UNIVERSIDADE DE SÃO PAULO

ESCOLA DE ENGENHARIA DE SÃO CARLOS

DEPARTAMENTO DE ENGENHARIA MECÂNICA

Alexandre Padilha Senni

PROJETO DE HARDWARE DEDICADO PARA O PROCESSAMENTO DE IMAGENS EM APLICAÇÕES DE NAVEGAÇÃO AUTÔNOMA DE ROBÔS MÓVEIS AGRÍCOLAS

São Carlos

2016

Alexandre Padilha Senni

PROJETO DE HARDWARE DEDICADO PARA O PROCESSAMENTO DE IMAGENS EM APLICAÇÕES DE NAVEGAÇÃO AUTÔNOMA DE ROBÔS MÓVEIS AGRÍCOLAS

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Mestre em Engenharia Mecânica.

Área de Concentração: Manufatura

Orientador: Prof. Associado Mario Luiz Tronco

ESTE EXEMPLAR TRATA-SE DA VERSÃO CORRIGIDA.
A VERSÃO ORIGINAL ENCONTRA-SE DISPONÍVEL JUNTO AO DEPARTAMENTO DE ENGENHARIA MECANICA DA EESC-USP.

São Carlos

2016

AUTORIZO A REPRODUÇÃO E DIVULGAÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE ESTUDO E PESQUISA, DESDE QUE CITADA A FONTE.

S478p

Senni, Alexandre Padilha
Projeto de hardware dedicado para o processamento de imagens em aplicações de navegação autônoma de robôs móveis agrícolas / Alexandre Padilha Senni; orientador Mario Luiz Tronco. -- São Carlos, 2016.

Dissertação (Mestrado em Engenharia Mecânica e Área de concentração Manufatura). -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2016.

1. Visão computacional. 2. Robôs móveis agrícolas. 3. Navegação autônoma. 4. FPGA. I. Título.

FOLHA DE JULGAMENTO

Candidato: Engenheiro **ALEXANDRE PADILHA SENNI.**

Título da dissertação: "Projeto de hardware dedicado para o processamento de imagens em aplicações de navegação autônoma de robôs móveis agrícolas".

Data da defesa: 05/08/2016

Comissão Julgadora:	**Resultado:**
Prof. Associado **Mario Luiz Tronco** **(Orientador)** (Escola de Engenharia de São Carlos/EESC)	APROVADO
Prof. Associado **Evandro Luis Linhari Rodrigues** (Escola de Engenharia de São Carlos/EESC)	APROVADO.
Prof. Dr. **Emerson Carlos Pedrino** (Universidade Federal de São Carlos/UFSCar)	Aprovado

Coordenador do Programa de Pós-Graduação em Engenheira Mecânica:
Prof. Associado **Gherhardt Ribatski**

Presidente da Comissão de Pós-Graduação:
Prof. Associado **Luis Fernando Costa Alberto**

Dedico aos meus pais.

AGRADECIMENTOS

Agradeço primeiramente a Deus por todas as bênçãos a mim concedidas. Agradeço a Nossa Senhora da Conceição Aparecida pela minha recuperação após um acidente ocorrido durante o período de desenvolvimento deste trabalho. Também agradeço a minha mãe por todo o apoio e dedicação. Agradeço ao meu orientador Professor Mario Luiz Tronco pela dedicada orientação. Agradeço aos amigos, professores e funcionários da EESC.

Resumo

SENNI, A. P. **Projeto de hardware dedicado para o processamento de imagens em aplicações de navegação autônoma de robôs móveis agrícolas**. São Carlos, 133 p. Dissertação (Mestrado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2016.

O emprego de veículos autônomos é uma prática comumente adotada para a melhoria da produtividade no setor agrícola. No entanto, o custo computacional é um fator limitante na implementação desses dispositivos autônomos. A alternativa apresentada neste trabalho consistiu no desenvolvimento de um dispositivo de *hardware* dedicado para a navegação de robôs móveis agrícolas, o qual indica áreas navegáveis e não navegáveis, além do ângulo de inclinação do veículo em relação à linha de plantio. O desenvolvimento do projeto foi baseado em um método de extração de características visuais locais por meio do processamento de imagens coloridas obtidas por uma câmera de vídeo. O circuito foi implementado por meio de uma ferramenta de desenvolvimento baseado em um FPGA de baixo custo. O circuito consiste nas etapas de classificação, processamento morfológico e extração das linhas de navegação. Na primeira etapa, os *pixels* são classificados a partir do modelo de cores HSL em classes que representam as áreas passíveis e não passíveis de navegação. Posteriormente, a etapa de processamento morfológico realiza as tarefas de filtragem, agrupamento e extração de bordas. O processamento morfológico é realizado por meio de um arranjo de unidades de processamento dedicadas. Cada unidade pode realizar uma operação básica de morfologia matemática. O elemento estruturante utilizado na operação, bem como a operação realizada pela unidade, é configurado por meio de parâmetros do projeto. O processo de extração das linhas de orientação é realizado por meio do método de regressão linear por mínimos quadrados. A arquitetura proposta no projeto permitiu o processamento em tempo real de imagens para a aplicação de navegação autônoma de robôs móveis em ambientes agrícolas.

Palavras chave: Visão Computacional. Robôs móveis agrícolas. Navegação Autônoma. FPGA.

Abstract

SENNI, A. P. **Dedicated hardware design for image processing in applications of autonomous agricultural robot navigation**. São Carlos, 133 p. Dissertation (Master of Science) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2016.

The use of autonomous vehicles is a generally adopted practice to improve the productivity in the agriculture sector. However, the computer requirements are a limiting factor for implementation of these autonomous devices. The alternative shown in this paper is the design of a dedicated hardware for the autonomous agricultural robot navigation. The project development was based on a local visual feature extraction method by processing digital images obtained from a color video camera. The circuit was implemented through a development tool based on a low cost FPGA. The circuit consists of stages of classification, morphological processing and guidance line extraction. In the first stage, the pixels are classified through HSL color model into classes that represent suitable and unsuitable area for navigation. Then, the morphological processing stage performs filtering, grouping and edge detection tasks. The morphological processing is carried out by an arrangement of dedicated processing units. Each unit can perform a basic operation of mathematical morphology. The structuring element used in the operation and the operation performed by the unit are configured through project parameters. The guidance line extraction process is performed through the linear regression method by least square. The architecture proposed in the design allowed the real-time image processing in autonomous robot navigation applications in agricultural environments.

Keywords: Computer vision. Agricultural mobile robots. Autonomous navigation. FPGA.

Lista de Figuras

Figura 2.2.1: Capacitor buried channel (TAYLOR, 1998).....28

Figura 2.2.2: Processo de transferência de carga (TAYLOR, 1998).....29

Figura 2.2.3: Processo de leitura em arquitetura full frame (TAYLOR, 1998) 29

Figura 2.2.4: Arquitetura frame transfer.....30

Figura 2.2.5: Arquitetura Split frame transfer 30

Figura 2.2.6: Arquitetura interline transfer.....31

Figura 2.2.7: Matrizes passive pixel e active pixel (TAYLOR, 1998).....32

Figura 2.3.1: Absorção da luz pelos cones vermelho, verde e azul (GONZALEZ & WOODS, 2007) 37

Figura 2.3.2: Mistura de cores primárias e secundárias (GONZALEZ & WOODS, 2007).....38

Figura 2.3.3: Diagrama de cromaticidade da CIE (GONZALEZ & WOODS, 2007) 39

Figura 2.3.4: Faixa de cores no modelo RGB (GONZALEZ & WOODS, 2007).....40

Figura 2.3.5: Esquema do cubo de cores RGB (GONZALEZ & WOODS, 2007).....41

Figura 2.3.6: Matiz e saturação no modelo de cores HSI (GONZALEZ & WOODS, 2007).....42

Figura 2.3.7: Representação do modelo de cores HSL RGB (adaptado de GONZALEZ & WOODS, 2007) 43

Figura 2.4.1: Representação de uma imagem binária através do conjunto formado por seu sistema de coordenadas (PEDRINO, 2008) 45

Figura 2.4.2: Operação morfológica de erosão: (a) conjunto A; (b) elemento estruturante B; (c) erosão de A por B; (d) elemento estruturante alongado; (e) erosão de A pelo elemento estruturante alongado (GONZALEZ & WOODS, 2007) 46

Figura 2.4.3: Usando a erosão para remover componentes da imagem: (a) imagem binária de resolução 486 × 486; (b) a (d) imagem erodida pelos elementos estruturantes quadrados de tamanhos 11 × 11, 15 × 15 e 45 × 45, respectivamente (GONZALEZ & WOODS, 2007) 46

Figura 2.4.4: Operação morfológica de dilatação: (a) conjunto A; (b) elemento estruturante B; (c) dilatação de A por B; (d) elemento estruturante alongado; (e) dilatação de A pelo elemento estruturante alongado (GONZALEZ & WOODS, 2007).....47

Figura 2.4.5: Usando a dilatação para restaurar caracteres quebrados: (a) texto com baixa resolução; (b) elemento estruturante; (c) resultado da operação de dilatação (GONZALEZ & WOODS, 2007) 48

Figura 2.4.6: Operação de abertura de um conjunto A por um elemento estruturante circular: (a) conjunto A; (b) processo de erosão; (c) erosão de A; (d) processo de dilatação; (e) resultado da operação de abertura (GONZALEZ & WOODS, 2007).....49

Figura 2.4.7: Operação de fechamento de um conjunto A por um elemento estruturante circular: (a) conjunto A; (b) processo de dilatação; (c) dilatação de A; (d) processo de erosão; (e) resultado da operação de fechamento (GONZALEZ & WOODS, 2007).....50

Figura 2.5.1: Programmable logic array (BROWN & VRANESIC, 2000) 51

Figura 2.5.2: Programmable array logic (BROWN & VRANESIC, 2000) 52

Figura 2.5.3: Complex programmable logic device (BROWN & VRANESIC, 2000) ..52

Figura 2.5.4: Field programmable gate array (BROWN & VRANESIC, 2000).....53

Figura 2.5.5: LUT de três entradas (BROWN & VRANESIC, 2000) 54

Figura 2.5.6: Bloco lógico composto por uma LUT e um flip-flop (BROWN & VRANESIC, 2000).....55

Figura 2.5.7: Bloco lógico MUX (HAIGANG *et al*., 2014) 55

Figura 2.5.8: Bloco lógico AIC (HAIGANG *et al*., 2014).....55

Figura 2.5.9: Célula de memória estática (HAIGANG *et al*., 2014).....56

Figura 2.5.10: Célula baseada em memória flash (HAIGANG *et al*., 2014) 57

Figura 3.2.1: Representação de imagem tridimensional: (a) mapa de profundidade; (b) nuvem de pontos (CORREA, 2013).....62

Figura 3.2.2: Modelo de câmera (NERIS, 2008).....63

Figura 3.2.3: Navegação de robô móvel em cultura de milho: (a) imagem original; (b) imagem binária; (c) imagem processada (XUE *et al*., 2010) 64

Figura 3.2.4: Plataforma robótica AgTracker (XUE *et al*., 2010).....65

Figura 3.2.5: Concepção do algoritmo para extração da linha de navegação (CHOI *et al*., 2015).....66

Figura 3.2.6: Linhas de orientação. Linha vermelha tracejada: regressão por mínimos quadrados, linha verde contínua: regressão robusta (CHOI *et al*., 2015).....66

Figura 3.2.7: Trajeto realizado pelo robô em ambiente semiestruturado. O ponto azul indica a posição de partida (HIREMATH *et al*., 2014).....67

Figura 3.2.8: Extração do mapa de disparidade: (a) Imagem obtida pela câmera estéreo (somente esquerdo); (b) cálculo da disparidade entre o par de imagens (KLASER, 2014).....68

Figura 3.2.9: Aplicação do filtro para redução de ruído: (a) ruído ondular na nuvem de pontos; (b) projeção da nuvem filtrada sobre a imagem real (KLASER, 2014) 69

Figura 3.2.10: Campo de colheita de arroz: (a) imagem original; (b) mapa de disparidade (ZHANG *et al*., 2005) 70

Figura 3.2.11: Transformação em escala de cinza: (a) imagem original; (b) imagem transformada (JIANG *et al*., 2010).....71

Figura 3.2.12: Imagem limiarizada através do método de Otsu (JIANG *et al*., 2010)....72

Figura 3.2.13: Imagem dividida em fatias horizontais (JIANG *et al*., 2010).....72

Figura 3.2.14: Cálculo da estimativa dos pontos centrais das linhas de plantio: (a) quarta fatia da imagem limiarizada; (b) soma das colunas; (c) limiarização; (d) diferenciação; (e) classificação das bordas; (f) resultado do processo (JIANG *et al*., 2010) 74

Figura 3.2.15: Detecção das linhas de cultura através da transformada de Hough (JIANG *et al*., 2010).....74

Figura 3.2.16: Ponto de fuga em uma cena de plantio de soja (NERIS, 2008).....75

Figura 3.2.17: Diferentes localizações do ponto de fuga: (a) ponto de fuga à direita; (b) ponto de fuga no centro; (c) ponto de fuga à esquerda; (d) somadas colunas em (a); (e) soma das colunas em (b); (f) soma das colunas em (c) (NERIS, 2008).....76

Figura 3.2.18: Resultados do processamento e estimativa do ponto de fuga em condições variadas (NERIS, 2008).....77

Figura 3.2.19: Plataforma de testes (NERIS, 2008) 79

Figura 3.2.20: Campos agrícolas com aparências extremamente distintas: (a) alto contraste entre as plantas e o solo; (b) período de pousio (ENGLISH *et al*., 2014) 80

Figura 3.2.21: Diagrama de blocos do algoritmo (ENGLISH *et al*., 2014) 82

Figura 3.2.22: Plataforma robótica para testes (ENGLISH *et al*., 2014) 83

Figura 3.2.23: Resultados do experimento em quatro ambientes distintos: (a) trigo; (b) restolho de sorgo; (c) restolho de sorgo no período da noite sob iluminação dos faróis; (d) grão de bico (ENGLISH *et al*., 2014).....84

Figura 3.2.24: Percurso de avalição do sistema de navegação em uma região de seis hectares (ENGLISH *et al*., 2014) 85

Figura 3.2.25: Erro de desvio lateral obtido no teste de navegação autônoma em um campo de plantação de milho (ENGLISH *et al*., 2014).....86

Figura 3.2.26: Erro de desvio lateral obtido no teste de navegação autônoma em um campo de restolho de sorgo (ENGLISH *et al*., 2014) 86

Figura 3.2.27: Estrutura do sistema de navegação (MENG *et al*., 2015).....87

Figura 3.2.28: Plataforma experimental (MENG *et al*., 2015).....88

Figura 3.2.29: Processo de segmentação de imagem (MENG *et al*., 2015).....89

Figura 3.2.30: Principio do algoritmo Linear Scanning (MENG *et al*., 2014).....89

Figura 3.2.31: Detecção das linhas de plantio em uma cena agrícola com baixa densidade de ervas daninha através de diferentes algoritmos: (a) transformada de Hought; (b) método dos mínimos quadrados; (c) Linear Scanning (MENG *et al*., 2014).....90

Figura 3.2.32: Detecção das linhas de plantio em uma cena agrícola com alta densidade de ervas daninha através de diferentes algoritmos: (a) transformada de Hought; (b) método dos mínimos quadrados; (c) Linear Scanning (MENG *et al*., 2014).....90

Figura 3.2.33: Diagrama de fluxo do algoritmo para navegação autônoma de robôs móveis (SHARIFI & CHEN, 2015) 92

Figura 3.2.34: Aplicação do algoritmo Main-Shift na imagem de um pomar: (a) imagem original; (b) imagem processada (SHARIFI & CHEN, 2015) 93

Figura 3.2.35: Imagem classificada em três classes: terra, céu e arvores (SHARIFI & CHEN, 2015).....94

Figura 3.2.36: Processo de otimização: (a) transformação em escala de cinza; (b) classificação da classe localizada na região central inferior da imagem (SHARIFI & CHEN, 2015).....95

Figura 3.2.37: Representação binária da imagem classificada (SHARIFI & CHEN, 2015).....95

Figura 3.2.38: Extração das linhas de navegação através da transformada de Hough (SHARIFI & CHEN, 2015).....95

Figura 4.1.1: Fluxograma do sistema de navegação.....97

Figura 4.1.2: Módulos periféricos da plataforma de desenvolvimento DE2-70 98

Figura 4.2.1: Imagem típica de corredor para a navegação em um pomar de laranjas (Adaptado de SHARIFI & CHEN, 2015).....99

Figura 4.2.2: Diagrama do circuito de classificação dos pixels 100

Figura 4.2.3: Arranjo proposto para a classificação de cores na faixa descontínua em torno de H = 0° 100

Figura 4.3.1: Simbologia da unidade de processamento morfológico 103

Figura 4.3.2: Circuito da unidade de processamento morfológico.....103

Figura 4.3.3: Princípio de funcionamento do circuito de acesso à janela de pixels.....104

Figura 4.3.4: Arranjo para a extração de bordas.....105

Figura 4.3.5: Fluxograma do algoritmo para extração de bordas.....105

Figura 4.4.1: Circuito para a extração das linhas de navegação.....107

Figura 4.4.2: Parâmetros de orientação 107

Figura 4.4.3: Representação das linhas de orientação em uma imagem típica de cena agrícola (Adaptado de SHARIFI & CHEN, 2015).....108

Figura 4.5.1: Resultados do processo de classificação dos pixels: (a), (e), (i), (m) e (q) imagens originais; (b), (f), (j), (n) e (r) imagens binárias da classe correspondente à região do céu; (c), (g), (k), (o) e (s) imagens binárias da classe correspondente à região do solo; (d), (h), (l), (p) e (t) imagens binárias da classe correspondente à região da vegetação 111

Figura 4.5.2: Resultados do processamento morfológico: (a), (d), (g), (j) e (m) imagens originais; (b), (e), (h), (k) e (n) operação de erosão; (c), (f), (i), (l) e (o) duas operações de erosão 112

Figura 4.5.3: Resultados do processamento morfológico em duas sequências de operações: (a), (d), (g), (j), (m) e (p) imagens originais; (b), (e), (h), (k), (n) e (q) sequência de duas operações de abertura, três operações de dilatação e oito operações de fechamento; (c), (f), (i), (l), (o) e (r) sequência de duas operações de abertura, três operações de dilatação, oito operações de fechamento e extração de bordas.....113

Figura 4.5.4: Resultados do processo de extração das linhas de navegação: (a), (c), (e), (g), (i) e (k) imagens originais; (b), (d), (f), (h), (j) e (l) imagens de saída do sistema 114

Figura 4.5.5: Resultados do processo de extração das linhas de navegação de corredores em um pomar de laranja: (a), (c), (e), (g), (i) e (k) imagens originais; (b), (d), (f), (h), (j) e (l) imagens de saída do sistema 115

Figura 4.5.6: Exemplo de imagens com resultados verdadeiros positivos: (colunas 1 e 3) imagens originais; (colunas 2 e 4) imagens de saída do sistema.....116

Figura 4.5.7: Exemplo de imagens com resultados verdadeiros positivos: (colunas 1 e 3) imagens originais; (colunas 2 e 4) imagens de saída do sistema.....117

Figura 4.5.8: Exemplo de imagens com resultados verdadeiros positivos: (colunas 1 e 3) imagens originais; (colunas 2 e 4) imagens de saída do sistema.....118

Figura 4.5.9: Exemplo de imagens com resultados verdadeiros positivos: (colunas 1 e 3) imagens originais; (colunas 2 e 4) imagens de saída do sistema.....119

Figura 4.5.10: Imagens com resultados falsos positivos: (colunas 1 e 3) imagens originais; (colunas 2 e 4) imagens de saída do sistema 121

Figura 4.5.11: Imagens com resultados falsos positivos: (colunas 1 e 3) imagens originais; (colunas 2 e 4) imagens de saída do sistema 122

Lista de Tabelas

Tabela 2.2.1: Informações relevantes sobre os sensores CCD e CMOS (SOUSA & CARDOSA, 2012).....34

Tabela 2.2.2: Principais eventos na evolução de sensores de imagens de1976 até 2012 (SOUSA & CARDOSA, 2012) 35

Tabela 2.5.1: Comparação entre as tecnologias de programação de FPGA (HAIGANG *et al*., 2014).....58

Tabela 3.2.1: Dados experimentais utilizando Linear Scanning (MENG *et al*., 2014)...91

Tabela 3.2.2: Dados experimentais utilizando a transformada de Hough (MENG *et al*., 2014).....91

Tabela 3.2.3: Dados experimentais utilizando o método dos mínimos quadrados (MENG *et al*., 2014).....91

Tabela 4.5.1: Parâmetros do circuito de classificação de cores.....110

Tabela 4.5.2: Parâmetros utilizados para classificar os pixels correspondentes ao solo nas imagens do corredor de terra.....113

Tabela 4.5.3: Classificação das imagens. 115

Tabela 4.5.4: Parâmetros estabelecidos para as classes C1 e C2 116

Sumário

1. INTRODUÇÃO 23
1.1. Considerações Iniciais 23
1.2. Objetivos 24
1.3. Organização do Trabalho 25
2. FUNDAMENTAÇÃO TEÓRICA 26
2.1. Considerações Iniciais 26
2.2. Sensores de Imagem Digital 26
2.2.1. Sensor de Imagem CCD 26
2.2.2. Sensor de Imagem CMOS 31
2.2.3. Principais Diferenças Entre os Sensores CCD e CMOS 33
2.3. Representação de Imagens Digitais 35
2.4. Morfologia Matemática 44
2.5. Dispositivos Lógicos Programáveis 50
2.6. Considerações Finais 58
3. REVISÃO BIBLIOGRÁFICA 59
3.1. Considerações Iniciais 59
3.2. Visão Computacional para Navegação de Robôs Móveis Agrícolas 60
3.3. Considerações Finais 96
4. IMPLEMENTAÇÕES 97
4.1. Considerações Iniciais 97
4.2. Classificador 98
4.3. Processamento Morfológico 102
4.4. Extração das Linhas de Navegação 105
4.5. Resultados 109
4.6. Considerações Finais 123
5. CONCLUSÕES 124
6. REFERÊNCIAS BIBLIOGRÁFICAS 126

APÊNDICE A – Código em VHDL do Circuito para Conversão do Espaço de Cores RGB para HSL 130

APÊNDICE B – Código em VHDL do Circuito para Limiarização dos Pixels 131

APÊNDICE C – Código em VHDL do Bloco “operador_logico” 132

APÊNDICE D – Diagrama Esquemático do Bloco “memoria_de_deslocamento”133

APÊNDICE E – Diagrama Esquemático do Bloco “delay_sincronismo” 134

APÊNDICE F – Código em VHDL do Bloco “regressão” 135

APÊNDICE G – Código em VHDL do Bloco “filtro” 139

APÊNDICE H – Código em VHDL do Bloco “linha_media” 141

1 INTRODUÇÃO

1.1 CONSIDERAÇÕES INICIAIS

Sistemas automáticos baseados em dispositivos robóticos são utilizados para a otimização da produção e execução de tarefas nos diversos setores da indústria. Na Agricultura de Precisão os sistemas automáticos auxiliam em tarefas voltadas tanto para a inspeção e análise de culturas quanto para a locomoção autônoma de robôs móveis. Tais sistemas frequentemente utilizam métodos que empregam técnicas de visão computacional. No entanto, a visão computacional é considerada um problema desafiador para a computação de alto desempenho, tornando a implementação desses sistemas um investimento de alto custo.

A visão computacional é o campo de pesquisa que inclui métodos para aquisição, processamento, análise e compreensão de imagens, a fim de produzir informações numéricas ou simbólicas que possam auxiliar na construção de descrições explícitas e claras de objetos em uma imagem. Para a navegação autônoma de robôs móveis agrícolas a visão computacional lida com padrões visuais em linhas de plantio que são utilizados para estimar a orientação da máquina em relação à área passível de navegação. O projeto proposto no presente trabalho consiste em um dispositivo de *hardware* dedicado para o processamento de imagens em aplicações de navegação de veículos em ambientes agrícolas. O *hardware* proposto foi implementado por meio de um FPGA de baixo custo, permitindo a redução dos custos na construção do sistema de visão.

O sistema proposto consiste primeiramente em um circuito de classificação dos *pixels* a partir do modelo de cores HSL. Nessa etapa do sistema os *pixels* são identificados em classes que representam a área passível e não passível de navegação. Posteriormente, a fronteira que delimita a área navegável é calculada por meio de um grupo de unidades dedicadas para o processamento morfológico. Por último, um

circuito dedicado calcula os parâmetros que definem uma reta para a orientação da navegação do veículo por meio do método de regressão linear.

O circuito para a classificação dos *pixels* consiste em um dispositivo para a conversão do modelo de cores RGB para o HSL e um conjunto de blocos para a classificação dos *pixels* por meio de um processo de limiarização. O circuito da unidade de processamento consiste em um arranjo de memórias de deslocamento que permite o acesso simultâneo aos *pixels* de uma janela, um circuito lógico combinacional que realiza a operação morfológica no *pixel* central da janela e um circuito para ajustar os sinais de sincronismo de vídeo. O dispositivo para a extração das linhas de orientação consiste em um circuito lógico sequencial que realiza a operação de regressão linear a partir de um acumulador. As etapas finais correspondem à filtragem por meio de média aritmética e ao cálculo dos parâmetros que definem a linha central de orientação. O sistema foi desenvolvido para processar imagens de 640 × 480 *pixels*.

