Milhares de pessoas morrem e muitas outras ficam gravemente feridas todos os dias devido a acidentes de trânsito em todo o mundo. Grande parte destes acidentes são provocados por distrações e imprudência dos motoristas. Muitas empresas automobilísticas investem em tecnologias, porém ainda há pouquíssimos modelos de veículos que fazem prevenção de acidentes, e os que fazem, tem custo inacessível para a maioria das pessoas. Este trabalho apresenta o desenvolvimento de um sistema de visão computacional utilizando apenas uma câmera simples acoplada a um veículo, que notifica o motorista, através de alertas visuais e sonoros, quando há risco de colisão do seu veículo com veículos à sua frente. É estabelecida uma zona de risco à frente do veículo, e quando algum veículo estiver dentro desta região e começar a se aproximar a uma determinada distância do veículo com o sistema, o sistema emite alertas ao motorista, que terá tempo hábil de evitar uma colisão frontal com o veículo que está à sua frente. A detecção dos veículos é feita em intervalos de quadros com as técnicas Haar-like com AdaBoost, MB-LBP com AdaBoost e HOG com SVM. Dentro dos intervalos, cada veículo é rastreado por um algoritmo rápido e preciso chamado de KCF. Um mecanismo de tolerância a falhas com rastreadores auxiliares monitora e valida cada veículo. A validação dos veículos possui checagem dupla, uma com um dos detectores, e outra com uma nova técnica baseada no nível de confiança dos centroides de linhas verticais e horizontais. Caso algum veículo desapareça ou diminua seu nível de confiança de que a região ainda contém um veículo, apenas aquele veículo é re-detectado e o rastreamento continua. Caso o veículo não seja mais encontrado, uma nova detecção ocorre para sincronizar os rastreadores. Resultados mostram que isto melhora a precisão da detecção. Também melhora o tempo de processamento devido ao algoritmo de rastreamento processar a mais de 100 quadros por segundo, enquanto que o melhor detector, Haar-like com AdaBoost, processa a 17 quadros por segundo. Obteve-se resultados de até 99,7% de precisão na detecção com rastreamento, e tempo de processamento chegando a 70 quadros por segundo.

Orientador: Dr. André Tavares da Silva

Joinville, 2017

ANO
2017

MÁRCIO KOCH | SISTEMA PARA ALERTA DE COLISÃO VEICULAR PARA DETECÇÃO E RASTREAMENTO COM TOLERÂNCIA A FALHAS

UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
MESTRADO EM COMPUTAÇÃO APLICADA

DISSERTAÇÃO DE MESTRADO

SISTEMA PARA ALERTA DE COLISÃO VEICULAR PARA DETECÇÃO E RASTREAMENTO COM TOLERÂNCIA A FALHAS

MÁRCIO KOCH

JOINVILLE, 2017

MÁRCIO KOCH

SISTEMA PARA ALERTA DE COLISÃO VEICULAR PARA DETECÇÃO E RASTREAMENTO COM TOLERÂNCIA A FALHAS

Dissertação submetida ao Programa de Pós-Graduação em Computação Aplicada do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, para a obtenção do grau de Mestre em Computação Aplicada.

Orientador: Prof. Dr. André Tavares da Silva

JOINVILLE

2017

Ficha catalográfica elaborada pelo(a) autor(a), com
auxílio do programa de geração automática da
Biblioteca Setorial do CCT/UDESC

Koch, Márcio
Sistema para Alerta de Colisão Veicular para Detecção e Rastreamento com Tolerância a Falhas / Márcio Koch. - Joinville , 2017.
164 p.

Orientador: André Tavares da Silva
Dissertação (Mestrado) - Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada, Joinville, 2017.

1. Alerta de colisão veicular. 2. Detecção de veículos. 3. Rastreamento de veículos. 4. AdaBoost. 5. Haar-like. I. Silva, André Tavares da. II. Universidade do Estado de Santa Catarina. Programa de Pós-Graduação. III. Título.

Sistema para Alerta de Colisão Veicular para Detecção e Rastreamento com Tolerância a Falhas

por

Márcio Koch

Esta dissertação foi julgada adequada para obtenção do título de

Mestre em Computação Aplicada

Área de concentração em "Ciência da Computação,
e aprovada em sua forma final pelo

CURSO Mestrado Acadêmico em Computação Aplicada
CENTRO DE CIÊNCIAS TECNOLÓGICAS DA
UNIVERSIDADE DO ESTADO DE SANTA CATARINA.

Banca Examinadora:

Prof. Dr. André Tavares da Silva
CCT/UDESC (Orientador/Presidente)

Prof. Dr. Marcelo da Silva Hounsell
CCT/UDESC

Lilian Berton

Profa. Dra. Lilian Berton
ICT/UNIFESP

Joinville,SC, 28 de agosto de 2017.

Dedico este trabalho aos meus familiares, amigos, colegas de trabalho, professores e a todos os motoristas e pedestres que um dia possam se beneficiar de um sistema que auxilie na prevenção de acidentes de trânsito.

AGRADECIMENTOS

Agradeço a Deus por estar vivo após mais de 100 viagens de Blumenau para a UDESC em Joinville, nessas estradas perigosas e por ficar ileso em um acidente em uma das vindas de Joinville para casa.

A toda minha família, especialmente aos meus pais, Eloiza e Hains. Também pela compreensão e ajuda da minha esposa Ana Cristina, meu filho Mateus e minha sogra Adelaide (*in memoriam*), que partiu um pouco antes de eu terminar este trabalho. Ainda, ao meu cunhado Daniel, aos meus amigos e de maneira geral a todos que me apoiaram de alguma forma. Agradeço pela ajuda e pela compreensão por todos os dias, meses e anos afastado deles fazendo graduação, especialização e mestrado.

Expresso meus agradecimentos aos professores pelos ensinamentos, em especial ao professor Dr. André Tavares da Silva, meu orientador de mestrado, ao professor Dr. Marcelo da Silva Hounsell e a Dra. Lilian Berton pelas dicas que fizeram a diferença.

Agradeço também a Senior sistemas, empresa onde trabalho, que sempre permitiu minhas ausências do trabalho para poder ir para Joinville estudar na UDESC e fazer o mestrado.

“Há uma força motriz mais poderosa que o vapor, a eletricidade e a energia atômica: a vontade.”

Albert Einstein

RESUMO

KOCH, Márcio. **Sistema para Alerta de Colisão Veicular para Detecção e Rastreamento com Tolerância a Falhas.** 164 f. Dissertação (Mestrado em Computação Aplicada) – Universidade do Estado de Santa Catarina. Programa de Pós-Graduação em Computação Aplicada, Joinville, 2017.

Milhares de pessoas morrem e muitas outras ficam gravemente feridas todos os dias devido a acidentes de trânsito em todo o mundo. Grande parte destes acidentes são provocados por distrações e imprudência dos motoristas. Muitas empresas automobilísticas investem em tecnologias, porém ainda há pouquíssimos modelos de veículos que fazem prevenção de acidentes, e os que fazem, tem custo inacessível para a maioria das pessoas. Este trabalho apresenta o desenvolvimento de um sistema de visão computacional utilizando apenas uma câmera simples acoplada a um veículo, que notifica o motorista, através de alertas visuais e sonoros, quando há risco de colisão do seu veículo com veículos à sua frente. É estabelecida uma zona de risco à frente do veículo, e quando algum veículo estiver dentro desta região e começar a se aproximar a uma determinada distância do veículo com o sistema, o sistema emite alertas ao motorista, que terá tempo hábil de evitar uma colisão frontal com o veículo que está à sua frente. A detecção dos veículos é feita em intervalos de quadros com as técnicas Haar-like com AdaBoost, MB-LBP com AdaBoost e HOG com SVM. Dentro dos intervalos, cada veículo é rastreado por um algoritmo rápido e preciso chamado de KCF. Um mecanismo de tolerância a falhas com rastreadores auxiliares monitora e valida cada veículo. A validação dos veículos possui checagem dupla, uma com um dos detectores, e outra com uma nova técnica baseada no nível de confiança dos centroides de linhas verticais e horizontais. Caso algum veículo desapareça ou diminua seu nível de confiança de que a região ainda contém um veículo, apenas aquele veículo é re-detectado e o rastreamento continua. Caso o veículo não seja mais encontrado, uma nova detecção ocorre para sincronizar os rastreadores. Resultados mostram que isto melhora a precisão da detecção. Também melhora o tempo de processamento devido ao algoritmo de rastreamento processar a mais de 100 quadros por segundo, enquanto que o melhor detector, Haar-like com AdaBoost, processa a 17 quadros por segundo. Obteve-se resultados de até 99,7% de precisão na detecção com rastreamento, e tempo de processamento chegando a 70 quadros por segundo.

Palavras-chave: Alerta de colisão veicular. Detecção de veículos. Rastreamento de veículos. Haar-like. AdaBoost. KCF. Centroides. MB-LBP. HOG. SVM.

ABSTRACT

Thousands of people die and many others are seriously injured every day due to traffic accidents around the world. Most of these accidents are caused by distractions and recklessness of drivers. Many automobile companies invest in technologies, but there are still very few models of vehicles that do accident prevention, and those that do, their cost is inaccessible to most people. This work presents the development of a computer vision system using only a simple camera coupled to a vehicle, which notifies the driver through visual and audible alerts when there is a risk of collision of his vehicle with vehicles in front of him. A risk zone is established in front of the vehicle, and when a vehicle is within this region and begins to approach a certain distance of the vehicle with the system, the system alerts the driver, who will have time to avoid a frontal collision with the vehicle in front of him. Vehicle detection is done at frame intervals using the techniques Haar-like with AdaBoost, MB-LBP with AdaBoost and HOG with SVM. Within the ranges, each vehicle is tracked by a fast and accurate algorithm called KCF. A failure tolerance mechanism with helper trackers monitors and validates each vehicle. Vehicle validation has double check, one with one of the detectors, and another with a new technique based on the trust level of the centroids of vertical and horizontal lines. If a vehicle disappears or the level of confidence that the region still contains a vehicle decreases, only that vehicle is re-detected and tracking continues. If the vehicle is no longer found, a new detection occurs to synchronize the trackers. Results show that this improves detection accuracy. It also improves processing time because the tracking algorithm processes at over 100 frames per second, while the best detector, Haar-like with AdaBoost, processes at 17 frames per second. Results of up to 99.7% accuracy were obtained in the detection with tracking, and processing time reaching 70 frames per second.

Key-words: Vehicle collision warning. Vehicle detection. Vehicle tracking. Haar-like. AdaBoost. KCF. Centroids. MB-LBP. HOG. SVM.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACM	*Association for Computing Machinery*
AdaBoost	*Adaptive Boosting*
ANN	*Artificial Neural Network*
API	*Application Program Interface*
Caltech	*California Institute of Technology*
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CAS	*Collision Avoidance System*
CCLVH	Confiança dos Centroides das Linhas Verticais e Horizontais
CDF	*Cumulative Density Function*
C-HOG	*Circular HOG*
CPU	*Central Processing Unit*
DAS	*Driver Assistance System*
DFT	*Discrete Fourier Transform*
DOT	*Degree Of Trust*
DPVAT	Seguro de Danos Pessoais Causados por Veículos Automotores de Via Terrestre
DVR	*Digital Video Recorder*
FHD	*Full High Definition*
FN	Falso Negativo
FNFN	*Functional Neural Fuzzy Network*
FOC	*Focus Of Contraction*
FOE	*Focus Of Expansion*
FP	Falso Positivo
FPS	*Frames Per Second*

GH	Geração de Hipótese
GHz	*Giga Hertz*
HOG	*Histogram of Oriented Gradients*
IDE	*Integrated Development Environment*
IEEE	*Institute of Electrical and Electronics Engineers*
iROADS	*Intercity Roads and Adverse Driving Conditions*
ITS	*Intelligent Transportation System*
KCF	*Kernelized Correlation Filter*
KITI	*Karlsruhe Institute of Technology*
LBP	*Local Binary Pattern*
LNBD	*Last Number of Border*
MBA	Mecânismo de Busca Acadêmico
MB-LBP	*Multi-scale Block Local Binary Patterns*
MIT	*Massachusetts Institute of Technology*
MIT CBCL	*Center for Biological and Computational Learning at MIT*
MS	Mapeamento Sistemático
NBD	*Number of Border*
ODB II	*On Board Diagnostics version 2*
OMS	Organização Mundial da Saúde
PIB	Produto Interno Bruto
PRF	Polícia Rodoviária Federal
PV	População total de Veículos
RAM	*Random Access Memory*
RANSAC	*RANdom SAmple Consensus*
RBF	*Radial Base Function*
R-HOG	*Rectangular HOG*

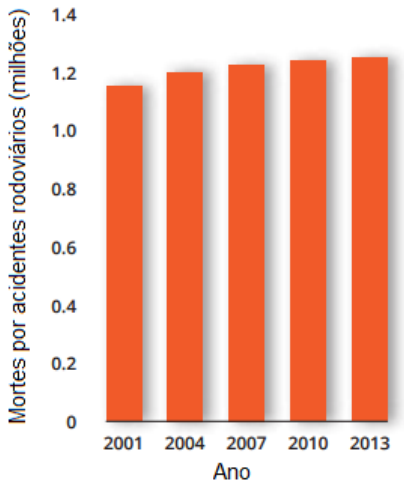
ROI	*Region Of Interest*
RS	Revisão Sistemática
SEMB-LBP	*Statistically Effective MB-LBP*
SURF	*Speeded Up Robust Features*
SVM	*Support Vector Machine*
TFN	Taxa de Falsos Negativo
TFP	Taxa de Falsos Positivo
TTC	*Time To Collision*
TVP	Taxa de Verdadeiros Positivo
UDESC	Universidade do Estado de Santa Catarina
UNN	*United Nations*
VH	Verificação de Hipótese
VP	Verdadeiro Positivo
VPN	*Virtual Private Network*
WHO	*World Health Organization*

SUMÁRIO

1 INTRODUÇÃO

Os acidentes de trânsito estão entre um dos maiores problemas de saúde pública e uma das principais causas de mortes, lesões e incapacidades em todo o mundo (UN, 2017). Conforme o “Relatório de situação global sobre a segurança rodoviária 2015”, o qual possui informações de 180 países, a cada ano mais de 1 milhão de pessoas morem em acidentes nas estradas (WHO, 2017). Apesar de entre 2010 e 2013 a população global ter aumentado 4% e o número de veículos registrados ter aumentado 16% no mesmo período, o gráfico da Figura 1 mostra estabilidade no número de mortes desde 2007 no contexto do aumento da frota motorizada. Porém,

Figura 1 – Número de mortes no trânsito em todo o mundo entre 2001 e 2013



Fonte: WHO (2017).

não percebe-se declínio no número de mortes, e este número em 2013 chega a 1,25 milhão de pessoas, o que é ainda um número muito expressivo.

É informado ainda por WHO (2017), que anualmente entre 20 e 50 milhões de pessoas sofrem lesões em acidentes nas estradas. Mais de 90% destes acidentes ocorrem em países de baixa e média renda, sendo que nesses países ficam menos da metade dos veículos do planeta.

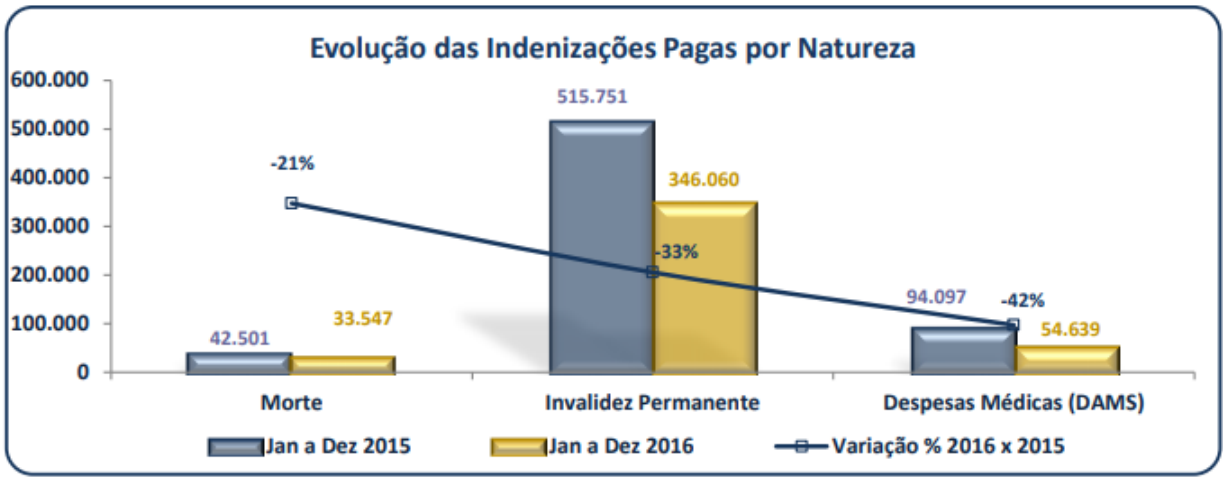
Conforme explicado em UN (2017), no mundo, os acidentes de trânsito estão entre as três principais causas de mortes para pessoas entre 5 e 44 anos de idade e é a principal causa de morte para jovens de 15 a 29 anos e a 9º causa de mortes para todos os grupos de idade, tendendo a se tornar a 7º causa de mortes até 2030. Isto deve-se ao crescente número de mortes em países de baixa e média renda, particular-

mente nas economias emergentes onde a urbanização e motorização acompanham o crescimento econômico. O excesso de velocidade, ultrapassagem perigosa e distração no volante estão entre as principais causas destes acidentes.

Isto afeta diretamente a economia e o desenvolvimento humano com uma estimativa global de perda anual em torno de 518 bilhões de dólares e custa para os governos de 1% a 3% do seu Produto Interno Bruto (PIB). Sem contar com os custos das famílias afetadas com despesas hospitalares, medicamentos, reabilitação, funeral, e ainda o extenso abalo emocional (UN, 2017).

No Brasil, baseado nos últimos anuários estatísticos divulgados por DPVAT (2017)[1], entre os anos de 2015 e 2016 mais de 1 milhão de pessoas foram vítimas de acidentes de trânsito. Deste número, foram mais de 76 mil mortes e quase 900 mil pessoas ficaram em estado de invalidez permanente, como mostra o gráfico da Figura 2. Mesmo que o gráfico mostre uma variação de queda para 2016 em relação a 2015, ainda assim, os valores são expressivos e preocupantes.

Figura 2 – Indenizações pagas pelo seguro DPVAT entre 2015 e 2016 por natureza



Fonte: DPVAT (2017).

Segundo dados da Polícia Rodoviária Federal (PRF)[2], em 2016 houveram mais de 19 mil acidentes por colisões traseiras nas rodovias federais. Destes acidentes, mais de 71% estavam relacionados à distrações do motorista e por não guardar uma distância segurança do seu veículo com os veículos à sua frente (dormindo, falta de atenção, não guardar distância de segurança e velocidade incompatível).

[1] Seguro de Danos Pessoais Causados por Veículos Automotores de Via Terrestre, mais conhecido como Seguro DPVAT.

[2] Arquivo em formato csv: <https://www1.prf.gov.br/arquivos/index.php/s/AhSKXYgrFtfXMK3>

Em WHO (2017) é apresentado que a velocidade dos veículos é um fator crítico de risco para acidentes de trânsito. À medida que a velocidade aumenta, também aumenta a probabilidade de um acidente. Se um acidente de trânsito acontecer, o risco de morte ou ferimento grave é maior se estiver em alta velocidade.

Outro ponto de atenção relatado em WHO (2017) é o número crescente de acidentes no trânsito devido a distração do motorista ao dirigir. Existem as distrações internas ao veículo (exemplo: fumar, comer, beber, mexer no som) e as distrações externas como as distrações visuais (exemplo: outdoors, e outras propagandas na estrada). Porém uma das distrações que mais tem crescido é a por uso de dispositivos tecnológicos, como manuseio de telefone celular enquanto dirige. O uso de telefone celular cria vários tipos de distrações como: visual, auditiva, manual e cognitiva. A Figura 3 mostra uma situação de uso de celular enquanto se está dirigindo.

Figura 3 – Manusear o celular enquanto dirige é um dos principais tipos de distrações

Fonte: Quatro Rodas (2013).

Resultados relatados em WHO (2017) mostram que distrações causadas por falar ao celular enquanto dirige podem prejudicar o desempenho da condução de várias maneiras, tais como: aumento do tempo de reação (notadamente no tempo de reação para frenagem), diminuição na capacidade de manter-se na pista correta e dificuldade de perceber distância curta entre os veículos. Mostra ainda que em média, um motorista dirigindo falando ao celular tem quatro vezes mais chances de ter um acidente do que um motorista que não esteja falando.

O relatório National Traffic Law Center (2016) descreve que um veículo andando a uma velocidade de aproximadamente 88 KM/h, anda cerca de 24 metros por segundo. Enviar ou ler uma mensagem de texto no celular, tira a atenção do motorista da estrada em média 4,6 segundos, o que equivale a cerca de 112 metros. Ou seja, o carro percorre cerca de um campo de futebol como se estivesse sem motorista.

No Brasil, um estudo feito pela revista Quatro Rodas (2013) mostra alguns dados (veja Tabela 1) da distância que um veículo percorre sem a atenção do motorista enquanto ele está distraído em alguns dos tipos mais comuns de distrações. A Tabela 1 mostra que a distância média percorrida pelo veículo sem a atenção do motorista em 60 KM/h e 100 KM/h é em torno de 30 e 50 metros respectivamente.

Tabela 1 – Tipos de distrações do motorista e distâncias percorridas pelo veículo sem sua atenção

Tipo de Distração	Velocidades (KM/h)	Tempo da distração (segundos)	Distância percorrida (metros)
Ajustar estação do rádio	60/100	1,5	25/42
Ler mensagem de texto	60/100	2	34/56
Escrever mensagem de texto	60/100	2,5	42/69
Buscar objetos no porta-luvas	60/100	1,5	25/42
Digitar números para ligação	60/100	2	34/56
Procurar endereço	40	3	36

Fonte: Quatro Rodas (2013)

Por fim, WHO (2017) relata ainda que tornar os carros mais seguros é um componente crítico para salvar vidas nas estradas. Enquanto os veículos em países de renda alta possuem cada vez mais tecnologias para segurança, quase 75% dos países ao redor do mundo, principalmente os países de média e baixa renda, não cumprem nem aos critérios internacionais mínimos em relação a segurança dos veículos.

1.1 PROBLEMA

O problema que se apresenta é como avisar antecipadamente ao motorista de um veículo sobre o risco dele se envolver em um acidente, como uma colisão com outro veículo, utilizando-se de tecnologias computacionais. Como discutido anteriormente, alguns tipos de situações que aumentam as chances de ocorrerem acidentes são: não manter distância segura entre os veículos e distrações do motorista. A probabilidade de ocorrer uma colisão pode aumentar consideravelmente, caso haja combinações destas situações, como por exemplo, dirigir em alta velocidade falando ao celular.

O estudo de WHO (2017), indica que tornar os veículos mais seguros, através de Sistemas de Prevenção de Colisão (*Collision Avoidance System* - CAS), é uma forma de prevenir e reduzir acidentes. Na literatura pesquisada sobre o assunto, apenas 13% dos estudos abordam este tipo de sistema. Isto indica uma carência de

sistemas computacionais que auxiliem o motorista na prevenção de acidentes, como as colisões.

Conforme explicado por Mukhtar, Xia e Tang (2015), um sistema CAS deveria notificar o motorista sobre o número de veículos nas proximidades, suas distâncias e velocidades relativas. Para extrair estas informações, o sistema pode usar diferentes sensores (radar, laser e câmera) para adquirir os dados do trafego na estrada e em seguida fazer a detecção e classificação dos veículos. Uma vez que uma colisão é prevista, este tipo de sistema pode tanto emitir um alerta ao motorista como frear o veículo automaticamente, ou executar ambas as ações.

Algumas questões que podem influenciar no desenvolvimento de sistemas CAS viáveis de serem aplicados em um ambiente real foram elencadas por Sivaraman e Trivedi (2013b), Mukhtar, Xia e Tang (2015) e Sreevishakh e Dhanure (2015), sendo elas:

a) dificuldade em encontrar sensores de qualidade com baixo custo;
b) dificuldade para acoplamento dos equipamentos aos veículos;
c) requerem altos recursos computacionais para processamento dos dados;
d) necessidade de execução rápida para poder ser empregado em um ambiente real;
e) sensores com limitações de distância e influenciados por condições climáticas e iluminação;
f) alguns sensores possuem dependência da infraestrutura das estradas.

No trabalho proposto, conforme recomendação da maioria dos trabalhos pesquisados, optou-se pela utilização de sensor óptico com câmera monocular, devido a ser um sensor encontrado facilmente no mercado com custo acessível, de fácil instalação nos veículos, sofre pouca influência da infraestrutura da estrada, não possuir limites de distância para extrair informações. Porém pode sofrer influências com diferenças de luminosidade e exigir alto consumo de recursos computacionais. (SREEVISHAKH; DHANURE, 2015).

1.2 OBJETIVOS

1.2.1 Objetivo Geral

O objetivo é desenvolver um sistema que detecte através de visão computacional os veículos à sua frente, e emita alertas sonoros e visuais quando identificar risco de colisão com os veículos detectados. Espera-se que com estes alertas, o motorista do veículo portando o sistema perceba a situação de risco e possa em tempo hábil tomar decisões e conseguir evitar acidentes por colisão.

1.2.2 Objetivos Específicos

Como objetivos específicos, tem-se:

a) investigar técnicas de visão computacional que envolvam a temática para prevenção de colisão veicular;

b) desenvolver um novo sistema para prevenção de colisão veicular utilizando visão computacional;

c) verificar se o rastreamento de veículos pode auxiliar na melhoria de desempenho, visto que a literatura não aborda o assunto de forma clara e objetiva;

d) avaliar o desempenho do sistema quanto a precisão na detecção dos veículos, tempo de processamento e emissão dos alertas.

1.3 ESCOPO

O escopo deste trabalho é propor e desenvolver um sistema que possa ser acoplado a um veículo e que ele seja capaz de emitir alertas ao motorista sempre que o seu veículo estiver se aproximando a uma determinada distância dos veículos à sua frente.

Para detectar os veículos, serão processados quadros de imagens adquiridos por uma câmera monocular, utilizando técnicas de visão computacional com extratores de características e classificadores. Podem ser utilizadas mais de uma técnica para poder evidenciar na prática se existe diferenças significativas no desempenho dependendo das técnicas empregadas.

Este trabalho não utilizará hardware específico para sua execução, poderá ser executado em qualquer computador (exemplo: um notebook). O programa executável do sistema gerado executará apenas em sistema operacional *Windows*.

A finalidade do sistema é para utilização somente durante o dia, podendo ser com tempo bom ou tempo chuvoso, desde que haja boa luminosidade.

A finalidade principal do sistema é detectar a parte traseira dos veículos à frente do veículo portando o sistema proposto.

As notificações com emissão de som estão limitadas a sons disponibilizados na internet e gratuitos. Não será avaliada a qualidade do som, mas sim se o alerta foi emitido atendendo aos critérios que forem estabelecidos para serem disparados.

A distância calculada pelo sistema entre o veículo portando o sistema e os demais veículos é apenas uma estimativa utilizada como um dos parâmetros para formular a heurística de risco de colisão.

Para efetuar os testes de detecção e rastreamento de veículos, e emissão de alertas do sistema, são utilizadas bases de imagens e vídeos públicos relatados nas bibliografias pesquisadas e disponibilizados na internet. Também são utilizados vídeos produzidos pelo próprio autor, portanto não há nenhuma garantia de que os testes efetuados com o sistema contemplem todos os cenários possíveis de riscos de colisão.

1.4 METODOLOGIA

Foram feitas pesquisas bibliográficas para as etapas inicias e aprofundamento do tema da pesquisa para conhecer o estado da arte dos sistemas para detecção de veículos. Verificou-se que há uma grande variedade de literatura sobre o tema. Por isto, optou-se por fazer um estudo mais aprofundado no assunto através de um mapeamento sistemático.

Dentre os dados extraídos do mapeamento sistemático, percebeu-se uma lacuna (*gap*) de sistemas de prevenção de colisão (CAS). Com isto foi levantado as principais técnicas e metodologias para serem utilizadas no desenvolvimento de um sistema voltado para este tipo de problema, justificado pela importância e gravidade já discutidas anteriormente na Seção 1 de introdução.

O sistema foi construído e testado em etapas. Iniciou-se com a detecção dos veículos, seguiu-se com a verificação de hipótese e rastreamento, após foi feita a estimativa da distância e por fim foi desenvolvida a heurística para estimativa de risco de colisão e a emissão dos alertas.

Para a detecção, rastreamento, e estimativa de distância entre os veículos, foram feitas provas de conceito (POC), principalmente com relação aos parâmetros de treinamento dos classificadores, pois alguns levaram semanas para treinar.

Os classificadores foram treinados e testados com objetos veículos marcados a partir de imagens obtidas de bases de imagens e vídeos públicos, relatados nos trabalhos encontrados na revisão bibliográfica e que estavam disponíveis na internet.

1.5 ESTRUTURA

A fim de apresentar a pesquisa realizada referente ao sistema desenvolvido, o texto foi dividido da seguinte forma: o segundo capítulo apresenta a fundamentação teórica de conteúdos envolvidos com a pesquisa abordando o reconhecimento de padrões, o processamento de imagens, a segmentação de regiões de interesse, métodos de extração de características, classificadores lineares, detecção e rastreamento de objetos com enfase em veículos. O terceiro capítulo apresenta os resultados do mapeamento sistemático realizado e introduz os trabalhos correlatos. No quarto capítulo

encontra-se o núcleo do trabalho, onde são apresentados três pontos importantes: Um novo método para verificação de hipótese, uma técnica simples para a estimativa de distância entre os veículos e o fluxo de funcionamento do sistema. No quinto capítulo são discutidos os testes e os resultados obtidos a partir dos experimentos realizados. Finalmente, no sexto capítulo são apresentadas as contribuições e considerações finais deste trabalho, seguido das referências bibliográficas.

2 FUNDAMENTAÇÃO

Neste capítulo serão apresentados alguns conceitos básicos sobre as principais técnicas utilizadas nos trabalhos relacionados e técnicas utilizadas no desenvolvimento deste trabalho. Iniciando-se com métodos para segmentação e detecção de bordas, em seguida reconhecimento de padrões, classificadores e por fim técnicas de detecção.

2.1 SEGMENTAÇÃO E DETECÇÃO DE BORDAS

Nesta seção serão apresentadas as técnicas utilizadas para segmentação e detecção de bordas, linhas verticais e horizontais. Inicia-se com o detector de bordas de Canny, em seguida é exposta a transformada de Hough, para detecção de formas como as linhas verticais e horizontais e por fim um seguidor de borda para extrair as coordenadas dos segmentos que formam cada uma das bordas.

2.1.1 Detector de Bordas de Canny

Esta técnica é utilizada na Seção 4.1.1 para fazer a segmentação dos contornos de uma região candidata a veículo, permitindo efetuar a detecção das linhas verticais e horizontais, as quais serão utilizadas em uma nova técnica para a Verificação de Hipótese (VH) de regiões candidatas a veículos.

O detector de bordas de Canny, como o próprio nome já diz, é uma técnica para detecção de bordas desenvolvida por Canny (1986). A detecção de bordas é uma etapa importante para a segmentação de imagens, detecção de objetos, detecção de linhas e extração de características. A técnica é baseada em três objetivos principais:

a) baixa taxa de erro: todas as bordas devem ser encontradas e o mais próximo possível das bordas verdadeiras;

b) os pontos de borda devem estar bem localizados: a distância entre um ponto marcado pelo detector como sendo de uma borda e o centro da borda verdadeira deve ser mínimo;

c) resposta de um único ponto de borda: o detector deve retornar apenas um ponto para cada ponto de borda verdadeiro. Ou seja, não deve identificar múltiplos pixels de borda onde apenas um ponto de borda existe.

Segundo Gonzales e Woods (2011, p. 474), é difícil encontrar uma solução fechada que atenda completamente os três objetivos acima. Porém é possível obter uma boa aproximação para um detector ótimo de bordas de degrau 1-D através da

primeira derivada de uma gaussiana, conforme Equação 2.1:

$$\frac{d}{dx}e^{-\frac{x^2}{2\sigma^2}} = \frac{-x}{\sigma^2}e^{-\frac{x^2}{2\sigma^2}}. \tag{2.1}$$

Para generalizar este resultado para 2-D, sendo que a direção da normal é desconhecida previamente, e para evitar a aplicação do detector de bordas 1-D em todas as direções possíveis, pode-se primeiro suavizar a imagem com uma função gaussiana circular 2-D, calcular o gradiente do resultado e, usar a magnitude do gradiente e a direção para estimar a intensidade da borda e a direção de cada ponto.

Sendo $f(x, y)$ a imagem de entrada e $G(x, y)$ a função gaussiana pela Equação 2.2:

$$G(x, y) = e^{\frac{x^2+y^2}{2\sigma^2}} \tag{2.2}$$

Uma imagem $f(x, y)$ é suavizada por convolução [1] de G e f pela Equação 2.3:

$$f_s(x, y) = G(x, y)\bigstar f(x, y). \tag{2.3}$$

Esta operação é seguida pelo cálculo da magnitude pela Equação 2.4

$$M(x, y) = \sqrt{g_x^2 + g_y^2} \tag{2.4}$$

e da direção (ângulo) do gradiente pela Equação

$$\alpha(x, y) = tg^{-1}\left[\frac{g_y}{g_x}\right] \tag{2.5}$$

com $g_x = \partial f_s/\partial x$ e $g_y = \partial f_s/\partial y$. Para obter g_x e g_y normalmente é utilizado o par de máscaras de Sobel como mostrado na Figura 4.

Figura 4 – Máscaras de filtragem de Sobel

−1	−2	−1	−1	0	1
0	0	0	−2	0	2
1	2	1	−1	0	1

Fonte: Gonzales e Woods (2011, p. 467).

[1] Segundo Gonzales e Woods (2011, p. 96), convolução representado pelo símbolo ★, serve para realizar filtragem espacial linear. É o processo de mover uma ou mais máscaras sobre uma imagem e calcular a soma dos produtos em cada posição onde a máscara se encontra sobre a imagem, sendo que o primeiro filtro é rotacionado em 180^o.

A Equação 2.2 é implementada usando uma máscara gaussiana de tamanho n x n, igual a imagem processada.

A utilização do gradiente $M(x, y)$ geralmente contém cristas largas em torno dos máximos locais, sendo que o próximo passo é afinar estas cristas utilizando um método de supressão dos não máximos, que especifica um número de orientações discretas da normal da borda (vetor gradiente).

Por exemplo, em uma região 3 x 3 podem ser definidas quatro direções para uma borda que passa pelo ponto central da região: horizontal, vertical, $+45^o$ e -45^o. A direção da borda é determinada a partir do vetor normal à borda, que é obtido a partir dos valores da imagem usando a Equação 2.5.

Considerando que d_1, d_2, d_3 e d_4 denotam as quatro direções básicas acima respectivamente. Pode ser formulado o seguinte esquema de supressão de não máximos de uma região 3 x 3 centrada em todos os pontos (x, y) de $\alpha(x, y)$:

a) encontre a direção d_k que está mais perto de $\alpha(x, y)$;

b) se o valor de $M(x, y)$ for inferior a pelo menos um dos seus vizinhos ao longo de d_k, deixe $g_N(x, y) = 0$ (supressão); caso contrário, deixe $g_N(x, y) = M(x, y)$. Onde $g_N(x, y)$ é a imagem com supressão de não máximos.

A etapa final é a limiarização de $g_N(x, y)$ para redução dos falsos pontos da borda. Para isto, o algoritmo de Canny utiliza a limiarização por histerese, que usa dois limiares: um limiar baixo, T_L, e um limiar alto, T_H. Canny sugeriu que a razão do limiar alto para o baixo seja de dois ou três para um.

A operação de limiarização pode ser visualizada como a criação de duas imagens adicionais, pela Equação 2.6:

$$g_{NH}(x, y) = g_N(x, y) \geq T_H, \tag{2.6}$$

e pela Equação 2.7:

$$g_{NL}(x, y) = g_N(x, y) \geq T_L \tag{2.7}$$

sendo que inicialmente, tanto $g_{NH}(x, y)$ como $g_{NL}(x, y)$ são definidas como zero. Elimina-se de $g_{NL}(x, y)$ todos os pixels diferentes de zero contidos em $g_{NH}(x, y)$ pela Equação 2.8:

$$g_{NL}(x, y) = g_{NL}(x, y) - g_{NH}(x, y). \tag{2.8}$$

Os pixels diferentes de zero em $g_{NH}(x, y)$ e $g_{NL}(x, y)$ são vistos como pixels de bordas fortes e fracos respectivamente.

Após a limiarização, os pixels fortes em $g_{NH}(x, y)$ são marcados como pixels de borda válidos. Dependendo do valor de T_H, as bordas em $g_{NH}(x, y)$ podem apresentar falhas, sendo que as bordas mais longas são formadas pelo seguinte procedimento:

a) localizar o próximo pixel de borda, p, a ser revisado em $g_{NH}(x, y)$;

b) marcar como pixel de borda válido todos os pixels fracos em $g_{NL}(x, y)$ que estão conectados a p, (por exemplo, conectividade de 8);

c) se todos os pixels diferentes de zero em $g_{NH}(x, y)$ foram revisados, vá para o passo b), senão, vá para o passo a);

d) atribuir zero para todos os pixels de $g_{NL}(x, y)$ que não foram marcados como pixels de borda válidos.

Ao final deste processo são atribuídos para $g_{NH}(x, y)$ todos os pixels diferentes de zero contidos em $g_{NL}(x, y)$.

Na prática, pode-se aplicar a limiarização por histerese diretamente durante a supressão não máxima e a limiarização pode ser aplicada diretamente sobre $g_N(x, y)$, formando uma lista dos pixels fortes e dos pixels fracos conectados a eles. Sendo as imagens $g_{NH}(x, y)$ e $g_{NL}(x, y)$, usadas apenas para simplificar a explicação do algoritmo.

O algoritmo de detecção de bordas de Canny pode ser resumido nas seguintes etapas:

a) suavizar a imagem de entrada com um filtro gaussiano;

b) calcular a magnitude do gradiente e os ângulos das imagens;

c) aplicar a supressão não máxima na imagem da magnitude do gradiente; e

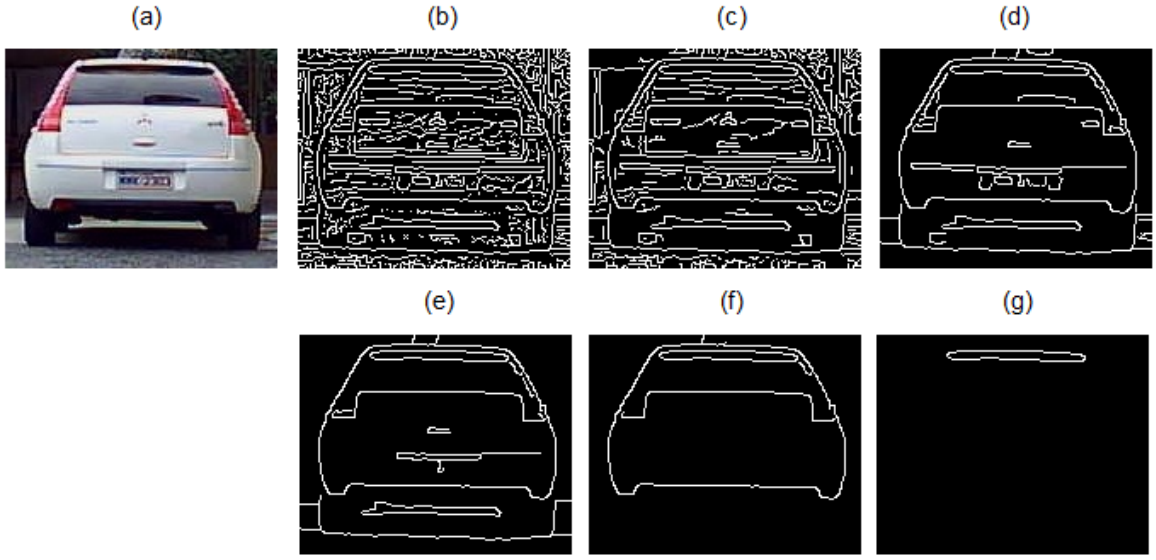
d) aplicar a limiarização dupla e a análise de conectividade para detectar e conectar as bordas.

Após a supressão não máxima ainda podem permanecer bordas com espessura maiores do que 1 pixel, para obter bordas de 1 pixel, após a etapa d), normalmente é aplicado um algoritmo de afinamento de bordas.

O resultado final obtido é uma imagem binária com fortes bordas finas. A Figura 5 exibe vários exemplos empregando vários valores diferentes para o limiar baixo, T_L e para o limiar alto, T_H. A razão utilizada para o limiar alto foi de três vezes o limiar baixo. Em cada amostra a imagem original foi suavizada primeiro com um filtro gaussiano com tamanho de *kernel* 5 x 5.

Analisando os resultados da Figura 5, é possível perceber claramente que com limiares muito baixos, são apresentadas muitas bordas falsas positivas e com limiares muito altos, várias bordas verdadeiras são eliminadas. Um limiar baixo entre 50 e 100, e um limiar alto de razão $T_L * 3$, mostra uma detecção de bordas equilibrada.

Figura 5 – Exemplos de detecção de bordas de Canny. (a) Imagem original. (b) $T_L = 1$. (c) $T_L = 10$. (d) $T_L = 50$. (e) $T_L = 100$. (f) $T_L = 170$. (g) $T_L = 195$.



Fonte: Elaborado pelo autor, 2017.

2.1.2 Transformada de Hough

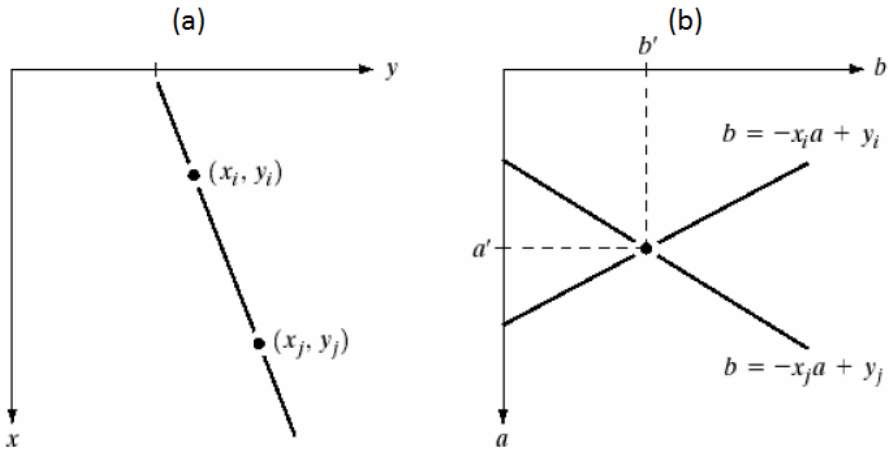
A transformada de Hough, conforme explica Gonzales e Woods (2011, p. 483), é uma transformada global utilizada para detectar formas como linhas verticais e horizontais. Alguns trabalhos relacionados como, Tang et al. (2015) e Rezaei, Terauchi e Klette (2015) utilizam esta técnica para detecção de linhas.

Seja um ponto (x_i, y_i) pertencente ao plano xy e a equação de uma reta na forma inclinação-interseção: $y_i = ax_i + b$. Infinitas retas podem passar pelo ponto (x_i, y_i), mas todas respeitando a equação $y_i = ax_i + b$ para diferentes valores de a e b. Entretanto, escrever esta equação como $b = -x_i a + y_i$ e considerando o plano ab (também chamado de espaço de parâmetros) produz a equação de uma única reta para um par fixo (x_i, y_i).

Um segundo ponto (x_j, y_j) também tem uma reta no espaço de parâmetros associada a ele e a menos que sejam paralelas, esta reta cruza a reta associada à (x_i, y_i) em algum ponto (a', b'), em que a' é a inclinação e b' é a interseção da reta contendo tanto (x_i, y_i) quanto (x_j, y_j) no plano xy. As Figuras 6(a) e (b) mostram um exemplo destes conceitos.

Em princípio, as retas do espaço de parâmetros que corresponda a todos os pontos (x_k, y_k) no plano xy podem ser traçadas e as retas principais neste plano poderiam ser determinadas identificando os pontos nos espaço de parâmetro nos quais uma grande quantidade de retas do espaço de parâmetros se intercepta.

Figura 6 – (a) Plano xy. (b) Espaço de parâmetros



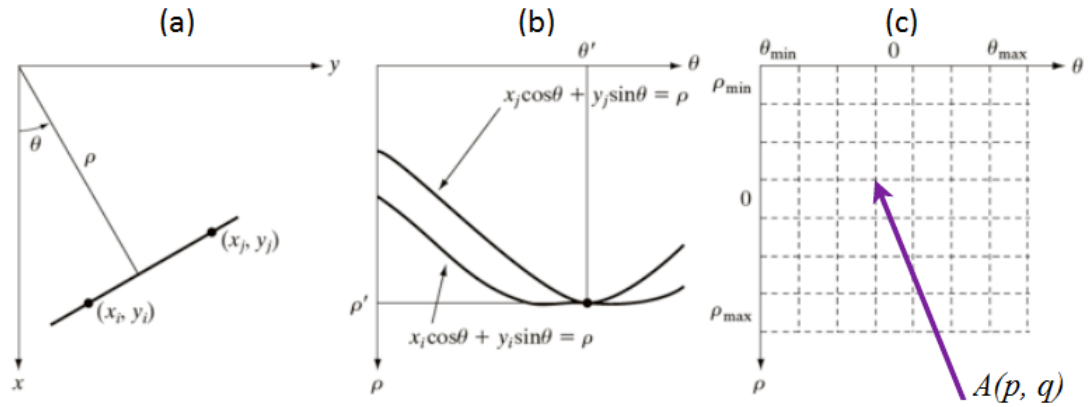
Fonte: Gonzales e Woods (2011).

Porém, a inclinação de uma reta se aproxima do infinito conforme se aproxima da direção (inclinação) vertical. Uma forma de solucionar esta questão é utilizar a representação normal de uma reta em coordenadas polares conforme a Equação 2.9:

$$x.\cos\theta + y.\text{sen}\theta = \rho \tag{2.9}$$

A Figura 7(a) ilustra a interpretação geométrica dos parâmetros ρ e θ. Uma reta horizontal tem $\theta = 0^o$, com ρ igual à interseção positiva de x. Uma reta vertical tem $\theta = 90^o$, com ρ igual à interseção positiva de y, ou $\theta = -90^o$, com ρ igual à interseção negativa de y. Cada curva senoidal da Figura 7(b) representa a família de retas que passam por um determinado ponto (x_k, y_k) no plano xy. O ponto de interseção (ρ', θ') na Figura 7(b) corresponde a reta que passa tanto por (x_i, y_i) quanto por (x_j, y_j) na Figura 7(a).

Figura 7 – (a) (ρ, θ) Parametrização da reta no plano xy. (b) Curvas senoidais no plano $\rho\theta$. (c) Divisão do plano $\rho\theta$ em células acumuladoras no formato $A(p, q)$



Fonte: Gonzales e Woods (2011).

