UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LUCAS FERNANDO DA SILVA CAMBUIM

SISTEMA DE PREVISÃO DE COLISÃO DE PEDESTRE BASEADO EM VISÃO COMPUTACIONAL

Recife
2022

LUCAS FERNANDO DA SILVA CAMBUIM

SISTEMA DE PREVISÃO DE COLISÃO DE PEDESTRE BASEADO EM VISÃO COMPUTACIONAL

Tese de Doutorado apresentada ao Programa de Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de Doutor em Ciência da Computação.

Área de Concentração: Engenharia da Computação
Orientadora: Profa. Dra. Edna Natividade da Silva Barros

Recife
2022

Catalogação na fonte
Bibliotecária Nataly Soares Leite Moro, CRB4-1722

C178s Cambuim, Lucas Fernando da Silva
Sistema de previsão de colisão de pedestre baseado em visão computacional / Lucas Fernando da Silva Cambuim. – 2022.
206 f.: il., fig., tab.

Orientador: Edna Natividade da Silva Barros.
Tese (Doutorado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2022.
Inclui referências e apêndices.

1. Engenharia da computação. 2. SVM. 3. FPGA. I. Barros, Edna Natividade da Silva (orientador). II. Título

621.39 CDD (23. ed.) UFPE - CCEN 2023 – 004

LUCAS FERNANDO DA SILVA CAMBUIM

"**SISTEMA DE PREVISÃO DE COLISÃO DE PEDESTRE BASEADO EM VISÃO COMPUTACIONAL**"

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação. Área de Concentração: Engenharia da Computação.

Aprovado em: 31/10/2022.

Orientadora: Profa. Dra. Edna Natividade da Silva Barros

BANCA EXAMINADORA

Prof. Dr. Carlos Alexandre Barros Mello
Centro de Informática / UFPE

Prof. Dr. Abel Guilhermino da Silva Filho
Centro de Informática / UFPE

Prof. Dr. Stefan Michael Blawid
Centro de Informática / UFPE

Prof. Dr. Filipe Rolim Cordeiro
Departamento de Estatística e Informática / UFRPE

Prof. Dr. Cesar Albenes Zeferino
Escola do Mar, Ciência e Tecnologia / UNIVALI

Eu dedico este trabalho à minha família, aos meus amigos, a minha futura esposa e professores que deram apoio incondicional, força, incentivo e amizade sem igual. Sem eles nada disto seria possível.

AGRADECIMENTOS

Dedico este momento à minha família, sobretudo ao meu querido e eterno pai, Fernando Cambuim e à minha mãe, Sônia Cambuim. Eles estiveram comigo em toda minha caminhada e, mesmo com todas as dificuldades, me deram tudo que eu precisei: o pão de cada dia, os livros do colégio, o carinho, os brinquedos. Não há como esquecer dos muitos beijos e cheiros na minha cabeça que meu pai me dava. Também não poderei jamais esquecer a difícil decisão que minha mãe precisou tomar de largar o emprego para cuidar de mim e me apoiar nos estudos, principalmente quando eu ainda era bem pequeno. Eles, juntos, me deram o principal: o Amor! A minha maior inspiração! O que me move a ser um bom filho, um bom amigo, um bom cidadão, um bom esposo e um bom profissional. Se hoje estou aqui, me formando, é graças aos meus pais. Papai do Céu resolveu antecipar a ida do meu Pai, mas eu sei que, onde ele estiver, estará muito feliz por mim. Pai, eu te Amo! E nunca me esquecerei de você!

Agradeço, acima de tudo, a Deus! Por me conceder este momento, e porque sem Sua permissão, nada disso poderia acontecer. Deus sempre me amparou nos momentos complicados durante estes anos de estudo. E se por algum momento estive preocupado, com medo, ansioso, me sentindo incapaz, Ele sempre me lembrava que eu sou mais forte do que eu penso. Agradeço aos anjos que Deus me enviou antes e durante esta minha caminhada na Universidade. Essas pessoas, com bonitos gestos, me apoiaram e acreditaram em mim. Entre esses anjos estão as minhas tias Fracinete Cambuim, Floripes Cambuim, Idalina Cambuim; e as minhas professoras Tatiana, Luzinete Kurtinaitis, e Edna Natividade. Quero ainda fazer um agradecimento especial ao meu irmão, Rodrigo Cambuim, que esteve sempre presente e me deu muita força para continuar firme. Agradeço a minha futura esposa, Valdeilma Oliveira, que sempre esteve ao meu lado e aguentou serenamente as minhas oscilações de humor, principalmente nos anos finais deste trabalho. E por fim, meu muito obrigado ao meu amigo de faculdade, Fernando Maciano, que sempre me deu força e que caminhou comigo desde a graduação. Se eu pudesse dar um único conselho, eu diria: estude e confie em Deus, que o resto Ele te acrescentará.

Obrigado a todos por acreditarem em mim!

RESUMO

Todos os anos, várias centenas de pessoas morrem devido a acidentes rodoviários. O fator humano é responsável por mais de 90% dos acidentes rodoviários. Sistemas de previsão de colisão de pedestres (PCP) baseados em câmeras são fundamentais para redução de acidentes pois permitem tomadas de decisão antecipadas. Conforme os veículos se tornam mais rápidos, o risco de acidentes fatais com pedestres aumenta, exigindo sistemas PCP com tempo de resposta cada vez menor e com capacidade de prever colisões cada vez mais distantes. Tais sistemas dão ao veículo a capacidade de reduzir a velocidade de colisão ou até mesmo evitá-la. Estudos afirmam que diminuir a velocidade de impacto em 30km/h reduz a chance de um adulto morrer em 40%. As etapas mais críticas e computação intensiva de dados em um sistema PCP são a detecção de pedestres e correspondência estéreo. Nesta Tese, nós propomos soluções otimizadas em taxa de processamento para estas etapas implementadas em plataformas de baixa dissipação de potência e com restrição de recursos como *Field-Programmable Gate Array* (FPGA). Para a etapa de detecção nós propomos um módulo que implementa as técnicas de *Histogram of Oriented Gradients* (HOG) e *Support Vector Machine* (SVM) com suporte para pirâmide de imagens e janelas de detecção de diferentes tamanhos para localização de pedestres próximos ou distantes. A arquitetura proposta do detector consegue processar pixels na vazão máxima e sem perda de *frames*, através de técnicas de pipeline e divisão espacial de pixels entre unidades de processamento paralelo. Estratégias de otimização de recursos foram propostas tais como o compartilhamento da memória de pesos e resultados entre as unidades. Para correspondência estéreo nós implementamos uma solução de alto desempenho da técnica *Semi Global Matching* (SGM) e propomos melhorias em precisão e adição do suporte à detecção de pixels de oclusão que permitiu alcançarmos alto desempenho e resultados precisos de localização de pedestre. O processamento de duas janelas de detecção de dimensões diferentes permitiu uma redução na taxa de falta de pelo menos 25% em comparação com um detector de janela de tamanho único. Os desempenhos alcançados pelo detector e pelo sistema PCP em resolução HD foram de 130 e 66,2 *frames* por segundo, respectivamente. Os resultados demonstram que o desempenho de ambos é constante independentemente da quantidade de janelas a serem processadas. Para demonstrar os benefícios do ganho de desempenho, nós desenvolvemos uma estratégia de avaliação que envolveu a criação de uma base de dados sintética de colisões com pedestres. A melhoria de desempenho alcançada pelo sistema PCP com a adição dos módulos de hardware propostos permitiu um aumento em distância para tomada de decisão de 6,5 metros em comparação com outros sistemas PCP mais precisos, mas que são mais lentos. Este trabalho cria alicerces para mais estudos em melhoria de previsão através de desempenho de processamento.

Palavras-chave: SVM; FPGA; alto desempenho; otimização de recursos; detecção de pedestres distantes; previsão de colisão eficiente.

ABSTRACT

Every year, several hundred people are killed due to road accidents. The human factor is responsible for more than 90% of road accidents. Pedestrian collision prediction (PCP) systems based on cameras are essential for reducing accidents because they allow for early decision-making. As vehicles become faster, the risk of fatal pedestrian accidents increases, requiring PCP systems with shorter response times and capacities to predict collisions further away. Such systems allow the vehicle to reduce collision speed or even avoid it. Studies claim that decreasing impact speed by 30km/h reduces the chance of an adult dying by 40%. Most existing solutions focus on efficiency improvement by prediction accuracy. However, solutions that address improvement by processing performance still remain open. The most critical and data-intensive steps in a PCP system are pedestrian detection and stereo matching. In this Thesis, we propose throughput-optimized solutions for these steps implemented in platforms based on Field-Programmable Gate Array (FPGA) with low power dissipation. For the detection step, we propose a module that implements the Oriented Gradient Histogram (HOG) and Support Vector Machine (SVM) techniques with support for images pyramid and detection windows of different sizes to locate nearby or distant pedestrians. The proposed detector architecture can process pixels serially at maximum throughput and without frames loss through techniques such as pipeline and spatial division of pixels between parallel processing units. Resource optimization strategies have been proposed, such as weight memory sharing among all parallel units. For stereo matching, we implement a high-performance solution of the Semi Global Matching (SGM) technique, and we propose improvements in accuracy and the addition of support for the detection of occlusion pixels that enabled us to achieve high-performance and accurate pedestrian location results. Processing two detection windows of different dimensions with a 7-level pyramid reduced the miss rate by at least 25% compared to a single-size window and 1-level pyramid detector. The performances achieved by the detector and the PCP system in HD resolution were 130 and 66.2 frames per second, respectively. Furthermore, results demonstrate that processing performances are constant regardless of the number of windows to be processed. To demonstrate the benefits of performance, we developed an assessment strategy that involved building a synthetic database of pedestrian collisions. The performance improvement achieved by the PCP system with the addition of the proposed hardware modules allowed an increase in decision-making distance of 6.5 meters compared to other more accurate but slower PCP systems.

Keywords: SVM; FPGA; high performance; resource optimization; distant pedestrian detection; efficient collision prediction.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ADAS	*Advanced driver-assistance system*
AEB	*Autonomous Emergency Braking*
AFU	*Accelerator Functional Unit*
BB	*Bounding Box*
CARLA	Simulador de código aberto para pesquisa em direção autônoma.
CNN	*Convolutional Neural Network*
CV	*Constant Velocity*
DPS	Janelas de detecção por segundo
EKF	*Extended Kalman Filter*
FIFO	*First In First Out*
FK	Filtro de Kalman
FKE	Filtro de Kalman Estendido
FPGA	*Field Programmable Gate Array*
FPPI	Falso Positivo por Imagem
FPS	*Frames* Por Segundo
GPIO	*General Purpose Input/Output*
GPP	*General Purpose Processor*
GPU	*Graphics Processing Unit*
GT	*Ground Truth*
HARPv2	*Open Programmable Acceleration Engine*
HOG	*Histograms of Oriented Gradients*
HW/SW	Hardware/Software
IoU	*Intersection over Union*
JPD	Joules por detecção
JPF	Joules por *frame*
KF	Kalman *Filter*
LBP	*Local Binary Patterns*
NMS	*Non-maximum Suppression*
OPAE	*Open Programmable Acceleration Engine*
OpenCV	*Open Source Computer Vision Library*

P-CCI	Protocolo *Core Cache Interface*
PPA	Proteção ao Pedestre Ativo
P-SVM	Pirâmide de SVMs
P-HOG	Pirâmide de HOGs
P-Imagem	Pirâmide de Imagens
PCP	Previsão de Colisão de Pedestre
RGB	*Red,Green,Blue*
ROI	*Region of Interest*
SGM	*Semi Global Matching*
SGM-FPGA	Versão da técnica SGM para plataforma FPGA
SGM-CUDA	Versão da técnica SGM para plataforma GPU
SVM	*Support Vector Machine*
TTC	*Time to Collision*
UI	Unidades de Interpolação
VRU	*Vulnerable Road User*
YOLO	*You Only Look Once*
YOLOv3	*You Only Look Once* versão 3

LISTA DE SÍMBOLOS

α	Letra grega Alfa
β	Letra grega Beta
γ	Letra grega Gama
vs	Versus
$\in$	Pertence
$\mathbb{R}$	O conjunto dos números reais.
$\geq$	Maior ou Igual
$\wedge$	Conectivo lógico de conjunção
δ	Delta
m	Metro
km	Quilômetro
s	Segundo
m/s	Metro por segundo
m/s^2	Metro por segundo ao quadrado
km/h	Quilômetro por hora
W_{img}	Largura em pixels da imagem.
H_{img}	Altura em pixels da imagem.
W_{SVM}	Largura em pixels da janela de detecção do SVM.
H_{SVM}	Altura em pixels da janela de detecção do SVM.
W_{cel}	Largura em pixels da célula HOG
H_{cel}	Altura em pixels da célula HOG
W_{bloc}	Largura em pixels do bloco HOG
H_{bloc}	Altura em pixels do bloco HOG
Q_{bins}	Quantidade de *bins* em uma célula HOG

D_{pir}	Profundidade da pirâmide de imagens
S_{pir}	Fator de escala da pirâmide de imagens
σ_{SVM}	Confiança de classificação usando a técnica SVM.
N_d	Intervalo de disparidades.
C_{inteiro}	Quantidade de bits para a parte inteira de um número em representação de ponto fixo
$C_{\text{fração}}$	Quantidade de bits para a parte fracionária de um número em representação de ponto fixo
V_{car}	Velocidade do veículo
(u, v, d)	coordenadas no espaço de imagem
(x, y, z)	coordenadas no espaço de câmera
(X, Y, Z)	coordenadas no espaço do mundo
σ_u	Desvio padrão do erro de medição da posição lateral no espaço de imagem
σ_d	Desvio padrão do erro de medição da disparidade

SUMÁRIO

1 INTRODUÇÃO

Os automóveis trouxeram muitos benefícios para sociedade tais como agilidade, conforto e segurança no deslocamento de pessoas e mercadorias para lugares distantes e remotos trazendo como consequência maior interação social e econômica. Contudo, com o crescimento da quantidade de veículos nas ruas e da tecnologia de motorização, o tráfego se tornou cada vez mais complexo e perigoso. Um relatório da Organização Mundial da Saúde (2018) informa que aproximadamente, no mundo, 1,35 milhão de pessoas morrem a cada ano como resultado de acidentes de trânsito. Neste mesmo relatório, os pedestres, considerados usuários de estrada vulneráveis (em inglês, *vulnerable road user* - VRU), representam 18% de um total de 180.000 mortes de trânsito no Brasil.

Acidentes de trânsito produzem altos custos emocionais e sociais para as vítimas e suas famílias. Eles são capazes de gerar danos mentais, dias de ausência e diminuição da produtividade do trabalho. Consequências psicológicas e transtornos de estresse pós-traumático resultantes de acidentes de trânsito são potencialmente incapacitantes a longo prazo.

A maioria dos acidentes tem sido ocasionada por erro do condutor que não agiu em tempo hábil. Este erro é devido à capacidade limitada dos seres humanos de adquirir informação, raciocinar e de agir adequadamente. Estudos (ROLISON et al., 2018) mostram que mais de 90% dos acidentes de trânsito são devidos a falha humana, com mais de 90% devido a problemas de aquisição de informação visual. Além disso, a falta de atenção do motorista é uma das causas mais comuns de acidentes com pedestres que resulta a partir de falar no celular, comer enquanto dirige, programar um GPS, de aplicar maquiagem ou várias outras atividades. Um relatório da Administração de Segurança de Tráfego de Estrada Nacional dos Estados Unidos (2017) afirma que somente em 2017, 3.166 pessoas foram mortas em acidentes de automóvel envolvendo motoristas distraídos.

Com o intuito de reduzir a quantidade e a severidade dos acidentes, empresas especializadas em segurança, produtores de veículos e universidades têm juntado esforços para desenvolver sistemas de proteção ao pedestre, que podem ser classificados com sistemas ativos e passivos. Os sistemas passivos envolvem estruturas de veículos (por exemplo, capô, para-choques) que se expandem durante a colisão para minimizar o impacto da perna ou da cabeça do pedestre ao bater no veículo. Contudo, os sistemas passivos de pedestres são limitados pelas leis da física em termos de capacidade de reduzir a energia de colisão e, portanto, o nível da lesão. Além disso, os sistemas passivos não levam em conta as lesões sofridas no impacto secundário do pedestre na pista ou calçada. Muito esforço tem sido empregado, portanto, para o desenvolvimento de sistemas de proteção ao pedestre ativos (PPA), que detectam situações perigosas envolvendo pedestres e tomam algum tipo de decisão como alertar o motorista ou controlar automaticamente o veículo

quando o motorista está desatento ou não foi capaz de visualizar uma situação de risco.

Os sistemas PPAs são componentes chaves dos sistemas avançados de assistência ao motorista (em inglês, *Advanced driver-assistance system* - ADAS) e direção autônoma (PAYALAN; GUVENSAN, 2019) (HAAS; BHATTACHARJEE; MÖLLER, 2020). Eles baseiam-se em sensores que extraem informações úteis do veículo e do ambiente, analisam e tomam alguma decisão. Diferentes tipos de sensores têm sido utilizados nos sistemas PPAs para obter informações úteis do ambiente, tais como radar, sonar, lidar e óptico. Cada sensor possui características que o diferencia dos outros sensores de tal maneira que a combinação deles é benéfica para aumentar a cobertura de todos os cenários de acidentes. Este sistema, juntamente com os sensores e a estratégia de processamento, precisam lidar com restrições impostas pelo veículo como baixa dissipação de potência e recursos de processamento limitados (LIU et al., 2021).

1.1 MOTIVAÇÃO

Hoje, ainda permanece um desafio desenvolver sistemas PPAs robustos devido à alta complexidade do tráfego. Os sistemas de previsão de colisão de pedestre (PCP) são fundamentais para que os sistemas PPAs consigam detectar, antecipadamente, uma possível situação de colisão e assim possibilitem que o sistema do veículo reaja mais rapidamente e consiga, na pior das hipóteses, mitigar o acidente. A habilidade de iniciar uma frenagem com alguns segundos de antecedência tem o potencial para reduzir a severidade do acidente (HAMDANE et al., 2015)(XIA; CHUNG; KASSIM, 2013). Um relatório da Organização Mundial da Saúde (2018) afirma que um adulto tem menos de 20% de chance de morrer se for atropelado por um carro a menos de 50 km/h, mas possui um risco de 60% de morrer se for atingido por um carro a 80 km/h.

Uma vez que os veículos estão se tornando cada vez mais rápidos, o risco de acidentes fatais envolvendo pedestres aumenta ainda mais de acordo com a Organização Mundial da Saúde (2018). Carros mais rápidos exigem uma distância ainda maior para tomada de decisão (LI et al., 2020). Dessa forma, dois aspectos são essenciais em sistemas PCP para que possam ser aplicados em veículos em velocidades mais altas: (1) cobertura em grandes intervalos de distância e (2) resposta rápida.

Uma maneira de cobrir pedestres mais distantes é processar dados em resoluções cada vez maiores (SULEIMAN; SZE, 2016). Sensores ópticos (câmeras) são uma escolha atraente, pois fornecem uma grande quantidade de informações do ambiente em forma de pixels. Comparado com radar e lidar, os sensores ópticos possuem resoluções de informações do ambiente e taxas de amostragem muito maiores (WANG, 2021)(LI et al., 2022) (XU, 2021)(DURINI, 2019) (BHOWMIK; PANTHO; BOBDA, 2021)(BHOWMIK; PANTHO; BOBDA, 2019). Câmeras de maior resolução facilitam a previsão de colisão ao suportar pedestres distantes, pois serão codificados em um número maior de pixels. Um aumento em resolução

em resolução de 480 × 320 para 1920 × 1080 pixels permite aumentar a distância de localização do pedestre em 77 metros (HELALI et al., 2020).

As abordagens para um sistema PCP precisam processar os *frames* eficientemente para alcançar altas taxas; que se traduz em menor perda de informação importante para a previsão de colisão. Soluções baseadas em hardware dedicado, como as plataformas formadas por Arranjo de Portas Programáveis em Campo (em inglês, *Field Programmable Gate Array* - FPGA) têm o potencial de alcançar alto desempenho de processamento devido à sua natureza de processamento estritamente paralela além da sua capacidade de processamento em fluxo contínuo de dados vindo diretamente da câmera (LIU et al., 2021; BAILEY, 2019; BAILEY, 2011).

1.2 PROBLEMA DE PESQUISA E HIPÓTESES

O desafio em aberto é desenvolver sistemas PCPs baseados em visão computacional que consigam se antecipar cada vez mais na previsão de uma colisão iminente envolvendo pedestre. Sistemas PCPs existentes não lidam diretamente com desempenho de processamento, nem com localização de pedestres distantes, além de não suportarem altas resoluções (WU, 2016; KELLER et al., 2011; LLORCA et al., 2011; LEE; SHIN; KWON, 2017; Park et al., 2017).

Mais especificamente, a informação fundamental do pedestre para a realização da previsão de colisão é a sua localização e sua obtenção depende de duas etapas críticas (OTTO, 2013; WU; ZHOU; SRIKANTHAN, 2016; LIU et al., 2021): a detecção de pedestres e a correspondência estéreo. Estas etapas impactam não somente na precisão, mas também no tempo de resposta do sistema PCP (WU; ZHOU; SRIKANTHAN, 2016; OTTO, 2013; KELLER et al., 2011). Por demandarem um uso intensivo de dados, estas etapas são os maiores gargalos de desempenho de processamento do sistema PCP. Além disso, a demanda por sistemas PCPs de baixo consumo de energia, impõem desafios ainda maiores na proposição de técnicas para estas duas etapas utilizando plataformas de baixo consumo de energia (ZENG et al., 2022).

Com relação à etapa de detecção de pedestres, os detectores baseados em aprendizagem profunda (GOODFELLOW; BENGIO; COURVILLE, 2016) possuem a capacidade para detectar pedestres em maiores variabilidades de poses e tamanhos. Porém tais detectores exigem um alto custo de memória e possuem grande dependência de dados que impede de obtermos soluções de processamento eficientes em maiores resoluções (NGUYEN et al., 2019). Por outro lado, os detectores que combinam histograma de gradientes orientados (em inglês, *Histogram of Oriented Gradients* - HOG) com classificadores lineares, como máquinas de vetor de suporte (em inglês, *Support Vector Machine* - SVM) têm demonstrado resultados promissores ao detectar pedestres (BILAL; HANIF, 2019). Estes detectores oferecem um ótimo compromisso entre precisão e complexidade (SULEIMAN; ZHANG; SZE, 2017). As características HOG têm persistido como padrão por muitos anos já que o HOG

captura densamente informações de gradiente, sendo robusto a rotações, translações e variações de luminosidade (SANGEETHA; DEEPA, 2017)(LUO; LIN, 2018).

Assim, alguns detectores baseados em FPGAs foram propostos que aceleram o processamento das técnicas HOG e SVM. O maior desafio, ainda em aberto, ao se desenvolver tais detectores é lidar com a grande sobreposição de dados que existe no processo de janelamento. Os detectores propostos por Luo e Lin (2018) e Helali et al. (2020) detectam pedestres em um intervalo estreito de distância. Além disso, algumas simplificações que diminuem a capacidade de detecção, como por exemplo redução do espaço de busca, foram adotadas para lidar com a sobreposição. Em Ma, Najjar e Roy-Chowdhury (2015), os autores propõem um detector capaz de aumentar o intervalo de distância de detecção, mas é fortemente dependente do desempenho da memória externa, aumentando o seu consumo de energia e reduzindo o seu desempenho de processamento, além de não lidar eficientemente com a sobreposição. Os detectores propostos por Hahnle et al. (2013) e Dürre, Paradzik e Blume (2018) suportam um intervalo maior de distância, similar ao detector de Ma, Najjar e Roy-Chowdhury (2015), e são independentes de memória externa. Porém, para lidar com sobreposição e alcançar uma taxa maior de processamento, os autores dobram a frequência de processamento do detector, reduzindo a sua escalabilidade para câmeras com frequências de operação cada vez maiores (DURINI, 2019).

Dessa forma, é necessário desenvolver detectores baseados em HOG e SVM mais eficientes para alcançar maiores intervalos de distâncias e maior desempenho de processamento. Além disso, melhorias de qualidade como redução de falsos positivos são necessários nestes detectores e são pouco investigados dentro do contexto de veículos em estradas (KELLER et al., 2011). A combinação de detectores baseados em HOG e SVM com etapas complementares de custo computacional simples, que extraem dicas da estrada, podem alcançar precisão similar aos detectores baseados em aprendizagem profunda, sem perder desempenho de processamento (WU; ZHOU; SRIKANTHAN, 2016).

Além da detecção de pedestres, a correspondência estéreo cumpre um papel fundamental para estimativa precisa da localização. Nós destacamos a técnica de correspondência estéreo semi global (em inglês, *Semi-Global Matching* - SGM) (HIRSCHMULLER, 2008) que realiza uma otimização em toda a imagem, produzindo dados de profundidade mais precisos para o contexto urbano. Devido à sua alta precisão, esta técnica tem sido usada em muitas aplicações como detecção/reconhecimento de objetos (GUINDEL; MARTíN; ARMINGOL, 2019) e navegação autônoma (RASHED et al., 2019) que requerem processamento em tempo real de dados de profundidade em imagens de alta resolução. A técnica SGM demanda um alto custo computacional e, dessa forma, nós desenvolvemos uma solução de alto desempenho que processa dados de profundidade usando esta técnica (CAMBUIM; BARBOSA; BARROS, 2017). Contudo, problemas de baixa textura e oclusão de pixels (HIRSCHMULLER; SCHARSTEIN, 2009) são enfrentados por esta solução que reduzem a precisão da estimativa de localização de pedestres. Além disso, a imprecisão da detecção de pedes-

tres também influencia diretamente na imprecisão da localização de pedestres.

Desta forma, os problemas de pesquisa desta tese são enumerados a seguir:

1. É possível ter detectores de pedestres baseados em HOG e SVM de alto desempenho e baixo consumo de energia capazes de detectar pedestres em intervalos de distâncias maiores do que os sistemas atuais conseguem detectar ?

2. É possível ter sistemas de correspondência estéreo baseados em SGM de alto desempenho que permitam estimar dados de localização precisos mesmo na presença de regiões de oclusão e que sejam de baixo consumo de energia ?

3. É possível ter sistemas PCPs com tempo de respostas menores do que os sistemas atuais cuja a detecção de pedestres e correspondência estéreo sejam de baixo consumo de energia ?

4. É possível que tais sistemas PCPs proporcionem maiores distâncias para tomada de decisão pelo condutor e/ou veículo do que os sistemas atuais ?

5. É possível que tais sistemas PCPs reduzam a quantidade de acidentes que envolvam colisão com pedestre comparado com os sistemas atuais ?

1.3 OBJETIVOS

1.3.1 Objetivo Geral

O objetivo desta tese é o desenvolvimento de um sistema PCP de alto desempenho com implementações eficientes do módulo de hardware de detecção baseado em HOG e *Support Vector Machine* e correspondência estéreo baseado em *Semi Global Matching*, para suportar a previsão de pedestres distantes e aumentar a distância para tomada de decisão.

1.3.2 Objetivos Específicos

Para o alcance do objetivo geral, os seguintes objetivos específicos são definidos:

- Desenvolvimento de abordagens eficientes em FPGA para explorar maiores intervalos de distâncias do que os detectores existentes.
- Desenvolvimento de abordagens em FPGA para detecção de profundidades erradamente estimadas.
- Desenvolvimento de estratégias eficientes para proporcionar menores falsos positivos de detecção do que os detectores atuais.
- Implementação de algoritmos largamente adotados para cada etapa do sistema PCP.

- Desenvolvimento de estratégias para integração eficiente dos módulos de hardware com as outras etapas do sistema PCP.
- Desenvolvimento de estratégias para avaliação de sistemas PCPs.

1.4 METODOLOGIA DE PESQUISA

Uma plataforma heterogênea baseada em um processador de propósito geral (em inglês, *general purpose processor* - GPP) e FPGA é utilizada para validar a implementação do sistema PCP proposto. O GPP processa as etapas menos dispendiosas computacionalmente.

Os módulos de detecção e correspondência estéreo propostos são avaliados com relação a capacidade do sistema PCP de localizar e prever uma colisão. Base de dados pública é utilizado para avaliação da precisão da localização e comparação com outras abordagens. Diversas análises e comparação de desempenho de processamento e ocupação de recursos computacionais são realizados para o sistema PCP e os módulos propostos.

Para avaliação e comparação da previsão de colisão, uma base de dados foi criada de colisões envolvendo travessia de pedestres obstruídos na frente do veículo em movimento. Esses cenários representam a maioria dos acidentes e são desafiadores devido à necessidade de uma reação rápida do veículo. Além disso, métricas para avaliação da eficiência da previsão de colisão foram propostas.

1.5 PUBLICAÇÕES RELACIONADAS A TESE

1.5.1 Periódico

- Cambuim, Lucas, and Edna Barros. FPGA-based pedestrian detection for collision prediction system. Sensors 22.12 (2022): 4421. <http://dx.doi.org/10.3390/s22124421>
- Cambuim, L.F.S., Oliveira, L.A., Barros, E.N.S. et al. An FPGA-based real-time occlusion robust stereo vision system using semi-global matching. J Real-Time Image Proc (2019). <http://doi.org/10.1007/s11554-019-00902-w>

1.5.2 Conferência

- Cambuim, Lucas FS, and Edna Barros. Supporting Detection of Near and Far Pedestrians in a Collision Prediction System. VISIGRAPP (4: VISAPP). 2021. <http://dx.doi.org/10.5220/0010253706690676>
- Lucas F. S. Cambuim, Severino J. B. Júnior, Edna N.S. Barros. A Strategy to Support Streaming Communication using the Intel HARPv2 Platform: A Case Study

in Stereo Vision Application. The 18th IEEE International NEWCAS Conference (2020). <http://dx.doi.org/10.1109/NEWCAS49341.2020.9159771>

- de Melo, Mirella Pessoa, Lucas Cambuim, and Edna Barros. "Occupancy Grid Map Estimation Based on Visual SLAM and Ground Segmentation."2021 Latin American Robotics Symposium (LARS), 2021 Brazilian Symposium on Robotics (SBR), and 2021 Workshop on Robotics in Education (WRE). IEEE, 2021. <http://dx.doi.org/10.1109/LARS/SBR/WRE54079.2021.9605417>

1.6 ORGANIZAÇÃO DO TRABALHO

Os capítulos restantes desta tese encontram-se estruturadas da seguinte forma:

Capítulo 2 - ESTADO DA ARTE: descreve com maiores detalhes o problema de previsão de colisão abordado neste trabalho de pesquisa, as diversas etapas envolvidas, conceitos básicos, desafios e soluções existentes em cada uma delas.

Capítulo 3 - PREVISÃO DE COLISÃO DE PEDESTRES: apresenta o sistema de previsão de colisão construído, detalhando as técnicas implementadas em cada etapa, as bases de dados utilizadas e construídas e as métricas adotadas para avaliação do detector e dos sistema de previsão de colisão.

Capítulo 4 - ABORDAGEM DE DETECÇÃO DE PEDESTRES PROPOSTA: Apresenta uma solução baseada em HOG e SVM para detectar pedestre em um intervalo de distância maior através da combinação de janelas de detecção de dimensões diferentes.

Capítulo 5 - ARQUITETURA DE HW PARA DETECÇÃO DE PEDESTRES: descreve a arquitetura de hardware com otimização de desempenho e recurso para detecção de pedestres baseada em HOG e SVM com suporte a janelas de detecção de dimensões diferentes.

Capítulo 6 - ARQUITETURA HW/SW PARA LOCALIZAÇÃO: descreve a arquitetura HW/SW para etapa de localização que combina o módulo SGM com etapa em software para estimativa de distância.

Capítulo 7 - CONCLUSÃO: apresenta as considerações finais sobre os principais tópicos abordados nesta tese, incluindo as contribuições alcançadas e as indicações de trabalhos futuros.

2 ESTADO DA ARTE

O sistema de previsão de colisão de pedestre (PCP) é um componente fundamental pra reduzir os riscos de acidentes ou mitigar o impacto da colisão. As escolhas de tecnologia para sensoriamento (por exemplo: lidar, sonar, radar, óptico) e técnicas para cada uma das etapas do sistema PCP definirá a precisão da previsão de colisão em diversos cenários reais. Os cenários incluem vários tipos de ambiente, de movimento de pedestres, quantidade de pedestres, oclusão de pedestres, distância do pedestre, velocidade do veículo entre outras.

A eficiência de um sistema PCP não depende somente da precisão, mas também do tempo de resposta para prever situações de risco de colisão. Os eventos que antecedem a colisão são repentinos e demandam uma reação do motorista ou do próprio veículo cada vez mais rápida. Três aspectos que tornam a tarefa de previsão de colisão desafiante são as dinâmicas de movimentação do pedestre que pode mudar de direção e velocidade rapidamente, aparecimento repentino do pedestre, e a movimentação do veículo que atinge velocidades cada vez mais altas. Assim, o tempo para previsão de uma colisão precisa ser cada vez mais reduzido.

Neste trabalho, nós focamos não somente no aspecto de tempo de resposta, para melhoria da eficiência da previsão de colisão, mas também em dissipação de potência, custo e resolução de dados. Estes últimos são cruciais para se ter soluções que possam ser adotados nos veículos comerciais. Os Sensores lidar obtém excelentes resultados com medições tridimensionais precisas. Contudo o alto custo e alta dissipação de potência desses sensores dificultam sua aplicação para implantação em larga escala em veículos inteligentes. O radar pode medir diretamente a posição e a velocidade dos objetos, ser robusto a condições climáticas adversas e ser de baixo custo. Contudo este sensor é ruim em modelar uma forma perfeitamente precisa do objeto. Como resultado, o sistema pode não ser capaz de identificar exatamente qual é o objeto (WANG, 2021). Além disso, tanto o radar como lidar possuem baixa taxa de atualização de dados em nível de *frame*, com taxas alcançando até 10 *frames* por segundo (FPS) (GRIMM et al., 2022).

As câmeras são tecnologias que proporcionam alta resolução, alta taxa de atualização de *frames*, transmissão de informações ricas, baixo custo e baixa dissipação de potência (WANG, 2021). A câmera é considerada em todos os sistemas PCPs como o sensor fundamental. Uma vez que câmeras possuem visão deficiente sob eventos climáticos extremos, tais como tempestades de neve, tempestades de areia ou outras condições que levam a baixa visibilidade, os sensores de radar são utilizados como um complemento da câmera para melhorar a precisão do sistema de previsão (Park et al., 2017). A adoção de câmeras permite a captura de uma grande quantidade de informação acerca dos pedestres. Além disso, os avanços em tecnologias de câmera em termos de resolução de imagem e taxa de processamento tem possibilitado aumentar a capacidade dos sistemas de previsão de

colisão, lidando com cenários que antes não eram possíveis. Em virtude destas vantagens, nós focamos em sensores ópticos.

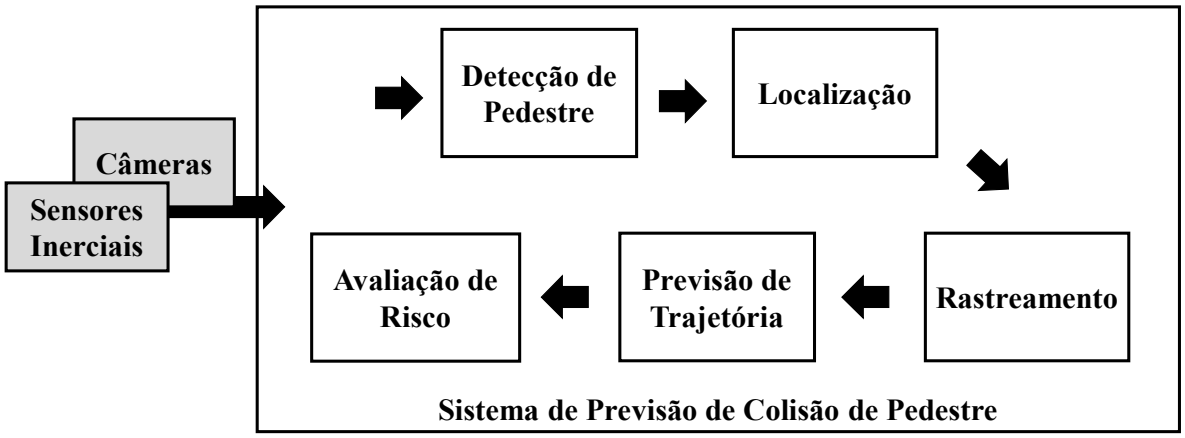
Nas próximas seções nós descrevemos um sistema de previsão de colisão típico baseado em câmeras, bem como os desafios e as técnicas existentes para cada uma das etapas deste sistema. Em seguida, nós descrevemos alguns trabalhos que propõem sistemas completos de previsão de colisão.

2.1 VISÃO GERAL DE SISTEMAS DE PREVISÃO DE COLISÃO

A base para uma previsão de colisão é a extração de informações relevantes do ambiente e do tráfego a partir de dados de sensores. Para atender aos requisitos de previsão robustos, os sistemas de previsão de colisão combinam e fundem entradas a partir de vários sensores. As configurações mais comuns processam dados individuais ou combinados de sensores lidar, radar e ópticos.

Com foco nos sensores ópticos, a parte de aquisição de informação é constituída de uma ou mais câmeras. As câmeras geram dados do ambiente em forma de sequência de *frames* a uma determinada taxa de coleta. É importante esclarecer que os termos *frame* e imagem têm o mesmo significado, ou seja, ao escolher um *frame* específico, nós teremos diretamente a formação de uma imagem, sem qualquer codificação específica de compressão de imagens (SALOMON; MOTTA, 2010). Dessa forma, esses termos podem ser utilizados, neste trabalho, como sinônimos sem qualquer mudança de interpretação.

Figura 1 – Etapas de um sistema de previsão de colisão típico



Fonte: O autor.

Além de câmeras, os sensores inerciais são fundamentais para conhecer a dinâmica do veículo. Eles estimam dados de movimentação do veículo, tipicamente, sua velocidade e taxa de guinada (em inglês, *yaw rate*), coletados a uma taxa específica dos sensores. A taxa de guinada descreve o quão rápido um veículo está mudando de direção. Os dados

de movimentação são fundamentais para as corretas interpretações das informações dos *frames*, mesmo com veículos em movimento.

A primeira etapa de processamento em um sistema de previsão de colisão baseado em visão computacional inclui a detecção de pedestres em imagem única, como mostrado na Figura 1. A tarefa de detecção de pedestre determina as regiões que delimitam cada pedestre de interesse em uma dada imagem.

A etapa de localização é responsável por estimar a localização do pedestre no espaço de coordenadas estabelecidas. A tarefa de rastreamento realiza as associações dos pedestres detectados ao longo de vários *frames*. O rastreamento também é importante para aumentar a robustez da detecção com o aproveitamento de informações temporais. A previsão de trajetória projeta posições futuras do pedestre e do veículo dentro de um horizonte de tempo. A partir dos dados projetados e aspectos físicos do veículo, tais como capacidade de desaceleração máxima, tempo de reação do motorista, taxa de desvio máxima, a probabilidade de colisão, quando ocorrerá, a posição de colisão e o valor de impacto podem ser calculados. Em cada uma destas tarefas, dados de *frame* e dos sensores inerciais podem ser utilizados.

2.1.1 Detecção de Pedestre

Para obter uma compreensão completa da imagem, não devemos apenas nos concentrar em classificar diferentes imagens, mas também em estimar as posições e dimensões dos objetos contidos em cada imagem. Esta tarefa é referida como detecção de objetos. A tarefa de estimar posições e dimensões distingue a detecção de objetos da classificação de objetos. O último é baseado em recortes de imagens, cada um contendo um objeto, o primeiro é baseado em imagens que podem incluir apenas objetos ou plano de fundo. Como a detecção de objetos requer a busca de potenciais candidatos a objetos em uma imagem completa, esta tarefa é computacionalmente mais cara do que a classificação de objetos e mais propenso à detecção de falsos positivos.

O nosso foco é na detecção de pedestre que é uma sub-tarefa da detecção de objetos no qual a classificação se reduz a uma decisão binária (ZHAO et al., 2019): pedestre ou plano de fundo. Por causa de suas aplicações diretas em segurança de automóveis, vigilância e robótica, a detecção de pedestres atraiu muita atenção nos últimos anos.

A detecção confiável de pedestres é considerada uma tarefa desafiadora em visão computacional e aprendizagem de máquina. A Figura 2 mostra um cenário típico de estrada em um ambiente urbano e apresenta vários desafios para detecção de pedestres. Os exemplos incluem pedestres em uma multidão, pedestres e crianças escondidos atrás de veículos estacionados ou cobertos por outros pedestres ou objetos e pedestres distantes.

Os requisitos mais importantes que os sistemas de detecção de pedestre precisam lidar a fim de serem aplicáveis em sistemas de previsão de colisão são listados a seguir (KARG; SCHARFENBERGER, 2020):

Figura 2 – Cenário típico de estrada em ambiente urbano

Fonte: Karg e Scharfenberger (2020)(ADAPTADO).

- Devem funcionar em qualquer condição de iluminação, incluindo dia e noite, pôr do sol, amanhecer e sol ausente.
- Devem funcionar em várias direções e distância a partir do veículo.
- Devem suportar diferentes tamanhos, poses, aparências, pontos de vistas do pedestre.
- Devem lidar com condições desafiadoras do ambiente e do clima, incluindo chuva leve e pesada, neve, neblina, sol e uma alta dinâmica na iluminação ambiente, como sombras projetadas.
- Devem funcionar de forma confiável em situações de tráfego e ambientes complexos.
- Os pedestres devem ser detectados com robustez, mesmo quando cobertos por objetos transportados, veículos ou outras pessoas.
- Devem lidar com pedestres individuais em uma multidão e extrair o pedestre mais relevante e crítico que um veículo pode precisar para tomar uma ação.

Além destes desafios, os sistemas de detecção de pedestres precisam ser cada vez mais rápidos e suportar câmeras com resolução mais altas. De fato, câmeras com maior resolução facilitam a tarefa de detecção e permitem que o sistema detecte pedestres distantes, pois eles serão codificados em um número maior de pixels. Por exemplo, usando uma resolução de 480 $\times$ 320 pixels, a distância para o qual um pedestre de 1,5 metros de altura pode ser detectado é de 24 metros. Enquanto com uma resolução de 1920 $\times$ 1080 pixels, esta distância aumentará para 101 metros (HELALI et al., 2020). No trabalho de Otto (2013) os autores afirmam que a precisão da distância é limitada pela resolução, especialmente a longas distâncias o que realmente mostra a importância de se ter resoluções maiores.

Além disso, uma taxa maior de processamento de *frames* melhora o rastreamento de obstáculos, reduz o tempo de resposta do sistema e minimiza a taxa de falsas detecções

(HELALI et al., 2020). Ambos os desafios, desempenho de processamento e resolução, são colocados como baixa prioridade na maioria dos trabalhos em detecção de objetos porque não se tem um requisito importante de segurança. Contudo quando temos um sistema de previsão de colisão é necessário redobrar a atenção nesse desafio. A detecção é um dos objetivos mais desafiadores para alcançar sistemas com maiores desempenhos e que são escaláveis em termos de resolução de imagem.

Na busca por detectores de pedestres que atendam a esses requisitos, diversos trabalhos têm sido propostos na literatura. Um conjunto de trabalhos adotam um fluxo tradicional constituído das etapas de seleção de hipótese, extração de características, classificação de objetos e pós-processamento.

A etapa de seleção de hipótese envolve a geração de um conjunto de janelas de detecção candidatas ou regiões de interesse (em inglês, *region of interest* - ROIs) nas quais serão analisadas a existência ou não de pedestres. A etapa de extração obtém características salientes dos pedestres. A etapa de classificação de pedestres decide, a partir de um conjunto de características, as regiões que contêm pedestres (ou seja, objetos que pertencem à classe de pedestre) e as que não contêm (ou seja, objetos que pertencem à classe de não-pedestre). Para cada região classificada como contendo pedestres são retornadas informações que permitem encontrar estes pedestres na imagem que em geral são pontos de extremidades de um retângulo, chamado de caixa envoltória (em inglês, *bounding box*). A etapa de pós-processamento realiza um refino dos resultados de classificação. Esta etapa permite melhorias na precisão e na redução de falsos positivos através de técnicas de análise espacial e temporal. A etapa de pré-processamento é opcional e envolve a remoção de ruídos da imagem, a remoção de distorções radiais e tangenciais, melhoria de contraste, bem como a remoção de desfoque que são úteis para aumentar o poder de discriminação dos pedestres em relação aos outros objetos e o plano de fundo na imagem.

Outros trabalhos são baseados em aprendizagem de máquina que podem seguir explicitamente o fluxo tradicional de detecção ou implicitamente através de arquiteturas que suportam aprendizagem profunda. A seguir nós explicamos mais estas etapas e abordagens.

2.1.1.1 Seleção de Hipótese

Como diferentes pedestres podem aparecer em várias posições da imagem e ter diferentes proporções ou tamanhos, é uma escolha natural varrer a imagem inteira com uma abordagem de janelas deslizantes multiescala. O conceito de multiescala é realizado a partir da técnica de pirâmide espaço-escala, comumente chamado de pirâmide de imagens, que cria representações da imagem original em várias escalas através de aplicações sucessivas de suavização gaussiana seguido de subamostragem (DOLLAR et al., 2011). A suavização é fundamental para remover os efeitos de *aliasing* que distorcem a imagem e consequentemente reduzem a qualidade da detecção (SZELISKI, 2022).

Uma outra abordagem para construção da pirâmide de imagens utiliza técnicas de interpolação (HAN, 2013) que servem tanto para subamostragem como para sobre-amostragem de imagens. As técnicas de interpolação mais conhecidas são a de vizinhos mais próximos, bilinear e bicúbica. A bilinear e bicúbica lidam bem com os efeitos de *aliasing* por causa da ponderação de pixels vizinhos. Um diferencial das técnicas de interpolação é a possibilidade de reamostrar mais facilmente a imagem pra qualquer tamanho que se deseje, comparado com a abordagem de suavização e subamostragem que apenas geram imagens cuja razão de redução é uma potência de 2.

Embora a estratégia de busca exaustiva associada com multiescala possa descobrir uma grande quantidade de posições possíveis dos pedestres, suas deficiências também são óbvias. Devido a um grande número de janelas candidatas, esta estratégia é computacionalmente cara e produz muitas janelas redundantes. Cabe salientar que em cada janela candidata um conjunto de algoritmos para checagem da existência do pedestre é executado, e, dessa forma, a contribuição da execução de cada janela adiciona um grande custo de processamento para a detecção. No entanto, se apenas uma quantidade fixa de janelas for aplicada, o custo computacional poderá ser reduzido, mas resultados insatisfatórios poderão ser produzidos, como a não localização do pedestre na imagem.

Abordagens adicionais são combinadas com a técnica de janelas deslizantes e pirâmide de imagens com o intuito de reduzir a quantidade de ROIs que são processados pela etapa de classificação de objetos. A chave desta etapa é garantir que pedestres sejam sempre destacados, mesmo a um custo de encontrar regiões que não contenham pedestres. Ao remover regiões que é impossível de conter um pedestre como por exemplo o céu, o detector poderá reduzir o número de falsos positivos bem como o custo computacional da tarefa de classificação. Outras estratégias são a busca seletiva (UIJLINGS et al., 2013) e geração de ROIs a partir de bordas (ZITNICK; DOLLÁR, 2014) que são comumente usadas na etapa de seleção hipótese, mas mostram ser muito custosas.

2.1.1.2 Extrator de Características

Para reconhecer diferentes objetos, nós precisamos extrair características visuais que forneçam uma representação semântica e robusta. Os extratores de características, também chamados de descritores de características, transformam um grande conjunto de dados, que podem ser redundantes, em um conjunto reduzido de características mais representativo. A intenção de um descritor de características para detecção de objetos é generalizar o objeto de tal maneira que o mesmo objeto (neste caso, um pedestre), quando visualizado em condições diferentes na imagem, produza as mesmas características ou similares.

No contexto de detecção de pedestres, os descritores que dependem de padrões de orientação espacial, tais como os histogramas de gradientes orientados (DALAL; TRIGGS, 2005)(MAMMERI; ZUO; BOUKERCHE, 2014) (em inglês, *Histograms of Oriented Gradient* - HOG) e padrões binários locais (TRICHET; BREMOND, 2018) (em inglês, *Local Binary*

Patterns - LBP), têm sido especialmente estudados. A ideia básica do LBP é codificar a vizinhança de cada pixel como um número binário por limiarização. Cada ROI é dividida em várias células. O vetor de características LBP é formado pela concatenação de histogramas normalizados sobre as células. A popularidade do LBP pode ser explicada devido à sua robustez às mudanças de iluminação e ao cálculo rápido de características. A ideia básica do HOG é que a aparência e a forma do pedestre possam ser descritas pela distribuição dos gradientes de bordas. Para o cálculo do HOG cada ROI é dividida em vários blocos sobrepostos e cada bloco contém uma grade de células. Os gradientes de cada célula são acumulados em um histograma baseado em orientação. O vetor de caraterísticas é formado pela concatenação de blocos normalizados localmente. Usar o descritor HOG torna o sistema robusto à mudanças de iluminação e tem demonstrado precisão significativa na detecção de pedestres (BILAL; HANIF, 2019).

Abordagens que combinam o descritor HOG com outros descritores provaram ser capazes de aumentar o poder de descrição dos pedestres (BENENSON et al., 2014a)(GOVARDHAN; PATI, 2014). Para uma visão mais abrangente de extratores de características para a detecção de pedestres, os trabalhos de Dollar et al. (2011), Zhang et al. (2016),Benenson et al. (2014a) e (DAI et al., 2021) são indicados.

2.1.1.3 Classificação de objetos

A etapa de classificação de objetos tem como objetivo identificar objetos, classificando-os em um dos conjuntos finitos de classes. Esse processo envolve a comparação de características de um novo objeto com as características ou modelos de objetos conhecidos para decidir a qual categoria específica este novo objeto pertence. No caso de detecção de pedestre, essa classificação é binária, ou seja, restrita a duas classes: (1) ausência de pedestre e (2) presença de pedestre.

Classificadores baseados em aprendizagem de máquina são geralmente utilizados nesta etapa. Máquina de vetor de suporte (em inglês, *Support Vector Machine* - SVM) é uma técnica bem-sucedida de aprendizagem de máquina supervisionada que têm fortes fundamentos teóricos na solução de problemas de classificação. Basicamente, a técnica SVM define uma linha de separação, mais comumente chamada de hiperplano entre os dados de duas classes.

Além do SVM, nós temos a abordagem chamada de modelo de partes deformáveis (em inglês, *Deformable Part-based Model* - DPM) (FELZENSZWALB et al., 2010) que combina partes de objetos classificados com custo de deformação para lidar com deformações severas. Essa abordagem lida melhor com casos difíceis como oclusões, poses difíceis ou ângulos de visão raros.

Alguns métodos exploram informações adicionais em treinamento e em teste para melhorar as detecções. Eles consideram imagens estéreo (KELLER et al., 2011), fluxo óptico

(PARK et al., 2013) ou dados de outros sensores, como lidar (PREMEBIDA et al., 2014) ou radar.

2.1.1.4 Pós-processamento

Muitos sistemas contêm uma etapa de pós-processamento responsável pela validação e filtragem das ROIs classificadas como pedestres para redução de falsos positivos. Uma técnica essencial nesta etapa é a de supressão não-máxima (em inglês, *non-maximum suppression* - NMS) que resolve um problema típico de abordagens de detecção que é detecção de vários *bounding-boxes* ao redor do mesmo pedestre. Isso acontece porque as janelas da vizinhança têm pontuações semelhantes o que leva a serem consideradas como regiões contendo pedestres.

Basicamente, a aplicação de algoritmos de supressão não-máxima resolve este problema agrupando as janelas detectadas através de proximidade espacial e a confiança das janelas. Diversas técnicas tem surgido para melhorar a capacidade de escolher a janela de detecção correta (SOLOVYEV; WANG; GABRUSEVA, 2021) (BODLA et al., 2017) (Gerónimo et al., 2010).

2.1.1.5 Detectores baseados em aprendizagem profunda

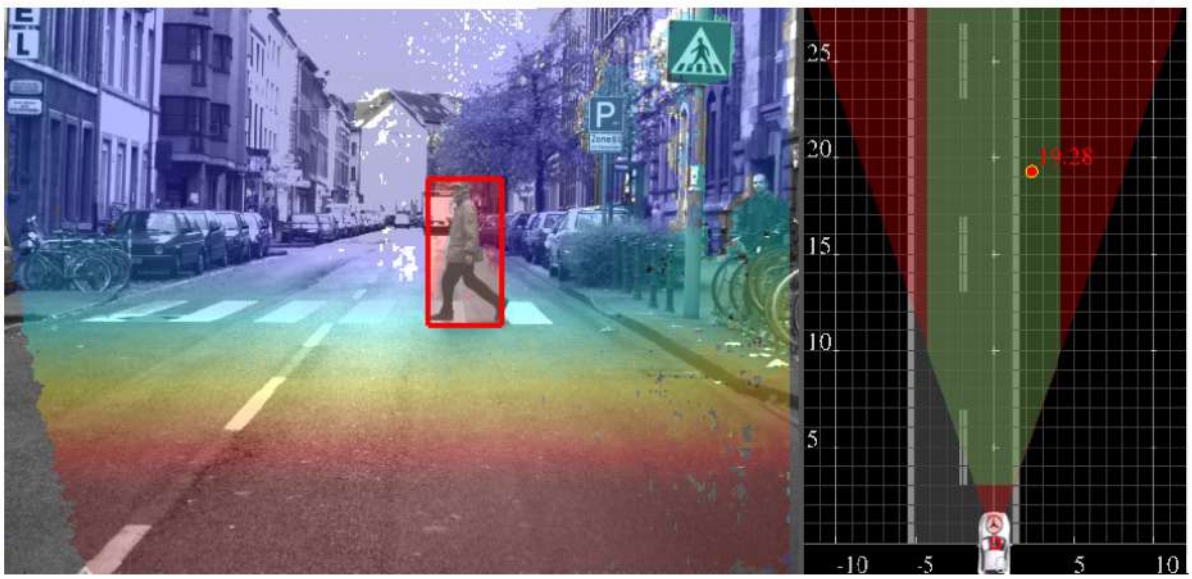
Em técnicas baseadas em aprendizagem profunda, nós temos as redes neurais profundas (em inglês, *Deep Neural Network* - DNN) que permitiu o surgimento das Redes Neurais Convolucionais (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), (em inglês, *Convolution Neural Network* - CNN). Ambos DNNs e CNNs, atuam de forma bem diferente das abordagens tradicionais. Eles têm arquiteturas mais profundas com capacidade de aprender características mais complexas do que as características feitas a mão (em inglês, *handed-craft*) utilizadas pelos classificadores baseados em aprendizagem máquina. Além disso, a expressividade e os algoritmos de treinamento robustos permitem aprender representações informativas de objetos sem a necessidade de projetar características manualmente.

A partir do surgimento das arquiteturas convolucionais profundas muitas variantes têm surgido que mudou o fluxo tradicional de detecção de objetos. Por exemplo, YOLO (*You Only Look Once*) (REDMON; FARHADI, 2018) e SSD (*Single Shot Detector*) (LIU et al., 2016) são abordagens que detectam pedestres em várias escalas e proporções na imagem incorporando o conceito de multiescala e seleção de hipóteses dentro da própria arquitetura neural.

2.1.2 Localização

Informações de localização do pedestre são cruciais em sistemas de previsão de colisão e proteção ao pedestre (GAO et al., 2018). A localização define as distâncias e profundidades dos pedestres em relação ao veículo como mostrado na Figura 3.

Figura 3 – Localização de pedestre



Fonte: Keller et al. (2011).

Vários sensores de profundidade baseados em visão foram desenvolvidos e podem ser classificados em duas categorias: sensores de profundidade ativos e passivos. Os sensores de profundidade ativos incluem principalmente tempo de voo (em inglês, *time-of-flight* - TOF), luz estruturada, escaneadores a laser e assim por diante. As câmeras de profundidade atuais são de alto custo ou de baixa robustez. Além disso, os resultados são tipicamente muito mais esparsos do que as imagens e, portanto, perdem muitos detalhes de profundidade visíveis nas imagens (WANG et al., 2019).

Os sensores passivos podem ser categorizados como: monocular ou binocular. Em monocular, inferir profundidade de pedestres é um problema fundamentalmente mal definido devido à variação de altura do pedestre. Se todos os pedestres tivessem a mesma altura, não haveria ambiguidade. Portanto, o conhecimento prévio sobre o pedestre deve ser empregado, como aparência típica, layout e tamanho. Um método bastante comum é estimar a distância usando as relações entre o tamanho real e a quantidade de pixels do objeto (BAO; WANG, 2016) (ALI; HUSSEIN, 2016). Claramente este tipo de abordagem só pode ser usado em condições restritas que envolve objetos com a mesma dimensão real do objeto que foi utilizado para definir a relação entre o tamanho e a quantidade de pixels.

A visão binocular (ZHANG et al., 2017)(KELLER et al., 2011) estima a distância do alvo usando o princípio da disparidade. A disparidade define a distância entre dois pixels que correspondem ao mesmo ponto no mundo real. Quanto maior a disparidade mais perto o pixel está da câmera. As abordagens binoculares têm se mostrado mais adequadas para sistemas que funcionam *onboard* no veículo devido à grande quantidade de variabilidade de pedestres.

A etapa núcleo da visão binocular é a busca pelos pares de pixels correspondentes nas

duas imagens. Uma vez que a busca se baseia em valores de intensidade de pixel fornecido a partir de câmeras, diversos obstáculos reais impõem dificuldades para encontrar valores das disparidades corretos como listados a seguir:

- Diferença de iluminação: ocorre quando imagens estéreos diferem entre si quanto ao padrão de luminosidade em certas partes da cena, quando visualizados por cada câmera (HIRSCHMULLER; SCHARSTEIN, 2009). Como mostrado na Figura 4 (a) e (b), regiões nas quais incidiu mais luz em uma imagem do que na outra faz com o que o pixel mais similar entre os candidatos tenha valor de intensidade bastante diferente do pixel de referência dificultando encontrá-lo.

- Distorções radiométricas: são diferenças na configuração do brilho, do gamma e da abertura entre as câmeras, que faz com que o ponto real 3D seja representado nas duas imagens por valores de intensidade bastante diferentes, podendo conduzir a erros na busca dos pixels correspondentes. Na Figura 4 (e) tem-se um exemplo de distorção radiométrica na qual a imagem da esquerda possui mais brilho que a imagem da direita.

- Regiões de baixa textura e estruturas repetidas, como mostrados nas Figuras 4 (c) e (d), dificultam a tarefa de decidir sobre o pixel mais similar pois vários pixels candidatos diferentes têm o mesmo valor de intensidade e logicamente terão os mesmos custos de correspondência.

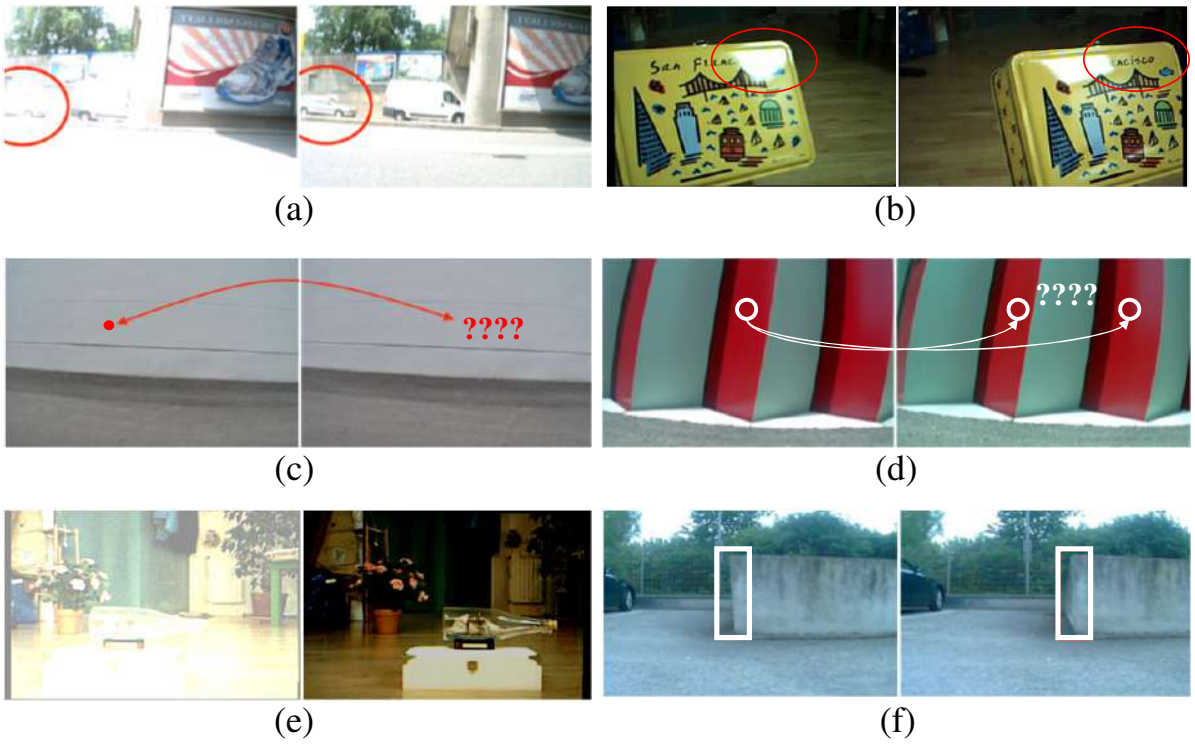
- Regiões de oclusão: são regiões que são visíveis apenas em uma câmera. Na Figura 4 (f) é possível observar que uma região do muro é apenas visível pela câmera da direita. Em regiões de oclusão não é possível encontrar os pares de pixels correspondentes e, consequentemente, são fontes de erros de disparidades.

- Imagens mal alinhadas podem comprometer todo o processo de correspondência estéreo se a abordagem adotada não for capaz de lidar com imagens desalinhadas.

Os algoritmos de correspondência estéreo podem ser classificados como densos ou esparsos. Os densos efetuam a correspondência para todos os pixels da imagem e a busca pela disparidade correta é realizada sempre em todo intervalo de disparidade definido. Por outro lado, os esparsos se destinam a encontrar a disparidade apenas analisando alguns pixels ou região de interesse e pode envolver a busca em um intervalo de disparidade reduzido. A ideia central das abordagens esparsas é reduzir o espaço de busca e custos de armazenamento e consequentemente complexidade computacional.

Por outro lado, as abordagens densas são mais úteis pois muitas informações interessantes para o sistema de previsão de colisão de pedestre podem ser descobertas a partir de imagens contendo o valor de disparidade para todos os pixels (MICLEA; NEDEVSCHI, 2019).

Figura 4 – Exemplo de situações que dificultam encontrar corretamente os pixels correspondentes. (a) Variação de iluminação entre os *frames*, (b) Reflexo da luz em posições diferentes, (c) Regiões de baixa textura, (d) Estrutura repetida, (e) Objetos transparentes. (f) Regiões de oclusão



Fonte: O autor.

Entre os algoritmos considerados densos temos as abordagens locais e as globais. As abordagens locais buscam a disparidade correta usando janela de suporte de tamanho finito. Em geral, as abordagens locais definem mapas de disparidade bastante ruidosos, no sentido de que regiões de mesma profundidade real que deveriam ter o mesmo valor de disparidade, na verdade possuem valores de disparidades estimados bastante diferentes. O aumento do tamanho da janela de suporte permite reduzir tais ruídos, mas ocasiona novos erros de disparidade em regiões de descontinuidade de profundidade, tal como bordas e declives acentuados. Em situações envolvendo pedestres distantes que se apresentam na imagem com pequenas dimensões, as abordagens locais com grandes janelas certamente não conseguirão estimar o valor de disparidade correta. Por outro lado, as abordagens globais e semi-globais conseguem lidar com tais questões pois a função de custo de dissimilaridade destas abordagens é tratada como um problema de minimização de uma função de custo global. Contudo a forte dependência de dados torna difícil sua implementação em processadores paralelos e alcançar taxa de processamento em tempo real.

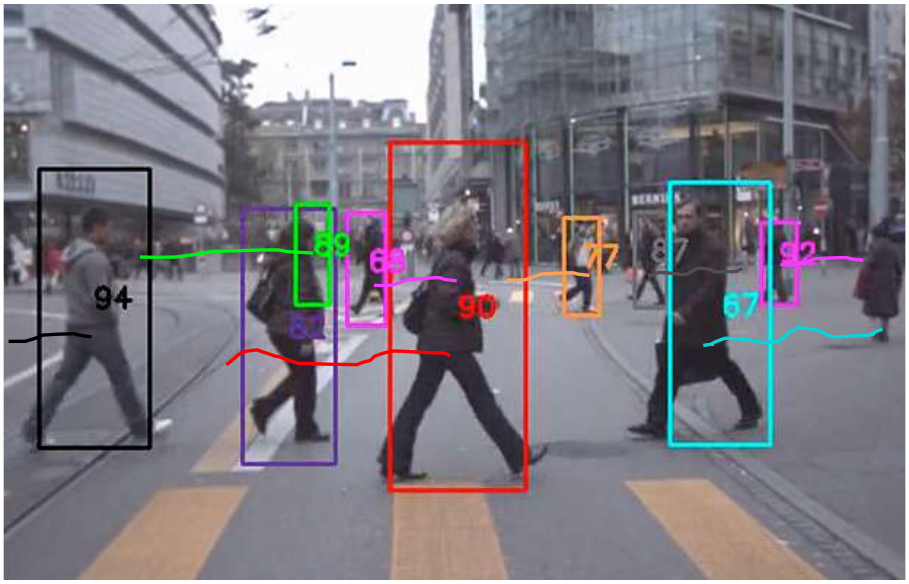
Além do cálculo de disparidade, uma etapa posterior, a de estimativa da distância, requer estratégias para estimar a distância lateral e longitudinal do pedestre. Abordagens de segmentação de imagens são úteis para separar a região de pedestre que chamamos de primeiro plano (em inglês, *foreground*) do restante da região que chamamos de segundo plano

(em inglês, *background*). Segmentação baseada em aprendizagem profunda demanda um alto custo computacional (BREHAR et al., 2018). Dicas úteis (NGUYEN; ROTTENSTEINER; HEIPKE, 2019) (MA; ZHU, 2022) (KELLY et al., 2006) podem ser obtidas a partir dos mapas de disparidades denso para a realização da segmentação poupando esforço computacional para um processamento completo de segmentação.

2.1.3 Rastreamento

Informações úteis sobre a dinâmica de movimentação dos pedestres tais como velocidade e direção de movimentação podem ser obtidas se o pedestre for rastreado ao longo do tempo. Para esta finalidade, a tarefa de rastreamento de objetos é fundamental. Ela é definida como o processo de observar objetos de interesse no momento atual e corresponder a objetos presentes em momentos anteriores. O rastreamento cria um identificador exclusivo para cada uma das detecções e, em seguida, rastreia cada um conforme eles se movem ao longo dos *frames*, mantendo a atribuição do identificador como mostrado na Figura 5.

Figura 5 – Rastreamento no contexto de pedestres.



Fonte: O autor.

O rastreamento fundamenta tarefas de alto nível, como estimativa de pose, reconhecimento de ação e análise de comportamento (LUO et al., 2021). Existem diversos desafios para o desenvolvimento de sistemas de rastreamento de pedestres robustos tais como (SMEULDERS et al., 2013)(HELD; THRUN; SAVARESE, 2016):

- Movimento de pedestres: Pedestres podem se movimentar de maneira não linear e imprevisível, resultando em rastreamento difícil. Por exemplo, em cenários do mundo real, pedestres mudam facilmente e inesperadamente o seu padrão de movimentação entre andar, correr, parar ou virar (SCHNEIDER; GAVRILA, 2013).

- Oclusão: Acontece quando um objeto rastreado ou seus principais atributos usados para reconhecer sua identidade não estão disponíveis no *frame* atual para que o rastreador possa continuar acompanhando seu estado enquanto o objeto ainda está presente na cena (LEE et al., 2014). Dependendo da técnica de rastreamento, quando dois ou mais pedestres rastreados ocluem, é possível que um rastro possa ser perdido temporariamente. O algoritmo de rastreamento deve ser capaz de levar isso em conta para que, após a oclusão, cada rastro mantenha o objeto apropriado (ou seja, aquele que era rastreado antes da oclusão).

- Bifurcação: Um rastreamento pode ser dividido em dois ou mais rastreios. Isso pode ser, por exemplo, devido a uma pessoa que depositou um objeto na cena.

- Novos rastros: Novos pedestres, que não foram rastreados anteriormente, devem ser reconhecidos como novo e não confundidos com pedestres rastreados anteriormente.

- Rastros perdidos: Um rastro pode ser perdido completamente, devido a ruídos, ou o pedestre saindo do campo de visão da câmera. Este rastro perdido deve ser marcado como inativo e não ser confundido com outros rastros.

Várias técnicas e algoritmos foram propostos para tentar resolver tais desafios. A maioria das abordagens segue uma estratégia de Rastreamento Baseado em Detecção (também chamada de Rastreamento por Detecção) no qual os *frames* são dados a um detector de objeto pré-treinado que fornece as observações que, por sua vez, são usadas pelo rastreador para conectar com observações vistas em momentos passados para formar as trajetórias (WANG et al., 2020).

As abordagens também são categorizadas em paradigma *on-line* ou *off-line*. A diferença está no fato de que as observações de *frames* futuros são ou não utilizadas ao manipular o *frame* atual. Os métodos de rastreamento *on-line* contam apenas com as informações passadas disponíveis até o *frame* atual, enquanto as abordagens *off-line*, também chamado de rastreamento baseado em lote, empregam observações no passado e no futuro.

Os métodos *off-line* são geralmente mais precisos (REZATOFIGHI et al., 2015)(KIM et al., 2015), pois as observações no futuro e no passado estão disponíveis para desambiguação. Dentro deste paradigma, trajetórias de objetos são normalmente encontradas em um problema de otimização global que processa lotes de vídeo inteiros de uma só vez. No entanto, devido ao processamento em lote, esses métodos não são aplicáveis em cenários de previsão de colisão em que um alvo apenas está disponível em cada etapa de tempo atual.

O problema de rastreamento pode ser visto como um problema de associação de dados onde o objetivo é associar detecções entre *frames* em uma sequência de vídeo. Os métodos mais tradicionais de associação de dados são o Rastreamento de Múltiplas Hipóteses

(em inglês, *Multiple Hypothesis Tracking* - MHT) (REID, 1979) e o Filtro de Associação de Dados Probabilístico Conjunto (em inglês, *Joint Probabilistic Data Association Filter* - JPDAF) (FORTMANN; BAR-SHALOM; SCHEFFE, 1983). Ambos os métodos foram revisitados em um cenário de rastreamento por detecção (REZATOFIGHI et al., 2015)(KIM et al., 2015) e mostraram resultados promissores. No entanto, o ganho em precisão desses métodos vem com o aumento da complexidade computacional.

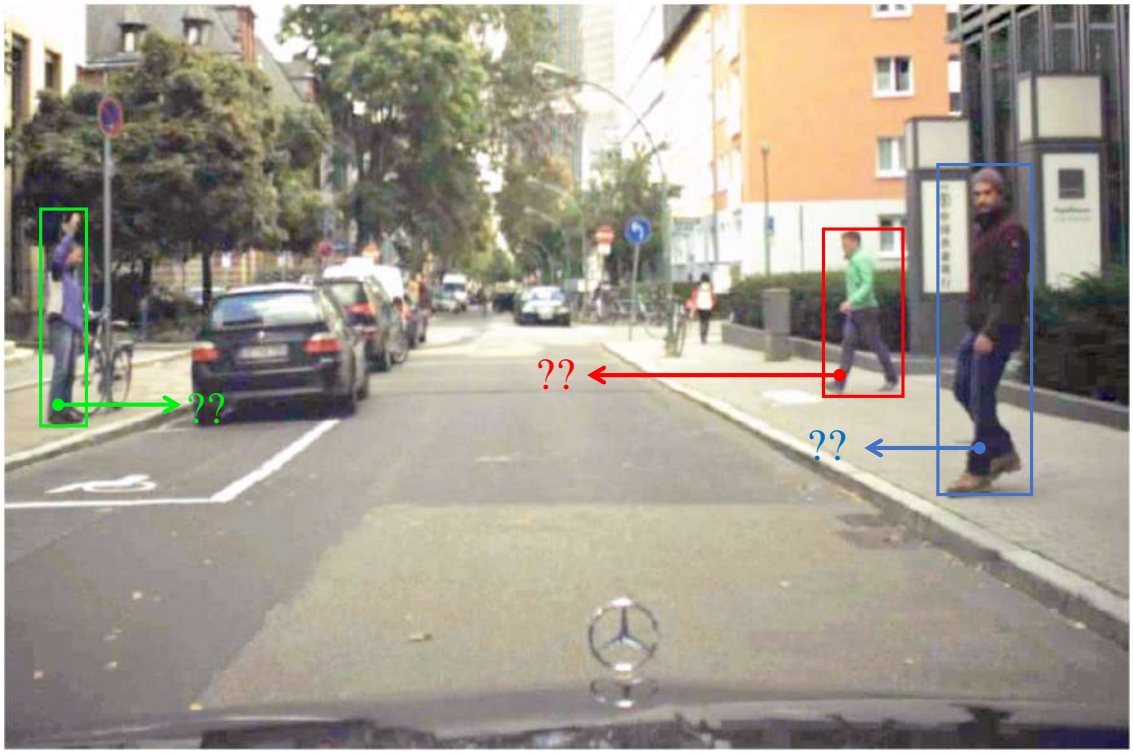
Para auxiliar no processo de associação de dados, os rastreadores usam vários métodos para modelar geometria e dinâmica do movimento (YOON et al., 2015) (BEWLEY et al., 2016). O modelo de movimento permite prever a posição potencial dos objetos nos *frames* futuros, reduzindo assim o espaço de busca. Na maioria dos casos, supõe-se que os objetos se movam suavemente no mundo real e, portanto, no espaço de imagem. No entanto, o modelo de movimento sozinho pode falhar em cenários onde os pedestres realizam mudanças abruptas de direção e velocidade ou quando estão ocluídos. Para contornar esta deficiência, outras abordagens utilizam modelos de aparência de pedestres juntamente com geometria e dinâmica (WOJKE; BEWLEY; PAULUS, 2017). Contudo, calcular dados de aparência e realizar as correspondências trazem um alto custo computacional (LUO et al., 2021). Outra maneira de mitigar o problema de mudança abrupta é de fato termos soluções que processem em altas taxas de *frames* porque o sistema conseguiria capturar essas mudanças abruptas em menos tempo.