1.2 OBJETIVOS

Trabalhos anteriores foram realizados pela comunidade científica a fim de propor metodologias de visão computacional para a navegação autônoma de veículos em ambientes agrícolas. Alguns autores desses trabalhos concentraram seus esforços no desenvolvimento de técnicas para o processamento das imagens com um custo computacional reduzido, viabilizando economicamente a implantação do sistema. Esses métodos em sua maioria consistem em algoritmos para a identificação de características locais de uma determinada área de cultivo. O objetivo principal do presente trabalho é demonstrar a viabilidade na implantação desses métodos por meio de um *hardware* dedicado que pode ser implementado por um dispositivo lógico programável de baixo custo, desenvolvendo um circuito baseado em arquitetura *pipeline*, que permita o processamento de imagens em tempo real, e que possa ser utilizado para indicar áreas passíveis de navegação e o ângulo de inclinação do veículo em relação à linha média entre áreas de cultivo.

1.3 ORGANIZAÇÃO DO TRABALHO

No Capítulo 2 são apresentados os principais fundamentos teóricos essenciais para o desenvolvimento do projeto proposto no presente trabalho. Primeiramente são abordados os principais aspectos construtivos de sensores de imagem digital. Em seguida são apresentados os conceitos fundamentais de imagens digitais, incluindo o espaço de cores HSL, que foi utilizado no projeto para o processo de classificação dos *pixels*. Posteriormente são apresentados os principais conceitos de morfologia matemática. As operações morfológicas fundamentais de erosão, dilatação, abertura e fechamento são esclarecidas por meio de exemplos de aplicações em imagens binárias. Essas operações são posteriormente implementadas no projeto por meio de um circuito dedicado e são utilizadas para a realização de tarefas como, por exemplo, filtragem, agrupamento de regiões e extração de bordas. Por fim, nesse capítulo, é apresentada uma breve revisão das principais tecnologias de dispositivos lógicos programáveis disponíveis no mercado.

No Capítulo 3 é realizada uma revisão em metodologias de visão computacional para a navegação autônoma de veículos agrícolas. Nesse capítulo são esclarecidas as etapas envolvidas em um sistema de visão computacional, como, por exemplo, aquisição de imagem, pré-processamento, extração de características e segmentação de imagem. Posteriormente são apresentadas algumas tecnologias para aplicações de visão computacional a partir de imagens tridimensionais, e o modelo pontual para aplicações bidimensionais. Por último, alguns trabalhos relevantes em aplicações de navegação de robôs móveis agrícolas por meio de metodologias de visão computacional são apresentados.

A arquitetura de *hardware* desenvolvida no presente trabalho é apresentada no Capítulo 4. Primeiramente são esclarecidos os principais aspectos construtivos do projeto, bem como um fluxograma descriminado as etapas envolvidas no sistema. Posteriormente são abordados detalhadamente os circuitos desenvolvidos no trabalho para as tarefas de classificação, processamento morfológico e extração das linhas de orientação. Por último, são apresentados os resultados obtidos nos testes para a validação dos circuitos.

Por último, as conclusões finais do trabalho são apresentadas no Capítulo 5.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 CONSIDERAÇÕES INICIAIS

Neste capítulo são apresentados os principais fundamentos teóricos envolvidos no desenvolvimento do sistema de navegação proposto no presente trabalho. Inicialmente, são abordados os principais aspectos construtivos de sensores de imagem digital. O Segundo item aborda os fundamentos para a representação de imagens digitais, tanto em escala de cinza quanto imagens coloridas. Na sequência são apresentados os conceitos básicos de morfologia matemática, incluindo as suas principais operações. No último item são apresentados os principais dispositivos lógicos programáveis, tal como o *field programmable gate array*, sendo esse o dispositivo utilizado para implementar o circuito proposto.

2.2 SENSORES DE IMAGEM DIGITAL

Sensores de imagem digital são dispositivos digitais fabricados com semicondutores e elaborados de tal forma que a luz incidente em sua superfície produz sinais elétricos os quais, por meio de processos específicos, são convertidos em uma imagem digital. Propriedades dos sensores, tais como qualidade da imagem, velocidade de captura de *frame*, consumo de energia, tamanho e custo, podem variar conforme a tecnologia empregada em sua fabricação. Dentre as principais tecnologias disponíveis atualmente no mercado, o CCD (*Charged Coupled Device*) e o CMOS (*Complementary Metal Oxide Semicondutor*) são os mais utilizados.

2.2.1 SENSOR DE IMAGEM CCD

O CCD foi inventado em 1970 por Willard Boyle e George Smith no laboratório da Bell (BOYLE & SMITH, 1970). A ideia surgiu a partir da pesquisa de memórias de

bolhas magnéticas. Atualmente essa tecnologia é encontrada em alta escala em aparelhos como máquinas de *fax*, fotocopiadoras, câmeras fotográficas e *scanners*.

O sensor consiste em um arranjo de linha ou matriz bidimensional de células sensíveis à luz, ou *pixels*, que são capazes de produzir uma carga elétrica proporcional à intensidade da luz incidente. Um parâmetro fundamental ao CCD é a sua resolução que corresponde à quantidade de *pixels* que constituem a área sensível do dispositivo. Um dos primeiros sensores CCD foi fabricado pela Fairchild em 1974 e possuía resolução de 100 × 100 *pixels* (FAIRCHILD, 2015). Atualmente são encontrados no mercado dispositivos CCDs com resoluções de aproximadamente 9000 × 7000 *pixels*.

Cada *pixel* do arranjo que forma o sensor é essencialmente um capacitor MOS onde cada fóton é convertido em uma carga e armazenado. Tipicamente esses elementos são encontrados em dois tipos: *surface channel* e *buried channel*. Os dois tipos diferem ligeiramente no seu processo de fabricação. No entanto, quase todos os CCDs fabricados atualmente usam capacitores *buried channel* devido a grandes vantagens que esses elementos oferecem (TAYLOR, 1998).

Tipicamente o CCD é fabricado com um substrato de silício tipo P de aproximadamente 300 µm de espessura com uma camada superior do tipo N de aproximadamente 1 µm de espessura. Posteriormente, uma fina camada de dióxido de silício é acrescida seguida de um eletrodo metálico, ou porta. A sessão transversal do dispositivo é ilustrada na Figura 2.2.1.

A aplicação de um potencial positivo na porta forma uma camada de depressão na junção entre as camadas do tipo N e tipo P. A luz incidente no sensor gera um par elétron-lacuna na região da camada de depleção que, devido ao potencial elétrico aplicado, aprisiona o elétron no material tipo N. A carga negativa acumulada nesse processo é proporcional à intensidade de luz incidente. O período de exposição do sensor à luz para o acúmulo das cargas é denominado de período de integração. Após esse período o potencial de cada carga é transferido para fora do CCD e convertido em um dado digital equivalente.

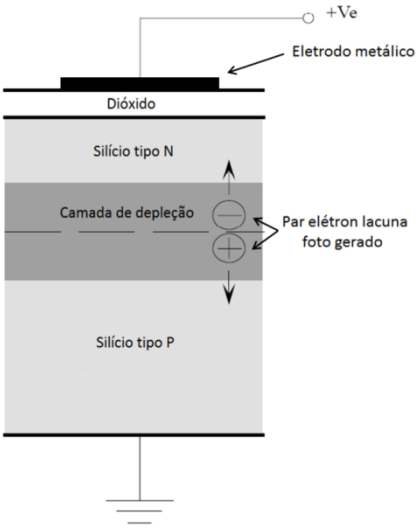


Figura 2.2.1: Capacitor buried channel (TAYLOR, 1998).

A transferência das cargas no sensor é realizada através de regiões distintas denominadas de *phase*. Embora outras configurações possam ser encontradas, tipicamente os *pixels* de um sensor CCD possuem três *phases* (sensor *three-phase*). A *phase* consiste em uma região do sensor formada por uma porta cujos sinais elétricos aplicados produzem o fluxo de carga. O processo de leitura da carga ocorre em dois estágios. No primeiro estágio as cargas elétricas de um grupo de *pixels* são transferidas através da superfície da matriz e no segundo estágio as cargas são armazenadas em um registrador antes de serem digitalizadas.

No primeiro estágio, tomando-se como exemplo um sensor *three-phase*, o processo de transferência de carga ocorre conforme ilustrado na Figura 2.2.2. No início desse processo as cargas obtidas no período de integração encontram-se distribuídas nos *phases* 1 e 2 de cada *pixel* enquanto o *phase* 3 é mantido no estado de bloqueio (Figura 2.2.2a). O *phase* 1 é então levado ao estado de bloqueio e as cargas presentes no *phase* 1 são transferidas para o *phase* 2 (Figura 2.2.2b). Posteriormente, o *phase* 3 deixa o estado de bloqueio e as cargas do *phase* 2 começam a ser transferidas para o *phase* 3 até atingir o estado de equilíbrio entre as suas cargas (Figura 2.2.2c). Nesse instante, as cargas inicialmente presentes nos *phases* 1 e 2 foram transferidas para os *phases* 2 e 3. Seguindo esse procedimento as cargas obtidas no processo de integração são

transferidas até o registrador dedicado. No segundo estágio, as cargas já transferidas para o registrador dedicado adjacente a matriz são digitalizadas e armazenadas.

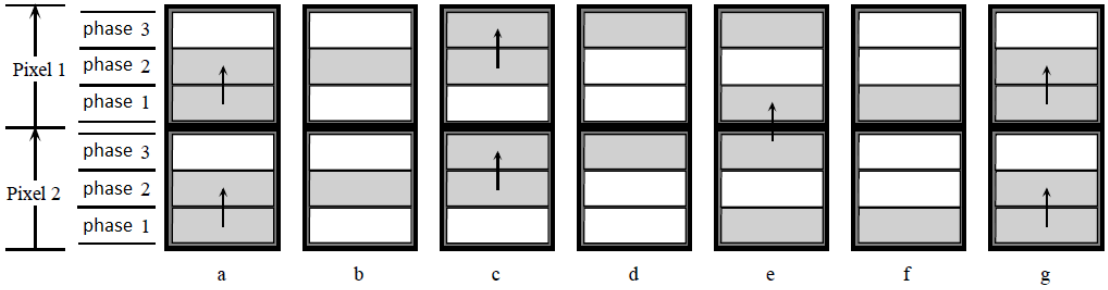


Figura 2.2.2: Processo de transferência de carga (TAYLOR, 1998).

O processo descrito anteriormente pode ocorrer em regiões individuais da matriz conforme a sua arquitetura. Normalmente essas arquiteturas são: *full frame*, *frame transfer*, *split frame transfer* e *interline transfer*.

Full frame: O processo de leitura ocorre individualmente linha por linha, as cargas são transferidas entre as linhas simultaneamente até o registrador de leitura. Os *pixels* presentes no registrador são digitalizados e assim o processo descrito ocorre até todas as linhas serem digitalizadas. O processo de leitura é ilustrado na Figura 2.2.3. Até o término desse processo os *pixels* que ainda não foram lidos estão expostos à luz incidente podendo gerar uma distorção da imagem. Uma solução comumente adotada para esse problema é a inclusão de um obturador mecânico que protege o sensor da luz enquanto ocorre a leitura da imagem.

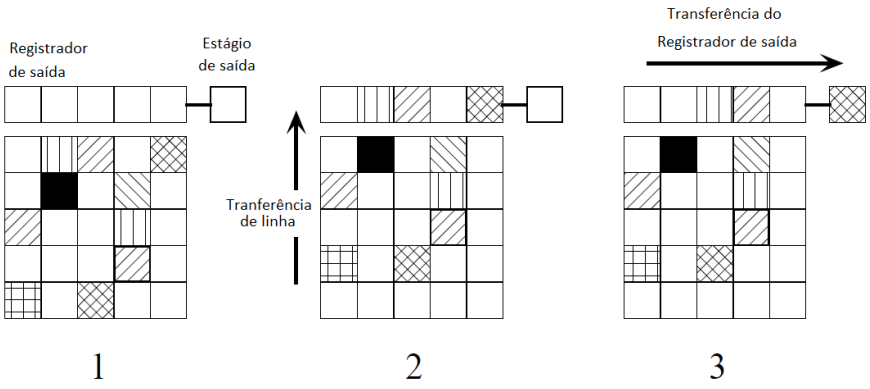


Figura 2.2.3: Processo de leitura em arquitetura full frame (TAYLOR, 1998).

Frame transfer: Nessa arquitetura o dispositivo possui uma sessão de armazenamento protegida da luz com no mínimo o mesmo tamanho da sessão de captura de imagem. A imagem registrada no período de integração é transferida para a sessão protegida de forma que uma nova imagem pode ser capturada enquanto ocorre o processo de leitura de carga da imagem anterior. O uso desta técnica permite a concepção de sensores com um período de integração maior. Na Figura 2.2.4 é ilustrado um exemplo de arquitetura *frame transfer*.

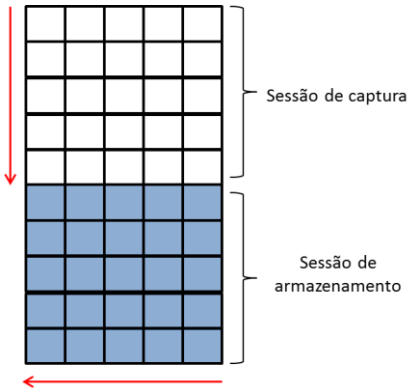


Figura 2.2.4: Arquitetura frame transfer.

Split frame transfer: Dispositivos com essa arquitetura são essencialmente semelhantes aos dispositivos apresentados anteriormente. Nessa arquitetura a sessão de armazenamento está dividida em duas partes com registradores de saída individuais. Desta forma, o processo de leitura de carga ocorre duas vezes mais rápido que em dispositivos com arquitetura *frame transfer*. Na Figura 2.2.5 é ilustrado um exemplo de arquitetura *split frame transfer*.

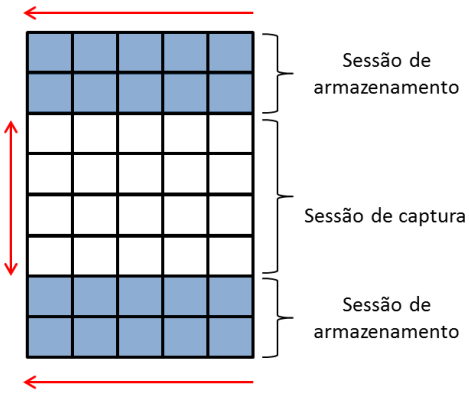


Figura 2.2.5: Arquitetura Split frame transfer.

Interline transfer: A matriz de dispositivos com essa arquitetura é constituída de colunas intercaladas de elementos fotossensíveis e de armazenamento. Ao final do período de integração todas as cargas armazenadas nos elementos sensíveis são transferidas simultaneamente para os elementos de armazenamento. Essa arquitetura possui algumas vantagens, tais como, menor tempo de exposição à luz dos elementos ainda não lidos durante o processo de transferência e capacidade de iniciar o armazenamento de cargas do próximo quadro enquanto ainda é realizada a leitura do quadro presente na região de armazenamento. No entanto, dispositivos com essa arquitetura apresentam uma redução típica de 40% em sua área sensível. Uma solução frequentemente adotada para minimizar os efeitos causados pela redução da área sensível é a utilização de microlentes posicionadas sobre cada par de regiões de forma a focar a luz apenas nas regiões fotossensíveis (TAYLOR, 1998). Na Figura 2.2.6 é ilustrado um exemplo dessa arquitetura.

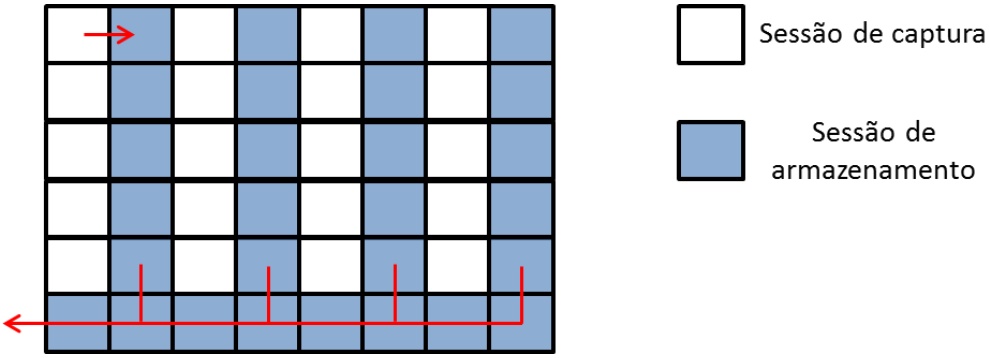


Figura 2.2.6: Arquitetura interline transfer.

2.2.2 SENSOR DE IMAGEM CMOS

Os sensores CMOS (*complimentary metal oxide semiconductor*) são conhecidos desde quase o mesmo período que os CCDs. No entanto, foi graças às pesquisas realizadas pelo *Jet Propulsion Laboratory* (JPL) da NASA que em 1993 foi possível construir um sensor CMOS com desempenho comparável aos sensores CCDs (HOSSAIN, 2010).

Sensores CMOS também são caracterizados por uma matriz de elementos sensitivos no qual cada elemento pode produzir um sinal ou carga proporcional à luz incidente. No entanto, nos sensores CMOS o sinal é obtido por meio de circuitos formados por diodos, transistores e capacitores. Esse processo ocorre nesse tipo de sensor da seguinte maneira: antes do processo de integração os capacitores recebem uma carga até uma tensão conhecida. Posteriormente, durante o período de integração essas cargas são drenadas pelos fotodiodos em uma taxa proporcional a luz incidente. As cargas remanescentes nos capacitores são lidas e digitalizadas após o término do período de integração.

Os sensores CMOS ainda podem ser classificados conforme a construção de seus *pixels*, basicamente esses são do tipo *passive pixel* ou *active pixel*. Os sensores *passive pixel* são caracterizados por um capacitor e um transistor em cada *pixel* e um amplificador em cada coluna da matriz. Após a captura de uma imagem, a carga de cada *pixel* de uma coluna é conduzida de forma intercalada pelos transistores até o amplificador respectivo. Nos dispositivos do tipo *active pixel* cada elemento da matriz possui um amplificador individual. Os dois modelos de sensores são ilustrados na Figura 2.2.7.

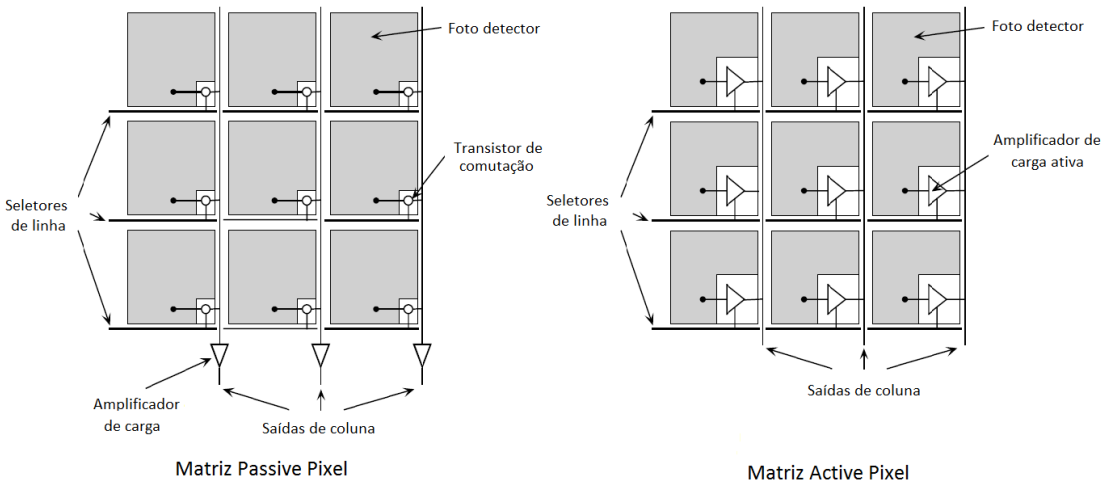


Figura 2.2.7: Matrizes passive pixel e active pixel (TAYLOR, 1998).

Em geral, os *pixels* que constituem um sensor CMOS são formados por circuitos elaborados no qual apenas um componente possui capacidade fotossensível. Por consequência, a sensibilidade desses sensores é reduzida de forma significativa. Até mesmo em sensores do tipo *passive pixel*, que possuem elementos menos elaborados, a área fotossensível é tipicamente 26% da sua área total. Uma alternativa comumente adotada para solucionar esse problema é o uso de microlentes posicionadas em cada *pixel* de forma a focar a luz incidente nas regiões fotossensíveis (TAYLOR, 1998).

2.2.3 PRINCIPAIS DIFERENÇAS ENTRE OS SENSORES CCD E CMOS

Os sensores CMOS permitem um alto nível de integração entre dispositivos no mesmo *chip*, incluindo, por exemplo, circuitos para compressão de imagens, conversores analógico-digital e controle de exposição. Embora a integração de um sistema completo também possa ser realizada por meio de vários circuitos integrados, a fabricação de um sistema completo em um único *chip* permite produzir dispositivos com o consumo até cem vezes menor que dispositivos baseados em CCDs. A integração das funções em um único *chip* também permite reduzir os efeitos de capacitância e indutância, reduzindo os atrasos de propagação no circuito. Esse fato juntamente com uma arquitetura paralela tornam os sensores CMOS mais rápidos. Os sensores CMOS podem processar cerca de 1 *Giga pixel* por segundo (SOUSA & CARDOSA, 2012).

Os CCDs apresentam um custo de produção elevado devido ao seu delicado processo de fabricação enquanto os dispositivos CMOS possuem o mesmo padrão de produção que a maioria dos circuitos integrados existentes no mercado, reduzindo o custo final desses dispositivos. Embora os dispositivos CMOS apresentem diversas vantagens quando comparados aos CCDS, tradicionalmente os CCDs oferecem melhores resultados em sensibilidade e nível de ruído, tornando-os dispositivos ideais para aplicações cientificas e industriais onde é necessário maior qualidade de imagem.

Algumas informações relevantes sobre os dois tipos de sensores são apresentadas na Tabela 2.2.1 e os principais eventos na evolução de sensores de imagens são apresentados na Tabela 2.2.2.

Variáveis	CCD (Charged Coupled Device)	CMOS (Complementary Metal Oxide Semiconductor)
(1) Ano de Invenção	1969	1990
(2) Criadores	Willard Boyle e George Smith	Eric R. Fossum
(3) Tamanho de Pixel	2,3 µm	1,3 µm
(4) Resolução	25 Mpx	25 Mpx
(5) Componente do Circuito	Capacitor MOS	Fotodiodo, capacitor, transistor
(6) Aplicações	Câmeras estáticas de alta resolução e filmadoras de alta qualidade	Câmeras de modos múltiplos, 3-D e imageamento com alta faixa dinâmica
(7) Fill Factor	80%	50 a 70%
(8) Consumo de energia	Consomem cerca de 100x mais que o CMOS por não ter um sistema dinâmico de transferência de carga	É um tipo de sensor onde no qual tem um sistema de transferência de carga mais simples e dinâmico acarretando um baixo consumo
(9) Velocidade	6,7 GB/s	34,8 GB/s novo modelo da Sony, 2012
(10) Custo de Fabricação	O CCD custa em média 2x mais que o CMOS	Os sensores CMOS hoje têm a sua produção barateada devido ao seu processo de fabricação quase 90% o mesmo padrão que os CI's existentes
(11) Qualidade e Ruído	Criam imagens de alta qualidade e baixo nível de ruído	Geralmente são mais suscetíveis a ruídos (interferência eletromagnética)

Tabela 2.2.1: Informações relevantes sobre os sensores CCD e CMOS (SOUSA & CARDOSA, 2012).

Descrição	Ano	Lugar	Autor	Tecnologia	Característica
(1) Estudo sobre o PPS	1967	Fairchild Semiconductor	Mr. Gene P. Weckler	Pixel Passivo	Cada pixel é um fotodiodo e um transistor
(2) Criado o primeiro CCD	1969	Bells Labs	Willard Boyle e George Smith	Capacitiva	Soldado manualmente
(3) Produzidos os CMOS	1974	U.S.A	Fairchild Electronics	Fotodiodos	Produção apenas comercial
(4) Produção dos CCD	1974	U.S.A	Fairchild Electronics	Capacitor MOS	Resolução 100x100 pixels
(5) Primeira câmera digital	1975	Eastman Kodak Company	Steve Sasson	Lente de câmera super8	Baixa velocidade
(6) Uso astronômico	1975	Planeta Urano	NASA	CCD	Primeira fotografia da astronomia
(7) Primeira máquina com CCD	1976	U.S.A	Sony	CCD	Primeira máquina com CCD
(8) Retina de silicone	1989	U.S.A	C. Mead	CMOS	Amplificadores dentro do pixel retina de silicone
(9) Comercialização APS	1995	U.S.A	Photobit Corp	APS	Tecnologia APS para uma série de aplicações
(10) Nova geração de sensor CMOS	2012	U.S.A	Sony	CMOS	Atinge velocidade de 34,8 GB/s

Tabela 2.2.2: Principais eventos na evolução de sensores de imagens de1976 até 2012 (SOUSA & CARDOSA, 2012).

