eventos compostos, que consiste na junção de eventos primitivos ordenados no tempo (ZHAI et al., 2009). As detecções de eventos em vídeo podem ser classificadas em duas grandes categorias: reconhecimento de eventos explícitos, na qual o sistema tem o conhecimento do evento que deve ser identificado e o reconhecimento de eventos anormais, em que o sistema cria ou recebe um modelo dos eventos normais e reconhece eventos não pertencentes ao modelo (PICIARELLI; FORESTI, 2011).

A detecção automática de eventos em vídeos tem sido abordada em várias pesquisas de formas diferentes, algumas utilizando classificadores que detém o conhecimento sobre o evento a ser detectado (MOTA et al., 2013), outras com técnicas com base em regras pré-definidas que relacionam eventos primitivos para a criação de eventos de mais alto nível, com (ONAL et al., 2013) ou sem (ZHAI et al., 2009) a ajuda de classificadores. Utilizando modelos de aprendizagem não supervisionadas para a obtenção de modelos e detecção de anomalias destes modelos, e também a detecção de eventos compostos por regras de relacionamento entre objetos e planos de fundos rotulados (GULER; LIANG; PUSHEE, 2003; ALMEIDA, 2010).

Pesquisas voltadas para a detecção de eventos explícitos comumente abordam eventos predefinidos, em nível de codificação (NA; QIN; WRIGHT, 2011; ONAL et al., 2013), tem linguagem de definição de regras complexas (GULER; LIANG; PUSHEE, 2003; ALMEIDA, 2010). Que necessita de um especialista não

só no domínio da aplicação como também na linguagem de construção das regras que definem um evento, ou fazem uso de classificadores que necessitam ser treinados para aprender padrões sobre os eventos que serão reconhecidos (KAMISHIMA; INOUE; SHINODA, 2013).

Entretanto, essa última abordagem denota uma inflexibilidade na reutilização do sistema em condições variadas, deixando-o estagnado para uma determinada finalidade para a qual tal sistema foi treinado. Para o estabelecimento das regras que definem os eventos, algumas técnicas de modelagem são utilizadas e.g. Redes de Petri (GHANEM et al., 2004), modelos de Markov (PIXI; HONGYAN; WEI, 2010) , linguagens de modelagem como CLIPS (GULER; LIANG; PUSHEE, 2003; ALMEIDA, 2010) e linguagem de definição de eventos (*Event Definition Language - EDL*) (XIAO et al., 2012).

Muitos dos trabalhos, principalmente os que fazem uso de uma EDL própria, têm como base XML (*eXtensible Markup Language*) para a especificação e armazenamento do conjunto de regras. Pesquisas voltadas para o tratamento de eventos em outras áreas, como redes de computadores, podem ser úteis quanto ao tratamento, descrição e armazenamento dos eventos (XIAO et al., 2012).

Os sistemas que não apresentam facilidade de reutilização e interface de fácil utilização para a descrição dos eventos, terminam estagnados em situações pontuais. Necessitando de adaptações realizadas por especialistas para englobar novas situações. Dessa forma, temos como principais problemas na área de detecção automática de eventos em vídeo a escassez de interfaces com o usuário para a criação e manipulação de regras para eventos e a dificuldade para adaptar os arcabouços existentes para novas situações.

1.1 Objetivos

Considerando o problema exposto anteriormente, as modelagens de eventos a partir de aprendizagem são de difícil personalização, devido à necessidade de um conjunto de treinamento, porém, os eventos com base em regras possibilitam uma adaptação mais rápida a novas situações. O objetivo geral deste trabalho é, portanto, criar um arcabouço para a detecção de eventos com base em regras, no qual o conjunto de regras é formado por meio de lógica de primeira ordem (Subseção 1.3.2) e contextualizado temporalmente com a álgebra de Allen (Subseção 1.3.3). Tais regras constituem o modelo do evento a ser detectado. A seguir, são listados os objetivos específicos.

1.1.1 Objetivos específicos

Os objetivos específicos dessa dissertação abordam a generalidade do arcabouço proposto, a forma de modelagem, a estrutura e funcionalidade. Assim, tem-se a seguinte lista de objetivos:

- Propor uma representação para a modelagem dos eventos com base em lógica de primeira ordem e álgebra de Allen;
- Propor uma forma de armazenamento para os modelos dos eventos;
- Propor um arcabouço extensível com base em componentes (módulos) de *software* para a detecção de eventos em vídeo;
- Provar a eficácia e a generalidade do arcabouço.

1.2 Relevância

Hoje em dia, sistemas de vigilância eletrônica em vídeo passaram a ser fundamentais para empresas de pequeno a grande porte. Babás eletrônicas com câmeras atualmente passaram a ser comuns nas casas com bebês. A sumarização de longos vídeos para demarcação de pontos importantes para o usuário, a monitoração de animais de estimação, dentre outras atividades, são alguns exemplos que evidenciam a importância da concepção de sistemas automáticos para processamento de vídeos.

Por isso, vários sistemas tanto no âmbito de *softwares* livres, como iSpyConnect (ISPYCONNECT, 2015) e Zoneminder (ZONEMINDER, 2015), quanto de privados, como DigiFort (DIGIFORT, 2015), trabalham com monitoramento e em formas de melhorar o trabalho de detecção de eventos que ocorrem nos ambientes monitorados, automatizando processos que antes eram realizados apenas por pessoas, auxiliando e/ou detectando eventos nos ambientes propostos.

Tais sistemas fazem uso de abordagens já difundidas em pesquisas, acoplando-os como extensões, de modo a detectar eventos pré-estabelecidos ou relacionando detecções de objetos com áreas demarcadas nos vídeos. Entretanto, ao ser necessário personalizar os eventos, os usuários ficam restritos a utilizarem apenas as funcionalidades providas por cada extensão separadamente (e.g. detecção de atributos como: velocidade, direção e áreas demarcadas manualmente nos vídeos). Dessa forma, o trabalho se faz relevante por proporcionar ao usuário relacionar as detecções de diferentes extensões em sequência temporal para formar o modelo do evento a ser detectado.

1.3 Fundamentação

Este trabalho tem como foco a criação de um arcabouço extensível por módulo que usa modelagem, e a detecção de eventos em vídeo por meio de regras seguindo um padrão especificado nesta dissertação. Para tal, foram utilizados conceitos de engenharia de *software* (Subseção 1.3.1), de lógica (Subseção 1.3.2), e contextualização temporal (Subseção 1.3.3). Porém, para que fosse possível a validação em vídeos reais, foi necessário criar um simples detector e rastreador de objetos. Na construção do detector foram utilizados conceitos de detecção de

objetos (Subseção 1.3.4). Também para validar com um cenário de trânsito, foi necessário a criação de um detector de estado do semáforo, a partir de conceitos de cores e intensidade (Subseção 1.3.5).

1.3.1 Arcabouços

Arcabouço, segundo explicações de Pressman (2001) é uma infraestrutura esquelética de aplicação específica, ou seja, uma miniarquitetura reutilizável que fornece estrutura e comportamento genéricos para uma família de sistemas e abstrações em um determinado domínio. Essa estrutura conta com uma coleção de pontos de conexões que permitem a adaptação a um domínio específico do problema. Esses pontos permitem a integração de classes ou funcionalidades de um sistema à estrutura do arcabouço.

Riehle e Gross (1998) consideram um arcabouço como um conceito central de desenvolvimento de *software* orientado a objeto em larga escala. O qual aumenta a produtividade, a qualidade das aplicações e reduz o tempo de desenvolvimento. Assim, o autor afirma que um arcabouço é um modelo de classes, juntamente com a integração de um conjunto de papéis e de classes adaptáveis e extensíveis. Abrange um domínio específico ou um aspecto significativo do domínio. É uma unidade coerente de reutilização, tanto pelo reuso dos relacionamentos ou por extensão de subclasses. Podendo ser caracterizado por ser de caixa-preta ou de caixa-branca ou ambos. O arcabouço de caixa-preta encapsula toda aplicação, criando pontos de acesso, os quais são instanciados. O de caixa-branca necessita do fornecimento de novas subclasses primeiro, para assim poder criar e compor instâncias. Muitas estruturas combinam essas duas características, provendo classes usáveis e classes abstratas.

Johnson (1992) afirma que um arcabouço é um *design* de um programa reutilizável ou parte de um programa expressado por um conjunto de classes, que como todos os programas, são uma mistura de classes concretas e abstratas. Arcabouços são estruturas reutilizáveis, não apenas código fonte, sendo eles mais abstratos que a maioria dos programas. Eles são criados por especialistas no domínio e utilizados por não especialistas para solução de problemas em domínio específico. O autor também considera que um bom arcabouço pode ser utilizado para fins que não foi projetado inicialmente.

Sommerville et al. (2008) afirma que a orientação a objetos é uma das chaves para o reuso de *software*, porém objetos são muitas vezes pequenos demais ou muito especializados. Dessa forma, temos que a reutilização orientada a objeto é melhor realizada por meio de abstrações de maior granularidade, chamadas de arcabouço. Um arcabouço é uma estrutura genérica que é estendida para criar uma aplicação ou subsistema mais específico. Arcabouço para o autor é uma coleção de classes concretas e abstratas que são adaptadas e estendidas para criar aplicações de sistema, que são implementados como uma coleção de objetos concretos e abstratos em uma linguagem de programação orientada a objetos. Arcabouços proveem suporte para funcionalidades genéricas que são

utilizadas em aplicações de tipos similares. Provendo suporte ao *design* de reuso e uma arquitetura esquelética para aplicações, como também o reuso de classes específicas do sistema. Toda a arquitetura é definida por objetos de classes e as interações entre eles. As classes são reutilizadas diretamente e podem ser estendidas usando funcionalidades como herança.

1.3.2 Lógica de primeira ordem

Segundo Allen (1995), a lógica foi desenvolvida como uma notação para capturar as propriedades essenciais da linguagem natural e do raciocínio. Assim, ambas as linguagens, lógica e natural, possuem propriedades em comum. Essas similaridades acarretam uma facilidade de conversão da linguagem natural para a lógica. Em ambas as linguagens, existem expressões que definem objetos, relacionamento entre objetos e possibilitam a obtenção de conclusões sobre os objetos. Na lógica, essas expressões são divididas em duas classes: os termos, que definem os objetos e os relacionamentos entre eles e as proposições, que fazem asserções sobre os termos.

Entre os termos, o autor afirma a existência de duas grandes classes: constantes e funções. Constantes são o que se aproximam mais dos nomes na linguagem natural. Entretanto, diferentemente do que ocorre em linguagem natural, não pode haver ambiguidade, ou seja, dois objetos não podem ter o mesmo nome. Funções correspondem a propriedades do objeto ou relacionamentos entre objetos. Por exemplo, uma função que corresponde ao relacionamento “Pai de João” seria Pai(João).

Na lógica de primeira ordem (LPO), conhecida também como cálculo de predicados de primeira ordem (CPPO), uma proposição é formada por um conjunto de termos, que por sua vez é formado por um conjunto de funções e propriedades, como por exemplo a frase “João gosta de seu pai” pode ser descrita em LPO como gosta(João,pai(João)). Outros operadores lógicos também podem ser utilizados como: ou $(\vee)$, e $(\wedge)$, não $(\neg)$, ou exclusivo $(\oplus)$ e implicação $(\Rightarrow)$. O que diferencia a LPO da lógica proposicional é a adição de quantificadores conhecidos também como variáveis lógicas. Tais quantificadores são o existencial $\exists$, que é usado em sentenças como “Existe um carro sobre a calçada”, que é representada em LPO como:

$\exists\, x\ \ CARRO(x)\ \wedge\ SOBRE(x,\ Calcada1)$

E o quantificador universal $\forall$, que pode ser empregado em sentenças como “todo carro é vermelho” que corresponde em LPO a:

$\forall\, x\ \ CARRO(x)\ \Rightarrow COR(x,\ Vermelho)$

Também, segundo Russell e Norvig (2002), ao se analisar a sintaxe da linguagem natural, os elementos mais óbvios são os substantivos e frases nominais que se referem a objetos (por exemplo, carros e calçadas), verbos (por exemplo, gostar e ser) e expressões verbais que se referem às relações entre objetos (por exemplo, é rica e é pobre). Em que alguns desses relacionamentos são funções que retornam um valor para uma determinada entrada. Semelhante ao exposto

anteriormente, a lógica de primeira ordem é formada por objetos e relacionamentos que são definidos pelo domínio da aplicação, e conectados ou modificados por operadores lógicos. Utilizando como exemplo a sétima frase da Tabela 1.1 “Se Sue é rica, então ela é feliz” temos como objeto Sue e como relacionamentos as expressões verbais “é Rica” e “é feliz” montando a expressão lógica Feliz(Sue) $\wedge$ Rica(Sue) formada pelos relacionamentos Feliz e Rica sobre o objeto Sue, conectados pelo operador $\wedge$. A Tabela 1.1 mostra alguns exemplos de frases e suas equivalências em lógica.

Tabela 1.1: Relação entre Linguagem natural e LPO (ALLEN, 1995).

Frase	Equivalente em Lógica de 1a Ordem
John	John1
Pai de John	Pai(John1)
John não gosta do Pai	¬Gosta(John1,Pai(John1))
Sue é feliz, ou ela é rica (ou os dois)	Feliz(Sue) $\vee$ Rica(Sue)
Sue é feliz e rica	Feliz(Sue) $\wedge$ Rica(Sue)
Sue é rica, ou ela é pobre	Rica(Sue) $\oplus$ Pobre(Sue)
Se Sue é rica, então ela é feliz	Rica(Sue) $\Rightarrow$ Feliz(Sue)

Este trabalho utiliza um subconjunto da lógica de primeira ordem, o cálculo proposicional, que mapeia todas as fórmulas para valores V (verdadeiro) ou F (Falso). Desta forma, toda saída de uma regra resulta em V, se houve o evento, ou F, se não houve.

1.3.3 Contexto temporal dos eventos

Como exposto por Guler, Liang e Pushee (2003), eventos podem ser formados pela junção de outros eventos, criando dessa forma, eventos complexos. Porém, essa junção de acontecimentos segue uma sequência temporal (FLEISCHMAN; DECAMP; ROY, 2006; CHUANG et al., 2014; CHEN; TSAI, 2014), na qual eventos primitivos acontecem sequencialmente para formar os eventos complexos. Sendo assim, muitos autores recorrem aos relacionamentos temporais descritos por Allen (1983) ou técnicas que os utilizam como base (WU; CHEN, 2007).

No trabalho de Allen (1983), ele define que qualquer relacionamento entre dois intervalos I_1 e I_2 pode ser representado por 7 relações temporais (precede, encontra, sobrepõe, inicia, finaliza, durante e igual) e as inversões (para o igual não existe inversão), totalizando 13 relacionamentos, como apresentado nas Figura 1.1. O autor demonstra a generalidade ao utilizar intervalos de tempo em pontos fixos no tempo, pois a imprecisão contida na primeira abordagem pode considerar eventos mesmo que estejam em pontos distantes no tempo, desde que obedeçam aos intervalos estabelecidos. Sendo assim, de acordo com Allen (1983), os eventos não necessitam acontecer em uma ordem pré-estabelecida, diferentemente de uma abordagem orientada a estados. O autor também apresenta como benefícios, em relação a uma abordagem orientada a estados, a fa-

cilidade de implementação dos intervalos e que eles são computacionalmente viáveis.

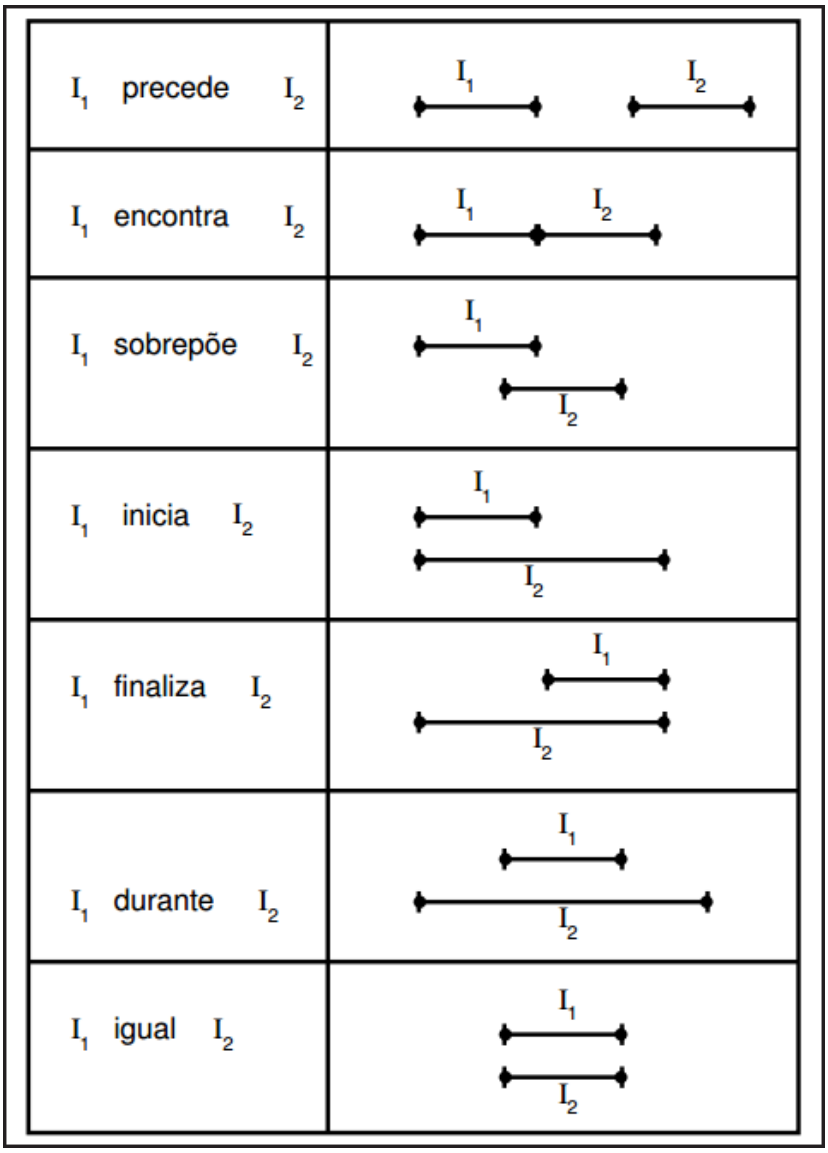


Figura 1.1: Relacionamentos temporais de Allen (ALLEN, 1983).

1.3.4 Detecção e rastreamento de objetos

Segundo Gonzalez e Woods (2006), o processo de detecção de objetos parte inicialmente da segmentação, processo em que é separado o objeto de interesse do restante da imagem. Essa fase é caracterizada por se ter uma imagem tanto de entrada, quanto de saída. Parte dos algoritmos de segmentação utilizam propriedades básicas de intensidade ou descontinuidade e similaridade. No primeiro caso, o particionamento da imagem é realizado onde há uma discrepância na mudança de intensidade, como linhas e contornos. O segundo particiona as imagens

em regiões que são similares a um determinado padrão ou grupo de características.

Um dos recursos utilizados para segmentação em vídeos, é o mesmo utilizado por pessoas e animais para separar o plano de fundo de objetos de interesse. Assim, como afirma Marr (1982), O movimento de um objeto contra o plano de fundo pode ser usado para delimitar os contornos de um objeto, e o sistema visual humano é bem eficiente nesta tarefa. Dessa forma, temos que uma das técnicas mais simples para a detecção de movimento é a utilização da diferença entre quadros de um vídeo para segmentar os objetos que se moveram, para depois classificá-los. Da mesma forma que no trabalho de Krausz e Herpers (2010), foi empregado nesta dissertação um Modelo de Mistura de Gaussianas adaptativa (*Adaptive Gaussian mixture models* (GMM)), que realiza subtração de plano de fundo. A técnica consiste na criação de um modelo para o quadro de vídeo atual, que será comparado com o modelo do quadro futuro. As diferenças entre esses modelos são consideradas movimento. Para a criação do modelo, a técnica utiliza um modelo de mistura de gaussianas para cada *pixel* contido na imagem. Criando uma função de densidade, que é utilizada na comparação com o quadro de vídeo seguinte, em que os *pixels* da imagem futura que forem descritos pela função de densidade são considerados como fazendo parte do plano de fundo e os outros como objeto em movimento.

Após a detecção dos objetos que se movimentaram, é realizada a classificação das áreas de interesse, que identifica a qual classe pertencem os objetos. Tal reconhecimento de padrões pode ser dividido em duas categorias (GONZALEZ; WOODS, 2006): teoria da decisão e estrutural. Na primeira categoria, utilizam-se descritores quantitativos que descrevem a classe, como tamanho, área e textura. No segundo, utilizam-se descritores qualitativos que descrevem o objeto (conjunto de vetores, linhas ou padrões). Nesta dissertação foi utilizada a categoria de descritor quantitativo, por ser genérico e de fácil implementação. Pois o foco da dissertação não é a criação de detectores superiores ao estado da arte.

Em seguida, para a detecção de eventos é necessário acompanhar o objeto. Para saber por onde ele se movimenta e identificar o objeto em todos os quadros do vídeo, atribuindo-lhe um identificador único, esta fase é chamada de rastreamento. Existem várias técnicas de rastreamento, porém as mais simples usam um limiar com base na distância entre os objetos detectados em cada quadro consecutivo. Tal distância pode ser calculada de várias formas (e.g. índice de Jaccard e distância euclidiana), como por exemplo, Krausz e Herpers (2010) utilizam a distância euclidiana entre os centroides dos objetos. Nesta dissertação é utilizado o índice de Jaccard ou coeficiente de similaridade de Jaccard, mesmo método utilizado por Jain e Learned-Miller (2010) para comparação dos resultados com os rótulos da base de dados, em que dados dois objetos A e B, essa medida consiste na união das áreas dos objetos sobre a intercessão, como descrito na fórmula $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$, que representa o quão próximo estão as marcações dos objetos. Esta técnica foi escolhida devido ser simples, genérica e de fácil implementação, como também devido à fase de detecção e rastreamento

não ser o foco desta pesquisa.

Utilizando o exemplo da Figura 1.2, temos na Figura (a) a detecção de um objeto, sua marcação e o respectivo centroide. Na Figura (b), temos a mesma marcação em preto que representa a detecção do quadro anterior e mais duas novas marcações, uma vermelha e a outra em verde, que representam as detecções do quadro atual e os respectivos centroides. Se considerarmos apenas a distância euclidiana a marcação menor seria considerada como sendo o mesmo objeto marcado de preto no quadro anterior. Pois é o com menor distância entre os centroides, porém usando o coeficiente de Jaccard é levada em consideração a área da marcação, o que consideraria a detecção em vermelho como a continuação da marcação em preto e a verde como uma nova detecção.

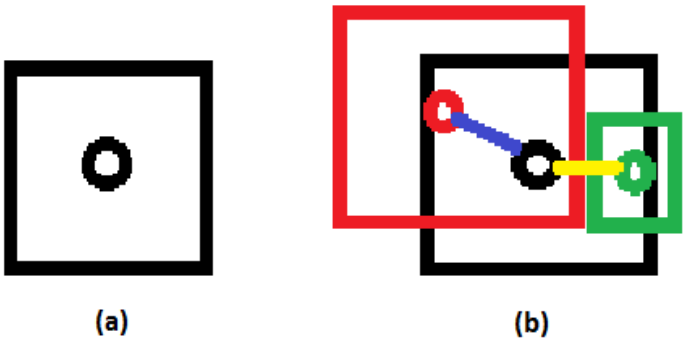


Figura 1.2: Exemplos de detecções de objetos.

1.3.5 Detecção de cor

Segundo Gonzalez e Woods (2006), o processamento de cores é dividido em duas grandes áreas: cores completas e pseudo-cores. A primeira especifica imagens que são adquiridas de sistemas que dão suporte a cores completas, como câmeras de vídeo e *scanners* coloridos, em que cores completas consistem no espectro contínuo de cores que abrange do violeta ao vermelho. A segunda, trabalha com imagens adquiridas em sistemas monocromáticos ou por intervalos de intensidade.

No presente trabalho, para a detecção da cor da luz que define o estado do semáforo, foi utilizado o modelo de pseudo-cores por meio de intervalos de intensidade, utilizando a escala de cinza, na qual o valor de cada pixel é uma única amostra de um espaço de cores. Nas imagens em tons de cinza, cada pixel resulta do cálculo da intensidade da luz do respectivo pixel na imagem colorida. Sendo assim, podemos medir a intensidade da luz em determinadas áreas da imagem, podendo por exemplo, predizer em que ponto da imagem há uma luz acesa, se é na área em que fica a luz vermelha, amarela ou verde.

1.4 Estrutura da dissertação

A dissertação continua com os trabalhos relacionados ao tema proposto no Capítulo 2, em seguida há a apresentação da arquitetura e funcionamento do arcabouço proposto no Capítulo 3, apresentação da análise dos resultados dos experimentos realizados e a validação dos objetivos no Capítulo 4. Por fim são apresentadas as conclusões, as considerações finais sobre o trabalho realizado e trabalhos futuros no Capítulo 5.

Capítulo 2

Trabalhos relacionados

Este capítulo apresenta os trabalhos relacionados ao tema de detecção de eventos que foram encontrados na revisão bibliográfica realizada. A estrutura do capítulo inicia com a apresentação na Seção 2.1 da discussão dos trabalhos encontrados e as definições dos conceitos utilizados no trabalho. Durante a seção são diferenciadas as formas de detecção de eventos com base em lógica (Subseção 2.1.2) e com base em aprendizagem (Subseção 2.1.3). Em seguida é apresentado a discussão sobre arcabouços na Seção 2.2, as aplicações relacionadas a pesquisa na Seção 2.3 e finalizando, é apresentado as considerações finais sobre o capítulo na Seção 2.4

2.1 Detecção de eventos

Segundo a definição do *Cambridge Dictionaries Online* (CAMBRIDGE, 2014), um evento é algo que acontece, em especial utilizado para denotar algo importante ou anormal. Um evento pode ser considerado um acontecimento único ou uma sequência de acontecimentos em um determinado tempo (GULER; LIANG; PUSHEE, 2003). Então, saber **quais** são os eventos e **quando** eles acontecem é essencial para o desenvolvimento de algumas atividades, algumas delas de suma importância para a sobrevivência, como em atividades de caça ou até mesmo ao andar no trânsito. Com o avanço das tecnologias da informação e da comunicação e o aumento da automatização das atividades antes executados por humanos, muitos eventos passaram a ser detectados de forma automática ou semiautomática (com supervisão humana), agilizando e/ou melhorando as atividades. Várias pesquisas têm se voltado para automatizar o processo de detecção (**quando**) e reconhecimento (**quais**) de eventos, sejam eles em vídeos, redes (XIAO et al., 2012), negócios (LINEHAN et al., 2011), entre outros.

