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

Pedro Jorge Américo Ishimaru

DESENVOLVIMENTO DE MÓDULO PARA SEGMENTAÇÃO DE ESPAÇO LIVRE EM IMAGENS ESTÉREO COM PROTOTIPAÇÃO EM FPGA

Recife
2021

Pedro Jorge Américo Ishimaru

DESENVOLVIMENTO DE MÓDULO PARA SEGMENTAÇÃO DE ESPAÇO LIVRE EM IMAGENS ESTÉREO COM PROTOTIPAÇÃO EM FPGA

Este trabalho foi apresentado à Pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Engenharia da Computação

Orientadora: Edna Natividade da Silva Barros

Recife

2021

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

I79d Ishimaru, Pedro Jorge Américo
Desenvolvimento de módulo para segmentação de espaço livre em imagens estéreo com prototipação em FPGA / Pedro Jorge Américo Ishimaru. – 2021.
98 f.: il., fig., tab.

Orientador: Edna Natividade da Silva Barros.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2021.
Inclui referências.

1. Engenharia da computação. 2. Sistemas embarcados. 3. Visão computacional estéreo. 4. Sistemas ciber-físicos. 5. Desenvolvimento em FPGA. I. Barros, Edna Natividade da Silva (orientador). II. Título

621.39 CDD (23. ed.) UFPE - CCEN 2022 – 25

Pedro Jorge Américo Ishimaru

"DESENVOLVIMENTO DE MÓDULO PARA SEGMENTAÇÃO DE ESPAÇO LIVRE EM IMAGENS ESTÉREO COM PROTOTIPAÇÃO EM FPGA"

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Aprovada em: 13/08/2021.

BANCA EXAMINADORA

Prof. Dr. Adriano Augusto de Moraes Sarmento
Centro de Informática / UFPE

Prof. Dr. Victor Wanderley Costa de Medeiros
Departamento de Estatística e Informática / UFRPE

Profa. Dra. Edna Natividade da Silva Barros
Centro de Informática / UFPE
Orientadora

Este trabalho é dedicado à memória de minhas avós, Miyoko Ishimaru e Maria do Carmo Aquino, e ao caminho que elas traçaram enquanto estiveram conosco.

AGRADECIMENTOS

As primeiras pessoas a quem devo os devidos agradecimentos são minha orientadora Edna Barros e a meu colega de pós graduação Lucas Cambuim, pelo incentivo, pelas contribuições incomensuráveis a este trabalho e, principalmente, por terem me instigado a continuar nos momentos de maior desânimo.

Agradeço a minha família. Aos meus pais Jorge Ishimaru e Ana Ishimaru, por serem uma inspiração sem igual e, por motivos incontáveis, terem possibilitado tudo que eu conquistei até hoje. À minha irmã Maria Eduarda Ishimaru, pelo amor, compreensão e conversas sempre enriquecedoras.

Agradeço à minha amada companheira, Lilian França, que além de ter sido a melhor pessoa do mundo para se ter ao lado ao longo dos últimos anos, dando muito amor, apoio e carinho no que estamos construindo, ajudou com a revisão do texto e tradução de figuras.

Agradeço à minha psicologa Fátima Melo, por ter me ajudado a desatar incontáveis nós e cuidar da saúde mental nesta aventura que é, não apenas viver, mas fazer pós-graduação no Brasil dos últimos anos.

Às amigas e amigos, que tiveram um papel fundamental durante o desenvolvimento deste trabalho. A Tarsila Lima, pelas conversas tarde da madrugada quando nos encontrávamos pela cozinha. A Ana Cristina Barreto, por ser um exemplo sem precedentes na minha vida acadêmica e ter ajudado com a construção de figuras, a Karl Souza, Elys Carvalho, Catarina Cataldi, Lorena Moraes, Julio e Lidiane Riedl, Nathalia Mota, Andrea Brandão e Sarah Oliveira.

A todas as pessoas que me acompanharam e acompanham durante minha jornada como trabalhador da pesquisa, desenvolvimento e inovação. No grupo da pós-graduação, no LINCS, no LaCETI e no ITEMM. A João Machado e Pyetro Ferreira por terem me dado a primeira oportunidade neste caminho. A Vanessa Ogg e Marvson Pontes por terem participado comigo do início de um novo laboratório na UFPE. A Marcos Malveira e João Fernandes por terem segurado as pontas e possibilitado que eu terminasse esse trabalho. Agradeço nominalmente também à Victor Sabino, Leon Pontes, Gabriel Freitas, Severino José, Flaviano Dias, Rogério Luiz, Lucas Amorim, Gustavo Charamba, Lucas Cavalcanti, Mirela Melo, Henrique Figueroa, Thiago Nogueira, Roshany Barreto e Rafael Nunes. Desde que comecei, todos os dias eu reaprendo o significado de camaradagem.

Agradeço a CAPES pelo fomento para realização deste trabalho.

Finalmente, gostaria de agradecer a todos os professores, estudantes, pesquisadores e funcionários da Universidade Federal de Pernambuco por possibilitar a realização de pesquisa, extensão e ensino público, gratuito e de qualidade. Os dias difíceis passarão e nós venceremos.

“eu me afastei e observei até que eu balançasse como uma onda entre a vida que eu sonhei e o sonho oscilante que vivi”. (ADONIS, 2000, p. 12).

RESUMO

A Segmentação de regiões livres e superfície do chão é uma questão de grande importância no contexto da robótica e de veículos terrestres autônomos. Este problema, relacionado à área de percepção de espaço, é amplamente discutido na literatura especializada, sendo abordado por meio de diferentes técnicas de processamento de imagem, como uso de Redes Neurais Artificiais, Métodos Estatísticos e Métodos Algébricos para processar vários possíveis tipos de sinais de entrada, como por exemplo, imagens monocromáticas e coloridas, som, Laser e imagens estéreo. Entretanto, os métodos propostos para realizar segmentação de espaço livre geralmente possuem altos custos computacionais, envolvendo operações de convolução e manipulações matriciais. Portanto, para obter melhores desempenhos, essas técnicas normalmente são implementadas em plataformas de maior poder computacional. Paradoxalmente, isto pode ser impeditivo para aplicações em robótica e veículos autônomos, uma vez que é comum que tecnologias embarcadas estejam sob rígidas restrições de capacidade computacional, memória, consumo de potência e custo. Neste contexto, este trabalho propõe uma arquitetura em FPGA para realizar segmentação da superfície do chão usando técnicas de visão computacional estéreo, isto é, usando mapas de disparidades gerados a partir de um par de imagens estéreo. Foi projetado uma arquitetura em FPGA baseada em um algorimo de referência usando técnicas algébricas que exploram informação tridimensional para detectar regiões de obstáculo, regiões livres e horizonte a partir da extração do perfil característico destes objetos. A validação do algoritmo de referência e do módulo proposto usou o *dataset* KITTI para detecção de superfície de estrada, que contém imagens do mundo real de cenários automotivos. Os resultados, comparados com outras técnicas da literatura, mostraram uma baixa perda de qualidade, considerando as métricas de precisão e sensitividade, com aumentos significativos, na ordem de 10 vezes, nas taxas de processamento em *frames* por segundo, consolidando o trabalho como um módulo promissor para aplicações envolvendo robótica e visão computacional estéreo.

Palavras-chaves: sistemas embarcados; segmentação de imagem; visão computacional estéreo; sistemas ciber-físicos; desenvolvimento em FPGA.

ABSTRACT

Free space and ground surface segmentation is an all-important matter regarding the design of robotics and self-driving vehicles. This problem, related to the field of space perception, is widely discussed in specialized literature with several different approaches, such as Artificial Neural Networks, graphs, and image processing techniques to deal with many types of possible input signals, for example, monochromatic and RGB pictures, sound, LASER, and stereo images. However, the proposed methods to perform free space segmentation usually have high computational costs, involving convolutions and matrix manipulation. Thus, in order to achieve acceptable performances, these techniques are typically implemented in more powerful platforms. Paradoxically, this could impede applications in robots and autonomous vehicles since these embedded technologies generally are under tight constraints of computational capabilities, memory, power consumption, and cost. Given this context, this work proposes an FPGA architecture to perform ground surface segmentation using stereo computer vision techniques, this is, using disparity maps generated from a pair of stereo pictures. The hardware design started from an algorithm that adapts to the implementation in FPGA algebraic techniques that exploit three-dimensional information to detect obstacle regions, free regions, and horizon detection in the image from the algebraic extraction of their profiles. The validation of both algorithm and proposed module used the KITTI dataset for road surface detection, which contains real-world pictures from automotive scenarios. Compared with other techniques, the results showed a low-performance loss regarding precision and recall metrics, with significant improvements in the Frames per Second processing rate, consolidating a promising module for applications involving robotics and stereo computer vision.

Keywords: embedded systems; image segmentation; stereo computational vision; cyber-physical systems; FPGA prototyping.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

A	Acurácia
AHDL	*Altera Hardware Description Language*
CIn	Centro de Informática
CRF	Conditional Random Field
DSP	*Digital Signal Processor*
F1	Medida F1
FN	False Negative
FP	False Positive
FPGA	*Field Programmable Gate Array*
FPS	Frames Por Segundo
GCC	GNU Compiler Collection
GPS	Global Positioning System
KITTI	*Karlsruher Institut für Technologie and Toyota Institute*
LiDAR	*Light Detection and Ranging*
LUT	*Look-up Tables*
MEF	Máquina de Estados Finita
mif	Memory Initialization File
P	Precisão
PM	Precisão Média
RNA	Rede Neural Artificial
S	Sensitividade
SER	Specify, Explore and Refine
t	Tempo
TN	True Negative
TP	True Positive
TPR	True Positive Rate
UFPE	Universidade Federal de Pernambuco
VHDL	*VHSIC Hardware Description Language*

LISTA DE SÍMBOLOS

A	Unidade de corrente Ampére
W	Unidade de potência Watt
Δ	Letra grega maiúscula Delta
$\geq$	Maior ou Igual
$\leq$	Menor ou Igual
$\sum$	Somatório
$\pm$	mais ou menos

SUMÁRIO

1 INTRODUÇÃO

Esta dissertação descreve o projeto, desenvolvimento e prototipação em hardware reconfigurável de um módulo para segmentação de espaço livre em imagens estéreo binoculares. Neste capítulo faz-se uma introdução ao tema abordado por meio da exposição, na seção 1.1, das motivações que justificaram este trabalho. Na seção 1.2 são expostos os objetivos gerais e específicos, a seção 1.3 descreve brevemente as contribuições trazidas e, finalmente, a seção 1.4 descreve como se dá a estruturação deste texto.

1.1 MOTIVAÇÃO

A habilidade da mobilidade espacial ativa, isto é, mover-se no espaço intencionalmente, é fortemente dependente da percepção espacial. Este conceito é descrito pela *Enciclopédia Britânica* da seguinte forma:

> A percepção espacial é o processo através do qual humanos e outros organismos se tornam cientes de da posição relativa de seus próprios corpos e dos objetos ao seu entorno. A percepção espacial provê indícios, tais como profundidade e distância, que são importantes para a movimentação e orientação no ambiente. (JäRVINEN, 2017).

Em organismos animais, a percepção se dá por meio do processamento das informações provenientes dos órgãos sensoriais principalmente visão, audição e tato. Para que a locomoção seja possível, os agentes devem levar em conta a posição e da dinâmica existente entre o seu corpo e o ambiente.

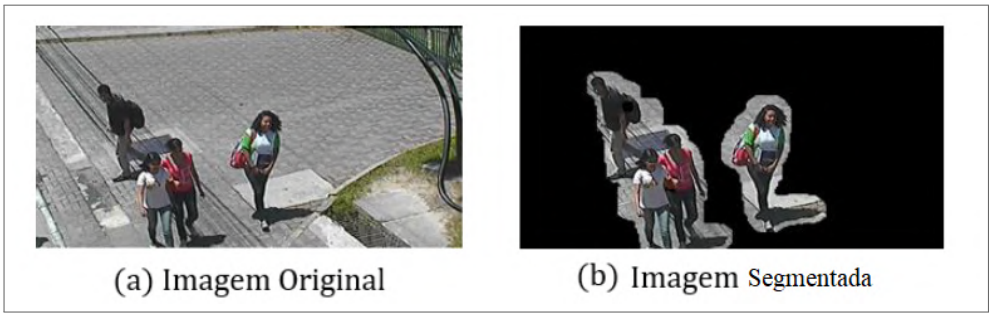
A mesma dependência existe para robôs autônomos móveis, que são agentes não-orgânicos. Na percepção espacial robótica, as informações são adquiridas por meio de sensores como por exemplo câmeras, sonares e escâneres a laser (KAGAN, 2020). O agente deve processar os dados de sensoriamento para calcular as informações espaciais necessárias para o movimento.

Em relação ao processamento visual, um dos passos que devem ser realizados é a identificação e separação de objetos. Para um robô terrestre com sensoriamento baseado em imagens se locomover de forma efetiva ele deve possuir a capacidade de identificar as regiões para qual é possível realizar a movimentação, ou seja, classificar, na imagem espaço livre e não-livre. Esta classificação pode ser realizada através da segmentação.

Segundo SHAPIRO (2001), segmentação é processo de particionar uma imagem digital em um ou múltiplos segmentos de modo a simplificar ou transformar a sua representação em algo mais significativo e fácil de analisar. Tal processo é feito por meio da classificação de pixeis de modo que pixeis da mesma classe compartilhem características de interesse. Uma das aplicações da segmentação de imagem é a detecção de regiões de interesse (BRINKMANN, 2008), ou seja a separação dos trechos da imagem que contém a informação útil para se

processar e analisar, diminuindo a quantidade de dados a serem operados. Na Figura 1 é possível observar um exemplo de segmentação de região de interesse.

Figura 1 – Resultado de trabalho desenvolvido por pesquisadores do CIn/UFPE para aceleração em hardware de segmentação de pedestres a partir de imagens provenientes de câmeras de segurança. Em (a) pode-se ver a imagem original e em (b) a imagem após a segmentação



Fonte: FERNANDES (2016).

Por reduzir a complexidade computacional da aplicação final, tal procedimento é muito importante para sistemas com limitações de capacidade de processamento, consumo energético ou que tenham restrições para responsividade em tempo real (VAHID; GIVARGIS, 2001), o que inclui os veículos auto-dirigíveis e robôs móveis autônomos (LIU et al., 2021; KAGAN, 2020).

Nas últimas décadas a robótica, campo multidisciplinar que envolve áreas da ciência da computação e engenharias (BETA, 2020), experimentou avanços significantes com grandes desenvolvimento em algoritmos, mecânica e plataformas de hardware (LIU et al., 2021). Na manufatura industrial, o uso de manipuladores robóticos se estabeleceu como um mercado que movimenta aproximadamente 2 bilhões de dólares anuais (SIEGWART ILLAH REZA NOURBAKHSH, 2011). Tais dispositivos conseguem mover grandes massas, com velocidades expressivas e grande precisão. Com essas características super-humanas, pode-se afirmar que robôs, de certa maneira, superaram o proposto pela sua primeira definição, do escritor tcheco Karel Capek na peça teatral R.U.R em 1921 (CAPEK CLAUDIA NOVACK-JONES, 2004), na qual o termo foi cunhado para indicar uma máquina capaz de realizar o trabalho humano como um humano. Uma definição mais moderna, que será utilizada por este trabalho é a proposta por Lumelsky (2006):

> Um robô é uma máquina automática ou semi-automática capaz de realizar movimentação proposital em resposta ao seu entorno em um ambiente não estruturado. (p. 15, tradução do autor).

A obra de Lumelsky (2006) busca ainda estabelecer distinções entre sistemas robóticos e sistemas computacionais tradicionais, segundo ele a capacidade de reação a dados sensoreados é uma condição *sine qua non* para um robô. A esta condição, Kagan (2020) adiciona as propriedades definidas por Brooks (1991):

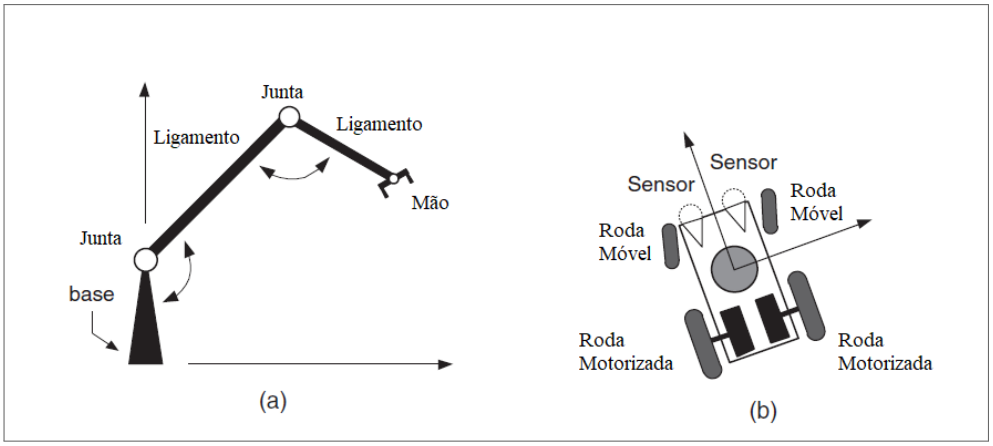
- Situabilidade. Os robôs estão situados no mundo, lidando com os conceitos de "aqu" e "agora" do ambiente que influencia diretamente o comportamento do sistema.
- Encorporamento. Os robôs têm um corpo físico e suas ações participam da dinâmica com o mundo.
- Emersão. As interações do agente robótico com o ambiente são inteligentes.

Por sua própria definição, portanto, os agentes robóticos são intrinsecamente ligados ao sensoriamento do ambiente. Sendo a segmentação um processo importante para todos aqueles, fixos ou móveis, que utilizam processamento de imagem como parte do seu conjunto de sensores.

A Figura 2 mostra exemplos de esquemas de robôs respectivamente fixos e móveis. Os robôs autônomos móveis, dos quais pode-se dizer que veículos auto-dirigíveis, são uma parte são a classe de maior interesse para este capítulo. Diferente dos robôs fixos, o sistema de coordenadas cartesianas locais é relativo ao agente estando a sua origem em movimento em relação as condições do ambiente, seus principais atuadores são os motores que possibilitam a sua movimentação a partir de dados captados por sensores que captam estímulos externos KAGAN.

Para construção de uma melhor representação do entorno do agente robótico, é importante a captação de informações de profundidade. Todavia câmeras monoculares só disponibilizam de dados bidimensionais. Para contornar este problema, pode-se usar recursos como *Light Detection and Ranging* (LiDAR), sonares, radares ou, os sensores mais importante para este trabalho, as câmeras estéreo.

Figura 2 – Exemplo de robôs autônomos fixos (a) e móveis (b)



Fonte: Kagan (2020).

A visão computacional estéreo é o recurso que utiliza duas ou mais câmeras para obter informações de profundidade. A ideia, de origem inspirada no funcionamento dos olhos dos animais mamíferos, é fundamentada no fato de que cada câmera capta a mesma imagem de

uma perspectiva diferente e a diferença entre o mesmo ponto captado pela câmera esquerda e pela direita, por triangulação, contêm a informação desejada. Este conceito é aprofundado na seção 2.2 do capítulo 2.

Um dos problemas relacionados a visão estéreo é a complexidade computacional associado ao cálculo do mapa de disparidades, que contêm a quantificação da diferença entre os pixeis do par conjugado formado pelas imagens esquerda e direita (KOK; RAJENDRAN, 2019). Os custos computacionais elevados impactam o desempenho do sensoriamento e pode inviabilizar o uso desta tecnologia em aplicações da robótica, visto que muitas vezes tais sistemas estão limitados por rígidas restrições em termos de tempo de resposta, capacidade computacional e eficiência energética (LIU et al., 2021).

Uma estratégia para solucionar este problema é a implementação de sistemas em FPGAs, hardware reconfigurável com capacidade de processamento massivo de dados. O grupo de pesquisa em sistemas embarcados do Centro de Informática (CIn) possui trabalhos envolvendo este tipo de tecnologia, em especial, pode-se citar a dissertação de mestrado de Cambuim (2017) na qual o autor implementou um módulo em FPGA para cálculo de mapas de disparidades usando a técnica *semi-global matching* (HIRSCHMULLER, 2005) obtendo excelente desempenho com taxas de processamento superiores a 100 Frames Por Segundo (FPS), para imagens full HD (768 x 1024), o que possibilita o uso do módulo para aplicações de tempo real.

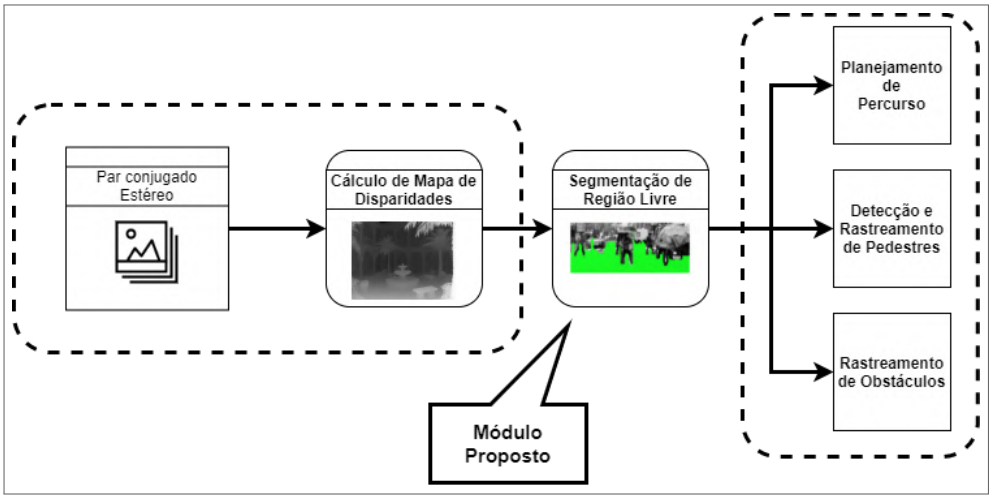
A partir do mapa de disparidades e da consequente informação de profundidade, é possível realizar processamentos para obter outras informações acerca do cenário sensoreado. Por meio de características geométricas do conjugado estéreo e do mapa de disparidades correspondente, pode-se realizar a segmentação da região livre (LABAYRADE; AUBERT; TAREL, 2002; ILOIE; GIOSAN; NEDEVSCHI, 2014; ZHANG et al., 2018b; WEDEL et al., 2008). Tal cálculo pode ser computacionalmente custoso, visto que envolve a computação de histogramas e a manipulação de matrizes.

Paralelamente, os grupos de pesquisa em desenvolvimento de sistemas embarcados e de robótica do Centro de Informática da Universidade Federal de Pernambuco (UFPE) desenvolvem trabalhos que utilizam de segmentação de região livre tanto para compor uma região de interesse a ser usada para uma busca mais refinada quanto para outros fins, como por exemplo a construção autônoma de mapas com *grids* de ocupação e algoritmos de planejamento de percurso (MELO, 2021).

Para estes trabalhos, que possuem restrições típicas do desenvolvimento de sistemas embarcados, a disponibilização de um módulo capaz de realizar a segmentação desejada, com qualidade aceitável, bom desempenho em termos de performance e eficiência energética seria de grande importância. A Figura 3 esquematiza como o módulo proposto por este trabalho contribui para as pesquisas em andamento do CIn. O módulo recebe como entrada os resultados do sistema proposto por Cambuim (2017), responsável pelo cálculo eficiente de mapas de disparidade, e provê uma solução eficiente para os trabalhos em andamento e futuros que

utilizam da segmentação de espaço livre.

Figura 3 – Diagrama de blocos mostrando o fluxo do par de imagem estéreo até a aplicação final. No meio, é ilustrado o módulo de segmentação proposto por este trabalho



Fonte: O autor.

1.2 OBJETIVOS DO TRABALHO

1.2.1 Objetivo geral

O objetivo geral deste trabalho é o desenvolvimento e prototipação em FPGA um módulo eficiente para segmentação de espaço livre em imagens estéreo. Por eficiente, entende-se que o módulo deve atingir taxas de processamento superiores a 30FPS, de modo que seja possível seu uso em aplicações de tempo real.

A qualidade da segmentação do módulo deve ser tal que as métricas de precisão e sensitividade, no processamento das imagens do *dataset* KITTI [1] sejam superiores a 70%, desta forma, o trabalho se insere entre os 20 melhores resultados entre os trabalhos que utilizam visão estéreo para segmentação de pista na lista disponível no site do *dataset*.

1.2.2 Objetivos específicos

São objetivos específicos deste trabalho:

- Proposição de um algoritmo de referência combinando técnicas descritas na literatura para segmentação de espaço livre em imagem estéreo
- Validação com pelo menos um *dataset* público
- Avaliação dos resultados obtidos em termos de qualidade, performance e eficiência energética em comparação com técnicas de terceiros.

[1] O dataset Daimler, também utilizado neste trabalho, não foi mencionado nos objetivos gerais pois foi usado apenas para validação qualitativa

1.3 CONTRIBUIÇÕES

Este trabalho traz contribuições para a área de processamento de imagem em sistemas embarcados, robótica e veículos autônomos. O Módulo proposto foi concebido baseando-se em técnicas amplamente difundidas na literatura da área, tendo atingido taxas elevadas de processamento, consumo de energia menor que os algoritmos executados em processador e métricas de qualidade satisfatórias. Os resultados obtidos viabilizam a sua aplicação em sistemas embarcados com restrições de potência e requisitos de resposta em tempo real.

Neste contexto, o trabalho também torna-se viável para integração com o módulo em FPGA para cálculo de mapas de disparidades proposto por CAMBUIM (2017) e para uso em projetos em desenvolvimento dos Grupos de Sistemas Embarcados e de Robótica do CIn da UFPE.

1.4 ORGANIZAÇÃO DO TRABALHO

Neste capítulo foram apresentadas e discutidas as motivações que justificam a realização deste trabalho, embasando-o fortemente nas pesquisas em andamento para percepção robótica e visão computacional do grupo de pesquisa em Engenharia de Computadores e Sistemas Embarcados do CIn-UFPE. Foram trazidos os objetivos gerais e específicos do trabalho, que serviram para embasar os requisitos e restrições do módulo proposto.

O capítulo 2 introduz e comenta sobre conceitos básicos necessários para o entendimento desta dissertação. Primeiramente é apresentada a noção de percepção para sistemas robóticos, em seguida, dentro do conceito de percepção é explorado o campo da visão computacional estéreo aprofundando o embasamento teórico de geometria epipolar usado para extração de informações por mapas *u e v disparidade*.

Ainda no segundo capítulo, são comentadas as restrições aos quais os sistemas robóticos autônomos estão submetidos, enquadrando-os no prisma do desenvolvimento de sistemas embarcados. Finalmente, é discutido o uso de FPGAs para a aceleração de algoritmos. Este capítulo além de apresentar conceitos, apresenta outra linha justificativa para o projeto.

Trabalhos relacionados são comentados e comparados no capítulo 3. É feita uma revisão breve e direta da literatura, tendo como foco as contribuições e aplicações do módulo proposto. Ao final, foi construída uma tabela comparativa entre os métodos analisados. Para efeito de comparação é dado foco na discussão de trabalhos que usaram o *dataset* KITTI para avaliação dos seus resultados, visto que com estes serão feitas comparações em termos de qualidade e desempenho.

O algoritmo e a arquitetura propostas no trabalho são apresentados no capítulo 4, onde são detalhados o fluxo de dados no algoritmo e o funcionamento de cada submódulo componente do sistema, detalha-se também as máquinas de estado responsáveis pelo controle geral e dos módulos inferiores.

O capítulo 5 descreve e justifica as metodologias de desenvolvimento e validação adotadas. É apresentada a metodologia de desenvolvimento de sistemas embarcados baseada em refinamentos consecutivos, sendo mostrado como se deu o desenvolvimento e implementação do módulo seguindo estes procedimentos. Para a validação, são apresentados os *datasets* e as métricas pelos quais o trabalho foi avaliado.

Resultados e discussões são trazidos no capítulo 6. Faz-se o estudo dos resultados de qualidade e desempenho de segmentação sendo realizado também discussões sobre a síntese gerada para a FPGA, sobre o consumo de recursos do chip e eficiência energética.

Finalmente, no capítulo 7 são trazidas as considerações finais acerca do trabalho realizado, são mencionadas e discutidas as contribuições do trabalho para a comunidade e apresentados possíveis trabalhos futuros.