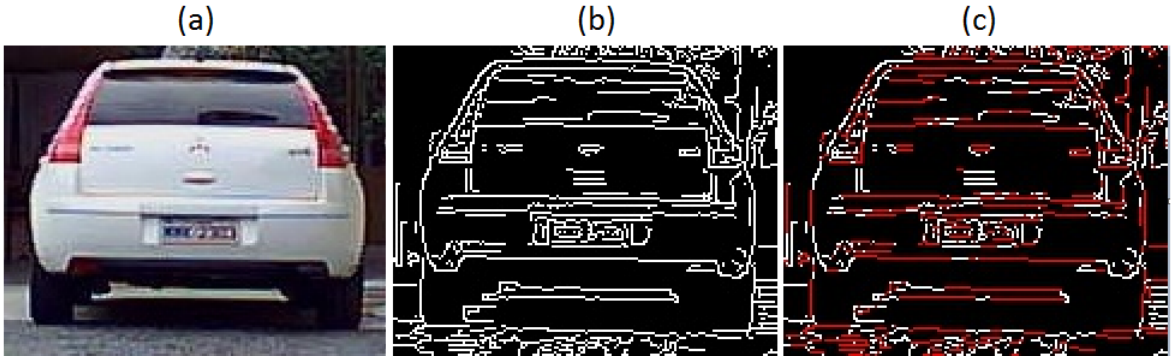
A transformada de Hough é atrativa computacionalmente devido a subdivisão do espaço de parâmetros $\rho\theta$ nas chamadas células acumuladoras, conforme Figura 7(c) ilustra, sendo que (ρ_{min}, ρ_{max}) e $(\theta_{min}, \theta_{max})$ são os esperados intervalos de valores dos parâmetros: $-90^o \leq \theta \leq 90^o$ e $-D \leq \rho \leq D$, em que D é as distância máxima entre os cantos opostos de uma imagem.

A célula nas coordenadas (i, j), como um valor de acumulador $A(i, j)$, corresponde ao quadrado associado às coordenadas do espaço de parâmetros (ρ_i, θ_j). Inicialmente estas células são definidas com o valor zero. Então, cada ponto (x_k, y_k) do plano xy, deixamos que θ seja igual a cada valor de subdivisão permitido no eixo θ e calculamos o ρ correspondente utilizando a equação $\rho = x_k.cos\theta + y_k.sen\theta$.

Os valores ρ resultantes são arredondados para o valor de célula permitida mais próximo no eixo ρ. Se a escolha de θ_p resultar em ρ_q, então tem-se $A(p, q) = A(p, q)+1$. No final deste processo, um valor de P em $A(i, j)$ significa que os pontos P no plano xy encontram-se na reta $x.cos\theta_j + y.sen\theta_j = \rho_i$. O número de subdivisões no plano $\rho\theta$ determina a precisão da colinearidade destes pontos.

A transformada de Hough é aplicada sobre uma imagem de bordas binária, a qual pode ser obtida através do detector de bordas de Canny ou dos operadores de Sobel (ou outra técnica equivalente). A Figura 8 (a) até (c) mostra um exemplo da aplicação da transformada de Hough sobre uma imagem de veículo com as bordas binária obtida com o detector de bordas de Canny. As linhas em vermelho sobre as

Figura 8 – (a) Imagem original. (b) Imagem binária. (c) Resultado da transformada de Hough



Fonte: Elaborado pelo autor, 2017.

linhas brancas da Figura 8(c) são as linhas detectadas pela transformada de Hough.

2.1.3 Seguidor de Borda

As técnicas de segmentação, como Canny e Hough, produzem dados primários em forma de pixels ao longo de uma fronteira (sua borda), porém não dizem nada

por si próprias sobre as informações destas fronteiras como entidades (BRADSKI; KAEHLER, 2011, p. 222), são apenas áreas muito finas de cor branca.

Aqui, o interesse maior é extrair destes dados primários suas características externas (sua fronteira ou contorno). Isto permite extrair características como sua posição, extensão e a orientação da linha reta que une seus pontos extremos. Esta técnica será utilizada na Seção 4.1.1 do Capítulo 4, como entrada para extração das linhas verticais e horizontais de veículos, em uma nova técnica proposta para verificação de hipótese (VH).

Um seguidor de bordas eficiente foi desenvolvido por Suzuki e Abe (1985). Esta técnica produz um conjunto de listas (ou árvore) de pontos sequências que representam as coordenadas das bordas de uma imagem binária.

Um seguidor de borda deriva de uma sequência de coordenadas, também chamados de códigos de cadeia, entre um componente-1 (conectado de 1-pixels) e um componente-0, conectado de 0-pixels, (fundo ou buraco). Assim, as informações são extraídas por dois tipos de bordas: borda externa e borda de buraco. A estrutura da topologia de uma borda pode ser determinada por existir uma relação entre uma borda externa e o componente-1, e entre uma borda de buraco e o componente-0.

A linha do topo e da base, e a coluna mais a esquerda e a coluna mais a direita da imagem compõe o seu quadro. Pixels com valor 0 e 1 são chamados de 0-pixel e 1-pixel, respectivamente. Assim, assume-se que 0-pixels preenchem o quadro da imagem binária. Também assume-se que pode ser atribuído qualquer valor inteiro para um pixel durante o processo do seguidor de borda (isto pode fazer com que a imagem de saída seja diferente da imagem de entrada).

O pixel localizado na linha ith e na coluna jth é representado pelo número da linha e coluna (i, j). A linha de número i é incrementada do topo para baixo e o número da coluna j da esquerda para a direita.

Um componente-1 e um componente-0 são os componentes conectados de 1-pixels e de 0-pixels, respectivamente. Se um componente-0, S, contém o quadro da imagem, S é chamado de fundo, caso contrário é um buraco. A fim de evitar contradições topológicas, 0-pixels devem ser considerados conectividade-4(-8) se 1-pixel são tratados como conectividade-8(-4).

Em um caso de conectividade-8(-4), um 1-pixel (i, j) tendo um 0-pixel (p, q) em sua vizinhança de 4 ou de 8 é chamado de ponto de borda. Também é descrito como um ponto de borda entre componente-1 S_1 e um componente-0 S_2, se (i, j) é um membro de S_1 e (p, q) é um membro de S_2.

Para dois componentes conectados S_1 e S_2 em uma imagem binária, se tiver um pixel pertencendo a S_2 para qualquer conectividade-4 de um pixel em S_1 para um pixel no quadro, então S_2 circunda S_1. Se S_2 circunda S_1 e existe um ponto de borda entre eles, então S_2 circunda S_1 diretamente.

Uma borda externa é definida como um conjunto de pontos de borda entre um componente-1 arbitrário e o componente-0 que o circunda diretamente. Semelhantemente, um conjunto de pontos de borda entre um buraco e um componente-1 que o circunda diretamente é definida como uma borda de buraco.

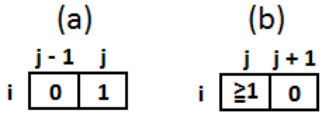
Uma borda pai de uma borda externa entre um componente-1 S_1 e o componente-0 S_2, que circunda S_1 diretamente, é definido como: uma borda de buraco entre S_2 e um componente-1 o qual circunda S_2 diretamente, se S_2 é um buraco; e o quadro da imagem, se S_2 é o fundo.

Dadas duas bordas B_0 e B_n de uma imagem binária, B_n circunda B_0 se existir uma sequência de borda B_0, B_1, ..., B_n, tal que B_k é a borda pai de B_{k-1} para todo k $(1 \leq k \leq n)$.

Esta técnica marca (rotula) unicamente cada borda encontrada, e consegue obter a borda pai do seguidor de borda corrente. Isto permite obter informações de relacionamento entre as bordas conectadas ao redor.

O algoritmo varre uma imagem binária, até encontrar um pixel (i, j) que satisfaz a condição de ponto inicial do seguidor de borda de uma borda externa, Figura 9(a), ou de uma borda de buraco, Figura 9(b). Caso satisfaça ambas as condições, deve ser considerado como o ponto de partida da borda externa. Associa um número sequencial único para a nova borda, denominado de NBD.

Figura 9 – As condições do ponto de partida para um pixel ser seguidor de borda. (a) Para uma borda externa. (b) Para um borda de buraco



Fonte: Suzuki e Abe (1985).

É mantido o número sequencial da última borda processada, denominado de LNBD (externa ou de buraco). Este número indica ou a borda pai ou uma borda que compartilha sua borda pai, com a borda sendo processada. Caso o tipo das duas bordas (borda LNBD e a borda corrente) sejam iguais, o pai da borda corrente será a borda de LNBD, caso contrário será a borda pai da borda LNBD.

Segue a borda encontrada a partir do ponto de partida, marcando os pixels na borda. O esquema do seguidor de borda usado por Suzuki e Abe (1985) é baseado no esquema clássico de Rosenfeld (1970), e Yokoi, Toriwaki e Fukumura (1975), com a diferença de ter a seguinte política de marcação:

a) se o seguidor de borda atual está entre o componente-0 o qual contém o pixel $(p, q+1)$ e o componente-1 o qual contém o pixel (p, q), o valor do pixel (p, q) deve ser mudado para -NBD;

b) caso contrário, atribua o valor NBD para o pixel (p, q), a menos que (p, q) esteja em uma borda já seguida.

A Figura 10 mostra um exemplo conceitual da técnica. Os passos vão da letra (a) até (e).

2.2 PADRÕES E CLASSES DE PADRÕES

Nesta seção será discutida a descrição de objetos através de vetores de características ou descritores, a sua categorização em classes de padrões e uma introdução básica sobre o reconhecimento de objetos com base no método de decisão teórica.

2.2.1 Padrões

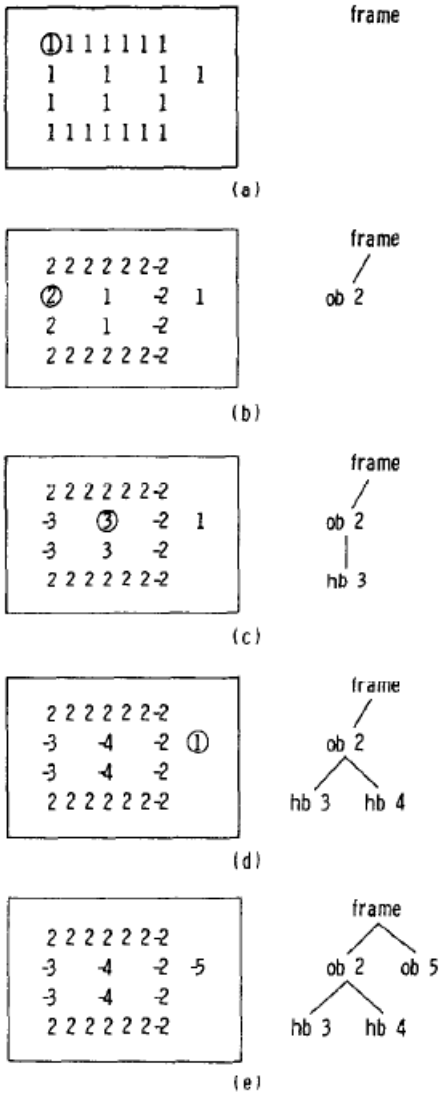
Um padrão é um arranjo de descritores, também chamado de características. Conforme pode ser constatado em Gonzales e Woods (2011, p. 568), Theodoridis e Koutroumbas (2009, p. 5) e Duda, Hart e Stork (2000, p. 7) o arranjo de padrões comumente utilizado é o vetor de características também chamados em inglês de *feature vector* ou *pattern vector*. Os vetores de características são descritores quantitativos, como comprimento, área e textura. Normalmente um vetor de características é representado pela letra **x** minúscula em negrito e assume a forma da Equação 2.10

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{2.10}$$

sendo que cada componente x_i representa a i-ésima característica, e n é o número total de características associados ao padrão. Ou seja, é uma matriz de n linhas por 1 coluna. Outra forma equivalente de representar um vetor de características apresentado na Equação 2.10 é **x**$= (x_1, x_2, \ldots, x_n)^T$, onde T indica transposição.

A natureza dos componentes do vetor de características **x** depende da metodologia utilizada para descrever o padrão físico (por exemplo, um objeto segmentado em

Figura 10 – Uma ilustração do processo do seguidor de bordas. As figuras da esquerda mostram os valores dos pixels e da direita as estruturas extraídas entre as bordas (ob: bordas externas, hb: borda de buraco). Os pixels circulados são os pontos inicias do seguidor de bordas.



Fonte: Suzuki e Abe (1985).

uma imagem como um veículo ou uma pessoa). Em uma imagem em níveis de cinza de 32 x 32 pixels e, tomando como metodologia de descrição do padrão de características o nível de intensidade de cada pixel, teríamos um vetor de características de 1024 componentes, ou seja, um vetor no espaço euclidiano de 1024 dimensões.

2.2.2 Classes de Padrões

Uma classe de padrão conforme explicado por Gonzales e Woods (2011, p. 568) é uma família de padrões que compartilham algumas propriedades comuns. Theodoridis e Koutroumbas (2009, p. 5) descrevem que objetos podem ser classificados em categorias ou classes pela descrição de seus padrões.

As classes de padrões são indicadas por $\omega_1, \omega_2, ..., \omega_W$, onde W é o número de classes. Técnicas de reconhecimento de padrões por máquina tem por objetivo associar vetores de características desconhecidos nas suas respectivas classes de forma mais automatizada e com menor erro possível.

2.2.3 Reconhecimento de Padrões

Gonzales e Woods (2011, p. 570) explicam que o reconhecimento de padrões baseado nas abordagens da decisão teórica utilizam uma função de decisão, também conhecida como função discriminante.

Para um padrão representado pelo vetor de características **x**$= (x_1, x_2, \ldots, x_n)^T$ n-dimensional e para W classes de padrões $\omega_1, \omega 2, ..., \omega_W$, o problema básico em reconhecimento de padrões é encontrar W funções de decisão $d_1(\mathbf{x}), d_2(\mathbf{x}), ..., d_W(\mathbf{x})$ com a propriedade que, se o padrão **x** pertence à classe ω_i deve atender a Equação 2.11

$$d_i(\mathbf{x}) > d_j(\mathbf{x}) \quad \text{para} \quad j = 1, 2, ..., W \quad \text{e} \quad j \neq i. \tag{2.11}$$

Ou seja, um padrão **x** de classe desconhecida pertence à i-ésima classe de padrões se a substituição de **x** em todas as funções de decisão fizerem com que $d_i(\mathbf{x})$ tenha o maior valor numérico. Casos de empates são tratados arbitrariamente.

As classes ω_i e ω_j são separadas por uma fronteira de decisão onde os valores de **x** resultam em $d_i(\mathbf{x}) = d_j(\mathbf{x})$, ou de forma equivalente, pela Equação 2.12

$$d_i(\mathbf{x}) - d_j(\mathbf{x}) = 0. \tag{2.12}$$

Assim, a fronteira de decisão entre duas classes pode ser identificada pela função $d_{ij}(\mathbf{x}) = d_i(\mathbf{x}) - d_j(\mathbf{x}) = 0$. Logo, $d_{ij}(\mathbf{x}) > 0$ indica que o padrão **x** pertence a classe ω_i, e $d_{ij}(\mathbf{x}) < 0$ indica que o padrão **x** pertence a classe ω_j.

2.3 CLASSIFICADORES

A maioria dos classificadores na sua essência implementam as definições dadas nas Seções 2.2.2 e 2.2.3. Conforme explicam Duda, Hart e Stork (2000, p. 11), os classificadores são reconhecedores de padrões. Em suma, dado algum dado, um classificador de padrões procura encontrar a hipótese mais provável de um conjunto de hipóteses - "esta região é provavelmente um veículo".

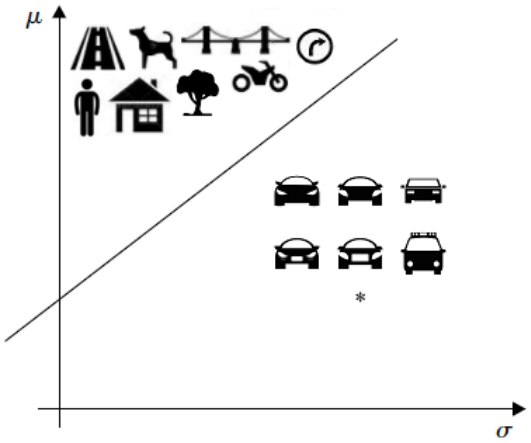
Um classificador de padrões difere do processamento de imagem. No processamento de imagem, a entrada é uma imagem e a saída é uma imagem, por vezes transformada, mas com as informações originais preservadas. Um classificador também difere do extrator de características, o qual perde informações da imagem original, porém preserva o suficiente para cumprir sua tarefa produzindo os valores das características.

Já um classificador, devido as regras de decisão, é um processo radical de redução de informações. Pode reduzir uma imagem de entrada representada por milhares de bits ou pixels, em poucos bits representado a categoria ou classe escolhida (um único bit no caso de classificar se uma região é um veículo ou não).

A seguir será apresentado um exemplo adaptado de Theodoridis e Koutroumbas (2009, p. 4). Supondo que temos duas classes, a classe A, que representa objetos não veículos, e a classe B, que representa somente objetos veículos. Supondo ainda que temos uma base de imagens que possuem vários objetos conhecidos que pertençam a classe A, e outros também conhecidos que pertençam a classe B.

O primeiro passo é identificar valores mensuráveis que fazem estas duas classes serem distintas entre si. A Figura 11 mostra a média das características extraídas (por exemplo, diâmetro e simetria dos buracos e das linhas verticais e horizontais dos objetos) versus o desvio padrão correspondente em torno da média. É possível perceber que os padrões da classe A tendem a se espalhar em um espaço diferente dos padrões da classe B. A linha reta parece ser uma boa candidata para separar os padrões das duas classes neste caso específico.

Figura 11 – Gráfico do valor da média versus desvio padrão para um número de imagens diferentes originadas da classe A (não veículos) e da classe B (veículos). Uma linha reta separa as duas classes



Fonte: Adaptado de Theodoridis e Koutroumbas (2009, p. 5). Imagens obtidas da internet.

Supondo que é apresentado um novo padrão (vetor de características) desconhecido e que não se sabe a qual das duas classe ele pertence, é calculada sua média e desvio padrão, analisando as mesmas características descritas acima. O padrão irá cair em um dos lados da linha reta e será possível saber se ele se parece mais com a classe A ou com a Classe B. Este é o caso do asterisco (*) da Figura 11.

A linha reta da Figura 11 é conhecida como linha de decisão. Ela constitui as regras de um classificador que divide o espaço de características da classe A e da classe B em regiões. Se um vetor de características **x** desconhecido cair na região da classe A, ele é classificado como sendo da classe A, caso contrário como classe B.

Caso a classificação não esteja correta, ocorreu um erro de classificação, também chamado de falso positivo. Para desenhar a linha reta da Figura 11 foi explorado o fato de já serem conhecidos os rótulos (classe A e B) para cada objeto da figura. Os padrões cuja classe verdadeira seja conhecida e que são usados para desenhar o classificador são chamados de padrões de treinamento.

Neste trabalho serão utilizados dois classificadores diferentes: AdaBoost e *Support Vector Machine* (SVM), ambos são lineares e binários, com funcionamento semelhante ao exemplo apresentado anteriormente para a classificação de veículos de não veículos. Foram escolhidos estes dois classificadores devido a serem os mais empregados na detecção de veículos, conforme pode ser visto no gráfico da Figura 37(b), onde foram encontrados 26 trabalhos utilizando AdaBoost e 24 trabalhos utilizando SVM.

2.3.1 AdaBoost

AdaBoost é uma técnica de aprendizado por máquina desenvolvida por Freund e Schapire (1995). Seu nome provêm do inglês *adaptive boosting* (impulso adaptativo). A técnica objetiva criar um classificador com alta precisão baseado na combinação de vários classificadores com precisão relativamente baixas (classificadores fracos). Estes classificadores fracos precisam ter sua taxa de erro de predição, no mínimo um pouco melhor do que um classificador com predição ao acaso.

Conforme explicado por Schapire e Freund (2012, p. 4), o algoritmo AdaBoost toma como entrada um conjunto de exemplos de treinamento $(x_1, y_1), ..., (x_m, y_m)$ onde cada x_i é uma instância do domínio X, e cada $y_i \in \{-1, +1\}$ é o rótulo ou classe associada a cada instância [2].

O algoritmo AdaBoost começa com um classificador base muito fraco, tão fraco que sozinho tem pouco a contribuir para a classificação. Entretanto, como regra, ele deve ser um pouco melhor do que predições ao acaso. A chamada do algoritmo é iterativa, baseada em várias rodadas, de forma que a cada nova iteração o classificador base aprenda algo novo sobre os dados e se torne cada vez mais forte, resultando em um novo classificador com mais conhecimento do que o classificador anterior. Para isto lhe são atribuídos, no seu conjunto de exemplos de entrada para treinamento, os exemplos classificados errados pelo classificador base antecessor.

[2] Uma extensão do AdaBoost com suporte a múltiplas classes é proposta por Schapire e Freund (2012)

Para escolher um conjunto de treinamento em cada rodada, AdaBoost mantém uma distribuição de pesos sobre os exemplos de treinamento. A distribuição usada na $t-$ésima rodada é D_t, e o peso atribuído para o exemplo i é denotado por $D_t(i)$. Este peso é uma medida que indica a importância da classificação correta do exemplo i na rodada corrente t.

Inicialmente todos os exemplos possuem o mesmo peso, mas a cada rodada, os exemplos classificados errados são penalizados com um aumento em seu peso. Isto quer dizer que exemplos difíceis de serem classificados corretamente, terão sucessivos incrementos em seus pesos, forçando o classificador base a manter sua atenção nestes exemplos. Por outro lado, os exemplos classificados corretamente tem seu peso decrementado a cada rodada. A Figura 12 apresenta o algoritmo AdaBoost. As linhas enumeradas na figura são para facilitar a didática na explicação das equações contidas no algoritmo.

A inicialização do algoritmo AdaBoost é feita na linha 1, onde recebe o conjunto de m exemplos de entrada para treinamento, cada exemplo com seu respectivo rótulo -1 ou +1 associado. Seguindo na linha 2, é calculado o peso inicial da distribuição D_1 de cada exemplo i, lembrando que cada exemplo é inicializado com o mesmo peso de distribuição $1/m$.

O algoritmo prossegue na linha 3, executando T iterações ou rodadas. O trabalho em cada rodada consiste em encontrar um classificador base $h_t : X \rightarrow \{-1, +1\}$ apropriado para a distribuição D_t conforme o que foi exposto anteriormente. A qualidade de um classificador base é medida pelo erro ponderado da distribuição D_t conforme a linha 7 do algoritmo. A expressão $\mathbf{Pr}_{i\sim D_t}[.]$ denota a probabilidade do i-ésimo exemplo ser selecionado aleatoriamente de acordo com a distribuição D_t. Assim, o erro ponderado ϵ_t, é a chance do classificador h_t classificar errado um exemplo aleatório se selecionado de acordo com a distribuição D_t. Basicamente, ϵ_t é a soma da distribuição dos exemplos classificados errados na rodada t.

Um classificador fraco tenta escolher uma hipótese fraca h_t com baixo erro ponderado ϵ_t, porém deve ser um pouco melhor do que uma escolha ao acaso, e fica tipicamente longe de zero. O objetivo do classificador franco é reduzir gradativamente o erro ponderado.

Se um classificador escolher entre duas classes -1 e $+1$ ao acaso, a probabilidade de ambas as classes serem escolhidas será igualmente de $1/2$. Para se obter uma probabilidade um pouco melhor do que ao acaso, o erro ponderado ϵ_t do classificador deve ser de pelo menos $1/2 - \gamma$, onde, γ é uma pequena constante positiva.

Na linha 8 do algoritmo AdaBoost, é calculado o parâmetro α_t que determina a importância associada para h_t. Observa-se que $\alpha_t > 0$ se $\epsilon_t < 1/2$, e que α_t fica maior

Figura 12 – Algoritmo AdaBoost

1 Dados: $(x_1, y_1), ..., (x_m, y_m)$ onde $x_i \in X$, $y_i \in \{-1, +1\}$;
2 Inicializar: $D_1(i) = 1/m$ para $i = 1, ..., m$;
3 Para $t = 1, ..., T$:
4 • Treinar o classificador fraco usando a distribuição D_t;
5 • Obter a hipótese fraca $h_t : X \to \{-1, +1\}$;
6 • Objetivo: selecionar h_t para minimizar o erro ponderado:
7 $$\epsilon_t \doteq \mathbf{Pr}_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i) \neq y_i} D_t(i);$$
8 • Escolher $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$;
9 • Atualizar, para $i = 1, ..., m$:
10 $$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{se } h_t(x_i) = y_i \text{ (classificação correta)} \\ e^{\alpha_t} & \text{se } h_t(x_i) \neq y_i \text{ (classificação incorreta)} \end{cases}$$
11 $$= \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t},$$
12 onde Z_t é um fator de normalização (escolhido para que D_{t+1} seja uma distribuição) na forma:
13 $$Z_t \approx \sum_{i=1}^{m} D_t(i) exp(-\alpha_t y_i h_t(x_i));$$
14 Apresentar a hipótese final:
15 $$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right);$$

Fonte: Elaborado pelo autor, 2017, com base em Schapire e Freund (2012, p. 5).

conforme ϵ_t fica menor. Assim, quanto maior for a precisão do classificador base h_t, maior importância lhe será atribuída.

O peso $D_t(i)$ associado a cada exemplo, determina a sua probabilidade de ser escolhido para compor o conjunto de exemplos de treinamento daquela rodada. Estes pesos são atualizados pelas expressões das linhas 10 até 13 do algoritmo. Na linha 10 os pesos são multiplicados por $e^{-\alpha_t}$ para os exemplos classificados corretamente e e^{α_t} para os exemplos classificados incorretamente. A atualização pode ser expressa de forma mais sucinta pela expressão da linha 11 do algoritmo. Em seguida, os pesos são normalizados, dividindo-os pelo fator Z_t (vide cálculo na linha 13), para garantir que a nova distribuição $D_t + 1$ some aproximadamente 1.

O efeito destas regras é incrementar os pesos dos exemplos classificados incorretamente por h_t, e decrementar os pesos dos exemplos classificados corretamente. Assim, os pesos maiores tendem a concentrar-se nos exemplos mais difíceis de serem classificados, os quais consequentemente terão maior probabilidade de serem escolhidos para a próxima rodada. Desta forma, o algoritmo AdaBoost tenta forçar o classificador base a aprender alguma coisa nova sobre os dados a cada rodada.

Após muitas chamadas ao classificador base, o algoritmo AdaBoost na linha 15, combina o resultado de cada chamada em um classificador final H. Isto é conseguido por um voto simples de peso ponderado de cada classificador base. Dado um novo exemplo x desconhecido (não rotulado com -1 ou +1), o classificador combinado avaliará todos os classificadores base, e predirá com o voto ponderado de cada um deles, a qual rótulo o exemplo desconhecido pertence. O voto do t-ésimo classificador base h_t é ponderado pelo seu parâmetro α_t previamente calculado.

Para ilustrar como o algoritmo AdaBoost funciona, será apresentado um exemplo baseado em Schapire e Freund (2012, p. 7). As instâncias são pontos em um plano rotulados com + ou -. Neste caso, tem-se $m = 10$ exemplos de treinamento, cinco rotulados com positivo e cinco com negativo, conforme pode ser visto na Figura 13.

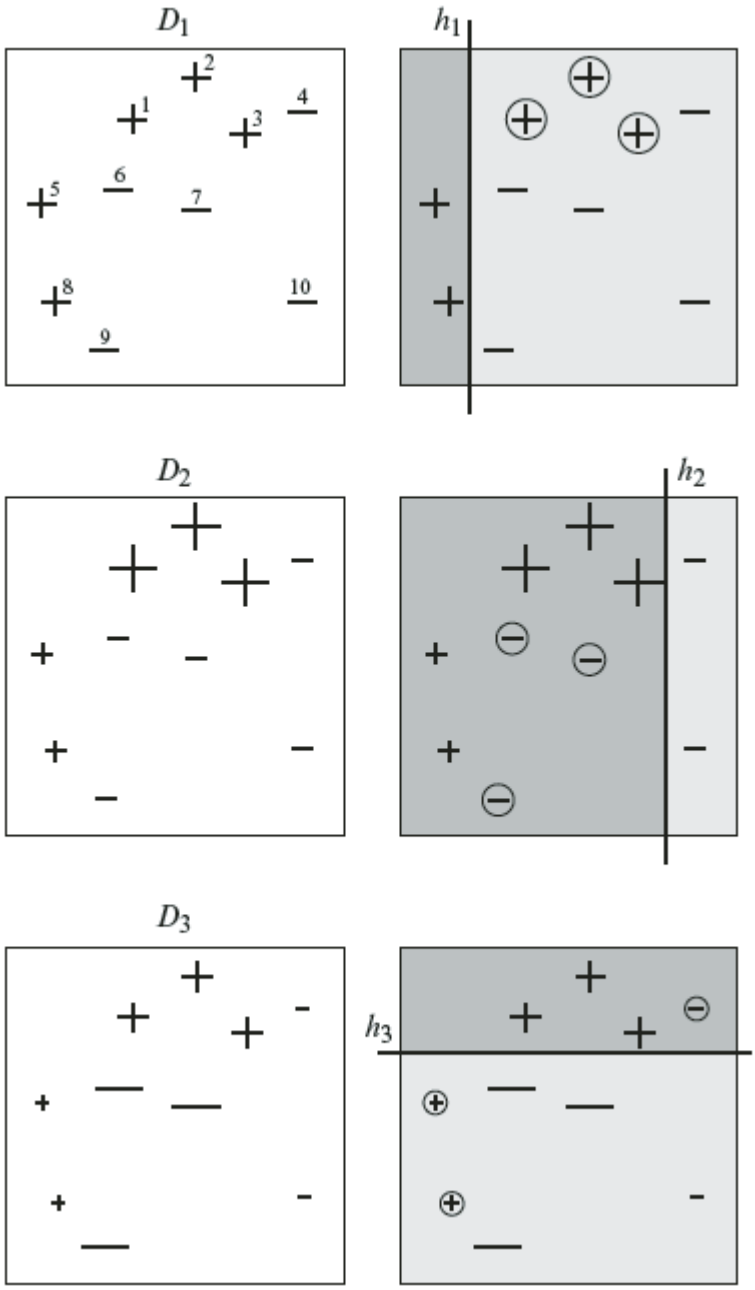
Supondo que o classificador base encontre classificadores definidos por linhas verticais ou horizontais através do plano. O classificador de linha vertical, separa para o lado esquerdo da linha todos os exemplos positivos e para o lado direito todos os exemplos negativos. É possível perceber que um classificador deste tipo, não consegue classificar mais de sete entre os dez exemplos de treinamento, significando que nenhum tem taxa de erro abaixo de 30%. Em cada rodada t, é suposto que o classificador base sempre encontre a hipótese base com mínimo erro ponderado em relação a distribuição D_t.

Neste exemplo será mostrado como é possível, utilizando uma combinação de classificadores fracos, em apenas $T = 3$ rodadas classificar corretamente todos os dez exemplos com o algoritmo AdaBoost. Também é possível acompanhar os valores calculados em cada rodada, através da Figura 14.

Na rodada 1, são atribuídos pesos iguais para todos os exemplos de treinamento, como indicado na Figura 13, pelo desenho de todos os exemplos com mesmo tamanho na caixa D_1. O classificador base escolhe a hipótese h_1, que classifica pontos como positivos apenas se caírem no lado esquerdo da linha. Esta hipótese classifica incorretamente os três pontos positivos que estão circulados. Assim, seu erro ϵ_1 é 0.30. Utilizando ϵ_1 na expressão da linha 8 do algoritmo da Figura 12 obtém-se $\alpha \approx 0.42$.

Na construção de D_2, os pesos dos três pontos classificados errados por h_1 são incrementados (retângulo vermelho na linha $D_2(i)$ da Figura 14), enquanto os pesos

Figura 13 – Uma ilustração de como AdaBoost funciona com $m = 10$ exemplos. Cada linha representa uma rodada, para $t = 1, 2, 3$. A caixa à esquerda em cada linha representa a distribuição D_t, com o tamanho de cada exemplo dimensionado em proporção com o peso de sua distribuição. Cada caixa à direita mostra a hipótese fraca h_t, onde a região mais escura indica a área predita para os exemplos positivos. Exemplos classificados errados por h_t estão circulados



Fonte: Schapire e Freund (2012, p. 8).

de todos os outros pontos são decrementados. Isto pode ser visto pelos tamanhos dos pontos na caixa D_2 da segunda linha da Figura 13.

Na rodada 2, o classificador escolhe a linha marcada por h_2. Desta vez, classifica corretamente os três pontos com pesos mais altos classificados errados por h_1, porém, com o custo de classificar errado outros três pontos com pesos mais baixos, os

Figura 14 – Cálculos do exemplo da Figura 13. Os pesos dos exemplos classificados errados pela hipótese h_t estão sublinhados nas linhas $D_t(i)$. Os pesos dentro de retângulos vermelhos, destacam seu incremento devido a sua classificação ter sido incorreta na rodada anterior

	1	2	3	4	5	6	7	8	9	10	
$D_1(i)$	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	0.10	$\epsilon_1 = 0.30, \alpha_1 \approx 0.42$
$e^{-\alpha_1 y_i h_1(x_i)}$	1.53	1.53	1.53	0.65	0.65	0.65	0.65	0.65	0.65	0.65	
$D_1(i)\ e^{-\alpha_1 y_i h_1(x_i)}$	0.15	0.15	0.15	0.07	0.07	0.07	0.07	0.07	0.07	0.07	$Z_1 \approx 0.92$
$D_2(i)$	0.17	0.17	0.17	0.07	0.07	0.07	0.07	0.07	0.07	0.07	$\epsilon_2 \approx 0.21, \alpha_2 \approx 0.65$
$e^{-\alpha_2 y_i h_2(x_i)}$	0.52	0.52	0.52	0.52	0.52	1.91	1.91	0.52	1.91	0.52	
$D_2(i)\ e^{-\alpha_2 y_i h_2(x_i)}$	0.09	0.09	0.09	0.04	0.04	0.14	0.14	0.04	0.14	0.04	$Z_2 \approx 0.82$
$D_3(i)$	0.11	0.11	0.11	0.05	0.05	0.17	0.17	0.05	0.17	0.05	$\epsilon_3 \approx 0.14, \alpha_3 \approx 0.92$
$e^{-\alpha_3 y_i h_3(x_i)}$	0.40	0.40	0.40	2.52	2.52	0.40	0.40	2.52	0.40	0.40	
$D_3(i)\ e^{-\alpha_3 y_i h_3(x_i)}$	0.04	0.04	0.04	0.11	0.11	0.07	0.07	0.11	0.07	0.02	$Z_3 \approx 0.69$

Fonte: Elaborado pelo autor, 2017, com base em Schapire e Freund (2012, p. 9).

quais foram classificados corretamente por h_1. Sob a distribuição D_2, estes três pontos tem peso de apenas 0.07, assim o erro de h_2 é $\epsilon_2 \approx 0.21$, e $\alpha_2 = 0.65$. Na construção de D_3, estes três pontos tem seus pesos incrementados (retângulos vermelhos na linha $D_3(i)$ da Figura 14), enquanto os pesos dos outros pontos são decrementados.

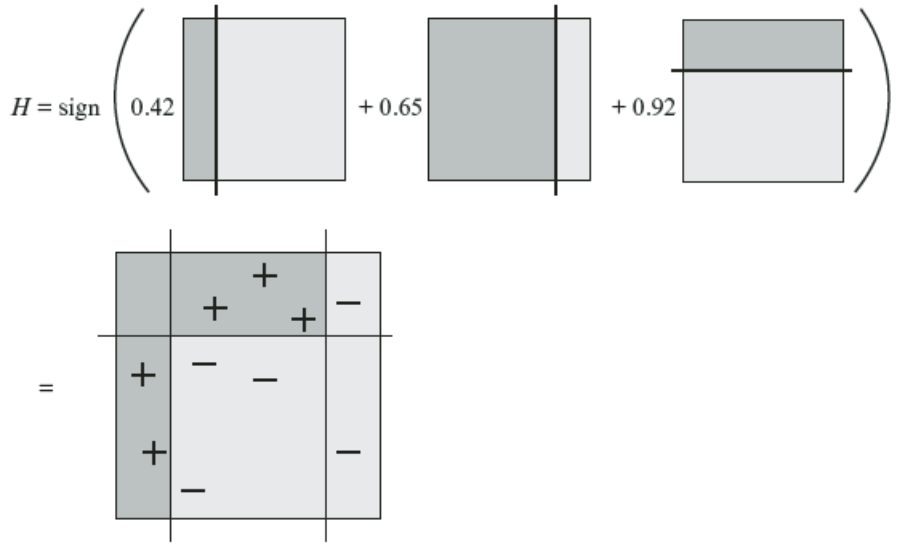
Na rodada 3, o classificador h_3 é escolhido. Este classificador não perde nenhum dos pontos classificados errados por h_1 e h_2, pois eles tem peso relativamente alto em D_3. Em vez disto, ele classifica incorretamente três pontos que foram classificados corretamente por h_1 e h_2, e portanto, possuem pesos bem baixos em D_3, com $\epsilon_3 \approx 0.14$ e $\alpha \approx 0.92$.

O classificador combinado H é um voto ponderado de h_1, h_2 e h_3, como mostrado na Figura 15, onde os pesos dos respectivos classificadores são α_1, α_2 e α_3. Embora cada um dos classificadores fracos, classifica três de dez exemplos incorretamente, o classificador combinado, como mostrado na Figura 15, classifica corretamente todas os exemplos de treinamento. Por exemplo, a classificação do exemplo no canto superior direito da Figura 13 (número 4), que é classificado como negativo por h_1 e h_2, mas positivo por h_3 é $sign(-\alpha_1 - \alpha_2 + \alpha_3) = sign(-0.15) = -1$.

2.3.2 Máquina de Vetor de Suporte (SVM)

A Máquina de Vetor de Suporte (*Support Vector Machine* - SVM), é explicada por Cortes e Vapnik (1995) como sendo um tipo de aprendizagem por máquina que mapeia vetores de entrada em um espaço de características de mais alta dimensionalidade através de um mapeamento não linear escolhido a priori. Nesse espaço é construída uma superfície de decisão linear com alta habilidade de generalização.

Figura 15 – Cálculos do exemplo da Figura 13. O classificador combinado para o exemplo da Figura 13 é calculado como o sinal da soma ponderada das três hipóteses fracas, $\alpha_1 h_1 + \alpha_2 h_2 + \alpha_3 h_3$, como mostrado no topo. Isto é equivalente ao classificador mostrado na parte inferior. As regiões mais escuras indicam as amostras classificadas como positivas



Fonte: Schapire e Freund (2012, p. 9).

Para Burges (1998), seja um conjunto de exemplos de treinamento rotulados e separáveis linearmente $\{x_i, y_i\}; i = 1, 2, ..., l; y_i \in \{-1, +1\}; x_i \in R^d$ com duas classes (separação binária) projetadas em um espaço d-dimensional. Supondo que haja um hiperplano que separe as amostras positivas das negativas. Todo ponto x que esteja no hiperplano satisfaz $\mathrm{w} \cdot x + b = 0$, onde w é a normal até o hiperplano, $|b|/||\mathrm{w}||$ é a distância perpendicular da origem até o hiperplano e $||\mathrm{w}||$ é a norma Euclidiana de w. Sendo d_+ e d_- a menor distância do hiperplano até a amostra positiva e negativa respectivamente mais próxima. A margem do hiperplano de separação é dada por $d_+ + d_-$.

O objetivo em treinar uma SVM é encontrar um hiperplano com a maior margem de separação possível entre os pontos das amostras positivas e negativas. Quanto maior for a margem de separação, melhor a capacidade de generalização do classificador (DUDA; HART; STORK, 2000, p. 49). Isto implica, segundo Burges (1998), que todos os dados de treinamento devem satisfazer as restrições das Equações 2.13 e 2.14:

$$x_i \cdot \mathrm{w} + b \geq +1 \quad \text{para} \quad y_i = +1 \tag{2.13}$$

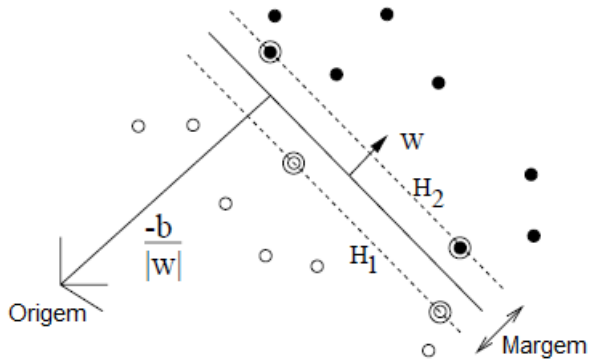
$$x_i \cdot \mathrm{w} + b \leq -1 \quad \text{para} \quad y_i = -1 \tag{2.14}$$

As Equações 2.13 e 2.14 podem ser combinadas na Equação 2.15 da seguinte forma:

$$y_i(x_i \cdot \mathrm{w} + b) - 1 \geq 0 \quad \forall i \tag{2.15}$$

Os pontos que atendam a igualdade da Equação 2.13 são projetados no hiperplano $H_1 : x_i \cdot \mathrm{w} + b = 1$ com normal **w** e a distância perpendicular da origem $|1-b|/||\mathrm{w}||$. Já os pontos que atendam a igualdade da Equação 2.14 são projetados no hiperplano $H2 : x_i \cdot \mathrm{w} + b = -1$ com normal novamente **w** e a distância perpendicular da origem $|-1-b|/||\mathrm{w}||$. Portanto $d_+ = d_- = 1/||\mathrm{w}||$ e a margem $2/||\mathrm{w}||$. H_1 e H_2 são paralelos e os pontos não são projetados entre eles. Assim, podem ser encontrados os pares de hiperplanos com a margem máxima minimizando $||\mathrm{w}||^2$ respeitando a Equação 2.15. Aqueles pontos que se encontram sobre os hiperplanos H_1 e H_2 e cuja remoção mudaria a solução encontrada, são os vetores de suporte. Eles estão circulados na Figura 16.

Figura 16 – Hiperplanos de separação linear. Os vetores de suporte estão circulados



Fonte: Burges (1998).

Muitas vezes os dados não tendem a separação linear diretamente devido a presença de ruídos e ao próprio domínio do problema. Isto pode ser ajustado suavizando as restrições das Equações 2.13 e 2.14 adicionando variáveis de folga positivas $\xi_i, i = 1, 2, ..., l$ nas restrições que passarão a ser:

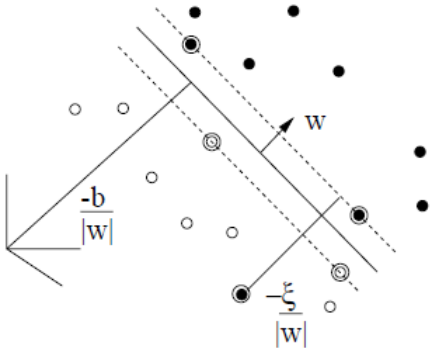
$$x_i \cdot \mathrm{w} + b \geq +1 - \xi_i \quad \text{para} \quad y_i = +1 \tag{2.16}$$

$$x_i \cdot \mathrm{w} + b \leq -1 + \xi_i \quad \text{para} \quad y_i = -1 \tag{2.17}$$

$$\xi_i \geq 0 \quad \forall_i. \tag{2.18}$$

Este procedimento permite que alguns pontos fiquem entre os hiperplanos e alguns erros de treinamento ultrapassem as fronteiras do seu hiperplano. A função de minimização é alterada de $||\mathrm{w}||^2$ para $||\mathrm{w}||^2/2 + C(\sum_i \xi_i)^k$, onde C é um parâmetro definido pelo usuário, e quanto maior for C maior será a penalidade para erros, conforme exibido na Figura 17.

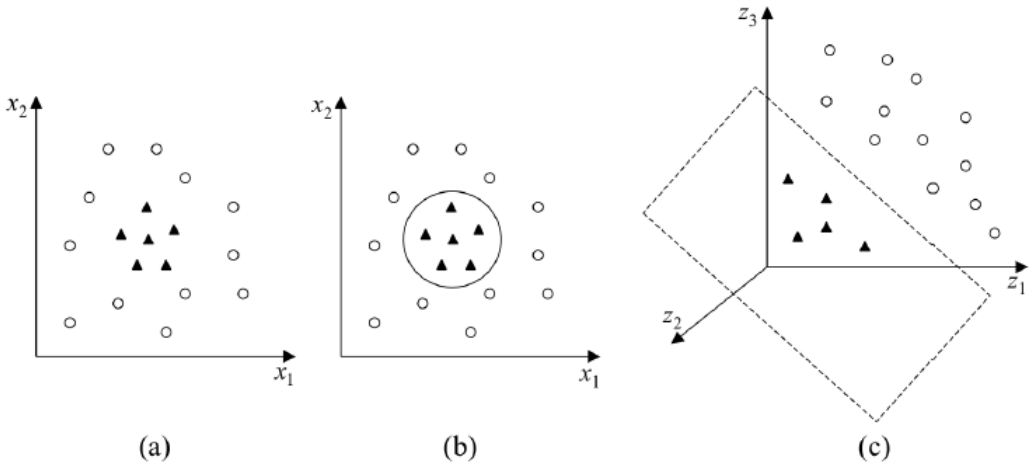
Figura 17 – Hiperplanos de separação linear para casos não separáveis



Fonte: Burges (1998).

Conforme explicado por Burges (1998), muitos problemas não permitem dividir os dados de treinamento de maneira satisfatório por um hiperplano. Esta questão pode ser resolvida projetando os dados de entrada R^d em um novo espaço de características H de alta dimensionalidade, veja Figura 18. Isto pode ser feito por uma função

Figura 18 – (a) Conjunto de dados não linear em um espaço 2-D; (b) Fronteira não linear no espaço de entrada 2-D; (c) Dados mapeados do espaço 2-D para um espaço 3-D, produzindo uma fronteira com separação linear



Fonte: Lorena e Carvalho (2007).

de mapeamento Φ, conforme apresentado pela Equação 2.19:

$$\Phi : R^d \mapsto H. \tag{2.19}$$

Assim, o algoritmo de treinamento dependerá apenas dos dados dos produtos escalares de H com funções na forma $\Phi(X_i) \cdot \Phi(X_j)$. Empregando uma função de *kernel* K tal que $K(X_i, X_j) = \Phi(X_i) \cdot \Phi(X_j)$, será necessário utilizar apenas K no algoritmo de treinamento, e não será necessário conhecer explicitamente Φ.

Burges (1998), explica que as primeiras funções de *kernel* a serem empregadas no reconhecimento de padrões foram: classificador polinomial de grau p (Equação 2.20), função de base radial gaussiana (Equação 2.21) e um tipo particular de rede neural sigmoidal de duas camadas (Equação 2.22).

$$K(x, y) = (x \cdot y + 1)^p \tag{2.20}$$

$$K(x, y) = e^{-||x-y||^2/2\sigma^2} \tag{2.21}$$

$$K(x, y) = \tanh(kx \cdot y - \delta). \tag{2.22}$$

Embora os classificadores SVM sejam binários (separam exemplos pertencentes a apenas duas classes), eles podem ser combinados para separar múltiplas classes. Em um conjunto de N classes a serem separadas, para cada classe faz-se o treinamento um contra o resto, onde o **um** é a classe positiva a ser separada do **resto** que representa a classe negativa e é formado pelas classes restantes. Após o treinamento, submetendo uma amostra de testes, ela corresponde a categoria para a maior distância positiva (mais próxima de 1).