2.3 REPRESENTAÇÃO DE IMAGENS DIGITAIS

A representação gráfica de uma imagem pode ser definida como uma função bidimensional $f(x,y)$ cujos componentes x e y são as coordenadas espaciais do plano onde a imagem é projetada, e a amplitude de f em qualquer par de coordenadas

representa a intensidade da luz nesse ponto. Em imagens monocromáticas o valor de intensidade f é denominado de nível de cinza. Denomina-se a representação de uma imagem em imagem digital quando x, y e os valores de intensidade de f são quantidades finitas e discretas. Os elementos finitos em uma imagem digital são denominados de *pixels* (GONZALEZ & WOODS, 2007).

Em imagens monocromáticas o valor de intensidade de luz pode ser representado conforme a equação:

$$l = f(x, y) \tag{2.3.1}$$

O nível de cinza l encontra-se no intervalo de valores definido pela equação:

$$L_{mín} \leq l \leq L_{máx} \tag{2.3.2}$$

O intervalo estabelecido por $L_{mín}$ e $L_{máx}$ é denominado de escala de cinza e na pratica os seus valores são respectivamente 0 e (L – 1), onde l = 0 é considerado preto e l = (L – 1) é considerado branco. Todos os valores intermediários variando do preto ao branco são denominados de tons de cinza.

Os valores de intensidade em uma imagem digital são representados em uma matriz de tamanho $M \times N$, onde M e N representam a largura e o comprimento da imagem, respectivamente. O número de níveis de intensidade L é tipicamente uma potência inteira de 2, conforme demonstrado na Equação 2.3.3. O número de *bits* necessários para o armazenamento da imagem digitalizada é dado pela Equação 2.3.4.

$$L = 2^k \tag{2.3.3}$$

$$b = M \times N \times k \tag{2.3.4}$$

A percepção de cores no sistema visual humano ocorre por meio de sensores denominados de cones. Os cones são divididos em três categorias, cujos picos correspondem aproximadamente às cores vermelho, verde e azul. Na Figura 2.3.1 são ilustradas as curvas experimentais médias detalhando a absorção da luz pelos cones do olho humano. Devido a essas características de absorção do olho humano, as cores são vistas como combinações das chamadas cores primárias aditivas: vermelho (R), verde (G) e azul (B). Em 1931 a Comissão Internacional de Iluminação (CIE) determinou os seguintes valores específicos como comprimento de onda para as cores primárias: R = 700 nm, G = 546,1 nm e B = 435,8 nm.

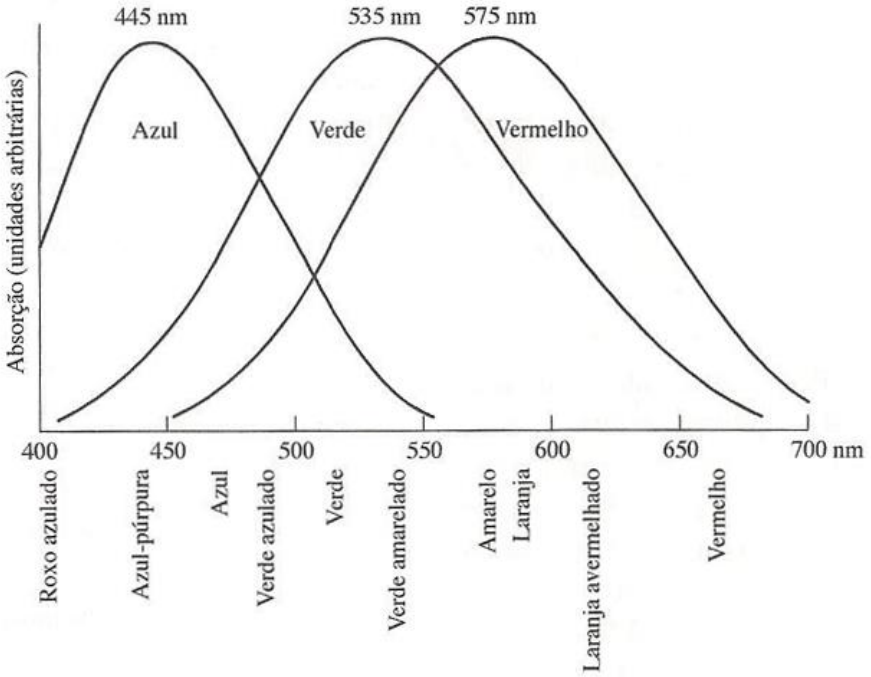


Figura 2.3.1: Absorção da luz pelos cones vermelho, verde e azul (GONZALEZ & WOODS, 2007).

As cores primárias aditivas podem ser adicionadas para produzir as cores secundárias: magenta, ciano e amarelo. A combinação entre as três cores primarias ou uma secundária com a sua primária oposta em intensidades corretas produzem a luz branca. Quando se trata de pigmentos ou corantes, uma cor primária é definida como uma cor que absorve uma cor primária de luz e reflete as outras duas. Nesse caso, as

cores primárias de pigmento, ou primárias subtrativas, são: magenta, ciano e o amarelo. As cores secundárias de pigmento são: vermelho, verde e azul. A combinação das três cores primárias de pigmento, ou uma secundária com a sua primária oposta, produz o preto. Os resultados obtidos por meio da combinação de cores são ilustrados na Figura 2.3.2.

Figura 2.3.2: Mistura de cores primárias e secundárias (GONZALEZ & WOODS, 2007).

As intensidades das cores primárias R, G e B necessárias para formar qualquer cor em particular são denominadas valores de triestímulo e expressas como X, Y e Z, respectivamente. Portanto, uma cor pode ser especificada por seus coeficientes tricromáticos x, y e z a partir dos valores de triestímulo conforme as equações:

$$x = \frac{X}{X + Y + Z} \tag{2.2.5}$$

$$y = \frac{Y}{X + Y + Z} \tag{2.2.6}$$

$$z = \frac{Z}{X + Y + Z} \tag{2.2.7}$$

A partir dessas equações, temos:

$$x + y + z = 1 \tag{2.3.8}$$

No diagrama de cromaticidade da CIE ilustrado na Figura 2.3.3 é apresentada a composição de cores como uma função de x e y. A cor denominada de verde no diagrama, por exemplo, corresponde aproximadamente a 62% de verde, 25% de vermelho e, a partir da Equação 2.3.8, o coeficiente z é calculado 13%. Os pontos pertencentes à fronteira do diagrama representam as cores puras e os pontos dentro do diagrama correspondem a uma mistura de cores do espectro. O ponto de energia igual representa o padrão da CIE para a luz branca.

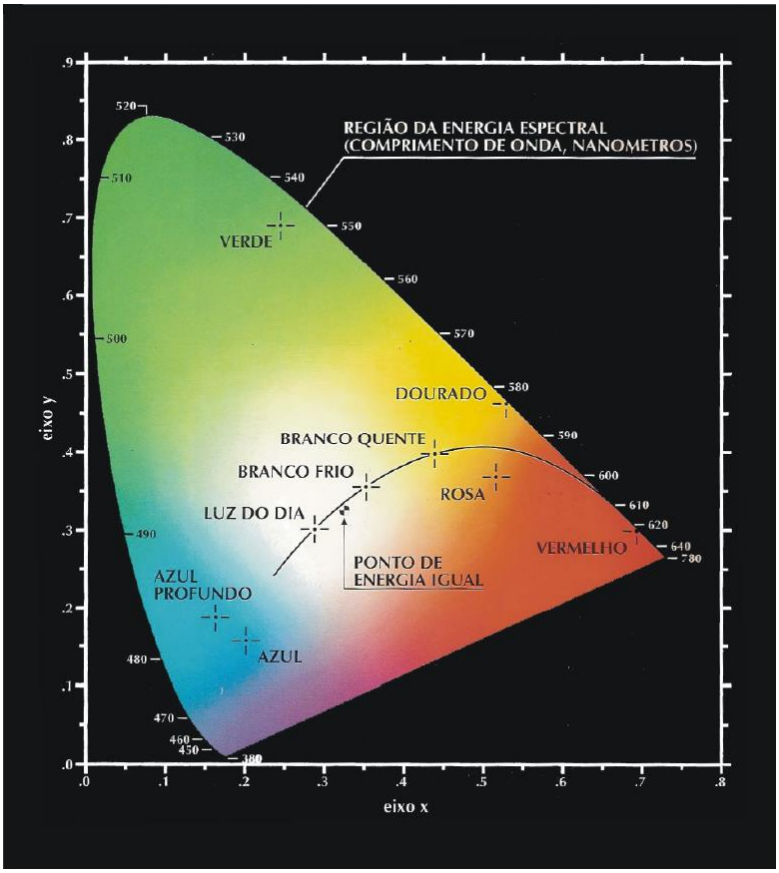


Figura 2.3.3: Diagrama de cromaticidade da CIE (GONZALEZ & WOODS, 2007).

A faixa de cores obtidas a partir da combinação de três cores pode ser determinada por meio do diagrama de cromaticidade da CIE. O triângulo formado pelas retas que ligam os três pontos de cores representa a área das cores que podem ser obtidas a partir da mistura de seus componentes. Na Figura 2.3.4 é ilustrada a faixa de cores obtidas por meio da combinação das cores primárias R, G e B.

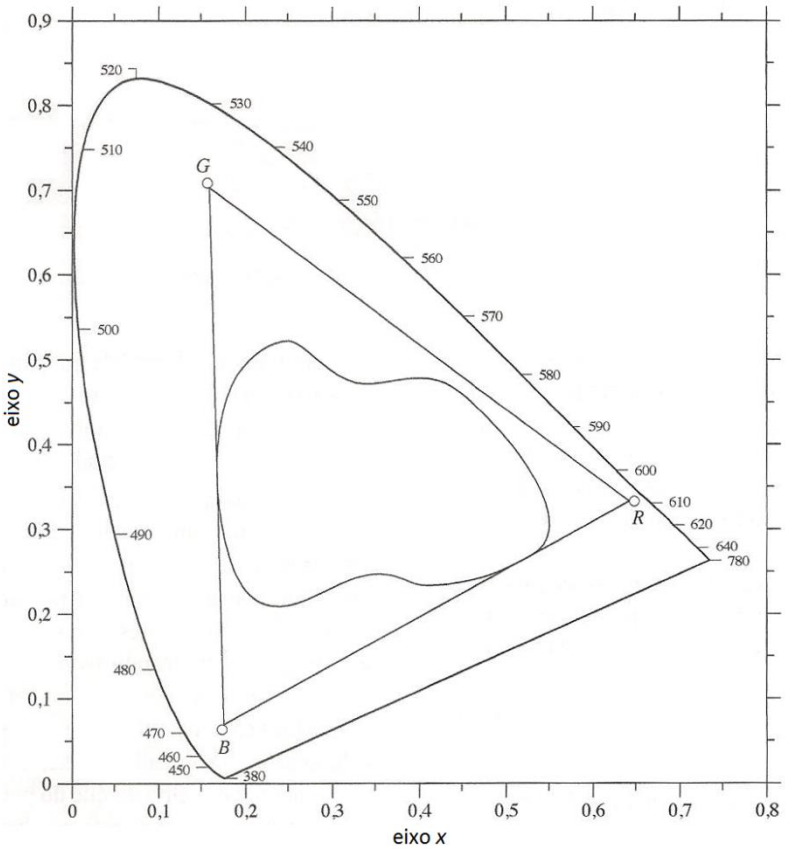


Figura 2.3.4: Faixa de cores no modelo RGB (GONZALEZ & WOODS, 2007).

A faixa de cores obtida a partir das componentes R, G e B também é frequentemente representada por um cubo unitário conforme ilustrado na Figura 2.3.5. No modelo RGB a intensidade em cada componente de cor é representada por uma matriz de tamanho $M \times N$ e, portanto, a quantidade de *bits* necessários para o armazenamento da imagem colorida é dada pela equação:

$$b = 3 \times M \times N \times k \tag{2.3.9}$$

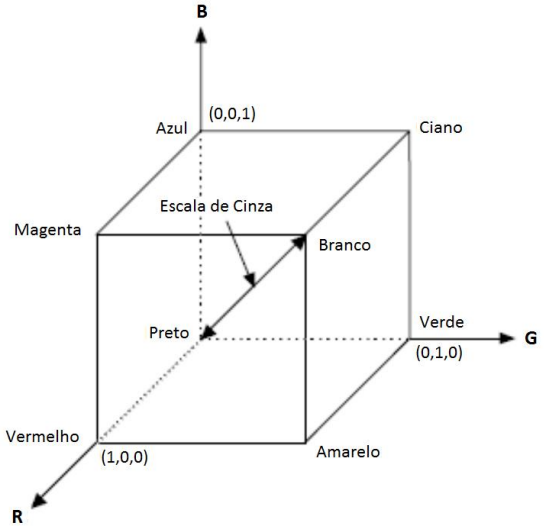


Figura 2.3.5: Esquema do cubo de cores RGB (GONZALEZ & WOODS, 2007).

O espaço de cores RGB é frequentemente utilizado em tarefas de geração de imagens coloridas, como, por exemplo, a aquisição de imagens por uma câmera colorida ou a sua representação em um monitor. No entanto, a sua utilização na descrição de cores é considerada limitada. As tarefas de processamento de imagens no modelo RGB apresentam limitações em desempenho devido à correlação existente entre os canais de cores, além de ser impossível medir similaridades de cores utilizando distância euclidiana (LULIO, 2011).

No processo de segmentação de imagens coloridas o espaço de cores comumente adotado é o HSI (matiz, saturação e intensidade). Nesse modelo as informações de intensidade são separadas das informações de cores, tornando relativamente simples e intuitivo a sua interpretação. O componente H define o comprimento de onda dominante da cor. O componente S define a quantidade de branco misturado à cor. Por último, o componente I define a intensidade de luz.

As informações de cores no modelo HSI podem ser representadas em um plano circular no qual as cores primárias estão separadas em 120° e as secundárias posicionadas em ângulos de 60° das primárias, conforme ilustrado na Figura 2.3.6. O componente S corresponde ao comprimento do segmento de reta que liga o centro da circunferência a um ponto de cor aleatório. Normalmente o ângulo de 0° indica o vermelho.

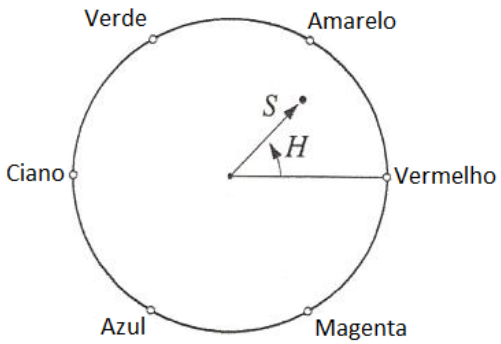


Figura 2.3.6: Matiz e saturação no modelo de cores HSI (GONZALEZ & WOODS, 2007).

A seguir são apresentadas as equações para a transformação do espaço de cores RGB para o HSI. O componente H é determinado pelas equações:

$$H = \begin{cases} \theta & se\ B \leq G \\ 360 - \theta & se\ B > G \end{cases} \tag{2.3.10}$$

$$\theta = cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B]}{[(R-G)^2+(R-B)(G-B)]^{1/2}}\right\} \tag{2.3.11}$$

O componente S é dado por:

$$S = 1 - \frac{3}{(R+G+B)}[mín(R,G,B)] \tag{2.3.12}$$

Por último, o componente I é dado por:

$$I = \frac{1}{3}(R+G+B) \tag{2.3.13}$$

As informações cromáticas no modelo HSI são irrelevantes quando em condições de baixa luminosidade ou baixa saturação. Portanto, as descrições de cores dos *pixels* nessas condições devem ser desconsideradas em algoritmos para classificação e segmentação de cores.

Os espaços de cores HSV e HSL são variações do modelo HSI. No modelo HSV o componente V (valor) é determinado pela equação:

$$V = máx(R, G, B) \tag{2.3.14}$$

No modelo HSL o componente L (luminosidade) é determinado pela equação:

$$L = \frac{1}{2}[máx(R, G, B) + mín(R, G, B)] \tag{2.3.15}$$

O modelo de cores HSL pode ser representado conforme ilustrado na Figura 2.3.7.

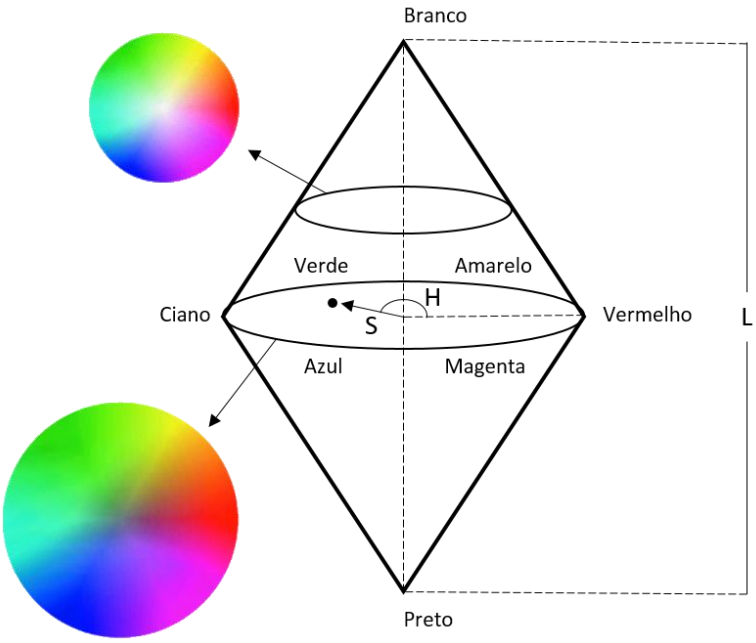


Figura 2.3.7: Representação do modelo de cores HSL RGB (adaptado de GONZALEZ & WOODS, 2007).

2.4 MORFOLOGIA MATEMÁTICA

A morfologia matemática consiste em uma ferramenta para extrair componentes das imagens que são úteis na representação e na descrição da forma de uma região, como fronteiras, esqueletos e fecho convexo. A morfologia pode ser aplicada em diversas áreas de processamento de imagens, como, por exemplo, realce, filtragem e segmentação. A palavra morfologia é derivada das palavras gregas *morphê* (forma) e *logos* (ciência) e, portanto, o seu significado é o estudo da forma, ou estrutura. O uso mais antigo dessa palavra está relacionado ao estudo de linguística e biologia. Em linguística, morfologia estuda a estrutura das palavras. Em biologia, morfologia está relacionada à forma de um organismo (GONZALEZ & WOODS, 2007; PEDRINO, 2008).

No contexto de imagens digitais, a morfologia matemática começou a ganhar destaque em 1964 por meio de pesquisas conjuntas dos pesquisadores franceses Georges Matheron e Jean Serra. Entre 1964 e 1968 foram estabelecidas as primeiras noções teóricas, incluindo operações, tais como, abertura, fechamento e *hit-miss*. Nesse período foi criado o centro de morfologia matemática na *École des Mines de Paris* em Fontainebleau, França (PEDRINO, 2003; FACON, 2011).

A linguagem da morfologia matemática é a teoria dos conjuntos. O princípio básico da morfologia consiste em extrair informações relativas à geometria de conjuntos desconhecidos de uma imagem, a partir do elemento estruturante. O elemento estruturante é um conjunto completamente definido e conhecido em forma e tamanho, que é comparado aos conjuntos desconhecidos da imagem, permitindo testar se o elemento estruturante está contido na imagem. Os conjuntos, em morfologia matemática, representam os objetos encontrados em uma imagem. O conjunto de todos os *pixels* pretos de uma imagem, por exemplo, é uma descrição morfológica completa da imagem. Em imagens binárias, tais conjuntos pertencem ao espaço bidimensional de números inteiros Z^2 e cada elemento do conjunto é formado por um vetor bidimensional cujas coordenadas são (x,y). As coordenadas x e y podem ser as coordenadas dos *pixels* pretos ou dos *pixels* brancos da imagem, dependendo da convenção. No exemplo ilustrado na Figura 2.4.1 o objeto em forma de “U” pode ser representado

completamente pelo conjunto A = {(1,1),(1,2),(1,3),(2,3),(3,1),(3,2),(3,3)}. No presente trabalho, por convenção, os conjuntos são caracterizados pelos *pixels* pretos da imagem. Em imagens digitais em níveis de cinza os conjuntos pertencem a Z^3, sendo que, um componente é o valor discreto de intensidade e os outros dois as suas coordenadas (PEDRINO, 2008; GONZALEZ & WOODS, 2007).

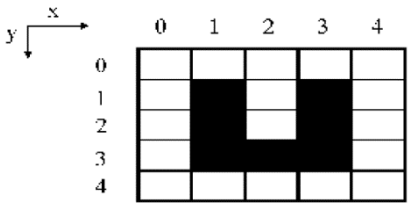


Figura 2.4.1: Representação de uma imagem binária através do conjunto formado por seu sistema de coordenadas (PEDRINO, 2008).

A seguir são apresentados os principais operadores de morfologia matemática.

Sendo A e B conjuntos de Z^2, a erosão do conjunto A pelo elemento estruturante B pode ser definida como:

$$A \ominus B = \{z|(B)_z \subseteq A\} \tag{2.4.1}$$

A erosão de A por B é o conjunto de todos os pontos *z* que ao transladar B configuram B contido em A. Na Figura 2.4.2 é ilustrado um exemplo de erosão. Os elementos A e B são ilustrados nas Figuras 2.4.2a e 2.4.2b, respectivamente. Na Figura 2.4.2c é ilustrado o resultado da erosão de A por B. Nas Figuras 2.4.2d e 2.4.2e é ilustrado, respectivamente, um elemento estruturante alongado e a erosão de A por esse elemento.

Um exemplo prático de aplicação da operação de erosão consiste em remover as linhas que ligam o componente central com os componentes das bordas de uma imagem, como ilustrado na Figura 2.4.3. Nas Figuras 2.4.3b, 2.4.3c e 2.4.3c são ilustrados os resultados da erosão da imagem pelos elementos estruturantes quadrados de tamanho 11 × 11, 15 × 15 e 45 × 45, respectivamente. Na primeira erosão foram

removidas quase todas as linhas. Na segunda, as linhas foram removidas completamente. Por último, o elemento quadrado de 45 × 45 removeu completamente as linhas e os elementos na borda da imagem. O exemplo indica que a operação de erosão diminui ou afina objetos em uma imagem binária. A erosão também elimina elementos da imagem cujo tamanho é menor que o elemento estruturante.

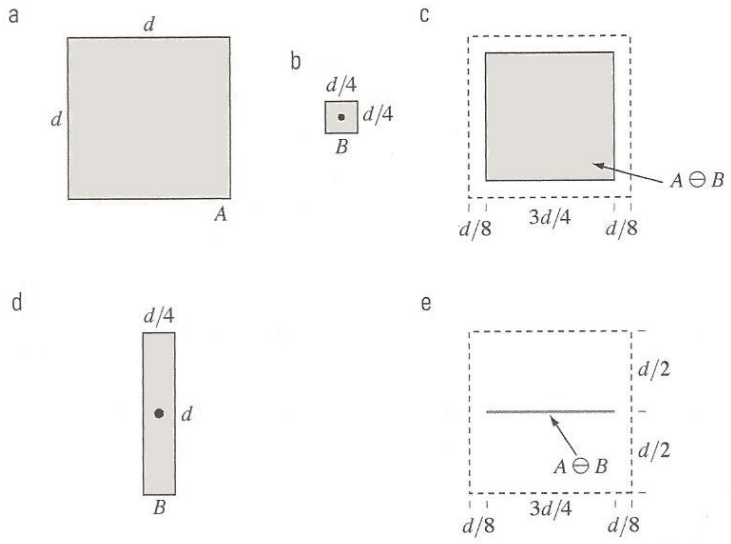


Figura 2.4.2: Operação morfológica de erosão: (a) conjunto A; (b) elemento estruturante B; (c) erosão de A por B; (d) elemento estruturante alongado; (e) erosão de A pelo elemento estruturante alongado (GONZALEZ & WOODS, 2007).

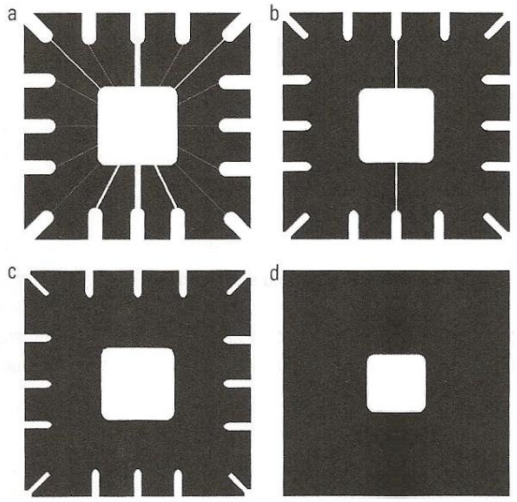


Figura 2.4.3: Usando a erosão para remover componentes da imagem: (a) imagem binária de resolução 486 × 486; (b) a (d) imagem erodida pelos elementos estruturantes quadrados de tamanhos 11 × 11, 15 × 15 e 45 × 45, respectivamente (GONZALEZ & WOODS, 2007).