2 CONCEITOS BÁSICOS

Este capítulo discorre sobre alguns conceitos básicos nos quais este trabalho se justifica e está fundamentado. A seção 2.1 faz uma breve introdução ao conceito de percepção no contexto da robótica, inserindo este trabalho neste contexto. A seção 2.2 apresenta a ideia de visão computacional estéreo, que são as técnicas em que se baseiam esta dissertação. A seção 2.3 introduz o conceito de mapas de *u-disparidade* e *v-disparidade* e a sua aplicação na segmentação de imagens a partir de mapas de disparidade. A seção 2.4 discute sobre as restrições sob as quais o projeto de sistemas robóticos está submetido, inserindo-o no macro-contexto de desenvolvimento de sistemas embarcados. Finalmente, a seção 2.5 discute a aceleração de algoritmos em FPGAs e sua importância para aplicações em sistemas embarcados.

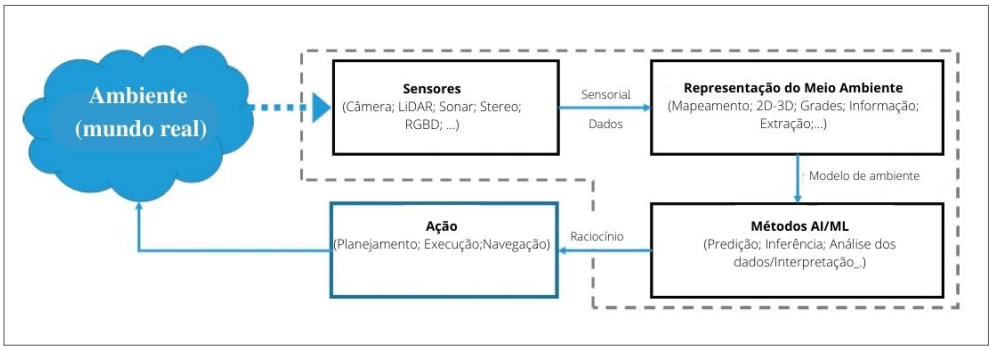
2.1 PERCEPÇÃO EM ROBÓTICA

No contexto da robótica, percepção é entendida como um conjunto de habilidades que envolvem perceber, compreender e raciocinar sobre o ambiente no entorno do agente. Estes sistemas envolvem processamento de dados de sensoriamento, modelagem matemática para representação do meio e algoritmos baseados em aprendizagem de máquina. Este conceito é crucial para a tomada de decisão, planejamento e operação em ambientes relevantes do mundo real(PREMEBIDA; AMBRUS; MARTON, 2019). Alguns exemplos de subáreas da percepção robótica, especialmente no contexto de veículos robóticos autônomos, são:

- detecção de obstáculos (RENNIE et al., 2016; BORE; JENSFELT; FOLKESSON, 2018)
- reconhecimento de objetos (HANDA et al., 2016; FIRMAN, 2016)
- representação tridimensional (SAARINEN; ANDREASSON; LILIENTHAL, 2012)
- classificação de terrenos (MANDUCHI et al., 2005)
- detecção de pista (FERNANDES et al., 2014)
- detecção de veículos (ASVADI et al., 2018)
- detecção de pedestres (PREMEBIDA; NUNES, 2013)
- rastreamento de objetos (BORE; JENSFELT; FOLKESSON, 2018)

A Figura 4 ilustra as partes que compõe um sistema robótico genérico, os blocos relacionados à percepção estão destacados pela linha tracejada. A informação é adquirida do ambiente por meio dos mais diversos sensores, tais como câmeras mono e estéreo, LiDAR e sonares. Os dados captados por sensores normalmente passam por uma etapa de pré-processamento

Figura 4 – Diagrama de blocos de um sistema de percepção robótico genérico

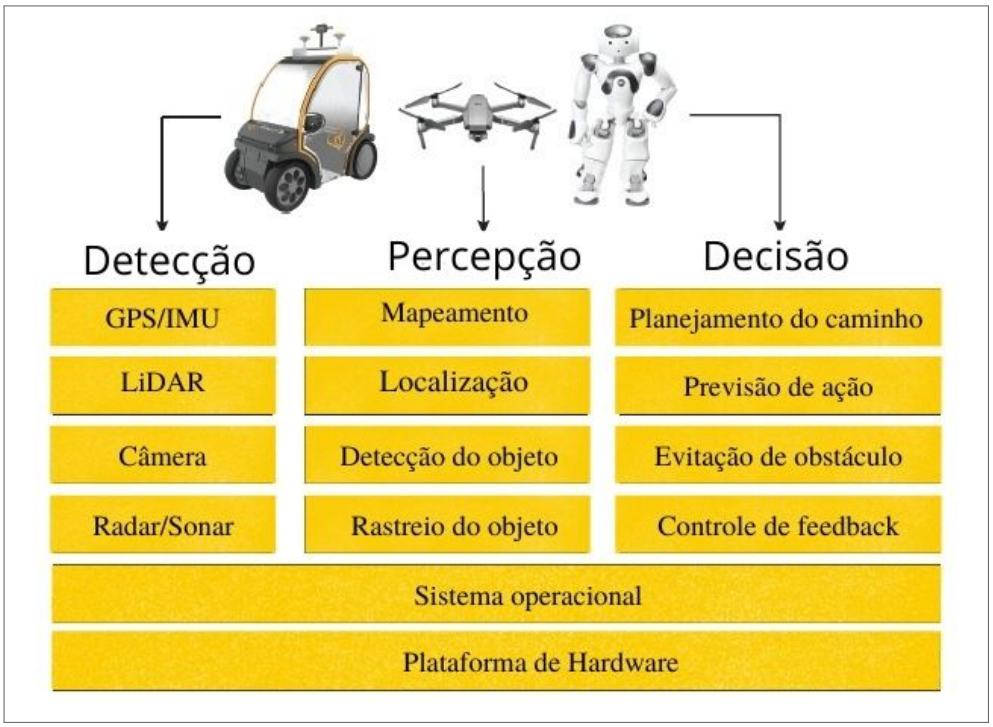


Fonte: Premebida, Ambrus e Marton (2019) (ADAPTADO).

na qual os sinais são filtrados e amplificados e são enviados para os módulos responsáveis pela representação de ambiente gerando um modelo que é, então, utilizado por métodos de aprendizagem de máquina e inteligência artificial para que o agente atue no ambiente por meio de planejamento, execução, navegação e etc.

Em Liu et al. (2021) é proposta uma representação complementar, na qual as tarefas de detecção, percepção e decisão ocorrem simultaneamente, conforme ilustrado na Figura 5. Nesta perspectiva a percepção estaria relacionada ao conceito de representação e interpretação do ambiente, estando separada das partes responsáveis pelo sensoriamento.

Figura 5 – Componentes de um sistema robótico



Fonte: Liu et al. (2021) (ADAPTADO).

Independente da abordagem, o sensoriamento é um elemento chave na percepção está-

tica e dinâmica dos objetos que compõe o ambiente e na construção de uma representação confiável e suficientemente detalhada do entorno.

Este trabalho se insere no âmbito da percepção robótica, uma vez que se utiliza de informações de tridimensionalidade geradas a partir de um par de imagens estéreo para realizar segmentação de espaço livre e não-livre (obstáculos) em imagens de aplicação em contexto automotivo.

2.2 VISÃO COMPUTACIONAL ESTÉREO

Visão computacional é uma subárea da ciência da computação que busca obter informações sobre o mundo real a partir de um conjunto formado por uma ou mais imagens por meio de técnicas e métodos para extração e interpretação de suas propriedades (SZELISKI, 2011).

Este campo é de suma importância para a implementação de agentes robóticos móveis e veículos autônomos, visto que cada vez mais tem se buscado obter e processar informações de profundidade, isto é, usar a tridimensionalidade para a elaboração de modelos mais precisos do ambiente(PREMEBIDA; AMBRUS; MARTON, 2019).

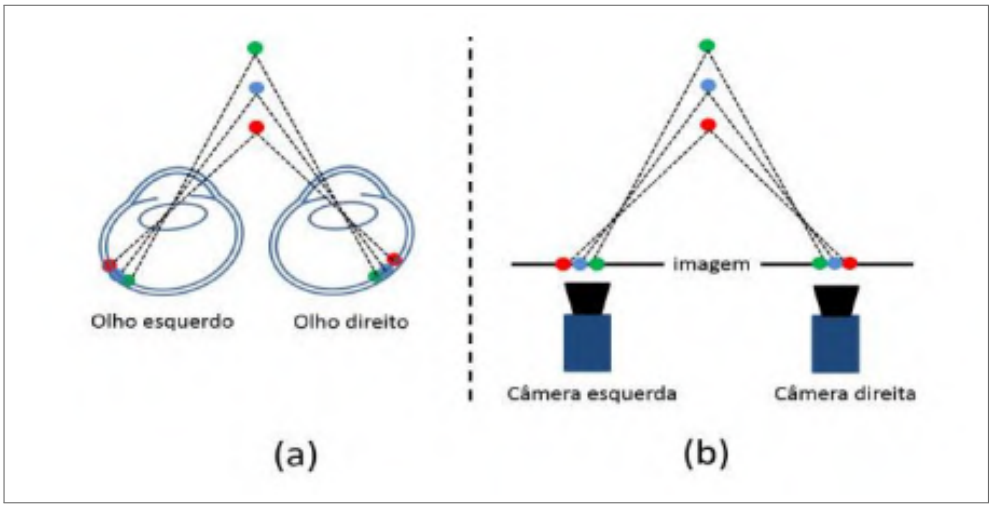
Uma das maneiras de obter estas informações de profundidade através de imagens é a utilização de duas ou mais imagens, obtidas simultaneamente de pontos espaciais ligeiramente diferentes e estimar, a partir de técnicas computacionais, a proximidade dos objetos captados. Tal técnica é inspirado (BENYUS, 2002) no sofisticado sistema de visão dos animais com visão binocular, no qual cada olho recebe informação do ambiente a partir de uma perspectiva diferente, possibilitando que o sistema nervoso extraia a informação de profundidade considerando o fato que, o quão maior for a proximidade do objeto do observador, maior o deslocamento entre as duas imagens geradas em cada retina. O cérebro consegue mapear este deslocamento, ou disparidade, na informação da distancia real entre o indivíduo e o objeto.

Os sistemas de visão computacional estéreo, com duas ou mais câmeras, utilizam-se deste mesmo princípio geométrico para obter informações tridimensionais de um cenário. A luz refletida nos objetos do mundo real é projetada em pixeis nas imagens captadas pelas lentes das câmeras, estes pixeis estarão projetados em locais diferentes nas imagens geradas pelas câmeras esquerda e direita. Da mesma forma ao que ocorre nos olhos, quanto mais próximo o objeto estiver das lentes, maior será a diferença espacial entre o local de projeção do mesmo ponto na câmera esquerda e direita.

Este mecanismo óptico esta ilustrado na Figura 6 em (a) para sistemas de visão biológica e em (b) para um par de câmeras estéreo. O problema se torna então encontrar a correspondência entre os pixeis das imagens esquerda e direita e calcular a diferença entre suas posições, de modo a obter a profundidade da imagem.

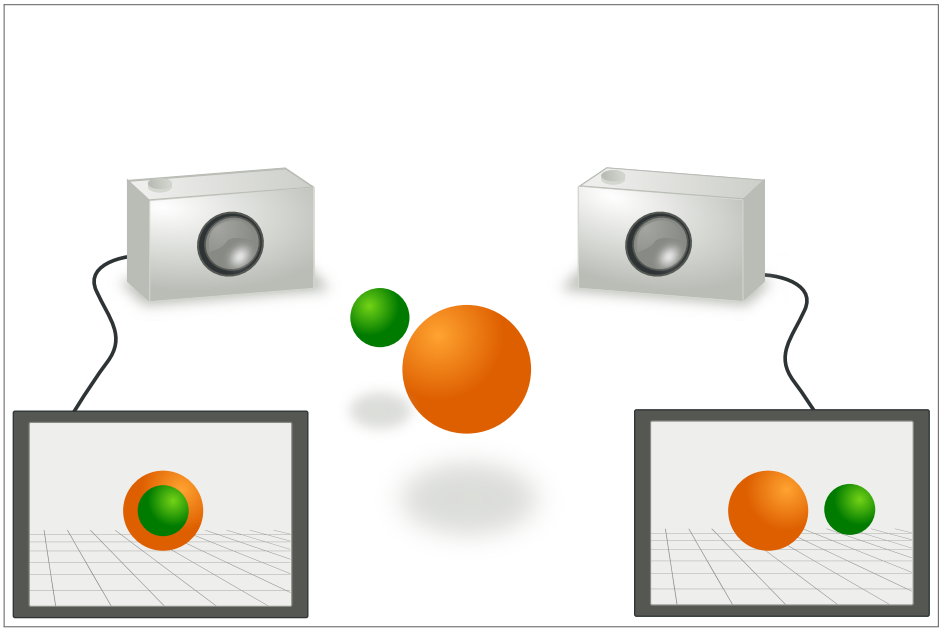
Sistemas de visão estéreo são descritos de acordo com a geometria epipolar (XU; ZHANG, 1996). Nesta representação, ilustrada nas Figuras 7 e 8, as câmeras são modeladas de acordo com um modelo de câmera *pinhole*, nos quais a abertura pontual das câmeras esquerda e direita são descritas respectivamente como os pontos O_l e O_r.

Figura 6 – Captação de imagens de diferentes perspectivas para aquisição de informações de profundidade em sistemas de visão estéreo binocular animal (a) e computacional (b)



Fonte: Cambuim (2017).

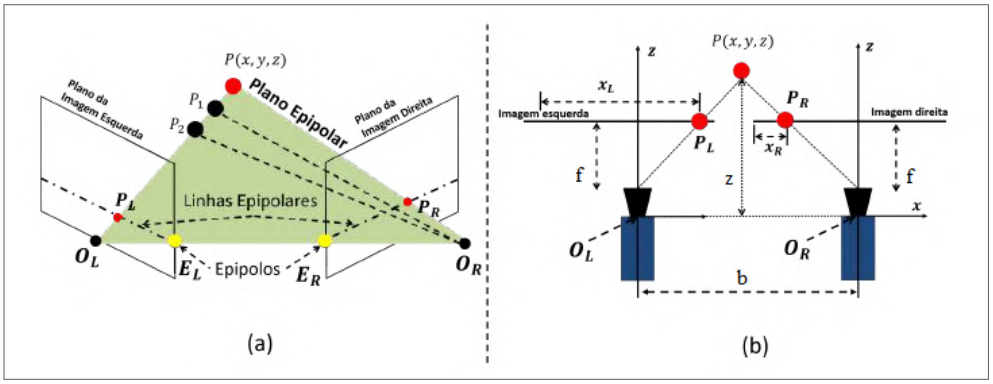
Figura 7 – Caso típico de geometria epipolar. Duas câmeras capturando uma imagem do mesmo cenário a partir de pontos de vista diferentes



Fonte: *Creative Commons*.

$P_{(x,y,z)}$ é um ponto no espaço tridimensional e os dois pontos E_l e E_r são os epipolos. Define-se epipolo como o ponto de intersecção da linha através dos centros óticos com o plano de imagem. Os pontos P, O_l e O_r definem o plano epipolar. A reta $P - O_l$ é vista pela câmera esquerda como um ponto P_l por estar alinhada com o centro da câmera O_l. No plano da imagem direita, a linha representada por $E_r - P_r$ é definida como linha epipolar. Para cada ponto observado em uma imagem, pode-se observar um ponto correspondente na

Figura 8 – Geometria epipolar em um sistema de câmeras estéreo binoculares. (a) Vista 3D e (b) 2D



Fonte: HADJITHEOPHANOUS et al. 2010 adaptado por CAMBUIM 2017.

linha epipolar da outra imagem. Esta restrição, chamada *restrição epipolar*, reduz o problema de encontrar o pixel correspondente a uma busca ao longo de linhas epipolares conjugadas.

A distância da posição dos pixeis correspondentes entre as duas imagens é definida como *disparidade* e é calculada de acordo com a equação 2.1.

$$\Delta = |x_l - x_r| \tag{2.1}$$

O processo de busca pelos pixeis correspondentes no par de imagens estéreo é chamado de *Correspondência Estéreo*. Dados dois pontos correspondentes conhecidos $P_l = (x_l, y_l)$ e $P_r = (x_r, y_r)$ e os parâmetros geométricos do par de câmeras, i.e distância focal e distância entre centros óticos, as coordenadas do ponto $P(x, y, z)$ são calculadas por triangulação, usando o princípio de semelhança dos triângulos definidos por (P, O_l e O_r) e (P, P_l e P_r) as coordenadas de P são dadas pelas Equações 2.2, 2.3 e 2.4.

$$x = \frac{x_l \cdot b}{\Delta} \tag{2.2}$$

$$y = \frac{y_l \cdot b}{\Delta} \tag{2.3}$$

$$z = \frac{b \cdot f}{\Delta} \tag{2.4}$$

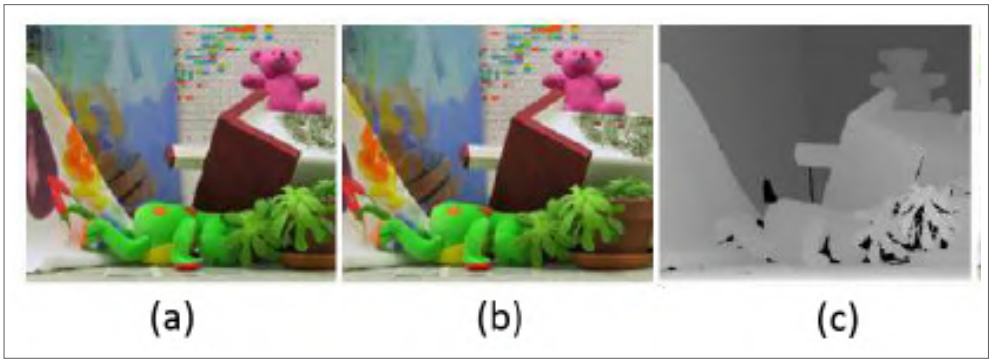
Nas equações o valor b indica a distância entre os centros óticos O_l e O_r. O valor f é o comprimento focal das lentes e o valor z é a distância entre o ponto P e a linha de base b, representando a profundidade do objeto.

A relação entre disparidade e profundidade dos pixeis é mostrada na Equação 2.4. Esta relação é diretamente proporcional à distancia entre as câmeras e ao comprimento focal da lente e inversamente proporcional à disparidade.

Calculando o valor da disparidade para todos os pixeis da imagem é obtido o mapa de disparidades, conforme as equações correspondentes, quanto mais próximo o objeto, maior a

disparidade no mapa. A Figura 9 mostra um exemplo de mapa de disparidades obtido a partir de um par de imagens estéreo. Neste tipo de representação gráfica, a intensidade do pixel representa a informação do valor da disparidade. No exemplo, pode-se perceber que os objetos mais próximos, como a pelúcia verde, são representados no mapa de disparidades (c) por pixeis de maior intensidade, de modo diferente da pelúcia rosa que por estar mais distante, possui disparidade menor e é representado por pixeis menos intensos.

Figura 9 – Construção do mapa de disparidades (c) a partir da imagem esquerda(a) e direita(b)

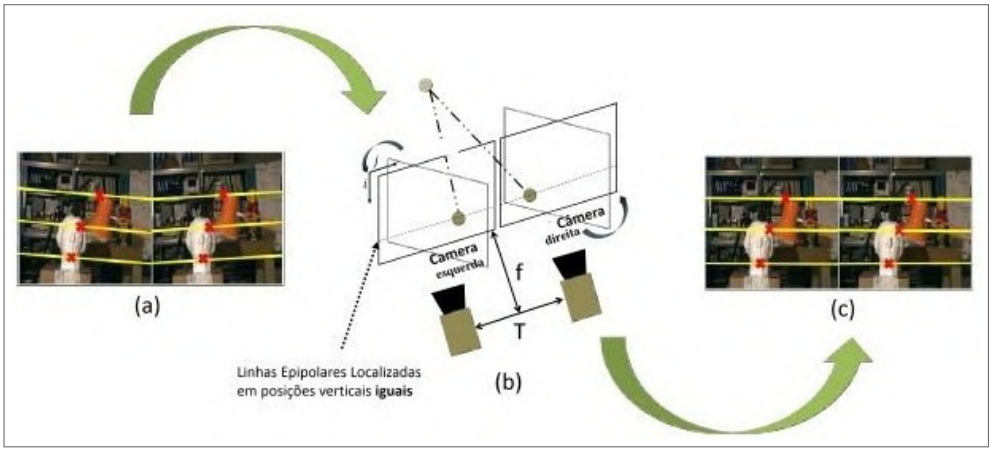


Fonte: Middlebury Stereo Datasets Scharstein e Szeliski (2003).

Para reduzir a complexidade da *Correspondência Estéreo* é realizado o processo de retificação de imagens estéreo. A retificação é feita a partir de parâmetros intrínsecos (comprimento focal, centros óticos e distorção das lentes) e extrínsecos (posições relativas e orientações de cada câmera do sistema) obtidos por meio de calibração.

A retificação alinha os pares de linhas epipolares conjugadas para um eixo de imagem comum, de modo que a busca por pixeis correspondentes é realizada apenas ao longo da linha epipolar ao invés de todo o espaço bidimensional. A Figura 10 ilustra as etapas do processo de retificação, nela é possível observar o resultado do alinhamento das linhas epipolares dos pares.

Figura 10 – Processo de retificação de imagens. (a) Imagens originais, (b) Processo de retificação e (c) Imagens retificadas



Fonte: CAMBUIM 2017

2.3 EXTRAÇÃO DE INFORMAÇÕES POR HISTOGRAMAS DE DISPARIDADE

A aplicação principal dos mapas de disparidade é, naturalmente, a extração de informação de profundidade para reconstrução do espaço tridimensional para navegação, inferência de textura e formato e aperfeiçoamento de dados de odometria. Todavia, os mapas de disparidades também possibilitam, por meio de técnicas de pós processamento, extrair várias características geométricas do cenário sensoreado como, por exemplo, o perfil de objetos perpendiculares e paralelos ao plano Z. Esta extração de características é feita através da construção dos mapas *v-disparidade* (LABAYRADE; AUBERT; TAREL, 2002) e *u-disparidade* (HU; LAMOSA; UCHIMURA, 2005) tendo aplicações para segmentação de pista, espaço livre e detecção e rastreamento de obstáculos e pedestres (HU; LAMOSA; UCHIMURA, 2005).

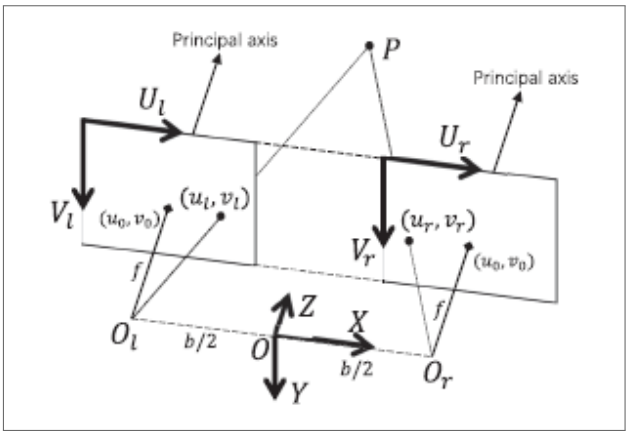
Para o par de imagens estéreo retificadas, é adotado o sistema de coordenadas $U - V$, ilustrado na Figura 11, no qual U_l e U_r são as variáveis do eixo das abscissas e V_l e V_r são as variáveis do eixo das ordenadas nas imagens esquerda e direita respectivamente. Na imagem, o par (u_0, v_0) é a coordenada da projeção do centro ótico da câmera, (X, Y, Z) representa os eixos do mundo real. Para simplificar as representações, assumimos X como paralelo a U_l e U_r, Y como paralelo a V_l e V_r e Z paralelo ao eixo principal. f é a distância focal das lentes e b é a distância entre as câmeras.

Por semelhança de triângulos, pode-se afirmar que os pontos tridimensionais do cenário capturado são projetados nas imagens esquerda e direita de acordo com o sistema de equações dado em 2.5.

$$\begin{cases} u_{l,r} = u_0 + \frac{f}{z}(x \pm \frac{b}{2}) \\ v = v_l = v_r = v_0 + \frac{f}{z} \end{cases} \tag{2.5}$$

A disparidade previamente definida na Equação 2.1, também pode ser calculada mani-

Figura 11 – Sistema de coordenadas adotado para pares de imagem estéreo calibrados e retificados



Fonte: ZHANG et al. 2018b

pulando algebricamente a Equação 2.4 obtendo a Equação 2.6.

$$\Delta = \frac{fb}{z} \tag{2.6}$$

Para efeitos de simplificação, nesta seção será adotado o modelo do chão, ou espaço livre, como uma região contida num plano π_1 paralelo a linha base (reta que conecta as duas câmeras), sendo o π_1 representado pela equação de plano 2.7

$$y = a_1 z + d_1 \tag{2.7}$$

Na qual a_1 é um pequeno valor que representa a inclinação do plano que contem o chão e d_1 um valor constate que indica deslocamento em relação à origem. Substituindo as equações 2.3 e 2.4 nos valores de y e z e considerando $y_l = v - v_0$, pode-se manipular a equação 2.7 de modo que ela assuma a forma mostrada na equação 2.8.

$$v = v_0 + f a_1 + \frac{d_1}{b}\Delta. \tag{2.8}$$

Portanto, percebe-se que o plano π_1 que contém o chão é representado por uma reta no domínio (v, Δ). A transformada que leva o mapa de disparidades para este domínio é a construção do mapa *v-disparidade* (LABAYRADE; AUBERT; TAREL, 2002).

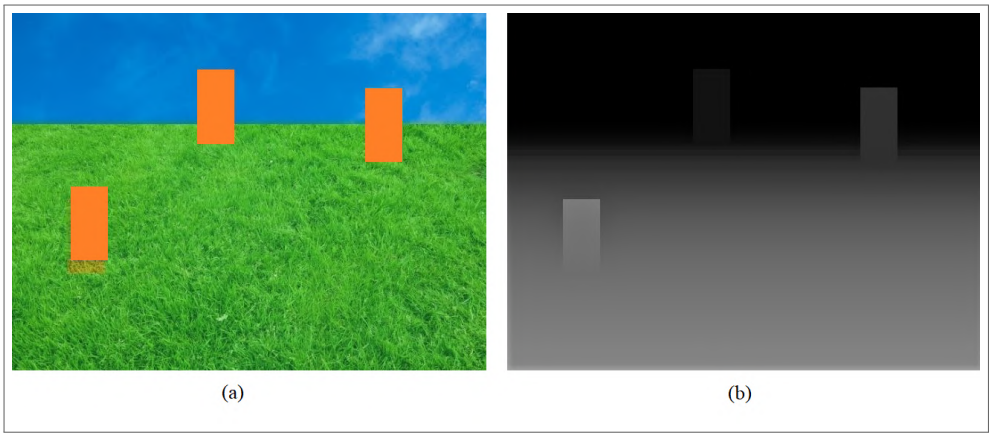
De maneira análoga, pode-se modelar um obstáculo como um anteparo contido no plano π_2, que é paralelo ao eixo Y. π_2 descrito pela equação de plano $z = a_2 x + d_2$. Substituindo as equações 2.2 e 2.4 na equação do plano e manipulando um pouco as variáveis, pode-se chegar a uma formulação alternativa na qual a relação entre Δ e u_l para π_2 é dada pela equação 2.9.

$$u_l = u_0 + \frac{f}{a_2} - \frac{d_2}{a_2} \cdot \frac{\Delta}{b} \tag{2.9}$$

Deduz-se portanto que o plano π_2 é projetado em uma reta no domínio (Δ, u_l) e os pontos do obstáculos são levados em um segmento desta. A transformação que leva o mapa de disparidades do domínio (v, u) para o domínio (Δ, u) é a construção do mapa *u-disparidade*.

Para entender a construção dos mapas *v-disparidade* e *u-disparidade* considere como ponto de partida o cenário gerado artificialmente mostrado na Figura 12 em (a) e o mapa de disparidades gerada a partir da imagem e seu par conjugado estéreo mostrado em (b).