2.1.4 Previsão de Trajetória

O início de uma manobra de emergência do veículo requer uma estimativa não apenas da atual, mas também da posição futura do pedestre em relação ao veículo como mostrado na Figura 6. Os seres humanos conseguem compreender diversas situações complexas de trânsito e prever com facilidade o comportamento de outros participantes. No entanto, desenvolver sistemas de previsão é bastante complicado devido à natureza altamente dinâmica do movimento de pedestres que pode abruptamente mudar sua direção de caminhada para qualquer lugar.

O desafio de fazer previsões precisas do movimento humano surge da complexidade do comportamento humano e da variedade de seus estímulos internos e externos. O comportamento do movimento pode ser conduzido pela própria intenção, pela presença e ações dos agentes vizinhos, pelas relações sociais entre os agentes, pelas regras e normas sociais ou pelo ambiente com sua topologia, geometria, possibilidades e semânticas. A maioria dos fatores não é diretamente observável e precisa ser inferida a partir de pistas perceptivas ruidosas ou modelada a partir de informações de contexto. Além disso, para ser eficaz na prática, a previsão de movimento deve ser robusta e operar em tempo real. Em geral, o problema a ser resolvido de previsão de movimento contém os três elementos a seguir (RUDENKO et al., 2019):

Figura 6 – O pedestre irá cruzar? Os sistemas PCPs precisam raciocinar rapidamente sobre intenções e locais futuros dos pedestres.



Fonte: Kooij et al. (2019) (ADAPTADO).

- Estímulos: Os estímulos internos e externos que determinam o comportamento do movimento incluem a intenção de movimento dos agentes e outras influências observáveis direta ou indiretamente. A maioria dos métodos de previsão se baseia em trajetórias parciais observadas ou, geralmente, em sequencias de observações do estado do agente, como posições, velocidades, ângulos ou atributos de articulações corporais. Frequentemente, isso é fornecido por um sistema de rastreamento de objetos e é comum assumir a identidade correta do rastro durante o período de observação.

- Modelagem de movimento: as abordagens para predição de movimento humano diferem na maneira como representam, parametrizam, aprendem e resolvem a tarefa.

- Previsão: Diferentes métodos produzem diferentes formas paramétricas, não paramétricas ou estruturadas de previsões, como gaussianas sobre estados de agentes, distribuições de probabilidade sobre grades (em inglês, *grid*), amostras de trajetória única ou múltipla ou padrões de movimento usando modelos gráficos.

Muitas abordagens para a previsão geram movimentos futuros considerando um modelo de movimento feito à mão, dinâmico e explícito f baseado em leis de movimento de Newton. Uma forma comum para f é $\dot{s}(t) = f(s(t), a(t), t) + w(t)$ onde $a(t)$ é a entrada

de controle e $w(t)$ é o ruído do processo. De fato, a previsão de movimento pode ser vista como inferir $s(t)$ e $a(t)$ a partir de várias pistas estimadas ou observadas. Os modelos mais simples são cinemáticos que representam estados de movimento como posição, orientação, velocidade e aceleração sem considerar as forças que governam o movimento. Exemplos populares incluem o modelo de velocidade constante (em inglês, *constant velocity* - CV) (LEIGH et al., 2015)(BERTOZZI et al., 2004) que assume velocidade constante por partes com aceleração sendo modelado como ruído branco, o modelo de aceleração constante (em inglês, *constant acceleration* - CA) (BINELLI et al., 2005) que assume aceleração constante por partes com impulso sendo modelado como ruído branco, o modelo de curva constante (em inglês, *costant turning* - CT) que assume a taxa de rotação e a velocidade constantes e a aceleração de rotação modelado como ruído branco.

O movimento complexo do pedestre é pobremente descrito por um modelo único dinâmico f. Uma abordagem comum para modelar o movimento geral de alvos que mudam seu comportamento é a definição e fusão de diferentes modelos de movimento, cada um deles descrito por um regime dinâmico diferente f. Uma vez que os modos de movimento dos pedestres não são diretamente observáveis, nós precisamos de técnicas para representar e raciocinar sobre a incerteza no modo de movimento. Os métodos baseados em multi-modelos (MM) são as abordagens mais aplicadas para este fim. Nos métodos MM, supõe-se que um conjunto de modelos represente os possíveis padrões ou estruturas de comportamento do sistema que podem saltar em momentos desconhecidos. Um banco de filtros é executado sempre em paralelo, cada um com base em um modelo específico, para obter estimativas condicionadas ao modelo. Os saltos no modo de sistema podem ser modelados como alternâncias entre os modelos assumidos. Por simplicidade, geralmente se supõe que a transição entre os modos do sistema segue uma cadeia de Markov (RUSSELL; NORVIG, 2016). Finalmente, a estimativa geral é dada pela fusão das estimativas desses filtros. O desenvolvimento de um preditor de MM consiste nas seguintes etapas: projeto de um conjunto de modelos, seleção de filtros, desenvolvimento de estratégias de cooperação e estimativa da fusão das saídas. O termo cooperação significa quaisquer ações cooperativas tomadas entre os filtros para alcançar uma melhor precisão. Na etapa final, a fusão estimada pode ser alcançada por um procedimento baseado em decisão *hard* ou decisão *soft* (por exemplo, soma ponderada).

A observação do comportamento do movimento de pedestres pode ajudar a desenvolver sistemas para prever a trajetória futura utilizando métodos de multi-modelos. Alguns trabalhos analisam a maneira de andar dos seres humanos, especialmente em caminhada em estado estacionário do ponto de vista biomecânico (BRENIERE; DO, 1986). A velocidade como parâmetro principal da caminhada em estado estacionário também é avaliada em vários estudos sobre pedestres no ambiente de tráfego. No trabalho de Köhler et al. (2012) os autores mostram que a velocidade média de caminhada é determinada entre 1,36 e 1,52 metro por segundo (m/s), dependendo das condições específicas do local do

teste. Em cenários de tráfego, as transições de marcha constituem uma parte essencial do movimento de pedestres, principalmente determinando o início de uma caminhada e o término da caminhada. No entanto, ainda existem apenas alguns estudos sobre transições de movimento. Por exemplo, os autores Köhler et al. (2012) abordam um estudo do início de uma caminhada no qual concluem que as acelerações médias estão em torno de 1,4 metro por segundo ao quadrado (m/s^2) com um desvio padrão de 0,7 m/s^2. É relatado nesse trabalho que a velocidade no estado estacionário é alcançada dentro do segundo passo apenas 1,3 segundos em média após o movimento inicial, mostrando a grande agilidade dos pedestres. Uma análise do comportamento de parada no tráfego é apresentada no trabalho de Goldhammer et al. (2014b). Os autores determinaram que os pedestres têm a capacidade de parar muito rapidamente em 2 passos, respectivamente entre 1 e 2 segundos, mas geralmente usam fases de desaceleração muito mais longas, até 7 segundos, quando a parada rápida não é necessária. A velocidade de estado estacionário medida também é mais baixa do que em outros estudos, portanto, a intenção da ação futura planejada de parar parece ter uma influência precoce no movimento da pessoa.

Para permitir detectar mudanças trajetória e consequentemente aumentar o poder da tarefa de previsão de trajetória, uma grande variedade de características e informações podem ser extraídas dos pedestres. No trabalho de Yannis, Papadimitriou e Theofilatos (2013) os autores realizam estudos sobre algumas dicas úteis. Informações de posicionamento, velocidade, orientações e posições de cabeça e articulações determinam as variáveis que um motorista normalmente usa para inferir intenções e saber se os pedestres estão cientes dos veículos que se aproximam.

2.1.5 Avaliação de Risco

A avaliação do risco de colisão ou previsão de colisão entre o veículo e o pedestre deve ser realizada com base em uma métrica adequada. Várias métricas de risco foram propostas na literatura. Entre elas, a mais popular, o tempo de colisão (em inglês, *Time-To-Collision* - TTC) corresponde ao tempo restante até a colisão. Quanto menor o TTC, maior o risco de colisão.

Dadas as trajetórias futuras do veículo e do pedestre, a colisão pode ser estimada resolvendo-se as equações dos modelos de movimento correspondentes ao veículo e o pedestre e localizando o ponto de interseção entre duas trajetórias. Se o veículo e o obstáculo chegarem ao ponto de interseção ao mesmo tempo, uma colisão é detectada e o TTC é derivado de acordo. Como o veículo ou outros obstáculos têm um certo volume, eles não podem ser simplesmente tratados como um ponto, como assumido na física. Tendo em conta o tamanho e outros fatores, como a incerteza de posicionamento, uma solução mais comum é representar os veículos como polígonos, círculos ou elipses e uma colisão é prevista ao definir uma condição de sobreposição entre a forma do veículo e o pedestre. Em geral, as incertezas advêm do processo de percepção e do comportamento desconhecido

dos participantes do tráfego no intervalo de tempo de previsão. Ao considerar essas incertezas, os sistemas mais avançados calculam a ocorrência de uma colisão prevista de maneira probabilística e os valores de TTC tornam-se uma distribuição de probabilidade.

A métrica TTC não é adequada para avaliar completamente a criticidade das situações de tráfego, uma vez esta métrica não envolve diretamente as possíveis ações do motorista. Uma métrica mais adequada é o tempo de reação (HILLENBRAND; SPIEKER; KROSCHEL, 2006) (em inglês, *Time-To-React* - TTR) que é o tempo que o motorista do veículo leva para começar uma manobra para contornar uma colisão com o objeto. O TTR permite uma melhor incorporação de outras medidas tipicamente baseadas no tempo, como, por exemplo, tempo morto, tempos de reação e tempos de pré-aviso. A métrica TTR pode ser aproximada para lidar com um pequeno conjunto de manobras que cobrem diferentes tipos de restrições físicas tais como frenagem máxima, aceleração máxima, direção com raio mínimo para a esquerda e direção com raio mínimo para a direita. Os tempos de reação restantes para começar cada uma dessas manobras são indicados como tempo para frear (em inglês, *Time-To-Break* - TTB), tempo para reacelerar (em inglês, *Time-To-Kickdown* - TTK) e tempo para desviar (em inglês, *Time-To-Steer* - TTS). TTB indica o tempo restante no qual o motorista ainda pode evitar uma colisão ao frear com desaceleração máxima. TTS é o tempo que permite a frente do veículo passar pelas bordas do objeto realizando um arco circular. Mais detalhes sobre estas medidas podem ser encontrados no trabalho de Hillenbrand, Spieker e Kroschel (2006).

2.2 SISTEMAS COMPLETOS EXISTENTES

Vários sistemas de proteção a pedestres têm sido propostos envolvendo a etapa de previsão de colisão. A seguir, nós descrevemos e analisamos alguns destes sistemas.

No trabalho de Keller et al. (2011) os autores aplicam a técnica de janelas deslizantes em várias escalas e restrições de localidade de pedestres para geração da lista de ROIs. Na tarefa de classificação, os autores empregam a técnica HOG como extrator de características e SVM como classificador de pedestre. Os autores aplicam a abordagem de supressão não-máxima para remover as múltiplas respostas de um mesmo pedestre. A localização dos pedestres detectados é estimada a partir da combinação da técnica de visão estéreo densa, SGM, e uma função de massa de probabilidade (FLOHR; GAVRILA et al., 2013) que pondera cada valor de disparidade. As localizações são enviadas para um filtro de Kalman Kalman (1960) para fornecer dados de posição e velocidade do pedestre com menos ruído de medição. O movimento do pedestre é compensado pelo movimento do carro usando dados de velocidade e taxa de guinada a partir de sensores inerciais. O rastreamento é feito usando a abordagem de vizinho mais próximo global (em inglês, *global nearest neighbor* - GNN) com restrição retangular *a priori* nas posições dos objetos. Para a avaliação de risco os autores identificam possíveis colisões interceptando as posições futuras do veículo com as posições futuras de todos os objetos, inclusive de outros pedestres, dentro de um

intervalo de tempo. Se uma colisão for detectada, a abordagem inicia uma busca pela manobra, que pode ser uma frenagem ou um desvio. Os autores informam que a abordagem de detecção de pedestres envolvendo HOG, SVM, SGM, filtragem de ROIs e localização, possui uma taxa de processamento variável dependendo da complexidade do cenário e que no pior caso alcança uma taxa de 15 FPS, operando em uma resolução de imagem de 640 × 480 pixels (ou seja, resolução VGA). Somente a implementação em hardware do algoritmo de visão estéreo, nessa resolução, leva 17 milissegundos (ms), ou, em termos de taxa de processamento, 59 FPS. Uma análise importante feita pelos autores diz que a estimativa de velocidade correta de um pedestre se movendo a 2 m/s leva em torno de 20 *frames*. Então, somente com a detecção, o sistema levaria aproximadamente 1,3 segundos para estimar a velocidade correta do pedestre. Para agravar ainda mais o desempenho, essa taxa de processamento tenderá a diminuir se adotarmos câmeras com resolução maiores. De fato, sistemas de detecção de pedestres capazes de processar imagens em altas resoluções são mais atraentes uma vez que permitem encontrar mais informações úteis sobre o pedestre e o cenário a frente (SULEIMAN; SZE, 2016)(HELALI et al., 2020).

No trabalho de Lee, Shin e Kwon (2017) os autores propõem uma fusão de câmera com radar e LIDAR para obter informações sobre a classe, velocidade relativa, distância relativa do pedestre. Para cada sensor é realizado o rastreamento dos objetos detectados com o filtro de Kalman e um método de porteira (em inglês, *gating*). O modelo de movimento de velocidade constante é adotado para prever o estado do pedestre no FK. Os dados dos rastros gerados por cada sensor são fundidos para definir um valor de estado final do pedestre. Os autores não descrevem a abordagem de detecção de pedestre baseada em visão. Na avaliação de risco, os autores realizam a previsão de trajetória do veículo e do pedestre considerando taxa de guinada e um modelo de movimento aproximado de velocidade constante ao longo de uma linha reta. Uma previsão de trajetória com um horizonte de tempo de três segundos é realizada para estimativa de colisão. A distância para frenagem do veículo é projetada considerando a possibilidade de o pedestre acelerar ou desacelerar o passo bem como a distância de frenagem. O sistema de detecção de pedestre opera *frames* de resolução VGA a uma taxa de 11 FPS. A baixa taxa de processamento impede, por exemplo, do sistema descobrir em tempo hábil a velocidade correta do pedestre para poder o sistema realizar a previsão de colisão e tomada de decisão correta.

No trabalho de Llorca et al. (2011) os autores propõem um sistema de proteção ao pedestre no qual, na etapa de detecção de pedestres, eles integram a sua própria solução (ALONSO et al., 2007) que realiza uma etapa de extração de pontos de interesse 3D e visão estéreo esparsa. Em seguida, os autores aplicam um algoritmo de agrupamento subtrativo nesses pontos a fim de encontrar grupos que representam candidatos a pedestre. Depois eles aplicam uma estratégia de geração de multi-candidato, onde para cada grupo, vários candidatos são gerados deslocando suavemente o *bounding box* original nas duas dimensões. A partir das regiões candidatas geradas os autores aplicam uma abordagem de

classificação baseada em partes do corpo do pedestre. Para cada parte, os autores utilizam um extrator de características diferente e um classificador SVM especificamente treinado. Os resultados dos classificadores são somados para encontrar o valor da classificação final daquela região. A abordagem de rastreamento é realizada a partir de filtro de Kalman (1960) e o método húngaro (KUHN, 1955), que inclui a posição do pedestre e a velocidade relativa no vetor de estado do filtro e assume um modelo de movimento de velocidade constante. Em termos de desempenho, os autores afirmam que a detecção processa imagens com resolução de 320 × 240 pixels em 20 FPS. A baixa resolução associada com e baixa taxa de processamento, pode impedir, por exemplo, de obter informações precisas de localização e consequentemente de TTC. Certamente o desempenho de processamento do sistema será degradado ainda mais se for necessário aumentar a resolução de imagem.

No trabalho de Park et al. (2017) os autores propuseram um sistema de proteção ao pedestre no qual, no componente de sensoriamento, os autores combinam o sensor de câmera e radar para detecção de pedestre. Segundo os autores, a fusão pode melhorar a robustez do sistema de reconhecimento em situações em que de fato o desempenho do sensor de câmera é ruim: nevoeiro, queda de neve, chuva forte ou condições de luz do sol. Os dados de rastreamento de objetos a partir do radar são fundidos com os dados de rastreamento de pedestres a partir da câmera. Os autores utilizam um algoritmo tradicional de fusão de sensores que consiste em três funções: sincronização, fusão de rastros e gerenciamento de rastros (STREUBEL; YANG, 2016). Um dos fatores que impede de estimar rapidamente a velocidade do pedestre é a taxa de reconhecimento. Os autores afirmam que o tempo de reconhecimento do pedestre é de 100 milissegundos (ou seja, 10 FPS) processando *frames* de 1280 × 720 pixels. Além disso, o tempo das outras etapas do sistema é de 100 milissegundos aumentando ainda mais o tempo de todo o sistema para reagir.

No trabalho Wu, Zhou e Srikanthan (2016) os autores desenvolveram uma abordagem baseado em visão para detecção e rastreamento de qualquer tipo de obstáculos usando SGM. Diferentemente dos trabalhos existentes que focam apenas na detecção de veículos ou pedestres, o método de detecção de obstáculos proposto conta com o conceito de mapas *u-v* disparidade para detectar todos os obstáculos na cena. Eles tomam proveito de algumas dicas específicas que existem na estrada que podem ser visualizadas com estes mapas para diferenciar os obstáculos da região de estrada livre. Um Espaço de Interesse é definido para reduzir bastante o espaço de busca de obstáculos antes de empregar técnicas de rotulagem de componentes conectados adaptativos para separar o espaço de interesse em conjuntos de obstáculos na imagem de u-disparidade e retornar os seus *bounding boxes*. Esse trabalho é utilizado como base para um sistema de previsão de colisão apresentado na tese de Wu (2016). A estratégia é promissora pois permite a detecção de qualquer obstáculo com alguns processamentos simples. Contudo, por outro lado, não faz distinção do tipo de obstáculo podendo gerar muitos falsos positivos. Além disso, essa estratégia é fortemente dependente da precisão da visão estéreo e da existência de estrada. Em um

ambiente em que a estrada não é fácil de ser detectada, através do mapa u-v disparidade, certamente essa abordagem falhará. É por isso que os detectores de objeto baseados em aprendizagem de máquina são extramente importantes pois são mais independentes da configuração geométrica do ambiente. O desempenho de processamento apenas desta estratégia, sem o SGM, em uma resolução de 1242×375 pixels é de 21 FPS.

2.3 RESUMO E CONSIDERAÇÕES

Este capítulo apresentou as diversas etapas envolvidas em um sistema de previsão de colisão típico detalhando os desafios de se propor métodos precisos, robustos e rápidos em cada uma delas. Por fim, apresentamos alguns sistemas práticos para evitar colisão com o pedestre que utilizam os princípios do sistema de previsão de colisão. Nós observamos que o aspecto de desempenho de processamento do sistema e suporte a detecção de pedestres em grandes intervalos de distância não têm sido abordados nestes trabalhos. Além disso, a resolução de *frame* é uma característica para a qual também não tem sido dado importância nestes sistemas onde a percepção de mais informações úteis do cenário pode ajudar na tarefa de previsão de colisão. Uma vez que pedestres mudam rapidamente seu movimento e criam situações de colisão em fração de segundos, os sistemas PCPs precisam ser capazes de capturar o máximo de informações possíveis referentes a todas essas mudanças através dos *frames*. Para alcançar este feito, todos os componentes do sistema PCP ou aqueles que são intensivos de dados tem que ser desenvolvidos de maneira otimizada para aumentar a taxa de processamento do sistema processando *frames* em maiores resoluções. Dessa forma, nesta Tese, nós exploramos estes aspectos em busca de melhorias na efetividade do sistema PCP para evitar e mitigar colisões com o pedestre.

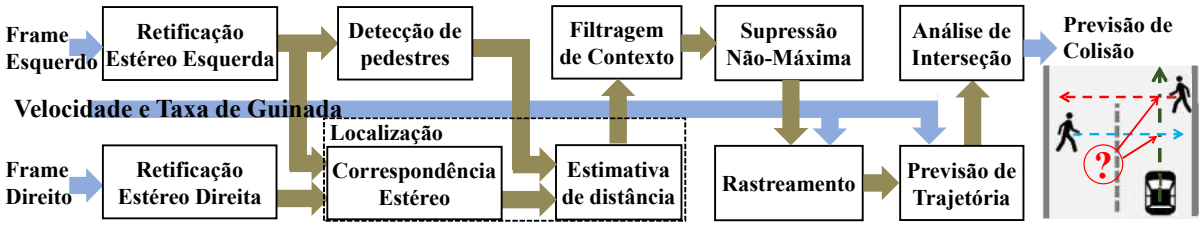
3 PREVISÃO DE COLISÃO DE PEDESTRES

Este capítulo descreve o sistema de previsão de colisão de pedestre (PCP) proposto, detalhando os algoritmos e técnicas adotadas em cada etapa. Este capítulo também descreve a estratégia proposta pra suportar a avaliação da etapa de previsão de colisão. Ambos, o sistema e a estratégia de avaliação, são usados como suporte para a demonstração das melhorias propostas em etapas específicas para a tarefa de previsão de colisão, tais como detecção e localização.

3.1 SISTEMA PCP

Este sistema compreende uma configuração de câmeras estéreo acoplado ao veículo para captura e processamento de pares de *frames* estéreo ao longo do seu trajeto. A taxa de captura depende da taxa de processamento de todo o sistema. Além disso, para cada par de *frames* coletado, o sistema também captura dados da movimentação do veículo como velocidade em metros por segundo (m/s) e a taxa de guinada em radianos por segundo (rad/s).

Figura 7 – Arquitetura geral do sistema PCP proposto.



Fonte: O autor.

Cada par de *frames* é processado primeiramente pela etapa de Retificação Estéreo para correção de distorções radiais e tangenciais, bem como para o alinhamento horizontal (HARTLEY; ZISSERMAN, 2003). Essa etapa é imprescindível para a localização do pedestre usando técnicas de correspondência estéreo. A etapa de Detecção de Pedestres encontra cada pedestre no *frame* atual retificado e destaca a presença do pedestre no *frame* através de *bounding boxes*. A etapa de Localização recebe o par de *frames* atual, todos os *bounding boxes* e estima suas distâncias.

A etapa de Filtragem de Contexto remove as detecções que não atendem às restrições de forma e localidade de pedestre no cenário urbano. A etapa de supressão não-máxima (em inglês, *non-maximum suppression* - NMS) remove várias detecções vizinhas do mesmo pedestre. A etapa de Rastreamento rotula as posições estimadas ao longo do tempo (ou discretamente, ao longo de vários *frames*) como pertencendo ao mesmo pedestre. A etapa de Previsão de Trajetória estima as trajetórias futuras dos pedestres e do veículo, a partir

de modelos de movimento e dados de velocidade e taxa de guinada, em um horizonte de tempo futuro discreto. Finalmente, a etapa de Análise de Interseção identifica possíveis posições de colisão quando as posições dos pedestres estiverem ao mesmo tempo no futuro tocando as posições futuras do veículo.

Nós implementamos este sistema PCP e utilizamos as diversas arquiteturas propostas para demonstrar as melhorias na capacidade de previsão de colisão com as soluções propostas para detecção e localização de pedestres. Nas seções seguintes, nós descrevemos as abordagens propostas para cada etapa deste sistema. Uma vez que a detecção de pedestres é uma das etapas mais críticas e o principal foco deste trabalho, esta etapa será mais detalhada nos Capítulos 4 e 5.

3.1.1 Localização

Esta etapa utiliza a correspondência estéreo para geração de mapas de disparidades denso e para cada *bounding box* vindo da etapa detecção de pedestres estima as suas distâncias laterais e longitudinais. A seguir são detalhadas a abordagem de correspondência estéreo adotada e a estratégia para estimativa de distância.

3.1.1.1 Correspondência Estéreo

Um sistema de correspondência estéreo típico (STANKIEWICZ; LAFRUIT; DOMAŃSKI, 2018) recebe dois *frames* que visualizam o mesmo ambiente no mesmo instante de tempo, mas tendo pontos de vistas ligeiramente diferentes. As câmeras que projetam esses *frames* têm um deslocamento horizontal pequeno para gerar essa diferença de ponto de vista. Qualquer ponto no mundo real 3D é representado por um par de pixels correspondentes de coordenadas diferentes no qual um pixel está localizado no *frame* esquerdo e o outro pixel no *frame* direito.

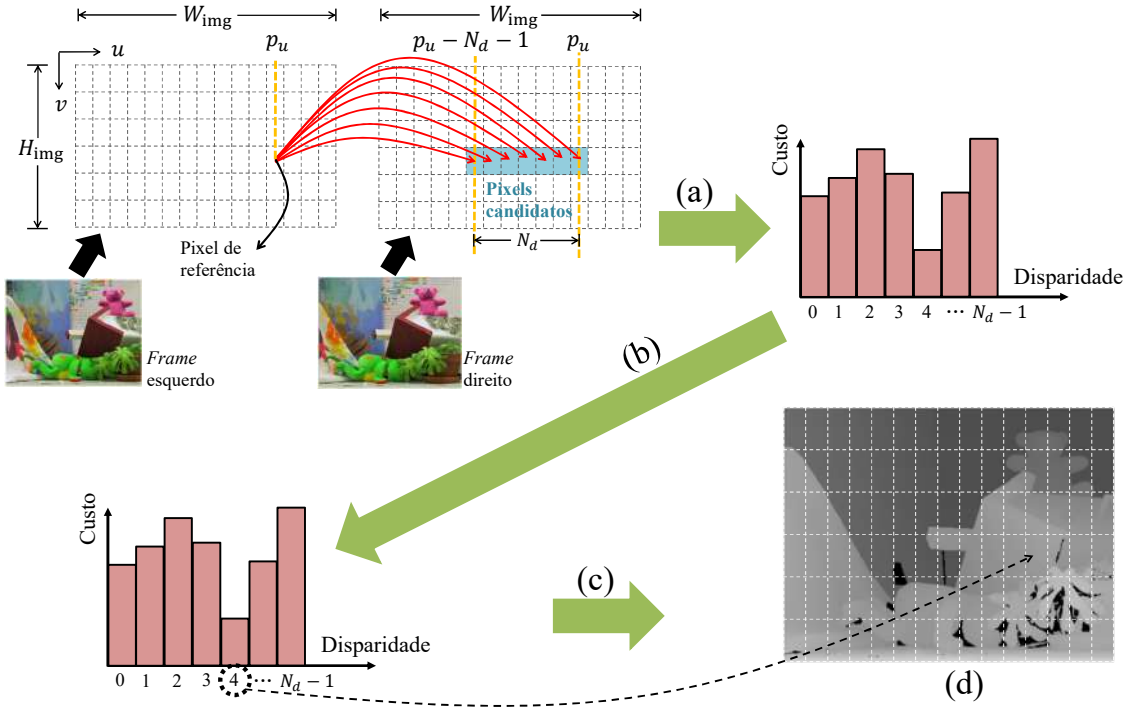
As técnicas de correspondência estéreo visam encontrar esses pixels correspondentes realizando uma busca horizontal e estimando um valor de custo de dissimilaridade entre cada par de pixels comparado, como mostrado na Figura 8.

A distância horizontal relativa entres pares de pixels correspondentes é chamada de disparidade. O intervalo de busca horizontal N_d é chamado de intervalo de disparidades. Quanto maior esse intervalo, maior é a possibilidade de encontrar objetos mais próximos da câmera, mas também maior é o custo computacional. O *frame* gerado que consiste em todos os valores de disparidades é chamado de mapa de disparidades.

A busca horizontal apenas é possível se o sistema de câmeras estiver calibrado e cada novo par de *frames* estéreo estiver perfeitamente alinhado horizontalmente e sem distorções de lentes. A etapa de retificação é responsável por corrigir os *frames* (HARTLEY; ZISSERMAN, 2003; HYUN; MOON, 2017).

A partir da disparidade $d(u, v)$ associada ao pixel de coordenada (u, v), é possível calcular sua profundidade $z(u, v)$, através do processo de triangulação (QUAN, 2010),

Figura 8 – Processo de busca pelo pixel mais similar: (a) Atribuição de custo de correspondência de pixel de referência para cada pixel candidato; (b) Busca pelo pixel candidato de menor custo; (c) Retorno do valor de disparidade do pixel de menor custo (mais similar); (d) Quando esse processo é repetido para todos os pixels do *frame*, obtém-se o mapa de disparidade. Os termos W, H, N_d, significam, respectivamente, largura, altura e o intervalo de disparidades.



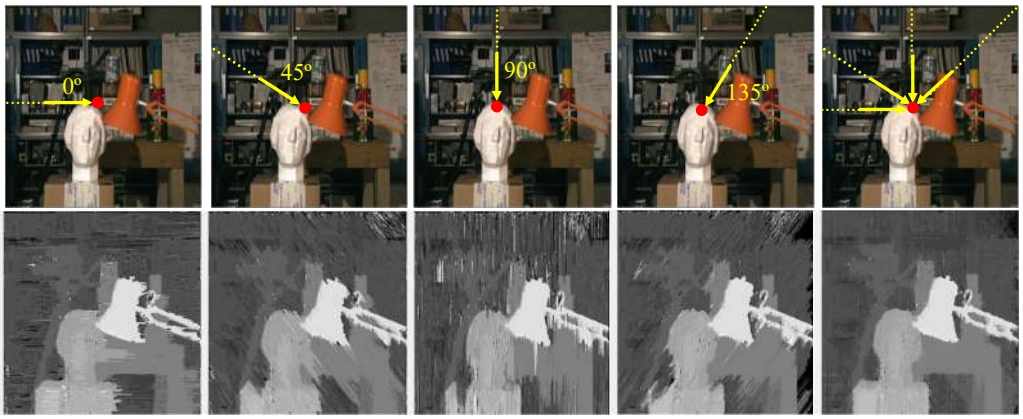
Fonte: O autor.

ou também chamado de transformação perspectiva, usando a Equação 3.1, onde f é a distância focal da câmera e b é a distância entre os centros ópticos das câmeras. Como pode ser constatado, quanto maior o valor de disparidade, ou seja, regiões mais claras no mapa de disparidade, mais próximo o objeto estará do sistema de câmeras estéreo.

$$z(u,v) = \frac{bf}{d(u,v)}, \text{para o pixel}(u,v) \tag{3.1}$$

A técnica que define o custo de dissimilaridade adotado neste trabalho é o *Stereo Global Matching* (SGM). Essa técnica usa uma combinação de técnicas locais para calcular os custos de dissimilaridades iniciais e uma abordagem de otimização de custos global que combina diferentes caminhos de propagação independentes ao longo do *frame*. A Figura 9 mostra a importância da propagação de custos e da combinação dessas propagações para a melhoria da precisão do mapa de disparidade. A capacidade de propagar os custos ao longo do *frame* permite incorporar informações de contexto nos custos finais de todos os pixels, consequentemente, gerando mapas de disparidade mais robustos para o contexto urbano, que em geral, possui os desafios descritos na Seção 2.1.2. Quanto maior a quantidade de caminhos de propagação e quanto mais simétricos são esses caminhos, maior a precisão alcançada pelo SGM.

Figura 9 – Resultado da otimização por caminhos individuais e a combinação de todos os caminhos da técnica SGM



Fonte: O autor.

Formalmente, a função geral da técnica SGM é definido da seguinte forma:

$$\begin{aligned} L_{\mathbf{r}^t}(\mathbf{p}, d) =& C(\mathbf{p}, d) + \min[L_{\mathbf{r}^t}(\mathbf{p} + \mathbf{r}^t, d), \\ & L_{\mathbf{r}^t}(\mathbf{p} + \mathbf{r}^t, d-1) + P_1, \\ & L_{r^t}(\mathbf{p} + \mathbf{r}^t, d+1) + P_1, \\ & \min_{i \in [0, N_d - 1]} L_{\mathbf{r}^t}(\mathbf{p} + \mathbf{r}^t, i) + P_2] - \min_{k \in [0, N_d - 1]} L_{\mathbf{r}^t}(\mathbf{p} + \mathbf{r}^t, k), \end{aligned} \tag{3.2}$$

onde $\mathbf{p} = (p_u, p_v)$ é a localização espacial do pixel na imagem de referência e d é o valor de disparidade. O termo do lado esquerdo da Equação, $L_{\mathbf{r}^t}(\mathbf{p}, d)$, representa as diversas funções de custos de caminho independentes para cada caminho de direção $\mathbf{r}^t = (r_u^t, r_v^t)$. A função é realizada ao longo de N_d pixels candidatos (ou seja, $d \in [0, N_d - 1]$).

Os termos do lado direito da Equação são custos de caminhos de um pixel anterior $\mathbf{p} + \mathbf{r}^t$ que são utilizados no cálculo do custo de caminho do pixel atual $L_{\mathbf{r}^t}(\mathbf{p}, d)$. A posição do pixel anterior é determinada pelo parâmetro $\mathbf{r}^t$ que está associado ao ângulo de reta t no qual a propagação de custo está sendo realizada. Alguns exemplos de $\mathbf{r}^t$ são: $\mathbf{r}^{0^\circ} = (-1, 0)$, $\mathbf{r}^{45^\circ} = (-1, -1)$, $\mathbf{r}^{90^\circ} = (0, -1)$, $\mathbf{r}^{135^\circ} = (1, -1)$.

O custo inicial $C(\mathbf{p}, d)$ é definido por abordagens locais que calculam o custo direto entre os pares de pixels como mostrado na Figura 8. Em abordagens locais, diversas soluções têm sido propostas para melhorar ainda mais precisão da técnica SGM (JIAO et al., 2014). Neste trabalho, propusemos uma solução que associa uma técnica de interpolação, chamada de diferença absoluta insensível a amostragem (BIRCHFIELD; TOMASI, 1998), e o filtro de gradiente de Sobel (KANOPOULOS; VASANTHAVADA; BAKER, 1988) na direção u do eixo de coordenadas do *frame* (CAMBUIM et al., 2019). Essa associação permite lidar com problemas apresentados na Seção 2.1.2 que dificultam encontrar os pixels correspondentes corretamente, como por exemplo, diferenças de iluminação na cena.

Os parâmetros P_1 e P_2 se referem a penalidades, para, respectivamente, pequenas e grandes mudanças de disparidades. P_1 é sempre menor que P_2. O uso das duas penalidades faz com que a otimização lide ao mesmo tempo com superfícies suaves e inclinadas (devido a P_1) e preserve descontinuidades de profundidade na cena (devido a P_2). O termo de subtração no final da equação evita que o custo de caminho aumente permanentemente ao longo do caminho, podendo conduzi-lo a valores infinitamente altos como observado por Hirschmüller (HIRSCHMULLER, 2008).

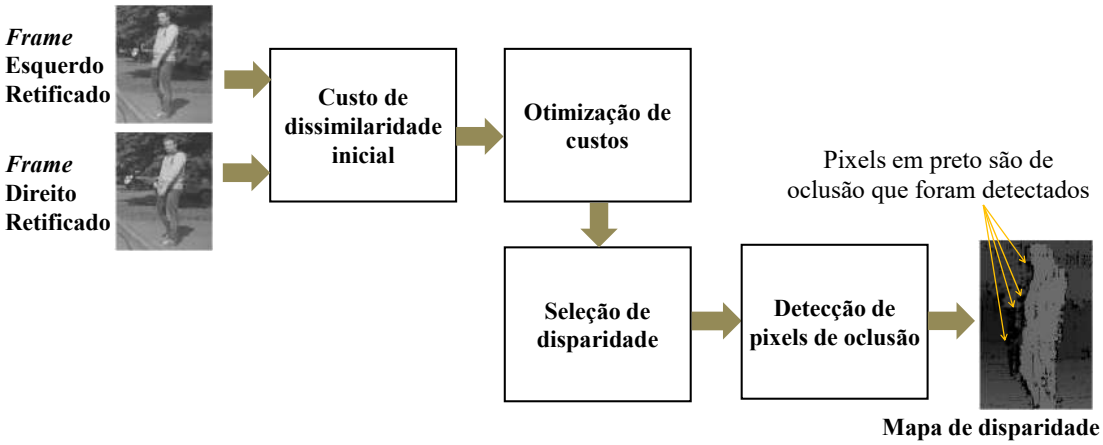
$$S(\mathbf{p}, d) = \sum_{t \in S_t} L_{\mathbf{r}^t}(\mathbf{p}, d) \tag{3.3}$$

O custo de dissimilaridade final otimizado em todo o *frame* é alcançado de acordo com a Equação 3.3. O termo S_t define o conjunto de direções das funções de custo de caminho envolvidas, como por exemplo, $S_t = \{0°, 45°, 90°, 125°\}$. Esta equação acumula os custos a partir de todos os custos de caminho $L_{\mathbf{r}^t}(\mathbf{p}, d)$ de direções diferentes em um único custo final S para todos os valores de disparidade $d \in [0, N_d - 1]$.

$$D_L(\mathbf{p}) = \underset{d \in [0, N_d - 1]}{\operatorname{argmin}} \; S(\mathbf{p}, d) \tag{3.4}$$

A partir da matriz de custos de dissimilaridade S, a seleção de disparidade constrói o mapa de disparidade resultante D_L. Essa seleção é realizada por meio da abordagem vencedor leva tudo (em inglês, *winner-takes-all*), conforme demonstrado na Equação 3.4. De acordo com esta equação, para cada pixel $\mathbf{p}$ a disparidade d de menor custo $S(\mathbf{p}, d)$ é escolhida para ser atribuída a $D_L(\mathbf{p})$.

Figura 10 – Etapas da correspondência semi global (SGM).



Fonte: O autor.

As etapas do SGM são mostradas na Figura 10. Além das etapas de custo inicial, otimização de custos e seleção de disparidade, nós temos a etapa de detecção de pixels

de oclusão. Tais pixels são apenas visíveis em uma câmera e, portanto, seus valores de disparidades não são possíveis de serem calculados. Por este motivo, pixels de oclusão conduzem a cálculos de disparidades errados pelas abordagens de correspondência estéreo, como descritos na Seção 2.1.2, que prejudicam a qualidade do mapa de disparidade e consequentemente reduz a precisão na estimativa de distância do pedestre.

Portanto, técnicas de detecção de pixels de oclusão são necessários. Neste trabalho, nós utilizamos a técnica denominada Checagem L/R, similar à técnica descrita em Hirschmuller (2008). Esta técnica cria, primeiramente, um segundo mapa de disparidade, indicado por D_R, a partir da matriz de custos de dissimilaridade, S, conforme a Equação 3.5.

$$D_R(\mathbf{p}) = D_R(p_u, p_v) = \underset{d \in [0, N_d - 1]}{\operatorname{argmin}} S(p_u, p_v - d, d) \tag{3.5}$$

Em seguida, verifica-se, para cada posição no mapa D_L, se a diferença do valor da disparidade para o valor da disparidade no mapa D_R é maior que um limiar fixo σ_{SGM}, chamado de limiar de oclusão, conforme demonstrado na Equação 3.6. Se sim, rotula-se esta disparidade como sendo de oclusão. A Figura 10 ilustra estes pixels detectados e invalidados através da Checagem L/R. O parâmetro de limiar de oclusão define a sensibilidade para detectar as disparidades erradas. Quanto menor esse parâmetro mais pixels são detectados como oclusão.

$$D(\mathbf{p}) = \begin{cases} D_{\text{inv}}, & \text{se } (D_L(\mathbf{p}) - D_R(\mathbf{p})) > \sigma_{\text{SGM}} \\ D_L(\mathbf{p}), & \text{caso contrário} \end{cases} \tag{3.6}$$

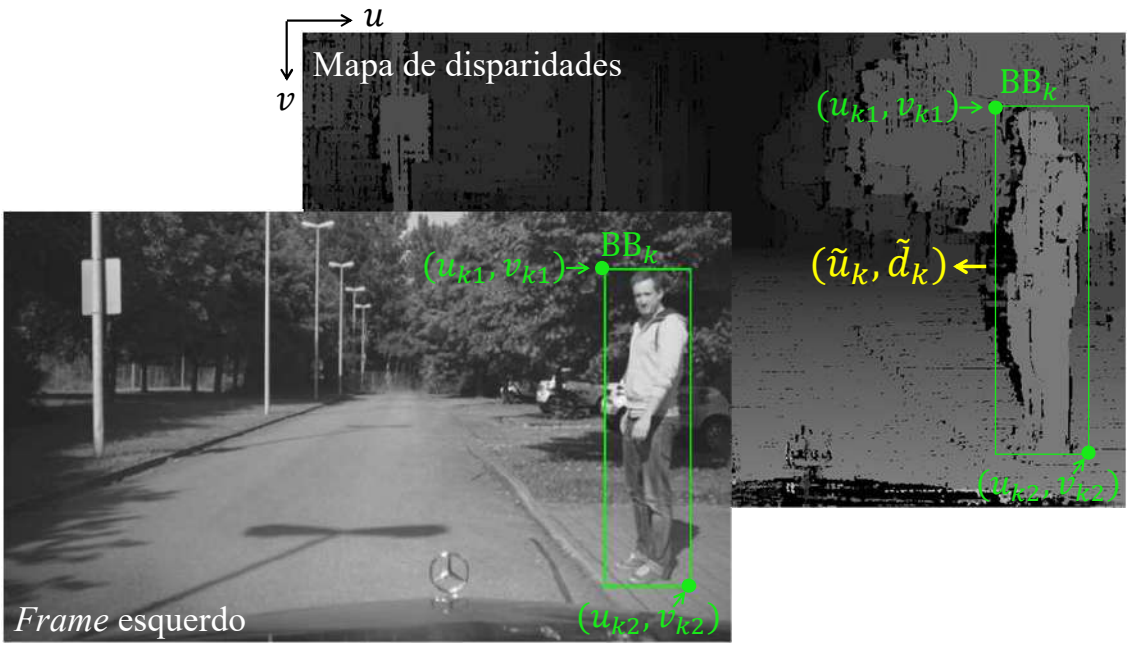
Como já mencionado, quanto mais caminhos de propagação mais preciso se torna o mapa de disparidade. Porém, a quantidade de caminhos impacta diretamente na complexidade da técnica SGM podendo inviabilizar soluções de alto desempenho. Além disso, técnicas mais robustas para a etapa de custo inicial e detecção de pixels de oclusão, também adicionam complexidade a abordagem de correspondência estéreo. Mais detalhes sobre a técnica SGM e melhorias propostas podem ser encontrados em Hirschmuller (2008), Cambuim et al. (2019) e Hernandez-Juarez et al. (2016).

3.1.1.2 Estimativa de Distância

A etapa de estimativa de distância recebe o mapa de disparidades, resultante da etapa de correspondência estéreo, e todos os *bounding boxes*, resultante da etapa de detecção de pedestres, calculados a partir do *frame* atual. Para cada *bounding box*, esta etapa estima a sua distância lateral e longitudinal, no espaço de imagem. Como mostrada na Figura 11, a partir de um dado *bounding box* BB_k, de índice k, cujas extremidades do retângulo são definidas pelas coordenadas (u_{k1}, v_{k1}) e (u_{k2}, v_{k2}), esta etapa calcula a sua posição lateral $\tilde{u}_k$, e sua posição longitudinal, $\tilde{d}_k$, que é um valor de disparidade.

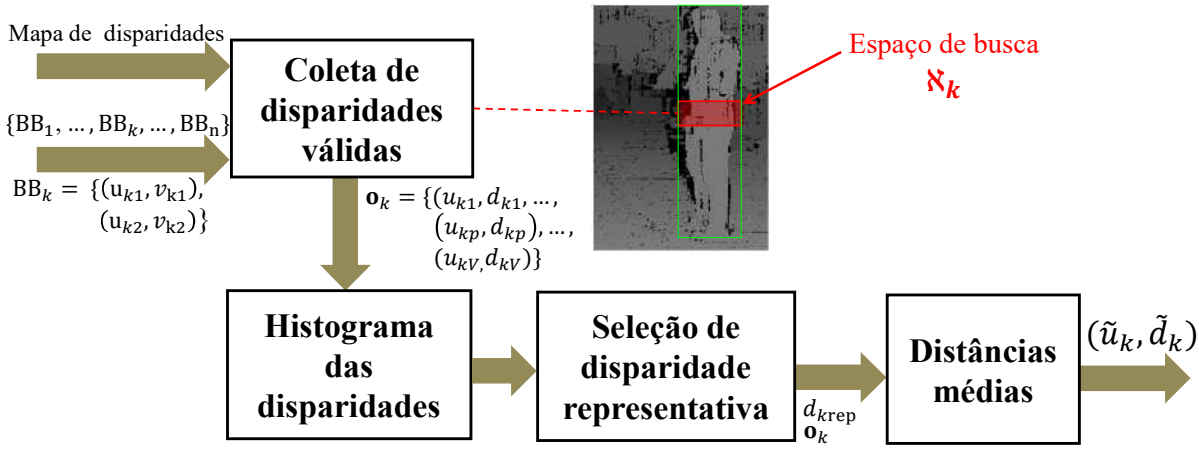
A estimativa de distância em cada *bounding box* é obtida em quatro passos, como mostrado na Figura 12: (1) filtragem de pixels ocluídos, (2) histograma das disparidades, (3) seleção da disparidade representativa e (4) distância média.

Figura 11 – Visão geral da estimativa de distância do pedestre e das variáveis envolvidas nesta etapa. O termo BB significa *Bounding Box*



Fonte: O autor.

Figura 12 – Abordagem para estimativa da distância do pedestre no espaço de imagem.



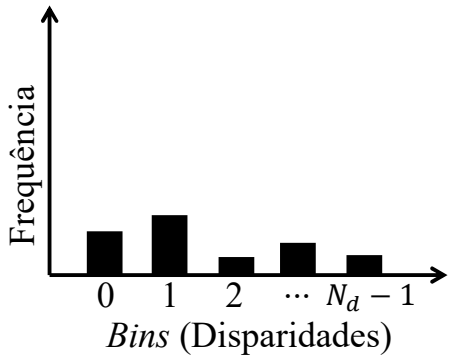
Fonte: O autor.

No primeiro passo, é realizada a coleta das disparidades válidas, ou seja, aquelas que possuem o valor diferente de D_{inv}, a partir da Equação 3.6. Esta coleta é realizada dentro de um espaço de busca de região retangular $\aleph$, centrado no meio do *bounding box* e é retornado um conjunto de V pares (u, d), contendo a coordenada lateral u e o valor de disparidade d correspondente. Os parâmetros de altura e largura do espaço de busca são definidos respectivamente, como H_{dist} e W_{dist}, limitados a dimensão do respectivo *bounding box* que está sendo processado.

Em seguida, um histograma das disparidades é calculado, a partir desse conjunto

de disparidades válidas, como mostrado na Figura 13. Cada posição na horizontal do histograma é chamada de *bin* e representa cada valor de disparidade possível. Em outras palavras, o total de *bins* do histograma é o intervalo de disparidades N_d definido pela abordagem de correspondência estéreo. Na vertical, tem-se a quantidade de vezes em que a respectiva disparidade foi contabilizada.

Figura 13 – Cálculo de histograma de disparidades a partir das disparidades válidas.



Fonte: O autor.

No terceiro passo, é realizada a seleção pela disparidade representativa $d_{k\text{rep}}$ referente ao um dado BB_k. Uma vez que o detector apenas encontra pedestres que estão totalmente visíveis, sem nenhum obstáculo a frente, então nós consideramos a disparidade representativa como sendo o maior valor de disparidade. Então é realizada a busca pelo *bin* de maior índice no histograma cujo valor de frequência é maior que uma quantidade mínima M_k definida por:

$$M_k = V_k \, \sigma_{\text{dist}}, \tag{3.7}$$

onde V_k é a quantidade de disparidades contabilizadas válidas do *bounding box* de índice k e σ_{dist} é um parâmetro com valores entre 0 e 1 que indica a porcentagem dessa quantidade que é considerada ser o mínimo para escolha de uma dada disparidade. A ideia de usar a validação por histograma é aumentar a robustez da estimativa da disparidade representativa na presença de disparidades estimadas erradamente no mapa de disparidade. No Capítulo 6 nós demonstramos que o uso do histograma permitiu reduzir a média e o desvio padrão do erro da estimativa das distâncias.

No quarto e último passo, as distâncias laterais e longitudinais do pedestre são estimadas através da média das disparidades e das distâncias laterais cuja diferença absoluta das disparidades válidas para a disparidade representativa $d_{k\text{rep}}$ é menor do que um dado limiar l_{dist}.

3.1.2 Filtragem de Contexto

Duas abordagens de filtragem são propostas baseadas no contexto de estrada e pedestres: a filtragem geométrica e filtragem por segmentação de estrada.

3.1.2.1 Filtragem geométrica

A Filtragem Geométrica considera restrições de localidade na estrada e restrição de dimensões de pedestres para definir se um determinado *bounding box* é valido ou não. Para a aplicação das restrições é necessário que conheçamos a altura e largura do *bounding box* no espaço de coordenadas do veículo. Um detalhamento sobre os espaços de coordenadas e transformações pode ser encontrado no Apêndice E. Para as duas localizações (u_{k1}, v_{k1}, d_k) e (u_{k2}, v_{k2}, d_k), referentes aos extremos do *bounding box* de índice k no espaço de coordenadas da imagem junto com a sua informação de disparidade d_k (ou seja, a distância longitudinal), nós calculamos os seus dois pontos extremos no espaço de coordenadas do veículo (X_{k1}, Y_{k1}, Z_k) e (X_{k2}, Y_{k2}, Z_k). Esse cálculo envolve transformação perspectiva de pixels (HARTLEY; ZISSERMAN, 2003) no espaço de imagem em pontos no espaço de câmera, seguido de uma transformação linear de pontos do espaço de câmera para pontos no espaço do veículo. Essas transformações estão detalhadas no apêndice E.

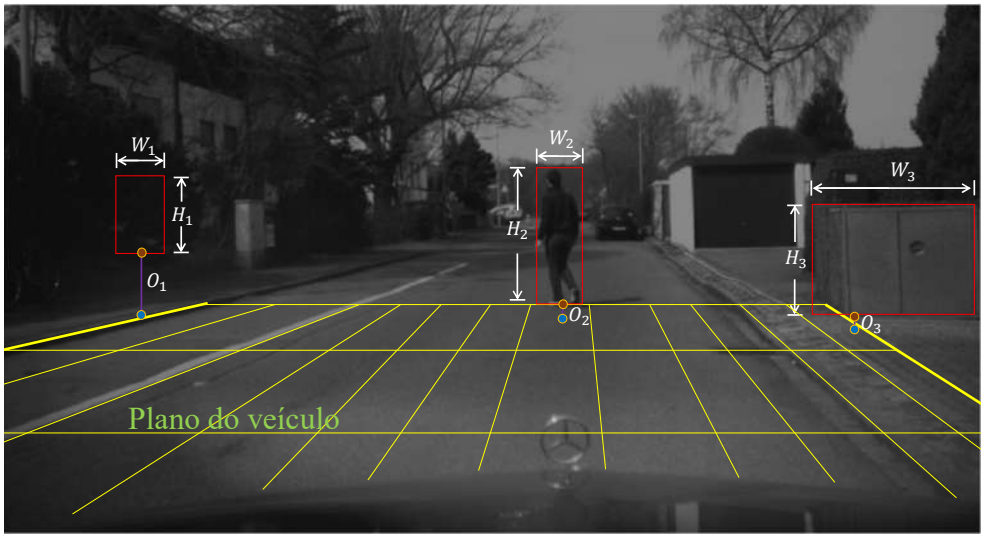
A partir dos pontos extremos no espaço de veículo, nós calculamos a altura H_k, largura W_k, e deslocamento vertical em relação a estrada O_k em unidades de metros de um dado *bounding box* de índice k usando as Equações 3.8, 3.9 e 3.10, respectivamente. A Figura 14 apresenta os dados de altura, largura e deslocamento vertical para cada *bounding box*.

$$H_k = |Y_{k1} - Y_{k2}| \quad \text{(metros)} \tag{3.8}$$

$$W_k = |X_{k1} - X_{k2}| \quad \text{(metros)} \tag{3.9}$$

$$O_k = \min(Y_{k1}, Y_{k2}) \quad \text{(metros)} \tag{3.10}$$

Figura 14 – Demonstração dos dados de largura, altura e deslocamento vertical de cada *bounding box* no espaço de coordenadas do veículo e o plano do veículo.



Fonte: O autor.

Com as informações de altura, largura e deslocamento, nós podemos aplicar as restrições geométricas usando a seguinte equação:

$$D(H_k, W_k, O_k) = \begin{cases} 1, & \text{se } (H_{\min} < H_k \leq H_{\max} \\ & \wedge\ W_{\min} < W_k \leq W_{\max}\ \wedge \\ & O_{\min} < O_k < O_{\max}) \\ 0, & \text{Caso contrário} \end{cases}, \tag{3.11}$$

onde $H_{\min}$ e $H_{\max}$ significam respectivamente a altura mínima e máxima do pedestre em metros, $W_{\min}$ e $W_{\max}$ significam respectivamente a largura mínima e máxima do pedestre em metros e $O_{\min}$ e $O_{\max}$ significam respectivamente o deslocamento vertical mínimo e máximo do pedestre em relação ao plano do veículo em metros. A ilustração do plano do veículo é apresentada na Figura 14. Como pode ser notado, este plano é o próprio plano da estrada.

A primeira, segunda e terceira clausula da Equação 3.11 verifica, respectivamente, se a altura, largura e deslocamento vertical do *bounding box* em coordenadas do veículo estão dentro de um determinado intervalo aceitável.

Figura 15 – Exemplo da aplicação da filtragem geométrica: (a) Resultado de detecção sem a filtragem e (b) com a filtragem.

Fonte: O autor.

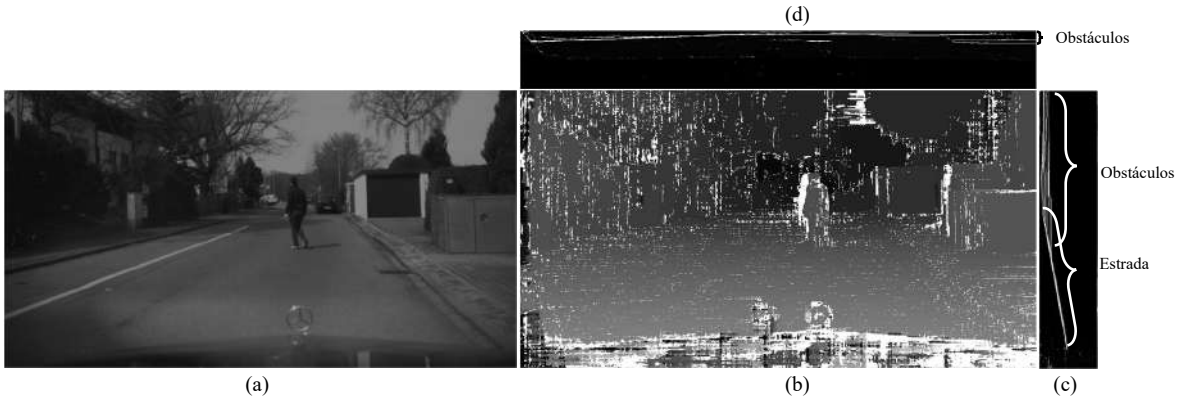
Uma detecção é considerada válida se a Equação 3.11 retornar valor 1. Caso contrário, o *bounding box* é descartado. Um exemplo da aplicação da filtragem geométrica é mostrado na Figura 15. É possível perceber a possibilidade de redução de falsos positivos com o uso dessa etapa.

3.1.2.2 Filtragem por segmentação de estrada

Uma segunda filtragem é realizada através da segmentação da estrada. A segmentação adotada neste trabalho segue uma estratégia de modelagem do perfil da estrada através

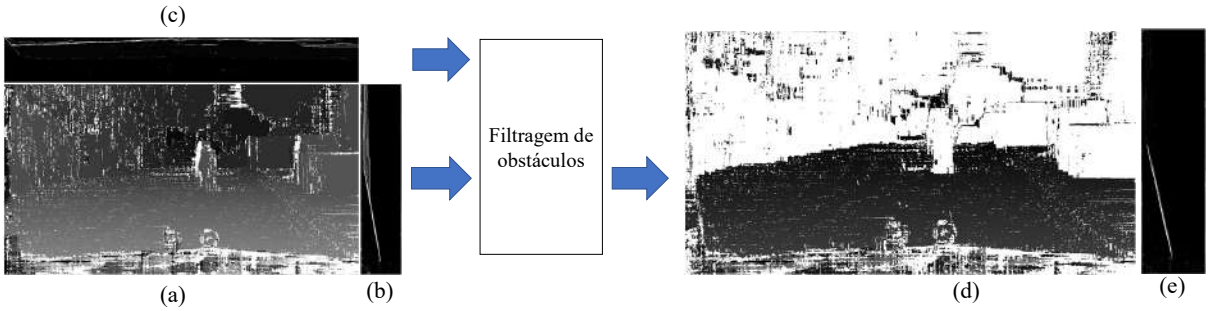
do mapa de disparidade. Essa modelagem utiliza o conceito de mapa v-disparidade que é uma imagem obtida através do cálculo de histograma de cada linha do mapa de disparidade de entrada. A Figura 16 ilustra esse conceito. Esse mapa v-disparidade revela o perfil longitudinal da estrada. Além disso, um histograma relacionado a cada coluna do mapa de disparidade forma o mapa u-disparidade, que destaca os obstáculos. Os obstáculos aparecem como uma linha vertical no mapa v-disparidade e horizontal no mapa u-disparidade.

Figura 16 – Ilustração do conceito de mapa v e u-disparidade e sua importância para segmentação da estrada: (a) *frame* de entrada; (b) mapa de disparidade; (c) o mapa de v-disparidade; (d) o mapa u disparidade. Os pixels brancos no mapa de disparidade são pixels invalidados pela abordagem de detecção de oclusão.



Fonte: O autor.

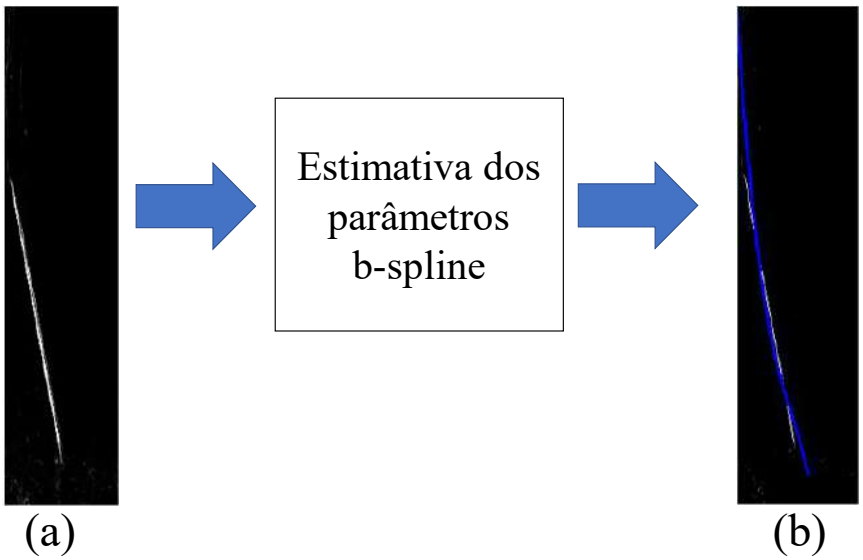
Figura 17 – Ilustração do conceito de filtragem de obstáculos do mapa de disparidade a partir do mapa u-disparidade: (a) mapa de disparidade não filtrado; (b) o mapa de v-disparidade; (d) o mapa u disparidade. Os pixels brancos no mapa de disparidade são pixels invalidados pela abordagem de detecção de oclusão.



Fonte: O autor.

A modelagem do perfil da estrada requer primeiro a remoção no mapa v-disparidade de regiões de ruídos referentes a obstáculos a fim de destacar a curva da estrada. Na verdade, essa remoção é feita diretamente no mapa de disparidade com o suporte do mapa u-disparidade. Qualquer pixel de coordenadas (u, v, d) no espaço de imagem quando mapeado no mapa u-disparidade possui um valor de *bin* b. Se a diferença do valor da

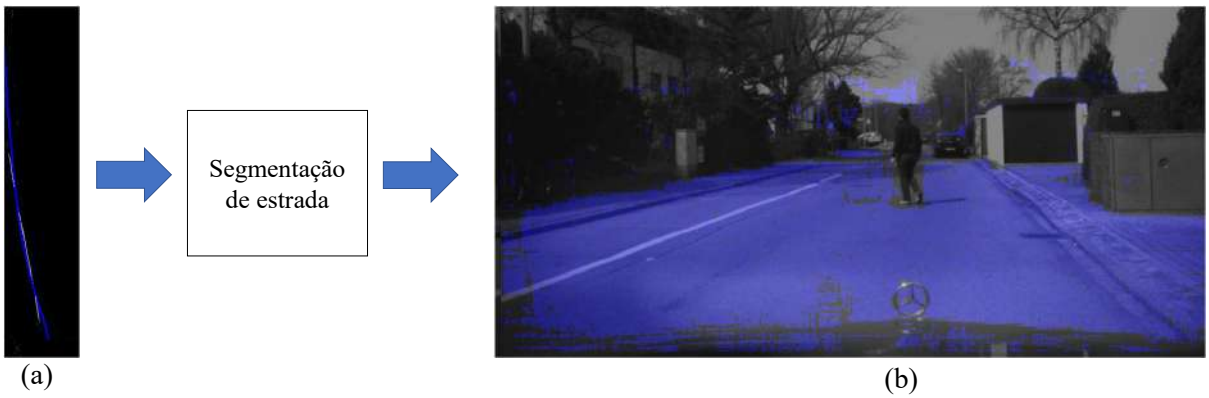
Figura 18 – Ilustração do resultado da estimativa dos parâmetros da função B-splines: (a) mapa v-disparidade; (b) função estimada em azul.



Fonte: O autor.

sua disparidade d com o valor b for maior que um limite ω_u, esse pixel é considerado como obstáculo, sendo então invalidado no mapa de disparidade. A Figura 17 ilustra esse processo e geração do mapa de disparidade filtrado. Consequentemente, o mapa v-disparidade sem as regiões de obstáculos, como mostrado na Figura 17(e), é calculado diretamente do mapa de disparidade filtrado. Quanto maior o valor do parâmetro ω_u, menos obstáculos são detectados.

Figura 19 – Ilustração do resultado de segmentação a partir da função B-splines estimada: (a) mapa v-disparidade filtrado com a função B-splines destacada em azul; (b) a segmentação da estrada destacada em azul.



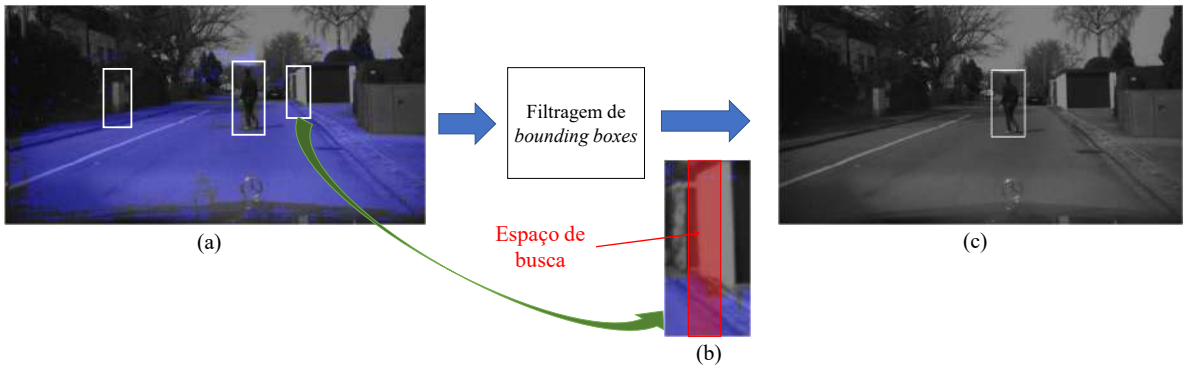
Fonte: O autor.

Para modelar a curva do perfil de estrada, nós adotamos a função B-spline (BROGGI et al., 2013), que é uma função polinomial por partes suavemente unida, no nosso caso, formada por cinco funções splines cúbicas. Essa estratégia permite criar e gerenciar formas e superfícies complexas, como terrenos planos, inclinados ou ondulados, todos presentes

em cenários reais. A curva azul na Figura 18 indica a função B-splines cujos parâmetros foram estimados a partir dos dados do mapa v-disparidade filtrado, usando o método dos mínimos quadrados (TIBSHIRANI et al., 2021).

A partir dessa função é possível saber o valor correto da disparidade que se refere a estrada para cada posição de linha no *frame*. Assim, verificamos para cada pixel se a sua localização no *frame* e sua disparidade no mapa de disparidade estão dentro da curva B-spline estimada. Em caso afirmativo, consideramos que é um pixel de estrada. O resultado é destacado em azul na Figura 19.

Figura 20 – Ilustração da estratégia de filtragem de *bounding boxes* a partir do *frame* segmentado: (a) a segmentação da estrada destacada em azul e os *bounding boxes* a partir do detector; (b) exemplo da aplicação da filtragem destacando o espaço no qual é realizada contagem dos pixels que são de estrada; (c) resultado da aplicação da filtragem



Fonte: O autor.

A partir do *frame* segmentado, nós verificamos para cada *bounding box* que vem do detector se este contém uma quantidade de mínima de pixels que são de estrada dentro de uma espaço de busca. Em caso positivo, nós descartamos esse *bounding box*. A Figura 20 ilustra o resultado da aplicação do filtragem de *bounding boxes* através da segmentação.

3.1.3 Supressão Não-Máxima

Nós adotamos a técnica proposta por Bodla et al. (2017) chamada de Soft-NMS que é uma versão melhorada do algoritmo tradicional NMS. A versão tradicional consiste em um procedimento iterativo guloso que começa selecionando o melhor *bounding box*, ou seja, aquele com maior pontuação de confiança. Como descrito no Capítulo 4, cada *bounding box* de resposta a partir do detector de pedestres possui um valor de confiança associado que indica a certeza de conter de fato um pedestre. Quanto maior esse valor, maior é a certeza. Em seguida, os *bounding boxes* que estão próximos do *bounding box* selecionado são suprimidos. A medida de proximidade entre *bounding boxes* é a interseção sobre a união (em inglês, *Intersection over Union* - IoU), definida pela Equação 3.12. Essa Equação define uma relação da área de sobreposição de dois *bounding boxes* a_i e a_j com a área total deles dois. Quanto mais sobrepostos estiverem, mais próximo do valor 1 essa

função retornará.

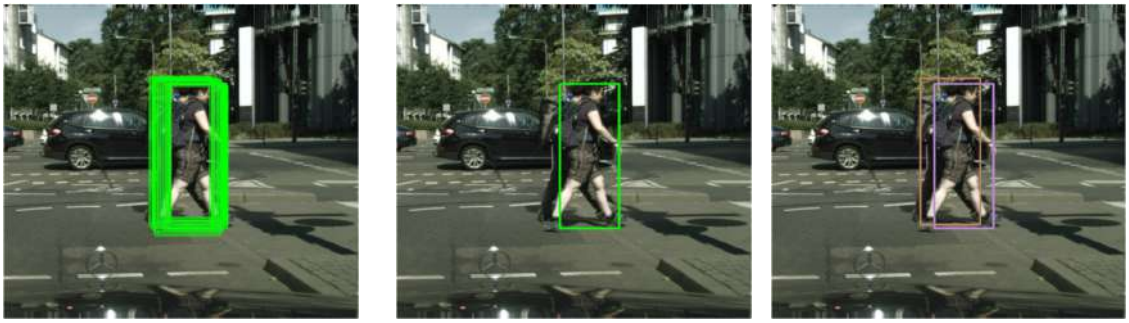
$$\Gamma(a_i, a_j) = \frac{A(a_i \cap a_j)}{A(a_i \cup a_j)} \tag{3.12}$$

Supondo que o *bounding box* a_j foi escolhido como sendo o melhor, então o *bounding box* a_i está sujeito a supressão não-máxima se o valor da função de cobertura dado pela Equação 3.12 estiver acima de um limiar proximidade θ_{nms}. Esta verificação é realizada em todos os *bounding boxes* disponíveis que ainda não foram suprimidos. Com os *bounding boxes* restantes, o próximo de maior pontuação é selecionado e o procedimento é repetido até que não restem mais *bounding boxes*.

$$s_i = \begin{cases} s_i, & \text{se } \Gamma(a_i, a_j) < \theta_{nms} \\ 0, & \text{Caso contrário} \end{cases} \tag{3.13}$$

Uma das principais desvantagens dessa abordagem tradicional é que ela suprime todos os *bounding boxes* dentro da vizinhança. Em termos numéricos, a pontuação de confiança s_i de um dado *bounding box* a_i, sujeito a supressão, é definida para zero, como descrito pela Equação 3.13. A definição de valor zero é equivalente a suprimir o *bounding box*. Em outras palavras, este *bounding box* nunca será selecionado como o próximo melhor *bounding box* nas próximas repetições.

Figura 21 – Ilustração do problema do NMS tradicional e o que o Soft-NMS busca resolver. (a) Todos os *bounding boxes* previstos pelo detector sem NMS. (b) Os *bounding boxes* após NMS tradicional. (c) Os *bounding boxes* corretos que o Soft-NMS pode fornecer.



(a) (b) (c)

Fonte: Zhang et al. (2020)(ADAPTADO).

Se dois ou mais pedestres estiverem próximos um do outro, todos, exceto um deles, serão suprimidos. O problema com a abordagem tradicional é ilustrado na Figura 21. Como consequência teremos uma taxa de faltas maior na detecção.

O Soft-NMS mitiga esse problema reduzindo a pontuação de confiança como uma função linear da proximidade para o *bounding box* de maior pontuação, como mostrado na Equação 3.14.

$$s_i = \begin{cases} s_i, & \text{se } \Gamma(a_i, a_j) < \theta_{nms} \\ s_i(1 - \Gamma(a_i, a_j)), & \text{Caso contrário} \end{cases} \tag{3.14}$$

Assim, estes *bounding boxes* vizinhos ainda teriam um valor de pontuação de confiança, embora menor, e poderiam ainda participar do processo iterativo e ser escolhido como *bounding box* de maior pontuação em uma outra iteração.

3.1.4 Rastreamento

O rastreamento identifica e rotula as medições que pertencem ao mesmo pedestre em vários *frames* consecutivos. As medições são as distâncias laterais e longitudinais $p_i = (u, d)$ no espaço de imagem. Um modelo de movimento de pedestres é crucial para a eficácia da associação de medições e rastros e também para a realização da previsão de trajetória. Assim, detalhamos o modelo de movimento e depois as etapas de associação.

3.1.4.1 Modelo de Movimento de Pedestres

Uma trajetória é uma representação espaço-temporal do deslocamento do pedestre. Com base no modelo cinemático de velocidade constante (em inglês, *constant velocity* - CV), a evolução da trajetória do pedestre é avaliada pelos parâmetros de movimento como posição e velocidade do pedestre. Em termos matemáticos, o modelo CV do movimento do pedestre é representado através da variável de estado $\mathbf{x} = (x, z, v_x, v_z)$, onde x, z, v_x e v_z são, respectivamente, as distâncias lateral e longitudinal; e as velocidades laterais e longitudinais no espaço de coordenadas de câmera. A relação entre a posição $p_c = (x, y, z)$, em coordenadas de câmera, e a medição $p_i = (u, v, d)$, em coordenadas de imagem é definida no apêndice E. A coordenada v é ignorada porque nós consideramos que o pedestre está sempre no plano do solo.

As medições (u, d) do pedestre obtidas a partir das abordagens de detecção de pedestre, visão estéreo e estimativa da distância são ruidosas. Para lidar com esse problema foi utilizado o filtro de Kalman na sua versão estendida (em inglês, *Extended Kalman Filter* - EKF) (BAR-SHALOM; LI; KIRUBARAJAN, 2004). O EKF é uma variação do filtro de Kalman (KF) tradicional, descrito detalhadamente no Apêndice A, que lida com funções não lineares como a da transformação perspectiva descrita na Equação E.1.

O EKF envolve uma iteração de dois estágios: previsão de estado e atualização de medição. A previsão de estado estima um estado a priori $\hat{\mathbf{x}}_k = (x, z, v_x, v_z)$ na etapa de tempo k a partir do estado anterior $\mathbf{x}_{k-1}$ através das seguintes Equações:

$$\hat{\mathbf{x}}_k = \mathbf{A}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{s}_k \tag{3.15}$$

$$\hat{\mathbf{P}}_k = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^\top + \mathbf{Q}, \tag{3.16}$$

onde $\mathbf{A}_k \in \mathbb{R}^{4\times4}$ é a matriz de transição que descreve o relacionamento entre $\mathbf{x}_k$ e $\mathbf{x}_{k-1}$, $\mathbf{s}_k \in \mathbb{R}^{4\times1}$ é uma matriz contendo qualquer entrada de controle. O termo $B = I_{4\times4}$ é uma matriz que descreve as componentes de controle que modifica o estado $\mathbf{x}_k$, enquanto o termo $\mathbf{P}_k$ é a matriz de covariância de estado e o termo $\mathbf{Q}$ é a covariância de ruído do

processo. O ruído do processo descreve a quantidade de desvio, ou incerteza, do movimento real do objeto a partir do modelo de movimento escolhido. O acento circunflexo (ˆ) em cima da variável indica que é uma estimativa *a priori*, ou seja, foi derivada antes que se saiba o valor real.