Em algumas pesquisas, como a de Xiao et al. (2012), a detecção e o reconhecimento de eventos auxiliam ou automatizam a tomada de decisão. Na pesquisa deste autor, ele estipulou regras que definiam os evento a partir de um conjunto de pequenos eventos detectados pelo monitoramento do sistema operacional de dispositivos móveis. Estes pequenos eventos são chamados pelo o autor de even-

tos primitivos. Os eventos definidos pelas regras são utilizados pra definir o uso da rede sem fio do dispositivo.

Em vídeos, Na, Qin e Wright (2011) utilizaram fluxo óptico para detectar objetos nos quadros do vídeo, criando um modelo dos objetos a serem detectados por meio de uma soma de vetores e assim reconhecendo-os e rastreando-os. Em seguida, relacionava os objetos detectados com outros objetos detectados e com áreas demarcadas no vídeo para detectar eventos que acarretam situações de perigo para crianças, como por exemplo, criança se aproximando de objetos desconhecidos parados, criança subindo em locais perigosos e crianças saindo da área de monitoramento.

Eventos também são muito utilizados na área de esportes, como na pesquisa de Hosseini e Eftekhari-Moghadam (2013). O autor extrai características do áudio e do vídeo, e as utilizam para detectar eventos em partidas de futebol e sumarizar a partida. As características extraídas variam desde histogramas até a indentificação de objetos complexos, como logomarcas.

Nas pesquisas de Qian et al. (2012a) e Qian et al. (2012b), os autores também utilizaram a detecção de vídeo por meio da extração de características, e do treinamento de uma HCRF (*Hidden Conditional Random Field*) e de uma HMM (*Hidden Markov Model*), respectivamente. Ambos os trabalhos trataram da detecção de eventos em vídeos de futebol, identificando eventos como faltas, gols, chutes normais, cobranças de faltas dentre outros.

Pixi, Hongyan e Wei (2010) e Tavassolipour, Karimian e Kasaei (2014) também fizeram uso de HMM para a detecção de eventos em partidas de futebol, com a diferença de que Tavassolipour, Karimian e Kasaei (2014) utilizaram o HMM na fase de treinamento da Redes Bayesianas (BN - *Bayesian Network*), juntamente com funções de distribuição cópula para a detecção dos eventos. Wang e Ngo (2012) sumarizaram por meio de uma HHMM (*hierarchical hidden Markov model*) trechos de vídeos para o reuso de cenas importantes.

2.1.1 Tipos de detecção de eventos

Um vídeo é uma sequência de imagens, e a detecção de eventos em um vídeo se dá não só pela interpretação dos componentes de uma imagem, mas sim, pela sequência de imagens que compõe o vídeo, o que caracteriza o mais alto nível de processamento de imagens. Tal afirmação está alinhada com a visão de Gonzalez e Woods (2006), que divide o processamento de imagem em 3 níveis: baixo, médio e alto, em que o mais alto nível visa extrair significados por meio da interpretação de componentes da imagem. Esses processos de detecção podem ser classificados, como sugere Piciarelli e Foresti (2011), em dois grandes grupos: (1) detecção de eventos explícitos, em que se tem conhecimento prévio do evento a detectar e (2) a detecção de eventos anormais, em que há apenas o conhecimento prévio do evento que não se deve detectar (detectando eventos que não são conhecidos). Destes dois grupos abordamos neste trabalho a detecção de eventos explícitos em quadros de vídeo. Para esse grupo, autores utilizam

abordagens com base em (1) lógica, assim como Na, Qin e Wright (2011), Shet, Harwood e Davis (2005), Albanese et al. (2008), Males e Zarnic (2011) e Ghanem et al. (2004), em (2) classificadores treinados, como Aoun, Elghazel e Amar (2011), Chen e Tsai (2014) e Chuang et al. (2014) e em (3) abordagens híbridas, como Guler, Liang e Pushee (2003) e Tjondronegoro e Chen (2010) que mesclam lógica e classificadores. Nas Seções 2.1.2 e 2.1.3 são apresentadas pesquisas que empregam lógica e classificadores para a detecção de eventos, respectivamente.

Eventos podem atingir um alto grau de complexidade, o que faz Onal et al. (2013) afirmar que detectar eventos complexos a partir de vídeos é uma tarefa difícil, uma vez que tais eventos são compostos por vários componentes diferentes, cada um complexo por si próprio. Sendo assim, é comum dividir um evento complexo em eventos mais simples, denominados por Zhai et al. (2009) de eventos primitivos. A detecção de determinados eventos primitivos com ou sem o contexto temporal pode resultar na detecção do evento complexo. Cada evento, seja ele simples ou complexo, é o resultado da soma e/ou de relacionamentos dos componentes detectados e/ou rastreados nos vídeos, quer seja por meio de fluxo óptico, subtração de plano de fundo, extração de características, entre outros.

Das técnicas mais utilizadas estão a SVM (Máquina de Vetor de Suporte) e os HMM (Modelos Ocultos de Markov) . Porém, ambas necessitam de treinamento. Tavassolipour, Karimian e Kasaei (2014) utilizam HMM, e fazem comparações com técnicas que utilizam SVM. Os autores mostram experimentalmente a superioridade de HMM. Alguns autores utilizam lógica ou modelos lógicos, alguns destes modelos fazem uso de recursos estatísticos como no caso das Redes Lógicas de Markov ou inclusão de propriedades encontradas na lógica difusa. Linguagens de programação lógica como CLIPS e Prolog ou até mesmo a utilização de lógica diretamente no fluxo de controle da aplicação, ou seja, programar as regras por meio da linguagem utilizada para programar o sistema, utilizando as estruturas da própria linguagem de programação.

2.1.2 Detecção de eventos com base em lógica

A detecção de eventos com base em lógica normalmente se utiliza de regras ou modelos pré-estabelecidos para detectar os eventos. Tais modelos podem ser codificados diretamente no programa, utilizando da própria linguagem de programação ou por arquivos externos que utilizam uma estrutura previamente conhecida para armazena-los. Autores como Na, Qin e Wright (2011) e Meghdadi e Irani (2013) fazem a descrição do modelo diretamente no código, em que toda a estrutura de definição do evento é criada no momento em que se codifica o software, a partir de estruturas de laço e repetição. Essa abordagem pode tornar o desenvolvimento mais simples, uma vez que não é necessário implementar toda uma estrutura de algum arcabouço ou até mesmo implementar técnicas ou teorias mais sofisticadas. Porém, esta abordagem dificulta a adaptação para outros fins, fora o que foi programado inicialmente. Outros autores fazem uso de estruturas mais complexas, como no caso das Redes de Petri utilizadas por Alba-

nese et al. (2008) e Ghanem et al. (2004), as quais consistem em um conjunto de lugares, arcos, transições e marcas formando uma espécie de grafo direcionado com comentários, e assim a presença de uma marca em um dos lugares irá ocasionar uma transição levando-a a outro lugar, como no exemplo da Figura 2.1 no qual o autor modela um pessoa entrando em um carro.

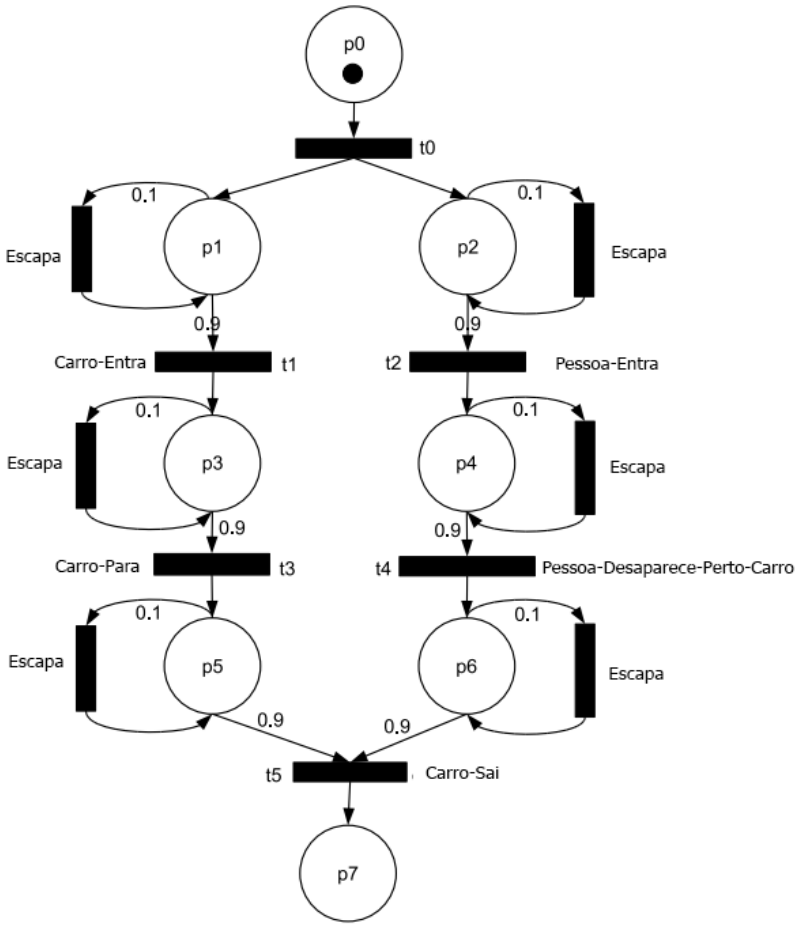


Figura 2.1: Exemplo de uma rede de Petri usada por Albanese et al. (2008) (ALBANESE et al., 2008).

Autores como, Onal et al. (2013) e Kardas, Ulusoy e Cicekli (2013) utilizaram Redes Lógicas de Markov, que consistem na combinação de lógica de primeira ordem e modelos probabilísticos dispostos em um modelo similar a uma máquina de estados ou grafo não direcionado, que permite modelar o conhecimento por meio da lógica e adicionar a incerteza por meio das probabilidades.

SanMiguel e Martinez (2011) utilizaram Máquinas de Estado Finito, que consistem em um conjunto de estados, para os quais, partindo-se de um estado inicial pode-se transitar para outros estados por meio de funções de transição. O que acarreta que o estado atual tem um histórico do estado anterior, como no exemplo da Figura 2.2, que modela um objeto sendo abandonado. Na pes-

quisa de Hakeem (2007), grafos são empregados na modelagem de eventos, em que as arestas são associadas a pesos e os vértices a eventos primitivos, que quando combinados disparam o evento modelado. Hosseini e Eftekhari-Moghadam (2013) utilizaram de lógica difusa, que adiciona valores intermediários entre os valores verdadeiro e falso da lógica tradicional, assim acrescentando incerteza à modelagem realizada. Também são utilizadas linguagens de programação lógicas como Prolog (SHET; HARWOOD; DAVIS, 2005) e CLIPS (*C Language Integrated Production System*) (ALMEIDA, 2010; GULER; LIANG; PUSHEE, 2003) as quais proporcionam estrutura lógicas para solução de proposições.

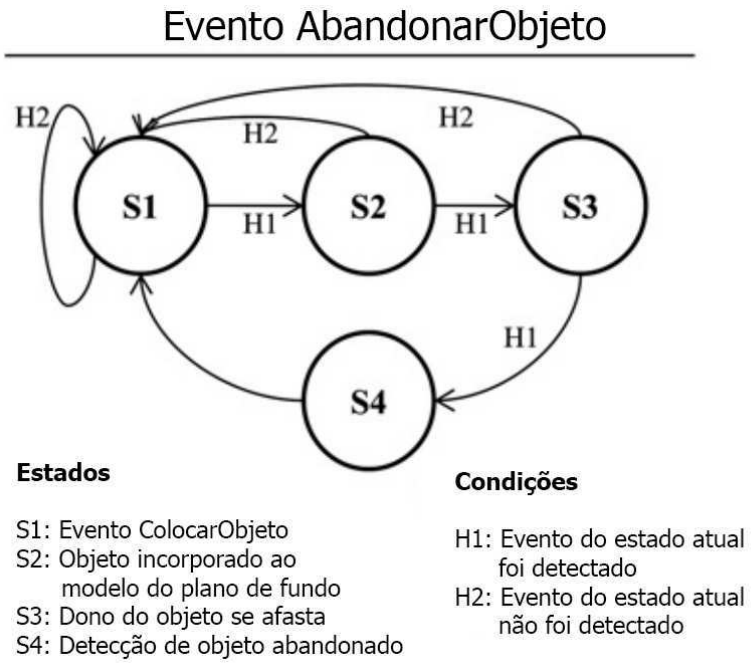


Figura 2.2: Exemplo de uma máquina de estados finitos usada por SanMiguel e Martinez (2011) (SANMIGUEL; MARTINEZ, 2011).

Obviamente esses sistemas costumam apresentar bons resultados quando as entradas para o sistema de inferência lógica estão corretas e os modelos dos eventos bem definidos, sendo assim pequenas falhas em um desses pontos podem fazer com que o evento não seja detectado.

2.1.3 Detecção de eventos com base em aprendizagem

As abordagens de detecção de eventos que utilizam aprendizagem têm como base as técnicas de aprendizagem de máquina, em que os eventos são detectados a partir de um modelo previamente aprendido. Técnicas de aprendizagem se apoiam no que se deve aprender e em como se deve guardar o conhecimento adquirido. Cada imagem é formada por vários *pixels*, o que pode tornar complexa a aprendizagem realizada em cada *pixel*. Dessa forma, com o propósito de

quebrar esta complexidade, é comum a aplicação de estratégias para a extração de características e representação, dessa forma, simplificando o modelo a ser armazenado, como por exemplo, grafos formados por características de movimento utilizados por Aoun, Elghazel e Amar (2011), os vetores de movimento e histogramas de gradiente de Chuang et al. (2014) e as caraterísticas SIFT (*Scale Invariant Feature Transform*), STIP(*Space-time interest points*) e MFCC (*Mel-Frequency Cepstral Coefficients*) de Jhuo et al. (2014).

Estratégias de aprendizagem são comumente dividias em três tipos (RUSSELL; NORVIG, 2002) : **supervisionada**, **não supervisionada** e **por reforço**, em que as duas primeiras são comumente aplicadas para detecção de eventos, assim temos que:

- **Supervisionada:** É utilizada quando a base de dados possui tanto entradas quanto as saídas, ou seja, possui os componentes detectados e os eventos a eles vinculados. A partir do treinamento é possível generalizar saídas para dados similares que não fizeram parte do treinamento. Esta forma de aprendizagem é largamente utilizada na detecção de eventos explícitos, quando se sabe qual evento se deve detectar.

- **Não supervisionada:** É utilizada quando a base de dados possui apenas as entradas e não se sabe as saídas. Assim, a aprendizagem normalmente ocorre a partir do agrupamento dos dados a partir da análise de similaridade entre as características de entrada. Esta forma de aprendizagem é comumente usada para a detecção de eventos anormais, em que os eventos são agrupados, sendo considerados anormais aqueles eventos cujas características diferem muito dos grupos formados.

- **Por reforço:** É utilizada quando a base de dados possui as entradas e indicativos qualitativos das saídas, Assim a aprendizagem ocorre a cada saída, que é avaliada qualitativamente. Esta avaliação é utilizada para retreinar os indicadores qualitativos de saída.

Definidos o tipo de componente e a forma de aprendizagem, são então aplicadas técnicas de aprendizagem específicas ao problema. Dentre os tipos de aprendizagem, o mais utilizado para detecção de evento é a aprendizagem supervisionada. Dentre as técnicas de aprendizagem supervisionada mais utilizadas, tem-se a Máquina de Vetor de Suporte utilizada por Aoun, Elghazel e Amar (2011), Chuang et al. (2014), Jhuo et al. (2014), Kamishima, Inoue e Shinoda (2013), Mota et al. (2013), Fleischman, Decamp e Roy (2006), Merler et al. (2012), Aoun, Elghazel e Amar (2011), em que, dado duas classes distintas, constrói-se um hiperplano de margem máxima de separação entre as classes. Há também os Modelos Ocultos de Markov utilizados por Chen e Tsai (2014), Guler, Liang e Pushee (2003), Pixi, Hongyan e Wei (2010), Tavassolipour, Karimian e Kasaei (2014), que constituem modelos estatístico de Markov com estados ocultos, nos quais a partir dos estados observáveis e das funções de probabilidades

de mudança de estados é possível inferir sobre novas entradas. Nesta técnica os dados da base são utilizados para formar as probabilidades de transição dos estados.

A Figura 2.3 apresenta um modelo de um HMM em que as funções de interação e de estados são os estados observáveis e as saídas são a ocorrência ou não do evento. Variações da técnica HMM, como os *Noise Hidden Markov Models* (Modelos Ocultos de Markov Ruidosos) de Mutschler e Philippsen (2012) que empregam estratégias para remover os ruídos indesejados da HMM e a *hierarchical hidden Markov model* (Modelos Ocultos de Markov Hierárquico) de Wang e Ngo (2012) também são utilizadas para detecção de eventos em vídeo.

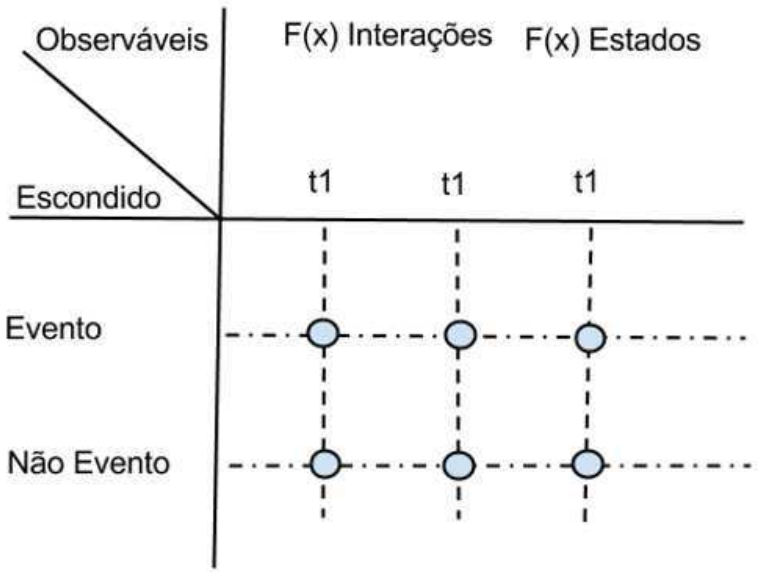


Figura 2.3: Exemplo de Modelos Ocultos de Markov.

As detecções com base em aprendizagem apoiam-se em características extraídas de um conjunto de treinamento composto por quadros de vídeo, nos quais ocorrem os eventos a serem detectados. Tais técnicas utilizam classificadores treinados por meio de algoritmos de aprendizagem (RUSSELL; NORVIG, 2002) para, a partir de características específicas, detectar a ocorrência de eventos em quadros que apresentem características similares às que foram treinadas. Essa abordagem é amplamente utilizada para a detecção de eventos em quadros de vídeo. Dentre os classificadores mais comumente utilizados tem-se HMM (Modelos ocultos de Markov) (CHEN; TSAI, 2014; GULER; LIANG; PUSHEE, 2003; PIXI; HONGYAN; WEI, 2010; TAVASSOLIPOUR; KARIMIAN; KASAEI, 2014), suas variações (MUTSCHLER; PHILIPPSEN, 2012) e SVM (AOUN; ELGHAZEL; AMAR, 2011; CHUANG et al., 2014; JHUO et al., 2014; KAMISHIMA; INOUE; SHINODA, 2013; MOTA et al., 2013; FLEISCHMAN; DECAMP; ROY, 2006; MERLER et al., 2012; AOUN; ELGHAZEL; AMAR, 2011). Essas técnicas costumam ter bons resultados quando as características escolhidas representam pontos importantes das imagens em questão (e.g. contornos, linhas e texturas), e os dados de trei-

namentos contém estas características. Porém a necessidade de uma base de dados para treinamento torna essas abordagens difíceis de se adaptar a novas situações. Devido a necessidade de encontrar novas bases de dados que contenham o novo evento a ser detectado.

2.2 Arcabouços

Para esta dissertação, um arcabouço é considerado uma estrutura genérica, com conexões ou configurável para um determinado domínio. Muitos autores definem seus sistemas como arcabouços devido à possibilidade de reconfiguração, configurações essas que muitas vezes não são simples de se realizar. Xu et al. (2011) criaram um arcabouço que detectava eventos simples por meio de HOG (Histogramas de gradientes orientados) (DALAL; TRIGGS, 2005) para detectar os objetos e *Mean-Shift* (COMANICIU; MEER, 2002) para rastreá-los criando uma lista de eventos ocorridos, ao final, comparando a ocorrência desses eventos com as regras anteriormente estipuladas, seguindo o fluxograma da Figura 2.4. Para a reconfiguração do sistema, adaptando-o para detectar outros eventos seria necessário um novo treinamento do HOG e novas regras.

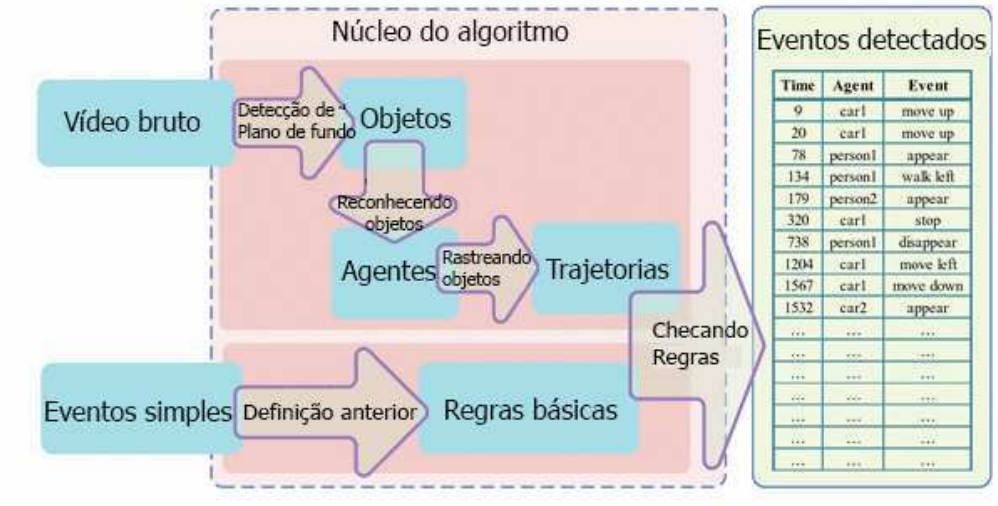


Figura 2.4: Fluxograma do arcabouço de Xu et al. (2011) (XU et al., 2011).

Albanese et al. (2008), Onal et al. (2013), Chen e Tsai (2014), Wang e Ngo (2012), Guler, Liang e Pushee (2003), Chattopadhyay e Das (2014),Lim, Tang e Chan (2014), Li e Porikli (2004) e Lu et al. (2009) apoiam-se no termo arcabouço na vertente da configurabilidade dos sistemas desenvolvidos por meio das pesquisas propostas, nos quais a partir de novo treinamento das máquinas de aprendizagens ou alteração em arquivos ou código fonte pode-se configurar a ferramenta para outras finalidades. Contudo, trabalhos como o de Bayat et al. (2013), que tem um objetivo muito específico, detectar gols em vídeos de futebol,

não possuem generalidade, devido conter um domínio bem definido, detectando apenas gols em partidas de futebol.

Trabalhos como o de Kovacs et al. (2009) fazem uso do conceito de extensibilidade, em que funções podem ser estendidas por meio de módulos ou adição de novas classes. Porém, o autor apenas alerta o usuário dos eventos detectados pelos módulos.

2.3 Aplicações relacionadas a esta pesquisa

Dos trabalhos revisados, os que mais se assemelham ao tema proposto são os de Yung e Lai (2001), Shet, Harwood e Davis (2005), Krausz e Herpers (2010), Almeida (2010), D'Odorico (2013) e Lim, Tang e Chan (2014) nos quais os autores utilizam de lógica para detectar os eventos. Todos os trabalhos partem da extração de características dos vídeos, as quais são utilizadas pela máquina de inferência. A partir de um conjunto de regras pré-estabelecidas, é possível verificar se houve ou não a ocorrência de um determinado evento.

Dentre as pesquisas citadas, temos a de Yung e Lai (2001) que detecta os veículos que ultrapassam o sinal vermelho. Para tal, é necessário uma inicialização do sistema para detectar o semáforo, o tempo de transição e a faixa de pedestre, para em seguida serem detectados os veículos que realizam determinada infração. Os autores utilizam vídeos de apenas um cenário e apenas vídeos diurnos, conseguindo uma detecção declarada de 100% dos eventos.

Shet, Harwood e Davis (2005) utilizaram de subtração de plano de fundo adaptativo para detectar e rastrear os objetos nos quadros e Prolog na construção e interpretação das regras. Assim, foi possível inferir se as pessoas entraram em área restrita, perseguição e furto de pacotes.

Entretanto, Krausz e Herpers (2010) afirmam que essa construção em Prolog utilizada por Shet, Harwood e Davis (2005) se torna lenta, pois a rotina que atualiza os fatos na máquina de inferência é chamada a cada 5 segundos. Para contornar este fato, os autores empregaram CLIPS permitindo que o sistema opere em tempo real. Os autores utilizam de subtração de plano de fundo adaptativa para detectar os objetos. Eles aumentam ou diminuem o quanto dos novos quadros de vídeos são incorporados ao plano de fundo e com isso conseguem remover ruídos antes de enviar os dados para a máquina de inferência. essa adaptação também é utilizada para compensar a mudança de iluminação e para conservar por mais tempo os objetos detectados que pararam de se movimentar.

Com o foco na vigilância de estações de trem e controle de riscos, o MetroSurv, sistema desenvolvido por Krausz e Herpers (2010) faz uso de demarcações de áreas no vídeo para relacionar com os objetos detectados e alertar as situações de perigo, e.g. pessoas andando nos trilhos e objetos sendo jogados entre as plataformas.