2.4 HAAR-LIKE

A técnica Haar-like foi proposta por Viola e Jones (2004) e empregada originalmente para detectar faces de pessoas em imagens, com processamento rápido e atinge alta precisão na detecção. Utiliza como algoritmo de aprendizagem o AdaBoost. Um dos pontos fortes desta técnica de detecção, é a sua capacidade de generalização, o que a torna viável para detecção de vários tipos de objetos além de faces. Neste trabalho, esta técnica será empregada para detecção de veículos.

Resumidamente, a técnica é fundamentada em três pontos principais, conforme relatado por Viola e Jones (2004). Primeiro foi feita uma nova representação da imagem, chamada de imagem integral, a qual permite calcular características muito rapidamente. A imagem integral é semelhante a “tabela de área somada” proposta por Crow (1984) para mapeamento de textura. Uma vez construída a imagem integral, as características podem ser calculadas em tempo constante.

O segundo ponto, é a construção de um classificador eficiente para extração apenas das características importantes dentre muitas potenciais candidatas, utilizando o algoritmo AdaBoost. Cada classificador fraco é restringido para depender de apenas uma característica. Assim, cada estágio (ou rodada) do algoritmo AdaBoost, que gera um novo classificador fraco, pode ser visto como um extrator de características.

Por fim, o terceiro ponto é a combinação de sucessivos classificadores mais complexos em uma estrutura de cascata, que descarta rapidamente regiões de fundo, e que gasta mais processamento sobre regiões com alto potencial de serem o objeto procurado.

2.4.1 Extração de Características Utilizando Haar-like

Para a extração de características, são utilizadas imagens em níveis de cinza. O cálculo das características é simples, foi motivado em partes pelo trabalho de Papageorgiou e Poggio (2000), que não trabalha diretamente sobre os níveis de intensidade dos pixeis da imagem, mas usam algumas funções bases de Haar como as *wavelets* 2D, nas direções: vertical, horizontal e diagonal.

Um ponto forte do trabalho de Viola e Jones (2004) é justamente a extração de características que ficou conhecida como Haar-like. Esta técnica para extração de características se tornou um algoritmo clássico usado para treinar o classificador AdaBoost.

As características são chamadas de características de retângulos. Utiliza-se regiões retangulares de mesmo tamanho e forma, podendo ser verticalmente ou horizontalmente adjacentes, onde soma-se os valores de áreas retangulares de cor clara e soma-se áreas retangulares de cor escura.

A característica é a diferença (subtração) entre os dois somatórios. Mais especificamente, são utilizados três tipos de características. Característica de dois retângulos, que consiste na diferença do somatório dos pixeis entre dois retângulos. Características de três retângulos, calcula a soma de dois retângulos externos e subtrai pela soma de um retângulo central.

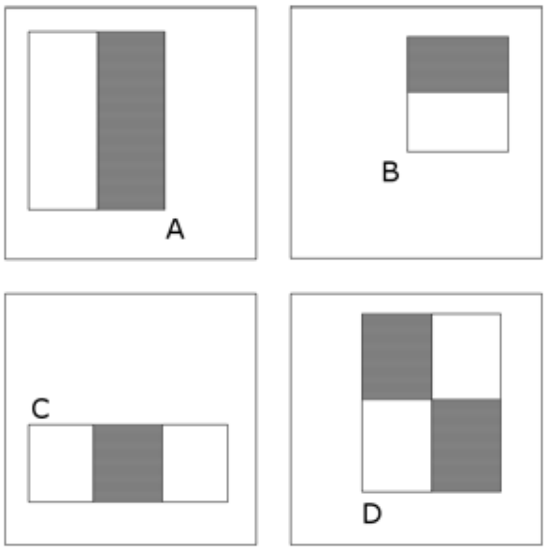
Por fim, características de quatro retângulos, onde é calculada a diferença entre a soma de dois pares de retângulos diagonais. A Figura 19 exibe exemplos dos tipos de retângulos.

A resolução da sub-janela base do detector de faces é de 24 x 24 pixeis. Considerando todas as possibilidades de posição, tipo e tamanho dos retângulos de características, são geradas para cada sub-janela de detecção, 160.000 características de retângulos.

2.4.2 Imagem Integral

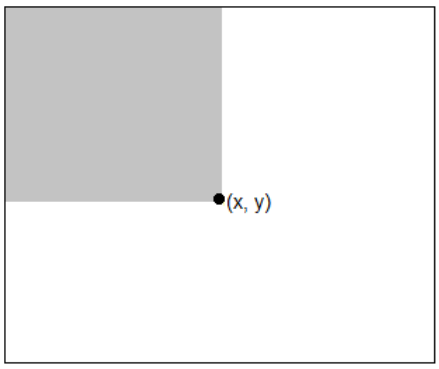
Características de retângulos podem ser calculadas muito rapidamente utilizando uma representação intermediária da imagem original chamada de imagem integral. A imagem integral na posição x e y (vide Figura 20) contém a soma dos pixels a esquerda e acima de x e y, inclusive, conforme Equação 2.23.

Figura 19 – Exemplos de características de retângulos mostrados em relação a sub-janela de detecção. São somados os pixeis que estão nas áreas dos retângulos claros, e esta soma é subtraída da soma dos pixeis que estão nas áreas dos retângulos escuros. Características de dois retângulos são mostrados em A e B. Em C, é mostrado um retângulo de três características, e D mostra uma característica de quatro retângulos.



Fonte: Viola e Jones (2004).

Figura 20 – O valor da imagem integral no ponto (x, y) é a soma de todos os pixels acima e a esquerda



Fonte: Viola e Jones (2004).

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y'), \tag{2.23}$$

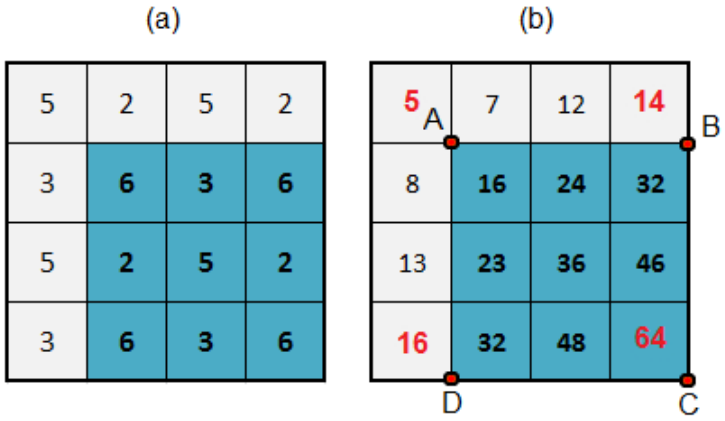
onde $ii(x,y)$ é a imagem integral e $i(x,y)$ é a imagem original. A imagem integral pode ser calculada em apenas uma passada sobre a imagem original pela Equação 2.24:

$$ii(x,y) = i(x,y) + ii(x-1,y) + ii(x,y-1) - ii(x-1,y-1), \tag{2.24}$$

sendo que, $ii(-1,y) = 0$, e $ii(x,-1) = 0$.

Usando a imagem integral, qualquer soma de área retangular pode ser feita com apenas quatro acessos a matriz de valores (vide Figura 21). O cálculo da diferença da soma de duas áreas retangulares pode ser feito com oito acessos. Como nesta técnica os retângulos são adjacentes, para características de dois retângulos a diferença da soma dos dois retângulos pode ser calculada em apenas seis acessos, oito no caso de características de três retângulos e nove para características de quatro retângulos.

Figura 21 – (a) Imagem original. (b) Imagem integral calculada pela Equação 2.24. A soma dos pixeis dentro da região retangular destacada na imagem original, pode ser feito com quatro acessos na imagem integral calculando C + A - (B + D), sendo 64 + 5 - (14 + 16) = 39



Fonte: Elaborado pelo autor, 2017, com base em Viola e Jones (2004).

2.4.3 Aprendizado AdaBoost para Haar-like

Mesmo que o cálculo de cada característica seja muito eficiente, o cálculo de todo o conjunto das características é muito carro computacionalmente, comprometendo o desempenho do detector. Assim, foi combinado um conjunto de poucas características para formar um classificador efetivo. Sendo que o principal desafio, seria encontrar quais são as melhores características para compor este pequeno conjunto. Como solução, foi utilizada uma variante do algoritmo AdaBoost, tanto para obter um conjunto pequeno contendo as melhores características, como para construir um classificador combinado eficiente.

Relembrando o algoritmo AdaBoost, apresentado na seção 2.3.1, proposto por Freund e Schapire (1995), são treinados uma série de classificadores fracos (cada um com taxa de erro um pouco menor do que 50%), que combinados produzem um classificador final muito eficiente que consegue rotular se uma amostra desconhecida pertence a classe de exemplos positivos ou a classe de exemplos negativos.

A chave para conseguir o melhoramento ou impulsão na combinação dos classificadores fracos, é que cada um deles é ensinado a resolver uma parte diferente do problema. Isto é alcançado, aumentado os pesos dos exemplos classificados incorretamente na rodada anterior, de forma que sejam adicionados aos exemplos de entrada na rodada seguinte da série de classificadores, isto força o novo classificador a aprender algo diferente do que o classificador anterior. Por fim, são combinados os votos ponderados de cada classificador fraco, resultando no parecer final para a classificação.

Schapire e Freund (2012) afirmam que, o erro de treinamento do classificador combinado forte se aproxima de zero exponencialmente ao número de rodadas. Também destacam como ponto relevante, a capacidade de generalização, a qual está relacionada com a margem de separação entre os exemplos, e que AdaBoost consegue produzir margens grandes rapidamente.

O desafio é associar pesos grandes para boas funções de classificação e pesos pequenos para funções fracas. Neste sentido, AdaBoost é uma técnica efetiva para selecionar um conjunto pequeno de boas funções de classificação com significativa variedade. Fazendo uma analogia entre classificadores fracos e características, AdaBoost pode ser empregado como um procedimento efetivo para encontrar um pequeno número de boas características com significativa variedade.

A fim de completar esta analogia, o classificador fraco pode ser restringido para que cada função do conjunto de classificação, dependa de apenas uma característica. Para atingir este objetivo, o algoritmo de aprendizagem fraco pode ser projetado para selecionar a característica de retângulo que melhor separa os exemplos positivos dos negativos. Para cada característica, o classificador fraco determina a função de classificação de limiar ótimo, que obtém o número mínimo de exemplos classificados incorretamente. A função é definida pela Equação 2.25.

$$h(x, f, p, \theta) = \begin{cases} 1 & \text{se} \quad pf(x) < p\theta \\ \\ 0 & \text{caso} \quad \text{contrário} \end{cases} \tag{2.25}$$

Onde x é uma sub-janela de 24 x 24 pixeis de uma imagem, f é uma característica, θ é um limiar e p é uma polaridade indicando a direção da desigualdade. O algoritmo proposto por Viola e Jones (2004) pode ser visto na Figura 22:

O algoritmo é usado para selecionar o melhor classificador fraco do conjunto de possíveis classificadores. Uma vantagem destacada por Viola e Jones (2004) sobre o AdaBoost como mecanismo de seleção de características, é a velocidade de aprendizado. Usando AdaBoost como classificador de 200 características, ele pode aprender em $O(MNK)$ ou aproximadamente 10^{11} operações.

Figura 22 – O algoritmo AdaBoost adaptado por Viola e Jones

Dados exemplos de imagens: $(x_1, y_1), ..., (x_n, y_n)$ onde $y_i = 0, 1$ para exemplos negativos e positivos respectivamente;
Inicializar pesos $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $y_i = 0, 1$ respectivamente, onde m e l são o número de exemplos negativos e positivos respectivamente;
Para $t = 1, ..., T$:

- Normalizar os pesos, $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$;
- Selecionar o melhor classificador fraco com relação ao erro ponderado: $\epsilon_t = min_{f,p,\theta} \sum_i w_i |h(x_i, f, p, \theta) - y_i|$;
  - Veja explicação que segue abaixo para detalhes de uma implementação eficiente.
- Definir $h_t(x) = h(x, f_t, p_t, \theta_t)$, onde f_t, p_t, e θ_t são minimizadores de ϵ_t;
- Atualizar os pesos:
  $w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$,
  onde $e_i = 0$ se o exemplo x_i é classificado corretamente, $e_i = 1$ caso contrário, e $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$;

- O classificador forte final é:

$$C(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{caso contrário} \end{cases},$$

onde $\alpha_t = log\frac{1}{\beta_t}$;

Fonte: Elaborado pelo autor, 2017, com base em Viola e Jones (2004).

Onde M é a quantidade de classificadores fracos, N é a quantidade de exemplos e K é a quantidade de características. Em cada rodada, as dependências das características selecionadas anteriormente são eficientemente e compactamente calculadas usando os pesos dos exemplos. Estes pesos podem então ser usados para avaliar um determinado classificador fraco em tempo constante.

Para cada característica, os exemplos são ordenados pelo valor da característica. O algoritmo AdaBoost calcula o limiar ótimo para a característica em apenas uma passada sobre a lista ordenada. O erro para o limiar que separa o limite entre o elemento atual e o anterior na lista ordenada é dado pela Equação 2.26

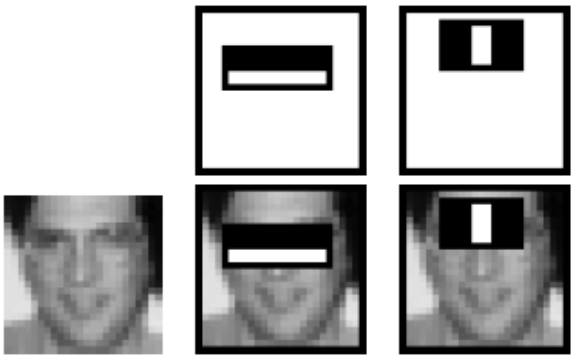
$$e = min(S^+ + (T^- - S^-), S^- + (T^+ - S^+)), \tag{2.26}$$

onde T^+ é a soma total dos pesos dos exemplos positivos, T^- é a soma total dos exemplos negativos, S^+ é soma dos pesos dos elementos positivos abaixo do elemento corrente e S^- a soma dos pesos dos elementos negativos abaixo do elemento corrente. Estas somas são mantidas e atualizadas para cada elemento da lista ordenada conforme o processo segue.

Para a tarefa de detecção de faces, as características de retângulo iniciais selecionadas pelo algoritmo AdaBoost são significativas e de fácil interpretação. A primeira característica foca na propriedade em que a região dos olhos é frequentemente mais escura que a região do nariz e bochechas. A segunda característica depende da pro-

priedade em que os olhos são mais escuros do que a ponte acima da região do nariz que os separam, conforme pode ser visto na Figura 23.

Figura 23 – A primeira e segunda características selecionadas pelo algoritmo AdaBoost. A primeira linha da figura mostra as duas características, as quais são sobrepostas sobre uma imagem típica de face na segunda linha a fim de calcular os valores das duas características



Fonte: Viola e Jones (2004).

O algoritmo consegue descartar a grande maioria das características. Testes iniciais foram feitos por Viola e Jones (2004), onde após rodar o algoritmo da Figura 22 remanesceram apenas 200 características, obtendo com elas resultados promissores com taxa de detecção de 95% e requerendo 0,7 segundos para escanear uma imagem de 384 x 288 pixels. Porém, Viola e Jones afirmam que adicionar características ao classificador, aumenta diretamente o tempo de processamento.

2.4.4 Cascata de Classificadores

Um algoritmo para construir uma cascata de classificadores, o qual aumenta a taxa de desempenho e diminui o tempo de processamento foi proposto por Viola e Jones (2004). Um ponto chave neste algoritmo, é que classificadores fracos podem rejeitar a maioria das sub-janelas negativas, enquanto detectam quase todas as instâncias positivas. São usados classificadores simples para rejeitar a maioria das sub-janelas negativas, antes que classificadores mais complexos sejam chamados para atingir baixas taxas de falsos positivos.

Começando com um classificador de duas características, pode ser obtido um classificador de faces efetivo, ajustando o limiar do classificador para minimizar os falsos negativos. O limiar inicial do algoritmo AdaBoost, $\frac{1}{2}\sum_{t=1}^{T}\alpha_t$, produz baixa taxa de erro nos exemplos de treinamento. Limiares com valores baixos produzem maiores taxas de detecção e maiores taxas de falsos positivos.

Baseado na validação através de um conjunto para treinamento, Viola e Jones (2004) obtiveram 100% de detecção com 50% de falso positivos usando um classifica-

dor com duas características. Afirmam ainda que, o classificador de duas características pode reduzir significativamente o número de sub-janelas com poucas operações:

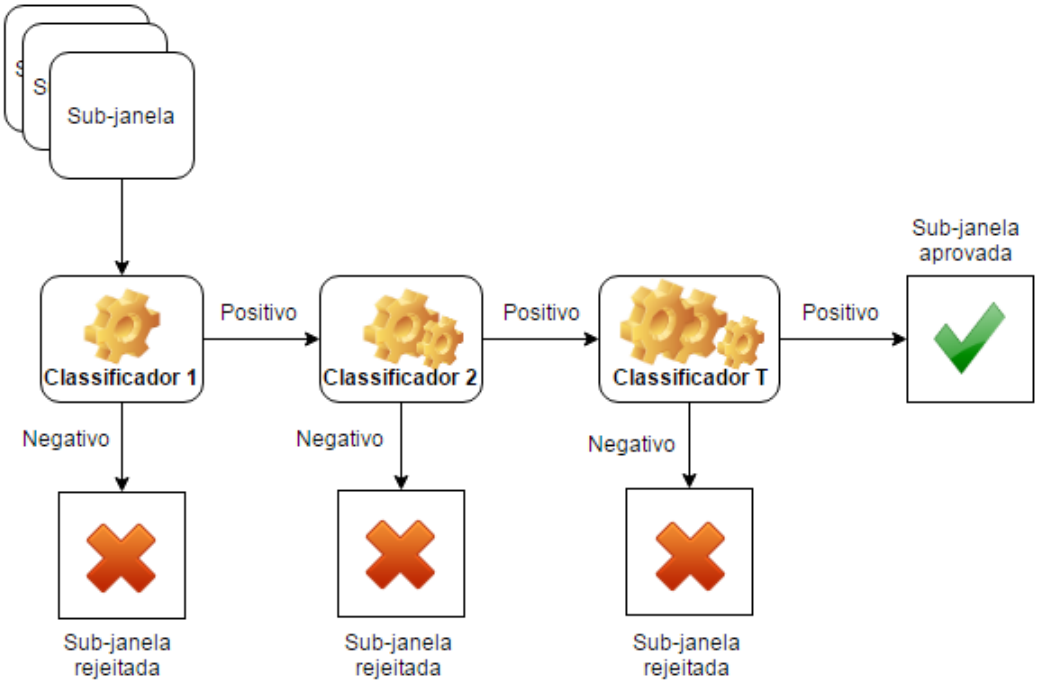
a) avaliar as características de retângulo (requer de 6 a 9 referências a matriz de valores por característica);

b) calcular o classificador fraco para cada característica (requer um cálculo de limiar por característica);

c) combinar os classificadores fracos (requer uma multiplicação por característica, uma adição, e um limiar).

O esquema dos classificadores em cascata funciona da seguinte maneira: um resultado positivo no primeiro classificador, aciona o segundo classificador que foi ajustado para atingir uma alta taxa de detecção. Havendo também resultado positivo no segundo classificador dispara um terceiro classificador e assim por diante. Ocorrendo resultado negativo em qualquer classificador, ele imediatamente rejeita a sub-janela e parte para outra. O esquema dos classificadores em cascata é apresentado na Figura 24.

Figura 24 – Esquema de detecção com classificadores em cascata.



Fonte: Elaborado pelo autor, 2016, com base em Viola e Jones (2004).

Normalmente em uma imagem, poucas regiões (tratadas aqui como sub-janelas) contém o objeto procurado, e a grande maioria das regiões são apenas de fundo. O classificador em cascata tenta rejeitar estas regiões de fundo rapidamente, já nos pri-

meiros estágios. Isto deve-se ao fato de que nos primeiros estágios os classificadores serem bem simples por analisarem poucas características de retângulo. Quanto mais profundo na cascata estiver o classificador, mais características ele analisará, portanto tornando-se mais complexo, e maior será o seu custo de processamento. É rara uma região ser aprovada por todos os classificadores da cascata.

Parecido como uma árvore de decisão, classificadores subsequentes são treinados com os exemplos que passaram por todos os classificadores anteriores. Como resultado, o classificador seguinte tem uma tarefa mais difícil do que o classificador anterior.

O processo do classificador em cascata é conduzido a partir de um conjunto de objetivos de detecção e desempenho. A definição do número de estágios da cascata e o tamanho de cada estágio deve ser suficiente para atingir desempenho suficiente e minimizar o custo computacional.

Dado um classificador em cascata treinado, a sua taxa de falso positivo é dada pela Equação 2.27

$$F = \prod_{i=1}^{K} f_i, \tag{2.27}$$

onde F é a taxa de falso positivo do classificador em cascata, K é o número de classificadores, e f_i é a taxa de falso positivo do $i-$ésimo classificador dos exemplos que passam por ele.

A taxa de detecção deste mesmo classificador em cascata treinado é dada pela Equação 2.28

$$D = \prod_{i=1}^{K} d_i, \tag{2.28}$$

onde D é a taxa de detecção do classificador em cascata, K é o número de classificadores, e d_i é a taxa de detecção do $i-$ésimo classificador dos exemplos que passam por ele.

Dadas as metas gerais para falso positivos e taxa de detecção, taxas específicas podem ser determinadas para cada estágio do classificador em cascata. Por exemplo, uma taxa de detecção de $0,9$ pode ser alcançada por um classificador com 10 estágios se cada estágio tiver uma taxa de detecção de $0,99$ (desde que $0,9 \approx 0,99^{10}$). Cada estágio precisa atingir somente um taxa de falso positivo de aproximadamente 30% ($0,30^{10} \approx 6 \times 10^{-6}$).

O número de características avaliadas durante o escaneamento de uma imagem é um processo probabilístico. Cada sub-janela analisada vai progredir através dos classificadores da cascata, um por vez, até que seja rotulada como negativa ou, em raros casos, se conseguir passar por todos os classificadores como positiva.

O comportamento esperado deste processo é determinado pela distribuição das sub-janelas de uma imagem em um conjunto de testes. A medida chave de cada classificador é sua taxa de positivos, a proporção de sub-janelas rotuladas com chance de conter uma face. O número esperado de características avaliadas é dado pela Equação 2.29

$$N = n_0 + \sum_{i=1}^{K} \left(n_i \prod_{j<i} p_j \right), \tag{2.29}$$

onde N é o número esperado de características avaliadas, K é o número de classificadores, p_j é a taxa de positivos do $i-$ésimo classificador, e n_i são o número de características do $i-$ésimo classificador.

Viola e Jones (2004) afirmam que o algoritmo AdaBoost visa minimizar erros, e não é desenhado para alcançar altas taxas de detecção ao custo de uma taxa grande de falso positivos. Uma maneira simples de resolver isto é ajustar o limiar relacionado aos erros do AdaBoost. Limiares mais elevados produzem classificadores com menos falso positivos e menor taxa de detecção. Limiares inferiores produzem classificadores com mais falso positivos, e maior taxa de detecção. Advertem ainda que procedendo desta forma, não fica claro a preservação do treinamento e generalização com garantia providas pelo algoritmo AdaBoost.

Explicam também que geralmente classificadores com mais características produzem alta taxa de detecção com poucos falso positivos, porém com maior custo computacional. Na prática isto pode ser otimizado provendo uma estrutura em que: o número de estágios do classificador, o número de características, n_i, de cada estágio, e o limiar de cada estágio são ajustados a fim de minimizar o número esperado de características N, dando um alvo para F e D. Porém, encontrar os valores que otimizam estes parâmetros não é tarefa trivial.

Para construir um classificador em cascata com alta eficiência, Viola e Jones (2004) propuseram um algoritmo conforme Figura 25. O usuário seleciona a taxa máxima aceitável para f_i e a taxa mínima aceitável para d_i. Cada estágio da cascata de classificadores é treinado pelo algoritmo AdaBoost (como descrito na Figura 22), com o número de características usadas sendo incrementado até que os valores alvo para as taxas de detecção e falso positivos sejam alcançados para o estágio atual. As taxas são determinadas testando o classificador atual em um conjunto de validação.

Se o alvo para a taxa global de falso positivos ainda não foi atingido, um novo estágio é adicionado na cascata de classificadores. O conjunto negativo para treinamento dos estágios subsequentes é obtido coletando todas as detecções falsas encontradas, rodando o classificador atual em um conjunto de imagens as quais não contém nenhuma instância com faces.

Figura 25 – O algoritmo de treinamento para a construção de um classificador em cascata

Usuário seleciona valor para f, a taxa de falso positivo máxima aceitável por estágio e d, a taxa de detecção mínima aceitável por estágio;
Usuário seleciona a taxa alvo global de falso positivo F_{target};
$P =$ conjunto de exemplos positivos;
$N =$ conjunto de exemplos negativos;
$F_0 = 1.0$; $D_0 = 1.0$;
$i = 0$;
while $F_i > F_{target}$ **do**
    $i \leftarrow i + 1$;
    $n_i = 0$; $F_i = F_{i-1}$;
    while $F_i > f \times F_{i-1}$ **do**
        $n_i \leftarrow n_i + 1$;
        Use P e N para treinar o classificador com n_i características usando AdaBoost;
        Avaliar o classificador em cascata atual no conjunto de validação para determinar F_i e D_i;
        Decrementar o limiar para o i-ésimo classificador até que o classificador em cascata atual tiver uma taxa de pelo menos $d \times D_{i-1}$ (isso também afeta F_i);
    end
    $N \leftarrow \theta$;
    if $F_i > F_{target}$ **then**
        Avaliar o classificador em cascata atual no conjunto de imagens sem faces e colocar quaisquer falsas detecções no conjunto N;
    end
end

Fonte: Elaborado pelo autor, 2016, com base em Viola e Jones (2004).

2.4.5 Considerações Finais Sobre a Técnica

Viola e Jones (2004) relatam a utilização de 4.916 imagens de faces, as quais foram coletadas aleatoriamente da internet, para o conjunto de exemplos positivos, com dimensões de 24 x 24 pixeis. A região contendo a face é cortada manualmente por um usuário de forma que pegue a face, um pouco dos cabelos, e um pouco abaixo do queixo. Essa borda em torno da face melhora o desempenho do classificador, principalmente na rejeição antecipada de regiões que não são faces, comparado a utilizar estritamente a região da face.

O detector final é formado por uma cascata de 38 classificadores em série e um total de 6.060 características. O primeiro classificador em cascata é construído utilizando duas características e rejeita cerca de 50% das regiões sem faces enquanto detecta corretamente perto de 100% das faces. O segundo classificador tem 10 características e rejeita 80% das regiões sem faces enquanto detecta quase 100% das faces. O terceiro e o quarto classificador tem 25 características cada e o quinto até o oitavo classificador possuem 50 características cada. Os classificadores seguintes

possuem quantidades de características variadas de acordo com o algoritmo da Figura 25.

As escolhas específicas das quantidades de características nos primeiros estágios foram feitas por um processo de tentativa e erro, em que, o número de características foi incrementado até que uma redução significativa na taxa de falso positivos fosse alcançada. Mais estágios foram adicionados até que a taxa de falso positivos no conjunto de validação chegasse próximo de zero, mantendo ainda uma elevada taxa de detecção correta.

As sub-janelas sem faces usadas para treinamento no primeiro estágio foram selecionadas aleatoriamente de um conjunto de 9.500 imagens que não continham faces. Os exemplos sem faces usados para treinamento dos estágios subsequentes foram obtidos escaneando a cascata parcial através de imagens sem faces e coletando até no máximo 6.000 sub-janelas de falso positivos para cada estágio.

A imagem é escaneada pelo detector da esquerda para a direita e do topo para baixo, deslocando o detector alguns pixeis de distância do detector anterior. No entanto, Viola e Jones (2004) explicam que se deslocar o detector mais de um pixel por vez, reduz ligeiramente a taxa de detecção e também o número de falso positivos. Vários escaneamentos podem ser aplicados com escalas diferentes do detector, isto é mais prático do que escalonar a imagem, visto que, uma característica pode ser calculada em qualquer escala com o mesmo custo.

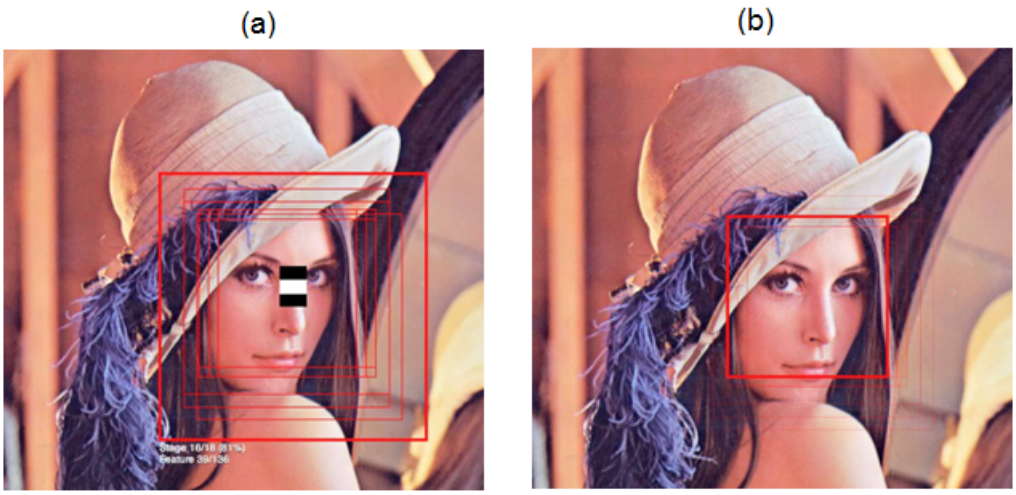
Devido ao detector não ser sensível a pequenas alterações de translação e escala é comum ocorrerem múltiplas detecções de uma mesma face nos vários escaneamentos realizados sobre a imagem, como pode ser visto na Figura 26 (a). Por isto, é feito um pós processamento, e às sub-janelas detectadas para a mesma face são combinadas em apenas uma área de detecção, o que pode ser visto na Figura 26 (b).

As áreas detectadas são particionadas em subconjuntos disjuntos. Duas áreas detectadas pertencem ao mesmo subconjunto se suas regiões delimitadoras se sobrepuserem. Cada subconjunto corresponde a uma face detectada e a sua região delimitadora final é definida pela média dos cantos de cada região do subconjunto.

2.5 PADRÃO BINÁRIO LOCAL DE BLOCOS MULTIESCALA

A técnica de padrão binário local de blocos de multi escala, do inglês *Multi-scale Block Local Binary Patterns* (MB-LBP) foi desenvolvida por Liao et al. (2007). A MB-LBP é baseada na técnica de padrão binário local, do inglês *Local Binary Pattern* (LBP). A LBP é uma técnica de extração de características local, baseada nos valores da intensidade dos pixels de uma vizinhança de 8. Já a MB-LBP é baseada nos valores médios da intensidade de sub-regiões de pixels, chamadas de blocos.

Figura 26 – Sub-janelas de detecção em escalas diferentes. (a) Várias sub-janelas em escalas diferentes para a mesma face. (b) Sub-janelas combinadas em apenas uma área de detecção



Fonte: Harvey (2017).

Três vantagens da técnica MB-LBP sobre a técnica LBP são apresentadas por Liao et al. (2007):

a) mais robusta do que a LBP;

b) codifica micro e macro estruturas, provendo representação mais completa da imagem do que a LBP; e

c) pode ser computada muito eficientemente usando o conceito de imagem integral, a mesma metodologia utilizada por Viola e Jones (2004).

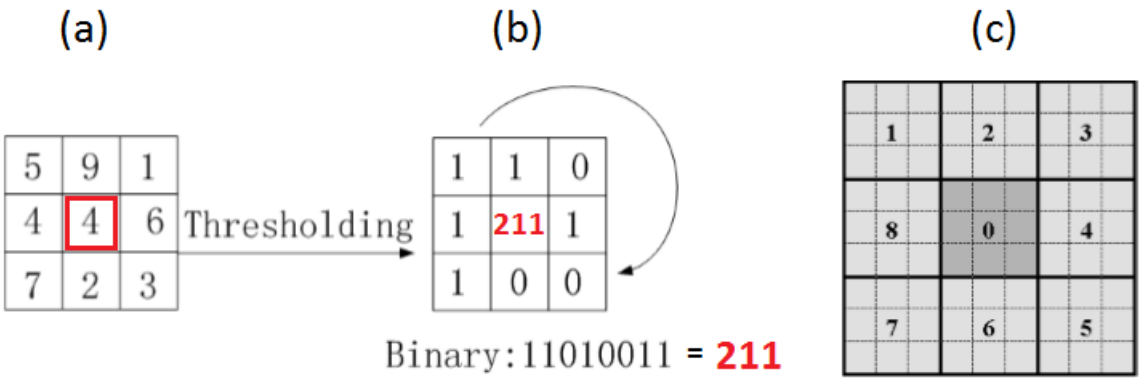
A MB-LBP foi empregada originalmente na detecção facial, juntamente com o classificador AdaBoost.

2.5.1 Funcionamento

A técnica LBP rotula os *pixels* de uma imagem analisando a intensidade dos pixels de uma vizinhança de 8 com o seu *pixel* central, conforme exibido na Figura 27(a). Caso o valor da intensidade do *pixel* vizinho seja maior que o valor do *pixel* central, ele recebe o rótulo 1, ou o rótulo 0, caso contrário. O resultado é uma *string* binária ou seu número decimal correspondente, conforme exibido na Figura 27(b). O histograma dos rótulos pode ser usado como descritor de textura da imagem.

A MB-LBP é uma extensão da LBP, onde em vez de comparar o valor da intensidade de cinza dos pixels da vizinhança com o valor do pixel central, são feitas comparações entre a média da intensidade de cinza dos pixels de sub-regiões com a média de uma sub-região central. Cada sub-região é um bloco, contendo um ou mais

Figura 27 – (a) LBP básico com vizinhança de 8 pixels, compara pixels vizinhos com o pixel central. (b) Caso a intensidade do pixel vizinho é maior do que do pixel central, vale 1, senão vale 0. (c) Operador MB-LBP com 9 sub-regiões (blocos) de vizinhança de 8 pixels. Calcula a intensidade média de cada sub-região e compara com a média da sub-região central. Segue com a mesma lógica do LBP. O resultado final é uma string de 8 bits e seu valor decimal correspondente.



Fonte: Liao et al. (2007).

pixels, conforme exibido na Figura 27(c). O filtro inteiro é formado por 9 blocos de mesmo tamanho, sendo um pixel o tamanho mínimo de cada bloco (LIAO et al., 2007).

Um filtro pode ter o seu tamanho parametrizado com uma variável s, onde s x s é a escala do filtro MB-LBP. Para $s = 3$ tem-se a LBP original com blocos de 1 pixel, valores comuns para um filtro MB-LBP são $s = 9$ e $s = 15$. O cálculo da média da intensidade de cinza de cada bloco é feita de maneira muito eficiente utilizando a imagem integral, mesmo método empregado por Viola e Jones (2004), o que faz da MB-LBP um extrator de características muito rápido.

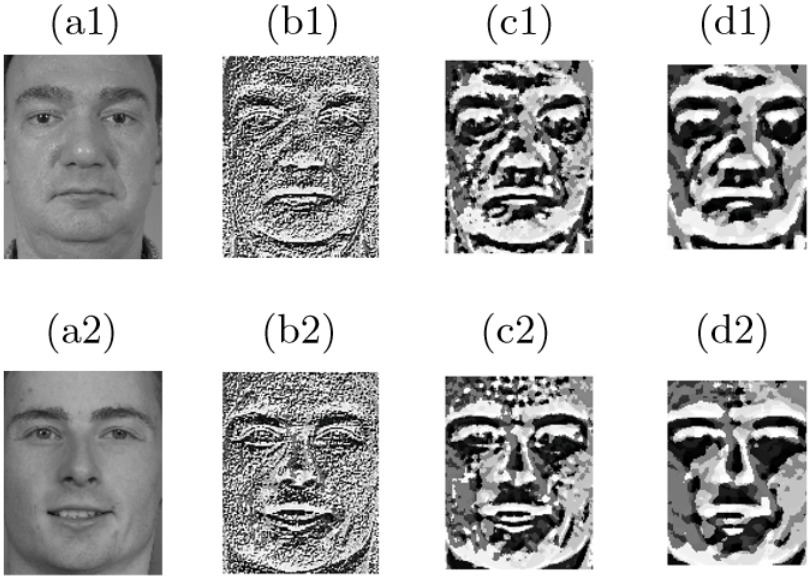
A Figura 28 mostra exemplos de duas imagens de face filtradas com $s = 3$, $s = 9$ e $s = 15$, respectivamente para as Figuras 28(b), (c) e (d). Através deste exemplo é possível perceber a influência que o parâmetro s pode fazer.

Conforme mostrado na Figura 28, filtros com valores de escalas pequenos favorecem a representação de detalhes locais com micro padrões, enquanto valores de escala maiores reduzem efeitos de ruído e fazem uma representação mais robusta provendo informações complementares, o que não é possível com escalas pequenas, porém muitas informações discriminantes podem ser perdidas. Normalmente podem ser usadas várias escalas diferentes e depois combinados os resultados para obter um melhor desempenho.

2.5.2 MB-LBP Estatisticamente Eficaz

É explicado por Liao et al. (2007) que para a LBP melhorar o desempenho na construção do histograma baseado na *string* binária, usa os chamados padrões uniformes que acontecem quando ocorre no máximo duas transições *bit* a *bit* de 1 para 0 ou 0 para 1, e a *string* de 8 *bits* for considerada circular. Na LBP existem 58

Figura 28 – Duas imagens filtradas com MB-LBP. (a) imagens originais. (b) MB-LBP com filtro 3 x 3. (c) MB-LBP com filtro 9 x 9. (d) MB-LBP com filtro 15 x 15.



Fonte: Liao et al. (2007).

intervalos (*bins*) no histograma em direção ao topo que correspondem a estes padrões uniformes.

Esta mesma abordagem não pode ser usada na MB-LBP devido a não ser um cálculo local (para blocos contendo mais de 1 *pixel*), mas sim um cálculo estatístico da média de cada bloco do filtro. Para a MB-LBP foram redefinidos os padrões uniformes para analise estatística a qual foi chamada de *Statistically Effective* MB-LBP (SEMB-LBP) baseada no percentual de distribuição dos níveis de cinza.

Liao et al. (2007) definem que seja $f_s(x, y)$ uma característica MB-LBP de escala s no local (x, y) calculado da imagem original. Um histograma da característica MB-LBP $f_s(., .)$ sobre certa imagem $I(x, y)$ pode ser definido como apresentado na Equação 2.30:

$$H_s(l) = 1_{[f_s(x,y)=l]}, l = 0, ..., L-1 \tag{2.30}$$

onde $1_{(S)}$ é o indicador do conjunto S, e l é o rótulo (*label*) do código MB-LBP. Como a codificação da MB-LBP são *strings* de 8 *bits*, tem um total de $2^8 = 256$ rótulos, assim o histograma tem 256 intervalos. Este histograma contém informações sobre a distribuição das características geradas pela MB-LBP sobre toda a imagem.

As caixas do histograma são ordenadas de acordo com sua porcentagem de ocorrências. Para um filtro LBP 3 x 3, os 58 intervalos superiores são chamados de padrões uniformes, porém para MB-LBP com blocos contendo mais de 1 pixel os últimos 58 intervalos são diferentes do que os obtidos pela LBP.

Para refletir uma aparência uniforme da MB-LBP, Liao et al. (2007) definiram o conjunto SEMB-LBP de escala s conforme a Equação 2.31:

$$SEMB - LBP_s = \{l | Rank[H_s(l)] < N\} \tag{2.31}$$

onde $Rank[H_s(l)]$ é o índice de $H_s(l)$ depois da ordenação decrescente, e N é o número dos padrões uniformes. Para o LBP padrão, $N = 58$, porém para a definição de Liao et al. (2007), N pode ter qualquer valor arbitrário entre 1 e 256. Valores grandes para N, causará grande dimensionalidade de características, enquanto com valores pequenos serão perdidos detalhes. Assim, a adoção do valor 63 para N apresentou-se como o mais equilibrado.

Os padrões restantes foram rotulados com um rótulo simples, sendo que a notação da Equação 2.32 foi utilizada para representar a SEMB-LBP:

$$u_s(x,y) = \begin{cases} Index_s[f_s(x,y)] & se \quad f_s(x,y) \in SEMB - LBP_s \\ N & caso \quad contrário \end{cases} \tag{2.32}$$

onde $Index_s[f_s(x,y)]$ é o índice de $f_s(x,y)$ no conjunto $SEMB - LBP_s$, iniciado em zero.

2.5.3 Aprendizado AdaBoost para MB-LBP

Conforme explicado por Liao et al. (2007), a representação das características através da metodologia SEMB-LBP pode conter muitas informações redundantes. A fim de remover estas redundâncias e construir um classificador efetivo para selecionar apenas as características mais relevantes das faces, utilizou-se um classificador AdaBoost.

O reconhecimento facial é um problema de multi-classes (uma classe para cada indivíduo), enquanto que AdaBoost é um classificador binário. Para resolver esta questão, usou-se um grande conjunto de treinamento com duas classes, uma descrevendo variações intra-pessoal e outra descrevendo variações extra-pessoal.

Uma diferença intra-pessoal ideal seria a diferença entre todos os pixels de duas imagens diferentes terem valor zero (pertencentes ao mesmo indivíduo), e uma diferença extra-pessoal deveria produzir valores de pixels bem maiores (indivíduos diferentes).

Porém no trabalho de Liao et al. (2007), em vez de obter a diferença entre duas imagens, os exemplos de treinamento para o algoritmo de aprendizagem AdaBoost são o conjunto de diferenças entre cada par de histogramas das regiões locais correspondentes. Os exemplos positivos são derivados da diferença entre pares

intra-pessoal e os exemplos negativos são derivados da diferença entre pares extra-pessoais.

Um classificador AdaBoost fraco é ensinado baseado na dissimilaridade entre duas barras de histograma correspondentes. Histogramas para estes padrões são construídos calculando a dissimilaridade: primeiro, uma sequência de m sub-janelas $R_0, R_1, ..., R_{m-1}$ variando a escala e localização são obtidas de uma imagem; em seguida, um histograma é calculado para cada código SEMB-LBP i sobre uma sub-janela R_j conforme Equação 2.33:

$$H_{s,j}(i) = 1_{[u_s(x,y)=i]} . 1_{[(x,y)\in R_j]}, i = 0, ..., N, j = 0, ..., m-1 \quad (2.33)$$

A diferença correspondente entre as barras dos histogramas é definida conforme a Equação: 2.34:

$$D(H^1_{s,j}(i), H^2_{s,j}(i)) = |H^1_{s,j}(i) - H^2_{s,j}(i)| i = 0, ..., N. \quad (2.34)$$

O melhor classificador fraco é aquele que consegue minimizar os pesos das diferenças das barras intra-pessoais e maximizar os pesos das extra-pessoais.

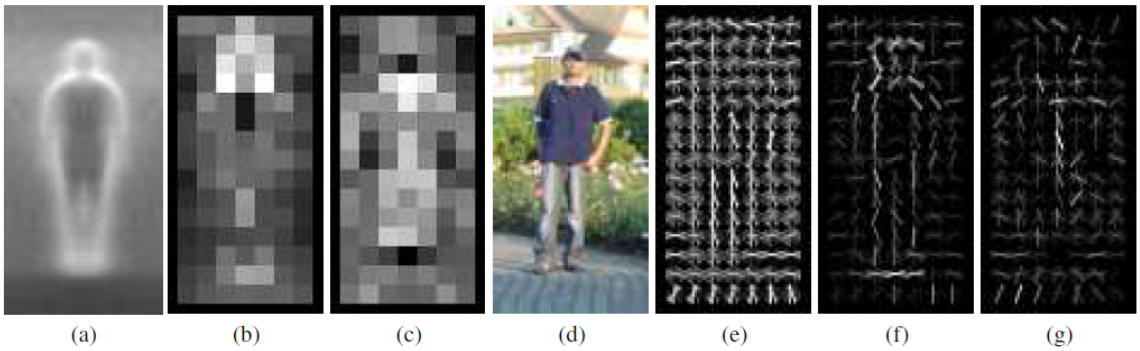
Este esquema de duas classes como procedimento para correspondência de faces funciona da seguinte forma: toma uma imagem de face desconhecida e uma imagem de face da galeria como entrada. Calcula um vetor de característica baseado na diferença a partir das duas imagens e, em seguida, calcula uma pontuação de similaridade para o vetor de características usando o classificador AdaBoost treinado. Finalmente, uma decisão é tomada com base na pontuação, para classificar o vetor de característica na classe positiva (vindo da mesma pessoa) ou na classe negativa (são pessoas diferentes).

2.6 HISTOGRAMA DE GRADIENTES ORIENTADOS

O Histograma de Gradientes Orientados (*Histogram of Oriented Gradients* - HOG), é uma técnica para extração de características que pode ser empregada na detecção de objetos em imagens digitais. A técnica foi desenvolvida por Dalal e Triggs (2005), que a aplicaram na detecção de pessoas utilizando um classificador SVM. Devido a capacidade de generalização da técnica, ela será utilizada neste trabalho para a detecção de veículos.

É essencial produzir um conjunto de características que identifiquem claramente a forma de um objeto (e.g. pessoa, veículo), mesmo sobre fundos desordenados e pouca iluminação. A ideia básica do HOG é que a aparência e forma de um objeto local pode ser caracterizada pela sua distribuição de intensidade dos gradientes ou direção das bordas. Em outras palavras, consegue identificar a silhueta do objeto em contraste com seu plano de fundo. A Figura 29 mostra um exemplo.

Figura 29 – Exemplo utilizando HOG para detectar uma pessoa. (a) Gradiente médio das amostras de treino. (b) Cada pixel mostra o peso positivo máximo da SVM no bloco centrado no pixel. (c) Equivalente a (b) mas para amostras negativas. (d) Uma imagem de teste. (e) Cálculo do descritor R-HOG. (f) Descritor dos pesos positivos da SVM e (g) Descritor dos pesos negativos da SVM.



Fonte: Dalal e Triggs (2005).

Dalal e Triggs (2005) explicam que a implementação do HOG é feita dividindo a imagem em regiões chamadas de células. Para cada célula é calculado o histograma 1-D das direções de gradientes locais ou orientações das bordas sobre os pixels da célula. A representação é formada pela combinação dos histogramas.