A covariância do processo $\mathbf{Q}$ atua como uma matriz de ponderação que relaciona a covariância entre a i-ésima e a j-ésima variável de estado. A matriz $\mathbf{Q}$ é modelada como aceleração de ruído branco discreto (OLSON; RUSSELL; CARPENTER, 2017; KOEHLER et al., 2013; GOLDHAMMER et al., 2014a). Essa distribuição é descrita pela Equação 3.17 que depende do parâmetro de desvio padrão do ruído de processo $\sigma_{\mathbf{x}}$. O termo Δt denota o tempo de ciclo entre *frames*.

$$\mathbf{Q} = \begin{bmatrix} \frac{1}{4}\Delta t^4\sigma_{\mathbf{x}}^2 & 0 & \frac{1}{2}\Delta t^3\sigma_{\mathbf{x}}^2 & 0 \\ 0 & \frac{1}{4}\Delta t^4\sigma_{\mathbf{x}}^2 & 0 & \frac{1}{2}\Delta t^3\sigma_{\mathbf{x}}^2 \\ \frac{1}{2}\Delta t^3\sigma_{\mathbf{x}}^2 & 0 & \Delta t^2\sigma_{\mathbf{x}}^2 & 0 \\ 0 & \frac{1}{2}\Delta t^3\sigma_{\mathbf{x}}^2 & 0 & \Delta t^2\sigma_{\mathbf{x}}^2 \end{bmatrix} \tag{3.17}$$

Uma vez que a cena é observada a partir do ponto de vista da câmera instalada no veículo, a origem do sistema de coordenadas está se movendo junto com o veículo. Assim a evolução da variável de estado do tempo t_{k-1} para o tempo t_k precisa ser compensada pelo movimento do veículo. Essa compensação é incorporada na etapa de predição do EKF (KELLER; ENZWEILER; GAVRILA, 2011), mais especificamente, nas matriz de transição de estado $\mathbf{A}_k$ e na matriz de controle $\mathbf{s}_k$, como mostrado, respectivamente, pelas seguintes Equações:

$$\mathbf{A}_k = \begin{bmatrix} \mathbf{R}_{M_c,k} & 0_{2\times 2} \\ 0_{2\times 2} & \mathbf{R}_{M_c,k} \end{bmatrix} \mathbf{A} \tag{3.18}$$

$$\mathbf{s}_k = \begin{bmatrix} \mathbf{t}_{M_c,k} \\ 0_{2\times 1} \end{bmatrix}, \tag{3.19}$$

onde $\mathbf{A}$ é a matriz de transição tradicional (RIBEIRO, 2004) para o modelo CV, descrita como:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.20}$$

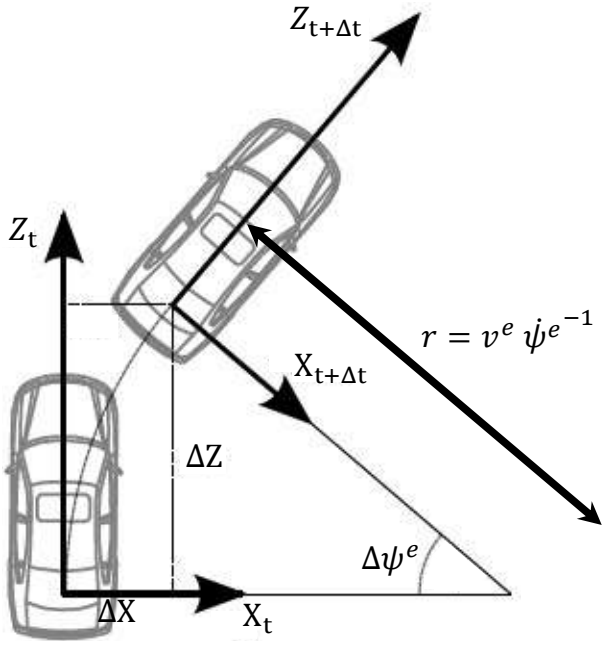
As matrizes $\mathbf{R}_{M_c,k} \in \mathbb{R}^{2\times 2}$ e $\mathbf{t}_{M_c,k} \in \mathbb{R}^{2\times 1}$ são, respectivamente, de rotação e translação de cena (HARTLEY; ZISSERMAN, 2003; KELLER; ENZWEILER; GAVRILA, 2011) no tempo t_k. Essas matrizes são obtidas a partir da matriz de homografia inversa do movimento do

veículo $\mathbf{M}_c$ descrita como:

$$\mathbf{M}_c = \mathbf{D}^{-1}\mathbf{M}_v\mathbf{D}, \tag{3.21}$$

onde a matriz $\mathbf{D}$ define a relação em coordenadas homogêneas entre o sistema de coordenada da câmera e o sistema de coordenada do veículo, descrita no apêndice E. A matriz $\mathbf{M}_v$ é a matriz de movimento inercial em coordenadas do veículo. Essa matriz é obtida através da velocidade do veículo v^e, da taxa de guinada $\dot{\psi}^e$ e do modelo de movimento do veículo que consideramos o de faixa única (em inglês, *single track*) (SCHRAMM; HILLER; BARDINI, 2014), como mostrado na Figura 22.

Figura 22 – Modelo de faixa única para modelagem do movimento do veículo.



Fonte: Keller, Enzweiler e Gavrila (2011)(ADAPTADO).

Entre período de tempo t e $t + \Delta t$, o veículo percorre a distância $(\Delta X, \Delta Z)$ com mudança de orientação $\Delta\psi^e = \dot{\psi}^e\Delta t$. Assim, $\mathbf{M}_v$ é descrita pela seguinte equação:

$$\mathbf{M}_v = \begin{bmatrix} \cos(\Delta\psi^e) & 0 & \sin(\Delta\psi^e) & -\Delta X \\ 0 & 1 & 0 & 0 \\ -\sin(\Delta\psi^e) & 0 & \cos(\Delta\psi^e) & -\Delta Z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.22}$$

onde ΔX e ΔZ são respectivamente:

$$\Delta X = v^e\,\dot{\psi}^{e^{-1}}\,[1 - \cos(\Delta\psi^e)] \tag{3.23}$$

$$\Delta Z = v^e\,\dot{\psi}^{e^{-1}}\,[\sin(\Delta\psi^e)] \tag{3.24}$$

A etapa de atualização de medição é responsável por corrigir a estimativa a priori do estado a partir de uma medição $\mathbf{z}_k = (u, d)$ no tempo k. As seguintes sequências de equações descrevem esta etapa:

$$\mathbf{K}_k = \hat{\mathbf{P}}_k \mathbf{H}_k^\top (\mathbf{H}_k \hat{\mathbf{P}}_k \mathbf{H}_k^\top + \mathbf{R})^{-1} \tag{3.25}$$

$$\mathbf{x}_k = \hat{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k)) \tag{3.26}$$

$$\mathbf{P}_k = (\mathbf{I}_k - \mathbf{K}_k \mathbf{H}_k)\hat{\mathbf{P}}_k, \tag{3.27}$$

onde $\mathbf{K}_k$ é o ganho de Kalman na etapa de tempo k e $\mathbf{R}$ é a matriz de covariância de ruído de medição. O ruído de medição descreve a quantidade de desvio, ou incerteza, do valor medido a partir do valor correto. A matriz $\mathbf{R}$ é dada pela Equação 3.28, que depende dos desvios padrões do erro de medição lateral, σ_u, e do erro de medição longitudinal, σ_d (SCHNEIDER; GAVRILA, 2013; FARAGHER, 2012).

$$\mathbf{R} = \begin{bmatrix} \sigma_u^2 & 0 \\ 0 & \sigma_d^2 \end{bmatrix} \tag{3.28}$$

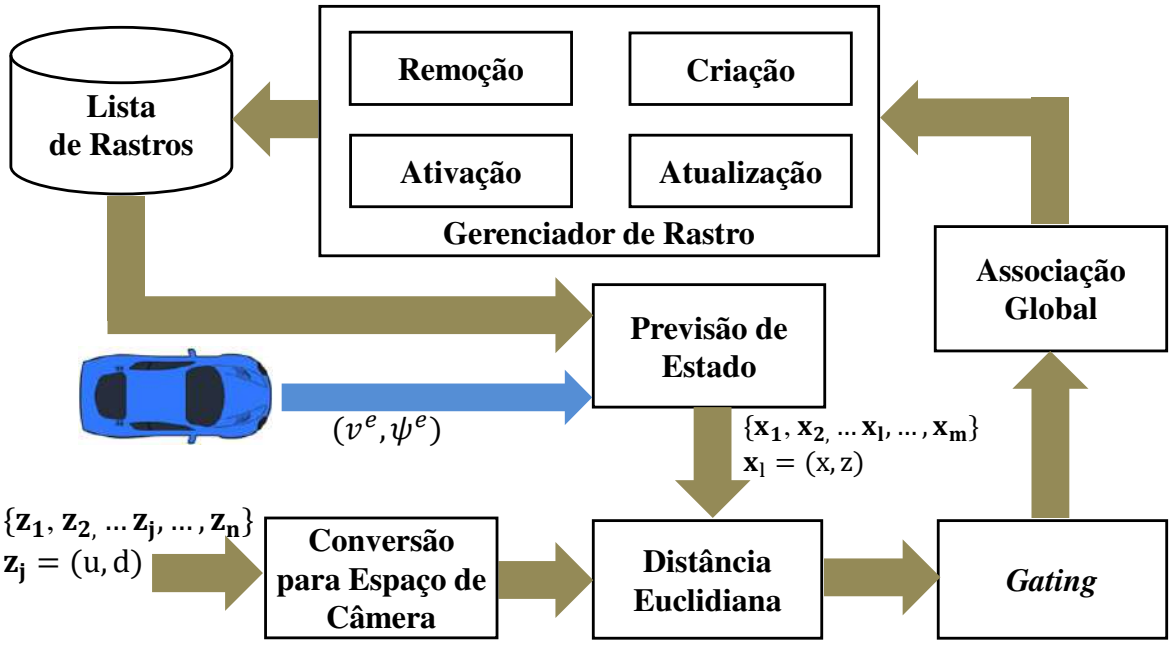
$$\mathbf{H}_k = \begin{bmatrix} \frac{f}{z} & -\frac{fx}{z^2} & 0 & 0 \\ 0 & -\frac{fb}{z^2} & 0 & 0 \end{bmatrix} \tag{3.29}$$

A função de transformação h na Equação E.1, a partir do espaço da câmera (x, z) para o espaço de imagem (u, d) é não-linear. Então a matriz $\mathbf{H}_k$, apresentada na Equação 3.29, é uma matriz jacobiana da função $h = (h_1, h_2, h_3)$. Novamente, a função h_2 é descartada porque nós consideramos que o pedestre está tocando o plano da estrada, que é o mesmo do veículo.

3.1.4.2 Associação e Gerenciamento de Rastros

As etapas para associação de rastros e medições são mostradas na Figura 23. Para cada rastro que está na lista de rastros é realizada a previsão de estado com compensação de movimento usando a Equação 3.15. Em seguida é calculada a dissimilaridade entre os rastros previstos e as novas medições usando a distância euclidiana. Estes valores de dissimilaridade são utilizados na etapa de *gating* para excluir associações improváveis cuja distância seja maior que um limiar fixo l_{gating} metros. Para as associações restantes, que não foram filtradas pela etapa de *gating*, foi utilizada a abordagem de Kuhn (KUHN, 1955)(FRANK, 2005), também chamado de método Húngaro, para associação global um para um de rastro com medições resultando em um conjunto contendo uma lista de rastros correspondidos com medições, uma lista de rastros não correspondidos e uma lista de medições não correspondidas. O método húngaro é uma técnica de otimização computacional que aborda o problema de atribuição em tempo polinomial. Esse método tem

Figura 23 – Abordagem para rastreamento de pedestres



Fonte: O autor.

sido adotado por vários sistemas de proteção ao pedestre (LEE; SHIN; KWON, 2017)(OTTO, 2013)(LLORCA et al., 2011).

Este conjunto é utilizado na etapa de gerenciamento de rastros para atualizar a lista de rastros existentes que inicialmente está vazia. Para os rastros não correspondidos, o gerenciador de rastros cria um novo rastro com o estado inicial $\mathbf{x} = (x, z, 0, 0)$ onde x e z são respectivamente as distâncias lateral e longitudinal referentes a respectiva localização no espaço de câmera. Para cada rastro i criado, existe um contador M_i que contabiliza o número de *frames* não associados desde a última associação bem-sucedida, um contador F_i que contabiliza a quantidade de associações bem-sucedidas desde a criação deste rastro e um estado para indicar se o rastro está confirmado ou não. O contador M_i é incrementado durante a etapa de previsão de estado e zerado quando o rastro for associado a uma medição. Os rastros que excedem uma idade máxima predefinida de $M_{\max}$ são consideradas como tendo deixado a cena e são excluídos da lista de rastros.

Os novos rastros são classificados como provisórios. Quando o contador F_i estiver maior do que um valor fixo $F_{\min}$ este rastro é considerado como confirmado. Porém, se enquanto o rastro estiver em provisório e em determinado *frame* ele não for correspondido, ele é removido da lista de rastros. Para cada rastro i que foi correspondido com a medição é realizada a atualização de medição do estado interno $\mathbf{x}$ deste rastro a partir do EKF e o contador F_i é incrementado. Somente os rastros considerados como confirmados são utilizados para previsão de trajetória.

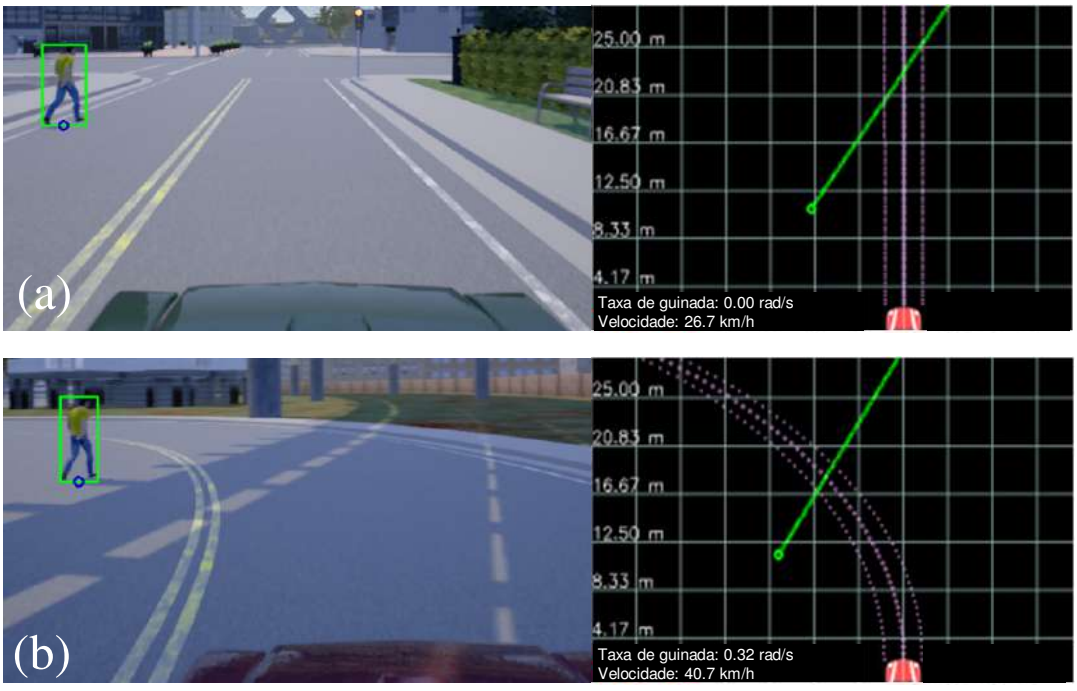
3.1.5 Previsão de Trajetória

A partir da variável de estado de cada rastro i confirmado, é realizada a propagação das suas posições futuras através da execução de sucessivas etapas de previsão de estado do EKF sem a etapa de atualização de medição. A cada variável estimada na etapa de tempo k é feita a previsão da próxima variável de estado na etapa de tempo $k+1$ através da Equação 3.30, onde $\mathbf{A}$ é a matriz de transição definida pela Equação 3.20.

$$\hat{\mathbf{x}}_{k+1} = \mathbf{A}\,\hat{\mathbf{x}}_k \tag{3.30}$$

O centro de coordenada do veículo está localizado no meio do eixo traseiro do veículo. Dessa forma, as posições atuais e futuras dos pedestres precisam ser transformadas para o mesmo espaço utilizado pelo veículo para permitir realizar estimativas de previsão de colisão baseadas na dimensão do veículo. A transformação é definida pela Equação E.7.

Figura 24 – Exemplos de propagação de posições futuras do pedestre e do veículo: (a) veículo se movendo em linha reta. (b) veículo realizando uma manobra de curva. Esses *frames* são obtidos através do simulador CARLA.



Fonte: O autor.

A trajetória futura do veículo é prevista a partir das medições atuais de taxa de guinada $\dot{\psi}^e$ e velocidade v^e, usando o modelo de faixa única (KELLER; ENZWEILER; GAVRILA, 2011), mostrado na Figura 22. Usando a relação de triângulo, um veículo movendo-se no raio de curva $r = v^e\,\dot{\psi^e}^{-1}$, terá a sua a posição lateral (X_t) e longitudinal (Z_t) em um tempo futuro t descritas conforme as Equações 3.31 e 3.32, respectivamente. Essas equações são similares as Equações 3.23 e 3.24, com a diferença que a variável t determina qualquer posição futura a partir da posição atual do veículo. Dessa forma, para cada valor de t,

podemos calcular a nova posição do veículo. Uma vez que os eventos são discretos então $t = k\ \Delta T_{\text{prev}}$, onde ΔT_{prev} é o intervalo de tempo de amostragem. O horizonte de tempo máximo de previsão $T_{\text{prev,max}}$ em segundos determina o tempo em que pode ser realizado a previsão de trajetória dos pedestres e do veículo.

$$X(t) = v^e\ \dot{\psi^e}^{-1}\ [1 - \cos(\dot{\psi^e} t)] \tag{3.31}$$

$$Z(t) = v^e\ \dot{\psi^e}^{-1}\ \sin(\dot{\psi^e} t) \tag{3.32}$$

Os pontos propagados do veículo são baseados no sistema de coordenadas que se encontra no meio do eixo traseiro do veículo. Uma vez que a colisão acontece na parte da frente do veículo, então para cada ponto propagado é necessário ser somada a distância do eixo traseiro até a parte da frente do veículo. Além disso, uma vez que toda extensão da dianteira do veículo é importante para verificar a possibilidade de colisão, então os pontos de extremidade do veículo foram propagados baseado na sua largura e na sua orientação em cada tempo t.

A Figura 24 mostra a propagação do movimento do pedestre e do veículo em etapas de tempo futuros em dois cenários diferentes. O cenário 1, mostrado na Figura 24 (a), apresenta um pedestre atravessando em diagonal ao veículo que se move em linha reta. O cenário 2, mostrado na Figura 24 (b), apresenta um pedestre atravessando em diagonal ao veículo que está realizando uma manobra de curva.

3.1.6 Análise de Interseção

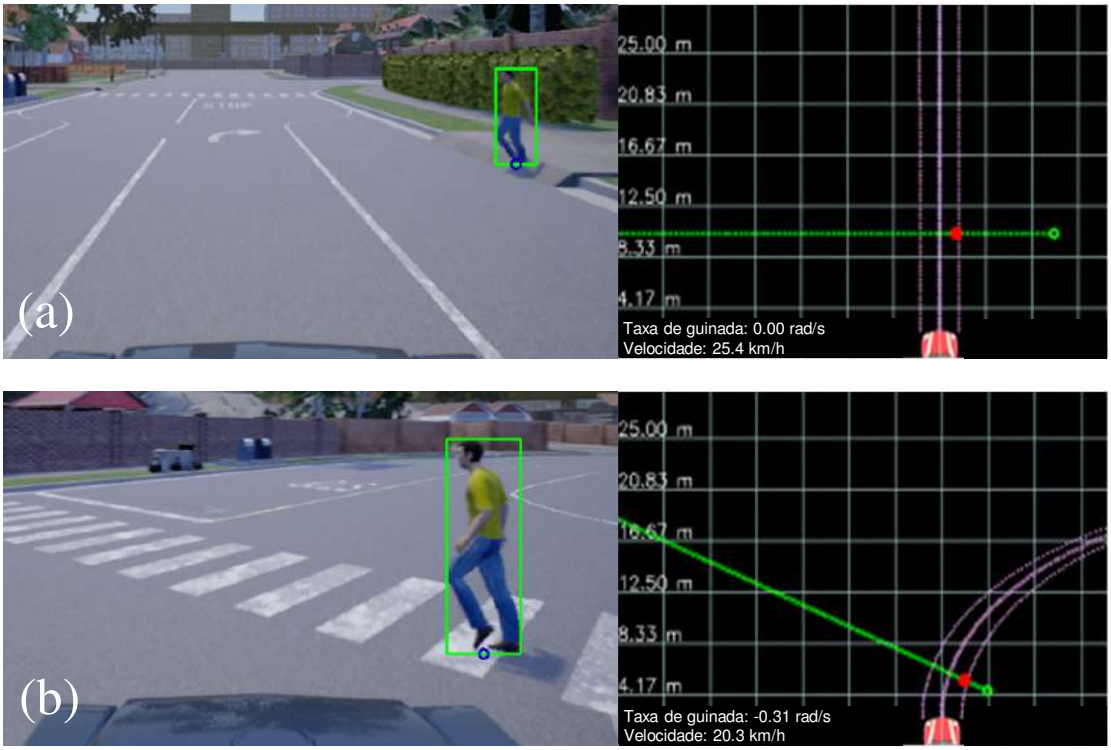
Uma colisão entre duas entidades acontecerá quando duas trajetórias que evoluem de seus locais atuais chegarem na mesma posição no mesmo instante de tempo no futuro. Dessa forma, para verificar essa situação de colisão em um instante futuro, são utilizadas as sequências futuras de todos os pedestres e do carro que foram calculadas na etapa de previsão de trajetória. Nós expressamos todas essas sequências através da Equação 3.33.

$$\begin{gathered}
\{1, \mathbf{z}_{\text{carro},1}, \mathbf{z}^1_{\text{ped},1}, \ldots, \mathbf{z}^q_{\text{ped},1}, \ldots, \mathbf{z}^M_{\text{ped},1}\} \\
\vdots \\
\{k, \mathbf{z}_{\text{carro},k}, \mathbf{z}^1_{\text{ped},k}, \ldots, \mathbf{z}^q_{\text{ped},k}, \ldots, \mathbf{z}^M_{\text{ped},k}\} \\
\vdots \\
\{H, \mathbf{z}_{\text{carro},H}, \mathbf{z}^1_{\text{ped},H}, \ldots, \mathbf{z}^q_{\text{ped},H}, \ldots, \mathbf{z}^M_{\text{ped},H}\}
\end{gathered} \tag{3.33}$$

O termo k indica a k-ésima etapa de tempo da previsão, H é a etapa de tempo máximo da previsão e $\mathbf{z}_{\text{carro},k}$ é o vetor que descreve as posições em coordenadas (X, Z) na etapa de tempo k da extremidade esquerda e direta do veículo. O termo H é definido como $H = T_{\text{prev,max}}/\Delta T_{\text{prev}}$, onde $T_{\text{prev,max}}$ é o horizonte de tempo máximo de previsão em segundos e ΔT_{prev} é o intervalo de tempo de amostragem em segundos. As localizações

$\mathbf{z}^1_{\text{ped},k}, \ldots, \mathbf{z}^M_{\text{ped},k}$ descrevem as posições em coordenadas (X, Z) dos M pedestres rastreados na etapa de tempo k.

Figura 25 – Exemplos de funcionamento da detecção de colisão envolvendo pedestre e o veículo em cenários que aconteceria uma colisão.



Fonte: O autor.

Possíveis posições de colisão são identificadas quando as posições futuras de cada pedestre estiverem no mesmo instante de tempo futuro tocando a parte da frente do veículo. Em outras palavras, se a posição de um pedestre q na etapa de tempo k, ou seja, $\mathbf{z}^q_{\text{ped},k}$, tocar a reta formada pelos pontos extremos do veículo $\mathbf{z}_{\text{carro},k}$, nesse mesmo instante de tempo k, a posição $\mathbf{z}^q_{\text{ped},k}$ é marcada como uma posição de colisão. Esse procedimento é repetido para todos os pedestres em todas as etapas futuras de tempo. Vale ressaltar que para cada posição marcada como colisão, também se conhece o tempo que levará para acontecer a colisão, ou seja, o seu TTC.

A Figura 25 mostra exemplos da técnica de análise de interseção envolvendo cenários que de fato resultariam em uma colisão entre veículo e pedestre. O ponto em vermelho indica onde aconteceria a colisão se o veículo e o pedestre mantivessem os seus movimentos.

3.2 PLATAFORMAS E ASPECTOS DE IMPLEMENTAÇÃO

No decorrer dos próximos capítulos nós utilizamos as plataformas descritas na Tabela 1. Quando nós estivermos utilizando o processamento em GPU ou em FPGA, nós estamos nos referindo as plataformas 1 e 2, respectivamente.

Tabela 1 – Plataformas para implementação

Plataforma	Descrição
1	GPP Core i5-9400F com 16 GB DDR3 GPU RTX 2070 de 8GB
2 (HARPv2)	GPP Xeon E5-2600v4 com 32 GB DDR3 FPGA Arria 10 GT (10AX115N2F40E2LG)

Fonte: O autor.

Todas as etapas do sistema PCP foram implementadas em linguagem C++ sem bibliotecas de alto nível. Para a implementação do SGM, inicialmente nós adaptamos a solução em linguagem CUDA, proposta em Hernandez-Juarez et al. (2016), e adicionamos a etapa de detecção de pixels ocluídos, descrita na Seção 3.1.1.1. Essa versão de implementação do SGM que chamamos de SGM-CUDA é utilizada na avaliação do algoritmo de detecção no Capítulo 4. Neste capítulo, o foco não é o desempenho de processamento e sim a precisão. Na medida em que otimizações de desempenho foram desenvolvidas para o sistema PCP através de uma abordagem HW/SW, nós integramos a nossa solução (CAMBUIM et al., 2019) para o cálculo do SGM baseado em FPGA com otimização de desempenho de processamento e suporte à detecção de pixels ocluídos. Nós chamamos essa segunda solução de SGM-FPGA.

3.3 BASE DE DADOS E MÉTRICAS DE AVALIAÇÃO

Para avaliação da capacidade e precisão do sistema PCP, nós investigamos, além da própria tarefa de previsão de colisão, a tarefa de localização. A tarefa de localização é uma combinação das tarefas de detecção de pedestre, correspondência estéreo e estimativa da distância. Qualquer mudança de abordagem na tarefa de localização tem um impacto significativo na qualidade da previsão de colisão. Os parâmetros de precisão e tempo de resposta da tarefa de localização e do sistema PCP precisam ser considerados para avaliação da qualidade na tarefa de previsão de colisão. Nós descrevemos a seguir as bases de dados e métricas adotadas para dar suporte a avaliação da localização e da previsão de colisão que são utilizadas nos capítulos seguintes.

3.3.1 Base de dados 1

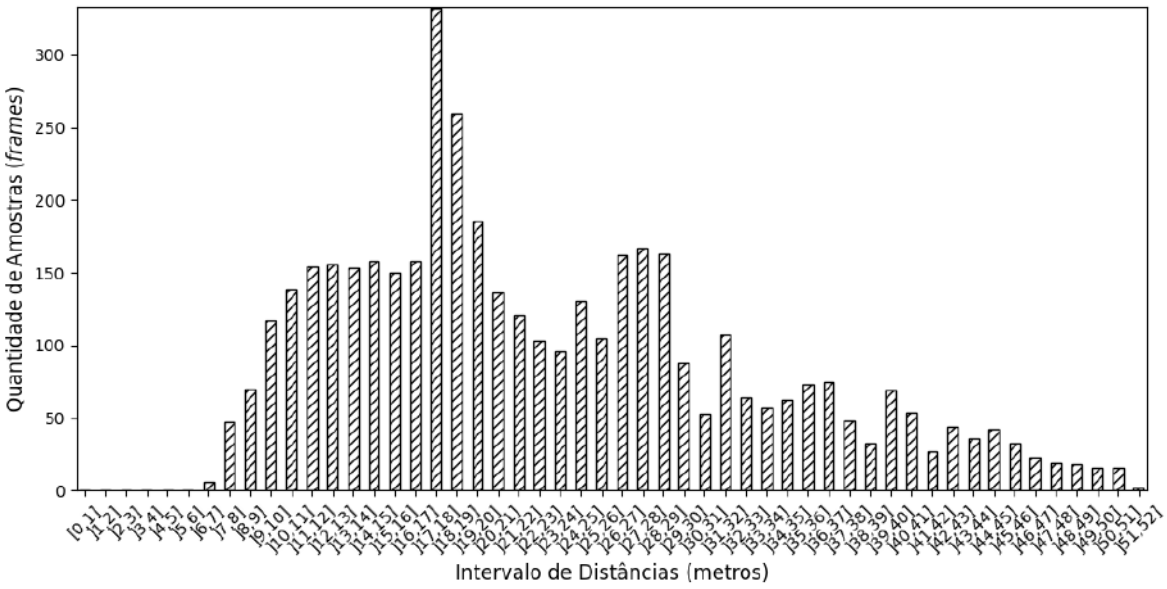
A base de dados 1 é fornecida pelos autores Schneider e Gavrila (2013). Esta base de dados permite a avaliação da tarefa de localização e consiste em 68 amostras, cada uma contendo uma sequência de *frames* registrados em cenários reais juntamente e sincronizados com a velocidade e taxa de guinada do veículo. Um total de 55 amostras foram registradas com veículo se movendo entre 20 e 30 km/h enquanto as outras foram registradas com o veículo parado. Todas as amostras foram registradas com um sistema de câmera estéreo

montado atrás do retrovisor traseiro. A resolução das imagens é de 1176 × 640 pixels e a taxa de coleta é de 16 FPS.

Figura 26 – Exemplos de alguns cenários a partir da base de dados 1. Cada *frame* contém os *bounding boxes* GT para localização do pedestre.

Fonte: Schneider e Gavrila (2013).

Figura 27 – Distribuição da quantidade de *frames* por intervalo de 1 metro de distância ao pedestre.



Fonte: O autor.

A base de dados 1 também fornece os *bounding boxes* e distância de referência do pedestre (em inglês, *ground truth* - GT) em cada *frame*. Todas as informações de geometria do veículo e das câmeras, necessárias para projetar as posições do pedestre no espaço de coordenadas do veículo, são fornecidas. Alguns dados são a altura da câmera ao solo, distância ao eixo de coordenadas comum, largura do veículo. Mais detalhes sobre geometria e parâmetros necessários podem ser encontrados no Apêndice E. Além disso, as amostras são divididas em amostras de treinamento e amostras se teste, o que permite ajuste de

parâmetros das etapas e treinamento de modelos preditivos e avaliação com a base de testes. A Figura 26 apresenta alguns exemplos a partir desta base de dados.

A Figura 27 apresenta a distribuição de quantidade de *frames* disponíveis por distância ao pedestre na base de dados de teste. Por exemplo, pedestres que estão entre 18 e 19 metros de distância para o veículo, a base de teste contém aproximadamente 275 *frames* disponíveis para avaliação. Essa distribuição auxilia no entendimento do impacto que a melhoria com a combinação das janelas de detecção de tamanhos diferentes traz para a melhoria de todo o conjunto de testes.

Uma vez que estamos avaliando a capacidade de localização dentro de certos intervalos de distância entre o pedestre e o veículo, a base de teste é dividida em grupos com relação a esses intervalos. O Grupo 1 é formado pelos *frames* com distâncias entre 6 e 20 metros o Grupo 2 contém *frames* com distâncias entre 20 e 52 metros e o Grupo 3 é formado por todos os *frames*, ou seja, com distâncias entre 6 e 52 metros. Existem 2085, 2244 e 4329 *frames* para os Grupos 1, 2 e 3, respectivamente.

3.3.2 Métrica de Avaliação para Localização

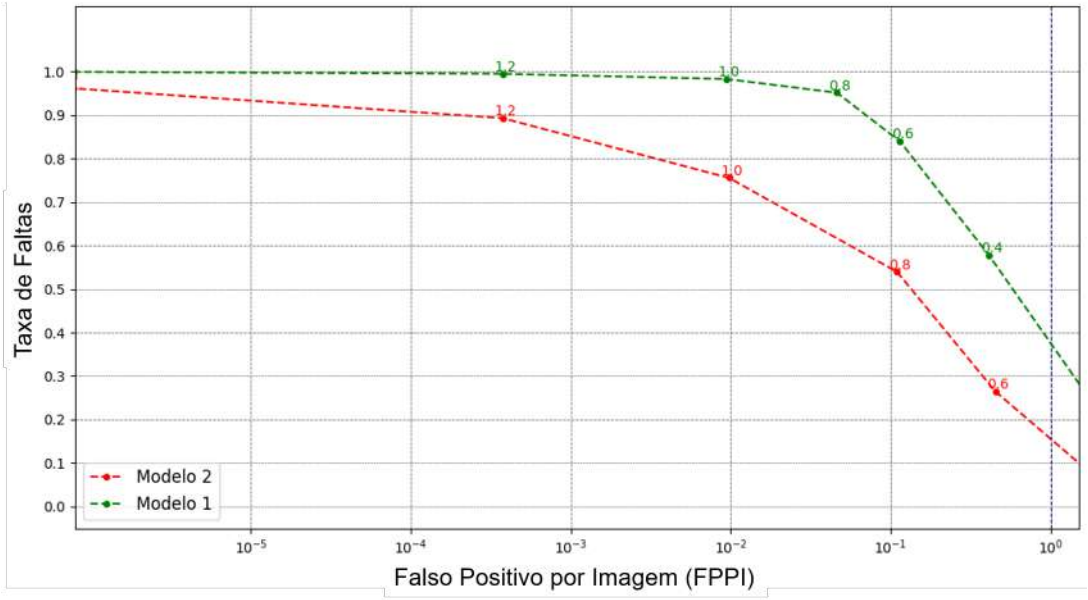
A avaliação da localização considera a posição lateral (X) e longitudinal (Z) do pedestre no espaço de coordenadas do veículo. A estratégia proposta em Keller, Enzweiler e Gavrila (2011) é adotada neste trabalho para comparar a saída do nosso sistema com a saída de referência. Esta estratégia especifica uma tolerância de localização, ou seja, o desvio posicional máximo que permite contabilizar uma localização correta. A tolerância de localização é necessária por causa do erro da estimativa de disparidade, devido a questões de calibração, que aumenta a medida que a distância do pedestre aumenta (SCHNEIDER; GAVRILA, 2013). Os valores de tolerância típicos de Z = 30% e X = 10% (KELLER; ENZWEILER; GAVRILA, 2011)(ENZWEILER et al., 2012) foram adotados neste trabalho, o que significa que, por exemplo, a uma distância de 10 metros, o erro de localização de ±3m e ±1m na posição longitudinal e lateral, respectivamente, é aceita como correta.

Os detectores são avaliados através da métrica de taxa de faltas (em inglês, *miss rate*) versus falsos positivos por imagem (FPPI) (BENENSON et al., 2014b) conforme mostrado na Figura 28. A taxa de faltas é a razão da quantidade de pedestres que não foram detectados pela quantidade total de *frames* e o FPPI é a razão da quantidade total de falsos positivos pela quantidade total de *frames*. Existem duas situações que são considerados falsos positivos. A primeira é quando o detector informa que há um pedestre em uma localização que não existe pedestre. A segunda é quando para um dado pedestre, o detector informa mais de uma localização próxima que estariam dentro da tolerância permitida. Apenas uma localização é considerada verdadeira e todas as outras são consideradas como falsos positivos.

É essencial esclarecer que a taxa de faltas (eixo das abscissas) e FPPI (eixo das ordenadas) são resultados do detector. O parâmetro do detector para obter esses dois resultados

é a confiança σ_{SVM} que possui valores no intervalo discreto de $[2; 0]$ com passos de 0,2. Para cada valor de confiança, a taxa de faltas e o FPPI são obtidos. Cada rótulo próximo ao ponto são valores de confiança na Figura 28.

Figura 28 – Exemplos da aplicação da métrica de avaliação FPPI por taxa de faltas em dois modelos de detectores hipotéticos.



Fonte: O autor.

O valor de confiança mais alto indica que o detector é mais preciso em afirmar que uma janela contém pedestres, mas menos pedestres são detectados e, consequentemente, uma alta taxa de faltas. Por outro lado, um FPPI mais alto é permitido com menor confiança, de modo que a taxa de faltas diminui, pois mais falsos positivos por imagem são aceitos e, portanto, os verdadeiros positivos têm uma chance maior de serem detectados. Deseja-se ter detectores com um baixo valor de *taxa de faltas* e um baixo valor de FPPI. Normalmente, os trabalhos de detecção de pedestres adotam 1 FPPI (ou seja, 10^0 FPPI) como um ponto de referência comum para comparar os resultados do detector.

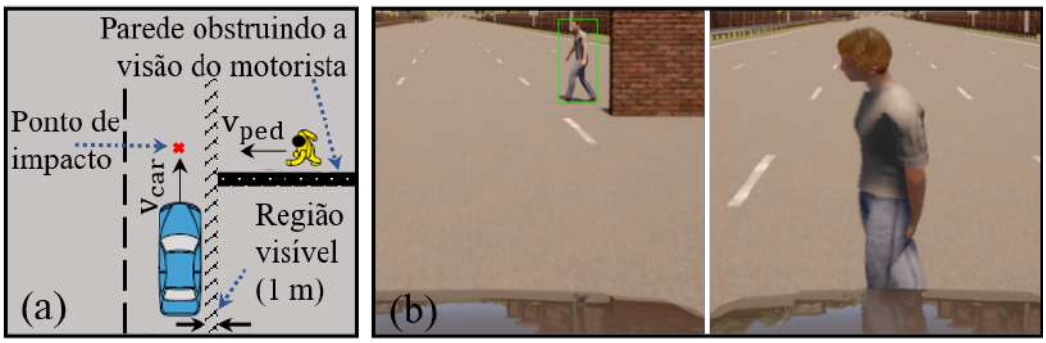
3.3.3 Base de dados 2

A base de dados 2 é direcionada exclusivamente para avaliação da tarefa de previsão de colisão. Até o momento da elaboração dessa tese não foi possível encontrar uma base de dados pública envolvendo cenários de colisão com pedestres com dados de câmeras estéreo e sensores inerciais registrados em tempo real. Desta forma, uma base de dados sintética foi criada neste trabalho utilizando o simulador CARLA na versão 0.9.7 (DOSOVITSKIY et al., 2017). O simulador CARLA é um simulador de código aberto implementado sobre o *Unreal Engine* 4 (SANDERS, 2016) que fornece dados sintéticos de cenários 3D para dar suporte ao desenvolvimento, treinamento, e validação de sistemas de direção autônoma. O simulador permite que o usuário instancie um conjunto de sensores do veículo que pode

incluir um número arbitrário de câmeras que coletam *frames* coloridos e de profundidade e permite leitura dos dados do sensor inercial do veículo que são fundamentais para avaliar o sistema completo de previsão de colisão proposto neste trabalho.

A base de dados foi construída a partir de um tipo de cenário definido por Jurecki e Stańczyk (2014), conforme mostrado na Figura 29(a). Este cenário envolve um pedestre movendo-se perpendicularmente em direção ao veículo e ocluído por uma parede. O carro atinge o pedestre em aproximadamente 50% da largura do veículo sem nenhuma ação de frenagem. Os parâmetros para a criação dos cenários são a velocidade do veículo (V_{car}) e o tempo para colisão (em inglês, *Time-To-Collision* - TTC). O parâmetro TTC, em segundos, é o quociente da distância do veículo ao pedestre (em metros) pela sua velocidade (em m/s) no momento do aparecimento do risco de colisão (JURECKI; STAŃCZYK, 2014).

Figura 29 – Cenário de avaliação a partir de Jurecki e Stańczyk (2014): (a) vista panorâmica (b) *frames* a partir do simulador CARLA.



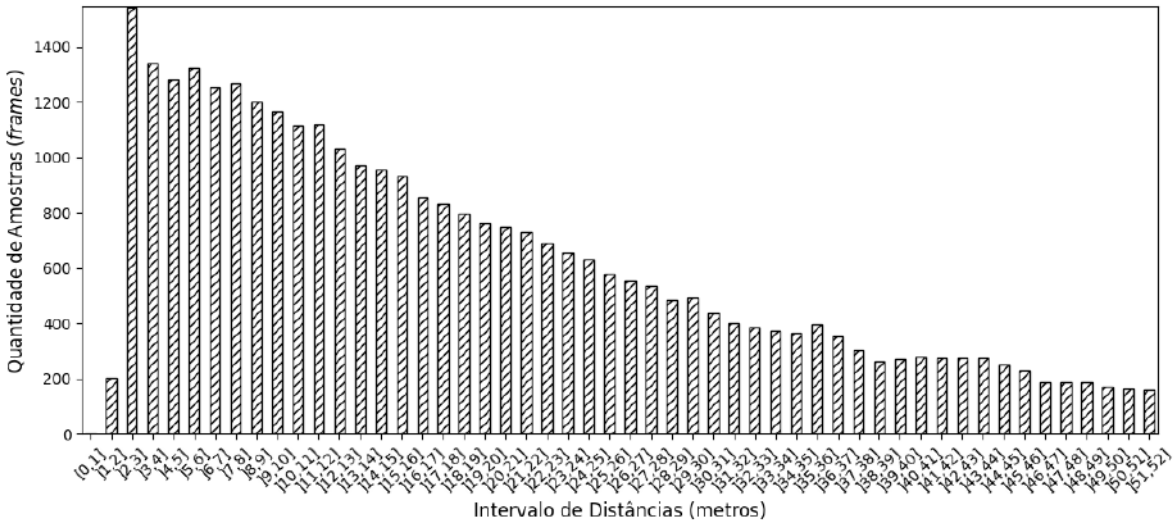
Fonte: O autor.

Seguindo Jurecki e Stańczyk (2014), os valores para V_{car} são 20, 30, 40, 50 e 60 km/h, e os valores de TTC são (0,6), (1), (1,4), (1,8), (2,2), (2,6) e (3). A taxa de amostragem dos *frames* é de 120 FPS e a resolução dos *frames* é de 1280 × 720 pixels. Os parâmetros de posição e movimentação do pedestre foram cuidadosamente ajustados para garantir que o tempo de colisão corresponda ao TTC de cada cenário quando o pedestre se torna visível e colide a aproximadamente 50% da largura do veículo.

Todas as combinações entre V_{car} e TTC foram realizadas, gerando 35 cenários para um tipo de pedestre. Essas combinações foram repetidas para mais dois pedestres diferentes, totalizando 105 cenários de testes criados. A base de treinamento foi criada sem envolver colisões. Foram introduzidos quatro tipos de pedestres diferentes da base de testes, veículos e pedestres em movimento e parados, e quatro ambientes do simulador. No total, foram construídos 100 cenários de treinamento. O *bounding box* de pedestres de cada *frame*, a velocidade do veículo e a taxa de guinada foram anotadas. Algumas capturas de tela do cenário CARLA são apresentadas na Figura 29(b).

A Figura 30 apresenta a distribuição da quantidade de *frames* disponíveis na base de dados de teste por intervalo de distância ao pedestre. Embora o foco desta base de dados

Figura 30 – Distribuição de *frames* por intervalo de distância para a base de dados sintética



Fonte: O autor.

seja a previsão de colisão com pedestres, avaliações de localização também podem ser realizadas.

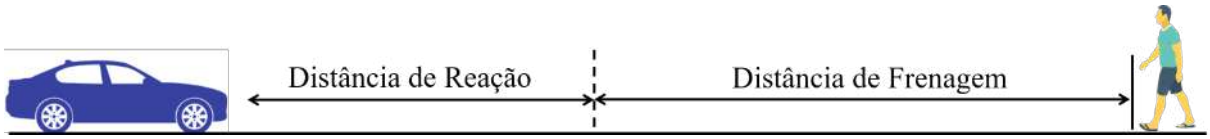
3.3.4 Métrica de Avaliação para Previsão de Colisão

A eficiência do sistema PCP é avaliada verificando a distância em que o sistema previu uma colisão pela primeira vez a partir do momento em que o pedestre apareceu. Nós denotamos esta métrica de previsão de colisão mais cedo. A distância segura, que dá suporte à avaliação de previsão, define a distância necessária para que o veículo inicia a frenagem e não colida com o pedestre. Se o sistema prever a colisão acima desta distância nós temos a garantia de que o veículo irá frear e não se chocará com o pedestre. Esta distância (CAFISO; GRAZIANO; PAPPALARDO, 2017) é definida como:

$$\text{distância}_{\text{segura}}(V_{\text{car}}, a_b, T_r) = \frac{V_{\text{car}}^2}{2\,a_b} + T_r\,V_{\text{car}} \quad \text{(metros)}, \tag{3.34}$$

onde V_{car} é a velocidade do carro medida em m/s, a_b é a desaceleração máxima do veículo medida em m/s^2, e T_r é o tempo de reação medido em segundos que pode ser o tempo para o motorista pisar no pedal do freio a partir de um aviso sonoro ou o tempo para o sistema de frenagem automática (em inglês, *Autonomous Emergency Braking* - AEB) acionar o sistema eletromecânico do freio. Nós adotamos o tempo para acionar sistema eletromecânico do freio como sendo de 100 ms a partir de Keller et al. (2011). A primeira componente da soma é a distância de frenagem a partir do momento que foi inciada a frenagem e a segunda componente é a distância necessária para reagir em um momento de risco de colisão. A Figura 31 ilustra esse conceito de distância segura. O sistema PCP deveria prever uma colisão acima da distância mínima.

Figura 31 – Distância segura para evitar colisão



Fonte: O autor.

O tempo médio de reação do motorista é em torno de 1 segundo (Park et al., 2017). Na realidade, este tempo de resposta varia em função da idade do condutor, da sua experiência e do seu estado. Por exemplo, um motorista cansado terá um tempo de resposta mais lento do que um motorista alerta. A desaceleração média em pista seca é em torno de $-10,0 m/s^2$ para veículos leves (KELLER et al., 2011). Esses valores foram usados para T_r e a_b. Os valores de distância segura para cada velocidade do veículo, partir da Equação 3.34, são apresentados na Tabela 2. Por exemplo, um veículo com velocidade de 60 km/h precisará de uma distância de 30,56 metros para que o motorista consiga evitar completamente uma colisão com pedestre e uma distância de 15,49 metros para que um sistema de automático de frenagem consiga evitar completamente uma colisão com pedestre. A importância de se ter sistemas PCP é para reduzir cada vez mais o tempo de reação da percepção uma vez que a distância de frenagem depende de aspectos físicos e eletrônicos do veículo.

Tabela 2 – Distância mínima para uma frenagem segura (metros) para cada valor de velocidade (km/h)

Velocidade	Distância de frenagem	Distância de reação do motorista	Distância de acionamento eletromecânico	Distância total de reação do motorista	Distância total para reação automática
20	1,54	5,55	0,55	7,09	2,09
30	3,47	8,33	0,83	11,80	4,30
40	6,17	11,11	1,10	17,28	7,27
50	9,64	13,89	1,41	23,53	11,04
60	13,89	16,67	1,60	30,56	15,49

Fonte: O autor.

Uma outra métrica importante é a quantidade de previsões de colisão que o sistema pode gerar desde o momento do aparecimento do pedestre até a colisão. Essa métrica traz um caráter de robustez para o sistema porque reduz a possibilidade de não detectar uma colisão. Além disso, essa métrica aumenta a confiança dos resultados obtidos pela métrica de previsão de colisão mais cedo porque garante que o sistema, se falhar ao detectar em um *frame* esperado, muito certamente no *frame* seguinte, conseguirá detectá-lo e a diferença em distância será pequena.

3.4 RESUMO E CONSIDERAÇÕES

Esse capítulo apresentou o sistema PCP proposto, detalhando as abordagens adotadas para cada etapa deste sistema. Este sistema é utilizado como base de avaliação das abordagens propostas nos capítulos seguintes para detecção, localização e previsão de colisão de pedestres. Este capítulo também apresentou a base de dados e as métricas para avaliação das abordagens propostas.

4 ABORDAGEM DE DETECÇÃO DE PEDESTRES PROPOSTA

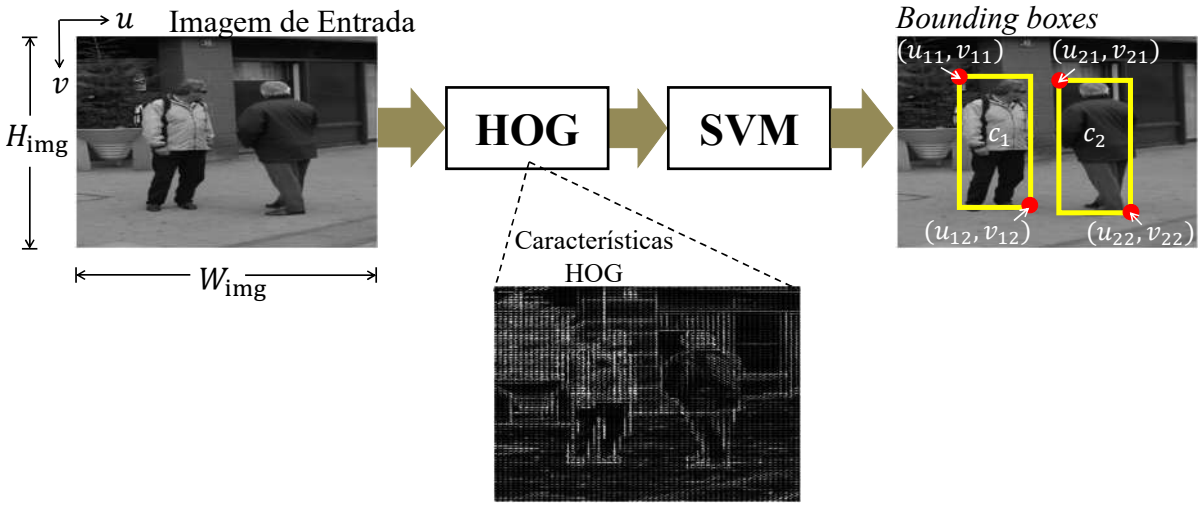
Este capítulo apresenta a abordagem proposta para a etapa de detecção de pedestre. Esta abordagem visa melhorar a eficiência do sistema PCP ao cobrir pedestres em grandes intervalos de distância e aumentando a distância para tomada de decisão segura.

Na Seção 4.1 nós descrevemos o detector base, baseado em HOG e SVM, detalhando e justificando as técnicas utilizadas em cada etapa deste detector. Em seguida, na Seção 4.2, nós apresentamos as melhorias propostas em relação ao detector base para aumentar o intervalo de distância de detecção. Na Seção 4.3, nós analisamos a precisão e desempenho do detector proposto e comparamos com outras abordagens de detecção existentes. Por fim, na Seção 4.4, descrevemos o resumo deste capítulo.

4.1 ARQUITETURA BASE: HOG/SVM

A visão geral do detector base é mostrada na Figura 36. O detector tem como objetivo encontrar pedestres e destacá-los por meio de *bounding boxes* que são os pontos extremos de um retângulo. Além disso, como parte da resposta do detector, existe um valor de confiança para cada *bounding box* que indica o nível de certeza do detector ao afirmar que existe um pedestre dentro dos limites do *bounding box*.

Figura 32 – Visão geral do detector de pedestres base. Os termos W, H e img significam, respectivamente, largura, altura e imagem. Um dado *bounding box* de índice k é formado pelos extremos (u_{k1}, v_{k1}) e (u_{k2}, v_{k2}) e por um valor de confiança c_k.



Fonte: O autor.

Como descrito nos Capítulos 1 e 2, os detectores baseados em aprendizagem profunda (GOODFELLOW; BENGIO; COURVILLE, 2016) são bastantes precisos pois possuem a capacidade de detectar pedestres em maiores variabilidades de poses e tamanhos. Porém tais

detectores exigem um alto custo de memória e possuem grande dependência de dados que impede de obtermos soluções de processamento eficientes em maiores resoluções (NGUYEN et al., 2019).

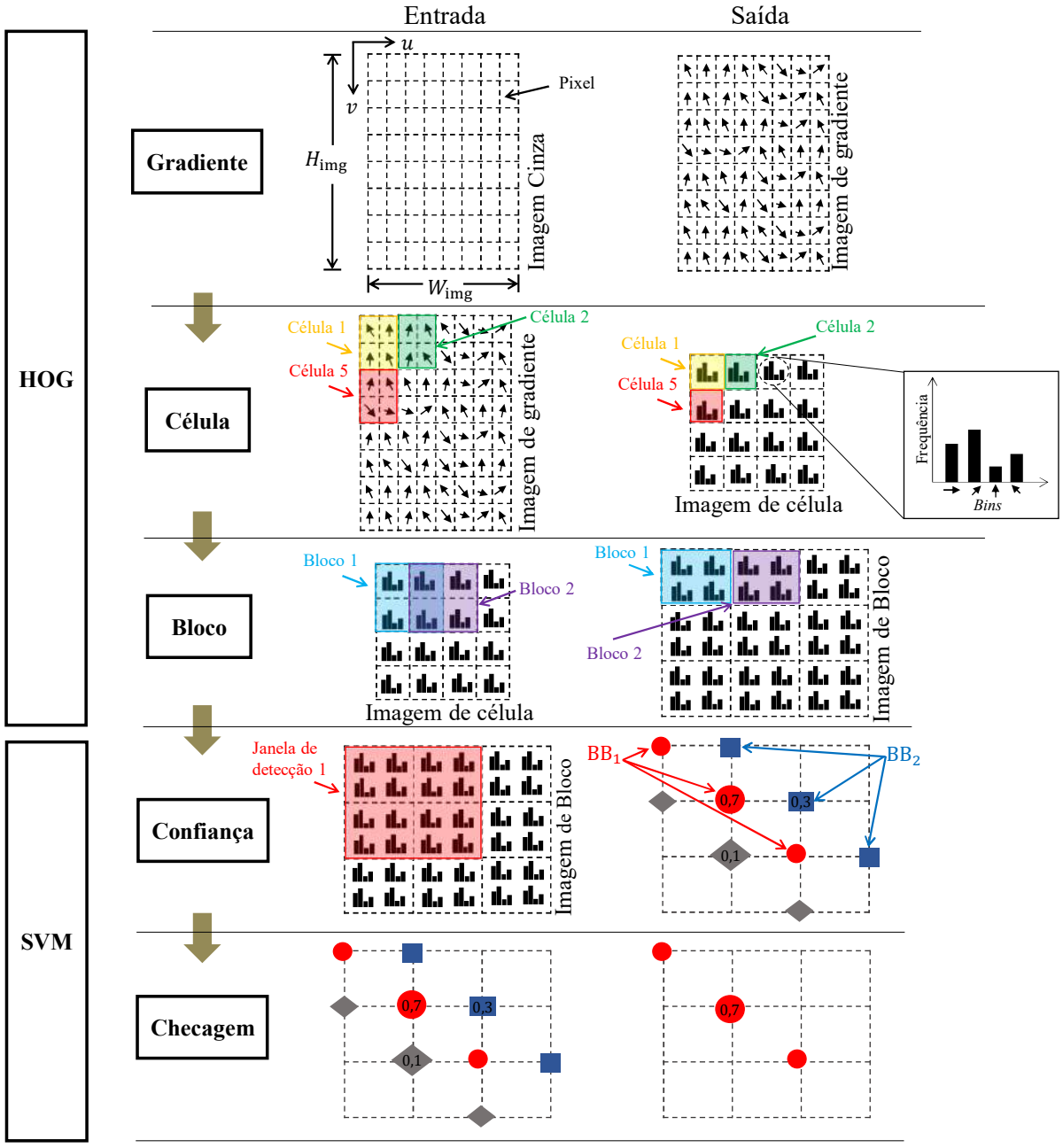
Por outro lado, os detectores baseados em aprendizagem de máquina, que combinam HOG e SVM oferecem um bom compromisso entre precisão e desempenho de processamento (SULEIMAN; ZHANG; SZE, 2017). Comparado com os detectores baseados em aprendizagem profunda, os detectores baseados em HOG e SVM têm um menor custo de memória e possuem menos dependência de dados. As características HOG facilitam a tarefa de classificação, destacando informações importantes que permitem distinguir o pedestre do plano de fundo, tais como regiões de bordas, que evidenciam a forma do pedestre, entre outras informações (SANGEETHA; DEEPA, 2017)(LUO; LIN, 2018)(DALAL; TRIGGS, 2005). Por serem tão promissores, estes detectores baseados em HOG e SVM têm sido adotados como um componente raiz em modelos mais complexos, tais como modelo de partes deformáveis que lida melhor com casos difíceis como oclusões, poses difíceis ou ângulos de visão raros (SULEIMAN; ZHANG; SZE, 2017).

A escolha de detector baseado em HOG e SVM não é somente pelo que este detector consegue oferecer até hoje, mas pelo que ele pode alcançar em termos de precisão em um contexto de veículos em estradas. Neste contexto, diversas informações úteis podem ser extraídas utilizando, por exemplo, correspondência estéreo (KELLER et al., 2011)(WU; ZHOU; SRIKANTHAN, 2016), que possibilitam melhorias na precisão da detecção sem prejudicar o seu desempenho de processamento. Outros extratores de características, como LBP, podem ser combinados com o HOG para aumentar a quantidade de características que destacam mais ainda o pedestres (BENENSON et al., 2014b)(YANG et al., 2021).

Outra razão para sua escolha, está no aspecto de processamento do SVM. Esta técnica configura como uma convolução que é um operador fundamental nos classificadores baseados em aprendizagem profunda. Dessa forma, ao propor soluções otimizadas em desempenho e recurso computacional para o SVM, estaremos contribuindo para termos detectores mais complexos que sejam escaláveis em desempenho e resolução de *frame*.

A Figura 33 apresenta uma visão geral do processamento das etapas de HOG e SVM. A etapa de gradiente calcula a magnitude e o ângulo de orientação do gradiente de cada pixel da imagem de entrada. A etapa de célula agrupa gradientes de pixels em célula e em cada célula calcula histogramas de orientações de gradiente. Cada histograma é formado por compartimentos (em inglês, *bins*) que contabilizam os gradientes em uma determinada direção. Um segundo agrupamento é realizado, formando os blocos, onde são aplicados funções de normalização de dados. Os dados normalizados é que definem as características HOG. O SVM é responsável por realizar a varredura de janela sobre as caraterísticas HOG, compor as características, e determinar se cada janela, chamada de janela de detecção, contém um pedestre ou não. Em cada janela de detecção, o SVM aplica o produto escalar entre o vetor de características HOG e o vetor de pesos SVM e

Figura 33 – Visão geral das etapas de HOG e SVM. As cores diferentes ajudam a diferenciar as diferentes células, blocos e janelas de detecção. Os termos W, H e img significam, respectivamente, largura, altura e imagem.



Fonte: O autor.

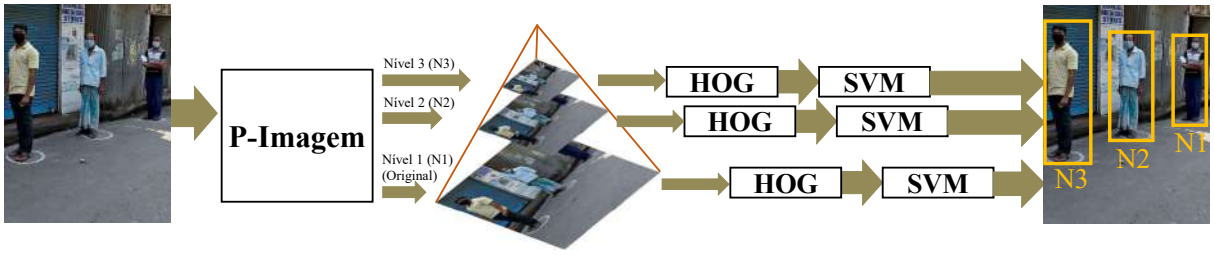
em seguida verifica se o resultado está acima de um limiar de confiança para afirmar que esta janela contém um pedestre. Nos Apêndices B e C nós detalhamos a estratégia de processamento, respectivamente, do HOG e SVM.

O conceito de pirâmide de imagens é fundamental para detectar pedestres que se apresentam com dimensões maiores que a dimensão da janela de detecção. Ela produz várias imagens redimensionadas de dimensões menores do que a imagem original. A Figura 34 mostra um exemplo do conceito de pirâmide de imagens (também chamada de P-

Imagem) na tarefa de detecção de pedestres. No nível 1 da pirâmide está a imagem original, enquanto nos outros níveis, as imagens são redimensionadas.

A técnica de redimensionamento de imagem é a interpolação bilinear (GRIBBON; BAILEY, 2004). Esta técnica calcula a intensidade dos pixels na imagem redimensionada através da soma ponderada da intensidade de quatro pixels mais próximos em torno de uma posição mapeada na imagem de origem. Como já descrito na Seção 2.1.1.1, esta técnica lida bem com efeitos de *aliasing* que dificultam o processo de detecção além de permitir facilmente redimensionar a imagem de origem para qualquer dimensão que se deseje. Suavização gaussiana pode ser incorporado para reduzir ainda mais os efeitos de *aliasing*. As equações desta técnica estão descritas no Apêndice D.

Figura 34 – Aplicação do conceito de pirâmide de imagens na detecção de pedestres.



Fonte: O autor.

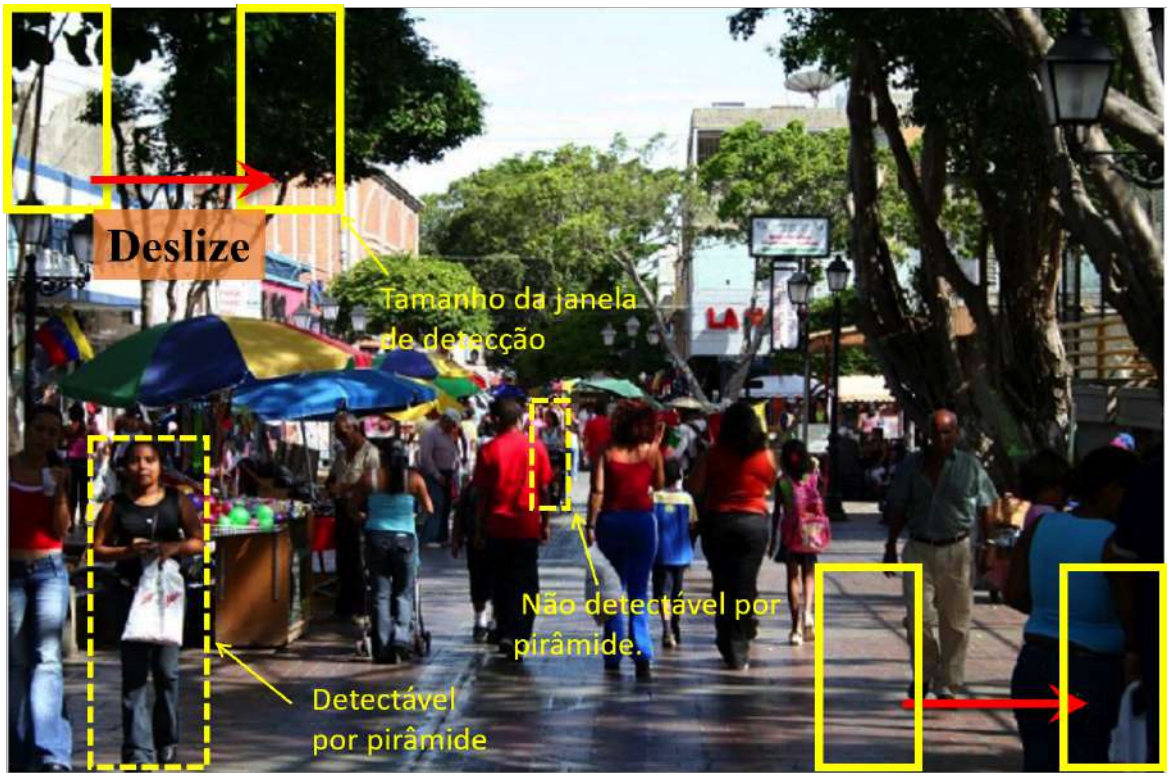
O fator de escala S_{pir} define a proporção do tamanho em ambos os lados da imagem a partir de um nível para o próximo, e a quantidade de níveis da pirâmide é determinada pela profundidade D_{pir}. Por exemplo, dado um *frame* de dimensão de 1280×720 pixels e $S_{\text{pir}} = 1,1$, o segundo nível da pirâmide terá a dimensão de 1163×654 pixels, o terceiro nível terá a dimensão de 1057×595 pixels.

4.2 ARQUITETURA PROPOSTA: JANELAS DE DIFERENTES DIMENSÕES

O detector base apresenta uma deficiência de apenas conseguir detectar pedestres que possuam dimensões em imagem próximas das dimensões de suas respectivas janelas de detecção. A pirâmide de imagens é uma abordagem que auxilia na detecção de pedestres que se apresentam com grandes dimensões na imagem reduzindo-os para dimensões que se ajustam a janela de detecção. Porém, pedestres que se apresentam na imagem com dimensões muito menores que a janela de detecção não são detectáveis, mesmo com o auxilio da pirâmide de imagens. A Figura 35 apresenta exemplos de pedestres que poderiam ser detectados com o auxilio da pirâmide e que não poderiam ser detectados. Embora as amostras de vários tamanhos sejam adicionadas na fase de treinamento supervisionado, o classificador SVM não converge para um modelo que captura todas essas variabilidades.

Por outro lado, janelas de detecção menores permitem a detecção de pedestres menores que geralmente estão mais distantes do veículo porque esses pedestres se encaixam melhor

Figura 35 – Demonstração da dificuldade da abordagem de janelas deslizantes de única dimensão. Pedestres menores que a janela de detecção tendem a não ser detectados.

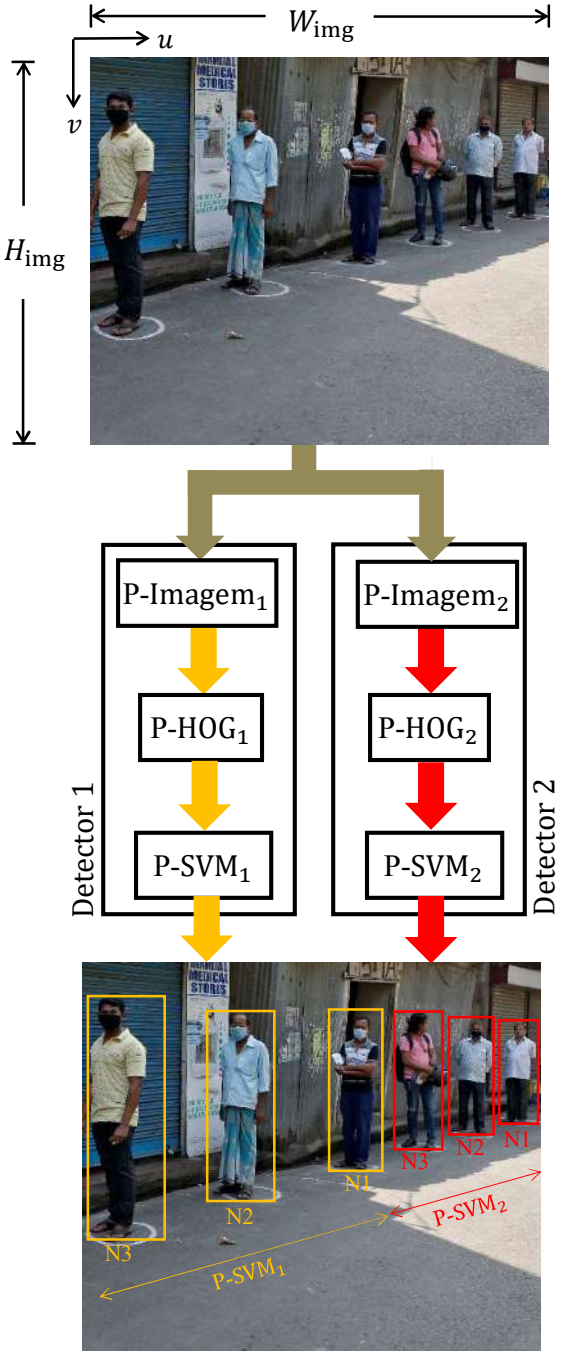


Fonte: O autor.

nessas janelas. A pirâmide de imagens ajuda a aumentar a capacidade de detecção, mas o aumento da profundidade da pirâmide acima de um limite faz com que o detector introduza muitos falsos positivos, além de não conseguir encontrar pedestres maiores. Isso ocorre porque janelas de detecção menores contêm menos informações de borda necessárias para generalizar a classificação de pedestres.

A solução proposta neste trabalho é ter múltiplos detectores, cada um com uma dimensão de janela de detecção diferente para ser capaz de localizar pedestres em intervalos de distância diferentes, como mostrado na Figura 36. As etapas de pirâmide HOG e SVM, chamadas de P-HOG e P-SVM, aplicam as técnicas HOG e SVM a cada imagem gerada pela pirâmide de imagens. Cada P-SVM$_k$ de índice k processa janelas de detecção de dimensão $W_{\text{P-SVM}_k} \times H_{\text{P-SVM}_k}$ pixels. Uma vez que a dimensão de cada P-SVM é diferente, os dados de pesos e *bias*, obtidos na fase de treinamento supervisionado, também serão diferentes. Cada P-HOG$_k$ processa células com dimensões $W_{\text{cel,k}} \times H_{\text{cel,k}}$ pixels, quantidade de *bins* definida por $Q_{bins,k}$ e blocos com dimensões $W_{\text{bloc,k}} \times H_{\text{bloc,k}}$ pixels. Cada P-Imagem$_k$ processa pirâmide de imagens com profundidade de $D_{\text{pir},k}$ e fator de escala de $S_{\text{pir},k}$.

Figura 36 – A arquitetura do detector de pedestres. Os termos W, H, P-Imagem, P-HOG e P-SVM significam, respectivamente, largura, altura, pirâmide de imagens, pirâmide HOG e pirâmide SVM.



Fonte: O autor.

No exemplo da Figura 36, o detector 1 possui classificadores em P-SVM$_1$ com uma dimensão de janela de detecção maior que a dimensão das janelas dos classificadores em P-SVM$_2$, do detector 2. Assim, o detector 2 pode encontrar pedestres em uma faixa de distância mais longe do que o detector 1. Além disso, ambos os detectores possuem profundidade de pirâmide 3, indicando que cada detector consegue encontrar pedestres mais próximos, que se apresentam com dois tamanhos maiores que a dimensão da janela

de detecção do respectivo detector.

4.3 RESULTADOS

Esta seção apresenta os resultados da solução de detecção proposta para as etapas de localização e previsão de colisão. Os resultados envolvem diversos experimentos, análises e comparações com outras abordagens usando as bases de dados definidas na Seção 3.3. Uma avaliação de desempenho também é realizada para dar suporte a análise de previsão de colisão.

4.3.1 Avaliação de Localização

Nós utilizamos a base de dados 1 para a avaliação da etapa de localização. Primeiramente, nós apresentamos as configurações dos detectores envolvidos nos experimentos, depois descrevemos a estratégia de avaliação geral e em seguida apresentamos os resultados dos experimentos e análises.

4.3.1.1 Definição dos Detectores

Nós avaliamos os detectores baseados em HOG e SVM com suporte de pirâmide de imagens e janelas de dimensões diferentes. Para explorar mais a comparações, nós avaliamos o detector baseado em *deep learning*, o YOLO (*You Only Look Once*) na versão 3 (YOLOv3) (REDMON; FARHADI, 2018). Nas próximas seções nós descrevemos as configurações destes detectores.

4.3.1.1.1 HOG e SVM

A fim de demonstrar melhorias na localização de pedestres próximos e distantes com o detector proposto, nós definimos duas unidades P-SVMs: P-SVM$_1$ com uma janela de detecção de dimensão 64 $\times$ 128 pixels e o P-SVM$_2$ com uma janela de detecção de dimensão de 48 $\times$ 96 pixels. As unidades P-SVM$_1$ e P-SVM$_2$ são responsáveis por detectar pedestres próximos e distantes, respectivamente. A partir destes dois P-SVMs nós implementamos em C++ três detectores, um com apenas o P-SVM$_1$, outro com apenas P-SVM$_2$ e o detector proposto com as duas unidades P-SVMs combinadas.

Para as implementações destes detectores nós utilizamos a função de detecção de objetos baseada em GPU a partir da biblioteca OpenCV 3.4. Esta função permite instanciar um detector em ponto flutuante com janelas de detecção de uma única dimensão tendo o extrator de características HOG e classificador SVM linear e a pirâmide de imagens usando interpolação linear. Para implementação dos detectores com janela de detecção de uma única dimensão nós instanciamos essa função apenas uma vez. Para a implementação do detector combinado nós instanciamos mais de uma vez esta função com parâmetros de

dimensão de janela diferentes, pirâmide de imagens e HOG e, no momento da inferência, nós juntamos os resultados de *bounding boxes* e confianças de ambas as instâncias em uma única lista para o processamento das etapas seguintes. A função do OpenCV é similar ao detector proposto neste trabalho com a diferença que a abordagem implementada para o HOG adiciona a técnica de interpolação trilinear que considera a espacialidade do *bin* dentro do bloco e a ponderação gaussiana que reduz o efeito de borda dentro do bloco.

Para treinar cada detector com a base de dados 1, nós criamos recortes positivos e negativos a partir de amostras de treinamento seguindo estratégias semelhantes ao proposto em Keller, Enzweiler e Gavrila (2011). Alguns exemplos são mostrados na Figura 37.

Figura 37 – Exemplos de amostras de recortes (a) contendo pedestres e (b) não contendo pedestres

Fonte: O autor.

A função da biblioteca OpenCV é usada para treinamento de cada P-SVM. A abordagem de *data augmentation* foi aplicada para cada recorte positivo usando espelhamento horizontal, rotação de imagem e alteração de contraste. Um algoritmo de *bootstrapping* foi aplicado para gerar os recortes negativos a partir dos *bounding boxes* erradamente detectados e retreinando o detector com a nova amostra. Os seguintes parâmetros de *data augmentation* foram cuidadosamente definidos para permitir a convergência da precisão do detector durante a fase de treinamento:

- Rotação (radiano): $\pm[0,1,\ 0,15,\ 0,20,\ 0,25]$
- Escala: $+[0,7,\ 0,75,\ 0,80,\ 0,85,\ 0,90]$
- Contraste: $+[0,7,\ 0,8,\ 1,2,\ 1,3]$

Os seguintes parâmetros são comuns a todos os três detectores avaliados. As dimensões da célula HOG e do bloco foram, respectivamente, 8×8 e 16×16 pixels. O parâmetro

stride $S_{\text{P-SVM}}$ de qualquer P-SVM foi definido como 8 pixels em ambas as direções. A quantidade de *bins* de histograma HOG foi definido como 9. O parâmetro de escala da pirâmide S_{pir} foi definido como 1,1.

4.3.1.1.2 YOLOv3

Nós seguimos os métodos dos autores Redmon e Farhadi (2018) para o treinamento da YOLOv3. Nós treinamos com imagens completas sem adição de amostras negativas. Nós empregamos as mesmas amostras de imagem usadas para treinar nossos P-SVMs. Nós usamos o *framework* de rede neural Darknet para treinamento e teste (REDMON, 2013–2016) que realiza treinamento em várias escalas, *data augmentation*, normalização, etc.

4.3.1.2 Estratégia de Avaliação

Para avaliação dos detectores nesta base de dados nós utilizamos a métrica de taxa de faltas por falso positivo por imagem (FPPI) a partir do resultado de localização do sistema PCP. Mais especificamente, nós consideramos as localizações ativas a partir da etapa de rastreamento como resultado para análise.

Nós consideramos a divisão de amostragem por grupos de distâncias, como descritos na Seção 3.3.2, para analisar cada detector. Um conjunto de avaliações foi realizado para demonstrar a importância de algumas etapas na melhoria da localização. Nós avaliamos a capacidade da pirâmide dos detectores com janelas detecção de dimensão única e a nossa solução que combina estes detectores. Em seguida, realizamos algumas avaliações de precisão com adição da filtragem de contexto no sistema PCP, proposto na Seção 3.1.2.

Com relação à configuração do sistema PCP, nós utilizamos os parâmetros descritos na Tabela 3. Para a etapa de correspondência estéreo, nós utilizamos o SGM-CUDA, descrito na Seção 3.2. Como nós estamos avaliando o impacto em precisão do detector na localização do pedestre, os parâmetros para a correspondência estéreo e estimativa de distâncias foram definidos buscando reduzir o erro destas duas etapas na localização. No Capítulo 6, mais especificamente, na Seção 6.3.1.6, nós avaliamos esta configuração e conseguimos uma taxa de faltas de menos de 2% e uma taxa de falsos positivos de aproximadamente 0%, independente do detector. Certamente estes parâmetros precisam ser redefinidos para uma resolução de *frames* diferente, a fim de garantir a mesma qualidade de localização.