Sendo A e B conjuntos de Z^2, a dilatação do conjunto A pelo elemento estruturante B pode ser definida como:

$$A \oplus B = \{z | (\hat{B})_z \cap A \neq \emptyset\} \tag{2.4.2}$$

Sendo $\widehat{B}$ a reflexão de B em torno de sua origem, a dilatação de A por B é o conjunto de todos os pontos *z* que ao transladar $\widehat{B}$ obtém-se pelo menos uma sobreposição entre A e $\widehat{B}$. Na Figura 2.4.4 é ilustrado um exemplo de dilatação. Os elementos A e B são ilustrados nas Figuras 2.4.4a e 2.4.4b, respectivamente. Na Figura 2.4.4c é ilustrado o resultado da dilatação de A por B. Nas Figuras 2.4.4d e 2.4.4e é ilustrado um elemento estruturante alongado e a dilatação de A por esse elemento, respectivamente.

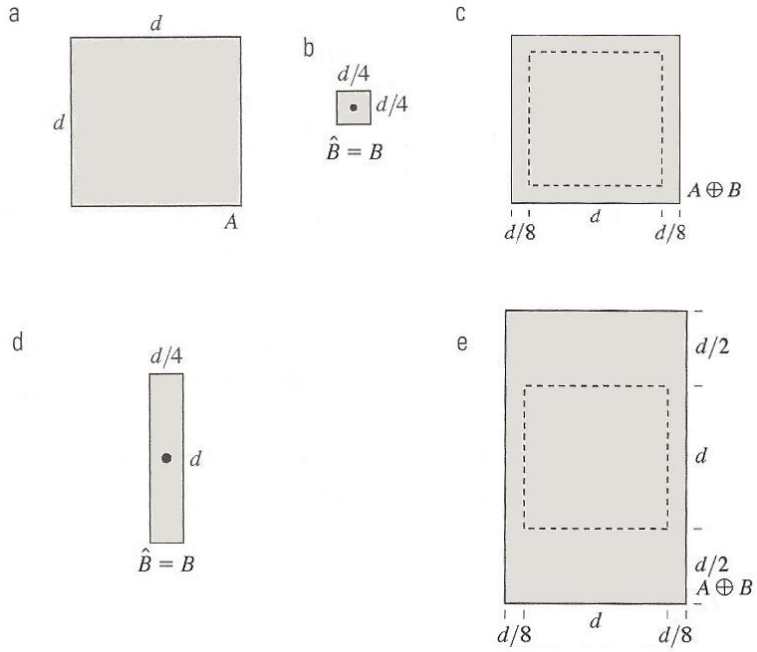


Figura 2.4.4: Operação morfológica de dilatação: (a) conjunto A; (b) elemento estruturante B; (c) dilatação de A por B; (d) elemento estruturante alongado; (e) dilatação de A pelo elemento estruturante alongado (GONZALEZ & WOODS, 2007).

Em um exemplo de aplicação, a operação de dilatação foi utilizada para restaurar caracteres quebrados em uma imagem de texto em baixa resolução, como ilustrado na Figura 2.4.5. Sabendo que os defeitos dos caracteres são formados por lacunas de no máximo dois pixels, o elemento estruturante ilustrado na Figura 2.4.5b pode ser utilizado para juntar os segmentos. O resultado da operação é ilustrado na Figura 2.4.5c.

A operação de abertura de A por B é definida como a erosão de A por B, seguida por uma operação de dilatação do resultado por B, conforme a equação:

$$A \circ B = (A \ominus B) \oplus B \tag{2.4.3}$$

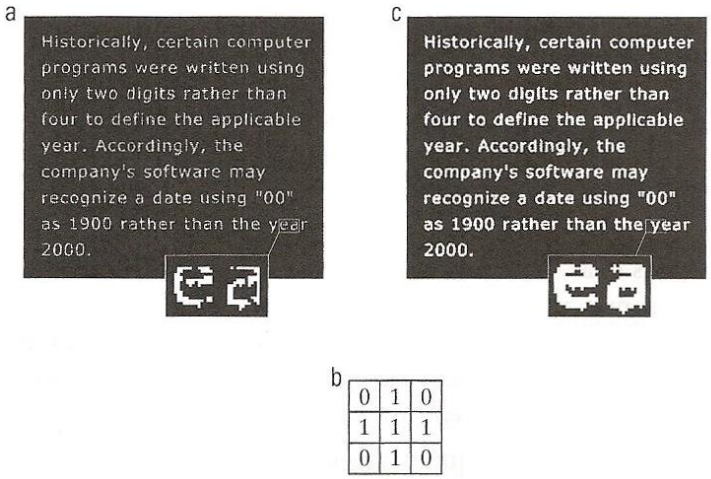


Figura 2.4.5: Usando a dilatação para restaurar caracteres quebrados: (a) texto com baixa resolução; (b) elemento estruturante; (c) resultado da operação de dilatação (GONZALEZ & WOODS, 2007).

Na Figura 2.4.6 é ilustrado a operação de abertura de um conjunto A por um elemento estruturante circular. A Figura 2.4.6b ilustra diferentes posições do elemento estruturante B transladando sobre A durante o processo de erosão. O resultado da erosão é ilustrado na Figura 2.4.6c. A Figura 2.4.6d ilustra o processo de dilatação do conjunto erodido e a Figura 2.4.6e ilustra o resultado final da operação de abertura.

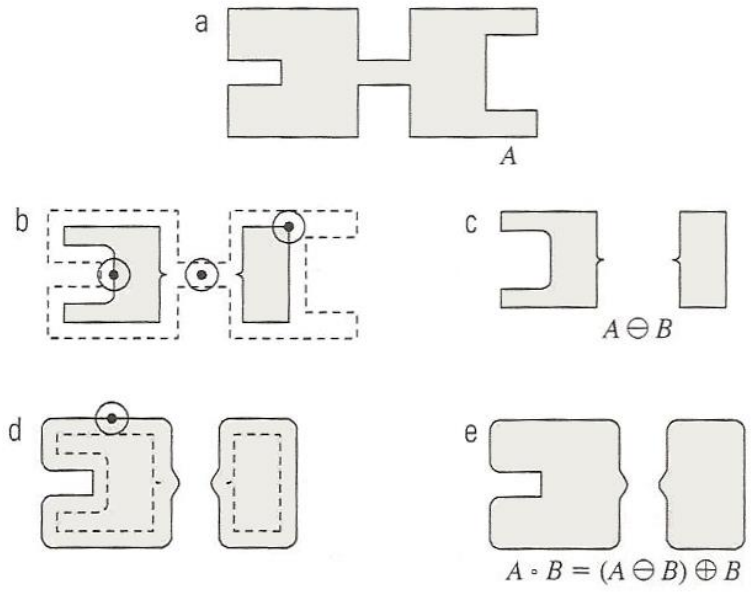


Figura 2.4.6: Operação de abertura de um conjunto A por um elemento estruturante circular: (a) conjunto A; (b) processo de erosão; (c) erosão de A; (d) processo de dilatação; (e) resultado da operação de abertura (GONZALEZ & WOODS, 2007).

A operação de fechamento de A por B é definida como a dilatação de A por B, seguida por uma operação de erosão do resultado por B, conforme a equação:

$$A \bullet B = (A \oplus B) \ominus B \tag{2.4.4}$$

Na Figura 2.4.7 é ilustrado a operação de fechamento de um conjunto A por um elemento estruturante circular. A Figura 2.4.7b ilustra diferentes posições do elemento estruturante B transladando sobre A durante o processo de dilatação. Na Figura 2.4.7c é ilustrado o resultado dessa operação. O processo de erosão e o resultado final da operação de fechamento são ilustrados nas Figuras 2.4.7d e 2.4.7e, respectivamente.

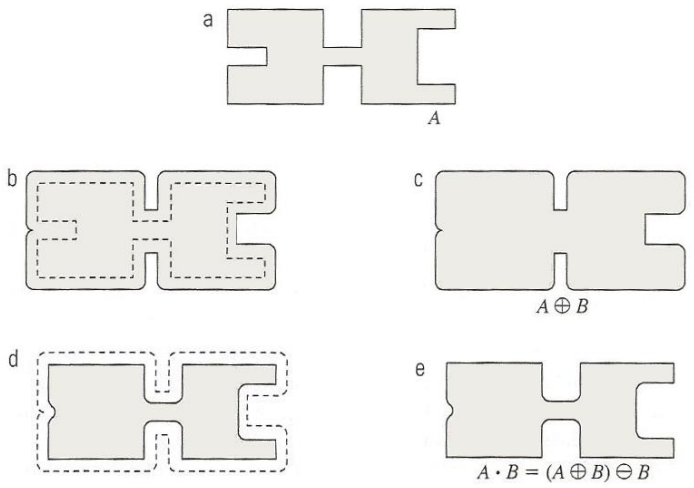


Figura 2.4.7: Operação de fechamento de um conjunto A por um elemento estruturante circular: (a) conjunto A; (b) processo de dilatação; (c) dilatação de A; (d) processo de erosão; (e) resultado da operação de fechamento (GONZALEZ & WOODS, 2007).

2.5 DISPOSITIVOS LÓGICOS PROGRAMÁVEIS

Os Dispositivos Lógicos Programáveis (PLDs) são circuitos integrados que podem ser configurados pelo usuário para implementar um circuito lógico digital. O dispositivo pode ser configurado e reconfigurado por meio de um software específico fornecido pelo seu fabricante. Os PLDs também podem ser programados no próprio sistema em que o dispositivo foi inserido por meio da tecnologia ISP *(in-system programmability*), facilitando-se assim possíveis mudanças de projetos (PEDRINO, 2003). Um PLD, de forma simplificada, pode ser visto pelo usuário como uma caixa preta que possui portas lógicas e chaves programáveis. Essas chaves permitem ao usuário conectar as portas lógicas de modo que se obtenha o circuito digital desejado.

Atualmente existem diversos PLDs disponíveis no mercado, dentre os quais o *field programmable gate array* (FPGA) tem se destacado devido a sua arquitetura de maior complexidade permitir a implementação de sistemas mais sofisticados.

A seguir são apresentadas as principais tecnologias de PLDs disponíveis no mercado:

***Programmable logic array* (PLA):** Dentre os diversos PLDs atualmente disponíveis no mercado, o primeiro a ser desenvolvido foi o PLA. A sua construção é baseada na ideia em que funções lógicas podem ser realizadas por meio de soma de produtos. O PLA compreende-se por um conjunto de portas AND conectados a um conjunto de portas OR, como ilustrado na Figura 2.5.1. As conexões entre as portas são realizadas por meio de chaves programáveis e o usuário pode configurar essas chaves de forma que o dispositivo realize a função lógica desejada (BROWN & VRANESIC, 2000).

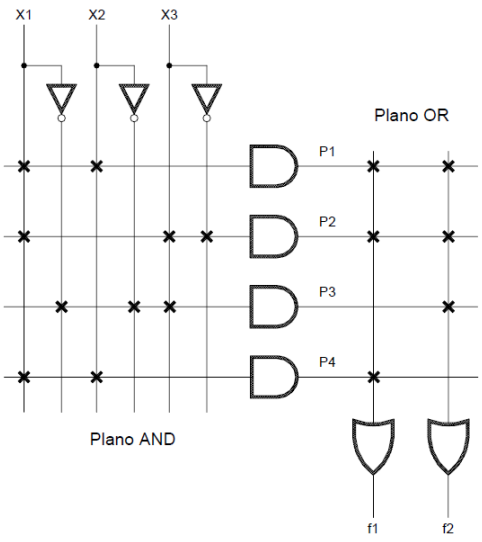


Figura 2.5.1: Programmable logic array (BROWN & VRANESIC, 2000).

***Programmable array logic* (PAL):** Dificuldades na produção de PLAs com maior desempenho levou ao desenvolvimento de uma nova tecnologia. Semelhante aos dispositivos mencionados anteriormente, os PALs também são circuitos constituídos por um grupo de portas AND e um grupo de portas OR. No entanto, tais dispositivos diferem-se no aspecto em que apenas as conexões das portas AND são programáveis. Essa característica permitiu reduzir os atrasos de propagação nos dispositivos. Alguns modelos de PALs possuem *flip-flops* conectados às saídas das portas OR, permitindo a implementação de circuitos sequenciais (PEDRINO, 2003). Na Figura 2.5.2 é ilustrado de forma simplificada o circuito de um PAL.

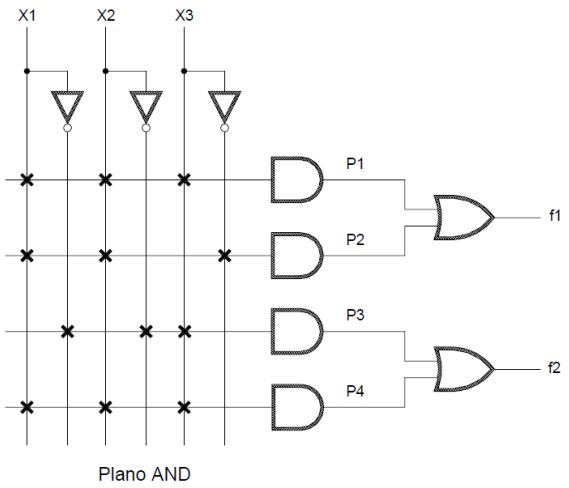


Figura 2.5.2: Programmable array logic (BROWN & VRANESIC, 2000).

***Complex programmable logic device* (CPLD):** Embora os PLAs e PALs permitam a implementação de uma ampla variedade de circuitos digitais, tais circuitos são limitados tanto em tamanho quanto em número de entradas e saídas. Os CPLDs surgem no mercado como uma solução para aplicações que exigem lógicas ainda mais sofisticadas. Esses dispositivos são caracterizados por um conjunto de blocos lógicos interconectados em um único *chip*, sendo cada bloco semelhante a um PLA ou a um PAL. Os CPLDs também possuem blocos dedicados para funções de entrada e saída de dados. Um exemplo de CPLD é ilustrado na Figura 2.5.3.

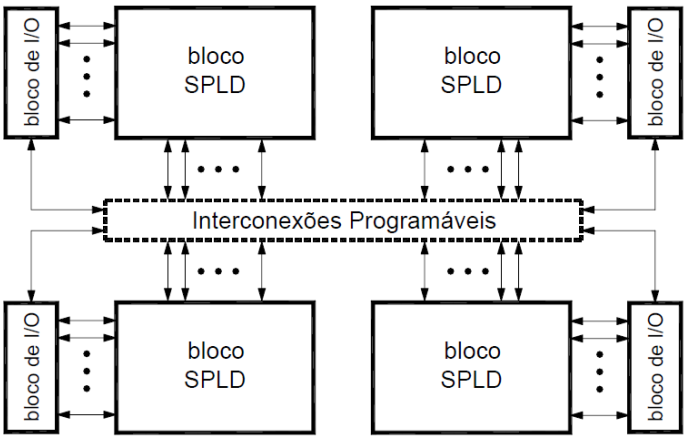


Figura 2.5.3: Complex programmable logic device (BROWN & VRANESIC, 2000).

***Field programmable gate array* (FPGA):** Introduzidos no mercado na década de 1980 pela empresa Xilinx, FPGAs são dispositivos fabricados de silício que podem ser eletricamente programados para implementar a maioria dos tipos de circuitos digitais ou sistemas (GOIS, 2011). Basicamente um FPGA é constituído de um arranjo de blocos lógicos (LB), uma rede de interconexões e blocos dedicados para entrada e saída de dados. Atualmente os FPGAs também incluem blocos sofisticados de circuitos dedicados diversos, como, por exemplo, DSPs, gerenciadores de *clock*, transceptores de alta velocidade, entre outros (HAIGANG *et al*., 2014). Os blocos lógicos são organizados em uma matriz bidimensional e uma rede de interconexões verticais e horizontais formada por chaves programáveis permite os blocos se conectarem de forma variada. A estrutura básica de um FPGA é ilustrada na Figura 2.5.4.

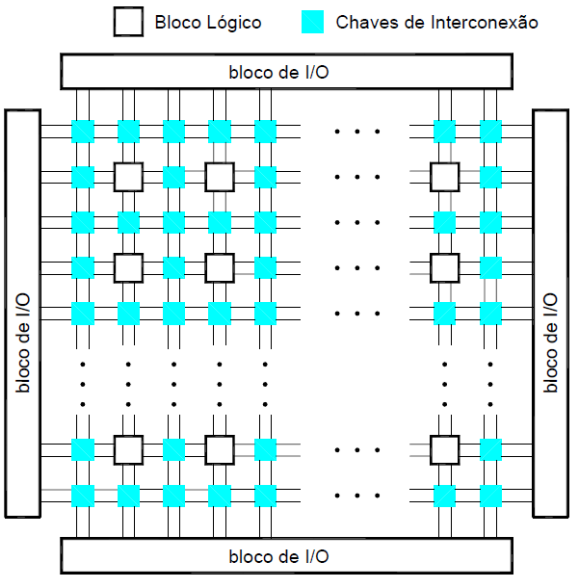


Figura 2.5.4: Field programmable gate array (BROWN & VRANESIC, 2000).

Os LBs são as principais fontes de processamento em um FPGA. Tipicamente esses elementos possuem um pequeno número de entradas e saídas que podem ser configuradas para executar pequenas lógicas combinacionais e operações sequenciais. Os FPGAs são fabricados com diferentes tipos de LBs. Os três principais tipos de blocos são descritos a seguir: *lookup Table* (LUT), *multiplexer* (MUX) e *and-inverter cone* (AIC) (HAIGANG *et al*., 2014).

O LUT é um arranjo de memória que armazena uma tabela verdade para funções de múltiplas entradas e saídas. Tipicamente o arranjo de memória é constituído de uma SRAM cujas linhas de endereço funcionam como entradas e as suas saídas provêm o valor da função lógica. Os FPGAs comerciais tipicamente possuem LUTs com quatro ou cinco entradas que correspondem respectivamente a SRAMs com 16 ou 32 células de memória (BROWN & VRANESIC, 2000). Na Figura 2.5.5 é ilustrado um exemplo de LUT com duas entradas, uma saída e quatro células de memória. Outra configuração frequente em FPGAs inclui blocos lógicos formados por um LUT e um *flip-flop*, como ilustrado na Figura 2.5.6. O *flip-flop* permite o armazenamento do resultado da lógica de forma síncrona.

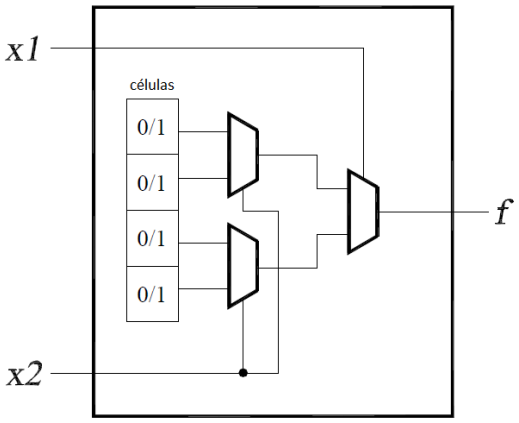


Figura 2.5.5: LUT de três entradas (BROWN & VRANESIC, 2000).

Outro tipo comum de LB, o MUX é baseado no fato em que todas as funções digitais de duas variáveis podem ser implementadas por multiplexadores do tipo 2 para 1 (2:1 mux). Esses dispositivos possuem a vantagem de prover um alto nível de funcionalidade com um número relativamente reduzido de transistores. No entanto, o custo para tal funcionalidade é um elevado número de entradas, o que exige recursos elevados de roteamento. Portanto, esses dispositivos são ideais para FPGAs que utilizam chaves programáveis de pequeno tamanho, como, por exemplo, memória *flash* e *anti-fuse* (HAIGANG *et al*., 2014). Na Figura 2.5.7 é ilustrado um exemplo de MUX.

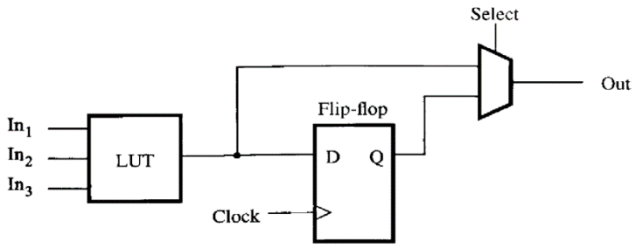


Figura 2.5.6: Bloco lógico composto por uma LUT e um flip-flop (BROWN & VRANESIC, 2000).

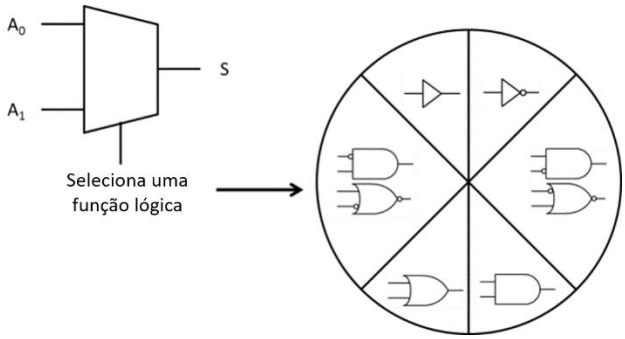


Figura 2.5.7: Bloco lógico MUX (HAIGANG et al., 2014).

Os blocos do tipo AIC são árvores de células que podem ser configuradas como portas lógicas AND ou NAND. Embora não possam implementar todas as possíveis funções lógicas de suas entradas, os AICs são mais eficientes em termos de velocidade de propagação de sinais quando comparado aos LUTs (HAIGANG *et al*., 2014). Na Figura 2.5.8 é ilustrado um exemplo de AIC.

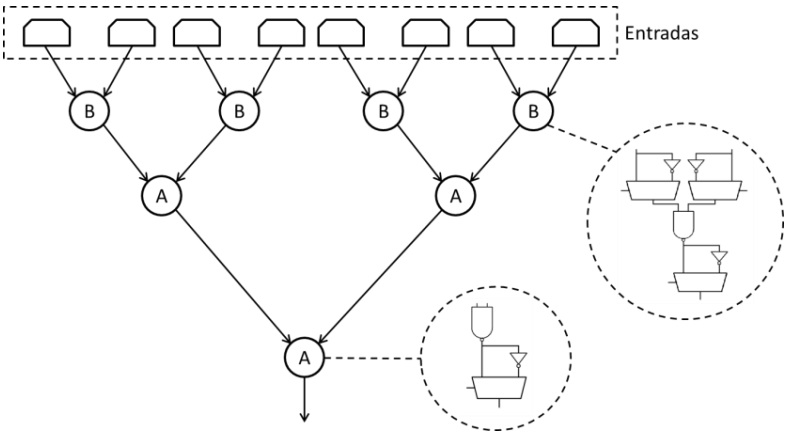


Figura 2.5.8: Bloco lógico AIC (HAIGANG et al., 2014).

Os FPGAs são configurados por meio de chaves eletricamente programáveis. As propriedades dessas chaves, como, por exemplo, resistência elétrica e capacitância, possuem um significante impacto em sua arquitetura e desempenho. A abordagem mais comum na construção dessas chaves inclui *static random access memory* (SRAM), embora dispositivos baseados em memória *flash* e *programmable read-only memory* (PROM) também podem ser encontrados.

A base para a tecnologia de FPGAs baseadas em SRAM é a célula de memória estática. A célula é constituída por seis transistores, como ilustrado na Figura 2.5.9. Os transistores M5 e M6 permitem o acesso à célula para ambas as operações de leitura e escrita e os dois inversores *cross-coupled* formados pelos transistores M1, M2, M3 e M4 retém a informação na célula. Os dispositivos fabricados com essa tecnologia também permitem uma rápida reconfiguração no próprio circuito. Algumas desvantagens dessa tecnológica incluem maior tamanho da célula de memória e volatilidade. A configuração em dispositivos com essa tecnologia deve ser realizada sempre que o dispositivo é ligado, tornado necessário o uso de uma memória externa. A compatibilidade desses dispositivos com a mais recente tecnologia CMOS permite a fabricação de dispositivos com maior velocidade, maior nível de integração e menor consumo de energia (HAIGANG *et al*., 2014).

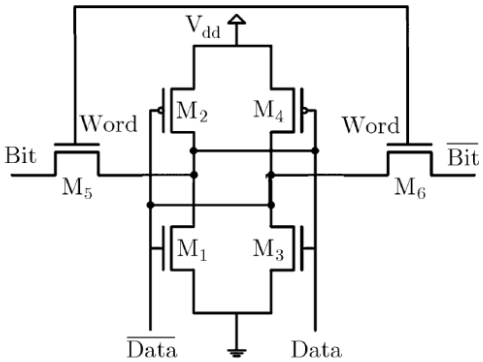


Figura 2.5.9: Célula de memória estática (HAIGANG et al., 2014).