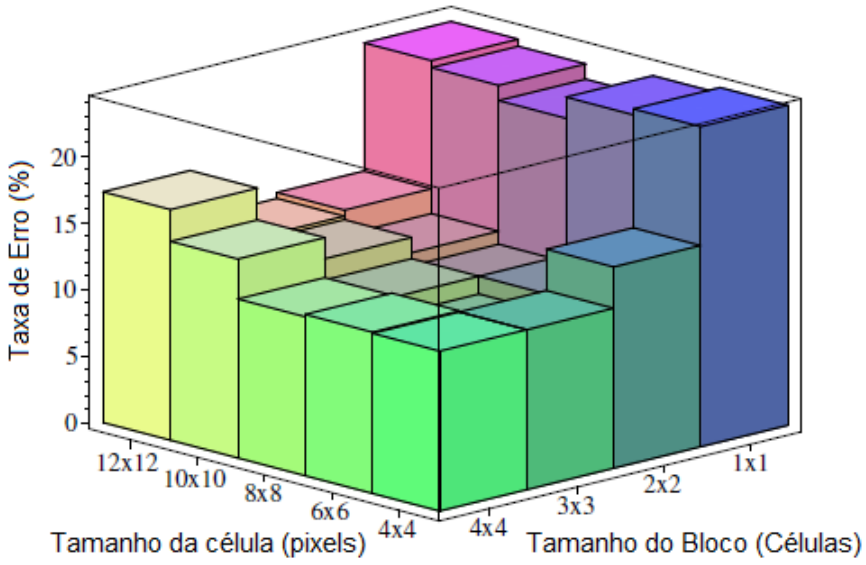
Para melhorar a precisão e diminuir a sensibilidade à variação de luminosidade e sombreamento, é feita a normalização do contraste por uma medida de intensidade de regiões maiores que uma célula, chamadas de blocos. Estes valores são usados para normalizar as células dentro dos blocos.

Primeiro é calculada a orientação do gradiente para cada pixel de uma célula. O gradiente é calculado aplicando-se uma máscara centralizada 1-D de derivada discreta $[-1, 0, 1]$ sobre cada pixel da célula em questão. Isto resulta no cálculo do ângulo do gradiente (sua direção) em relação ao pixel central com os seus pixels vizinhos. Os valores utilizados nos cálculos são a intensidade da cor do pixel, por exemplo, em uma imagem em níveis de cinza, varia de 0 até 255, ou seja, da cor preta até a cor branca respectivamente.

No passo seguinte é calculado para cada pixel, um voto ponderado para a orientação da borda. O voto é uma função da magnitude do gradiente para o pixel, e representa a presença suave ou ausência de borda naquele pixel. Estes votos são acumulados como a frequência de um histograma para cada célula.

As frequências dos histogramas podem estar distribuídas entre $0^0 - 180^0$ graus (o sinal do ângulo do gradiente é ignorado), ou entre $0^0 - 360^0$ graus (o sinal do ângulo do gradiente é considerado). Para detecção de pessoas, blocos de 3 x 3 células com células de 6 x 6 pixels apresentaram melhor desempenho com taxa de erro de 10,4%, conforme exibido na Figura 30.

Figura 30 – Desempenho do detector HOG com SVM em relação ao tamanho dos blocos e suas células. Blocos 3 x 3 com células de 6 x 6 pixels tem melhor desempenho, com taxa de erro de 10,4%.



Fonte: Dalal e Triggs (2005).

As células são agrupadas em blocos conectados para normalização local. Os descritores são os vetores concatenados das frequências dos histogramas das células normalizadas em cada bloco. Estes blocos se sobrepõem, o que faz com que cada célula contribua mais de uma vez para o descritor final (esta sobreposição melhora o desempenho).

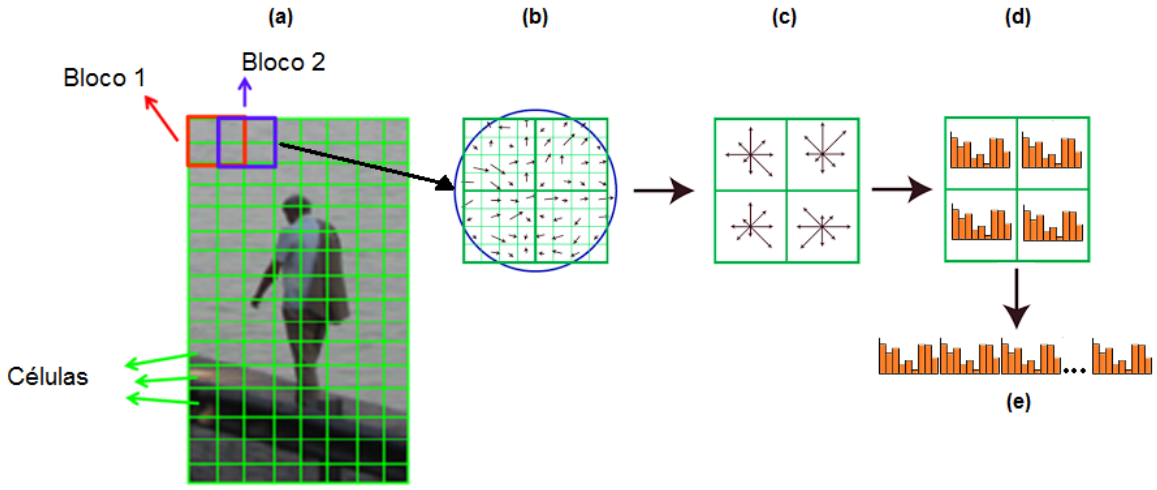
Os blocos podem ser retangulares (R-HOG) ou circulares (C-HOG). Os R-HOG são geralmente grades quadradas representadas por três parâmetros: número de células por bloco; o número de pixels por célula e o número de intervalos de frequência por célula do histograma. As C-HOG são divididas em célula central e célula central com divisão angular, e podem ser descritas com os seguintes parâmetros: o número de compartimento radial e angular; o raio do centro do compartimento; e o fator de expansão para o raio do compartimento radial adicional.

Os descritores mais relevantes encontram-se nas regiões onde ocorre o contraste das bordas da silhueta do objeto com seu fundo. O melhor desempenho obteve-se com uma janela de detecção de 64 x 128 pixels. Como classificador foi usado uma SVM linear.

Considerando uma janela de detecção de 64 x 128 pixels, veja Figura 31(a), com blocos de 4 células de 8 x 8 pixels por célula e com sobreposição de 50% das células (2 células), resulta em 105 blocos. Para cada célula são calculadas as orientações dos gradientes com 9 sub-divisões espaçadas igualmente de 20 em 20 graus ($0^0 - 180^0$), Figura 31(b), formando para cada célula um histograma com as frequências

das orientações e intensidades do gradiente de cada pixel da célula, conforme Figura 31(c).

Figura 31 – (a) Janela de detecção formada por blocos de quatro células com sobreposição de 50%. (b) Orientação e magnitude do gradiente de cada pixel das células de um bloco. (c) Descritores com a direção e a magnitude dos gradientes de cada célula. (d) Histogramas com 9 sub-divisões de frequência representando os descritores para cada célula de um bloco. (e) Concatenação dos histogramas normalizados formando o vetor final de características



Fonte: Elaborado pelo autor, 2017, com base em imagens da web.

Cada frequência acumulada do histograma representa uma característica. Isto resulta em um vetor de características com 3.780 (105 x 4 x 9) posições. O vetor de características é aplicado à uma SVM linear para a detecção das pessoas.

2.7 FILTROS DE CORRELAÇÃO DE *KERNEL*

Conforme explica Henriques et al. (2015), o componente central de um rastreador é um classificador que consegue distinguir entre o objeto alvo sendo rastreado e o ambiente circundante. Um classificador deste tipo normalmente é treinado com fragmentos de amostras escalonadas e transladadas. Tais conjuntos de amostras são repletos de redundâncias, onde qualquer sobreposição de pixels são tratados como idênticos.

Baseado nestas observações, Henriques et al. (2015) propuseram um modelo analítico para conjunto de dados de milhares de fragmentos transladados. Mostraram que a matriz de dados resultante é circulante e que pode ser diagonalizada com a transformada discreta de Fourier[3]. Reduzindo tanto o armazenamento como o custo computacional em várias ordens de grandeza.

[3] Um vasto material sobre Fourier pode ser encontrado em Gonzales e Woods (2011, p. 131)

Notou-se também que a regressão linear desta formulação é equivalente a um filtro de correlação, usado por alguns dos mais rápidos rastreadores competitivos como Hare et al. (2016) e Kalal, Mikolajczyk e Matas (2012). No entanto, para a regressão do *kernel*, Henriques et al. (2015) derivaram um novo filtro de correlação de *kernel* chamado de *Kernelized Correlation Filter* (KCF), que ao contrário de outros algoritmos de *kernel* tem exatamente a mesma complexidade que sua contrapartida linear.

O KCF utiliza HOG para extração das características, e o filtro de *kernel* pode ser aplicado sobre múltiplos canais com custo computacional $O(n \log n)$ devido a transformação rápida de Fourier em vez de álgebra linear custosa.

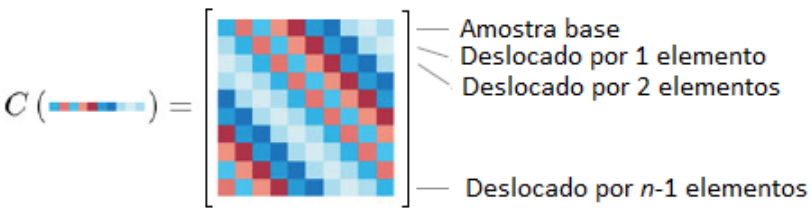
Considerando um vetor $n \text{ x } 1$ representando um fragmento de imagem (também chamado de *patch*) com o objeto de interesse, referenciado aqui como amostra base e denotado por x. O objetivo principal do KCF é treinar um classificador com uma amostra base (exemplo positivo) e várias amostras virtuais obtidas por transformações sobre esta amostra (servirão como exemplos negativos).

Translações uni-dimensionais deste vetor podem ser modeladas por um operador de deslocamento cíclico, o qual pode ser obtido pela matriz de permutação dada em 2.35:

$$P = \begin{bmatrix} 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{2.35}$$

O produto $P_x = [x_n, x_1, x_2, ..., x_{n-1}]^T$ desloca x por um elemento, modelando uma translação pequena. Podem ser encadeados u deslocamentos para atingir uma translação maior usando a potência da matriz $P^u x$. Um u negativo irá deslocar na direção reversa. Um sinal 1D transladado horizontalmente com este modelo é ilustrado na Figura 32.

Figura 32 – Ilustração de uma matriz circulante. As linhas são deslocamentos cíclicos de uma imagem



Fonte: Henriques et al. (2015).

A Figura 33 mostra exemplos de deslocamentos cíclicos verticais de uma amostra de base. A formulação de Henriques et al. (2015) do domínio de Fourier permite treinar um rastreador com todos os possíveis deslocamentos cíclicos de uma amostra de base, tanto vertical como horizontal, sem iterar sobre eles explicitamente.

Figura 33 – Exemplos de deslocamentos cíclicos verticais de uma amostra base 2D

Fonte: Henriques et al. (2015).

Devido ao fato do deslocamento ser cíclico é obtido o mesmo sinal de x periodicamente a cada n deslocamentos. Isto significa que o conjunto completo de deslocamentos é obtido com a Equação 2.36.

$$\{P^u x | u = 0, ..., n-1\}. \tag{2.36}$$

A primeira metade deste conjunto pode ser visto como deslocamentos na direção positiva, e a segunda metade como deslocamentos na direção negativa (veja Figura 33).

A matriz circulante dos deslocamentos das amostras pode ser calculada usando um conjunto da Equação 2.36 como linhas da matriz X conforme Equação 2.37.

$$X = C(x) = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \\ x_n & x_1 & x_2 & \cdots & x_{n-1} \\ x_{n-1} & x_n & x_1 & \cdots & x_{n-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_2 & x_3 & x_4 & \cdots & x_1 \end{bmatrix} \tag{2.37}$$

Um exemplo é dado na Figura 32. Pode ser observado que o padrão é determinístico e especificado pelo vetor gerador x, na primeira linha da matrix.

O fato mais interessante e útil é que todas as matrizes circulantes podem ser diagonalizadas pela transformada discreta de Fourier (DFT), independente do vetor gerador, pela Equação 2.38.

$$X = F \mathrm{diag}(\hat{\mathrm{x}}) F^H, \tag{2.38}$$

onde F é uma matriz constante que não depende de x, e $\hat{\mathrm{x}}$ denota a DFT do vetor gerador, $\hat{\mathrm{x}} = \mathcal{F}(x)$. A matriz F é conhecida como a matriz DFT, e é a única matriz que calcula a DFT para qualquer vetor de entrada, como $\mathcal{F}(z) = \sqrt{n}Fz$. Isto é possível porque a DFT é uma operação linear.

2.7.1 Regressão não Ninear

O objetivo do treinamento é encontrar uma função $f(z) = w^T z$ que minimize o erro quadrático em relação às amostras x_i e seus alvos de regressão y_i. Uma maneira de conseguir funções de regressão não lineares mais poderosas é com o chamado truque de *kernel*.

Um ponto atrativo disto é que o problema de otimização é linear, embora em um conjunto diferente de variáveis, conhecido também como espaço dual. Como ponto negativo, avaliar $f(z)$ normalmente cresce em complexidade com o número de amostras. Porém, Henriques et al. (2015) afirmam superar esta limitação e obtiveram filtros não lineares tão rápidos quanto filtros de correlação linear, tanto para treinamento como para avaliação.

Mapear as entradas de um problema linear para um espaço de características não linear $\varphi(x)$ com dica de *kernel* consiste em:

a) expressar a solução w como uma combinação linear das amostras pela Equação 2.39.

$$w = \sum_i \alpha_i \varphi(x_i). \tag{2.39}$$

   As variáveis para otimização são α, as quais são ditas estarem em um espaço dual.

b) escrevendo o algoritmo em termos de produto escalar $\varphi^T(x)\varphi(x^{'}) = k(x, x^{'})$, que é calculado usando a função de *kernel* k (por exemplo, Gaussiana ou polinomial).

Os produtos escalares entre todos os pares de amostras são normalmente armazenados em uma matriz de *kernel* k de $m \times n$, com os elementos calculados pela Equação 2.40:

$$K_{i,j} = k(x_i, x_j). \tag{2.40}$$

O poder da dica de *kernel* vem do uso implícito de um espaço de características com alta dimensionalidade $\varphi(x)$, sem sempre instanciar um vetor neste espaço. Porém, isto também é sua fraqueza, sendo que a complexidade das funções de regressão crescem com o número de amostras, conforme Equação 2.41:

$$f(z) = w^T z = \sum_{i=1}^{n} \alpha_i k(z, x_i), \tag{2.41}$$

sendo que estas desvantagens podem ser superadas utilizando dados circulantes.

2.7.2 Regressão Rápida de *Kernel*

A solução da versão de regressão de *kernel*, conforme explicado por Henriques et al. (2015), é dada pela Equação 2.42.

$$\alpha = (K + \lambda I)^{-1} y, \tag{2.42}$$

onde α é o vetor de coeficientes α_i, que representa a solução no dito espaço dual e K é a matriz de *kernel*. O λ é um parâmetro de regularização para controlar *overfitting* no treinamento e evitar divisão por zero, I é a matriz identidade e y é o alvo de regressão. Se K for circulante para conjuntos de deslocamentos cíclicos, a Equação 2.42 pode ser diagonalizada e pode ser obtido uma solução rápida, semelhante à de um caso linear.

Entretanto é necessário impor uma condição para garantir que K seja circulante. Pois o mapeamento arbitrário não linear $\varphi(x)$ não dá garantia de preservar qualquer tipo de estrutura. Assim, tendo dados circulantes $C(x)$, a matriz de *kernel* correspondente K é circulante se a função de *kernel* satisfaz $k(x, x^{'}) = k(Mx, Mx^{'})$, para qualquer matriz de permutação M.

Isto significa que o *kernel* precisa tratar todas as dimensões de dados igualmente para preservar a estrutura circulante. Felizmente a maioria dos *kernels* úteis atendem esta condição, como por exemplo:

a) *kernels* de funções de bases radias (por exemplo: Gaussiana);

b) *kernels* de produto escalar (por exemplo: linear, polinomial);

c) *kernels* aditivos exponenciais.

Conhecendo o *kernel* que pode ser usado para fazer K circulante, é possível diagonalizar a Equação 2.42, como em casos lineares e obter a Equação 2.43.

$$\hat{\alpha} = \frac{\hat{y}}{\hat{k}^{xx} + \lambda}, \tag{2.43}$$

onde $\hat{k}^{xx}$ é a primeira linha da matriz de *kernel* $K = C(k^{xx})$, e um chapéu ˆ denota a DFT de um vetor.

2.7.3 Detecção Rápida

Denotado por K^z tem-se a matriz de *kernel* entre todas as amostras de treinamento e todos os candidatos de fragmento. Desde que as amostras e os fragmentos sejam deslocamentos cíclicos da amostra base x e do fragmento base z, respectivamente, cada elemento de K^z é dado por $k(P^{i-1}z, P^{j-1}x)$. Esta matriz de *kernel* satisfaz a condição necessária para dados circulantes.

É necessário apenas a primeira linha para definir a matriz de *kernel*, conforme Equação 2.44:

$$K^z = C(k^{xz}), \tag{2.44}$$

onde k^{xz} é o *kernel* de correlação de x e z. A partir da Equação 2.41 é possível calcular a função de regressão para todos os candidatos a fragmentos pela Equação 2.45:

$$f(z) = (K^z)^T \alpha. \tag{2.45}$$

Pode ser notado que $f(z)$ é um vetor contendo a saída para todos os deslocamentos cíclicos de z, ou seja, a resposta de detecção completa. Para calcular a Equação 2.45 com mais eficiência ela pode ser diagonalizada pela Equação 2.46:

$$\hat{f}(z) = \hat{k}^{xz} \odot \hat{\alpha}. \tag{2.46}$$

A avaliação de $f(z)$ em todos os locais pode ser visto como uma operação de filtragem espacial sobre os valores de *kernel* k^{xz}. Cada $f(z)$ é uma combinação linear dos valores de *kernel* vizinhos de k^{xz}, ponderados pelos coeficientes aprendidos de α. Como isto é uma operação de filtragem, ela pode se calculada com mais eficiência no domínio de Fourier.

2.7.4 *Kernel* Gaussiano com Função de Base Radial

Um *Kernel* de função de base radial (RBF) tem a forma $k(x, x^{'}) = h(||x - x^{'}||^2)$, para alguma função h. Um caso especial é o chamado *kernel* Gaussiano com a forma $k(x, x^{'}) = exp\left(-\frac{1}{\sigma^2}||x - x^{'}||^2\right)$, que é obtido pela Equação 2.47:

$$k^{xx^{'}} = exp\left(-\frac{1}{\sigma^2}\left(||x||^2 + ||x^{'}||^2 - 2\mathcal{F}^{-1}(\hat{x}^* \odot \hat{x}^{'})\right)\right). \tag{2.47}$$

É possível calcular a correlação completa do *kernel* em tempo $O(n \log n)$.

Trabalhar no domínio de características chamado dual tem a vantagem de permitir múltiplos canais, tais como a orientação das barras do descritor HOG, simplesmente somando sobre eles no domínio de Fourier.

Assumindo que um vetor x concatena vetores individuais para C canais (por exemplo, como as barras do histograma com as orientação dos gradientes do descritor HOG), como $x = [x_1, ..., x_C]$.

Normalmente os *kernels* são baseados ou em produto escalar ou na norma dos argumentos. O produto escalar pode ser calculado simplesmente somando os produtos escalares individuais de cada canal. Pela linearidade da DFT, isto permite somar o resultado de cada canal no domínio de Fourier. Como exemplo concreto, isto

pode ser aplicado no *kernel* Gaussiano, obtendo multicanal análogo à Equação 2.47, pela Equação 2.48.

$$k^{xx'} = exp\left(-\frac{1}{\sigma^2}\left(||x||^2 + ||x'||^2 - 2\mathcal{F}^{-1}\left(\sum_c \hat{x}_c^* \odot \hat{x}_c'\right)\right)\right). \tag{2.48}$$

Vale ressaltar que a integração de múltiplos canais não resulta em um problema de inferência mais difícil, simplesmente são somados os canais ao calcular a correlação do *kernel*.

2.7.5 Modelo de Implementação do Rastreador KCF

Uma implementação do rastreador KCF utilizando um *kernel* Gaussiano multicanais foi proposta por Henriques et al. (2015) conforme descrito abaixo:

a) fazer o treinamento dos fragmentos de imagens para treinamento utilizando a Equação 2.43;

b) detectar o alvo utilizando a Equação 2.46;

c) calcular o *kernel* de correlação com a Equação 2.48.

Onde os fragmentos para treinamento e de teste para detecção tem dimensões m x n x c e o alvo de regressão tem dimensões m x n.

No primeiro quadro de um vídeo de testes é treinado um modelo com o fragmento da imagem na posição inicial do objeto alvo. Este fragmento deve ser maior do que o alvo a ser rastreado para obter algum contexto de rastreabilidade. Em cada novo quadro é detectado sobre o fragmento na posição anterior, e a posição do alvo é atualizada para a nova posição com máximo valor de correspondência. Finalmente, é treinado um novo modelo na nova posição, e assim por diante.

O fator mais importante para o alto desempenho e velocidade acima de 100 quadros por segundo do KCF, relata Henriques et al. (2015), é a incorporação de milhares de amostras negativas produzidas a partir do ambiente alvo com custo computacional muito baixo, próximo de $O(n \log n)$. A velocidade do rastreador é diretamente proporcional ao tamanho da região a ser rastreada.

O KCF é bem susceptível a mudanças de iluminação, oclusões e não é afetado por irregularidades do fundo das imagens, o que geralmente afeta severamente outros rastreadores. Isto deve-se ao fato de produzir milhares de amostras negativas em torno da região alvo. Porém, uma fraqueza reportada por Henriques et al. (2015) é a falta de um mecanismo de recuperação de falhas.

Pelo alto desempenho e velocidade de processamento do algoritmo de rastreamento KCF, ele parece ser muito promissor para o rastreamento de veículos. Em vez

de utilizar um algoritmo lento para detectar os veículos em cada quadro, pode ser detectado os veículos em um quadro inicial, criar rastreadores KCF para rastrear as ROIs dos veículos detectados, manter o rastreamento por vários quadros, e só depois de alguns quadros decorridos, fazer uma nova detecção para atualizar os rastreadores, que continuam rastreando as novas ROIs detectadas.

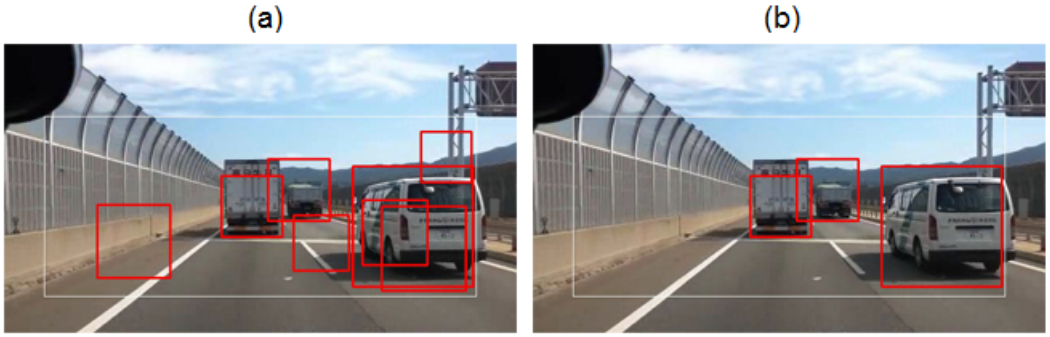
Isto vai ser mais rápido do que a detecção em cada quadro, e sem perder quase nenhuma precisão na detecção. Uma questão que precisa ser resolvida é a falta de um mecanismo de recuperação de falhas no rastreador KCF.

2.8 GERAÇÃO DE HIPÓTESE E VERIFICAÇÃO DE HIPÓTESES

Conforme Mukhtar, Xia e Tang (2015), usualmente são empregados dois estágios chaves para a detecção de veículos em vídeo, um chamado de Geração de Hipótese (GH) e o outro chamado de Verificação de Hipótese (VH).

A GH objetiva identificar Regiões de Interesse (*Region Of Interest* - ROI) no quadro do vídeo processado, como sendo prováveis regiões candidatas a veículos. Geralmente neste estágio são aplicados algoritmos rápidos e com alta taxa de detecção, porém com elevado número de falso positivos, como mostrado na Figura 34(a).

Figura 34 – (a) Exemplo de detecção no estágio de GH utilizando a técnica de Viola e Jones (2004). Observa-se cinco ROIs com falso positivos. (b) Exemplo de aplicação da fase VH apenas sobre as ROIs detectadas em (a) com o algoritmo HOG com SVM. Percebe-se a eliminação das cinco ROIs com falso positivos de (a).



Fonte: Elaborado pelo autor, 2017.

Por outro lado, no estágio de VH são verificadas cada ROI candidata obtida na GH, se realmente é um veículo. O algoritmo aplicado no estágio de VH geralmente possui alta precisão na detecção, eliminando assim, os falso positivos deixados pela GH, o que pode ser visto na Figura 34(b), porém seu custo para processamento geralmente é mais elevado do que na GH. A combinação dos dois estágios proporciona

equilíbrio entre precisão na detecção com poucos falso positivos, e com custo de processamento admissível para aplicações cotidianas.

Neste trabalho é proposta uma nova técnica para VH, chamada de Confiança dos Centroides das Linhas Verticais e Horizontais (CCLVH). A técnica é utilizada juntamente com outras técnicas como Viola e Jones (2004) e Liao et al. (2007) a fim de fazer uma verificação dupla melhorando a confiança da verificação de hipótese, sem comprometer muito o tempo de processamento.

2.9 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Como segmentador de bordas, a técnica mais robusta é o detector de bordas de Canny, pela sua precisão e por ser rápido computacionalmente. Ainda Gonzales e Woods (2011) afirmam que os resultados obtidos com o detector de bordas de Canny, tem feito dele a técnica ideal para a detecção de bordas.

Aplicar o seguidor de bordas de Suzuki (1985) sobre as bordas detectadas com Canny, permitirá extrair com precisão as coordenadas de cada segmento das bordas. Com isto será possível separar as linhas verticais das horizontais para poder utilizar no algoritmo CCLVH.

A transformada de Hough também é uma alternativa promissora, principalmente por ser rápida, porém exige que as linhas verticais e horizontais encontradas sejam separadas, sem contar que encontrar uma parametrização estável que extraia apenas as linhas retas verticais e horizontais, linhas de curvas suaves e descontinuidades não é simples.

A combinação dos extratores de características Haar-like e MB-LPB juntamente com o classificador AdaBoost, e a combinação do extrator de característica HOG com o classificador SVM, são promissores para empregar em um sistema de detecção de veículos devido a eficiência na extração das características e da capacidade de generalização dos classificadores.

A alta velocidade e precisão apresentadas para a técnica de rastreamento KCF, que pode chegar a mais de 100 quadros por segundo, faz desta técnica a opção ideal para ser usada no rastreamento dos veículos.

Por fim, aplicar as técnicas de geração e verificação de hipótese diminuem significativamente a quantidade de veículos falso positivos. O rastreamento de veículos tem sua velocidade de processamento reduzida dependendo da quantidade e tamanho das ROIs rastreadas, por este motivo, quanto menos falso positivos forem rastreados, mas rápido o sistema será.

3 TRABALHOS RELACIONADOS

Tomando como base o foco desta pesquisa no desenvolvimento de um sistema de notificação de risco de colisão veicular, foram realizadas pesquisas com o intuito de encontrar trabalhos que envolvessem esta temática. As pesquisas iniciaram analisando revisões sistemáticas (RS) relacionadas a temática a fim de encontrar os principais trabalhos e consequentemente as principais técnicas e metodologias empregadas para resolver este tipo de problema.

Na análise das RS, percebeu-se grande variedade de técnicas e metodologias empregadas, porém apresentando poucas informações quantitativas das técnicas empregadas e seus resultados. Apresentam nas sua maioria apenas informações descritivas e representativas, o que se não for analisado minuciosamente pelo pesquisador, co-relacionado as RS, pode levá-lo a conclusões equivocadas quanto a escolha das técnicas e metodologias mais maduras e que produzem os melhores resultados.

Visando obter uma visão geral e quantitativa das principais técnicas e metodologias empregadas na temática deste trabalho, e guiado pelo estudo co-relacionado das RS analisadas, foi desenvolvido um estudo em forma de mapeamento sistemático (MS), seguindo as propostas de Petersen et al. (2008) e Bailey et al. (2007), o qual será discutido na sequência.

3.1 REVISÃO SISTEMÁTICA E MAPEAMENTO SISTEMÁTICO

Inspirados nas pesquisas médicas, Petersen et al. (2008) explicam que os MS e as RS são trabalhos secundários que estudam um conjunto de trabalhos primários em uma área de pesquisa, porém de pontos de vista diferentes. Ambos objetivam levantar informações do estado da arte de uma área de pesquisa, sendo fonte de embasamento para elaboração de novas pesquisas na área em questão.

As RS fazem um estudo mais aprofundado em um tópico de pesquisa, exigindo mais esforço na leitura de um grupo mais seleto de trabalhos primários. Como resultado descreve as principais técnicas e metodologias utilizadas, apresenta os trabalhos mais relevantes e seus resultados (PETERSEN et al., 2008).

Já os MS dão uma visão geral de uma área de pesquisa. Fazem um estudo em largura, analisando geralmente uma quantidade maior de trabalhos, porém sem aprofundamento nos detalhes técnicos (apenas leitura do *abstract* dos trabalhos primários). Tem como principais resultados a classificação e categorização da área e a frequência das publicações em cada categoria de forma visual (BAILEY et al., 2007).

Uma boa prática indicada por Petersen et al. (2008), é de realizar MS como primeiro passo na elaboração de uma RS. Neste trabalho foram analisadas as seguintes RS: Mukhtar et al. (2013); Sivaraman e Trivedi (2013b); Wu et al. (2012); Saini (2014); Hadi, Sulong e George (2014); Tian et al. (2017); Sreevishakh e Dhanure (2015) e Mukhtar, Xia e Tang (2015).

Normalmente os MS não se aprofundam muito nos trabalhos primários analisados, devido a darem apenas uma visão geral e estrutural da área de pesquisa. Porém, como foi percebido na análise das RS uma grande variedade de técnicas e metodologias empregas e com poucos resultados quantitativos, foi necessário uma analise mais detalhada das técnicas empregadas e seus resultados. Para isto fez-se necessário a leitura de praticamente todo o conteúdo dos trabalhos primários analisados.

3.2 DESENVOLVIMENTO DO MAPEAMENTO SISTEMÁTICO

O desenvolvimento do mapeamento sistemático iniciou-se com a elaboração de perguntas a fim de guiar o mapeamento na temática da pesquisa. As seguintes perguntas de pesquisa foram elaboradas para este trabalho:

a) existem quantidades consideráveis de publicações nos últimos anos relacionadas à detecção e rastreamento de veículos utilizando sistema óptico?

b) entre os trabalhos existentes, há trabalhos que notificam o motorista sobre o risco de colisão?

c) quais são as técnicas mais empregadas e com melhores resultados para detecção e rastreamento de veículos com sistema óptico?

Ao final do processo de mapeamento deve ser possível responder de forma consistente com dados quantitativos cada uma destas perguntas.

O mapeamento é realizado com base na análise de publicações primárias relacionadas com o tema da pesquisa deste trabalho. Estas publicações serão buscadas em mecanismos de busca acadêmicos (MBAs). Para efetuar a busca é necessário definir uma expressão de busca com palavras chaves, de forma que os MBAs tragam apenas trabalhos relevantes que respondam as perguntas da pesquisa elaboradas anteriormente. A seguinte expressão de busca foi definida:
(*vehicle* OR *car*) AND (*detect** OR *recogni** OR *track*) AND (*vision* OR *system* OR CAS OR DAS OR ITS) AND (*road* OR *traffic* OR *highway*) AND (*collision* OR *crash*) AND (*safety* OR *avoid**).

Considerando o período entre 2012 a 2016 e utilizando-se de quatro MBAs: ACM Digital Library [1], IEEE Xplore [2], Science Direct [3] e Periódicos da CAPES [4]. Foi optado por estes MBAs devido a avaliação de Buchinger, Cavalcanti e Hounsell (2014) e pela possibilidade de serem acessados de forma gratuita através da VPN da UDESC.

Com as propriedades de busca definidas, efetuaram-se as consultas nos MBAs no mês de junho de 2016. Foram encontradas 763 publicações com a seguinte distribuição por MBA: 19 em ACM Digital Library, 534 em IEEE Xplore, 71 em Science Direct e 139 nos Periódicos da CAPES.

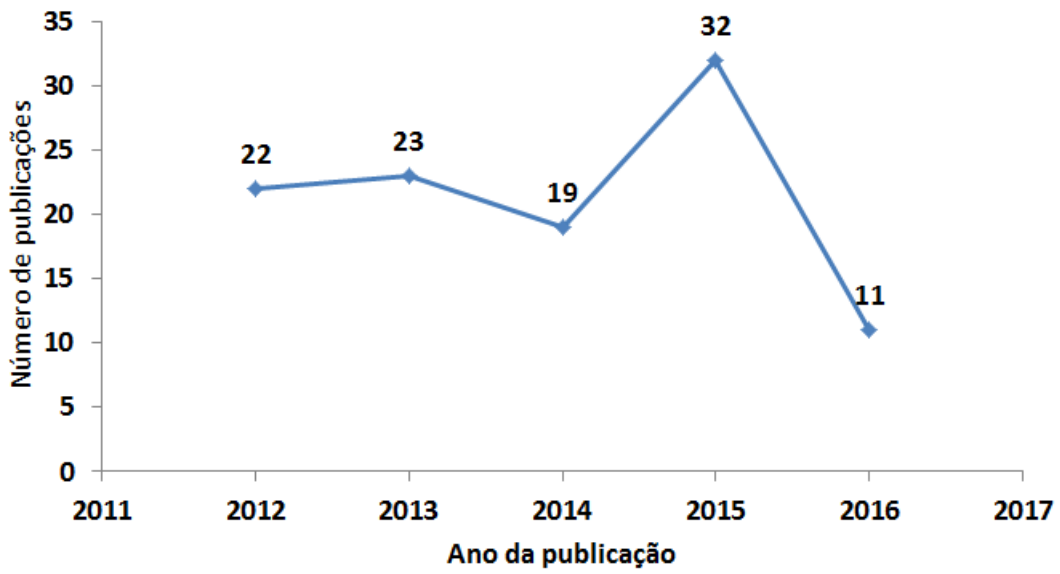
O número elevado de publicações encontradas normalmente possuem uma série de ruídos, isto é, publicações que entraram nos resultados, porém não respondem as perguntas da pesquisa (não correspondem com a temática da pesquisa). A fim de remover estes trabalhos indesejados, são aplicados critérios de inclusão e exclusão. Os critérios de inclusão foram: estar dentro do contexto da pesquisa; possuir as palavras chaves no *abstract* e título do trabalho e novidades sobre a temática. Já os critérios de exclusão foram: estar fora do contexto da pesquisa; ano de publicação anterior a 2012; trabalhos em duplicatas; não ser trabalho primário; e idioma não for inglês.

Na maioria dos trabalhos foram analisados o título, o *abstract* e as palavras chaves (*keywords* ou *index terms*) das publicações, sendo o *abstract* o mais relevante. Porém, em muitos casos foi necessário ler outras partes do texto para poder identificar se o contexto do trabalho estava de acordo com os objetivos da pesquisa, isto devido a muitos *abstract* dos trabalhos serem pobres de conteúdo, conforme já adverte Petersen et al. (2008). Após aplicar os critérios de inclusão e exclusão restaram 107 publicações relevantes à pesquisa, conforme pode ser visto pela Figura 35.

Várias características foram investigadas nas publicações e inseridas em uma planilha eletrônica a fim de facilitar o levantamento quantitativo. As principais características levantadas foram: Ano da publicação, Precisão (%), Velocidade (Quadros Por Segundo - fps), Técnica de Detecção, Técnica de extração de características, Técnica de rastreamento, Detecção de Dia e/ou à Noite, Faz contagem de veículos, Estima a distância entre os veículos, Estima a velocidade, Estima a direção, Tipo de objeto detectado (Veículo, Moto, Bicicleta, Pedestre e Animal), Faz calibração da câmera, Usa câmera monocular, Usa câmera estéreo, Trata oclusões, Tipo do sistema (*Collision*

[1] Disponível em: <http://dl.acm.org>
[2] Disponível em: <http://ieeexplore.ieee.org>
[3] Disponível em: <http://www.sciencedirect.com>
[4] Disponível em: <http://www.periodicos.capes.gov.br>

Figura 35 – Mapeamento sistemático aplicado sobre 107 publicações, entre os anos de 2012 a 2016



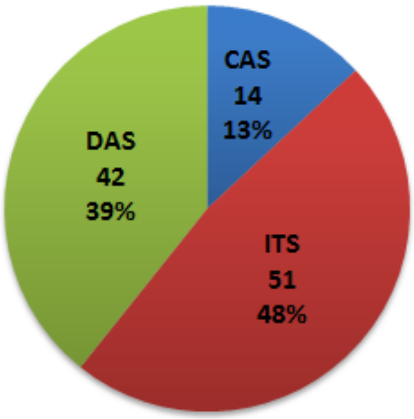
Fonte: Elaborado pelo autor, 2017.

Avoidance System (CAS), *Driver Assistance System* (DAS) e *Intelligent Transportation System* (ITS)).

Todas as características foram investigadas em cada publicação, as características não encontradas em alguma publicação foram deixadas em branco na planilha a fim de não influenciar nos resultados gerais e médias calculadas.

A seguir são expostos alguns dos principais resultados obtidos com o mapeamento sistemático. Conforme mostra a Figura 36, a maioria das publicações são

Figura 36 – Publicações por tipo de sistema.



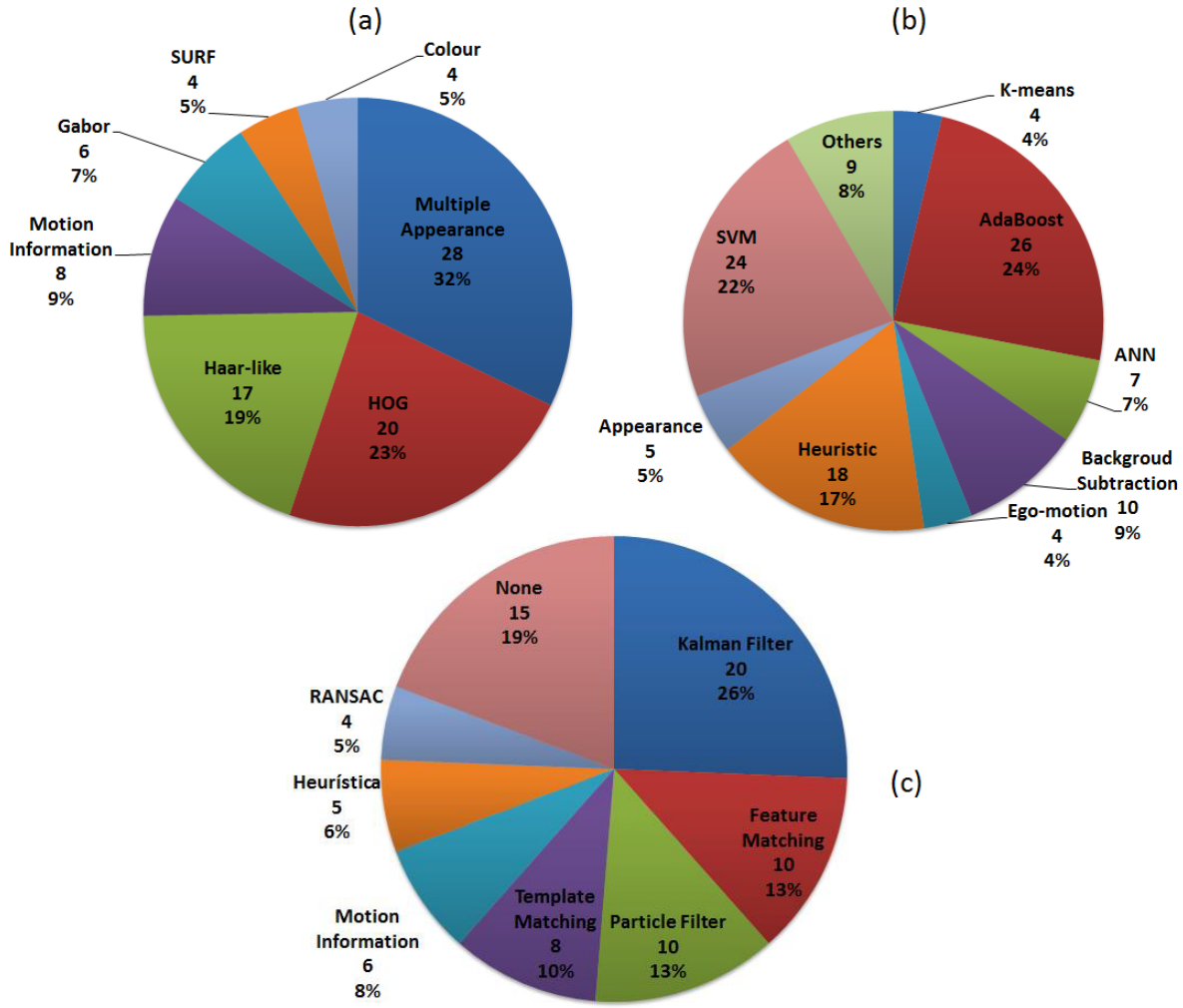
Fonte: Elaborado pelo autor, 2017.

empregadas nos sistemas ITS e DAS e apenas 13% em sistemas CAS, o que indica a

necessidade de mais trabalhos nesta área, o que é o caso deste trabalho de pesquisa proposto.

A Figura 37(a) mostra maior uso dos extratores de características HOG e Haar-like. *Multiple Appearence* é um agrupador de várias técnicas com heurísticas diferentes. A Figura 37(b) indica que os dois classificadores mais utilizados para a detecção de veículos são AdaBoost e SVM, os quais possuem dualidade com os extratores de características Haar-like e HOG respectivamente. Já a Figura 37(c) mostra que as as principais técnicas para rastreamento de veículos empregadas são Kalman *Filter*, *Feature Matching* e *Particle Filter*. Também é possível verificar que 19% dos trabalhos não fazem rastreamento dos veículos.

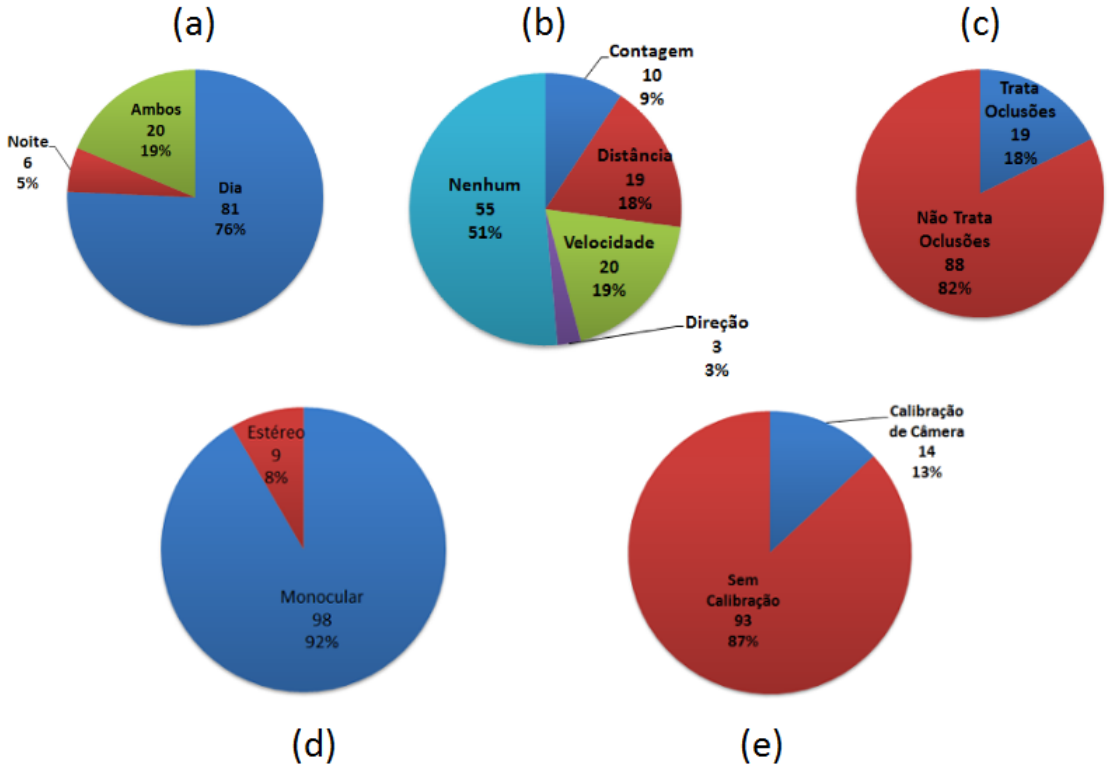
Figura 37 – (a) Publicações por extrator de características. (b) Publicações por classificador. (c) Publicações por rastreador



Fonte: Elaborado pelo autor, 2017.

A Figura 38 mostra uma série de informações interessantes para identificar características que estão sendo exploradas e as que são pouco exploradas. Como por exemplo, poucos trabalhos explorando: detecção à noite, que estimam a direção dos veículos, Tratam oclusões, uso de câmera estéreo e com calibração de câmera, como mostram as Figuras 38(a)(b)(c)(d) e (e), respectivamente.

Figura 38 – Publicações por: (a) Ambiente de iluminação. (b) Comportamentos. (c) Tratamento de oclusões. (d) Tipo de câmera. (e) Calibração de câmera



Fonte: Elaborado pelo autor, 2017.

A Tabela 2 mostra os resultados relatados nas publicações quanto as técnicas de detecção e as suas respectivas técnicas de extração de características. Para a precisão e o desempenho (quarta e quinta colunas respectivamente) foi calculado o valor médio dos valores extraídos das publicações que apresentam estes valores (terceira coluna), e as duas últimas colunas apresentam a aplicação durante o dia e/ou durante à noite respectivamente. As linhas destacadas mostram as informações avaliadas como sendo as mais relevantes do ponto de vista deste trabalho.

A combinação da técnica de detecção *Heurisitc* (heurística) com a técnica de extração de características *Brightness* (brilho) obteve o melhor desempenho de precisão e velocidade de processamento. No entanto, esta combinação de técnicas é empregada apenas em ambiente noturno conforme: (Hajimolahoseini, Soltanian-Zadeh e Amirfattahi (2014), Choi et al. (2014) e Zhou et al. (2013). Seu desempenho elevado

Tabela 2 – Resultados quantitativos por técnica de detecção e extração de características

Téc. Detecção	Téc. Extração	Nº Pub.	Prec. (%)	Vel. (fps)	D	N
Heuristic	*Brightness*	3	97,31	61,94	1	3
	Multiple Appearance	10	88,99	37,61	9	7
SVM	**HOG**	**10**	**93,22**	**18,78**	**10**	**1**
	Multiple Appearance	5	87,17	14,22	5	-
	SURF	2	70	25	2	-
AdaBoost	HOG	5	92,32	12	5	1
	Haar-like	**14**	**91,66**	**18,95**	**13**	**4**

Fonte: Elaborado pelo autor, 2017.

deve-se ao fato de precisar basicamente diferenciar três situações: grandes regiões de pixels com baixa intensidade de níveis de cinza (escuridão), pequenas regiões com alta intensidade de níveis de cinza (luzes) e a simetria entre elas.