Figura 12 – Cenário artificial. Em (a) imagem, parte de um conjugado estéreo, com uma planície com obstáculos e em (b) mapa de disparidades gerado a partir da imagem. Quanto mais claro um ponto, maior a sua disparidade e mais próximo ele se encontra da câmera.



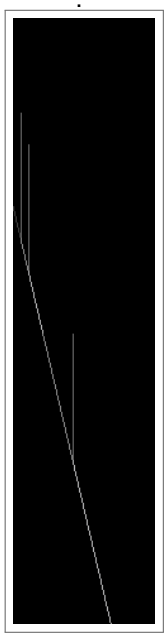
Fonte: O Autor

O *v-disparidade* é calculado de forma que cada linha do mapa é o histograma das disparidades da linha correspondente do mapa de disparidades, sendo os valores nas colunas a intensidade para as disparidades no histograma. Como mencionado, o chão, ou superfície livre, é levado em uma reta neste mapa. Um exemplo disto pode ser visto na Figura 13, que mostra o mapa *v-disparidade* gerado a partir do cenário da Figura 12. Os segmentos perpendiculares a reta são efeito dos obstáculos existentes na imagem, na seção 4.1 do Capítulo 4 será discutido uma forma de remover a presença de obstáculo no cálculo do mapa v-disparidade.

O mapa *u-disparidade*, que pode ser usado para calcular o perfil de obstáculos, é calculado de forma que cada coluna do mapa corresponda ao histograma da coluna correspondente no mapa de disparidades. Os pixeis do obstáculo são projetados em segmentos de reta, que terão valores mais intensos no domínio (Δ, u). Com isso, uma forma simples de estimar obstáculo é realizando uma operação de *thresholding*. O exemplo mostrado na Figura 14, ilustra o resultado para o mapa de disparidades da Figura 12. É possível observar os segmentos de reta correspondente aos três obstáculos na imagem.

A correspondência de obstáculos e pista nos mapas *u e v-disparidade* pode ser melhor compreendida na representação mostrada na Figura 15. É possível observar os obstáculos levados em segmentos de reta nos mapas *u e v-disparidade* e a pista sendo levada em uma reta.

Figura 13 – Mapa *v-disparity* calculado a partir do mapa de disparidades mostrado na Figura 12 (b)



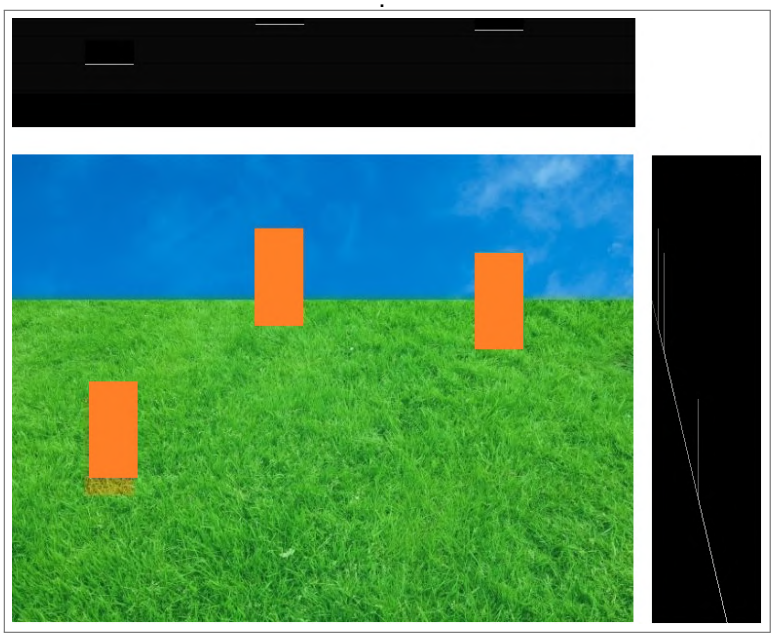
Fonte: O Autor

Figura 14 – Mapa *u-disparity* calculado a partir do mapa de disparidades mostrado na Figura 12.

Fonte: O Autor

Uma forma de entender a ideia das transformações *u e v disparidade* é interpretá-las como uma mudança de perspectiva. Sendo o mapa *u-disparidade* uma vista superior e o mapa *v-disparidade* uma vista lateral. Ambas com a informação de profundidade codificadas na informação da disparidade.

Figura 15 – Cenário artificial e os mapas *u e v-disparidade* gerados a partir de seu mapa de disparidades



Fonte: O Autor

2.4 RESTRIÇÕES EM SISTEMAS ROBÓTICOS AUTÔNOMOS

Os sistemas robóticos podem também ser entendidos como um tipo de sistema embarcado. Segundo VAHID; GIVARGIS (2001), os sistemas embarcados apresentam algumas características em comum que os distinguem de outros sistemas de computação:

1. Função definida: Um sistema embarcado normalmente é projetado para executar uma determinada aplicação específica. Por exemplo, um sistema embarcado para controlar a temperatura de um aquário é projetado para executar esta função, em contraste, um computador de mesa realiza uma grande gama de funções.

2. Restrições de projeto: Todos os sistemas de computação tem restrição de projeto, entretanto, as restrições impostas aos projetos de sistema embarcado são normalmente muito mais estritas. Existem diversas métricas de projeto tais como custo, tamanho, performance temporal e consumo energético que devem ser respeitadas para a realização do dispositivo projetado.

3. Reatividade em tempo real: É comum que sistemas embarcados devem ser projetados para apresentar resposta rápida à mudanças no ambiente, computando os resultados em tempo real.

O projeto de um sistema embarcado deve resultar numa implementação que preencha as requisições de funcionalidade solicitadas e simultaneamente esteja otimizada dentro das

diversas métricas de projeto (VAHID; GIVARGIS, 2001; GAJSKI et al., 2009). Dentre as mais importantes métricas de projeto, estão:

- *Custo de Engenharia Não Recorrente (NRE)*: Custo financeiro de projetar o sistema, acontece apenas uma vez e, por isso, é chamado não recorrente.
- *Custo Unitário*: Custo financeiro da manufatura de uma cópia do sistema, estando excluído o custo NRE.
- *Tamanho*: Dimensão física do dispositivo, frequentemente medido em bytes para *software* e em número de portas lógicas para *hardware*.
- *Performance*: Tempo de execução da função realizada pelo sistema.
- *Consumo*: Potência consumida pelo sistema.
- *Flexibilidade*: Possibilidade de mudar a funcionalidade do sistema sem grande esforço e custo.
- *Tempo de prototipação*: Tempo necessário para construir uma versão funcional do sistema.
- *Tempo de mercado*: Tempo necessário para desenvolver o sistema a ponto deste poder ser incluído no mercado.
- *Manutenibilidade*: Possibilidade de modificar o sistema após este já ter sido lançado.
- *Exatidão*: Confiança que o sistema executa sua função de maneira correta.
- *Segurança*: Probabilidade de que o produto não causará acidentes

Normalmente há uma relação de *trade off* entre as métricas citadas, isto é, melhorar o desempenho do sistema em um quesito significa prejudicar as outras. Nas aplicações envolvendo visão computacional estéreo, por exemplo, pode-se melhorar executar algoritmos com boa precisão e em tempo real usando processadores gráficos dedicados(GPUs), em contrapartida o custo do projeto e o consumo de energia tendem a subir. O desafio dos projetistas de sistemas embarcados é encontrar a melhor otimização do projeto dentro das métricas citadas.

Veículos robóticos autônomos funcionam através da integração de diversas técnicas, o que inclui sensoriamento, percepção, mapeamento, tomada de decisão, controle linear e não-linear, etc. Tal complexidade faz da robótica moderna um dos campos mais desafiadores das ciências da computação. Os sistemas robóticos precisam, simultaneamente, processar uma quantidade muito grande de dados, proveniente de múltiplos sensores, em tempo real, estando submetidos a severas restrições de acurácia e sincronização. Todavia, para serem viáveis, normalmente a implementação é feita em ambientes com limitações de capacidade

de processamento, memória, largura de banda e energia, tornando difícil o cumprimento dos requisitos de tempo-real.

As aplicações robóticas, em geral possuem requisitos severos de latência temporal, com fortes limitações em termos de consumo de potência, neste cenário, um dos grandes desafios dos pesquisadores do campo é desenvolver sistemas que possam suportar a quantidade de computação necessária para executar tarefas como sensoriamento tridimensional, localização, navegação e planejamento de rota dentro dos recursos disponíveis na plataforma e cumprindo abaixo do tempo crítico definido.

2.5 ACELERAÇÃO EM FPGAS

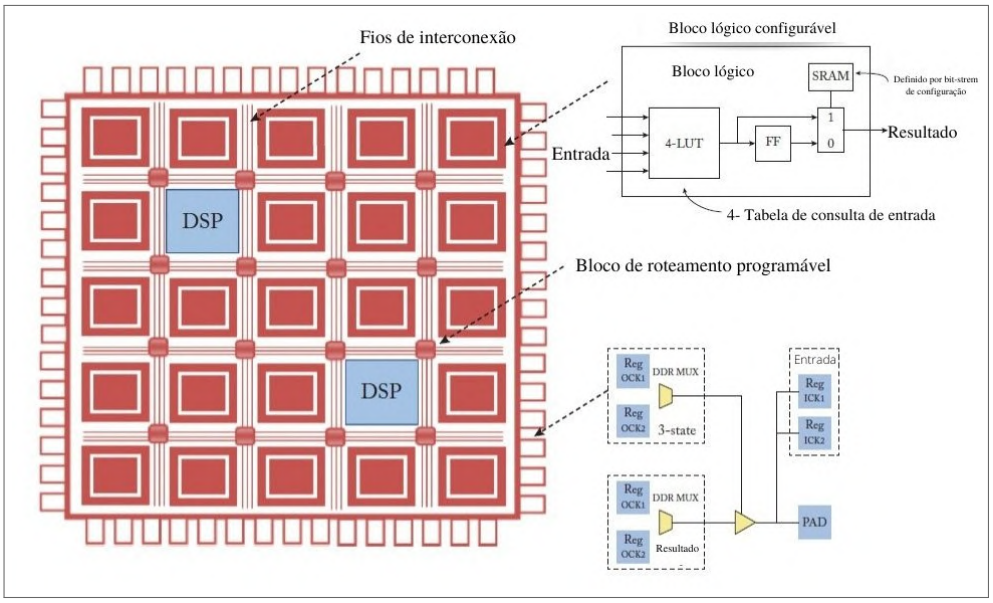
Uma tecnologia importante para o desenvolvimento de sistemas capazes de cumprir os requisitos de processamento dentro do cenário de restrições da robótica é a *Field Programmable Gate Array* (FPGA). Inicialmente, estes dispositivos foram concebidos como um tipo de *hardware* que pudesse ser configurado diversas vezes para implementar as funcionalidades de diferentes tipos de circuitos. Enquanto a maioria dos CIs tem suas funcionalidades fixas, uma FPGA é fabricada de modo que ela possa ser configurada e, sempre que necessário, reconfigurada para determinada aplicação. Tal programação é realizada via software, por meio de alguma das diversas linguagens de descrição de *hardware* existentes, como por exemplo o SystemVerilog e o Verilog (ACCELLERA, 2004), o *VHSIC Hardware Description Language* (VHDL) (MICROSOFT, 1997) ou o *Altera Hardware Description Language* (AHDL) (ALTERA, 1997). Esta capacidade de configuração rápida é usada para a implementação de protótipos de circuitos eletrônicos diminuindo drasticamente o tempo de prototipação (GOKHALE, 2005).

As FPGAs são uma plataforma de computação que permite processar uma quantidade massiva de dados de forma paralela, sendo muito útil para aplicações de alta performance como o de processamento digital de sinais e imagem (KILTS, 2007) e controle (KOCUR; KOZAK; DVORSCAK, 2014). Com os avanços na tecnologia de fabricação e a popularização das FPGAs, que hoje possuem um grande quantidade de recursos, surge a linha de pesquisa e desenvolvimento da computação reconfigurável (FERNANDES, 2016). Nesta linha, são desenvolvidos algoritmos que de forma total ou parcial, buscando explorar suas características para obter melhores desempenhos de processamento.

A Figura 16 mostra os componentes básicos que compõe a estrutura de uma FPGA moderna. São eles:

- **Blocos lógicos configuráveis** são os elementos lógicos básicos de repetição em uma FPGA. São conectados pelos blocos de roteamento programáveis de modo a exercer funções, sendo capazes de implementar funções lógicas complexas, memórias e realizar sincronização. Estes blocos são compostos de elementos ainda menores como *flip-flops*, *Look-up Tables* (LUT), e multiplexadores. Os *flip-flops* são o menor elemento de armazenamento na FPGA, sendo registradores de um bit usados para salvar estados lógicos

Figura 16 – Arquitetura geral de um dispositivo FPGA



Fonte: LIU et al. 2021 (ADAPTADO).

entre ciclos de *clock*. Os LUTs armazenam saídas pré-definidas para cada entrada possível, provendo uma forma rápida para obter resultados de operações lógicas. Multiplexadores são circuitos lógicos seletores. Qualquer lógica pode ser implementada por meio de *flip-flops*, LUTs e multiplexadores.(KILTS, 2007; LIU et al., 2021; MEYER, 2007).

- **Blocos de roteamento programáveis** são os elementos responsáveis por prover a conectividade programável entre os blocos lógicos configuráveis entre si e com os elementos de interface. São basicamente elementos de dois tipos: os blocos de conexão e a matrix de *switches*, que realiza o reteamento de fios para a configuração das ligações.
- **Blocos de interface I/O** são *buffers* de entrada e saída com lógica *three-state* usados para passagem de sinal nas entradas e saídas do chip.
- **Processadores de Sinal Digital**. *Digital Signal Processor* (DSP) são blocos otimizados para processar funções típicas do processsamento digital de sinal, estes blocos contém multiplicadores, adicionadores, acumuladores entre outros. (MEYER, 2007; LIU et al., 2021).

As FPGAs são plataformas com característica reprogramável e, portanto, personalizável, que permitem o desenvolvimento de sistemas de computação paralela com capacidade de realizar processamento massivo de dados em pouco tempo, e com baixo consumo energético. Dentre as possíveis aplicações está o contexto no qual se insere este trabalho: o processamento de imagem aplicado à robótica.

3 TRABALHOS RELACIONADOS

A área da segmentação de obstáculos e espaço livre é amplamente discutida na literatura relacionada às áreas de processamento de imagem e veículos autônomos. A maior parte das técnicas, naturalmente, está direcionada para aplicações em imagens monoculares, sendo este também um cenário com maior disponibilidade de *datasets* (FRITSCH; KüHNL; GEIGER, 2013; RATEKE; JUSTEN; WANGENHEIM, 2019; ENZWEILER et al., 2010; ZHANG et al., 2018a). Sistemas baseados em imagem monocular podem ser adicionados de outros tipos de sinais para a obtenção de melhores resultados, sendo possível encontrar dados voltados para a segmentação de espaço livre contendo LiDAR (CHEN; ZHANG; TAO, 2019; GU; YANG; KONG, 2021; CALTAGIRONE et al., 2018) e informações de Global Positioning System (GPS) (Wang et al., 2020). Pode-se adicionar também uma imagem conjugada complementar (esquerda ou direita) que, junto com a imagem inicial, transforma o sistema em um par binocular estéreo.

O trabalho de FRITSCH; KüHNL; GEIGER (2013) introduz o *dataset Karlsruher Institut für Technologie and Toyota Institute* (KITTI) para detecção de estrada em contexto automotivo. O conjunto de imagens possui extensões para visão binocular, LiDAR e sinal de GPS e possui *groundtruth* para as imagens do subconjunto de treinamento. O trabalho ainda detalha e justifica as métricas a serem usadas para avaliar métodos pelo dataset, disponibilizando em seu *website* uma espécie de competição[1] listando o desempenho de cada trabalho a ele submetido.

Dada a motivação deste trabalho, para efeito de comparação foram selecionados os métodos e técnicas para segmentação de espaço livre que utilizem como entrada **imagens estéreo binoculares** com ou sem combinação com outras informações. A seção 3.1 traz uma breve revisão da literatura científica com trabalhos mais relevantes relacionados a esta dissertação, seguido de discussões e análises comparativas realizadas na seção 3.2. As métricas adotadas para avaliação dos trabalhos são Precisão (P), Sensitividade (S), Precisão Média (PM) e Medida F1 (F1). Essas medidas são detalhadas na seção 5.4 do capítulo 5.

3.1 REVISÃO DA LITERATURA

Os trabalhos voltados para a segmentação de espaço livre a partir de imagens estéreo utilizam diferentes técnicas, das quais as mais comuns envolvem aprendizagem de máquina, a geometria epipolar do mapa de disparidades, características de textura e cor do par conjugado estéreo ou ainda a combinação de uma ou mais destas técnicas. Para efeito de organização, este seção separa as técnicas que usam métodos de aprendizagem de máquina das técnicas que não as usam.

[1] <http://www.cvlibs.net/datasets/kitti/eval_road.php>

3.1.1 Técnicas com aprendizagem de máquina

Nesta subseção estão descritos os trabalhos que tem como principal característica de sua abordagem o uso de algoritmos de aprendizagem de máquina para realizar a segmentação. Aprendizagem de máquina é o subconjunto dos algoritmos de Inteligência Artificial em que o modelo é aperfeiçoado a partir da experiência e do uso de informação, sendo necessário uma etapa de treinamento com dados pré conhecidos (MITCHELL, 1997). Exemplos desses tipos de algoritmos são Redes Neurais Artificiais, Árvores de Decisão, Regressão Linear e técnicas estatísticas como Naive Bayes.

Um dos trabalhos que utiliza esse tipo de técnica é o proposto por CHEN et al. (2015). Os autores usam do conceito de *superpixel*, um ênuplo associado a cada pixel contendo o seu valor RGB médio, a média da posição 2D em cada imagem do par estéreo, a posição tridimensional estimada a partir do mapa de disparidades, as inclinações angulares em relação ao plano da câmera, informações se o pixel está ou não acima da linha do horizonte e o desvio padrão da coloração e do posicionamento.

A partir de um conjunto de imagens do KITTI, foram gerados estes superpíxeis que, por sua vez, foram utilizados para o treinamento de uma Rede Neural Artificial (RNA), usando a *toolbox* para aprendizagem de máquina do software de computação científica *Matlab*. O método obteve os melhores resultados em termos de qualidade da segmentação, dentre os que estão listados na página de resultados do *dataset* KITTI, os valores podem ser vistos na Tabela 1. Um exemplo de resultado de segmentação é mostrado na Figura 17. Na imagem, os locais destacados em vermelho são pixeis erroneamente classificados como negativos. Tal padrão se repete em outras imagens o que leva a inferir que o algoritmo é sensível a presença de sombras.

O foco deste trabalho não foi desempenho, sendo o método executado em computador com 4 núcleos de processamento e usando o matlab, que possui tempos de processamento mais longos que programas feitos em c, c++ ou mesmo Python. Para o método, o tempo de processamento de cada frame foi de 5 segundos.

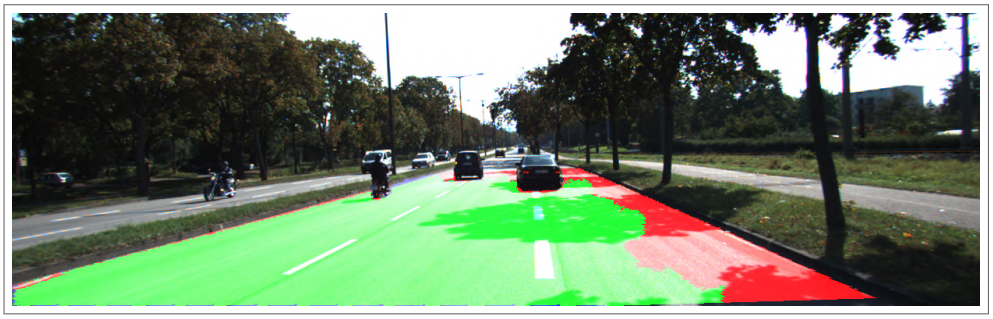
Tabela 1 – Resultados de qualidade de segmentação do método proposto por CHEN et al. (2015) para o dataset KITTI

MaxF1	PM	P	S
90.50 %	87.95 %	91.43 %	89.59 %

Fonte: The KITTI Vision Benchmark Suite

O trabalho de WANG et al. (2016) propõe uma abordagem estatística para determinação de fronteira de uma pista por meio de múltiplos modelos de Naïve Bayes (RUSSELL; NORVIG, 2009) que são construídos utilizando informações de profundidade obtidas através do mapa de disparidades, no caso o vetor normal do pixel no espaço tridimensional, e de características da imagem como cor e intensidade. O modelo com mais alto nível de confiança indicará se um pixel pertence a uma região de fronteira e, estando determinada a região, o algoritmo

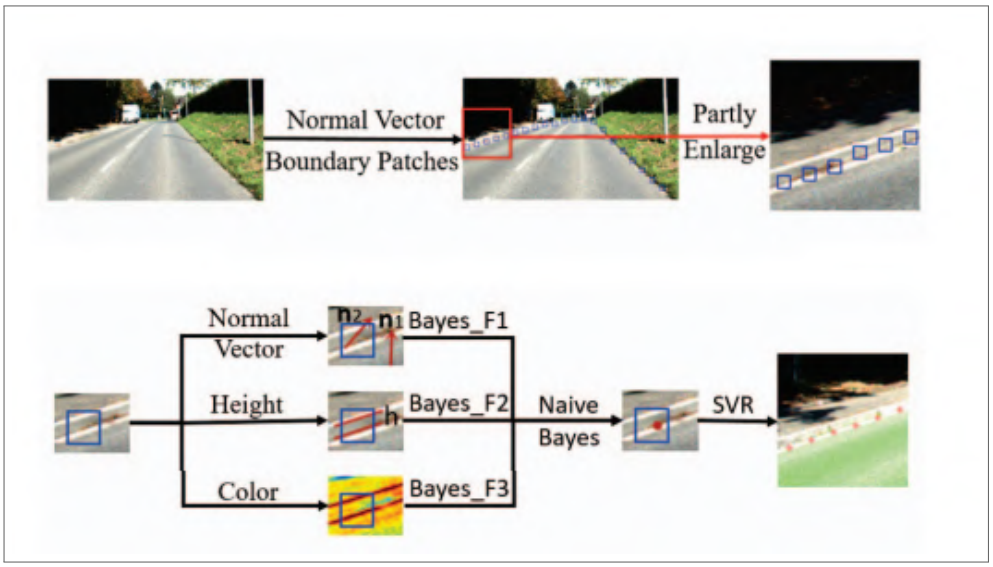
Figura 17 – Resultado de segmentação em imagem do dataset KITTI pelo algoritmo proposto em (CHEN et al., 2015)



Fonte: CHEN et al. 2015

utiliza de regressão por vetor de suporte para ajustar uma curva que represente os limiares da estrada. Este fluxo é ilustrado na Figura 18

Figura 18 – Diagrama de blocos mostrando o fluxo de funcionamento do algoritmo proposto em (WANG et al., 2016)



Fonte: WANG et al. 2016

A técnica foi implementada utilizando um núcleo de processamento em linguagem C/C++ e tem um tempo de processamento de 2,5s por frame. Os resultados de qualidade de segmentação, mostrados na Tabela 2 e com exemplo mostrado na Figura 19 foram similares aos de CHEN et al. (2015), com uma ligeira queda na precisão e sensitividade, porém obtendo melhor performance em termos de taxa de processamento, mesmo sendo executado em uma plataforma com menos capacidade computacional.

Alguns métodos combinam múltiplas técnicas para efetuar a segmentação da região desejada. Os autores de VITOR et al. 2014 desenvolveram uma técnica complexa que realiza a construção de superpixeis usando características da geometria epipolar do sistema, por meio dos mapas *u- e v-disparidade*, e da extração de características da imagem usando a trans-

Tabela 2 – Resultados de qualidade de segmentação do método proposto por WANG et al. (2016) para o dataset KITTI

MaxF1	PM	P	S
89.42 %	83.13 %	88.31 %	90.55 %

Fonte: The KITTI Vision Benchmark Suite

Figura 19 – Resultado de segmentação em imagem do dataset KITTI pelo algoritmo proposto em (WANG et al., 2016)

Fonte: WANG et al. 2016

formada *watershed*(NAJMAN; SCHMITT, 1994) e da construção de mapas *textons* (ZHU et al., 2005).

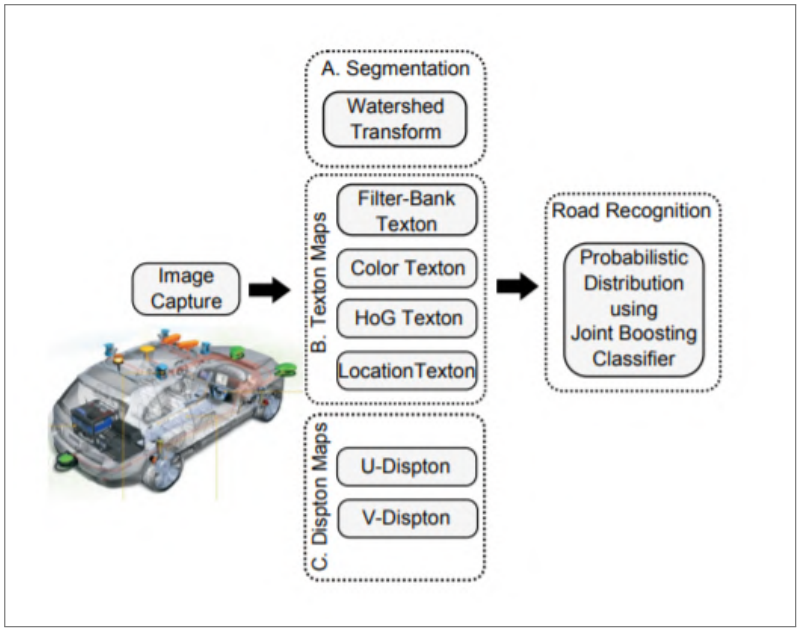
A transformada *watershed* opera sobre uma imagem em escala cinza utilizando o mesmo princípio da construção de mapas topográficos de relevo, considerando como a altura a intensidade do brilho de cada pixel, desta forma ela constrói linhas com separação de regiões por brilho. Por sua vez, um *texton* é um elemento formado por agrupamento de pixeis, contendo micro informações sobre a textura da imagem.

O resultado deste processo é a geração de uma distribuição de probabilidades multidimensional que é o modelo de entrada para o algoritmo de classificação por comitê do tipo *Joint Boosting*(ZHOU, 2012) que é implementado para realizar a segmentação da pista. Essa técnica consiste na junção de vários classificadores fracos, isto é com baixos níveis de certeza, para formar um forte.

A Figura 20 ilustra o fluxo de informação no método proposto por Vitor et al. (2014). Devido a sua complexidade, com uso de múltiplos algoritmos para processamento de imagem com realização de transformadas, filtros e construção de mapas de significado, a técnica tem a pior performance em taxas de processamento das estudadas, demorando em média 150 segundos para processar um frame, mesmo rodando em uma plataforma com arquitetura x86 de 8 núcleos.

Os resultados da qualidade de segmentação podem ser vistos na Tabela 3, com exemplo ilustrado na Figura 21. O algoritmo mantém os bons valores de sensitividade observados nos métodos anteriores, porém a pequena queda na precisão e, principalmente, a seu altíssimo custo computacional contam como fortes pontos negativos. O método contudo mostra como pode-se usar a combinação de técnicas diferentes para a obtenção de resultados precisos.

Figura 20 – Diagrama de blocos mostrando fluxo de funcionamento do algoritmo proposto em (VITOR et al., 2014)



Fonte: VITOR et al. 2014

Tabela 3 – Resultados de qualidade de segmentação do método proposto por VITOR et al. (2014) para o dataset KITTI

MaxF1	PM	P	S
87.48 %	80.13 %	85.02 %	90.09 %

Fonte: The KITTI Vision Benchmark Suite

Figura 21 – Resultado de segmentação em imagem do dataset KITTI pelo algoritmo proposto em (VITOR et al., 2014)

Fonte: VITOR et al. 2014

Ainda no conjunto de algoritmos que utilizam de aprendizagem de máquina, a tese de doutorado de GHEORGHE 2015 propõe um método para segmentação semântica de terrenos e estradas usando classificadores estatísticos do tipo Conditional Random Field (CRF), uma técnica de aprendizagem de máquina na qual o modelo discriminativo é construído de tal forma que a predição é realizada baseada na informação contextual ou nos estados dos elementos na

vizinhança. Da mesma forma que os demais trabalhos citados até aqui, este trabalho também utilizou-se da construção de *superpixeis* com informações advindas tanto de características da imagem quanto dos histogramas que compõem os mapas *u e v disparidade*.