Similar ao trabalho de Keller e Gavrila (2014), os pedestres envolvidos na avaliação da localização possuem alturas de 1,4 até 2,5 metros em pé. Similar ao trabalho de Srinivas (2016) pedestres com deslocamento vertical máximo de $\pm$ 0,5 metros em relação ao plano do chão são considerados. Para reduzir o risco de eliminar algum pedestre erroneamente a partir do sistema PCP nós definimos a restrição de intervalo um pouco maior de altura e deslocamento vertical da etapa de Filtragem Geométrica. A largura do pedestre varia

Tabela 3 – Parâmetros adotados para o sistema PCP

Etapa	Parâmetro	Descrição	Valor
Correspondência Estéreo	P_1	Penalidade	24
	P_2	Penalidade	96
	N_d	Intervalo de disparidades	128
	σ_{SGM}	Limiar de oclusão	20
Estimativa de Distâncias	H_{dist}	Altura do espaço de busca	5
	W_{dist}	Largura do espaço de busca	200
	σ_{dist}	Porcentagem de disparidades válidas	1%
	l_{dist}	Limiar para distâncias médias	2
Filtragem Geométrica	H_{min}	Altura mínima aceitável	1,1
	H_{max}	Altura máxima aceitável	2,8
	W_{min}	Largura mínima aceitável	0
	W_{max}	Largura máxima aceitável	2,5
	O_{min}	Deslocamento vertical mínimo	-1,5
	O_{max}	Deslocamento vertical máximo	1,5
Filtragem por Segmentação	ω_u	Limiar para filtragem de obstáculos	5
Rastreamento (FKE)	$\sigma_{\mathbf{x}}$	Ruído de processo	4,0
	σ_u	Ruído de medição lateral	6,15
	σ_d	Ruído de medição longitudinal	0,32
	P_0	Covariância inicial	0,01
Rastreamento (Associação e Gerenciamento de Rastros)	M_{max}	Quantidade de frames perdidos para remoção de rastro	2
	F_{min}	Quantidade de *frames* consecutivos detectados para considerar um rastro válido	3
Previsão de Trajetória	$T_{\text{prev,max}}$	Horizonte de previsão máximo (segundos)	4
	ΔT_{prev}	Intervalo de tempo de amostragem (segundos)	0,008

Fonte: O autor.

a medida em que ele se movimenta, que vai desde o momento em que os braços estão próximos ao corpo até o momento em que estão bem afastados. Visualmente, quando os braços estão próximos um do outro, a largura do pedestre se aproxima do zero e quando estão bem afastados a sua largura se aproxima da altura do pedestre. Dessa forma, para evitar a eliminação errada de detecção, nós consideramos valores para largura mínima e máxima, respectivamente em 0 e 2,5 metros.

O parâmetro ω_u de limiar para filtragem de obstáculos tem que ser pequeno o suficiente para remover o máximo de regiões de obstáculos, sem remover a região de estrada. A região de estrada é crítica para a estimativa da função de perfil da estrada. Nós definimos $\omega_u = 5$, um valor similar ao definido no trabalho de Wu (2016), que removeu uma boa parte dos obstáculos e ainda preservou a região de estrada. Certamente este parâmetro precisa ser

redefinido para uma resolução de *frames* diferente.

Valores acima de 1 para validação e invalidação de rastros, da etapa de Associação e Gerenciamento de Rastros, aumentam a confiança da detecção do pedestre. Seguindo os parâmetros do trabalho de Keller e Gavrila (2014), novos rastros são iniciados após 3 detecções consecutivas do mesmo pedestre e finalizados após 2 detecções sucessivas perdidas. O parâmetro de horizonte de previsão foi definido como 4 segundos para cobrir todos os cenários de colisão.

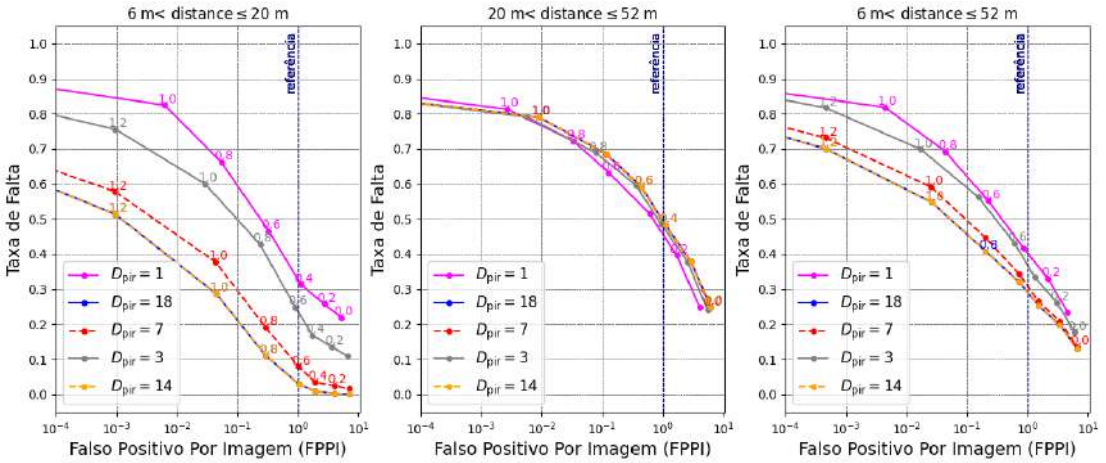
4.3.1.3 Experimentos, análises e comparações

A partir da definição dos detectores, do sistema PCP e da estratégia de avaliação, nós conduzimos os experimentos nas seções seguintes.

4.3.1.3.1 Pirâmide de Imagens e Janelas de Detecção de Dimensão Única

Este experimento avalia a qualidade dos detectores com janelas de uma única dimensão variando o parâmetro de profundidade da pirâmide. Uma vez que nós queremos, não somente analisar a melhoria da qualidade, mas também aspecto de convergência da detecção nós adotamos valores de profundidade que vão desde valores pequenos até um valor suficiente para perceber esse comportamento de convergência. Assim, os valores adotados de profundidade para avaliação são 1, 3, 7, 14 e 18. Neste experimento, nós consideramos a filtragem geométrica.

Figura 38 – Resultados da qualidade do detector baseado em HOG e SVM com janela de detecção de dimensão 64×128 pixels para cada grupo de distância. Para obter cada resultado de taxa de faltas e FPPI, os valores de confiança σ_{SVM} foram gerados no intervalo de $[2\,,\,0]$ com passos de 0,2.

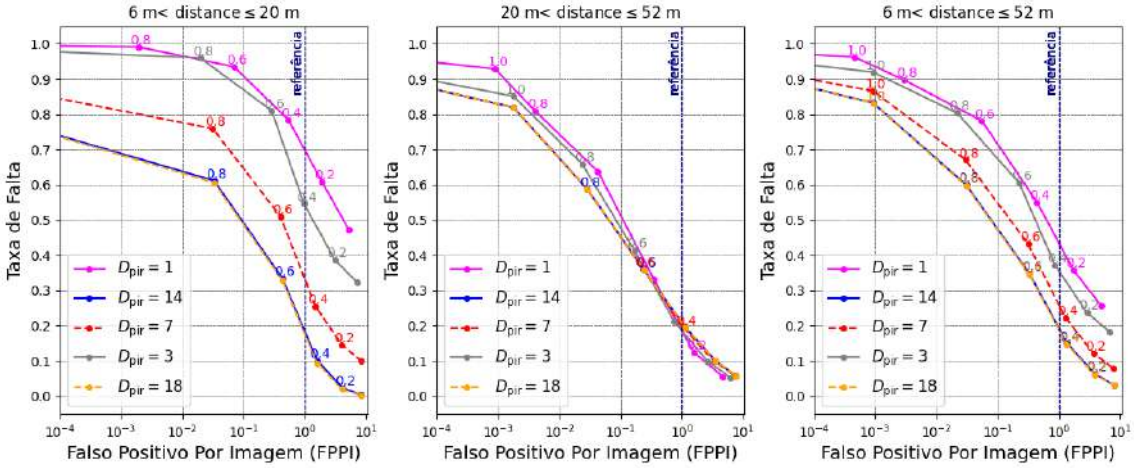


Fonte: O autor.

Nós utilizamos a base de dados 1 dividida em grupos de distâncias como descrito na Seção 3.3.1. O Grupo 1 contém os *frames* cuja distância está entre 6 e 20 metros, contabilizando 2085 *frames*; o Grupo 2 contém os *frames* cuja distância está entre 20 e

52 metros, contabilizando 2244 *frames*. O Grupo 3 contém os *frames* cuja distância está entre 6 e 52 metros contabilizando 4329 *frames*.

Figura 39 – Resultados de qualidade para o detector baseado em HOG e SVM com janela de detecção de dimensão 48 × 96 pixels para cada grupo de distância. Para obter cada resultado de taxa de faltas e FPPI, os valores de confiança σ_{SVM} foram gerados no intervalo de [2 , 0] com passos de 0,2.



Fonte: O autor.

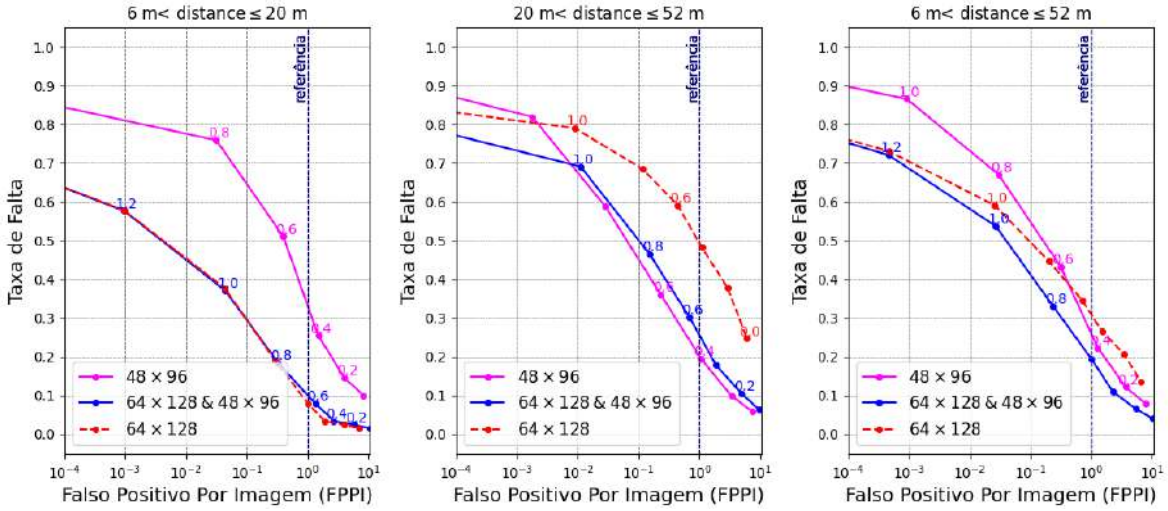
As Figuras 38 e 39 apresentam os resultados para o detector com janela de dimensão 64 × 128 e 48 × 96 pixels respectivamente. Nós podemos observar dois aspectos a partir dos resultados apresentados nestas Figuras. O primeiro aspecto é com relação ao aumento da profundidade de pirâmide que reduziu a taxa de faltas para o mesmo FPPI. Como pode ser observado na Figura 38, o detector com janela de detecção de 64 × 128 pixels, com a profundidade de 14, considerando o grupo 3 e o FPPI de referência, alcançou a taxa de falta de 30% enquanto com a profundidade de 1, o detector alcançou uma taxa de 40%. O mesmo comportamento acontece para o detector com janela de detecção de 48 × 96 pixels, como observado na Figura 39. Com a profundidade de 14, considerando o grupo 3 e o FPPI de referência, o detector alcançou a taxa de faltas de 28% enquanto com a profundidade de 1, o detector alcançou uma taxa de 43%.

O segundo aspecto é com relação à especificidade do detector para um dado intervalo de distância. Como pode ser observado na Figura 38 o detector com janela de detecção de 64 × 128 pixels, considerando o FPPI de referência, tem a melhor taxa de faltas de 3% para o grupo 1 e 45% para o grupo 2. Já o detector com janela de detecção de 48 × 96 pixels, tem a melhor taxa de faltas de 18% para o grupo 1 e 19% para o grupo 2, como observado na Figura 39. Mesmo que a profundidade de pirâmide seja aumentada, a taxa de faltas converge para o melhor resultado que é possível para uma dada dimensão de janela de detecção, como observado nessas duas figuras.

4.3.1.3.2 Janelas Combinadas

A partir dos resultados e análises da seção anterior, nós analisamos o detector 3 que combina os detectores 1 e 2. Nesse terceiro experimento nós comparamos os três detectores. A profundidade da janela dos três detectores foi definido como 7. Neste experimento, nós consideramos a filtragem geométrica. Nós avaliamos os detectores usando a divisão por grupos de distância.

Figura 40 – Resultados de qualidade para o detector baseado em HOG e SVM com janela de detecção de dimensão única de 64 × 128 e 48 × 96 pixels e com combinação de janelas de detecção. Para obter cada resultado de taxa de faltas e FPPI, os valores de confiança σ_{SVM} foram gerados no intervalo de [2 , 0] com passos de 0,2.



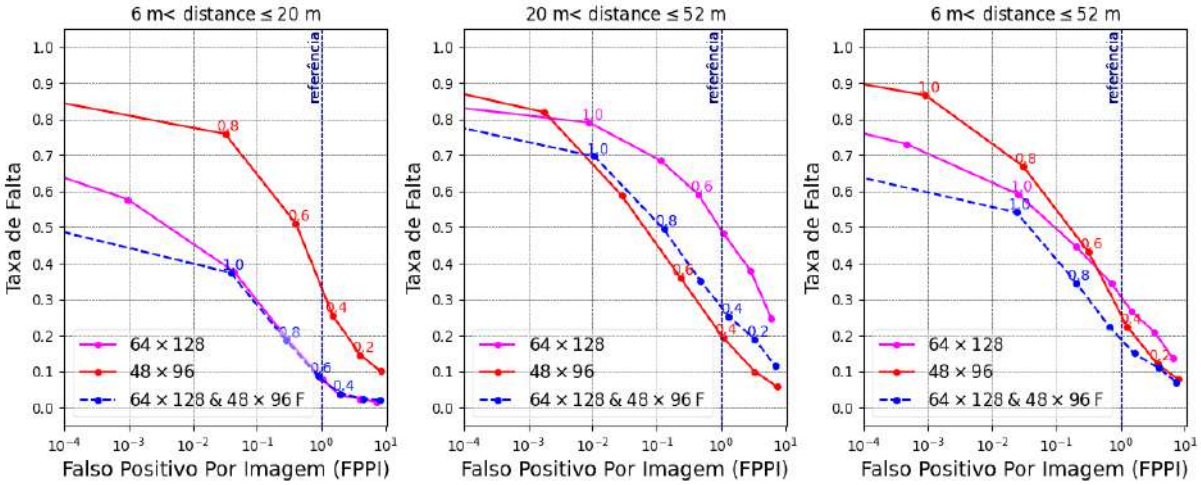
Fonte: O autor.

Como pode ser observado na Figura 40, o detector combinado obtém resultados próximos aos resultados dos detectores individuais nos intervalos de distância em que estes detectores são melhores. No grupo 1, o detector combinado obteve uma taxa de faltas para 1 FPPI de 10% contra 9% do detector com janela de dimensão 64 × 128 pixels. No grupo 2, o detector combinado obteve uma taxa de faltas para 1 FPPI de 25% contra 20% do detector com janela de dimensão 48 × 96 pixels. Como consequência, o detector combinado consegue ser melhor do que os outros dois detectores no grupo 3 que envolve todos os *frames* da base de dados. Para 1 FPPI, o detector combinado obteve uma taxa de 19% contra 25% obtidos pelos outros dois detectores. Essa redução em porcentagem em taxa de faltas equivale a um aumento de 259 *frames* detectados. Além disso, a combinação permitiu uma redução da área sob a curva em relação aos outros detectores.

É possível observar também que o detector combinado, gera mais falsos positivos do que os detectores de janelas únicas. Essa característica se justifica porque os falsos positivos dos detectores individuais são adicionados para o detector combinado. Então se os detectores de janelas únicas não forem bons, no sentido de ter valores baixos para FPPI e uma baixa taxa de faltas, a combinação também não será interessante. A combinação

apenas funciona bem quando os detectores são bons nos seus respectivos intervalos de distâncias.

Figura 41 – Resultados de qualidade para o detector baseado em HOG e SVM com janela de detecção de dimensão única de 64 × 128 e 48 × 96 pixels e com combinação de janelas de detecção usando a filtragem a partir da segmentação de estrada. Para obter cada resultado de taxa de faltas e FPPI, os valores de confiança σ_{SVM} foram gerados no intervalo de [2 , 0] com passos de 0,2.



Fonte: O autor.

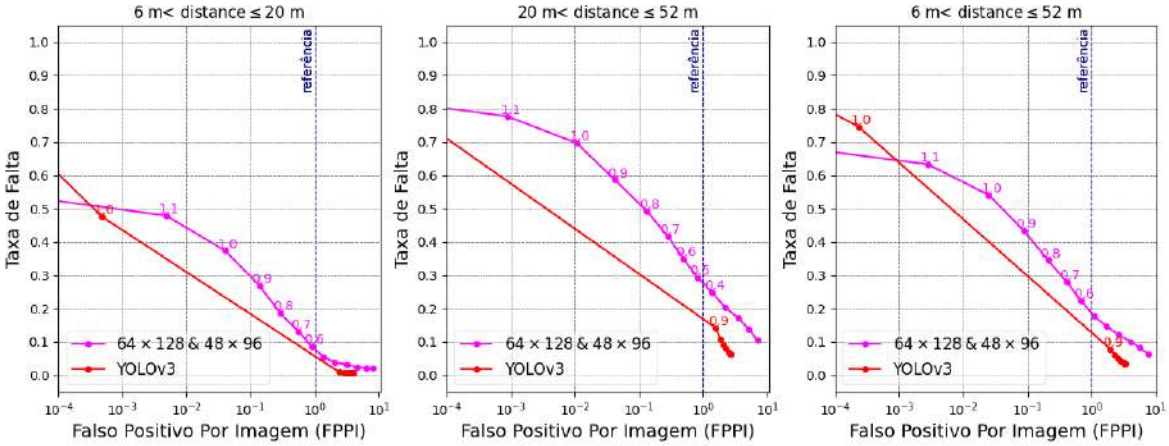
Assim, nós propomos a abordagem de filtragem por segmentação de estrada para redução de falsos positivos dos detectores, descrita na Seção 3.1.2.2. A Figura 41 mostra o resultado do uso desse tipo de filtragem no detector combinado. Como pode ser observado foi possível reduzir ainda mais a curva comparado com o detector combinado sem esta filtragem. As filtragens propostas neste trabalho podem ser facilmente implementadas em plataformas embarcadas, uma vez tendo os resultados do mapa de disparidade e do detector. Certamente estas filtragens são opções bem atraentes para compor o detector proposto. Uma outra sugestão de estratégia é estimar o melhor intervalo de distância para cada detector a partir do perfil de estrada já desenvolvido neste trabalho para reduzir ainda mais os falsos positivos do detector combinado.

4.3.1.3.3 Comparação de detectores: YOLOv3 vs Janelas Combinadas

Nós comparamos o nosso detector de janelas combinadas com o detector YOLOv3. A profundidade da janela do detector combinado foi definido como 7. Neste experimento, nós consideramos a filtragem geométrica.

Como pode ser observado na Figura 42, o detector YOLOv3, de fato, consegue obter resultados melhores que o nosso detector combinado principalmente para distâncias maiores devido a sua capacidade de capturar grandes variabilidades de pedestres. Por outro lado, para distância menores os resultados de ambos detectores são parecidos. Outras

Figura 42 – Resultados de qualidade para o detector baseado em HOG e SVM com combinação de janelas de detecção de dimensões de 64 × 128 e 48 × 96 pixels e o detector YOLOv3. Para obter cada resultado de taxa de faltas e FPPI, os valores de confiança σ_{SVM} foram gerados no intervalo de [2 , 0] com passos de 0,1.



Fonte: O autor.

formas de melhorar a taxa de faltas a partir do detector proposto é unir o HOG com outros extratores de características. Essa abordagem aumenta o número de características de pedestres salientes permitindo melhorar a capacidade de classificação (BENENSON et al., 2014b). Padrões binários locais (LBP) (YANG et al., 2021), características de disparidade estéreo (KELLER; ENZWEILER; GAVRILA, 2011), redes neurais convolucionais (CNN) (KALAKE et al., 2022) são exemplos de extratores de características.

4.3.2 Avaliação de Desempenho de Processamento

Nós estimamos o tempo de processamento do sistema PCP com o nosso detector e o detector YOLOv3 para processar imagens com resoluções de 1176 × 640 e 1280 × 720 pixels, as mesmas resoluções da base de dados 1 e 2, respectivamente. O ambiente de execução foi a plataforma descrita na Seção 3.2, que envolve um computador com processador de propósito geral (GPP), core I5-9400F de 2,90GHz, com 16 GB de memória RAM, e uma GPU RTX 2070 de 8 GB de memória.

A detecção de pedestres e correspondência estéreo são as etapas que exigem os maiores custos computacionais para processamento do sistema PCP. Como descrito na Seção 4.3.1.1.1, usamos uma função pronta implementada em plataforma GPU fornecida pela biblioteca OpenCV para executar o detector proposto. Nós estimamos o tempo médio de relógio para cada sistema PCP processar 1.000 *frames*.

Nosso detector, leva em média 15,6 ms para processar um *frame*, enquanto o processamento do YOLOv3 leva em média 60,8 ms. Além disso, como descrito na Seção 3.2, nós adaptamos a implementação SGM-CUDA para suportar a detecção de disparidades ocluídas. O processamento de correspondência estéreo leva 10,4 ms, em média. Ao somar os tempos de todas as etapas de processamento, o sistema PCP com o nosso detector alcan-

çou um desempenho de 40 e 30 FPS para as resoluções de 1176 × 640 e 1280 × 720 pixels, respectivamente. Com o detector YOLOv3, o sistema PCP alcançou um desempenho de 15 e 12 FPS, respectivamente.

4.3.3 Avaliação de Previsão de Colisão

Nós utilizamos a base de dados 2, descrita na Seção 3.3.3, para avaliação da tarefa de previsão de colisão. Primeiramente, nós apresentamos as configurações dos detectores envolvidos nos experimentos e estratégia de análise e em seguida apresentamos os experimentos, resultados e análises.

4.3.3.1 Definição dos Detectores

As abordagens, parâmetros e implementações dos detectores são similares aos adotados para avaliação da base de dados 1. Algumas diferenças estão nos parâmetros das unidades P-SVMs. Nós definimos P-SVM$_1$ com uma janela de detecção de dimensão 48 × 96 pixels e P-SVM$_2$ com uma janela de detecção de dimensão de 24 × 48 pixels. O limiar de classificação foi definido experimentalmente como 0,7 levando em consideração o compromisso entre um baixo FPPI e baixa taxa de faltas em localização.

Os seguintes parâmetros são comuns a todos os três detectores avaliados. As dimensões da célula e bloco HOG foram, respectivamente, 4 × 4 e 8 × 8 pixels. O parâmetro *stride* $S_{\text{P-SVM}}$ de qualquer P-SVM foi definido como sendo 4 pixels em ambas as direções. A quantidade de *bins* de histograma HOG foi definida como 9. O parâmetro de escala e profundidade da pirâmide foram definidos, respectivamente como 1,1 e 7.

A estratégia de treinamento é a mesma descrita nas seções 4.3.1.1.1 e 4.3.1.1.2. As amostras de treinamento sintéticas foram usadas para treinar cada P-SVM e a YOLOv3.

4.3.3.2 Estratégia de Avaliação

Para avaliação de previsão de colisão nós utilizamos a métrica de previsão de colisão mais cedo, descrita na Seção 3.3.4. Diferentemente das avaliações que foram conduzidas para base de dados 1, na base de dados 2 nós consideramos também a taxa de processamento dos sistemas PCPs com detectores diferentes. Como a taxa de amostragem da base de dados 2 é de 120 FPS, nós definimos o intervalo de captura de *frames* que corresponde a taxa real do sistema PCP. Os intervalos para o sistema com o nosso detector e o YOLOv3 foram 4 e 10 *frames*, respectivamente.

Os parâmetros do sistema PCP foram os mesmos definidos na Seção 4.3.1.2. A diferença está no parâmetro de ruído do processo $\sigma_{\mathbf{x}}$ do filtro EKF. Nós executamos um método de otimização definido em Schneider e Gavrila (2013) com as amostras de treinamento a partir da base de treinamento da base de dados 2 para encontrar o melhor valor

de $\sigma_{\mathbf{x}}$ para o sistema PCP. Para o sistema com o nosso detector e o detector YOLOv3 os valores estimados de $\sigma_{\mathbf{x}}$ foram 1,12 e 7,2 respectivamente.

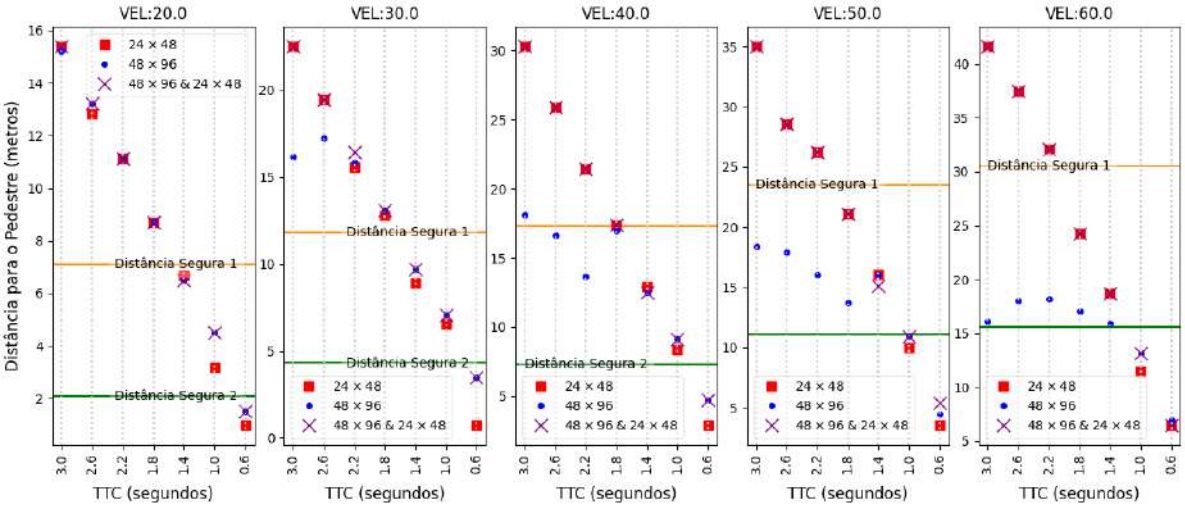
4.3.3.3 Experimentos, análises e comparações

A partir da definição dos detectores, do sistema PCP e da estratégia de avaliação, nós conduzimos os experimentos, cujos resultados serão descritos na seções seguintes.

4.3.3.3.1 Janelas de Detecção de Dimensão Única e Combinadas

Esse experimento envolveu três detectores, os de janelas de dimensão única e o detector com janelas de detecção combinadas. A Figura 43 apresenta os resultados destes três detectores. As distâncias seguras 1 e 2, detalhadas na Seção 3.3.4, definem as distâncias mínimas para frenagem segura sem colisão, respectivamente, do motorista se for avisado pelo sistema PCP e do sistema frenagem automática sendo acionado pelo sistema PCP. Os valores e conceitos sobre estas distâncias estão detalhados na Seção 3.3.4.

Figura 43 – Resultados a partir da métrica de previsão de colisão mais cedo para os detectores baseados em HOG e SVM com janela de detecção de dimensão única de 64 × 128 e 48 × 96 pixels e dimensões combinadas. Os termos Distância Segura 1 e 2 definem as distâncias mínimas para frenagem segura sem colisão, respectivamente, do motorista e do sistema de frenagem automática.



Fonte: O autor.

Nós podemos observar alguns aspectos nesta Figura já notados nos experimentos com a base de dados 1 que envolve as distâncias exigidas para previsão de colisão. Veículos cada vez mais rápidos exigem que a previsão de colisão detecte a colisão envolvendo pedestres mais distantes porque os veículos percorrem uma distância maior em um mesmo TTC. Além disso, pedestres mais distantes se apresentam no *frame* cada vez menores. Nós observamos que o detector com janela maior (48 × 96 pixels), a partir de uma certa velocidade de veículo, não consegue mais acompanhar a tendência de crescimento na

distância de previsão. Com valores maiores de TTC esta deficiência se agrava. Por outro lado, o detector de janela menor (24 × 48 pixels) consegue acompanhar melhor esta tendência e detectar pedestre mais distantes.

De maneira oposta, os veículos cada vez mais lentos exigem distâncias menores para previsão de colisão. Para distâncias menores, os pedestres se apresentam no *frame* com dimensões maiores. Além da velocidade do veículo, os cenários com menores TTCs envolvem pedestres mais perto. O detector de maior dimensão (48 × 96 pixels) consegue prever colisões em cenários de distâncias menores, como por exemplo envolvendo os TTCs de 0,6 e 1,0 segundo e as velocidades de 20 até 50km/h, garantindo uma distância para tomada de decisão maior do que o detector de janela menor (24 × 48 pixels).

Quando analisamos a resposta do detector combinado, podemos observar que este detector tomou proveito dos melhores resultados dos dois detectores individuais. Com o detector combinado é possível evitar colisões através do aviso ao motorista. Por exemplo, veículos com velocidade de 50km/h e 60km/h e TTC de (2,2), (2,6) e (3) segundos o sistema PCP com o detector combinado previu a colisão em distâncias acima da distância segura, o que significa que o sistema garantiu completamente a colisão. Esse tipo de análise deixa claro a importância das combinações de dimensões de janelas para aumentar a distância para tomada de decisão em vários TTCs e velocidades de veículo e, consequentemente, aumentar a quantidade de cenários que é possível evitar uma colisão.

4.3.3.3.2 Comparação de detectores: YOLOv3 vs Janelas Combinadas

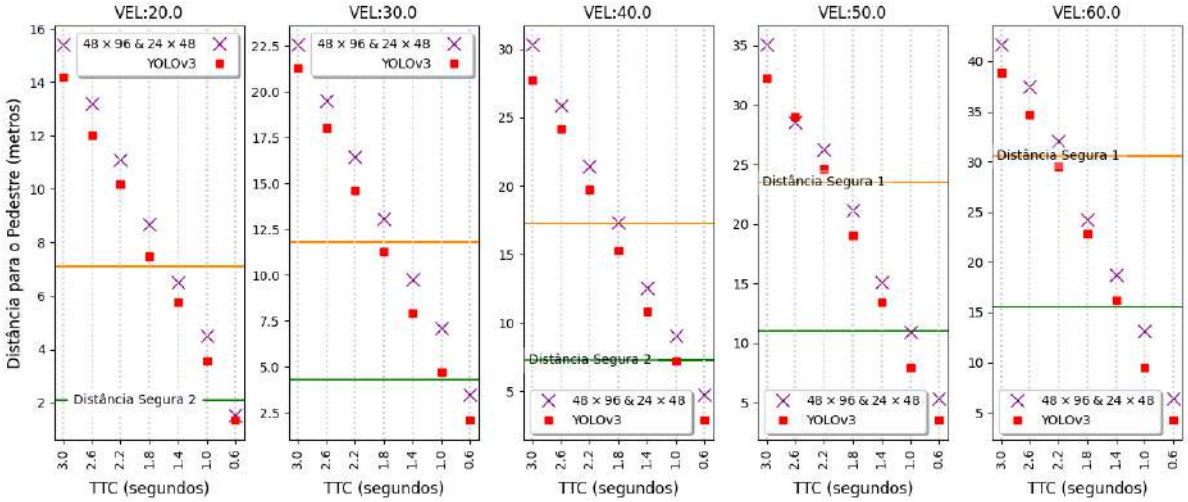
Nós comparamos o nosso detector de janelas combinadas com o detector YOLOv3. Além da análise usando a métrica de previsão mais cedo nós também utilizamos a métrica de quantidade de previsão de colisão, descrita na Seção 3.3.4. Essa métrica contabiliza a quantidade de previsões realizadas pelo sistema desde o aparecimento do pedestre até o momento da colisão.

Como podemos observar na Figura 44, o sistema PCP com detector combinado permitiu um aumento na distância para tomada de decisão em praticamente todos os cenários de teste. Em alguns cenários, como por exemplo, velocidade de 30 km/h e TTC de 1,8 segundos, o uso do detector combinado garantiu a distância segura 1 que evita, de fato, a colisão se o motorista for avisado para que ele tome a decisão de frear.

Esse ganho em distância se deve ao fato de o detector combinado possuir uma taxa de processamento 2,5 vezes maior, que permitiu um processamento de mais informações de localização e uma convergência do filtro de Kalman estendido (EKF) para encontrar velocidade real do pedestre de forma mais rápida.

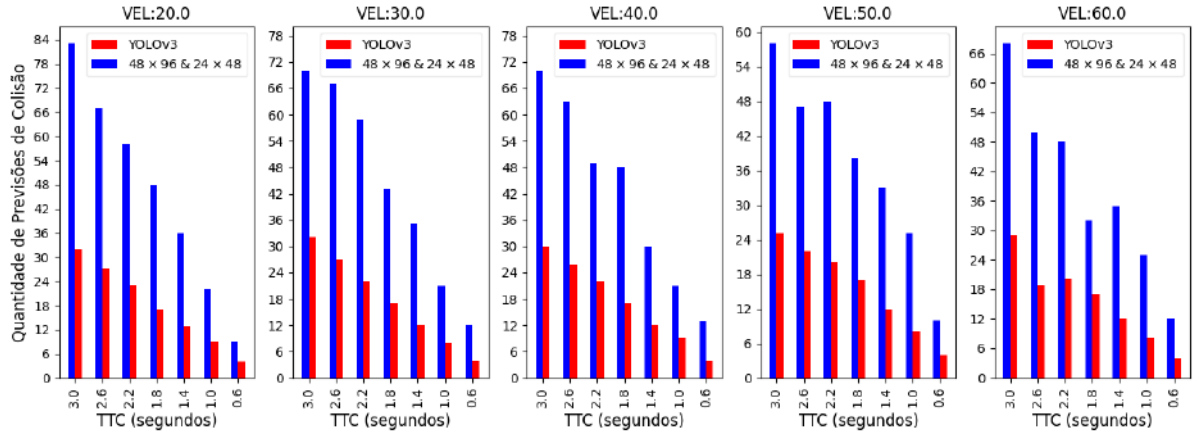
Outro importante resultado é com relação a quantidade de previsões de colisão, apresentado na Figura 45. O sistema PCP com o detector combinado gerou uma quantidade de previsão consideravelmente maior do que o sistema com o detector YOLOv3. Esse resultado indica que o nosso sistema tem uma chance maior de prever uma colisão antes que

Figura 44 – Resultado de qualidade a partir da métrica de previsão de colisão mais cedo para o detector baseado em HOG e SVM com janelas de detecção de dimensões combinadas e o detector YOLOv3. Os termos Distância Segura 1 e 2 definem as distâncias mínimas para frenagem segura sem colisão, respectivamente, do motorista e do sistema de frenagem automática.



Fonte: O autor.

Figura 45 – Resultados de qualidade a partir da métrica de quantidade de previsão de colisão para o detector baseado em HOG e SVM de dimensões combinadas e o detector YOLOv3



Fonte: O autor.

a colisão aconteça. Além disso, capacidade de processar mais *frames* aumenta a confiança dos resultados de previsão mais cedo pois reduz a distância de erro por falha na previsão de colisão.

4.4 RESUMO

Esse capítulo apresentou o detector baseado em HOG e SVM com combinação de janelas de detecção de múltiplas dimensões. Resultados mostraram que a combinação permitiu aumentar a precisão na localização e eficiência na previsão de colisão em um amplo intervalo de distâncias. Além disso, esse detector conseguiu resultado em precisão superior comparado com o detector YOLOv3 para tarefa de previsão de colisão na base de dados

sintética. A melhoria na eficiência da previsão de colisão devido ao ganho em desempenho de processamento do sistema PCP usando o detector combinado motivou o desenvolvimento de uma solução de processamento de alto desempenho baseado em FPGA para esse detector que será detalhado no próximo capítulo.

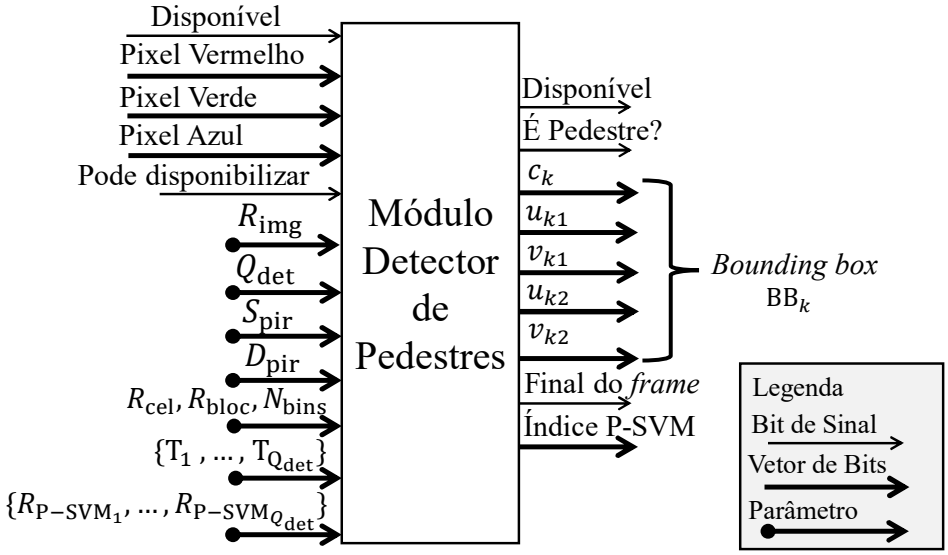
5 ARQUITETURA DE HW PARA DETECÇÃO DE PEDESTRES

Neste capítulo nós detalhamos a arquitetura de hardware da abordagem de detecção de pedestres proposto no Capítulo 4. Na Seção 5.1 nós descrevemos arquitetura geral do hardware. Na Seção 5.2 nós detalhamos os componentes chave deste detector, descrevemos os desafios para garantir o processamento em fluxo contínuo, como a arquitetura de cada componente foi concebida para lidar com tais desafios. Estratégias para redução de recurso de hardware são apresentadas também neste capítulo. Por fim, na Seção 5.4, nós analisamos a arquitetura do ponto de vista de precisão, ocupação de recursos, dissipação de potência, desempenho e comparamos com outras abordagens em hardware de detecção de pedestres existentes.

5.1 ARQUITETURA GERAL DE HARDWARE

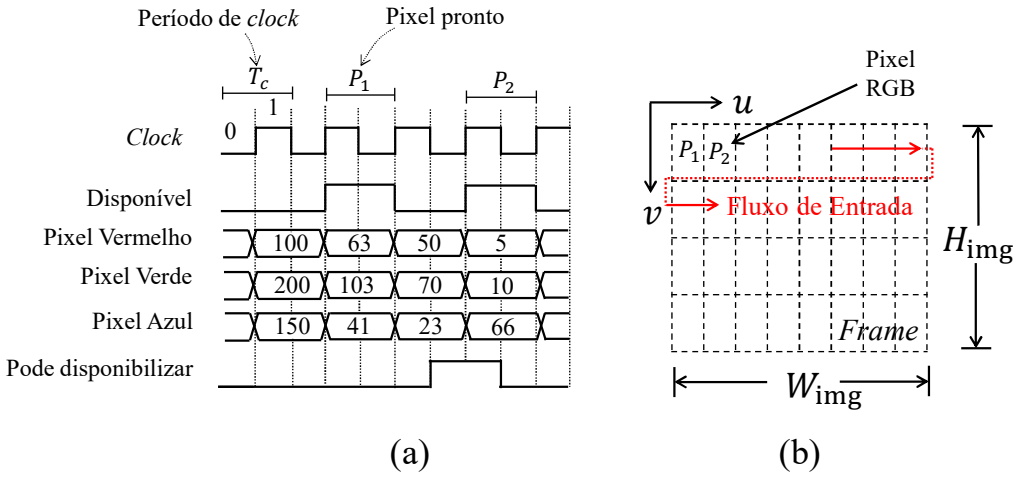
Como parte da descrição da arquitetura geral, nós apresentamos, na Figura 46, a interface do detector de pedestre. Nós apresentamos também, nas Figuras 47 e 48, o fluxo de entrada de pixels e saída dos *bounding boxes* do detector.

Figura 46 – Descrição da interface de entrada e saída do detector de pedestres. O parâmetro Q_{det} define a quantidade de classificadores P-SVM diferentes. Os parâmetros R_{img}, R_{cel}, R_{bloc}, $R_{\text{P-SVM}_h}$ significam, respectivamente, a resolução em pixels do *frame* de entrada, da célula HOG, do bloco HOG e da janela de detecção do classificador P-SVM$_h$. O parâmetro T_h indica o conjunto de dados de treinamento de um dado classificador P-SVM$_h$. Os parâmetros D_{pir} e S_{pir} definem, respectivamente, a profundidade e escala da pirâmide. Os pares u_{k1}, v_{k1} e u_{k2}, v_{k2}, e o termo c_k descrevem, respectivamente os dois extremos do retângulo e o valor de confiança do *bounding box* BB$_k$.



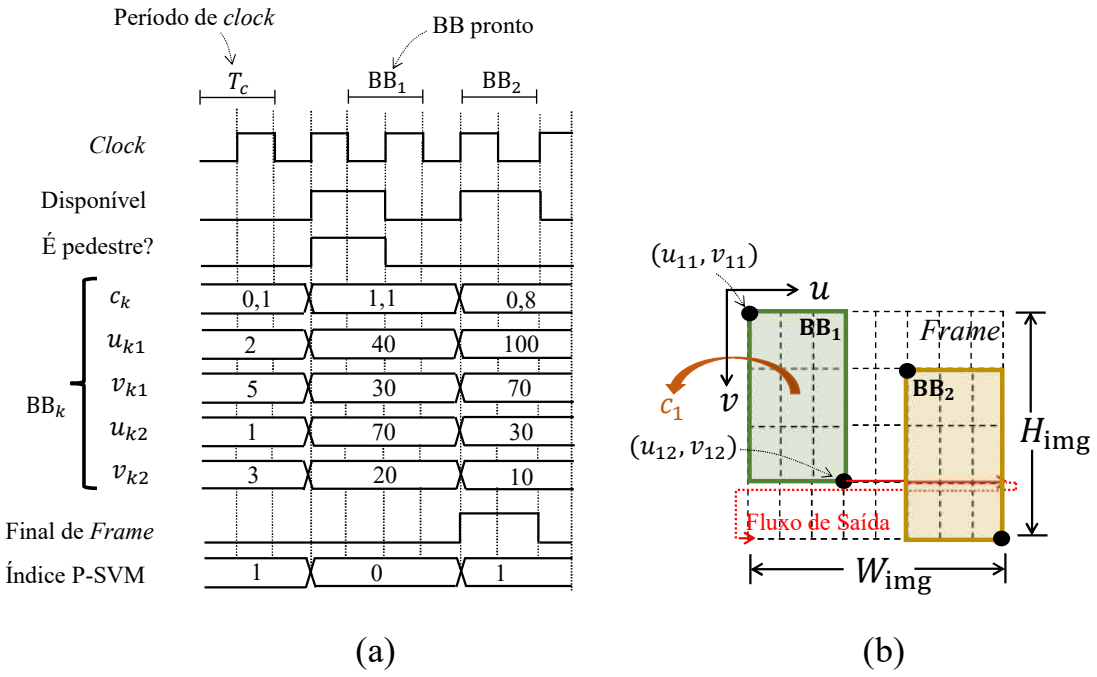
Fonte: O autor.

Figura 47 – Fluxo de entrada de pixels do detector de pedestres: (a) Visão em forma de onda (b) Visão em forma de varredura de pixels na imagem. Os termos W_{img} e H_{img} significam, respectivamente, largura e altura da imagem. O detector processa os valores de um determinado pixel quando o sinal de disponível estiver em nível lógico 1. Por exemplo, os valores do pixel P_1 que são processados são 63, 103 e 41. O fluxo de entrada dos pixels precisa obedecer a ordem de varredura de imagem da esquerda para direita e de cima para baixo.



Fonte: O autor.

Figura 48 – Fluxo de saída de *bounding boxes* (BB) do detector de pedestres: (a) Visão em forma de onda (b) Visão em forma de varredura de pixels em imagem. Os termos W_{img} e H_{img} significam, respectivamente, largura e altura da imagem. Sempre que o sinal de disponível estiver em nível lógico 1, em um ciclo de *clock*, então, nesse ciclo, os dados do BB de índice k, ou seja, c_k,u_{k1},v_{k1},u_{k2} e v_{k2}, estão prontos para serem lidos. Além desses dados, a informação do índice do módulo P-SVM que gerou o *bounding box* também é disponibilizado. O fluxo de saída dos *bounding boxes* obedece a ordem do fluxo de entrada, ou seja, da esquerda para direita e de cima para baixo. Os dados de confiança são de representação de ponto fixo.



Fonte: O autor.

O detector recebe pixels no espaço de cor RGB (espaço formado pelos canais *Red*, *Green* e *Blue*) e retorna os *bounding boxes* (BB), ou seja, as duas posições no espaço de imagem dos extremos de um retângulo e o seu valor de confiança. Juntamente com cada BB é retornado o índice do conjunto P-SVM de classificadores que o gerou. Diversos parâmetros são necessários para geração do detector tais como: resolução de *frame*; quantidade de conjunto de classificadores P-SVM; profundidade e escala da pirâmide de imagens; resolução de célula e bloco da técnica HOG; e o conjunto de dados de treinamento de cada conjunto de classificadores P-SVM.

A entrada dos pixels obedece a um fluxo tradicional de varredura de *frame*, da esquerda para direita e de cima para baixo. A entrega dos *bounding boxes* também segue esse fluxo. A arquitetura foi concebida para suportar qualquer vazão de entrada de pixels, ou seja, pode entrar pixel ciclo após ciclo ou, em intervalos de ciclos espaçados. Além disso, esta arquitetura funciona em modo escravo, sempre aguardando a fonte externa informar que possui um pixel disponível. Toda vez que o sinal de disponível estiver no nível lógico 1, então existe um pixel de entrada disponível e o detector prontamente já lê os três canais deste pixel e processa-o. Essa característica torna o detector facilmente integrável em plataformas que possuem câmeras de hardware de pixel serial conectadas diretamente ao detector por pinos GPIO e em plataformas heterogêneas HW/SW (SARKAR; BHAIRANNAWAR; KB, 2021).

O processamento e entrega de resultados seguem o ritmo da entrada dos dados. Sempre que existem dados de entrada para processamento de um dado *bounding box*, o detector faz o processamento e, se este *bounding box* tiver sido classificado como contendo pedestre, então este será disponibilizado na saída. Nós reiteramos que o detector só disponibiliza *bounding boxes* que foram classificados como contendo pedestres.

Em termos de interface, o sinal Disponível, na saída do detector, informa quando existe um dado pronto. Este sinal é levantado por um ciclo de *clock*. Este sinal pode ser levantado quando tem um *bounding box* classificado como contendo pedestre ou quando o processamento do *frame* foi finalizado. Todos os outros sinais de saída são alinhados temporalmente com este sinal Disponível. O sinal É Pedestre? tem valor 1 quando o *bounding box* é classificado como contendo um pedestre (ou seja, obedece a Equação C.2) e 0, caso contrário. O dado Confiança informa o nível de confiança do *bounding box*, o dado *Bounding box* contém os valores das extremidades do *Bounding box*. O sinal Final do *Frame* informa que o processamento de todo o *frame* foi concluído. Quando este sinal é levantado por um ciclo, então sabemos que o processamento do *frame* foi finalizado.

O sinal de entrada chamado de pode disponibilizar informa para o detector que as saídas já podem ser disponibilizadas. Ele é levantado em um ciclo de *clock* para cada *frame* por uma fonte externa. Quando um *frame* é processado, os *bounding boxes* somente serão disponibilizados quando esse sinal tiver sido levantado pelo menos uma vez para este *frame*. Caso contrário, todos os resultados são mantidos internamente pelo detetor.

Esse sinal é importante para gerenciar outros sistemas que estão sendo processados em paralelo com o detector (no nosso caso, a correspondência estéreo) e temos apenas um única via ou barramento de comunicação para processamento das outras etapas a frente.

A Figura 49 apresenta a arquitetura de hardware do detector incluindo seus módulos e suas conexões. Os módulos P-Imagem, P-HOG e P-SVM, instanciam respectivamente os módulos de Redimensionamento (em inglês, *Resize*), HOG e SVM para processar cada *frame* da pirâmide de imagens.

A quantidade de módulos é definida pelo parâmetro de profundidade da piramide D_{pir}. Os módulos P-HOG e P-SVM instanciam uma quantidade D_{pir} de módulos HOG e SVM respectivamente. Uma vez que no nível 0 não existe o módulo *Resize*, o módulo P-Imagem instancia $D_{\text{pir}} - 1$ módulos *Resizes*. O parâmetro de escala S_{pir} define a resolução de *frame* com que cada módulo de *Resize* irá gerar o *frame* reescalado bem como os outros módulos que irão processá-lo.

As funções $H(k)$ e $W(k)$, apresentadas, respectivamente, pelas Equações 5.1 e 5.2, definem a altura e largura do *frame* de um dado nível $k \in [1, D_{\text{pir}}]$ da pirâmide. A constante S_{pir} descreve a escala da pirâmide, ou seja, a razão entre largura ou altura do *frame* no nível $k-1$ e a largura ou altura do *frame* no nível k. No nível $k = 0$, a altura e largura é a mesma que a resolução do *frame* de entrada e não é necessário redimensionar o *frame*.

$$W(k) = W_{\text{img}}\, S_{\text{pir}}^{-(k-1)} \tag{5.1}$$

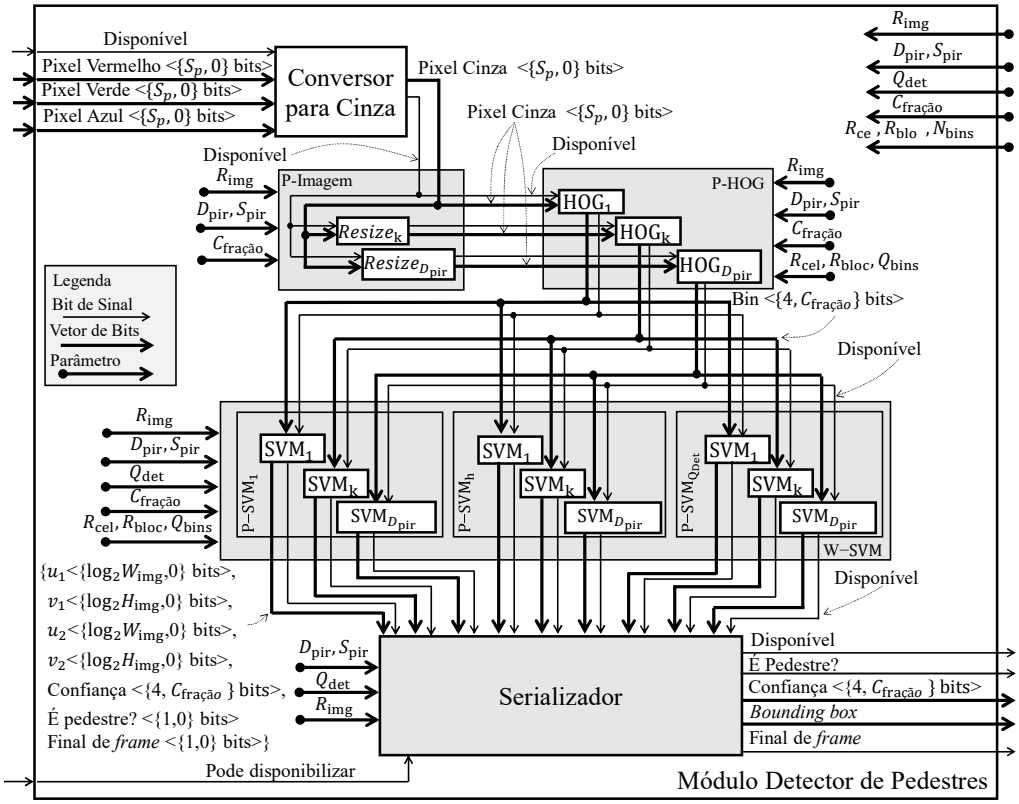
$$H(k) = H_{\text{img}}\, S_{\text{pir}}^{-(k-1)} \tag{5.2}$$

O detector processa os dados em um fluxo contínuo. Sempre que algum módulo tiver dados prontos na entrada, ele prontamente irá processá-los e armazenará os resultados parciais. Se algum resultado já puder ser fornecido, o módulo sinalizará a disponibilidade para que o próximo módulo receba estes dados e processe-os.

Todos os módulos do sistema operam com números em representação de ponto fixo. Todas as operações aritméticas são de operadores inteiros adaptados para ponto fixo. Nós definimos $\{C_{\text{inteiro}}, C_{\text{fração}}\}$ como a largura de bits da parte inteira e fracionária de um dado. Por simplicidade, nós adotamos o mesmo comprimento da parte fracionária $C_{\text{fração}}$ para todos os módulos do detector. Um estudo do comprimento mais adequado para cada módulo pode ser realizado, para redução de recursos e melhoria em precisão. A largura de bits da parte inteira C_{inteiro} depende da faixa de valores possíveis de cada módulo. Em uma visão geral de cada módulo nós apresentamos a quantidade de bits da parte inteira necessária.

Em termos de interface e fluxo de dados, o pixel RGB de entrada possui três canais, em que a intensidade de cada canal possui $\{S_p, 0\}$ bits de largura, com S_p sendo igual a 8. Quando o pixel de entrada está disponível, o módulo Conversor para Cinza converte-o para cinza com a largura de $\{S_p, 0\}$ bits. Os módulos de *Resize* processarão esse pixel

Figura 49 – A arquitetura geral de hardware do detector de pedestre. Os módulos P-SVMs são instanciados para processamento em paralelo de janelas de detecção de dimensões iguais e diferentes com o suporte de pirâmide de imagens.



Fonte: O autor.

e depois que alguns pixels necessários tiverem sido processados, eles disponibilizarão o pixel reescalado. Em seguida, os módulos HOG processarão o pixel reescalado e quando um descritor HOG tiver sido finalizado, este será disponibilizado. Como descrito na Seção 4.1 e detalhado no Apêndice B, um descritor HOG é um *bin* de um histograma de um dado bloco HOG. A largura de qualquer *bin* é $\{4, C_{\text{fração}}\}$ bits. O descritor HOG disponível, por sua vez, é processado por todos os módulos SVM e quando uma janela de detecção for finalizada, esta será disponibilizada. Uma vez que várias janelas de detecção podem ser disponibilizadas ao mesmo tempo por módulos SVMs diferentes, o módulo Serializador gerencia todos elas entregando na saída do detector uma janela de detecção por vez.

A arquitetura de hardware apresentada na Figura 49 possui uma diferença com a arquitetura do algoritmo da Figura 36. Os módulos Conversor para Cinza P-Imagem e o P-HOG são instanciados apenas uma vez e os seus resultados são reutilizados igualmente por todos os módulos P-SVM à frente. Nós propomos essa reutilização uma vez que os parâmetros destes módulos são os mesmos para todos os módulos P-SVM. Pequenas modificações podem ser realizadas para ter estes módulos independentes para cada P-SVM.

5.2 ARQUITETURA DOS MÓDULOS CHAVE

Nesta seção nós detalhamos a arquitetura dos módulos chave do detector que são o *Resize*, o HOG, o SVM e o Serializador. Nós também descrevemos a interface e o fluxo de entrada e saída dos dados destes módulos.

5.2.1 *Resize*

Esta etapa redimensiona o *frame* de origem em escala de cinza para uma dimensão menor usando a técnica de interpolação bilinear (GRIBBON; BAILEY, 2004). Esta técnica calcula a intensidade em escala de cinza de cada pixel no *frame* redimensionado através da soma ponderada da intensidade de quatro pixels vizinhos no *frame* de origem que são mais próximos de uma posição mapeada. Como descrito no Apêndice D, para cada posição do novo pixel a ser calculado no *frame* redimensionado, a Equação D.1 define a posição do pixel correspondente mapeado no *frame* de origem, a Equação D.2 define o peso de cada pixel vizinho a partir do pixel correspondente e a Equação D.3 define o valor final da intensidade para o novo pixel no *frame* redimensionado.

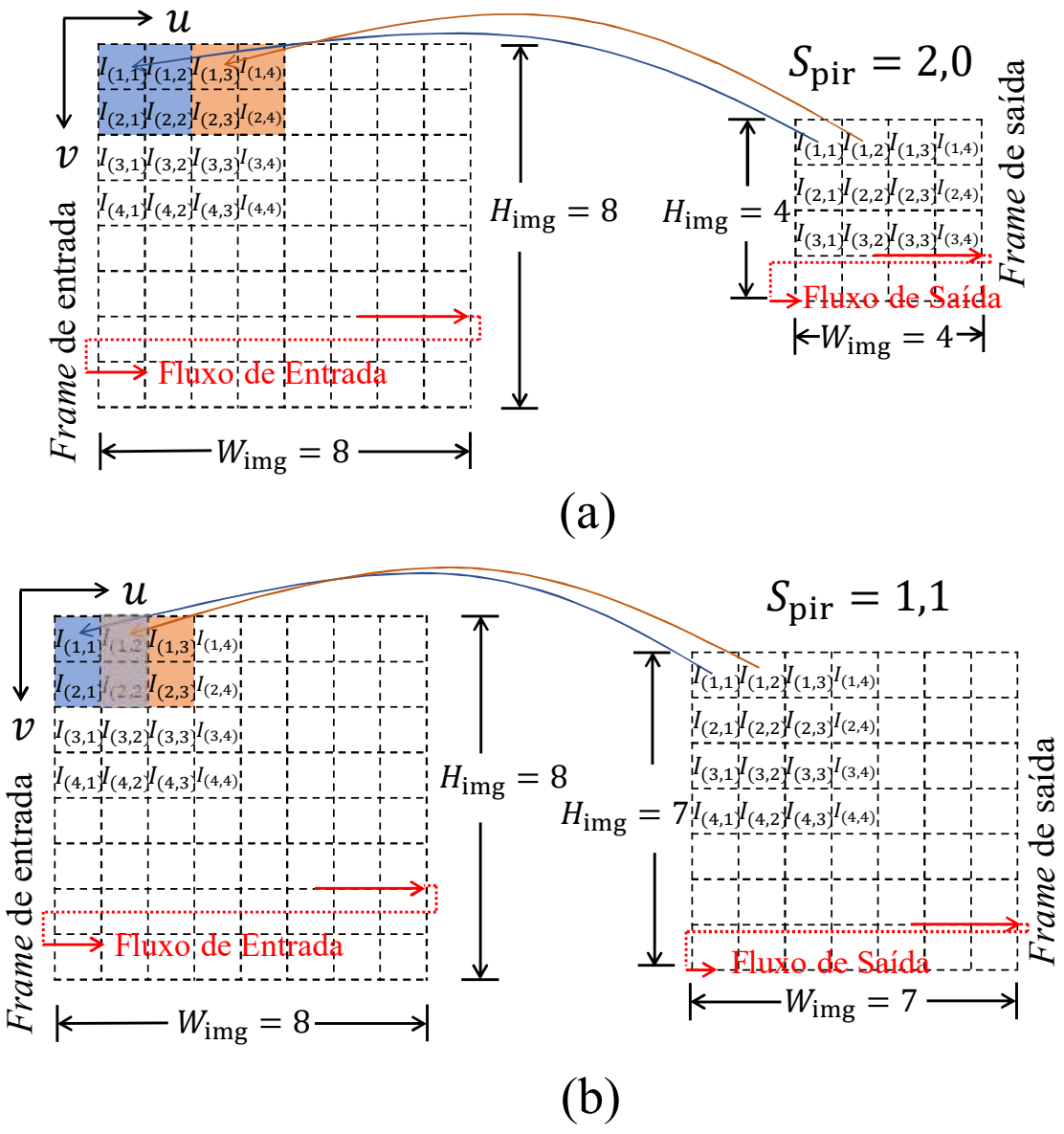
Para melhorar o entendimento da aplicação destas equações, a Figura 50 ilustra o mapeamento a partir de dois novos pixels no *frame* de saída (ou seja, o *frame* redimensionado) para os pixels no *frame* de entrada. A Figura 50(a) mostra o mapeamento com o fator de escala $S_{\text{pir}} = 2,0$. Como pode ser visto, para o pixel de posição (1,1) no *frame* redimensionado, a posição mapeada no *frame* de entrada é a (1,1). Assim, para processar este pixel são necessários o pixel esquerdo de posição (1,1), o direito de posição (1,2), o esquerdo abaixo de posição (2,1) e o direito abaixo de posição (2,2). Para processar o pixel de posição (1,2), são necessários os pixels de posição (1,3), (1,4), (2,3) e (2,4), iniciando na posição (1,3).

Como pode ser visto pela Equação D.3, nós temos quatro parcelas referente a cada pixel vizinho que são somadas para termos o resultado final de um dado pixel no *frame* de saída. Ao seguir o fluxo de entrada de pixels, ilustrado na Figura 50, sempre que um pixel mapeado de entrada estiver disponível este será processado para obtenção de uma das parcelas da Equação e acumulado para geração do resultado parcial.

Tomando o exemplo da Figura 50(a), quando o pixel de posição (1,1) estiver disponível, então o processamento da primeira parcela da Equação D.3 é realizado para o pixel de posição (1,1) no *frame* de saída, ou seja, $I_{\text{src}}(\lfloor u_{\text{src}} \rfloor, \lfloor v_{\text{src}} \rfloor) \cdot \mu_l \cdot \mu_t$, e esse resultado parcial é armazenado. Quando o pixel de posição (1,2) estiver disponível na entrada então o processamento da segunda parcela é realizado, ou seja, $I_{\text{src}}(\lfloor u_{\text{src}} \rfloor + 1, \lfloor v_{\text{src}} \rfloor) \cdot \mu_r \cdot \mu_t$ para esse mesmo pixel de posição (1,1) no *frame* saída, somado com o resultado parcial e armazenado novamente. Quando o pixel de posição (2,1) estiver disponível no *frame* de entrada então o processamento da terceira parcela é realizado, ou seja $I_{\text{src}}(\lfloor u_{\text{src}} \rfloor, \lfloor v_{\text{src}} \rfloor + 1) \cdot \mu_l \cdot \mu_b$, para esse mesmo pixel de posição (1,1), somado com o resultado parcial e armazenado. Por

fim, quando o pixel de posição (2,2) estiver disponível então o processamento da última parcela é realizado, ou seja, $I_{\text{src}}(\lfloor u_{\text{src}} \rfloor + 1, \lfloor v_{\text{src}} \rfloor + 1) \cdot \mu_r \cdot \mu_b$, somado com o resultado parcial e disponibilizado o resultado final na saída do módulo *Resize*.

Figura 50 – Mapeamento de pixels e fluxo de processamento do módulo *Resize*. O termo $I_{(v,u)}$ significa a intensidade do pixel na posição (u, v) no *frame*.



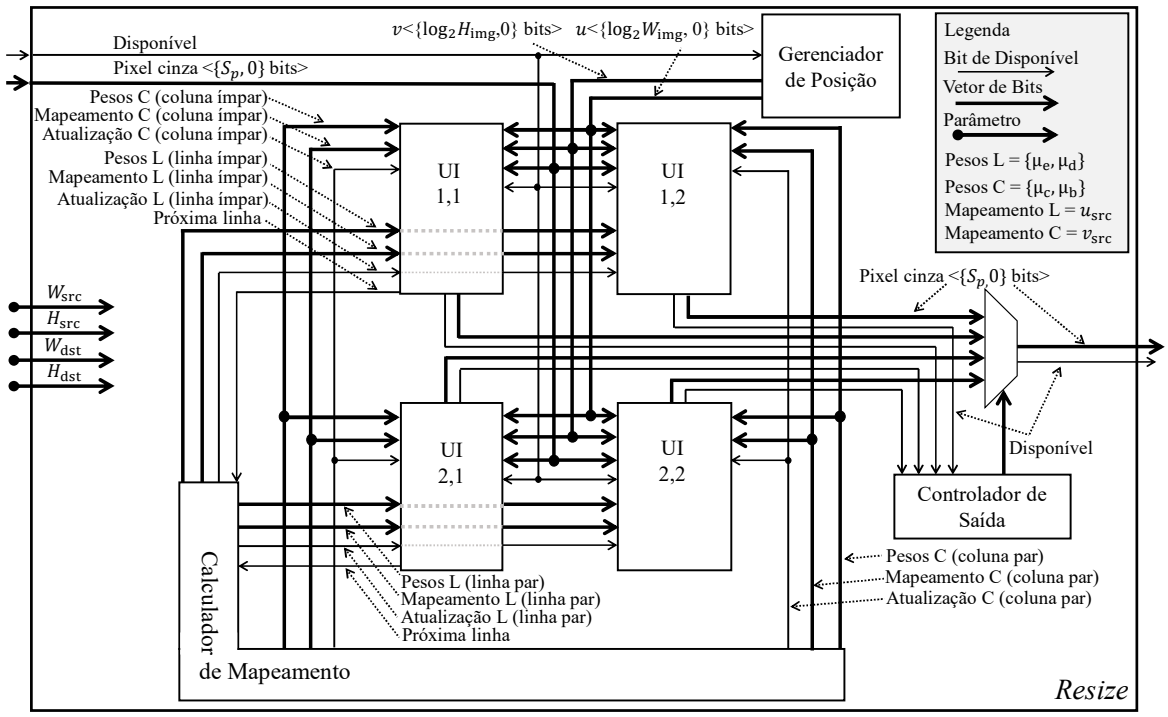
Fonte: O autor.

Nós podemos observar dois aspectos críticos no processamento dos pixels no *frame* redimensionado. O primeiro é que se S_{pir} for menor que 2, existirão pixels de posições adjacentes no *frame* de saída que precisam dos mesmos pixels no *frame* de entrada. Como pode ser visto na Figura 50(b), no qual $S_{\text{pir}} = 1,1$, os pixels de posição (1,1) e (1,2) no *frame* de saída precisam dos mesmos pixels de posição (1,2) e (2,2) no *frame* de entrada. O segundo aspecto diz respeito às operações que precisam ser realizadas para processar as Equações D.1 e D.3. Essas equações requerem operações aritméticas que demandam

muitos ciclos de *clock* que podem levar à perda de pixels de entrada enquanto essas operações estão sendo realizadas, se a vazão de entrada de pixels for de um pixel por ciclo de *clock*.

Para suportar o cenário de vazão máxima de entrada de pixels serial (1 pixel a cada ciclo de *clock*) e evitar a perda de pixels, é necessário processar as duas posições adjacentes ao mesmo tempo no *frame* de saída para o mesmo pixel de entrada. Dessa forma, nós propusemos uma estratégia de múltiplas unidades de interpolação (UI), como mostrado na Figura 51. Essas unidades são módulos que processam pixels em posições intercaladas no *frame* redimensionado, usando a Equação D.3, como mostrado na Figura 52.

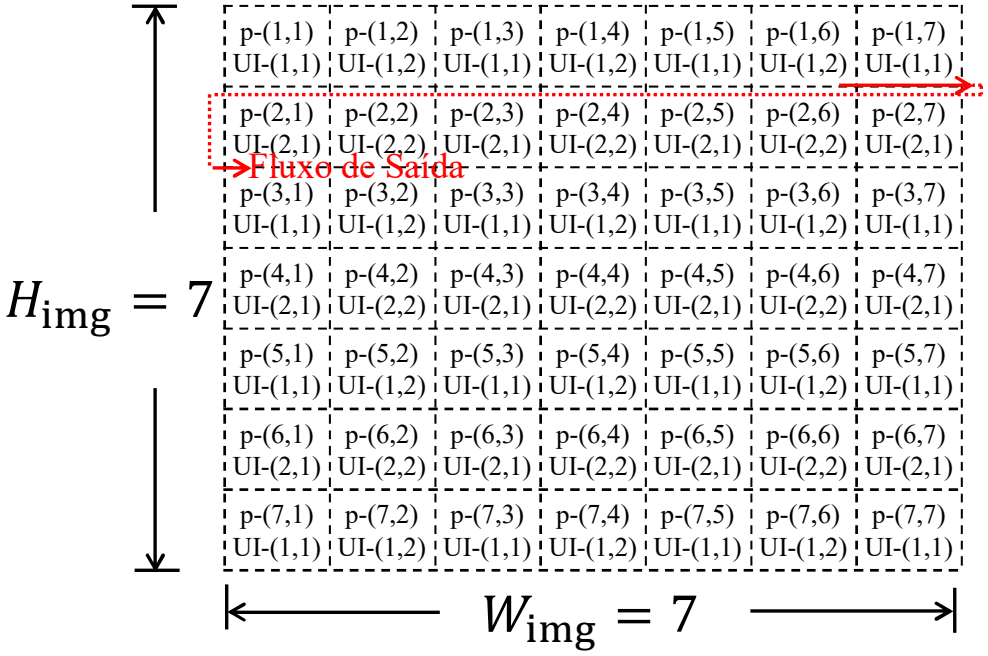
Figura 51 – A arquitetura de hardware do módulo *Resize*. O termo UI significa Unidade de Interpolação. Os termos L e C significam dados referentes, respectivamente, à linha e coluna. Quatro unidades de interpolação (UI) são suficientes para garantir um fluxo de processamento sem interrupção.



Fonte: O autor.

As unidades de índices 1,1 e 2,1 processam colunas ímpares e as unidades de índices 1,2 e 2,2 processam colunas pares de pixels no *frame* redimensionado. As unidades de índices 1,1 e 1,2 processam linhas ímpares de pixels e as unidades de índices 2,1 e 2,2 processam linhas pares de pixels. Quando existem pixels no *frame* de entrada que precisam ser processados mais de uma vez para calcular pixels em posições adjacentes no *frame* redimensionado, essas quatro unidades de interpolação processarão esses pixels em paralelo. O módulo Gerenciador de Posição, mostrado na Figura 51, mantém a informação da posição correta (u, v) de cada pixel de entrada. Essa posição é utilizada pelas Unidades de Interpolação para selecionar quais pixels serão processados.

Figura 52 – Lógica de distribuição das Unidades de Interpolação (UI) entre os pixels (p) que precisam ser calculados no *frame* redimensionado. O termo p-(v, u) indica a posição do pixel na imagem e o termo UI-(i, j) indica a localização da unidade de interpolação na organização do módulo de *Resize*



Fonte: O autor.

O módulo Controlador de Saída gerencia os resultados a partir das Unidades de Interpolação para disponibilização dos resultados ordenados no módulo *Resize*. Uma vez que as Unidades de Interpolação entregam os resultados ordenados dos seus respectivos pixels intercalados, o Controlador de Saída incrementa um contador interno de coluna e linha para saber quando é uma posição ímpar ou par e, assim, definir qual unidade escolher.

Nós resolvemos o segundo aspecto crítico através de duas estratégias conjuntas. A primeira é o armazenamento antecipado do mapeamento e dos pesos e a segunda é a abordagem de processamento em *pipeline* das operações. Com relação a primeira estratégia, nós nos baseamos no fato de que a escala S_{pir} é constante e, portanto, os mapeamentos são sempre os mesmos de um *frame* para o outro. Portanto, nós propomos um módulo, chamado de Calculador de Mapeamento, mostrado na Figura 51, que calcula os mapeamentos e pesos, armazena-os em memória e registradores antes que o módulo de *resize* comece a processar qualquer *frame*.

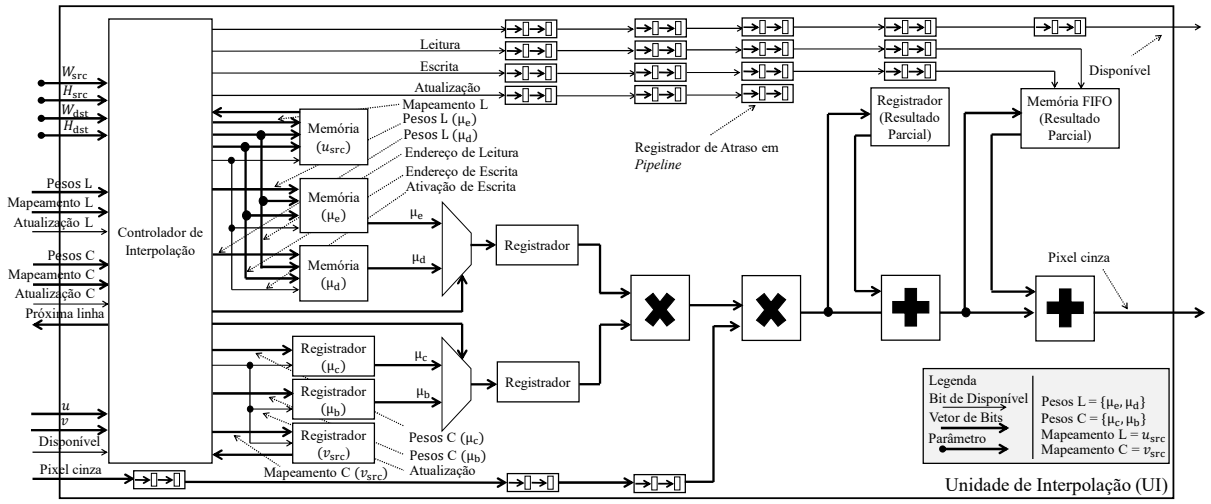
O armazenamento dos dados de mapeamento e peso não ocorre para todas as posições do *frame*, pois não seria uma solução escalável em termos de uso de memória com relação à resolução do *frame*. Em vez disso, o módulo Calculador de Mapeamento define uma linha inteira de mapeamentos de cada coluna $u_{\text{dst}} \rightarrow u_{\text{src}}$ seguindo a Equação D.1, divide-os e armazena-os na memória do FPGA de cada Unidade de Interpolação. Esta estratégia é possível pois o mapeamento no eixo u não varia conforme descemos na linha do *frame*.

Por exemplo, na Figura 50(a), os pixels de coluna 2 no *frame* de saída são mapeados sempre na coluna 3 no *frame* de entrada independente da linha em que o pixel esteja sendo processado.

O mapeamento de cada linha $v_{\text{dst}} \rightarrow v_{\text{src}}$ é definido ao longo do processamento do *frame*. O módulo Calculador de Mapeamento define a próxima linha v_{src} da Unidade de Interpolação sempre que a linha atual é finalizada. Uma vez que as Unidades de Interpolação processam linhas intercaladas, a próxima linha a ser definida é sempre duas linhas à frente com relação a linha atual que a Unidade de Interpolação estava processando. Consequentemente, a estratégia de Unidades de Interpolação intercaladas garante tempo suficiente para cada Unidade obter a próxima linha antes que os pixels de entrada estejam disponíveis.