Outro item em destaque, é a combinação da detecção com o classificador SVM, juntamente com o extrator de características HOG. Esta combinação apresenta desempenho médio inferior a apresentada anteriormente, porém é amplamente aplicada durante o dia (Guo et al. (2015), Khairdoost, S e Jamshidi (2013), Arrospide, Salgado e Nieto (2012), etc.), onde a complexidade de processamento é superior se comparado com à noite devido a possuir mais objetos visíveis na cena que precisam ser identificados.

O último item em destaque, e também amplamente utilizado durante o dia (Rezaei, Terauchi e Klette (2015), Gu et al. (2015), Li et al. (2013), Sivaraman e Trivedi (2013a), etc.), é a combinação do classificador AdaBoost com o extrator de características Haar-like. Esta combinação apresenta precisão e velocidade de processamento semelhante a combinação SVM com HOG.

Os classificadores SVM e AdaBoost e os extratores de características HOG e Haar-like estão em pelo menos 50% das publicações, apresentaram desempenho médio acima de 91,5%, velocidade média acima de 18,5 fps e são aplicados durante o dia. Isto faz destas técnicas as alternativas mais indicadas para serem aplicadas neste trabalho de pesquisa.

O mapeamento sistemático desenvolvido para este trabalho de dissertação de mestrado pode ser acessado na íntegra no seu endereço eletrônico [5].

[5] <https://drive.google.com/file/d/0B9DOjnXN1LOBZDdUSHhHQ3hwNGc/view?usp=sharing>

3.3 REVISÃO DOS TRABALHOS ENCONTRADOS

Tendo em vista a grande quantidade de publicações primárias encontradas no mapeamento sistemático discutido anteriormente, serão revisados os principais trabalhos que serviram como base em termos de técnicas de detecção e extração de características, rastreamento, desempenho, estimativa de distância entre veículos e notificação de risco de colisão veicular.

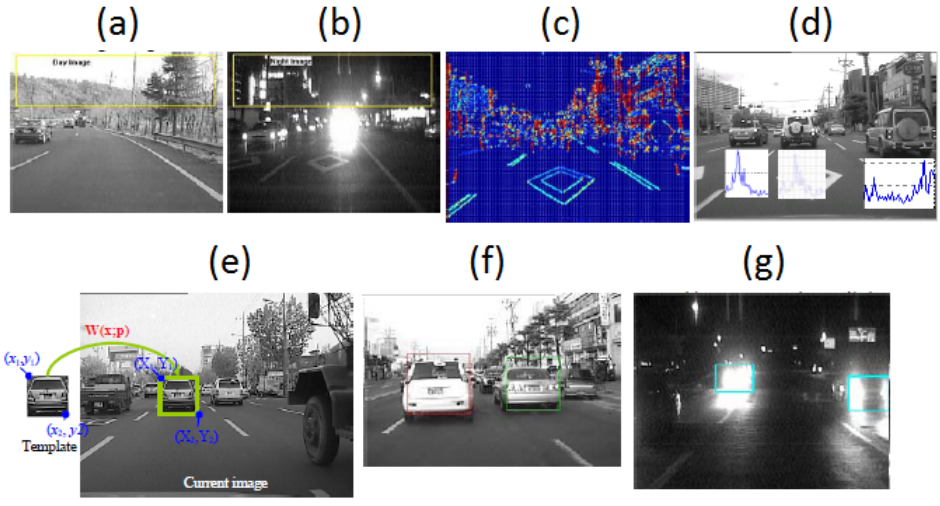
A fim de melhorar o entendimento, facilitar a comparação e padronizar os pontos mais relevantes das publicações, todas as revisões estão estruturadas com os mesmos tópicos, sendo eles: Autores e Título do trabalho, Objetivos principais, Técnicas de GH, Técnicas de VH, Rastreamento, Estimativa de distância, Taxa de detecção, Tempo de processamento, Alertas e Observações.

- Kim et al. () - *Front and Rear Vehicle Detection and Tracking in the Day and Night Times Using Vision and Sonar Sensor Fusion*:

- Objetivos principais: Detectar e rastrear veículos na frente e atrás do veículo portando o sistema, tanto de dia como de noite, utilizando câmera para longe e sensor de sonar para perto.

- Técnicas de GH: Primeiro determina se é de dia ou de noite. Para isto calcula a média da intensidade dos pixels do topo de uma série de quadros consecutivos, caso esta média seja maior do que um determinado limiar, é considerado dia, do contrário é noite (Figura 39 (a) e (b)). Com base nisto utiliza métodos diferentes para detectar os veículos de dia ou de noite.

Figura 39 – (a) e (b) Determinação da condição de luminosidade. (c) Mapa de ângulo de bordas. (d) Taxa de simetria. (e) Rastreamento por modelo. (f) Detecção de dia. (g) Detecção à noite



Fonte: Kim et al. ().

Durante o dia, a detecção das regiões candidatas a veículos é feita pela sombra debaixo dos veículos. Aplica equalização do histograma para equilibrar a intensidade das regiões mais claras e mais escura da estrada, a fim de favorecer a detecção das sombras.

Depois é criada uma imagem binária com limiar baixo para eliminar as regiões claras. A sombra é detectada quando a intensidade dos pixels de uma determinada região é menor do que um certo limiar.

A detecção de veículos à noite é feita pela análise do brilho gerado pela luzes dos veículos. A intensidade dos pixels nas regiões das luzes é maior do que das outras regiões.

- Técnicas de VH: Para a validação durante o dia, usa operadores de Sobel para gerar um mapa de ângulos das bordas (Figura 39 (c)) e extrair a simetria dos limites esquerdo e direito do veículo. A região onde a sombra foi detectada é escaneada para cima a fim de encontrar as delimitações.

São analisadas as duas linhas verticais nas laterais esquerda e direita da região candidata a veículo, através de histogramas das bordas (Figura 39 (d)). Se os histogramas verticais em ambos os lados excederem determinada proporção da largura de um veículo (em pixels), a região candidata é válida.

Para validar durante à noite, as regiões das luzes são analisadas em pares e de acordo com suas formas (Figura 39 (g)). Onde, pequena: luz traseira e de freio; grande: luz refletida por outro veículo; e enorme: luz do farol dianteiro.

- Rastreamento: Detectar veículos pela sombra é custoso computacionalmente e às vezes falha por uma série de motivos como: mudança de iluminação, pontes, sombra de árvores, etc. Para resolver esta questão o veículo é rastreado e se em algum ponto ele é perdido, ele pode ser recuperado por um modelo e pelo seu grau de confiança (*degree of trust* - DOT). Cada veículo detectado possui um identificador (ID) e um DOT. O DOT é inicializado com zero. No caso de detecção e rastreamento contínuo com o mesmo ID, o DOT é incrementado em 1. Caso o veículo seja perdido, o DOT é decrementado em 1 e o modelo é atualizado.

Para rastreamento é utilizado o algoritmo de Lucas e Kanade (1981) que minimiza o erro entre o modelo pré-estabelecido e a sub-região rastreada no quadro corrente (Figura 39 (e)). Caso um veículo rastreado é perdido, ele pode ser restaurado se possuir um DOT com peso alto. Reduz-se as restrições dos limiares para detecção da sombra e das linhas verticais no quadro corrente e decrementa-se o peso do DOT correspondente.

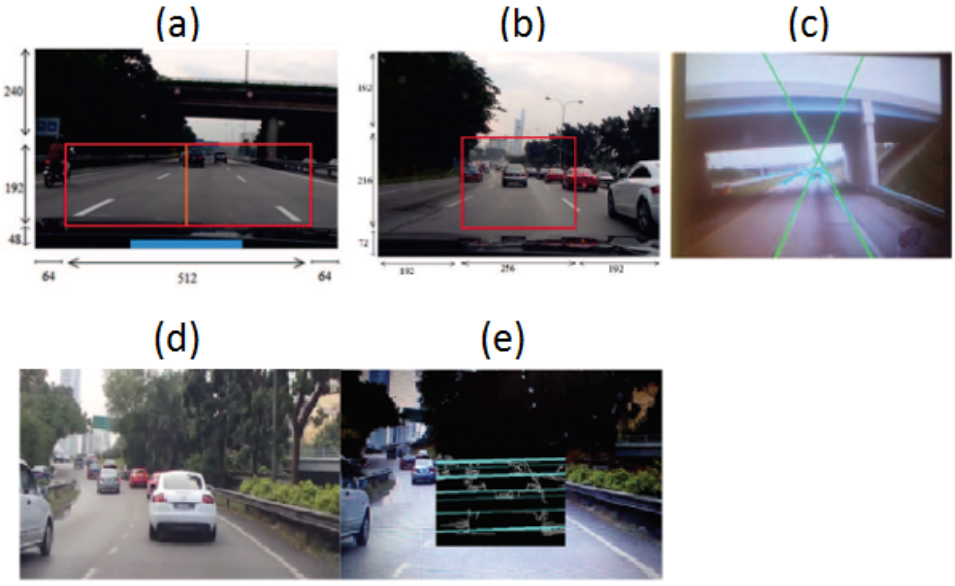
- Estimativa de distância: Somente relata estimativa de distância até 10 metros com sensor de sonar.

- Taxa de detecção: 85% de detecção sem rastreamento. 96% de detecção com rastreamento.
- Tempo de processamento: 50 ms ($\approx$ 20 fps).
- Alertas: Não emite alertas ao motorista.
- Observações: Com exceção do limiar para o DOT que é 3, não informa os valores dos outros limiares utilizados nos experimentos. Caso ocorra impasse entre determinar se é dia ou noite, por motivos de iluminação ou outro qualquer, são aplicados os dois algoritmos (detecção de dia e detecção de noite) e é selecionado aquele que conseguir criar candidatos a veículo. Caso ambos algoritmos extraiam candidatos, é utilizado o algoritmo para de dia. Apesar deste trabalho ser mais antigo, ele foi incluído na revisão por tratar tolerância a falhas quando um veículo é perdido durante o rastreamento (uso dos DOTs), e foram encontrados pouquíssimos trabalhos que tratam esta questão.

- Tang et al. (2015) - *Real-Time Lane Detection and Rear-End Collision Warning System On A Mobile Computing Platform*:

- Objetivos principais: Desenvolver um sistema em dispositivo móvel para detectar as faixas da estrada e os veículos à frente do veículo portando à câmera. Também estimar a distância da câmera até os veículos à sua frente e emitir alerta de perigo de colisão.
- Técnicas de GH: Primeiramente é calculada a ROI de detecção das faixas da estrada (Figura 40(a)), onde são ignoradas regiões nos quatro lados da imagem a fim de reduzir a área de processamento.

Figura 40 – (a) ROI das faixas. (b) ROI para detecção de veículos. (c) Potenciais faixas quando as linhas convergem. (d) e (e) Detecção de veículo



Fonte: Tang et al. (2015).

Um pixel é candidato a pertencer a uma faixa se tiver valor de intensidade maior que um limiar de 175. A fim de eliminar falso positivos com objetos grandes de cor branca (por exemplo: veículos), foi desenvolvido um método de limiarização adaptativa binária, onde uma janela deslisante de 17 x 17 pixels percorre a imagem e calcula a intensidade média dos pixels naquela região.

Caso a intensidade do pixel central somando com 10, for maior do que a média da região da janela, então ele é considerado como candidato a pertencer a uma faixa. Porém, objetos pequenos ainda podem ser detectados errados, assim é feito uma operação *AND* entre os dois métodos anteriores e a região candidata a faixa é extraída.

Para a detecção de regiões candidatas a veículos, primeiramente também é defina uma ROI (Figura 40(b)). Em seguida, linhas horizontais são detectadas utilizado o detector de bordas de Canny. A ROI é dividida em três regiões: base, meio e topo. Cada região é relacionada a um limiar de comprimento das linhas horizontais, sendo 20, 40 e 60 pixels respectivamente, linhas maiores que estes limiares, são candidatas a região (Figuras 40(d) e (e)).
- Técnicas de VH: Usa a transformada de Hough para validar as linhas das faixas mesmo com inclinações. Do ponto de vista da câmera, a faixa do lado esquerdo e a do lado direito do veículo formam o contorno de um triângulo (Figura 40(c)).

Para a validação de regiões candidatas a veículos são utilizadas informações temporais, onde são acumulados os valores de quatro em quatro quadros a fim de distribuir o ruído uniformemente e incrementar a confiança na detecção dos veículos.
- Rastreamento: Usa informações temporais.
- Estimativa de distância: Usa semelhança de triângulos utilizando os parâmetros da câmera.
- Taxa de detecção: Não informado.
- Tempo de processamento: $\approx$ 10-25 fps.
- Alertas: Cria uma zona de perigo em cada lado do veículo. Avisa de forma sonora caso o veículo mudar de faixa e entrar na zona de perigo. Para alertas de colisão, divide a distância estimada até o veículo à frente pela sua velocidade, e caso esteja a menos de 4 segundos do veículo, é emitido um alerta sonoro ao motorista do veículo.
- Observações: As linhas apresentadas são exibidas apenas em ambiente de testes. Em um ambiente de uso prático elas são ocultas para diminuir custos de processamento.

• Di e He (2016) - *Forward Collision Warning System Based on Vehicle Detection and Tracking*:
- Objetivos principais: Apresentar uma proposta de sistema para aviso de colisão fron-

tal que auxilia o motorista a evitar colisões com veículos à sua frente. Aplicar técnicas para detecção e rastreamento de veículos e também estimar a distância entre eles.
- Técnicas de GH: Uma região candidata a veículo é baseada na sombra debaixo do veículo, pois está região sempre parecerá mais escura do que o resto da estrada (Figura 41(a)). A ROI para detecção é de até 15 metros à frente do veículo e com a mesma largura do veículo. A câmera é fixada atrás do espelho retrovisor do parabrisas frontal.

Figura 41 – (a) Imagem original. (b) Imagem binária com bodas. (c) Imagem com a caixa delimitadora. (d) Extração de características com HOG. (e) Correspondência de pontos de características

Fonte: Di e He (2016).

É coletado o histograma desta região para determinar as bordas laterais da região da sombra. As bordas são extraídas utilizando os operadores de Sobel, e após é aplicado um método de binarização da imagem baseada na média local da intensidade dos pixels de uma vizinhança-8 (Figura 41(b)).

Por fim é determinada a caixa delimitadora em torno da região do veículo. Para isto, um ponto inicial apropriado da detecção é necessário. Localiza-se o centro da sombra, combina-se a região da sombra com as bordas para localizar a base do veículo. Então a análise segue da base em direção a linha horizontal no topo do veículo, delimitado pelas linhas verticais laterais (Figura 41(c)).
- Técnicas de VH: Esta etapa é essencial por filtrar falsas regiões candidatas a veículos geradas pela fase de GH. Por isto precisa de um extrator de características com alto poder de discriminação entre imagens de veículos e não veículos, como é o caso do HOG.

As imagens dos veículos são redimensionadas para 64 x 64 pixels e fatiadas em 9 blocos. Cada bloco é dividido em 4 células das quais são obtidas 20 direções igualmente distribuídas do histograma, gerando um vetor de 80 dimensões (Figura 41(d)).

Estes vetores de características são entradas para um classificador AdaBoost, o qual é treinado para separar veículos de não veículos. É montada uma cascata de classificadores fracos, onde para uma amostra ser classificada como veículo, ela precisa passar como positiva por todos os estágios da cascata. Se em algum estágio ela

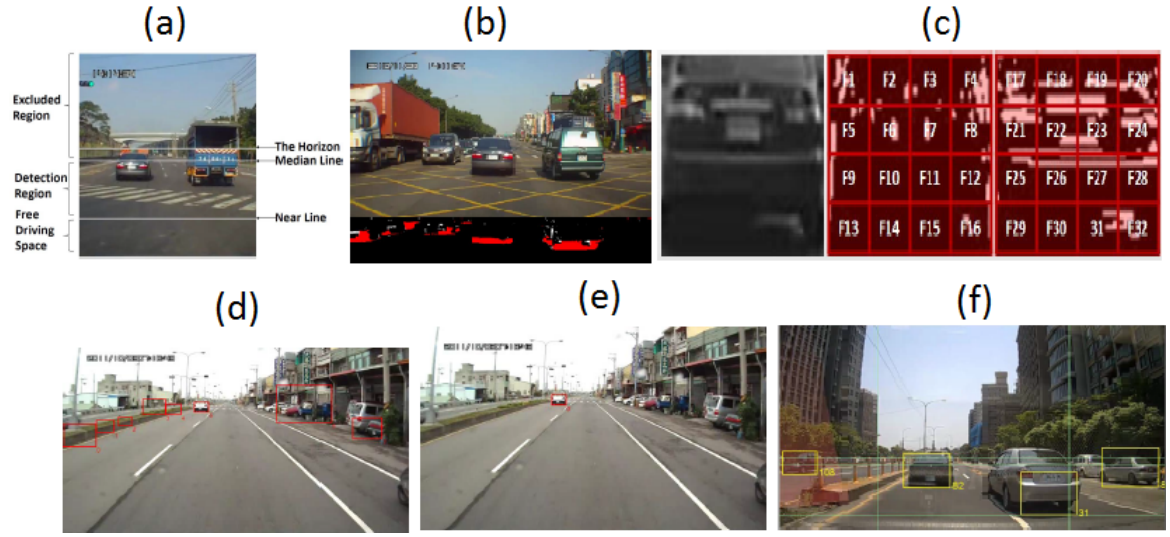
for avaliada como negativa, o processo é abortado e segue para uma nova imagem.
- Rastreamento: Rastrear um veículo é estimar a nova posição de um determinado veículo já detectado no quadro anterior, no quadro atual. Para valer a pena, o processo de rastreamento deve ser mais barato computacionalmente do que o processo de detecção. O que normalmente é verdadeiro, pois existem técnicas com bom desempenho que conseguem estimar a nova posição do veículo e encontrar pontos na nova região que combinem com os pontos da região anterior.

Foi utilizado o algoritmo de Harris e Stephens (1988) para detectar cantos em várias escalas. Este algoritmo extrai pontos de características e usa correspondência entre estes pontos em dois quadros adjacentes. Procura correspondência apenas na vizinhança para diminuir o custo computacional. Os pontos são normalizados no intervalo -1 até 1.

Cada par de pontos com correspondência maior que 0,9 são considerados correspondentes e produzem um vetor de movimento para avaliar o movimento do veículo à medida que o tempo passa. Se mais da metade dos pontos de características combinarem, o veículo foi detectado e é gerada a caixa delimitadora na nova posição (Figura 41(e)).
- Estimativa de distância: Não deixa bem claro o cálculo para estimar a distância. Relata apenas que a distância é dada pela relação fixa da posição vertical da linha base da caixa delimitadora do veículo. Relata ainda que o erro máximo relativo da distância calculada é de 5%.
- Taxa de detecção: 91,37%.
- Tempo de processamento: Relata apenas que a taxa de gravação é de 24 fps.
- Alertas: Não relata como notifica o motorista.
- Observações: A metodologia em cascata aplicada no classificador AdaBoost, apesar de ser com o extrator de características HOG, é semelhante a metodologia aplicada por Viola e Jones (2004).

• Hsiao, Yeh e Chen (2014) - *Design and Implementation of Multiple-Vehicle Detection and Tracking Systems with Machine Learning*:
- Objetivos principais: Desenvolver um sistema para detecção e rastreamento de veículos em duas fases: Geração de Hipótese (GH) e Verificação de Hipótese (VH). Propor uma arquitetura de hardware que acelere o cálculo da intensidade dos pixels das bordas verticais e horizontais de uma região da imagem. Estas informações geram vetores de características para serem usadas com uma SVM na fase de VH.
- Técnicas de GH: Primeiramente divide cada quadro da imagem em três regiões: zona excluída (topo da imagem), região de detecção (meio da imagem) e região de condução livre (base da imagem) (Figura 42(a)).

Figura 42 – (a) Definição das áreas. (b) Detecção das sombras (áreas vermelhas). (c) Extração das características do mapa de bordas. (d) Depois da GH. (e) Depois da VH. (f) Sistema rodando na rua



Fonte: Hsiao, Yeh e Chen (2014).

A detecção de regiões candidatas a veículos é feita analisando as sombras de baixo dos veículos. Para isto usa uma estrutura com o percentual da intensidade dos níveis de cinza dos pixels a qual é denominada de função de densidade cumulativa (CDF). Assim, as regiões com percentuais maiores que um limiar e que sejam de intensidades baixas, e cujos pixels tenham a vizinhança conectada, são sombras e as outras não são sombras (Figura 42(b)).

No passo seguinte são calculadas as linhas verticais e horizontais das bordas, utilizando o operador de Sobel. Baseado na largura da sombra encontrada é estimada a caixa delimitadora da região, sendo a largura igual a da sombra, e a altura 0,7 vezes a largura.

- Técnicas de VH: A verificação das regiões extraídas na fase de GH é feita por uma SVM. A extração das características é baseada nos mapas de bordas com as linhas verticais e horizontais. Cada mapa de bordas é dividido em uma grade de 16 elementos, cada um com uma característica, formando assim vetores de características com 32 componentes escalares (Figura 42(c)).

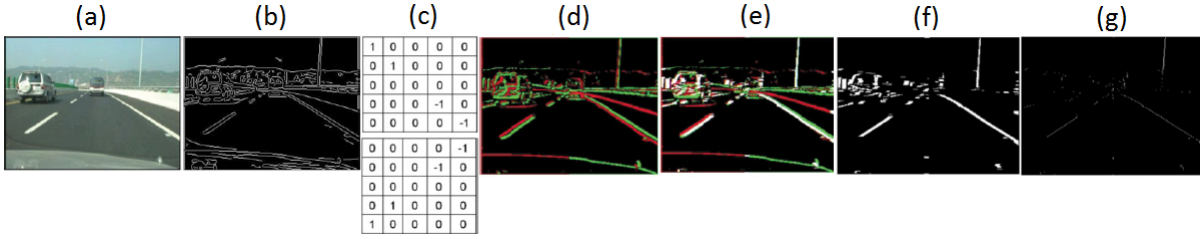
A Figura 42(d) mostra um exemplo após a fase de GH, há vários retângulos vermelhos com regiões falsa positivas. Já a Figura 42(e) mostra o resultado após a fase de VH, percebe-se a remoção das regiões falsa positivas. Na Figura 42(f) é mostrado outra aplicação real do sistema.

- Rastreamento: O rastreamento é feito utilizando correspondência de modelos (*template matching*). O rastreamento permite predizer com baixa margem de erro a localização do veículo no quadro seguinte. Isto reduz o uso de recursos computacionais. Porém é estipulado um tempo máximo para manter o rastreamento, após este período

exceder, volta a detectar e em seguida pode rastrear novamente.
- Estimativa de distância: Não estima distância.
- Taxa de detecção: ≈ 95%.
- Tempo de processamento: No computador: 91,6 ms (≈ 11 fps). No hardware e microcontrolador: 375,5 ms (≈ 2,6 fps)
- Alertas: Não emite nenhum alerta ao motorista.
- Observações: No hardware foi implementado o CDF e o operador de Sobel para identificar as linhas verticais e horizontais. Os resultados dos cálculos no hardware foram armazenados em uma memória compartilhada por um microcontrolador. A detecção e o rastreamento rodam no microcontrolador. O CDF de 3,9% do tempo do processamento no computador foi para 38% no hardware, e o operador de Sobel foi de ≈ 80% do tempo do processamento no computador para ≈ 4% no hardware.

• Wu, Lin e Lee (2012) - *Applying a Functional Neurofuzzy Network to Real-Time Lane Detection and Front-Vehicle Distance Measurement*:
- Objetivos principais: Detectar as faixas e os veículos à frente na estrada. Estimar a distância entre a câmera e os veículos. Emitir alertas ao motorista devido ao veículo trocar de faixa.
- Técnicas de GH: Primeiro são detectadas as faixas da estrada. Começa aplicando-se sobre a imagem original o detector de bordas de Canny (Figuras 43(a) e (b) respectivamente).

Figura 43 – (a) Imagem original. (b) Detector de bordas de Canny. (c) Máscaras 5 x 5. (d) Linha positiva e negativa. (e) Identificação da linha branca. (f) Imagem binária. (g) Redução



Fonte: Wu, Lin e Lee (2012).

No trabalho foi desenvolvido e aplicado um algoritmo específico para detectar as linhas das faixas da estrada. Este algoritmo faz uma transformação de cores e consegue identificar linhas pelo seu grau de inclinação. São aplicadas duas máscaras 5 x 5 para identificar as linhas, uma máscara para a linha do lado esquerdo e uma máscara para a linha do lado direito da faixa (Figura 43(c), topo e base respectivamente).

As imagens analisadas são em nível de cinza. Bordas com intensidade maior que um limiar de valor 30 recebem cor verde, e são consideradas positivas, e as bordas

com intensidade menor do que outro limiar -30 são consideradas negativas, e recebem cor vermelha (Figura 43(d)). As demais áreas recebem cor preta.

Caso a distância entre a borda verde e vermelha seja menor do que cinco pixels, esta região recebe a cor branca (Figura 43(e)). Para gerar a imagem binária, exceto as regiões de cor branca, todo o resto da imagem recebe a cor preta (Figura 43(f)). Por fim é aplicado um algoritmo de redução, o qual vare a imagem de baixo para cima e do meio para ambos os lados. O primeiro encontro com um pixel de borda em cada linha é salvo e todos os outros pontos são excluídos (Figura 43(g)).

Para detectar veículos, uma ROI de busca é estimada pelas faixas detectadas na estrada conforme já apresentado. A detecção é iniciada procurando a sombra dos veículos. A intensidade dos pixels da região da sombra é bem menor do que a intensidade da estrada.

Para cada linha da imagem é calculada a intensidade média do nível de cinza. Se a proporção entre a intensidade média e a intensidade de um pixel for menor do que um determinado limiar, o pixel recebe a cor branca, caso contrário ele recebe a cor preta (Figura 44(a)).

Figura 44 – (a) Detecção das sombras. (b) Fechamento morfológico. (c) Redução. (d) Detecção noturna. (e) Filtro de Sobel para detectar bordas. (f) Resultado final

Fonte: Wu, Lin e Lee (2012).

Isto muda uma imagem em níveis de cinza para uma imagem binária. Experimentalmente foi optado pelo valor 0,5 para o limiar. Para fechar pequenos buracos foi aplicado operações de fechamento morfológico (Figura 44(b)). Como nesta etapa o interesse é só na região da sombra, partes brancas do veículo logo acima da sombra são removidas. Para fazer isto é utilizando um algoritmo de redução, análogo ao já descrito para a redução das faixas (Figura 44(c)).

Em ambiente noturno os veículos são detectados através das suas luzes. A intensidade dos pixels nas regiões das luzes é maior do que das outras regiões. Utiliza-se processo análogo, porém com o limiar de intensidade dos pixels inversa aos procedimentos descritos para detecção da sombra em baixo dos veículos (Figura 44(d)).
- Técnicas de VH: A região da sombra é validada de forma que regiões maiores ou

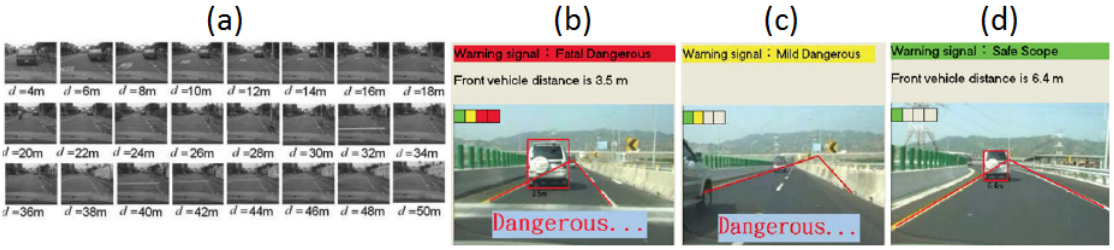
muito fora da região das faixas da estrada, detectadas anteriormente, sejam descartadas.

Para estimar a região do veículo, parte-se da região da sombra detectada, é utilizado um filtro de Sobel para detectar as bordas horizontais e verticais em torno daquela região (Figura 44(e)). Com isto é possível estimar as regiões delimitadoras do veículo e desenhar um retângulo em volta da região (Figura 44(f)).

Para ambiente noturno, é utilizado o algoritmo de agrupamento de luzes traseiras proposto por Kuo e Chen (2010). Onde basicamente é estipulada uma ROI de busca por simetria das luzes do lado esquerdo e direito do veículo. Regiões fora das faixas da estrada são descartadas.

- Rastreamento: Não relata técnica de rastreamento.
- Estimativa de distância: Para estimar à distância entre a câmera e os veículos não são usados nenhum parâmetro da câmera, é treinada uma rede neurofuzzy funcional (FNFN), conforme o modelo proposto por Lin, Liu e Lee (2008). A rede fuzzy é treinada baseada no relacionamento da distância em pixel e a distância real da câmera até o veículo (Figura 45(a)). Para reduzir erros, as distâncias são medidas mais de uma vez e utiliza-se os valores médios para treinamento da rede. Após treinada, a FNFN é capaz de estimar a distância real a partir da distância em pixels baseada em regras fuzzy *IF-THEN*.
- Taxa de detecção: 89,96%.
- Tempo de processamento: Não informado efetivamente, a captura é de 20 fps. Porém relata que o sistema pode ser usado em um ambiente de tempo real para detecção de veículos.
- Alertas: Notifica o motorista toda vez que o veículo muda de faixa (sai da área das linhas das faixas detectadas). Esta situação também ocorre quando o veículo entra em uma curva. Exemplos de alertas são mostrados nas Figuras 45(b), (c) e (d).
- Observações: Relata problemas na detecção das faixas com veículos próximos ao ponto de fuga da faixa e de veículos por causa da sombra ao passar em baixo de ponte e baixa iluminação à noite.

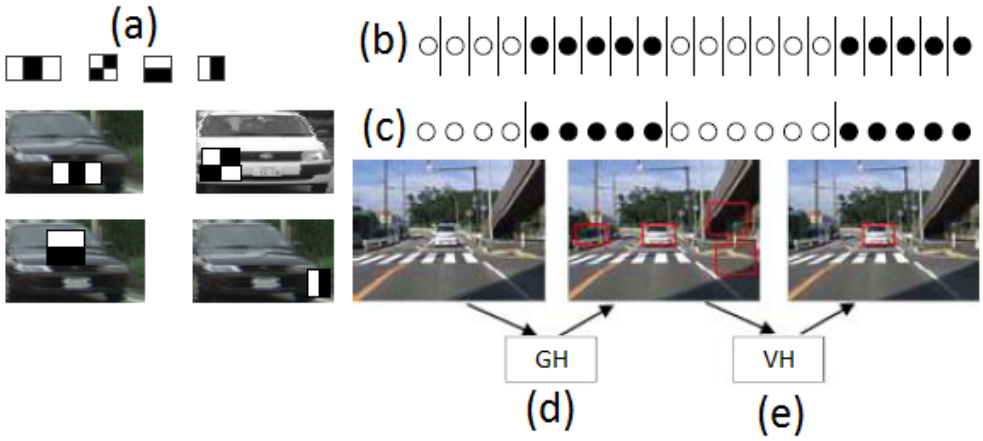
Figura 45 – (a) Medidas de distância real. (b) Alerta de perigo fatal. (c) Alerta de perigo médio. (d) Alerta de ambiente seguro



Fonte: Wu, Lin e Lee (2012).

• Wen et al. (2015) - *Efficient Feature Selection and Classification for Vehicle Detection*:
- Objetivos principais: Detectar regiões candidatas a veículos, fase de geração de hipótese (GH), e melhorar o desempenho do algoritmo Haar-like com AdaBoost e SVM para a verificação das regiões candidatas, fase de verificação de hipótese (VH), combinando as características extraídas das amostras com seus rótulos de classe.
- Técnicas de GH: Detecta regiões candidatas, baseado em Wen et al. (2006) e Liu et al. (2007), onde analisa a sombra de baixo dos veículos, suas bordas e simetria. Algumas operações *AND* entre as validações são feitas para melhorar a confiança.
- Técnicas de VH: É utilizado Haar-like para extrair as características das amostras que descrevem as aparências dos veículos (Figura 46(a)). Foi desenvolvido um algoritmo para normalizar os valores das características selecionadas de forma que reduza a diferença intra-classe enquanto aumenta a variabilidade entre-classes (Esquema da Figura 46(b) e (c), círculos vazados e com preenchimento pretos denotam duas classes respectivamente).

Figura 46 – (a) Extração de características com Haar-like. (b) Haar-like com AdaBoost tradicional. (c) Haar-like com AdaBoost proposto. (d) Geração de hipótese. (e) Verificação de hipótese



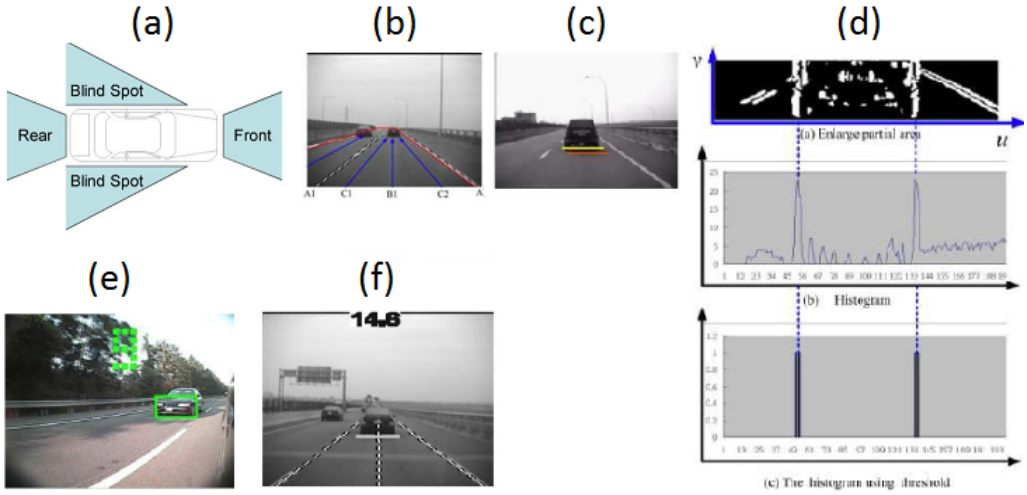
Fonte: Wen et al. (2015).

Os valores das características são normalizados para [0, 1], é construído um vetor de características para treinar um classificador SVM, o qual obtém o resultado da classificação e faz a verificação de hipótese (Figura 46(d) e (e)).
- Rastreamento: Não efetua rastreamento.
- Estimativa de distância: Não estima distância.
- Taxa de detecção: 94,94%.
- Tempo de processamento: Reduziu o tempo de treinamento de 117,43 horas para 101,91 horas ($\approx$ 15 horas).

- Alertas: Não emite nenhum tipo de alerta.
- Observações: Treina as imagens de amostras com os tamanhos 32 x 32 pixels em níveis de cinza.

- Wu et al. (2012) - *A Vision-Based Collision Warning System by Surrounding Vehicles Detection*:
- Objetivos principais: Prevenir através de avisos ao motorista, possíveis colisões entre veículos. Para isto monitora o estado dos veículos a sua volta, incluindo a distância dos outros veículos à frente e à atrás, do lado esquerdo e do lado direito (Figura 47(a)).
- Técnicas de GH: A fim de processar regiões menores do que toda a imagem, é determinada uma ROI baseado na detecção das faixas da estrada, determina também a faixa do veículo *host* (Figura 47(b)). Com uma taxa de quadros por segundo alta, o deslocamento de um veículo muda pouco de um quadro para o outro, o que permite estimar a ROI do veículo no quadro subsequente.

Figura 47 – (a) Áreas monitoradas. (b) Detecção da ROI. (c) Detecção das linhas horizontais. (d) Detecção das linhas verticais. (e) Detecção lateral traseira. (f) Detecção frontal



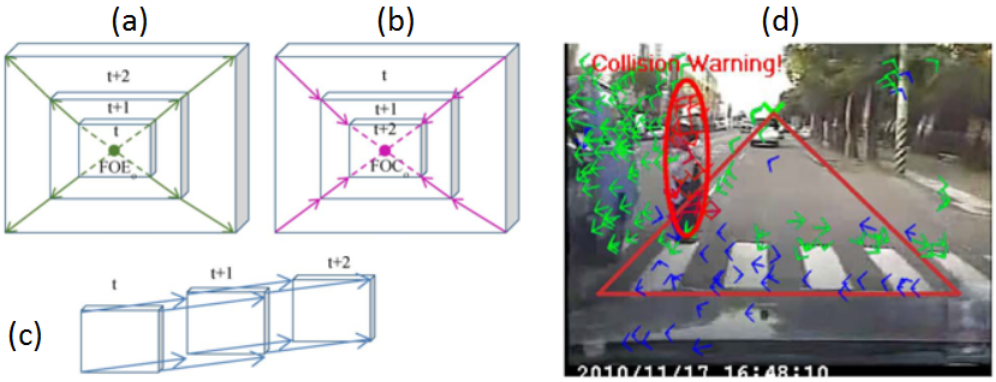
Fonte: Wu et al. (2012).

As regiões candidatas a veículos são identificadas pela diferença dos níveis de pixel identificados dentro da região das faixas já identificadas. É analisada a sombra debaixo do veículo, isto devido a apresentar elevada diferença na intensidade dos níveis de cinza se comparado com o resto da estrada. Também são detectadas as linhas horizontais através da aplicação de um filtro de Sobel (Figura 47(c)).
- Técnicas de VH: Para reforçar a certeza da região detectada e estimar a caixa delimitadora do veículo, é detectado o par de bordas verticais através do histograma gerado pela detecção de bordas verticais com o filtro de Sobel. Onde regiões com intensidade dentro de um determinado limiar são consideradas bordas verticais (Figura 47(d)).

- Rastreamento: Não apresenta técnica de rastreamento.
- Estimativa de distância: A distância é calculada baseado nos parâmetros da câmera, onde projeta os pontos do espaço 3-D para o espaço 2-D. Para determinar a distância nas laterais é necessário calibração da câmera. Para isto são utilizadas as faixas detectadas na estrada como referência.
- Taxa de detecção: 97%.
- Tempo de processamento: $\approx$ 30 fps.
- Alertas: Para veículos detectados à frente e atrás do veículo *host*, é disparado alerta caso a distância entre o veículo *host* e os veículos detectados seja menor que a metade da velocidade do veículo *host*. Para veículos detectados nos pontos cegos, é disparado alerta de perigo de colisão caso a distância entre eles seja menor que 10 metros.
- Observações: As Figuras 47(e) e 47(f) mostram exemplos da detecção lateral esquerda e à frente respectivamente. Não apresenta explicitamente como são os alertas.

• Yang e Zheng (2015) - *On-Road Collision Warning Based on Multiple FOE Segmentation Using a Dashboard Camera*:
- Objetivos principais: Desenvolver um sistema que visa detectar os padrões gerais de movimento de qualquer objeto que se aproxima da câmera acoplada ao veículo.
- Técnicas de GH: Usa Fluxo óptico (direção do movimento de grupos de pixels), onde é feita a correspondência (casamento) das características extraídas (pequena vizinhança de pixels com maioria significativa de cantos) de um quadro com quadros subsequentes, a fim de determinar a semelhança e a direção da região se movimentando. As regiões em movimento são os objetos.
- Técnicas de VH: Usa uma árvore de decisão em cascata para filtrar falso positivos o mais breve possível. Criou o termo foco de expansão (FOE) de objetos, que aumenta conforme o objeto se aproxima da câmera, e foco de contração (FOC) que diminui conforme o objeto se distância da câmera. Objetos em estado FOE (Figura 48(a)) são candidatos a colisão, já objetos FOC (Figura 48(b)) e transladando (Figura 48(c)) são descartados por não oferecerem riscos.
- Rastreamento: Utiliza fluxo óptico.
- Estimativa de distância: Calcula o tempo estimado antes de um objeto colidir com a câmera, com uma técnica chamada *time-to-collision* (TTC). Em um ambiente 2-D, o TTC pode ser obtido pela primeira derivada dos vetores de movimento do fluxo óptico.
- Taxa de detecção: 91%.
- Tempo de processamento: $\approx$ 50 ms ($\approx$ 20 fps).
- Alertas: Duas condições disparam alertas: Primeira, se há múltiplos objetos em estado FOE. Segundo, foi estipulada uma zona de perigo, onde se há objetos nesta área, emite alerta de perigo de colisão (Figura 48(d)).

Figura 48 – (a) Objeto FOE. (b) Objecto FOC. (c) Objeto transladando, não possui FOE e nem FOC. (d) Zona de perigo de colisão com emissão de alerta



Fonte: Yang e Zheng (2015).

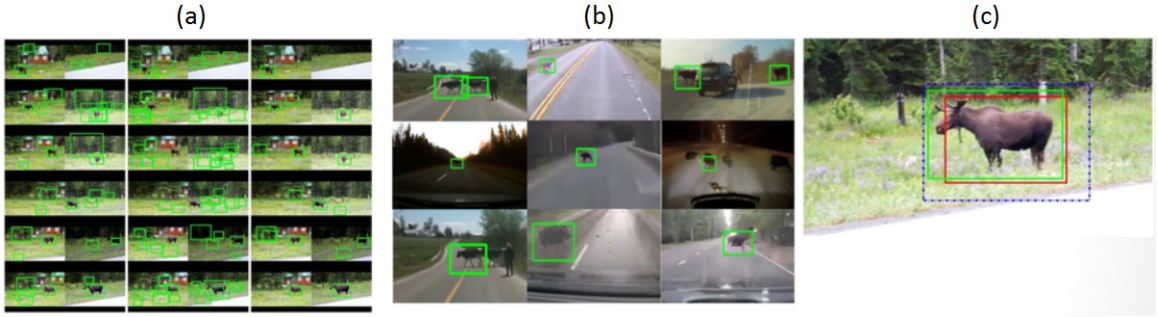
- Observações: Outra alternativa relatada ainda para TTC é baseada no treinamento de pilotos de avião para evitar colisões aéreas, a qual é detectada quando um objeto permanece visível e crescente na região do para-brisas ao longo do tempo.

- Mammeri, Zhou e Boukerche (2016) - *Animal–Vehicle Collision Mitigation System for Automated Vehicles*:

- Objetivos principais: Desenvolver um sistemas para prevenção de colisão de veículos com animais grandes na estrada. Fazer comparações entre os extratores de características Haar-like, HOG e MB-LBP com o classificador AdaBoost, e HOG com o classificador SVM.

- Técnicas de GH: Utiliza três extratores de características Haar-like, HOG e MB-LBP. Cada um dos extratores é combinado individualmente com o classificador AdaBoost. Resultados podem ser vistos na Figura 49(a), da esquerda para a direita respectivamente. Também foi utilizado HOG com o classificador SVM. O trabalho faz comparações de precisão na detecção e tempo de processamento.

Figura 49 – (a) Da esquerda para direita: Haar-like, HOG e MB-LBP com AdaBoost. (b) Exemplos de detecção. (c) Ajustes nas dimensões das ROIs para as fases de GH e VH

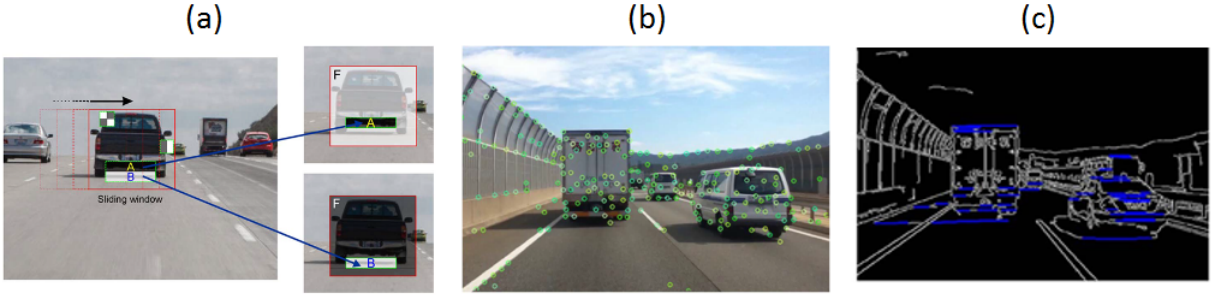


Fonte: Mammeri, Zhou e Boukerche (2016).

- Técnicas de VH: A combinação MB-LBP com AdaBoost na fase de GH, e HOG com SVM na fase de VH obteve melhor desempenho considerando taxa de detecção versus tempo de processamento e redução de falso positivos (Figura 49(b)).
- Rastreamento: Não há rastreamento.
- Estimativa de distância: Não há estimativa de distância.
- Taxa de detecção: Taxa média de falsos positivos por imagem de 1 x 10^{-4}.
- Tempo de processamento: 64.32 ms ($\approx$ 15.5 fps).
- Alertas: Não emite alertas.
- Observações: A ROI detectada pelo classificador AdaBoost na fase de GH (Figura 49(c) retângulo vermelho ao centro) pode ser insuficiente para que o classificador SVM, na fase de VH, consiga detectar (Figura 49(c) do centro para fora o segundo retângulo, de cor verde). Para resolver esta questão, a ROI obtida na GH foi expandida em cerca de 20% (Figura 49(c) retângulo azul pontilhado mais externo). Esta expansão para ROIs grandes pode aumentar o custo computacional, assim as ROIs são redimensionadas para 70 x 50 pixels.

- Rezaei, Terauchi e Klette (2015) - *Robust Vehicle Detection and Distance Estimation Under Challenging Lighting Conditions*:
- Objetivos principais: Desenvolver um sistema capaz de detectar veículos sobre diferentes condições de tempo, de dia ou de noite, e que avise o motorista sobre o risco de colisão veicular detectando os veículos a frente e estimando a distância ente eles.
- Técnicas de GH: AdaBoost com Haar-like. Fez uma adaptação no Haar-like, introduzindo o conceito de Haar-like com adaptação global. Cada característica local Haar-like é acompanhada por duas características globais. Global em relação a janela deslizante. Uma característica global é calculada pela diferença da imagem integral da janela deslizante por cada retângulo da característica local (Figura 50(a)). Isto permite, por exemplo, obter uma intensidade quase constante de um veículo distante.

Figura 50 – (a) Características Haar-like globais adaptáveis. (b) Detecção de cantos. (c) Detecção de linhas horizontais



Fonte: Rezaei, Terauchi e Klette (2015).

- Técnicas de VH: Analisa características de cantos (Figura 50(b)), detecção de linhas horizontais utilizando a transformada de Hough (Figura 50(c)), simetria entre as luzes traseiras da região candidata. Faz fusão das técnicas usando a teoria de Shafer (1976), que atribui pesos baseados na expertise humana a cada metodologia utilizada.
- Rastreamento: Não é feito.
- Estimativa de distância: Utiliza parâmetros da câmera e triangulação.
- Taxa de detecção: 95%.
- Tempo de processamento: Entre 25 e 28 fps.
- Alertas: Não relata a emissão de alertas ao motorista.
- Observações: Usa fusão de várias técnicas (cantos, linhas horizontais, simetria dos faróis) na fase de VH.