Em dispositivos baseados em tecnologia de memória *flash* as células são do tipo não volátil e, portanto, permitem que as configurações sejam mantidas mesmo quando o dispositivo é desligado, tornando desnecessário o uso de dispositivos externos para armazenamento dos dados. Os dispositivos fabricados com essa tecnologia também

apresentam uma melhor eficiência de área, uma vez que as suas células são constituídas apenas por dois transistores, como ilustrado na Figura 2.5.10. Como principal desvantagem, dispositivos com esta tecnologia apresentam limitações quanto ao número de programações, tipicamente de 500 a 1000 ciclos.

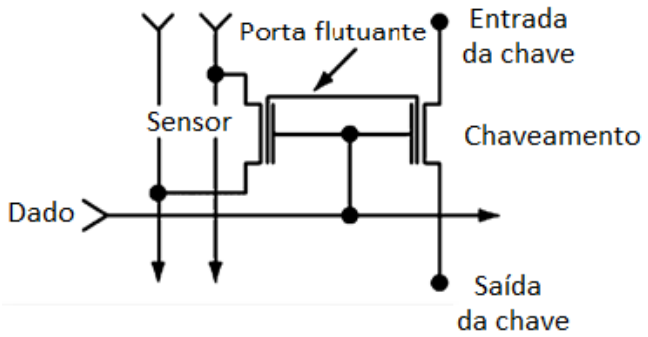


Figura 2.5.10: Célula baseada em memória flash (HAIGANG et al., 2014).

Os FPGAs baseados em tecnologia *anti-fuse* são caracterizados por estruturas que apresentam alta resistência elétrica em condições normais e, convenientemente, uma série de *links* de baixa resistência pode ser criada para formar as conexões necessárias a um determinado circuito. Contudo, o processo de construção do *link* é irreversível. Uma vez estabelecido um *link* torna-se permanente, permitindo que a programação do dispositivo seja realizada apenas uma única vez. Esses FPGAs requerem um processo de fabricação CMOS não padrão e por esta razão estão várias gerações atrás de dispositivos baseados em SRAM e memória *flash* (HAIGANG *et al.*, 2014). Como vantagem, essa tecnologia de programação possui baixa resistência e capacitância parasita, permitindo um melhor desempenho dos dispositivos. Na Tabela 2.5.1 são apresentadas algumas das principais características das tecnologias de programação citadas.

O surgimento dos FPGAs, entre outros dispositivos reconfiguráveis, deu origem ao um novo paradigma para computação denominado de computação reconfigurável. Os sistemas reconfiguráveis possuem um alto grau de paralelismo, o que os tornam dispositivos ideais para aplicações de alto desempenho. Os FPGAs vêm demonstrando

resultados satisfatórios em aplicações que envolvem sistemas computacionalmente complexos e aplicações que exijam processamento em tempo real (BOBDA, 2007).

Parâmetro	Tecnologia de Programação		
	SRAM	Flash	Anti-fuse
Não-volátil	×	✓	✓
Reprogramável	✓	✓	×
Transistores por célula	5 ou 6	1 ou 2	-
Tempo de Leitura/Escrita	1 ns/1 ns	100 ns/200 ns	-
Tecnologia de fabricação	20 nm	65 nm	130 nm
Ciclo de programação	∞	500 ~ 1000	1

Tabela 2.5.1: Comparação entre as tecnologias de programação de FPGA (HAIGANG et al., 2014).

2.6 CONSIDERAÇÕES FINAIS

Nesse capítulo foram apresentados os principais fundamentos teóricos necessários para o entendimento do presente trabalho. No capítulo seguinte são apresentados os principais trabalhos relacionados à navegação de robôs móveis agrícolas. Alguns dos trabalhos mencionados abordam métodos de visão computacional relativamente simples que utilizam características visuais comuns em cenas de navegação de robôs em linhas de plantio, tais métodos foram utilizados como base para o desenvolvimento do circuito proposto no presente trabalho.

3 REVISÃO BIBLIOGRÁFICA

3.1 CONSIDERAÇÕES INICIAIS

Na área de Visão computacional, estuda-se a forma como uma máquina percebe o meio a sua volta, extraindo informações significativas a partir de imagens capturadas por câmeras de vídeo, sensores, *scanners*, entre outros dispositivos. Os trabalhos de visão computacional aliados à inteligência artificial foram iniciados por volta da década de 70. Na época acreditava-se que em pouco tempo seria possível representar o sentido da visão biológica de forma completa em máquinas. Na década seguinte os pesquisadores verificaram que haveria uma complexidade muito maior para tal propósito, principalmente pela falta de informações e modelos que representassem a forma como as imagens são interpretadas no cérebro humano (DE MILANO & HONORATO, 2010).

O olho humano consegue perceber e interpretar objetos em uma imagem de forma muito rápida. Isso acontece no córtex visual do cérebro, uma das partes mais complexas no sistema de processamento do cérebro. Alguns cientistas concentram seus estudos na tentativa de entender o funcionamento dessa parte do cérebro, para então trazer tais ideias para a visão computacional. Dessa forma, a visão computacional fornece ao computador uma infinidade de informações a partir de imagens e vídeos, de forma que o computador consiga executar tarefas inteligentes, simulando e aproximando-se da inteligência humana (DE MILANO & HONORATO, 2010).

O emprego de sistemas computacionais na agricultura de precisão fomenta a automação de processos e tarefas aplicadas nesta área, precisamente voltadas à inspeção e análise de culturas agrícolas e locomoção de robôs móveis (LULIO, 2011). A navegação autônoma de robôs agrícolas móveis por métodos de visão computacional é uma das principais questões a ser considerada neste contexto e por esta razão é um assunto importante a ser abordado no presente trabalho.

3.2 VISÃO COMPUTACIONAL PARA NAVEGAÇÃO DE ROBÔS MÓVEIS AGRÍCOLAS

Embora os sistemas de visão computacional tenham chamado a atenção de pesquisadores desde o início da computação, esta ciência ainda é considerada muito recente. A maioria das aplicações que utilizam visão computacional é provinda de outras áreas de pesquisa, resolvendo problemas particulares de forma específica. Por esta razão não existe um modelo padrão bem definido para implementação em aplicações que utilizam visão computacional. Desta forma, dependendo da aplicação, o projetista escolhe dentre as várias técnicas existentes a mais adequada para a solução do problema (BONATO, 2004). Em consenso, os sistemas de visão computacional envolvem o reconhecimento de objetos em imagens e a transformações destes objetos em informações (DE MILANO & HONORATO, 2010). As tarefas envolvidas nesse processo estão divididas da seguinte maneira:

- Aquisição de imagem: é o primeiro passo em um sistema de visão. Trata-se do processo de aquisição de uma imagem ou de um conjunto de imagens a partir de sensores para obter uma representação da informação visual da cena. Normalmente tal representação é construída a partir de uma fonte de radiação, como, por exemplo, calor, raios-X, luz visível e micro-ondas (DE MILANO & HONORATO, 2010).

- Pré-processamento: consiste no processo realizado antes de se obter informações de uma imagem, de forma a aplicar métodos específicos que facilitem a identificação de um objeto, como, por exemplo, o destaque de contornos e bordas (DE MILANO & HONORATO, 2010).

- Extração de características: é o processo de extração de características matemáticas que compõem uma imagem, tais como, textura, formato e movimento (DE MILANO & HONORATO, 2010).

- Detecção e segmentação: nesse processo, divide-se a imagem em regiões que devem corresponder às áreas de interesse da aplicação (DE MILANO & HONORATO, 2010).
- Processamento de alto nível: nesse processo, inclui-se a validação da satisfação dos dados obtidos, a estimativa de parâmetros sobre a imagem e a classificação dos objetos obtidos em diferentes categorias (DE MILANO & HONORATO, 2010).

As aplicações de navegação de robôs agrícolas envolvem diferentes técnicas de processamento de imagens. Em sua grande maioria essas técnicas são baseadas em algoritmos para o processamento de imagens bidimensionais, embora também existam métodos para navegação por meio de imagens tridimensionais (MOUSAZADEH, 2013).

Para a navegação a partir de imagens tridimensionais existem basicamente duas técnicas: visão estéreo e sistemas de varredura a laser (*laser scanner 3D*). No primeiro caso a construção da imagem tridimensional da cena é obtida por meio do processamento de um par de imagens capturadas simultaneamente com um pequeno deslocamento lateral. Os objetos que compõe a cena são representados no par de imagens com um deslocamento proporcional à distância que eles se apresentam da câmera. Conhecendo esse deslocamento é possível aferir a profundidade relativa de cada objeto representado na cena. A obtenção da estimativa de profundidade por meio dessa técnica possui algumas limitações inerentes. O cálculo da disparidade se torna impossível para objetos próximos a câmera a partir do ponto em que eles não podem ser representados por ambas as lentes simultaneamente. Para o caso de objetos muito distantes o cálculo também não é possível porque as diferenças entre as imagens se tornam nulas. As câmeras estéreo também possuem as limitações típicas de câmeras comuns, tal como a necessidade de iluminação (MENDES, 2012).

Em sistemas de varredura a *laser* a estimativa de profundidade é obtida por intermédio da reflexão de pulsos de *laser* emitidos com o auxílio de um espelho de varredura. Basicamente existem três métodos para medição a *laser*: triangulação, tempo de percurso (*time of light*) e diferença de fase. Os sensores baseados no princípio de triangulação possuem um ou mais sensores CCDs e uma fonte *laser*. Dependendo da distância do objeto o pulso de *laser* é refletido para diferentes regiões da imagem

capturada. Dessa forma, a distância até o objeto é calculada a partir do conhecimento das posições do *laser* e da câmera (GONÇALES, 2007).

No método de tempo de percurso o sensor emite pulsos de *laser* permitindo o instrumento medir as distâncias dos objetos a partir do conhecimento do tempo decorrido pelo percurso do *laser* e a velocidade de propagação do pulso no meio. O terceiro método utiliza um feixe de *laser* normalmente dividido em três comprimentos de ondas diferentes. O cálculo das coordenadas dos pontos refletidos pelos objetos é realizado por meio da comparação de fase entre os diferentes comprimentos de onda.

A representação tridimensional do ambiente a partir dos sensores mencionados pode ser realizada por meio do mapa de profundidade. O mapa consiste em uma imagem bidimensional com informações de distância, ou seja, cada pixel da imagem representa a distância entre a câmera e o ponto correspondente. As informações também podem ser representadas por meio da nuvem de pontos. Nesta representação é utilizado o sistema de coordenadas cartesiano (z, y e z) no qual cada ponto que representa o ambiente possui sua posição neste sistema (CORREA, 2013).

Nas Figuras 3.2.1(a) e 3.2.1(b) são ilustrados exemplos de mapa de profundidade e nuvem de pontos.

(a) (b)

Figura 3.2.1: Representação de imagem tridimensional: (a) mapa de profundidade; (b) nuvem de pontos (CORREA, 2013).

Em aplicações de navegação autônoma de veículos a partir de imagens bidimensionais o modelo de câmera comumente utilizada é o *pinhole model*, ou modelo pontual. Esse modelo baseia-se na passagem dos raios luminosos por um pequeno

orifício na entrada da câmera e na sua projeção em uma superfície plana, como ilustrado na Figura 3.2.2. Nesse processo as descrições dos objetos da cena são transferidas para o plano de projeção utilizando um método de conversão de coordenadas por meio de operações geométricas simples de translação e rotação. A conversão dos sistemas de coordenadas permite projetar objetos tridimensionais no plano de projeção bidimensional eliminando as informações de profundidade da cena (NERIS, 2008). A partir das imagens obtidas por esse método diferentes técnicas de visão computacional são aplicadas de acordo com as características específicas de uma dada aplicação. O modelo de visão empregado em aplicações de navegação de máquinas agrícolas através dos corredores formados entre linhas de plantio é um exemplo dessas aplicações. Nessa abordagem uma solução comumente adotada é o emprego de técnicas de segmentação de imagem para o cálculo da estimativa da linha de navegação por meio da extração de características comuns da área de plantio e a área guiável.

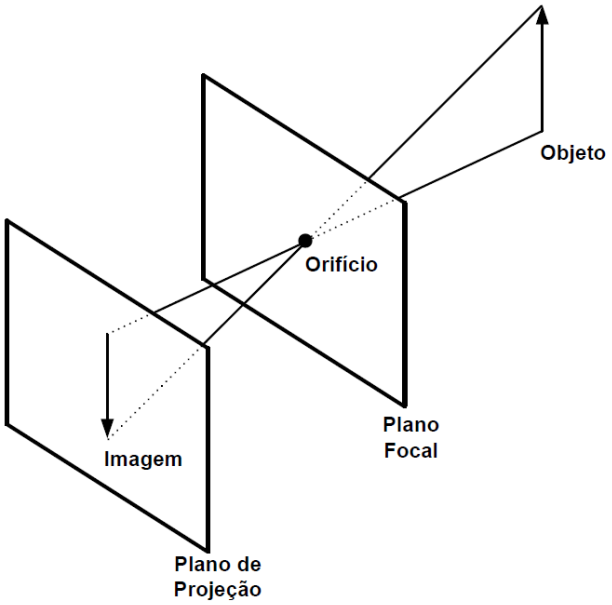


Figura 3.2.2: Modelo de câmera (NERIS, 2008).

A seguir, são apresentados alguns trabalhos relevantes em aplicações de navegação de robôs agrícolas com o propósito de esclarecer algumas das principais metodologias adotadas para navegação por meio de visão computacional.

Em Xue (XUE *et al*., 2012) um método de visão computacional foi desenvolvido para a navegação de um robô móvel agrícola entre as linhas de plantio em culturas de milho. As linhas de orientação foram detectadas utilizando um algoritmo de

processamento de imagem, empregando características morfológicas. O robô é guiado através das linhas de orientação por meio de um controle de lógica *fuzzy*.

Para o cálculo das linhas de navegação o primeiro passo distingue as plantas da região do solo por meio da segmentação de imagem por classificação de cores, gerando uma imagem binária. Posteriormente, as operações morfológicas são aplicadas para determinar os limites da área de plantio à direita e esquerda. Nesta etapa as operações morfológicas também permitem a redução de ruído presente na imagem preservando a forma e tamanho dos objetos.

Na Figura 3.2.3 são ilustradas as tarefas do processamento de imagem em uma cena real de navegação.

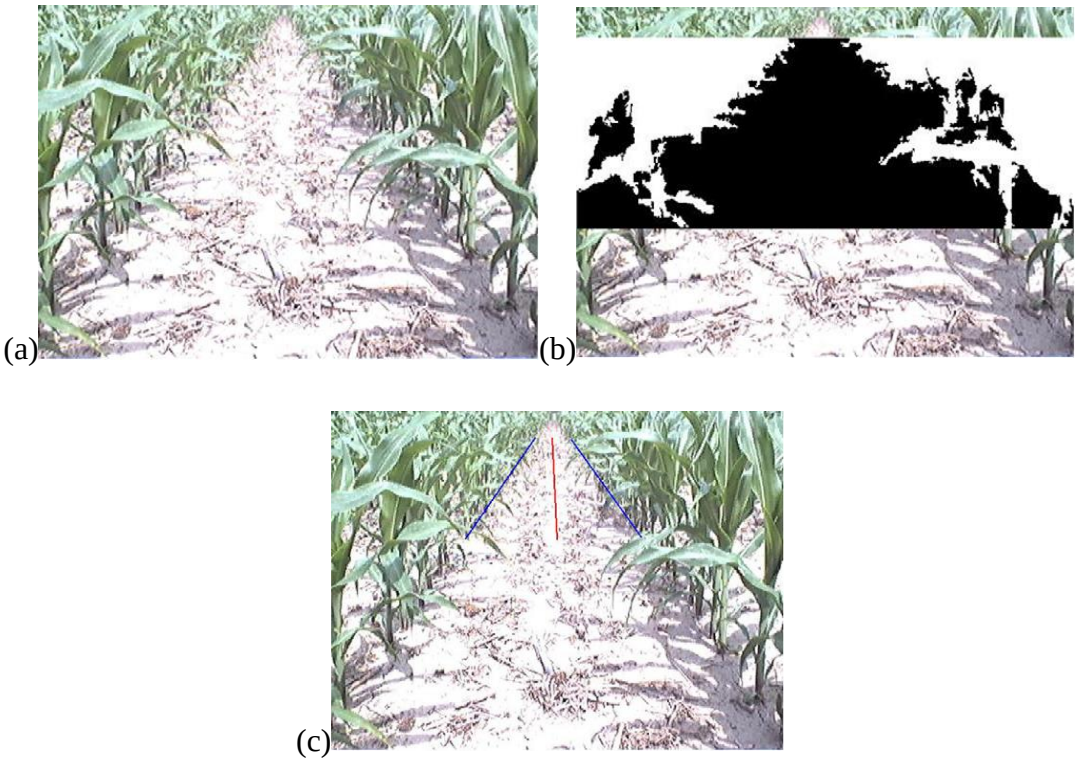


Figura 3.2.3: Navegação de robô móvel em cultura de milho: (a) imagem original; (b) imagem binária; (c) imagem processada (XUE et al., 2010).

A plataforma robótica utilizada neste trabalho foi a *AgTracker* desenvolvida pelo departamento de agricultura e engenharia biológica da Universidade de Illinois. Para esta aplicação o robô foi equipado com um *laptop* equipado com um processador AMD Turion 64 de dois núcleos e uma câmera digital de baixo custo com resolução de 640 por 480 pixels. A plataforma *AgTracker* é ilustrada na Figura 3.2.4. Os resultados

mostrados pelo autor demonstraram que a metodologia utilizada foi capaz de guiar o robô através de uma cultura de milho com precisão e estabilidade aceitável sem danificar o plantio. O autor enfatizou que o erro máximo de orientação obtido nos testes foi de 15,8 mm.

Figura 3.2.4: Plataforma robótica AgTracker (XUE et al., 2010).

Outra metodologia para navegação de robôs agrícolas através de linhas de cultivo em culturas de arroz foi apresentada em Choi (CHOI *et al.*, 2015). O algoritmo proposto identifica as regiões centrais das plantas utilizando a característica morfológica na qual as folhas normalmente convergem sua orientação no sentido da região do tronco central. Segundo o autor, encontrar a região de convergência permite aumentar a precisão da estimativa da linha de navegação.

Para esta finalidade, o primeiro processo cria um conjunto de retas virtuais a partir do processo de extração de bordas da imagem original das plantas. As retas são estendidas de forma que as suas intersecções permitem calcular uma estimativa da região central de cada planta, como ilustrado na Figura 3.2.5. Posteriormente, as linhas de orientação são estimadas a partir da região central de cada planta por meio do método de regressão robusta modificada. Os resultados apresentados indicam o erro do ângulo de inclinação de navegação com a linha de orientação inferior a 1°, o que o autor define como um bom resultado.

Na Figura 3.2.6 é ilustrado o resultado do cálculo de extração das linhas de orientação por meio da regressão robusta e regressão por mínimos quadrados.

Em Hiremath (HIREMATH *et al.*, 2014) a metodologia proposta permitiu ao robô navegar em duas situações: na primeira situação o robô é guiado através do

corredor formado entre as linhas de plantio; na segunda situação o robô realiza uma manobra sobre a cabeceira e retorna no próximo corredor em condições normais. O trajeto realizado pelo robô durante a fase de teste é ilustrado na Figura 3.2.7. O método é baseado em um filtro de partículas onde um modelo de imagem de partícula é comparado diretamente da imagem de medição após a etapa de processamento elementar.

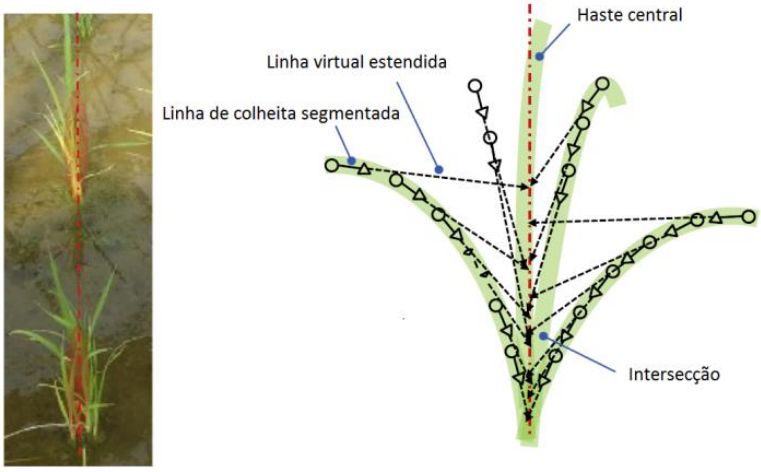


Figura 3.2.5: Concepção do algoritmo para extração da linha de navegação (CHOI et al., 2015).

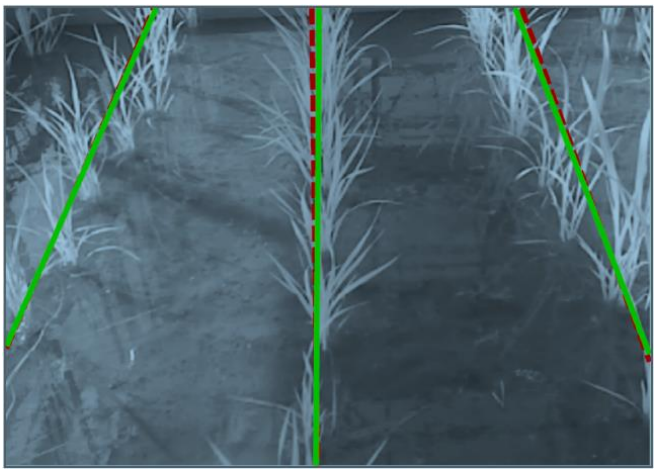


Figura 3.2.6: Linhas de orientação. Linha vermelha tracejada: regressão por mínimos quadrados, linha verde contínua: regressão robusta (CHOI et al., 2015).

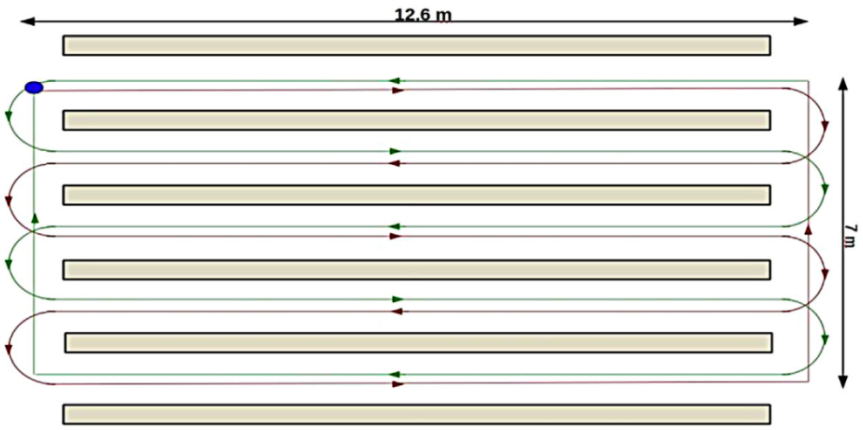


Figura 3.2.7: Trajeto realizado pelo robô em ambiente semiestruturado. O ponto azul indica a posição de partida (HIREMATH et al., 2014).

O autor destacou que a utilização do filtro de partículas minimizou os efeitos da incerteza que ocorrem no processo de extração de características. Os testes realizados comprovaram a robustez do algoritmo por intermédio de experiências em campos com diferentes padrões de linhas, diferentes tamanhos de plantas e condições de iluminação diversificadas. O método apresentado nesse trabalho foi desenvolvido em linguagem C# utilizando a biblioteca OpenCV para o processamento das imagens. O robô foi equipado com um computador Intel Core2 Duo de 2,4 GHz e uma câmera RGB µEye UI-1220 com resolução de 752 por 480 pixels.

Em Klaser (KLASER, 2014) foi apresentado um sistema de navegação autônomo com foco em ambientes não estruturados. A navegação é realizada nesse método por meio do sistema de posicionamento global (GPS) associado ao mapeamento local. A visão computacional nesse trabalho possui a função de detectar os obstáculos encontrados no trajeto. As informações obtidas pelo sistema de visão computacional são então utilizadas por um sistema de ocupação probabilístico (OctoMap) permitindo o planejamento da trajetória e o desvio de obstáculos durante a navegação.

O sistema de visão computacional consiste primeiramente no cálculo das disparidades dos pixels entre o par de imagens obtidas por uma câmera estéreo. Para esta finalidade é aplicado o algoritmo de *Block Matching* que consiste na busca local de correspondências entre os pixels dentro de uma janela. Segundo o autor, o método escolhido para o cálculo do mapa de disparidade permitiu um custo computacional compatível com aplicações em tempo real. Na Figura 3.2.8 é mostrado o resultado obtido nessa etapa do processamento.