Com relação a segunda estratégia, o *pipeline* na Unidade de Interpolação, nós implementamos as operações de interpolação de uma maneira para garantir uma vazão de entrada de pixels de um ciclo por *clock*. Essa característica é necessária para suportar a vazão de entrada de pixels que, no caso mais crítico, ocorre a um ciclo por *clock*, ou seja, sem interrupção.

Figura 53 – A arquitetura de processamento da Unidade de Interpolação (UI)



Fonte: O autor.

A Figura 53 mostra a arquitetura para a Unidade de Interpolação. Cada Unidade de Interpolação possui o módulo Controlador de Interpolação que verifica quais pixels de entrada participarão do cálculo dos novos pixels no *frame* redimensionado e define qual peso será multiplicado pelos respectivos pixels.

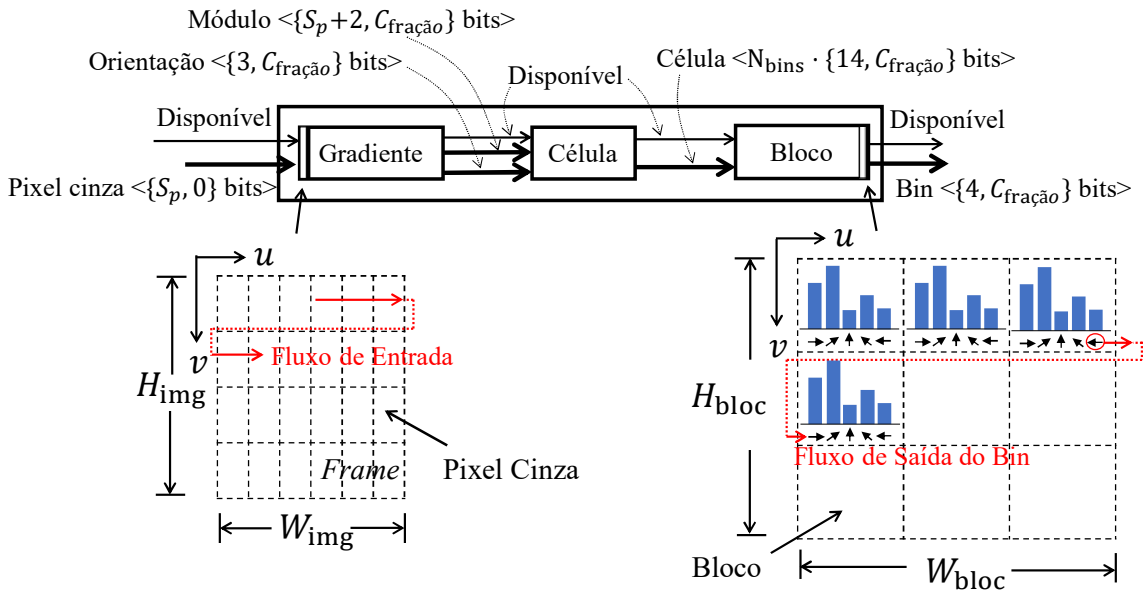
Considere o cálculo de um determinado pixel q. Quando o pixel do *frame* de entrada referente a primeira parcela de q estiver disponível, ele é multiplicado pelos pesos μ_l, μ_t, seguindo a Equação D.3, e armazenado no registrador Resultado Parcial. Quando o pixel da próxima coluna estiver disponível, ele é multiplicado pelos pesos μ_r, μ_t, somado ao resultado anterior e armazenado na memória de FPGA do tipo FIFO (em inglês, *First-*

In-First-Out). Esta memória armazena uma linha de resultados parciais. Quando o pixel da linha seguinte, que faz parte do mesmo pixel q, estiver disponível (ou seja, o terceiro pixel), ele é multiplicado pelos pesos μ_l e μ_b e armazenado no registrador Resultado Parcial. Quando o último pixel estiver disponível, ele é multiplicado pelas pesos μ_r e μ_b, somado ao resultado que está no registrador e em seguida somado ao resultado parcial que está na memória FIFO referente a linha anterior e o resultado é disponibilizado na saída da Unidade de Interpolação. Sem perda de generalidade, os próximos pixels vizinhos também seguirão esse processamento e, como não há dependência de dados entre eles, eles estarão sendo processados sem necessidade de esperar.

5.2.2 HOG

O módulo HOG inclui três submódulos seguindo os passos detalhados na Seção 4.1 e detalhados no Apêndice B: Gradiente, Célula e Bloco. A Figura 54 mostra a interface e fluxo de entrada e saída do módulo HOG bem como as interfaces que conectam cada submódulo.

Figura 54 – Descrição da interface e fluxo de entrada e saída de dados da arquitetura de hardware do módulo HOG



Fonte: O autor.

Sempre que o pixel cinza, vindo do módulo *Resize*, estiver disponível, o módulo Gradiente irá processá-lo. Sempre que o gradiente de um dado pixel tiver sido processado, seu módulo e orientação são disponibilizados para serem processados pelo módulo Célula. Sempre que uma célula estiver disponível, todos os seus *bins* de histograma são disponibilizados em um único ciclo de *clock* para serem processados pelo módulo de bloco. Quando

um bloco tiver sido processado, seus *bins* serão disponibilizados um por vez, a cada ciclo de *clock*.

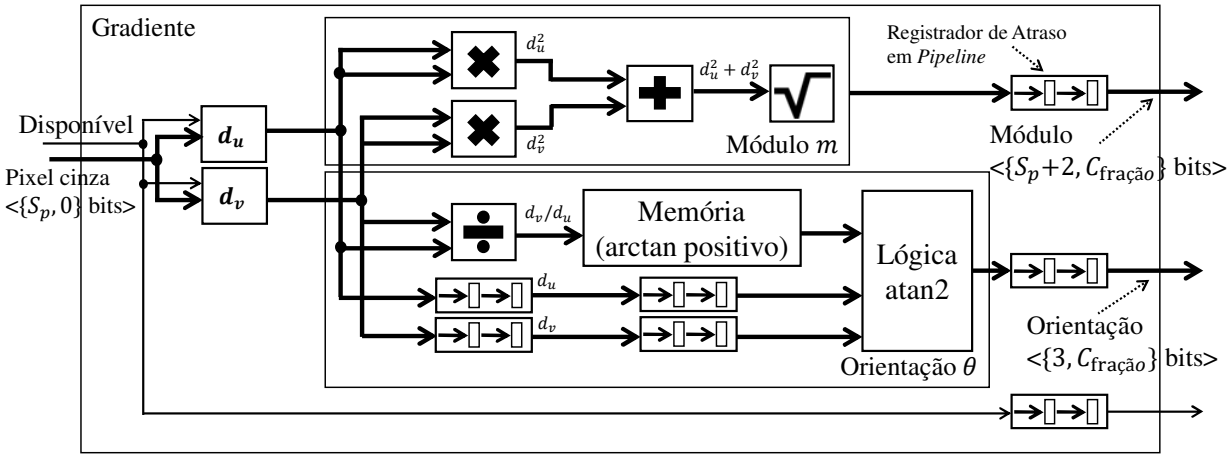
A seguir descrevemos detalhadamente os módulos Gradiente, Célula e Bloco.

5.2.2.1 Gradiente

Na medida em que os pixels estão prontos na entrada do módulo Gradiente, estes vão sendo processados por este módulo para a obtenção das orientações e módulos de gradiente de acordo com a Equação B.2. A arquitetura do módulo Gradiente é mostrada na Figura 55.

Primeiramente as componentes $d_u(u, v)$ e $d_v(u, v)$ descritas pela Equação B.1 são calculadas. Este cálculo usa a técnica de janelas deslizantes com uma janela de *kernel* de dimensão 3×3. As janelas de *kernel* para d_u e d_v são definidas pela Equação 5.3.

Figura 55 – Arquitetura do módulo Gradiente. A Equação B.2 descreve o cálculo que é realizado neste módulo.



Fonte: O autor.

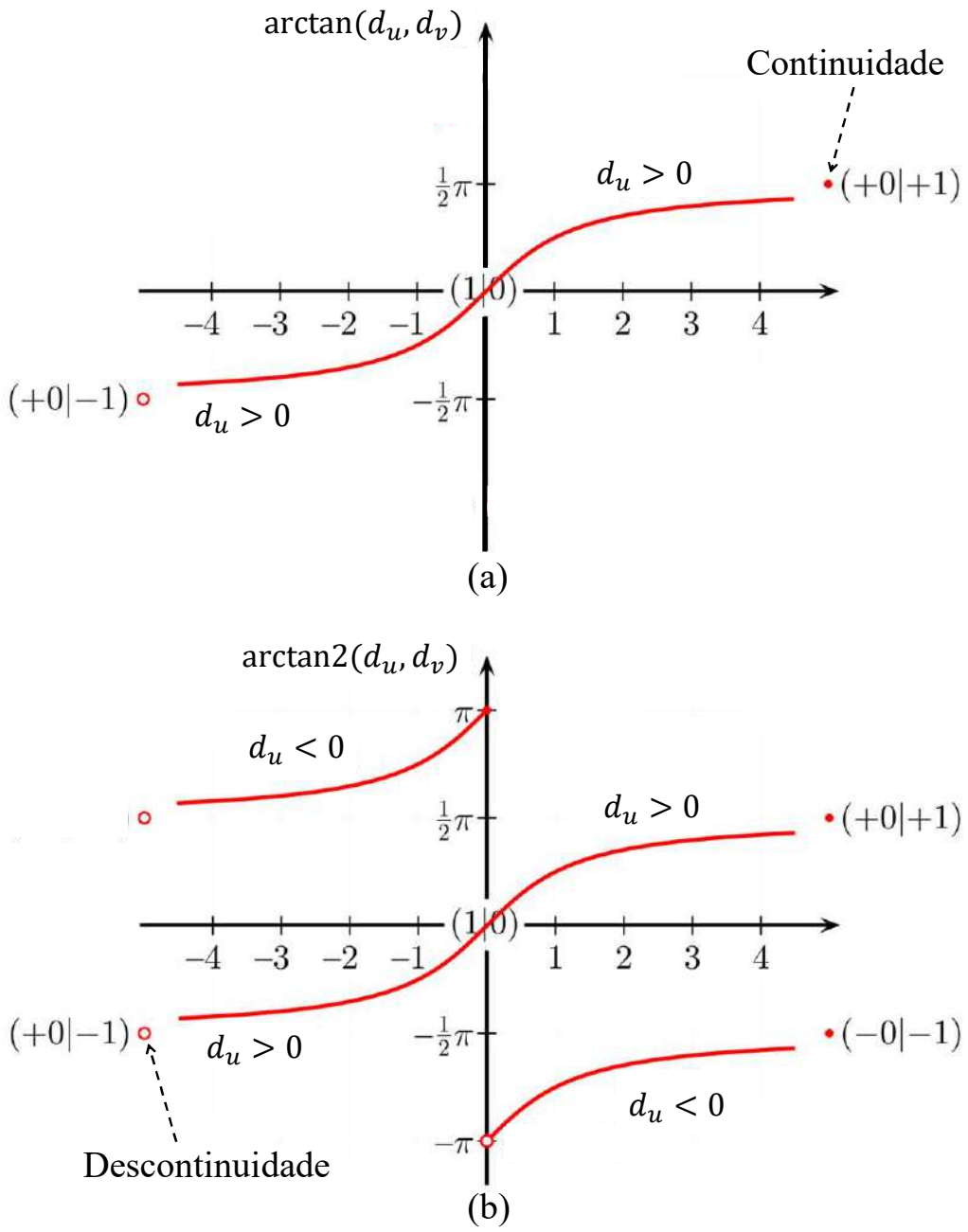
A janela deslizante utiliza *buffers* de linha que mantém 3 linhas de pixels mais recentes do *frame* de entrada. Esses *buffers* permitem o acesso aos pixels vizinhos de um dado pixel que está sendo processado, que são necessários para a realização do cálculo das componentes deste pixel de acordo com o respectivo *kernel*. A partir do momento que duas linhas de pixels e três pixels da terceira linha já estiverem disponíveis nos *buffers* de linha, os valores das componentes $d_u(u, v)$ e $d_v(u, v)$ do primeiro pixel do *frame* já podem ser disponibilizados.

$$d_u = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline -1 & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \quad d_v = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \tag{5.3}$$

Com as componentes $d_u(u, v)$ e $d_v(u, v)$ disponíveis, o próximo processamento é realizado pelos módulos que calculam o módulo m e orientação θ do gradiente, seguindo

a Equação B.2. Para calcular o módulo do gradiente, primeiramente obtém-se as componentes d_u^2 e d_v^2 a partir da multiplicação de d_u e d_v por ele mesmo. Em seguida, os resultados são somados para gerar $d_u^2 + d_v^2$. Finalmente, sua raiz quadrada é calculada. Nós adaptamos o módulo de raiz quadrada para números inteiros para calcular valores em ponto fixo.

Figura 56 – Diferença entre o (a) arctan e o (b) arctan2. A notação $(d_u|d_v)$ indica se os valores de d_u e d_v são positivos (+1), negativos (-1) ou zero (0).



Fonte: O autor.

Para calcular a componente θ nós implementamos a função de arco tangente de duas variáveis, também chamado de arctan2. É importante diferenciar o arctan2 a partir do

arco tangente tradicional, também chamado de arctan. O arctan é uma função definida no intervalo de $[-\pi/2, \pi/2]$, que é obtida a partir de d_v/d_u. O arctan2 é uma função definida no intervalo de $[-\pi, \pi]$ e que é obtida a partir das duas componentes individuas d_v e d_u. A Equação 5.4 apresenta a função arctan2 e a Figura 56 ilustra a diferença entre as estas duas funções.

$$\text{arctan2}(d_u, d_v) = \begin{cases} \arctan(\frac{d_v}{d_u}), & \text{se } d_u > 0, \\ \arctan(\frac{d_v}{d_u}) + \pi, & \text{se } d_u < 0 \wedge d_v \geq 0, \\ \arctan(\frac{d_v}{d_u}) - \pi, & \text{se } d_u < 0 \wedge d_v < 0, \\ +\frac{\pi}{2}, & \text{se } d_u = 0 \wedge d_v > 0, \\ -\frac{\pi}{2}, & \text{se } d_u = 0 \wedge d_v < 0, \\ 0, & \text{se } d_u = 0 \wedge d_v = 0. \end{cases} \tag{5.4}$$

Os módulos disponíveis que processam a função arctan utilizam o algoritmo de base chamado de computador digital para rotação de coordenadas (em inglês, *Coordinate Rotation Digital Computer* - CORDIC) que demanda muito recurso de processamento e limita a frequência máxima de operação (MANOR; BEN-DAVID; GREENBERG, 2022)(NGUYEN et al., 2016)(KAUR; SINGH, 2012). Uma vez que é necessário instanciar várias vezes o módulo Gradiente, devido a abordagem de pirâmide de imagem, a solução para calcular o arctan2 precisa ser escalável em termos de recursos de processamento e frequência.

Assim, nós propomos um módulo que utiliza a abordagem baseada em tabela *look-up*. Como pode ser observado na Equação 5.4, a função arctan2 pode ser calculada a partir dos resultados da função arctan. Além disso a função arctan é simétrica em torno do eixo das ordenadas. Então, apenas o lado positivo da função arctan é armazenada em memória de FPGA.

$$\text{índice}_{\text{arctan}}(d_u, d_v) = \text{round}(d_v/d_u) \tag{5.5}$$

$$\text{arctan2}(d_u, d_v) = \begin{cases} +\text{arctan}_{\text{tabela}}(\text{índice}_{\text{arctan}}), & \text{se } d_u > 0 \wedge d_u \geq 0, \\ -\text{arctan}_{\text{tabela}}(\text{índice}_{\text{arctan}}), & \text{se } d_u > 0 \wedge d_u < 0, \\ \pi - \text{arctan}_{\text{tabela}}(\text{índice}_{\text{arctan}}), & \text{se } d_u < 0 \wedge d_v > 0, \\ +\text{arctan}_{\text{tabela}}(\text{índice}_{\text{arctan}}) - \pi, & \text{se } d_u < 0 \wedge d_v < 0, \\ +\pi, & \text{se } d_u < 0 \wedge d_v = 0, \\ +\frac{\pi}{2}, & \text{se } d_u = 0 \wedge d_v > 0, \\ -\frac{\pi}{2}, & \text{se } d_u = 0 \wedge d_v < 0, \\ 0, & \text{se } d_u = 0 \wedge d_v = 0. \end{cases} \tag{5.6}$$

Os valores armazenados estão no domínio de intervalo $[0, +\text{limite}_{\text{arctan}}]$ com saltos de $\text{step}_{\text{arctan}}$. A entrada $\text{índice}_{\text{arctan}}$ na tabela, é definida na Equação 5.5. O restante da

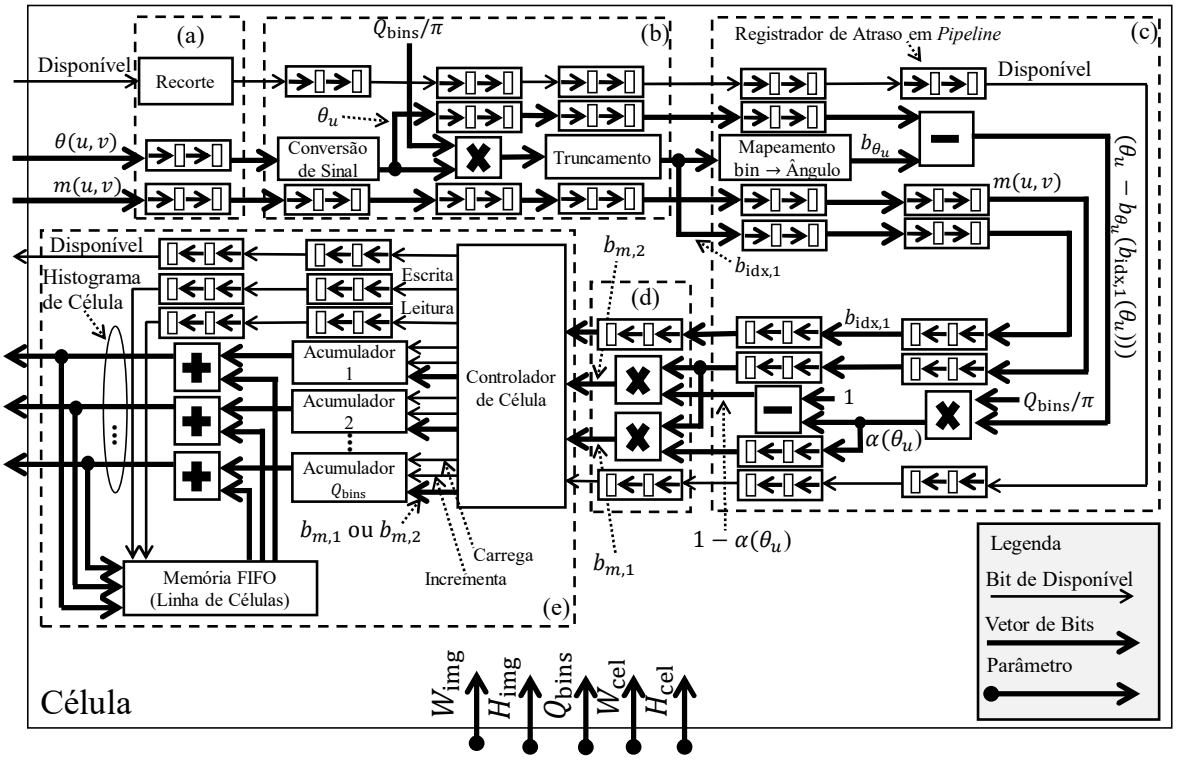
função arctan2, mostrado na Equação 5.6, é coberto através de um conjunto de lógicas de comparação usando os sinais das componentes d_u e d_v. Cada condição dessa Equação é realizada em um estágio diferente de *pipeline* para reduzir a complexidade de operação por ciclo de *clock*.

Os registradores de atraso no final do módulo Gradiente, mostrado na Figura 55, são utilizados para alinhar os resultados do processamento de módulo e orientação e o sinal de disponível na saída.

5.2.2.2 Célula

Uma vez que o gradiente está disponível, o módulo Célula calcula os histogramas de cada célula do *frame*. A arquitetura de processamento da célula é mostrada na Figura 57.

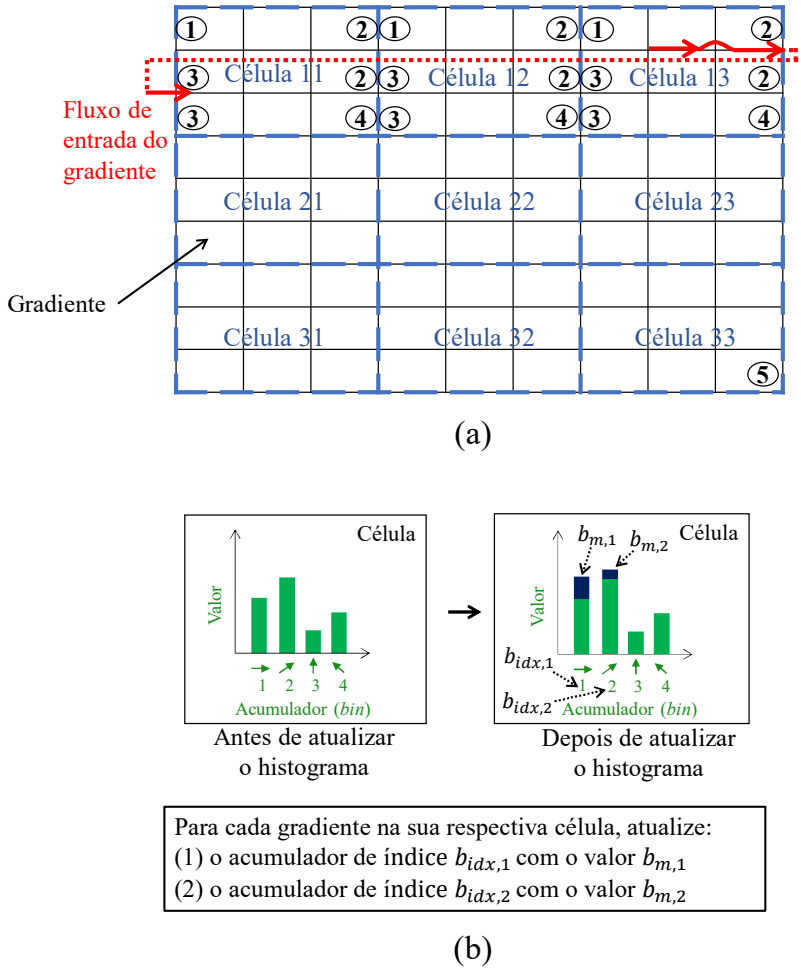
Figura 57 – Arquitetura de processamento do módulo Célula. As Equações B.3, B.4, B.5, B.6, B.7, B.8 e B.9 descrevem o cálculo do histograma de célula que é realizado neste módulo. As principais tarefas realizadas neste módulo são: (a) recorte de imagem, (b) cálculo de índice, (c) cálculo do peso, (d) cálculo dos valores ponderados e (e) gerenciamento de células.



Fonte: O autor.

A primeira etapa executa o recorte do *frame* de gradiente, removendo linhas e colunas à direita e abaixo para garantir que o *frame* recortado seja um múltiplo da dimensão da célula HOG. Este recorte é realizado em hardware através de registradores de linha e coluna que mantém a posição correta do gradiente no *frame* e de lógicas de verificação se a posição do gradiente de entrada está dentro dos limites permitidos para o novo *frame* recortado. Se sim, então o pixel é disponibilizado na saída do módulo de recorte.

Figura 58 – Explicação (a) do gerenciamento de todas as células do *frame* e (b) da atualização de *bins* no histograma a partir de cada gradiente. As marcações numeradas indicam situações diferentes que requerem um lógica de tratamento diferente: (1) indica que é o primeiro gradiente da primeira linha a ser processado, (2) indica que é o último gradiente da linha atual de uma determinada célula, (3) indica o primeiro gradiente das linhas seguintes, (4) indica o último gradiente da última linha de uma determinada célula e (5) indica o último gradiente do *frame*.



Fonte: O autor.

As próximas etapas são referentes ao processamento da atualização dos *bins* e o gerenciamento das diversas células no *frame* para a garantia do fluxo de processamento de um pixel por ciclo sem interrupção. A Figura 58 auxilia na explicação dessas etapas.

5.2.2.2.1 Atualização de bins

Para realizar a atualização de *bins*, cada gradiente tem sua orientação com sinal (θ) convertida em orientação sem sinal (θ_u), conforme definida pela Equação B.3. Essa conversão é feita através da soma da constante π para os ângulos θ que estão entre [180°,360°]. Em seguida, de acordo com a Equação B.4, a orientação sem sinal (θ_u) é multiplicada pela constante Q_{bins}/π. Depois, são realizados o truncamento do resultado e extração da

parte inteira para obtenção do índice do *bin* principal $b_{\text{idx},1}$. Este valor é usado no módulo mapeamento bin $\rightarrow$ ângulo para procurar o respectivo ângulo b_{θ_u}, conforme descrito pela Equação B.6. Em vez de realizar uma multiplicação, este módulo de mapeamento define uma memória indexável contendo valores constantes, em que cada posição $i \in [1, Q_{\text{bins}}]$ é igual a $(i-1)(\pi/Q_{\text{bins}})$.

Em seguida, o fator $\alpha(\theta_u)$ é obtido, realizando primeiramente o cálculo de $(\theta_u - b_{\theta_u}(b_{\text{idx},1}(\theta_u)))$ e o resultado multiplicado por Q_{bins}/π. A partir de $\alpha(\theta_u)$, obtém-se $1-\alpha(\theta_u)$ que é o peso do *bin* vizinho. Então, os valores dos módulos de gradiente proporcional para os dois *bins*, $b_{m,1}$ e $b_{m,2}$, são calculados através de duas multiplicações de $\alpha(\theta_u)m(u,v)$ e $(1-\alpha(\theta_u))m(u,v)$, respectivamente.

A partir do índice do *bin* principal $b_{\text{idx},1}$, o módulo Controlador de Célula define quais *bins* terão seus valores parciais incrementados simultaneamente com os módulos de gradientes proporcionais, como mostrado na Figura 58(a). Esses *bins* são representados pelos acumuladores que incrementam seu valor parcial com o valor de gradiente de entrada ou podem carregar um novo valor parcial.

5.2.2.2.2 Gerenciamento de células

Ao longo do processamento das diversas células várias situações acontecem que são tratadas de maneira diferente pelo módulo Controlador de Célula. A Figura 58(b) mostra através de rótulos numerados essas situações. Antes do processamento da primeira linha de cada célula, mostrado pelo rótulo 1, os valores iniciais dos acumuladores estão zerados. Cada gradiente atualiza os valores dos acumuladores e quando o último gradiente da primeira linha de cada célula é processado, mostrado pelo rótulo 2, o Controlador de Célula armazena os resultados temporários dos acumuladores em memória do tipo FIFO. Esse armazenamento é feito, aglutinando todos os acumuladores em um único vetor de bits e ativando por um ciclo de *clock* o sinal de escrita com esse vetor na entrada da memória. No final do processamento de uma linha inteira do *frame*, todos os resultados temporários de todas as células que foram parcialmente processadas estarão armazenados nessa memória.

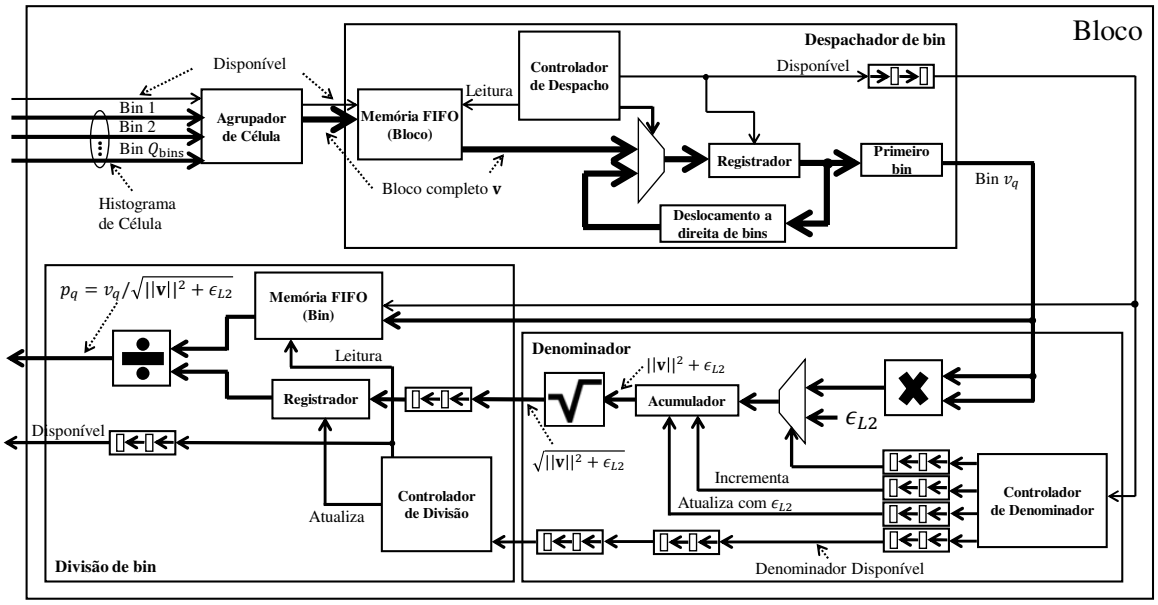
No início das linhas seguintes, mostrado pelo rótulo 3, o valor parcial das células que estão na memória, são carregados de volta para os respectivos acumuladores. Esse carregamento é feito pelo Controlador de Célula que levanta o sinal Carrega e o sinal de leitura da memória, por um ciclo de *clock*, para respectivamente, atualizar os acumulares com os *bins* que estão na memória e remover esses valores da memória. Ao término do processamento do último gradiente de cada célula, mostrado pelo rótulo 4, os *bins* respectivos já estarão prontos e serão disponibilizados na saída do módulo Célula. Além disso, uma vez que o processamento da célula foi finalizado, os resultados não são armazenados de volta para a memória. Assim, ao término da última linha inteira, a memória estará vazia, podendo dar início ao processamento das células das linhas seguintes.

5.2.2.3 Bloco

A partir do resultado de cada célula, o módulo Bloco realiza a normalização dos *bins* dentro de um bloco, seguindo a Equação B.10, apresentada no Apêndice B. È possível destacar dois desafios para garantir o processamento sem interrupção. O primeiro envolve o cálculo do denominador que precisa processar todos os *bins* usando operações de multiplicação e raiz quadrada. O segundo desafio envolve o cálculo do *bin* normalizado que envolve operação de divisão. Todas estas operações demandam muitos ciclos de *clock* e impõem sérios desafios para implementação de soluções que consigam garantir um fluxo de processamento sem interrupção.

Nós conseguimos alcançar tal solução através da arquitetura apresentada na Figura 59. O primeiro módulo, o Agrupador de Célula, executa a estratégia de janela deslizante para obter as quatro células que compõem cada bloco. A Figura 60 ilustra esta estratégia.

Figura 59 – Arquitetura de processamento do módulo Bloco. A Equação B.10 descreve o cálculo que é realizado neste módulo.



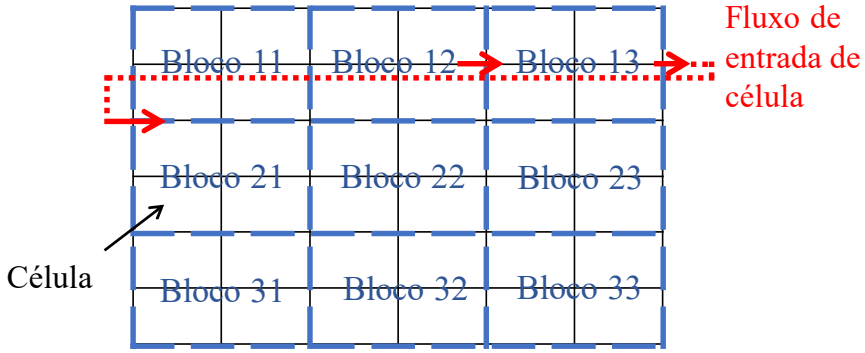
Fonte: O autor.

Cada bloco formado é enviado para o módulo Despachador de *Bin* que despacha um *bin* por vez. Primeiramente, este módulo armazena em memória do tipo FIFO todos os blocos que chegam a partir do módulo Agrupador de Célula. Em seguida, o Controlador de Despacho lê um bloco desta memória e armazena no registrador. A cada ciclo de *clock*, o primeiro *bin* do registrador é disponibilizado na saída e o vetor de bits do registrador é deslocado para a direita para descartar este *bin*. Quando todos os *bins* tiverem sido despachados, o Controlador de Despacho lê o próximo bloco disponível a partir da memória.

Cada *bin* despachado vai para o módulo Denominador e para o módulo Divisão de *bin*. O módulo Denominador calcula o denominador da Equação B.10 e o módulo Divisão

de *bin* calcula a divisão de cada *bin* pelo denominador para obter o *bin* normalizado.

Figura 60 – Explicação da varredura de célula e composição de blocos.



Fonte: O autor.

O módulo Denominador processa, primeiramente, a potência de dois de cada *bin*, através do módulo de multiplicação. Em seguida, cada resultado de potência de dois é acumulado em um registrador Acumulador. Antes de acumular, o registrador é carregado com o valor inicial de ϵ_{L2}. Quando as potências de dois de todos os *bins* do bloco tiverem sido acumuladas, sua raiz quadrada é então processada. O resultado do módulo raiz quadrada compõe o denominador. O módulo Controlador de Denominador é responsável por gerenciar os dados do acumulador e definição do valor inicial e também sinalizar na saída do módulo Denominador quando existe um valor de denominador disponível.

Quando um valor de denominador está disponível este é enviado ao módulo Divisão de *bin* que armazena-o em um registrador. O módulo Controlador de Divisão despacha um *bin* a partir da memória FIFO a cada ciclo de *clock* para ser dividido pelo denominador. O resultado da divisão, constituindo o *bin* normalizado, é disponibilizado na saída do módulo Bloco. No momento em que o último *bin* de um dado bloco tiver sido normalizado, o novo valor de denominador já estará disponível e será armazenado no registrador para processamento dos *bins* normalizados da próxima célula.

A estratégia de processamento de um *bin* por ciclo de *clock* proposta para o módulo Bloco exige menos recursos computacionais do que abordagens que processariam ao mesmo tempo *bins*. Porém essa estratégia apenas é possível se o número de ciclos necessários para processar uma linha inteira de blocos for menor que o número de ciclos até a chegada da próxima linha de células que formarão os próximos blocos. Uma vez que o fluxo de entrada do detector é de um pixel por ciclo de *clock*, entre uma linha de bloco e a próxima existem H_{cel} linhas de células a serem formadas, e este é o intervalo de tempo livre que o módulo de bloco tem para calcular todos os blocos da linha atual. Então este espaço de tempo em ciclos de *clock* é igual a H_{cel} multiplicado por W_{img}. Se não fosse possível garantir o processamento dentro desse espaço de tempo, seria necessária alguma estratégia para bloquear a entrada de pixels ou alguma técnica para armazenar em memória as células ainda não processadas. Ambas as estratégias não são interessantes porque

prejudicam o desempenho de todo o detector e a escalabilidade em termos de resolução de imagem.

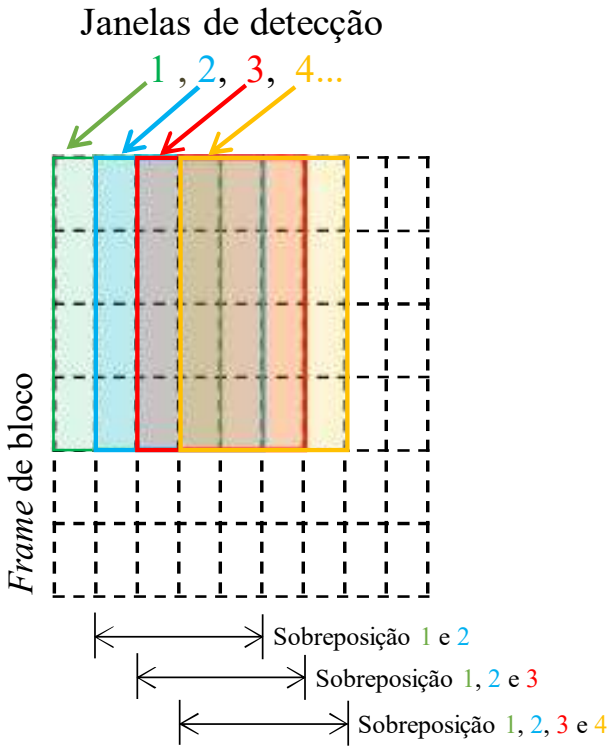
5.2.3 SVM

A proposição da arquitetura do módulo SVM passa por duas fases das quais foram obtidas duas versões deste módulo. Na primeira fase nós propomos uma arquitetura completa do SVM com foco no paralelismo para alcançar alto desempenho de processamento. Na segunda fase nós propomos estratégias de compartilhamento de unidades lógicas e memórias, para redução de recursos computacionais da arquitetura inicial, sem prejudicar o desempenho de processamento alcançado.

5.2.3.1 Versão 1: Estratégia de paralelismo

Uma vez que as características calculadas pelo HOG (ou seja, os *bins*) estão disponíveis, o módulo SVM calcula a confiança $f(\mathbf{u})$ de cada janela de detecção seguindo a Equação C.1 e então verifica se cada janela contém pedestres, seguindo a Equação C.2.

Figura 61 – Sobreposição de janelas de detecção.



Fonte: O autor.

A sobreposição entre janelas é um aspecto desafiador para alcançar o processamento em fluxo contínuo de entrada de pixels sem interrupção. Uma vez que o *stride* em pixels, ou seja, o salto em pixels entre janelas, é menor que a dimensão da janela de detecção,

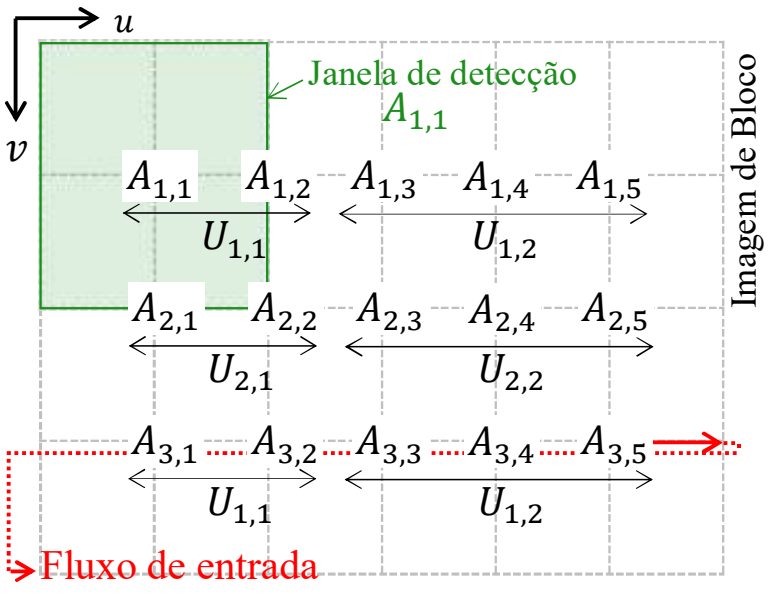
várias janelas irão se sobrepor, como mostrado na Figura 61. As sobreposições acontecem tanto na horizontal como na vertical.

Consequentemente, quando uma característica calculada pelo HOG está pronta, ela precisa ser processada por várias janelas de detecção. Para ser mais preciso, a quantidade máxima de diferentes janelas sobrepostas O_{janelas} que precisam ser processadas, por um módulo SVM de dimensão de janela $W_{\text{SVM}} \times H_{\text{SVM}}$ pixels e célula com dimensão $W_{\text{cel}} \times H_{\text{cel}}$ pixels, a partir de uma característica HOG é dada como:

$$O_{\text{windows}} = \frac{\text{W}_{\text{SVM}}}{\text{W}_{\text{cel}}} \times \frac{\text{H}_{\text{SVM}}}{\text{H}_{\text{cel}}} \tag{5.7}$$

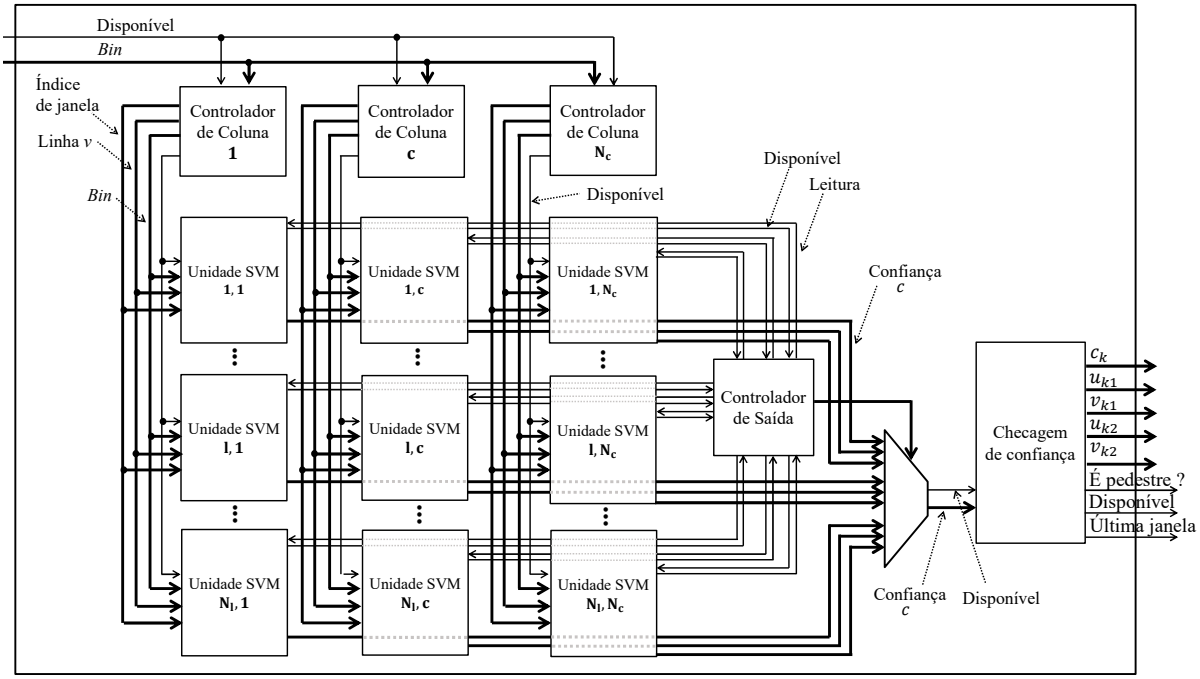
Devido à sobreposição, sempre que uma característica HOG estiver pronta, o módulo SVM gastará uma quantidade de ciclos definida pela Equação 5.7 vezes o número de ciclos para processar uma única janela. Por exemplo, considere uma janela de detecção com dimensões de 64×128 pixels e uma célula HOG com dimensões de 8×8 pixels. Aplicando a Equação 5.7, nós teremos 128 janelas de detecção sobrepostas. Se considerarmos que o processamento de uma característica leva 10 ciclos, então precisaríamos de 1280 ciclos para processar uma característica para todas as 128 janelas. Não é difícil ver que, sem nenhum tipo de paralelismo, a vazão de entrada de características será muito maior do que a vazão de processamento de todas as janelas. Essa diferença na taxa de transferência na vazão certamente torna o processamento em fluxo contínuo sem interrupção inviável.

Figura 62 – Exemplo de distribuição de unidades SVM. Neste exemplo, a dimensão da janela de detecção é 2×2 blocos. Como a quantidade de linhas de unidades SVM é igual a quantidade de linhas de bloco de janela de detecção, então $N_l = 2$. A quantidade de colunas pode ser definida livremente para atender a demanda por processamento sem interrupção. Neste exemplo, nós definimos $N_c = 2$. O termo $U_{l,c}$ significa unidade SVM na posição (l, c) na matriz de unidades SVM. O termo $A_{v,u}$ significa janela de detecção na posição (u, v) no *frame* de bloco.



Fonte: O autor.

Figura 63 – Arquitetura proposta do módulo SVM, destacando a estratégia de organização em matriz de unidades SVM ($U_{l,c}$). As tarefas deste módulo envolvem: o gerenciamento dos dados em coluna na matriz, realizado pelo módulo Controlador de Coluna; ordenamento dos resultados, realizado pelo módulo Controlador de Saída; checagem da existência de pedestres em cada *bounding box*, realizado pelo módulo Checagem de confiança; e o cálculo de confiança, realizado pelas unidades SVM. Esta arquitetura é capaz de lidar com muitas janelas de detecção sobrepostas por meio das unidades SVM, que processam em paralelo e em pipeline um conjunto diferente de janelas.



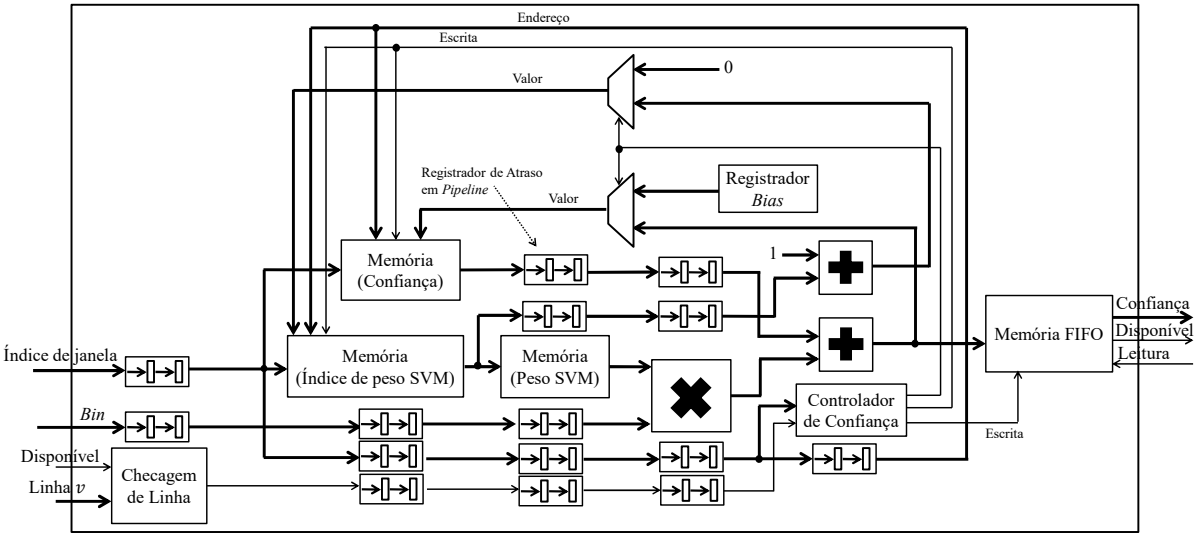
Fonte: O autor.

Para lidar com o desafio da sobreposição, nós propomos unidades SVM para processarem um subconjunto diferente de janelas de detecção em paralelo. Essas unidades são organizadas em uma matriz com N_l linhas e N_c colunas. A quantidade de janelas de detecção dispostas em uma mesma linha é dividida igualmente entre as N_c unidades SVM. As unidades SVM dispostas em linhas processam janelas de detecção das linhas seguintes do *frame* de blocos HOG. A linha inicial v_0 de janelas de detecção, que uma unidade SVM localizada em uma dada linha l na matriz de unidades SVM, é $v_0 = l$.

Um exemplo da distribuição de janelas de detecção por unidades SVM é mostrado na Figura 62. Neste exemplo, nós temos $N_c = 2$ e $N_l = 2$, e a dimensão da janela de detecção é de 2×2 blocos. Nós denotamos $U_{l,c}$ como sendo a unidade SVM localizada na linha l e coluna c na matriz de unidades SVM e $A_{u,v}$ como sendo a janela de detecção cuja borda inferior esquerda está na posição localizada na linha v e coluna u no *frame* de blocos. Como pode ser visto, a unidade $U_{1,1}$ processa as janelas de detecção $A_{1,1}$ e $A_{1,2}$, a unidade $U_{1,2}$ processa janelas de detecção $A_{1,3}$, $A_{1,4}$ e $A_{1,5}$, a unidade $U_{2,1}$ processa as janelas de detecção $A_{2,1}$ e $A_{2,2}$, e a unidade $U_{2,2}$ processa as janelas de detecção $A_{2,3}$, $A_{2,4}$ e $A_{2,5}$. Como pode ser notado, quando a divisão de janelas de detecção em uma determinada

linha não pode ser igual para cada unidade SVM, as unidades SVM da última coluna conterão as janelas restantes da divisão exata.

Figura 64 – Arquitetura proposta da unidade SVM. A tarefa da unidade SVM envolve o cálculo da Equação C.1 para um dado conjunto de janelas de detecção.



Fonte: O autor.

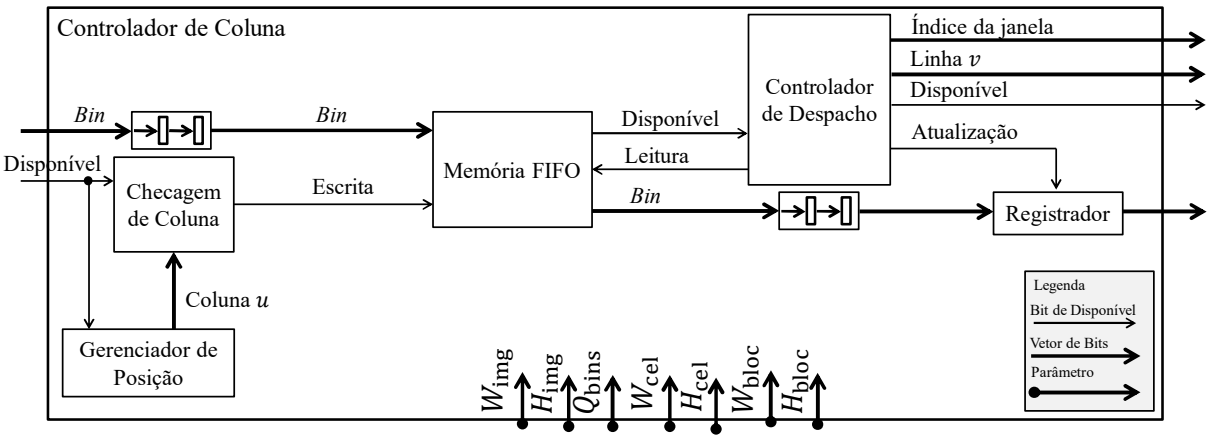
Sempre que qualquer unidade SVM finaliza todas as suas janelas de detecção da sua linha atual, ela salta para processar uma linha adiante que ainda não foi processada por nenhuma outra unidade SVM. A próxima linha a ser processada é a soma da linha atual v de uma determinada unidade SVM e a quantidade de linhas N_l da matriz de unidades SVM. No exemplo da Figura 62, pode-se ver que além das janelas $A_{1,1}$ e $A_{1,2}$, a unidade SVM $U_{1,1}$ processa as janelas $A_{3,1}$ e $A_{3,2}$. Se a nova linha exceder os limites do *frame* de bloco, então a posição reverte para o valor inicial v_0. Esse salto traz uma característica de reusabilidade das unidades SVM, permitindo processar todo o *frame* com apenas uma matriz com poucas unidades SVM.

O parâmetro N_c deve ser definido de acordo com a vazão de pixels de entrada da fonte externa, para garantir a finalização do processamento de todas as janelas de detecção da linha atual antes da chegada dos blocos da linha seguinte. Esse requisito de finalização garante um fluxo de processamento ininterrupto e sem perda de *frames*. Se não for satisfeito, então algum mecanismo precisa ser incorporado para bloquear a entrada de pixels ou armazenar estes pixels em memórias maiores. A definição do parâmetro N_l é diferente. A quantidade de janelas de detecção sobrepostas em linhas diferentes é $\mathrm{H_{SVM}}/\mathrm{H_{cel}}$, onde $\mathrm{H_{SVM}}$ e $\mathrm{H_{cel}}$ são as alturas em pixels da janela de detecção e da célula HOG. Logo, o parâmetro N_l precisa ser, pelo menos, essa quantidade de linhas, para garantir o processamento das janelas sobrepostas das linhas vizinhas.

A arquitetura proposta que processa esta lógica é apresentada na Figura 63. Os módulos Controladores de Coluna armazenam temporariamente as características HOG (ou

seja, os *bins* de blocos) das janelas de detecção em memória FIFO para cada coluna c da matriz de unidades SVM. O módulo Controlador de Coluna é apresentado na Figura 65. Sempre que uma caraterística está disponível nesta FIFO, o módulo Controlador de Despacho o envia para todas as unidades SVM de sua respectiva coluna c. Esse módulo gerencia a posição de linha v no *frame* de bloco de cada característica e envia essa posição junto com a característica. O controlador também envia os índices das janelas de detecção que pertencem a esta característica. O processamento dentro da unidade SVM segue uma estratégia sequencial. Assim, a mesma característica é enviada várias vezes a cada ciclo de *clock* com o valor do índice da janela de detecção incrementado para cobrir todas as janelas que precisam ser processadas. Quando todos os índices são enviados, então o processamento desta característica está finalizado e o Controlador de Coluna busca a próxima característica disponível na memória FIFO.

Figura 65 – Controlador de Coluna



Fonte: O autor.

A arquitetura de processamento da unidade SVM é apresentada na Figura 64. Processamento na unidade SVM segue uma estratégia sequencial e de *pipeline*. O processamento da janela de detecção através da unidade SVM começa acessando o índice de peso SVM a partir do índice da janela de detecção. O valor do peso SVM, obtido a partir do índice de peso, é multiplicado pela característica de entrada. Esse valor é adicionado ao valor de confiança temporário armazenado em uma memória indexável e armazenado de volta nessa memória. Esta memória armazena os valores de confiança temporários de todas as janelas de detecção.

O módulo Checagem de Linha é responsável por permitir as atualizações da memória de confiança e índices. Este módulo verifica se as características de entrada pertencem às unidades SVM de sua respectiva linha. Por exemplo, as características da linha 1 não pertencem às unidades SVM da linha 2, conforme mostrado na Figura 62. Quando pertence, este módulo emite um sinal para o módulo Controlador de Confiança para atualizar a memória de confiança e a memória de índice com o novo valor de confiança

temporário e o próximo índice de peso do SVM.

Quando a última característica de uma determinada janela de detecção é processada, a confiança desta janela está pronta e já pode ser disponibilizada na saída da unidade SVM. Neste ponto, o índice do peso SVM é zerado para processar uma nova janela em outra linha. Além disso, o valor de confiança temporário armazenado na memória retorna ao valor inicial, que é o *bias* a partir da Equação C.1.

As unidades SVM vizinhas podem finalizar algumas janelas de detecção simultaneamente e fora ordem. Essa situação acontece principalmente no processamento da última linha da janela de detecção em que as primeiras janelas da unidade SVM seguinte finalizam primeiro do que as últimas janelas da unidade SVM anterior. Como a leitura de todos os resultados de confiança é feita uma de cada vez e na ordem disposta pelo fluxo do *frame*, essa simultaneidade das respostas das unidades SVM pode levar à perda de resultados. Para garantir que os resultados das unidades SVM não sejam perdidos, eles são armazenados em uma pequena memória FIFO de saída, conforme mostrado na Figura 64.

O módulo Controlador de Saída mostrado na Figura 63 gerencia as saídas de todas as unidades SVM. Este módulo garante que a leitura e entrega dos resultados das unidades SVM estejam na ordem do fluxo de entrada das características. Por exemplo, se a unidade SVM a ser lida tiver resultados armazenados na memória de saída, o módulo Controlador de Saída chaveia a saída do multiplexador para que sejam lidos os dados dessa unidade SVM a fim de serem entregues ao módulo Checagem de Confiança. Este módulo, por sua vez, verifica se a janela contém ou não um pedestre seguindo a Equação C.2 e também verifica se é a última janela do *frame* atual. Como os resultados já estão em ordem, ou seja, seguindo o fluxo da esquerda para a direita e de cima para baixo, o módulo de Checagem de confiança gerencia e entrega as posições das extremidades do *bounding box* de cada valor de confiança.

5.2.3.2 Versão 2: Estratégia de compartilhamento de recursos

Uma vez entendido o fluxo de processamento proposto pela versão 1 da arquitetura SVM, nós descrevemos, nesta seção, otimizações realizadas para redução dos recursos computacionais de FPGA. Várias abordagens foram implementadas para reduzir a exigência de memória, tais como: (1) reuso da memória de pesos e índices pelas unidades SVM, (2) reuso de memória de pesos por colunas adjacentes na matriz SVM e, (3) reuso de memória de resultados por unidades SVM, como descrito a seguir. É importante ressaltar que todas as otimizações não impactaram no desempenho de processamento do módulo SVM, uma vez que fluxo de processamento das janelas não foi alterado.

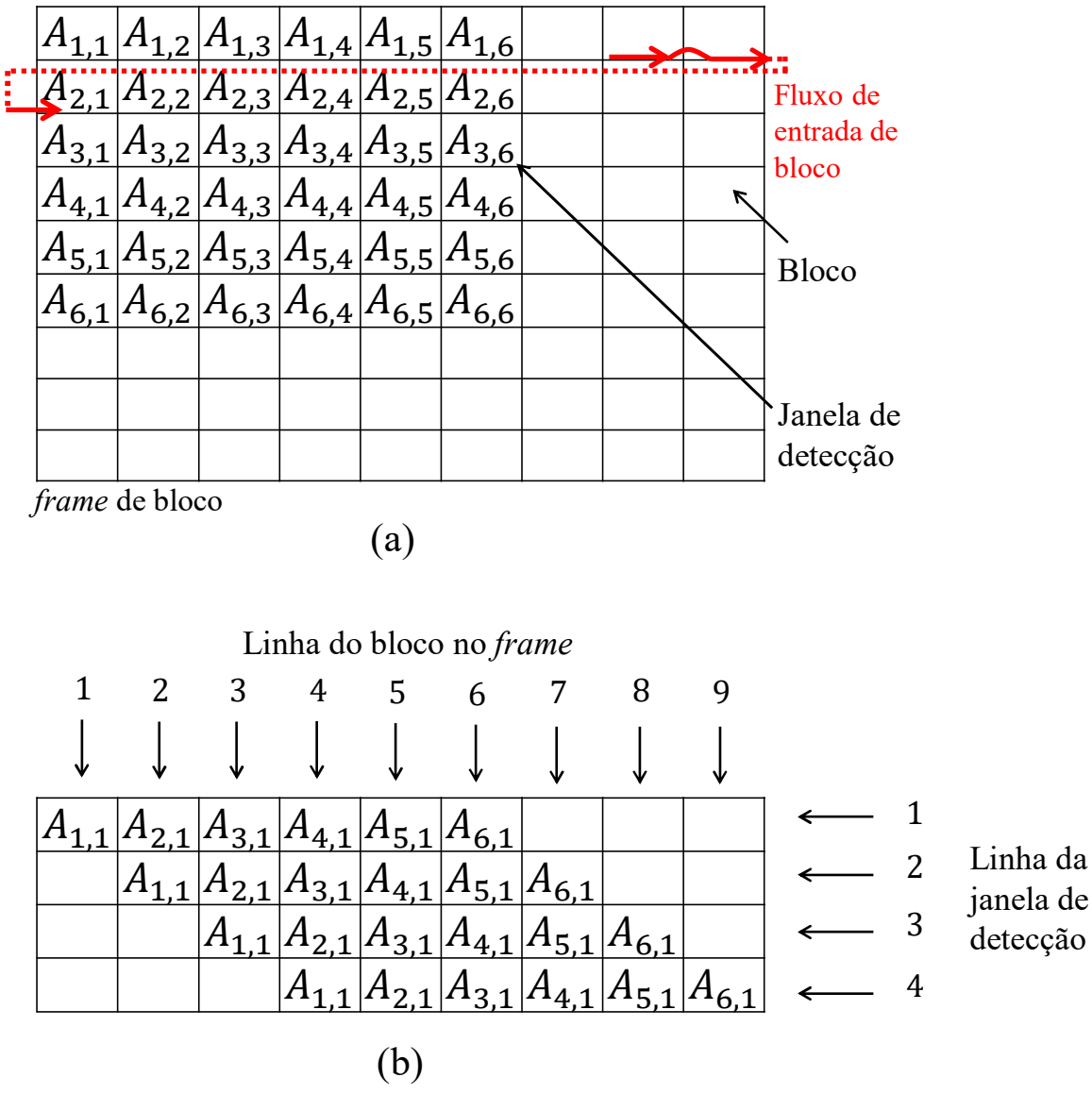
5.2.3.2.1 Reuso de memória de pesos e índices

Como pode ser observado na versão 1 da arquitetura, as memórias de peso SVM contendo os mesmos pesos estavam replicadas para todas as unidades SVM. Uma vez que precisamos garantir o fluxo de processamento de um pixel por ciclo sem interrupção, a quantidade de unidades SVM necessária depende da dimensão da janela de detecção, da quantidade de *bins* e da dimensão da célula. Para entender melhor, considere, como exemplo, um detector com célula de dimensão 8×8 pixels, 9 *bins* e uma janela de detecção de 64×128 pixels. A quantidade N_l de linhas de unidades SVM é a quantidade de blocos na vertical, necessária para compor a janela de detecção, ou seja, é a altura da janela dividida pela altura da célula menos 1. Assim, no exemplo, $N_l = 128/8 - 1 = 15$. A quantidade de colunas N_c é definida de maneira experimental e nós observamos que 4 colunas seria uma quantidade suficiente para suportar, no pior caso, um fluxo de entrada de um pixel por ciclo sem parar. Logo, nós teremos $15 \times 4 = 60$ unidades SVM. Essa quantidade de unidades exige uma grande quantidade de memória devido a replicação da memória de pesos SVM, armazenando os mesmos pesos SVM. Essa quantidade dificulta a escalabilidade do detector para grandes profundidades de pirâmide e quantidade de P-SVMs.

Para reduzir esta quantidade de memória replicada, nós exploramos aspectos temporais e espaciais a partir do fluxo de entrada de pixel adotado. Importante relembrar que o fluxo de entrada de pixel é sempre da esquerda para direita e de cima para baixo. Esse fluxo se propaga para o fluxo de entrada de bloco no módulo SVM. Com esse fluxo nós podemos perceber dois aspectos com relação às localizações dos blocos que estão sendo processados por cada unidade SVM.

O primeiro aspecto é com relação ao processamento das linhas de uma janela de detecção. Quando uma unidade SVM está processando uma determinada linha da janela de detecção, a unidade SVM da linha seguinte na mesma coluna da matriz SVM está processando a linha anterior da próxima janela de detecção. A Figura 66 exemplifica esse comportamento. Como pode ser visto nessa figura, quando o processamento do SVM está localizado na linha de bloco 1, apenas a linha 1 da janela de detecção $A_{1,1}$ está, na verdade, sendo processada. Quando o processamento está localizado na linha de bloco 2, a linha 2 da janela de detecção $A_{1,1}$ e a linha 1 da janela de detecção $A_{2,1}$ estão sendo processadas. Quando o processamento está localizado na linha de bloco 3, a linha 3 da janela de detecção $A_{1,1}$, a linha 2 da janela de detecção $A_{2,1}$ e a linha 1 da janela de detecção $A_{3,1}$ estão sendo processadas. Essa lógica se repete ao longo de todas linhas do *frame* de bloco. Quando nós referenciamos a linha de uma janela de detecção nós estamos também referenciando a linha do peso SVM. Assim, as unidades SVMs dispostas na mesma coluna na matriz de unidades SVM estarão sempre processando uma linha de peso diferente.

Figura 66 – Ilustração do aspecto das janelas de detecção de linhas diferentes sendo processadas pelas unidades SVM: (a) Distribuição das janelas na imagem de blocos. (b) Descrição da linha dentro da janela que está sendo processada por cada linha de bloco da imagem de bloco. Nessa ilustração temos um *frame* de bloco de 9 × 9 blocos e janelas de detecção de 4 × 4 blocos. O termo $A_{v,u}$ significa janela de detecção na posição (u, v) no *frame* de bloco.



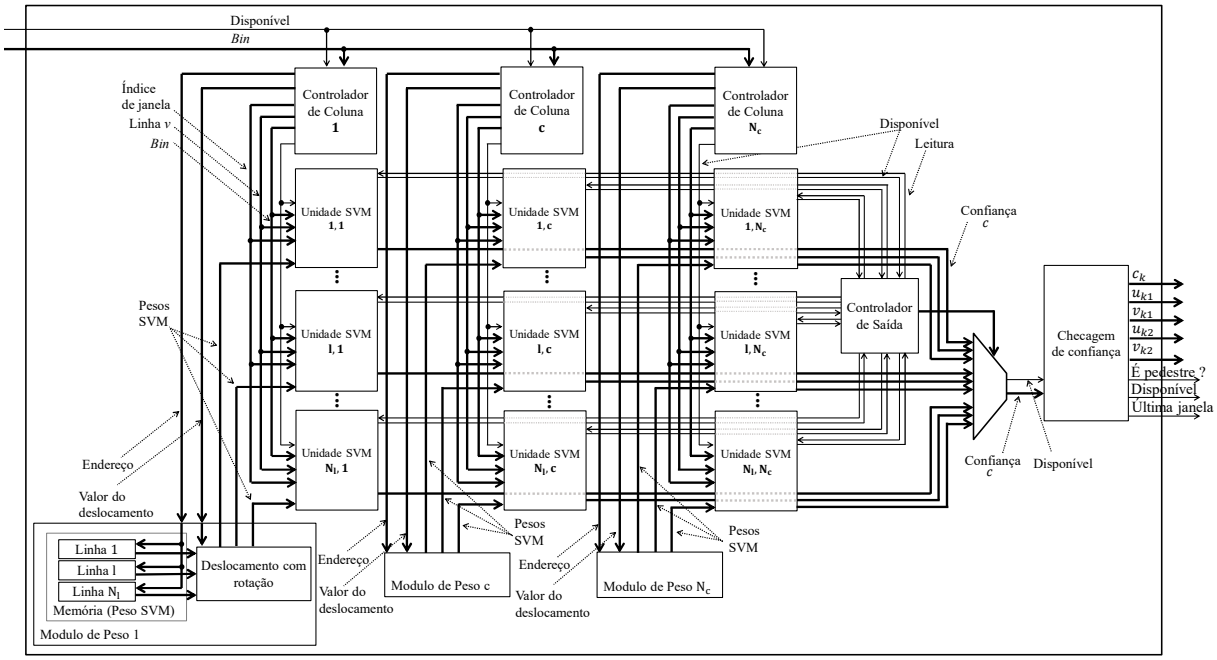
Fonte: O autor.

O segundo aspecto é com relação ao processamento de uma determinada coluna de uma janela de detecção. Nós podemos observar na Figura 66(a) que as unidades SVM de uma determinada coluna da matriz SVM estão sempre processando a mesma posição de coluna. Esse comportamento deve-se ao fato de que sempre as unidades SVM recebem o mesmo fluxo de dados, dessa forma, têm o mesmo padrão de processamento, mesmo processando linhas diferentes da janela de detecção.

A partir desse entendimento, nós propomos a modificação da versão 1 da arquitetura do módulo SVM, removendo as memórias de pesos SVM e índices, a partir das unidades SVM,

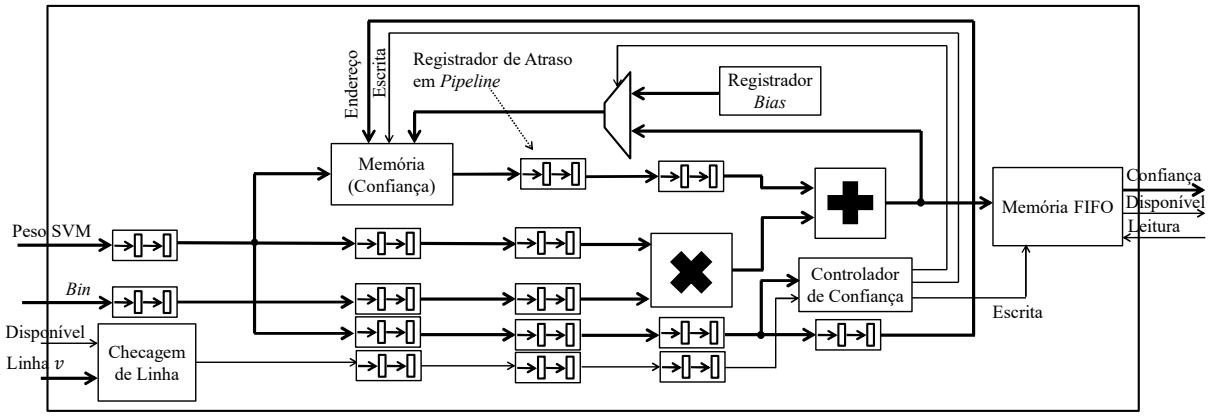
e definindo uma memória de pesos SVM e índices compartilhadas com todas as unidades SVM de mesma coluna na matriz. As Figuras 67 e 68 apresentam, respectivamente, a matriz de unidades SVM e na unidade SVM com essa estratégia de compartilhamento. Essas memórias compartilhadas residem dentro do módulo de Pesos que foi criado para cada coluna de unidades SVM e a memória de índice foi transferida para dentro do módulo Controlador de Coluna que agora irá gerenciar os índices de acesso a memória de pesos.

Figura 67 – Módulo SVM modificado a partir da versão 1, com a adição do módulo de Peso em cada coluna da matriz de unidades SVM, para o compartilhamento de memória de pesos, e do compartilhamento dos índices de janela entre colunas, que agora reside dentro do Controlador de Coluna.



Fonte: O autor.

Figura 68 – Unidade SVM modificada a partir da versão 1 com adição da estratégia de compartilhamento de memória de pesos e índices.

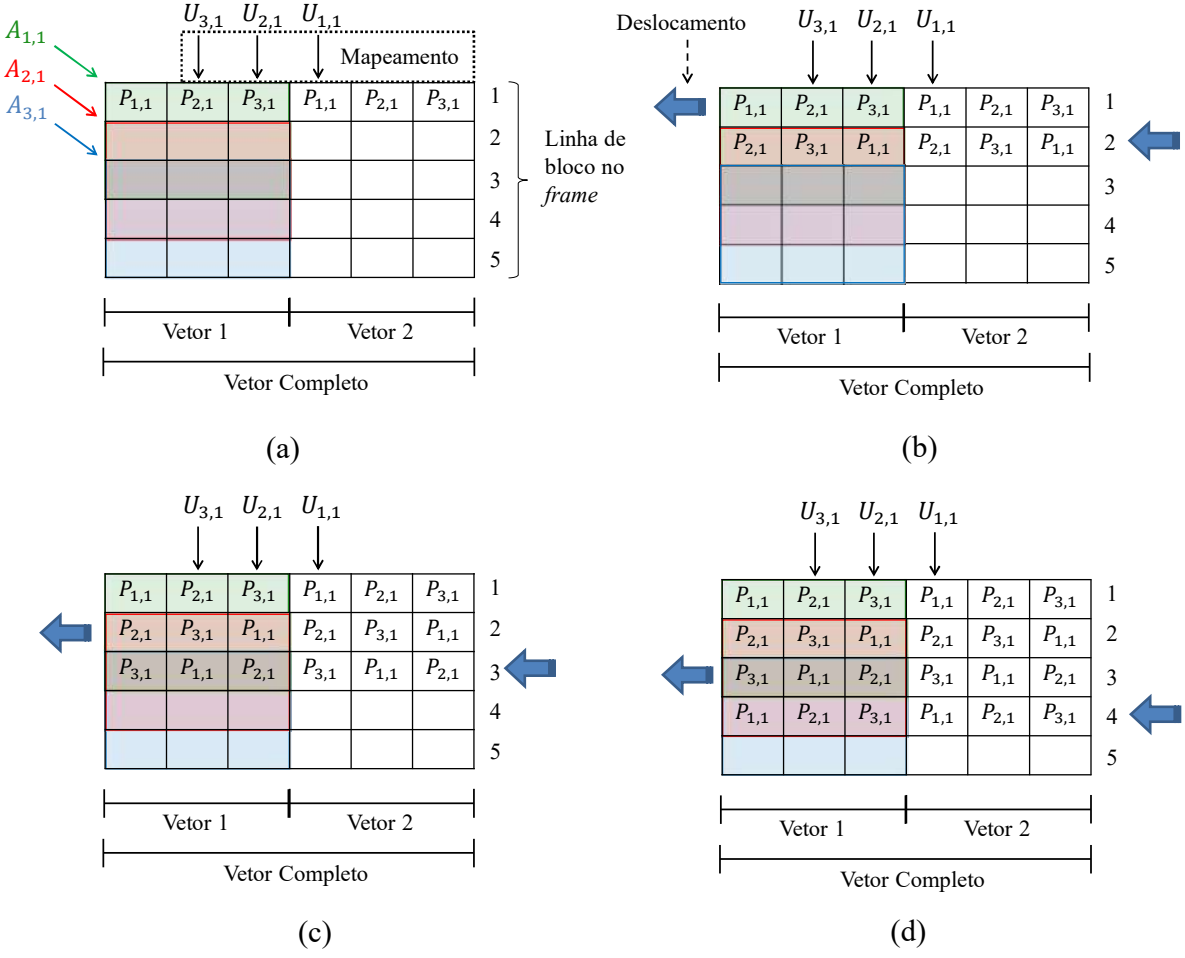


Fonte: O autor.

Para que as unidades SVM tenham acesso a sua respectiva linha de pesos SVM, a memória de pesos, que antes era única na versão 1 da arquitetura, é dividida em linhas que correspondem as linhas da janela de detecção. No módulo de Pesos, os pesos que saem de cada memória são enviados para o módulo de deslocamento rotacional que monta esses dados como um vetor e realiza deslocamentos para esquerda com rotação para garantir o correto alinhamento dos pesos pelas unidades SVM. A quantidade de deslocamentos D é definida pela Equação 5.8, onde v é a posição da linha do bloco que está sendo processado no *frame*. Essa equação é calculada pelo módulo Controlador de Coluna.

$$D(v) = v \bmod \left(\frac{H_{\text{SVM}}}{H_{\text{cel}}} - H_{\text{bloc}} + 1 \right) \tag{5.8}$$

Figura 69 – Exemplo da lógica de vetorização dos pesos e deslocamento rotacional: (a) primeira, (b) segunda, (c) terceira e (d) quarta linha do *frame* de blocos que está sendo processada. O termo $U_{l,c}$ significa unidade SVM na posição (l, c) na matriz de unidades SVM. O termo $P_{l,c}$ significa peso da linha l da coluna c na matriz de unidades SVM. O termo $A_{v,u}$ significa janela de detecção na posição (u, v) no *frame* de bloco. Nesse exemplo nós temos três linhas de unidades SVM e janela de detecção de 3×3 blocos.