3.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Através da revisão dos trabalhos relacionados anteriormente, pôde-se perceber pontos comuns e distintos entre eles, conforme mostra a Tabela 3. As partes destacadas são as principais contribuições utilizadas neste trabalho.

Tabela 3 – Resumo comparativo das características entre os trabalhos relacionados

Trabalho	Técnicas GH	Técnicas VH	Rastreamento	Distância	Alertas
Kim et al. ()	Sombra	**Simetria linhas**	**DOT**, Correspondência de modelo, **informações temporais**	Parcial, com sonar	Não
Tang et al. (2015)	**Linhas**	**Informações temporais**	Informações temporais	Parâmetros da câmera, **triangulação**	Mudança de faixa, **alerta sonoro**
Di e He (2016)	Sombra	**HOG+Adaboost**	Correspondência de cantos	Cálculo não explicado	Não
Hsiao, Yeh e Chen (2014)	Sombra, **linhas**	Mapa de bordas + **SVM**	Correspondência de modelo	Não	Não
Wu, Lin e Lee (2012)	Sombra	largura da sombra, **linhas**	Não	Rede fuzzy, **proporção medida real vs da câmera**	Mudança de faixa, **alerta visual**
Wen et al. (2015)	Sombra, **linhas**	**Haar-like+AdaBoost** e **SVM**	Não	Não	Não
Wu et al. (2012)	Sombra, **linhas**	**Simetria das linhas**	Não	Parâmetros da câmera, **triangulação**	**Veículo à frente** e atrás, não diz o tipo
Yang e Zheng (2015)	Fluxo óptico	Árvore de decisão, **FOE**, FOC	Fluxo óptico	Tempo para colisão	**FOE** e **zona de perigo**, **visual**
Mammeri, Zhou e Boukerche (2016)	**Haar-like**, **HOG** e **MB-LBP com AdaBoost**	**SVM**	Não	Não	Não
Rezaei, Terauchi e Klette (2015)	**Haar-like+AdaBoost**	Cantos e **linhas**, simetria das luzes	Não	Parâmetros da câmera, **triangulação**	Emite, mas não relata como

Fonte: Elaborado pelo autor, 2017.

Percebe-se que, seis trabalhos analisados utilizam a região da sombra como principal estratégia na fase de GH; sete trabalhos utilizam as linhas verticais e/ou horizontais para GH e/ou VH, geralmente analisando a simetria entre elas; quatro trabalhos empregam o classificador AdaBoost nas fases de GH e/ou VH; o classificador SVM é utilizando na fase de VH de três trabalhos; a metade dos trabalhos não fazem rastreamento; não foi encontrado nenhum trabalho que utilizasse alguma técnica de rastreamento mais recente, como é o caso do rastreador KCF proposto por Henriques et al. (2015), o qual é utilizado neste trabalho.

Três trabalhos não fazem estimativa de distância, um faz parcial até 10 metros com sonar, um não explica como faz e um só calcula o tempo estimado para colisão. Dos quatro trabalhos que realmente estimam a distância, três deles utilizam semelhança de triângulos utilizando os parâmetros da câmera. Destes três trabalhos, dois foram implementados para testes (Rezaei, Terauchi e Klette (2015) e Tang et al. (2015)), porém nenhum dos dois calculou corretamente utilizando as equações dos respectivos trabalhos.

Em relação a emissão de alertas, cinco trabalhos não reportam alertas ao motorista e dois não exibem ou relatam como eles são feitos. Apesar da maioria destes trabalhos nos seus resumos e até nos seus títulos proporem sistema que auxiliam o motorista a evitar colisões através de alertas, poucos o fazem efetivamente, o que pode acabar inviabilizando a aplicação prática de alguns destes sistemas no dia a dia dos motoristas.

No próximo capítulo serão apresentadas as técnicas e métodos utilizados neste trabalho para desenvolver um sistema para alerta de colisão veicular. Muitas das técnicas explicadas são baseadas nas contribuições dos trabalhos apresentados acima como: detecção dos veículos com AdaBoost e SVM; o grau de confiança, baseado no DOT; foco de expansão baseado no FOE; definição de uma zona de risco; detecção de linhas verticais e horizontais e cálculo da simetria entre elas; estimativa de distância entre veículos baseado em triangulação e proporção das medidas do mundo real e da câmera.

4 SISTEMA PARA ALERTA DE COLISÃO VEICULAR

Neste capítulo será detalhado o funcionamento do sistema para alerta de colisão veicular. Primeiro será explicada a proposta de uma nova técnica, chamada de CCLVH, para verificação de hipótese (VH) de uma região candidata a veículo obtida na etapa de geração de hipóteses (GH). Em seguida será apresentado um cálculo para estimativa da distância entre veículos, e finalmente o fluxo de funcionamento do sistema proposto.

4.1 VERIFICAÇÃO DE HIPÓTESE UTILIZANDO CCLVH

Conforme relatado por Hsiao, Yeh e Chen (2014), Hadi, Sulong e George (2014), Tian et al. (2017), Sreevishakh e Dhanure (2015) e Mukhtar, Xia e Tang (2015), a fase de VH é importante na detecção de veículos, pois é nela que é valida se uma região candidata obtida na fase de GH, é ou não um veículo. Um algoritmo robusto e rápido nesta etapa, não só diminui a taxa de falso positivos como deixa o sistema mais rápido por remanescerem menos regiões para serem monitoradas pelo sistema de alertas de colisão.

Nesta seção será discutida a elaboração de uma nova técnica para efetuar verificação de hipótese (VH) de veículos. A nova técnica proposta é chamada de Confiança dos Centroides das Linhas Verticais e Horizontais (CCLVH). O conjunto de procedimentos propostos é específico para ser aplicado sobre uma região candidata a veículo e dar seu voto com um valor de confiança se a região é ou não um veículo. A técnica é baseada nos seguintes passos principais sobre a região candidata a veículo:

a) pré-processamento da região;

b) detecção das linhas verticais e horizontais;

c) cálculo do centroide vertical e horizontal;

d) cálculo do limiar de confiança.

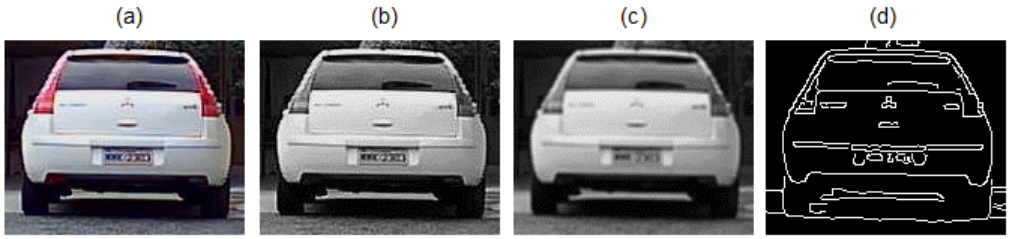
A seguir será feito o detalhamento dos passos da técnica e dos algoritmos envolvidos.

4.1.1 Pré-processamento da Região Candidata

Para poder detectar as linhas na região candidata a veículo com mais velocidade e precisão, é necessário aplicar algumas operações de pré-processamento e segmentação. Primeiro a imagem colorida é convertida em tons de cinza, com um canal simples de 8 bits. Em seguida é aplicado um filtro gaussiano para suavizar a imagem, reduzindo assim os ruídos. Então, é feita a segmentação dos contornos e

gerado uma imagem binária aplicando o detector de bordas de Canny, conforme pode ser visto na Figura 51.

Figura 51 – Imagens: (a) Original. (b) Tons de cinza. (c) Suavizada. (d) Binária



Fonte: Elaborado pelo autor, 2017.

Como passo final, são extraídas as informações dos contornos da imagem binária aplicando o algoritmo seguidor de borda de Suzuki e Abe (1985). Cada contorno gera uma lista de pontos, onde cada ponto possui coordenadas x e y.

4.1.2 Detecção das Linhas Verticais e Horizontais

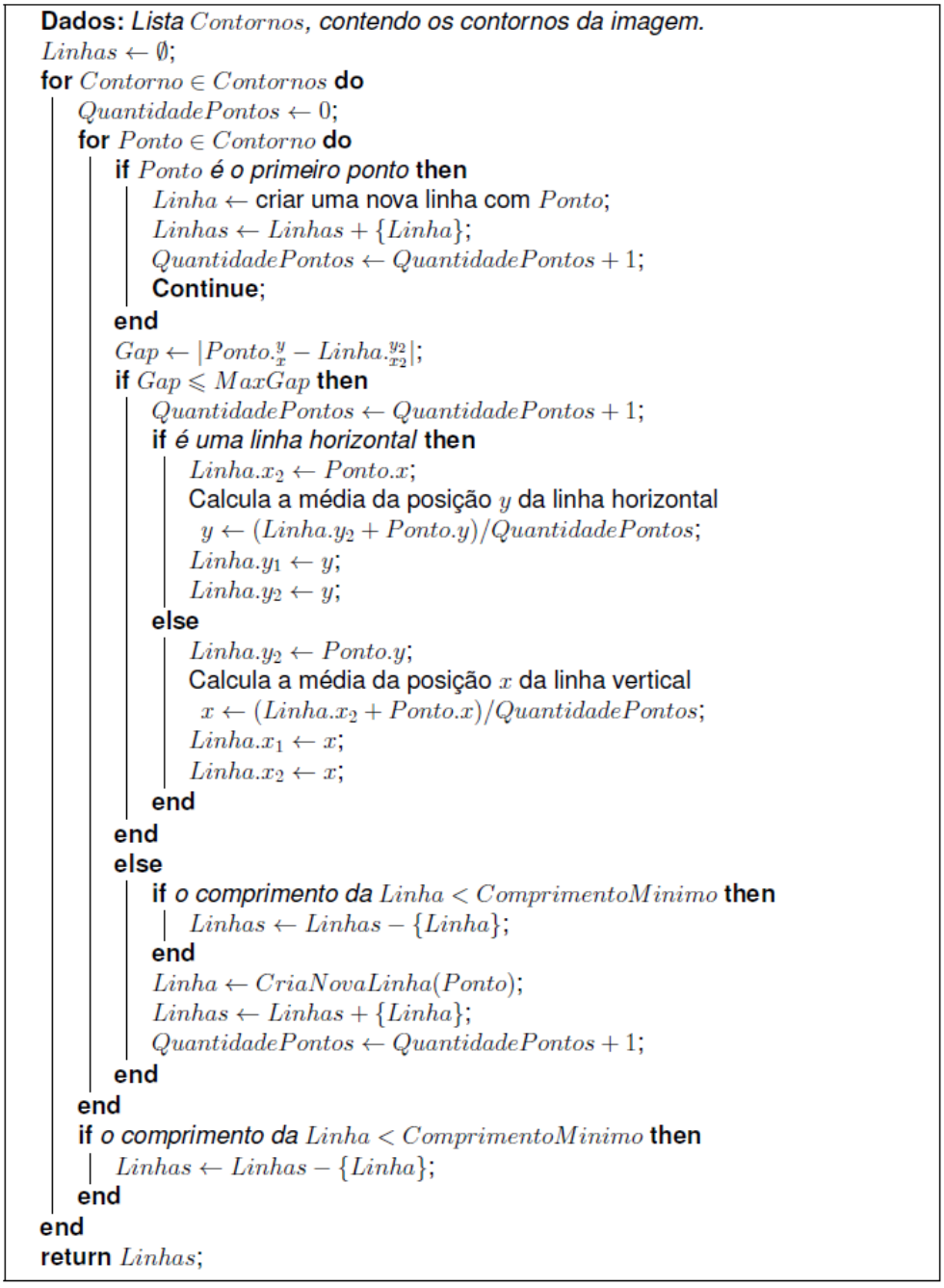
As linhas verticais e horizontais podem ser detectadas analisando as listas de pontos obtidas pelo algoritmo de Suzuki e Abe (1985), conforme algoritmo da Figura 52.

O algoritmo inicia com uma lista de linhas vazias. Itera por cada contorno do vetor de contornos, e para cada contorno, itera no vetor de pontos que o forma. Uma linha é formada por um ponto inicial com as coordenadas (x_1, y_1) e por um ponto final com as coordenadas (x_2, y_2).

O primeiro ponto de cada vetor de pontos inicia a primeira linha. A linha aumenta seu comprimento com a mudança das coordenadas do ponto final, conforme o algoritmo segue analisando cada ponto do vetor de pontos. O próximo ponto é concatenado com o ponto anterior, desde que a distância (Gap) entre eles não ultrapasse o limiar máximo de distância ($MaxGap$). Este limiar serve para desconsiderar ruídos e leves curvaturas nas linhas, entre o final da linha atual e o próximo ponto a ser concatenando a ela.

O algoritmo também garante que cada linha fique reta, mesmo com $Gap > 0$. Para isto o algoritmo calcula a média entre os pontos concatenados à linha e faz (x_1 = x_2) para linhas verticais, e ($y_1 = y_2$) para linhas horizontais. Assim, por exemplo, em uma linha com curvatura leve, pode ser produzida apenas uma linha, onde sem o Gap poderiam ser produzidas três linhas, cada uma com $1/3$ da curva.

Figura 52 – Algoritmo para detecção das linhas verticais e horizontais



```
Dados: Lista Contornos, contendo os contornos da imagem.
Linhas ← ∅;
for Contorno ∈ Contornos do
    QuantidadePontos ← 0;
    for Ponto ∈ Contorno do
        if Ponto é o primeiro ponto then
            Linha ← criar uma nova linha com Ponto;
            Linhas ← Linhas + {Linha};
            QuantidadePontos ← QuantidadePontos + 1;
            Continue;
        end
        Gap ← |Ponto.y_x − Linha.y2_x2|;
        if Gap ⩽ MaxGap then
            QuantidadePontos ← QuantidadePontos + 1;
            if é uma linha horizontal then
                Linha.x2 ← Ponto.x;
                Calcula a média da posição y da linha horizontal
                 y ← (Linha.y2 + Ponto.y)/QuantidadePontos;
                Linha.y1 ← y;
                Linha.y2 ← y;
            else
                Linha.y2 ← Ponto.y;
                Calcula a média da posição x da linha vertical
                 x ← (Linha.x2 + Ponto.x)/QuantidadePontos;
                Linha.x1 ← x;
                Linha.x2 ← x;
            end
        end
        else
            if o comprimento da Linha < ComprimentoMinimo then
                Linhas ← Linhas − {Linha};
            end
            Linha ← CriaNovaLinha(Ponto);
            Linhas ← Linhas + {Linha};
            QuantidadePontos ← QuantidadePontos + 1;
        end
    end
    if o comprimento da Linha < ComprimentoMinimo then
        Linhas ← Linhas − {Linha};
    end
end
return Linhas;
```

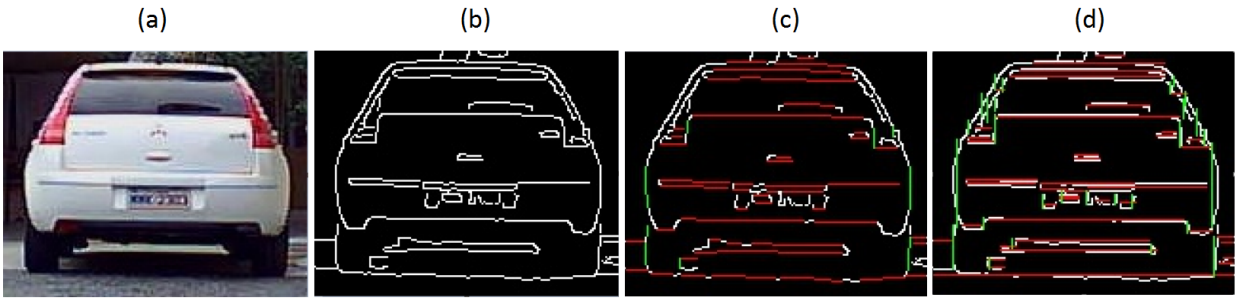
Fonte: Elaborado pelo autor, 2017.

Linhas com comprimento menor do que o limiar $ComprimentoMinimo$ são descartadas. As linhas que não infringem os limiares são adicionadas na lista de linhas, que é retornada ao final do algoritmo da Figura 52.

A Figura 53 mostra um exemplo com o resultado da aplicação do algoritmo da Figura 52 para a detecção das linhas verticais e horizontais em uma imagem contendo um veículo. As parametrizações utilizadas neste exemplo são: $ComprimentoMinimo =$

5 pixels e $MaxGap$ com os valores 0 e 3 pixels para as imagens (c) e (d) respectivamente.

Figura 53 – (a) Imagem original. (b) Detecção de bordas de Canny. (c) Linhas verticais e horizontais detectadas com $MaxGap = 0$. (d) Linhas verticais e horizontais detectadas com $MaxGap = 3$.



Fonte: Elaborado pelo autor, 2017.

As linhas de cor verde indicam linhas verticais detectadas, e as linhas de cor vermelha indicam as linhas horizontais detectadas. É possível perceber que na imagem (c), existem uma quantidade maior de segmentos de linhas detectadas e as linhas detectadas coincidem exatamente com os seguimentos das linhas brancas das bordas, isto devido ao limiar $MaxGap$, possuir o valor zero. Já na imagem (d), como o mesmo limiar possui o valor três, mais segmentos de linhas são concatenados, produzindo menos linhas e com a posição sendo a média em relação a cada segmento que foi concatenado.

4.1.3 Cálculo dos Centroides das Linhas Verticais e Horizontais

A etapa seguinte à detecção das linhas verticais e horizontais é o cálculo do centroide vertical C_v e do centroide horizontal C_h, para as linhas verticais e horizontais respectivamente.

Seja pm o ponto médio de uma linha, ele é calculado através da média das somas das coordenadas x e y do ponto inicial e final da linha, como apresentado na Equação 4.1:

$$pm = \left(\frac{x_1 + x_2}{2}, \frac{y_1 + y_2}{2}\right). \tag{4.1}$$

O centroide C_v das N_v linhas verticais é calculado pela Equação 4.2:

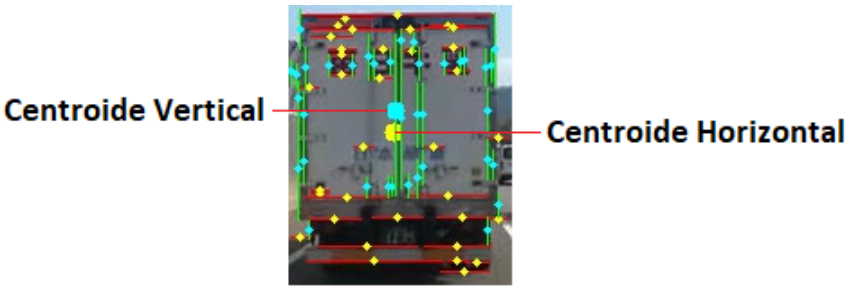
$$C_v = \frac{1}{N_v} \sum_{i=1}^{N_v} pm_i, \tag{4.2}$$

e o centroide C_h das N_h linhas horizontais é calculado pela Equação 4.3:

$$C_h = \frac{1}{N_h} \sum_{i=1}^{N_h} pm_i. \tag{4.3}$$

A Figura 54 mostra um exemplo do cálculo dos pontos médios das linhas verticais e horizontais (pontos menores amarelos e azul claro, respectivamente), e dos dois centroides, pontos maiores no centro da imagem. Quando mais próximos um do outro forem os dois centroides, e mais próximo do centro da imagem, maior é a simetria entre as linhas verticais e horizontais.

Figura 54 – Cálculo dos pontos médios das linhas e do centroide vertical e horizontal



Fonte: Elaborado pelo autor, 2017.

4.1.4 Limiar de Confiança

No contexto deste trabalho, o valor da confiança é um limiar quantitativo que qualifica o quanto uma região candidata pode ser um veículo. Para isto são extraídas e analisadas características desta região candidata. Quanto mais próximo de 100 for o limiar de confiança, maior é a confiança de que aquela região é um veículo. Quanto mais próximo de zero for o valor do limiar, menor será a confiança de que aquela região é um veículo.

O limiar de confiança, denominado de $Confiança$, é inicializado com o valor 100. Este é o valor que indica a máxima confiança de que uma região candidata realmente seja um veículo. Porém, o valor da confiança pode ser decrementado, caso as características analisadas nas linhas verticais e horizontais, ponto médio das linhas e centroides, não atendam a alguns critérios de posicionamento e relacionamento entre si, conforme é descrito nos passos a seguir.

O valor para o decremento do limiar de confiança utilizado foi 25, devido a serem analisados quatro fatores principais, sendo eles: distância de cada centroide em relação ao centro da região candidata; distância entre os centroides; quantidade de linhas verticais simétricas; e quantidade de linhas horizontais simétricas.

O primeiro critério analisado, é o percentual de linhas verticais e horizontais próximas das bordas da região candidata. Para esta tarefa são identificadas e contadas separadamente as linhas verticais e horizontais a uma determinada distância das bordas da região candidata. Se a detecção da região foi bem sucedida pelo algoritmo

de GH, o veículo deve ocupar a maior parte da região, e obviamente seu contorno vai estar próximo das bordas da região. Próximas do topo e da base da região, estarão a maioria das linhas horizontais. Já nas laterais estarão a maioria das linhas verticais.

O algoritmo da Figura 55 efetua a contagem e marca as linhas próximas das bordas da região candidata.

Figura 55 – Algoritmo para contar e marcar as linhas próximas das bordas da região candidata

Dados: $Altura$*: altura da região candidata a veículo em pixels;*
$Largura$*: largura da região candidata a veículo em pixels;*
$Linhas$*: lista contendo as linhas (horizontais ou verticais) a serem analisadas.*

$$Contagem \leftarrow 0;$$

$$Comprimento \leftarrow \begin{cases} Altura & \text{se linhas horizontais} \\ Largura & \text{se linhas verticais} \end{cases}$$

$$LimiteInicial \leftarrow \begin{cases} Comprimento * 0.2 & \text{se linhas horizontais} \\ Comprimento * 0.15 & \text{se linhas verticais} \end{cases}$$

$$LimiteFinal \leftarrow \begin{cases} Comprimento * 0.60 & \text{se linhas horizontais} \\ Comprimento * 0.85 & \text{se linhas verticais} \end{cases}$$

for $Linha \in Linhas$ **do**

$$Posicao \leftarrow \begin{cases} Linha.y_1 & \text{se linhas horizontais} \\ Linha.x_1 & \text{se linhas verticais} \end{cases}$$

  if $Posicao \notin \{LimiteInicial, ..., LimiteFinal\}$ **then**
    Marca $Linha$ como próxima da borda da região candidata a veículo;
    $Contagem \leftarrow Contagem + 1$;
  end
end
return $Contagem$ e linhas atualizadas com marcação;

Fonte: Elaborado pelo autor, 2017.

Conforme o algoritmo, uma linha vertical entra na contagem se a sua coordenada x estiver distante da lateral esquerda e direita a um limiar de pixels menor que 15% da largura da imagem. Já uma linha horizontal entra na contagem se sua coordenada y estiver distante do topo e da base da região candidata um limiar de pixels menor que 20% e 40% respectivamente da altura da imagem. Chegou-se a estes valores de limiares por várias tentativas até atingir um valor que abrangesse as maiores linhas próximas das bordas da região candidata.

O algoritmo da Figura 55 é chamado duas vezes, uma vez para contar as linhas horizontais, $Contagem_h$, e uma vez para contar as linhas verticais, $Contagem_v$. A Figura 56 mostra um exemplo da aplicação do algoritmo. Onde linhas claras no centro da imagem são desconsideradas e as linhas vermelhas e verdes são as linhas hori-

zontais e verticais, respectivamente, marcadas como próximas das bordas do veículo.

Figura 56 – Exemplo da aplicação do algoritmo da Figura 55 para marcação de linhas próximas das bordas

Fonte: Elaborado pelo autor, 2017.

Seja $Percentual_v$ e $Percentual_h$, os percentuais de linhas verticais e horizontais respectivamente, que estão próximas das bordas da região candidata a veículo. Caso o $Percentual_v$ e/ou o $Percentual_h$ seja < 40, a $Confiança$ é penalizada com um decremento de 25 em cada caso.

O próximo passo é calcular a distância entre o centro da região candidata e os centroides. Devido a simetria das linhas verticais e horizontais de um veículo, os centroides não deveriam estar afastados mais de 30% do centro da região candidata. Seja pc o ponto central da região candidata a veículo, suas coordenadas são calculadas com: $pc.x = 1/2 * Largura$ e $pc.y = 1/2 * Altura$ da região candidata. Dados dois pontos, $p_1(x_1, y_1)$ e $p_2(x_2, y_2)$, a distância D entre eles é calculada pela distância euclidiana conforme Equação 4.4:

$$D = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{4.4}$$

Seja D_{pc} a distância entre a origem $(0, 0)$ e o ponto central pc, e D_{max} a distância máxima aceitável entre o centro da região candidata e cada um dos dois centroides, onde $D_{max} = D_{pc} * 0.3$. Seja D_{cv} a distância entre pc e C_v e D_{ch} a distância entre pc e C_h, e ainda seja D_{cvch} a distância entre os centroides C_v e C_h. Caso as distâncias D_{cv} e/ou D_{ch} sejam maiores que D_{max}, ou ainda D_{cvch} seja maior que D_{max} e o deslocamento horizontal absoluto entre os centroides $|C_v.x - C_h.x|$ seja maior que 30 pixels, a $Confiança$ é decrementada em 25 para cada um dos três casos.

Por fim, é calculada a simetria entre as linhas verticais nas proximidades das bordas laterais da região candidata a veículo. A simetria é calculada com base no ponto médio das linhas. Observando um veículo é possível perceber que geralmente

ele possui maior simetria em suas laterais do que na sua base e topo. O algoritmo da Figura 57 mostra o cálculo da simetria.

Figura 57 – Algoritmo para cálculo da simetria entre as linhas verticais

Dados: $Linhas$*: lista contendo as linhas verticais a serem analisadas;*
C_v*: centroide das linhas verticais.*

$Contagem \leftarrow Contagem_e \leftarrow Contagem_d \leftarrow 0$;

$Linhas_v$ = Obtém de $Linhas$ as linhas marcadas como próximas das bordas;
Ordena $Linhas_v$ da esquerda para a direita (coordenada x);

$i \leftarrow 1$;
$Linha_e \leftarrow Linhas_v[i]$;
while $Linha_e.x_1 < C_v.x$ **do**
  $Contagem_e \leftarrow Contagem_e + 1$;
  $y_e \leftarrow \frac{1}{2}(Linha_e.y_1 + Linha_e.y_2)$;
  $D_e \leftarrow C_v.x - Linha_e.x_1$;
  $j \leftarrow |Linhas_v|$;
  $Linha_d \leftarrow Linhas_v[j]$;
  while $Linha_d.x_1 > C_v.x$ **do**
    $Contagem_d \leftarrow Contagem_d + 1$;
    $y_d \leftarrow \frac{1}{2}(Linha_d.y_1 + Linha_d.y_2)$;
    $D_d \leftarrow Linha_d.x_1 - C_v.x$;
    if $(|y_e - y_d| < Limiar_y)$ **and** $(|D_e - D_d| < Limiar_d)$ **then**
      $Contagem \leftarrow Contagem + 1$;
    end
    $j \leftarrow j - 1$;
    $Linha_d \leftarrow Linhas_v[j]$;
  end
  $i \leftarrow i + 1$;
  $Linha_e \leftarrow Linhas_v[i]$;
end

return $Contagem, Contagem_e, Contagem_d$;

Fonte: Elaborado pelo autor, 2017.

As linhas verticais são ordenadas pela coordenada x, assim elas podem ser divididas pelo centroide vertical. O algoritmo pega uma linha à esquerda do centroide, $Linha_e$, calcula seu ponto médio, y_e, e a distância até o centroide, D_e. Depois navega pelas linhas à direita do centroide, $Linha_d$, também calculando o ponto médio, y_d e a distância até o centroide, D_d.

Para as duas linhas serem consideradas simétricas, a diferença vertical absoluta entre os dois pontos médios, $|y_e - y_d|$, devem ser menor do que o limiar $Limiar_y$, e a diferença absoluta entre as distâncias de ambas as linhas até o centroide $|D_e - D_d|$ deve ser menor do que o limiar $Limiar_d$. A quantidade de linhas que atenderem estas condições são contabilizadas em $Contagem$. Também são contabilizadas todas as linhas à esquerda do centroide em $Contagem_e$ e as linhas à direita do centroide em $Contagem_d$. Caso $Contagem$ seja igual a zero ou um dos lados do centroide tenha

mais de 60% das linhas verticais, a $Confiança$ é penalizada com um decremento de 25.

A simetria entre as linhas horizontais é calculada com um algoritmo simples. Para cada linha o algoritmo calcula o ponto médio e a distância horizontal deste ponto com o ponto médio das demais linhas. Se a distância absoluta for menor ou igual a um limiar de distância máxima $limiar_{dmax}$, e o comprimento da linha avaliada for maior ou igual a um limiar de comprimento mínimo $limiar_{cmin}$, a duas linhas são simétricas e é contabilizado no contador $contador_{LSH}$.

Valores baseados em tentativa e erro dos limiares para obter a maioria das linhas simétricas horizontais são: $limiar_{dmax} = 10$ e $limiar_{cmin} =$ 25% da largura da imagem. Um valor de $contador_{LSH} < 3$, penaliza a confiança com um decremento de 25. Caso as penalidades aplicadas produzirem um valor menor do que zero para a confiança, o valor zero é assumido. Com isto é produzido um valor de confiança entre 0 e 100 para uma região candidata a veículo.

4.2 ESTIMATIVA DA DISTÂNCIA ENTRE VEÍCULOS

Para emitir alertas de risco de colisão mais eficientes, é necessário estimar a distância entre o veículo portando a câmera e demais veículos detectados à sua frente. A distância entre os veículos servirá como um dos parâmetros para determinar o nível de risco de colisão que um determinado veículo oferece ao veículo portando o sistema.

Dos quatro trabalhos apresentados no Capítulo 3 que estimam a distância, três deles utilizam semelhança de triângulos utilizando os parâmetros da câmera. Destes três trabalhos, dois foram implementados para testes (Rezaei, Terauchi e Klette (2015) e Tang et al. (2015)), porém nenhum dos dois calculou corretamente utilizando as equações dos respectivos trabalhos.

Devido a isto, neste trabalho optou-se por desenvolver uma solução mais simples, sem utilizar os parâmetros da câmera. A solução é baseada na semelhança de triângulos[1], mais especificamente, na proporção das medidas reais, como a largura do veículo e na distância dele até a câmera, e a estas medidas proporcionais em pixels no plano de imagem.

A calibração da câmera pode ser feita da seguinte maneira: seja V um veículo com a largura L_M conhecida. O veículo V é posicionado a uma distância D_M da câmera. Adquire-se uma foto do veículo nesta distância, e mede-se a largura L_P em pixels do veículo na foto tirada. A distância focal F da câmera pode ser obtida com a Equação 4.5:

$$F = \frac{L_P \text{ x } D_M}{L_M}. \tag{4.5}$$

A distância D de um veículo V_X pode ser estimada por semelhança pela Equação 4.6:

$$D = \frac{L_M \text{ x } F}{L_{XP}}, \tag{4.6}$$

onde L_{XP} é a largura em pixels do veículo V_X. A largura L_M e as distâncias D_M e D são na mesma unidade de medida (por exemplo, metros).

Para calibrar a câmera e fazer alguns testes, foi calculada a largura média de 210 veículos[2] de passeio diferentes, onde obteve-se $\mu = 1,8$ como largura média em metros e $\sigma = 0,15$ como desvio padrão, também em metros. Como o objetivo é evitar colisões, é preferível uma margem de erro de forma que os veículos pareçam mais próximos da câmera do que mais afastados. Assim, L_M é calculado pela Equação 4.7:

[1] http://www.pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv

[2] Fonte: http://g1.globo.com/carros/noticia/2010/11/veja-veiculos-avaliados-pelo-g1.html

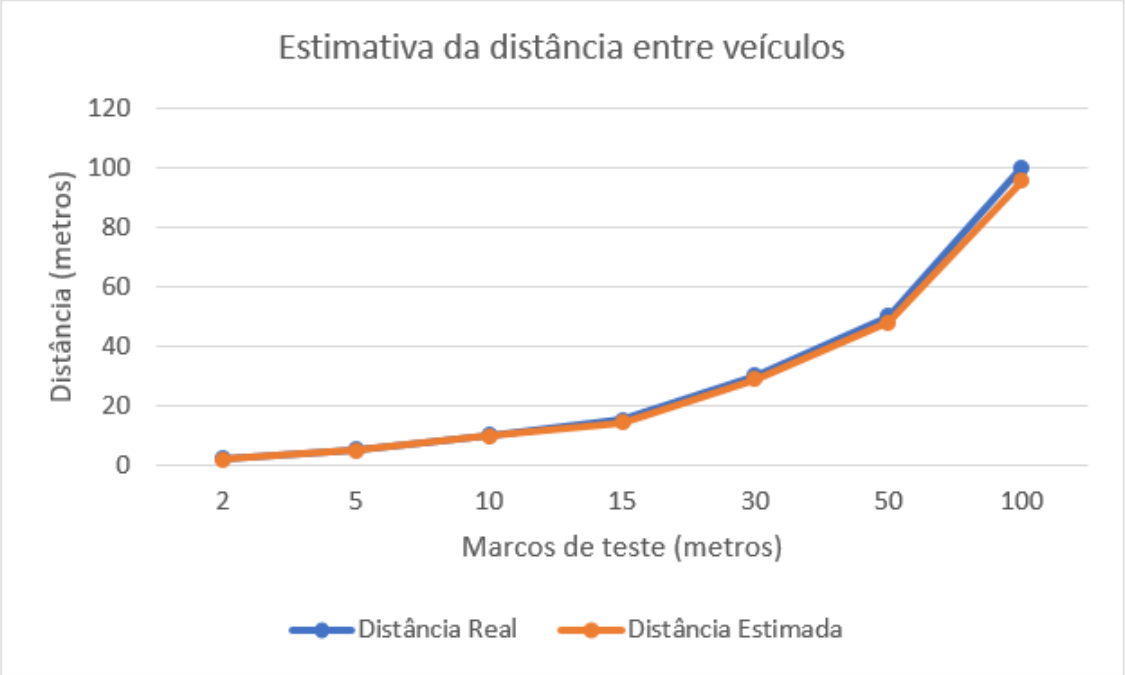
$$L_M = \mu - \sigma, \tag{4.7}$$

onde obteve-se $L_M = 1,65$ metros de largura. Foram executados testes com dois veículos diferentes, um C4 de 1,7 metros de largura e um Sandero de 1,75 metros de largura. Os testes para calibração da câmera foram feitos a 2, 5, 10, 15, 30, 50 e 100 metros de distância da câmera. Foi medida a largura em pixels dos dois veículos em cada um destes marcos de distância e feita a proporcionalidade para obter a largura L_P em pixels da largura L_M. Sendo que o melhor resultado foi obtido com $D_M = 50$ metros e $L_P = 32$ pixels.

Com estes parâmetros foi obtido $E_\sigma = -4,5\%$ de erro médio, assim um veículo a uma distância de 30 e 50 metros da câmera, parecerá como se estivesse a 28,65 e 47,75 metros de distância respectivamente, ou seja, parecerá um pouco mais próximo do que a distância real, o que é um erro tolerável, pois na pior das hipóteses serão disparados alertas de risco de colisão um pouco antecipadamente.

O gráfico da Figura 58 mostra o resultado médio final da comparação das distâncias reais com as distâncias estimadas em cada um dos marcos de teste para os dois veículos utilizados durante a calibração da câmera utilizando os parâmetros conforme descrito anteriormente.

Figura 58 – Resultado final dos testes pós calibragem da câmera, comparando distância real versus distância estimada.

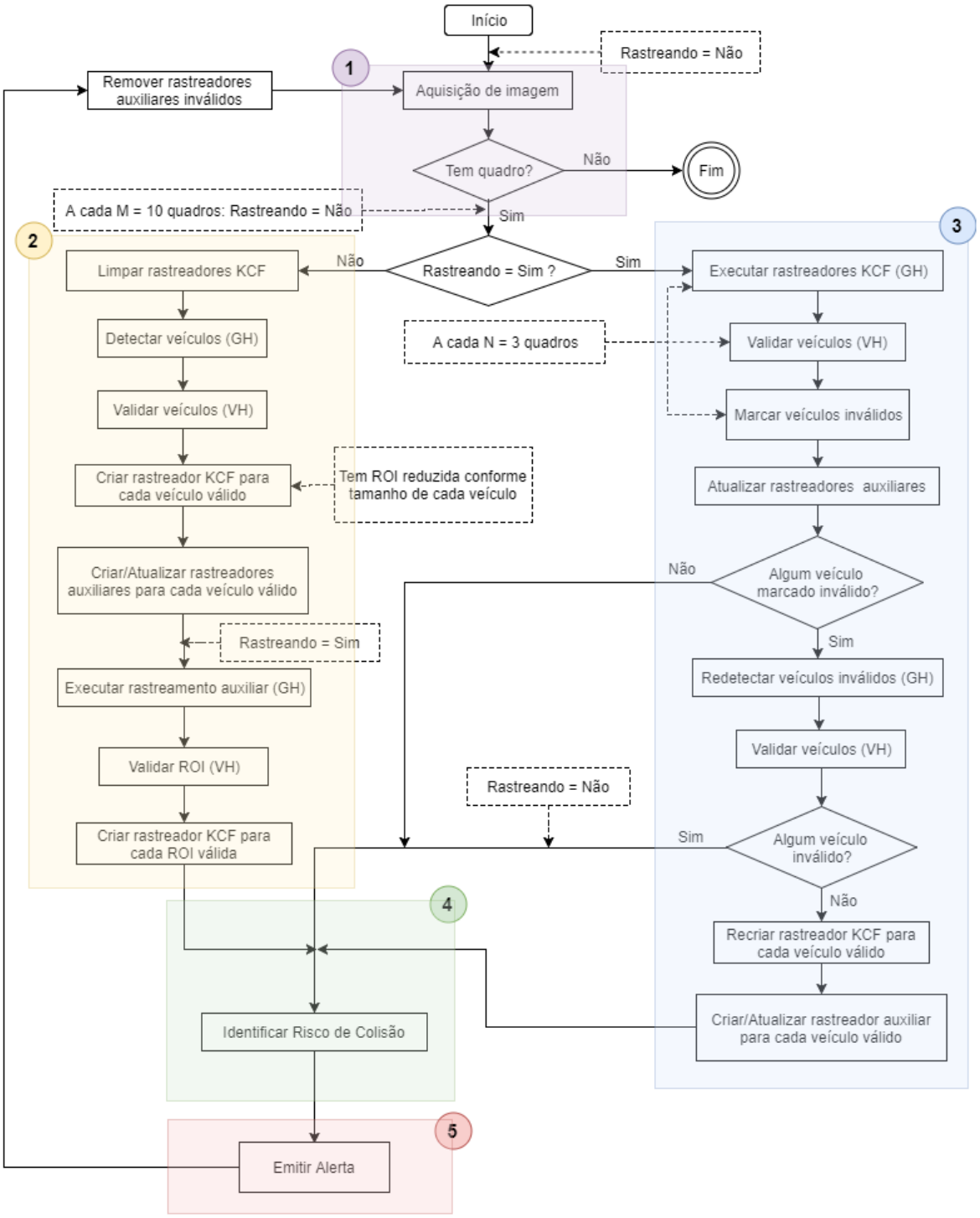


Fonte: Elaborado pelo autor, 2017.

4.3 FLUXO DE FUNCIONAMENTO DO SISTEMA

Nesta seção será apresentado o fluxo para execução do sistema para alerta de colisão veicular. O fluxograma do sistema é mostrado na Figura 59.

Figura 59 – Fluxograma do sistema para alerta de colisão veicular



Fonte: Elaborado pelo autor, 2017.

As etapas base do fluxo do sistema são: 1-Aquisição de imagem, 2-Detecção de veículos, 3-Rastreamento de veículos, 4-Identificação de risco de colisão e 5-Emissão de alertas. As caixas de fundo em cores distintas no fluxograma da Figura 59 representam cada uma destas etapas. O sistema é inicializado com os seguintes estados: não possui nenhum quadro capturado, não há risco de colisão e não está fazendo rastreamento.

4.3.1 Aquisição de Imagem

A aquisição da imagem é feita através de uma câmera monocular. A taxa de aquisição dos quadros deve ser em torno de 30 fps. A câmera é fixada no meio do para-brisa dianteiro, e com um ângulo de inclinação vertical de 90^0 (apontada reto para frente).

Processar toda a área do quadro da imagem adquirida pela câmera se torna custoso computacionalmente e desnecessário tanto para a detecção como para o rastreamento dos veículos, pois aqueles veículos aos quais se tem interesse ocupam geralmente a área central da imagem. Devido a isto são definidas regiões de interesse específicas para cada uma das duas operações: detecção e rastreamento de veículo.

Uma região de interesse, normalmente chamada de ROI, do inglês *Region Of Interest*, é a região de pixels da imagem que se deseja processar em um determinado momento, onde o restante da imagem só tomaria tempo de processamento e poderia influenciar negativamente nos resultados esperados.

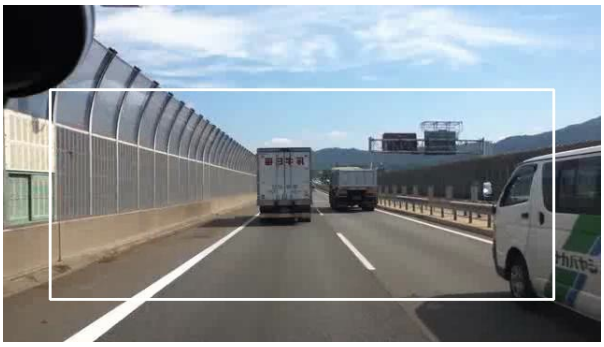
É definida uma ROI menor do que o tamanho do quadro adquirido. Onde a partir desta região serão feitas as detecções e rastreamentos. Seja Q um quadro adquirido de largura W e altura H. Uma ROI $R \in Q$ pode ser definida com algumas propriedades. Seja $x = 0$ e $y = 0$, a primeira coluna e linha respectivamente da matriz de pixels de Q. A ROI R com largura w e altura h para detecção de veículos é definida com os seguintes parâmetros: $x = Limiar_x$, $y = 1/4H$, $w = W - 2x$ e $h = H - 1.5y$. O valor 50 para $Limiar_x$ foi utilizado na maioria dos testes realizados.

Uma ROI R com $Limiar_x = 50$ é representada pelo retângulo (branco) da Figura 60. Para fazer a detecção de veículos, esta região da imagem é convertida para níveis de cinza. As ROIs para rastreamento e para detecção de veículos específicos serão explicadas mais adiante.

4.3.2 Detecção de Veículos

A detecção de veículos ocorre em duas etapas, inicia com a geração de hipótese (GH) e em seguida é feita a verificação de hipótese (HV). A GH é o ponto de partida, pois é ela que fornece inicialmente as coordenadas das regiões candidatas

Figura 60 – Retângulo (branco) representando a ROI para detecção de veículos



Fonte: Elaborado pelo autor, 2017.

a veículos. A detecção é efetuada utilizando uma das seguintes técnicas já discutidas anteriormente (extrator de características + classificador): Haar-like + AdaBoost, MB-LBP + AdaBoost ou HOG + SVM.

Como o objetivo deste sistema é alertar o motorista sobre risco dele colidir seu veículo com os veículos à sua frente, o ideal é que a técnica consiga detectar o máximo de veículos à sua frente, pelo menos aqueles que estiverem mais próximos da câmera.

Na etapa de GH, as regiões detectadas são apenas regiões candidatas a veículos. Por isto, a técnica utilizada nesta etapa deve detectar o máximo possível de candidatos a veículos, mesmo que produza alguns falso positivos. Esta etapa consome muito tempo de processamento, pois o classificador tem que varrer, inclusive em várias escalas diferentes, toda a ROI definida em busca de veículos.

O que não deveria ocorrer nesta etapa é uma alta taxa de falso negativos, isto é, veículos próximos da câmera que não são detectados. Por isto é muito importante que a etapa de GH gere poucos falso negativos e alguns falso positivos. Estes últimos serão eliminados na etapa de VH.

Neste trabalho, a fim de deixar o processo de detecção de veículos mais rápido, e também mais preciso, a detecção é combinada com o rastreamento de veículos. Para cada veículo detectado e validado pela etapa de VH, é criado um rastreador KCF, e um rastreador auxiliar que possui um mecanismo de tolerância a falhas para regiões que desaparecerem subitamente. O funcionamento dos rastreadores será explicado com mais detalhes adiante.

No início de cada detecção a lista de rastreadores é limpa. A detecção de veículos pode ocorrer em três casos:

1. ao atingir o limiar de intervalo máximo de quadros decorridos, L_{IMQ}. Neste caso a detecção é feita sobre toda a ROI R;

2. durante um rastreamento, quando uma região rastreada for rejeitada na verificação de hipótese e precisa ser re-detectada. Neste caso a ROI para detecção é a ROI atual do rastreador, porém expandida em alguns pixels conforme um limiar de expansão, LD_{exp};

3. quando uma região re-detectada não consegue ser detectada ou é rejeita na verificação de hipótese. Para o sistema, o veículo sumiu. O veículo pode ter se deslocado para além da expansão feita no caso 2, ou o detector pode ter falhado. Isto dispara uma detecção do caso 1.

Após uma região ser detectada, ela pode ser um veículo, como também pode ser um falso positivo. A fim de eliminar os falso positivos, a região candidata é validada por um ou dois verificadores de hipótese. Uma região precisa de dois votos verdadeiros para ser considerada uma região válida. O primeiro voto já é do próprio detector que a detectou. O segundo voto é dado primeiramente por uma das duas técnicas de detecção restantes, sempre tendo preferência aquela que possuir o melhor desempenho entre as duas.

A ROI candidata é expandida em alguns pixels conforme o limiar de expansão LD_{exp}, isto deve-se ao fato de que técnicas de detecção diferentes poderem necessitar de ROIs com dimensões diferentes para detectar o mesmo objeto, conforme explicado por Mammeri, Zhou e Boukerche (2016) e isto foi comprovado nos experimentos feitos. Caso o segundo detector retorne verdadeiro, a região é validada.

Caso o segundo detector retorne falso, indicando que não conseguiu detectar, a região é validada pelo validador CCLVH. Para este validador a ROI não é expandida pois ele analisa características específicas de linhas e seus centroides da região para determinar o limiar de confiança se é um veículo ou não. Caso o validador CCLVH valide como verdadeiro, a região é tida como válida, e inválida caso contrário. Assim, cada região tem dois votos verdadeiros para ser considerada válida, um do próprio detector e o segundo por um dos dois validadores.

Para cada veículo válido é criado um rastreador KCF. A ROI do veículo é a ROI alvo do rastreador. Uma ROI é expandida a partir do alvo, conforme um limiar LR_{exp}, a qual será a ROI do fragmento do rastreador, esta ROI é necessária para o rastreador ter um contexto de treinamento e aprendizagem para aquele alvo, conforme explicado por Henriques et al. (2015). Como o rastreador KCF utiliza três canais de cores, as suas ROIs precisam ser extraídas de Q, que é colorido, em vez de R que é em níveis de cinza. Lembrando que o tamanho da ROI do rastreador influencia na sua velocidade de execução.