Almeida (2010) também faz uso do CLIPS, porém ele adiciona um elemento novo, que é a contextualização temporal do evento. O autor, assim como outros

autores em trabalhos já citados, faz uso da álgebra de intervalo de Allen (ALLEN, 1983) para modelar o tempo em que os acontecimentos ou eventos primitivos são detectados, de forma a criar uma conexão temporal entre as detecções e assim validar o disparo da regra em questão. Porém, diferentemente dos dois trabalhos citados anteriormente, a validação é realizada com dados sintéticos, os quais nem sempre correspondem a uma cena real.

D'Odorico (2013), em seu trabalho, desconsidera a fase de extração de elementos dos quadros de vídeos, mantendo o foco na máquina de inferência e na análise da vaguidão (o quão ambíguo pode ser) dos verbos de movimento (verbos que definem ações, e.g. mover e tocar). Dessa forma, as entradas para o sistema são proveniente de anotações dos vídeos. O autor contextualiza os verbos de movimento selecionados no trabalho atribuindo às ações descritas um significado bem definido. O sistema tem como base a linguagem Prolog e utiliza como base para contextualização temporal o *Event Calculus* (KOWALSKI; SERGOT, 1986), que é um formalismo em que os pontos no tempo são explícitos, possibilitando que uma proposição seja verdadeira em um determinado ponto no tempo. Devido ao foco do trabalho ser a análise da vaguidão dos verbos de movimento, só foram implementados e testados apenas dois verbos: *Approach* (Aproximar) e *Hold* (Segurar).

O sistema proposto e implementado na pesquisa de Lim, Tang e Chan (2014) assemelha-se a esta pesquisa por ser um arcabouço com foco na detecção de múltiplos eventos, utilizando de lógica para as regras que determinam a ocorrência ou não do evento. Ele utiliza técnicas para detecção de movimento, detecção de classes de objetos, junção e separação de objetos entre outras técnicas para determinar a ocorrência de um dos 5 eventos propostos em uma área de interesse previamente mapeada no vídeo.

Dos trabalhos citados nesta seção, os autores utilizaram bases próprias ou eventos diferentes, não possibilitando realizar testes nas mesmas bases. O que acarretou a criação de uma pequena base pública para esta dissertação. Os demais trabalhos revisados utilizaram de bases públicas como, a CANTATA (KARDAS; ULUSOY; CICEKLI, 2013) , UFC (MOTA et al., 2013), TRECVID (JHUO et al., 2014) e INRA (MEGHDADI; IRANI, 2013), que são voltadas para detecção de ações (por exemplo, andar, correr, jogar e fazer um bolo), ou criaram as próprias bases (por exemplo, gravações de trânsito, de jogos e de pessoas). Na Tabela 2.2 pode-se observar que a maioria das bases são próprias. Na Tabela 2.1, são listadas as principais técnicas e resultados obtidos pelos autores.

Relacionando os dados apresentados nas tabelas citadas anteriormente temos que, parte dos autores optam por criar as próprias bases. Os autores que utilizam de bases públicas se restringem aos rótulos nelas contidas. Também constata-se que a maioria dos autores fazem uso de técnicas com base em aprendizagem.

Esta pesquisa tem como diferencial, a reprodutibilidade dos resultados, a extensibilidade do arcabouço e a não necessidade de treinamento. Quando comparada com as pesquisas citadas nesta seção, temos que: a pesquisa desenvolvida nesta dissertação diferencia-se das pesquisas de Shet, Harwood e Davis (2005) e

Krausz e Herpers (2010) por trazer a possibilidade de extensão das funcionalidades por meio de *plug-ins*; diferencia-se do trabalho de D'Odorico (2013) por realizar tratamento em vídeos de situações reais, não apenas com anotações ou com dados sintéticos, como no trabalho apresentado por Almeida (2010); diferencia-se do trabalho de Lim, Tang e Chan (2014) por apresentar formas de extensão dos módulos de detecção e dos eventos que podem ser detectados, como também apresenta formas de modificar as regras estipuladas; e de Yung e Lai (2001) por utilizar mais de um cenário e ter uma análise de resultado mais robusta.

2.4 Considerações finais

Ao final do capítulo de revisão, conclui-se que há uma dificuldade em encontrar bases públicas para detecção de eventos em vídeos, para os eventos abordados nessa pesquisa. A maioria das bases encontradas na revisão que se assimilavam ao propósito desta pesquisa, ou que poderiam ser utilizadas nos testes eram próprias e não disponibilizadas para download. Isso acarretou uma dificuldade em comprar os resultados desta pesquisa com outras existentes.

A maioria dos trabalhos revisados destacam a utilização de técnicas com base em aprendizagem, o que torna a adaptação para outros eventos dependente de uma nova base de dados para testes. Entre as técnicas com base em lógica ou modelos lógicos, as mais adaptáveis contam com modelos ou regras que podem ser alterados para se adaptar a novas situações, porém esses modelos ficam limitados às funções ou objetos detectados pelos sistemas, que na maioria dos trabalhos apresentados não são extensíveis.

Assim temos uma lacuna em sistemas com base em lógica, adaptáveis e extensíveis. O sistema proposto nesta pesquisa aborda as três características citadas anteriormente.

Tabela 2.1: Técnicas e resultados.

Autor	Técnica	Resultados
Chen e Tsai (2014)	Modelos ocultos de Markov	Acurácia de 52,1%.
Lu et al. (2009)	SVM com pesos	Sem experimento
Bayat et al. (2013)	Mistura de gaussianas em modelos ocultos de Markov	Taxa de detecção: 93% e de erro: 3%.
Mota et al. (2013)	Máquina de vetores de suporte	Taxa de acerto de 92,5% no KTH.
Jhuo et al. (2014)	Máquina de vetores de suporte e Grafos	*mean of average precision* entre 1,16% a 7,36%.
Tavassolipour, Karimian e Kasaei (2014)	Modelos ocultos de Markov e redes Bayesianas	Acurácia de 91,04%.
Kamishima, Inoue e Shinoda (2013)	Máquina de vetores de suporte	MNDC de 0,510.
Hosseini e Eftekhari-Moghadam (2013)	Modelos ocultos de Markov e árvores de decisão	Acurácia e F1-Score médios de 81,78% e 78,67%.
Aoun, Elghazel e Amar (2011)	Grafos e Máquina de vetores de suporte	Acurácia entre 73,8% e 76,2%.
Qian et al. (2012a)	Campo aleatório condicional oculto (HCRF)	F1-Score de 85,5% .
Qian et al. (2012b)	Modelos ocultos de Markov	F1-Score de 82,92%.
Meghdadi e Irani (2013)	Cubos de espaço tempo	*Normalized time-to-completion* entre 0,1 a 0,15.
Tjondronegoro e Chen (2010)	Modelo estatístico	Precisão e revocação entre 68,53% a 89,06% e 82,42% a 90,32%.
Wu e Hu (2014)	*Temporal Pyramid Model* (TPM)	Acurácia entre 49,46% e 86,01%.
Hakeem (2007)	Grafos	Precisão média de 79,71% e 91,19%.
Fleischman, Decamp e Roy (2006)	Máquina de vetores de suporte	Precisão entre 60% a 90% e revocação entre 66% e 76%.
Chuang et al. (2014)	Máquina de vetores de suporte	Acurácia entre 52,8% a 100%.
Pixi, Hongyan e Wei (2010)	Modelos ocultos de Markov	Precisão entre 86,36% e 91,67% e revocação entre 91,67% e 100%.
Wang e Ngo (2012)	Modelos ocultos de Markov hierárquicos (HHMM)	Acurácia entre 70% e 96%
Chattopadhyay e Das (2014)	Support Vector Data Description (SVDD)	Taxa de erro média de 14,35%
Merler et al. (2012)	Máquina de vetores de suporte	*mean of average precision* de 0,46
SanMiguel e Martinez (2011)	Máquina de estado finito	Precisão entre 0,60 e 0,64 e revocação entre 0,45 a 0,55.
Onal et al. (2013)	Redes lógicas de Markov	Tabela com eventos encontrados
Shet, Harwood e Davis (2005)	lógica com Prolog	Eventos encontrados.
Albanese et al. (2008)	Redes de Petri	Obteve um F1-Score entre 60 e 100 para blocos de vídeos e 40 a 60 para quadros de vídeos, dados apresentados no gráfico.
Tian et al. (2010)	Lógica	Precisão de 92% e revocação de 78%.
Guler, Liang e Pushee (2003)	Modelos ocultos de Markov e CLIPS	Acurácia de 97%.
Yung e Lai (2001)	Lógica	Taxa de detecção de 100% e taxa de erro de 0%.
D'Odorico (2013)	lógica com Prolog	Taxa de detecção de 67,26% e taxa de erro de 12,27% para o evento "aproximar" e 65,30% e 38,33% para o evetno "segurar".
Zhai et al. (2009)	Lógica	Taxa de acerto de 100% porém com taxa de falso positivo de 60%.
Na, Qin e Wright (2011)	Lógica	Taxa de acerto entre 75,51% a 1,43%.
Lim, Tang e Chan (2014)	Lógica	Acurácias entre 80% e 88,46%.
Kardas, Ulusoy e Cicekli (2013)	Redes lógicas de Markov	Precisão e revocação de 100%.
Krausz e Herpers (2010)	lógica com CLIPS	Taxa de acerto entre 83% e 100%.
Almeida (2010)	lógica com CLIPS	Demonstra que os eventos são encontrados.

Tabela 2.2: Eventos detectados e bases de dados.

Autor	Eventos	Base de dados
Albanese et al. (2008)	Roubo de Bancos e atividades do aeroporto	bank Dataset e TSA Dataset
Onal et al. (2013)	Deixar objetos e encontro de pessoas	PET-2006, CANTATA e base própria.
Chen e Tsai (2014)	Eventos de futebol	Base própria.
Bayat et al. (2013)	Congestionamento pesado, alta densidade com e sem alta velocidade, alta densidade com e sem alta velocidade e vacância	Base própria.
Tian et al. (2010)	Gol	Gravação da TV, base própria.
citeonlineGuler2003	Juntar, separar, Aglomeração, dispersão, pegar objeto e largar objeto	Base própria.
Yung e Lai (2001)	Ultrapassagem sinal vermelho	Base própria.
D'Odorico (2013)	Aproximar e segurar	Anotações da base de dados da DARPA.
Zhai et al. (2009)	Falso escaneamento do produto no caixa	Base própria.
Mota et al. (2013)	Eventos rotulados nas bases de dados	KTH Database, UFC e Hollywood2.
Na, Qin e Wright (2011)	Criança perto da mobilha, fora do chão e seguro no chão	Base própria.
Jhuo et al. (2014)	Eventos rotulados nas bases de dados	TRECVID e CVV Dataset.
Tavassolipour, Karimian e Kasaei (2014)	Gol, falta, cartão e etc..	Base própria.
Kamishima, Inoue e Shinoda (2013)	Eventos contidos na base	TRECVID MED.
Hosseini e Eftekhari-Moghadam (2013)	Eventos de futebol	Gravação de jogos de futebol.
Aoun, Elghazel e Amar (2011)	Pessoa caminhando e correndo	TRECVID e INRIA Dataset.
Qian et al. (2012a)	Eventos de futebol	Base própria.
Qian et al. (2012b)	Eventos de futebol	Base própria.
Meghdadi e Irani (2013)	Apresenta uma tabela de eventos detectados	Base própria.
Lim, Tang e Chan (2014)	Vadiagem, invasão, cair ao dormir, atividade anormal em multidões e Abandonar objetos	Vídeos selecionados das bases PTES, CANTATA, UMN e do Youtube.com.
Tjondronegoro e Chen (2010)	Eventos de futebol	Base própria.
Kardas, Ulusoy e Cicekli (2013)	Encontrar e deixar objetos	CANTATA e Base própria.
Wu e Hu (2014)	Eventos rotulados nas bases	Base de dados: HMDB51 e UCF50.
Hakeem (2007)	Encontros, vigilância e monitoramento de trilhos de trem	Base de dados: NIST; Kojima; Sadiye; VACE; PETS; CAVIAR; ETISEO; Alexei; FDOT; Meeting; Surveillance; Railroad.
Krausz e Herpers (2010)	Pessoa presa na porta de um trem em movimento e caminhando nos trilhos.	Base própria.
Fleischman, Decamp e Roy (2006)	Fazer café, guardando os pratos e bebendo algo	HSP dataset.
Chuang et al. (2014)	Eventos rotulados nas bases de dados	Bases de dados: KTH , Weizmann, UFC e UFC50.
Pixi, Hongyan e Wei (2010)	Chutes e faltas	Gravações de jogos de futebol.
Wang e Ngo (2012)	Eventos rotulados nas bases de dados	Bases de dados: TRECVID 2007, TRECVID 2008 e BBC rushers.
Chattopadhyay e Das (2014)	Eventos rotulados nas bases de dados	Bases de dados: UCSD e Anomalous Behavior.
Merler et al. (2012)	Eventos rotulados nas bases de dados	TRECVID MED.
Almeida (2010)	Violação do semáforo e abordagem a carro	Dados sintetizados.
SanMiguel e Martinez (2011)	Colocar objeto, pegar objeto, abandonar objeto e roubar objeto	Bases de dados: PETS 2006, PETS 2007, AVSS 2007 , VISOR, CANDELA, WCAM e ChromaVSG.
Shet, Harwood e Davis (2005)	Escolta, entrada autorizada e entrada não permitida	Base própria.

Capítulo 3

Arcabouço proposto

Neste capítulo, é apresentado o arcabouço proposto para a detecção de eventos em vídeo. O arcabouço é formado por estruturas funcionais e componentes lógicos com interface de comunicação previamente especificada. O projeto do arcabouço objetivou prover um alto nível de abstração entre os componentes. Esse modelo arquitetural foi escolhido para facilitar a extensibilidade do arcabouço, possibilitando: (1) a reutilização em contextos diferentes daqueles para os quais o arcabouço foi inicialmente proposto; (2) facilitar a extensibilidade do arcabouço por meio da adição de quaisquer componentes que implementem a interface do sistema; (3) substituição de componentes sem a necessidade de alteração de outros componentes do sistema.

A construção do arcabouço foi realizada em C++, utilizando componentes externos à linguagem, como OpenCV (OPENCV, 2015), Jsoncpp (JSONCPP, 2015) e Clipper C++ (CLIPPER, 2015). A escolha da linguagem C++ se deu pela velocidade de processamento, a portabilidade e a integração com o OpenCV. O Jsoncpp e o Clipper foram escolhidos por apresentarem simplicidade na forma de uso e uma alta produtividade. O resultado desse trabalho gerou o arcabouço denominado RulesFramework. Ao final desta dissertação são apresentados nos apêndices um exemplo de um arquivo de configuração do detector de carros no Apêndice A, do detector de objetos fixos no Apêndice B e do detector da cor da luz do semáforo no Apêndice C. Também é apresentado no apêndice D uma descrição dos componentes externos a linguagem utilizados.

3.1 Arquitetura do arcabouço

O RulesFramework foi concebido como um arcabouço. Não foi projetado com uma ferramenta para usuário final, e sim, como meio para a criação de um produto final. Para tal, o arcabouço necessita de uma aplicação que faça uso do mesmo. Esse uso consiste na alimentação do sistema com dados e o tratamento das respostas à detecção. O arcabouço necessita receber quadro a quadro dos vídeos a serem analisados, para isso, necessita de fontes de vídeo e.g. câmeras ou arquivos de vídeo. Também necessita de um equipamento que possa interligar a

fonte de vídeo à aplicação que utiliza o arcabouço e.g. microcomputador ou um gravador digital de vídeo. A Figura 3.1 mostra o diagrama de implementação do arcabouço, que demonstra a necessidade de uma aplicação externa que receba as entradas e as repasse para o arcabouço

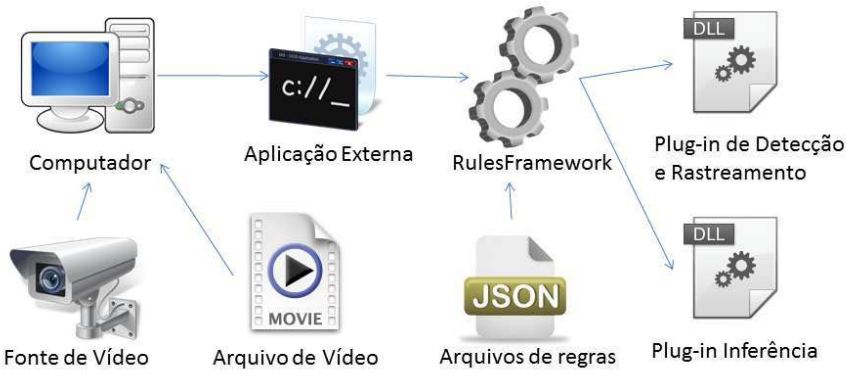


Figura 3.1: Visão de Implantação.

3.1.1 Visão geral

O arcabouço foi concebido sobre o modelo arquitetural com base em componentes (PRESSMAN, 2001). Tendo em vista a generalidade de uso, o arcabouço não contempla um programa principal executável que gerencia a aplicação como um todo. Tal componente de *software* deve ser fornecido por um desenvolvedor, o qual fará uso das bibliotecas do arcabouço para obter o resultado das detecções por ele encontradas com base no conjunto de regras recebidas.

Pacotes de *design* significativos do ponto de vista da arquitetura

Os componentes do arcabouço são formados por uma ou mais bibliotecas, das quais apenas a biblioteca principal, que contém o gerenciador de módulos, deve ser utilizada pela aplicação externa. Todo o arcabouço divide-se em três componentes principais: (1) componente de gerenciamento de módulos, que realiza o controle da interface e comunicação com aplicações externas; (2) componentes de detecção e rastreamento; e (3) componentes de inferência.

- **O componente principal RulesFramework** contém uma interface de comunicação com aplicações externas chamada IRules, que qualquer aplicação pode utilizar para acessar as funções disponibilizadas, instanciando um objeto do tipo IRules. Também contém uma interface para comunicação com os MÓDULOS, denominada IPlugin. Tal interface determina a forma de comunicação entre os módulos (tanto os de inferência como os de detecção) e o componente principal

- **O componente de detecção e rastreamento**, assim como o componente principal, este pode ser adicionado e removido sem que haja modificação no núcleo do arcabouço, como também pode ser construído por qualquer desenvolvedor, desde que ele implemente a interface IPlugin. Estes componentes em suma, recebem o quadro do vídeo provido pelo RulesFramework, o processa e retorna como resultado uma estrutura de dados com os resultados do processamento.

- **O componente de inferência**, assim como o componente principal, pode ser adicionado e removido sem modificação no núcleo do arcabouço. Também pode ser construído por qualquer desenvolvedor, desde que ele implemente a interface IPlugin. Estes componentes recebem as regras e as detecções de objetos do RulesFramework, as processam e retornam uma estrutura de dados com os resultados da detecção dos eventos. A Figura 3.2 apresenta o diagrama de classes contemplando a visão lógica do arcabouço.

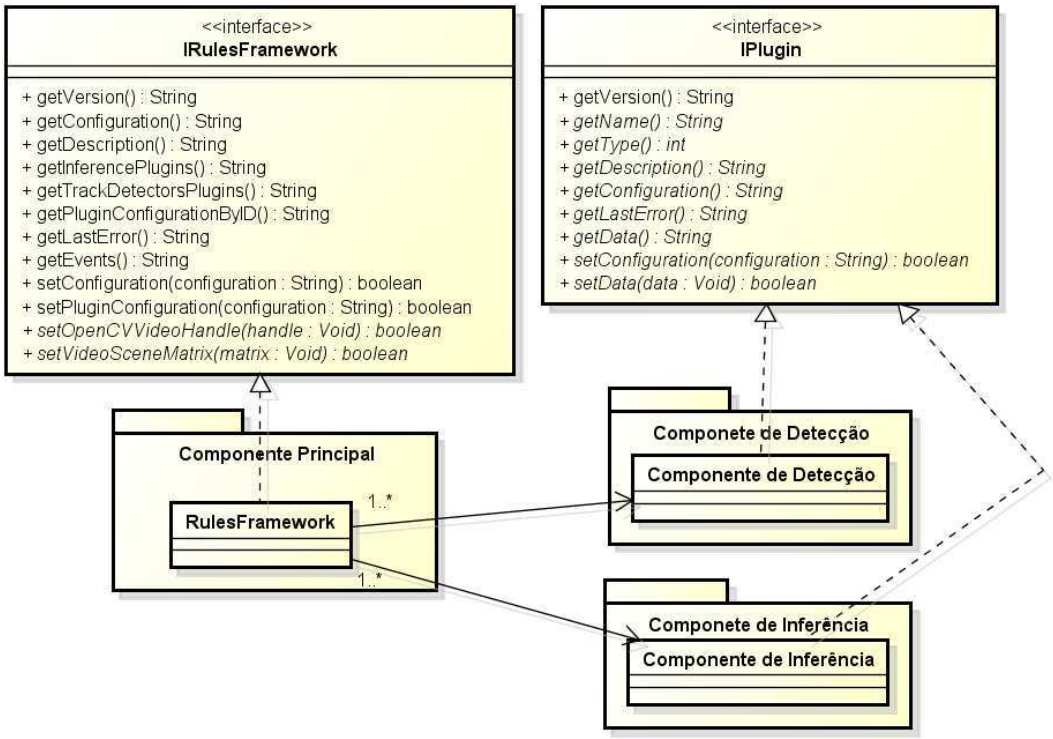


Figura 3.2: Visão lógica do arcabouço proposto.

3.1.2 Processo de detecção de eventos

O funcionamento do arcabouço segue os processos descritos nesta secção, que tem início com a inicialização da biblioteca principal do arcabouço (RulesFramework) por uma aplicação externa. Após a inicialização, a aplicação

carrega o RulesFramework, que é utilizado por meio da interface IRules. O processo segue com a configuração do arcabouço, em que as configurações estruturadas, seguindo o formato JSON (padrão escolhido para troca de mensagens), são solicitadas. A aplicação, se necessário, edita as configurações e reenvia o texto. Estas configurações contêm os parâmetros principais do arcabouço, tais como a localização dos arquivos de regras e a localização da pasta que contém os módulos. Em seguida, o RulesFramework carrega os arquivos de regras e os módulos e disponibiliza para a aplicação externa a lista dos módulos e as configurações individuais de cada um deles, sempre seguindo o mesmo padrão de troca de mensagens.

Após o processo de configuração, o arcabouço espera o recebimento de vídeo pela aplicação externa, o recebimento deve ser quadro-a-quadro. Após o recebimento, cada quadro é repassado para os módulos de detecção e rastreamento. O resultado pode ser reorganizado (caso necessário) e enviado para os módulos de inferência, juntamente com as regras que foram carregadas anteriormente. Com esses dados, o módulo de inferência realiza os testes, e retorna os eventos encontrados. Ao receber os eventos detectados, o arcabouço agrupa as mensagens recebidas dos módulos em uma única mensagem e a envia para a aplicação externa. O processo segue repetindo até a finalização do uso. As figuras a seguir apresentam os diagramas BPM (*Business Process Modeling*) dos processos citados anteriormente. A Figura 3.3 para o processo de uso e a Figura 3.4 para o de configuração.

3.1.3 Processo de inicialização dos módulos

O arcabouço, ao ser inicializado, necessita de configurações básicas para o funcionamento. Tais configurações podem ser passadas pela aplicação externa diretamente para o arcabouço ou requisitadas, editadas e reenviadas.

O processo ilustrado no diagrama da Figura 3.5, inicia-se com a criação da instância a ser utilizada. Em seguida, as configurações são enviadas (caso já tenha um padrão salvo) ou solicitadas para edição e envio (o arcabouço retorna todos os parâmetros em branco ou com valores padrão para o sistema). Após as configurações básicas, o arcabouço localiza os módulos (na pasta de módulos) e as regras (na pasta de regras). O arcabouço carrega os módulos em listas separadas os módulos de detecção e rastreamento e os módulos de inferência, deixando disponíveis à aplicação externa as configurações de cada módulo. o qual é editado ou enviado para o arcabouço reconfigurá-los.

3.1.4 Processo de utilização dos módulos

Com o arcabouço configurado e pronto para o uso, o sistema espera o recebimento do quadro de vídeo. Em seguida, o arcabouço percorre a lista de módulos de detecção e rastreamento, passando o quadro para cada um deles e esperando os resultados das detecções. Após isso, o sistema percorre a lista de módulos de

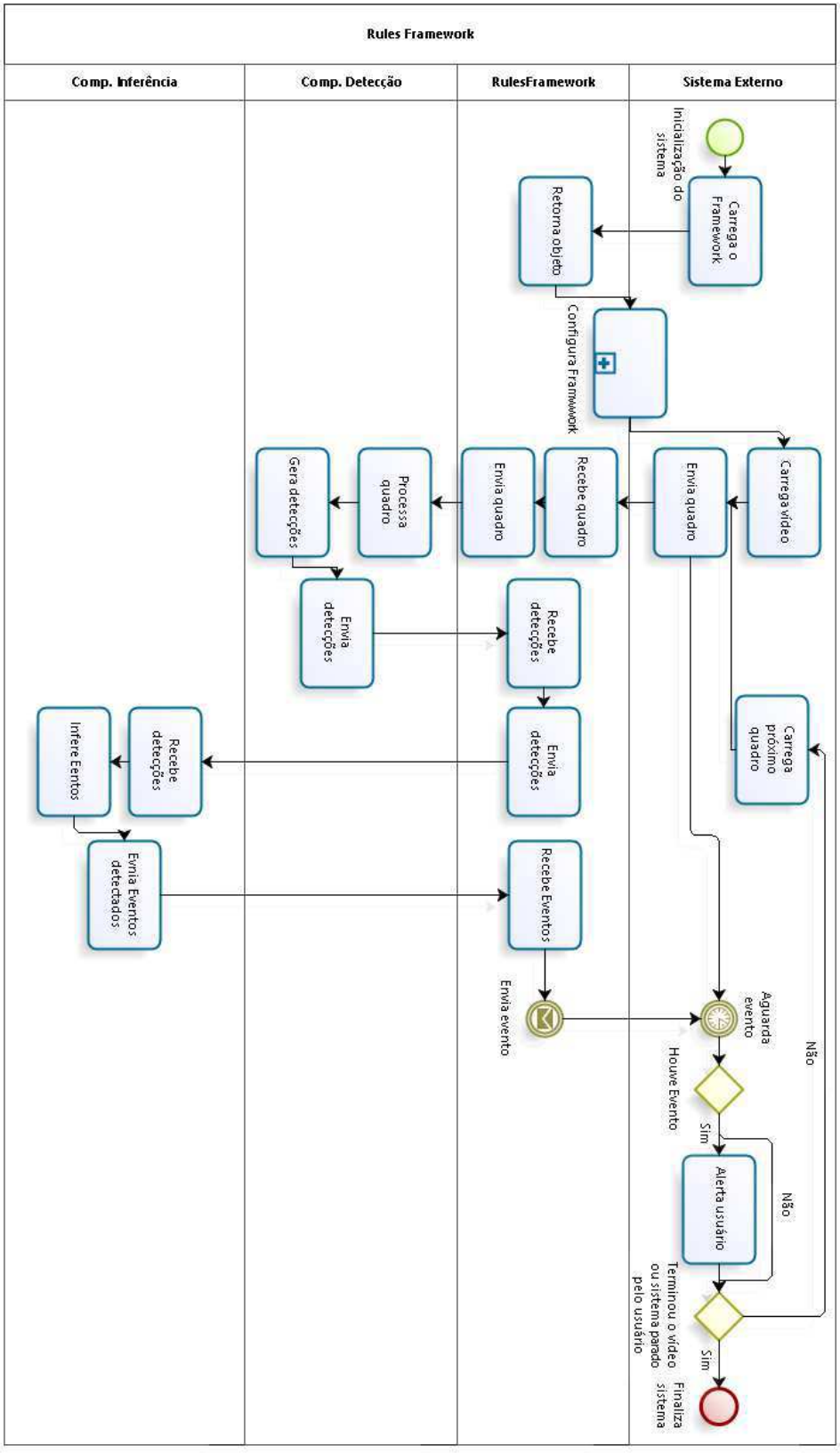


Figura 3.3: Diagrama BPM detalhando o processo de utilização do RulesFramework.