Os resultados de qualidade de segmentação para o algoritmo são mostrados na Tabela 4. A Figura 22 mostra um exemplo de segmentação de estrada usando este método.

Figura 22 – Resultado de segmentação em imagem do dataset KITTI pelo algoritmo proposto em (GHEORGHE, 2015)

Fonte: GHEORGHE 2015

Tabela 4 – Resultados de qualidade de segmentação do método proposto por GHEORGHE (2015) para o dataset KITTI

MaxF1	PM	P	S
83.73 %	72.89 %	82.13 %	85.39 %

Fonte: The KITTI Vision Benchmark Suite

O método foi implementado usando C/C++ em processador de 8 núcleos. O tempo de execução é de 5 segundos por frame.

3.1.2 Técnicas sem aprendizagem de máquina

A técnicas discutidas até então tem como principal característica o uso de algoritmos de aprendizagem de máquina para realizar a segmentação de pista, usando parâmetros de tridimensionalidade e características da imagem para gerar as entradas que alimentam estes algoritmos. No geral, estes métodos são inerentemente custosos e precisam de um grande conjunto de imagens para realizar um treinamento e ajustar o seu modelo. Os próximos trabalhos a serem discutidos são puramente algébricos, apresentando desempenhos melhores e tendo a vantagem de não necessitar de um treinamento para ajuste do modelo da região a ser segmentada.

Uma destas técnicas é o trabalho proposto por ZHANG et al. (2018b). Apesar de não estar inserido na página de resultados do site do KITTI, os resultados deste método se mostraram melhores que os demais listados para segmentação de imagens estéreo. Este algoritmo se baseia no uso do mapa de disparidades para, por meio da geração dos mapas *u- e v- disparidade* obter

uma estimativa inicial para a região de chão e calcular a linha do horizonte. A linha do horizonte é então usada como uma região reduzida para estimar o ponto de fuga da imagem, um conceito da geometria descritiva que é o ponto no horizonte para o qual as retas convergem (GRAY, 2009), conforme ilustrado na Figura 23.

Figura 23 – Representação da ideia de ponto de fuga

Fonte: GRAY (2009)

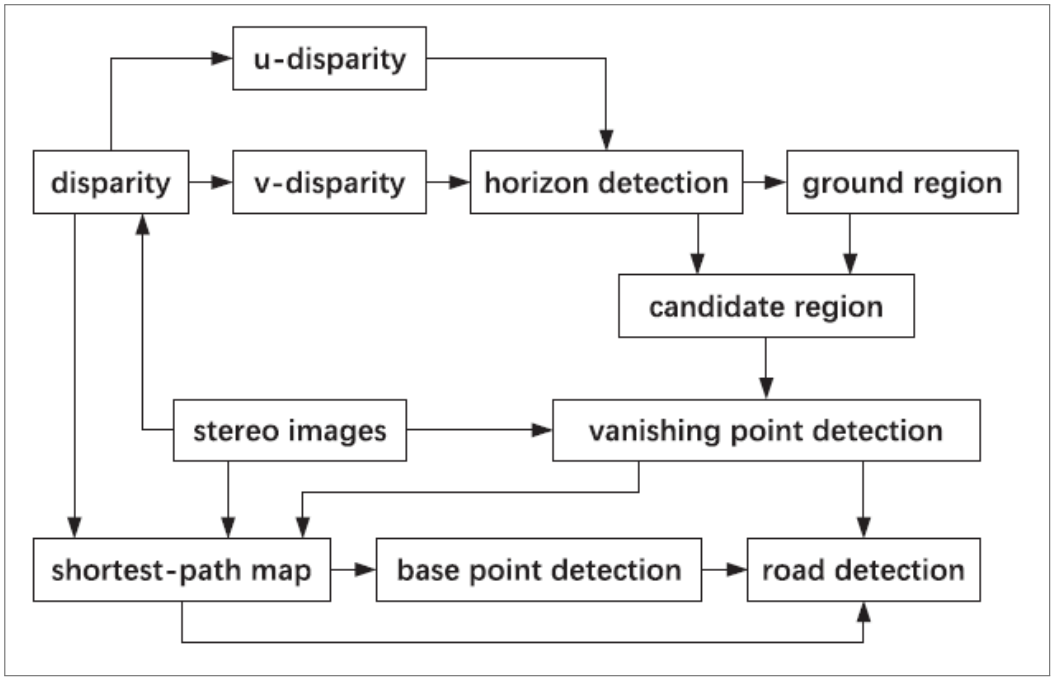
O método então extrai informações da imagens para, combinado com informações de disparidade, calcular 6 funções de custo entre dois pixeis vizinhos: custo de gradiente, custo de orientação, custo de planitude, custo de disparidade e custo de direção de gradiente. Os valores são usados para modelar a imagem como um grafo no qual os pixeis são os nós e para calcular dois pontos bases que são os locais na imagem em que as fronteiras da estrada tocam o agente com as câmeras. Finalmente, aplica-se o algoritmo de Dijkstra (CORMEN et al., 2009) para para calcular os dois caminhos de menor custo entre o ponto de fuga e os pontos base, estes caminhos coincidem com os limitadores da região que se deseja segmentar. A Figura 24 ilustra o fluxograma do método da geração do mapa de disparidades até a detecção da região de estrada.

A Figura 25 mostra vários resultados de segmentação para imagens usando a técnica proposta.

Como previamente anunciado, este método obteve resultados gerais melhores, tanto em termos de qualidade de segmentação quanto em termos de desempenho, para segmentação que as técnicas previamente citadas nessa seção. A implementação foi realizada em linguagem C++ sendo executada em uma CPU Intel Core i5-6500 (4 núcleos), o tempo de execução do método foi 0,25s por frame. Os resultados de qualidade são detalhados na Tabela 5.

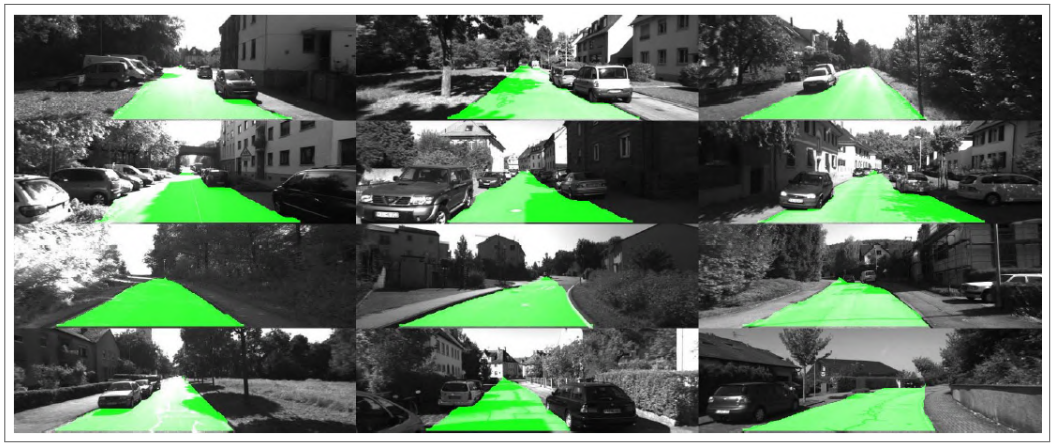
Outra técnica interessante é o método de segmentação proposto por Joos (2011). Nesta

Figura 24 – Diagrama de blocos com fluxo de funcionamento do algoritmo proposto por ZHANG et al. (2018b)



Fonte: ZHANG et al. (2018b)

Figura 25 – Resultados de segmentação em imagens do dataset KITTI pelo algoritmo proposto em (ZHANG et al., 2018b)



Fonte: ZHANG et al. 2018b

Tabela 5 – Resultados de qualidade de segmentação do método proposto por ZHANG et al. (2018b) para o dataset KITTI

MaxF1	PM	P	S
92,02%	Não Informado	91,84%	92,21%

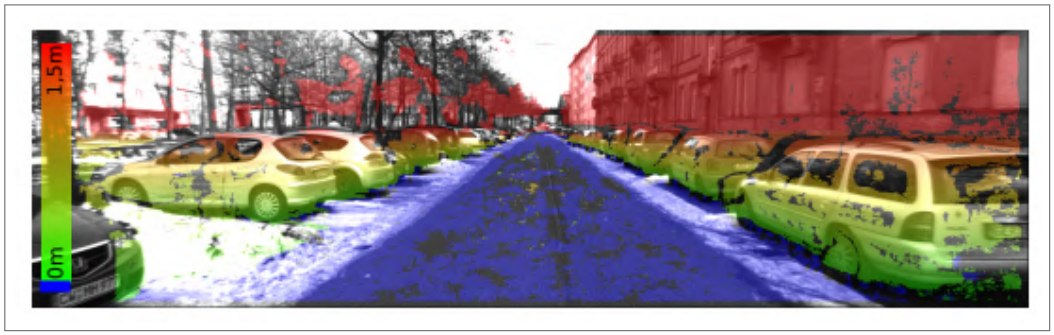
Fonte: ZHANG et al. (2018b)

tese, o autor propõe um algoritmo não somente para detecção em tempo real de pista e espaço livre mas também o rastreamento da pista em uma sequência de *frames* advinda de entrada

de vídeo.

Para a segmentação da pista, o autor também usa a ideia de cálculo de *v-disparidade* otimizado por filtragem de obstáculos por meio do *u-disparidade*. O diferencial da proposta em relação às técnicas tradicionais de modelagem do perfil da pista é que são usadas funções *B-Spline* (WEDEL et al., 2008) para modelar o perfil no mapa *v-disparidade*, de modo a obter uma aproximação muito superior para o modelo da pista, visto que esta função permite detectar de forma precisa não conformidades como relevos e declives existentes no espaço livre.

Figura 26 – Resultados de segmentação alcançados pelo trabalho de JOOS 2011. O sistema foi testado em datasets próprios fechados



Fonte: JOOS 2011

Outra contribuição deste trabalho foi a implementação eficiente que explorou paralelismos de *thread*, de modo a obter tempo melhores do que as técnicas até então mencionadas, processando imagens de resolução 1344 x 373, superiores às do dataset KITTI, em um tempo médio de 94 ms, usando um *cluster* com 2 processadores *Intel Xeon E5410* de 4 núcleos.

Um ponto fraco do trabalho é que não foram feitas análises quantitativas da qualidade de segmentação, sendo os resultados avaliados apenas por inspeção visual das imagens segmentadas.

Em termos de desempenho é importante destacar a arquitetura mista com Hardware e Software proposta por KAKEGAWA et al. (2018) para segmentação de espaço livre em tempo real. O trabalho utiliza da combinação dos mapas *u- e v-disparidade* com um terceiro tipo de mapa construído a partir do cálculo de histogramas de disparidades na região de vizinhança de um pixel de referência para encontrar a fronteira entre a superfície e os obstáculos, removendo-os de maneira mais efetiva do cenário. A região que permanece é considerada a superfície de interesse. Os resultados da qualidade de segmentação podem ser vistos na Figura 27.

Infelizmente, a publicação fornece poucos detalhes sobre a implementação do hardware, limitando-se a mencionar que foi utilizada FPGA para acelerar o processamento. Segundo o artigo os resultados de desempenho utilizando uma plataforma com processador ARM e FPGA no mesmo chip foram um tempo de processamento de 1 ms por *frame*, para imagens de baixa resolução (100 x 240).

Os pontos fracos do trabalho de KAKEGAWA et al. (2018) foram a falta de detalhamento sobre a arquitetura proposta e o uso de imagens de baixa resolução, destoando dos *datasets*

públicos existentes.

Figura 27 – Exemplo de segmentação de região de espaço livre com o método proposto por KAKEGAWA et al. (2018)

Fonte: KAKEGAWA et al. 2018

3.2 DISCUSSÕES

Este capítulo apresentou alguns trabalhos relevantes no contexto de segmentação de espaço livre em imagens estéreo de modo a reforçar as motivações do trabalho e fundamentar critérios para comparação. A maioria destes trabalhos foi avaliado pelos *benchmarks* do dataset KITTI, que é melhor detalhado e tem as métricas explicadas na seção 5.4 do capítulo 5. A tabela 6 mostra um comparativo dos trabalhos apresentados nesta seção que avaliaram seus resultados pelo KITTI.

Dentre estes algoritmos, percebe-se que o que teve melhor desempenho em todas as métricas de avaliação foi o trabalho de Zhang et al. (2018b), destacando o fato de se tratar de um algoritmo puramente algébrico, que não usa técnicas estatísticas ou de aprendizagem de máquina e, portanto, não necessita passar por uma etapa prévia de treinamento. Outra abordagem que também se destacou neste comparativo é o trabalho proposto por WANG et al. (2016). O método foi implementado em uma plataforma com apenas um núcleo de processamento e ainda assim obteve um dos melhores resultados de qualidade de segmentação, mantendo desempenho temporal similar a outros métodos.

Considerando os resultados das métricas de desempenho de processamento (FPS), a compilação de resultados é mostrada na Tabela 7. Inicialmente, para o contexto deste trabalho, pode-se descartar o algoritmo de VITOR et al. 2014, cujo o custo computacional afasta qualquer possibilidade de aplicação em ambiente embarcado.

O sistema descrito por Kakegawa et al. (2018) se mostrou muito promissor pois, apesar de usar imagens com resolução cerca de 15 vezes menor, atingiu taxas quase 1000 vezes maior que a média das demais técnicas avaliadas. Sugere-se o estudo mais aprofundado do sistema proposto por este autor, com uso em outros cenários, para avaliar se há perdas significativas

Tabela 6 – Comparativo dos trabalhos relacionados quando submetidos ao *benchmark* do dataset KITTI

Método	P	S	PM	Max F1	t	Ambiente
NNP (CHEN et al., 2015)	91,43%	89,59%	87,95%	90,50%	5 s	CPU 4 cores @ 2,5 GHz (Matlab)
BMCF (WANG et al., 2016)	88,31%	90,55%	83,13%	89,42%	2,5 s	CPU 1 core @ 2,5 GHz (C/C++)
ProbBoost (VITOR et al., 2014)	85,02%	90,09%	80,13%	87,48%	150 s	CPU 8 cores @3,0 GHz (C/C++)
SCRFFPFHGSP (GHEORGHE, 2015)	82,13%	85,39%	72,89%	83,73%	5 s	CPU 8 cores @ 2,5 GHz (Matlab)
VP Dijkstra (ZHANG et al., 2018b)	91,84%	92,21%	NI	92,02%	0,25 s	CPU 4 cores @ 2,8 GHz (C/C++)

Fonte: KITTI, com entradas adicionadas pelo Autor

em termos de qualidade de segmentação. O algoritmo usado para segmentação descrito em (JOOS, 2011) também pode teve bom desempenho especialmente considerando o uso de uma imagem de resolução maior que as imagens do *dataset* KITTI, seria importante verificar o desempenho desta técnica em plataformas com menor capacidade computacional.

Tabela 7 – Comparativo dos trabalhos relacionados em termos de performance

Método	**Resolução Frame**	**Desempenho**	**Ambiente**
NNP (CHEN et al., 2015)	375 x 1242	0,2 FPS	CPU 4 cores @ 2,5 GHz (Matlab)
BMCF (WANG et al., 2016)	375 x 1242	0,4 FPS	CPU 1 core @ 2,5 GHz (C/C++)
ProbBoost (VITOR et al., 2014)	375 x 1242	0,006 FPS	CPU 8 cores @3,0 GHz (C/C++)
SCRFFPFHGSP (GHEORGHE, 2015)	375 x 1242	0,2 FPS	CPU 8 cores @2,5 GHz (Matlab)
VP Dijkstra citeUVdispZhang2018	375 x 1242	4 FPS	CPU 4 cores @ 2,8 GHz (Matlab)
BSplines (JOOS, 2011)	373 x 1344	10,6 FPS	CPU 8 cores @3,0 GHz (C/C++)
VLDHist FPGA (KAKEGAWA et al., 2018)	100 x 240	1000 FPS	CPU 1 core (C/C++) FPGA (VHDL)

Fonte: O Autor

4 ARQUITETURA PROPOSTA

Este capítulo descreve o sistema proposto por este trabalho. A seção 4.1 detalha o algoritmo proposto baseado em técnicas da literatura modificadas para o fluxo de implementação em FPGA. A seção 4.2 descreve a arquitetura de hardware desenvolvida, explanando o funcionamento de seus submódulos.

4.1 ALGORITMO PARA SEGMENTAÇÃO DE ESPAÇO LIVRE

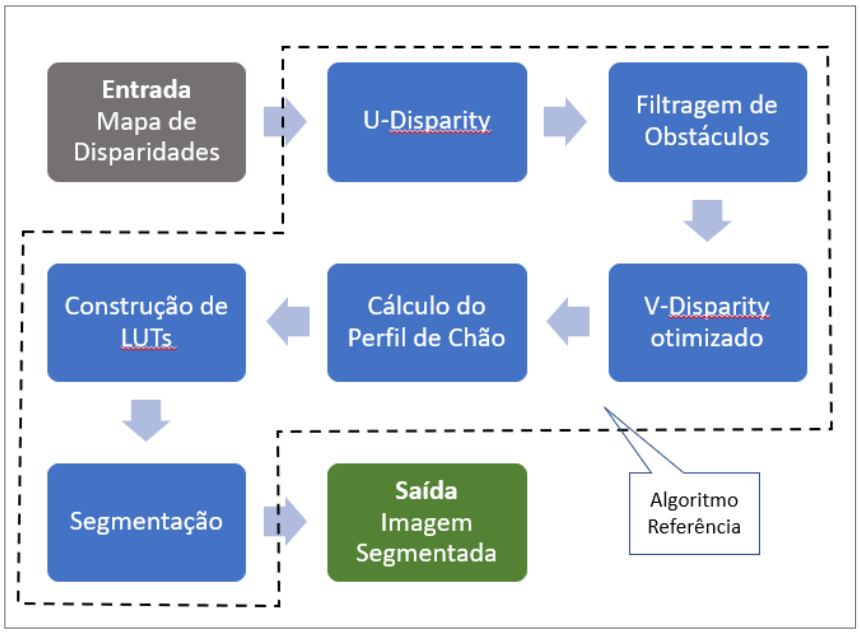
4.1.1 Visão Geral do algoritmo proposto

O algoritmo proposto explora o conceitos introduzidos no capítulo 2 de utilizar os mapas u e v-disparidade para extrair os parâmetros de um modelo que represente a superfície que do espaço livre no plano tridimensional. A partir deste modelo consegue-se determinar se determinado pixel pertence ou não a região de interesse a ser segmentada.

O método proposto neste capítulo entretanto, utiliza de algumas técnicas para obter uma representação mais fiel da região a ser segmentada e foi concebido de forma que seu fluxo facilitasse o desenvolvimento de uma arquitetura em FPGA.

A Figura 28 mostra o fluxo da informação no algoritmo. A partir de um mapa de disparidades, calcula-se o mapa *U-disparidade*. Em seguida o *U-disparidade* é usado para filtrar os obstáculos, resultando em um mapa de disparidades filtrado.

Figura 28 – Diagrama de blocos mostrando o funcionamento do algoritmo de segmentação proposto



Fonte: O Autor

O mapa filtrado é usado em um algoritmo para calcular uma versão melhorada do mapa *v-disparidade*. Utiliza-se um modelo de função linear definida por partes para representar o perfil do chão no mapa *v-disparidade*, os parâmetros desta função são calculados e então ela é usada para construir as *Look-up Tables* usadas para segmentação. Para determinar se um pixel de disparidade pertence pertence à região segmentada, busca-se o valor da variável v para a sua disparidade na *Look-up Table* e compara-se com o valor o seu valor (a linha do pixel). Se estiver dentro de um limiar de tolerância, o algoritmo classifica o pixel como região de chão, retornando o valor um, caso contrário o algoritmo retorna o valor zero.

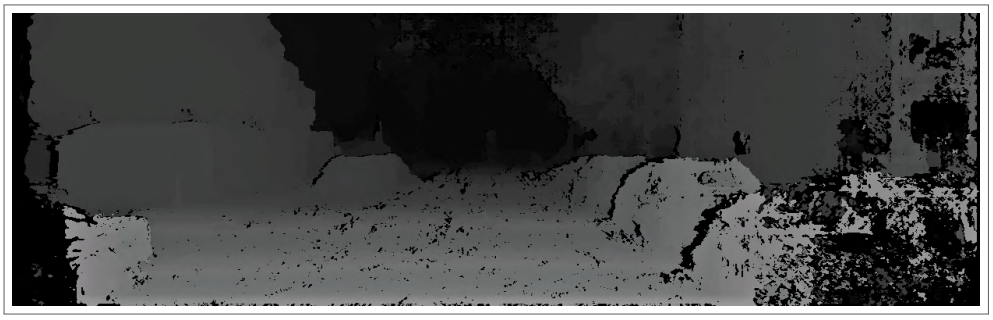
4.1.2 Mapa V-disparidade com filtragem de obstáculos

De modo a se ter um melhor entendimento do algoritmo considere a imagem esquerda de um par conjugado estéreo do *dataset* KITTI mostrada na Figura 29 e o mapa de disparidade correspondente mostrado na Figura 30.

Figura 29 – Imagem de cenário automotivo do *dataset KITTI*

Fonte: GEIGER; LENZ; URTASUN 2012

Figura 30 – Mapa de disparidades gerado a partir do conjugado estéreo relacionado ao cenário mostrado na Figura 29

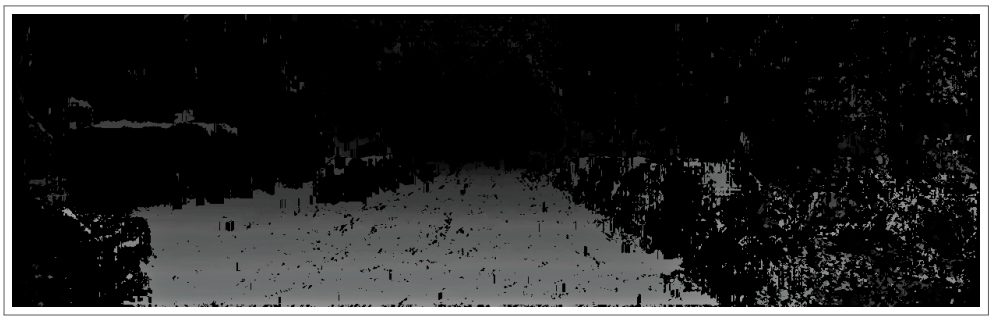


Fonte: O Autor

Como mencionado na seção 2.3 do capítulo 2, os obstáculos são representados no domínio (Δ, v) como segmentos de reta verticalizados, estes segmentos são informações ruidosas que prejudicam a extração do perfil do chão. Uma forma de atenuar este problema é realizar a filtragem de obstáculos por meio do cálculo do mapa *u-disparidade* seguido de uma operação

de *thresholding*. A Figura 31 mostra o mapa de disparidades da Figura 30 com os obstáculos removidos.

Figura 31 – Mapa de disparidades gerado a partir da Figura após a remoção de obstáculos por meio do mapa u-disparidade 29



Fonte: O Autor

Com a filtragem de obstáculos, o mapa *v-disparidade* é calculado com um resultado bem menos ruidoso. Faz-se também um procedimento de binarização deste mapa, mantendo apenas o ponto mais significativo de cada coluna.

A Figura 32 mostra melhorias em um mapa *v-disparidade* após a aplicação destes passos.

Figura 32 – Processo de cálculo do mapa v-disparidade por meio de mapa de disparidades com obstáculos filtrados. Em (a) o mapa calculado sem a filtragem, em (b) o mapa com filtragem e em (c) o mapa após o processo de binarização.

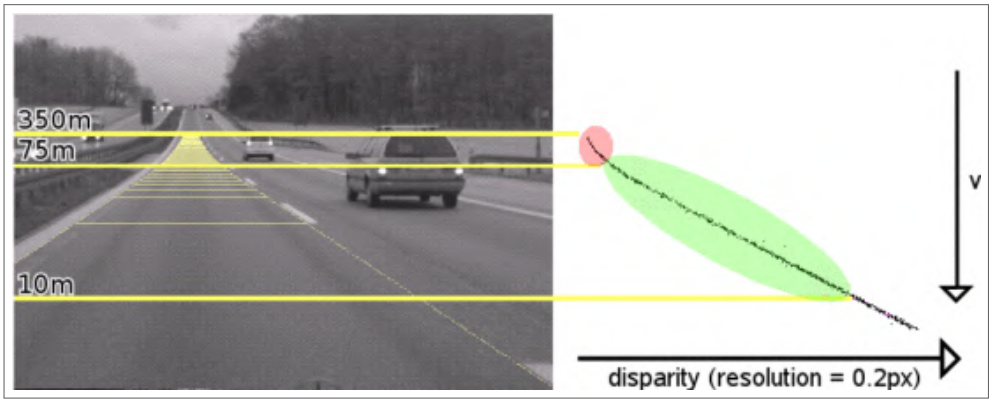
Fonte: O Autor

4.1.3 Cálculo de perfil de chão por função linear definida por partes

Na seção 2.3 do capítulo 2, assumiu-se que o chão consistiria em uma região contida em um plano paralelo a linha base do conjugado estéreo.

Esta aproximação, naturalmente, é muito simplista para generalizar cenários mais reais de aplicações em robótica. Terrenos, pisos, estradas e caminhos muitas vezes apresentam características não planas. Nestes casos, uma abordagem mais correta seria considerar que o espaço livre seria uma região contida não em um plano, mas em uma superfície no mapa de disparidades. Esta superfície, seria representada no domínio $((\Delta, v))$ como uma curva, como ilustrado na Figura 33.

Figura 33 – Cenário de pista não plana e mapa v-disparidades correspondente.



Fonte: WEDEL et al. 2008

Na literatura há propostas disponíveis para modelar esta função curvilínea, como funções polinomiais (SAPPA et al., 2007) ou funções do tipo *spline* e *b-spline* (WEDEL et al., 2008). O ajuste destas curvas é, naturalmente, mais complicado do que o cálculo de coeficientes de uma reta, podendo envolver a necessidade do cálculo de matriz inversa. Esta característica aumenta consideravelmente a complexidade do hardware.

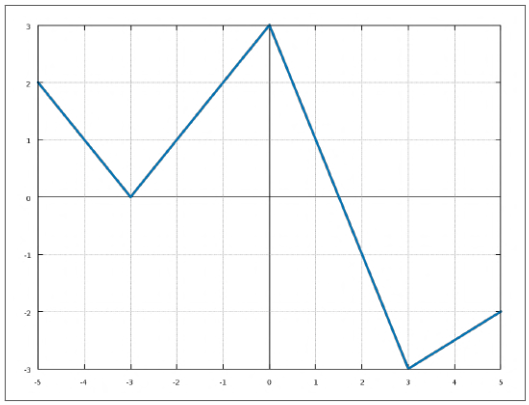
Uma alternativa para conseguir representar superfícies não-planas e não aumentar muito a complexidade da arquitetura de hardware, proposta neste trabalho, é modelar o perfil do chão como uma função linear definida por partes.

Uma função linear definida por partes é uma função de domínio e co-domínio reais definida nos intervalos por diferentes retas. A Figura 34 ilustra o gráfico de uma função deste tipo. Para as aplicações deste trabalho, ainda, as funções são definidas e contínuas em todo o intervalo.

Para calcular a função, divide-se o domínio Δ do mapa *v-disparidade* em partes, doravante chamadas pelo termo *splits*. Para a primeiro segmento, parte-se de um ponto no meio e são geradas várias retas ligando-o a outros pontos no segmento, o número de retas geradas é um parâmetro do método. Esta ideia está ilustrada na Figura 35.