O passo seguinte é criar ou atualizar um rastreador auxiliar para cada rastreador KCF. Este rastreador auxiliar armazenará algumas informações temporais impor-

tantes da ROI rastreada pelo rastreador KCF. É através das informações temporais armazenadas no rastreador auxiliar que é possível elaborar um mecanismo de tolerância a falhas de uma região rastreada por uma série de quadros. Neste momento é ativado um estado indicando que o rastreamento está ativo, o qual só ocorrerá efetivamente no quadro seguinte à detecção.

4.3.3 Rastreadores Auxiliares

Existe uma situação indesejada que pode ocorrer onde um veículo que vinha sendo rastreado por vários quadros simplesmente some após uma detecção ou durante o próprio rastreamento com o rastreador KCF. Isto pode ocorrer por fatores intrínsecos a cada técnica ou devido à mudanças do ambiente, como por exemplo, a influência da sombra de uma árvore, passar sob uma ponte ou a própria trepidação da câmera acoplada ao veículo (REZAEI; TERAUCHI; KLETTE, 2015) e (KIM et al.,).

A fim de desenvolver um mecanismo tolerável a falhas deste tipo, foi criada uma lista auxiliar de rastreadores. Estes rastreadores auxiliares estão sincronizados com os rastreadores KCF em relação a ROI do objeto alvo. Porém, guardam mais alguns estados necessários que são:

a) ROI: coordenadas x e y, largura e altura da região do objeto alvo (veículo) do rastreador KCF vinculado;

b) Índice do quadro: guarda qual foi o último quadro em que o rastreador auxiliar foi atualizado e consequentemente o rastreador KCF rastreou;

c) Contador de referências: é incrementado em 1 a cada vez (quadro) que a ROI do objeto alvo, vinculada a um rastreador KCF, foi atualizada. Após processar um quadro, é decrementado drasticamente (pela metade) caso o índice do quadro seja menor do que o índice do quadro atual (indica que não foi atualizado no quadro atual). Quando atingir valor zero, é removido da lista de rastreadores auxiliares;

d) Valor de confiança: Baseado nos DOTs de Kim et al. (). Incrementado toda vez que a ROI for atualizada e considerada valida na validação de hipótese. Decrementado toda vez que a ROI for atualizada e considerada inválida na validação de hipótese. Decrementado também, toda vez que for utilizado devido a uma ROI ter desaparecido na detecção ou rastreamento. Neste último caso, seu Índice do quadro é atualizado para o quadro corrente. Quando o valor de confiança atingir zero, o rastreador é eliminado da lista de rastreadores auxiliares;

e) FOE: Indica foco de expansão (valor > 0), baseado em Yang e Zheng (2015). Incrementado caso a ROI do veículo esteja mais próxima da câmera do que

estava no quadro anterior e decrementado (mínimo zero) caso contrário. São analisadas as coordenadas e as dimensões da ROI do quadro atual e comparados com as do quadro anterior.

Após fazer uma detecção, são analisados os rastreadores auxiliares que estão desatualizados, ou seja, seus índices de quadro são menores do que o índice do quadro atual. Como permanecem na lista de rastreadores, foram atualizados em vários quadros e possuem contador de referências e valor de confiança maiores do que zero.

Assim, são feitas as validações de hipótese de cada ROI que esteja neste estado, e para cada ROI validada com sucesso, é criado um rastreador KCF e vinculado à ela. Lembrando que os classificadores auxiliares são penalizados quando utilizados nestas circunstâncias tendo o seu valor de confiança decrementado.

4.3.4 Rastreamento de Veículos

No sistema, quando o estado de rastreamento estiver ativo, o rastreamento dos veículos é feito com o algoritmo KCF. Este algoritmo possui alta velocidade de rastreamento, chegando a processar centenas de quadros por segundo (HENRIQUES et al., 2015). O objetivo do rastreamento é encontrar no quadro corrente a ROI mais parecida com a ROI do veículo encontrada no quadro anterior.

A ROI encontrada no quadro corrente pelo rastreamento é uma região candidata a veículo, ou seja, é uma GH e deverá passar por uma validação de hipótese. A VH da região candidata obtida no rastreamento, segue o mesmo esquema usado na VH da detecção, que já foi discutida anteriormente. Sempre dando preferência para a técnica de detecção com o melhor desempenho para fazer a segunda validação.

Devido ao bom desempenho do rastreador KCF, ao deslocamento do veículo mudar pouco de um quadro para outro, e ao rastreamento também possuir um certo custo computacional (principalmente com vários veículos sendo rastreados simultaneamente), percebeu-se que não há necessidade de efetuar o rastreamento e a VH a cada quadro durante o intervalo de rastreamento. Assim, foi definido um limiar, L_{VH}, de intervalos de quadros para executar o rastreamento e a VH durante o rastreamento.

Na VH durante o rastreamento, caso uma ROI seja rejeitada, ela é marcada como inválida e o sistema tentará redetectá-la posteriormente. Assim, pretende-se despender menos recursos computacionais detectando apenas a área de um veículo do que os recursos computacionais que seriam despendidos para detectar o veículo analisando toda a ROI R.

Os rastreadores auxiliares também são atualizados com as novas coordenadas e dimensões das ROIs obtidas pelos rastreadores KCF. Para as ROIs inválidas o

rastreador auxiliar é penalizado tendo o seu valor de confiança decrementado, caso contrário este valor é incrementado.

Seguindo com o processo de rastreamento, são analisadas as ROIs que ficaram inválidas pela VH após o rastreamento dos rastreadores KCF executarem e é tentado detectá-las novamente. Para isto, cada ROI inválida é expandida conforme o limiar LD_{exp}, e é feita a detecção apenas nesta ROI, com a mesma técnica de detecção usada no passo de detecção. São feitas duas tentativas, caso a primeira falhe, na segunda tentativa a ROI é expandida proporcionalmente em mais 50% da sua área.

A nova ROI detectada deve ter pelo menos a metade da sua área interseccionando com a área antiga (a inválida) para ir para a etapa de VH. Caso detecte mais de um veículo, é usada a ROI com maior área de interseção com a ROI do rastreador KCF que falhou (mínimo de 50% da área também). Caso não exista interseção, é tido como não detectado.

Para as regiões detectadas, é feita a VH da mesma forma como é feita na etapa de detecção já descrita anteriormente na Seção 4.3.2. Para as regiões detectadas e que obtiveram sucesso pelo validador de hipótese, são recriados os rastreadores KCF e recriados (ou atualizados) os rastreadores auxiliares, para que possam continuar armazenando os estados dos veículos rastreados.

Porém, caso alguma região não seja redetectada ou seja invalidada pela VH, o processo de rastreamento é interrompido, e o estado de rastreamento é desativado. Isto significa que no quadro seguinte ocorrerá uma detecção completa, assim todos os rastreadores KCF serão recriados e os rastreadores auxiliares serão recriados ou atualizados.

4.3.5 Identificação de Risco de Colisão

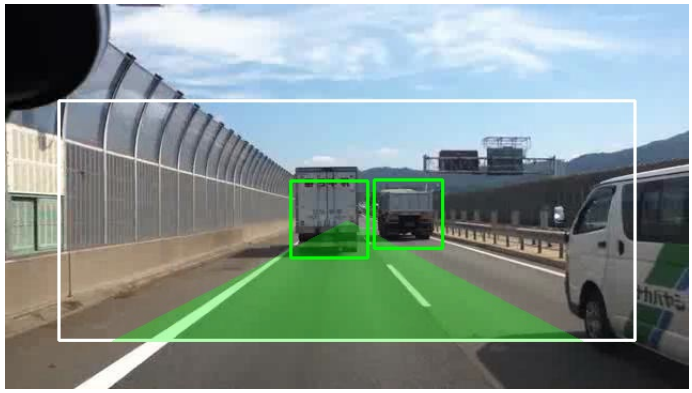
Esta etapa é uma das mais críticas do sistema, pois é nela que é estimado se existe risco de colisão. Isto pode resultar na emissão de um alerta que pode evitar um grave acidente, e até salvar as vidas dos ocupantes dos veículos.

Primeiramente é definida um zona de risco de colisão à frente do veículo. A zona de risco, em alguns aspectos, como a sua forma e o monitoramento dos veículos que a intersectam, é baseada no trabalho de Yang e Zheng (2015). Veículos que interseccionam esta zona são potenciais candidatos a risco de colisão e devem ser monitorados pelo sistema.

A zona de risco se parece com a figura de um triângulo, com a base sendo a base da ROI R do quadro da imagem, afastada em um limiar de pixels, $afastamento_{zr}$, em cada lado e o fim do triângulo alcançando o centro de R. A Figura 61 mostra um exemplo da zona de risco, com $afastamento_{zr} = 50$, como um triângulo verde com

transparência. Esta forma da zona de risco à frente do veículo é suficiente para que o

Figura 61 – O triângulo marca a zona de risco de colisão



Fonte: Elaborado pelo autor, 2017.

sistema consiga controlar veículos com potenciais risco de colisão.

Os seguintes critérios são analisados para estimar o risco de colisão:

a) risco alto: existir um ou mais veículos no estado de foco em expansão (FOE), interseccionando a zona de risco, e a uma distância em metros menor ou igual a um limiar de $risco\_alto$;

b) risco moderado: existir um ou mais veículos com foco em expansão, interseccionado a zona de risco, e a uma distância em metros menor ou igual a um limiar de $risco\_moderado$;

c) seguro: qualquer outra situação.

O resultado desta etapa é um valor numérico de 0 até 2, significando risco alto, risco moderado ou seguro, respectivamente. Como base no valor deste resultado, é possível acionar a etapa de emissão de alertas e notificar de forma adequada o motorista.

4.3.6 Emissão de Alertas

Esta é a etapa de interação do sistema com o motorista, onde o alerta é emitido. O alerta emitido pode ser tanto sonoro como visual, ou ambos. O propósito do alerta visual é para pessoas com limitações auditivas, porém exige que o motorista tire o foco da estrada para olhar a tela do sistema. O propósito do alerta sonoro é bem mais amplo, pois evita que o motorista se distraia, tendo que desviar o olhar da estrada para a tela do sistema, e permite que facilmente todos os ocupantes do veículo tomem conhecimento da situação de risco de colisão.

Os alertas são emitidos com as características de cor de fundo, texto e efeito sonoro conforme apresentado na Figura 62. As cores e textos são baseados no traba-

Figura 62 – Tipos de alertas e suas características visuais e sonoras

	Risco alto	Risco moderado	Seguro
Cor de fundo	Vermelho	Amarelo	Verde
Texto	Risco Alto	Risco Moderado	Seguro
Exemplo	Risco Alto	Risco Moderado	Seguro
Som	Intermitência curta	Intermitência longa	Sem som

Fonte: Elaborado pelo autor, 2017.

lho de Wu, Lin e Lee (2012). É utilizado a cor vermelha para risco crítico, a cor amarela para risco moderado e a cor verde quando não há risco de colisão.

Na pesquisa bibliográfica efetuada sobre a temática deste trabalho, não foi encontrado nenhum trabalho que descrevesse os critérios adotados para a emissão de alertas de risco de colisão sonoro. Devido a isto foi definido pelo próprio autor deste trabalho de pesquisa o tipo de som para alerta de risco alto e risco moderado, conforme explicado a seguir.

O som[3] emitido para risco alto de colisão possui intermitências com sinal mais longo e agudo, dando uma impressão de maior risco e perigo ao motorista. Já o som[4] de risco moderado possui intermitências com sinal mais curto e espaçados dando uma impressão mais branda, porém despertando a atenção do motorista.

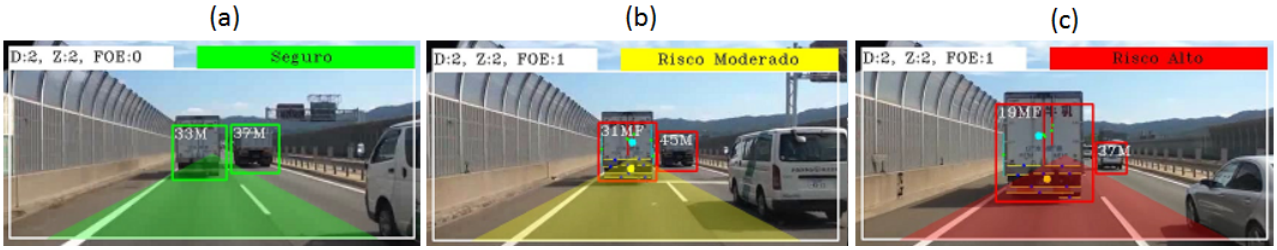
A Figura 63 mostra exemplos práticos do sistema emitindo alertas de colisão. No canto superior esquerdo de cada figura são exibidos alguns estados, onde: **D** indica a quantidade de veículos detectados no quadro; **Z** indica a quantidade de veículos intersectando a zona de risco; e **FOE** a quantidade de veículos dentro da zona de risco em foco de expansão, isto é, se aproximando do veículo.

Na Figura 63(a) o sistema indica que não há risco de colisão, pois mesmo havendo dois veículos a uma distância menor do que o limiar de $risco\_moderado$ (50 metros neste exemplo), nenhum deles está em foco de expansão. Já na Figura 63(b) o sistema está emitindo alerta de colisão com Risco Moderado devido a ter um veículo dentro da zona de risco, com foco em expansão e com distância menor que o limiar de $risco\_moderado$.

Os veículos com foco em expansão possuem um indicativo com a letra **F** após a marcação da distância. Finalmente na Figura 63(c) o sistema está emitindo alerta de

[3] Som de risco alto: https://drive.google.com/open?id=0B9DOjnXN1LOBa093TzVHbEp6SEE
[4] Som de risco moderado: https://drive.google.com/open?id=0B9DOjnXN1LOBc3pXREE0MTRyZlk

Figura 63 – Exemplos de emissão de alerta de risco de colisão. (a) Seguro. (b) Risco moderado. (c) Risco Alto



Fonte: Elaborado pelo autor, 2017.

colisão com Risco Alto. Isto devido a ter um veículo dentro da zona de risco, com foco em expansão e com distância menor que o limar $risco\_alto$ (30 metros neste exemplo).

4.4 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Neste capítulo foi apresentado o fluxo de funcionamento do sistema para alertas de colisão veicular. A emissão dos alertas precisa ser eficiente, ou seja, ser emitida quando realmente existir algum risco de colisão com os veículos à frente e os alertas precisam ser disparados o mais rápido possível. Assim o motorista do veículo perceberá rapidamente a situação de risco e poderá tomar de forma antecipada ações para evitar acidentes como uma colisão com o veículo à sua frente.

Para melhorar a eficiência e emitir alertas rápidos, foram apresentadas neste capítulo algumas metodologias, onde algumas delas são propostas inéditas deste trabalho. Uma das novas metodologias propostas é a técnica CCLVH, que auxilia na etapa de VH a eliminar falso positivos. Diminuir a quantidade de falso positivos melhora tanto a qualidade dos alertas emitidos, pois são disparados alertas apenas por regiões que realmente são veículos, como contribui para emissão de alertas mais rápidos, pois terão menos ROIs para analisar e rastrear.

Outra metodologia apresentada por este capítulo, é a estimativa da distância entre os veículos, utilizando a proporção da largura do veículo e da sua distância da câmera no mundo real com estes valores em pixels no plano de imagem. Conhecendo a distância aproximada entre os veículos possibilita emitir alertas mais eficientes.

Uma das contribuições mais importantes deste capítulo, é o emprego de um modelo de detecção dos veículos com VH e rastreamento de veículos intercalados com a detecção, e um mecanismo de tolerância a falhas com a restauração dos veículos que desapareceram subitamente. O modelo permite o emprego de qualquer algoritmo para detecção, rastreamento e VH.

O fluxo de execução proposto para o sistema potencializa principalmente a velocidade de execução, e também em vários casos a precisão, dos algoritmos utilizados.

Isto porque cada um deles é utilizado no momento mais adequado para sua função e somente sobre a região necessária.

Outra contribuição importante apresentada neste capítulo é uma proposta para a emissão de alertas de risco de colisão alto e moderado, tanto sonoros como visuais que é feita utilizando uma zona de risco que monitora veículos aumentando a proximidade da câmera. Isto é feito utilizando o conceito de foco de expansão (FOE) e da estimativa da distância dos veículos em relação a câmera.

5 TESTES E RESULTADOS

Nesta seção serão discutidos os testes realizados e os resultados obtidos na detecção, verificação, rastreamento e emissão de alertas para o risco de colisão veicular. Foram feitos testes comparando técnicas para geração de hipótese (GH) e técnicas para verificação de hipótese (VH). Também foram realizados testes sem e com rastreamento dos veículos, outro teste é relacionado a emissão de alertas para risco de colisão. Sendo que a metodologia empregada para o rastreamento de veículos da Seção 4.3 e a verificação de hipótese com a técnica CCLVH, serem as principais contribuições deste trabalho.

Os testes foram implementados com a IDE Microsoft Visual Studio Community 2017, e foi utilizada a linguagem de programação C++. Também foi compilada e utilizada a API OpenCV versão 3.2.0[1]. O computador utilizado foi um notebook Dell Inspiron com processador Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90 GHz, com 16 GB de memória RAM e sistema operacional Microsoft Windows 10 Pro 64 bits.

A fim de facilitar as terminologias relacionadas as técnicas utilizadas para os testes, são adotados os seguintes termos:

a) a técnica Haar-like com AdaBoost, proposta por Viola e Jones (2004), será tratada como Haar+AdaBoost;

b) a técnica MB-LBP com AdaBoost, proposta por Liao et al. (2007), será tratada como MB-LBP+AdaBoost; e

c) a técnica HOG com SVM, proposta por Dalal e Triggs (2005), será tratada como HOG+SVM.

5.1 TREINAMENTO DOS CLASSIFICADORES

O treinamento dos classificadores foi realizado utilizando a API OpenCV. Esta API exige para treinamento uma lista contendo as anotações das coordenadas e as dimensões dos objetos a serem treinados para classificação, no caso deste trabalho, são as regiões dos veículos nos quadros de imagens.

A fim de tornar mais práticas e precisas as anotações dos objetos veículos nas amostras para treinamento, foi desenvolvida uma ferramenta[2] visual que permitisse marcar estas regiões com o cursor do mouse. A ferramenta também permite para-

[1] OpenCV: http://opencv.org/

[2] https://drive.google.com/open?id=0B9DOjnXN1LOBekpYLUowb3JYckU

metrizar e chamar a execução do treinamento executando os aplicativos auxiliares da API OpenCV. As funcionalidades para marcação das regiões dos veículos servem tanto para o classificador AdaBoost como para o classificador SVM.

Para amostras positivas, foram anotados 4.736 partes traseiras de veículos das bases de imagens iROADS Set 10[3], KITTI[4], Caltech [5] e MIT CBCL [6]. As imagens anotadas foram convertidas para níveis de cinza e foram padronizadas para o tamanho 32 x 32 pixels, para Haar+AdaBoost e MB-LBP+AdaBoost, e 64 x 64 pixels para SVM.

Foram obtidas também 4.754 amostras de imagens negativas da internet[7]. Nenhuma destas amostras continham veículos e também foram convertidas para níveis de cinza.

As seguintes parametrizações foram efetuadas para o treinamento com Haar +AdaBoost e MB-LBP+AdaBoost. A taxa de acerto desejada foi configurada em 0,999, para a taxa máxima de alarmes falsos estipulou-se 0,5% e o número máximo de estágios foi de 20 estágios. Estes parâmetros foram utilizados devido a ter vários trabalhos relatando bom desempenho utilizando eles, como por exemplo: Lienhart e Maydt (2002); Wen et al. (2015) e Mammeri, Zhou e Boukerche (2016). Quanto ao treinamento do classificador SVM, foi utilizado uma SVM linear com as mesmas amostras positivas e negativas.

A Tabela 4 mostra o tempo que cada técnica levou para fazer o treinamento utilizando um total de 9.490 amostras. Através da Tabela 4 é possível perceber que

Tabela 4 – Tempo de treinamento por técnica de extração de características e classificador

Técnica	Tempo de treinamento
Haar+AdaBoost	25 dias
MB-LBP+AdaBoost	1h35min
HOG+SVM	≈ 5min

Fonte: Elaborado pelo autor, 2017.

há grandes diferenças no tempo de treinamento entre as técnicas.

5.2 TESTES EM AMBIENTE COM TEMPO BOM

Foram realizados seis experimentos separadamente na base de imagens iROADS Set 10 Daylight. Nesta base o clima é de tempo bom com sol, produzindo sombra

[3] iROADS: https://cerv.aut.ac.nz/set-10/
[4] KITTI: http://www.cvlibs.net/datasets/kitti/raw_data.php
[5] Caltech: http://www.vision.caltech.edu/html-files/archive.html
[6] MIT CBCL: http://cbcl.mit.edu/cbcl/software-datasets/CarData.html
[7] https://github.com/handaga/tutorial-haartraining/tree/master/data/negatives

do lado esquerdo dos veículos (sombra localizada entre 7 e 8 horas). Possui veículos de passeio, vans e caminhões, ultrapassagens e oclusões ao fundo. A base possui 903 quadros. Foram contados 2.240 objetos veículos (algumas oclusões foram consideradas) por um operador humano.

Os três primeiros experimentos foram feitos utilizando as técnicas Haar + AdaBoost, MB-LBP+AdaBoost e HOG+SVM, apenas com geração de hipótese (GH). Nestes experimentos não houve verificação de hipótese (VH). Isto foi feito para verificar a precisão da detecção e o tempo de execução de cada uma das técnicas de forma isolada.

As preferências por Haar+AdaBoost e MB-LBP+AdaBoost na fase de VH é devido a elas apresentarem os melhores desempenhos nos três primeiros experimentos, respectivamente. Conforme pode ser visto nas Tabelas 5 e 6.

Nos três últimos experimentos, foi seguido o fluxograma da Seção 4.3, onde foi feita a fusão de cada uma das três técnicas com a técnica de rastreamento e verificação de hipótese (VH). A detecção foi feita individualmente para cada uma das três técnicas acima, com intervalos máximo de quadros com o limiar $L_{IMQ} = 10$ quadros.

Após a detecção ocorrer em um quadro, o rastreamento foi feito com a técnica KCF. Onde um valor 3 (a cada três quadros) para o limiar L_{VH}, apresentou o desempenho apresentado nas Tabelas 5 e 6, mantendo precisão e velocidade de processamento. Os rastreadores auxiliares foram utilizados sempre que, após efetuar uma detecção em um quadro, algum veículo desaparecia (não era detectado), porém possuía valor de confiança e contador de referência maior do que zero.

A verificação de hipóteses foi feita em duas situações, inclusive podendo ter checagem dupla. A primeira situação é sempre após detectar veículos. Primeiro foi verificado com Haar+AdaBoost (menos quando esta já era a técnica de detecção, neste caso foi utilizado MB-LBP+AdaBoost) e caso a região fosse reprovada, foi feita uma segunda checagem com a técnica CCLVH. Caso a região seja aprovada por uma das duas técnicas, ela é tida como aprovada pela VH, por possuir dois votos a favor e um contra.

Na segunda situação, a VH foi feita durante o rastreamento dos veículos. Porém não a cada quadro, pois os veículos não mudam significativamente suas dimensões e posicionamento de um quadro para outro. Foi utilizado o limiar L_{VH}, configurado com três quadros de intervalo para VH.

Fica evidente que nos últimos três experimentos são adicionados passos extras se comparado com os três primeiros experimentos, e que em tese, isto pode deixar o processo mais lento, o que seria indesejado. Esta situação é por vezes um desafio

para sistemas de reconhecimento que precisam executar em tempo hábil (próximo de 30 quadros por segundo para detecção de veículos).

Seja VP a quantidade de verdadeiro positivos, FN a quantidade de falso negativos e FP a quantidade de falso positivos. Seja ainda, PV = (VP + FN), a população total de veículos existentes na base de testes. A taxa de verdadeiro positivos (TVP) é dada por VP/PV, a taxa de falso negativos (TFN) é dada por FN/PV e a taxa de falso positivos (TFP) é dada por FP/(FP+VP).

Foi comparado o desempenho quanto ao percentual da TVP, TFN e TFP entre as técnicas. Também foi comparado o tempo de processamento em quadros por segundo (FPS) entre as técnicas. A Tabela 5 mostra o resultado dos seis experimentos. Pelos resultados apresentados é possível perceber que com o uso dos rastreadores

Tabela 5 – Resultados dos experimentos utilizando as técnicas Haar+AdaBoost, MB-LBP+AdaBoost e HOG+SVM em ambiente com tempo bom

Técnica	TVP (%)	TFN (%)	TFP (%)	FPS
Haar+AdaBoost	95,63	3,62	5,44	17,36
MB-LBP+AdaBoost	94,38	4,36	9,08	15,18
HOG+SVM	59,68	40,31	9,05	3,89
Haar+AdaBoost+Rastreadores+VH	99,74	0,25	0,87	41,71
MB-LBP+AdaBoost+Rastreadores+VH	99,03	0,97	3,54	27,75
HOG+SVM+Rastreadores+VH	67,77	32,22	3,79	13,37

Fonte: Elaborado pelo autor, 2017.

e verificação de hipótese, houve um aumento de 4,11%, 4,65% e 8,09% na taxa de verdadeiro positivos para Haar+AdaBoost, MB-LBP+AdaBoost e HOG+SVM, respectivamente. De forma inversa, houve decremento na taxa de falso negativos e falso positivos, o que é desejado.

Outro ponto importante a ser observado é o aumento na velocidade de processamento em quadros por segundo em 140,26%, 82,8% e 243,7% para Haar+AdaBoost, MB-LBP+AdaBoost e HOG+SVM respectivamente. Esta melhora no tempo de processamento deve-se principalmente a dois fatores principais conforme descrito adiante.

Primeiramente, o rastreamento feito com os rastreadores KCF e auxiliares executam em média a mais de 100 FPS independentemente do método para detecção, já a detecção é mais lenta, tomando como base a detecção mais rápida, que foi Haar+AdaBoost, teve média de 17 FPS.

O segundo fator, o qual é diretamente influenciado pelo primeiro fator, é que em média foram feitos 88% de rastreamento contra 12% de detecção para obtenção das

ROIs dos veículos com as duas técnicas que utilizam o classificador AdaBoost, e 85% contra 15% para a técnica que utiliza o classificador SVM, respectivamente.

A VH analisando apenas a ROI do veículo com Haar+AdaBoost leva em média 1,47 ms, enquanto que com CCLVH leva em média 3 ms, onde destes 1,41 ms são da aplicação de Canny e a extração das listas de contornos e o restante do tempo é gasto com a heurística da técnica em si. Porém como o fluxo de execução rastreia muito mais do que detecta, o tempo de processamento é pouco impactado pela VH.

Conforme destacado na Tabela 5, as técnicas Haar+AdaBoost e MB-LBP + AdaBoost com rastreadores individuais para cada veículo, mesmo com a VH, obteve a melhor precisão e maior velocidade de processamento.

Um fato interessante foi que a contagem de objetos veículos classificados corretamente cresceu de 2.240 (contagem por operador humano) para 2.363 (MB-LBP) e 2.279 (Haar-like+AdaBoost), houve aumento de 5,5% e 2,63% respectivamente. Isto deve-se ao mecanismo de rastreamento (KCF e rastreadores auxiliares), pois regiões com oclusões onde o classificador falharia, o rastreamento continua achando a ROI e certos quadros dentro do intervalo de rastreamento onde o classificador falha, os rastreadores não falham.

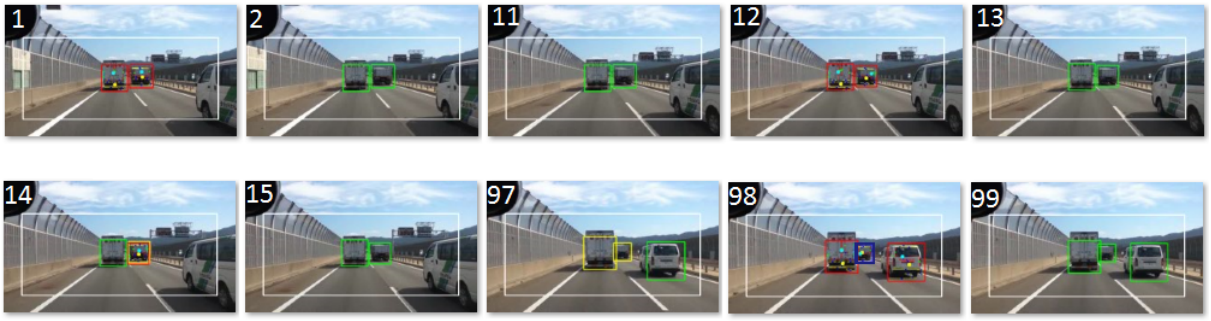
Com os resultados destes experimentos pode ser percebido que utilizar técnicas de rastreamento que sejam precisas e rápidas, como é o caso da KCF, intercaladas com a técnica de detecção, pode melhorar a precisão da detecção e o tempo de processamento.

A Figura 64 mostra um exemplo de como o esquema de detecção funciona através de uma sequência de dez quadros. Cada imagem possui no canto superior esquerdo o número do quadro processado. Região com retângulo vermelho indica detecção de veículo com uma das três técnicas: Haar+AdaBoost, MB-LBP+AdaBoost ou HOG+SVM e VH com a técnica CCLVH. Região com retângulo verde indica rastreamento com KCF e VH bem sucedida com a respectiva técnica de detecção utilizada. Região com retângulo amarelo indica que a ROI obtida pelo rastreador KCF foi rejeitada pela VH. Região com retângulo amarelo e outro vermelho por cima, significa uma região redetectada com sucesso.

Já a região com retângulo azul indica que uma região detectada ou rastreada anteriormente foi armazenada no rastreador auxiliar, porém o detector não consegue mais detectar esta região no quadro atual. Assim, o rastreador auxiliar entrega esta região para o rastreador KCF manter o rastreamento enquanto a região possuir um limiar de confiança maior do que zero.

Cada vez que uma região armazenada no rastreador auxiliar precisa ser restaurada devido a alguma falha na detecção ou do próprio rastreador KCF e reatribuída ao

Figura 64 – Sequências de detecção com rastreamento



Fonte: Elaborado pelo autor, 2017.

rastreador KCF, ela é penalizada tendo seu limiar de confiança decrementando. Caso o limiar de confiança atingir o valor zero, a região é removida do rastreador auxiliar.

Conforme pode ser visto na Figura 64, o processo inicia no quadro 1, onde é feita a detecção de dois veículos e são criados dois rastreadores KCF, um para a região de cada veículo. O rastreamento ocorre do quadro 2 até o quadro 11, isto devido ao limiar de intervalo para detecção estar configurado de 10 em 10 quadros.

No quadro 12 ocorre a detecção dos dois veículos novamente. No quadro 13 continua o rastreamento. No quadro 14 o veículo da direita tem sua ROI rejeitada pelo validador de hipótese, sendo assim, é solicitado uma nova detecção em torno daquela região. A detecção ocorre com sucesso e no quadro 15 segue o rastreamento das duas regiões.

No quadro 97 as duas regiões rastreadas mais a esquerda são rejeitadas pelo validador de hipótese. No entanto, no quadro 98 ocorre uma detecção e atualiza as regiões dos veículos. Porém, apenas o veículo maior a esquerda é detectado, dos dois que foram rejeitados no quadro anterior. A região do veículo mais à frente não é detectada. Assim, o rastreador auxiliar entra em ação novamente e resgata a última posição desta região e a entrega para o rastreador KCF que consegue rastreá-la conforme consta no quadro 99.

5.3 TESTES EM AMBIENTE COM TEMPO CHUVOSO

Empregando os mesmos dados de treinamento da Seção 5.1 e a mesma metodologia apresentada na Seção 5.2, com ambiente de tempo bom, foram realizados seis experimentos separadamente na base de imagens iROADS Set 10 *RainyDay*, sendo esta um ambiente com tempo chuvoso. Novamente os três primeiros experimentos foram feitos com detecção quadro a quadro, um experimento para cada técnica de detecção separadamente (Haar+AdaBoost, MB-LBP+AdaBoost e HOG+SVM). Nos

outros três experimentos foi seguido o fluxograma da Seção 4.3 para cada uma das técnicas de detecção separadamente.

Como na base de imagens iROADS Set 10 *RainyDay* o clima é de tempo chuvoso, há muitas gotas de chuva no para-brisa. Tem veículos de passeio e vans, ultrapassagens e oclusões ao fundo. A base possui 1.049 quadros. Foram contados 1.574 objetos veículos (algumas oclusões foram consideradas) por um operador humano. Porém um ambiente com tempo chuvoso é mais escuro e mais difícil de visualizar os veículos, principalmente os de cores escuras. A Tabela 6 mostra o resultado dos seis experimentos.

Tabela 6 – Resultados dos experimentos utilizando as técnicas Haar+AdaBoost, MB-LBP+AdaBoost e HOG+SVM em ambiente com tempo chuvoso

Técnica	TVP (%)	TFN (%)	TFP (%)	FPS
Haar+AdaBoost	95,05	4,95	2,38	15,63
MB-LBP+AdaBoost	97,1	2,9	11,46	13,95
HOG+SVM	93,31	6,69	86,47	3
Haar+AdaBoost+Rastreadores+VH	96,11	3,88	0,74	70,96
MB-LBP+AdaBoost+Rastreadores+VH	95,11	4,88	3,29	39,85
HOG+SVM+Rastreadores+VH	88,36	11,63	34,05	16,85

Fonte: Elaborado pelo autor, 2017.

Através da Tabela 6 é possível perceber que com a utilização dos rastreadores e VH, há diminuição significativa na TFP, porém com aumento da TFN e uma leve diminuição da TVP (com exceção de Haar+AdaBoost).

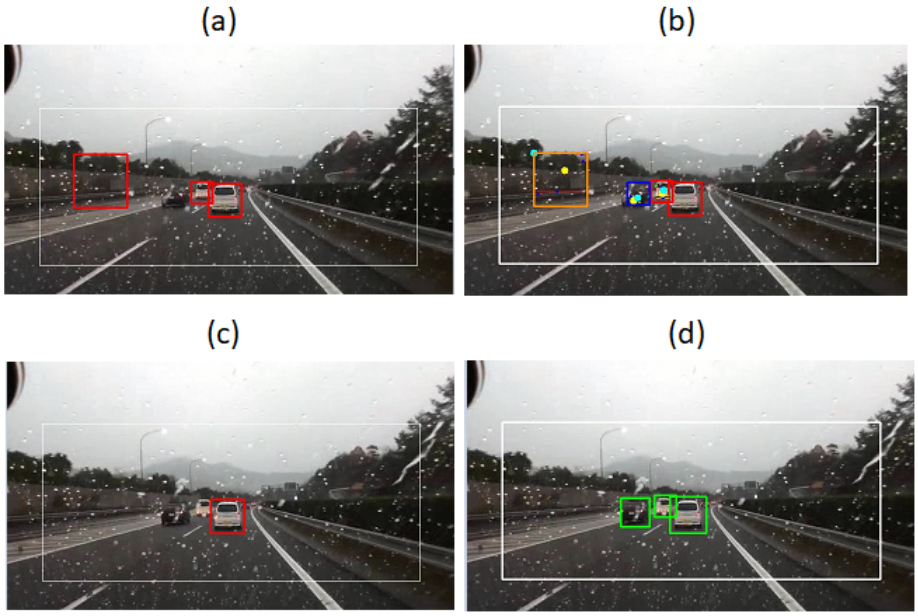
Isto deve-se ao efeito colateral do rastreamento, quando o classificador possui falhas fragmentadas na detecção, onde justamente no quadro da detecção, o classificador falha e propaga o efeito da falha pelo intervalo de rastreamento. Também devido a VH rejeitar mais regiões de veículos com cores escuras em ambiente com baixa luminosidade, por detectar menos linhas verticais e horizontais, a região do veículo fica muito parecida com a região de fundo.

Porém, o rastreamento e a VH mostraram contribuir para uma melhora na detecção e na eliminação de falso positivos mesmo em ambiente chuvoso, conforme pode ser visto na Figura 65. A primeira coluna da figura mostra quadros somente com a detecção e a segunda coluna mostra quadros com rastreamento e VH. Na Figura 65(a) tem-se um falso positivo a esquerda, e um falso negativo para o veículo mais escuro com algumas gotas de chuva o encobrindo.

Na Figura 65(b), a VH elimina o falso positivo e o falso negativo é restaurado pelo rastreador auxiliar que possui um limiar de confiança maior do que zero. Já na

Figura 65(c) tem-se dois falsos negativos, os quais são identificados com sucesso pelos rastreadores KCF na Figura 65(d). Incorporando o uso de rastreadores e da VH, é possível manter uma detecção de objetos veículos mais uniforme, sem desaparecimentos ou aparecimentos abruptos, tornando o processo mais confiável e eficiente.

Figura 65 – (a) e (c) Quadros sem rastreamento e VH. (b) e (d) Quadros com rastreamento e VH



Fonte: Elaborado pelo autor, 2017.

A VH elimina a grande maioria dos falso positivos e entrega as regiões validadas aos rastreadores. Na maioria das vezes, é na detecção que se dará origem a falso positivos e/ou falso negativos. Como é reduzido o número de detecções, devido ao rastreamento, consequentemente reduz a ocorrência de falso positivos e falso negativos.

Isto ficou muito evidente no uso de HOG+SVM com rastreadores e VH, como pode ser visto na Figura 66. Onde em um quadro Q_t (Figura 66(a)) é feita a detecção (todas as regiões retangulares) e aplicado a VH (retângulos laranja são falso positivos e retângulos vermelhos são verdadeiro positivos). No quadro seguinte, Q_{t+1} (Figura 66(b)), são rastreadas apenas as regiões validadas em Q_t, isto pode eliminar novas ocorrências de falso positivos até Q_{t+10}, devido a detecção ser em intervalos de 10 quadros.

Por outro lado, usar intervalo para detecções pode aumentar a quantidade de falso negativos. Isto é um efeito colateral por fazer menos detecções, pois caso o classificador não detecte algum veículo justamente no quadro Q_t, este erro pode ser

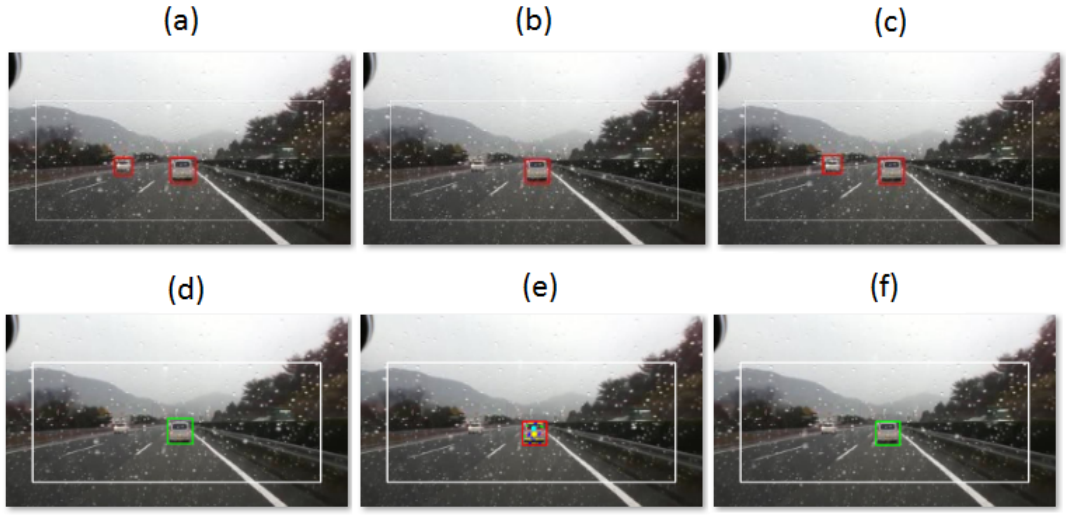
Figura 66 – (a) Quadro Q_t com detecção e VH. (b) Quadro Q_{t+1} rastreando apenas as regiões validadas em Q_t

Fonte: Elaborado pelo autor, 2017.

propagado até o quadro Q_{t+10}. Isto é um problema caso não detecte veículos próximos (ocorreu raras vezes) e sem muita relevância caso não detecte veículos longe (ocorreu poucas vezes).

Esta situação pode ser vista na Figura 67. Onde as figuras (a), (b) e (c) são

Figura 67 – (a) até (c) sequência de quadros com detecção sem VH. (d) até (f) sequência de quadros com rastreamento e VH



Fonte: Elaborado pelo autor, 2017.

uma sequência de quadros com detecção sem VH, e as figuras (d), (e), e (f) são uma sequência de quadros com: rastreamento; detecção com VH; e rastreamento, respectivamente.

É possível perceber que a não detecção do veículo da esquerda, na coluna do meio (letras (b) e (e)), pode aumentar a taxa de falsos positivos significativamente utilizando VH e rastreamento, enquanto fazendo apenas a detecção prejudicará em

apenas um quadro. Contudo, o veículo com maior potencial de risco de colisão, que é o da direita, foi detectado em ambos os casos.

Outro fator importante é a melhora significativa no tempo de processamento, onde a taxa de quadros por segundo aumentou em mais de 500%, 400% e 200% para HOG+SVM, Haar+AdaBoost e MB-LBP+AdaBoost respectivamente, utilizando rastreadores e VH. Isto deve-se novamente ao fato de fazer a detecção em intervalos de quadros e o rastreamento nos quadros dentro do intervalo.

5.4 TESTES DE EMISSÃO DE ALERTAS

Os testes para emissão de alertas de risco de colisão foram feitos utilizando a mesma base de imagens da Seção 5.3. As características visuais e sonoras dos alertas seguem o esquema da Figura 62. A zona de risco e o rótulo que exibe o texto do alerta também seguem o mesmo esquema de cores conforme é a severidade do alerta a ser emitido.

Quanto aos retângulos que identificam os veículos detectados, cada um possui a cor conforme o risco de colisão que oferece, e obedecem também o esquema de cores da Figura 62. Contudo, para Risco Moderado e Risco Alto as bordas dos retângulos são mais grossas, isto para facilitar a sua visualização.

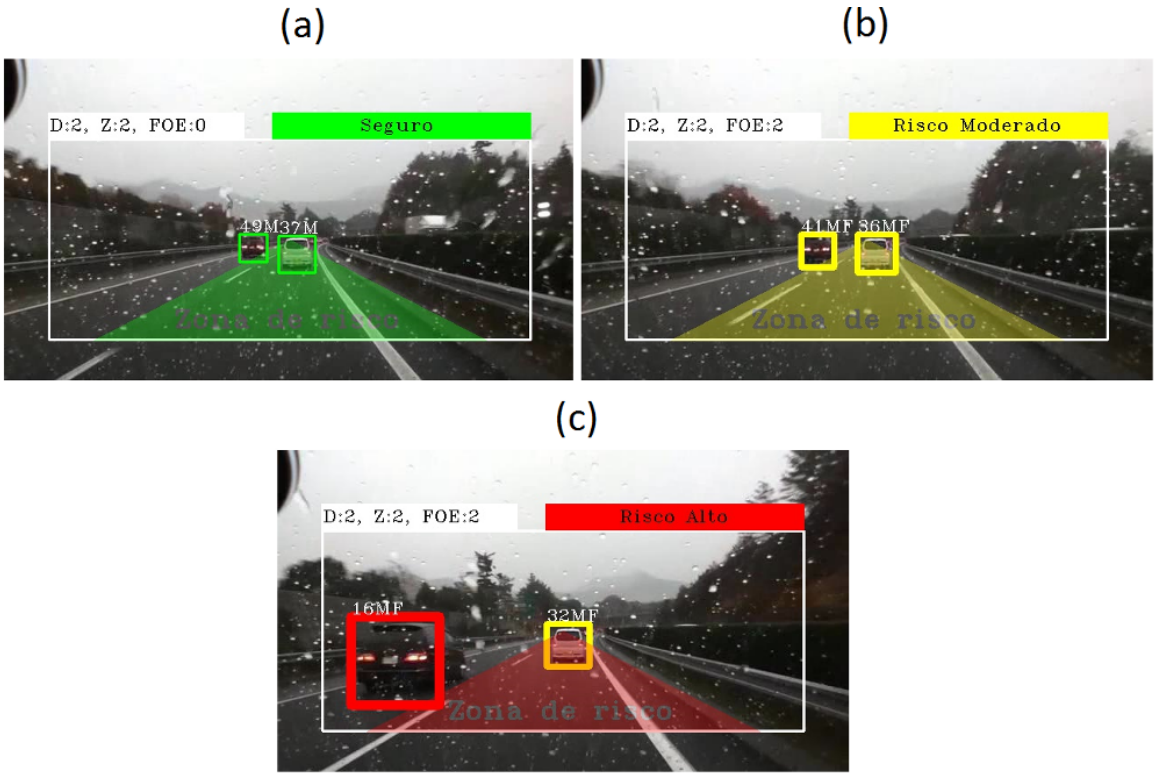
Com base nas informações do relatório da *National Traffic Law Center (2016)*, na Tabela 1, o limiar de $risco\_moderado$ foi definido com o valor de 50 metros e o limiar de $risco\_alto$ foi definido com o valor de 30 metros.

A Figura 68 mostra um exemplo com as três situações possíveis para a emissão de alertas. A Figura 68(a) indica que não há risco de colisão (cor verde). Mesmo existindo dois veículos detectados e dentro da zona de risco (D:2 e Z:2, respectivamente). Ambos estão com distância menor que 50 metros da câmera, porém eles estão sem foco em expansão (FOE:0), ou seja, ou os veículos estão se distanciando da câmera ou estão em velocidade constante em relação ao veículo com a câmera.

Já na Figura 68(b), os veículos continuam dentro da zona de risco, estão a mais de 30 metros e a menos de 50 metros da câmera, e ambos os veículos estão com foco em expansão (FOE:2). Isto indica que os veículos estão ficando cada vez mais próximos da câmera, o que começa a se tornar um risco de colisão, neste caso a cor muda para amarelo e é emitido alerta sonoro.

Por fim, na Figura 68(c) é emitido alerta de Risco Alto. As cores mudam para vermelho e também é emitido um alerta sonoro, porém diferente do alerta sonoro quando for Risco Moderado. Percebe-se que ambos os veículos estão em foco em expansão (FOE:2), porém o da esquerda está á uma distância menor do que 30 metros

Figura 68 – Alertas de risco de colisão. (a) Seguro. (b) Risco Moderado. (c) Risco Alto



Fonte: Elaborado pelo autor, 2017.

da câmera, o que torna o alerta de Risco Alto e seu retângulo fica na cor vermelha. O veículo mais à frente permanece na cor amarela, indicando que ele oferece Risco Moderado de colisão.

Executando o sistema ativando a zona de risco, a estimativa de distância, a verificação de FOE e com os alertas de colisão sonoros e visuais, são executados alguns processamentos extras, se comparado a executar o sistema apenas com detecção e rastreamento. Abaixo são listados os processamentos extras realizados:

a) calcular e desenhar a zona de risco;

b) calcular os veículos que intersectam a zona de risco;

c) calcular o foco de expansão dos veículos;

d) calcular a distância entre os veículos;

e) determinar o nível do risco;

f) desenhar os rótulos informativos no topo do retângulo branco grande;

g) desenhar a distância e a borda diferenciada em torno dos veículos baseado no risco que oferecem;

h) carregar e executar o arquivo de som com o alerta sonoro.