Fonte: O autor.

A estratégia para montagem dos pesos em um vetor junto com o deslocamento rotacional é crucial para o correto alinhamento dos pesos nas unidades SVM ao longo de todo

o processamento do *frame*. Para entender essa estratégia, a Figura 69 ilustra um exemplo da sua aplicação. Os pesos são replicados em um vetor e as unidades SVM são mapeadas a partir do primeiro peso do vetor 2. No exemplo da Figura, na linha de bloco 1 do *frame*, a primeira unidade $U_{1,1}$ aponta para a primeira linha de peso $P_{1,1}$, a segunda unidade $U_{2,1}$ aponta para a terceira linha de peso $P_{3,1}$ e a unidade $U_{3,1}$ aponta para a segunda linha de peso $P_{2,1}$. Embora as unidades $U_{2,1}$ e $U_{3,1}$ estejam apontando para algum peso, elas ainda não começaram a processar nenhuma janela de detecção, até que a sua primeira linha de bloco esteja disponível.

Quando a linha de bloco 2 na imagem está sendo processada, o vetor de pesos referente a linha 1 é deslocado para a esquerda em uma posição. Como é um deslocamento rotacional, o último peso $P_{1,1}$ é posto no início do vetor. Com esse deslocamento, as unidades $U_{1,1}$, $U_{2,1}$ e $U_{3,1}$ apontam, respectivamente, para as linhas de pesos $P_{2,1}$, $P_{1,1}$ e $P_{3,1}$. Quando a linha de bloco 3 está sendo processada, o vetor de pesos referente a linha 2 é deslocado para a esquerda em uma posição. Como é um deslocamento rotacional, o último peso $P_{2,1}$ é posto no início do vetor. Com esse deslocamento, as unidades $U_{1,1}$, $U_{2,1}$ e $U_{3,1}$ apontam, respectivamente, para as linhas de pesos $P_{3,1}$, $P_{2,1}$ e $P_{1,1}$.

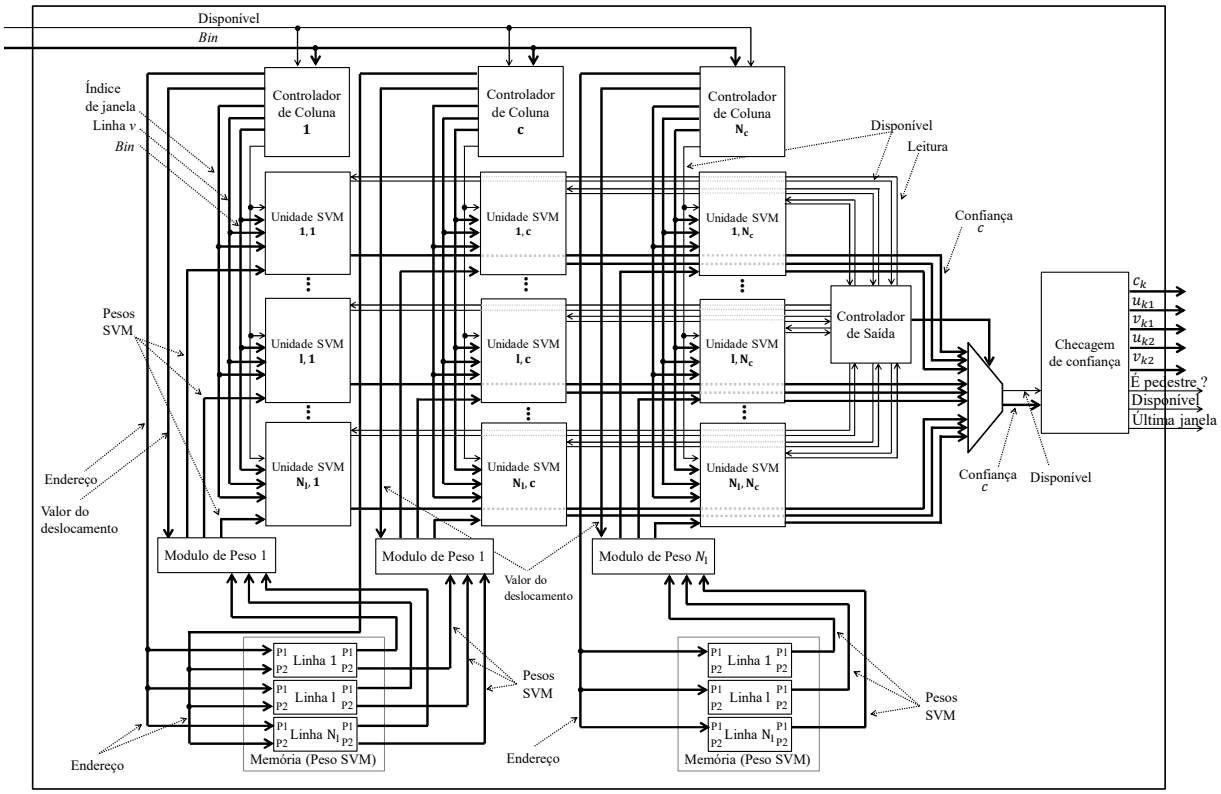
É possível observar na Figura 69 que a unidade $U_{1,1}$ termina o processamento da janela $A_{1,1}$ na linha de bloco 3 e começa a processar a próxima janela, $A4, 1$, na linha de bloco 4. Nessa linha, o vetor de pesos terá dado uma volta completa, estando corretamente alinhado para que a primeira linha de pesos seja lida novamente pela unidade $U_{1,1}$. Assim, essa lógica se repete até o final do *frame*. Quando a última janela da coluna c da matriz for processada, o vetor é reiniciado para o estado da linha de bloco 1 para processar os próximos *frames*, mesmo que o vetor não tenha dado uma volta completa.

Nós tomamos o exemplo do começo dessa seção, onde nós temos um detector com célula de dimensão 8×8 pixels, 9 *bins* e uma janela de detecção de 64×128 pixels. Comparado com a versão 1 da arquitetura SVM, essa arquitetura otimizada permitiu uma redução de 60 memórias de pesos e índices para apenas 4 memórias, uma para cada coluna da matriz de unidades SVM, considerando que precisamos de 4 colunas nesta matriz. Uma vez que estas memórias possuem internamente lógicas de processamento, essa redução de memória provocou uma redução também na quantidade de elementos lógicos, como está evidenciado na Seção 5.4.4.1.2 de resultados.

5.2.3.2.2 Reuso de memória de pesos entre colunas vizinhas

A memória utilizada para armazenamento dos pesos SVM é por padrão de duas portas para leitura e escrita de dados. Assim, nós propomos, uma redução de memória de pesos, a partir da versão descrita na Seção 5.2.3.2.1, através do uso da mesma memória para duas colunas vizinhas na matriz de unidades SVM, como mostrado na Figura 70. Nós utilizamos as duas portas para leitura de pesos, uma para cada coluna da matriz.

Figura 70 – Arquitetura do SVM com a redução de memória de peso através do uso da mesma memória para duas colunas de unidades SVM.



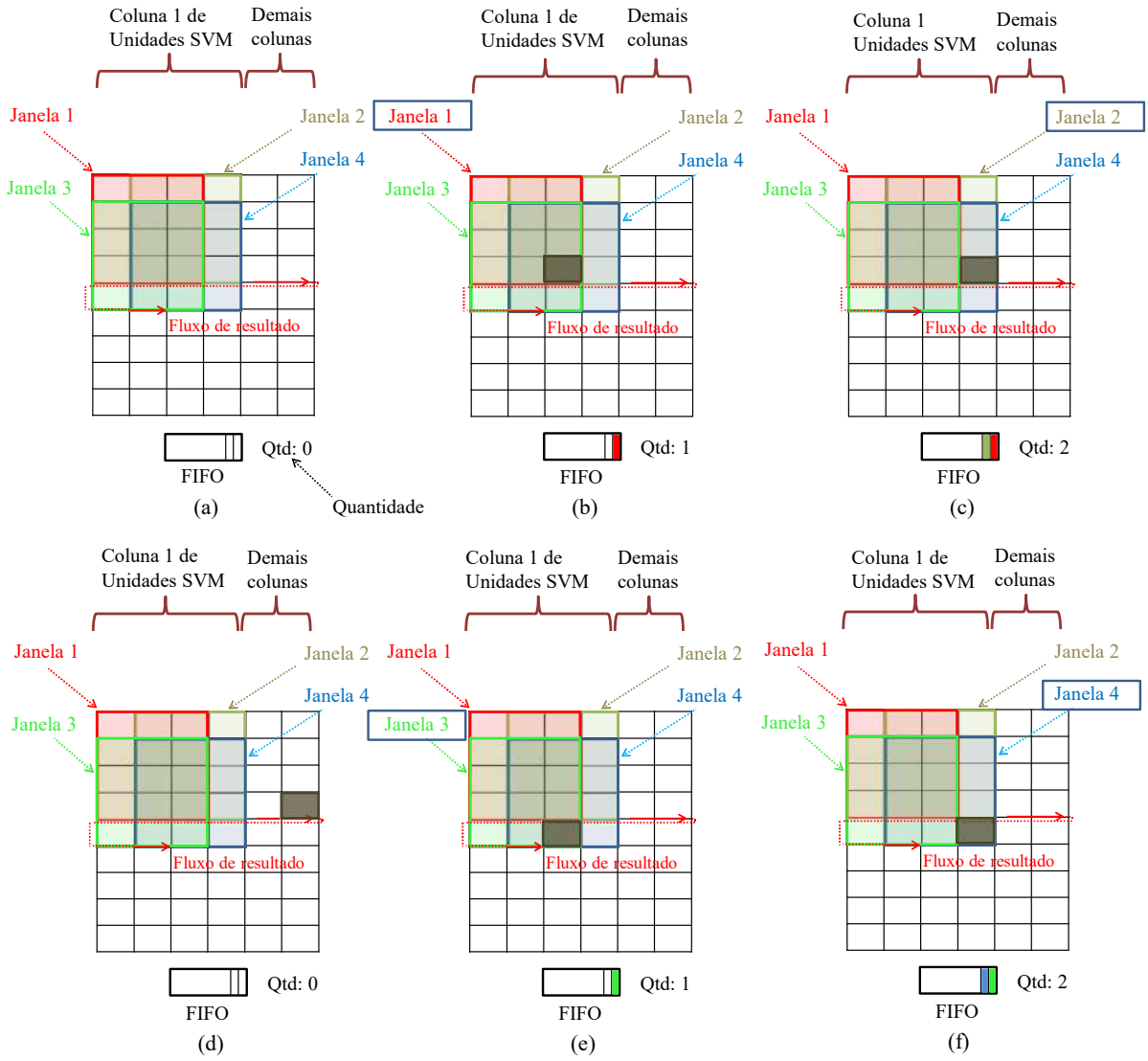
Fonte: O autor.

Essa solução não precisou de adição de nenhuma lógica de processamento, uma vez que a memória já disponibiliza as duas portas para leitura. Como não há necessidade de mudança de pesos ao longo do processamento, então não é necessário o uso da porta para escrita de dados. Nós podemos carregar os pesos através de um arquivo de inicialização no momento da síntese de código. Se a quantidade de colunas na matriz SVM não for par, a última memória estará sendo utilizada por apenas uma coluna de unidades SVM. Essa solução reduz pela metade a quantidade de memória de pesos a partir da solução otimizada da Seção 5.2.3.2.1.

5.2.3.2.3 Reuso de memória de resultados

Como descrita na Seção 5.2.3.1, o resultado de cada unidade SVM é armazenado em uma memória do tipo FIFO para resolver questões de ordenamento dos dados entre as unidades SVM. Esse tipo de memória, além de exigir espaço de armazenamento de FPGA, demanda recurso de processamento para gerenciamento da abordagem de FIFO. Uma vez que nós temos uma grande quantidade de unidades SVM replicadas, a soma dos recursos demandados por este tipo de memória é considerável para todo o detector.

Figura 71 – Ilustração do fluxo de escrita de resultados nas memórias FIFO pelas unidades SVM de linhas diferentes: (a) Cenário inicial; (b) Chegada do último dado da janela 1, finalização dessa janela e escrita em memória do resultado; (c) Chegada do último dado da janela 2, finalização dessa janela e escrita em memória desse resultado; (d) Chegada do último dado de linha do *frame* e esvaziamento da memória de resultados; (e) Chegada do último dado da janela 3, finalização dessa janela e escrita em memória do resultado; (f) Chegada do último dado da janela 4, finalização dessa janela e escrita em memória do resultado.

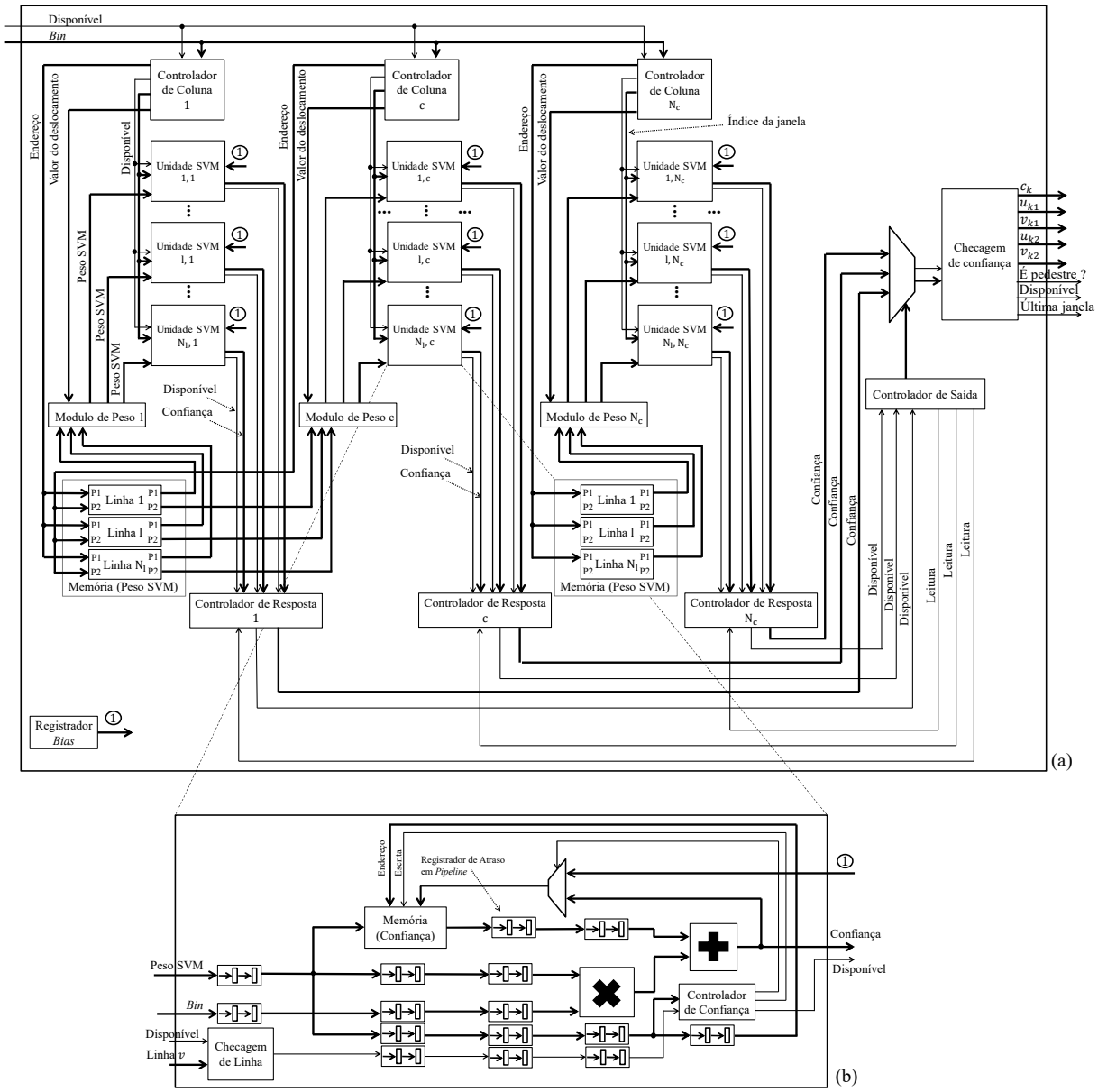


Fonte: O autor.

Seguindo o fluxo adotado de entrada de pixels, nós verificamos dois tipos de comportamento de entrega dos resultados das unidades SVM. O primeiro tipo de comportamento é com relação às unidades SVM de colunas diferentes para a mesma linha. Estas unidades entregam dados ao mesmo tempo no momento de transição entre uma coluna para a próxima. Utilizar uma memória para uma linha não é possível por conta dessa situação de concorrência na transição.

O segundo tipo de comportamento é com relação às unidades SVM de linhas diferentes para a mesma coluna. A Figura 71 exemplifica esse segundo tipo de comportamento. Nesse

Figura 72 – Arquitetura do módulo SVM com o compartilhamento da memória de resultado por coluna na matriz. A diferença da arquitetura descrita na Figura 63 para esta arquitetura está no surgimento do módulo Controlador de Resposta que contém a memória de resultados, que foi removida da unidade SVM, a lógica para escrita dos resultados das unidades de sua respectiva coluna e disponibilização destes resultados para o Controlador de Saída.



Fonte: O autor.

exemplo temos duas janelas de detecção para cada unidade SVM de linhas diferentes. Na Figura 71(a) nós temos o cenário inicial, com a memória FIFO vazia. Quando chegam os últimos dados da janela 1 e 2, ilustrados pelas Figuras 71(b) e (c), os seus resultados são armazenados na memória. Na mesma linha de blocos, outros dados de característica chegam, mas não são para as janelas desta coluna de unidades, como mostrado pela Figura 71(d). Enquanto estes dados estão sendo disponibilizados e processados pelas outras unidades SVM, a memória de resultados está sendo esvaziada. Quando os dados de característica da linha seguinte estão disponíveis, a memória já está vazia e pode armazenar

os resultados de novas janelas, como mostrado nas Figuras 71(e) e (f). Essa situação de esvaziamento da memória sempre acontece ao longo de todo *frame*, pois entre duas linhas de janelas detecção existem muitos dados para receber e muita espera para formação de células e blocos HOG.

A partir desta verificação, nós percebemos que é possível ter uma memória do tipo FIFO por coluna na matriz SVM para armazenar os resultados das unidades SVM de cada coluna. Dessa forma, nós propomos a remoção das memórias de resultados a partir das unidades SVM e a utilização de memórias por coluna na matriz SVM. A arquitetura otimizada com essa solução é mostrada na Figura 72.

A memória reside no módulo Controlador de Resposta e é gerenciado por este módulo. A lógica de gerenciamento é baseada em registrador de índice e lógica de seleção que escolhe qual a unidade SVM que terá seus resultados enviados para a memória do tipo FIFO. Quando todos os resultados da unidade SVM atual tiverem sido enviados então o índice é incrementado. Quando a unidade SVM da última linha envia todos os resultados, então o índice volta para o início.

5.2.4 Serializador

O módulo Serializador gerencia os resultados de detecção de todos os módulos P-SVMs em uma única saída. Este módulo tem um conjunto de memórias do tipo FIFO e uma memória para cada módulo SVM. Os módulos SVM escrevem seus resultados nestas memórias. Como descrito na Seção 5.2.3.1, o módulo Checagem de Confiança a partir do módulo SVM, envia os *bounding boxes* bem como os valores de confiança das janelas de detecção classificadas como contendo pedestres. Este módulo também sinaliza o processamento da última janela de detecção de um *frame*. Todas essas informações são enviadas para a memória.

Um sinal de uma fonte externa é definido no módulo Serializador para sincronização do detector com os sistemas externos. Quando a fonte externa não levanta esse sinal os resultados dos módulos P-SVMs são acumulados na memória. Quando a fonte externa levanta por um ciclo de clock esse sinal então o módulo Serializador começa a ler os dados a partir dessa memória. O módulo Serializador lê esses resultados um de cada vez e os envia para a saída do detector. Quando a informação de fim de *frame* a partir da memória atual é lida, então o módulo Serializador busca a próxima memória. Ao finalizar o processamento da última memória, o Serializador informa o término do processamento de um *frame*, ou seja, todos os resultados de todos os módulos SVMs foram enviados. Em seguida, o módulo Serializador volta para apontar para a primeira memória e espera novamente o sinal externo para ler novamente as memórias, agora com os dados do próximo *frame*.

O módulo Serializador é importante por dois motivos. O primeiro é com relação ao ambiente externo cuja restrições de paralelismo variam por aplicação. A serialização de

dados cria uma interface comum de fácil integração do módulo detector de pedestres para qualquer aplicação através de memórias FIFO e módulos de empacotamento de dados. A memória cria o padrão de dados necessário para a aplicação. Por exemplo, considere um barramento de dados que envia quatro *bounding boxes* de uma vez para serem utilizados por outro processador. Nós inserimos um módulo de empacotamento, que agrupa em um único conjunto de dados quatro *bounding boxes*, e armazenamos na FIFO para ser consumidos pelo barramento. Uma explicação mais detalhada do uso de módulos de empacotamento e memórias FIFO pode ser encontrada no Capítulo 6.

O segundo motivo é com relação ao fluxo de dados e desempenho de processamento. Claramente, nós sabemos que a serialização pode criar um gargalo de desempenho. Contudo, a quantidade de dados, ou seja, os *bounding boxes*, que é trafegada é muito pequena, comparado com a quantidade de janelas que é processada pelos módulos SVM. É importante lembrar que os *bounding boxes* que são enviados para o serializador são apenas aqueles cujo valor de confiança passou do parâmetro de limiar de confiança do SVM. Para valor alto de limiar, menos *bounding boxes* são serializados. Assim, o impacto no desempenho de processamento do detector se torna cada vez mais imperceptível à medida que aumentamos o limiar de confiança. Além disso, o atraso do processamento do ambiente externo, que consome estes dados, pode compensar esse pequeno atraso na serialização.

5.3 IMPLEMENTAÇÃO E INTEGRAÇÃO

A arquitetura proposta de hardware do detector de pedestre foi implementada na linguagem SystemVerilog (BERGERON et al., 2006)(SUTHERLAND; DAVIDMANN; FLAKE, 2006) e validada através de simulações de tempo usando o ModelSim 10.5b e Quartus 17.1 *Standard* (PEDRONI, 2020) para síntese de código. O módulo detector juntamente com módulo de correspondência estéreo SGM foram integrados ao sistema PCP usando a plataforma Intel HARP versão 2 (HARPv2) (GUPTA, 2016). Esta plataforma, como descrita na Seção 3.2, consiste em uma plataforma híbrida composta de um processador Intel Xeon E5-2699 v4 e uma placa de aceleração programável (PAC) com um FPGA Intel Arria 10 GX. Nós utilizamos a arquitetura de integração detalhada no Capítulo 6 e adicionamos o detector de hardware. Por simplicidade, nós utilizamos a arquitetura de *buffer* simples.

Cada par de pixels RGB, a partir do *frame* esquerdo e direito, é enviado para o módulo SGM, e o pixel esquerdo é enviado para o módulo de detecção de pedestres. Ambos os módulos estão processando em paralelo e entregando seus resultados ao mesmo tempo. Como a escrita em memória usando a plataforma HARPv2 é realizada um pacote por vez, é necessário que a escrita dos resultados de um módulo termine para que os resultados do próximo módulo sejam escritos. O módulo Escrita em Hardware, mostrado na Figura 82, realiza esse gerenciamento e envio de pacotes para a memória através do barramento. Primeiro, ele envia os dados de todo o mapa de disparidade. Quando termina de enviar, ele solicita ao módulo de detecção, por meio do sinal externo do módulo Serializador,

descrito na Seção 5.2.4, para enviar os resultados de detecção que estão armazenados nas memórias FIFO do módulo Serializador.

5.4 RESULTADOS

Esta seção apresenta os resultados do detector de pedestre implementado em hardware e integrado ao hardware proposto para a localização e previsão de colisão. O detector de hardware inclui a versão 2 do módulo SVM cujas otimizações propostas são descritas na Seção 5.2.3.2. A estratégia de avaliação é similar a estratégia descrita na Seção 4.3. Algumas outras avaliações foram realizadas como ocupação de recursos em FPGA, desempenho de processamento e dissipação de potência. Por fim, apresentamos nas Seções 5.4.6 e 5.4.7 comparações com alguns trabalhos em hardware de detectores de pedestres e sistemas PCP.

5.4.1 Avaliação da Localização

Nós utilizamos a base de dados 1 para avaliação da abordagem de localização. Primeiramente, nós apresentamos as configurações dos sistemas envolvidos nos experimentos e em seguida apresentamos os resultados e análises.

5.4.1.1 Definição dos Sistemas e Avaliação Geral

Os três detectores avaliados são similares aos adotados na Seção 4.3.1. Nós definimos P-SVM$_1$ com tendo uma janela de detecção de dimensão 64 $\times$ 128 pixels e P-SVM$_2$ com uma janela de detecção de dimensão de 48 $\times$ 96 pixels. A partir destes dois módulos P-SVMs nós instanciamos em nível de síntese de FPGA os três detectores, um com apenas o P-SVM$_1$, outro com apenas P-SVM$_2$ e o detector com os dois módulos P-SVMs combinados.

As dimensões de célula e bloco do módulo HOG são, respectivamente, 8 $\times$ 8 e 16 $\times$ 16 pixels. O parâmetro *stride* $S_{\text{P-SVM}}$ de qualquer P-SVM é definido como sendo igual a 8 pixels em ambas as direções do *frame*. O parâmetro de quantidade de *bins* de histograma HOG é definido como 9. O parâmetro de escala de pirâmide é definido como 1,1. A parte fracionária de todos os detectores, para todas as etapas de processamento, possui 8 bits de largura (ou seja, $C_{\text{fração}} = 8$), que é uma largura que mantém a precisão do detector (SULEIMAN; SZE, 2016).

A abordagem de treinamento para todos os detectores avaliados é similar às abordagens descritas nas Seções 4.3.1.1.1 e 4.3.1.1.2.

5.4.1.2 Estratégia de Avaliação

Para avaliação dos detectores nós seguimos a mesma estratégia do capítulo anterior. Nós avaliamos a capacidade da pirâmide do detector de hardware, dos detectores com janelas

de detecção de dimensão única isoladas e do detector com combinação destas janelas. Além disso, nós comparamos os resultados do detector do capítulo anterior implementado usando a biblioteca OpenCV com os resultados obtidos com detector proposto em hardware. Com relação ao sistema PCP nós utilizamos o módulo SGM-FPGA, descrito na Seção 3.2, para etapa de correspondência estéreo. Todos os parâmetros do SGM-FPGA e do sistema PCP são iguais aos utilizados no capítulo anterior, definidos na Seção 4.3.1.2.

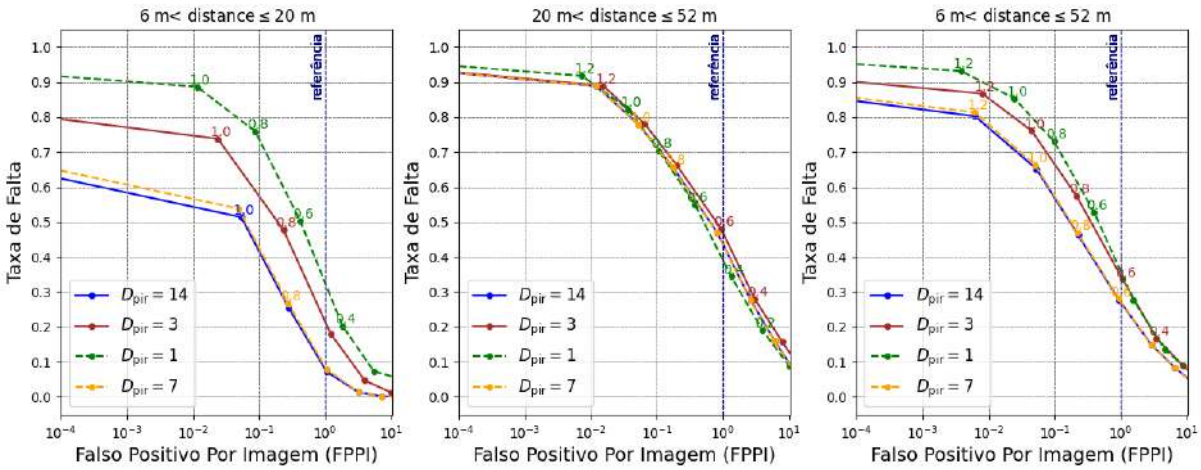
5.4.1.3 Experimentos, análises e comparações

A partir da definição dos módulos detectores, do sistema PCP e da estratégia de avaliação, nós realizamos os experimentos que serão descritos na seções seguintes.

5.4.1.3.1 Pirâmide de Imagem e Janelas de Detecção de Dimensão Única

Esse experimento foi realizado nos detectores de hardware com janelas de uma única dimensão variando o parâmetro de profundidade de pirâmide. O intuito desse experimento foi analisar o impacto da pirâmide de imagens implementada em hardware e das janelas de dimensões diferentes na qualidade da detecção de pedestres. Os valores de profundidade da pirâmide utilizada são 1, 3, 7 e 14. Nesse experimento, a filtragem geométrica é considerada. A base de dados é dividida em grupos de distâncias como descrito na Seção 3.3.1.

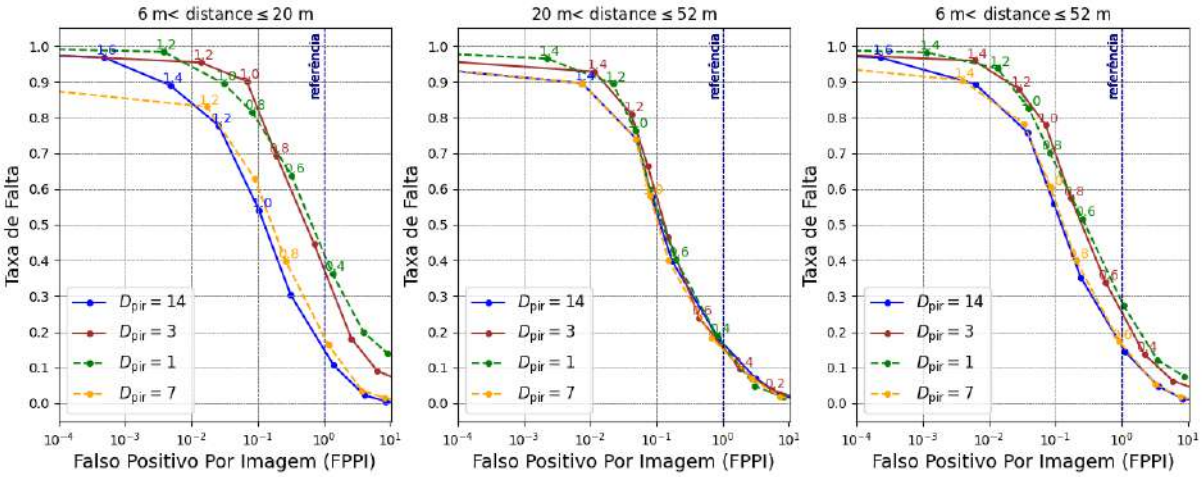
Figura 73 – Resultados da qualidade do detector de hardware com janela de detecção de dimensão 64 × 128 pixels. Para obter cada resultado da taxa de faltas e FPPI, os valores de confiança σ_{SVM} são gerados no intervalo de [2 , 0] com passos de 0,2.



Fonte: O autor.

As Figuras 73 e 74 apresentam os resultados para o detector com janela de dimensão 64 × 128 e 48 × 96 pixels respectivamente. Nós podemos evidenciar as mesmas tendências de redução da taxa de faltas com o aumento da profundidade da pirâmide, já observados na Seção 4.3.1.3.1. O detector de janela de 64 × 128 pixels, com a profundidade de 14,

Figura 74 – Resultados da qualidade detector de hardware com janela de detecção de dimensão 48 × 96 pixels. Para obter cada resultado de taxa de faltas e FPPI, os valores de confiança σ_{SVM} foram gerados no intervalo de [2 , 0] com passos de 0,2.



Fonte: O autor.

considerando o FPPI de referência, alcançou a taxa de faltas de 8% para o grupo 1 e com a profundidade de 1, o detector alcançou uma taxa de 32%. O detector de janela de 48 × 96 pixels, com a profundidade de 14, considerando o FPPI de referência, alcançou a taxa de falta de 16% para o grupo 1 enquanto que com a profundidade de 1, o detector alcançou uma taxa de faltas de 40%.

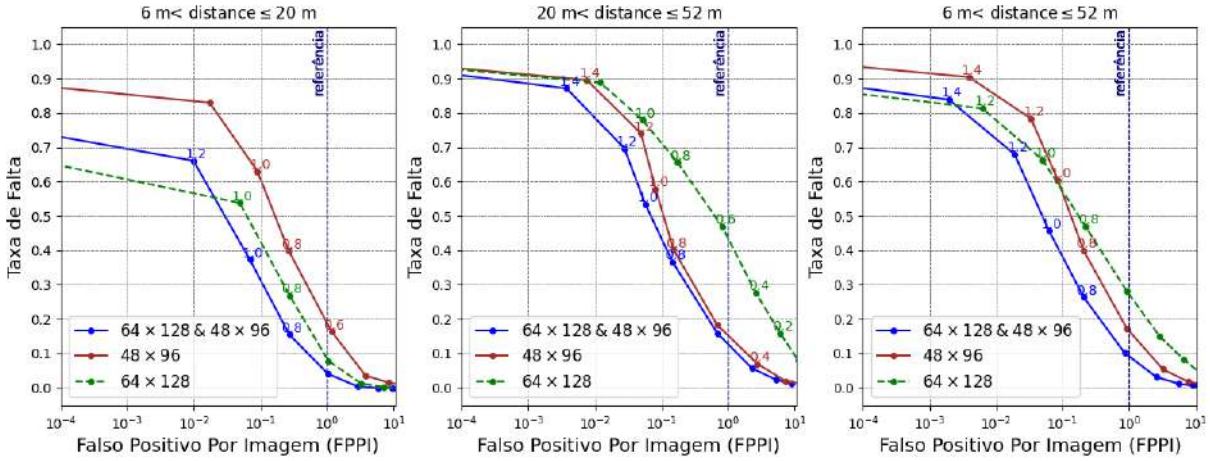
Nós também evidenciamos a importância de termos janelas menores para localizar pedestres mais distantes. Para o grupo 2, o detector com janela de dimensão 64 × 128, considerando o FPPI de referência, tem a melhor taxa de faltas de 40% enquanto para o detector com janela de dimensão 48 × 96, tem a melhor taxa de faltas de 15%.

5.4.1.3.2 Combinação de Janelas de Detecção

Nós analisamos o detector de hardware que combina as janelas das duas dimensões. Nesse terceiro experimento nós comparamos os três detectores de hardware. A profundidade da janela dos três detectores foi definida como 7. A filtragem geométrica é considerada. Nós avaliamos os detectores usando a divisão por grupo de distância.

Como pode ser observado na Figura 75, o detector combinado obteve resultados próximos aos resultados dos detectores individuais nos intervalos de distância em que estes detectores são melhores. No grupo 1, o detector combinado obteve uma taxa de faltas para 1 FPPI de 6% contra 8% do detector com janela de dimensão 64 × 128 pixels. No grupo 2, o detector combinado obteve uma taxa de falta para 1 FPPI de 13% contra 15% do detector com janela de dimensão 48 × 96 pixels. Como consequência, o detector combinado consegue ser melhor do que os outros dois detectores no grupo 3 que envolve

Figura 75 – Resultados de qualidade medidos através da relação de FPPI e taxa de faltas para o detector de hardware com janela de detecção de dimensão única de 64 × 128 e 48 × 96 pixels e dimensões combinadas. Os valores σ_{SVM} são gerados no intervalo de $[2.0, 0.0]$ com passos de 0,2 para obter cada resultado.



Fonte: O autor.

todos os *frames*. Para 1 FPPI, o detector combinado obteve uma taxa de 10% contra 16% e 28% obtidos pelos outros dois detectores.

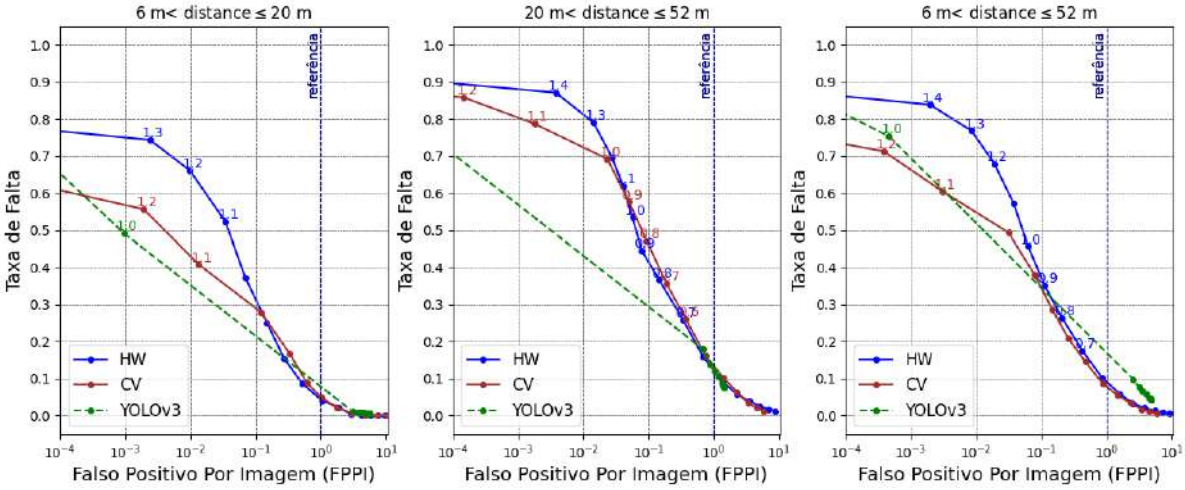
5.4.1.3.3 Comparação de detectores: Hardware vs Software vs YOLOv3

Nós comparamos o detector de hardware proposto com janelas combinadas com o detector baseado em OpenCV com janelas combinadas e com o detector YOLOv3. A profundidade da janela dos detectores combinados foi definida como 7. A filtragem geométrica foi considerada para os dois detectores.

É possível notar na Figura 76 uma redução na qualidade do detector de hardware com relação ao detector baseado em OpenCV, ou seja, o de software, sobretudo para baixos valores de FPPI. Existem algumas razões para esta redução. A primeira é a adoção de ponto fixo no lugar de ponto flutuante que existe na versão do OpenCV. O uso do ponto fixo adiciona erros nos dados calculados. Porém, a adoção de ponto flutuante demanda muito recurso de hardware (MA; NAJJAR; ROY-CHOWDHURY, 2015), comparado com ponto fixo, que inviabilizou o desenvolvimento de detectores escaláveis em termos de profundidade de pirâmide e janelas de detecção de dimensões diferentes.

A segunda razão é a não implementação de técnicas de interpolação trilinear e ponderação gaussiana na abordagem HOG que existem na versão do OpenCV. A interpolação trilinear considera a espacialidade do *bin* dentro de um bloco para definição da ponderação e não somente entre *bins* vizinhos, e a ponderação gaussiana reduz o efeito de borda entre blocos que insere falsos positivos na detecção (DALAL; TRIGGS, 2005). A implementação de tais técnicas em hardware envolve uma estratégia de processamento mais desafiadora, envolvendo um cálculo de histograma em três dimensões, as duas dimensões da imagem

Figura 76 – Resultados de qualidade medidos através da relação de FPPI e taxa de faltas para o detector com combinação de janela de detecção de dimensões de 64 × 128 e 48 × 96 pixels implementado em hardware, com o detector em OpenCV e com detector baseado em YOLO. Os valores σ_{SVM} foram gerados no intervalo de $[2.0, 0.0]$ com passos de 0,1 para obter cada resultado.



Fonte: O autor.

e uma dimensão do histograma, que certamente aumentará o custo de hardware. Além disso, as operações gaussianas para ponderação são necessárias, que acrescentam ainda mais o custo de hardware.

5.4.2 Avaliação de Desempenho do Processamento

Nesta seção nós avaliamos o desempenho de processamento unicamente do detector de hardware e do sistema PCP com o detector na plataforma HARPv2.

5.4.2.1 Detector

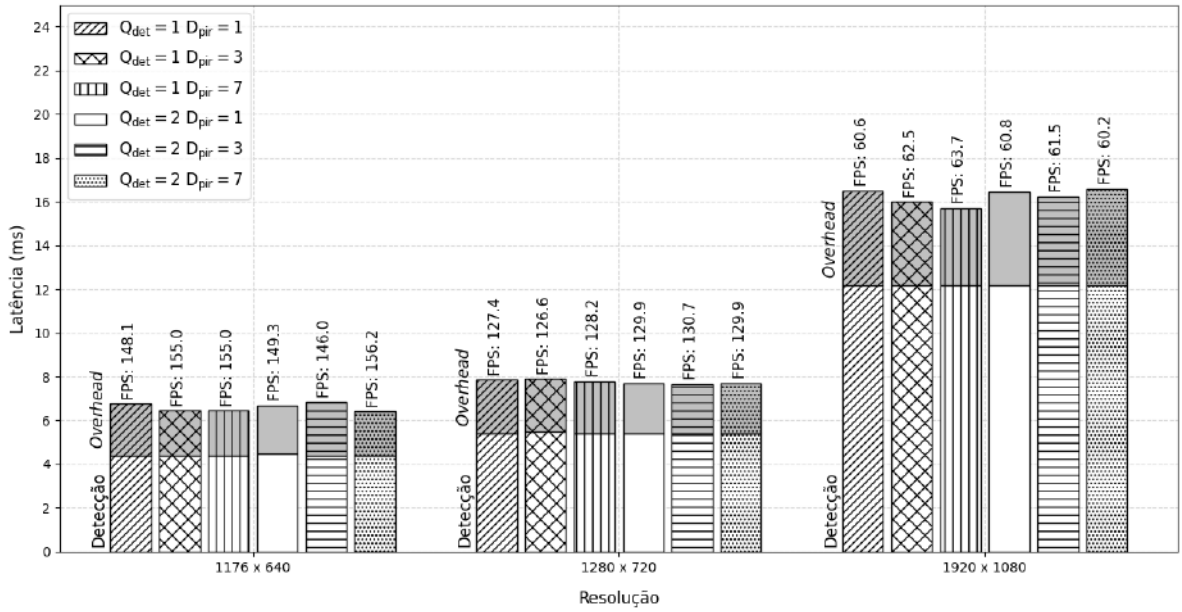
Dois tipos de avaliação do detector foram conduzidos: Real e Teórica. Além disso, nós também comparamos o detector de hardware com a versão do detector em software.

5.4.2.1.1 Real

Nesse tipo de avaliação nós medimos o desempenho do detector integrado na plataforma HARPv2. Os resultados de desempenho são em termos de latência e *frames* por segundo (FPS), estimados através de uma média de 1.000 *frames* processados. No lado do software, são enviados todos os *frames*, um por vez, e coletadas as respostas do detector.

A Figura 77 mostra os resultados do detector em várias resoluções de *frame*, profundidade de pirâmide (D_{pir}) e quantidade de P-SVM (Q_{det}). O tempo gasto preparando o *frame* na memória compartilhada e construindo os *bounding boxes* (*Overhead*) e o tempo de processamento unicamente do detector foram estimados e apresentados em cada barra vertical.

Figura 77 – Desempenho de processamento do detector na plataforma HARPv2. O termo *overhead* significa o tempo gasto para realizar a comunicação. A frequência de operação do detector em todas as configurações foi de 150 MHz.



Fonte: O autor.

Um resultado significativo observado na Figura 77 diz respeito ao tempo para realizar o processamento da detecção no hardware, que permanece aproximadamente constante em todas as configurações testadas para uma determinada resolução de *frame*. Esse comportamento acontece porque todos os módulos de *Resize*, HOGs e SVM de cada camada da pirâmide estão processando o *frame* de entrada em paralelo.

5.4.2.1.2 Teórica

Nesse tipo de avaliação nós estimamos o desempenho máximo que o detector pode alcançar independente de plataforma. Os parâmetros fundamentais para essa avaliação é a frequência de operação do detector e a quantidade de ciclos de *clock* que o detector precisa para processar um *frame*. O processamento do detector acompanha a vazão de entrada serial de pixels de tal maneira que ao término de entrada de todos os pixels o detector já terá finalizado o processamento do *frame* atual. Desta forma, a quantidade de ciclos de *clock* necessária é aproximadamente a quantidade de pixels a ser processados de um determinado *frame*. Então o desempenho máximo é aquele obtido quando a vazão de entrada é de um pixel por ciclo de *clock* ininterrupto. A partir dessas definições, a taxa de processamento (V_{det}) do detector é definida pela seguinte Equação:

$$V_{\text{det}}(f) = \frac{f}{W_{\text{img}}\, H_{\text{img}}}\ (\text{FPS}), \tag{5.9}$$

onde f é a frequência de operação do detector, W_{img} é a largura do *frame* e H_{img} é a altura do *frame*. A partir dessa Equação nós apresentamos na Tabela 4 a estimativa de desempenho teórica do detector para algumas configurações de resolução e frequência de operação.

Tabela 4 – Desempenho Teórico do Detector de Hardware.

Resolução ($W_{\text{img}} \times H_{\text{img}}$)	Frequência (f MHz)	R_{det} (FPS)
1176 × 640	100	132,9
	200	265,7
	300	398,6
	400	531,5
1280 × 720	100	108,5
	200	217,0
	300	325,5
	400	434,0
1920 × 1080	100	48,2
	200	96,4
	300	144,7
	400	192,9

Fonte: O autor.

Na síntese da ferramenta Quartus, o detector atingiu uma frequência máxima de 400 MHz com o FPGA Arria 10. Assim, o detector consegue alcançar, por exemplo, uma taxa máxima de 192 FPS para a resolução de 1920 × 1080 pixels.

É importante destacar que a arquitetura do detector foi pensada para processar o fluxo de pixels serial em qualquer vazão de entrada e sem perda de *frames*. Dessa forma, o desempenho de processamento de todo o sistema não é mais limitado pelo módulo detector, mas pela fonte externa, que gera os pixels seriais, tais como câmeras e barramentos de comunicação.

5.4.2.1.3 Comparação de detectores: versão 1 vs versão 2

As melhorias realizadas para o detector a partir da versão 1 do módulo SVM para a versão 2 não adicionaram qualquer atraso no processamento dos pixels. As modificações para redução dos recursos de memória e elementos lógicos tomam vantagem da característica temporal do fluxo de entrada de pixels que é o mesmo que a primeira versão do módulo SVM. Dessa forma, foi evidenciado em alguns experimentos que não houve qualquer redução de desempenho de processamento entre essas versões.

5.4.2.1.4 Comparação de detectores: hardware vs software

Para termos uma comparação de desempenho mais justa, nós comparamos sob a mesma plataforma HARPv2 o detector de hardware com a versão de software fornecida pela biblioteca OpenCV para execução em plataforma GPP. A tabela 5 mostra a comparação deste dois detectores.

Tabela 5 – Comparação de desempenho em FPS entre o detector de hardware e detector de software na plataforma HARPv2

Resolução ($W_{\text{img}} \times H_{\text{img}}$)	SW (FPS)	HW (FPS)
1176 × 640	10,1	156,2
1280 × 720	8,3	130,7
1920 × 1080	5,2	62,5

Fonte: O autor.

Nós podemos observar nesta tabela um ganho de desempenho em FPS de aproximadamente 15 vezes do detector de hardware em relação ao detector de software.

5.4.2.2 Sistema PCP

Uma avaliação do sistema PCP hardware/software com o detector de hardware e o módulo SGM foi conduzida. Além disso, comparações foram realizadas deste sistema PCP com o sistema PCP tendo o detector e o módulo SGM na versão de software.

5.4.2.2.1 O sistema PCP hardware/software

Para avaliação do sistema PCP nós utilizamos a mesma estratégia realizada para avaliar o detector real, na Seção 5.4.2.1.1, ou seja, em termos de latência e *frames* por segundo (FPS), estimados através de uma média de 1.000 *frames* processados. As diferenças estão no envio dos dados que agora o software envia pares de *frames* estéreo, um por vez, e na coleta, que agora o software obtém o mapa de disparidade e respostas do detector para o processamento das outras etapas do sistema PCP.

A Tabela 6 apresenta os resultados do sistema PCP em várias resoluções de imagem. Como o desempenho é constante em relação a Q_{Det} e D_{pir}, esses parâmetros foram definidos para 2 e 7, respectivamente. Nós coletamos a latência, não somente do sistema completo, mas das partes de software, de hardware e do *overhead* de comunicação. Os tempos destas partes, somados, resultam no tempo do sistema completo.

Um resultado importante é com relação ao tempo de resposta unicamente do processamento do conjunto detector de pedestres e correspondência estéreo que permaneceu

Tabela 6 – Desempenho de processamento do Sistema PCP na plataforma HARPv2. A frequência de operação dos módulos de detecção e correspondência estéreo em todas as configurações foi de 150 MHz. A latência do sistema completo é a soma dos tempos das etapas de software, do *overhead* de comunicação e das etapas de hardware.

Configuração	**Desempenho**			
Resolução	**Sistema Completo[1]**	**Etapas de Software[2]**	***Overhead*[3]**	**Etapas de Hardware[4]**
1176 × 640	74,1 (13,5 ms)	0,1 ms	9,8 ms	3,6 ms
1280 × 720	66,2 (15,1 ms)	0,1 ms	10,1 ms	4,9 ms
1920 × 1080	28,2 (35,4 ms)	0,2 ms	24,6 ms	10,6 ms

[1] Desempenho de processamento de todo o sistema PCP heterogêneo avaliado através de FPS e latência em milissegundos (ms). [2] Tempo de processamento de todas as etapas de software executadas em GPP, conforme descrito na Seção 5.3, medido em milissegundos. [3] *Overhead* de comunicação é o tempo em milissegundos para: organizar os dados do *frame* na memória, no lado do software; ler os dados da memória compartilhada, no lado do hardware; escrever a resposta na memória, no lado do hardware; e montar os resultados na memória do GPP para pós-processamento, no lado do software. [4] Tempo de processamento dos módulos de detecção de pedestres e correspondência estéreo, medido em milissegundos.

Fonte: O autor.

aproximadamente igual ao módulo detector de pedestres. A razão desse resultado é que os dois módulos processam em paralelo e entregam os resultados simultaneamente.

Além disso, o tempo de processamento dos dois módulos é bastante pequeno, em comparação com a latência de todo o sistema PCP. O maior gargalo do desempenho está nas funções de comunicação entre hardware e software. Qualquer melhoria nestas funções ou até mesmo a redução da quantidade de dados transmitidos resultará em ganhos de desempenho de processamento do sistema PCP.

5.4.2.2.2 Comparação de Sistemas PCP: hardware/software vs software

Nós comparamos o sistema PCP com o detector e a correspondência estéreo sendo executados no FPGA e o sistema PCP puramente em software. A versão de software destes dois componentes é baseada na biblioteca OpenCV que implementa as funções de maneira otimizada, usando paralelismo de dados e de *threads*.

Tabela 7 – Comparação de desempenho em FPS entre o sistema PCP hardware/software e o sistema PCP puramente software na plataforma HARPv2

Resolução ($W_{\text{img}} \times H_{\text{img}}$)	SW (FPS)	HW/SW (FPS)
1176 × 640	1,1	74,1
1280 × 720	0,8	66,2
1920 × 1080	0,5	28,2

Fonte: O autor.

Nós observamos na Tabela 7 um ganho do sistema em FPS do PCP hardware/software proposto em torno de 60 vezes comparado com a versão puramente em software.

5.4.3 Avaliação de Previsão de Colisão

Nós utilizamos a base de dados 2 para avaliação da técnica de previsão de colisão. Primeiramente, nós apresentamos as configurações dos sistemas envolvidos nos experimentos e estratégia de análise e em seguida apresentamos os experimentos, resultados e análises.

5.4.3.1 Definição dos Sistemas e Avaliação Geral

As abordagens, parâmetros e implementações dos detectores são similares aos adotados para avaliação do algoritmo de detecção na Seção 4.3.3. Apenas para recapitular, nós definimos P-SVM$_1$ com uma janela de detecção de dimensão 48 × 96 pixels e P-SVM$_2$ com uma janela de detecção de dimensão de 24 × 48 pixels. A partir desses parâmetros, instanciamos dois detectores de hardware com janela de dimensão única e um outro detector com janela de dimensão combinada. O limiar de classificação foi definido experimentalmente como 0,7 levando em consideração o compromisso entre um baixo FPPI e baixa taxa de faltas de localização. As dimensões da célula HOG e do bloco são, respectivamente, 4 × 4 e 8 × 8 pixels. O parâmetro *stride* $S_{\text{P-SVM}}$ de qualquer P-SVM foi definido como sendo de 4 pixels em ambas as direções. A quantidade de *bins* de histograma HOG foi definido como 9. O parâmetro de escala e a profundidade da pirâmide foram definidos como 1,1 e 7.

5.4.3.2 Estratégia de Avaliação

Para avaliação de previsão de colisão nós seguimos a mesma estratégia da Seção 4.3.3.2, na qual usamos a métrica de previsão de colisão mais cedo e quantidade de previsão de colisão. Os intervalos de captura de *frames* para o sistema com o detector combinado de hardware, detector combinado baseado em OpenCV e o detector baseado em YOLO são 2, 4 e 10 *frames*, respectivamente.

Os parâmetros do sistema PCP são os mesmos definidos na Seção 4.3.1.2. Nós adicionamos o parâmetro de ruído do processo $\sigma_{\mathbf{x}}$ do filtro EKF para o detector de hardware como sendo 0,5, que não tinha sido definido antes.

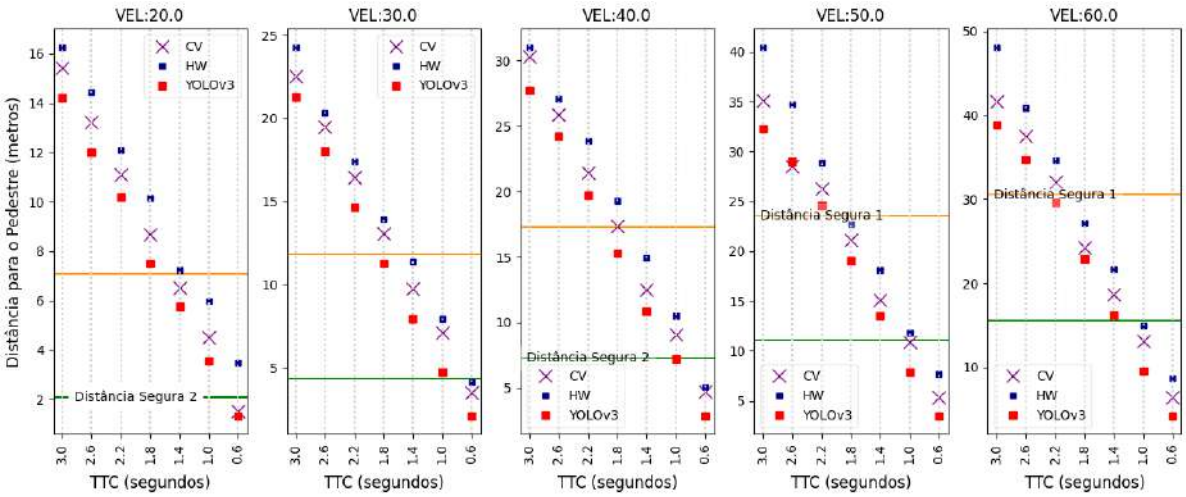
5.4.3.3 Experimentos, análises e comparações

A partir da definição dos detectores, do sistema PCP e da estratégia de avaliação, nós conduzimos os experimentos descritos na seções seguintes.

5.4.3.3.1 Comparação de detectores: hardware vs software vs YOLOv3

Esse experimento foi uma continuação do experimento realizado na Seção 4.3.3 com a adição dos resultados de previsão de colisão mais cedo e quantidade de previsão de colisão do detector de hardware.

Figura 78 – Resultado de qualidade a partir da métrica de previsão de colisão mais cedo. Os detectores avaliados são baseados em HOG e SVM de dimensões combinadas a partir da biblioteca OpenCV (CV) e a partir do hardware (HW) e também o detector baseado em YOLO. O termo Distância Segura 1 e 2 definem as distâncias mínimas para frenagem segura sem colisão, respectivamente, do motorista e do sistema de frenagem automática.



Fonte: O autor.

A Figura 78 apresenta os resultados de previsão de colisão mais cedo. Nós contabilizamos a quantidade de previsões de colisão dentro de uma distância segura, que garante uma tomada de decisão sem colisão. Essas previsões precisam estar acima da marcação de distância segura adotada. É importante destacar que existem cenários que não é possível evitar uma colisão usando sensores de linha de visada, como as câmeras. A distância a partir do momento em que o pedestre se torna visível até o momento da colisão precisa pelo menos ser maior que a distância segura. A Tabela 8 apresenta os cenários são possíveis de se evitar uma colisão. Por exemplo, um veículo com velocidade de 60 km/h possui apenas 10 metros de distância até a colisão, quando o pedestre surge em um TTC de 0,6

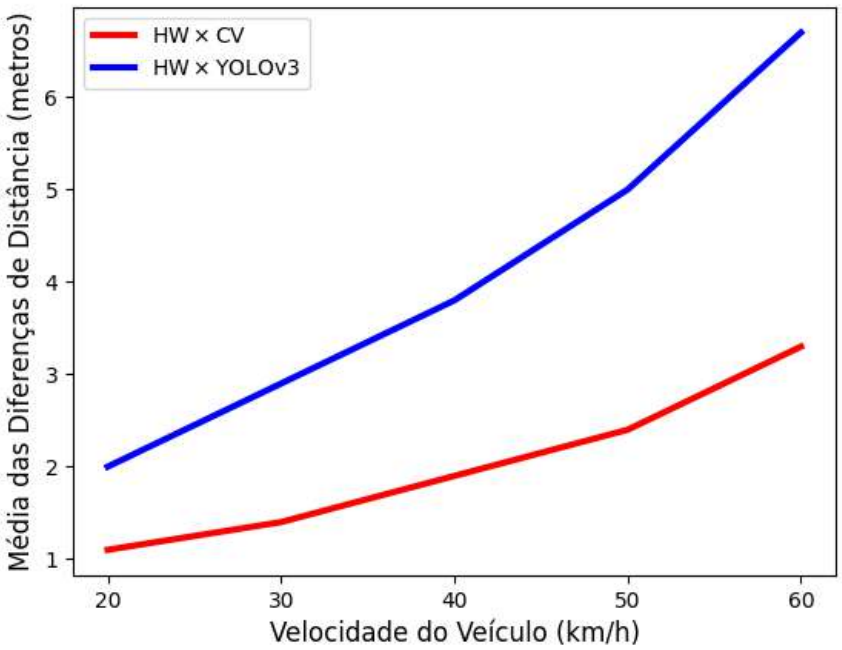
segundos. Essa distância é insuficiente para o motorista raciocinar, tomar a decisão e o veículo começar a frenagem.

Tabela 8 – Distância entre o momento da aparição do pedestre até a colisão com o veículo. As células marcadas em vermelhos e azul são os cenários em que, respectivamente, não são e são possíveis de evitar a colisão usando a distância segura 1.

Velocidade	TTC						
	0,6 s	1,0 s	1,4 s	1,8 s	2,2 s	2,6 s	3,0 s
20 km/h	3,33 m	5,55 m	7,78 m	10,00 m	12,22 m	14,44 m	16,67 m
30 km/h	5,00 m	8,33 m	11,67 m	15,00 m	18,33 m	21,67 m	25,00 m
40 km/h	6,67 m	11,11 m	15,56 m	20,00 m	24,44 m	28,89 m	33,33 m
50 km/h	8,33 m	13,89 m	19,44 m	25,00 m	30,55 m	36,11 m	41,66 m
60 km/h	10,00 m	16,67 m	23,33 m	30,00 m	36,67 m	43,33 m	50,00 m

Fonte: O autor.

Figura 79 – Média das diferenças de distância em metros para tomada de decisão para todos os TTCs em cada velocidade.



Fonte: O autor.

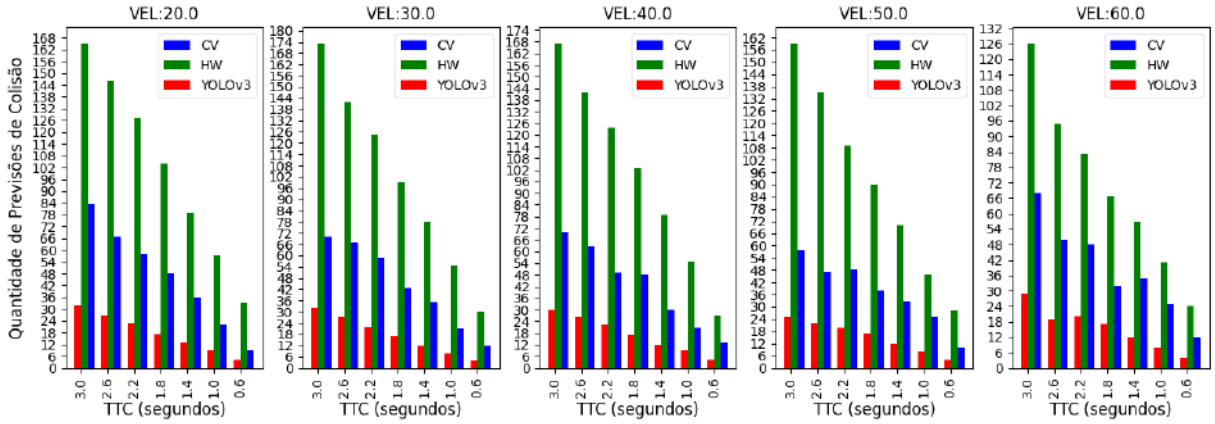
É possível notar que o detector de hardware aumentou a quantidade de previsões de colisão nos dois cenários de distância segura, em comparação com os outros detectores. Por exemplo, com a distância segura 1, ou seja, quando o motorista é quem toma a decisão de frenagem, 19 previsões seguras foram contabilizadas com o detector de hardware contra 17 com o detector de OpenCV e 15 usando YOLOv3. Em outras palavras, o detector de

hardware garantiu 100% de previsão segura nos cenários que de fato são possíveis de evitar colisão, contra 89% com o detector de OpenCV e 79% usando YOLOv3. A principal razão para esse aumento na segurança do pedestre é que uma vez que o tempo de processamento é menor, a estimativa da velocidade correta do pedestre também levará menos tempo.

Embora nem sempre seja possível garantir a previsão de colisão em uma distância acima da distância segura, é possível observar um aumento dessa distância com o uso do detector de hardware em todas as velocidades e configurações de TTC. Para melhor evidenciar esse ganho de distância, nós calculamos a média, pra cada velocidade, das diferenças de distância entre o detector de hardware e os outros detectores (ou seja, HW vs (CV) e HW vs YOLOv3) para todos os valores de TTC, como mostrado na Figura 79.

Na Figura 79 fica claro que o detector de hardware resultou em um aumento na distância para tomada de decisão em comparação com os outros detectores. Por exemplo, comparado com o YOLOv3, o nosso detector pode detectar uma colisão com 6 metros de antecedência com um veículo a 60 km/h de velocidade. Esse aumento em distância é de extrema importância para mitigar o impacto e a gravidade do acidente.

Figura 80 – Resultado de qualidade a partir da métrica de quantidade de previsão de colisão para os detectores baseados em HOG e SVM de dimensões combinadas e baseado em YOLO.



Fonte: O autor.

Além disso, é possível observar que a diferença aumenta quase linearmente com a velocidade do veículo. Isso se deve ao tempo de resposta do sistema PCP com o detector de hardware, que é menor que o tempo de resposta dos outros dois sistemas. Como são necessários vários *frames* para a estimativa de colisão, essa diferença de tempo leva a um aumento no ganho de distância com veículos mais rápidos. Esse crescimento linear traz mais segurança ao pedestre, principalmente em veículos em velocidades mais altas. Os esforços para reduzir o tempo de resposta do sistema PCP aumentarão ainda mais a distância de segurança. Especificamente, a redução dos gargalos de comunicação da integração, conforme mencionado na Seção 5.4.2, significará um tempo de resposta menor do sistema PCP.

Outro importante resultado é com relação a quantidade de previsões de colisão, apresentado na Figura 80. Uma vez que o detector de hardware permitiu o sistema PCP alcançar uma taxa de processamento maior, esse sistema conseguiu gerar uma quantidade de previsão consideravelmente maior do que o sistema com o detector baseado em OpenCV e em YOLO, indicando que o nosso sistema tem uma chance maior de prever uma colisão antes que a colisão aconteça. Além disso, a capacidade de processar mais *frames* aumenta a confiança dos resultados de previsão pois reduz a distância de erro por falta na previsão de colisão.

5.4.4 Avaliação do Uso de Recursos do FPGA

Esta seção apresenta os resultados de síntese em FPGA do detector de hardware isoladamente e de toda a parte de hardware do sistema PCP através da plataforma Arria 10.

5.4.4.1 Detector

A avaliação na Seção 5.4.4.1.1 foi realizada com a versão otimizada do detector, ou seja, com o módulo SVM na versão 2, e comparações são apresentadas na Seção 5.4.4.1.2 em relação à versão 1 do módulo SVM.

5.4.4.1.1 Detector com a versão 2 do SVM

A Tabela 9 apresenta os resultados do detector com o SVM na versão 2, detalhado na Seção 5.2.3.2.3, avaliado com diferentes parâmetros de resolução de imagem, quantidade de P-SVMs e profundidade da pirâmide de imagens. Como pode ser observado nesta tabela, o aumento na resolução da imagem quase não impactou na quantidade de blocos de memória, elementos lógicos e registradores. Essa característica se deve à reutilização dos módulos e unidades para processar todos os pixels da imagem.

Outro aspecto importante é a quantidade de recursos de processamento do detector, que não aumentou na mesma proporção aumentou da quantidade de P-SVMs. Por exemplo, a partir do detector com 7 níveis de pirâmide e $Q_{det} = 1$, para o detector com 7 níveis de pirâmide e $Q_{det} = 2$ o fator de crescimento de elementos lógicos foi 1,4. Este resultado se deve ao reaproveitamento da mesma pirâmide de imagem e HOG para ambos os módulos P-SVMs.

Tabela 9 – Uso de recurso do detector sintetizado no FPGA Arria 10. A frequência máxima obtida na síntese para todas as configurações é de 400 MHz.

Resolução[6]	**Módulo**	**Parâmetros**	**Recurso**			
			KALMs[3]	**KReg.**[4]	**RAM**[5]	**DSPs**[7]
1280 × 720	*Resize*	-	1,3 (0,3%)	2,8	0,38 (0,7%)	9 (0,6%)
	Gradiente	-	1,2 (0,3%)	3,0	0,11 (0,2%)	2 (0,1%)
	Célula	-	0,8 (0,2%)	2,0	0,05 (0,1%)	4 (0,3%)
	Bloco	-	2,7 (0,6%)	8,3	0,26 (0,5%)	1 (0,1%)
	SVM	N_c[1] $= 4$	6,4 (1,5%)	13,4	0,20 (0,4%)	60 (3,9%)
	Completo[2]	$Q_{det} = 1$[8] $D_{pir} = 1$	11,4 (2,6%)	27,7	0,76 (1,4%)	70 (4,6%)
	Completo	$Q_{det} = 1$ $D_{pir} = 3$	35,6 (8,3%)	84,7	3,01 (5,4%)	238 (15,7%)
	Completo	$Q_{det} = 1$ $D_{pir} = 7$	83,0 (19,4%)	206,2	6,20 (11,2%)	574 (37,9%)
	Completo	$Q_{det} = 1$ $D_{pir} = 14$	165,6 (38,6%)	411,7	10,35 (11,2%)	1162 (76,5%)
	Completo	$Q_{det} = 2$ $D_{pir} = 3$	49,9 (11,7%)	122,9	3,90 (7,0%)	386 (25,4%)
	Completo	$Q_{det} = 2$ $D_{pir} = 7$	116,9 (27,4%)	289,0	8,21 (14,8%)	930 (61,2%)
1920 × 1080	*Resize*	-	1,3 (0,3%)	2,8	0,49 (0,9%)	9 (0,6%)
	Gradiente	-	1,2 (0,3%)	3,0	0,16 (0,3%)	2 (0,1%)
	Célula	-	0,8 (0.2%)	2,0	0,05 (0,1%)	4 (0,3%)
	Bloco	-	2,8 (0,6%)	8,4	0,26 (0,5%)	1 (0,1%)
	SVM	N_c[1] $= 4$	6,7 (1,6%)	13,6	0,30 (0,6%)	60 (3,9%)
	Completo	$Q_{det} = 1$ $D_{pir} = 1$	11.5 (2,7%)	28,3	0,93 (1,7%)	70 (4,6%)
	Completo	$Q_{det} = 1$ $D_{pir} = 3$	35,5 (8,3%)	87,8	3,61 (6,5%)	238 (15,7%)
	Completo	$Q_{det} = 1$ $D_{pir} = 7$	84,0 (19,7%)	208,3	8,27 (14,9%)	574 (37,9%)
	Completo	$Q_{det} = 1$ $D_{pir} = 14$	167,2 (39,1%)	416,0	13,61 (24,5%)	1162 (76,5%)
	Completo	$Q_{det} = 2$ $D_{pir} = 3$	50,2 (11,8%)	123,6	4,75 (8,6%)	386 (25,4%)
	Completo	$Q_{det} = 2$ $D_{pir} = 7$	118,7 (27,8%)	292,5	10,66 (19,2%)	930 (61,2%)

[1] Quantidade de colunas de processamento paralelo de SVMs. [2] Para todas as configurações de detector, o parâmetro N_c é igual a 4. [3] Quantidade de módulos de lógica adaptativa (em inglês, *adaptive logic modules* - ALMs), apresentados em unidades de 10^3 e porcentagem de ocupação do FPGA. [4] Quantidade de registradores em unidades de 10^3. [5] Quantidade de armazenamento na RAM interna da FPGA, medida em megabits e porcentagem de ocupação do FPGA. [6] Resolução. [7] Quantidade de blocos de DSPs e porcentagem de ocupação do FPGA. [8] Para $Q_{det} = 1$ a janela possui dimensão de 64×128 pixels e para $Q_{det} = 2$ as janelas possuem dimensões de 64×128 e 48×96 pixels.

Fonte: O autor.

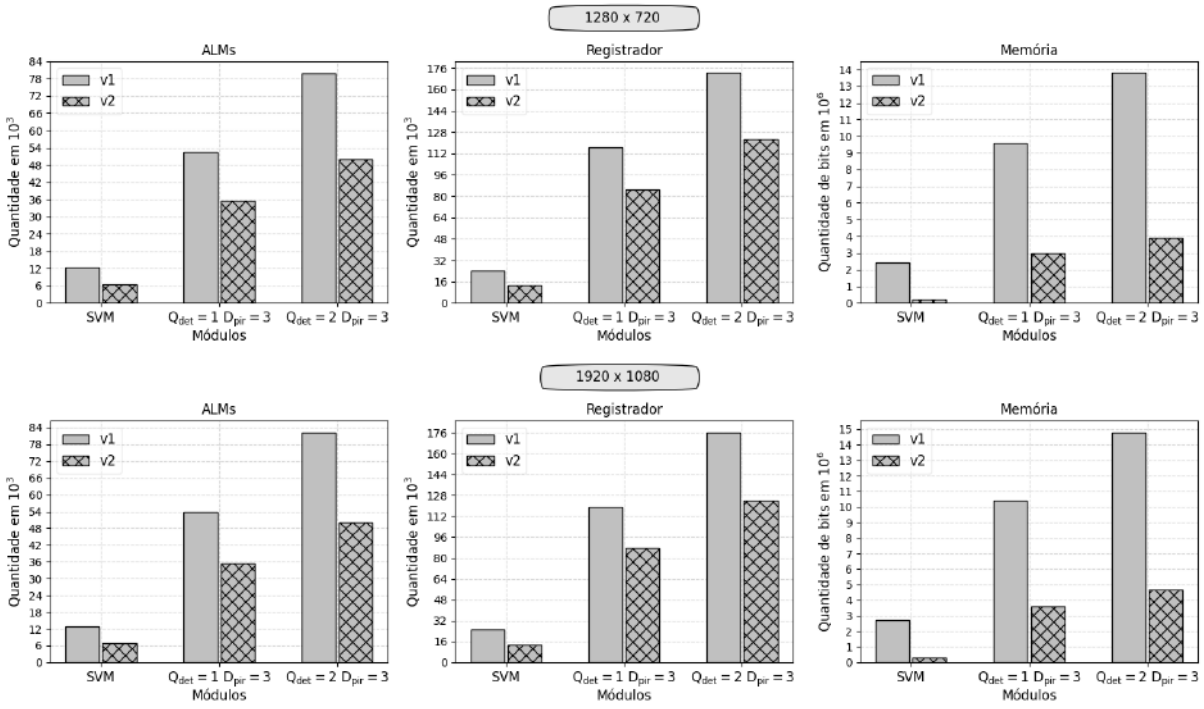
Os módulos *Resize*, Gradiente, Célula e Bloco, juntos, exigem uma quantidade de recursos significativa. Nós não focamos em otimizações de recursos nestes módulos. Isso significa que o aumento da profundidade da pirâmide gera um aumento na exigência de recursos no detector completo.

5.4.4.1.2 Comparação de detectores: versão 1 vs versão 2

A Figura 81 apresenta a comparação dos detectores com as duas versões do módulo SVM, a primeira versão sem otimização de hardware, apresentada na Seção 5.2.3.1 e a

versão 2, com a estratégia de reuso de memória.

Figura 81 – Comparação do módulo SVM na versão 1 e a versão 2 isoladamente, seguida de duas comparações do detector completo com o módulo SVM nas duas versões.



Fonte: O autor.

Nós podemos observar um ganho significativo com estratégia de reuso de memória, em termos de consumo de unidades lógicas, registradores e memória. A redução de consumo de memória é mais da metade com a versão de 2 do módulo SVM. A quantidade de elementos lógicos e registradores diminuiu porque as memórias do tipo FIFO possuem lógicas de processamento de dados.

5.4.4.2 Partição de Hardware do Sistema PCP

A Tabela 10 apresenta a ocupação de recurso de toda a parte de hardware do sistema PCP que inclui o modulo detector de pedestre, o módulo de correspondência estéreo SGM, e o módulo de comunicação HW/SW.

O intuito de apresentar estes resultados é destacar a parcela de uso do hardware de cada módulo que participa do sistema PCP. Esta tabela ajuda a orientar novos esforços para redução de recursos em módulos que estão com parcelas maiores de ocupação. Além disso, nós reforçamos a obtenção, nesta Tese, de um módulo maior capaz de processar mapas de disparidades e detecção de pedestres ao mesmo tempo.

Tabela 10 – Uso de recursos de todo o módulo de hardware do sistema PCP, que envolve o detector de pedestres, correspondência estéreo e a comunicação HW/SW, sintetizados no FPGA Arria 10.