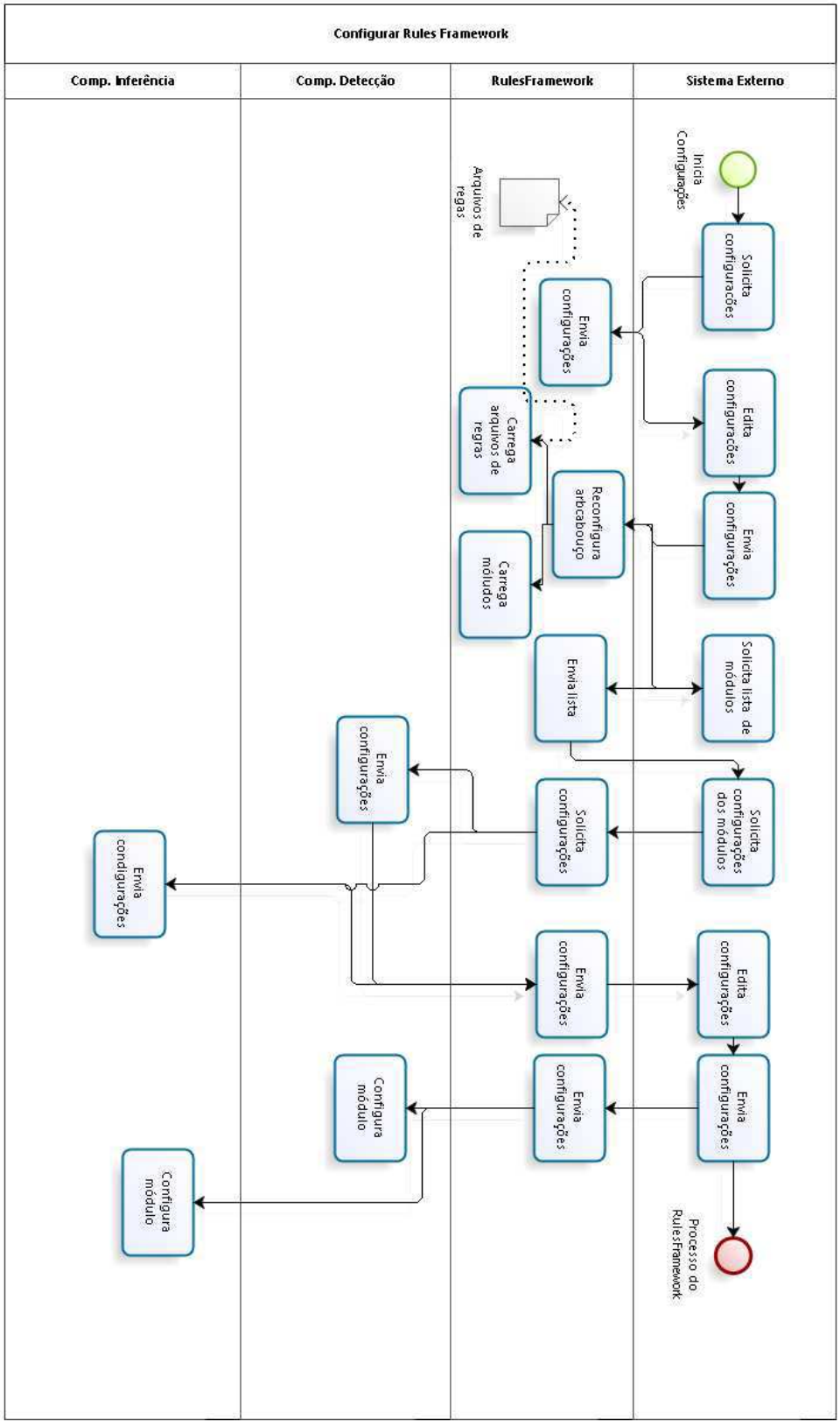


Figura 3.4: Diagrama BPM detalhando o processo de configuração do RulesFramework.

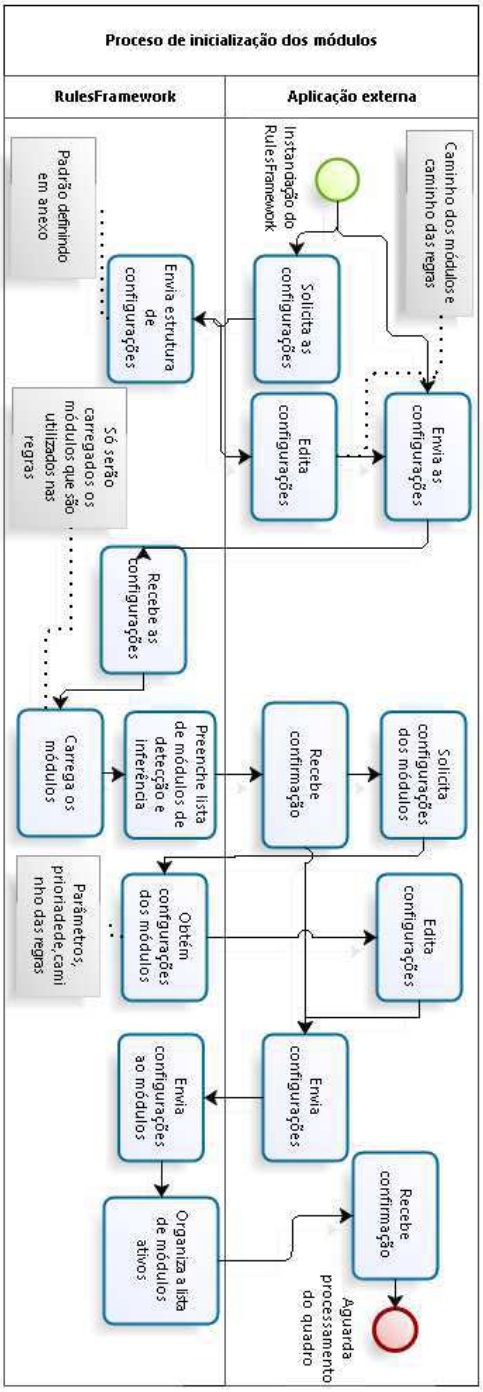


Figura 3.5: Diagrama BPM detalhando o processo de configuração dos módulos.

inferência e passa os resultados das detecções. Os quais infere sobre as detecções utilizando as regras recebidas anteriormente no processo de carregamento da lista de módulos de inferência. Todo esse processo segue o diagrama BPM, apresentado na Figura 3.6.

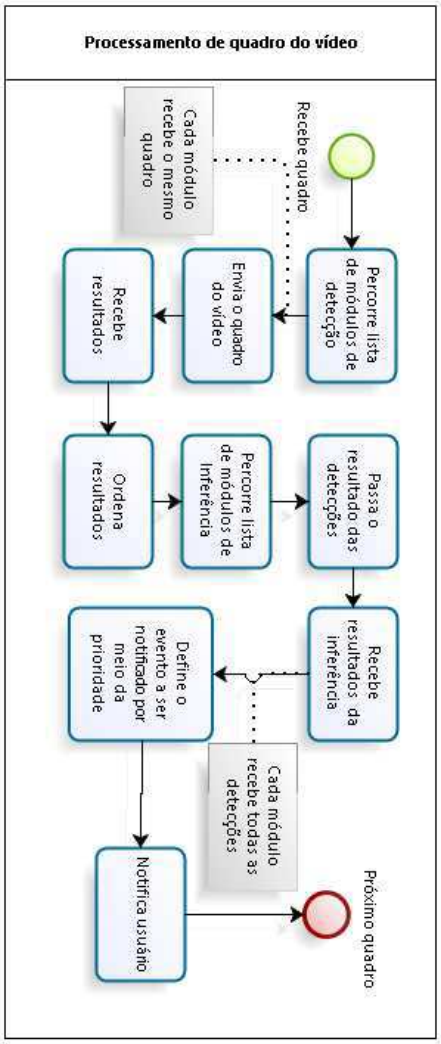


Figura 3.6: Diagrama BPM detalhando o processo de utilização dos módulos.

3.2 Implementação

Esta seção descreve a estrutura geral do modelo de implementação, a divisão do *software* em camadas e os subsistemas no modelo de implementação e todos os componentes significativos do ponto de vista da arquitetura.

O arcabouço é dividido em duas camadas: camada de gerenciamento e a camada de módulos. A camada de gerenciamento é responsável pelo controle dos módulos e a aquisição, repasse e retorno dos dados. A camada de módulos é responsável pelo tratamento dos dados.

3.2.1 Camada de gerenciamento

A Camada de gerenciamento é composta por um único componente, a biblioteca RulesFramework, que possui os componentes básicos definidos no diagrama de classe da Figura 3.7.

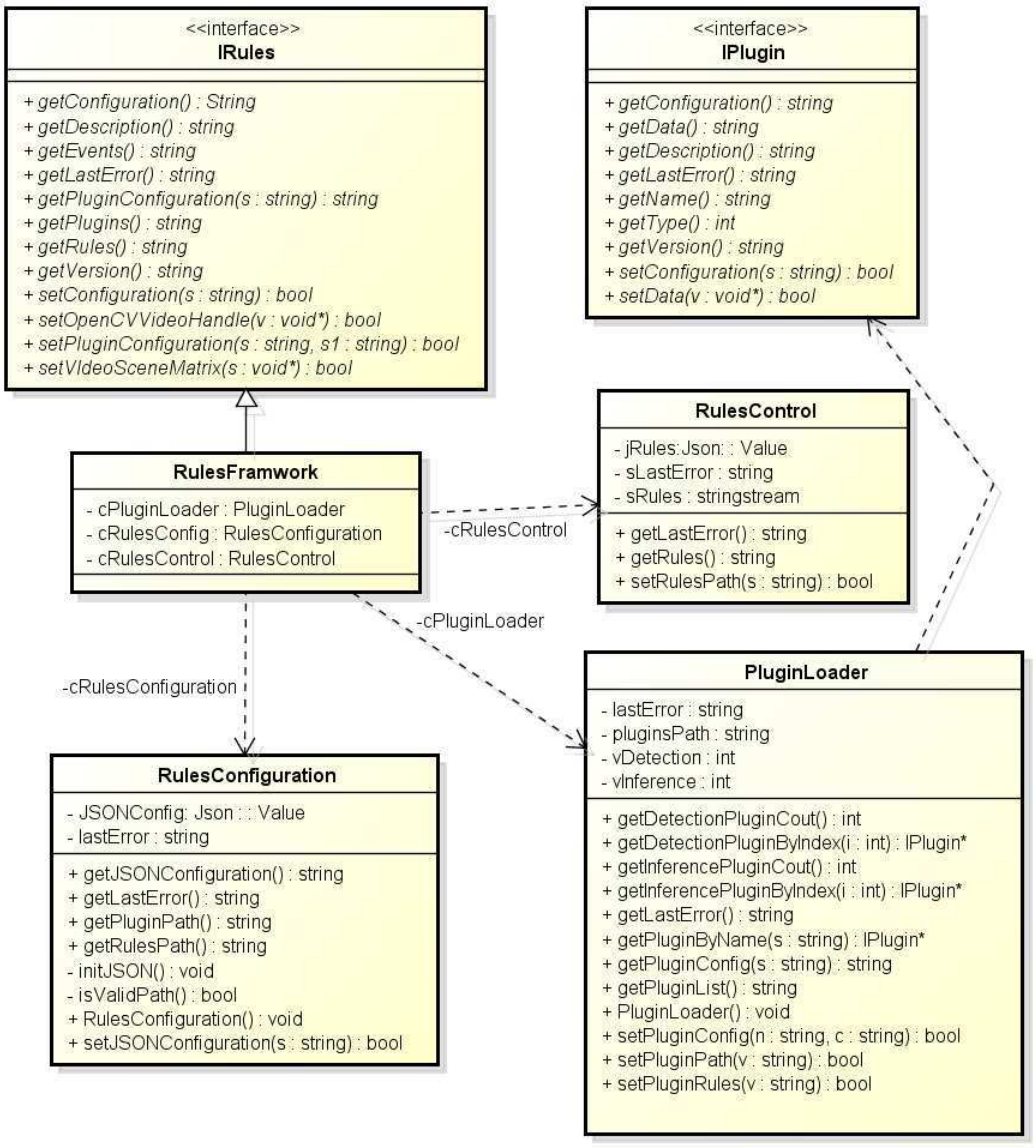


Figura 3.7: Diagrama de Classes da biblioteca RulesFramework.

O RulesFramework foi criado como uma biblioteca dinâmica, desse modo a estrutura utilizada por ele possibilita o carregamento do mesmo em tempo de execução, usando o arquivo de cabeçalho (extensão .h ou .hpp) e o caminho do arquivo de biblioteca de vínculo dinâmico (extensão .dll, padrão Windows). Por meio dessa interface podem ser acessadas as funcionalidades do arcabouço, assim como as propriedades e configurações. Sobre a interface, temos como principais métodos:

- **getConfiguration:** Recebe as configurações do arcabouço seguindo o padrão de mensagem;
- **setConfiguration:** Envia as configurações do arcabouço seguindo o padrão de mensagem;
  Exemplo:

```
{"Configuration" : {
      "PluginsPath" : "e:\\Worksapce\\Plugins\\",
      "RulesPath" : "e:\\Worksapce\\RulesRails\\",
      "OutPut" :"e:\\Worksapce\\RulesRails\\Rails1_Events.txt",
      "Simulator" : false
}}
```

- **getPlugins:** Recebe a lista de módulos dividida entre detecção e inferência;
  Exemplo:

```
{
   "Detection" : [ "CarDetect" ],
   "inference" : [ "HMMInference", "RulesInference" ]
}
```

- **getPluginConfiguration:** Recebe as configurações de um determinado módulo;
- **setPluginConfiguration:** Envia as configurações de um determinado módulo;
- **setVideoSceneMatrix:** Envia a matriz do OpenCV que será passada para os detectores;
- **getEvent:** Recebe as detecções de eventos e os objetos detectados;
  Exemplo:

```
{"Events" :
        [{"Event" : {
                "Name" : "Car overlap crosswlak",
                "Object" : "",
                "Type" : "Car",
                "Value" : false
        }}],
        "objects" :
        [{"Object" : {
                "Name" : "Car0",
                "Points" :
                [{"Point" : {   "x" : 302, "y" : 91}},
                 {"Point" : {"x" : 369, "y" : 91}},
                 {"Point" : {"x" : 369, "y" : 225}},
                 {"Point" : {"x" : 302, "y" : 225}}],
                "Type" : "Car"
}}]}
```

O restante do arcabouço é gerenciado pela classe RulesFramework, que implementa a interface IRules e é auxiliada pela **RulesConfiguration**. Das classes

apresentadas temos que: A **RulesConfiguration** recebe, trata e valida as configurações; a **PluginsLoader** carrega e gerencia os módulos; a **RulesControl** controla as regras a serem utilizadas e demais classes são auxiliares.

3.2.2 Camada de módulos

Para que o gerenciador do arcabouço possa carregá-los e utilizá-los, os módulos devem implementar a interface IPlugin. Este carregamento é realizado pela classe **PluginsLoader** e inicia quando o arcabouço recebe a localização da pasta de módulos. Após isso, a classe realiza uma leitura de todas as bibliotecas de vínculo dinâmico (extensão .dll, padrão Windows), testando qual delas implementam a interface IPlugin e as carrega. No momento do carregamento, é testado se o módulo encontrado é utilizado por alguma das regras previamente carregadas, se sim, a biblioteca é inserida na lista de módulos de detecção ou de inferência.

Entre as funções implementadas da interface temos como principais:

- **setData:** Envia os dados a serem tratados no módulo, as detecções no caso de inferência e o quadro no caso de detecção;
- **getData:** Recebe as detecções no caso do módulo de detecção e eventos no caso do módulo de inferência;
- **getType:** Recebe o tipo do módulo, se de inferência ou de detecção;
- **getName:** Recebe o nome do módulo;
- **getConfiguration:** Recebe as configurações do módulo seguindo o padrão estabelecido pelo arcabouço;
- **setConfiguration:** Envia as configurações do módulo seguindo o padrão estabelecido pelo arcabouço.

Camada de detecção

A camada de detecção abrange todos os módulos que realizam detecções sobre os quadros dos vídeos, e retorna mensagens contendo os resultados das detecções realizadas. Os módulos dessa camada têm como entrada um quadro do vídeo a ser analisado e como saída a mensagem no formato estabelecido pelo sistema. Mais detalhes sobre os módulos criados para esta dissertação são encontrados no Apêndice E.

Camada de inferência

O RulesInference é o responsável por receber as regras dos eventos e os objetos detectados, e com essas informações inferir se houve ou não o evento descrito nas regras. Tais eventos são escritos seguindo o padrão JSON e as especificações contidas na Seção 3.2.3. As regras são descritas formando expressões da

lógica de primeira ordem, usando conectivos AND para concatenar os termos das expressões. O relacionamento temporal dos termos é realizado por meio da Álgebra de Allen (ALLEN, 1983). A Figura 3.8 apresenta o diagrama de classes do módulo de inferência. O exemplo a seguir ilustra a estrutura do arquivo de entrada de objetos, que apresenta o objeto e os pontos que formam o polígono contendo a área do objeto.

```
{"objects" : [{
"Object" : {
"Name" : "Road1",
"Points" : [
{"Point" : {"x" : 175,"y" : 260}},
{"Point" : {"x" : 630,"y" : 290}},
{"Point" : {"x" : 505,"y" : 458}},
{"Point" : {"x" : 0,"y" : 458}},
{"Point" : {"x" : 0,"y" : 295}}
],
"Type" : "Road"
}
},{"Object" : {
"Name" : "Crosswalk1",
"Points" : [
{"Point" : {"x" : 0,"y" : 235}},
{"Point" : {"x" : 675,"y" : 270}},
{"Point" : {"x" : 675,"y" : 290}},
{"Point" : {"x" : 0,"y" : 245}}
],
"Type" : "Crosswalk"}}]}
```

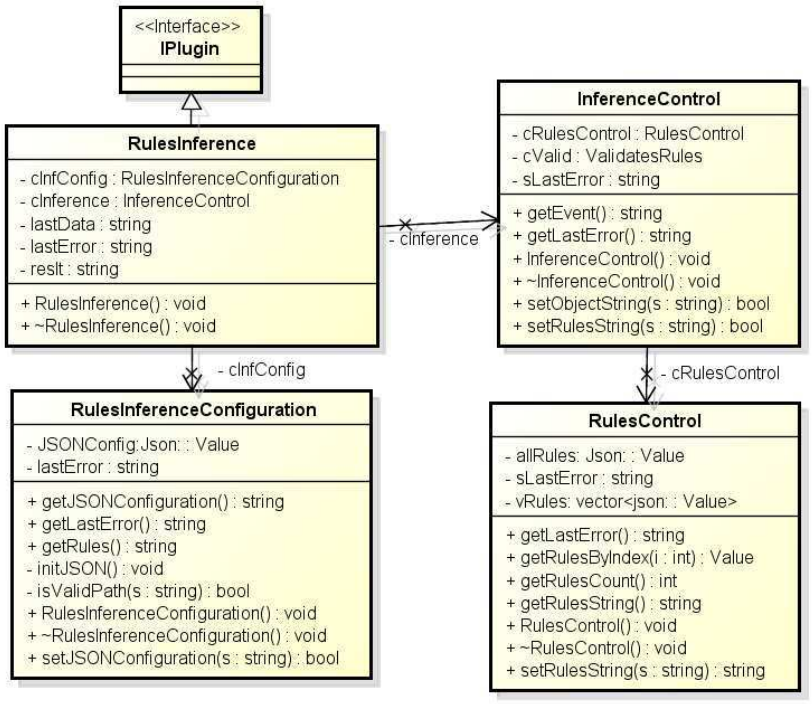


Figura 3.8: Diagrama de Classes do módulo de inferência.

A classe RulesInference é a responsável pela comunicação entre o módulo e o gerenciamento. A classe **RulesInferenceConfiguration** recebe as configurações assim como as regras (na forma de configuração) e as repassa para a classe

RulesControl que organiza as regras por objetos e as guarda em uma lista, com as regras e as detecções. A classe **InferenceControl** juntamente com classes auxiliares, realizam a validação das regras e envia para o arcabouço se o evento foi encontrado ou não.

3.2.3 Modelos das regras

As regras no RulesFramework também seguem o padrão JSON em relação à estruturação, porém dois outros conceitos foram utilizados: Lógica de Primeira Ordem e álgebra de Allen. A seguir, apresentaremos como cada um desses conceitos é utilizado e como ocorre a representação das regras.

Modelando em lógica de primeira ordem

Todas as regras a serem detectadas pelo sistema são proposições, as quais são formadas por vários termos, sendo eles funções ou constantes. Para o presente trabalho foram criadas duas funções: “COLOR” e “OVERLAP” e várias constantes, que pertencem aos tipos: “Estrada”, “Faixa”, “Carro” e “Semaforo” (Semáforo como objeto por conveniência é tratado sem acento) cada constante é um objeto de cada um desses tipos detectados nos quadros, que são nomeados de acordo com o tipo e um número sequencial, como por exemplo: Estrada1, Carro4 e Faixa6. Cada um desses tipos e constantes são enviados ao sistema pelos detectores, permitindo extensibilidade caso novos detectores sejam adicionados. Assim temos como provedores de constantes os módulos: CarDetect (Carros), TrafficLight (Semaforo), FixedObject (Faixa, Estrada e Vaga). Por outro lado as funções são dadas pelos módulos de inferência. Dessa forma, o RulesInference fornece as funções COLOR, que identifica a cor da luz acesa dado um objeto do tipo semáforo e OVERLAP, que determina se dado dois objetos de qualquer tipo, o primeiro sobrepõe a área do segundo.

Dentre as funções criadas para esta dissertação, temos que a função COLOR realiza a leitura dos dados oriundos do detector de luz do semáforo e retorna a cor encontrada. A função OVERLAP recebe como entrada dados da detecção de dois objetos e por meio do posicionamento e do tamanho, compara se houve intercessão entre a área dos dois objetos.

O evento ilustrado na Figura 3.9 representa: “O carro ultrapassa o sinal vermelho”, porém se o evento for subdividido, tem-se a seguinte sequência de subeventos:

- Carro D sobrepôs a estrada A
- Carro D sobrepôs a Faixa B
- O semáforo C esta com a luz vermelha

Após a divisão do evento em eventos primitivos tem-se, em linguagem natural:

Figura 3.9: Carro ultrapassando sinal vermelho.

> "Se o carro D sobrepôs a estrada A e sobrepôs a Faixa B e o semáforo C está com a luz vermelha acesa, então o carro ultrapassou o sinal vermelho"

A expressão anterior em lógica de primeira ordem torna-se:
$Sobrepos(CarroD, EstradaA) \quad \wedge \quad Sobrepos(CarroD, FaixaB) \quad \wedge \quad Cor(SemaforoC) = VERMELHO \Rightarrow UltapassaSinal$

Porém, mesmo representando os acontecimentos do evento a LPO não permite representar aspectos temporais no relacionamento entre os termos.

Álgebra de Allen

Seguindo o modelo lógico apresentado na seção anterior, tem-se que o verbo que define a função está no passado para que a proposição pudesse resultar na detecção dos eventos. Para abordar esse problema da contextualização temporal, os intervalos de Allen sequenciam temporalmente a expressão lógica, conforme apresentado a seguir:

$Durante\,[Cor(SemaforoC) = VERMELHO,\ (Antes\,[Sobrepos(CarroD, EstradaA), Sobrepos(CarroD, FaixaB)]\,] \quad \Rightarrow UltapassaSinal$

Esta abordagem implica duas validações, uma lógica e outra temporal para poder inferir a detecção do evento em questão.

Estrutura do arquivo

A estrutura do arquivo de regras segue o padrão JSON e abrange tanto a lógica de primeira ordem como os intervalos de Allen, O arquivo com a regra descrita nas seções anteriores é apresentado a seguir:

```
1  { "Rules" : {
2  "Name" : "Carro ultrapassa",
3  "Object" : "Car",
4  "Plugins": {"Detection": [ "CarDetect", "FixedObjects", "TrafficLight"],
5              "inference": ["RulesInference"]},
6  "Operators" : {"OP1" : "AND", "OP2" : "AND"},
7  "Sequences" : {
8     "S1" : {"Function" : "COLOR",
9                  "Param" : {
10                      "V1" : {"Name" : "TfA", "Type" : "TrafficLight"},
11                      "V2" : {"Type" : "CONST", "Value" : "RED"} },
12            "Temporal" : {}
13    },"S2" : {"Function" : "OVERLAP",
14                 "Param" : {
15                    "V1" : { "Type" : "THIS"},
16                "V2" : {"Name" : "Road1", "Type" : "Road"},
17                "V3" : {"Type" : "CONST", "Value" : "20"}},
18            "Temporal" : {}
19    },"S3" : {"Function" : "OVERLAP",
20                 "Param" : {
21              "V1" : {"Type" : "THIS"},
22                   "V2" : {
23                           "Name" : "Crosswalk1", "Type" : "Crosswalk"},
24              "V3" : {
25                           "Type" : "CONST", "Value" : "20" } },
26            "Temporal" : {
27                   "AFTER" : "S2",
28                   "DURING" : "S1"}}}}}
```

No arquivo de regras tem-se nas linhas de 2 a 12 o escopo de uma regra “Rules”, o nome da regra, o tipo objeto ao qual a regra pertence (a regra é validada para cada objeto do tipo) e os módulos que serão utilizados pela regra (só serão carregados os módulos que forem utilizados na regra, os demais não serão carregados). Por sua vez, nas linhas de 15 a 18 são descritos os operadores que relacionam os termos, de 19 a 76 são descritos os termos, cada um recebe um identificador “S” mais o número de sequência. Cada sequência é formada pela função, os parâmetros que a função recebe e o relacionamento temporal com os outros termos. Os parâmetros passados podem ser o próprio objeto testado, todos os objetos de um tipo, um objeto específico ou uma constante.