Algumas destas etapas, como carregar e executar o arquivo de som, ocorrem apenas quando for Risco Moderado ou Risco Alto. Estes processamentos extras não impactam na precisão da detecção, porém impactam na velocidade de processamento, deixando o sistema mais lento, conforme pode ser percebido se for comparado a quantidade de quadros por segundo (fps) da Tabela 6 com a Tabela 7.

Tabela 7 – Comparativo de tempo de processamento sem e com emissão de alertas

Técnica	FPS sem alertas (Tabela 6)	**FPS com alertas (sonoro)**	FPS com alertas (sonoro+visual)
Haar+AdaBoost+Rastreadores+VH	70,96	**68,18**	54,63
MB-LBP+AdaBoost+Rastreadores+VH	39,85	**39,26**	34,21
HOG+SVM+Rastreadores+VH	16,85	**16,1**	15,63

Fonte: Elaborado pelo autor, 2017.

Haar+AdaBoost teve a maior redução no tempo de processamento (mais de 20%) com alertas sonoros e visuais. As outras duas técnicas também tiveram redução, mas menor. Porém com a utilização apenas dos alertas sonoros (coluna do meio destacada) a redução na velocidade de processamento foi pequena em todas as três técnicas, o que é ótimo, pois este tipo de alerta é o mais interessante.

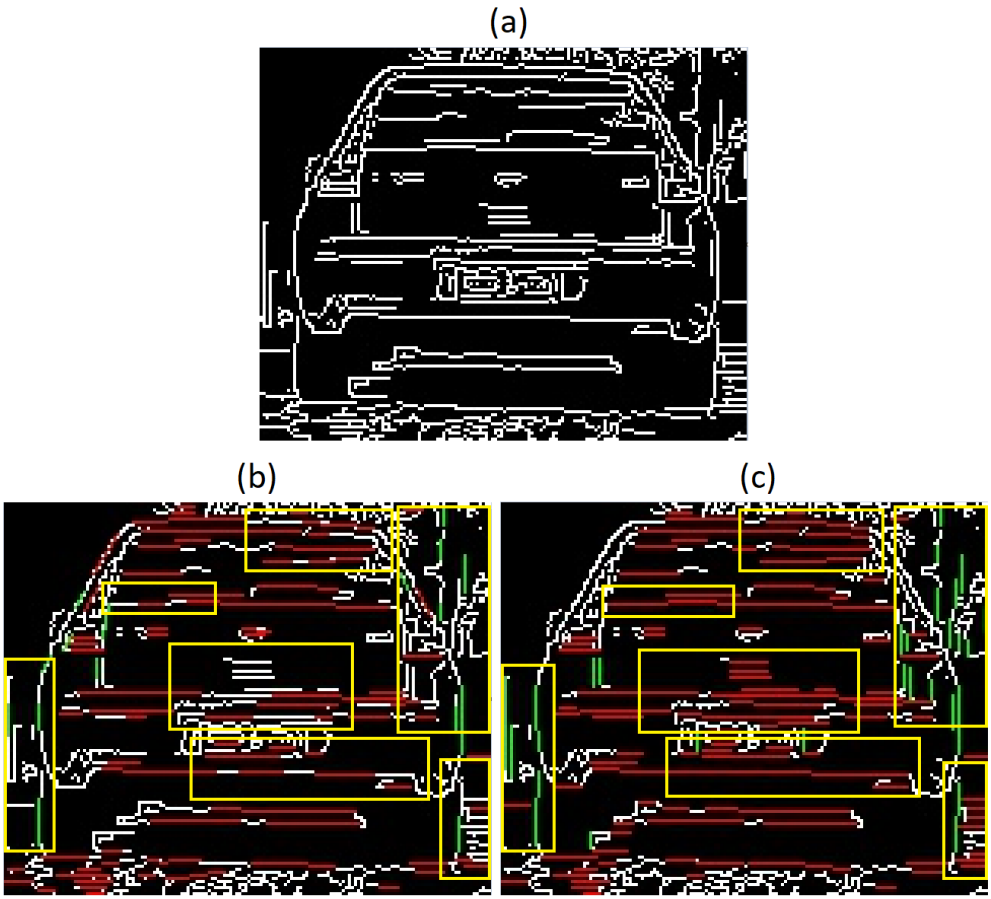
Executando o sistema, a detecção dos veículos e os alertas foram emitidos corretamente conforme os critérios definidos na Seção 4.3.5, tanto os alertas sonoros como os alertas visuais. A emissão destes alertas, principalmente dos sonoros, despertam a atenção do motorista na estrada, permitindo que ele analise a situação e tome ações de forma antecipada a fim de evitar um acidente em situações de risco de colisão com outros veículos a sua frente.

5.5 DETECÇÃO DE LINHAS COM CCLVH

Foram efetuados algumas comparações entre o algoritmo detector de linhas verticais e horizontais CCLVH (vide Seção 4.1.2, Algoritmo 52) e o algoritmo Hough (vide Seção 2.1.2). Foram utilizadas parametrizações semelhantes para as duas técnicas nos testes realizados. Sendo $\rho = 1$, $\theta = \pi/180$, $limiarAcumuladorMinimo = 25$, $tamanhoMinimoLinha = 5$ pixels e $gapMaximoLinha = 0$ pixels. A Figura 69 mostra o resultado da execução.

A Figura 69(a) é a imagem binária de entrada para ambos os algoritmos para efetuar a detecção das linhas. A imagem binária é produzida pelo detector de bordas de Canny. A Figura 69(b) mostra o resultado utilizando o algoritmo de Hough e a Figura 69(c) mostra o resultado utilizando o algoritmo CCLVH. As linhas de cor vermelha e as

Figura 69 – (a) Imagem binária de entrada (Canny). (b) Linhas detectadas com Hough. (c) Linhas detectadas no CCLVH



Fonte: Elaborado pelo autor, 2017.

linhas de cor verde indicam linhas horizontais e verticais detectadas, respectivamente. Os retângulos amarelos nas figuras (b) e (c) mostram algumas regiões com diferenças de detecção entre as duas técnicas. Os segmentos que permaneceram na cor branca são os que não foram identificados.

A Tabela 8 mostra algumas informações quantitativas, sendo elas o tempo de processamento, a quantidade de linhas verticais, e a quantidade de linhas horizontais detectadas por cada técnica. Pelos dados da Tabela 8, é possível perceber que a

Tabela 8 – Resultados da comparação da detecção de linhas utilizando Hough e CCLVH

Técnica	Quantidade de Linhas Verticais	Quantidade de Linhas Horizontais	Tempo de processamento
Hough	16	90	28 ms
CCLVH	**29**	**108**	**28 ms**

Fonte: Elaborado pelo autor, 2017.

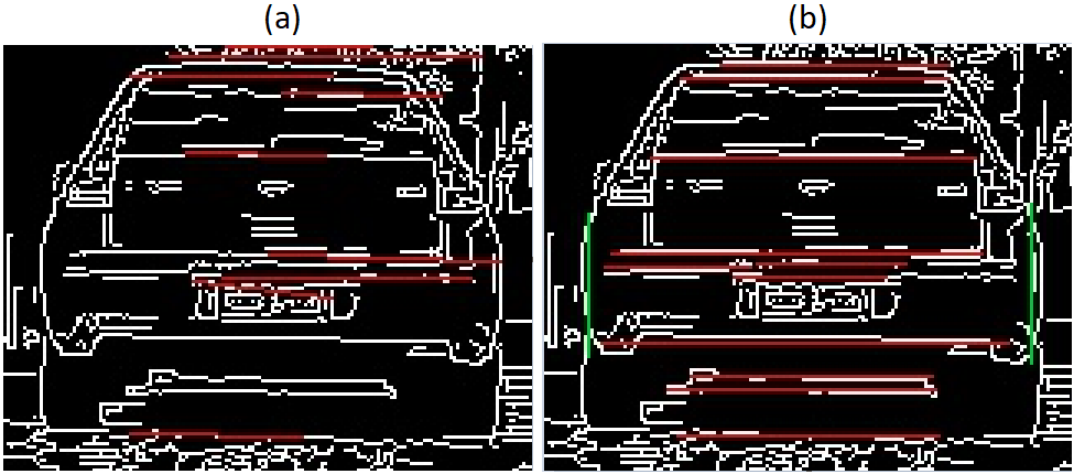
técnica CCLVH (linha destacada) detectou mais linhas horizontais e quase o dobro

de linhas verticais do que Hough, e com o mesmo tempo de processamento. Isto é desejado, pois quanto mais linhas forem detectadas, mais dados sobre as linhas verticais e horizontais podem ser utilizados para calcular o valor de confiança de que uma região candidata é um veículo.

Um segundo experimento foi feito explorando um dos principais diferenciais da técnica CCLVH, que é a junção de segmentos. Onde o objetivo é revelar linhas maiores para identificar melhor as principais linhas verticais e horizontais do veículo, mesmo que os segmentos possuam *gaps* (lacunas) entre eles.

As Figuras 70(a) e (b) mostram os resultados das técnicas Hough e CCLVH, respectivamente. Os seguintes parâmetros foram alterados em ambas as técnicas: $tamanhoMinimoLinha = 40$ pixels e $gapMaximoLinha = 5$ pixels. É possível perceber

Figura 70 – (a) Linhas detectadas com Hough. (b) Linhas detectadas no CCLVH



Fonte: Elaborado pelo autor, 2017.

que a técnica CCLVH conseguiu fazer a identificação das maiores linhas verticais e horizontais do veículo. Enquanto que a técnica de Hough não detectou várias linhas importantes, como as verticais e nem várias linhas do para-choques.

Com as linhas identificadas na Figura 70(b) pela técnica CCLVH, é muito simples determinar a caixa delimitadora do veículo, basta prolongar as linhas verticais e horizontais das extremidades até se intersectarem, conforme mostrado na Figura 71, onde o prolongamento das linhas foi feito na cor amarela para destacar. Também é possível perceber que há simetria entre as linhas verticais e entre as linhas horizontais que foram prolongadas para formar a caixa delimitadora em torno do veículo.

Figura 71 – Identificação da caixa delimitadora do veículo utilizando a técnica CCLVH

Fonte: Elaborado pelo autor, 2017.

5.6 CONSIDERAÇÕES FINAIS DO CAPÍTULO

Com base nos resultados expostos nas Tabelas 5 e 6, a combinação das técnicas Haar+AdaBoost+Rastreadores+VH mostrou-se superior, se comparada com as outras duas técnicas, tanto em tempo ensolarado como em tempo chuvoso.

Os resultados apresentados utilizando a combinação das técnicas MB-LBP+ AdaBoost+Rastreadores+VH ficaram próximos dos resultados das técnicas Haar+ AdaBoost+Rastreadores+VH. Com o diferencial que é possível melhorar o desempenho da primeira combinação de técnicas, adicionando-se milhares de imagens positivas e negativas ao conjunto de treinamento, sem comprometer muito o tempo de treinamento, pois o treinamento é mais rápido com MB-LBP do que com Haar-like com o classificador AdaBoost.

O rastreador KCF mostrou-se preciso e rápido para execução. Não foi percebida nenhuma detecção incorreta no processo de rastreamento. Inclusive tolerando certo nível de oclusões. Porém, quanto maior for a ROI do fragmento para rastreamento da ROI alvo (veículo), maior é o tempo de processamento, o que era esperado, conforme exposto por Henriques et al. (2015) na Seção 2.7. Também verificou-se que quanto maior for o limiar L_{VH} (intervalo de rastreamento e VH), menor é a precisão do casamento da região rastreada com a região do veículo sendo rastreado.

Os rastreadores auxiliares, apesar de acionados apenas em caso de falhas na detecção, cumprem um papel complementar importante. Através deles é possível restaurar veículos que vinham sendo monitorados quadro a quadro e desaparecem subitamente no quadro atual, por algum problema do detector. Isto faz dos rastreadores auxiliares um mecanismo de tolerância à falhas que contribui para a melhora na precisão da detecção dos veículos.

Os rastreadores auxiliares, apesar de acionados apenas em caso de falhas na detecção, cumprem um papel complementar importante, pois veículos que estavam visíveis no(s) quadro(s) anterior(es), dificilmente somem subitamente no quadro atual. Assim, é possível restaurar veículos não detectados pelo classificador, mas que vinham sendo monitorados pelos rastreadores auxiliares, e submete-los ao rastreamento, o que o torna um esquema interessante para tolerância à falhas na detecção.

Um efeito colateral da detecção intervalada com rastreamento, é que este esquema pode propagar os falso positivos (e falso negativos) de regiões detectadas (e não detectadas) incorretamente por vários quadros rastreados até que se faça uma VH ou uma nova detecção. Por isto é importante que o classificador tenha boa precisão.

Na maioria dos casos, a emissão de alertas sonoros parece ser mais indicada do que de alertas visuais. Isto devido aos alertas sonoros desviarem menos a atenção do motorista da estrada e atingir todos os ocupantes do veículo. No entanto o alerta visual pode ser útil para portadores de dificuldades auditivas e para copilotos.

A principal contribuição deste trabalho de pesquisa é a proposta de um esquema para detecção de veículos intervalada com rastreamento utilizando um rastreador rápido e preciso chamado KCF, proposto por Henriques et al. (2015). Tanto as regiões detectadas como as rastreadas podem ser submetidas a uma etapa de VH com o novo algoritmo CCLVH, que serve para eliminar falso positivos analisando a simetria dos centroides das linhas verticais e horizontais das bordas dos veículos. Regiões rastreadas que são reprovadas pela VH, são re-detectadas pontualmente, evitando que todo o quadro da imagem seja analisado pelo detector, economizando assim recursos computacionais. Um mecanismo de tolerância a falhas é acionado utilizando rastreadores auxiliares que monitoram as informações temporais das ROIs rastreadas quadro a quadro, quando uma região rastreada desaparece subitamente durante uma nova detecção, ela consegue ser restaurada e é mantido seu rastreamento, melhorando assim a precisão da detecção dos veículos (vide fluxograma da Figura 59).

A Tabela 9 mostra um comparativo de funcionalidades do trabalho proposto (linha destacada na tabela) com os trabalhos relacionados. Através da tabela é possível perceber que vários trabalhos contemplam apenas partes das funcionalidades mínimas esperadas para um sistema que visa auxiliar o motorista a prevenir colisões com outros veículos. Tolerância à falhas e Re-detecção de veículos apenas em regiões específicas, são funcionalidades implementadas apenas pelo trabalho proposto, o que o diferencia dos trabalhos comparados.

O gráfico da Figura 72 mostra os resultados de precisão dos trabalhos relacionados (somente os que divulgam seus resultados de precisão) e do trabalho proposto.

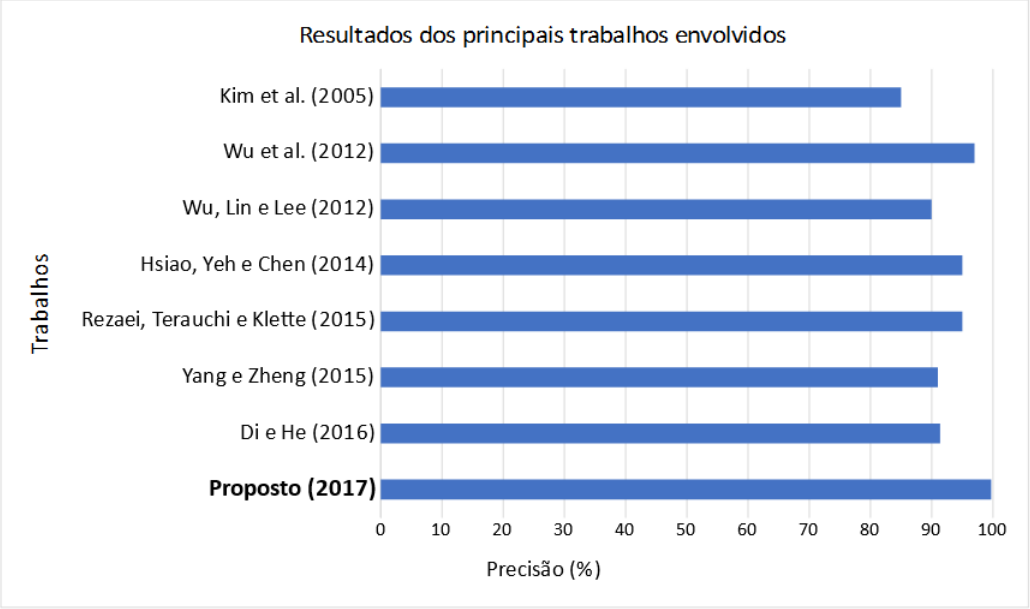
Tabela 9 – Comparativo das funcionalidades dos trabalhos relacionados com o trabalho proposto

Trabalho	GH	VH	Rastrea-mento	Tolerância à Falhas	Re-detecção de Regiões	Estimativa Distância	Alerta Visual	Alerta Sonoro
Kim et al. ()	✓	✓	✓			✓		
Tang et al. (2015)	✓	✓	✓			✓		✓
Di e He (2016)	✓	✓	✓			✓		
Hsiao, Yeh e Chen (2014)	✓	✓	✓					
Wu, Lin e Lee (2012)	✓	✓				✓	✓	
Wen et al. (2015)	✓	✓						
Wu et al. (2012)	✓	✓				✓		✓
Yang e Zheng (2015)	✓	✓	✓			✓	✓	
Mammeri, Zhou e Boukerche (2016)	✓	✓						
Rezaei, Terauchi e Klette (2015)	✓	✓				✓		
Proposto	✓	✓	✓	✓	✓	✓	✓	✓

Fonte: Elaborado pelo autor, 2017.

Onde, no trabalho proposto, a precisão apresentada é com base nos resultados dos testes efetuados em ambiente com tempo bom (Seção 5.2). É possível perceber que

Figura 72 – Desempenho dos trabalhos relacionados e do trabalho proposto



Fonte: Elaborado pelo autor, 2017.

a precisão do trabalho proposto é semelhante aos demais trabalhos, inclusive com superação.

6 CONCLUSÃO

Este trabalho apresentou um sistema que detecta e rastreia veículos para emissão de alertas de risco de colisão. A emissão dos alertas é feita somente quando realmente existir algum risco de colisão com os veículos à frente e os alertas são disparados. Assim o motorista do veículo percebe rapidamente a situação de risco e toma, de forma antecipada, ações que evitem acidentes como uma colisão com o veículo à sua frente.

Para melhorar a eficiência e emitir alertas rápidos, são apresentadas algumas metodologias, onde algumas delas são propostas inéditas deste trabalho. Uma das novas metodologias propostas é a técnica Confiança dos Centroides das Linhas Verticais e Horizontais (CCLVH), que auxilia no estágio de verificação de hipótese (VH) a eliminar falso positivos.

Outra metodologia apresentada neste trabalho, é a estimativa da distância entre os veículos, utilizando a proporção da largura do veículo e da sua distância da câmera no mundo real com estes valores em pixels no plano de imagem. Conhecendo a distância aproximada entre os veículos possibilita emitir alertas mais eficientes.

Uma das contribuições mais importantes deste trabalho, é o emprego de um modelo de detecção dos veículos com VH e rastreamento de veículos intercalados com a detecção, e um mecanismo de tolerância a falhas com a restauração dos veículos que desapareceram subitamente. O modelo permite o emprego de qualquer algoritmo para detecção, rastreamento e VH. Quanto mais rápido e eficiente for cada um destes algoritmos, mais rápido e preciso será o sistema.

Neste trabalho, para a detecção dos veículos foram testadas três conjuntos de técnicas: Haar-like com AdaBoost, MB-LBP com AdaBoost e HOG com SVM. Já a técnica utilizada para o rastreamento dos veículos foi o KCF, que é muito rápido, podendo processar a mais de 100 quadros por segundo (HENRIQUES et al., 2015).

Outra contribuição importante apresentada neste trabalho é uma proposta para a emissão de alertas de risco alto e moderado, tanto sonoros como visuais que é feita utilizando uma zona de risco que monitora veículos aumentando a proximidade da câmera. Isto é feito utilizando o conceito de foco de expansão (FOE) e da estimativa da distância dos veículos em relação a câmera.

Com base nos resultados do Capítulo 5, implementando os passos do fluxograma da Figura 59, é possível concluir que para a detecção e rastreamento de veículos com verificação de hipótese, as técnicas Haar+AdaBoost e MB-LBP+AdaBoost

apresentaram desempenhos promissores para uso na prática, tanto em precisão como no tempo de processamento. A precisão média destas duas técnicas são equiparadas, atingindo mais de 97%. Já a taxa média de processamento de quadros por segundo (fps) foi maior para Haar+AdaBoost, com 56 fps, enquanto MB-LBP processou a 33 fps.

A técnica de rastreamento KCF mostrou-se bastante tolerante às mudanças de iluminação e translação, obtendo precisão e processando em torno de 100 fps. Não foram percebidos rastreamentos incorretos nas regiões submetidas, inclusive conseguindo rastrear regiões com certo nível de oclusão. Porém, percebeu-se que quanto mais e maiores forem as ROIs para rastrear, maior é o tempo de processamento. Também foi percebido que a precisão do rastreamento diminui, conforme aumenta o intervalo de rastreamento e da VH.

Efetuar a detecção de veículos em toda a ROI principal da imagem somente em intervalos de quadros, realizar o rastreamento dos veículos detectados nos intervalos, e aplicar validação de hipótese intervalada, com re-detecção apenas dos veículos com baixo nível de confiança de serem realmente veículos, mostrou manter (e até melhorar em vários casos) a precisão da detecção e principalmente reduzir o tempo de processamento nas três técnicas de detecção utilizadas.

Utilizar rastreadores auxiliares para armazenar o estado dos veículos detectados e rastreados, com objetivo de restaurá-los caso desapareçam subitamente (não são mais detectados pelo classificador), foi efetivo como mecanismo de tolerância à falhas.

Utilizar VH descarta a grande maioria dos falso positivos, diminuindo assim a emissão de alertas falsos. Utilizar a técnica CCLVH proposta neste trabalho para VH, mostrou-se promissora com boa precisão na detecção das linhas verticais e horizontais, no cálculo de seus centros e centroides, possibilitando boa precisão no cálculo de um limiar de confiança para a hipótese da região ser ou não um veículo.

Utilizar o conceito de foco de expansão (FOE) dos veículos, permite identificar veículos que estejam crescendo na direção do veículo portando a câmera. Isto significa que o veículo portando a câmera e o veículo à sua frente estão diminuindo gradativamente a distância entre eles. Se este crescimento não parar, uma colisão entre os veículos irá ocorrer dentro de um determinado espaço de tempo. Este trabalho não calcula o tempo para colisão.

Construir uma zona de risco na frente do veículo portando o sistema, a fim de monitorar os veículos que estão em FOE, é uma estratégia que funciona bem para emitir alertas e prevenir acidentes de colisão. A região desta zona de risco não pode ser pequena demais, senão pode ocorrer aproximações súbitas, e nem demasiadamente

extensa, de maneira que permita disparar alertas excessivamente sem necessidade. A zona estimada neste trabalho atende a estes critérios.

Estimar a distância entre os veículos utilizando como base a proporção da sua largura e a sua distância real, com a sua largura e distância em pixels, é bastante dependente da precisão das ROIs detectadas pelo classificador. Porém, com isto solucionado, a estimativa possui uma boa precisão e é bem mais simples de calcular e calibrar do que usando vários parâmetros da câmera, que a qualquer mudança de posição ou trepidação precisa ser recalibrada.

Emitir alertas sonoros é mais efetivo que o visual, por desviarem menos a atenção do motorista da estrada, serem perceptíveis para todos os passageiros do veículo, com exceção daqueles que possuírem alguma dificuldade auditiva, e necessitarem menos custo computacional no processamento. Já os alertas visuais, exigem que o motorista desvie mais sua atenção da estrada, o que pode aumentar o risco de acidentes, e também aumenta o tempo necessário para processamento devido a necessidade de desenho gráfico.

Assim, é possível concluir que o sistema proposto detecta e rastreia veículos à frente do veículo portando o sistema, monitora os veículos que intersectam uma zona de risco definida à frente e identifica se há risco de colisão com estes veículos, analisando as distâncias e aumento de proximidade em relação à câmera. Tendo o sistema identificado risco de colisão, são emitidos alertas sonoros e visuais ao motorista, com o intuito de despertar sua atenção na estrada e este ter tempo hábil para tomar uma ação defensiva que o auxilie a evitar acidentes por colisão traseira com veículos à sua frente.

6.1 CONTRIBUIÇÕES/RESULTADOS

Com a realização deste trabalho pôde-se alcançar algumas contribuições:

- Um mapeamento sistemático com a análise de mais de 100 trabalhos relevantes sobre o tema da pesquisa deste trabalho, servindo como ponto de partida para outros pesquisadores;
- Desenvolvimento de um fluxo completo para aquisição de imagem, detecção e rastreamento de veículos e emissão de alertas eficazes em situações de risco de colisão entre o veículo portando o sistema com os veículos à sua frente;
- Criação de um novo algoritmo chamado CCLVH, que detecta com precisão as linhas verticais e horizontais de um veículo e marca quais destas linhas estão nas bordas da região. Calcula também os centros das linhas e os centroides horizontais e verticais, permitindo facilmente calcular a simetria entre as linhas

e um limiar de confiança para determinar se a região candidata é um veículo ou não;

- Apresentação de uma forma simples para estimar a distância entre veículos utilizando apenas uma câmera e as medidas do mundo real e a proporção em pixels;
- Comparação entre três técnicas diferentes de extração de características e classificação de veículos com e sem validação de hipótese;
- Aplicação de estratégias inovadoras de rastreamento e tolerância a falhas no rastreamento e detecção com o algoritmo KCF e rastreadores auxiliares para guardarem o contexto dos objetos detectados e rastreados. Isto permite melhorar a precisão na detecção e aumentar muito a velocidade de processamento;
- Constatação da eficácia do emprego de uma zona de risco de colisão e percepção da aproximação do veículo com o sistema dos demais veículos, permitindo assim emitir alertas mais eficazes;
- Desenvolvimento de uma ferramenta visual para facilitar e agilizar a marcação de imagens para treinamento tanto para AdaBoost como para SVM.

6.2 CONSIDERAÇÕES FINAIS

Segundo a Organização Mundial da Saúde (OMS), cerca de 3.400 pessoas morem por dia no mundo por acidentes nas estradas. Segundo a seguradora DPVAT, no Brasil entre os anos de 2015 e 2016 foram pagos seguros para quase 1 milhão de pessoas por invalidez permanente devido a acidentes com veículos. Isto são números impressionantes e alarmantes. As pessoas depara-se com acidentes de trânsito fatais quase que diariamente, seja presencialmente ou pelos noticiários. Isto já se tornou rotineiro, dando uma falsa impressão de que é um fato comum e que só ocorre com os outros.

Este trabalho de mestrado mostra que é possível desenvolver um sistema de visão computacional para emitir alertas de risco de colisão com outros veículos. O sistema apresentado é capaz de emitir alertas de risco de colisão ao condutor do veículo, de forma que ele fique mais atento à estrada e tenha tempo hábil de evitar um acidente.

Como os governos gastam bilhões em indenizações, aposentadorias por invalides, entre outros gastos relacionados, projetos deste tipo, são uma ótima oportunidade de investimento com ganhos incalculáveis, tanto financeiros como para as pessoas, as quais estariam mais seguras nas estradas.

6.3 TRABALHOS FUTUROS

Partindo do projeto desenvolvido e apresentado neste trabalho, existem ainda várias oportunidades de melhoramentos e extensões que trabalhos futuros podem fazer:

6.3.1 Melhoramentos no Sistema

- O treinamento dos classificadores foi feito com quase 10 mil imagens de amostras entre positivas e negativas. Para rodar o sistema nas estradas efetivamente, os classificadores deveriam ser treinados utilizando uma base com mais amostras positivas e negativas. É difícil estabelecer um número exato de amostras, mas cerca de 30 mil imagens positivas e 40 mil imagens negativas é uma sugestão para começar. No conjunto de imagens positivas deveriam ser adicionados exemplos com várias poses diferentes, como veículos de frente, de lado, tamanhos e modelos distintos. O treinamento das quase 10 mil imagens com Haar+AdaBoost levou 25 dias para treinar, porém com MB-LBP+AdaBoost levou menos de duas horas, e também com bons resultados;

- Ajustar a profundidade da perspectiva da zona de risco, tornando mais realista e precisa. Isto pode ser feito detectando as faixas da estrada e aplicando a perspectiva com base nelas. Um trabalho promissor para detecção de faixas é o de Wu, Lin e Lee (2012);

- Aumentar a robustez da técnica de estimativa da distância entre os veículos. No método apresentado, é necessário solucionar as questões com veículos de lado e veículos de grande porte, como caminhões e ônibus, pois vão parecer mais perto. Podem ser analisadas e testadas mais técnicas como a de Liu, Fang e Chen (2017), que faz uma relação entre o número da linha dos pixeis da imagem e a distância real até o veículo;

- Adaptar o sistema para rodar em dispositivos móveis. Assim, o motorista poderia acoplar seu dispositivo (por exemplo, um celular) no para-brisas do seu veículo, e ele emitiria os alertas de risco de colisão. Um trabalho relacionado ao assunto é o de Tang et al. (2015);

- Utilizar outras técnicas recentes e com processamento rápido para o rastreamento dos veículos (acima de 100 quadros por segundo), como por exemplo Hare et al. (2016) e Held, Thrun e Savarese (2016), a fim de melhorar ainda mais o desempenho do sistema;

- Utilizar informações intrínsecas do veículo, como a velocidade. Podendo assim, obter e utilizar mais variáveis na análise do risco de colisão entre os veículos.

Caso rode em um dispositivo com GPS é possível estimar a velocidade, outra alternativa é obter a velocidade e até outras informações do veículo via OBD II[1], que é a sigla para a expressão em inglês *On Board Diagnostics*, que significa "diagnóstico de bordo". Alguns trabalhos iniciais sobre o assunto são Fernandes et al. (2015) e Chen et al. (2015).

6.3.2 Extensões do Trabalho

- Veículos muito próximos da câmera são difíceis de serem detectados com classificadores convencionais como AdaBoost e SVM. Para isto pode ser utilizada a técnica CCLVH para detectar a caixa delimitadora de veículos próximos baseados nas linhas verticais e horizontais que formam seu contorno. Este algoritmo pode ser acionado, quando o classificador utilizado não encontrar nenhum veículo à sua frente, pois isto pode significar a existência de um veículo muito próximo da câmera. Para este fim, poderia ser utilizado ainda a detecção das luzes traseiras e a placa do veículo, tomando como base os trabalhos de Rezaei, Terauchi e Klette (2015) e Gu et al. (2015), respectivamente;

- Detectar os veículos se aproximando atrás e nas zonas cegas do veículo portando o sistema, para isto deve ser utilizado mais de uma câmera e ajustar o sistema para interpretar os quadros das outras câmeras também, ou ter mais instâncias do sistema rodando, uma para cada câmera. O trabalho de Dooley et al. (2016) possui uma abordagem interessante utilizando câmera olho de peixe;

- Detectar motocicletas, principalmente se aproximando por trás do veículo. Algumas referências iniciais podem ser os trabalhos de Fernández et al. (2013) e Thai et al. (2014);

- Detectar pedestres à frente do veículo. É comum acontecer muitos acidentes com pedestres, mesmo com a travessia na faixa de pedestres. Como trabalhos iniciais, podem ser apontados os de Zangenehpour, Miranda-Moreno e Saunier (2015), Wang et al. (2014) e Chavez-Garcia e Aycard (2016);

- Detectar veículos também à noite. Um solução indicada para isto é analisar a simetria de regiões com alta luminosidade, ou seja alta intensidade nos níveis dos pixeis (são as luzes dos veículos). Alguns trabalhos encontrados e relacionados a este assunto foram Zhou et al. (2013), Hajimolahoseini, Soltanian-Zadeh e Amirfattahi (2014), Rezaei, Terauchi e Klette (2015) e Choi et al. (2014).

[1] https://pt.wikipedia.org/wiki/OBD

REFERÊNCIAS

ARROSPIDE, J.; SALGADO, L.; NIETO, M. Video analysis-based vehicle detection and tracking using an MCMC sampling framework. **EURASIP Journal on Advances in Signal Processing**, v. 2012, n. 1, p. 1–20, 2012.

BAILEY, J. et al. Evidence relating to object-oriented software design: A survey. In: **First International Symposium on Empirical Software Engineering and Measurement**. [S.l.]: IEEE, 2007. p. 482–484.

BRADSKI, G. R.; KAEHLER, A. **Learning OpenCV: [computer vision with the OpenCV library]**. 1. ed. [S.l.]: O'Reilly, 2011. (Software that sees). ISBN 978-0-596-51613-0.

BUCHINGER, D.; CAVALCANTI, G. A. d. S.; HOUNSELL, M. D. S. Mecanismos de busca acadêmica: uma análise quantitativa. **Revista Brasileira de Computação Aplicada**, v. 6, n. 1, 2014.

BURGES, C. J. A tutorial on support vector machines for pattern recognition. **Data Mining and Knowledge Discovery**, v. 2, n. 2, p. 121–167, 1998.

CANNY, J. A computational approach to edge detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 8, n. 6, p. 679–698, 1986.

CHAVEZ-GARCIA, R. O.; AYCARD, O. Multiple sensor fusion and classification for moving object detection and tracking. **IEEE Transactions on Intelligent Transportation Systems**, v. 17, n. 2, p. 525–534, 2016.

CHEN, L.-B. et al. An intelligent vehicular telematics platform for vehicle driving safety supporting system. In: **International Conference on Connected Vehicles and Expo (ICCVE)**. [S.l.]: IEEE, 2015. p. 210–211.

CHOI, K.-H. et al. State machine and downhill simplex approach for vision-based nighttime vehicle detection. **ETRI Journal**, v. 36, n. 3, p. 439–449, 2014.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, v. 20, n. 3, p. 273–297, 1995.

CROW, F. C. Summed-area tables for texture mapping. **SIGGRAPH Comput. Graph.**, ACM, New York, NY, USA, v. 18, n. 3, p. 207–212, 1984.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: **2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)**. [S.l.: s.n.], 2005. v. 1, p. 886–893.

DI, Z.; HE, D. Forward collision warning system based on vehicle detection and tracking. In: **Optoelectronics and Image Processing (ICOIP), International Conference on**. [S.l.: s.n.], 2016. p. 10–14.

DOOLEY, D. et al. A blind-zone detection method using a rear-mounted fisheye camera with combination of vehicle detection methods. **IEEE Transactions on Intelligent Transportation Systems**, v. 17, n. 1, p. 264–278, 2016.

DPVAT. **Seguradora Líder-DPVAT**: Boletim estatístico. 2017. Disponível em: <https://www.seguradoralider.com.br/Pages/Boletim-Estatistico.aspx>. Acesso em: 21 jan. 2017.

DUDA, R. O.; HART, P. E.; STORK, D. G. **Pattern Classification (2Nd Edition)**. [S.l.]: Wiley-Interscience, 2000. ISBN 0471056693.

FERNANDES, B. et al. Mobile application for automatic accident detection and multimodal alert. In: **Vehicular Technology Conference (VTC Spring), 2015 IEEE 81st**. [S.l.]: IEEE, 2015. p. 1–5.

FERNáNDEZ, C. et al. Real-time vision-based blind spot warning system: Experiments with motorcycles in daytime/nighttime conditions. **International Journal of Automotive Technology**, v. 14, n. 1, p. 113–122, 2013.

FREUND, Y.; SCHAPIRE, R. E. A desicion-theoretic generalization of on-line learning and an application to boosting. In: **European conference on computational learning theory**. [S.l.]: Springer, 1995. p. 23–37.

GONZALES, R. C.; WOODS, R. E. **Processamento digital de imagens**. 3. ed. [S.l.]: Pearson Education, 2011. ISBN 978-85-7605-401-6.

GU, Q. et al. Vision-based multi-scaled vehicle detection and distance relevant mix tracking for driver assistance system. **Optical Review**, v. 22, n. 2, p. 197–209, 2015.

GUO, C. et al. A multimodal ADAS system for unmarked urban scenarios based on road context understanding. **IEEE Transactions on Intelligent Transportation Systems**, v. 16, n. 4, p. 1690–1704, 2015. ISSN 1524-9050, 1558-0016. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6971167>.

HADI, R. A.; SULONG, G.; GEORGE, L. E. Vehicle detection and tracking techniques : A concise review. **Signal & Image Processing : An International Journal**, v. 5, n. 1, p. 1–12, 2014.

HAJIMOLAHOSEINI, H.; SOLTANIAN-ZADEH, H.; AMIRFATTAHI, R. Robust vehicle tracking algorithm for nighttime videos captured by fixed cameras in highly reflective environments. **IET Computer Vision**, v. 8, n. 6, p. 535–544, 2014.

HARE, S. et al. Struck: Structured output tracking with kernels. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 38, n. 10, p. 2096–2109, 2016.

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. In: **Alvey vision conference**. [S.l.]: Citeseer, 1988. v. 15, p. 10–5244.

HARVEY, A. **CV Dazzle**: Camouflage from face detection. 2017. Disponível em: <https://cvdazzle.com/>. Acesso em: 16 jan. 2017.

HELD, D.; THRUN, S.; SAVARESE, S. Learning to track at 100 fps with deep regression networks. In: ____. **14th European Conference, Amsterdam, The Netherlands Proceedings, Part I**. [S.l.: s.n.], 2016. p. 749–765.

HENRIQUES, J. F. et al. High-speed tracking with kernelized correlation filters. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 37, n. 3, p. 583–596, 2015.

HSIAO, S.-F.; YEH, G.-F.; CHEN, J.-C. Design and implementation of multiple-vehicle detection and tracking systems with machine learning. In: **17th Euromicro Conference on Digital System Design**. [S.l.: s.n.], 2014. p. 551–558.

KALAL, Z.; MIKOLAJCZYK, K.; MATAS, J. Tracking-learning-detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 34, n. 7, p. 1409–1422, 2012.

KHAIRDOOST, N.; S, A. M.; JAMSHIDI, K. Front and rear vehicle detection using hypothesis generation and verification. **Signal & Image Processing : An International Journal**, v. 4, n. 4, p. 31–50, 2013.

KIM, S. et al. Front and rear vehicle detection and tracking in the day and night times using vision and sonar sensor fusion. In: **Intelligent Robots and Systems, 2005.(IROS 2005). IEEE/RSJ International Conference on**. [S.l.: s.n.]. p. 2173–2178.

KUO, Y.-C.; CHEN, H.-W. Vision-based vehicle detection in the nighttime. In: **International Symposium on Computer Communication Control and Automation (3CA)**. [S.l.: s.n.], 2010. v. 2, p. 361–364.

LI, W.-h. et al. Co-training algorithm based on on-line boosting for vehicle tracking. In: **Information and Automation (ICIA), 2013 IEEE International Conference on**. [S.l.]: IEEE, 2013. p. 592–596.

LIAO, S. et al. Learning multi-scale block local binary patterns for face recognition. In: **Advances in Biometrics: International Conference, ICB 2007, Seoul, Korea**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 828–837.

LIENHART, R.; MAYDT, J. An extended set of haar-like features for rapid object detection. In: **Proceedings. International Conference on Image Processing**. [S.l.: s.n.], 2002. v. 1, p. I–900–I–903 vol.1.

LIN, C.; LIU, Y.; LEE, C. An efficient neural fuzzy network based on immune particle swarm optimization for prediction and control applications. **International Journal of Innovative Computing, Information and Control**, v. 4, n. 7, p. 1711–1722, 2008.

LIU, L.-C.; FANG, C.-Y.; CHEN, S.-W. A novel distance estimation method leading a forward collision avoidance assist system for vehicles on highways. **IEEE Transactions on Intelligent Transportation Systems**, v. 18, n. 4, p. 937–949, 2017.

LIU, W. et al. Rear vehicle detection and tracking for lane change assist. In: **Intelligent Vehicles Symposium, 2007 IEEE**. [s.n.], 2007. p. 252–257. Disponível em: <http://ieeexplore.ieee.org/abstract/document/4290123/>.

LORENA, A. C.; CARVALHO, A. C. de. Uma introdução às support vector machines. **Revista de Informática Teórica e Aplicada**, v. 14, n. 2, p. 43–67, 2007.

LUCAS, B. D.; KANADE, T. An iterative image registration technique with an application to stereo vision. In: **Proceedings of Imaging Understanding Workshop**. [S.l.: s.n.], 1981. p. 121–130.

MAMMERI, A.; ZHOU, D.; BOUKERCHE, A. Animal-vehicle collision mitigation system for automated vehicles. **IEEE Transactions on Systems, Man, and Cybernetics: Systems**, v. 46, n. 9, p. 1287–1299, 2016.

MUKHTAR, A. et al. On-road approaching motorcycle detection and tracking techniques: A survey. In: **International Conference on Control System, Computing and Engineering (ICCSCE)**. [S.l.: s.n.], 2013. p. 63–68.

MUKHTAR, A.; XIA, L.; TANG, T. B. Vehicle detection techniques for collision avoidance systems: A review. **IEEE Transactions on Intelligent Transportation Systems**, v. 16, n. 5, p. 2318–2338, 2015.

National Traffic Law Center. PDF, **Investigation and prosecution of distracted driving cases (Report No. DOT HS 812 407)**. 2016. 30 p. Disponível em: <https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/812407-distracteddrivingreport.pdf>. Acesso em: 11 jun. 2017.

PAPAGEORGIOU, C.; POGGIO, T. A trainable system for object detection. **International Journal of Computer Vision**, v. 38, n. 1, p. 15–33, 2000.

PETERSEN, K. et al. Systematic mapping studies in software engineering. In: **Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering**. [S.l.: s.n.], 2008. (EASE'08), p. 68–77.

Quatro Rodas. **Distração ao volante**. 2013. Disponível em: <http://quatrorodas.abril.com.br/noticias/distracao-ao-volante/>. Acesso em: 11 jun. 2017.

REZAEI, M.; TERAUCHI, M.; KLETTE, R. Robust vehicle detection and distance estimation under challenging lighting conditions. **IEEE Transactions on Intelligent Transportation Systems**, v. 16, n. 5, p. 2723–2743, 2015.

ROSENFELD, A. Connectivity in digital pictures. **Journal of the ACM**, v. 17, n. 1, p. 146–160, 1970.

SAINI, M. Survey on vision based on-road vehicle detection. **International Journal of U-& E-Service, Science & Technology**, 2014.

SCHAPIRE, R. E.; FREUND, Y. **Boosting: Foundations and Algorithms**. [S.l.]: The MIT Press, 2012. ISBN 0262017180, 9780262017183.

SHAFER, G. **A Mathematical Theory of Evidence**. [S.l.]: Books on Demand, 1976.

SIVARAMAN, S.; TRIVEDI, M. M. Integrated lane and vehicle detection, localization, and tracking: A synergistic approach. **IEEE Transactions on Intelligent Transportation Systems**, v. 14, n. 2, p. 906–917, 2013.

SIVARAMAN, S.; TRIVEDI, M. M. Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis. **IEEE Transactions on Intelligent Transportation Systems**, v. 14, n. 4, p. 1773–1795, 2013.

SREEVISHAKH, K.; DHANURE, S. A review paper on automotive crash prediction and notiication technologies. In: **International Conference on Computing Communication Control and Automation**. [S.l.: s.n.], 2015. p. 999–1002.

SUZUKI, S. Topological structural analysis of digitized binary images by border following. **Computer vision, graphics, and image processing**, v. 30, n. 1, p. 32–46, 1985.

SUZUKI, S.; ABE, K. Topological structural analysis of digitized binary images by border following. **Computer Vision, Graphics, and Image Processing**, v. 30, n. 1, p. 32 – 46, 1985.

TANG, S. J. W. et al. Real-time lane detection and rear-end collision warning system on a mobile computing platform. In: **IEEE 39th Annual International Computers, Software and Applications Conference**. [S.l.: s.n.], 2015. p. 563–568.

THAI, N. D. et al. Learning bag of visual words for motorbike detection. In: **Control Automation Robotics & Vision (ICARCV), 13th International Conference on**. [S.l.: s.n.], 2014. p. 1045–1050.

THEODORIDIS, S.; KOUTROUMBAS, K. **Pattern recognition**. 4. ed. ed. [S.l.]: Elsevier Acad. Press, 2009. OCLC: 550588366. ISBN 978-1-59749-272-0.

TIAN, B. et al. Hierarchical and networked vehicle surveillance in its: A survey. **IEEE Transactions on Intelligent Transportation Systems**, v. 18, n. 1, p. 25–48, 2017.

UN. **The United Nations and Road Safety**. 2017. Disponível em: <http://www.un.org/en/roadsafety/>. Acesso em: 21 jan. 2017.

VIOLA, P.; JONES, M. J. Robust real-time face detection. **International journal of computer vision**, v. 57, n. 2, p. 137–154, 2004.

WANG, S. et al. A novel approach to design the fast pedestrian detection for video surveillance system. **International Journal of Security and Its Applications**, v. 8, n. 1, p. 93–102, 2014.

WEN, X. et al. Efficient feature selection and classification for vehicle detection. **IEEE Transactions on Circuits and Systems for Video Technology**, v. 25, n. 3, p. 508–517, 2015.

WEN, X. et al. A rear-vehicle detection system for static images based on monocular vision. In: **2006 9th International Conference on Control, Automation, Robotics and Vision**. [S.l.: s.n.], 2006. p. 1–4.

WHO. **Global status report on road safety 2015**. 2017. Disponível em: <http://www.who.int/violence_injury_prevention/road_safety_status/2015/en/>. Acesso em: 21 jan. 2017.

WU, B.-F. et al. A vision-based collision warning system by surrounding vehicles detection. **KSII Transactions on Internet and Information Systems**, v. 6, n. 4, p. 1203–1222, 2012.

WU, C.-F.; LIN, C.-J.; LEE, C.-Y. Applying a functional neurofuzzy network to real-time lane detection and front-vehicle distance measurement. **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, v. 42, n. 4, p. 577–589, 2012.

WU, J. et al. A survey on video-based vehicle behavior analysis algorithms. **Journal of Multimedia**, v. 7, n. 3, 2012.

YANG, M.-T.; ZHENG, J.-Y. On-road collision warning based on multiple FOE segmentation using a dashboard camera. **IEEE Transactions on Vehicular Technology**, v. 64, n. 11, p. 4974–4984, 2015.

YOKOI, S.; TORIWAKI, J.-I.; FUKUMURA, T. An analysis of topological properties of digitized binary pictures using local features. **Computer Graphics and Image Processing**, v. 4, n. 1, p. 63–73, 1975.

ZANGENEHPOUR, S.; MIRANDA-MORENO, L. F.; SAUNIER, N. Automated classification based on video data at intersections with heavy pedestrian and bicycle traffic: Methodology and application. **Transportation Research Part C: Emerging Technologies**, v. 56, p. 161–176, 2015.

ZHOU, S. et al. A night time application for a real-time vehicle detection algorithm based on computer vision. **Research Journal of Applied Sciences, Engineering and Technology**, v. 10, n. 5, p. 3037–3043, 2013.