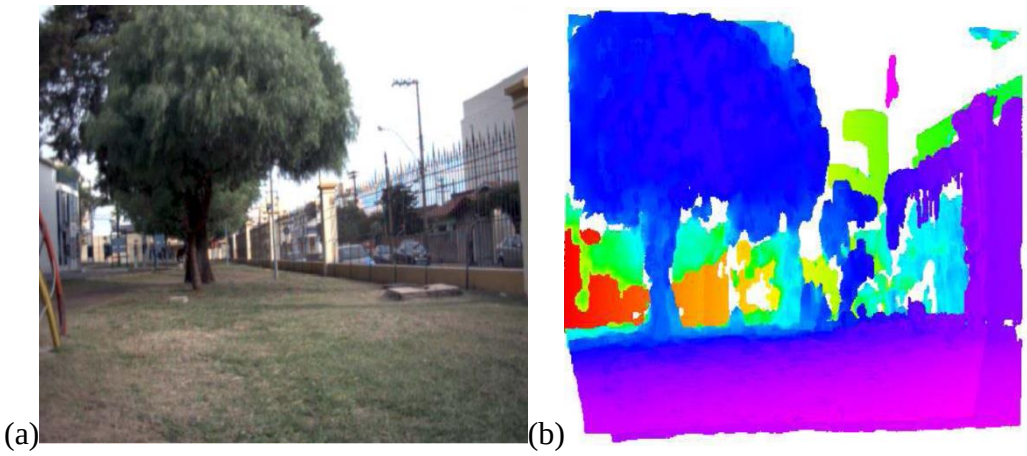


Figura 3.2.8: Extração do mapa de disparidade: (a) Imagem obtida pela câmera estéreo (somente esquerdo); (b) cálculo da disparidade entre o par de imagens (KLASER, 2014).

Após o mapa de disparidade ser computado, as coordenadas tridimensionais para cada ponto são estimadas com base nos parâmetros da câmera. O resultado obtido neste processo é uma nuvem de pontos tridimensionais representando a cena observada pelas câmeras.

As nuvens de pontos obtidas a partir de câmeras estéreo apresentam um ruído característico. Esse ruído causa um espelhamento ondular no eixo de profundidade dos objetos que representam a cena. Tal efeito é intensificado em objetos mais distantes, visto que a disparidade entre os pontos diminui à medida que a distância aumenta. Para o tratamento desse ruído o autor desenvolveu um filtro que normaliza as distâncias em relação ao eixo de profundidade. O filtro consiste no cálculo da média e desvio padrão das distâncias dos pontos de cada estrutura agrupada e ajuste da distância dentro do intervalo de um desvio padrão. Na Figura 3.2.9a é possível observar o ruído nos pontos que representam a estrutura de uma arvore e na Figura 3.2.9b é observado o resultado obtido após a aplicação do filtro.

A nuvem de pontos obtida a partir das disparidades é uma representação não estruturada dos dados da cena. Sendo assim, a nuvem contém pontos relacionados a todas as estruturas da cena. Dessa forma, um método de segmentação é aplicado a fim de separar as estruturas a partir de uma busca por vizinhos em regiões com densidades maiores de pontos, separando assim cada estrutura em conjuntos de pontos.

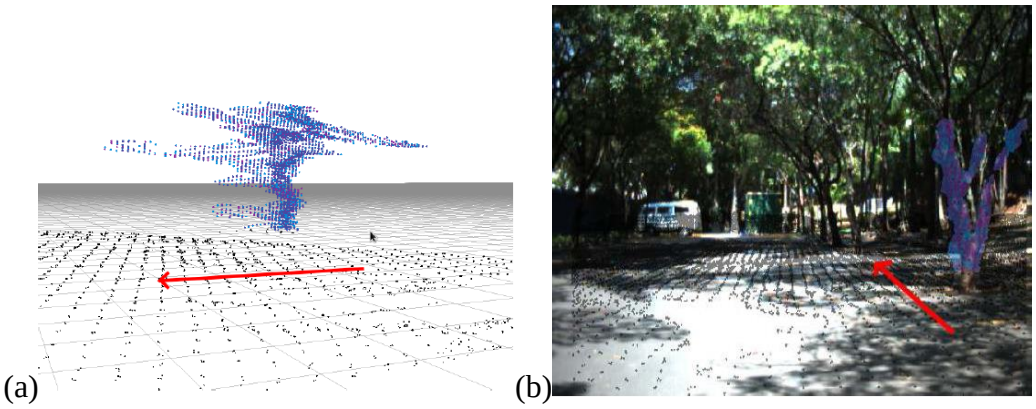


Figura 3.2.9: Aplicação do filtro para redução de ruído: (a) ruído ondular na nuvem de pontos; (b) projeção da nuvem filtrada sobre a imagem real (KLASER, 2014).

Para a validação do método o autor primeiramente realizou a simulação do projeto permitindo a execução de testes de forma mais versátil e rápida, sem expor o equipamento a possíveis danos. Para essa finalidade foi utilizado dois computadores Intel Core I5 com 4 Gb de memória ligados em rede. Para a validação em ambiente real o veículo foi equipado com um computador Intel I7 com 16 Gb de memória. Para o autor, a utilização do OctoMap para representar probabilisticamente a ocupação espacial permitiu que os ruídos apresentados pelo sistema de percepção e localização fossem incorporados à operação do sistema sem um maior prejuízo na execução das tarefas atribuídas. Desta forma, o mapa de navegabilidade gerado permitiu o planejamento da trajetória e a navegação autônoma do veículo em ambientes externos não estruturados.

Outro trabalho relacionado à visão estéreo foi apresentado em Zhang (ZHANG *et al.*, 2005). O algoritmo desenvolvido nesse trabalho foi aplicado para o reconhecimento do campo de colheita de arroz. A intensidade do nível de cinza e a correlação foram integrados para produzir as disparidades em imagens estéreo. A superfície do terreno e o plantio de arroz foram conjugados como dois planos acidentados e suas disparidades oscilaram numa baixa faixa de variação. As bordas contínuas e descontínuas do plantio de arroz foram inicialmente detectadas e posicionadas nas imagens. Em seguida, os pontos posicionados além da borda foram combinados para obter os valores de disparidade utilizando o método de correlação de área. A partir do mapa de disparidade, as coordenadas tridimensionais da câmera foram calculadas e as coordenadas do veículo obtidas pela multiplicação das coordenadas da

câmera e uma fórmula de transformação. Na Figura 3.2.10 é ilustrado uma imagem do campo de colheita de arroz e o mapa de disparidade correspondente.

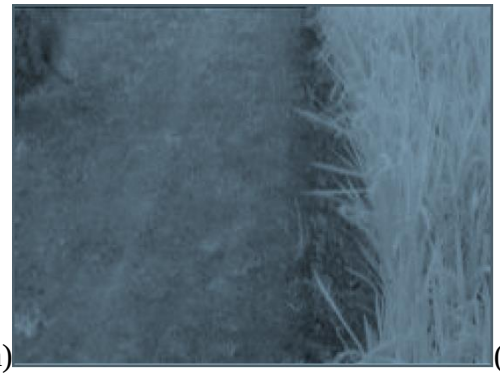
(a)  (b)

Figura 3.2.10: Campo de colheita de arroz: (a) imagem original; (b) mapa de disparidade (ZHANG et al., 2005).

O autor concluiu que o método de visão estéreo baseado na detecção de bordas no mapa de disparidade apresentado em seu trabalho pode fornecer informações confiáveis para aplicações de navegação autônoma em campos de colheita de arroz. A precisão na estimativa dos pontos foi de 95% para a região do solo e de 70% para a região de plantio.

Em Jiang (JIANG *et al.*, 2010) foi proposto outro algoritmo de visão computacional para a navegação autônoma de robôs agrícolas em linhas de cultura. Para o autor, diversos trabalhos anteriores realizaram estudos concisos propondo metodologias para a aplicação. No entanto, os requerimentos computacionais são uma das principais limitações em aplicações de navegação autônoma. Portanto, o objetivo do seu trabalho foi propor um método capaz de guiar um robô agrícola em linhas de cultura em tempo real e em condições adversas de iluminação.

O método apresentado em Jiang foi dividido em quatro etapas: transformação em escala de cinza, limiarização da imagem, divisão da imagem e detecção das linhas de cultura.

Na primeira etapa a imagem colorida foi transformada em uma imagem em escala de cinza por meio de um processo que busca enfatizar a cor verde. Para esse propósito o modelo *2G-R-B* foi utilizado conforme demonstrado na Equação 3.2.1.

$$pixe(x,y) = \begin{cases} 0 & 2G \leq R + B \\ 2G - R - B & outros \\ 255 & 2G \geq R + B + 255 \end{cases} \quad (3.2.1)$$

G, *R* e *B* são respectivamente os componentes verde, vermelho e azul do espaço de cores RGB e *pixel (x, y)* é o valor da escala de cinza para um determinado ponto nas coordenadas *x* e *y*. Na Figura 3.2.11 é ilustrado o resultado da transformação realizada.

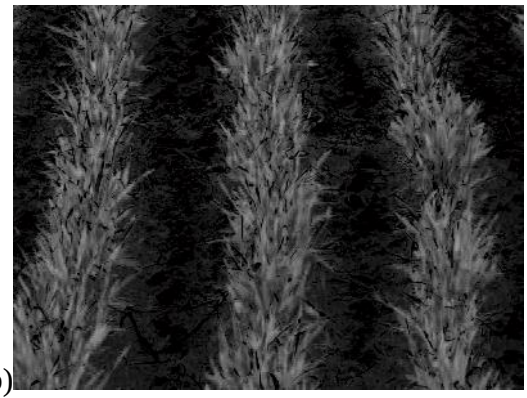
(a) (b) 

Figura 3.2.11: Transformação em escala de cinza: (a) imagem original; (b) imagem transformada (JIANG et al., 2010).

Na etapa seguinte a imagem em escala de cinza sofre um processo de limiarização a fim de reduzir as interferências causadas por ruídos e reduzir o custo computacional das etapas sequentes. O método de Otsu (HUI, 2006) é citado pelo autor como uma das melhores opções para a tarefa de limiarização. O seu princípio básico é procurar pelo valor de limiar ideal que divide o histograma da imagem em dois grupos com menor variância possível. Na Figura 3.2.12 é ilustrado o resultado da limiarização por meio do método de Otsu.

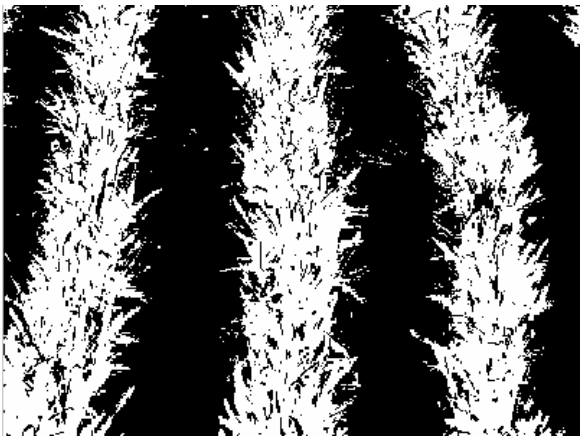


Figura 3.2.12: Imagem limiarizada através do método de Otsu (JIANG et al., 2010).

A transformada de *Hough* é frequentemente utilizada para a detecção das linhas de plantio. No entanto, segundo o autor, o custo computacional para essa tarefa é relativamente elevado em aplicações que necessitam de processamento em tempo real. Dessa forma, o autor propôs a divisão da imagem binária gerada na etapa anterior em um conjunto de fatias horizontais como ilustrado na Figura 3.2.13. Para cada fatia da imagem foram estimados os pontos centrais das linhas de plantio reduzindo o número de dados utilizados na etapa de detecção das linhas de cultura.

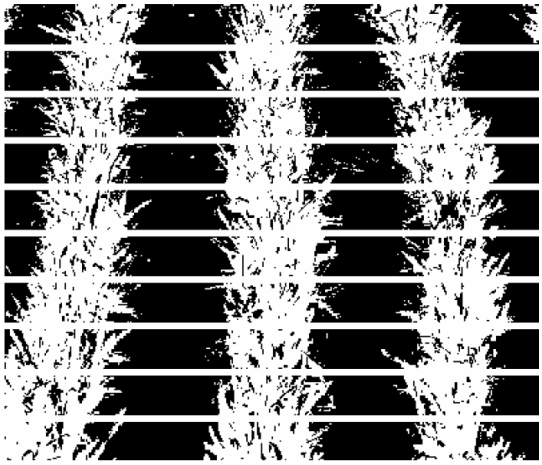


Figura 3.2.13: Imagem dividida em fatias horizontais (JIANG et al., 2010).

Na sequência são apresentadas as etapas realizadas no cálculo dos pontos centrais em uma fatia da imagem. Os pontos centrais das demais fatias foram calculados seguindo o mesmo procedimento.

Para o cálculo dos pontos centrais, a soma das colunas *S(j)* e o valor médio *m* da fatia foram calculados segundo as equações 3.2.2 e 3.2.3.

$$S(j) = \sum_{i=1}^{h} I(i,j) \qquad J = 1, 2, \dots M \tag{3.2.2}$$

$$m = \frac{1}{M}\sum_{j=1}^{M} S(j) \qquad j = 1, 2, \dots M \tag{3.2.3}$$

Onde $M \times N$ corresponde à resolução da imagem original, $M \times h$ corresponde ao tamanho da fatia e *I(i,j)* corresponde ao valor da imagem limiarizada nas coordenadas *i* e *j*.

A quarta fatia da imagem limiarizada foi utilizada no exemplo do cálculo dos pontos centrais. Na Figura 3.2.14a é ilustrado a fatia da imagem limiarizada e na Figura 3.2.14b é ilustrado o gráfico da soma das colunas *S(j).*

Na próxima etapa o valor médio *m* foi utilizado para limiarizar *S(j)* conforme a equação:

$$S(j) = \begin{cases} h & S(j) \geq m \\ 0 & outros \end{cases} \tag{3.2.4}$$

Na Figura 3.2.14c é ilustrado o resultado da limiarização de *S(j).* As próximas etapas foram a diferenciação e classificação das colunas que representam as bordas laterais a esquerda e a direita da linha de plantio. Esses processos foram ilustrados nas Figuras 3.2.14d e 3.2.14e respectivamente. O ponto médio entre as bordas laterais correspondem aos pontos centrais da linha de plantio conforme ilustrado na Figura 3.2.14f.

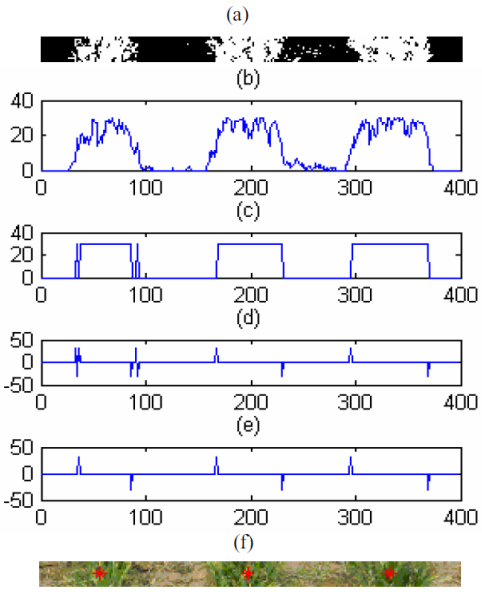


Figura 3.2.14: Cálculo da estimativa dos pontos centrais das linhas de plantio: (a) quarta fatia da imagem limiarizada; (b) soma das colunas; (c) limiarização; (d) diferenciação; (e) classificação das bordas; (f) resultado do processo (JIANG et al., 2010).

Após o cálculo dos pontos centrais em todas as fatias da imagem, a última etapa do sistema de visão computacional aplica a transformada de *Hough* nos pontos centrais para identificar as linhas de cultura. O resultado obtido nessa etapa é ilustrado na Figura 3.2.15.

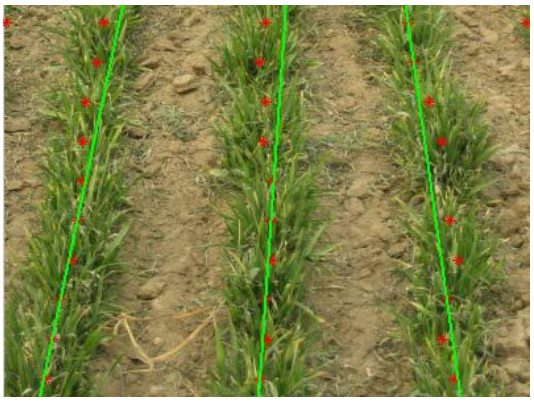


Figura 3.2.15: Detecção das linhas de cultura através da transformada de Hough (JIANG et al., 2010).

Os experimentos realizados no trabalho demonstraram que o algoritmo proposto é capaz de identificar as linhas de cultura em diferentes condições climáticas. No entanto, se as distâncias entre as linhas de cultura ou as distâncias entre as copas das plantas em linhas de plantio vizinhas forem pequenas, os centros das linhas de plantio não serão detectados satisfatoriamente. O algoritmo levou 70 ms para processar uma imagem colorida de 400 x 300 pixels em um computador com um processador de 1,8 GHz.

Para a navegação autônoma de máquinas agrícolas em linhas de cultura, em Neris (NERIS, 2008) foi utilizado um método para estimar o ponto de fuga da imagem. Devido às deformações da projeção perspectiva, as linhas paralelas verticais em uma imagem convergem a um determinado ponto conhecido como ponto de fuga. Em aplicações de navegação agrícola o ponto de fuga fornece informações da posição da câmera em relação às linhas de cultivo e pode ser utilizado como entrada em um sistema de direcionamento automático responsável por manter a rota da máquina agrícola. Na Figura 3.2.16 é ilustrado o ponto de fuga em uma cena de plantio de soja.

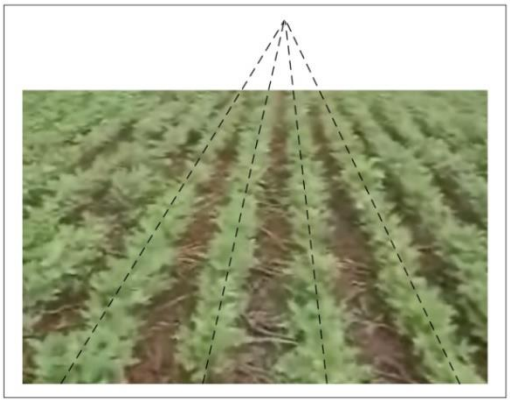


Figura 3.2.16: Ponto de fuga em uma cena de plantio de soja (NERIS, 2008).

O autor citou diferentes técnicas utilizadas para a obtenção dos pontos de fuga. Segundo ele, essas técnicas variam em termos de complexidade e tempo de processamento, inviabilizando, em muitos casos, a sua utilização em sistemas de direcionamento automático. O autor obteve o ponto de fuga por meio da soma dos pixels de cada coluna da imagem. O menor valor entre as somas indica a coluna na qual está localizado o ponto de fuga da imagem. O valor do pixel utilizado no somatório das colunas foi obtido a partir da seguinte equação:

$$P = 2 \times G - R - B \tag{3.2.5}$$

P é o valor do pixel. *R*, *G* e *B* são respectivamente os componentes vermelho, verde e azul do espaço de cor RGB.

Essa equação permite realçar as plantas verdes e é relativamente insensível às variações de luminosidade. Para o autor, o realce do verde proporcionado e a sua simplicidade são as principais razões para o uso dessa equação. Essas características contribuem para o objetivo do método proposto que é a redução do tempo de processamento da imagem e consequentemente o aumento da taxa de dados.

Na Figura 3.2.17 são ilustrados três imagens de uma cultura de soja obtidas por uma câmera digital e os respectivos gráficos da soma dos pixels das colunas. A câmera foi posicionada em três posições: à direita, ao centro e à esquerda da entrelinha de plantio selecionada. Para maximizar a distorção gerada pela projeção em perspectiva é necessário posicionar a câmera de forma cobrir regiões distantes da posição da câmera.

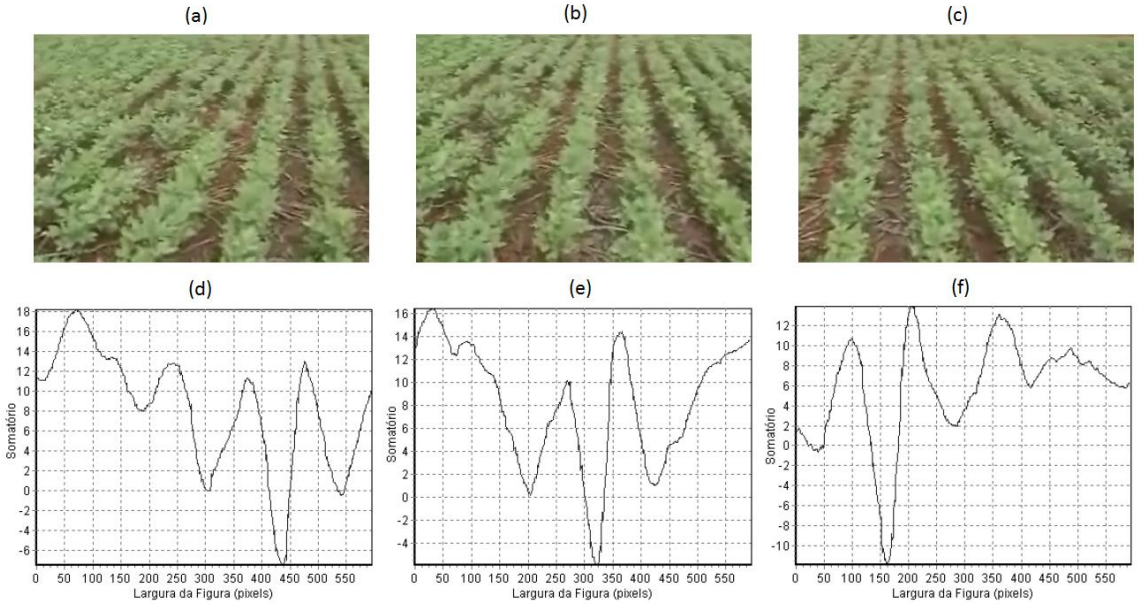
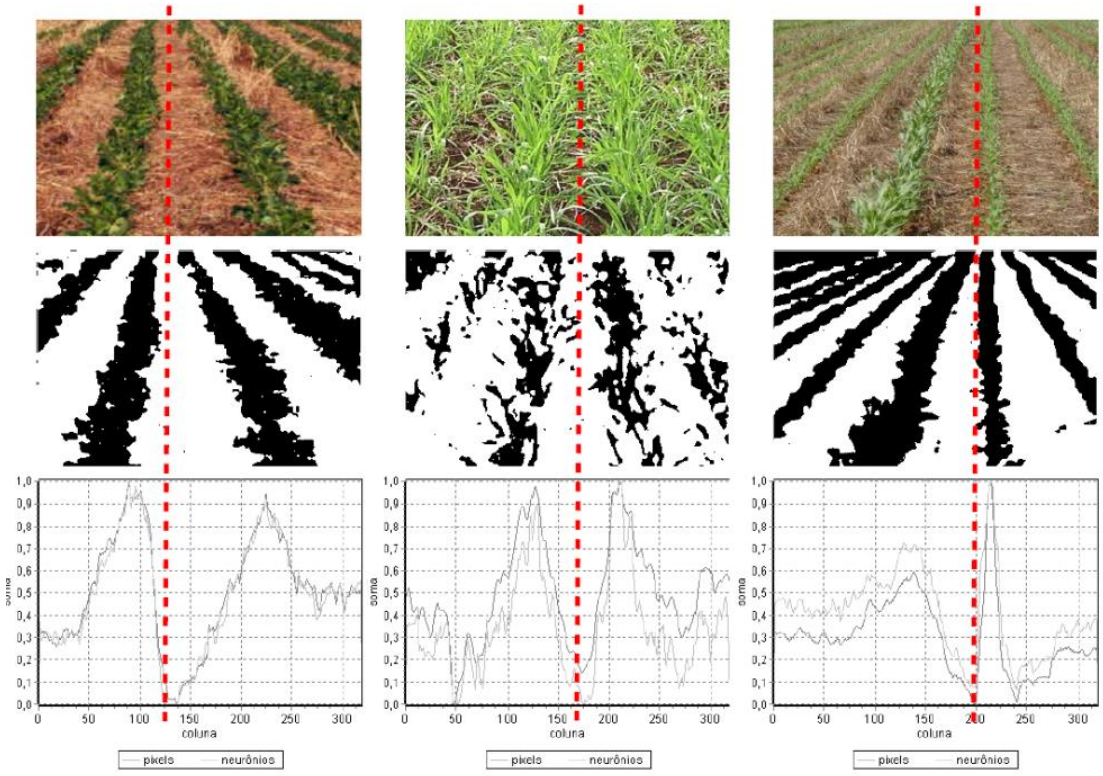


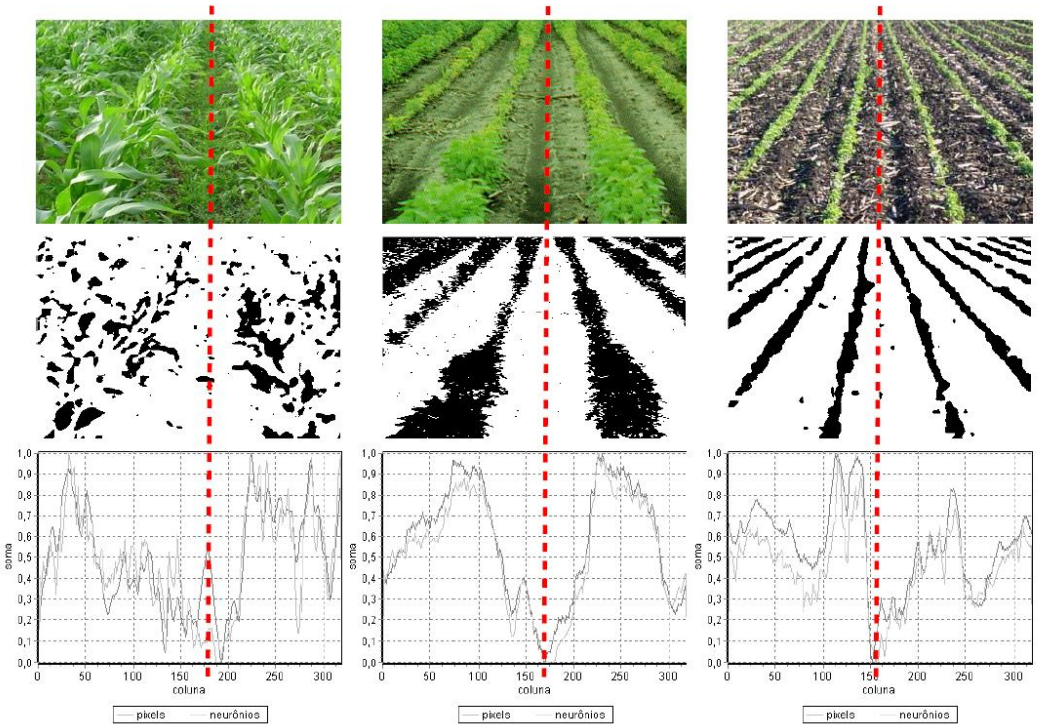
Figura 3.2.17: Diferentes localizações do ponto de fuga: (a) ponto de fuga à direita; (b) ponto de fuga no centro; (c) ponto de fuga à esquerda; (d) somadas colunas em (a); (e) soma das colunas em (b); (f) soma das colunas em (c) (NERIS, 2008).