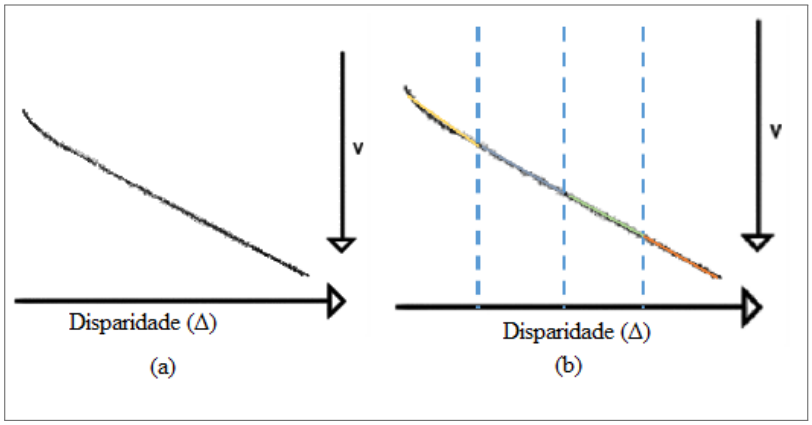
A reta que tiver o menor erro total em relação a todos os pontos no segmento é escolhida. Para a próxima reta, para cumprir o critério de continuidade da função, o ponto inicial é aquele

Figura 34 – Exemplo de função linear contínua definida por partes



Fonte: Creative commons

Figura 35 – Exemplo de modelagem de perfil de pista usando função linear definida por partes. Em (a) o perfil mostrado na Figura 33 de uma superfície livre não plana. Em (b) a modelagem do perfil usando 4 segmentos de retas.



Fonte: O Autor

no qual a reta anterior cruza a fronteira de *splits*. Repete-se este procedimento até que tenham se ajustado todos os splits definidos. O número de splits também é um parâmetro do algoritmo, experimentalmente, para problemas com disparidade máxima de 64 e 128, verificou-se que o ideal é usar, no máximo 4 splits.

4.1.4 Segmentação de espaço livre otimizada por detecção de horizonte

Estando o perfil da região de interesse modelada matematicamente e com os parâmetros extraídos, é possível realizar a etapa final da segmentação de espaço livre.

A segmentação é feita comparando o par (Δ, v) do mapa de disparidades com os valores da função perfil calculada anteriormente, isto é, dada um pixel no mapa de disparidades verifica-se se a linha na qual ele está contido está dentro de um intervalo aceitável quando comparado com a linha(valor de v) da disparidade correspondente no mapa *v-disparidade*. Este

comportamento é descrito pela Equação 4.1.

$$Res = \begin{cases} 1, \text{ se } |v_{pixel} - v_{function}| < \alpha * (v_{pixel} - hor) \\ 0, \text{caso contrário.} \end{cases} \tag{4.1}$$

Para melhoria no desempenho do processamento tanto em termos de precisão quanto desempenho, pode-se usar o cálculo da linha do horizonte para diminuir o espaço relevante na execução do algoritmo.

O cálculo da linha do horizonte foi implementado seguindo a técnica descrita por LABAYRADE; AUBERT; TAREL (2002). De uma maneira intuitiva, esta ideia baseia-se no fato que no horizonte os objetos estão muito distantes e, por isso, a disparidade tende à zero, desta forma o ponto que o perfil da curva toca o eixo das ordenadas coincide com a linha do horizonte.

A Figura 36 ilustra este conceito colocando uma imagem de contexto automotivo ao lado do seu mapa *v-disparidade* filtrado.

Figura 36 – Uso de mapa v-disparidades para encontrar o horizonte da imagem

Fonte: O Autor

Possuindo a informação do horizonte, o espaço de segmentação é reduzido para a disparidade abaixo da linha, visto que busca-se o chão. Esta redução traz duas melhorias para o sistema proposto:

1. Aumenta o desempenho do sistema, visto que só é necessário segmentar a imagem abaixo do horizonte.

2. Aumenta a precisão do sistema, pois é certo que não há chão acima do horizonte.

Feito este aperfeiçoamento, aplica-se a equação 4.1 para cada pixel abaixo da linha do horizonte, os valores marcados com 1 são segmentados, os demais são considerados obstáculos.

O algoritmo descrito foi implementado em linguagem C/C++ usando a biblioteca OpenCV 3.4.8 (BRADSKI, 2000) e usado como modelo de referência. Partindo dele, foi desenvolvida uma arquitetura em FPGA descrita na seção 4.2.

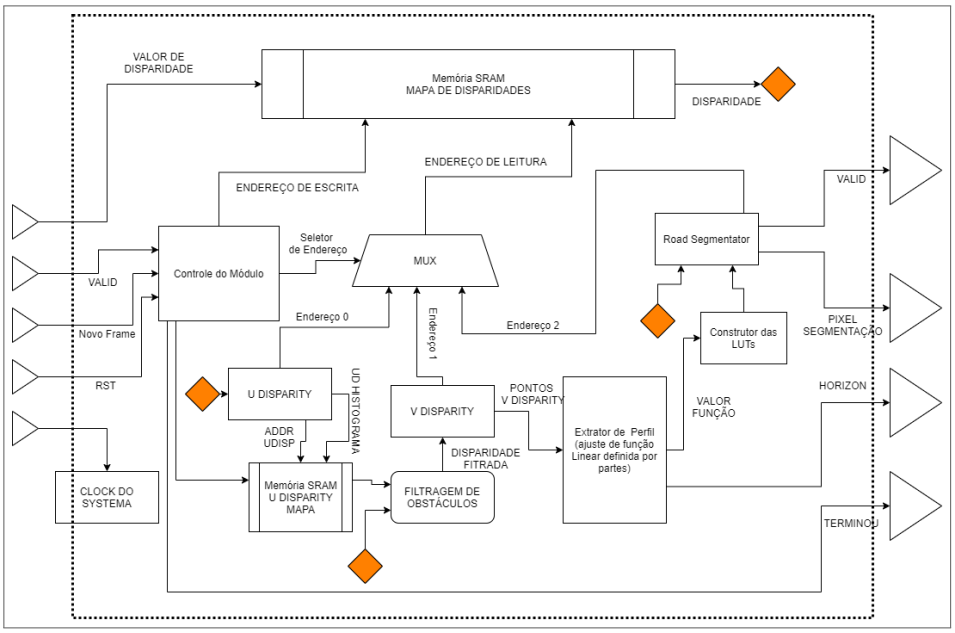
4.2 ARQUITETURA EM FPGA

4.2.1 Visão Geral

O algoritmo de segmentação de pista descrito na seção 4.1 fundamentou o desenvolvimento da arquitetura de hardware descrita neste trabalho.

O sistema consiste em um módulo que recebe pixeis advindos de um mapa de disparidades e indica se eles pertencem ou não à região de pista. A Figura 37 consistem em um diagrama de blocos com os principais módulos e conexões da arquitetura principal proposta para o sistema. Na imagem, os elementos em laranja representam a saída da memória que armazena o mapa de disparidades. É importante ressaltar que na figura não estão presentes os registradores de atraso colocados entre módulos usados para sincronização de sinal.

Figura 37 – Diagrama de blocos representando, de forma simplificada, a arquitetura da entidade *Top Level* do sistema desenvolvido em Hardware



Fonte: O Autor

As entradas e saídas do módulo são descritas mais detalhadamente na Tabela 8.

O controle principal do módulo é feito de forma sequencial por meio de uma Máquina de Estados Finita (MEF) de 6 estados cuja estrutura está ilustrada na Figura 38 e resumida na Tabela 9.

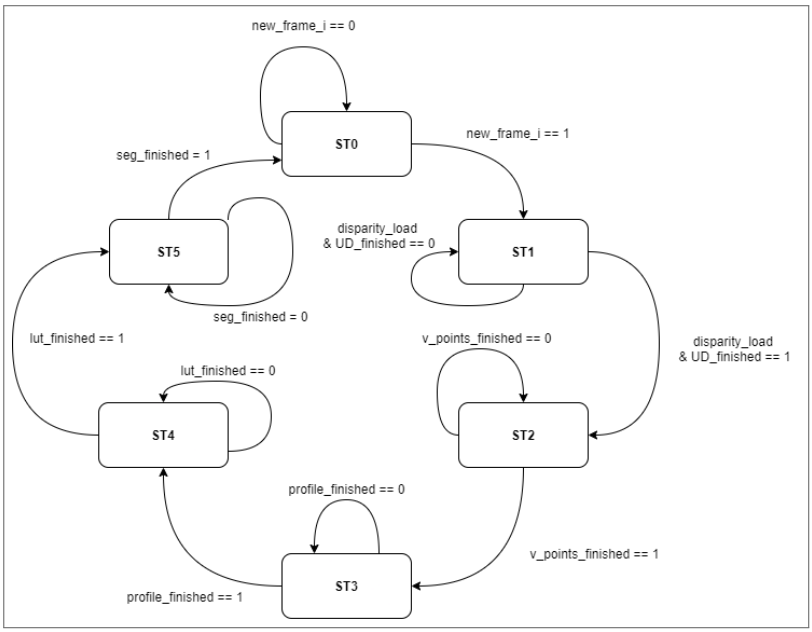
- No estado **St0**, a máquina está em ociosidade. Todos os sinais são configurados para o valor inicial e o processo começa quando é recebido um sinal de chegada de frame.
- No estado **St1**, os pixeis de disparidade que chegam na entrada são salvos em uma memória interna da FPGA e é calculado o mapa *u-disparidade*.

Tabela 8 – Entradas e saídas do módulo para segmentação de espaço livre

Sinal	Tipo	Descrição
clk_i	entrada	clock do sistema
rst_i	entrada	sinal de reset
pixel_i	entrada	pixel de mapa de disparidades
new_frame_i	entrada	sinal que indica a chegada de um novo frame para processamento
validpx_i	entrada	sinal que indica que um valor de pixel de entrada é válido
valid_o	saída	sinal que indica que o valor na saída é válido
is_road_o	saída	sinal que indica que um pixel pertence ou não a região de interesse
horizon_o	saída	coordenada em v da linha do horizonte
framefinished_o	saída	sinal que indica o término do processamento de um frame.

Fonte: O Autor

Figura 38 – Máquina de estados da entidade superior da arquitetura proposta



Fonte: O Autor

- No estado **St2**, é calculado o mapa v-disparidade aperfeiçoado, a filtragem de disparidades é feita no próprio módulo superior, antes do pixel ser passado para o módulo que calcula o v-disparidade, acessando o valor correspondente no histograma do mapa u-disparidades da coluna lida no momento.

Tabela 9 – Detalhamento da máquina de estados finita responsável pelo controle do sistema de segmentação proposto

Estado	Nome	Descrição	Próximo Estado	Condição de Transição
St0	Idle	Aguarda a chegada de um frame	St1	new_frame_i $==$ 1
St1	Udisp	Salva disparidade na memória SRAM e constrói mapa u-disparity	St2	disparity_load $==$ 1 e UD_finished $==$ 1
St2	Vdisp	Constrói mapa v-disparity aperfeiçoado	St3	v_points_finished $==$ 1
St3	Fitting	Ajusta função de linear definida por partes	St4	profile_finished $==$ 1
St4	Lut	Constroi LUTs	St5	lut_finished $==$ 1
St5	Seg	Realiza a segmentação do chão	St0	seg_finished $==$ 1

Fonte: O Autor

- O estado **St3** é no qual é feito, no módulo extrator de perfil, o ajuste dos segmentos de reta que definem o perfil do espaço livre. Os dados do mapa *v-disparidade* binarizado são passados para este módulo para ajuste das curvas.
- Com a função linear definida por partes calculada, para cada valor de disparidade é calculado o valor correspondente da coordenada v que representaria o perfil da pista, este valor é salvo em look-up-tables. A construção dessas LUTs é feita no estado **St4**, no módulo construtor de LUTs, neste estado é salvo também o valor do horizonte.
- Para finalizar, no estado **St5** é feita, no módulo *Road Segmentator* a segmentação da pista a partir da função de perfil.

4.2.2 Módulo para cálculo do mapa u-disparidade

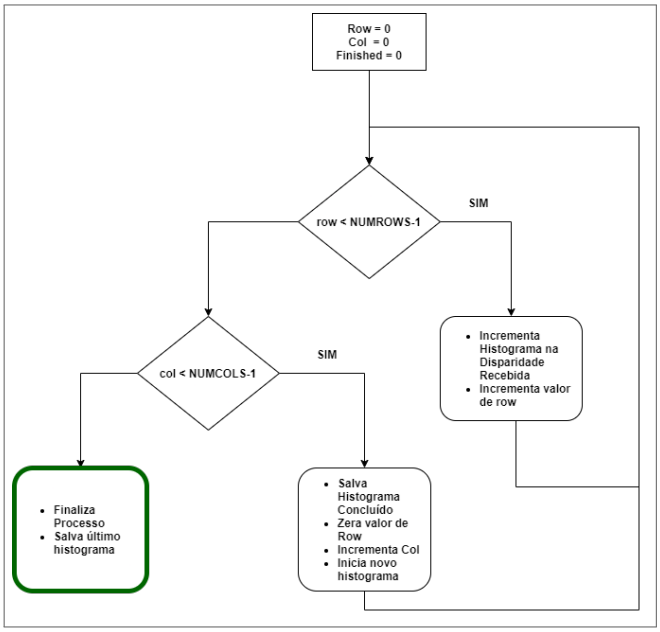
No estado **St1**, assim que se inicia o processo de salvar a disparidade em memória, é iniciada a construção do mapa *u-disparidade*. O fluxo do módulo é relativamente simples, estando mostrado na Figura 39

Como a memória de disparidades é preenchida por colunas, o endereço da memória correspondente a um pixel de disparidade de coordenadas (row, col) é dado pela equação 4.2

$$Addr = col * \mathrm{NUMROWs} + row; \tag{4.2}$$

Para a construção do *u-disparidade* o mapa de disparidades é percorrido por coluna. Leva-se em conta o atraso de um ciclo para o endereço ser calculado e um ciclo para o dado ficar

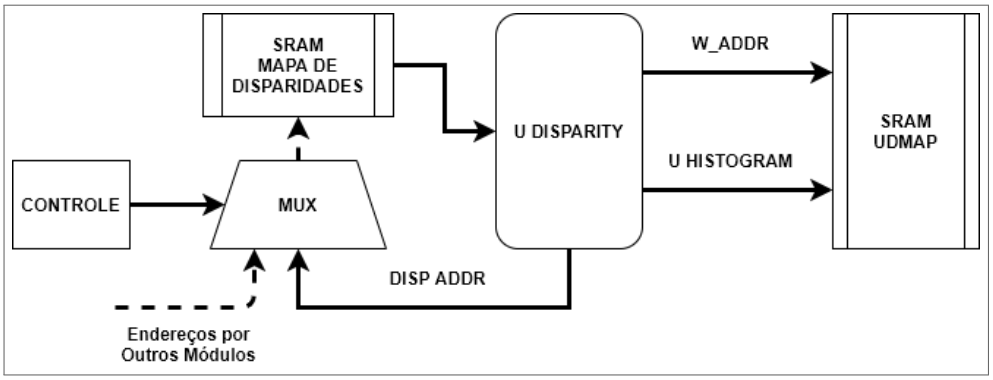
Figura 39 – Fluxograma de controle do módulo responsável pelo cálculo dos mapas u-disparidade



Fonte: O Autor

disponível, considerando esta latência são calculados os histogramas que compõem o mapa[1]. O resultado é salvo em uma memória de *u-disparidades* em que cada palavra corresponde a um histograma. Os valores de cada *slot* do histograma são aglutinados, por meio de deslocamentos binários e somas, para a construção da palavra. A Figura 40 ilustra o cálculo de u-disparidade na arquitetura proposta.

Figura 40 – Diagrama de blocos que destaca o papel do módulo u-disparity na arquitetura



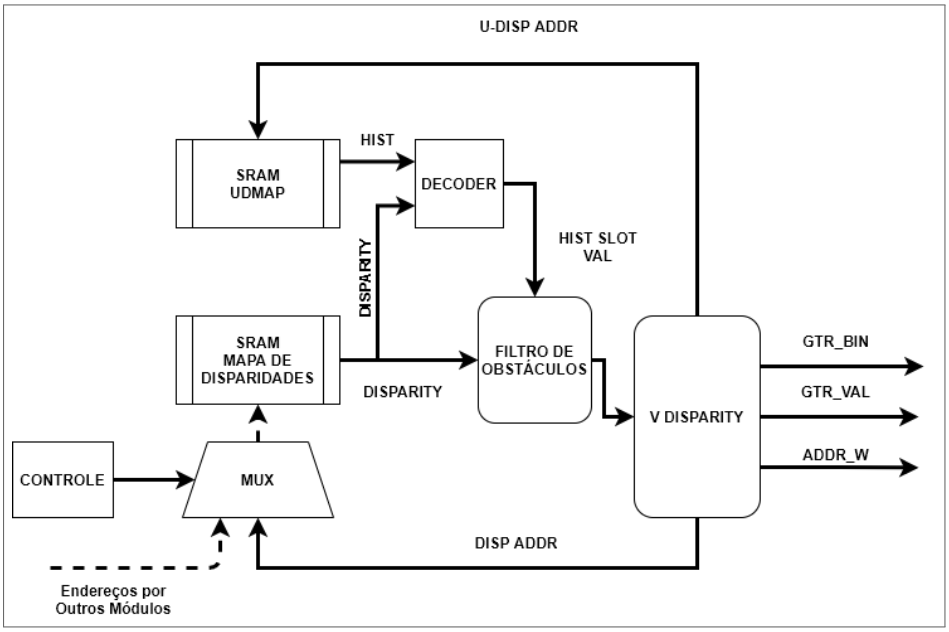
Fonte: O Autor

[1] Os histogramas são incrementados a medida que os pixeis de disparidades são recebidos na entrada, adicionando uma unidade no *splot* correspondente

4.2.3 Módulo para cálculo do mapa v-disparidade com filtragem de obstáculos

A primeira característica do módulo *v-disparidade* a ser entendida é o acesso aos dados. O endereço da memória do mapa de disparidades é calculado seguindo a equação 4.2, este cálculo é registrado e é feito em um ciclo de clock. Em seguida, solicita-se paralelamente a informação da disparidade e o histograma u-disparidade nas memórias. Espera-se mais um clock para obter estes dados e usa-se o valor da disparidade para, por meio de uma operação de máscara, encontrar o valor do histograma correspondente à disparidade. Se este valor estiver acima de um limiar parametrizável, considera-se o pixel da memória um obstáculo e é passado um valor inválido para o módulo v-disparidade, caso contrário, o histograma é computado.

Figura 41 – Diagrama de blocos que destaca o papel do módulo v-disparity na arquitetura



Fonte: O Autor

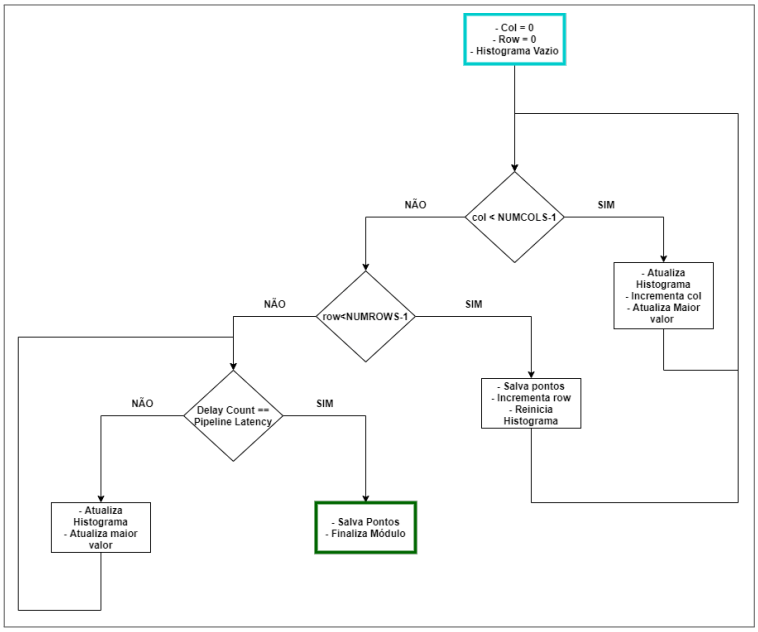
A Figura 41 mostra os módulos e conexões na arquitetura para o cálculo do mapa v-disparidade aperfeiçoado.

O fluxo para variação de coordenadas e cálculo de histogramas no v-disparidade é mostrado na Figura 42. A construção do mapa é feita por linhas, portanto os valores de disparidade são lidos fixando a linha e variando a coluna, de modo a obter o histograma relacionado à coordenada fixada. Sempre que é finalizada uma linha, o maior valor e a coordenada do histograma são salvos e o histograma é reiniciado.

4.2.4 Módulo para ajuste de função linear definida por partes

O módulo para ajuste de função linear é carregado com os resultados do módulo *v-disparidade* binarizado durante o estado **St2** da máquina de estados do controle principal.

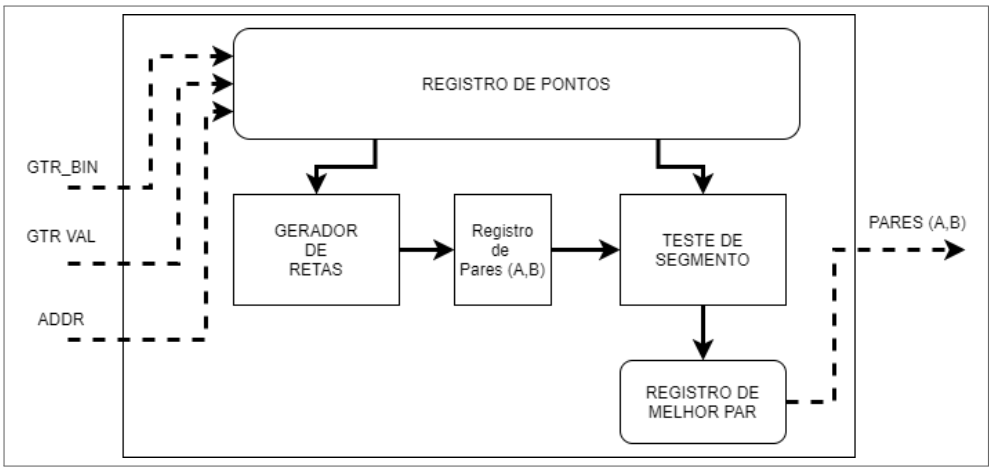
Figura 42 – Diagrama de blocos mostrando o fluxo do funcionamento do módulo que calcula o mapa *v-disparidade*



Fonte: O Autor

No estado **St3** o módulo de ajuste realiza o ajuste da função linear definida por partes. O funcionamento do módulo foi implementado de acordo com o proposto na seção 4.1 e tem a arquitetura mostrada na Figura. 43

Figura 43 – Diagrama de blocos que ilustra a arquitetura interna do módulo para ajuste de função de perfil

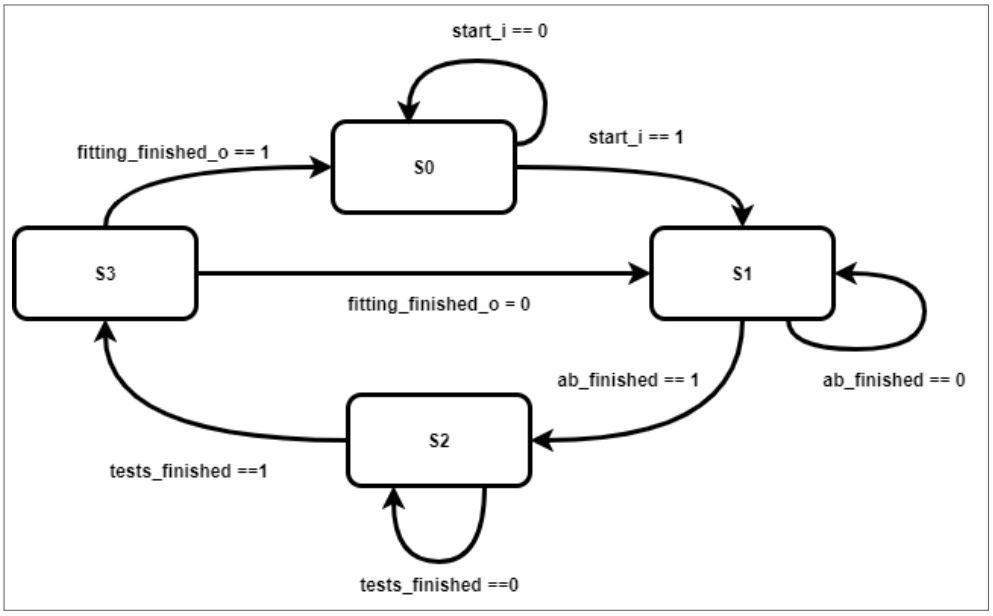


Fonte: O Autor

Este módulo tem o funcionamento intrinsecamente sequencial, visto que para testar retas a mesma tem que ser gerada, para definir a melhor reta, todas tem que ser testadas e o término de um segmento é o ponto inicial do seguinte, portanto, para implementar o controle deste módulo usa-se uma MEF, ilustrada na Figura 44

A descrição dos estados deste módulo é feita na Tabela 10.

Figura 44 – Máquina de estados finitos que controla o funcionamento do módulo que ajusta a função de perfil



Fonte: O Autor

Tabela 10 – Entradas e saídas do módulo para segmentação de espaço livre

Estado	Nome	Descrição	Próximo Estado	Condição de Transição
S0	Idle	Aguarda início do processo de ajuste	S1	start $==$ 1
S1	GenLine	Gera os parâmetros para os candidatos a segmento do split	S2	ab_finished $==$ 1
S2	Testes	Testa os candidatos e define o melhor (a,b)	S3	test_finished $==$ 1
S3	Final	Inicia próximo segmento ou finaliza o processo	S0 ou S1	S0 se fitting_finished $==$ 1, S1 se 0

Fonte: O Autor

No desenvolvimento deste módulo foi necessário realizar operações com ponto flutuante. Para acelerar os desenvolvimento foram utilizados os *IP Cores* da Altera Intel (INTEL, 2021). Os IPs utilizados foram:

- **FP_ADD**: Adicionador de ponto flutuante;
- **FP_SUB**: Subtrator de ponto flutuante;
- **FP_MULT**: Multiplicador de ponto flutuante;

- **FP_DIV**: Divisor de ponto flutuante;
- **FP_COMP**: Comparador de ponto flutuante;
- **FP_ABS**: Valor absoluto de ponto flutuante;
- **FP_INT**: converte o valor de ponto flutuante para inteiro e vice-versa.

4.2.4.1 Geração de candidatos a segmento

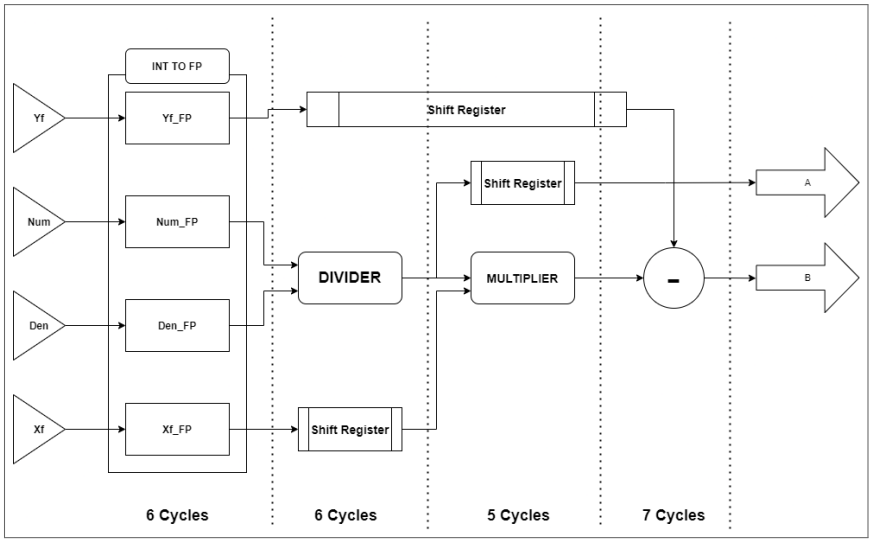
Este módulo consiste na geração dos parâmetros a e b da equação de reta 4.3.

$$v = a \cdot \Delta + b \tag{4.3}$$

Dado um par de pontos (x_1, y_1) e (x_2, y_2), substituindo os valores na equação de reta e solucionando um sistema simples logo temos temos o sistema mostrado em 4.4

$$\begin{cases} a = \frac{y_2 - y_1}{x_2 - x_1} \\ b = y_2 - \frac{y_2 - y_1}{x_2 - x_1} x_2 \end{cases} ; x_2 \neq x_1 \tag{4.4}$$

Figura 45 – Arquitetura do *pipeline* do módulo que computa os parâmetros de uma reta a partir de dois pontos



Fonte: O Autor

A Figura 45 mostra a arquitetura do pipeline usado para gerar todos os candidatos a serem o par (a, b) para o segmento do *split* a ser ajustado.