Resolução	**Módulo**	**Parâmetros**	**Recurso**		
			KALMs[2]	**KReg.[3]**	**RAM[4]**
1280 × 720	Detector	$Q_{det} = 2$ $D_{pir} = 7$	116,9 (27,4%)	289,0	8,2 (14,8%)
	SGM	N_d[1] $= 128$	73,7 (17,4%)	96,2	5,5 (10,0%)
	Comunicação[5]	-	41,8 (9,8%)	55,1	0,9 (1,6%)
	HW Completo	$Q_{det} = 2$ $D_{pir} = 7$ $N_d = 128$	199,4 (46,7%)	450,2	17,3 (31,1%)
1920 × 1080	Detector	$Q_{det} = 2$ $D_{pir} = 7$	118,7 (27,8%)	292,5	10,66 (19,2%)
	SGM	$N_d = 128$	79,1 (18,7%)	106,9	8,8 (16,0%)
	Comunicação	-	41,8 (9,8%)	55,1	0,9 (1,6%)
	HW Completo	$Q_{det} = 2$ $D_{pir} = 7$ $N_d = 128$	215,2 (50,4%)	462,1	22,6 (40,7%)

[1] Intervalo de Disparidade. [2]Quantidade de módulos de lógica adaptativa (ALMs), apresentados em unidades de 10^3 e porcentagem de ocupação do FPGA. [3] Quantidade de registradores em unidades de 10^3. [4] Quantidade de armazenamento em memória RAM do FPGA, medida em megabits e porcentagem de ocupação do FPGA. [5] Módulo de comunicação hardware/software.

Fonte: O autor.

5.4.5 Avaliação de Dissipação de Potência em FPGA

Nós também realizamos uma avaliação de dissipação de potência dos módulos de detecção e correspondência estéreo. A medição foi realizada com a ferramenta Altera PowerPlay. Para obter medidas precisas da dissipação de potência partir dessa ferramenta nós precisamos ter dados suficientes de atividades de chaveamento em nível de portas lógicas gerados pelo módulo detector. Dessa forma, utilizando a ferramenta ModelSim, nós geramos esses dados de chaveamento em simulação, enviando um fluxo de pixels de *frame* para o módulo detector. Ao final da simulação, nós obtivemos um arquivo de chaveamento, com extensão .vcd (*Value Change Dump*), que utilizamos como entrada para a ferramenta PowerPlay, que por sua vez, estimou a potência estática, a potência dinâmica e a potência total dissipada. Nós tivemos o cuidado de enviar uma quantidade de pixels para o módulo suficiente para a ferramenta PowerPlay afirmar que as estimativas de potência geradas são confiáveis. Os resultados são apresentados na Tabela 11.

Como pode ser observado, o aumento na profundidade da pirâmide e da quantidade de janelas de dimensões diferentes afetam diretamente a dissipação de potência dinâmica e de entrada e saída. Por outro lado, a parcela dessas dissipações é pequena em relação a potência estática que é pouco afetada por essas configurações. Isso significa que o aumento do tamanho do módulo não aumentará na mesma proporção a dissipação de potência total. Um resultado atrativo é com relação ao sistema de hardware completo que dissipou

Tabela 11 – Dissipação de potência do módulo detector e de todo o módulo de hardware do sistema PCP, que envolve o detector, SGM e a comunicação HW/SW. A frequência utilizada é de 150 MHz.

Resolução	**Módulo**	**Parâmetros**	**Potência (mW)[3]**			
			Estática	**Dinâmica**	**E/S[4]**	**Total**
1280 × 720	Detector	$Q_{det} = 1^{1,2}$ $D_{pir} = 1$	1714,99	0,11	1,37	1716,46
		$Q_{det} = 1$ $D_{pir} = 3$	1764,02	0,34	3,69	1768,05
		$Q_{det} = 1$ $D_{pir} = 7$	1854,06	0,73	9,13	1863,92
		$Q_{det} = 1$ $D_{pir} = 14$	2039,13	1,51	18,24	2058,88
		$Q_{det} = 2$ $D_{pir} = 3$	1820,15	0,68	7,82	1828,65
		$Q_{det} = 2$ $D_{pir} = 7$	2004,32	1,43	17,92	2023,67
	Completo	$Q_{det} = 2$ $D_{pir} = 7$ $N_d = 128$	2389,25	2,78	34,74	2426,47
1920 × 1080	Detector	$Q_{det} = 1$ $D_{pir} = 1$	1715,74	0,11	1,37	1717,22
		$Q_{det} = 1$ $D_{pir} = 3$	1764,25	0,34	3,95	1768,54
		$Q_{det} = 1$ $D_{pir} = 7$	1859,44	0,83	9,21	1869,48
		$Q_{det} = 1$ $D_{pir} = 14$	2041,05	1,62	18,40	2061,07
		$Q_{det} = 2$ $D_{pir} = 3$	1835,49	0,71	7,91	1844,11
		$Q_{det} = 2$ $D_{pir} = 7$	2007,09	1,58	18,14	2026,81
	Completo	$Q_{det} = 2$ $D_{pir} = 7$ $N_d = 128$	2429,12	3,02	35,22	2467,36

[1] Para todos os detectores, o parâmetro de quantidade de colunas N_c nas unidades SVM é igual a 4. [2] Para $Q_{det} = 1$ a janela de detecção possui dimensão de 64 × 128 pixels e para $Q_{det} = 2$ as janelas possuem dimensões de 64 × 128 e 48 × 96 pixels. [3] Potência dissipada medida em miliwatts (mW). [4] Potência de *Standby* é a potência que um dispositivo dissipa quando não está em uso, mas conectado a uma fonte de energia e pronto para ser usado.

Fonte: O autor.

apenas 2,5 Watts para imagens com resolução de 1920 × 1080 pixels.

5.4.6 Comparação com Detectores de Hardware Existentes

As Tabelas 12 e 13 apresentam os resultados de alguns detectores de pedestres existentes baseados em hardware quanto à, respectivamente, desempenho de processamento e ocupação de recursos, para algumas configurações de resolução de *frame* e profundidade da pirâmide.

Além da métrica de FPS, nós também avaliamos a velocidade de processamento através da métrica de janelas de detecção por segundo (DPS), que significa o produto da quantidade de janelas de detecção verificadas em um *frame* e o FPS. O resultado em termos de DPS do nosso detector foi medido na plataforma de prototipagem, HARPv2, considerando o *overhead* de comunicação da plataforma. Nosso sistema possui um DPS maior do que os outros trabalhos, devido principalmente pela quantidade de SVMs ins-

Tabela 12 – Dados de desempenho de processamento obtidos a partir de detectores de pedestres existentes.

Trabalho	**Plat.**[2]	**Resol.**[3]	**Pirâmide**		**Quantidade de *Clocks***	**MHz**	**Desempenho**	
			Profundidade	**Escala**			**FPS**	**MDPS**[4]
HELALI (2020) (HOG + SVM)	Zynq 7020 (28 nm)	1920 × 1080	1 nível	n_a[1]	1	150	60	1,9
LUO (2018) (HOG + SVM)	Cyclone IV (60 nm)	800 × 600	1 nível	n_a	1	150	162	1,2
DURRE (2018) (Pirâmide + HOG + SVM)	Cyclone IV (60 nm)	1920 × 1080	9 níveis	1,25	2	140	33	9,5
MAGGIANI (2018) (HOG + SVM)	Cyclone V (28 nm)	640 × 480	1 nível	n_a	1	84	273	1,3
NGUYEN (2019) (YOLO)	Virtex 7 (28 nm)	416 × 416	n_a	n_a	1	200	60,1	n_a
PROPOSTO (Pirâmide + + HOG + SVM)	Arria 10 (20 nm)	1920 × 1080	7 níveis	1,1	1	150	62,2	12,0
PROPOSTO (Pirâmide + + HOG + SVM)[5]	Arria 10 (20 nm)	1920 × 1080	14 níveis	1,1	1	150	63,7	24,7

[1] Os termos n_i, n_u e n_a significam, respectivamente, não informado, não utilizado e não aplicado. [2] Plataforma FPGA utilizada e sua tecnologia de circuito integrado na forma de dimensão em nanômetro (nm) do menor componente constituinte. [3] Resolução de *frame* em pixels [4] Milhão de detecção por segundo. [5] Janelas de detecção com tamanhos diferentes.

Fonte: O autor.

tanciados.

Do ponto de vista de ocupação de recursos, o trabalho de (DÜRRE; PARADZIK; BLUME, 2018) apresenta resultados interessantes, uma vez que houve muitas simplificações de hardware nas etapas anteriores ao módulo SVM, tais como *resize*, gradiente, célula e bloco. Nós não exploramos simplificações nestas etapas. Porém, os autores não informaram a precisão do ponto fixo adotada que impacta diretamente na taxa de ocupação de recursos dos módulos. Além disso, os autores utilizam uma estratégia de frequência dupla para redução de recursos no módulo SVM que prejudica a escalabilidade em termos de frequência de operação. Nós não aplicamos frequência dupla no nosso detector. Ao invés disso, utilizamos a estratégia de unidades SVM de processamento em paralelo que certamente

Tabela 13 – Dados de ocupação de recursos em FPGA obtidos a partir de detectores de pedestres existentes.

Trabalho	**Plat.**[2]	**Resol.**[3]	**Pirâmide**		**Quantidade de *Clocks***	**Recursos**			
			Profundidade	**Escala**		**KEL**[6]	**KReg**[7]	**RAM**[8]	**ERAM**[9]
HELALI (2020) (HOG + SVM)	Zynq 7020 (28 nm)	1920 × 1080	1 nível	n_a[1]	1	11,5	12,2	n_u[1]	n_u
LUO (2018) (HOG + SVM)	Cyclone IV (60 nm)	800 × 600	1 nível	n_a	1	16,0	7,2	0,3	n_u
DURRE (2018) (Pirâmide + HOG + SVM)	Cyclone IV (60 nm)	1920 × 1080	9 níveis	1,25	2	47,2	25,1	4,2	n_u
MAGGIANI (2018) (HOG + SVM)	Cyclone V (28 nm)	640 × 480	1 nível	n_a	1	8,4	n_i[1]	0,2	n_u
NGUYEN (2019) (YOLO)	Virtex 7 (28 nm)	416 × 416	n_a	n_a	1	155,0	n_i	21,9	n_u
PROPOSTO (Pirâmide + + HOG + SVM)	Arria 10 (20 nm)	1920 × 1080	7 níveis	1,1	1	84,0	208,3	8,2	n_u
PROPOSTO (Pirâmide + + HOG + SVM)[5]	Arria 10 (20 nm)	1920 × 1080	14 níveis	1,1	1	167,2	416,2	13,6	n_u

[1] Os termos n_i, n_u e n_a significam, respectivamente, não informado, não utilizado e não aplicado. [2] Plataforma FPGA utilizada e sua tecnologia de circuito integrado na forma de dimensão em nanômetro (nm) do menor componente constituinte. [3] Resolução de *frame* em pixels [4] Milhão de detecção por segundo. [5] Janelas de detecção com tamanhos diferentes. [6] O termo KEL significa quantidade de elementos lógico medidos em unidades de 10^3 que podem ser ALMs ou LUTs. [7] Quantidade de registradores em unidades de 10^3. [8] Quantidade de armazenamento RAM medida em megabits. [9] O termo ERAM significa memória externa.

Fonte: O autor.

exige mais recursos de FPS, mas permite alcançar frequências de processamento cada vez mais altas.

Nós também realizamos comparações de dissipação de potência. Como medida de eficiência de potência, nós adotamos a de Joules por *frame* (JPF) (MA; NAJJAR; ROY-CHOWDHURY, 2015) que é obtida dividindo a potência dissipada pela taxa de processamento de *frame* (FPS). Essa medida significa o consumo de energia para processar um *frame*. A medida de Joules por *frame* não diferencia detectores que avaliaram diferentes quantidade de janelas de detecção. Dessa forma, nós adotamos também a medida de Joules por detecção (JPD), que é obtida dividindo a potência dissipada pela taxa de processamento da janela de detecção (DPS). A Tabela 14 apresenta comparações de dissipação de

Tabela 14 – Dados de dissipação de potência e consumo de energia obtidos a partir de detectores de pedestres existentes.

Trabalho	**Plat.**[2]	**Resol.**[3]	**Pirâmide**		**Desempenho**		**Dissipação/Energia**		
			Profundidade	**Escala**	**FPS**	**MDPS**	**Watts**	**JPF**[6]	**μJPD**[7]
HELALI (2020) (HOG + SVM)	Zynq 7020 (28 nm)	1920 × 1080	1 nível	n_a[1]	60	1,9	1,5	0,025	0,789
LUO (2018) (HOG + SVM)	Cyclone IV (60 nm)	800 × 600	1 nível	n_a	162	1,2	n_i	n_i	n_i
DURRE (2018) (Pirâmide + HOG + SVM)	Cyclone IV (60 nm)	1920 × 1080	9 níveis	1,25	33	9,5	n_i	n_i	n_i
MAGGIANI (2018) (HOG + SVM)	Cyclone V (28 nm)	640 × 480	1 nível	n_a	273	1,3	n_i	n_i	n_i
NGUYEN (2019) (YOLO)	Virtex 7 (28 nm)	416 × 416	n_a	n_a	60,1	n_a	11,1	0,184	n_i
PROPOSTO (Pirâmide + + HOG + SVM)	Arria 10 (20 nm)	1920 × 1080	7 níveis	1,1	62,2	12,0	1,9	0,031	0,158
PROPOSTO (Pirâmide + + HOG + SVM)[5]	Arria 10 (20 nm)	1920 × 1080	14 níveis	1,1	63,7	24,7	2,0	0,032	0,081

[1] Os termos n_i, n_u e n_a significam, respectivamente, não informado, não utilizado e não aplicado. [2] Plataforma FPGA utilizada e sua tecnologia de circuito integrado na forma de dimensão em nanômetro (nm) do menor componente constituinte. [3] Resolução de *frame* em pixels [4] Milhão de detecção por segundo. [5] Janelas de detecção com tamanhos diferentes. [6] Joules por *Frame*. [7] Micro Joules por detecção.

Fonte: O autor.

potência e eficiência dos detectores usando estas métricas. Podemos observar que a nossa solução consegue ter uma eficiência de consumo de energia melhor do que os detectores mencionados ao processar cada janela de detecção.

Não é possível realizar uma comparação em termos de precisão devido à falta de compatibilidade da base de dados. A base de dados INRIA (DALAL; TRIGGS, 2005), usada pela maioria desses trabalhos, é destinada a avaliação puramente da etapa de detecção e, dessa forma, não fornece dados de *frames* estéreo, nem informações sobre o movimentação e a geometria do veículo que são necessários para a avaliação da localização no sistema PCP que contem o detector.

5.4.7 Comparação com Sistemas PCP Existentes

Nós também comparamos os sistemas PCP existentes na literatura, descritos na Seção 2.2.

Tabela 15 – Dados de desempenho de processamento obtidos a partir de sistemas PCP existentes.

Trabalho	**Plataforma**[2]	**Resolução**[3]	**Método de estimativa de profundidade**	**FPS**
Keller (2011)	GPP (Quad Core de 4 GHz)	640 × 480	Correspondência estéreo denso semi global (SGM)	8,0
Lee (2017)	n_i	640 × 480	Radar	11,0
Llorca (2011)	n_i	320 × 240	Correspondência estéreo esparsa local	20,0
Park (2017)	n_i[1]	1280 × 960	Radar	10,0
Wu (2016)	GPP (Xeon de 3,5 GHz)	1242 × 375	Correspondência estéreo denso semi global (SGM)	1,2
PROPOSTO	GPP (Xeon de 3,5 GHz) FPGA (Arria 10)	1280 × 720	Correspondência estéreo denso semi global (SGM)	66,2

[1] O termo n_i significa não informado. [2] Plataforma de processamento. [3] Resolução de *frame* em pixels.

Fonte: O autor.

Como pode ser visto na Tabela 15, o desempenho de processamento e resolução de *frame* do nosso sistema PCP são ambos superiores a todos os outros sistemas. Nós destacamos o sistema de Wu, Zhou e Srikanthan (2016) que é totalmente baseado em câmera e com técnicas de baixo custo computacional detecção de obstáculos. Este sistema utiliza correspondência estéreo SGM para a estimativa de distância. A plataforma GPP é a mesma que nós adotamos neste trabalho. Como pode ser visto na tabela, o desempenho do nosso sistema com os módulos de detecção de pedestres e correspondência estéreo em FPGA, é bastante superior.

5.5 RESUMO

Este capítulo apresentou a implementação em FPGA do módulo de detecção de pedestres baseado nas técnicas HOG e SVM, suportando pirâmide de imagens e janelas de detecção de diferentes dimensões, proposta nessa Tese. Foi demonstrado que um detector com janelas de detecção de dois tamanhos diferentes obteve uma taxa de faltas de 6% menor do que um detector com janelas de detecção de dimensões únicas. Estratégias específicas para cada módulo foram propostas para garantir uma vazão de processamento de um pixel por ciclo de *clock*.

Com relação ao módulo mais crítico, o SVM, uma nova abordagem foi proposta para processar uma grande quantidade de janelas de detecção sobrepostas. Esta abordagem envolve uma nova arquitetura formada de várias unidades SVMs, dispostas em matriz, apresentada na Seção 5.2.3.1, que processam em paralelo um conjunto diferente de janelas de detecção. Além disso, uma estratégia para redução do uso de recursos em FPGA baseada em reuso de memória foi proposta, apresentada na Seção 5.2.3.2. Esta estratégia gerou uma redução significativa em recursos de memória e de elementos de processamento, mantendo o mesmo desempenho de processamento da primeira versão do módulo SVM.

O módulo detector proposto proporcionou um ganho em desempenho de processamento em comparação com outros trabalhos existentes. Além disso, o detector apresentou um desempenho constante independente da profundidade da pirâmide e da quantidade de janelas de detecção de diferentes tamanhos. O ganho de desempenho de processamento do detector deu ao sistema PCP uma capacidade de evitar mais colisões em relação aos sistemas PCP com o detector YOLOv3 e o detector baseado em HOG e SVM, fornecida pela biblioteca OpenCV. Além disso, foi observado uma melhora na distância de tomada de decisão de 3,3 metros com este sistema PCP integrado.

6 ARQUITETURA HW/SW PARA LOCALIZAÇÃO

Neste capítulo nós apresentamos uma abordagem baseada na integração hardware/software para aceleração do processamento da etapa de localização. É importante lembrar que a etapa de localização é constituída da geração do mapa de disparidade e estimativa da distância lateral e longitudinal do pedestre. Essa etapa recebe como entrada pares de *frames* esquerda e direita retificados e os *bounding boxes* resultantes da etapa de detecção de pedestres.

A geração do mapa de disparidade é processada através da técnica de correspondência estéreo SGM que realiza uma otimização global através da propagação de custos de dissimilaridades em vários caminhos unidimensionais independentes. Quanto mais caminhos de propagação existirem e quanto mais simétricos esses caminhos forem, maior será a precisão do mapa de disparidade (HIRSCHMULLER; SCHARSTEIN, 2009). Contudo, a complexidade da técnica SGM aumenta com o aumento da quantidade de caminhos de propagação. Além disso, cada caminho de propagação impõe dependências de dados ao longo do processamento do *frame* que dificulta a obtenção de implementações otimizadas em desempenho dessa técnica.

Nós propomos nesta Tese uma solução, descrita em Cambuim et al. (2019), baseada em FPGA, para o processamento de alto desempenho da técnica SGM. A solução suporta qualquer vazão de entrada serial de pixels. Alguns caminhos de propagação a partir da técnica SGM não são considerados, principalmente aqueles que são contrários ao fluxo de entrada, pois impõe grandes desafios para implementações em hardware que garantam o fluxo de processamento contínuo. Essa redução da quantidade de caminhos certamente reduziu a precisão do mapa de disparidade. Contudo, como será demonstrado nos resultados, essa redução da precisão não prejudica a estimativa de localização do pedestre devido à robustez da abordagem da etapa de estimativa de localização.

Nós avaliamos em Cambuim et al. (2019) o desempenho da solução de hardware (FPGA) para o processamento do SGM em plataformas heterogêneas formada por um processador de propósito geral GPP e FPGA, e constatamos que um dos maiores gargalos que dificulta alcançar maiores taxas de processamento é a troca de dados entre estes núcleos. Para reduzir o gargalo de comunicação, nós propomos uma abordagem que suporta comunicação em fluxo contínuo. Nós utilizamos a plataforma Intel HARP na versão 2 (HARPv2) (GUPTA, 2016) que consiste em um processador Intel Xeon E5-2699v4 e uma placa de aceleração programável com a FPGA Intel Arria 10 GX. Nós apresentamos uma estratégia de gerenciamento de *buffer* duplo para aumentar o desempenho do processamento da localização que pode ser aproveitado em qualquer outra plataforma HW/SW. Essa estratégia permitiu aumentar em pelo menos 1,4x o desempenho do processamento de localização comparado com uma estratégia sem *buffer* duplo. Além disso, nós também

propomos uma estratégia de ordenação de dados necessária para o módulo SGM e uma estratégia de gerenciamento de vários intervalos de endereços de memória para suportar *frames* em altas resoluções.

A plataforma e a estratégia de comunicação são utilizadas para validação da integração do detector de pedestre e do módulo SGM, apresentados no Capítulo 4. Neste capítulo nós focamos na estratégia de comunicação e na integração do módulo SGM. Além disso, nós também realizamos um estudo da precisão da técnica de estimativa da distância proposta e apresentada no Capítulo 4 que é importante para conhecer a sua parcela de contribuição na precisão do sistema PCP. Nas seções seguintes nós descrevemos a estratégia de comunicação, apresentamos os resultados de precisão de localização e desempenho de processamento.

6.1 ESTRATÉGIA DE COMUNICAÇÃO COM HARP

O OPAE (*Open Programmable Acceleration Engine*) é o *framework* da plataforma HARPv2 que permite a comunicação entre o software executado no GPP e o módulo de aceleração ou também chamado de Unidade Funcional de Acelerador (em inglês, *Accelerator Functional Unit* - AFU) executado no FPGA através da alocação de regiões de memória compartilhadas no GPP, às quais o FPGA tem acesso para enviar ou receber dados.

No lado do software, qualquer escrita de novas informações é feita como uma simples atribuição a uma variável ou *array*. Em aplicações de visão computacional, cada novo *frame* deve ser escrito neste *array* compartilhado para que o módulo AFU possa acessá-los. No lado do hardware, o protocolo chamado *Core Cache Interface* (P-CCI) permite que o AFU leia e grave dados em endereços na RAM do GPP.

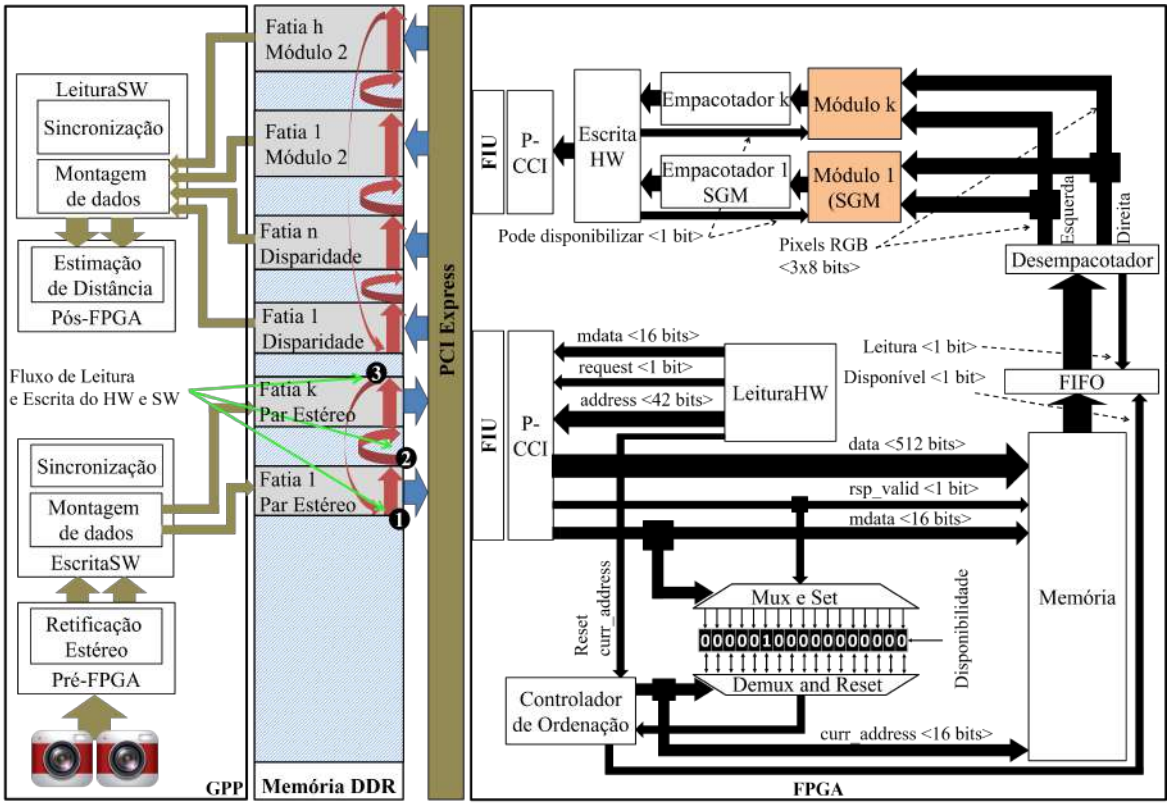
As solicitações de escrita são executadas pelo módulo Gerenciador de Endereços de Escrita de Hardware (EscritaHW), conforme mostrado na Figura 82, que fornece os dados junto com o endereço que será escrito sem a necessidade de aguardar qualquer resposta do P-CCI. Essa característica é atrativa para módulos aceleradores que geram grandes fluxo de dados de forma contínua. As solicitações de leitura são realizadas pelo módulo Gerenciador de Endereços de Leitura de Hardware (LeituraHW). O módulo LeituraHW solicita a leitura de vários endereços, enviando cada um separadamente, e então aguarda as respostas a partir do P-CCI. Contudo, a ordem em que os dados chegam a partir do P-CCI não preserva a mesma ordem que em que os dados foram solicitados. O ordenamento dos dados é uma característica necessária para a maioria das abordagens de visão computacional principalmente aquelas que possuem uma forte dependência de dados, como é o caso do módulo SGM e o do módulo de detecção de pedestres apresentados nos capítulos 4 e 5.

Para resolver a falta de ordem da resposta, nós utilizamos um campo chamado mdata a partir do P-CCI. Esse campo existe tanto na solicitação de leitura como na resposta. No momento da solicitação de leitura, o valor que está em mdata é mantido na resposta dessa

solicitação também em um campo de mdata. Dessa forma, nós utilizamos esse campo para definir índices ordenados como um *tag* para cada endereço lido. Assim, para cada dado de resposta, o P-CCI também informa a *tag* que foi fornecida anteriormente durante a requisição de leitura. Esta *tag* é usada para armazenar os dados na ordem correta em uma memória de dados indexável do FPGA.

O módulo Controlador de Ordenação gerencia a disponibilidade desses dados através de registradores de estado para cada endereço de memória requisitado. Na medida em que cada dado de resposta vêm do P-CCI, o registrador de disponibilidade, cuja posição, localizado pela *tag* do dado, é atualizado com o nível lógico 1 para indicar que esse dado agora está disponível na memória RAM. O Controlador de Ordenação mantém um registrador de índice que aponta para o próximo registrador indisponível na sequência ordenada. Esse registrador também indexa a requisição de leitura da memória. Quando o registrador de disponibilidade informa que há dados disponíveis na posição apontada pela registrado de índice, o Controlador de Ordenação zera este registrador, incrementa o registrador de índice e envia os dados para a AFU.

Figura 82 – Estratégia para leitura e escrita de dados envolvendo estratégia de ordenação de dados e gerenciamento de múltiplos endereços de memória. Essa arquitetura suporta grandes quantidades de dados, como imagens em altas resoluções, por causa dos múltiplos endereços de memória, e lida com aplicações que possuem forte dependência de dados, que exigem o processamento de dados ordenados.



Fonte: O autor.

O campo mdata possui uma largura fixa de 16 bits definida pelo padrão P-CCI. Com

esse tamanho, é possível mapear até 2^{16} endereços na memória compartilhada. Para aplicações de visão computacional, essa quantidade pode ser insuficiente para processar *frames* em grandes resoluções. No caso do módulo SGM, esta situação é ainda pior, pois esta técnica necessita de pares de *frames* em que cada *frame* é representado em um espaço de cores RGB. Para aumentar a capacidade de mapeamento de dados, nós propomos um esquema de paginação para gerenciar vários espaços de endereçamento contíguos, conforme mostrado na Figura 82. Do lado do software, o módulo Gerenciador de Endereços de Escrita de Software (EscritaSW) aloca vários espaços com o mesmo tamanho pré-definido que possibilita o acesso pelo lado do hardware através do P-CCI a todos os endereços.

A sequência de leituras do módulo LeituraHW passa por três etapas essenciais rotuladas pelos círculos 1, 2 e 3 na Figura 82. A solicitação de leitura dos endereços sobre uma fatia de dados (etapa 1) ocorre incrementando um contador que armazena inicialmente o endereço base da fatia atual. Ao atingir a última posição da fatia, o módulo de LeituraHW atualiza com o endereço base da próxima fatia, redefine o índice de endereço disponível no Controlador de Ordenação e inicia a leitura sequencial de novos endereços (etapa 2). Quando o módulo LeituraHW atinge o último endereço da última fatia, ele informa o software e atualiza o contador para o endereço base da fatia 1 (passo 3). A lógica de escrita da resposta do módulo SGM para várias regiões de memória é semelhante ao módulo LeituraHW e é feita pelo módulo EscritaHW.

Para enviar ou receber vários *frames*, tanto o software quanto o hardware implementam um mecanismo de sincronização baseado na troca de parâmetros e variáveis de controle. Cada vez que o software termina de escrever todos os dados na memória, ele modifica uma variável de controle. Quando o módulo LeituraHW reconhece os dados disponíveis, ele solicita a leitura desses dados em modo rajada. Da mesma forma, quando o módulo SGM termina de processar e gravar todos os resultados na memória, o módulo EscritaHW modifica uma variável de controle para que a aplicação executando no processador GPP saiba que todos os resultados estão prontos e podem ser usados para algum processamento posterior.

6.2 ESTRATÉGIA DE OTIMIZAÇÃO COM O *BUFFER* DUPLO

Esta seção apresenta uma estratégia envolvendo o *buffer* duplo para aumentar o desempenho de processamento da etapa de localização. Em qualquer plataforma HW/SW existe a questão do tempo de comunicação entre os núcleos de processamento que impacta diretamente no desempenho do processamento do sistema. Embora a estratégia tenha sido implementada para a plataforma HARPv2 ela se aplica a qualquer plataforma HW/SW. Na Seção 6.2.1 nós definimos o *overhead* de comunicação, Na Seção 6.2.2 nós apresentamos a solução de *buffer* duplo para diminuir esse *overhead*.

6.2.1 *Overhead* de Comunicação

Sempre que o módulo SGM termina de ler um determinado par de *frames* estéreo, este módulo tem que sinalizar para o software para que este armazene um novo par de *frames* na memória compartilhada e sinalize para o hardware sobre os novos dados. Esse processo de escrita e sinalização adiciona uma espera que faz com que o módulo SGM fique ocioso sem processar nenhum *frame*.

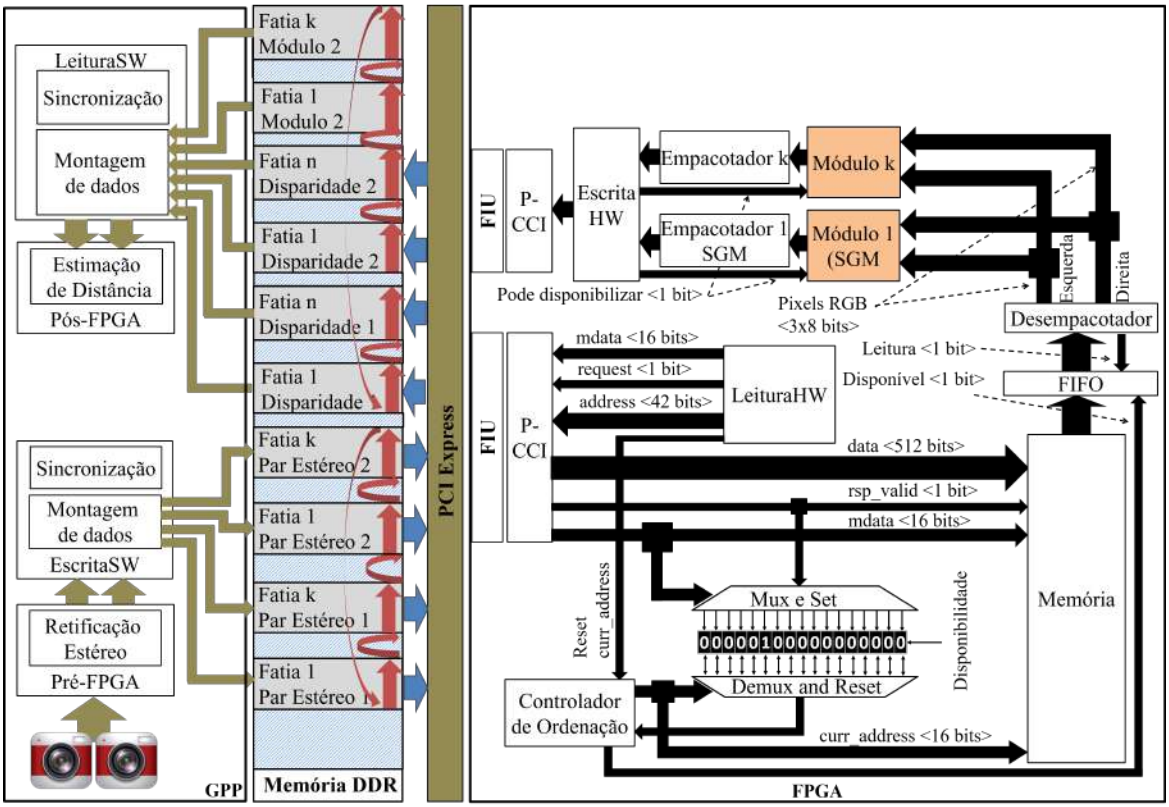
Da mesma forma acontece com o resultado do módulo SGM. O módulo EscritaHW só pode escrever um novo mapa de disparidade na memória compartilhada quando o software já comunicou que consumiu o mapa de disparidade anterior. Enquanto não consumiu, o módulo SGM não pode processar um novo mapa de disparidade senão a resposta sobrescreverá dados que não foram consumidos pelo software. Dessa forma, mais atrasos são adicionados no processamento do mapa de disparidade.

6.2.2 *Buffer* duplo

Para contornar esses atrasos de espera de comunicação, nós propomos uma abordagem que envolve a alocação de duas regiões de memória, de igual tamanho para armazenamento de dados de entrada e saída que são compartilhados com o módulo SGM, como mostrado na Figura 83. Nós chamamos esta estratégia de *buffer* duplo. A versão com uma região de memória compartilhada para entrada e saída nós chamamos de *buffer* único.

Essas regiões permitem armazenar dois conjuntos de dados. Um conjunto pode ser dois ou mais pares de *frames*. As variáveis de estado são atribuídas a cada *buffer* e compartilhadas com o módulo LeituraHW para indicar que o software acabou de preencher o *buffer*, ou seja, há dados disponíveis para processamento pelos módulos de hardware. Outras variáveis de estado também permitem que o módulo SGM sinalize o término da leitura de um *buffer* específico, ou seja, indicando que o espaço está disponível neste *buffer* para a escrita, pelo software, de um novo conjunto de dados.

Figura 83 – Estratégia de *buffer* duplo. A diferença para a arquitetura da Figura 82 está na duplicidade da memória para armazenamento das imagens de entrada como para armazenamento das respostas dos módulos. Esta estratégia de *buffer* duplo, associada com a estratégia de reordenamento de dados e paginação de memória, permite aumentar o desempenho da integração.



Fonte: O autor.

Este esquema de *buffer* duplo garante que sempre haverá dados disponíveis em qualquer *buffer*, evitando que o módulo SGM não espere que o software adicione o próximo conjunto de *frames* estéreo para iniciar seu processamento. Essa estratégia de *buffer* duplo também é realizada na comunicação da resposta do módulo SGM, para evitar atrasos do software ao ler os dados de resposta. O *buffer* duplo na resposta é gerenciado pelo módulo EscritaHW, no lado do hardware, e pela função LeituraSW, no lado do software.

6.3 RESULTADOS

Esta seção apresenta os resultados do módulo de hardware para visão estéreo e da estimativa de distância para a tarefa de localização.

6.3.1 Avaliação

Nós utilizamos a base de dados 1 para a avaliação da tarefa de localização. Primeiramente, nós apresentamos as configurações das abordagens para correspondência estéreo envolvidas nos experimentos, depois descrevemos a estratégia de avaliação geral e em seguida apresentamos os resultados e análises.

6.3.1.1 Definição das Abordagens de Visão Estéreo

Nós avaliamos duas abordagens de SGM. A primeira é a abordagem proposta em Hernandez-Juarez et al. (2016). Essa abordagem possui 8 caminhos de propagação de custos simétricos e foi implementada em plataforma GPU. A técnica de transformada Census é proposta nessa abordagem com janela de custo local de 9×7. Nós adicionamos o suporte para detecção de pixels ocluídos nessa primeira abordagem. Os parâmetros P_1, P_2, N_d e σ_{SGM} para essa primeira abordagem são definidos, respectivamente, como 24, 120, 128 e 20. Mais informações e conceitos sobre o SGM podem ser encontrados na Seção 3.1.1.1. Nós chamamos essa versão de SGM-CUDA. A segunda abordagem é a técnica proposta pelo autor em (CAMBUIM et al., 2019) implementada em FPGA com 4 caminhos de propagação não simétricos com uma janela de custo local de 2×1 e com suporte a pixels ocluídos. Os parâmetros P_1, P_2, N_d, σ_{SGM} para essa abordagem são definidos, respectivamente, como 5, 40, 128 e 1. Nós chamamos essa segunda solução de SGM-FPGA.

6.3.1.2 Estratégia de Avaliação

Nós utilizamos a base de dados 1 para a avaliação da tarefa de localização. Nós utilizamos a métrica de erro médio e taxa de faltas por falsos positivos por imagem (FPPI) para avaliação. Para a realização de análises, nós consideramos as componentes lateral e longitudinal a partir da etapa de estimativa da distância.

Uma vez que, nesse capítulo, nós não avaliamos a componente de detecção de pedestres, nós consideramos como entrada da etapa de localização os *bounding boxes* de referência a partir da base de dados. Com esses *bounding boxes*, nós adicionamos alguns erros como redução de *bounding box* para analisar a robustez da localização tanto das abordagens de correspondência estéreo como do algoritmo de estimativa da distância. Esse erro nós chamamos de Fator de Erro que multiplica cada lado do *bounding box* de referência para gerar o *bounding box* modificado. Nós variamos o parâmetro de Fator de Erro do *bounding box* no intervalo de 0,3 até 1,0 com passo de 0,1. Nós consideramos a divisão de amostragem por grupos de distâncias, como descritos na Seção 3.3.2, para analisar o comportamento das medições.

6.3.1.3 Experimentos, análises e comparações

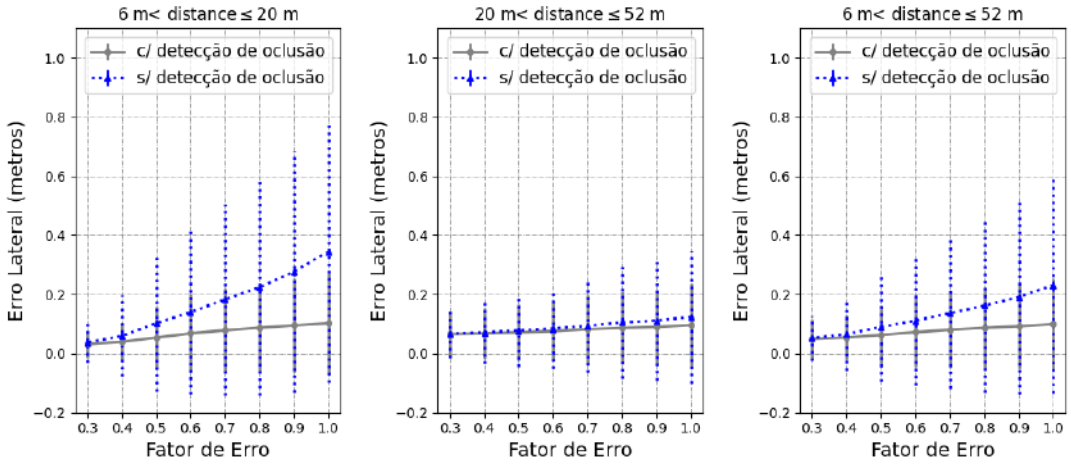
A partir da definição das abordagens de correspondência estéreo e da estratégia de avaliação nós conduzimos os experimentos nas seções seguintes.

6.3.1.4 Detecção de Oclusão e Filtragem de Histograma

Primeiramente nós analisamos a precisão da localização com a adição da detecção de oclusão implementada no módulo SGM. O conceito de pixels de oclusão pode ser visto na Seção 2.1.2. Sendo breve, estes pixels geram valores errados de disparidades e precisam

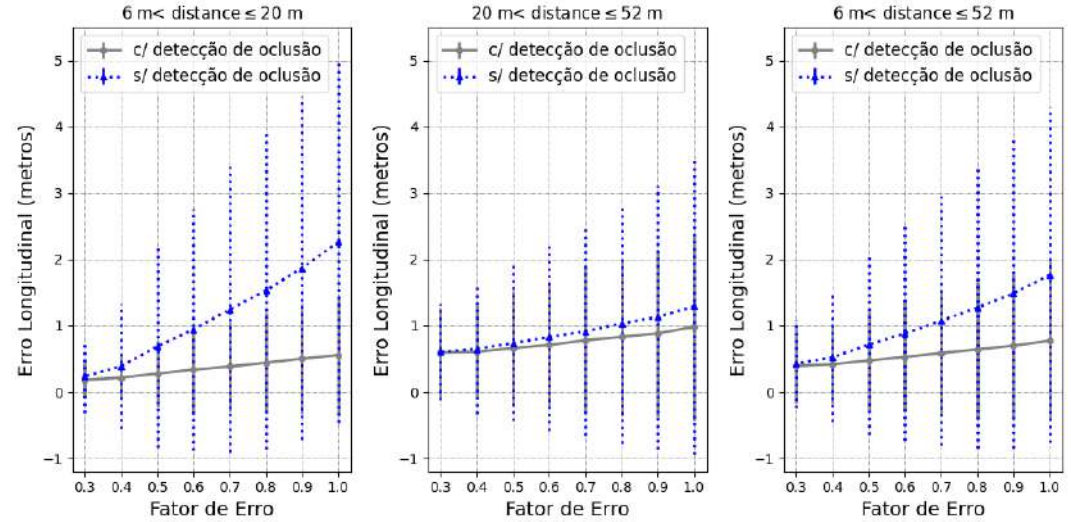
ser detectados e desconsiderados na estimativa de localização. As Figuras 84 e 85 mostram respectivamente o erro médio e desvio padrão da estimativa de localização lateral e longitudinal, respectivamente.

Figura 84 – Resultados de qualidade medidos através do erro médio e desvio padrão da estimativa de localização lateral com o módulo SGM em FPGA com e sem a detecção de oclusão



Fonte: O autor.

Figura 85 – Resultados de qualidade medidos através do erro médio e desvio padrão da estimativa de localização longitudinal com o módulo SGM em FPGA com e sem a detecção de oclusão

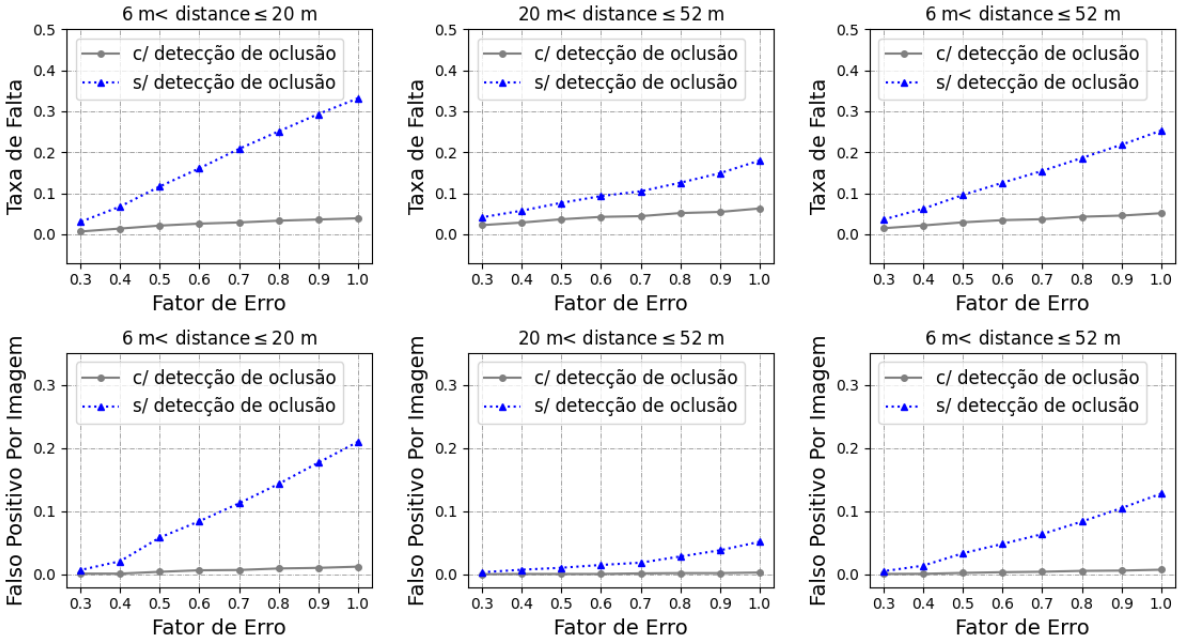


Fonte: O autor.

A Figura 86 mostra a relação de taxa de faltas e FPPI. Todas as figuras envolvem resultados da abordagem de correspondência estéreo SGM-FPGA variando o parâmetro de Fator de Erro do *bounding box*. Para não influenciar nos resultados, nós desativamos a abordagem de filtragem de histograma da etapa de estimativa da distância.

Nós podemos observar que a detecção de oclusão permitiu reduzir consideravelmente o erro na estimativa da localização lateral e longitudinal e que também se refletiu na

Figura 86 – Resultados de qualidade da estimativa de localização medidos através da taxa de faltas e falso positivo por imagem usando o módulo SGM em FPGA com e sem a detecção de oclusão.



Fonte: O autor.

redução da taxa de faltas e falso positivo por imagem. Nós também podemos observar que na medida em que a dimensão da janela se aproxima do tamanho original, o erro é ainda maior sem a detecção de oclusão.

6.3.1.5 Filtragem por Histograma

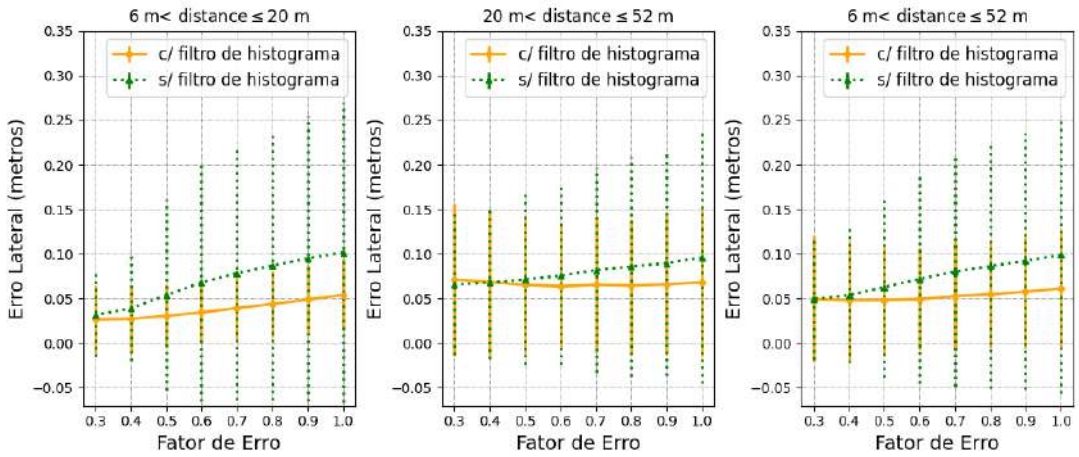
Nesse experimento nós analisamos o impacto da etapa de filtragem por histograma. As Figuras 87 e 88 mostram respectivamente o erro médio e desvio padrão da estimativa de localização lateral e longitudinal, respectivamente.

A Figura 89 mostra a relação da taxa de faltas e FPPI. Todas as figuras envolvem resultados da abordagem de correspondência estéreo SGM-FPGA variando o parâmetro de fator de erro do *bounding box*. Nós consideramos a detecção oclusão na avaliação com e sem a abordagem de filtragem de histograma.

Nós podemos observar mais reduções de erro com a adição da filtragem de histograma. Isso acontece porque a abordagem de filtragem exige uma quantidade mínima de pixels para serem escolhidos como pixels representativos do *bounding box*.

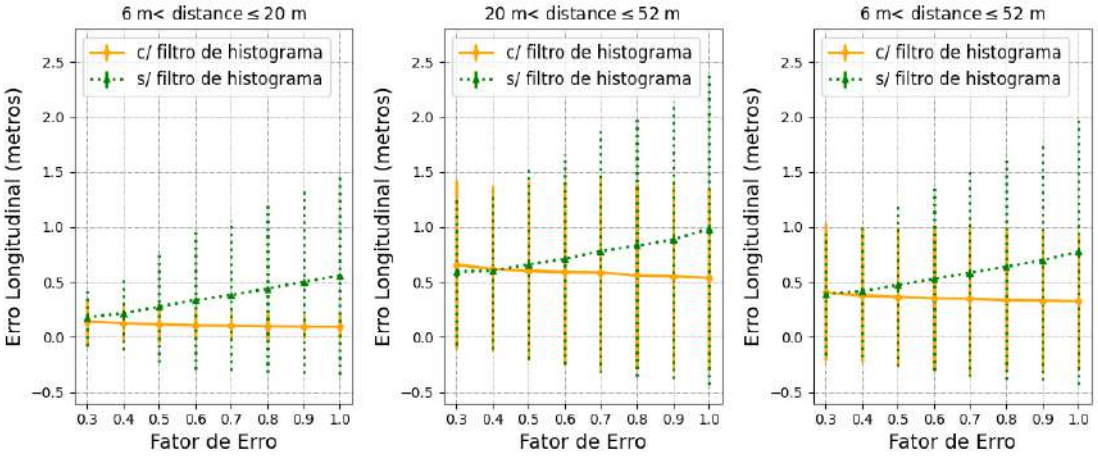
Outra observação é com relação a distância dos pedestres. Pedestres mais distantes geram estimativas de distância menos precisas em ambas os componentes. Contudo quando observamos a taxa de faltas e FPPI, na Figura 89 o erro é quase zero porque na medida em que a distância aumenta, a tolerância do erro, definido na Seção 3.3.2, também é aumenta, aceitando medições mais imprecisas como um acerto.

Figura 87 – Resultados de qualidade medidos através do erro médio e desvio padrão da estimativa de localização lateral com o módulo SGM em FPGA com e sem a filtragem de histograma



Fonte: O autor.

Figura 88 – Resultados de qualidade medidos através do erro médio e desvio padrão da estimativa de localização longitudinal com o módulo SGM em FPGA com e sem a filtragem de histograma



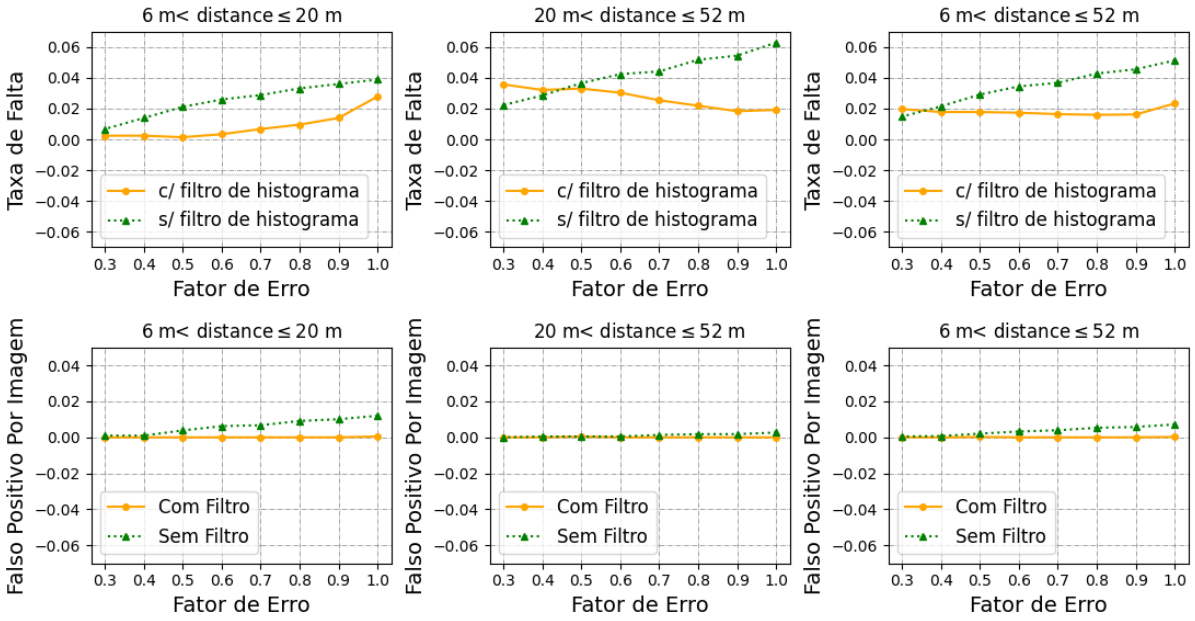
Fonte: O autor.

Outra análise diz respeito ao fator de erro sobre o *bounding box* de referência. Nós podemos observar que, mesmo com *bounding boxes* com dimensões erradas, o erro na estimativa das componentes laterais e longitudinais não aumentou. Esse resultado demonstra a robustez do algoritmo de estimativa de distância.

6.3.1.6 Comparação de precisão: SGM-CUDA vs SGM-FPGA

As Figuras 90 e 91 mostram respectivamente o erro médio e desvio padrão da estimativa de localização lateral e longitudinal, respectivamente. A Figura 92 mostra a relação da taxa de faltas e FPPI. Todas as figuras envolvem resultados das abordagens de correspondência estéreo SGM-CUDA e SGM-FPGA variando o parâmetro de fator de erro do *bounding box*. O parâmetro de estimativa de distância é o mesmo para as duas abordagens, a abordagem

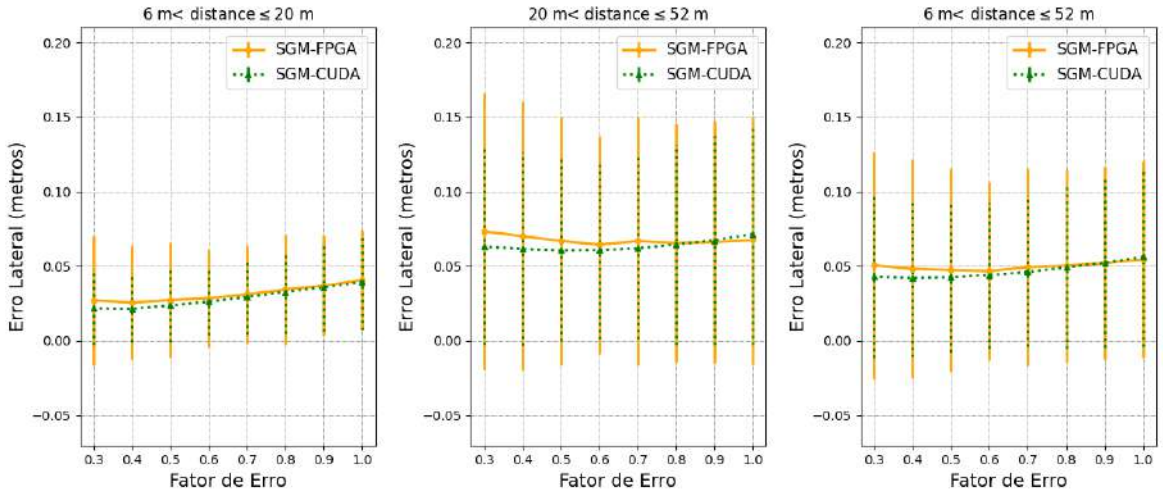
Figura 89 – Resultados de qualidade medidos através da taxa de faltas e falso positivo por imagem e da estimativa de localização com o módulo SGM em FPGA com e sem a filtragem de histograma.



Fonte: O autor.

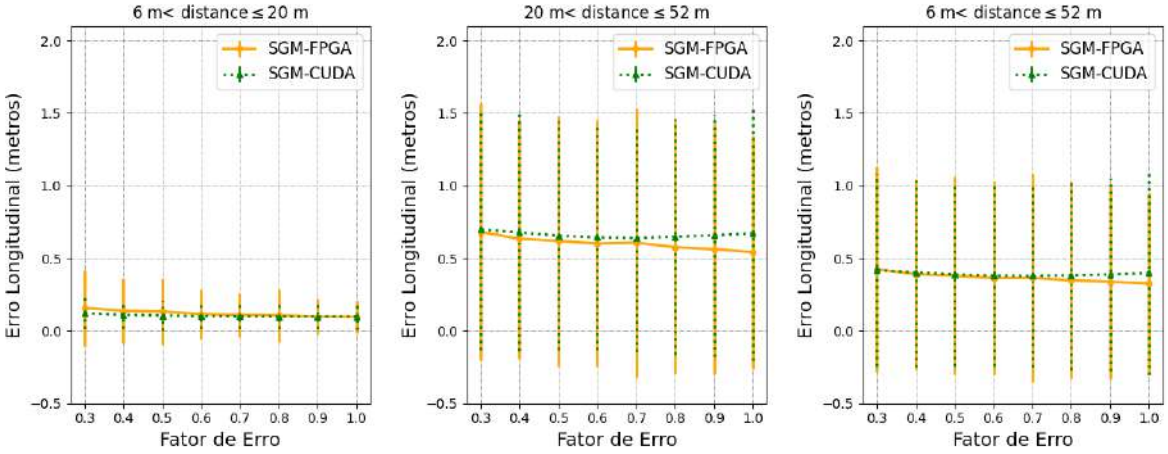
de filtragem por histograma e detecção de oclusão são consideradas.

Figura 90 – Resultados de qualidade medidos através do erro médio e desvio padrão da estimativa de localização lateral com o módulo SGM em FPGA e a implementação em GPU.



Fonte: O autor.

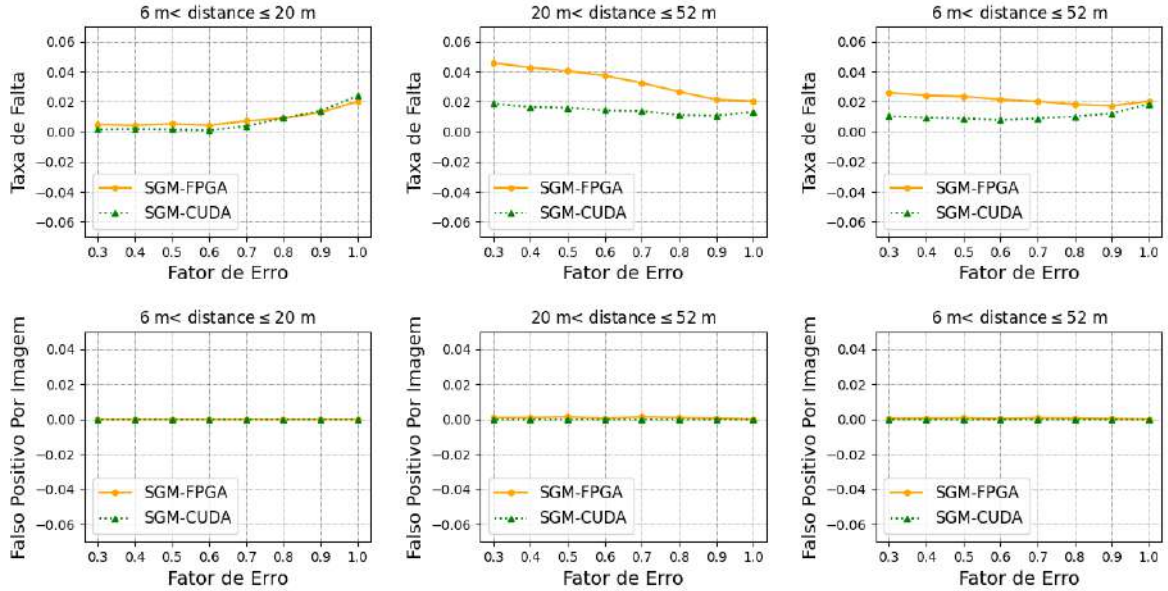
Figura 91 – Resultados de qualidade medidos através do erro médio e desvio padrão da estimativa de localização longitudinal com o módulo SGM em FPGA e a implementação em GPU.



Fonte: O autor.

A observação a ser realizada diz respeito a estimativa de distâncias com as abordagens SGM-CUDA e SGM-FPGA. É mostrado por Cambuim et al. (2019) que a abordagem SGM-CUDA gera mapas de disparidades mais precisos do que a abordagem SGM-FPGA por conta de ter mais caminhos de propagação simétricos (ou seja, 8 caminhos) e também ter dimensões de janelas locais maiores.

Figura 92 – Resultados de qualidade medidos através da taxa de faltas e falso positivo por imagem e da estimativa de localização com o módulo SGM em FPGA e a implementação em GPU.



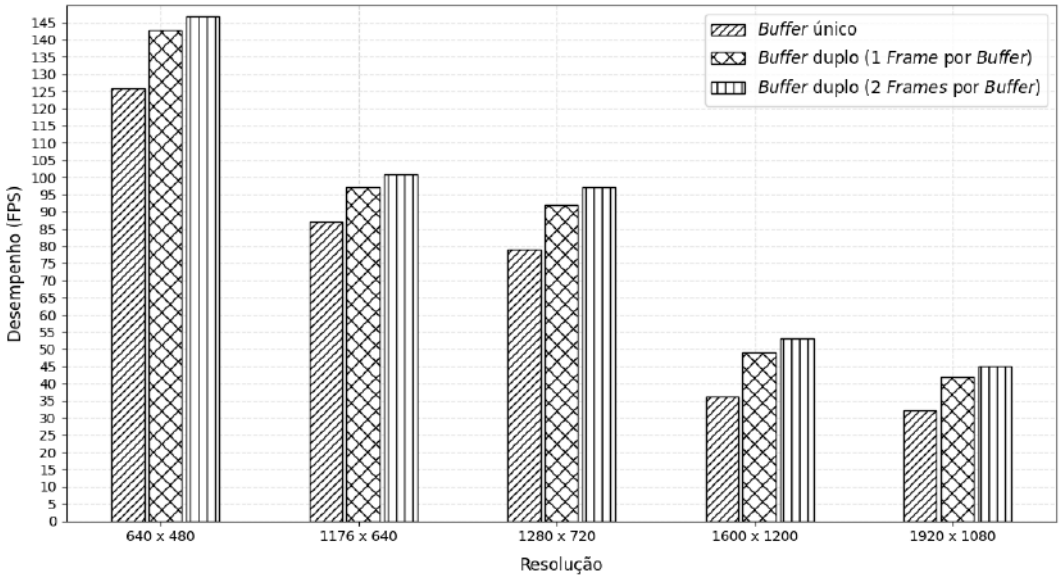
Fonte: O autor.

Contudo ao observar as Figuras 90 e 91 nós evidenciamos que essa perda reduz suavemente a precisão na estimativa de localização. Quando observamos a Figura 92, a diferença é ainda menor entre as duas abordagens. No pior caso, nós temos uma perda de 4% na taxa de faltas na abordagem SGM-FPGA contra 2% da abordagem SGM-CUDA.

6.3.1.7 Comparação de desempenho: *Buffer* único vs *Buffer* duplo

Nós realizamos uma análise comparativa do desempenho de processamento envolvendo duas estratégias de compartilhamento de dados: (1) *buffer* único e (2) *buffer* duplo.

Figura 93 – Comparação de desempenho de processamento com o *buffer* único e o *buffer* duplo.



Fonte: O autor.

No *buffer* único, nós temos um espaço de memória compartilhado que contém apenas um par de *frames* estéreo e um espaço de memória compartilhado para armazenar o mapa de disparidade resultante. Nós instanciamos duas threads em que uma thread é responsável por enviar os pares de imagem estéreo, enquanto a outra é responsável por receber o mapa de disparidade processado.

A etapa de retificação foi implementada usando as funções da biblioteca OpenCV. Estimamos o tempo médio de retificação e o tempo médio de comunicação para enviar e receber 1.000 *frames* pelo barramento PCIe. Para ter a mesma quantidade de dados transferidos entre hardware e software com o módulo SGM, enviamos um par de *frames* de 3 canais e, em seguida, no hardware, enviamos de volta um *frame* de um canal.

Nós repetimos este experimento para várias resoluções de imagem. A frequência de operação no módulo SGM em todas as configurações foi definida como 150 MHz. Como o intervalo de disparidade (N_d) não afeta o desempenho do sistema, conforme evidenciado em Cambuim et al. (2019), foram considerados 128 intervalos.

Com o *buffer* duplo, estimamos o desempenho para diferentes quantidades de *frames* por *buffer*. Como pode ser observado na Figura 93, a técnica proposta usando uma estratégia de *buffer* duplo resultou em um ganho de desempenho em relação à estratégia de *buffer* único para duas configurações de *buffer* duplo. A adição da estratégia de *buffer* duplo usando dois *frames* por buffer resultou em ganhos de desempenho de processamento de 1,2x e 1,6x nas resoluções de 640 $\times$ 480 e 1920 $\times$ 1080 pixels, respectivamente. Esse ganho ocorreu porque a espera para processar os próximos *frames* no hardware foi reduzida. Nós acreditamos que o aumento na quantidade de *frames* por *buffer* permitirá aumentar ainda mais na taxa de processamento.

6.4 RESUMO

Esse capítulo apresentou uma solução para localização de pedestres baseado na integração hardware/software do módulo hardware de correspondência estéreo SGM. Nós apresentamos uma solução baseada em *buffer* duplo para aumentar o desempenho de processamento do mapa de disparidade. Nós utilizamos a plataforma HARPv2 no qual introduzimos algumas camadas de soluções para garantir o processamento de *frames* em alta resolução e o ordenamento dos dados no lado do hardware. Nós mostramos que a estimativa de localização usando módulo de hardware praticamente é a mesma que uma solução de SGM que processa em GPU com mais caminhos de propagação. Nós mostramos que a estratégia de *buffer* duplo garantiu um ganho em desempenho de até 1,6x comparada com a abordagem de *buffer* único.

7 CONCLUSÃO

Este capítulo tem como objetivo apresentar as considerações finais sobre os principais tópicos abordados nesta tese, incluindo as contribuições alcançadas e indicações para trabalhos futuros.

7.1 CONSIDERAÇÕES FINAIS

Sistemas de Previsão de Colisão de Pedestre (PCP) são fundamentais para garantir a maior segurança dos pedestres prevendo colisões com antecedência e dessa forma reduzindo a quantidade de colisões. Sistemas PCP atrativos são aqueles capazes de garantir distâncias maiores para tomada de decisão. Dois aspectos são fundamentais para aumentar esta distância: tempo de resposta e cobertura de pedestres em grandes intervalos de distância.

7.2 PRINCIPAIS CONTRIBUIÇÕES

Para endereçar esses dois aspectos, nós construímos nesta Tese um sistema PCP baseado em visão computacional com otimização de desempenho e precisão. Este sistema compreendeu as etapas de retificação estéreo, detecção de pedestres, correspondência estéreo, localização, filtragem de contexto, rastreamento, previsão de trajetória e previsão de colisão. Por serem intensivas em processamento de dados e críticas para a precisão da previsão, as etapas otimizadas nesta Tese foram: a detecção de pedestres e correspondência estéreo.

Para a etapa de detecção de pedestres, nós implementamos um módulo em FPGA que processa a técnica baseada em HOG e SVM com suporte a pirâmide de imagens e janelas de detecção de diferentes dimensões. A fim de alcançar alto desempenho de processamento desse detector, diversas estratégias de otimização em nível de *frame* e em nível de janelas de detecção foram propostos. Em nível de *frame*, os módulos são instanciados em cada nível da pirâmide para processamento paralelo de janelas de detecção de dimensões iguais e diferentes. No nível de janelas de detecção, um conjunto de unidades SVM é projetado para processar em paralelo várias janelas de detecção sobrepostas. Essas estratégias garantem uma vazão máxima de entrada de pixels serial, sem perda de *frames* e sem interrupção de entrada de pixels. Essa abordagem proporcionou um ganho no desempenho de processamento em comparação com outros detectores existentes de pelo menos 3 vezes e um desempenho independente da profundidade da pirâmide e da quantidade de janelas de detecção de diferentes tamanhos. Além disso, resultados mostraram que a combinação de janelas de detecção de diferentes dimensões permitiu detectar mais pedestres distantes com um aumento de 10% na taxa de detecção.

Para aumentar a escalabilidade do detector ao aumentar a profundidade da pirâmide e a quantidade de janelas de detecção de diferentes dimensões, nós implementamos várias estratégias para redução de recursos de hardware. A principal estratégia envolveu a redução da quantidade de memória, através de uma abordagem de reuso da memória de pesos e da divisão dessa memória em linhas para serem compartilhadas por várias unidades SVM diferentes. Nós conseguimos reduzir em pelo menos 3,2 vezes, com 3 níveis de pirâmide e 2 janelas de dimensões diferentes, a quantidade de memória exigida. Todas as estratégias de redução não impactaram no desempenho de processamento porque o fluxo normal de processamento não foi alterado.

Para a etapa de localização, nós realizamos melhorias no módulo de hardware que processa a técnica SGM, para tornar o mapa de disparidade mais preciso, e adicionamos o suporte para detecção de pixels de oclusão. Essas melhorias estão detalhadas em nosso trabalho (CAMBUIM et al., 2019). Nós propusemos uma abordagem para estimativa da distância do pedestre baseada em checagem de histograma, filtragem de pixels de oclusão e filtragem por segmentação de estrada, que garantiu robustez na localização do pedestre, mesmo na presença de *bounding boxes* imprecisos, reduziu os falsos positivos e aumentou a taxa de detecção de pedestres. Outros resultados mostraram que o nosso módulo de hardware SGM, integrado com a abordagem de estimativa de distância proposta, conseguiram alcançar uma precisão na estimativa de distâncias semelhante a técnica SGM de maior custo computacional, que processa mais direções de propagação.

Nós realizamos a integração dos módulos de detecção e correspondência estéreo no sistema PCP e validamos essa integração na plataforma HARPv2 da Intel na qual apresentamos resultados de precisão e desempenho do detector para localização e previsão de colisão do pedestre. Para aumentar o desempenho da integração, nós propusemos estratégias usando um *buffer* duplo, reordenamento de dados e paginação de memória que garantiu um ganho de desempenho de 1,6 vezes, processando *frames* com resolução de 1920×1080 pixels.

Nós propusemos cenários sintéticos de colisão com pedestres que permitiram analisar a qualidade de sistemas de previsão de colisão levando-se em consideração aspectos de tempo de processamento. Nós demonstramos que o uso de pirâmide de imagens e as janelas de dimensões diferentes aumentaram a capacidade do detector para localizar e prever colisão com pedestres, respectivamente, mais próximos e mais distantes. O ganho de desempenho de processamento do detector de hardware proposto deu ao sistema PCP uma capacidade de evitar mais colisões em relação a outros sistemas PCP mais lentos. Por exemplo, o detector de hardware garantiu 100% de previsão segura, ou seja, sem colidir, nos cenários que de fato são possíveis de evitar colisão, contra 89% com o detector baseado em OpenCV e 79% com o detector baseado em YOLOv3. Além disso, esse ganho de processamento também garantiu um aumento na distância de tomada de decisão de 6,5 metros que permite mitigar o impacto de colisões quando não é possível evitar a colisão.

7.3 TRABALHOS FUTUROS

Para dar continuidade ao trabalho de pesquisa descrito nesta Tese, lista-se, nesta seção, propostas de trabalhos futuros a serem realizadas.

Como demonstrado na Seção 5.4.2, o gargalo de comunicação impede que o sistema PCP alcance taxas maiores de processamento. A razão principal desse gargalo é a necessidade de envio de pares de *frames* estéreo e o recebimento de mapa de disparidade. A implementação em hardware da etapa de estimativa de distância reduzirá este gargalo, uma vez que reduzirá a necessidade de envio do mapa de disparidade pelo barramento. Outra solução é conectar câmeras diretamente no hardware, uma vez que evitará a necessidade de envio pelo barramento dos pares de frames estéreos.

Não somente a implementação da etapa de estimativa de distância, mas de todas as etapas do sistema PCP, seguindo a abordagem de alto desempenho proposta para o módulo detector e correspondência estéreo, permitirá aumentar ainda mais as taxas de processamento do sistema PCP. Dessa forma, nós teremos um sistema totalmente embarcado que pode ser integrado em plataformas FPGA de mais baixo porte, alcançando altas taxas de processamento. Uma vez que provamos que taxas maiores de processamento garantem melhor eficiência na previsão de colisão, certamente, mais melhorias no desempenho do sistema PCP trarão mais benefícios para a qualidade da previsão.

Trabalhos podem ser realizados na etapa de extração de características do detector para combinar o HOG com outros extratores em hardware a fim de aumentar a precisão e robustez do detector e ainda manter o alto desempenho proposto nesse trabalho. A adição das abordagens de interpolação trilinear e ponderação gaussiana permitirá também alcançar melhorias na precisão do módulo de detecção de pedestres. A abordagem baseada em partes deformáveis pode tomar proveito do nosso detector para lidar com pedestres com mais variabilidade de poses.

REFERÊNCIAS

Administração de Segurança de Tráfego de Estrada Nacional dos Estados Unidos. *Direção Distraída em Acidentes Fatais*. 2017. <https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812700/>.

ALI, A. A.; HUSSEIN, H. A. Distance estimation and vehicle position detection based on monocular camera. In: IEEE. *2016 Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications (AIC-MITCSA)*. [S.l.], 2016. p. 1–4.

ALONSO, I. P.; LLORCA, D. F.; SOTELO, M. Á.; BERGASA, L. M.; TORO, P. R. de; NUEVO, J.; OCAÑA, M.; GARRIDO, M. Á. G. Combination of feature extraction methods for svm pedestrian detection. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, v. 8, n. 2, p. 292–307, 2007.