3.3 Considerações finais

Ao final deste capítulo destacamos que os detectores foram criados apenas para testar o arcabouço. Porém, os mesmos não fazem parte do foco desta pesquisa. O que acarretou que os detectores são funcionais, porém não melhores que os encontrados no estado-da-arte. O arcabouço é constituído de bibliotecas dinâmicas e podem ser carregados em tempo de execução. Assim pode ser adicionado, removido ou alterado qualquer componente sem a necessidade de compilar todo o resto do projeto. Essa abordagem também possibilita a criação de novos componentes por qualquer desenvolvedor, sem a necessidade de alterar ou conter o código fonte dos outros componentes. Os modelos das regras foram construídos

de forma a dar suporte a uma futura interface visual para a criação e edição das regras.

Capítulo 4

Avaliação experimental

Para os cenários propostos, não foi encontrada base de dados pública adequada, o que acarretou a criação e rotulamento de uma base pública local. Tal base é considerada pública devido aos vídeos terem sido obtidos do serviço Youtube[1], o qual disponibiliza vídeos a qualquer usuário da Internet e do *site* Camerite[2], que disponibiliza visualização de câmeras públicas *online*. A Tabela 4.1 apresenta uma descrição detalhada da base de dados para eventos de ultrapassagem do sinal vermelho. Para o evento de carro estacionando na vaga, todos os vídeos utilizados nos testes foram obtidos de uma câmera do Camerite, que pode ser visualizada no endereço eletrônico <https://goo.gl/77HIih>. Estes vídeos foram obtidos em horário comercial no período da manhã ou da tarde, e editados para contar apenas o evento a ser detectado. Todos os vídeos e arquivos de configurações utilizados nesta dissertação estão disponibilizados neste endereço eletrônico <https://goo.gl/Ycu5hk>. Alguns vídeos tiveram sua resolução alterada no momento da edição, o que faz com que as marcações utilizadas nesse trabalho não estejam exatamente iguais às marcações realizadas nos vídeos originais, porém, todas são proporcionais aos originais e todos os vídeos editados estão contidos no endereço citado anteriormente. A base contém 61 vídeos, sendo 31 para evento de ultrapassagem do sinal e 30 para estacionamento, com tamanhos que variam entre 6 e 30 segundos.

Para os testes foram criados dois detectores, um para as cores dos semáforos e outro para a detecção de objetos, ora usado para detectar os carros. Sendo assim, os resultados foram organizados em três seções, uma para o detector de cor, outra para o detector de objetos em movimento e a última para a detecção de eventos.

4.1 Detector de cor do semáforo

Como exposto no capítulo anterior, é utilizada uma medida da intensidade da luz em determinada região do semáforo para reconhecer qual o estado do sinal. Para

[1] <https://www.youtube.com/>
[2] <https://www.camerite.com/>

Tabela 4.1: Base de vídeos de carros ultrapassando sinal vermelho.

Link	Tempo	Video
<http://y2u.be/NCmlbDaj-uE>	0:11 - 0:19	Hazelwood Traffic Cameras
<http://y2u.be/NCmlbDaj-uE>	0:32 - 0:44	Hazelwood Traffic Cameras
<http://y2u.be/NCmlbDaj-uE>	0:46 - 0:56	Hazelwood Traffic Cameras
<http://y2u.be/NCmlbDaj-uE>	0:56 - 1:04	Hazelwood Traffic Cameras
<http://y2u.be/NCmlbDaj-uE>	1:30 - 1:41	Hazelwood Traffic Cameras
<http://y2u.be/NCmlbDaj-uE>	1:42 - 1:54	Hazelwood Traffic Cameras
<http://y2u.be/NCmlbDaj-uE>	3:45 - 3:55	Hazelwood Traffic Cameras
<http://y2u.be/Hf_N3abhQf0>	0:09 - 0:14	Dangers of Right On Red
<http://y2u.be/vimo-OCQODU>	0:19 - 0:24	Nassau Stops On Red 2011 PSA
<http://y2u.be/zpGGVtBUK0I>	Todo	Red-Light Running Incident in Miami, FL
<http://y2u.be/njA9cBZQQ3Q>	Todo	OrangePark, FL Red-Light Safety Cameras Capture Near Miss of Bicyclist
<http://y2u.be/PeGSEuGYM9g>	Todo	Sarasota, FL Red-Light Running Collision
<http://y2u.be/M7JrtCrKNuE>	Todo	Montgomery Near Miss Caught by Red-Light Safety Cameras
<http://y2u.be/VTW8WYR_nFI>	Todo	Gloucester Township, NJ Near miss
<http://y2u.be/9SUh9cXk2mk>	Todo	Laurel, MD close call with bicyclist
<http://y2u.be/OmEEhXOqoiE>	Todo	Florissant catches red-light runner nearly miss pedestrian
<http://y2u.be/UcBpcpsBoV0>	Todo	Excelsior Springs, Missouri Near Miss
<http://y2u.be/KCdecXnfzcY>	Todo	Florissant, Missouri red light running near miss 2
<http://y2u.be/5O3oZro73_Q>	Todo	East Cleveland Cameras Capture Close Call
<http://y2u.be/sh3WbdCthBs>	0:30 - 0:52	Granite City, IL Red-Light Running Compilation
<https://www.youtube.com/watch?v=jbTbrxvo8dM>	Todo	Jacksonville, FL - Red Light Runner Of The Week - 01/29/2015 - Normandy and Lane
<https://www.youtube.com/watch?v=a9MQWvtWWCE>	00:52 - 00:57	2015's Worst Red Light Runners - Funny Videos
<https://www.youtube.com/watch?v=YEUcBnTT0VQ>	00:38 - 00:44	Red-light Running in New Jersey 2011
<https://www.youtube.com/watch?v=YEUcBnTT0VQ>	00:20 - 00:25	Red-light Running in New Jersey 2011
<https://www.youtube.com/watch?v=63kHSU5cFCI>	00:12 - 00:19	Red Light Runners - Round 5
<https://www.youtube.com/watch?v=63kHSU5cFCI>	00:21 - 00:27	Red Light Runners - Round 5
<https://www.youtube.com/watch?v=63kHSU5cFCI>	01:20 - 01:28	Red Light Runners - Round 5
<https://www.youtube.com/watch?v=63kHSU5cFCI>	01:59 - 02:08	Red Light Runners - Round 5
<https://www.youtube.com/watch?v=BsYBbJs4hCQ>	00:56 - 01:04	Red Light Runners - Round 6
<https://www.youtube.com/watch?v=BsYBbJs4hCQ>	01:07 - 01:16	Red Light Runners - Round 6

validar essa abordagem foram utilizados quadros selecionados arbitrariamente de cada um dos vídeos da base. De cada vídeo da base foi obtido um quadro de cada cor que o sinal assumia no vídeo, a Tabela 4.2 mostra a relação de quadros utilizados para o teste, rótulos/resultados e posicionamento do semáforo. Dentre as colunas temos o nome de vídeo, o quadro do vídeo que foi utilizado, a cor existente no sinal "/" cor detectada pelo detector e o posicionamento da marcação do semáforo, contendo posicionamento nos eixos X e Y, assim como a altura e largura da marcação.

Tabela 4.2: Vídeos, quadros, detecções e posicionamentos dos semáforos.

Vídeo	Quadro	Cor/Detecção	Eixo X	Eixo Y	Altura	Largura
Hazelwood01	37	Vermelho/Vermelho	551	43	20	9
Hazelwood02	15	Amarelo/Amarelo	542	34	22	9
Hazelwood02	148	Vermelho/Vermelho	542	34	22	9
Hazelwood03	76	Vermelho/Vermelho	411	96	22	8
Hazelwood04	9	Amarelo/Amarelo	480	39	22	5
Hazelwood04	122	Vermelho/Vermelho	480	39	22	5
Hazelwood06	13	Verde/Verde	162	53	29	5
Hazelwood06	75	Amarelo/Amarelo	162	53	29	5
Hazelwood06	192	Vermelho/Vermelho	162	53	29	5
Hazelwood07	116	Vermelho/Vermelho	146	71	25	8
Hazelwood09	133	Vermelho/Vermelho	380	31	23	6
DangersOnRed	73	Vermelho/Vermelho	534	83	21	6
Nassau01	10	Amarelo/Amarelo	387	102	27	4
Nassau01	46	Vermelho/Vermelho	387	102	27	4
Miami01	3	Amarelo/Amarelo	337	55	16	5
Miami01	35	Vermelho/Vermelho	337	55	16	5
OrangePark	3	Amarelo/Amarelo	464	15	14	4
OrangePark	194	Vermelho/Vermelho	464	15	14	4
Sarasota	159	Vermelho/Vermelho	396	27	19	9
Montgomery	162	Vermelho/Vermelho	43	19	15	5
Gloucester	5	Amarelo/Amarelo	211	38	19	7
Gloucester	213	Vermelho/Vermelho	211	38	19	7
Laurel	189	Vermelho/Vermelho	273	15	12	4
Florissant	192	Vermelho/Vermelho	188	16	12	4
Excelsior	244	Vermelho/Vermelho	266	23	19	8
Florissant02	5	Amarelo/Amarelo	178	21	15	4
Florissant02	166	Vermelho/Vermelho	178	21	15	4
East	16	Verde/Verde	283	28	16	9
East	71	Amarelo/Amarelo	283	28	16	9
East	172	Vermelho/Vermelho	283	28	16	9
Granite	4	Amarelo/Amarelo	520	145	39	13
Granite	32	Vermelho/Vermelho	520	145	39	13
Granite	163	Amarelo/Amarelo	520	145	39	13
Granite	191	Vermelho/Vermelho	520	145	39	13
Granite	349	Amarelo/Amarelo	520	145	39	13
Granite	389	Vermelho/Vermelho	520	145	39	13
Granite	621	Vermelho/Vermelho	520	145	39	13

Os resultados que são verificados nesses quadros, se repetem por todos os vídeos. Assim, quando utilizadas as demarcações manuais de tamanho e posicionamento contidos na Tabela 4.2, para todos os quadros dos vídeos, têm-se uma taxa de acerto de 100% em relação à detecção da cor do semáforo.

4.2 Detector de objetos

Como exposto no Capítulo 3, o detector utiliza subtração de plano de fundo por meio de misturas de gaussianas para detectar o movimento que houve entre os

quadros do vídeo. Para avaliar o detector, foram utilizadas marcações realizadas nos vídeos da base, as quais abrangem os carros que estavam presentes na área de interesse previamente mapeada. As marcações foram feitas nos carros observáveis, estando eles em movimento ou não (desde que tenham se movimentado pelo menos uma vez no vídeo). Tal fato contribui para uma diminuição na taxa de detecção, devido ao fato de que quando um objeto parar de se movimentar, o mesmo não será mais detectado. Os testes foram realizados nas primeiras 20 amostras de vídeos de trânsitos coletadas, mesmo não sendo foco do trabalho criar detectores, os testes foram realizados para ilustrar a viabilidade do projeto. A Tabela 4.3 mostra os resultados das detecções, indicando os falsos positivos (FP), falsos negativos (FN), verdadeiros positivos (VP) e verdadeiros negativos (VN).

Tabela 4.3: Detector de objetos detectando carros.

Video	FP	FN	VP	VN	Acurácia	Precisão	Revocação	Valor-F
DangersOnRed01	12	28	124	38	80,20	91,18	81,58	85,34
East	169	285	207	1	31,42	55,05	42,70	40,01
Excelsior	37	141	127	136	59,64	77,44	47,39	67,38
Florissant	3	39	113	204	88,30	97,41	74,34	92,62
Florissant02	38	118	34	147	53,71	47,22	22,37	50,26
Gloucester	167	358	353	0	40,21	67,88	49,65	51,50
Granite	168	126	352	211	65,69	67,69	73,64	66,68
Hazelwood01	5	21	124	89	89,12	96,12	85,52	92,49
Hazelwood02	143	257	222	60	41,35	60,82	46,35	49,23
Hazelwood03	5	142	218	101	68,45	97,76	60,56	80,52
Hazelwood04	32	49	110	7	59,09	77,46	69,18	67,04
Hazelwood06	311	802	401	0	26,49	56,32	33,33	36,03
Hazelwood07	28	145	119	78	53,24	80,95	45,07	64,24
Hazelwood09	66	278	177	96	44,25	72,84	38,90	55,05
Laurel	13	39	65	254	85,98	83,33	62,50	84,64
Miami01	27	172	214	110	61,95	88,80	55,44	72,98
Montgomery	8	41	84	274	87,96	91,30	67,20	89,60
Nassau01	23	126	240	0	61,70	91,25	65,57	73,62
OrangePark	174	374	320	0	36,87	64,78	46,11	46,99
Sarasota	93	368	275	14	35,53	74,73	42,77	50,85

As detecções de carros possuem um Valor-F médio de 65,85 e tem um desvio padrão de 16,95. Estes valores são ocasionados devido a diversidade da base, que contém vídeos com uma variedade de cenários e comportamentos. Os melhores resultados (acurácia acima de 80%) foram atingidos nos vídeos mais simples, como Hazelwood01, Florissant, Montgomery, Laurel, DangersOnRed01, os quais só possuem um ou dois carros se movimentando na área de interesse ao mesmo tempo e com uma certa distância entre eles. Os resultados com acurácia piores que 40% como aqueles associados aos vídeos OrangePark e Sarasota, se dão pela presença de carros que param durante o vídeo e deixam de ser detectados. Os vídeos East e Hazelwood06 contam com veículos parados e também com objetos próximos o suficiente para serem tratados incorretamente pelo detector como sendo o mesmo objeto.

4.3 Detecção de eventos

Todos os vídeos da base possuem o evento acontecendo em alguns quadros do vídeo, assim como alguns quadros antes e depois do evento em questão. Para os eventos de carro ultrapassando o sinal vermelho, os quadros considerados como contendo os eventos foram os que o intervalo de quadros compreendeu desde o instante em que o carro ultrapassou a faixa de pedestres até o último quadro do vídeo contendo o rótulo do carro que disparou o evento. Para os eventos de carros estacionando na vaga, os quadros considerados como contendo o evento foram os quadros em que o carro rotulado manualmente ocupa mais de 30% da vaga em questão.

Devido à complexidade e a variedade de cenários utilizados para detecção de carros ultrapassando o sinal vermelho e devido o foco do trabalho não ser a criação de um detector genérico de veículos. O módulo de detecção teve os parâmetros (e.g. histórico e taxa de aprendizagem) ajustados para cada um dos cenários propostos. Para os eventos de estacionamento, foram utilizados os mesmos parâmetros para todos os vídeos, pois todos foram obtidos no mesmo cenário (capturados de uma única câmera). Como diferença em arquivos de configuração nesses vídeos tem-se apenas o remapeamento das vagas devido a alteração no posicionamento da câmera, pois alguns vídeos foram obtidos dias depois e nesses vídeos a câmera teve o ângulo alterado.

Para análise dos resultados foram utilizadas métricas como acurácia, precisão, revocação e Valor-F, como também a taxa normalizada do custo de detecção (*Normalized Detection Cost Rate*[3] - NDCR) utilizada na competição TRECVID[2], que é uma combinação linear entre a probabilidade das falhas de detecção (Falso Negativo) e a taxa de falso alarme (Falso Positivo) com atribuição de pesos. Por ser uma métrica normalizada, a mesma vária no intervalo [0..1], no qual 0 é um desempenho perfeito e 1 é o custo do sistema não realizar qualquer detecção. Todas as métricas utilizam como base para os cálculos os valores de Verdadeiro Positivos (VP, quando o sistema detectar algo que realmente ocorreu), Verdadeiro Negativo (VN, quando o sistema não detectar nada e realmente não houve nada), Falso Positivo (FP, quando o sistema detectar algo e realmente não houve nada) e Falso Negativo (FN, quando o sistema não detecta nada e realmente houve algo). As métricas de acurácia (Equação 4.1), precisão (Equação 4.2), revocação (Equação 4.3), valor-F (Equação 4.4) e NDCR (Equação 4.5) são calculadas de acordo com as seguintes equações:

$$Acuracia = \frac{VP + VN}{VP + VN + FP + FN} \tag{4.1}$$

$$Precisao = \frac{VP}{VP + FP} \tag{4.2}$$

[3] <http://www.itl.nist.gov/iad/mig/tests/trecvid/2011/doc/SED11-EvalPlan-v01.htm>

$$Revocacao = \frac{VP}{VP + FN} \tag{4.3}$$

$$Valor - F = \frac{Precisao \times Revocacao}{Precisao + Revocacao} \tag{4.4}$$

$$NDCR = (\frac{FN}{TotalDeQuadrosPositivos}) + Beta \times (\frac{FP}{TotalDeQuadros}) \tag{4.5}$$

em que $TotalDeQuadrosPositivos$ representa quantidade de quadros em que realmente ocorre o evento, $TotalDeQuadros$ representa a quantidade total de quadros do vídeo e $Beta$ representa uma medida de custo das falhas definida na Equação 4.6.

$$Beta = \frac{CustoFa}{CostoErr \times TaxaPrioridade} \tag{4.6}$$

em que $CustoFa$ é uma constante que define o custo de um falso alarme (FN) (utilizado na dissertação o padrão do TRECVID que é 1), $CustoErr$ é a constante que define o custo para uma detecção errada (FP) (utilizado na dissertação o padrão do TRECVID que é 10) e $TaxaPrioridade$ é uma constante que define a taxa de prioridade dos eventos observados (utilizado na dissertação o padrão do TRECVID que é 20).

Os resultados das detecções dos eventos de carros estacionando na vaga estão apresentados na Tabela 4.4. Na Tabela 4.5 são apresentados os resultados das detecções dos eventos de carros ultrapassando os sinal vermelho.

Os testes com os carros ultrapassando o sinal vermelho foram realizados com o detector e com um simulador que emulava os rótulos como sendo as detecções, a coluna tipo especifica **“Det”** para o detector e **“Sim”** para os dados simulados. Os dados simulados foram utilizados para validar a máquina de inferência e comprovar que a qualidade dos resultados da máquina de inferência depende da qualidade das detecções recebidas. A mesma análise não é apresentada na tabela com os resultados dos eventos de estacionamento por terem obtido o resultado de 100% para todas as métricas, assim, para simplificar a segunda tabela não foram inseridos estes dados.

A etapa inicial da análise dos resultados foi a geração de gráficos Q-Q Plots para acurácia e Valor-F de todos os eventos juntos (Figura 4.1), dos eventos dos vídeos envolvendo ultrapassagem (Figura 4.2) e dos eventos envolvendo estacionamento (Figura 4.3). Entre os resultados apresentados, temos que apenas os gráficos com os valores-F de todos os eventos e dos eventos de ultrapassagem aparentam ser normais. Também foi realizada as mesmas análises para o NDCR dos eventos, os quais apenas o dos eventos de ultrapassagem indicaram aparente normalidade nas distribuições.

Tabela 4.4: Resultados para o evento **estacionar na vaga**.

Vídeo	Detectado	NDCR	Acurácia	Precisão	Revocação	Valor-F
Estacionamento01	SIM	0,00	94,51	90,21	100,00	94,85
Estacionamento02	SIM	0,13	88,24	95,29	87,36	91,15
Estacionamento03	SIM	0,00	63,43	50,69	100,00	67,28
Estacionamento04	SIM	0,10	91,14	90,00	90,00	90,00
Estacionamento05	SIM	0,00	88,67	79,70	100,00	88,71
Estacionamento06	SIM	0,40	77,04	100,00	59,79	74,83
Estacionamento07	SIM	0,52	61,46	70,00	48,13	57,04
Estacionamento08	SIM	0,07	93,96	95,93	92,70	94,29
Estacionamento09	SIM	0,47	34,63	35,51	52,73	42,44
Estacionamento10	SIM	0,00	74,32	68,86	100,00	81,56
Estacionamento11	SIM	0,00	97,34	96,40	100,00	98,17
Estacionamento12	SIM	0,00	83,89	72,36	100,00	83,96
Estacionamento13	SIM	0,35	81,27	90,65	65,10	75,78
Estacionamento14	SIM	0,03	84,76	75,00	96,77	84,51
Estacionamento15	SIM	0,02	94,68	92,82	98,25	95,45
Estacionamento16	SIM	0,00	98,06	97,12	100,00	98,54
Estacionamento17	SIM	0,00	95,68	92,66	100,00	96,19
Estacionamento18	SIM	0,00	87,80	74,65	100,00	85,49
Estacionamento19	SIM	0,00	94,07	91,01	100,00	95,29
Estacionamento20	SIM	0,00	98,52	96,36	100,00	98,15
Estacionamento21	SIM	0,00	97,89	95,60	100,00	97,75
Estacionamento22	SIM	0,00	98,34	96,55	100,00	98,25
Estacionamento23	SIM	0,00	97,42	95,17	100,00	97,53
Estacionamento24	SIM	0,00	94,48	87,65	100,00	93,42
Estacionamento25	SIM	0,00	95,85	92,42	100,00	96,06
Estacionamento26	SIM	0,00	95,97	92,37	100,00	96,03
Estacionamento27	SIM	0,00	94,68	89,33	100,00	94,37
Estacionamento28	SIM	0,00	94,48	89,13	100,00	94,25
Estacionamento29	SIM	0,00	96,68	93,83	100,00	96,82
Estacionamento30	SIM	0,00	95,43	91,27	100,00	95,44

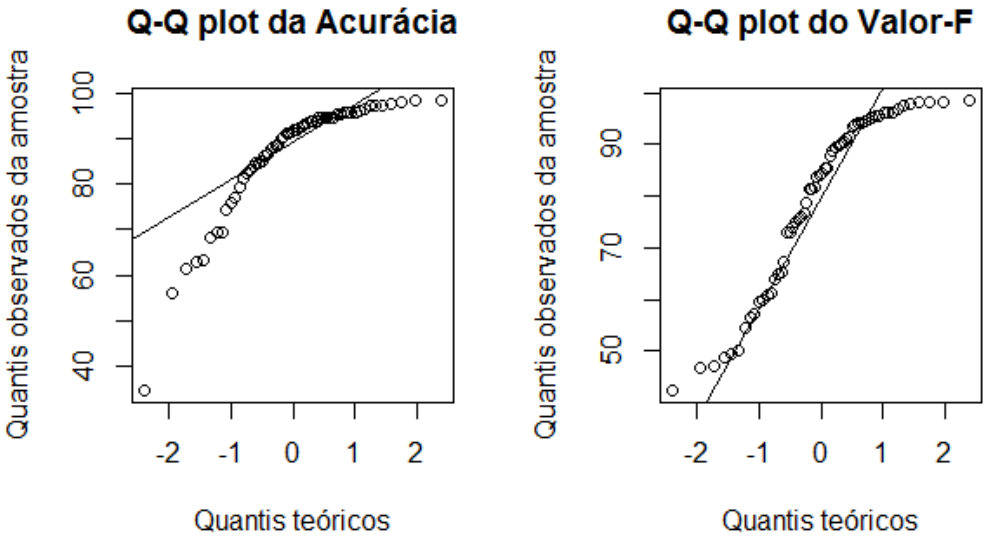


Figura 4.1: Q-Q Plot da acurácia e Valor-F de todos os eventos.

Tabela 4.5: Resultados para o evento de **ultrapassar o sinal vermelho**.