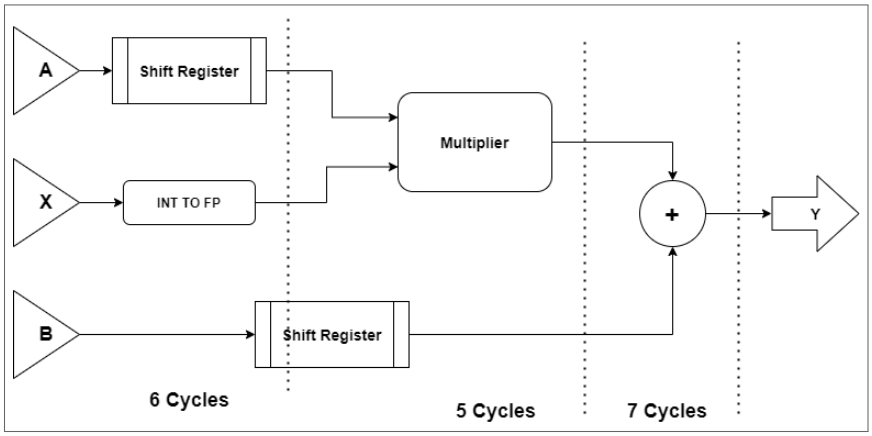
4.2.4.2 Teste de candidatos

O módulo que testa os candidatos implementa as seguintes atividades:

- Calcular o valor da reta candidata a segmento para os pontos existentes no registro de pontos
- Atualizar o erro total atrelado a reta
- Salvar o candidato com o menor erro total

Para calcular o valor da reta foi implementado um módulo para calcular funções lineares, a arquitetura do módulo é mostrada na Figura 46.

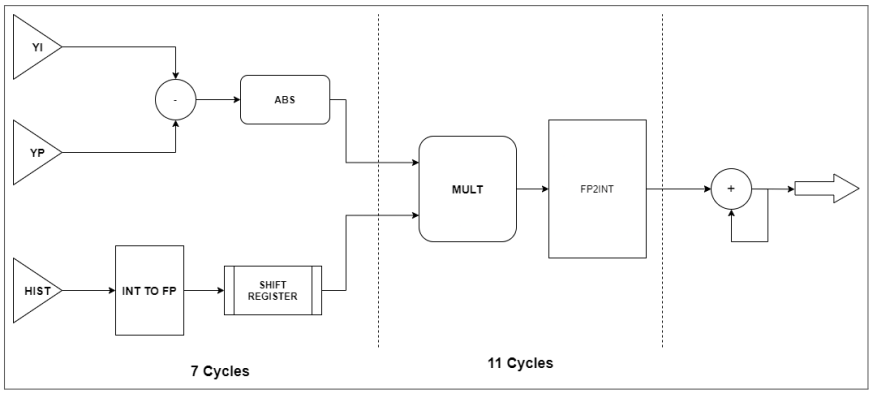
Figura 46 – Módulo desenvolvido para cálculo de função linear



Fonte: O Autor

O módulo acumulador de erro recebe o valor da reta candidata e o ponto de referência e vai atualizando o erro a medida que são calculados novos pontos. Quando vai ser calculado o erro de um novo candidato, o módulo é reiniciado. A Figura 47 mostra a arquitetura deste módulo.

Figura 47 – submódulo responsável por avaliar e acumular o erro de uma dada reta testada



Fonte: O Autor

Para cada segmento que compõe a função de perfil, o módulo salva o par de parâmetros (a, b) com menor erro total nos testes realizados. A escolha da função de erro total em detrimento de erro médio ou erro quadrático médio dar-se para simplificação do hardware do módulo construído para exercer este papel.

4.2.5 Módulo para construção de *luts*

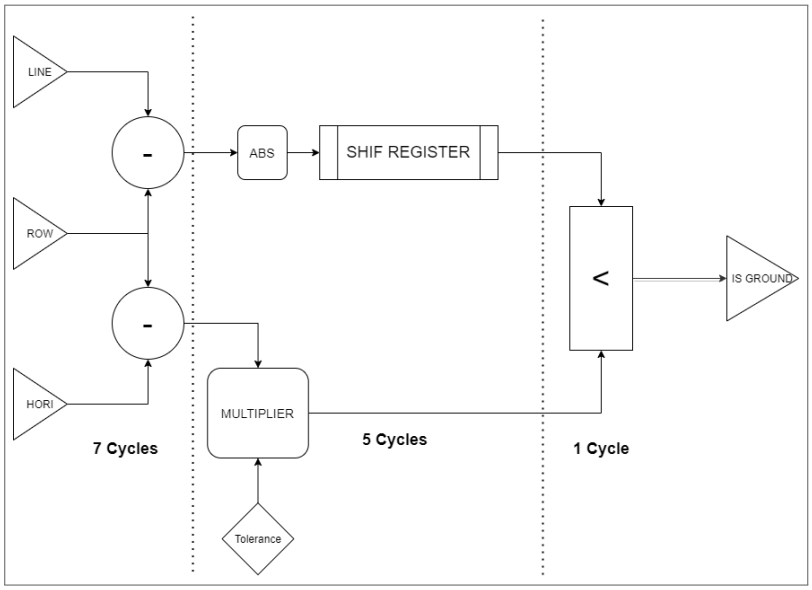
Para construção de LUTs usam-se os parâmetros das retas encontradas no módulo descrito na seção anterior para calcular, por meio de outra instância do módulo que calcula função linear, o valor da reta função previsto para cada disparidade.

A sua saída é convertida novamente para inteiro, que é o formato no qual é feito o processo de segmentação.

4.2.6 Módulo para segmentação de espaço livre

O módulo para segmentação é uma implementação em hardware da equação 4.1. Faz-se acesso a memória de disparidades e compara-se o valor da disparidade com o valor da linha, se ela estiver dentro de um intervalo de tolerância de acordo com o previsto pela função de perfil, as coordenadas do mapa de disparidades correspondentes ao valor testado consistem em uma região de chão. A Figura 48 mostra a arquitetura do módulo implementado.

Figura 48 – Arquitetura em pipeline do módulo responsável pela segmentação do chão a partir de função do perfil



Fonte: O Autor

O módulo realiza somente a segmentação a partir da linha do horizonte, considerando como região fora do interesse os pixeis acima desta. A saída do módulo é enviada diretamente para a saída da entidade superior e, junto com o horizonte, consiste no resultado do hardware desenvolvido neste trabalho.

5 METODOLOGIA DE IMPLEMENTAÇÃO E VALIDAÇÃO

Este capítulo apresenta a metodologia usada para o desenvolvimento deste trabalho, como o módulo proposto foi projetado, implementado e validado. A seção 5.1 explica o método de projeto adotado, a seção 5.2 disserta sobre a metodologia de síntese. Na seção 5.3 são discutidas as plataformas de implementação usadas. Finalmente, a seção 5.4 é discutida a metodologia de validação do sistema, na qual discorre-se também sobre os *datasets utilizados* e as métricas usadas na avaliação.

5.1 METODOLOGIA DE PROJETO

A metodologia de projeto é a forma que se dá o fluxo de trabalho durante o projeto do sistema implementado. Neste trabalho foi utilizada a técnica *especificar, explorar e refinar* (GERSTLAUER et al., 2008). Esta técnica, consolidada nos projetos modernos de sistemas embarcados, pode ser usada para o desenvolvimento de hardware e software e consiste no constante refinamento do projeto por meio de modelos que são adicionados. Sempre que adicionado um modelo, são feitos os ajustes para que o sistema mantenha as funcionalidades e cumpra os requisitos.

A metodologia parte de um modelo comportamental desejado, que, quando aplicável, pode estar acompanhado de um modelo de referência algébrico. A partir deste deste modelo é desenvolvido uma versão em alto nível executável.

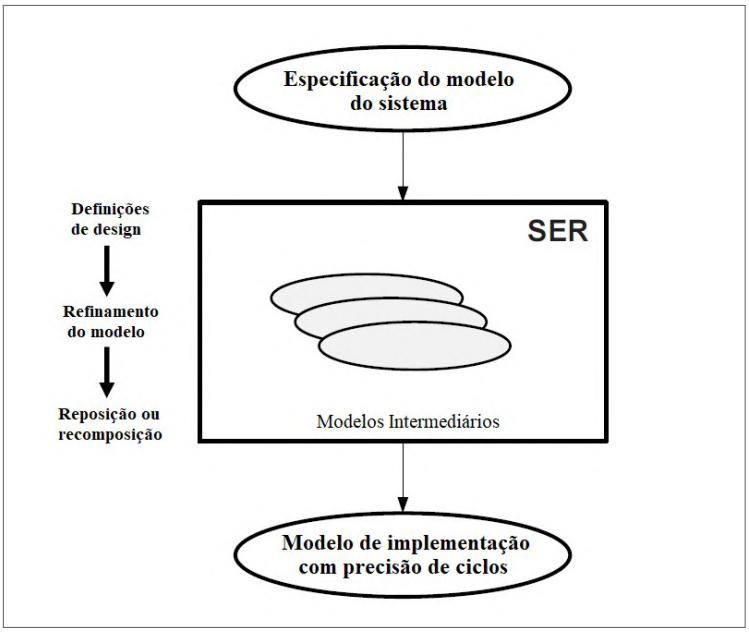
A Figura 49 mostra o fluxo de desenvolvimento pelo método SER. O projeto é realizado por passos incrementais e simuláveis, havendo possibilidade de testar o sistema em diferentes níveis de abstração. Para cada modelo, explora-se as decisões de projeto e refina-se o modelo após a exploração. Após isso, o modelo refinado é usado como especificação para a construção do modelo do próximo nível.

No contexto deste trabalho, partiu-se do comportamento desejado do sistema: Realizar segmentação de chão e espaço livre usando as técnicas cujo modelo algébrico está descrito descrito na seção 2.3 para extração de perfil de chão a partir dos mapas *u e v-disparidade*.

O modelo de alto nível, doravante chamado *modelo de referência*, foi implementado em linguagem C/C++. Este modelo continha todas as funcionalidades do sistema e o processo de refinação foi feito para otimizar o desempenho.

Em seguida, foi implementado um modelo em *systemverilog* para verificação funcional na ferramenta de simulação *Intel Modelsim*. Neste modelo, foi possível verificar se o código em linguagem de descrição de hardware cumpria as funcionalidades requisitadas, além disso, em systemverilog é possível implementar trechos que, apesar de não sintetizáveis, facilitam a verificação, como por exemplo a leitura de estímulos simulacionais de arquivos e a também escrita de resultados intermediário. Esta característica foi essencial para o desenvolvimento de

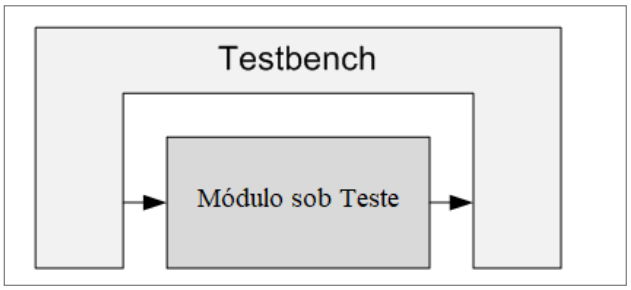
Figura 49 – Fluxo de projeto seguindo a metodologia especificar, explorar e refinar ou SER.



Fonte: GAJSKI et al. (2009) (ADAPTADO)

testbenches.

Figura 50 – Conceito de testbench de um módulo



Fonte: LAND (2014) (ADAPTADO)

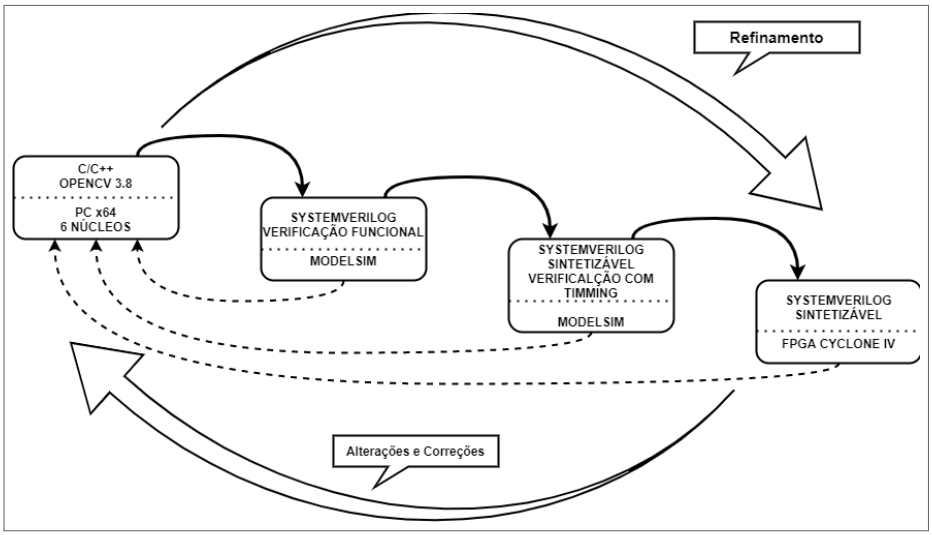
Um *testbench* é um módulo não sintetizável em linguagem de descrição de hardware que fornece estímulos para um módulo sob validação e capta a saída podendo realizar pós processamento para indicar características do sinal, apontar erros e outros. A Figura 50 ilustra o conceito do *testbench*.

Após o modelo em *systemverilog* estar finalizado, a especificação foi refinada de forma a ser sintetizável. Nesta etapa foram feitos os refinamentos necessários para que o sistema cumprisse as restrições de tempo, podendo haver a necessidade de modificar a lógica de operação várias vezes. Após sintetizado, com as restrições de tempo, passou a ser capaz de descrever as funcionalidades com precisão de ciclos.

Finalmente, foi feita a implementação do sistema na plataforma final com prototipação em FPGA. Sempre que era necessário realizar mudanças em determinado modelo foi preciso modificar os modelos de nível mais alto de modo a garantir que as versões estariam coerentes.

A Figura 51 esquematiza o modelo de refinamento usado no projeto deste trabalho. Pode-se entender também que foi utilizada uma metodologia *Top-down*, na qual o refinamento do modelo se deu através da implementação em níveis de cada vez mais baixa abstração. Isto é, partindo do modelo de referência em C++ até o sistema sintetizável, pronto para prototipação em FPGA.

Figura 51 – Método de projeto aplicado ao desenvolvimento do módulo segmentação de espaço livre



Fonte: O Autor

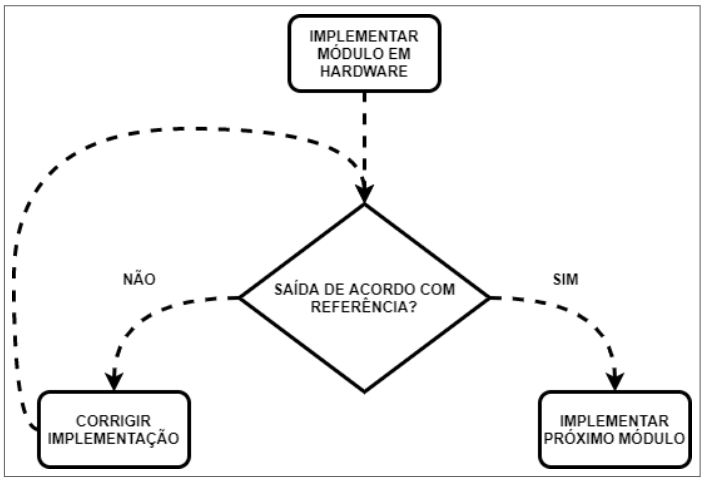
5.2 METODOLOGIA DE SÍNTESE

A metodologia de síntese adotada consistiu na implementação gradual de cada bloco funcional do algoritmo de referência como módulos em systemverilog. Para validação, foi feita a comparação com resultados intermediários do modelo de alto nível ou, quando possível, com o modelo algébrico do módulo a ser implementado. Isso garante coerência do modelo implementado com o algoritmo no qual ele foi baseado e diminui as chances de falhas, visto que o modelo de referência já está totalmente validado no momento da síntese. A figura 52 ilustra esta estratégia de implementação.

A verificação da saída dos módulos se deu de duas diferentes maneiras, a depender da complexidade do módulo sob teste.

- Inspeção da forma de onda no *modelsim*: em módulos mais simples é a forma mais rápida de verificar a exatidão. Esta ferramenta também é fundamental para realizar *debbuging* pois permite visualizar todo o histórico dos sinais selecionados desde o começo da simulação. Pode-se ver um exemplo de simulação de uma especificação em *systemverilog* com *modelsim* na Figura 53.

Figura 52 – Fluxo de implementação de um módulo em SystemVerilog a partir de um módelo de referência em alto nível



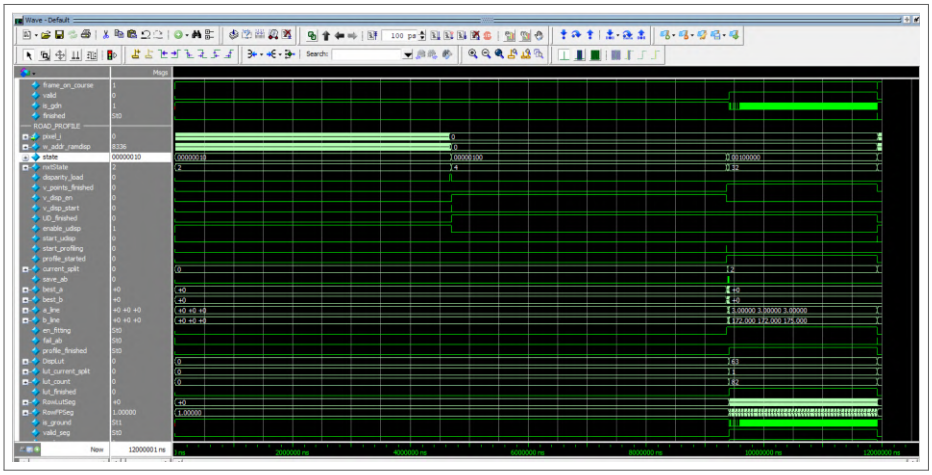
Fonte: O Autor

- Escrita de sinais em arquivos. O systemverilog tem recursos não-sintetizáveis para manipulação da memória do computador no qual está sendo realizada a simulação, usando isso, é possível escrever os sinais de interesse em arquivo para, posteriormente, utilizar scripts para validá-los.

Ambos os recursos foram usados tanto no modelo funcional não-sintetizável quanto no modelo sintetizável com precisão de ciclo. Neste segundo, é importante ressaltar, que os recursos não sintetizáveis foram usados apenas no testbench e que para visualizar os sinais internos do módulo é preciso externalizá-los até que os mesmos cheguem a instância mais alta.

A Tabela 11 detalha o procedimento de validação dos principais módulos do sistema. Demais módulos, por terem sua estrutura demasiada simples, foram considerados válidos com o teste de instâncias superiores que os continham.

Figura 53 – Visualizador de onda da ferramenta Intel Modelsim. No exemplo, pode-se ver alguns sinais da simulação de um frame no modelo funcional em systemverilog



Fonte: O Autor

Tabela 11 – Protocolo para implementação e validação individual dos módulos em hardware

Módulo	Descrição	Entrada	Saída	Validação
u_disparity	Cálculo do mapa u-disparity	mapa de disparidades	mapa u-disparity	comparação com mapa u-disparity do modelo de ref.
v_disparity	Cálculo do mapa v-disparity aperfeiçoado e binarizado	mapa de disparidades e u-disparity	mapa v-disparity binarizado	comparação com mapa v-disparity do modelo de ref.
line_ generator	Gera parâmetros de reta a partir de dois pontos	dois pontos no espaço Delta-v	parâmetros de reta	comparação com modelo algébrico
linear_func	executa função linear	paramêtros de reta e valor de variável independente	saída da y da função linear	comparação com modelo algébrico
error_ acummulator	calcula o erro total de um candidato a segmento	parâmetro de retas e conjunto de pontos a ser testado	erro total relacionado ao conjunto	comparação com modelo algébrico
line_tester	testa todas candidatas dentro do segmento	candidatos a reta e conjunto de pontos no segmento	reta com menor erro total para o segmento em análise	comparação com a reta gerada no modelo de referência
profilefitting	constrói a função linear definida por partes que representa o perfil da reta	v-disparity	função-perfil de espaço livre	comparação com modelo de referência
LUT_gen	controi LUT a partir da função encontrada	perfil de chão	Look-up-table	comparação com modelo algébrico
segmentator	Segmenta a região de chão	LUT do perfil do chão	imagem segmentada	comparação com modelo de referência
control_ statemachine	Controla o funcionamento do sistema e a integração dos módulos	-	-	-

Fonte: O autor

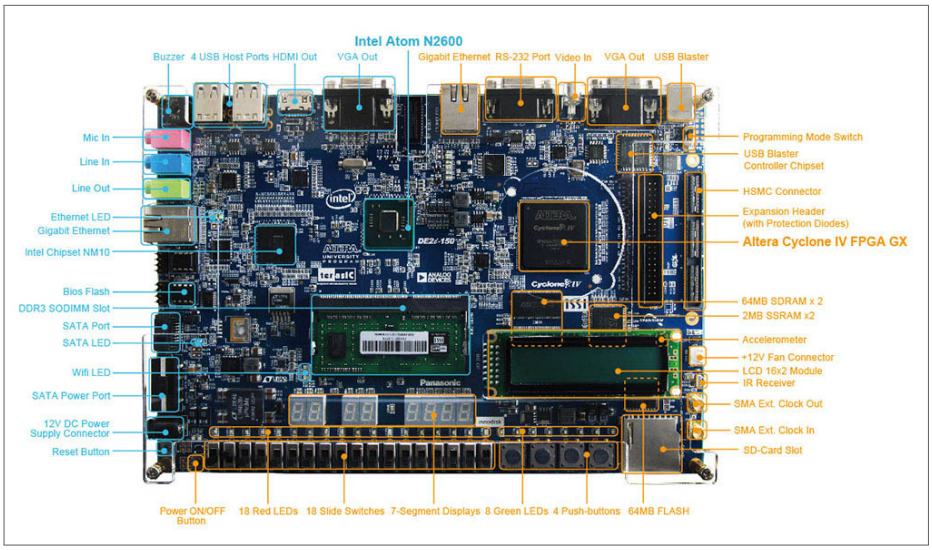
5.3 PLATAFORMAS DE IMPLEMENTAÇÃO E PROJETO

A escolha das plataformas de prototipação é um importante passo a ser definido num projeto de sistema embarcado, visto que ao mesmo tempo que deve possibilitar o cumprimento dos requisitos, a plataforma insere novas restrições para serem lidadas pelo projetista. Foram utilizadas duas plataformas diferentes do desenvolvimento deste trabalho:

1. **Plataforma de Desenvolvimento** Computador Avell modelo A60 com processador Intel Core I7-9750H (6 núcleos, 2.6 GHz), com sistema operacional Linux Ubuntu 20.04. Usado para implementar e executar o modelo de referência, implementar o código de descrição de hardware e realizar as simulações funcionais e de simulações com informações temporais(*timming*).

2. **Plataforma de Implementação** Placa de desenvolvimento Terasic DE2i-150, com FPGA Cyclone IV GX, modelo EP4CGX150DF31C7. Modelo no qual o hardware proposto foi sintetizado.

Figura 54 – Placa de desenvolvimento FPGA Terasic DE2i150



Fonte: Terasic

5.3.1 Ferramentas de projeto

As ferramentas e bibliotecas utilizadas na implementação do projeto foram:

- Editor de texto e IDE Vscode: escrita de código em c/c++ e descrição do hardware SystemVerilog.
- Compilador GNU Compiler Collection (GCC): Compilação do modelo de referência.
- Biblioteca OpenCV versão 3.4.8

- Simulador Intel Modelsim: verificação funcional e com precisão de tempo
- Intel Quartus Prime 18.1.0: Síntese de Hardware e programação de FPGA.

5.4 METODOLOGIA DE VALIDAÇÃO

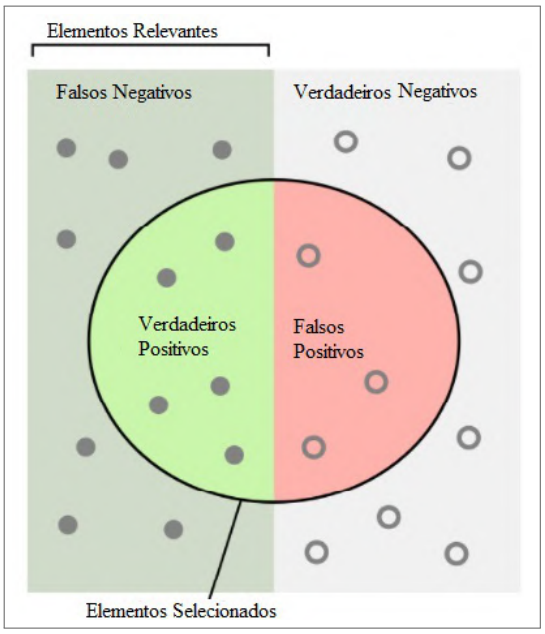
Nesta seção é apresentada a estratégia para validar se a arquitetura atende aos requisitos de projeto, mensurando, de maneira quantitativa os principais aspectos de interesse para o desenvolvimento do sistema proposto:

- **Qualidade**: o quão boa é a segmentação da imagem. Avaliam-se os resultados binários da saída do sistema comparando-os com o *ground-truth* fornecido pelo dataset.
- **Desempenho**: o quão rápido o sistema segmenta as imagens.

5.4.1 Métricas de qualidade

A segmentação de imagens é um problema da área de classificação binária, podendo ser resumido em determinar se um pixel pertence ou não à área investigada. As métricas de qualidade, portanto, devem mensurar a eficiência na classificação dos pixeis comparando o resultado com o *ground truth*.

Figura 55 – Possíveis resultados em problemas de classificação binários



Fonte: Creative Commons (ADAPTADO)

A Figura 55 ilustra os possíveis resultados no problema da classificação binária que são dados por:

- Verdadeiro Positivo - True Positive (TP): quando um pixel pertence a área que se deseja segmentar sendo classificado como tal.
- Verdadeiro Negativo - True Negative (TN): quando um pixel não pertence a área que se deseja segmentar e é corretamente classificado desta forma.
- Falso Positivo - False Positive (FP): quando um pixel não pertence a área que se deseja segmentar mas é classificado como se pertencesse.
- Falso Negativo - False Negative (FN): quando um pixel pertence a área que se deseja segmentar mas é classificado como se não pertencesse.

A partir destes valores, são calculadas as métricas do projeto. Foram escolhidas para avaliação de qualidade as métricas comumente usadas para classificação descritas em (POWERS, 2011). Esses critérios também foram os usados no *benchmark* KITTI para quantificar resultados (FRITSCH; KüHNL; GEIGER, 2013). São elas:

- P: Precisão, definido pela equação 5.1 como a razão de todos os pixeis que foram corretamente classificados como chão dentre todos os que foram classificados como chão.
- PM: Precisão média, definida pela equação 5.2 como a média da precisão dos *frames* no conjunto avaliado.
- S: Sensibilidade, definida pela equação 5.3 como a razão de todos os pixeis que foram corretamente classificados como chão dentre todos os pixeis que realmente são chão. Também conhecida como taxa de verdadeiros positivos - True Positive Rate (TPR).
- F1: Métrica que avalia simultaneamente os valores de P e S. Definida pela equação 5.4.
- $\mathrm{F1}_{max}$: Maior valor de F1 dentro de todos os frames avaliados no conjunto.
- Acurácia (A): Definida pela equação 5.5 como a razão dos pixeis avaliados corretamente dentre todos os pixeis avaliados.

$$\mathrm{P} = \frac{TP}{TP + FP} \tag{5.1}$$

$$\mathrm{PM} = \frac{1}{N} \sum_{n=1}^{n=N} \mathrm{P}_n \tag{5.2}$$

$$\mathrm{S} = \frac{TP}{TP + FN} \tag{5.3}$$

$$\mathrm{F1} = 2 \cdot \frac{\mathrm{P} \cdot \mathrm{S}}{\mathrm{P} + \mathrm{S}} \tag{5.4}$$

$$\mathrm{A} = \frac{TP + TN}{TP + FP + TN + FN} \tag{5.5}$$

5.4.2 Métricas de Desempenho

As métricas de desempenho buscam avaliar o tempo e as taxas de processamento do algoritmo. Foram considerados então:

- O Tempo (t) de processamento de um frame.
- O número de FPS alcançados pelo sistema em determinado *dataset*.