Segundo o autor, a estimativa do ponto de fuga a partir da soma de pixels das colunas minimiza alguns problemas tipicamente presentes em imagens de áreas de cultivo. Imagens contendo falhas de plantio e ervas daninha entre as linhas de cultivo podem ser utilizadas diretamente sem a necessidade de calibrações ou ajustes prévios.

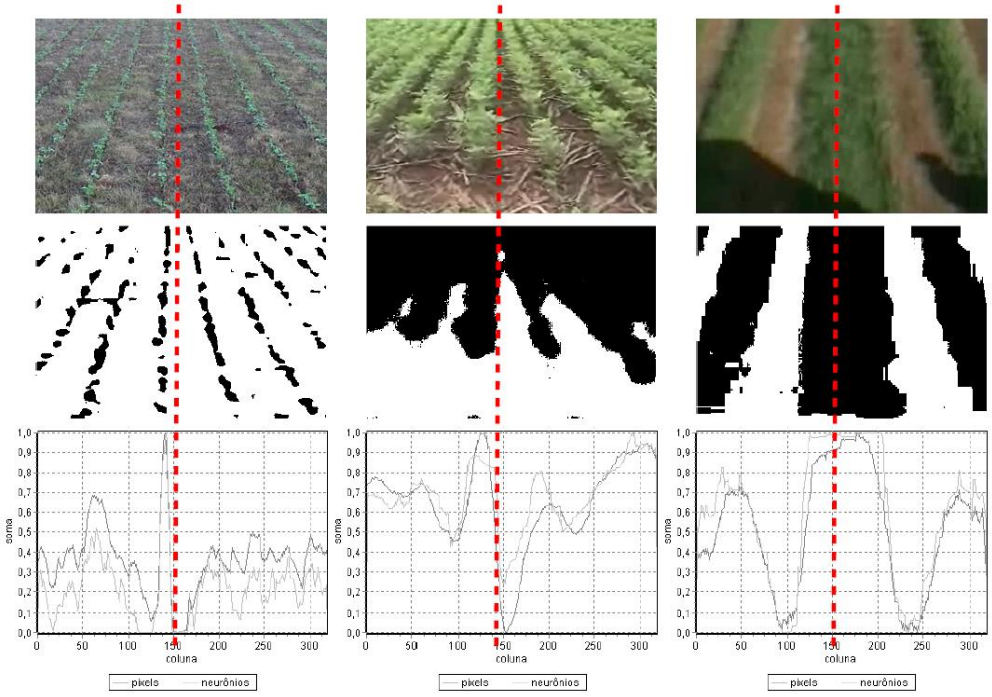
Os experimentos realizados pelo autor também avaliaram o desempenho de uma Rede Neural *Pulse-Coupled* (PCNN), para as tarefas de processamento de imagem envolvidas na metodologia apresentada. Experimentos anteriores demonstraram que as PCNNs podem ser utilizadas em várias aplicações relacionadas ao processamento de imagens digitais. Na Figura 3.2.18 são ilustrados alguns testes preliminares em condições distintas. O pondo de fuga nas imagens corresponde ao pico na contagem dos pixels, ou contagem de neurônios ativos para a implementação em rede neural artificial.



(a) (b) (c)


soma
coluna
pixels
neurônios

(d) (e) (f)


soma
coluna
pixels
neurônios

(g) (h) (i)

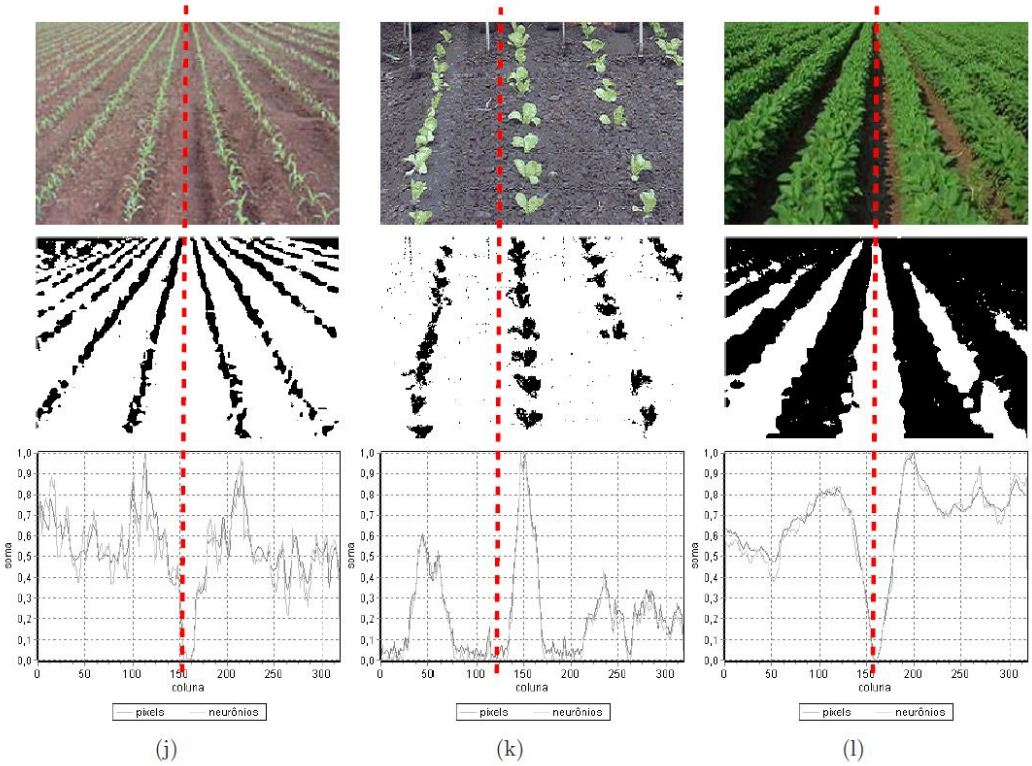


Figura 3.2.18: Resultados do processamento e estimativa do ponto de fuga em condições variadas (NERIS, 2008).

Como plataforma de testes foi utilizado um triciclo de carga modificado para ter tração e controle de direção por meio de motores elétricos. O triciclo suporta uma carga máxima de 50 kg e possui tração nas duas rodas traseiras. A plataforma de testes é ilustrada na Figura 3.2.19. A câmera utilizada foi um *webcam* colorido fabricado pela Creative. A sua resolução máxima é de 640 x 480 pixels a 30 quadros por segundo.

Figura 3.2.19: Plataforma de testes (NERIS, 2008).

O autor concluiu citando que o método de extração da posição de imagens da área de cultivo proposto em seu trabalho apresentou resultados satisfatórios em aplicações de navegação em linhas de cultura. A redução do tempo de processamento devido à simplicidade do algoritmo permite processar um número maior de imagens por unidade de tempo melhorando a resposta dos controladores. Desta forma, a metodologia apresentada em seu trabalho permite o desenvolvimento de sistemas de direcionamento autônomo de veículos que operam em velocidades elevadas, como, por exemplo, pulverizadores.

A maioria das técnicas de visão computacional utilizadas para navegação autônoma de veículos agrícolas através de linhas de plantio requer suficiente diferença visual entre as linhas de plantio e o solo no processo de segmentação de imagem. Dessa forma, o processo de segmentação de imagem muitas vezes se torna uma tarefa desafiadora devido às condições visuais da área de plantio. Em English (ENGLISH *et al.*, 2014) foi apresentado um método de visão computacional baseado em textura capaz de identificar o trajeto através de campos com condições visuais extremamente variadas. O método não utiliza nenhum processo de segmentação de imagem. Dessa forma, não é necessária qualquer suposição quanto à aparência do plantio, como, por exemplo, espaçamento entre as linhas de cultura, periodicidade, coloração e condições de iluminação. Na Figura 3.2.20 são ilustradas duas condições distintas de cena agrícola em que o método proposto foi capaz de navegar sem a necessidade de realizar modificações.

(a) (b)

Figura 3.2.20: Campos agrícolas com aparências extremamente distintas: (a) alto contraste entre as plantas e o solo; (b) período de pousio (ENGLISH et al., 2014).

O método proposto identifica as linhas de cultura por meio da estimativa da textura paralela planar dominante em uma vista aérea simulada. Para o desenvolvimento do método foi assumido que as linhas de plantio são aproximadamente retas e o terreno razoavelmente plano, características que são encontradas em uma ampla variedade de culturas.

No diagrama de blocos da Figura 3.2.21 é apresentada uma visão geral do algoritmo proposto. São as suas etapas:

- Pré-processamento da imagem para corrigir as distorções da lente e redução da resolução para melhorar o tempo de processamento.
- Uso de uma IMU e/ou rastreamento visual do horizonte para estimar o posicionamento da câmera.
- Estabilização da imagem a partir dos dados de posicionamento da câmera.
- Transformação da imagem estabilizada em uma vista aérea simulada por meio da transformada de perspectiva inversa.
- Estimar a direção do veículo com relação às linhas de cultura a partir da direção da textura paralela dominante.
- Corrigir a direção da imagem da vista aérea utilizando a estimativa da direção das linhas de plantio.
- Gerar um *frame template* por meio das somas das colunas da imagem corrigida.
- Estimar o movimento lateral relativo à área de plantio por meio da comparação entre o *frame template* e o *crop template*.

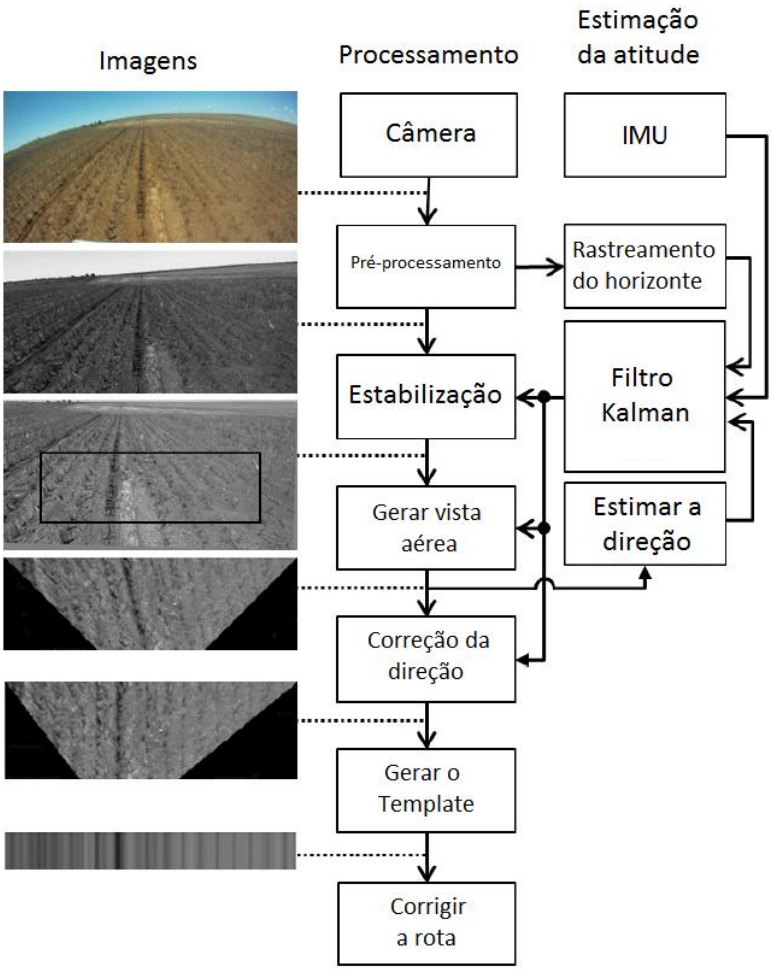


Figura 3.2.21: Diagrama de blocos do algoritmo (ENGLISH et al., 2014).

O método para a estimativa da direção das linhas de cultura consiste em inclinar interativamente a imagem da vista aérea simulada. Para cada ângulo em que a imagem é inclinada são calculadas as somas das colunas e posteriormente determinada a variância entre os resultados. O ângulo que obtiver a menor variância é considerado a melhor estimativa do ângulo das linhas de plantio em relação à posição da câmera.

Na etapa para estimativa do desvio lateral em relação às linhas de plantio as imagens de vista aérea simulada são transformadas em funções denominadas de *templates*. O primeiro *template* gerado é denominado *crop template* e os *templates* sequentes são denominados de *frame template*. Para estimar o desvio lateral, uma comparação é realizada entre um frame *template* e o *crop template* por meio do *Zero Normalised Cross Correlation* (ZNCC).

A plataforma robótica utilizada, ilustrada na Figura 3.2.22, é um veículo elétrico utilitário *John Deere Gator TE* modificado para operações autônomas. O processamento dos dados foi realizado por dois computadores com sistema operacional *Ubunto* 12.04 e *Robot Operating System* (ROS). O algoritmo de navegação foi implementado em C++ utilizando *OpenCV*. Nos testes foram utilizadas as câmeras *IDS uEye CP* e o modelo de baixo custo *Microsoft LifeCam Cinema*.

Figura 3.2.22: Plataforma robótica para testes (ENGLISH et al., 2014).

O primeiro experimento realizado mostra a capacidade de navegação do algoritmo em uma ampla variedade de condições. Nesse experimento o robô foi guiado manualmente através das linhas de plantio e as imagens capturadas foram processadas pelo algoritmo desenvolvido. Os resultados obtidos no processamento são comparados com os resultados obtidos por meio do GPS. Na Figura 3.2.23 são ilustrados os resultados obtidos no experimento.

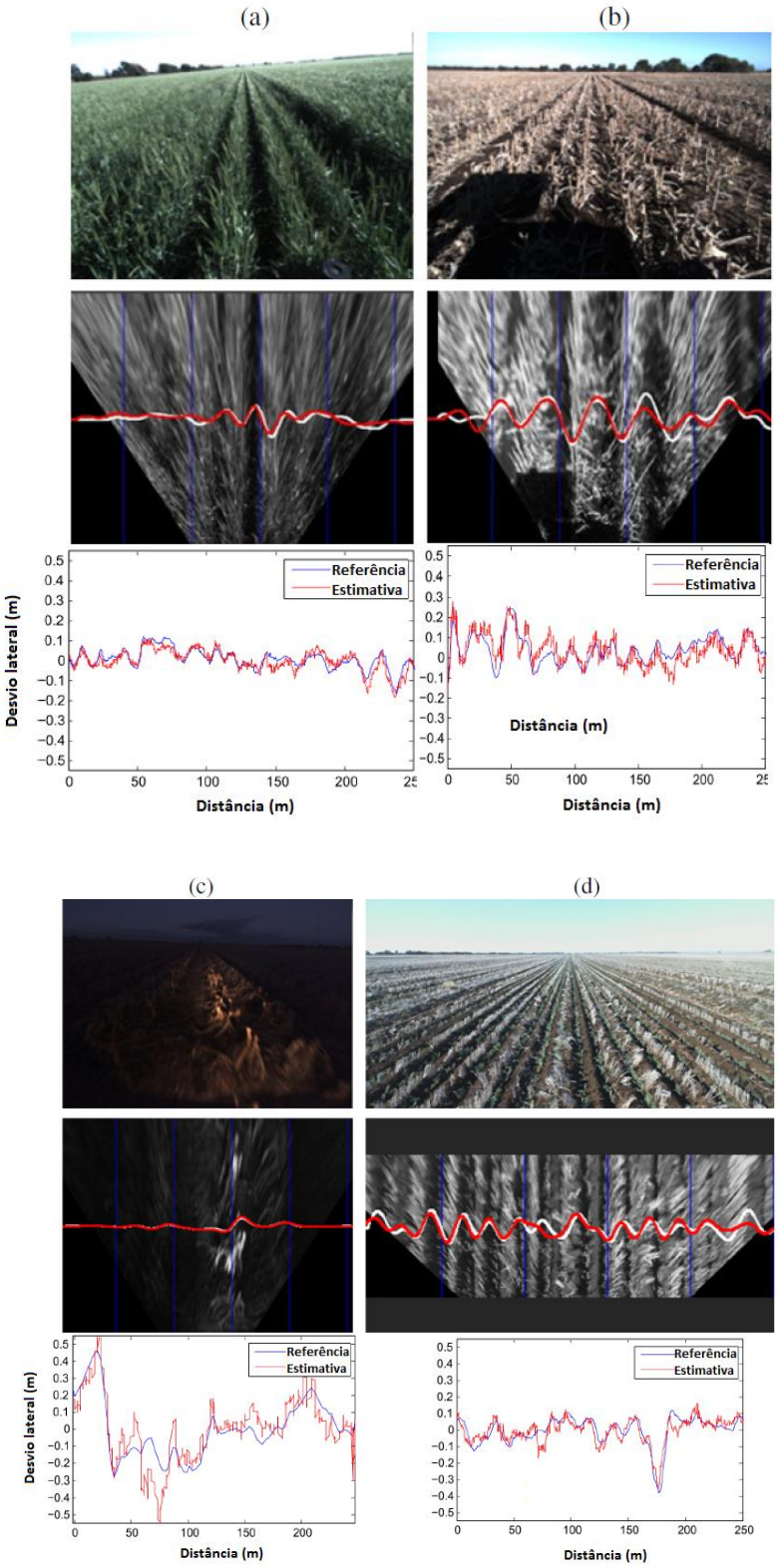


Figura 3.2.23: Resultados do experimento em quatro ambientes distintos: (a) trigo; (b) restolho de sorgo; (c) restolho de sorgo no período da noite sob iluminação dos faróis; (d) grão de bico (ENGLISH et al., 2014).

O segundo experimento foi realizado para avaliar a capacidade de navegação do sistema através do restolho de sorgo em uma área de seis hectares. O percurso realizado pelo veículo em um intervalo de duas horas é ilustrado na Figura 3.2.24. Na figura pode-se identificar as regiões onde foi possível extrair os dados para a navegação autônoma do veículo.

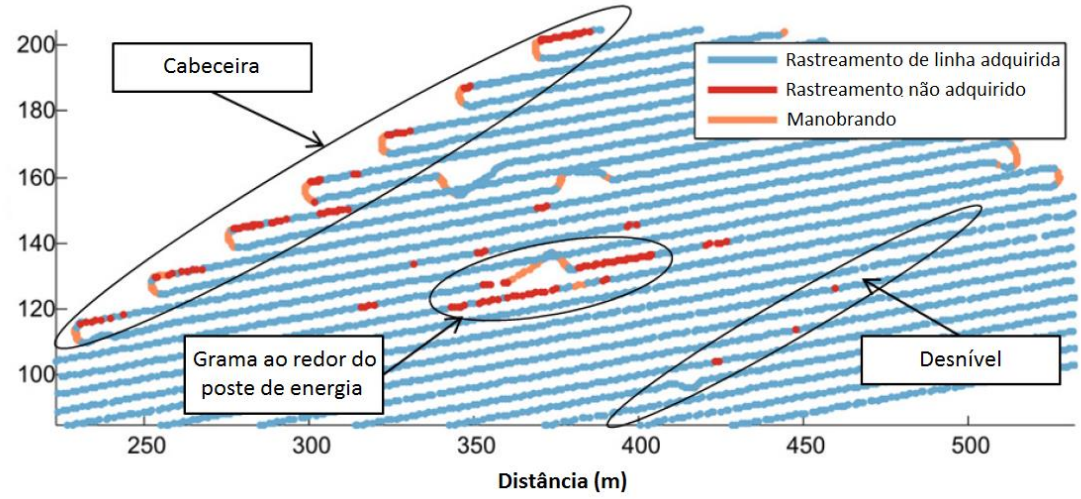


Figura 3.2.24: Percurso de avalição do sistema de navegação em uma região de seis hectares (ENGLISH et al., 2014).

No último experimento realizado o veículo foi guiado de forma totalmente autônoma. Os dados extraídos pelo algoritmo foram utilizados para guiar o veículo por meio de um controlador proporcional-integral (PI). Os resultados obtidos nesse experimento foram comparados com os resultados fornecidos pelo GPS. Nas Figuras 3.2.25 e 3.2.26 são apresentados os resultados dos erros de desvio lateral na navegação do veículo em um campo de trigo e de restolho de sorgo respectivamente. O erro quadrático médio foi de 18 mm para o trigo e 120 mm para o restolho de sorgo.

O autor concluiu afirmando que o método para a navegação autônoma de máquinas agrícolas proposto em seu trabalho pode ser facilmente aplicado em uma ampla variedade de culturas sem a necessidade de ajustes ou modificações. A não utilização de processos de segmentação de imagem também permite que o algoritmo funcione em ambientes onde as características visuais das linhas de plantio são fracas ou ruidosas.

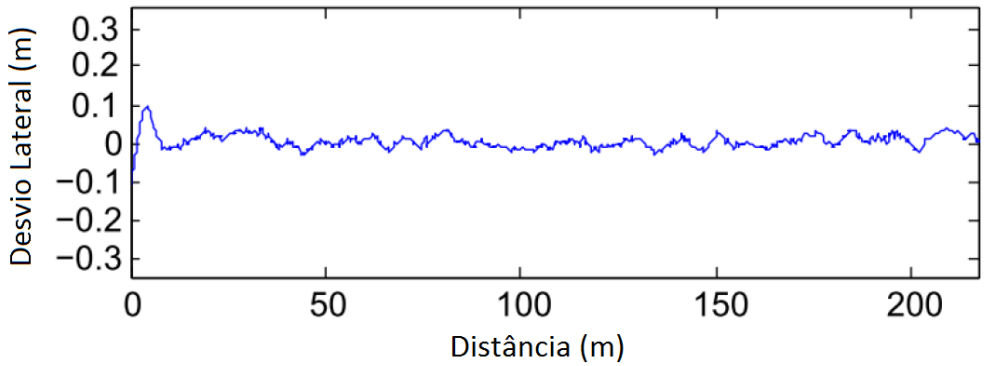


Figura 3.2.25: Erro de desvio lateral obtido no teste de navegação autônoma em um campo de plantação de milho (ENGLISH et al., 2014).

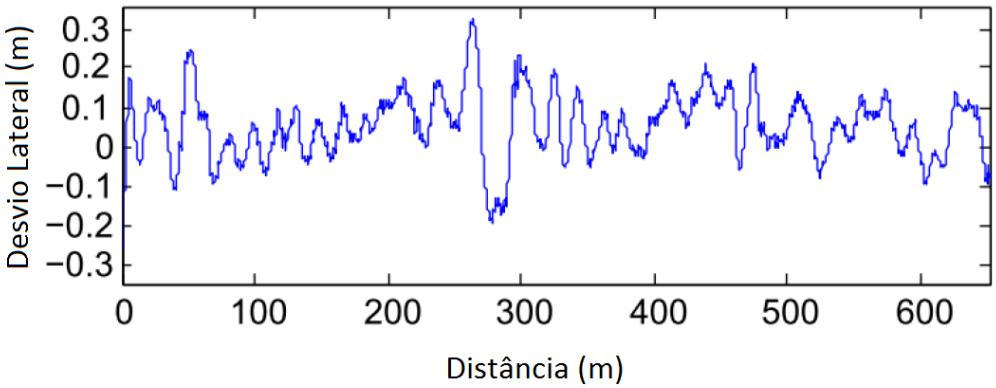


Figura 3.2.26: Erro de desvio lateral obtido no teste de navegação autônoma em um campo de restolho de sorgo (ENGLISH et al., 2014).