Tipo	Vídeo	Detectado	NDCR	Acurácia	Precisão	Revocação	Valor-F
Det.	DangersOnRed01	SIM	0,19	93,29	100,00	80,77	89,36
Sim.	DangersOnRed01	SIM	0,00	100,00	100,00	100,00	100,00
Det.	East	SIM	0,22	88,61	55,88	77,55	64,96
Sim.	East	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Excelsior	SIM	0,31	95,86	100,00	68,75	81,48
Sim.	Excelsior	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Florissant	SIM	0,29	96,11	81,48	70,97	75,86
Sim.	Florissant	SIM	0,00	100	100	100,00	100
Det.	Florissant02	SIM	0,57	93,71	56,25	42,86	48,65
Sim.	Florissant02 S	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Gloucester	SIM	0,51	85,00	100,00	48,57	65,38
Sim.	Gloucester	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Granite	SIM	0,56	82,30	95,88	44,50	60,78
Sim.	Granite	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Hazelwood01	SIM	0,18	92,92	100,00	82,47	90,40
Sim.	Hazelwood01	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Hazelwood02	SIM	0,23	94,55	92,00	76,67	83,64
Sim.	Hazelwood02	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Hazelwood03	SIM	0,19	92,42	100,00	81,48	89,80
Sim.	Hazelwood03	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Hazelwood04	SIM	0,43	90,00	100,00	57,50	73,02
Sim.	Hazelwood04	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Hazelwood06	SIM	0,67	84,68	100,00	32,93	49,54
Sim.	Hazelwood06	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Hazelwood07	SIM	0,67	62,95	100,00	33,50	50,19
Sim.	Hazelwood07	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Hazelwood09	SIM	0,69	69,26	100,00	30,83	47,13
Sim.	Hazelwood09	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Laurel	SIM	0,58	90,56	100,00	42,37	59,52
Sim.	Laurel	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Miami01	SIM	0,09	95,36	92,11	90,91	91,50
Sim.	Miami01	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Montgomery	SIM	0,70	91,11	100,00	30,43	46,67
Sim.	Montgomery	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Nassau01	SIM	1,00	38,00	0,00	0,00	0,00
Sim.	Nassau01	SIM	0,00	100,00	100,00	100,00	100,00
Det.	OrangePark	SIM	0,10	97,49	98,59	89,74	93,96
Sim.	OrangePark	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Sarasota	SIM	0,57	68,29	100,00	42,77	59,91
Sim.	Sarasota	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Jerssey01	SIM	0,63	55,93	100,00	37,35	54,39
Sim.	Jerssey01	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Jerssey02	SIM	0,43	69,44	100,00	57,36	72,91
Sim.	Jerssey02	SIM	0,00	100,00	100,00	100,00	100,00
Det.	RedLight01	SIM	0,52	83,33	93,94	48,44	63,92
Sim.	RedLight01	SIM	0,00	100,00	100,00	100,00	100,00
Det.	RedLight02	SIM	0,61	79,44	100,00	39,34	56,47
Sim.	RedLight02	SIM	0,00	100,00	100,00	100,00	100,00
Det.	RedLight03	SIM	0,41	86,19	100,00	58,57	73,87
Sim.	RedLight03	SIM	0,00	100,00	100,00	100,00	100,00
Det.	RedLight04	SIM	0,34	93,33	97,06	66,00	78,57
Sim.	RedLight04	SIM	0,00	100,00	100,00	100,00	100,00
Det.	RedLight06	SIM	0,53	75,73	88,46	46,94	61,33
Sim.	RedLight06	SIM	0,00	100,00	100,00	100,00	100,00
Det.	RedLight07	SIM	0,25	91,67	100,00	74,58	85,44
Sim.	RedLight07	SIM	0,00	100,00	100,00	100,00	100,00
Det.	RedLight08	SIM	0,31	91,90	100,00	68,52	81,32
Sim.	RedLight08	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Worst2015	SIM	0,22	86,58	100,00	78,02	87,65
Sim.	Worst2015	SIM	0,00	100,00	100,00	100,00	100,00
Det.	Jacksonville	SIM	0,38	91,79	100,00	62,16	76,67
Sim.	Jacksonville	SIM	0,00	100,00	100,00	100,00	100,00

As observações realizadas nos gráficos não são determinantes para comprovar ou não a normalidade das distribuições dos dados. Para obter mais evidên-

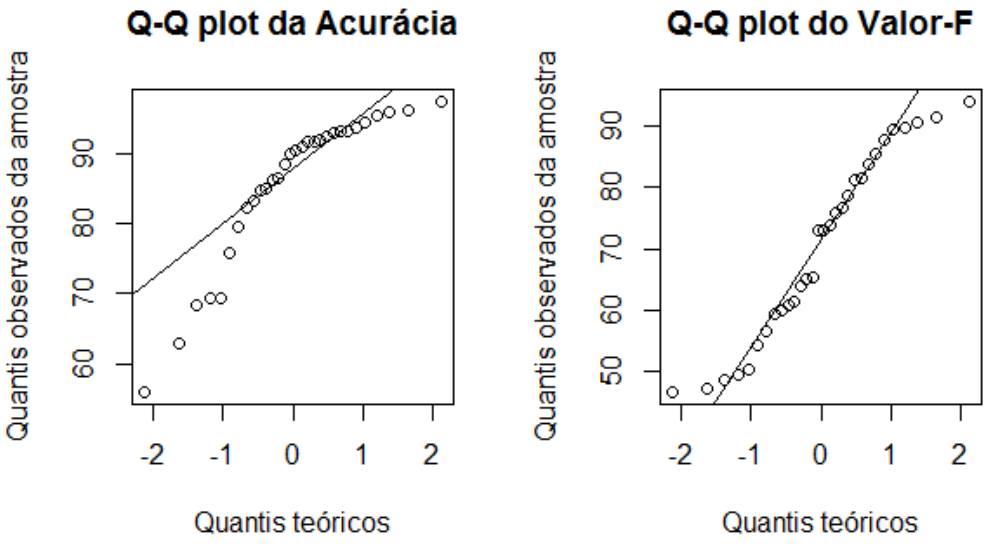


Figura 4.2: Q-Q Plot da acurácia e Valor-F dos eventos de ultrapassagem.

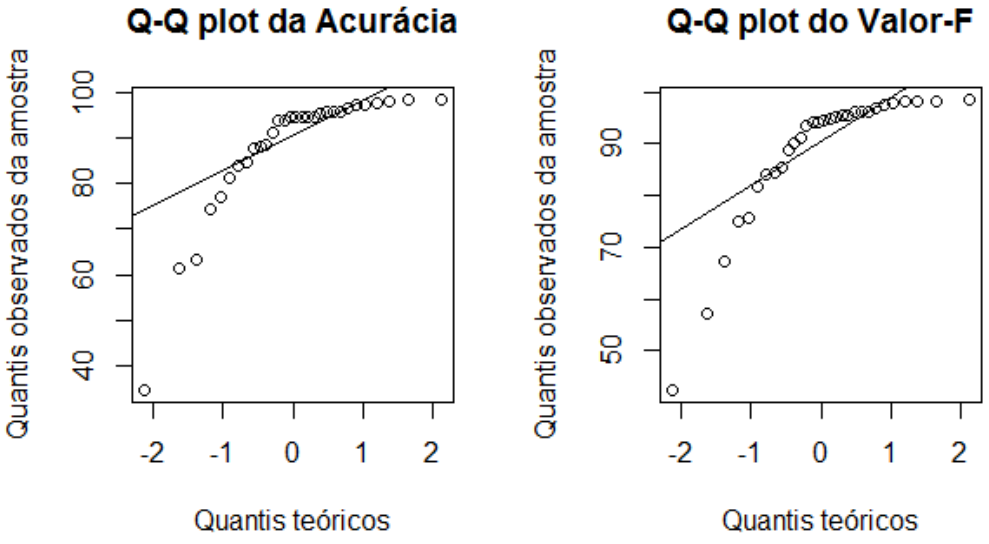


Figura 4.3: Q-Q Plot da acurácia e Valor-F dos eventos de estacionamento.

cias sobre a normalidade da distribuição dos dados, foram realizados três testes de normalidade: Shapiro-Wilk, Kolmogorov-Smirnov e Anderson-Darling. Os resultados dos testes são apresentados na Tabela 4.6. Os testes evidenciam a normalidade das distribuições dos dados das métricas Valor-F e NDCR para os eventos de ultrapassagem do sinal vermelho, pois o valor da estatística W para o teste Shapiro-Wilk de ambas consegue ser maior que o valor de 0,927 contido na tabela de valor crítico para 30 amostras e significância de 5%. As estatísticas do teste Kolmogorov-Smirnov de ambas conseguem ser menor que o valor 0,24 con-

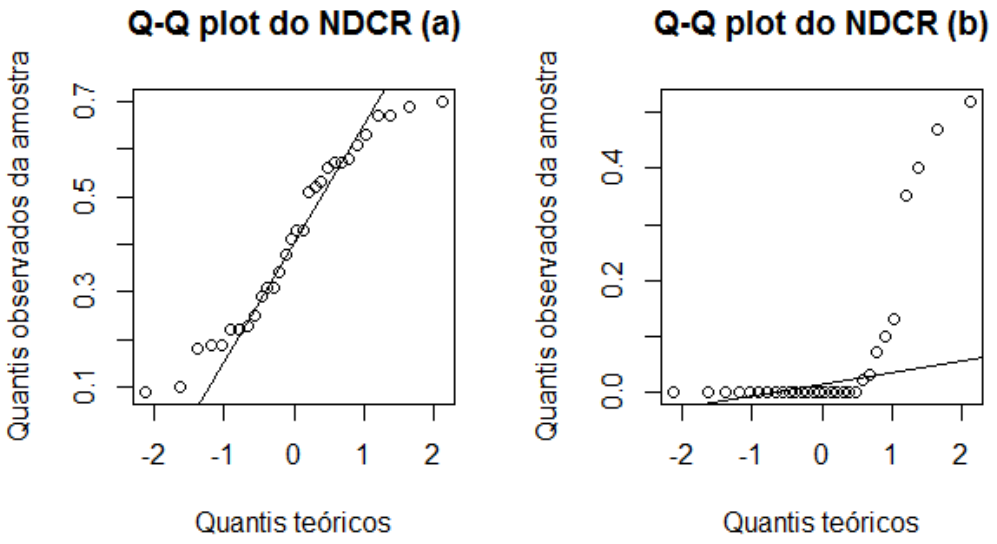


Figura 4.4: Q-Q Plot do NDCR dos eventos de estacionamento e ultrapassagem.

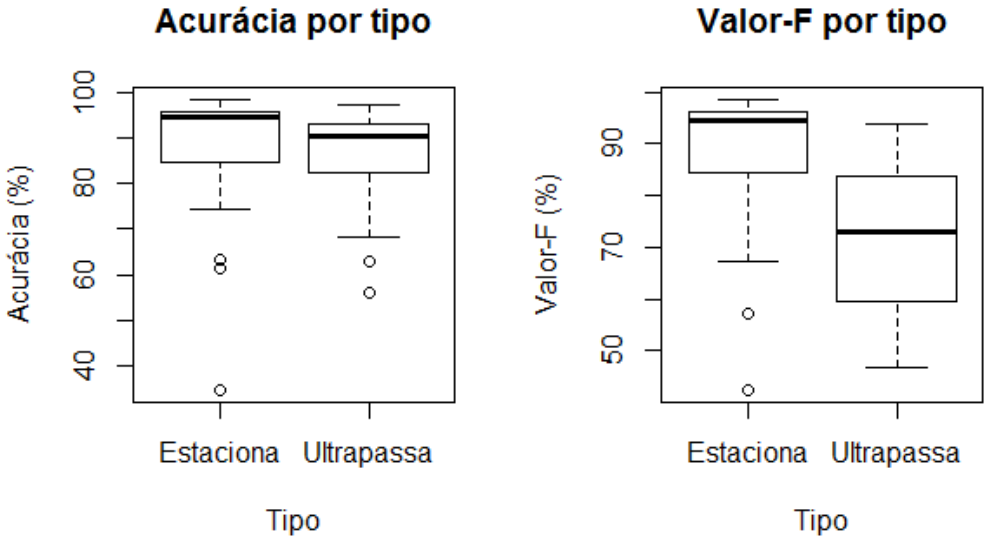


Figura 4.5: Boxplots da acurácia e Valor-F por tipo de eventos.

tido na tabela de valor crítico para 30 amostras e significância de 5%. No teste de Anderson-Darling o valor P de ambas conseguem ser maior que a significância de 5%. Sendo assim, tem-se indícios que apenas essas duas distribuições são normais.

Após determinar quais dados seguem uma distribuição normal, foi utilizado o teste t-student (SEWARD, 2014) para obtenção dos intervalos de confiança das distribuições normais, e para os demais foi utilizado a técnica de BOOTSTRAP (SEWARD, 2014), que consiste na reamostragem dos dados de forma a poder

Tabela 4.6: Resultados dos testes de normalidade.

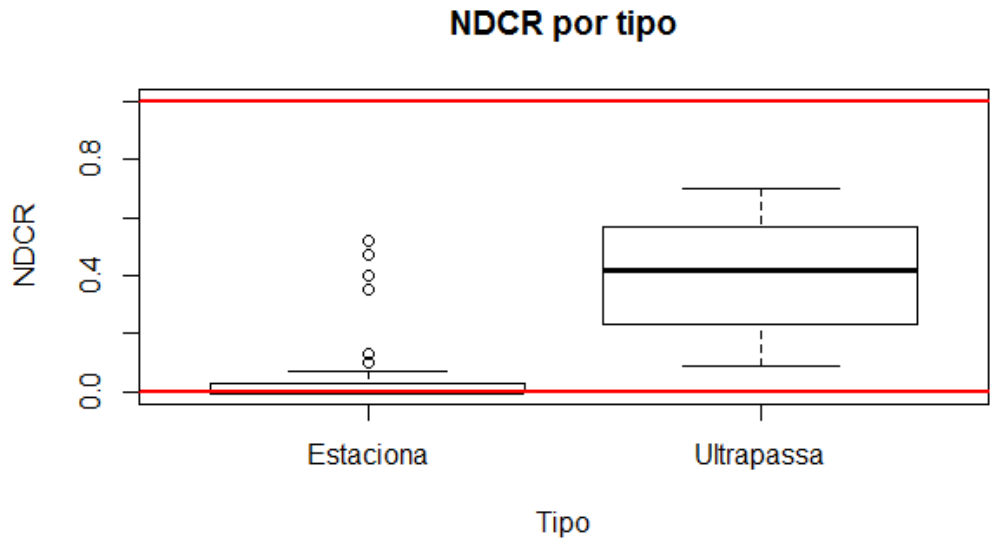
Teste	Evento	Métrica	Estatística	Valor P	Normal
Shapiro-Wilk	Ultrapassar	Acurácia	0,8468	0,00053	NÃO
Kolmogorov-Smirnov	Ultrapassar	Acurácia	0,1893	0,00763	SIM
Anderson-Darling	Ultrapassar	Acurácia	1,7253	0,00016	NÃO
Shapiro-Wilk	Estacionar	Acurácia	0,2600	1,72E-05	NÃO
Kolmogorov-Smirnov	Estacionar	Acurácia	0,6992	1,52E-06	NÃO
Anderson-Darling	Estacionar	Acurácia	3,0565	7,16E-08	NÃO
Shapiro-Wilk	Ultrapassar	Valor-F	0,9376	0,07851	SIM
Kolmogorov-Smirnov	Ultrapassar	Valor-F	0,0988	0,64032	SIM
Anderson-Darling	Ultrapassar	Valor-F	0,5366	0,15552	SIM
Shapiro-Wilk	Estacionar	Valor-F	0,7283	4,14E-06	NÃO
Kolmogorov-Smirnov	Estacionar	Valor-F	0,2458	7,04E-05	NÃO
Anderson-Darling	Estacionar	Valor-F	2,8607	2,21E-07	NÃO
Shapiro-Wilk	Ultrapassar	NDCR	0,9384	0,08245	SIM
Kolmogorov-Smirnov	Ultrapassar	NDCR	0,1293	0,22577	SIM
Anderson-Darling	Ultrapassar	NDCR	0,6043	0,10604	SIM
Shapiro-Wilk	Estacionar	NDCR	0,5274	1,06E-08	NÃO
Kolmogorov-Smirnov	Estacionar	NDCR	0,3777	4,16E-12	NÃO
Anderson-Darling	Estacionar	NDCR	6,5974	1,34E-16	NÃO

estimar os intervalos de confiança para distribuições normais e não normais. Por meio das técnicas citadas, foi possível estimar os intervalos de confiança de todas as amostras de dados. A Tabela 4.7 apresenta os intervalos de confiança com 5% de significância para todas as métricas utilizadas, exceto para precisão e revocação, pois o Valor-F é a média harmônica de precisão e revocação.

Tabela 4.7: Intervalos de confiança.

Evento	Métrica	Intervalos
Evento	Métrica	Intervalos
Ultrapassar	Acurácia	(81,95 - 89,36)
Estacionar	Acurácia	(83,02 - 93,17)
Ultrapassar	Valor-F	(64,84 - 76,11)
Estacionar	Valor-F	(83,72 - 83,72)
Ultrapassar	NDCR	(0,342 - 0,484)
Estacionar	NDCR	(0,016 - 0,122)

Seguindo a métrica NDCR, um resultado de detecção de evento com valor igual a 1 indica que o sistema não realizou nenhuma detecção e o valor igual a 0 indica que detectou todos os quadros em que ocorre o evento. A partir desta métrica pode-se comprovar que todos os eventos de ultrapassagens do sinal vermelho e de estacionamento foram detectados, como demonstrado pela Figura 4.3, na qual a linha vermelha superior e inferior indicam respectivamente o 1 e o 0 do boxplot. Nenhum dos resultados assume o valor 1, o que demonstra que ao menos um quadro de cada evento foi encontrado, e indica que o arcabouço não vai falhar em todas as situações. Como apresentado na Tabela 4.7, tem-se que a acurácia de todos os eventos contém intervalos que iniciam acima de 80% e intervalos de Valor-F que iniciam acima de 60%. Pode-se também observar nas Tabelas 4.5 e 4.4, que todas as métricas indicam que pelo menos alguns quadros dos vídeos que continham o evento foram detectados. Dessa forma, com



intervalos posicionados acima de 50% tanto para acurácia quanto para o Valor-F, e com valores de NDCR que são inferiores a 1, pode-se com os dados analisados afirmar que o arcabouço foi capaz de detectar a ocorrência dos eventos nos dois ambientes propostos.

Porém, para demonstrar que os dados obtidos não foram meramente acaso, foram utilizados o (1) Coeficiente de Correlação de Matthews (CCM) (MATTHEWS, 1975), que é em essência um coeficiente de correlação entre as classificações binárias existentes e as detectadas, que retorna valores entre -1 e +1, em que +1 representa uma detecção perfeita, 0 uma detecção não superior ao acaso e -1 representa total falha nas detecções e o (2) Índice de Youden (J)(YOUDEN, 1950), que utiliza como base a taxa de verdadeiros positivos (revocação) e a taxa de verdadeiros negativos (especificidade), que retorna valores entre 0 e 1, sendo 0 quando o sistema retorna a mesma proporção para verdadeiros positivos e para falsos positivos e 1 quando não há falsos positivos nem falsos negativos, considerando o resultado 1 como uma detecção perfeita. As equações para o coeficiente de correlação de Matthews (Equação 4.7) e para o índice de Youden (Equação 4.8) são definidas a seguir:

$$CCM = \frac{VP \times VN - FP \times FN}{\sqrt{((VP + FP)(VP + FN)(VN + FP)(VN + FN))}} \tag{4.7}$$

$$J = \frac{VP}{VP + FN} + \frac{VN}{FP + VN} - 1 \tag{4.8}$$

A partir destas duas métricas calculadas sobre todos os eventos juntos, tem-se indícios de que as detecções realizadas pelo arcabouço não foram mero acaso, pois com os intervalos de confiança com significância de 5% para o CCM é de 0,66 a 0,78, sendo assim superior ao zero, que representa mero acaso. Também

com significância de 5% o intervalo de confiança do índice de Youden é de 0,61 a 0,73. Em ambas as métricas os resultados foram superiores a zero, o que indica que não houve perfeição nos resultados, porém os resultados obtidos não foram mero acaso, nem uma falha total de detecção. Cumprindo assim um dos objetivos específicos desta pesquisa, que é provar a eficácia e a generalidade do arcabouço.

Durante os experimentos constatou-se que as falhas nos detectores afetavam os resultados das detecções de eventos. Porém, ao se analisar a correlação entre as métricas Valor-F do detector de carros e o do detector de eventos para os eventos de ultrapassagens do sinal vermelho, tem-se um coeficiente de correlação de 0,087, que indica que a correlação entre as variáveis é muito baixa. O mesmo é visto no gráfico de dispersão na Figura 4.6. No entanto, vídeos com discrepâncias entre essas métricas ocorrem devido a uma limitação do detector, pois o mesmo só detecta objetos em movimento, porém os rótulos marcam objetos que pararam após o movimento, como pode ser observado na Figura 4.7. Na qual estão marcados como *ground truth* todos os veículos que estão ou já estiveram em movimento, e a Figura 4.8 em que o sistema deixa de marcar os veículos que pararam de se mover (car1 e car2). Dessa forma, o detector falha, no entanto, a falha não é no objeto significante para o evento (o objeto que cruza a faixa), o que faz com que o Valor-F do evento seja mais alto que o do detector. Também temos casos inversos, onde o Valor-F do detector é melhor que o do evento, o que pode ser explicado pela falta de precisão na área do objeto detectado, como pode ser observado na Figura 4.9, retirada do vídeo Excelsior.

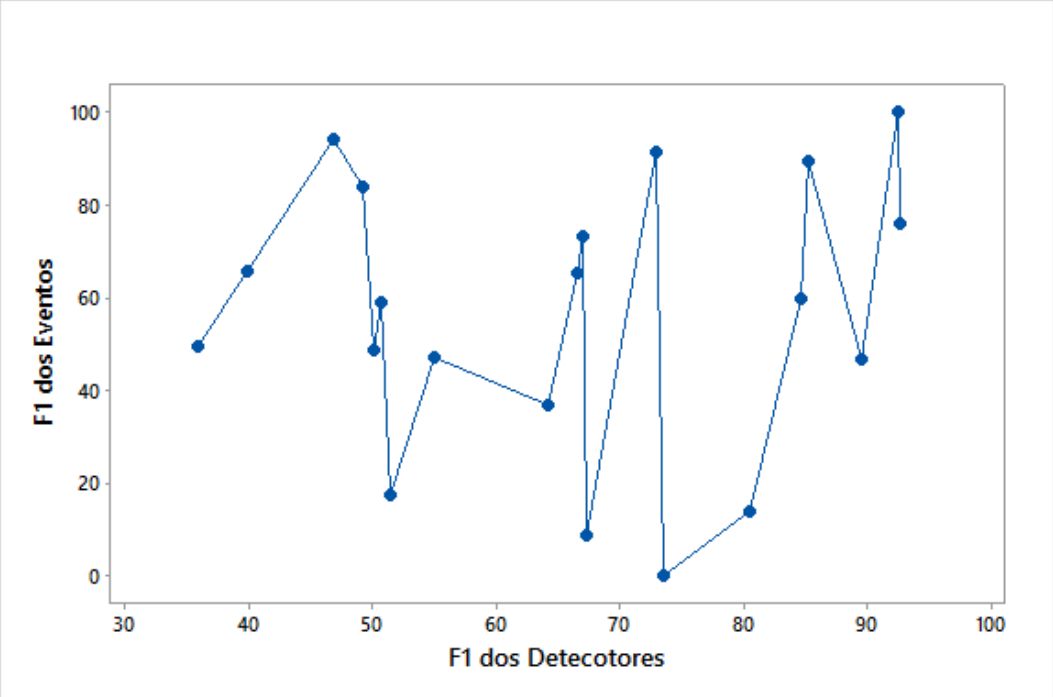


Figura 4.6: Gráfico de dispersão de F1 dos eventos versus F1 do detector.

Figura 4.7: Detector de eventos utilizando os rótulos manuais, os polígonos brancos são as detecções e o texto é o tipo e o identificador numérico do objeto detectado (ATS, 2015).

Figura 4.8: Detector de eventos utilizando dados do detector de carros, os polígonos brancos são as detecções, e o texto é o tipo e o identificador numérico do objeto detectado (ATS, 2015).

Figura 4.9: Falha no detector de objetos, a detecção “Car3” demarca dois objetos como sendo um único objeto (ATS, 2015).

Então, como apresentado anteriormente, os experimentos para os eventos de ultrapassagem foram realizados de duas formas: utilizando as detecções realizadas pelo detector e utilizando os rótulos manuais como se fossem detecções realizadas pelo detector (simulando). Dessa forma, com os dados simulados (rótulos manuais), que não contém falhas de detecção no **objeto que dispara a regra quando o evento acontece**, os resultados se mantiveram constantes, sempre alcançando os resultados máximos, como observado nos resultados do tipo **Sim.**, na Tabela 4.5. Porém, com os dados detectados (retorno do detector), que contêm falhas de detecção no **objeto que dispara a regra quando o evento acontece**, foram obtidos os resultados do tipo **Det.**, na Tabela 4.5. Com isso, pode-se concluir que se houver falhas de detecção no objeto que dispara a regra quando o evento acontece, haverá falhas nos resultados das detecções do evento.

Portanto, podemos formar a partir dos dados da Tabela 4.5 duas proposições verdadeiras em LPO, predicando sobre eventos x:

$$\forall\, x\, .\, SIMULADO(x) \;\Rightarrow \neg FALHA\_EVENTO(x)$$
$$\exists\, x\, .\, DETECTADO(x)\; \&\; FALHA\_EVENTO(x)$$

Nas quais os predicados SIMULADO e DETECTADO determinam o tipo de detecção, e o predicado FALHA_EVENTO determina se há falhas na detecção do evento. Desta forma, pode-se afirmar que para qualquer vídeo usando dados simulados não se tem falhas nas detecções de eventos. E que existem vídeos usando dados detectados que tem falhas nas detecções de eventos. A partir das expressões lógicas apresentadas anteriormente, pode-se com os dados analisados considerar que as falhas nos detectores afetam os resultados das detecções

de eventos. Porém, para afetar a detecção de eventos, as falhas nos detectores devem ocorrer com o objeto que dispara a regra no momento em que o evento está acontecendo, ou ocorrer de modo a gerar um falso positivo (quando o sistema acusa haver um evento e o mesmo não existe).

Como apresentado anteriormente, tem-se que as falhas oriundas dos detectores afetam as detecções de eventos, assim, quando o detector falha em alguns dos quadros, o evento deixa de ser detectado. Os vídeos utilizados nos testes contêm taxas e 29 e 30 quadros por segundo, o que indica que a cada segundo pode-se obter em média 30 detecções com resultados de verdadeiros ou falsos. Porém, segundo Thorpe, Fize e Marlot (1996) ao estudarem sobre a velocidade do processamento no sistema visual humano, a média de tempo de reação e de 445 ms, com o menor tempo em 382 ms e o maior com 567 ms. Com esse tempo de reação de 445 ms ou 0,445 segundos, pode-se considerar que um humano ao analisar uma imagem ou os resultados de uma detecção, demora aproximadamente meio segundo para identificar a informação que é apresentada com saída de um sistema de detecção de eventos em vídeo.

Como todos os vídeos têm uma taxa de amostragem de em média aproximadamente 30 quadros por segundo e que o tempo de reação de um humano é de aproximadamente 0,5 segundos, tem-se que uma pessoa demora em média aproximadamente 15 quadros para interpretar uma informação oriunda da aplicação externa que utiliza o arcabouço. Considerando essas afirmações com base na pesquisa de Thorpe, Fize e Marlot (1996), foram realizados testes agrupando as detecções de eventos em blocos, para assim reduzir o efeito das falhas de detecções sobre a detecção de eventos.

Para o experimento, foram criado blocos, os quais foram organizados de forma a não superar 0,5 segundos, ou seja, 15 quadros. Essa divisão ficou organizada segundo a Tabela 4.8, que contempla blocos de 5, 10 e 15 quadros e uma quantidade de acertos de 1%, 25%, 50%, 75% e 100%, quando o valor resultante desta porcentagem não for fracionado e for passivo de arredondamento sem duplicar a quantidade de acertos.

Tabela 4.8: Experimentos por blocos.