Ambas as métricas são co-dependentes visto que o FPS é o inverso do t, vide equação 5.6.

$$FPS = \frac{1}{t} \tag{5.6}$$

Esta métrica é importante pois traduz a melhoria do desempenho em relação ao modelo de referência executado em processador, que é uma das justificativas para implementação do sistema em hardware com prototipação em FPGA.

A comparação é feita em relação ao modelo de referência executado em processador Intel I7 de 9ª geração em duas versões: uma totalmente sequencial e outra explorando paralelismos de núcleos do processador.

Os paralelismos foram explorados usando a biblioteca de instruções intrínsecas para processadores x86 "x86intrin"(INTEL, 2015), que permite executar instruções algébricas inseridas em laços de repetição de maneira paralelas nos núcleos do processador. Este recursos foi utilizado nas funções de cálculo dos cálculos de histograma para computar os mapas u- e v-disparidade.

5.4.3 Datasets

Foram utilizados dois *datasets* de visão estéreo para cenário automotivo:

- o dataset KITTI, desenvolvido através da parceria do Instituto de Tecnologia de Karlsruhe, Alemanha e o Instituto Toyota de Tecnologia, com sede em Chicago, nos Estados Unidos. (FRITSCH; KüHNL; GEIGER, 2013). Este dataset foi usado para mensurar precisão e desempenho.
- o dataset Daimler, desenvolvido pela Universidade de Heidelberg, Alemanha e pela Universidade de Amsterdã, Holanda. (ENZWEILER et al., 2010) Este dataset foi usado apenas para mensuração de desempenho.

.

Os trechos a seguir aprofundam um pouco o detalhamento sobre cada dataset. Um resumo das informações mais relevantes deles pode ser encontrado na Tabela 12

Tabela 12 – Comparativo dos datasets usados para validação neste trabalho

Dataset	Tamanho	Resolução	Formato	Groundtruth
KITTI	285 imagens	1242 x 375	.png	SIM
Daimler	7129 imagens	640 x 480	.pgm	NÃO

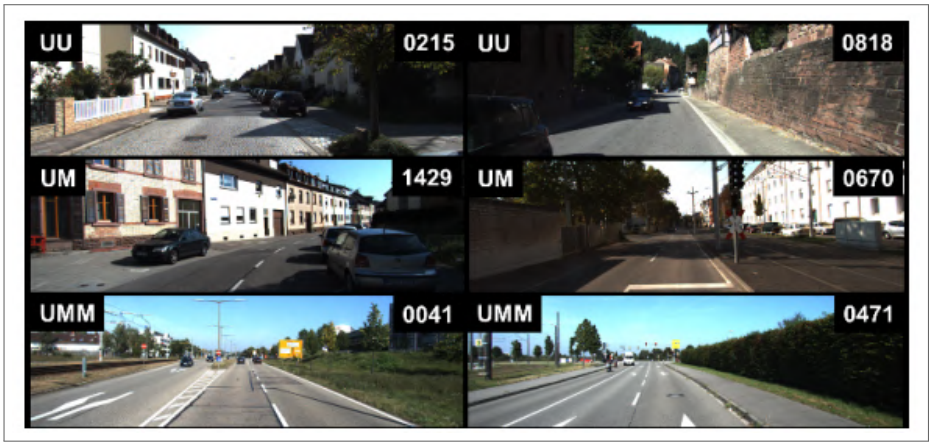
Fonte: O Autor

5.4.3.1 Dataset KITTI

O Dataset KITTI é um conjunto de *benchmarks* de uso aberto para visão computacional no contexto de veículos autônomos. Os dados foram adquiridos nos arredores da cidade de Karlsruhe na Alemanha utilizando um veículo equipado com uma grande quantidade de câmeras RGB e monocromáticas, scanners a laser e aparelhos de geolocalização de precisão. As informações adquiridas foram armazenadas em uma base de dados contendo imagens de cenários rurais, urbanos e de autoestradas, com dados adicionais de visão estéreo, LiDAR e geolocalização.

Para este trabalho, foram utilizadas as imagens captadas adicionadas do conjunto de conjugados estéreo. A Figura 56 mostra algumas imagens do *dataset* com pista marcada (UM), multiplas pistas marcadas (UMM) e pista não-marcada(UU).

Figura 56 – Subconjunto de imagens do *dataset* KITTI.



Fonte: FRITSCH; KüHNL; GEIGER (2013)

É importante frisar que este é um dataset para encontrar pista, não a região de chão, como a pista é necessariamente um subconjunto contido no chão, o desempenho do módulo proposto neste projeto pode ser pior nestas métricas que de fato é na realidade. A escolha deste dataset para validação foi uma decisão do projeto para usar um conjunto de imagens de cenário estéreo no contexto automotivo, com *ground truth* que é amplamente difundido na literatura.

5.4.3.2 Dataset Daimler

O *dataset Daimler* é um dataset que contém informação de visão estéreo monocromáticas de contexto automotivo concebido para validação de algoritmos de detecção de pedestres. As imagens foram gravadas por câmera de vídeo a uma taxa de 15 fps com qualidade VGA em cenários urbanos da cidade Alemã de Aachen. As imagens do Daimler não possuem *ground truth* aplicável ao contexto deste trabalho, sendo usados, portanto para efeito de análise qualitativa e para análise de desempenho.

Figura 57 – subconjunto de imagens do *dataset* Daimler.

Fonte: ENZWEILER et al. (2010)

5.4.4 Validação por código em systemverilog por testbenches

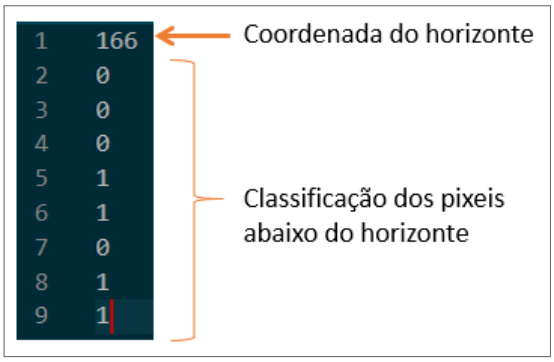
A validação dos modelos implementados em systemverilog se deu por meio da construção de *testbenches* usados para geração dos sinais de estímulo e recepção dos resultados. A partir dos pares conjugados estéreo que compõe os datasets, foi gerado um *dataset* secundário com os mapas de disparidade correspondentes. Os mapas gerados foram convertidos para texto para poderem alimentar os *testbenches* utilizados para validar tanto o modelo funcional quanto o modelo pós-sintese com precisão de ciclo.

As saídas foram captadas pelos testbenches e escritas também em arquivo de texto com o seguinte formato:

- A primeira linha continha o horizonte detectado pelo algoritmo e marca o início do procedimento de segmentação, toda a informação acima do horizonte é considerada não-pista.
- As demais linhas correspondem à classificação dos pixeis segmentados, sendo a imagem reconstruída **por coluna**.

A Figura 58 mostra um exemplo com a estrutura de saída proposta.

Figura 58 – Estrutura da saída em texto dos testbenches desenvolvidos para simulação no modelsim



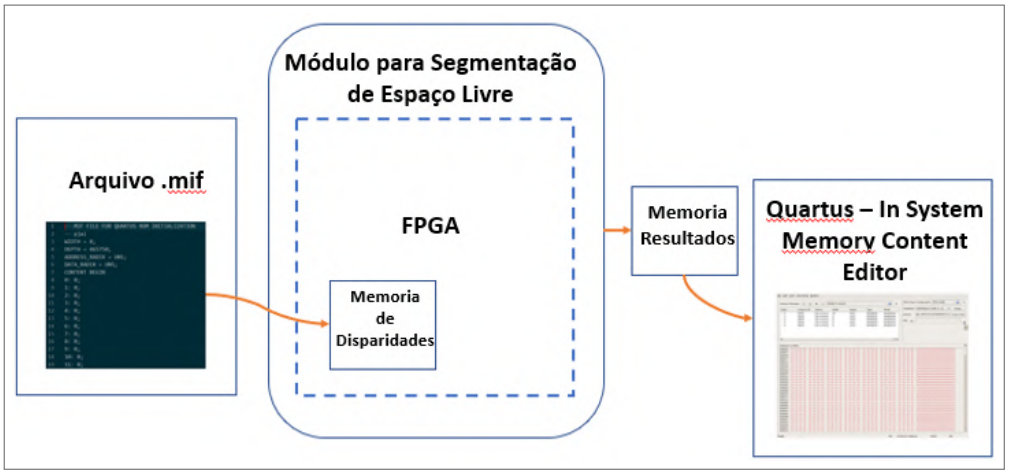
Fonte: O Autor

5.4.5 Validação do Hardware

Para validação do hardware, as informações de disparidade foram carregadas utilizando um arquivo do tipo Memory Initialization File (mif), usado para carregar previamente memórias SRAM em FPGAs Intel. Estes dados são usados para alimentar os módulos que realizam a segmentação de espaço livre.

Os resultados são adquiridos usando a ferramenta *In-system Memory Content Editor* do Quartus prime. Ela permite acessar e editar, em tempo de execução, o conteúdo de memórias SRAM da FPGA instanciadas como IPs da Intel e salvar seu conteúdo em computadores conectados por JTAG.

Figura 59 – Diagramas de blocos representando o procedimento de validação dos resultados de Hardware



Fonte: O Autor

A Figura 59 mostra o fluxo de validação em hardware, do carregamento do arquivo .mif a aquisição do resultado usando a ferramenta *In-system Memory Content Editor*.

6 RESULTADOS E DISCUSSÕES

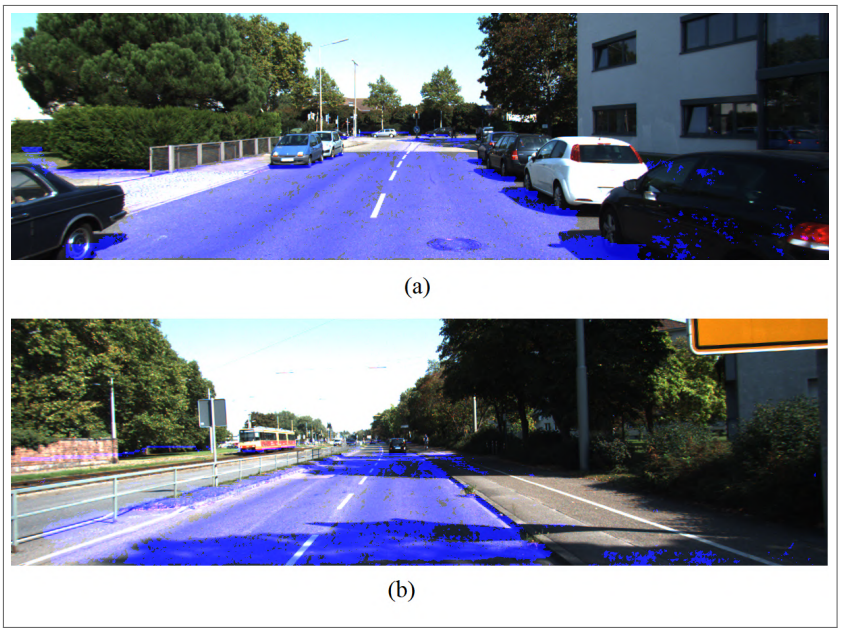
Este capítulo apresenta e discute os resultados do sistema proposto por este trabalho, submetidos às métricas apresentadas na seção 5.4 do capítulo 5. A seção 6.1 apresenta os resultados em termos de qualidade da segmentação das imagens no *dataset* KITTI. A seção 6.2 discute o desempenho do sistema ao segmentar os dados provenientes de ambos os datasets usados o KITTI e o Daimler. A seção 6.3 apresenta dados da compilação do sistema em FPGA, discutindo consumo de recursos do chip, consumo e dissipação de potência. Para finalizar, a seção 6.4 traz um comparativo com alguns trabalhos relacionados discutidos previamente no capítulo 3 que usam o *dataset* KITTI para avaliação.

6.1 QUALIDADE DA SEGMENTAÇÃO

Os modelos desenvolvidos no contexto deste trabalho foram usados para realizar segmentação de região de chão das imagens do *dataset* KITTI para obtenção de análise quantitativa e sobre as imagens do dataset Daimler para análise qualitativa por inspeção visual.

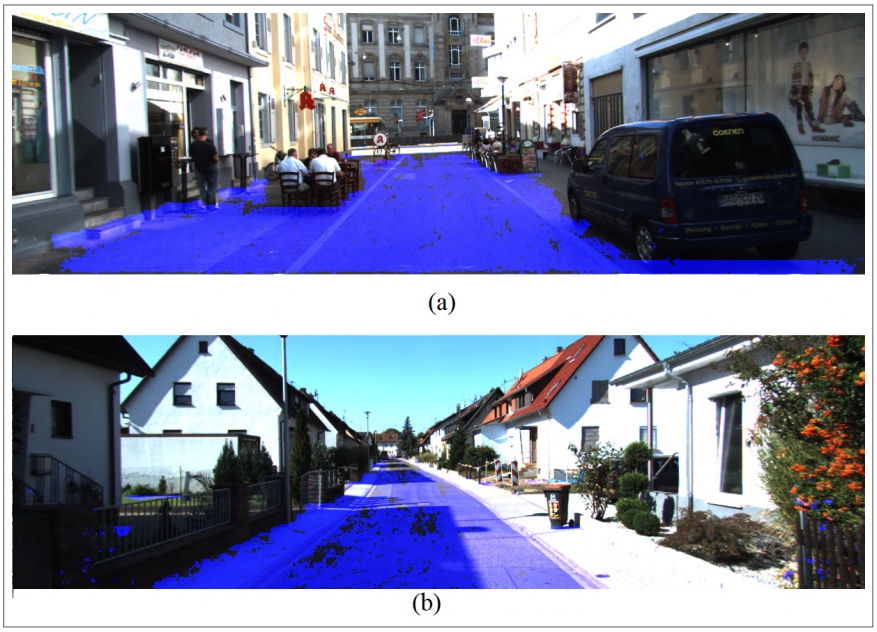
Alguns resultados de segmentação de chão para o *dataset* KITTI podem ser vistos nas Figuras 60 e 61, que mostram respectivamente a segmentação de imagens no cenário rodoviário com uma via delimitada (UM) e para cenário urbano sem delimitação de via. A região segmentada está destacada nas imagens por coloração azul.

Figura 60 – Exemplo de segmentação, executada pelo modelo de referência, do algoritmo proposto em imagens do dataset KITTI subconjunto *UM*



Fonte: O Autor

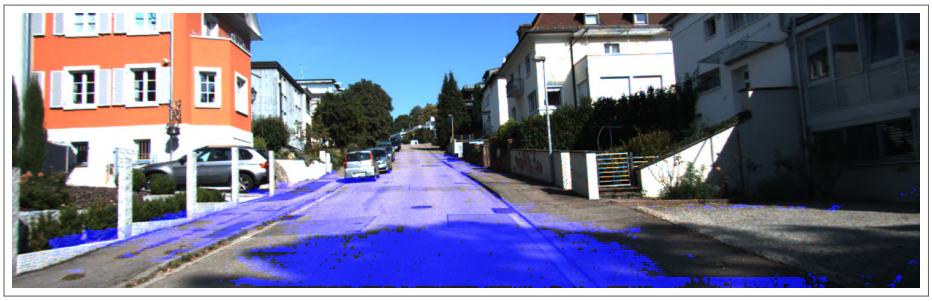
Figura 61 – Exemplo de segmentação, executada pelo modelo de referência, do algoritmo proposto em imagens do dataset KITTI subconjunto *UU*



Fonte: O Autor

Também é possível observar, ainda no *dataset* KITTI, o resultado do método para cenários não-planos. Na Figura 62 é possível observar uma região de pista ascendente com inclinação variável e a região segmentada pelo método destacada em azul.

Figura 62 – Resultado de segmentação usando o algoritmo proposto em cenário de terreno não-plano

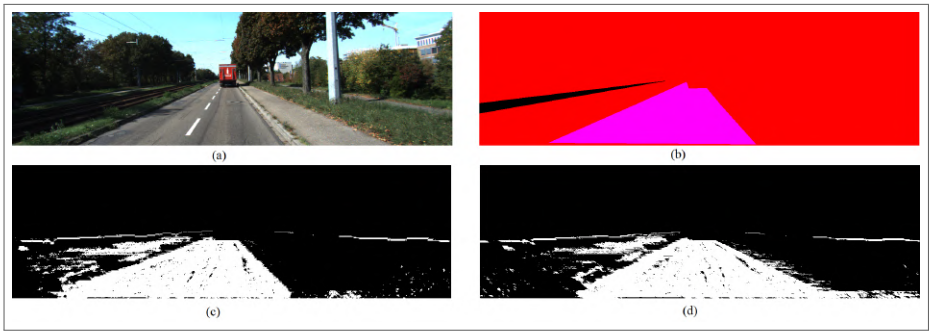


Fonte: O Autor

Para o módulo prototipado em FPGA observou-se uma pequena queda de desempenho em termos de qualidade de segmentação em relação ao modelo de referência. Na Figura 63 é possível observar os resultados da segmentação de uma imagem do KITTI (a), subconjunto de imagens rodoviárias com uma pista, do modelo de referência (c) e do módulo em hardware (d) quando comparados com o *ground truth*(b). Nos resultados do módulo de hardware observou-se uma maior quantidade de falsos positivos(FP), apesar de uma ligeira queda no número de falsos negativos(FN). Os resultados podem ser melhor observados na Tabela 13.

Os resultados mostram uma pequena queda de desempenho em relação ao modelo de

Figura 63 – Comparativo de segmentação do modelo de referência com o módulo implementado. Em (a) é mostrado uma imagem de um par conjugado do dataset KITTI, em (b) o *groundtruth* referente a esta imagem, em (c) o resultado da segmentação do modelo de referência e em (d) o resultado da segmentação do módulo em hardware



Fonte: O Autor

referência, isto se deve a uma menor precisão do modelo de ajuste dos segmentos de reta que compõem a função perfil de chão, visto que no modelo de referência, os pontos são selecionados de forma randômica e no módulo em hardware, por questões de escolha de projeto, os pontos são selecionados em um intervalo determinado e parametrizável pré-síntese. Ainda assim, a redução de qualidade se justifica com melhorias significativas apresentadas com os resultados das seções 6.2 e 6.3.

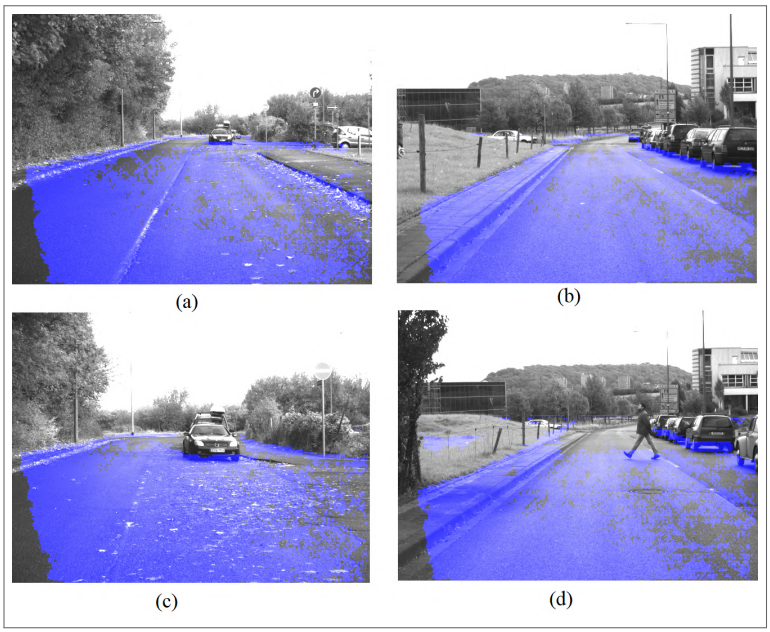
Tabela 13 – Comparativo dos resultados de qualidade de segmentação entre o modelo de referência e a implementação proposta.

Modelo	P	S	F1	A	PM	Max F1
Referência	77,89%	81,81%	79,80%	93,04%	77,18%	85,46%
FPGA	72,29%	83,64%	77,56%	91,86%	71,36%	84,96%

Fonte: O Autor

Para o *dataset* Daimler, foi feita uma validação de qualidade da segmentação apenas visual, uma vez que, como comentado no capítulo 5, o conjunto não dispõe de *ground-truth*. Os resultados de segmentação podem ser vistos na Figura 64, com a região de espaço livre destacada em azul. As regiões falhas na segmentação são decorrência da geração de uma mapa de disparidades de menor qualidade.

Figura 64 – (a), (b), (c) e (d) são exemplos de segmentação com o algoritmo em imagens do dataset Daimler



Fonte: O Autor

6.2 DESEMPENHO

O desempenho do sistema foi mensurada em termos da taxa FPS, que é o inverso do tempo de processamento do frame somado com o tempo de mudança de frames, e a frequência máxima de clock que o módulo desenvolvido consegue funcionar. Estes resultados estão mostrados na Tabela 14.

Tabela 14 – Performance do módulo em FPGA para imagens dos *datasets* KITTI e Daimler

Dataset	**Resolução**	**Frequência máxima de clock (Pior Caso)**	**Frequência de clock**	**Tempo frame**	**FPS**
Daimler	480 x 640	58,72 MHz	50,00 MHz	15,96ms	62,63 FPS
Kitti	375 x 1242	54,38 MHz	50,00 MHz	23,83ms	41,96 FPS

Fonte: O Autor

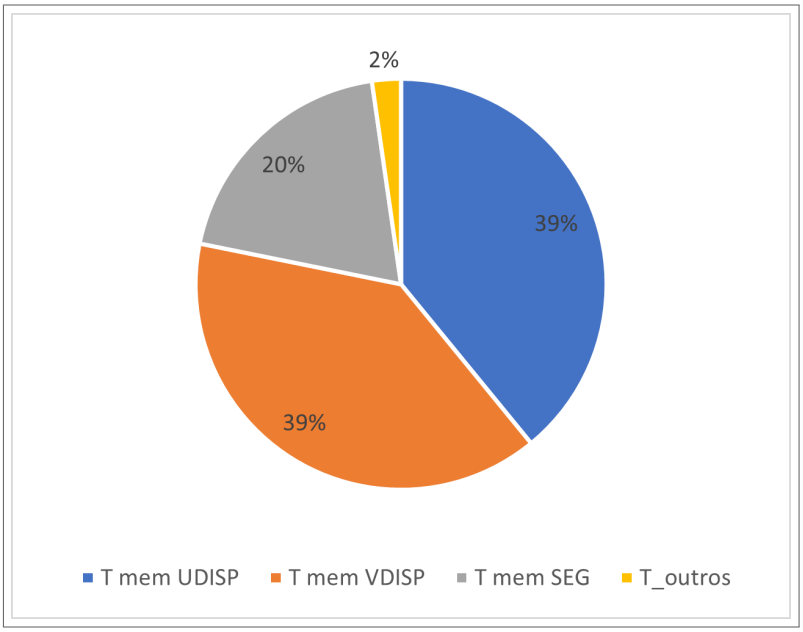
Tais resultados mostram um ganho significativo quando comparados com o modelo de referência, cujo o tempo de processamento por frame é, na versão sequencial, **0,45s** resultando em uma taxa de **2,2 FPS** e na versão com recursos de paralelismo **0,11s** ou **9,09 FPS**.

Foi possível, também, identificar que o grande gargalo limitante do desempenho do sistema é o processo de acesso aos dados da memória do mapa de disparidades. Na arquitetura, faz-se a leitura da memória completa duas vezes, nos módulos para cálculo dos mapas *u- e v-disparidade* e de aproximadamente metade da memória no módulo para segmentação, visto que, na maioria dos casos, é onde a linha do horizonte se encontra.

A memória guarda um pixel de disparidade por posição, portanto, no *dataset* KITTI, são 465.750 endereços para serem acessados, sendo realizada uma leitura por clock, na frequência de 50 MHz leva-se aproximadamente 9,32 ms para realizar uma leitura completa.

O gráfico mostrado na Figura 65 ilustra a dimensão do gargalo. É possível observar que o módulo passa 98% do tempo realizando leituras de memória. Esta informação permite orientar as melhorias na arquitetura discutidas na seção 7.2 do capítulo 7.

Figura 65 – Gráfico da ocupação do tempo durante a segmentação de um frame pelo módulo proposto. Em Azul, o percentual do tempo gasto realizando acesso de memória durante o cálculo do U-disparity, em laranja do v-disparity e em cinza o percentual do tempo gasto acessando a memória durante o processo de segmentação. Em amarelo, os 2% do tempo no qual é realizado outro tipo de processamento.



Fonte: O Autor

6.3 HARDWARE

A resolução do mapa de disparidades define os parâmetros usados na síntese do módulo desenvolvido. Nesta seção, são mostrados resultados em termos de uso de recursos da FPGA Cyclone IV GX para qual o sistema foi sintetizado e as estimativas de consumo e dissipação de potência geradas pela ferramenta *Quartus Power Analizer*.

6.3.1 Consumo de recursos da FPGA

Os relatórios de uso de recursos da síntese do módulo de segmentação para FPGA nas sínteses realizadas para os *datasets* KITTI e Daimler são respectivamente mostrados nas Tabelas 15 e 16. O consumo elevado de bits de memória é em decorrência da resolução do

mapa de disparidades relacionado às imagens. Os módulos multiplicadores são usados pelas instâncias para processamento em ponto flutuante usadas na arquitetura.

Tabela 15 – Relatório de uso de recursos da FPGA pelo sistema ajustado para imagens do dataset KITTI

Recurso	Uso
Elementos lógicos	30.997/149.760 (21%)
Registradores	19.182
Pinos	223/508 (44%)
Bits de memória	4.449.329/6.635.520 (67%)
Multiplicadores Embarcados	58/720 (8%)

Fonte: O Autor

Tabela 16 – Relatório de uso de recursos da FPGA pelo sistema ajustado para imagens do dataset Daimler

Recurso	Uso
Elementos lógicos	32.967/149.760 (21%)
Registradores	21.041
Pinos	221/508 (44%)
Bits de memória	2.834.160/6.635.520 (43%)
Multiplicadores Embarcados	58/720 (8%)

Fonte: O Autor

Apesar de ter uma resolução menor, a síntese para o *dataset* Daimler teve um uso maior de registradores, isso se ocorre devido às imagens deste conjunto possuírem mais linhas, implicando em um mapa *v-disparidade* maior e, consequentemente, em um maior número de pontos salvos em registradores para a construção da função perfil de chão.

6.3.2 Estimativas energéticas

As tabelas 17 e 18 mostram as estimativas de consumo energético e dissipação térmica de potência na FPGA para as sínteses do módulo para os dois *datasets* utilizados. Em ambos os casos, observa-se um consumo de corrente inferior à 500 mA e baixa potência dissipada, quando se compara com o consumo usual de uma CPU com arquitetura x64, plataforma na qual foram executados outros métodos para segmentação de espaço livre em imagem estéreo.

A tabela 19 traz um comparativo dos resultados do módulo com a estimativa de consumo médio para um processador Intel i7 de 9ª geração, no qual foi implementado o modelo de referência.

A tabela 19 trás ainda uma relação de desempenho pela potência dissipada comparando o modelo de referência executado no processador Intel Core i7 de 9ª geração e o módulo em

Tabela 17 – Relatório da ferramenta Quartus Power Analizer com estimativas para dissipação de potência e consumo de corrente no dispositivo quando compilado para imagens do dataset KITTI

Descrição	Potência
Consumo de corrente	431,60 mA
Dissipação térmica Dinâmica	389,17 mW
Dissipação térmica Estática	121,29 mW
Dissipação térmica nos pinos	118,35 mW
Dissipação térmica Total	**628,81 mW**

Fonte: Quartus Power Analizer

Tabela 18 – Relatório de compilação para uso de recursos da FPGA pelo sistema ajustado para imagens do dataset Daimler

Descrição	Potência
Consumo de corrente	377,49 mA
Dissipação térmica Dinâmica	339,25 mW
Dissipação térmica Estática	121,04 mW
Dissipação térmica nos pinos	103,45 mW
Dissipação térmica Total	**563,74 mW**

Fonte: Quartus Power Analizer

Tabela 19 – Comparação de dissipação de potência do módulo proposto com o consumo médio de uma CPU de arquitetura x64 Intel I7 de 9ª geração

Plataforma	Potência média dissipada	Desempenho/Potência
Intel Core i7	85,0 W	0,026 FPS/W
Módulo para Segmentação em FPGA	0,59 W	71 FPS/W

Fonte: LINUXHINT; ANANDTECH e o Autor.