BAILEY, D. G. *Design for Embedded Image Processing on FPGAs*. 1st. ed. [S.l.]: Wiley Publishing, 2011. ISBN 0470828498, 9780470828496.

BAILEY, D. G. *Image processing using FPGAs*. [S.l.]: MDPI, 2019. 53 p.

BAO, D.; WANG, P. Vehicle distance detection based on monocular vision. In: IEEE. *2016 International Conference on Progress in Informatics and Computing (PIC)*. [S.l.], 2016. p. 187–191.

BAR-SHALOM, Y.; LI, X. R.; KIRUBARAJAN, T. *Estimation with applications to tracking and navigation: theory algorithms and software*. [S.l.]: John Wiley & Sons, 2004.

BENENSON, R.; OMRAN, M.; HOSANG, J.; SCHIELE, B. Ten years of pedestrian detection, what have we learned? In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2014. p. 613–627.

BENENSON, R.; OMRAN, M.; HOSANG, J.; SCHIELE, B. Ten years of pedestrian detection, what have we learned? In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2014. p. 613–627.

BERGERON, J.; CERNY, E.; HUNTER, A.; NIGHTINGALE, A. *Verification methodology manual for SystemVerilog*. [S.l.]: Springer, 2006. v. 1.

BERTOZZI, M.; BROGGI, A.; FASCIOLI, A.; TIBALDI, A.; CHAPUIS, R.; CHAUSSE, F. Pedestrian localization and tracking system with kalman filtering. In: IEEE. *IEEE Intelligent Vehicles Symposium, 2004*. [S.l.], 2004. p. 584–589.

BEWLEY, A.; GE, Z.; OTT, L.; RAMOS, F.; UPCROFT, B. Simple online and realtime tracking. In: IEEE. *2016 IEEE international conference on image processing (ICIP)*. [S.l.], 2016. p. 3464–3468.

BHOWMIK, P.; PANTHO, M. J. H.; BOBDA, C. Event-based re-configurable hierarchical processors for smart image sensors. In: IEEE. *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. [S.l.], 2019. v. 2160, p. 115–122.

BHOWMIK, P.; PANTHO, M. J. H.; BOBDA, C. Harp: Hierarchical attention oriented region-based processing for high-performance computation in vision sensor. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 21, n. 5, p. 1757, 2021.

BILAL, M.; HANIF, M. S. Benchmark revision for hog-svm pedestrian detector through reinvigorated training and evaluation methodologies. *IEEE transactions on intelligent transportation systems*, IEEE, v. 21, n. 3, p. 1277–1287, 2019.

BINELLI, E.; BROGGI, A.; FASCIOLI, A.; GHIDONI, S.; GRISLERI, P.; GRAF, T.; MEINECKE, M. A modular tracking system for far infrared pedestrian recognition. In: IEEE. *IEEE Proceedings. Intelligent Vehicles Symposium, 2005.* [S.l.], 2005. p. 759–764.

BIRCHFIELD, S.; TOMASI, C. A Pixel Dissimilarity Measure That Is Insensitive to Image Sampling. *IEEE Trans. Pattern Anal. Mach. Intell.*, IEEE Computer Society, Washington, DC, USA, v. 20, n. 4, p. 401–406, abr. 1998. ISSN 0162-8828. Disponível em: <http://dx.doi.org/10.1109/34.677269>.

BODLA, N.; SINGH, B.; CHELLAPPA, R.; DAVIS, L. S. Soft-nms–improving object detection with one line of code. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2017. p. 5561–5569.

BREHAR, R.; VANCEA, F.; MARIŢA, T.; NEDEVSCHI, S. A deep learning approach for pedestrian segmentation in infrared images. In: IEEE. *2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP)*. [S.l.], 2018. p. 253–258.

BRENIERE, Y.; DO, M. When and how does steady state gait movement induced from upright posture begin? *Journal of biomechanics*, Elsevier, v. 19, n. 12, p. 1035–1040, 1986.

BROGGI, A.; CARDARELLI, E.; CATTANI, S.; SABBATELLI, M. Terrain mapping for off-road autonomous ground vehicles using rational b-spline surfaces and stereo vision. In: IEEE. *2013 IEEE Intelligent Vehicles Symposium (IV)*. [S.l.], 2013. p. 648–653.

CAFISO, S.; GRAZIANO, A. D.; PAPPALARDO, G. In-vehicle stereo vision system for identification of traffic conflicts between bus and pedestrian. *Journal of traffic and transportation engineering (English edition)*, Elsevier, v. 4, n. 1, p. 3–13, 2017.

CAMBUIM, L. F.; BARBOSA, J. P.; BARROS, E. N. Hardware module for low-resource and real-time stereo vision engine using semi-global matching approach. In: *Proceedings of the 30th Symposium on Integrated Circuits and Systems Design: Chip on the Sands*. [S.l.: s.n.], 2017. p. 53–58.

CAMBUIM, L. F. S.; OLIVEIRA, L. A.; BARROS, E. N. S.; FERREIRA, A. P. A. An fpga-based real-time occlusion robust stereo vision system using semi-global matching. *Journal of Real-Time Image Processing*, Jul 2019. ISSN 1861-8219. Disponível em: <https://doi.org/10.1007/s11554-019-00902-w>.

DAI, X.; HU, J.; ZHANG, H.; SHITU, A.; LUO, C.; OSMAN, A.; SFARRA, S.; DUAN, Y. Multi-task faster r-cnn for nighttime pedestrian detection and distance estimation. *Infrared Physics & Technology*, Elsevier, v. 115, p. 103694, 2021.

DALAL, N.; TRIGGS, B. Histograms of oriented gradients for human detection. In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*. USA: IEEE Computer Society, 2005. (CVPR '05), p. 886–893. ISBN 0769523722. Disponível em: <https://doi.org/10.1109/CVPR.2005.177>.

DOLLAR, P.; WOJEK, C.; SCHIELE, B.; PERONA, P. Pedestrian detection: An evaluation of the state of the art. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 34, n. 4, p. 743–761, 2011.

DOSOVITSKIY, A.; ROS, G.; CODEVILLA, F.; LOPEZ, A.; KOLTUN, V. CARLA: An open urban driving simulator. In: *Proceedings of the 1st Annual Conference on Robot Learning*. [S.l.: s.n.], 2017. p. 1–16.

DURINI, D. *High performance silicon imaging: fundamentals and applications of cmos and ccd sensors*. [S.l.]: Woodhead Publishing, 2019.

DÜRRE, J.; PARADZIK, D.; BLUME, H. A hog-based real-time and multi-scale pedestrian detector demonstration system on fpga. In: *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. [S.l.: s.n.], 2018. p. 163–172.

ENZWEILER, M.; HUMMEL, M.; PFEIFFER, D.; FRANKE, U. Efficient stixel-based object recognition. In: IEEE. *2012 IEEE Intelligent Vehicles Symposium*. [S.l.], 2012. p. 1066–1071.

FARAGHER, R. Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes]. *IEEE Signal processing magazine*, IEEE, v. 29, n. 5, p. 128–132, 2012.

FELZENSZWALB, P. F.; GIRSHICK, R. B.; MCALLESTER, D.; RAMANAN, D. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, IEEE Computer Society, USA, v. 32, n. 9, p. 1627–1645, set. 2010. ISSN 0162-8828. Disponível em: <https://doi.org/10.1109/TPAMI.2009.167>.

FLOHR, F.; GAVRILA, D. et al. Pedcut: an iterative framework for pedestrian segmentation combining shape models and multiple data cues. In: *BMVC*. [S.l.: s.n.], 2013.

FORTMANN, T.; BAR-SHALOM, Y.; SCHEFFE, M. Sonar tracking of multiple targets using joint probabilistic data association. *IEEE journal of Oceanic Engineering*, IEEE, v. 8, n. 3, p. 173–184, 1983.

FRANK, A. On kuhn's hungarian method—a tribute from hungary. *Naval Research Logistics (NRL)*, Wiley Online Library, v. 52, n. 1, p. 2–5, 2005.

GAO, H.; CHENG, B.; WANG, J.; LI, K.; ZHAO, J.; LI, D. Object classification using cnn-based fusion of vision and lidar in autonomous vehicle environment. *IEEE Transactions on Industrial Informatics*, IEEE, v. 14, n. 9, p. 4224–4231, 2018.

Gerónimo, D.; López, A. M.; Sappa, A. D.; Graf, T. Survey of pedestrian detection for advanced driver assistance systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 32, n. 7, p. 1239–1258, July 2010. ISSN 1939-3539.

GOLDHAMMER, M.; DOLL, K.; BRUNSMANN, U.; GENSLER, A.; SICK, B. Pedestrian's trajectory forecast in public traffic with artificial neural networks. In: IEEE. *2014 22nd international conference on pattern recognition*. [S.l.], 2014. p. 4110–4115.

GOLDHAMMER, M.; HUBERT, A.; KOEHLER, S.; ZINDLER, K.; BRUNSMANN, U.; DOLL, K.; SICK, B. Analysis on termination of pedestrians' gait at urban intersections. In: IEEE. *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. [S.l.], 2014. p. 1758–1763.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep learning*. [S.l.]: MIT press, 2016.

GOVARDHAN, P.; PATI, U. C. Nir image based pedestrian detection in night vision with cascade classification and validation. In: IEEE. *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies*. [S.l.], 2014. p. 1435–1438.

GRIBBON, K. T.; BAILEY, D. G. A novel approach to real-time bilinear interpolation. In: IEEE. *Proceedings. DELTA 2004. Second IEEE International Workshop on Electronic Design, Test and Applications*. [S.l.], 2004. p. 126–131.

GRIMM, C.; FEI, T.; WARSITZ, E.; FARHOUD, R.; BREDDERMANN, T.; HAEB-UMBACH, R. Warping of radar data into camera image for cross-modal supervision in automotive applications. *IEEE Transactions on Vehicular Technology*, IEEE, v. 71, n. 9, p. 9435–9449, 2022.

GUINDEL, C.; MARTíN, D.; ARMINGOL, J. M. Traffic scene awareness for intelligent vehicles using convnets and stereo vision. *Robotics and Autonomous Systems*, v. 112, p. 109 – 122, 2019.

GUPTA, P. Accelerating datacenter workloads. In: *26th International Conference on Field Programmable Logic and Applications (FPL)*. [S.l.: s.n.], 2016.

HAAS, R. E.; BHATTACHARJEE, S.; MÖLLER, D. P. F. Advanced driver assistance systems. In: _____. *Smart Technologies : Scope and Applications*. Singapore: Springer Singapore, 2020. p. 345–371. ISBN 978-981-13-7139-4. Disponível em: <https://doi.org/10.1007/978-981-13-7139-4_27>.

HAHNLE, M.; SAXEN, F.; HISUNG, M.; BRUNSMANN, U.; DOLL, K. Fpga-based real-time pedestrian detection on high-resolution images. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. [S.l.: s.n.], 2013. p. 629–635.

HAMDANE, H.; SERRE, T.; MASSON, C.; ANDERSON, R. Issues and challenges for pedestrian active safety systems based on real world accidents. *Accident Analysis & Prevention*, Elsevier, v. 82, p. 53–60, 2015.

HAN, D. Comparison of commonly used image interpolation methods. In: ATLANTIS PRESS. *Conference of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*. [S.l.], 2013. p. 1556–1559.

HARTLEY, R.; ZISSERMAN, A. *Multiple View Geometry in Computer Vision*. 2. ed. New York, NY, USA: Cambridge University Press, 2003. ISBN 0521540518.

HELALI, A.; AMEUR, H.; GÓRRIZ, J.; RAMÍREZ, J.; MAAREF, H. Hardware implementation of real-time pedestrian detection system. *Neural Computing and Applications*, Springer, v. 32, n. 16, p. 12859–12871, 2020.

HELD, D.; THRUN, S.; SAVARESE, S. Learning to track at 100 fps with deep regression networks. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 749–765.

HERNANDEZ-JUAREZ, D.; CHACÓN, A.; ESPINOSA, A.; VÁZQUEZ, D.; MOURE, J. C.; LÓPEZ, A. M. Embedded real-time stereo estimation via semi-global matching on the gpu. *Procedia Computer Science*, Elsevier, v. 80, p. 143–153, 2016.

HILLENBRAND, J.; SPIEKER, A. M.; KROSCHEL, K. A multilevel collision mitigation approach—its situation assessment, decision making, and performance tradeoffs. *IEEE Transactions on intelligent transportation systems*, IEEE, v. 7, n. 4, p. 528–540, 2006.

HIRSCHMULLER, H. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 30, n. 2, p. 328–341, Feb 2008. ISSN 0162-8828.

HIRSCHMULLER, H.; SCHARSTEIN, D. Evaluation of stereo matching costs on images with radiometric differences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 31, n. 9, p. 1582–1599, Sept 2009. ISSN 0162-8828.

HYUN, J.; MOON, B. A simplified rectification method and its hardware architecture for embedded multimedia systems. *Multimedia Tools and Applications*, Springer, v. 76, n. 19, p. 19761–19779, 2017.

JIAO, J.; WANG, R.; WANG, W.; DONG, S.; WANG, Z.; GAO, W. Local stereo matching with improved matching cost and disparity refinement. *IEEE MultiMedia*, IEEE, v. 21, n. 4, p. 16–27, 2014.

JURECKI, R. S.; STAŃCZYK, T. L. Driver reaction time to lateral entering pedestrian in a simulated crash traffic situation. *Transportation research part F: traffic psychology and behaviour*, Elsevier, v. 27, p. 22–36, 2014.

KALAKE, L.; DONG, Y.; WAN, W.; HOU, L. Enhancing detection quality rate with a combined hog and cnn for real-time multiple object tracking across non-overlapping multiple cameras. *Sensors*, MDPI, v. 22, n. 6, p. 2123, 2022.

KALMAN, R. E. A new approach to linear filtering and prediction problems. 1960.

KANOPOULOS, N.; VASANTHAVADA, N.; BAKER, R. L. Design of an image edge detection filter using the sobel operator. *IEEE Journal of solid-state circuits*, IEEE, v. 23, n. 2, p. 358–367, 1988.

KARG, M.; SCHARFENBERGER, C. Deep learning-based pedestrian detection for automated driving: Achievements and future challenges. In: *Development and Analysis of Deep Learning Architectures*. [S.l.]: Springer, 2020. p. 117–143.

KAUR, S.; SINGH, K. Implementation of high speed fixed point cordic techniques. *International Journal of Computer Applications*, Citeseer, v. 56, n. 3, p. 35–41, 2012.

KELLER, C. G. *Stereo-based Pedestrian Detection and Path Prediction*. Tese (Doutorado) — University of Heidelberg, Germany, 2014.

KELLER, C. G.; DANG, T.; FRITZ, H.; JOOS, A.; RABE, C.; GAVRILA, D. M. Active pedestrian safety by automatic braking and evasive steering. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, v. 12, n. 4, p. 1292–1304, 2011.

KELLER, C. G.; ENZWEILER, M.; GAVRILA, D. M. A new benchmark for stereo-based pedestrian detection. In: IEEE. *2011 IEEE Intelligent Vehicles Symposium (IV)*. [S.l.], 2011. p. 691–696.

KELLER, C. G.; ENZWEILER, M.; ROHRBACH, M.; LLORCA, D. F.; SCHNORR, C.; GAVRILA, D. M. The benefits of dense stereo for pedestrian detection. *IEEE transactions on intelligent transportation systems*, IEEE, v. 12, n. 4, p. 1096–1106, 2011.

Keller, C. G.; Gavrila, D. M. Will the pedestrian cross? a study on pedestrian path prediction. *IEEE Transactions on Intelligent Transportation Systems*, v. 15, n. 2, p. 494–506, April 2014. ISSN 1558-0016.

KELLY, P.; COOKE, E.; O'CONNOR, N.; SMEATON, A. Pedestrian detection using stereo and biometric information. In: SPRINGER. *International Conference Image Analysis and Recognition*. [S.l.], 2006. p. 802–813.

KIM, C.; LI, F.; CIPTADI, A.; REHG, J. M. Multiple hypothesis tracking revisited. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2015. p. 4696–4704.

KOEHLER, S.; GOLDHAMMER, M.; BAUER, S.; ZECHA, S.; DOLL, K.; BRUNSMANN, U.; DIETMAYER, K. Stationary detection of the pedestrian? s intention at intersections. *IEEE Intelligent Transportation Systems Magazine*, IEEE, v. 5, n. 4, p. 87–99, 2013.

KÖHLER, S.; GOLDHAMMER, M.; BAUER, S.; DOLL, K.; BRUNSMANN, U.; DIETMAYER, K. Early detection of the pedestrian's intention to cross the street. In: IEEE. *2012 15th International IEEE Conference on Intelligent Transportation Systems*. [S.l.], 2012. p. 1759–1764.

KOOIJ, J. F.; FLOHR, F.; POOL, E. A.; GAVRILA, D. M. Context-based path prediction for targets with switching dynamics. *International Journal of Computer Vision*, Springer, v. 127, n. 3, p. 239–262, 2019.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2012. p. 1097–1105.

KUHN, H. W. The hungarian method for the assignment problem. *Naval research logistics quarterly*, Wiley Online Library, v. 2, n. 1-2, p. 83–97, 1955.

LEE, B. Y.; LIEW, L. H.; CHEAH, W. S.; WANG, Y. C. Occlusion handling in videos object tracking: A survey. In: IOP PUBLISHING. *IOP conference series: earth and environmental science*. [S.l.], 2014. v. 18, n. 1, p. 012020.

LEE, H.-K.; SHIN, S.-G.; KWON, D.-S. Design of emergency braking algorithm for pedestrian protection based on multi-sensor fusion. *International Journal of Automotive Technology*, Springer, v. 18, n. 6, p. 1067–1076, 2017.

LEIGH, A.; PINEAU, J.; OLMEDO, N.; ZHANG, H. Person tracking and following with 2d laser scanners. In: IEEE. *2015 IEEE international conference on robotics and automation (ICRA)*. [S.l.], 2015. p. 726–733.

LI, Y.; ZHENG, Y.; MORYS, B.; PAN, S.; WANG, J.; LI, K. Threat assessment techniques in intelligent vehicles: A comparative survey. *IEEE Intelligent Transportation Systems Magazine*, IEEE, 2020.

LI, Z.; ZHU, K.; HUANG, X.; ZHAO, J.; XU, K. All silicon microdisplay fabricated utilizing 0.18 μm cmos-ic with monolithic integration. *IEEE Photonics Journal*, IEEE, v. 14, n. 2, p. 1–5, 2022.

LIU, S.; WAN, Z.; YU, B.; WANG, Y. Robotic computing on fpgas. *Synthesis Lectures on Computer Architecture*, Morgan & Claypool Publishers, v. 16, n. 1, p. 1–218, 2021.

LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S.; FU, C.-Y.; BERG, A. C. Ssd: Single shot multibox detector. In: SPRINGER. *European conference on computer vision*. [S.l.], 2016. p. 21–37.

LLORCA, D. F.; MILANÉS, V.; ALONSO, I. P.; GAVILÁN, M.; DAZA, I. G.; PÉREZ, J.; SOTELO, M. Á. Autonomous pedestrian collision avoidance using a fuzzy steering controller. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, v. 12, n. 2, p. 390–401, 2011.

LUO, J. H.; LIN, C. H. Pure fpga implementation of an hog based real-time pedestrian detection system. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 18, n. 4, p. 1174, 2018.

LUO, W.; XING, J.; MILAN, A.; ZHANG, X.; LIU, W.; KIM, T.-K. Multiple object tracking: A literature review. *Artificial Intelligence*, Elsevier, v. 293, p. 103448, 2021.

MA, W.; ZHU, S. A multifeature-assisted road and vehicle detection method based on monocular depth estimation and refined uv disparity mapping. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, 2022.

MA, X.; NAJJAR, W. A.; ROY-CHOWDHURY, A. K. Evaluation and acceleration of high-throughput fixed-point object detection on fpgas. *IEEE Transactions on Circuits and Systems for Video Technology*, v. 25, n. 6, p. 1051–1062, 2015.

MAGGIANI, L.; BOURRASSET, C.; QUINTON, J.-C.; BERRY, F.; SÉROT, J. Bio-inspired heterogeneous architecture for real-time pedestrian detection applications. *Journal of Real-Time Image Processing*, Springer, v. 14, n. 3, p. 535–548, 2018.

MAMMERI, A.; ZUO, T.; BOUKERCHE, A. Extending the detection range of vision-based driver assistance systems application to pedestrian protection system. In: IEEE. *2014 IEEE Global Communications Conference*. [S.l.], 2014. p. 1358–1363.

MANOR, E.; BEN-DAVID, A.; GREENBERG, S. Cordic hardware acceleration using dma-based isa extension. *Journal of Low Power Electronics and Applications*, MDPI, v. 12, n. 1, p. 4, 2022.

MICLEA, V.-C.; NEDEVSCHI, S. Real-time semantic segmentation-based stereo reconstruction. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, v. 21, n. 4, p. 1514–1524, 2019.

NGUYEN, D. T.; NGUYEN, T. N.; KIM, H.; LEE, H.-J. A high-throughput and power-efficient fpga implementation of yolo cnn for object detection. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, IEEE, v. 27, n. 8, p. 1861–1873, 2019.

NGUYEN, H.-T.; NGUYEN, X.-T.; PHAM, C.-K.; HOANG, T.-T.; LE, D.-H. A parallel pipeline cordic based on adaptive angle selection. In: IEEE. *2016 International Conference on Electronics, Information, and Communications (ICEIC)*. [S.l.], 2016. p. 1–4.

NGUYEN, U.; ROTTENSTEINER, F.; HEIPKE, C. Confidence-aware pedestrian tracking using a stereo camera. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 4 (2019), Nr. 2/W5*, Göttingen: Copernicus GmbH, v. 4, n. 2/W5, p. 53–60, 2019.

OLSON, C. G.; RUSSELL, R. P.; CARPENTER, J. R. Precomputing process noise covariance for onboard sequential filters. *Journal of Guidance, Control, and Dynamics*, American Institute of Aeronautics and Astronautics, v. 40, n. 8, p. 2062–2075, 2017.

Organização Mundial da Saúde. *Acidentes de Tráfego nas Estradas*. 2018. <https://www.who.int/violence_injury_prevention/road_safety_status/2018/en/>.

OTTO, C. *Fusion of data from heterogeneous sensors with distributed fields of view and situation evaluation for advanced driver assistance systems*. [S.l.]: KIT Scientific Publishing, 2013. v. 8.

PARK, D.; ZITNICK, C. L.; RAMANAN, D.; DOLLÁR, P. Exploring weak stabilization for motion feature extraction. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2013. p. 2882–2889.

Park, M.; Lee, S.; Kwon, C.; Kim, S. Design of pedestrian target selection with funnel map for pedestrian aeb system. *IEEE Transactions on Vehicular Technology*, v. 66, n. 5, p. 3597–3609, May 2017. ISSN 1939-9359.

PAYALAN, Y. F.; GUVENSAN, M. A. Towards next-generation vehicles featuring the vehicle intelligence. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, 2019.

PEDRONI, V. A. *Circuit design with VHDL*. [S.l.]: MIT press, 2020.

PREMEBIDA, C.; CARREIRA, J.; BATISTA, J.; NUNES, U. Pedestrian detection combining rgb and dense lidar data. In: IEEE. *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.], 2014. p. 4112–4117.

QUAN, L. *Image-Based Modeling*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2010. ISBN 1441966781, 9781441966780.

RASHED, H.; SALLAB, A. E.; YOGAMANI, S.; ELHELW, M. Motion and depth augmented semantic segmentation for autonomous navigation. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. [S.l.: s.n.], 2019.

REDMON, J. *Darknet: Open Source Neural Networks in C*. 2013–2016. <http://pjreddie.com/darknet/>.

REDMON, J.; FARHADI, A. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

REID, D. An algorithm for tracking multiple targets. *IEEE transactions on Automatic Control*, IEEE, v. 24, n. 6, p. 843–854, 1979.

REZATOFIGHI, S. H.; MILAN, A.; ZHANG, Z.; SHI, Q.; DICK, A.; REID, I. Joint probabilistic data association revisited. In: *Proceedings of the IEEE international conference on computer vision*. [S.l.: s.n.], 2015. p. 3047–3055.

RIBEIRO, M. I. Kalman and extended kalman filters: Concept, derivation and properties. *Institute for Systems and Robotics*, v. 43, p. 46, 2004.

ROLISON, J. J.; REGEV, S.; MOUTARI, S.; FEENEY, A. What are the factors that contribute to road accidents? an assessment of law enforcement views, ordinary drivers' opinions, and road accident records. *Accident Analysis & Prevention*, Elsevier, v. 115, p. 11–24, 2018.

RUDENKO, A.; PALMIERI, L.; HERMAN, M.; KITANI, K. M.; GAVRILA, D. M.; ARRAS, K. O. Human motion trajectory prediction: A survey. *CoRR*, abs/1905.06113, 2019. Disponível em: <http://arxiv.org/abs/1905.06113>.

RUSSELL, S. J.; NORVIG, P. *Artificial intelligence: a modern approach*. [S.l.]: Malaysia; Pearson Education Limited,, 2016.

SALOMON, D.; MOTTA, G. *Handbook of data compression*. [S.l.]: Springer, 2010.

SANDERS, A. *An introduction to Unreal engine 4*. [S.l.]: AK Peters/CRC Press, 2016.

SANGEETHA, D.; DEEPA, P. A low-cost and high-performance architecture for robust human detection using histogram of edge oriented gradients. *Microprocessors and Microsystems*, Elsevier, v. 53, p. 106–119, 2017.

SARKAR, S.; BHAIRANNAWAR, S. S.; KB, R. Fpgacam: A fpga based efficient camera interfacing architecture for real time video processing. *IET Circuits, Devices & Systems*, v. 15, n. 8, p. 814–829, 2021.

SÄRKKÄ, S. *Bayesian filtering and smoothing*. [S.l.]: Cambridge University Press, 2013. v. 3.

SCHNEIDER, N.; GAVRILA, D. M. Pedestrian path prediction with recursive bayesian filters: A comparative study. In: WEICKERT, J.; HEIN, M.; SCHIELE, B. (Ed.). *Pattern Recognition*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 174–183. ISBN 978-3-642-40602-7.

SCHRAMM, D.; HILLER, M.; BARDINI, R. Single track models. In: *Vehicle Dynamics*. [S.l.]: Springer, 2014. p. 223–253.

SMEULDERS, A. W.; CHU, D. M.; CUCCHIARA, R.; CALDERARA, S.; DEHGHAN, A.; SHAH, M. Visual tracking: An experimental survey. *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 36, n. 7, p. 1442–1468, 2013.

SOLOVYEV, R.; WANG, W.; GABRUSEVA, T. Weighted boxes fusion: Ensembling boxes from different object detection models. *Image and Vision Computing*, Elsevier, v. 107, p. 104117, 2021.

SRINIVAS, G. *Pedestrian detection in front of the ego vehicle using (stereo) camera in the urban scene: Deep versus Shallow learning approaches*. Tese (Doutorado) — Deutsches Zentrum für Luft-und Raumfahrt, 2016.

STANKIEWICZ, O.; LAFRUIT, G.; DOMAŃSKI, M. Multiview video: Acquisition, processing, compression, and virtual view rendering. In: *Academic Press Library in Signal Processing, Volume 6*. [S.l.]: Elsevier, 2018. p. 3–74.

STREUBEL, R.; YANG, B. Fusion of stereo camera and mimo-fmcw radar for pedestrian tracking in indoor environments. In: IEEE. *2016 19th International Conference on Information Fusion (Fusion)*. [S.l.], 2016. p. 565–572.

SULEIMAN, A.; SZE, V. An energy-efficient hardware implementation of hog-based object detection at 1080hd 60 fps with multi-scale support. *Journal of Signal Processing Systems*, Springer, v. 84, n. 3, p. 325–337, 2016.

SULEIMAN, A.; ZHANG, Z.; SZE, V. A 58.6 mw 30 frames/s real-time programmable multiobject detection accelerator with deformable parts models on full hd 1920 × 1080 videos. *IEEE Journal of Solid-State Circuits*, IEEE, v. 52, n. 3, p. 844–855, 2017.

SUTHERLAND, S.; DAVIDMANN, S.; FLAKE, P. *SystemVerilog for Design Second Edition: A Guide to Using SystemVerilog for Hardware Design and Modeling*. [S.l.]: Springer Science & Business Media, 2006.

SZELISKI, R. *Computer vision: algorithms and applications*. [S.l.]: Springer Nature, 2022.

TIBSHIRANI, R.; HASTIE, T.; WITTEN, D.; JAMES, G. *An introduction to statistical learning: With applications in R*. [S.l.]: Springer, 2021.

TRICHET, R.; BREMOND, F. Lbp channels for pedestrian detection. In: IEEE. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. [S.l.], 2018. p. 1066–1074.

UIJLINGS, J. R.; SANDE, K. E. V. D.; GEVERS, T.; SMEULDERS, A. W. Selective search for object recognition. *International journal of computer vision*, Springer, v. 104, n. 2, p. 154–171, 2013.

WANG, P. Research on comparison of lidar and camera in autonomous driving. In: IOP PUBLISHING. *Journal of Physics: Conference Series*. [S.l.], 2021. v. 2093, n. 1, p. 012032.

WANG, X.; ZHOU, B.; JI, J.; PU, B. Recognition and distance estimation of an irregular object in package sorting line based on monocular vision. *International Journal of Advanced Robotic Systems*, SAGE Publications Sage UK: London, England, v. 16, n. 1, p. 1729881419827215, 2019.

WANG, Z.; ZHENG, L.; LIU, Y.; LI, Y.; WANG, S. Towards real-time multi-object tracking. In: SPRINGER. *European Conference on Computer Vision*. [S.l.], 2020. p. 107–122.

WOJKE, N.; BEWLEY, A.; PAULUS, D. Simple online and realtime tracking with a deep association metric. In: IEEE. *2017 IEEE international conference on image processing (ICIP)*. [S.l.], 2017. p. 3645–3649.

WU, M. *Vision based Scene Understanding for Collision Avoidance on Roadway*. 2016.

WU, M.; ZHOU, C.; SRIKANTHAN, T. Robust and low complexity obstacle detection and tracking. In: IEEE. *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. [S.l.], 2016. p. 1249–1254.

XIA, L.; CHUNG, T. D.; KASSIM, K. A. B. A. A review of automated emergency braking system and the trending for future vehicles. In: *Proceedings of the Southeast Asia Safer Mobility Symposium*. [S.l.: s.n.], 2013.

XU, K. Silicon electro-optic micro-modulator fabricated in standard cmos technology as components for all silicon monolithic integrated optoelectronic systems. *Journal of Micromechanics and Microengineering*, IOP Publishing, v. 31, n. 5, p. 054001, 2021.

YANG, R.; WANG, Y.; XU, Y.; QIU, L.; LI, Q. Pedestrian detection under parallel feature fusion based on choquet integral. *Symmetry*, Multidisciplinary Digital Publishing Institute, v. 13, n. 2, p. 250, 2021.

YANNIS, G.; PAPADIMITRIOU, E.; THEOFILATOS, A. Pedestrian gap acceptance for mid-block street crossing. *Transportation planning and technology*, Taylor & Francis, v. 36, n. 5, p. 450–462, 2013.

YOON, J. H.; YANG, M.-H.; LIM, J.; YOON, K.-J. Bayesian multi-object tracking using motion context from multiple objects. In: IEEE. *2015 IEEE Winter Conference on Applications of Computer Vision*. [S.l.], 2015. p. 33–40.

ZENG, K.; MA, Q.; WU, J. W.; CHEN, Z.; SHEN, T.; YAN, C. Fpga-based accelerator for object detection: A comprehensive survey. *The Journal of Supercomputing*, Springer, p. 1–41, 2022.

ZHANG, J.; LIN, L.; ZHU, J.; LI, Y.; CHEN, Y.-c.; HU, Y.; HOI, S. C. Attribute-aware pedestrian detection in a crowd. *IEEE Transactions on Multimedia*, IEEE, v. 23, p. 3085–3097, 2020.

ZHANG, R.; WANG, W.; ZENG, L.; CHEN, J. A real-time obstacle detection algorithm for the visually impaired using binocular camera. In: SPRINGER. *International Conference in Communications, Signal Processing, and Systems*. [S.l.], 2017. p. 1412–1419.

ZHANG, S.; BENENSON, R.; OMRAN, M.; HOSANG, J.; SCHIELE, B. How far are we from solving pedestrian detection? In: *Proceedings of the iEEE conference on computer vision and pattern recognition*. [S.l.: s.n.], 2016. p. 1259–1267.

ZHAO, Z.-Q.; ZHENG, P.; XU, S.-t.; WU, X. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, IEEE, v. 30, n. 11, p. 3212–3232, 2019.

ZITNICK, C. L.; DOLLÁR, P. Edge boxes: Locating object proposals from edges. In: SPRINGER. *European conference on computer vision*. [S.l.], 2014. p. 391–405.

APÊNDICE A – FILTRO DE KALMAN

O filtro Kalman (KALMAN, 1960) calcula recursivamente os valores de uma variável de estado variante no tempo com variância de erro mínimo na presença de dados ruidosos em um sistema de tempo discreto. Ele foi projetado para variáveis de estado que são distribuídas por gaussianas. Mas detalhadamente, o filtro de Kalman assume que o estado de um sistema no momento k evoluiu do estado anterior no tempo $k-1$ de acordo com modelo de sistema linear de tempo discreto:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k + \boldsymbol{\omega}_k \tag{A.1}$$

onde:

- $\mathbf{x}_k$ é o vetor de estado que contém os termos de interesse do sistema (por exemplo, posição, velocidade, direção do pedestre) no momento k.
- $\mathbf{u}_k$ é o vetor que contém todas as entradas de controle (ângulo de direção, configuração do acelerador, força de frenagem do carro)
- $\mathbf{A}$ é a matriz de transição de estado que aplica o efeito de cada parâmetro de estado do sistema no tempo $k-1$ no estado do sistema no tempo k (por exemplo, a posição e a velocidade no tempo $k-1$, que afetam a posição no tempo k)
- $\mathbf{B}$ é a matriz de entrada de controle que aplica o efeito de cada parâmetro de entrada de controle no vetor $\mathbf{u}_k$ no vetor de estado (por exemplo, aplica o efeito da configuração do acelerador na velocidade e posição do sistema).
- $\boldsymbol{\omega}$ é o vetor que contém os termos de ruído do processo para cada parâmetro no vetor de estado. Presume-se que o ruído do processo seja obtido de uma distribuição normal multivariada de média zero com covariância dada pela matriz de covariância $\mathbf{Q}$, ou seja, $\boldsymbol{\omega} \sim \mathcal{N}(0, \mathbf{Q})$.

As medições são relacionadas ao vetor de estado pelo modelo de medição expresso através da seguinte equação linear:

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \boldsymbol{\nu}_k \tag{A.2}$$

onde

- $\mathbf{z}_k$ é o vetor de medições

- $\mathbf{H}$ é a matriz de transformação que mapeia os parâmetros do vetor de estado no domínio de medição

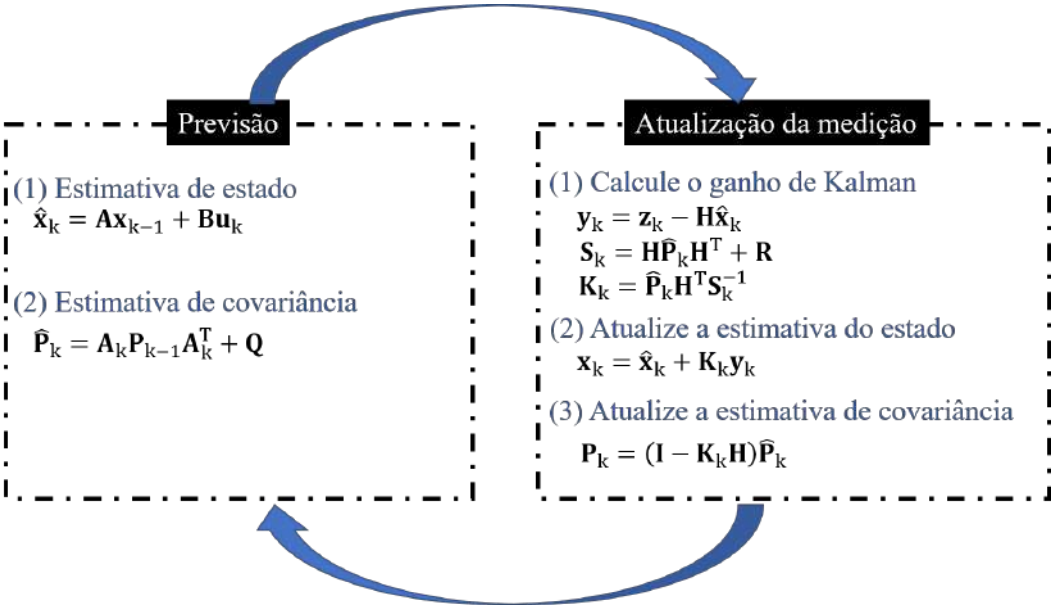
- $\boldsymbol{\nu}$ é o vetor que contém os termos do ruído de medição para cada observação no vetor de medição. Assim como o ruído do processo, o ruído de medição é assumido como sendo ruído branco gaussiano de média zero com covariância $\mathbf{R}$, ou seja, $\boldsymbol{\nu} \sim \mathcal{N}(0, \mathbf{R})$.

O vetor de estado estimado $\hat{\mathbf{x}}_k$ se desvia do vetor de estado real e desconhecido $\mathbf{x}_k$. Esse erro é modelado como ruído gaussiano branco com matriz de covariância $\mathbf{P}_k$, ou seja, $(\hat{\mathbf{x}}_k - \mathbf{x}_k) \sim \mathcal{N}(0, \mathbf{P}_k)$. Os termos ao longo da diagonal principal de $\mathbf{P}_k$ são as variâncias associados aos termos correspondentes no vetor de estado. Os termos fora da diagonal de $\mathbf{P}_k$ fornecem as covariâncias entre os termos no vetor de estado.

A Equação A.1, chamada de modelo de processo, diz que cada $\mathbf{x}_k$ (nossos valores de sinal) pode ser avaliado usando uma equação estocástica linear (de primeira ordem). Qualquer $\mathbf{x}_k$ é uma combinação linear de seu valor anterior mais um sinal de controle $\mathbf{u}_k$ e um ruído de processo. A Equação A.2, chamada de modelo de medição, diz que qualquer valor de medição (que não temos certeza de sua precisão) é uma combinação linear do valor do sinal e do ruído de medição. As funções lineares permitem uma representação simplificada do modelo de processo e do modelo de medição usando multiplicações de matrizes

O algoritmo de filtro de Kalman envolve uma iteração de dois estágios: previsão e atualização de medição como mostrado na Figura 94.

Figura 94 – Etapas de processamento do filtro de Kalman



Fonte: O autor.

A.1 ETAPA DE PREVISÃO

Dado um vetor de estado estimado $\mathbf{x}_{k-1}$ e uma matriz de covariância de estado $\mathbf{P}_{k-1}$, o estado no próximo passo é previsto a partir do modelo de sistema descrito pela Equação A.1 como:

$$\hat{\mathbf{x}}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k \tag{A.3}$$

e a matriz de covariância do estado previsto é calculada como:

$$\hat{\mathbf{P}}_k = \mathbf{A}_k\mathbf{P}_{k-1}\mathbf{A}_k^\top + \mathbf{Q} \tag{A.4}$$

A.2 ETAPA DE ATUALIZAÇÃO DA MEDIÇÃO

A medição prevista do vetor de estado previsto $\hat{\mathbf{x}}_k$ é derivado a partir do modelo de medição descrito pela equação A.2 como:

$$\hat{\mathbf{z}}_k = \mathbf{H}_k\hat{\mathbf{x}}_k \tag{A.5}$$

e sua matriz de covariância associada é dado como:

$$\hat{\mathbf{R}}_k = \mathbf{H}_k\hat{\mathbf{P}}_k\mathbf{H}_k^\top \tag{A.6}$$

O vetor de inovação descrito pela Equação:

$$\mathbf{s}_k = \mathbf{z}_k - \hat{\mathbf{z}}_k \tag{A.7}$$

descreve a diferença entre a medição real e medição prevista. Sua matriz de covariância é:

$$\mathbf{S}_k = \mathbf{R} + \hat{\mathbf{R}}_k \tag{A.8}$$

e é chamada de matriz de covariância da inovação. O novo estado é derivado a partir da atualização do estado previsto com uma versão transformada e ponderada do vetor de inovação:

$$\mathbf{x}_k = \hat{\mathbf{x}}_k + \mathbf{K}_k\mathbf{s}_k \tag{A.9}$$

e simplificado a partir das Equações A.7 e A.5 para:

$$\mathbf{x}_k = \hat{\mathbf{x}}_k + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k)) \tag{A.10}$$

O termo $\mathbf{K}_k$ é chamado de ganho de Kalman. O ganho ótimo de Kalman é ideal no sentido de minimizar a magnitude quadrática esperada do erro de estado $\hat{\mathbf{x}}_k - \mathbf{x}_k$, que é equivalente a minimizar a matriz de covariância estimada posterior $\mathbf{P}_k$. Através de alguns procedimentos matemáticos, o ganho ótimo de Kalman é obtido como:

$$\mathbf{K}_k = \hat{\mathbf{P}}_k \mathbf{H}_k^\top \mathbf{S}_k^{-1} \tag{A.11}$$

e simplificado a partir das Equações A.8 e A.6 para:

$$\mathbf{K}_k = \hat{\mathbf{P}}_k \mathbf{H}_k^\top (\mathbf{H}_k \hat{\mathbf{P}}_k \mathbf{H}_k^\top + \mathbf{R}_k)^{-1} \tag{A.12}$$

Com o ganho de Kalman ótimo, a matriz de covariância estimada a posteriori é simplificada para:

$$\mathbf{P}_k = (\mathbf{I}_k - \mathbf{K}_k \mathbf{H}_k)\hat{\mathbf{P}}_k \tag{A.13}$$

Em resumo, a etapa de atualização de medição envolve o calculo das equações A.12, A.10 e A.13. Caso o leitor queira entender melhor a matemática que fundamenta as equações de Kalman, poderá buscar o livro de Särkkä (2013) e o trabalho de Kalman (1960).

APÊNDICE B – HOG

Nesta etapa, são calculadas as características necessárias para a classificação. A técnica HOG compreende três etapas principais: gradiente, célula e bloco.

B.1 GRADIENTE

Esta etapa calcula a magnitude e o ângulo de orientação do gradiente de cada pixel da imagem de entrada. Usando a formulação proposta por Dalal e Triggs (2005), as componentes d_u e d_v do gradiente são primeiramente calculadas de acordo com a Equação B.1.

$$\begin{aligned} d_u(u,v) &= I_{\text{gray}}(u+1,v) - I_{\text{gray}}(u-1,v) \\ d_v(u,v) &= I_{\text{gray}}(u,v+1) - I_{\text{gray}}(u,v-1) \end{aligned} \tag{B.1}$$

A partir de d_u e d_v, a magnitude $m(u,v)$ e a orientação $\theta(u,v)$ são calculadas através da Equação B.2.

$$\begin{aligned} m(u,v) &= \sqrt{d_u(u,v)^2 + d_v(u,v)^2} \\ \theta(u,v) &= \arctan\frac{d_v(u,v)}{d_u(u,v)} \end{aligned} \tag{B.2}$$

A imagem de gradiente é obtida calculando os gradientes de todos os pixels usando as Equações B.1 e B.2, conforme mostrado na Figura 33.

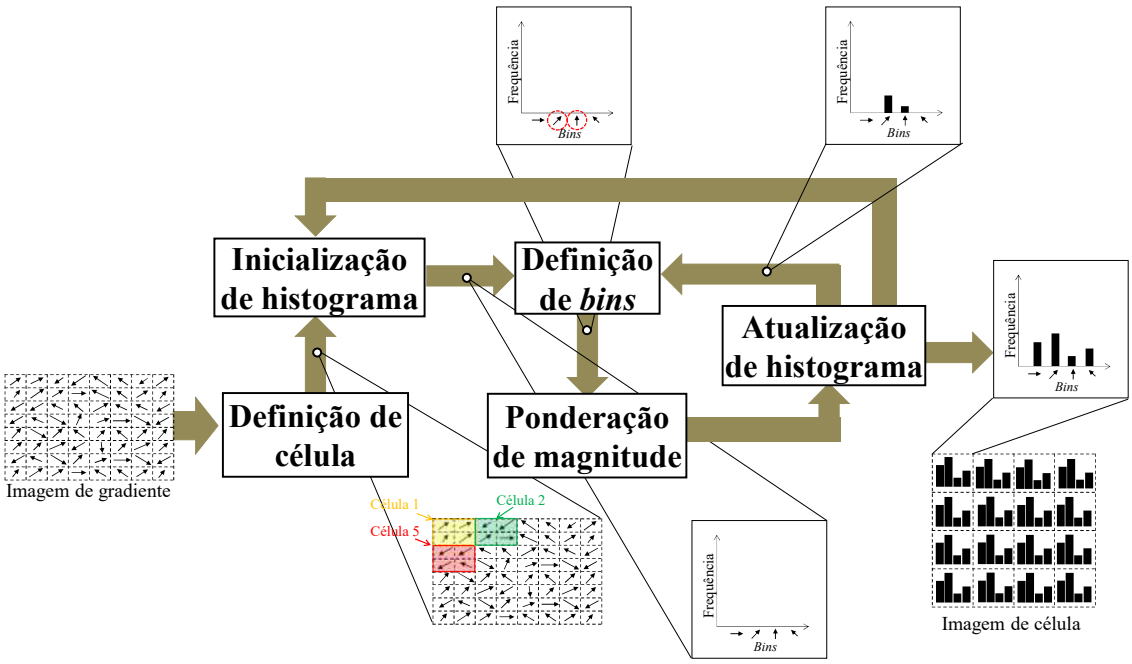
B.2 CÉLULA

A Figura 95 apresenta a estratégia de cálculo de célula. A primeira etapa define o grupo de gradientes que irão compor cada célula. A dimensão de cada célula, é definida como $W_{\text{cel}} \times H_{\text{cel}}$ pixels.

Em seguida, um histograma de orientação é inicializado para cada célula. A quantidade de *bins* do histograma é definida por Q_{bins}. As três etapas seguintes processam cada gradiente em sua respectiva célula para definição do histograma final. A etapa de definição de *bins* calcula os índices dos dois *bins* vizinhos.

Na etapa de ponderação de magnitude, é calculado o valor ponderado da magnitude do gradiente que será atribuído a cada *bin* selecionado. Por fim, a etapa de atualização de histograma realiza o acúmulo dos valores ponderados nos *bins* selecionados. Se todos os gradientes de uma determinada célula tiver sido processado, então o histograma final será disponibilizado e o fluxo de processamento volta para a etapa de inicialização para começar a calcular a próxima célula.

Figura 95 – Etapas de processamento da célula HOG



Fonte: O autor.

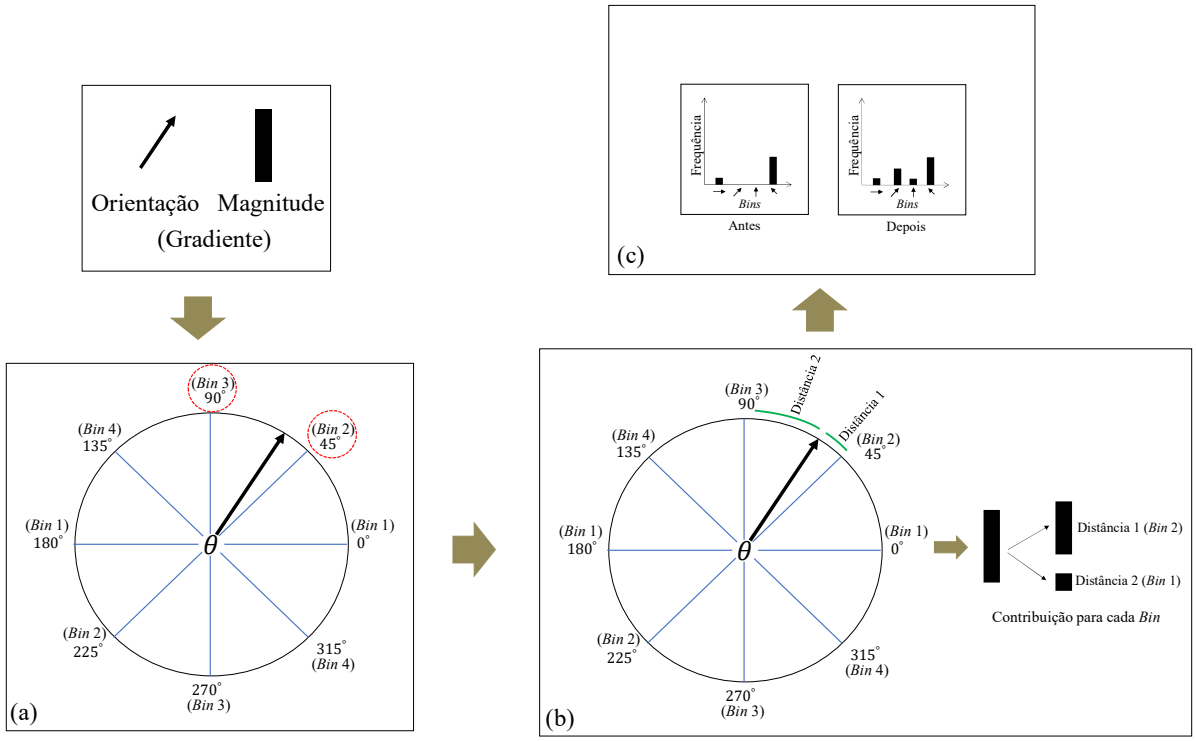
A Figura 96 detalha o cálculo do histograma a partir de um gradiente. A magnitude e o ângulo do gradiente dentro dos limites de uma determinada célula definem quais *bins* vizinhos serão acumulados dentro desta célula e quanto serão acumulados. Os *bins* representam ângulos de direção bem espaçados no intervalo [0°,180°]. Os valores dos dois *bins* vizinhos escolhidos são acumulados através de valores de magnitude ponderados com base na distância em ângulo do gradiente para cada *bin*. Quanto maior a distância menor é o peso. Esse tipo de ponderação é chamada de interpolação linear (DALAL; TRIGGS, 2005).

As equações para cálculo de célula são detalhadas a seguir. A magnitude e o ângulo de cada gradiente dentro dos limites de uma determinada célula definem quais *bins* serão acumulados dentro desta célula e quanto serão acumulados. As orientações não têm sinal, o que significa que os ângulos em [180°, 360°] são os mesmos que [0°, 180°]. Em termos matemáticos, o ângulo de orientação sem sinal θ_u de um determinado gradiente de localização (u, v) é dado por:

$$\theta_u(u,v) = \begin{cases} \theta(u,v), & \text{if } 0° \leq \theta(u,v) < 180° \\ \theta(u,v) + \pi, & 180° \leq \theta(u,v) < 360° \end{cases} \tag{B.3}$$

Os valores dos dois *bins* vizinhos são acumulados através de valores de magnitude ponderados com base na diferença entre o ângulo θ_u e o *bin* principal. Para obter esses valores de magnitude ponderada, primeiro o índice $b_{idx,1}$ é calculado através da Equação

Figura 96 – Detalhamento do cálculo do histograma a partir de um gradiente: (a) etapa de definição de *bins*, (b) etapa de ponderação de magnitude e (c) etapa de atualização do histograma.



Fonte: O autor.

B.4 que localiza o *bin* principal.

$$b_{idx,1}(\theta_u) = \lfloor \theta_u \frac{Q_{bins}}{\pi} \rfloor \tag{B.4}$$

O cálculo da diferença entre o ângulo θ_u e o *bin* principal é dado pela seguinte Equação:

$$\alpha(\theta_u) = (\theta_u - b_{\theta_u}(b_{\text{idx},1}(\theta_u))) \frac{Q_{bins}}{\pi}, \tag{B.5}$$

onde $b_{\theta_u}(\text{i})$ é uma função descrita pela Equação B.6 que define o ângulo de um determinado *bin* de índice $i \in [0, Q_{bins} - 1]$.

$$b_{\theta_u}(i) = i \frac{\pi}{Q_{\text{bins}}} \tag{B.6}$$

A partir de $\alpha(\theta_u)$, as porções da magnitude de gradiente que irão para o *bin* principal $b_{m,1}$ e o próximo *bin* vizinho $b_{m,2}$ são calculados através das Equações B.7 e B.8, respectivamente.

$$b_{m,1}(\theta_u, m) = (1 - \alpha(\theta_u))\, m \tag{B.7}$$

$$b_{m,2}(\theta_u, m) = \alpha(\theta_u)\, m \tag{B.8}$$

É importante destacar que o índice do *bin* vizinho $b_{\text{idx},2}(\theta_u)$ é calculado como:

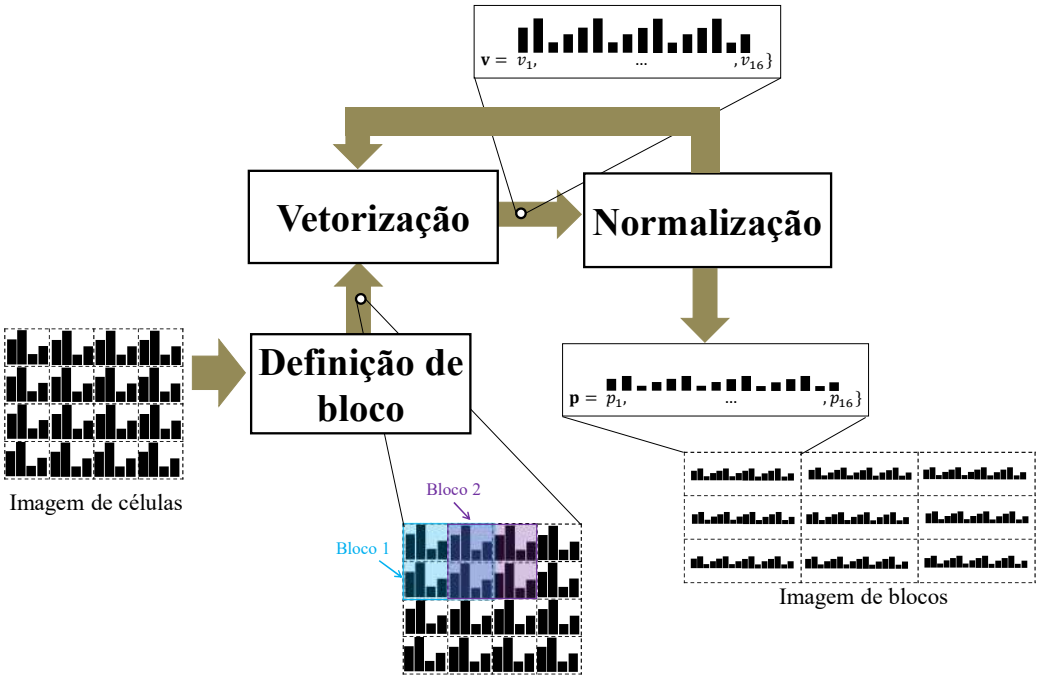
$$b_{\text{idx},2}(\theta) = \begin{cases} b_{\text{idx},1}(\theta_u) + 1, & \text{if } b_{\text{idx},1}(\theta_u) < Q_{\text{bins}} \\ 0, & \text{caso contrário} \end{cases} \tag{B.9}$$

A Equação B.9 garante a rotação do ângulo vizinho, ou seja, se o *bin* principal escolhido for o último do histograma, seu vizinho será o primeiro *bin* do histograma.

B.3 BLOCO

Os histogramas de células são normalizados para aumentar a imunidade às variações de iluminação, consequentemente, reduzindo falsos positivos da detecção. A Figura 97 apresenta as etapas do cálculo de bloco.

Figura 97 – Etapas de processamento de bloco HOG



Fonte: O autor.

A primeira etapa agrupa os histogramas de célula em blocos. A dimensão do bloco é dada por $W_{\text{bloco}} \times H_{\text{bloco}}$ pixels. Em seguida, cada bloco é vetorizado. Por fim, o vetor é normalizado. Para a normalização nós usamos a norma euclidiana, chamada de norma L2, cujo detalhamento matemático se encontra no Apêndice B. Como os blocos são compostos de células, a dimensão do bloco em pixels deve ser um múltiplo da dimensão da célula. Portanto, a dimensão do bloco também pode ser expressa como unidades de célula. Todos os vetores de bloco normalizados formam as características HOG.

As equações para cálculo de bloco são detalhadas a seguir. A função de normalização é a norma euclidiana, chamada de norma L2, definida como:

$$\mathbf{p} = \frac{\mathbf{v}}{\sqrt{\|\mathbf{v}\|^2 + \epsilon_{L2}}}, \tag{B.10}$$

onde $\mathbf{p} = \{p_1, ..., p_q, ..., p_s\}$ é o vetor de bloco unidimensional normalizado, $\mathbf{v} = \{v_1, ..., v_q, ..., v_s\}$ é o vetor unidimensional não normalizado formado pela concatenação

dos *bins* das células dentro do bloco, conforme mostrado na Figura 97, e $\epsilon_{L2} = 0,01$ é uma pequena constante para evitar divisões por zero.

APÊNDICE C – SVM

Esta etapa executa a varredura na imagem de características HOG usando a técnica de janelas deslizantes. Em cada janela de detecção, é verificada a existência ou não de um pedestre. As janelas de detecção do classificador têm dimensões $W_{\text{SVM}} \times H_{\text{SVM}}$ pixels. O parâmetro de salto (em inglês, *stride*), representado por $S_{\text{SVM}} = (S_{\text{SVM},u}, S_{\text{SVM},v})$, define o espaço em pixels entre uma janela de detecção e a próxima nas direções u e v, respectivamente. Este parâmetro deve ser um múltiplo da dimensão da célula HOG.

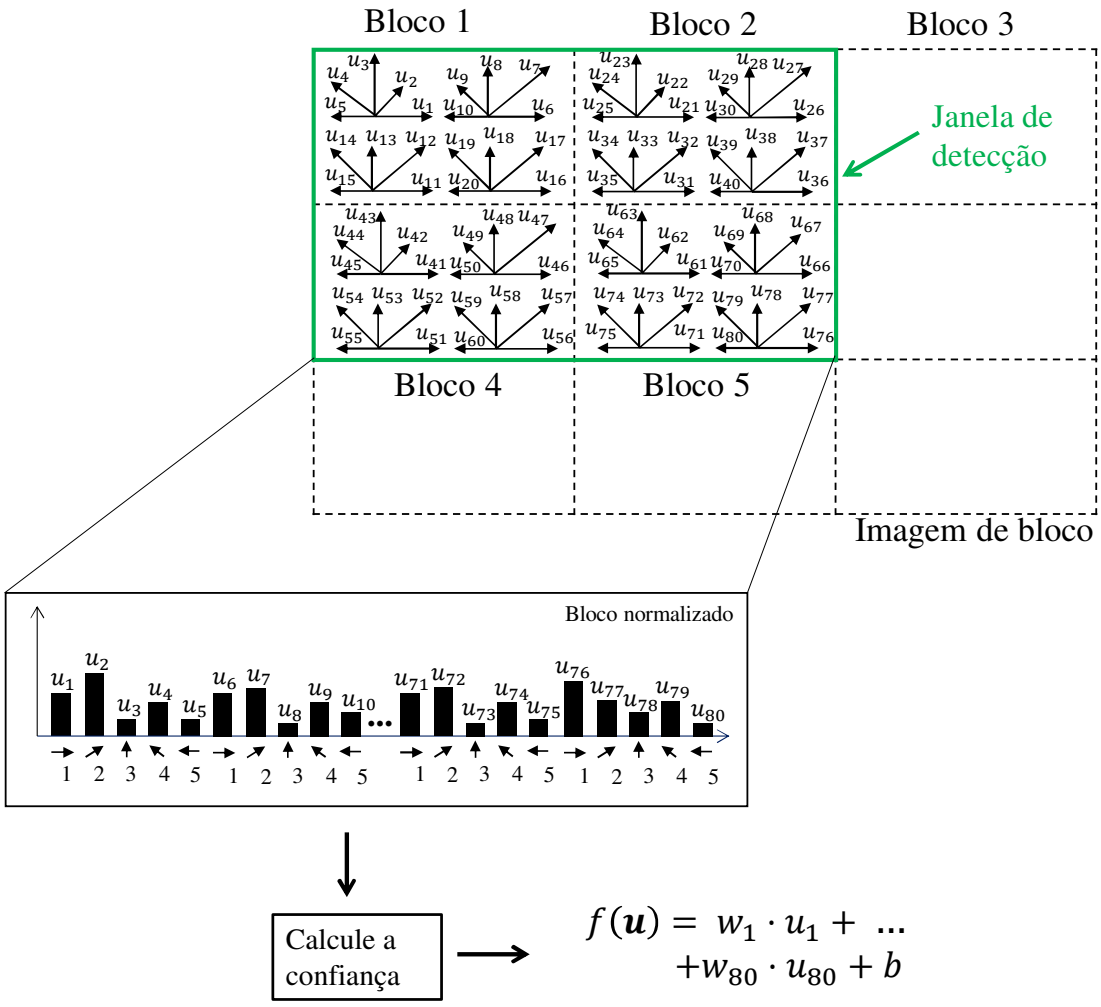
Em cada janela de detecção, os descritores são formados pela concatenação das características HOG em um único vetor unidimensional. Em seguida, a existência de um pedestre é verificada através do classificador SVM linear que compara esses descritores com um modelo de referência produzido pela fase de aprendizagem supervisionado. Uma vez que é um classificador binário, o SVM linear retornará o resultado se a janela contém ou não um pedestre. É crucial garantir que a ordem de composição das características do HOG no vetor unidimensional na fase de teste seja a mesma da fase de treinamento do SVM. Caso contrário, a classificação não funcionará corretamente.

Em termos matemáticos, uma janela de detecção é indicada por $\mathbf{u} = \{u_1, ..., u_r, ..., u_d\}$, onde cada u_r é uma característica HOG, ou seja, um *bin*. A classificação desta janela é realizada pela função SVM linear descrita pela Equação C.1.

$$f(\mathbf{u}) = \mathbf{w}^T\mathbf{u} + b = \sum_{i=1}^{d} w_i u_i + b \tag{C.1}$$

Os parâmetros $\mathbf{w}$ e b são, respectivamente, o peso e o *bias* do SVM linear, definidos através da fase de aprendizagem supervisionada. O resultado $f(\mathbf{u})$ indica um valor de distância da entrada $\mathbf{u}$ até o hiperplano que separa as duas classes. Quanto maior esse valor, maior é a certeza em informar que a janela de detecção contém um pedestre. A Figura 98 mostra um exemplo de uso da função SVM linear com uma janela de detecção sobre a imagem de bloco.

Figura 98 – Exemplo da aplicação da função SVM em uma janela de detecção. Cada característica u corresponde à frequência de seu respectivo *bin* no histograma de bloco



Fonte: O autor.

A função $g(\mathbf{u}, \sigma_{\text{SVM}})$, descrita pela Equação C.2 é usada para definir a classe de resposta do classificador.

$$g(\mathbf{u}, \sigma_{\text{SVM}}) = \begin{cases} 1, & \text{if } f(\mathbf{u}) > \sigma_{\text{SVM}} \\ -1, & \text{otherwise} \end{cases} \tag{C.2}$$

A constante σ_{SVM} define a pontuação de confiança. Para cada janela de detecção na qual o detector declarou que contém um pedestre, seu *bounding box* é retornado.

APÊNDICE D – REDIMENSIONAMENTO

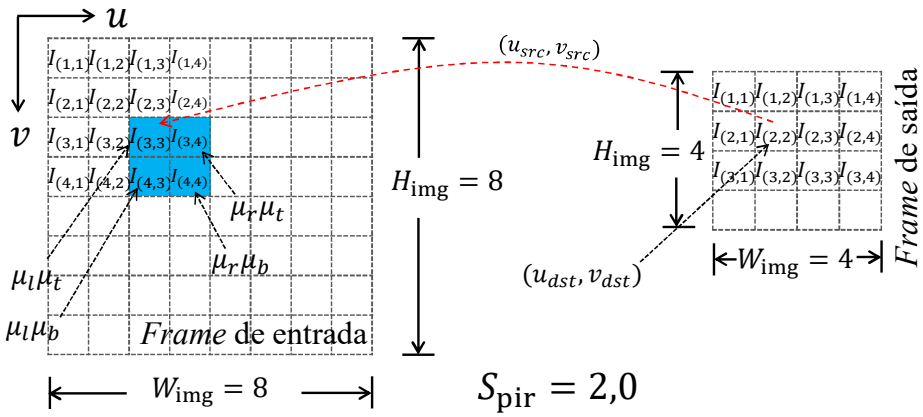
Esta etapa redimensiona a imagem de origem para uma dimensão de destino menor usando a técnica de interpolação bilinear (GRIBBON; BAILEY, 2004). Esta técnica calcula a intensidade em escala de cinza I_{dst} de cada pixel na imagem redimensionada através da soma ponderada da intensidade em escala de cinza dos quatro pixels mais próximos em torno de uma posição mapeada $(u_{\text{src}}, v_{\text{src}})$ na imagem de origem. Esta posição é calculada usando a Equação D.1.

$$\begin{aligned} u_{\text{src}} &= ((u_{\text{dst}} + 0.5)S_{\text{pir},u} - 0.5) \\ v_{\text{src}} &= ((v_{\text{dst}} + 0.5)\text{S}_{\text{pir},v} - 0.5) \end{aligned} \tag{D.1}$$

As constantes $S_{\text{pir},u}$ e $S_{\text{pir},v}$ são as razões da largura de entrada pela largura de saída e altura de entrada pela altura de saída. Essas constantes são iguais a S_{pir}, o que significa que a razão de redução nas duas dimensões é a mesma. Os termos src e dst significam, respectivamente, fonte e destino. A Figura 99 ilustra as variáveis envolvidas no cálculo de um dado pixel no *frame* redimensionado.

A partir da posição $(u_{\text{src}}, v_{\text{src}})$ os valores dos quatro pesos dos pixels vizinhos são calculados: peso esquerdo, peso direito, peso superior e peso inferior. Eles são descritos, respectivamente, pela Equação D.2 como μ_l (esquerdo), μ_r (direito), μ_t (superior) e μ_b (inferior).

Figura 99 – Cálculo de um novo pixel no *frame* redimensionado.



Fonte: O autor.

$$\begin{aligned} \mu_r &= (u_{\text{src}} - \lfloor u_{\text{src}} \rfloor) \\ \mu_l &= 1.0 - \mu_r \\ \mu_b &= (v_{\text{src}} - \lfloor v_{\text{src}} \rfloor) \\ \mu_t &= 1.0 - \mu_b \end{aligned} \tag{D.2}$$

Finalmente, a intensidade $I_{\text{dst}}(u_{\text{dst}}, v_{\text{dst}})$ do pixel alvo localizado em $(u_{\text{dst}}, v_{\text{dst}})$ é obtida a partir da Equação D.3.

$$\begin{aligned} I_{\text{dst}}(u_{\text{dst}}, v_{\text{dst}}) = {} & I_{\text{src}}(\lfloor u_{\text{src}} \rfloor, \lfloor v_{\text{src}} \rfloor)\ \mu_l\ \mu_t + \\ & I_{\text{src}}(\lfloor u_{\text{src}} \rfloor + 1, \lfloor v_{\text{src}} \rfloor)\ \mu_r\ \mu_t + \\ & I_{\text{src}}(\lfloor u_{\text{src}} \rfloor, \lfloor v_{\text{src}} \rfloor + 1)\ \mu_l\ \mu_b + \\ & I_{\text{src}}(\lfloor u_{\text{src}} \rfloor + 1, \lfloor v_{\text{src}} \rfloor + 1)\ \mu_r\ \mu_b \end{aligned} \tag{D.3}$$

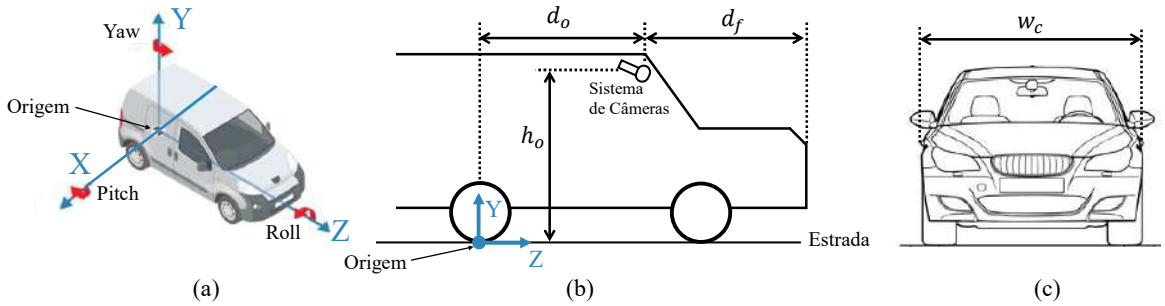
O *frame* redimensionado é obtido calculando todos os pixels dentro da nova dimensão, usando as Equações D.1–D.3.

APÊNDICE E – SISTEMA DE COORDENADAS

Quando existem vários sensores envolvidos que estão em posições e orientações de montagem diferentes, é necessário que os dados sejam transformados dos sistemas de coordenadas dos sensores para um sistema de coordenadas comum do veículo. Uma vez que se define a localização do sistema de coordenadas comum, todas as informações de distância e orientação de cada sensor para este eixo precisam ser definidos.

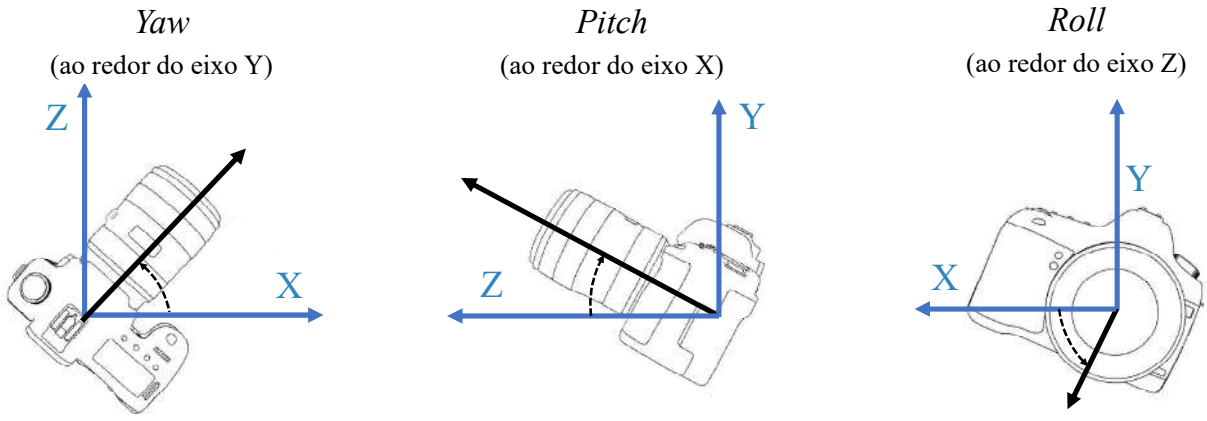
Para melhorar o entendimento, considere a origem do sistema de coordenadas comum do veículo como sendo no meio e abaixo do eixo traseiro, sobre o solo como mostrado na Figura 100. Assim, o eixo z aponta longitudinalmente para a frente do veículo, o eixo x constrói o componente lateral perpendicular apontando para a direita, enquanto o eixo y aponta para o topo.

Figura 100 – Ilustração de um sistema de coordenadas: (a) Visão de cima, (b) Visão lateral e (c) Visão de frente do veículo. Os parâmetros d_o, d_f, h_o e w_c significam, respectivamente, distância longitudinal para o eixo comum, distância longitudinal para a frente do veículo, altura para o eixo comum e largura do veículo.



Fonte: O autor.

Figura 101 – Ilustração dos ângulos *yaw*, *pitch* e *roll*.



Fonte: O autor.

Os ângulos *yaw*, *pitch* e *roll* definem a relação, em termos de rotação nas três di-

mensões, entre o sistema de coordenadas comum e o sistema de coordenadas do objeto. Este objeto pode ser o veículo, a câmera, ou outros sensores. Essa relação é mostrada na Figura 101. Esses ângulos são parâmetros necessários para o processo de transformação de coordenadas da câmera para o sistema de coordenadas comum.

Os pixels no sistema de coordenada de imagem precisam ser transformados no sistema de coordenada de câmera para depois ser transformados no sistema de coordenada do veículo. A primeira transformação, chamada de perspectiva (HARTLEY; ZISSERMAN, 2003), entre o ponto $p_c = (x, y, z)$ nas coordenadas de câmera e o pixel $p_i = (u, v, d)$ nas coordenadas de imagem é a seguinte:

$$\begin{pmatrix} u \\ v \\ d \end{pmatrix} = \begin{pmatrix} h_1(p_c) \\ h_2(p_c) \\ h_3(p_c) \end{pmatrix} = \begin{pmatrix} \frac{fx}{z} + u_0 \\ \frac{fy}{z} + v_0 \\ \frac{bf}{z} \end{pmatrix}, \tag{E.1}$$

onde os parâmetros b, f e (u_0, v_0) são, respectivamente, a distância entre os centros focais, a distância focal e o ponto principal do sistema de câmera estéreo.

A segunda transformação é realizada através de operações de rotação e a translação tridimensionais. Nesse tipo de transformação é recomendado que as coordenadas sejam representadas como homogêneas $\mathbf{x} = (x, y, z, 1)^T \in \mathbb{R}^4$. A transformação para coordenadas do veículo é uma multiplicação única das coordenadas homogêneas do sensor de câmera com a matriz de transformação $\mathbf{D}$ definida como:

$$\mathbf{D} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times4}, \tag{E.2}$$

onde

$$\mathbf{R} = \mathbf{R}_x(\alpha)\mathbf{R}_y(\beta)\mathbf{R}_z(\gamma), \tag{E.3}$$

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & \text{-}\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}, \tag{E.4}$$

$$\mathbf{R}_y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ \text{-}\sin(\beta) & 0 & \cos(\beta) \end{bmatrix}, \tag{E.5}$$

$$\mathbf{R}_z(\gamma) = \begin{bmatrix} \cos(\psi) & \text{-}\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{E.6}$$

$\mathbf{R} \in \mathbb{R}^{3\times3}$ é a matriz de rotação, que depende do ângulo de *pitch* α, do ângulo de *roll* β, do ângulo *yaw* ψ e $\mathbf{t} \in \mathbb{R}^{3\times1}$ é o vetor de translação.

$$\mathbf{X}_{\text{veículo}} = \mathbf{D}\, \mathbf{x}_{\text{câmera}} \tag{E.7}$$

Dessa forma, qualquer localização do pedestre $\mathbf{x}_{\text{câmera}}$ em coordenadas da câmera poderá ser transformada para $\mathbf{X}_{\text{veículo}}$ de acordo com a Equação E.7. Mais detalhes sobre transformações de espaço podem ser encontrados nos trabalhos de Otto (2013) e Keller (2014).