Nome	Bloco/Quadros	Acertos	Porcentagem
B1-A1	1	1	100%
B5-A1	5	1	1%
B5-A2	5	2	50%
B5-A5	5	5	100%
B10-A1	10	1	1%
B10-A2	10	2	25%
B10-A5	10	5	50%
B10-A7	10	7	75%
B10-A1	10	10	100%
B15-A1	15	1	1%
B15-A4	15	4	25%
B15-A7	15	7	50%
B15-A11	15	11	75%
B15-A15	15	15	100%

A partir dos blocos foram gerados gráficos mostrando a curva de desempenho dos blocos em relação aos resultados sem o agrupamento, os gráficos são os do Bloco de 5 quadros na Figura 4.10, em que o valor sem blocos é expresso por um gráfico de barras, e as linhas indicam os blocos. Essas mesmas indicações são utilizadas para os outros blocos, sendo assim, tem-se a Figura 4.11 com os blocos de 10 quadros e a Figura 4.12 com os blocos de 15 quadros. Para uma melhor visualização dos dados, os mesmos foram organizados no gráfico de forma crescente.

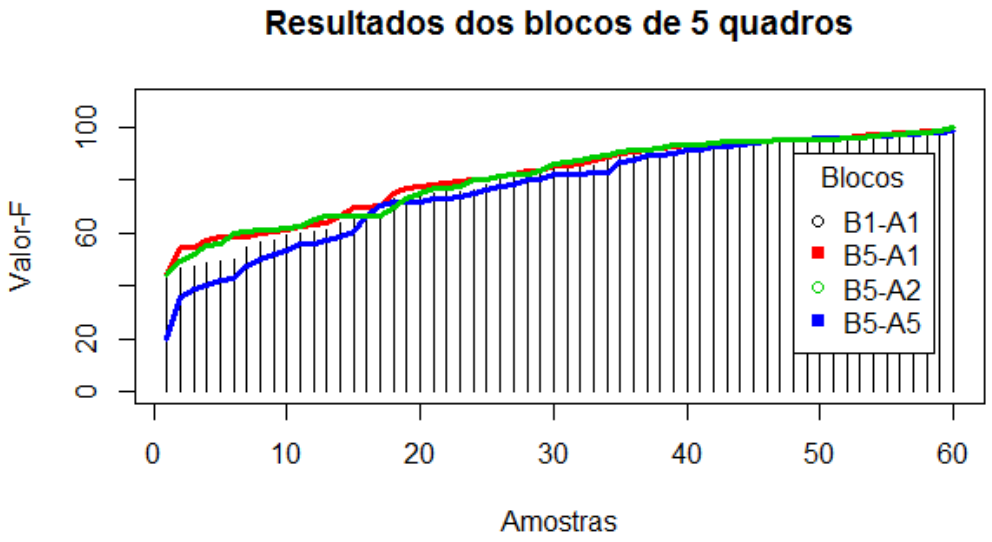


Figura 4.10: Resultados por blocos de 5 quadros.

Para comprovar se a blocagem dos quadros realmente causava efeito nos resultados e que os resultados obtidos não seriam iguais em variação, foi realizada a análise de variância (ANOVA) dos dados. A análise da variância dos dados obteve um valor P de 0,0006, valor esse menor que a significância de 5%. A análise também obteve um valor F de 2,7214, que é superior ao valor de F crítico que é de 1,7029. Dessa forma, podemos afirmar que a blocagem dos dados impacta nas taxas de detecções de eventos. Porém, para analisar se essa variância tem impacto positivo nos resultados do Valor-F (métrica selecionada para a análise) a Tabela 4.9 apresenta a soma da área sob a curva de desempenho de cada bloco, ordenada do menor para o maior, assim como o gráfico da Figura 4.13 com os 5 melhores resultados. Para uma análise mais detalhada, o Gráfico 4.14 mostra os resultados dos dois melhores blocos e o resultado das detecções sem blocos sem ordenação, assim é visível o desempenho dos blocos em cada um dos vídeos

A partir dos dados apresentados, tem-se fortes indícios que há variação nos resultados, e que há melhoramento nos resultados dos blocos comparados aos resultados sem blocagem. O que evidencia que o agrupamento em blocos dos resultados das detecções de eventos quadro a quadro influencia nas taxas de detecção dos eventos, devido haver variância nas amostras e os resultados de

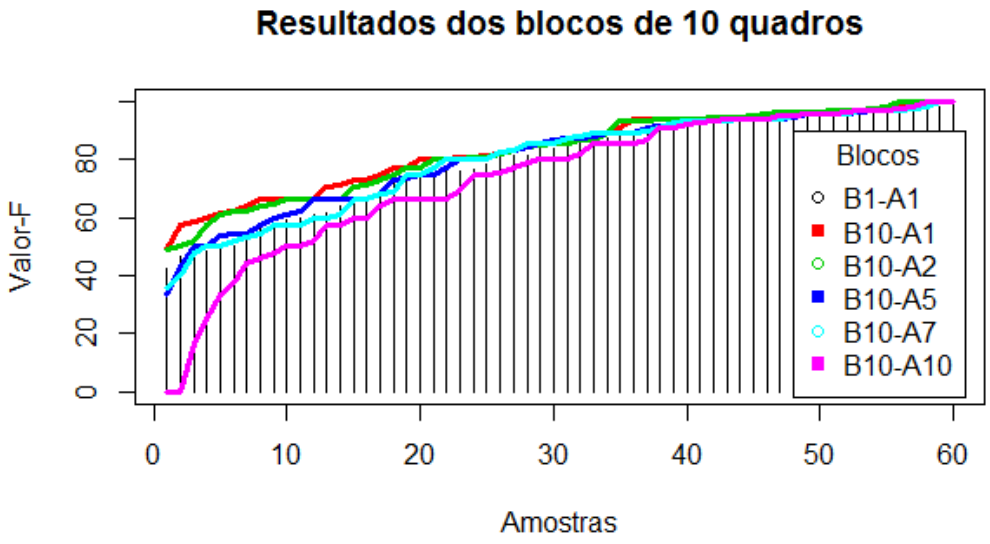


Figura 4.11: Resultados por blocos de 10 quadros.

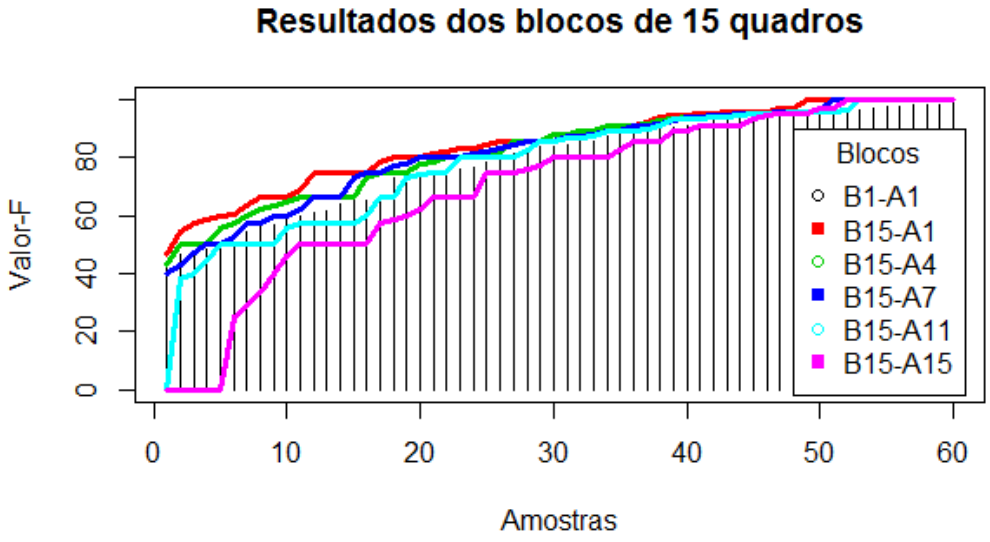


Figura 4.12: Resultados por blocos de 15 quadros.

alguns blocos serem superiores aos resultados sem blocagem. Como exemplo tem-se que o valor médio do Valor-F dos eventos é de 79,46% e que o melhor Valor médio do Valor-F do melhor bloco é de 84,84%, porém, ao analisar os intervalos de confiança não podemos afirmar com significância estatística que os resultados dos blocos são superiores aos resultados sem blocos. Pois como demonstrado no Boxplot 4.15 há intersecção entre os intervalos de confiança, que com significância de 5% são de 81,29% a 88,33% para o melhor bloco e de 74.93 a 83.84 sem blocos. A intersecção entre os blocos é destacada por meio das

Tabela 4.9: Área sob a curva dos experimentos por blocos.

Bloco	Área sob a curva
B15-A15	4154,56
B10-A10	4384,69
B5-A5	4550,65
B15-A11	4644,69
B1-A1	4697,40
B10-A7	4723,90
B10-A5	4764,83
B5-A1	4805,27
B5-A2	4805,27
B5-A1	4831,00
B15-A7	4848,47
B15-A4	4878,24
B10-A2	4898,04
B10-A1	4943,75
B15-A1	5016,89

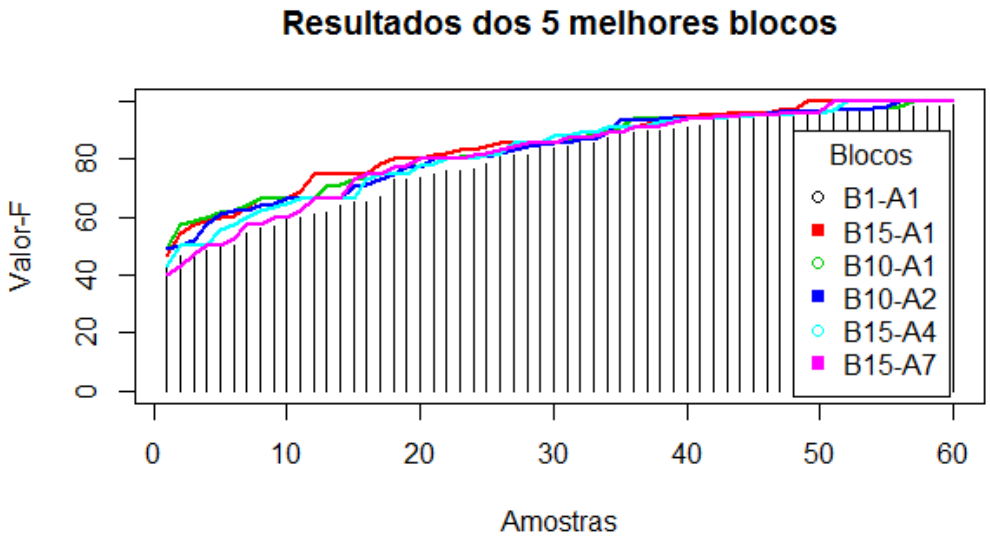


Figura 4.13: Resultados dos 5 melhores blocos.

linhas vermelhas no gráfico.

Também foi possível observar que o evento não foi encontrado com os dados detectados nem com os dados simulados para o vídeo Nassau01, com a regra apresentada no Capítulo 3. Tal fato se dá devido ao modelo da regra não contemplar a situação em que, a transição da cor amarela para a vermelha acontece no momento após a marcação do carro entrar na demarcação da faixa. Pois no modelo apresentado na Seção 3.2.3, as transições entre a estrada e a faixa devem acontecer durante o sinal vermelho (o termo “durante” significa que o evento deve ocorrer depois do início e antes do final), e no vídeo o início da transição acontece quando o sinal está amarelo.

Sendo assim, ajustando o modelo com a remoção do intervalo de Allen “durante” da regra, temos que o evento precisa apenas que o sinal esteja vermelho

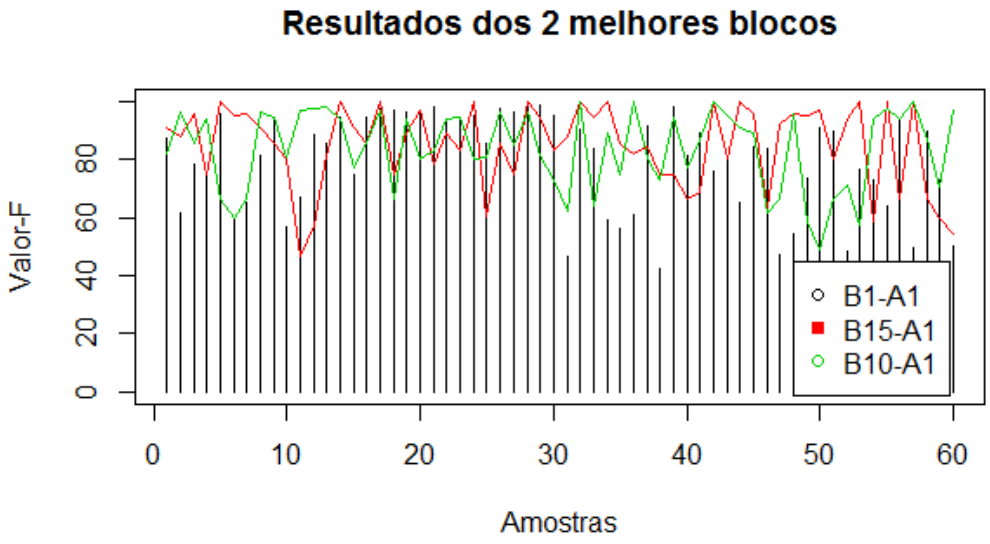


Figura 4.14: Resultados dos 2 melhores blocos sem ordenação.

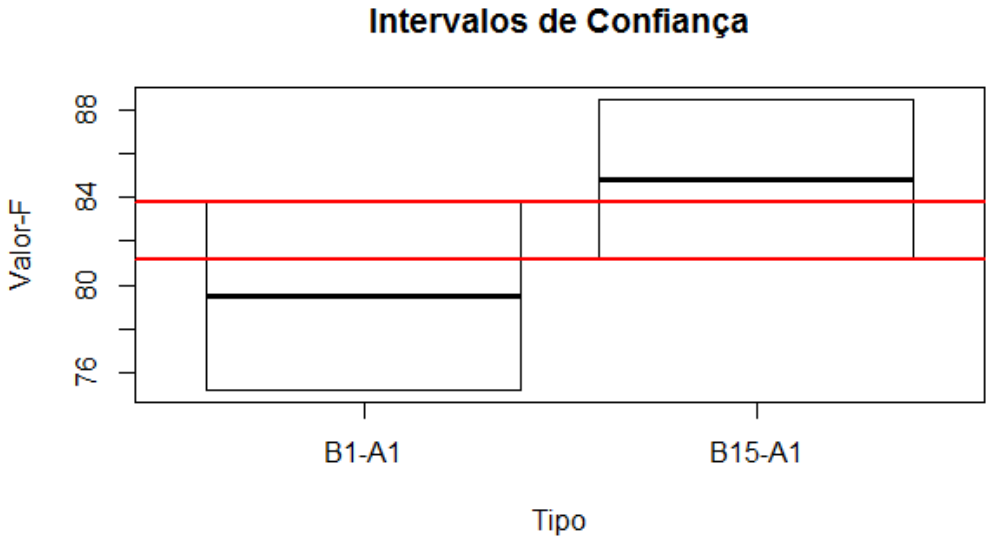


Figura 4.15: Boxplot dos intervalos de confiança do melhor bloco e sem bloco.

no momento em que for validada a transição. Com essa mudança tem-se como resultado a detecção do evento, o que mostra a não completude do modelo do evento afeta a detecção do mesmo, em que um modelo completo é definido por um modelo que abranja as situações necessárias para detecção do evento em todos os cenários propostos. A Tabela 4.10 mostra os resultados para o detector com a nova regra para o vídeo Nassau01, e em seguida a regra modificada.

```
{ "Rules" : {
        "Name" : "Carro ultrapassa",
        "Object" : "Car",
        "Plugins": {"Detection": [
                        "CarDetect",
                        "FixedObjects", Valor–F
                        "TrafficLight" ],
        "inference": ["RulesInference"] },
        "Operators" : {"OP1" : "AND",
                "OP2" : "AND"
        },
        "Sequences" : {
           "S1" : {
          "Function" : "COLOR",
          "Param" : {
                "V1" : {"Name" : "TfA",
                        "Type" : "TrafficLight" },
                "V2" : {"Type" : "CONST",
                        "Value" : "RED"}},
          "Temporal" : {}},
           "S2" : {
          "Function" : "OVERLAP",
          "Param" : {
                "V1" : {"Type" : "THIS" },
                "V2" : {"Name" : "Road1",
                        "Type" : "Road" },
                "V3" : {"Type" : "CONST",
                        "Value" : "20" }},
          "Temporal" : {}},
           "S3" : {
          "Function" : "OVERLAP",
          "Param" : {
                "V1" : {"Type" : "THIS" },
                "V2" : {"Name" : "Crosswalk1",
                        "Type" : "Crosswalk" },
                "V3" : {"Type" : "CONST",
                        "Value" : "20" }},
          "Temporal" : { "BEFORE" : "S2"}
}}}}
```

Tabela 4.10: Resultados no evento de ultrapassagem de sinal com a utilização da nova regra após alteração.

Tipo	Vídeo	Detectado	NDCR	Acurácia	Precisão	Revocação	Valor-F
Det.	Nassau01	SIM	0,41	74,67	100,00	59,14	74,32
Sim.	Nassau01	SIM	0,00	100,00	100,00	100,00	100,00

4.4 Considerações finais

Os testes realizados nesta seção avaliaram a eficácia e a generalidade do arcabouço proposto, pois o mesmo foi capaz de detectar eventos para dois ambientes diferentes, atingindo uma acurácia média para a soma de todos os eventos de 86,91%. Comparado com as aplicações que mais se assemelham ao tema proposto que são apresentadas na Secção 2.3. Tem-se que a pesquisa proposta nesta dissertação é utilizada em mais de um cenário, e não utiliza um ambiente controlado e um único ponto de visão para a câmera como o trabalho de Yung e Lai (2001), dessa forma, a pesquisa proposta difere da anteriormente citada por

se mostrar genérica. As pesquisas de Shet, Harwood e Davis (2005) e Krausz e Herpers (2010) tem como foco a detecção de pessoas em região de perigo e atitudes suspeitas em estações de trens, não apresentando forma de reconfiguração para outros tipos de eventos nem ambientes. A pesquisa de Almeida (2010) e D'Odorico (2013) não apresenta resultados utilizando detectores reais em tempo real de processamento, dado que o primeiro utiliza dados simulados e o segundo apenas rótulos manuais. A aplicação proposta na pesquisa de Lim, Tang e Chan (2014) realiza detecção de múltiplos eventos e utiliza de lógica para as regras que determinam a ocorrência ou não dos eventos, porém não é apresentado formas de extensão dos módulos de detecção nem dos eventos que podem ser detectados, como também não apresenta formas de modificar as regras estipuladas. O arcabouço proposto por Lim, Tang e Chan (2014) apresenta uma acurácia média para todos os eventos de 83,02% e é comparado com o estado-da-arte em termos de funcionalidades, dessa forma a pesquisa compara apenas se tem mais funcionalidades. Comparado com esta pesquisa, o presente arcabouço provê a possibilidade de extensão dos eventos por meio da adição de novos módulos e a possibilidade de modificar as regras para abranger outros eventos, não fixados a totalidade de eventos que podem ser detectados pelo trabalho desenvolvido nesta pesquisa.

Em comparação com as pequisas citadas, tem-se que o arcabouço proposto difere-se do arcabouço de Lim, Tang e Chan (2014) por ser explicitamente extensível, apresentar formas de modificar as regras de detecção de evento e utilizar de contextualização temporal para estas regras. Diferencia-se dos demais por ser genérico e configurável, podendo ser configurado para aplicações diversas no domínio de vigilância.

Capítulo 5

Conclusão e trabalhos futuros

O presente trabalho de dissertação objetivou a criação de um arcabouço genérico e extensível para a detecção de eventos com base em regras. A principal contribuição do trabalho foi uma arquitetura extensível e configurável, que contempla a possibilidade de utilização para diferentes cenários, como por exemplo, a partir dos módulos existentes é possível fazer um controle entrada de veículos em local proibido, com apenas a criação do modelo da regra e configuração dos objetos fixos. Da mesma forma, com a adição de um módulo de detecção de placas, é possível fazer a identificação do carro, ou com um detector de faces, saber se uma pessoa não autorizada entra em uma área restrita.

Também há a possibilidade de se adicionar novos módulos de inferência, e com isso fazer uso de métodos estatísticos ou de aprendizagem de máquina para melhorar os resultados das detecções, assim como também a possibilidade de adicionar novas funções de interação, como por exemplo aproximar, esperar, parar, dentre outras. Cada nova função pode aumentar o leque de possibilidades de utilização do arcabouço proposto.

Os arquivos de configurações do arcabouço foram projetados para ser também extensíveis, então os módulos podem mudar os parâmetros internos sem interferir diretamente no núcleo do arcabouço. Tanto as regras quanto as configurações foram projetadas de forma a facilitar o entendimento por humanos, e também possibilitar uma fácil tradução da linguagem das regras para uma linguagem visual, o que contribui para a possibilidade de criação, como trabalho futuro, de um ambiente visual para a modelagem das regras e configurações do sistema.

Levando em consideração os objetivos propostos, foi possível empregar o arcabouço proposto para modelar eventos com base em lógica e álgebra de Allen. Também foi proposta uma estrutura de armazenamento das regras e dos arquivos de configuração. A avaliação experimental comprovou a eficácia do arcabouço, o qual detectou ao menos alguns quadros de todos os eventos especificados nos modelos, alcançando F1-Score de 100% quando utilizando os dados dos rótulos manuais como detecções. Com detectores reais, o F1-Score ficou no intervalo de 75,14% a 83,70% com 95% de confiança.

Finalizando, a pesquisa realizada atingiu os objetivos planejados, tendo sido

apresentado um arcabouço genérico e extensível para detecção de eventos, o qual foi experimentalmente validado e sua eficácia foi comprovada em mais de um ambiente.

Diante dos resultados apresentados no Capítulo 4 e da arquitetura proposta no Capítulo 3, abre-se um leque de possibilidades devido à generalidade do arcabouço. Como propostas para trabalhos futuros, tem-se:

- Desenvolvimento de um ambiente gráfico para criação, edição das regras e configurações do arcabouço, de modo que um usuário não especialista no domínio possa criar novas regras e modificar o sistema de modo a atender as necessidades dele;
- Criação de módulos para funções de interações, para que assim não seja necessário alterar o módulo de inferência para adição dessas funções;
- Adição de métodos mais robustos para a detecção, rastreamento e reconhecimento dos objetos;
- Criação de estratégias para contornar as falhas dos detectores;
- Criar uma estratégia para utilização de eventos com elementos extraídos de câmeras diferentes, ou seja, uma abordagem multi-câmera.

Bibliografia

ALBANESE, M. et al. A constrained probabilistic petri net framework for human activity detection in video. *Multimedia, IEEE Transactions on*, v. 10, n. 8, p. 1429–1443, Dec 2008. ISSN 1520-9210.

ALLEN, J. *Natural Language Understanding (2Nd Ed.)*. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc., 1995. ISBN 0-8053-0334-0.

ALLEN, J. F. Maintaining knowledge about temporal intervals. *Commun. ACM*, ACM, New York, NY, USA, v. 26, n. 11, p. 832–843, nov. 1983. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/182.358434>.

ALMEIDA, F. F. M. *Uma Abordagem Baseada em Regras para Detecção Automática de Eventos Pré-Definidos em Vídeos*. Dissertação (Mestrado) — Programa de Pós-Graduação em Ciência da Computação, Universidade Federal de Campina Grande, Universidade Federal de Campina Grande, UFCG, 2010.

AOUN, N. B.; ELGHAZEL, H.; AMAR, C. B. Graph modeling based video event detection. In: *Innovations in Information Technology (IIT), 2011 International Conference on*. [S.l.: s.n.], 2011. p. 114–117.

ATS, A. T. S. *ATSRoadSafety*. 2015. <https://www.youtube.com/user/ATSRoadSafety>. Último acesso em:ponível em: 19 de Julho de 2015.

BAYAT, F. et al. A new framework for goal detection based on semantic events detection in soccer video. In: *Machine Vision and Image Processing (MVIP), 2013 8th Iranian Conference on*. [S.l.: s.n.], 2013. p. 184–189. ISSN 2166-6776.

CAMBRIDGE. *Cambridge Dictionaries Online*. 2014. Disponível em: <http://dictionary.cambridge.org/>.

CHATTOPADHYAY, C.; DAS, S. Seva- a salient event detection framework from video shots using support vector data description. In: *Emerging Applications of Information Technology (EAIT), 2014 Fourth International Conference of*. [S.l.: s.n.], 2014. p. 273–278.

CHEN, H.-S.; TSAI, W.-J. A framework for video event classification by modeling temporal context of multimodal features using hmm. *Journal of Visual Communication and Image Representation*, v. 25, n. 2, p. 285 – 295, 2014. ISSN 1047-3203. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1047320313002150>.

CHUANG, C.-H. et al. Model-based approach to spatial-temporal sampling of video clips for video object detection by classification. *Journal of Visual Communication and Image Representation*, v. 25, n. 5, p. 1018 – 1030, 2014. ISSN 1047-3203. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1047320314000522>.

CLIPPER. *Clipper - an open source freeware library for clipping and offsetting lines and polygons*. 2015. <http://www.angusj.com/delphi/clipper.php>. Último acesso em:ponível em: 19 de Julho de 2015.

COMANICIU, D.; MEER, P. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, IEEE Computer Society, Washington, DC, USA, v. 24, n. 5, p. 603–619, maio 2002. ISSN 0162-8828. Disponível em: <http://dx.doi.org/10.1109/34.1000236>.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*. Washington, DC, USA: IEEE Computer Society, 2005. (CVPR '05), p. 886–893. ISBN 0-7695-2372-2. Disponível em: <http://dx.doi.org/10.1109/CVPR.2005.177>.

DIGIFORT. *Digifort IP Surveillance System*. 2015. <http://www.digifort.com/>. Último acesso em:ponível em: 19 de Julho de 2015.

D'ODORICO, T. *An Ontological Analysis of Vague Motion Verbs, with an Application to Event Recognition*. Tese (Doutorado) — The University of Leeds School of Computing, 2013.

FLEISCHMAN, M.; DECAMP, P.; ROY, D. Mining temporal patterns of movement for video content classification. In: *Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*. New York, NY, USA: ACM, 2006. (MIR '06), p. 183–192. ISBN 1-59593-495-2. Disponível em: <http://doi.acm.org/10.1145/1178677.1178704>.

GHANEM, N. et al. Representation and recognition of events in surveillance video using petri nets. In: *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW '04. Conference on*. [S.l.: s.n.], 2004. p. 112–112.

GONZALEZ, R. C.; WOODS, R. E. *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. ISBN 013168728X.