FPGA. Por este ponto de vista, os ganhos são consideravelmente altos, sendo os resultados obtidos em FPGA mais de 2 mil vezes os do algoritmo de referência em processador.

6.4 COMPARATIVO

Para avaliar o desempenho do sistema em termos de qualidade de segmentação, foi realizada uma análise comparativa com os trabalhos relacionados que usaram o dataset KITTI para classificação de pista. Os resultados da comparação podem ser vistos na Tabela 20. O módulo proposto por este trabalho teve uma queda de desempenho em termos de precisão e sensibilidade de aproximadamente 20% e 10% quando comparado com o trabalho de ZHANG et al. (2018b), contudo a redução considerável no tempo de processamento por frame foi um ganho obtido que, a depender da aplicação, compensar a redução nas métricas de qualidade.

Tabela 20 – Resultados do método de referência e do módulo comparados com os principais trabalhos de segmentação de pista em visão estéreo submetidos ao *benchmark* do dataset KITTI

Método	**P**	**S**	**PM**	**Max F1**	**t**	**Ambiente**
NNP (CHEN et al., 2015)	91,43%	89,59%	87,95%	90,50%	5 s	CPU 4 cores @ 2,5 GHz (Matlab)
BMCF (WANG et al., 2016)	88,31%	90,55%	83,13%	89,42%	2,5 s	CPU 1 core @ 2,5 GHz (C/C++)
ProbBoost (VITOR et al., 2014)	85,02%	90,09%	80,13%	87,48%	150 s	CPU 8 cores @3,0 GHz (C/C++)
SCRFFPFHGSP (GHEORGHE, 2015)	82,13%	85,39%	72,89%	83,73%	5 s	CPU 8 cores @ 2,5 GHz (Matlab)
VP Dijkstra (ZHANG et al., 2018b)	91,84%	92,21%	NI	92,02%	0,25 s	CPU 4 cores @ 2,8 GHz (C/C++)
Módulo Proposto (Modelo de Referência)	77,89%	81,81%	77,18%	85,46%	0,45 s	CPU 6 cores @ 2,4 GHz (C/C++)
Módulo Proposto (Modelo de Referência com paralelismo)	77,89%	81,81%	77,18%	85,46%	0,11 s	CPU 6 cores @ 2,4 GHz (C/C++)
Módulo Proposto (FPGA)	72,29%	83,64%	71,36%	84,96%	0,02 s	FPGA CYCLONE IV @ 50 MHz

Fonte: KITTI, com entradas adicionadas pelo Autor

Um comparativo de desempenho pode ser observado na Tabela 21. Entre as técnicas avaliadas que utilizaram imagens do KITTI ou de resolução similar, o módulo desenvolvido obteve o melhor desempenho.

Apesar do trabalho de KAKEGAWA et al. (2018) ter atingido taxas de 1000 FPS, há de se considerar que a resolução das imagens suportadas por este trabalho é consideravelmente inferior. Para efeitos de comparação, pode-se usar a métrica de pixeis processados por segundo. Nestes termos, o método proposto por Kakegawa consegue processar, em um segundo, 24 milhões de pixeis enquanto o trabalho proposto consegue processar no máximo 19,5 milhões. Ainda assim, não é possível afirmar que o trabalho de Kakegawa et al. (2018) é superior, visto que a complexidade do processamento não cresce de maneira linear com o aumento da resolução.

Tabela 21 – Comparativo do modelo de referência e do método implementado com as demais técnicas citadas no 3.

Método	Resolução Frame	Desempenho	Ambiente
NNP (CHEN et al., 2015)	375 x 1242	0,2 FPS	CPU 4 cores @ 2,5 GHz (Matlab)
BMCF (WANG et al., 2016)	375 x 1242	0,4 FPS	CPU 1 core @ 2,5 GHz (C/C++)
ProbBoost (VITOR et al., 2014)	375 x 1242	0,006 FPS	CPU 8 cores @3,0 GHz (C/C++)
SCRFFPFHGSP (GHEORGHE, 2015)	375 x 1242	0,2 FPS	CPU 8 cores @2,5 GHz (Matlab)
VP Dijkstra(ZHANG et al., 2018b)	375 x 1242	4 FPS	CPU 4 cores @ 2,8 GHz (Matlab)
BSplines (JOOS, 2011)	373 x 1344	10,6 FPS	CPU 8 cores @3,0 GHz (C/C++)
VLDHist FPGA (KAKEGAWA et al., 2018)	100 x 240	1000 FPS	CPU 1 core (C/C++) FPGA (VHDL)
Módulo Proposto (Modelo de Referência)	375 x 1242	2,22 FPS	CPU 6 cores @2,4 GHz (C/C++)
Módulo Proposto (FPGA)	375 x 1242	41,96 FPS	FPGA CICLONE IV @50 MHz (SystemVerilog)

Fonte: O Autor

7 CONSIDERAÇÕES FINAIS

7.1 CONCLUSÕES

Este trabalho apresentou o projeto e desenvolvimento da arquitetura de um módulo para segmentação de espaço livre em imagens estéreo com prototipação em FPGA. Neste capítulo, são feitas as considerações finais sobre os principais tópicos discutidos nesta dissertação e as conclusões gerais mencionando as contribuições trazidas e discutindo trabalhos futuros.

O escopo deste projeto foi embasado nas pesquisas desenvolvidas pelos grupos de robótica e de desenvolvimento sistemas embarcados do CIn-UFPE. Na dissertação, foram também apresentados os conceitos básicos nos quais os métodos se baseiam e, então, mostrado como o módulo foi desenvolvido de acordo com metodologias de projeto de sistemas embarcados existentes na literatura. Os resultados foram discutidos de forma quantitativa, nas análises do dataset KITTI, e qualitativa, nas análises do dataset Daimler, realizando comparações com outras tecnologias disponíveis.

Uma das principais contribuições deste trabalho foi a implementação, a partir de conceitos consolidados na literatura para segmentação de espaço livre em imagens estéreo, da arquitetura de um módulo para segmentação de chão com prototipação em FPGA obtendo ganhos consideráveis, de até 20 vezes em relação a versão sequencial e 5,5 vezes em relação a versão com exploração de recursos de paralelismos, mantendo parâmetros de qualidade como precisão acima de 70% e sensibilidade acima de 80%. As estimativas de potência fornecidas pela ferramenta de desenvolvimento indicam que o módulo possui boa eficiência energética, com consumo de corrente e potência dissipada pelo chip da FPGA abaixo dos 500 mA e 700 mW respectivamente, o que confirma a aplicabilidade em sistemas embarcados com restrições energéticas.

Outra contribuição importante foi o desenvolvimento de um algoritmo de referência que compila métodos para aperfeiçoamento na detecção de regiões livres planas e não planas usando mapas *u e v disparidade* em um fluxo que permitisse o projeto de um módulo em hardware.

O trabalho cumpre os seus objetivos ao apresentar um meio eficiente para detecção de regiões livres para serem usados como base para trabalhos futuros dos grupos de pesquisa em robótica e sistemas embarcados da UFPE e para a comunidade científica como um todo.

7.2 TRABALHOS FUTUROS

A realização deste trabalho abriu uma série de possibilidades para serem exploradas em projetos futuros, esta seção menciona alguns dos principais pontos a serem investigados, incluindo aplicações, melhorias e contribuições adicionais que podem ser feitas.

7.2.1 Integração do módulo

Dada as motivações deste trabalho, o próximo passo imediato é a integração do módulo desenvolvido com o módulo em FPGA para geração de disparidades proposto por CAMBUIM e ajustar a saída do modulo para facilitar a comunicação com eventuais módulos posteriores.

Deve-se também buscar integrar o módulo a um framework de comunicação como, o RIFFA (JACOBSEN et al., 2015), para possibilitar o uso do módulo em plataformas Hardware e Software combinando FPGAs e CPUs e, explorando todos os recursos da placa de desenvolvimento DE2i150.

7.2.2 Otimizações na arquitetura

A discussão sobre os resultados de desempenho mostrados no Capítulo 6, explicitam que o gargalo que atualmente limita o desempenho do sistema são os acessos à memória. Como tal mecanismo é inerente à arquitetura desenvolvida, sugerem-se duas possibilidades a serem exploradas para contornar a situação, possibilitando atingir melhores taxas de FPS.

- Otimizar as lógicas utilizadas na implementação, de modo a conseguir que o módulo sintetizado funcione com *clocks* mais elevados como 100 MHz o que poderia dobrar o desempenho do sistema.
- Ajustar a arquitetura para o processamento paralelo de pixeis de disparidade, reduzindo o número necessário de acessos à memória.

7.2.3 Implementação em FPGAs com mais recursos

A plataforma se mostrou um dos impeditivos para implementação de arquiteturas mais complexas. O número limitado de bits de memória permitia processar apenas um frame de mapas de disparidade por vez. Deve-se ponderar a possibilidade de implementar uma versão do sistema em um modelo de FPGA com mais recursos, como as FPGAs da família *Stratix*, de forma que se possa projetar uma arquitetura em *pipeline* para processar múltiplos frames simultaneamente.

7.2.4 Geração de novos datasets

Durante o desenvolvimento deste projeto, verificou-se que há poucos *datasets* disponíveis publicamente para segmentação de espaço livre com imagens estéreo. Dentro dos *datasets* disponíveis, ainda, todos são voltadas para o contexto automotivo, não havendo dataset voltado para aplicações de robótica em outros cenários. Uma contribuição futura que sugere-se fazer é a construção de um dataset deste tipo, com dados estéreo e mono, aplicações de robótica indoor e outdoor e *groundtruth* de espaço livre e obstáculos.

REFERÊNCIAS

ACCELLERA. *SystemVerilog 3.1a Language Reference Manual*. 2004. Disponível em: <http://www.ece.uah.edu/~gaede/cpe526/SystemVerilog_3.1a.pdf>.

ADONIS, S. H. *The Pages of Day and Night*. 1. ed. [S.l.]: Marlboro Press, 2000. 12-13 p. ISBN 0810160811,9780810160811.

ALTERA, I. *Altera AHDL Language Reference*. 1997. Disponível em: <https://www.intel.com/content/dam/altera-www/global/en_US/uploads/c/c9/Altera_AHDL_Language_Reference.pdf>.

ANANDTECH. *The Intel 9th Gen Review: Core i9-9900K, Core i7-9700K and Core i5-9600K Tested*. 2019. Disponível em: <https://www.anandtech.com/show/13400/intel-9th-gen-core-i9-9900k-i7-9700k-i5-9600k-review/21>.

ASVADI, A.; GARROTE, L.; PREMEBIDA, C.; PEIXOTO, P.; J. Nunes, U. Multimodal vehicle detection: fusing 3d-lidar and color camera data. *Pattern Recognition Letters*, v. 115, p. 20–29, 2018. ISSN 0167-8655. Multimodal Fusion for Pattern Recognition. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0167865517303598>.

BENYUS, J. M. *Biomimicry: Innovation Inspired by Nature*. Harper Perennial, 2002. ISBN 0060533226,9780060533229. Disponível em: <http://gen.lib.rus.ec/book/index.php?md5=e450772751c1a83b5480b37c9e74a521>.

BETA, B. *Robotics Technology*. 2020. Disponível em: <https://builtin.com/robotics>.

BORE, N.; JENSFELT, P.; FOLKESSON, J. *Multiple Object Detection, Tracking and Long-Term Dynamics Learning in Large 3D Maps*. 2018.

BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

BRINKMANN, R. *The Art and Science of Digital Compositing, Second Edition: Techniques for Visual Effects, Animation and Motion Graphics (The Morgan Kaufmann Series in Computer Graphics)*. 2. ed. [S.l.]: Morgan Kaufmann, 2008. (The Morgan Kaufmann Series in Computer Graphics). ISBN 9780123706386,0123706386.

BROOKS, R. New approaches to robotics. *Science (New York, N.Y.)*, v. 253, p. 1227–32, 10 1991.

CALTAGIRONE, L.; BELLONE, M.; SVENSSON, L.; WAHDE, M. Lidar-camera fusion for road detection using fully convolutional neural networks. *Robotics and Autonomous Systems*, 2018.

CAMBUIM, L. *Um módulo de Hardware de Tempo Real de Correspondência Semi Global para um Sistema de Visão Estéreo*. Dissertação (Dissertação de Mestrado) — Universidade Federal de Pernambuco, Recife, PE, Brasil, 2017.

CAPEK CLAUDIA NOVACK-JONES, I. K. K. *R.U.R. (Rossum's Universal Robots) (Penguin Classics)*. Penguin Classics, 2004. (Rossum's Universal Robots Penguin Classics). ISBN 0141182083,9780141182087. Disponível em: <http://gen.lib.rus.ec/book/index.php?md5=1bf8aced4bebe69149f350ecd6cb3b2f>.

CHEN, X.; KUNDU, K.; ZHU, Y.; BERNESHAWI, A.; MA, H.; FIDLER, S.; URTASUN, R. 3d object proposals for accurate object class detection. In: *NIPS*. [S.l.: s.n.], 2015.

CHEN, Z.; ZHANG, J.; TAO, D. Progressive lidar adaptation for road detection. *IEEE/CAA Journal of Automatica Sinica*, IEEE, v. 6, n. 3, p. 693–702, 2019.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. *Introduction to Algorithms, Third Edition*. 3rd. ed. [S.l.]: The MIT Press, 2009. ISBN 0262033844.

ENZWEILER, M.; EIGENSTETTER, A.; SCHIELE, B.; GAVRILA, D. M. Multi-cue pedestrian classification with partial occlusion handling. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. [S.l.: s.n.], 2010. p. 990–997.

FERNANDES, J. P. *PROJETO E DESENVOLVIMENTO DE UMA ARQUITETURA EM HARDWARE RECONFIGURÁVELPARA SEGMENTAÇÃO DE VÍDEOS*. Dissertação (Dissertação de Mestrado) — Universidade Federal de Pernambuco, Recife, PE, Brasil, 2016.

FERNANDES, R.; PREMEBIDA, C.; PEIXOTO, P.; WOLF, D.; NUNES, U. Road detection using high resolution lidar. In: *2014 IEEE Vehicle Power and Propulsion Conference (VPPC)*. [S.l.: s.n.], 2014. p. 1–6.

FIRMAN, M. RGBD datasets: Past, present and future. *CoRR*, abs/1604.00999, 2016. Disponível em: <http://arxiv.org/abs/1604.00999>.

FRITSCH, J.; KüHNL, T.; GEIGER, A. A new performance measure and evaluation benchmark for road detection algorithms. In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. [S.l.: s.n.], 2013. p. 1693–1700.

GAJSKI, D. D.; ABDI, S.; GERSTLAUER, A.; SCHIRNER, G. *Embedded System Design: Modeling, Synthesis and Verification*. 1. ed. [S.l.]: Springer US, 2009. ISBN 1441905030,9781441905031.

GEIGER, A.; LENZ, P.; URTASUN, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2012.

GERSTLAUER, A.; PENG, J.; SHIN, D.; GAJSKI, D.; NAKAMURA, A.; ARAKI, D.; NISHIHARA, Y. Specify-explore-refine (ser): From specification to implementation. In: *2008 45th ACM/IEEE Design Automation Conference*. [S.l.: s.n.], 2008. p. 586–591.

GHEORGHE, I. *Semantic Segmentation of Terrain and Road Terrain for Advanced Driver Assistance Systems*. Tese (Doutorado) — Coventry University, Priory St, Coventry CV1 5FB, UK, 9 2015.

GOKHALE, P. S. G. M. B. *Reconfigurable Computing: Accelerating Computation with Field-Programmable Gate Arrays*. 1. ed. [S.l.]: Springer, 2005. ISBN 9780387261058,0-387-26105-2.

GRAY, J. Kirsti andersen, the geometry of an art: The history of the mathematical theory of perspective from alberti to monge , springer, new york (2007) isbn 10:0-387-25961-9. isbn 13:978-0387-25961-1. xviii+802. *Historia Mathematica - HIST MATH*, v. 36, p. 182–183, 05 2009.

GU, S.; YANG, J.; KONG, H. A cascaded lidar-camera fusion network for road detection. In: *ICRA*. [S.l.]: IEEE, 2021.

HADJITHEOPHANOUS, S.; TTOFIS, C.; GEORGHIADES, A. S.; THEOCHARIDES, T. Towards hardware stereoscopic 3d reconstruction a real-time fpga computation of the disparity map. In: *2010 Design, Automation Test in Europe Conference Exhibition (DATE 2010)*. [S.l.: s.n.], 2010. p. 1743–1748.

HANDA, A.; PATRAUCEAN, V.; BADRINARAYANAN, V.; STENT, S.; CIPOLLA, R. Understanding real world indoor scenes with synthetic data. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. [S.l.: s.n.], 2016.

HIRSCHMULLER, H. Accurate and efficient stereo processing by semi-global matching and mutual information. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. [S.l.: s.n.], 2005. v. 2, p. 807–814 vol. 2.

HU, Z.; LAMOSA, F.; UCHIMURA, K. A complete u-v-disparity study for stereovision based 3d driving environment analysis. In: *Fifth International Conference on 3-D Digital Imaging and Modeling (3DIM'05)*. [S.l.: s.n.], 2005. p. 204–211.

ILOIE, A.; GIOSAN, I.; NEDEVSCHI, S. Uv disparity based obstacle detection and pedestrian classification in urban traffic scenarios. In: *2014 IEEE 10th International Conference on Intelligent Computer Communication and Processing (ICCP)*. [S.l.: s.n.], 2014. p. 119–125.

INTEL. *Intell Parallel Studio XE 2015 Composer Edition for C++ Linux*. 2015. Disponível em: <https://software.intel.com/content/dam/develop/external/us/en/documents/rn-c-comp-150-update5-lin.pdf>.

INTEL. *Floating-Point IP Cores User Guide*. 20. ed. 2200 Mission College Blvd. Santa Clara, CA 95054-1549; USA, 2021.

JACOBSEN, M.; RICHMOND, D.; HOGAINS, M.; KASTNER, R. Riffa 2.1: A reusable integration framework for fpga accelerators. *ACM Trans. Reconfigurable Technol. Syst.*, Association for Computing Machinery, New York, NY, USA, v. 8, n. 4, set. 2015. ISSN 1936-7406. Disponível em: <https://doi.org/10.1145/2815631>.

JOOS, M. *Modellierung und echtzeitf äahiges Tracking befahrbarer Fl äachen unter Verwendung von Stereo-Kameras*. Tese (Doutorado) — Fakultät für Maschinenbau, Institut für Mess- und Regelungstechnik, KIT – Universit äat des Landes Baden-Wurttemberg und nationales Forschungszentrum der Helmholtz-Gesellschaft, 9 2011.

JäRVINEN, E. J. *Space perception*. 2017. Disponível em: <https://www.britannica.com/science/space-perception>.

KAGAN, E. *Autonomous mobile robots and multi-robot systems : motion-planning, communication, and swarming*. [S.l.]: Wiley, 2020. ISBN 9781119213154.

KAKEGAWA, S.; MATONO, H.; KIDO, H.; SHIMA, T. Road surface segmentation based on vertically local disparity histogram for stereo camera. *International Journal of Intelligent Transportation Systems Research*, v. 16, n. 2, p. 90–97, May 2018. ISSN 1868-8659. Disponível em: <https://doi.org/10.1007/s13177-017-0140-8>.

KILTS, S. *Advanced FPGA Design: Architecture, Implementation, and Optimization*. [S.l.]: Wiley-IEEE Press, 2007. ISBN 9780470054376,9786468600,3175723993,2006033573.

KOCUR, M.; KOZAK, S.; DVORSCAK, B. Design and implementation of fpga - digital based pid controller. In: *Proceedings of the 2014 15th International Carpathian Control Conference (ICCC)*. [S.l.: s.n.], 2014. p. 233–236.

KOK, K. Y.; RAJENDRAN, P. A review on stereo vision algorithm: Challenges and solutions. v. 13, 11 2019.

LABAYRADE, R.; AUBERT, D.; TAREL, J.-P. Real time obstacle detection in stereovision on non flat road geometry through "v-disparity"representation. In: *Intelligent Vehicle Symposium, 2002. IEEE*. [S.l.: s.n.], 2002. p. 646–651 vol.2.

LAND, N. *Tutorial - What is a Testbench*. 2014. Disponível em: <https://www.nandland.com/articles/what-is-a-testbench-fpga.html>.

LINUXHINT. *Raspberry Pi 3 Power Requirements*. 2019. Disponível em: <https://linuxhint.com/raspberry_pi_3_power_requirements/#:~:text=Raspberry\%20Pi\%203\%20Model\%20B\%20consumes\%20about\%20260\%20mA\%20of,and\%20it's\%20in\%20idle\%20state.>

LIU, S.; WAN, Z.; YU, B.; WANG, Y. *Robotic Computing on FPGAs*. [S.l.]: Morgan & Claypool Publishers, 2021. v. 16. 1–218 p.

LUMELSKY, V. J. *Sensing, intelligence, motion: how robots and humans move in an unstructured world*. J. Wiley, 2006. ISBN 0-471-70740-6,978-0-471-70740-0. Disponível em: <http://gen.lib.rus.ec/book/index.php?md5=8afacea239d23912f5d918a43bbcc57e>.

MANDUCHI, R.; CASTANO, A.; TALUKDER, A.; MATTHIES, L. Obstacle detection and terrain classification for autonomous off-road navigation. *Autonomous Robots*, v. 18, p. 81–102, 01 2005.

MELO, M. S. P. de. *MAPEAMENTO DE REGIÃO NAVEGÁVEL A PARTIR DE UM SISTEMA SLAM E SEGMENTAÇÃO DE IMAGEM*. Dissertação (Dissertação de Mestrado) — Universidade Federal de Pernambuco, Recife, PE, Brasil, 2021.

MEYER, U. B. *Digital Signal Processing with Field Programmable Gate Arrays*. [S.l.]: Springer, 2007.

MICROSOFT. *VHDL Reference Manual*. 1997. Disponível em: <https://www.ics.uci.edu/~jmoorkan/vhdlref/Synario\%20VHDL\%20Manual.pdf>.

MITCHELL, T. M. *Machine Learning*. 1. ed. McGraw-Hill, 1997. (McGraw-Hill series in computer science). ISBN 9780070428072,0070428077. Disponível em: <http://gen.lib.rus.ec/book/index.php?md5=f3aa83fb7adab9c8675871a717db6231>.

NAJMAN, L.; SCHMITT, M. Watershed of a Continuous Function. *Signal Processing*, Elsevier, v. 38, n. 1, p. 99–112, 1994. Special issue on Mathematical Morphology. Disponível em: <https://hal-upec-upem.archives-ouvertes.fr/hal-00622129>.

POWERS, D. M. W. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies*, v. 2, n. 1, p. 37–63, 2011.

PREMEBIDA, C.; AMBRUS, R.; MARTON, Z.-C. Intelligent robotic perception systems. In: HURTADO, E. G. (Ed.). *Applications of Mobile Robots*. Rijeka: IntechOpen, 2019. cap. 6. Disponível em: <https://doi.org/10.5772/intechopen.79742>.

PREMEBIDA, C.; NUNES, U. Fusing lidar, camera and semantic information: A context-based approach for pedestrian detection. *The International Journal of Robotics Research*, v. 32, n. 3, p. 371–384, 2013. Disponível em: <https://doi.org/10.1177/0278364912470012>.

RATEKE, T.; JUSTEN, K.; WANGENHEIM, A. V. Road surface classification with images captured from low-cost camera - road traversing knowledge (rtk) dataset. v. 26, 12 2019.

RENNIE, C.; SHOME, R.; BEKRIS, K. E.; SOUZA, A. F. D. *A Dataset for Improved RGBD-based Object Detection and Pose Estimation for Warehouse Pick-and-Place*. 2016.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 3rd. ed. USA: Prentice Hall Press, 2009. ISBN 0136042597.

SAARINEN, J.; ANDREASSON, H.; LILIENTHAL, A. Independent markov chain occupancy grid maps for representation of dynamic environments. In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems :*. [S.l.: s.n.], 2012. (IEEE International Conference on Intelligent Robots and Systems), p. 3489–3495. ISBN 978-1-4673-1736-8.

SAPPA, A. D.; HERRERO, R.; DORNAIKA, F.; GERÓNIMO, D.; LÓPEZ, A. Road approximation in euclidean and v-disparity space: A comparative study. In: DÍAZ, R. M.; PICHLER, F.; ARENCIBIA, A. Q. (Ed.). *Computer Aided Systems Theory – EUROCAST 2007*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 1105–1112. ISBN 978-3-540-75867-9.

SCHARSTEIN, D.; SZELISKI, R. High-accuracy stereo depth maps using structured light. In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* [S.l.: s.n.], 2003. v. 1, p. I–I.

SHAPIRO, G. C. S. L. G. *Computer Vision*. Prentice Hall, 2001. ISBN 9780130307965,0130307963. Disponível em: <http://gen.lib.rus.ec/book/index.php?md5=6c4c16c124f47da4a941a691aa0c86aa>.

SIEGWART ILLAH REZA NOURBAKHSH, D. S. R. *Introduction to Autonomous Mobile Robots*. 2nd. ed. The MIT Press, 2011. (Intelligent Robotics and Autonomous Agents series). ISBN 0262015358,9780262015356. Disponível em: <http://gen.lib.rus.ec/book/index.php?md5=827c6c13d75863c8a048672b43471db6>.

SZELISKI, R. *Computer vision algorithms and applications*. London; New York: Springer, 2011. ISBN 9781848829343 1848829345 9781848829350 1848829353. Disponível em: <http://dx.doi.org/10.1007/978-1-84882-935-0>.

VAHID, F.; GIVARGIS, T. *Embedded System Design: A Unified Hardware Software Introduction*. [S.l.]: Wiley, 2001. ISBN 0471386782,9780471386780.

VITOR, G. B.; VICTORINO, A. C.; FERREIRA J. V., G. B.; VICTORINO. A probabilistic distribution approach for the classification of urban roads in complex environments. In: *Workshop on Modelling, Estimation, Perception and Control of All Terrain Mobile Robots on IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.: s.n.], 2014.

WANG, L.; WU, T.; XIAO, Z.; XIAO, L.; ZHAO, D.; HAN, J. Multi-cue road boundary detection using stereo vision. In: IEEE. *2016 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. [S.l.], 2016. p. 1–6.

Wang, X.; Qian, Y.; Wang, C.; Yang, M. Map-enhanced ego-lane detection in the missing feature scenarios. *IEEE Access*, v. 8, p. 107958–107968, 2020.

WEDEL, A.; FRANKE, U.; BADINO, H.; CREMERS, D. B-spline modeling of road surfaces for freespace estimation. In: . [S.l.: s.n.], 2008. p. 828 – 833. ISBN 978-1-4244-2568-6.

XU, G.; ZHANG, Z. *Epipolar Geometry in Stereo, Motion, and Object Recognition: A Unified Approach*. USA: Kluwer Academic Publishers, 1996. ISBN 0792341996.

ZHANG, Y.; CHEN, H.; HE, Y.; YE, M.; CAI, X.; ZHANG, D. Road segmentation for all-day outdoor robot navigation. *Neurocomputing*, v. 314, p. 316–325, 2018. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925231218307975>.

ZHANG, Y.; SU, Y.; YANG, J.; PONCE, J.; KONG, H. When dijkstra meets vanishing point: A stereo vision approach for road detection. *IEEE Transactions on Image Processing*, v. 27, n. 5, p. 2176–2188, 2018.

ZHOU, Z.-H. *Ensemble Methods: Foundations and Algorithms*. 1st. ed. [S.l.]: Chapman and Hall CRC, 2012. ISBN 1439830037.

ZHU, S.-C.; GUO, C.-E.; WANG, Y.; XU, Z. What are textons? *Int. J. Comput. Vision*, Kluwer Academic Publishers, USA, v. 62, n. 1–2, p. 121–143, abr. 2005. ISSN 0920-5691. Disponível em: <https://doi.org/10.1007/s11263-005-4638-1>.