GULER, S.; LIANG, W. H.; PUSHEE, I. A. A video event detection and mining framework. In: *Computer Vision and Pattern Recognition Workshop, 2003. CVPRW '03. Conference on*. [S.l.: s.n.], 2003. v. 4, p. 42–42. ISSN 1063-6919.

HAKEEM, A. *Learning, Detection, Representation, Indexing and Retrieval of Multi-agent Events in Videos*. Tese (Doutorado), Orlando, FL, USA, 2007. AAI3256923.

HOSSEINI, M.-S.; EFTEKHARI-MOGHADAM, A.-M. Fuzzy rule-based reasoning approach for event detection and annotation of broadcast soccer video. *Applied Soft Computing*, v. 13, n. 2, p. 846 – 866, 2013. ISSN 1568-4946. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1568494612004565>.

ISPYCONNECT. *Open Source Video Surveillance Software*. 2015. <http://www.ispyconnect.com>. Último acesso em:ponível em: 19 de Julho de 2015.

JAIN, V.; LEARNED-MILLER, E. *FDDB: A Benchmark for Face Detection in Unconstrained Settings*. [S.l.], 2010.

JHUO, I.-H. et al. Último acesso em:covering joint audio—visual codewords for video event detection. *Mach. Vision Appl.*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 25, n. 1, p. 33–47, jan. 2014. ISSN 0932-8092. Disponível em: <http://dx.doi.org/10.1007/s00138-013-0567-0>.

JOHNSON, R. E. Documenting frameworks using patterns. *SIGPLAN Not.*, ACM, New York, NY, USA, v. 27, n. 10, p. 63–76, out. 1992. ISSN 0362-1340. Disponível em: <http://doi.acm.org/10.1145/141937.141943>.

JSONCPP. *A C++ library for interacting with JSON*. 2015. <https://github.com/open-source-parsers/jsoncpp>. Último acesso em:ponível em: 19 de Julho de 2015.

KAMISHIMA, Y.; INOUE, N.; SHINODA, K. Event detection in consumer videos using gmm supervectors and svms. *EURASIP J. Image and Video Processing*, v. 2013, p. 51, 2013. Disponível em: <http://dblp.uni-trier.de/db/journals/ejivp/ejivp2013.html\#KamishimaIS13>.

KARDAS, K.; ULUSOY, I.; CICEKLI, N. K. Learning complex event models using markov logic networks. In: *Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on*. [S.l.: s.n.], 2013. p. 1–6.

KOVACS, L. et al. Digital video event detector framework for surveillance applications. In: *Advanced Video and Signal Based Surveillance, 2009. AVSS '09. Sixth IEEE International Conference on*. [S.l.: s.n.], 2009. p. 565–570.

KOWALSKI, R.; SERGOT, M. A logic-based calculus of events. *New Gen. Comput.*, Ohmsha, Tokyo, Japan, Japan, v. 4, n. 1, p. 67–95, jan. 1986. ISSN 0288-3635. Disponível em: <http://dx.doi.org/10.1007/BF03037383>.

KRAUSZ, B.; HERPERS, R. Metrosurv: Detecting events in subway stations. *Multimedia Tools Appl.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 50, n. 1, p. 123–147, out. 2010. ISSN 1380-7501. Disponível em: <http://dx.doi.org/10.1007/s11042-009-0367-8>.

LI, X.; PORIKLI, F. A hidden markov model framework for traffic event detection using video features. In: *Image Processing, 2004. ICIP '04. 2004 International Conference on*. [S.l.: s.n.], 2004. v. 5, p. 2901–2904 Vol. 5. ISSN 1522-4880.

LIM, M. K.; TANG, S.; CHAN, C. S. isurveillance: Intelligent framework for multiple events detection in surveillance videos. *Expert Systems with Applications*, v. 41, n. 10, p. 4704 – 4715, 2014. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417414000530>.

LINEHAN, M. H. et al. Controlled english language for production and event processing rules. In: *Proceedings of the 5th ACM International Conference on Último acesso em:tributed Event-based System*. New York, NY, USA: ACM, 2011. (DEBS '11), p. 149–158. ISBN 978-1-4503-0423-8. Disponível em: <http://doi.acm.org/10.1145/2002259.2002281>.

LU, J. et al. A framework for video event detection using weighted svm classifiers. In: *Artificial Intelligence and Computational Intelligence, 2009. AICI '09. International Conference on*. [S.l.: s.n.], 2009. v. 4, p. 255–259.

MALES, L.; ZARNIC, B. A spatiotemporal model of events within a bdi. In: *EUROCON - International Conference on Computer as a Tool (EUROCON), 2011 IEEE*. [S.l.: s.n.], 2011. p. 1–4.

MARR, D. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. New York, NY, USA: Henry Holt and Co., Inc., 1982. ISBN 0716715678.

MATTHEWS, B. Comparison of the predicted and observed secondary structure of {T4} phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure*, v. 405, n. 2, p. 442 – 451, 1975. ISSN 0005-2795. Disponível em: <http://www.sciencedirect.com/science/article/pii/0005279575901099>.

MEGHDADI, A.; IRANI, P. Interactive exploration of surveillance video through action shot summarization and trajectory visualization. *Visualization and Computer Graphics, IEEE Transactions on*, v. 19, n. 12, p. 2119–2128, Dec 2013. ISSN 1077-2626.

MERLER, M. et al. Semantic model vectors for complex video event recognition. *Multimedia, IEEE Transactions on*, v. 14, n. 1, p. 88–101, Feb 2012. ISSN 1520-9210.

MOTA, V. F. et al. Combining orientation tensors for human action recognition. In: *Graphics, Patterns and Images (SIBGRAPI), 2013 26th SIBGRAPI - Conference on*. [S.l.: s.n.], 2013. p. 328–333. ISSN 1530-1834.

MUTSCHLER, C.; PHILIPPSEN, M. Learning event detection rules with noise hidden markov models. In: *Adaptive Hardware and Systems (AHS), 2012 NASA/ESA Conference on*. [S.l.: s.n.], 2012. p. 159–166.

NA, H.; QIN, S.; WRIGHT, D. Detecting fall risk factors for toddlers. *Pervasive Computing, IEEE*, v. 10, n. 1, p. 82–89, 2011. ISSN 1536-1268.

ONAL, I. et al. A framework for detecting complex events in surveillance videos. In: *Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on*. [S.l.: s.n.], 2013. p. 1–6.

OPENCV. *OpenCV (Open Source Computer Vision)*. 2015. <http://opencv.org/>. Último acesso em:ponível em: 19 de Julho de 2015.

PICIARELLI, C.; FORESTI, G. Surveillance-oriented event detection in video streams. *Intelligent Systems, IEEE*, v. 26, n. 3, p. 32–41, 2011. ISSN 1541-1672.

PIXI, Z.; HONGYAN, L.; WEI, W. Research on event detection of soccer video based on hidden markov model. In: *Computational and Information Sciences (ICCIS), 2010 International Conference on*. [S.l.: s.n.], 2010. p. 865–868.

PRESSMAN, R. S. *Software Engineering: A Practitioner's Approach*. 5th. ed. [S.l.]: McGraw-Hill Higher Education, 2001. ISBN 0072496681.

QIAN, X. et al. Hidden conditional random field-based soccer video events detection. *Image Processing, IET*, v. 6, n. 9, p. 1338–1347, December 2012. ISSN 1751-9659.

QIAN, X. et al. Hmm based soccer video event detection using enhanced mid-level semantic. *Multimedia Tools Appl.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 60, n. 1, p. 233–255, set. 2012. ISSN 1380-7501. Disponível em: <http://dx.doi.org/10.1007/s11042-011-0817-y>.

RIEHLE, D.; GROSS, T. Role model based framework design and integration. *SIGPLAN Not.*, ACM, New York, NY, USA, v. 33, n. 10, p. 117–133, out. 1998. ISSN 0362-1340. Disponível em: <http://doi.acm.org/10.1145/286942.286951>.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, 2002. Hardcover. ISBN 0137903952. Disponível em: <http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&path=ASIN/0137903952>.

SANMIGUEL, J.; MARTINEZ, J. Use of feedback strategies in the detection of events for video surveillance. *Computer Vision, IET*, v. 5, n. 5, p. 309–319, September 2011. ISSN 1751-9632.

SEWARD, D. P. D. L. E. *Estatística Aplicada à Administração e Economia*. 4th. ed. [S.l.]: McGraw-Hill Higher Education, 2014.

SHET, V.; HARWOOD, D.; DAVIS, L. Vidmap: video monitoring of activity with prolog. In: *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference on*. [S.l.: s.n.], 2005. p. 224–229.

SOMMERVILLE, I. et al. *Engenharia de software*. São Paulo: Pearson Prentice Hall, 2008. ISBN 9788588639287 8588639289. Disponível em: <http://www.worldcat.org/search?qt=worldcat_org_all&q=9788588639287>.

TAVASSOLIPOUR, M.; KARIMIAN, M.; KASAEI, S. Event detection and summarization in soccer videos using bayesian network and copula. *Circuits and Systems for Video Technology, IEEE Transactions on*, v. 24, n. 2, p. 291–304, Feb 2014. ISSN 1051-8215.

THORPE, S.; FIZE, D.; MARLOT, C. Speed of processing in the human visual system. *Nature*, v. 381, p. 520, 1996.

TIAN, Y. et al. A novel generic framework of event detection in unmanned aerial videos. In: *Computational Intelligence and Natural Computing Proceedings (CINC), 2010 Second International Conference on*. [S.l.: s.n.], 2010. v. 1, p. 357–360.

TJONDRONEGORO, D.; CHEN, Y.-P. Knowledge-Último acesso em:counted event detection in sports video. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, v. 40, n. 5, p. 1009–1024, Sept 2010. ISSN 1083-4427.

WANG, F.; NGO, C.-W. Summarizing rushes videos by motion, object, and event understanding. *Multimedia, IEEE Transactions on*, v. 14, n. 1, p. 76–87, Feb 2012. ISSN 1520-9210.

WU, J.; HU, D. Learning effective event models to recognize a large number of human actions. *Multimedia, IEEE Transactions on*, v. 16, n. 1, p. 147–158, Jan 2014. ISSN 1520-9210.

WU, S.-Y.; CHEN, Y.-L. Mining nonambiguous temporal patterns for interval-based events. *Knowledge and Data Engineering, IEEE Transactions on*, v. 19, n. 6, p. 742–758, June 2007. ISSN 1041-4347.

XIAO, Y. et al. Power management for wireless data transmission using complex event processing. *Computers, IEEE Transactions on*, v. 61, n. 12, p. 1765–1777, Dec 2012. ISSN 0018-9340.

XU, W. et al. A framework of simple event detection in surveillance video. In: CHEN, R. (Ed.). *Intelligent Computing and Information Science*. Springer Berlin Heidelberg, 2011, (Communications in Computer and Information Science, v. 135). p. 556–561. ISBN 978-3-642-18133-7. Disponível em: <http://dx.doi.org/10.1007/978-3-642-18134-4_88>.

YOUDEN, W. J. Index for rating diagnostic tests. *Cancer*, Wiley Subscription Services, Inc., A Wiley Company, v. 3, n. 1, p. 32–35, 1950. ISSN 1097-0142. Disponível em: <http://dx.doi.org/10.1002/1097-0142(1950)3:1<32::AID-CNCR2820030106>3.0.CO;2-3>.

YUNG, N. H. C.; LAI, A. H. S. An effective video analysis method for detecting red light runners. *Vehicular Technology, IEEE Transactions on*, v. 50, n. 4, p. 1074–1084, Jul 2001. ISSN 0018-9545.

ZHAI, Y. et al. Chapter 19 - composite event detection in multi-camera and multi-sensor surveillance networks. In: AGHAJAN, H.; CAVALLARO, A. (Ed.). *Multi-Camera Networks*. Oxford: Academic Press, 2009. p. 457 – 480. ISBN 978-0-12-374633-7. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780123746337000215>.

ZONEMINDER. *ZoneMinder*. 2015. <http://www.digifort.com/>. Último acesso em:ponível em: 19 de Julho de 2015.

Apêndice A

Arquivo de configuração do detector de carros

Segue neste apêndice um exemplo do arquivo de configuração utilizado pelo detector de carros.

```
{
        "Configuration" : {
                "CarDetectFile" : "c:\\caroutput.txt",
                "CarHist" : 10,
                "Historic" : 100,
                "LearnRate" : 0.008,
                "Mixtures" : 3,
                "MinCarWidth" : 30,
                "MinCarHeight" : 30,
                "PerReduction" : 0.9,
                "Points" : [
                    {"Point" : {"x" : 3, "y" : 308}},
                    {"Point" : {"x" : 463, "y" : 136}},
                    {"Point" : {"x" : 620, "y" : 137}},
                    {"Point" : {"x" : 399, "y" : 358}},
                    {"Point" : {"x" : 4, "y" : 357}}
                ],
                "Threshold" : 10
        }
}
```

Apêndice B

Arquivo de configuração do detector de objetos fixos

Segue neste apêndice um exemplo do arquivo de configuração utilizado pelo detector de objetos fixos.

```
{"objects" : [{
        "Object" : {
                "Name" : "Road1",
                "Points" : [
                {"Point" : {"x" : 237,  "y" : 187}},
                {"Point" : {"x" : 544,  "y" : 198}},
                {"Point" : {"x" : 340,  "y" : 359}},
                {"Point" : {"x" : 3,    "y" : 357}},
                {"Point" : {"x" : 3,    "y" : 255}}
                ],
                "Type" : "Road"
        }
        }, {
        "Object" : {
                "Name" : "Crosswalk1",
                "Points" : [
                {"Point" : {"x" : 569,  "y" : 176}},
                {"Point" : {"x" : 551,  "y" : 192}},
                {"Point" : {"x" : 241,  "y" : 188}},
                {"Point" : {"x" : 254,  "y" : 173}}
                ],
                "Type" : "Crosswalk"
        }
        }
]}
```

Apêndice C

Arquivo de configuração do detector de cor do semáforo

Segue neste apêndice um exemplo do arquivo de configuração utilizado pelo detector de cor da luz do semáforo.

```
{"objects" : [{
        "Object" : {
                "Color" : "AZUL",
                "Name" : "TfA",
                "Points" : [
                {"Point" : {"x" : 536,  "y" : 45}},
                {"Point" : {"x" : 536,  "y" : 59}},
                {"Point" : {"x" : 540,  "y" : 59}},
                {"Point" : {"x" : 540,  "y" : 45}}
                ],
                "Type" : "TrafficLight"
        }}
]}
```

Apêndice D

Tecnologias utilizadas no arcabouço

Segue neste apêndice a descrição de tecnologias utilizadas para construção do sistema.

A construção do arcabouço foi realizada em C++, utilizando componentes externos à linguagem, como OpenCV (OPENCV, 2015) para o controle do fluxo do vídeo, assim como estruturas de controle, Jsoncpp (JSONCPP, 2015) para leitura e escrita de arquivos e variáveis no formato Json e o Clipper C++ (CLIPPER, 2015) para operações com polígonos. A utilização de cada um desses componentes faz parte das decisões arquiteturais que definem o funcionamento do sistema, são elas:

OpenCV: O arcabouço utiliza as estruturas de matrizes da biblioteca para armazenar e manipular as imagens a serem tratadas pelos módulos, assim como funções de subtração de plano de fundo, operação com retângulos e funções auxiliares.

Jsoncpp: A arquitetura com base em componentes realiza a comunicação entre os módulos por meio de mensagens, neste contexto, Jsoncpp permite simplificar o manuseio dessas mensagens e torná-las facilmente extensíveis. O padrão json, utilizado por esse pacote, foi adotado para troca de informações entre todos os módulos do sistema, tal decisão implica que o sistema que utilizar o arcabouço também terá que interpretar as mensagens nesse mesmo formato.

Clipper: A biblioteca Clipper realiza operações com linhas e polígonos, tendo sido utilizada para realizar operações com as áreas dos objetos detectados nos quadros para obtenção da detecção dos eventos.

A escolha da linguagem C++ se deu pela velocidade de processamento, a portabilidade e a integração com o OpenCV. O Jsoncpp e o Clipper foram escolhidos por apresentarem simplicidade na forma de uso e uma alta produtividade.

Apêndice E

Detalhamento da camada de detecção

O presente apêndice trata do detalhamento dos módulos criados para a camada de detecção. Para essa camada, foram implementados três Módulos: um para a detecção de carros, outro para a detecção da cor dos semáforos e um último para representação de objetos fixos. Mais detalhes são apresentados a seguir.

CarDetect: Utiliza subtração de plano de fundo implementada pela biblioteca OpenCV para detectar objetos se movendo nos quadros dos vídeos. Em seguida identifica a partir do tamanho se o objeto encontrado é um carro e retornar os dados das detecções para o arcabouço. A Figura E.1 apresenta o diagrama de classes desse módulo.

A classe **CarDetect** é responsável por implementar a interface IPlugin e o gerenciamento interno do módulo. A classe **carDetectConfiguration** é responsável por gerenciar as configurações do módulo e a **CarDetector** é responsável por guardar o modelo do plano de fundo e realizar a detecção e rastreamento dos carros. A classe **CarDetector** juntamente com classes auxiliares detectam o movimento no vídeo e classifica-os como sendo carro ou não. Para cada carro detectado é armazenado um modelo contendo o posicionamento e tamanho por um limiar de tempo configurável. Cada novo carro detectado é comparado com cada um dos modelos existentes, utilizando o índice de Jaccard. Ao comparar a detecção com o modelo, se o índice de Jaccard superar um determinado limiar, a detecção é considerada como sendo o mesmo carro descrito no modelo. Uma vez que a detecção é encontrada no conjunto de modelos, o modelo é atualizado com os dados da nova detecção, caso contrário, é criado um novo modelo para representar esse novo carro detectado.

TrafficLight: Recebe as marcações do semáforo e divide a área demarcada em três regiões, uma para cada cor do semáforo. Em seguida converte a imagem para tons de cinza, e calcula a intensidade da luz de cada região, assim determinando espacialmente qual a luz ativa, entre verde, vermelho e amarelo. A Figura E.2 apresenta o diagrama de classe deste módulo.

A classe **TrafficLight** é responsável por implementar a interface IPlugin e pelo gerenciamento interno do módulo. A classe **TrafficLightConfiguration** é

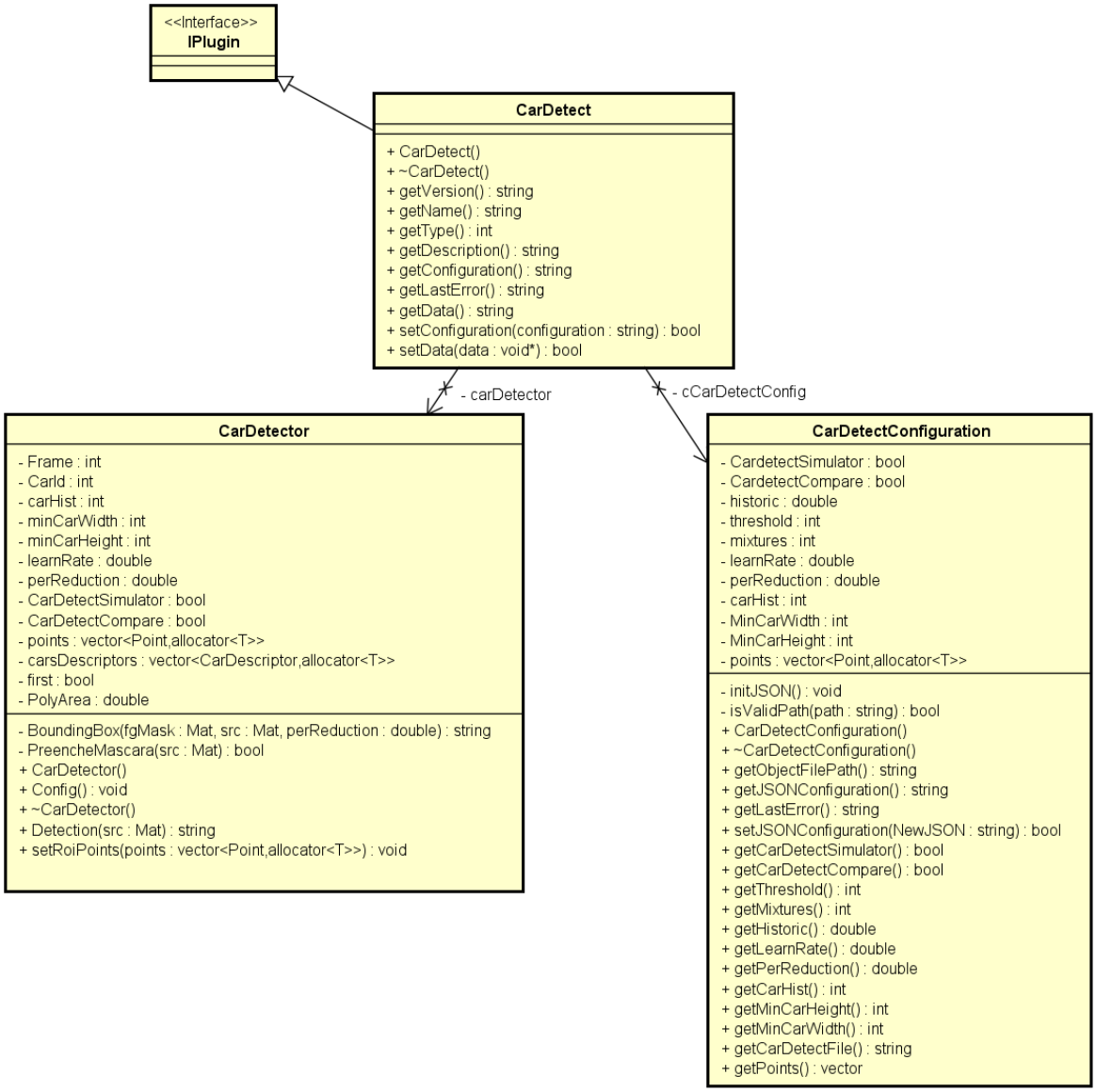


Figura E.1: Diagrama de Classes do módulo de detecção de carros.

responsável por gerenciar as configurações do módulo e a **TrafficLightColor** é responsável por analisar espacialmente a intensidade da luz e realizar a detecção da cor do sinal. A análise espacial é realizada ao dividir a área demarcada do semáforo em três áreas iguais: a área superior, a área intermediária e a área inferior. Para a análise da intensidade da luz, a imagem do semáforo é convertida para tons de cinza. Em seguida é realizado a soma de todas as intensidades dos *pixels* que compõe cada uma das três áreas. A área que contiver o maior intensidade é considerada como a área que contém a luz acesa, sendo verde para área superior, amarelo para área intermediária e vermelho para área inferior.

FixedObject: é responsável por receber o caminho de arquivo de demarcações fixas (objetos fixos na tela que interagem com as detecções dos outros detectores) e retornar para o RulesFramework a lista de objetos carregados como

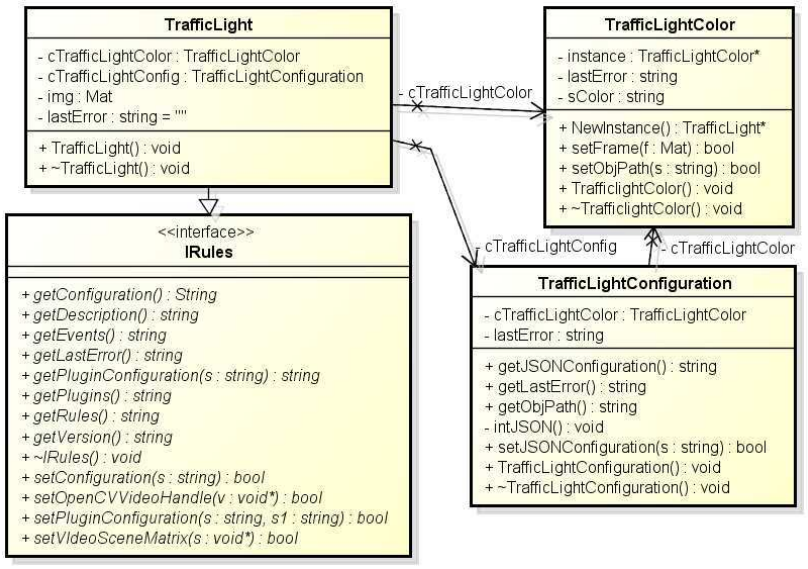


Figura E.2: Diagrama de Classes do módulo de cor do semáforo.

se fossem detecções providas por um detector, como por exemplo, a faixa de pedestre e a estrada. A Figura E.3 apresenta o diagrama de classes do módulo.

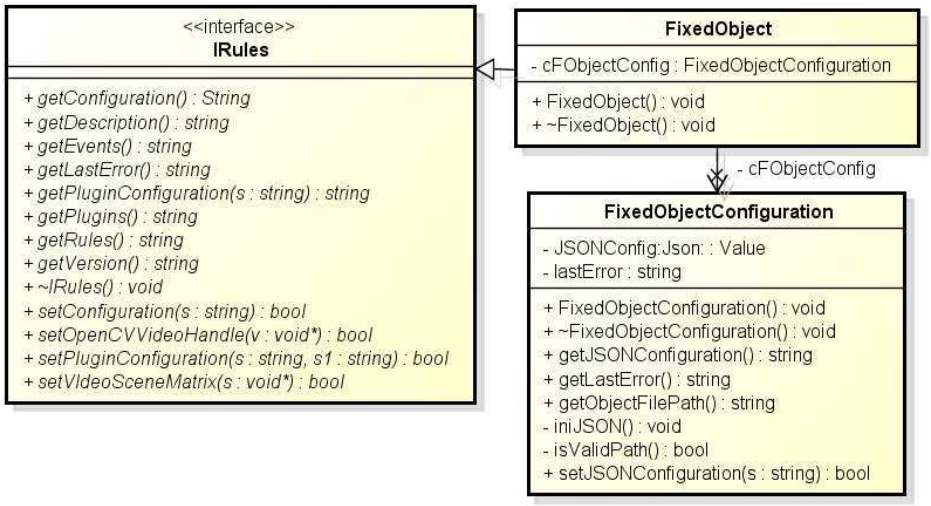


Figura E.3: Diagrama de Classes do módulo de detecção de objetos fixos.

A classe **FixedObject** é responsável por implementar a interface IPlugin, pelo gerenciamento interno do módulo e por retornar os objetos fixos para o arcabouço. A classe **FixedObjectConfiguration** é responsável por gerenciar as configurações do módulo e receber os arquivos com as marcações a serem retornadas.