Em Meng (MENG *et al*., 2015) foi desenvolvida uma estrutura para a navegação autônoma de máquinas agrícolas em linhas de cultura empregando um sistema de visão computacional e um sistema de posicionamento global GPS. A estrutura do sistema de navegação é ilustrada na Figura 3.2.27. O *hardware* utilizado no sistema de navegação consiste em uma câmera de vídeo colorida, um computador industrial, um sistema de controle de deslocamento lateral, um módulo GPS e um sistema hidráulico.

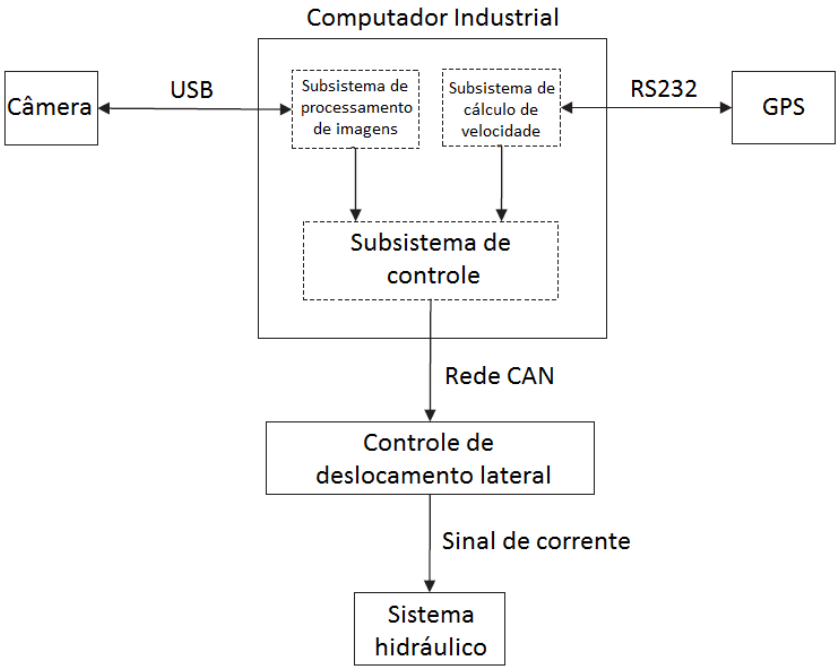


Figura 3.2.27: Estrutura do sistema de navegação (MENG et al., 2015).

O software do sistema de navegação está dividido em três subsistemas. São eles: subsistema de processamento de imagens, subsistema de cálculo de velocidade e subsistema de controle. O subsistema de processamento de imagens extrai as linhas de navegação das imagens da área de cultivo. O subsistema de cálculo de velocidade estima a velocidade do veículo a partir das informações fornecidas pelo GPS. De acordo com as linhas de navegação e a velocidade do veículo o sistema de controle calcula o movimento lateral necessário e envia instruções para o módulo de deslocamento lateral.

A câmera utilizada no projeto foi a *OKAC1310"block"* com lentes padrão. O modelo é uma câmera analógica colorida com CCD simples e resolução de 640 x 480 pixels. O computador utilizado possui *chipset Intel 945GSE*, 2 GB de memória RAM, sistema operacional *Windows XP* e grau de proteção IP67. O receptor GPS utilizado foi o AC12 fabricado pela *Thales*. Na Figura 3.2.28 é ilustrada a plataforma experimental utilizada nos experimentos.

Figura 3.2.28: Plataforma experimental (MENG et al., 2015).

A câmera foi focada na superfície do solo em um ângulo determinado de forma a cobrir aproximadamente quatro linhas de cultura. As imagens capturadas são continuamente transferidas para o computador onde são realizadas as etapas de processamento de imagem e controle do veículo. O GPS fornece informações dinâmicas de navegação ao computador através de uma rede de comunicação serial RS232 em uma taxa de 115.000 bits/s. A comunicação entre o computador e o sistema de controle de deslocamento lateral é realizado através de uma rede CAN. O módulo de controle recebe informações do computador e as convertem em sinais de corrente elétrica para o sistema hidráulico.

O sistema de visão desenvolvido no projeto utiliza o espaço de cores HSI. O Algoritmo proposto segmenta as imagens a partir do componente H. Para estabelecer os parâmetros ideias para o processo de segmentação das imagens em diversas condições climáticas o autor amostrou imagens de uma área de plantio de milho a cada dois dias durante um período de dois meses. As imagens foram amostradas em intervalos de meia hora desde o período da manhã até o período da tarde. A partir dos dados foram selecionados de forma aleatória 2000 pixels correspondentes ao plantio e 2000 pixels correspondentes ao solo. Com base nos pixels selecionados o autor define o *range* da componente H de 120° a 160° para o plantio e 40° a 70° para a região do solo.

A partir dos parâmetros estabelecidos o algoritmo de segmentação produz uma imagem binária atribuindo valor 1 para os pixels que compreendem o *range*

estabelecido para o plantio e 0 para os demais pixels. Na figura 3.2.29 é ilustrado o resultado da etapa de segmentação de imagens.

Figura 3.2.29: Processo de segmentação de imagem (MENG et al., 2015).

O algoritmo utilizado para a identificação das linhas de cultura foi o *Linear Scanning* (MENG *et al*., 2013). O algoritmo procura todas as possibilidades de linhas que iniciam em um pixel da região inferior da imagem e terminam em um pixel da região superior da imagem, como ilustrado na Figura 3.2.30. Os pixels de valor 1 pertencentes a cada linha são contados e as linhas que tiverem a maior contagem são consideradas as linhas de plantio. Nas Figuras 3.2.31 e 3.2.32 pode ser realizada uma comparação entre diferentes métodos utilizados para a identificação das linhas de cultura.

Figura 3.2.30: Principio do algoritmo Linear Scanning (MENG et al., 2014).

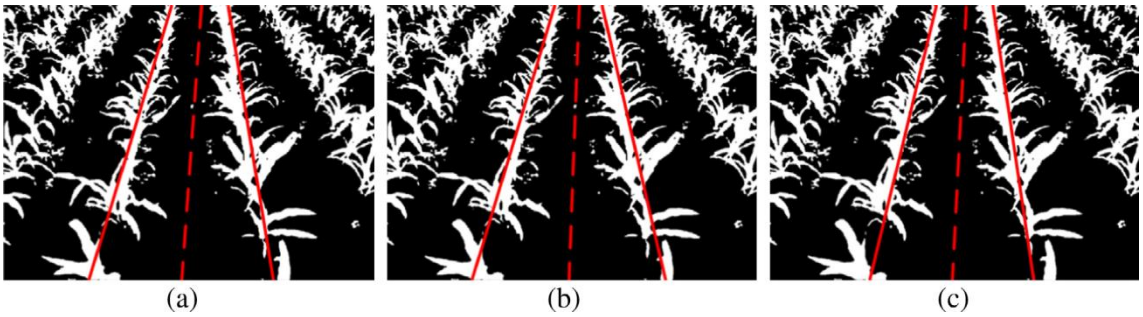


Figura 3.2.31: Detecção das linhas de plantio em uma cena agrícola com baixa densidade de ervas daninha através de diferentes algoritmos: (a) transformada de Hough; (b) método dos mínimos quadrados; (c) Linear Scanning (MENG et al., 2014).

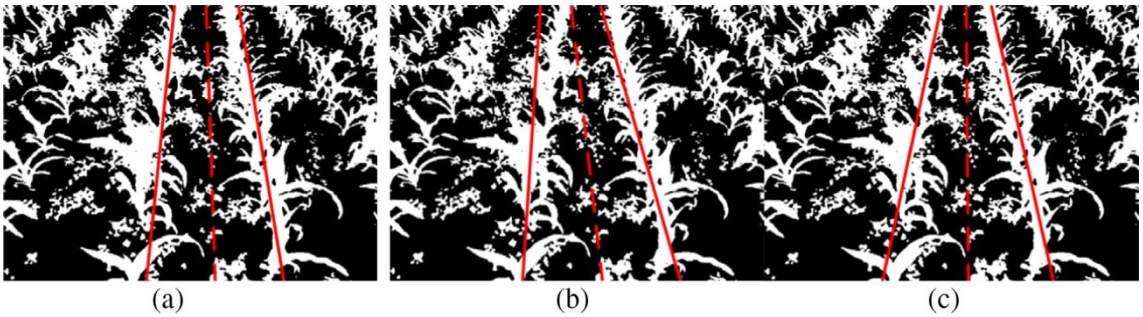


Figura 3.2.32: Detecção das linhas de plantio em uma cena agrícola com alta densidade de ervas daninha através de diferentes algoritmos: (a) transformada de Hought; (b) método dos mínimos quadrados; (c) Linear Scanning (MENG et al., 2014).

Os testes de navegação foram realizados em um clima ensolarado. O comprimento do campo experimental foi de aproximadamente 20 m com 0,6 m de espaço entre cada linha de plantio. A altura das plantas era de aproximadamente 20 cm e a velocidade de deslocamento do veículo foi de 0,6 m/s, 1,0 m/s e 1,4 m/s. Nas Tabelas 3.2.1, 3.2.2 e 3.2.3 são ilustrados os resultados do sistema de navegação utilizando o algoritmo *Linear Scanning*, transformada de *Hough* e método dos mínimos quadrados respectivamente. O autor mencionou que os resultados demostram um desempenho aceitável para o sistema de navegação em diferentes condições de iluminação para todas as velocidades do experimento. O erro médio menor foi de 2,7 cm e o erro máximo foi de 6,8 cm.

Velocidade (m/s)	Desvio lateral (cm)		
	Máximo	Média	Desvio Padrão
0,60	4,50	1,88	1,31
1,00	5,50	2,15	1,38
1,40	6,80	2,62	1,90

Tabela 3.2.1: Dados experimentais utilizando Linear Scanning (MENG et al., 2014).

Velocidade (m/s)	Desvio lateral (cm)		
	Máximo	Média	Desvio Padrão
0,60	5,40	1,95	1,26
1,00	8,20	3,42	2,35
1,40	9,60	3,80	2,77

Tabela 3.2.2: Dados experimentais utilizando a transformada de Hough (MENG et al., 2014).

Velocidade (m/s)	Desvio lateral (cm)		
	Máximo	Média	Desvio Padrão
0,60	5,70	2,19	1,46
1,00	7,80	3,57	2,47
1,40	9,90	4,03	2,86

Tabela 3.2.3: Dados experimentais utilizando o método dos mínimos quadrados (MENG et al., 2014).

Uma nova técnica de visão computacional para navegação de robôs móveis em pomares foi apresentada em Sharifi (SHARIFI & CHEN, 2015). A técnica consiste primeiramente na segmentação da imagem colorida por meio do algoritmo *Mean-Shift* (COMANICIU & MEER, 2002). Posteriormente, um novo método de classificação baseado na teoria de particionamento gráfico classifica as regiões da imagem em classes definidas como, por exemplo, terra, árvores e céu. Em seguida, a transformada de *Hough* é utilizada para extrair as características necessárias que permitem traçar um trajeto no corredor do pomar. O fluxograma do algoritmo proposto é ilustrado na Figura 3.2.33.

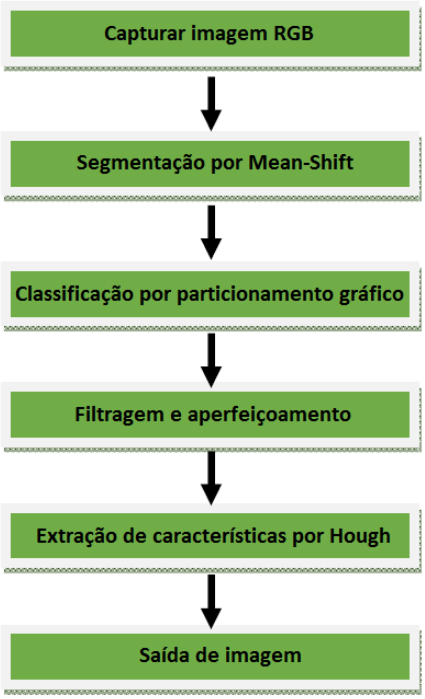


Figura 3.2.33: Diagrama de fluxo do algoritmo para navegação autônoma de robôs móveis (SHARIFI & CHEN, 2015).

Para os experimentos realizados no trabalho foram selecionadas imagens aleatórias para diferentes condições climáticas. A resolução das imagens foi de 640 x 480 pixels. O computador utilizado para o processamento possui um processador *Intel Core i7-4770* de 3,4 GHz e memória RAM de 16 GB. A plataforma de simulação utilizada foi o Matlab R2014a.

O algoritmo *Mean-Shift* é um método de análise não paramétrica que permite agrupar regiões da imagem com características semelhantes. Nesse método as imagens são convertidas em um espaço de características por meio de um *Kernel* não paramétrico. O algoritmo procura por funções de densidade de modos e atribui para cada ponto um modo específico.

Dado um conjunto de dados $\{xi\}_{i=1}^{n}$ com *n* pontos, a estimativa da densidade do *kernel* para um *kernel K* e um raio *h* é determinada pela equação:

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} k\left(\left\|\frac{x - x_i}{h}\right\|\right)^2 \tag{3.2.6}$$

O vetor *Mean-Shift* foi calculado por meio da equação:

$$m(x) = \frac{\sum_{i=1}^{n} x_i k\left(\left\|\frac{x - x_i}{h}\right\|\right)^2}{k\left(\left\|\frac{x - x_i}{h}\right\|\right)^2} - x \tag{3.2.7}$$

O autor afirmou que o uso do algoritmo *Main-Shift* permitiu reduzir o custo computacional. A sua utilização permite o processamento de grupos de pixels ao invés de pixel por pixel. O resultado da aplicação do algoritmo em uma imagem de cena agrícola é ilustrado na Figura 3.2.34.

(a) (b)

Figura 3.2.34: Aplicação do algoritmo Main-Shift na imagem de um pomar: (a) imagem original; (b) imagem processada (SHARIFI & CHEN, 2015).

A etapa seguinte classifica as regiões utilizando um método de particionamento gráfico. Em uma imagem particionada, cada pixel é considerado um nó e os nós adjacentes são conectados por um link. Cada link recebe um valor que define a similaridade entre os dois nós conectados. Dessa forma, o método de particionamento gráfico tenta formar classes com nós de maior similaridade.

O particionamento pode ser realizado por meio de parâmetros de corte. A técnica para o particionamento utilizado no trabalho foi desenvolvido em um trabalho anterior. Essa técnica é obtida por meio de uma modificação no algoritmo de corte normalizado permitindo realizar um processo de classificação ao invés de uma segmentação.

A Figura 3.2.35 ilustra o resultado do processo de classificação da imagem em três classes. São elas: terra, céu e arvores.

Figura 3.2.35: Imagem classificada em três classes: terra, céu e arvores (SHARIFI & CHEN, *2015).*

Na etapa de otimização da extração por operação de filtragem algumas operações foram realizadas na imagem classificada para facilitar a extração de características por meio da transformada de *Hough*. Nessa etapa, primeiramente foi realizado a transformação da imagem classificada em uma imagem em escala de cinza conforme ilustrado na Figura 3.2.36a. Posteriormente, foi realizada uma busca por pixels semelhantes ao pixel da região central inferior (localização próxima do robô). O resultado obtido nessa etapa é ilustrado na Figura 3.2.36b. Por fim, a imagem em escala de cinza é transformada em uma imagem binária como ilustrado na Figura 3.2.37.

A última etapa do sistema de visão computacional utiliza a transformada de *Hough* para encontrar as linhas entre as regiões da imagem compreendidas pelas árvores e terra. A linha central que inicia no centro inferior da imagem e termina na intersecção das linhas extraídas é compreendida como o trajeto para a navegação do robô. O resultado do método de extração das linhas de navegação é ilustrado na Figura 3.2.38.

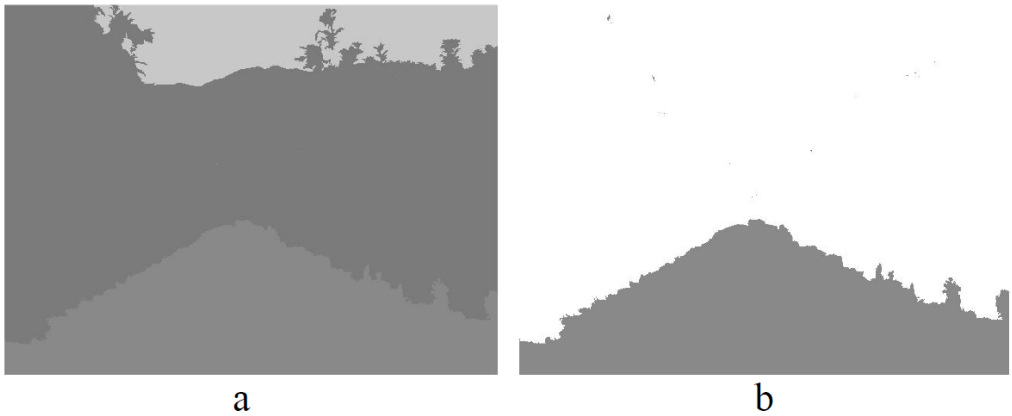


Figura 3.2.36: Processo de otimização: (a) transformação em escala de cinza; (b) classificação da classe localizada na região central inferior da imagem (SHARIFI & CHEN, *2015).*

Figura 3.2.37: Representação binária da imagem classificada (SHARIFI & CHEN, *2015).*

Figura 3.2.38: Extração das linhas de navegação através da transformada de Hough (SHARIFI & CHEN, 2015).

3.3 CONSIDERAÇÕES FINAIS

Diversos trabalhos anteriores envolvendo sistemas de visão computacional foram apresentados. Em sua maioria, esses trabalhos utilizam visão monocromática e exigem um elevado custo computacional para extrair as informações necessárias para a navegação autônoma de veículos. No contexto agrícola, um dos grandes desafios na automação é a redução de custos, uma vez que esses sistemas só serão aceitos no mercado se houver benefícios financeiros a custos razoáveis (NERIS, 2008). Por essa razão, neste capítulo foram apresentados alguns trabalhos relevantes que demonstram algumas das principais metodologias adotadas para a simplificação de sistemas de visão computacional em aplicações de navegação em linhas de plantio.

4 IMPLEMENTAÇÕES

4.1 CONSIDERAÇÕES INICIAIS

O *hardware* proposto no presente trabalho consiste em um circuito dedicado para a navegação autônoma de robôs móveis agrícolas. O *hardware* desenvolvido pode ser reprogramado de forma a adequar o sistema para as condições específicas de uma determinada área de plantio. O método implementado consiste na classificação de cores e extração das linhas de navegação a partir de características locais da cena.

O fluxograma do sistema proposto é ilustrado na Figura 4.1.1. A primeira etapa consiste em um circuito para a classificação dos *pixels*. A segunda etapa consiste em um grupo de processadores morfológicos que podem ser configurados para diversas tarefas, tais como a filtragem e a extração de bordas. Por último, um circuito dedicado permite a extração das linhas de navegação por meio do método de regressão linear. A linha de navegação é representada na saída do sistema por meio dos coeficientes angular e linear. A arquitetura proposta também inclui um canal auxiliar, destacado em azul no fluxograma da Figura 4.1.1, que permite a supervisão do processamento nas etapas de classificação, processamento morfológico e regressão linear, enquanto em fase de calibração.

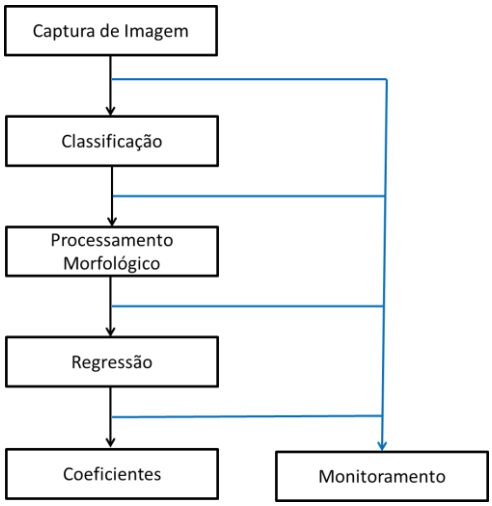


Figura 4.1.1: Fluxograma do sistema de navegação.

O circuito foi implementado por meio da plataforma de desenvolvimento DE2-70 fabricada pela Terasic. O DE2-70 é baseado no FPGA Cyclone II 2C70 da Altera. O FPGA possui 4 PLLs, 1152000 células de memórias, 150 multiplicadores e 68416 LEs. Além disso, a plataforma possui um conjunto de módulos periféricos integrados ao FPGA para a expansão dos recursos, conforme ilustrado no diagrama da Figura 4.1.2. Dentre esses periféricos, inclui-se o decodificador de vídeo NTSC/PAL/SECAM TV ADV7180 e o decodificador analógico-digital de vídeo ADC7123 que são utilizados para a captura de imagens e monitoramento das etapas de processamento, respectivamente.

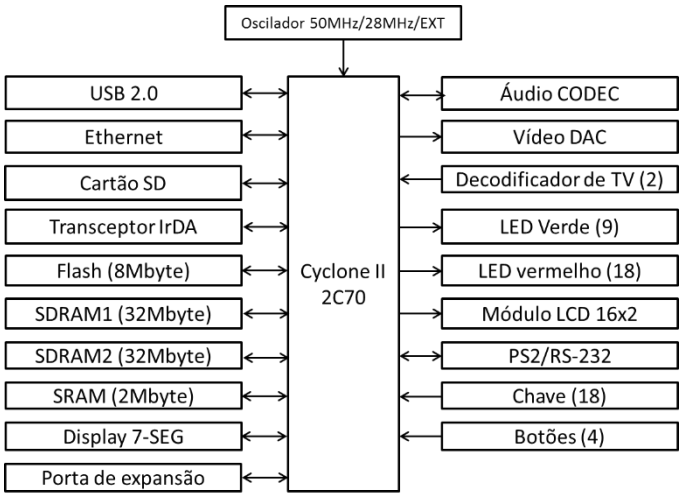


Figura 4.1.2: Módulos periféricos da plataforma de desenvolvimento DE2-70.

4.2 CLASSIFICADOR

Na etapa de classificação é utilizado o método de limiarização global para classificar os *pixels* da imagem dentre várias classes de interesse. As imagens de cena agrícola tipicamente possuem regiões com características de cor distintas, tais características podem ser utilizadas para a identificação da área navegável. No exemplo ilustrado na Figura 4.2.1 é possível identificar três regiões específicas, sendo essas correspondentes ao solo, céu e vegetação. A região do solo na imagem, por exemplo, pode ser identificada a partir da classificação dos *pixels* vermelhos.

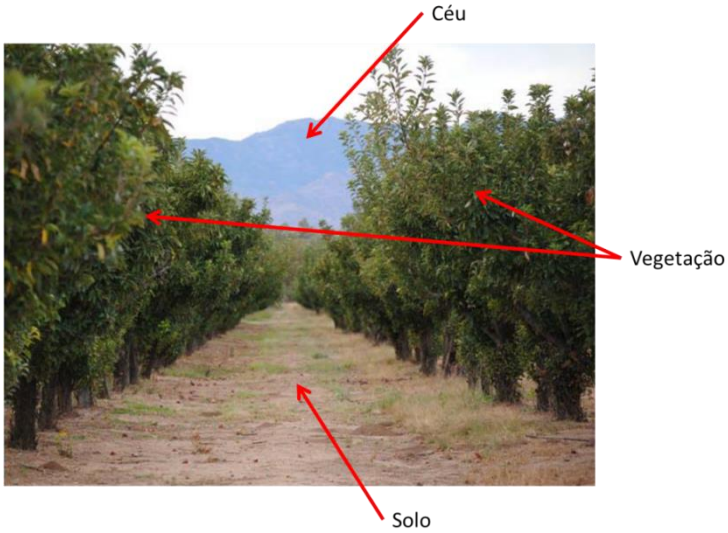


Figura 4.2.1: Imagem típica de corredor para a navegação em um pomar de laranjas (Adaptado de SHARIFI & CHEN, 2015).

O espaço de cor utilizado para o processo de classificação dos *pixels* é o HSL. Esse modelo de cores apresenta vantagens práticas na descrição de cores, permitindo medir a similaridade entre cores de forma simples e intuitiva. O circuito desenvolvido para o processo de classificação é ilustrado no diagrama da Figura 4.2.2. A primeira etapa do circuito consiste em um conversor do espaço de cores RGB para o HSL, bloco "rgb_2_hsl". Os blocos "classificador" configuram o arranjo desenvolvido para classificar os *pixels* da imagem dentre três classes definidas pelo usuário. Essas classes são representadas no circuito por meio das imagens binárias presentes nas saídas C1, C2 e C3 do diagrama da Figura 4.2.2.

A representação das cores no componente H por um plano circular é dada no intervalo de 0° a 360°. Em termos práticos, os tons cujo componente H encontra-se em uma faixa de valores em torno de 0° apresentam limitações quanto a sua classificação devido à descontinuidade numérica presente em 0°. Uma solução simples consiste no arranjo formado por dois blocos "classificador" e uma porta lógica OU, conforme ilustrado na Figura 4.2.3. Para classificar os *pixels* cujo componente H encontra-se na faixa que inicia em 350° e termina em 10°, por exemplo, os blocos classificadores são configurados para estabelecerem dois conjuntos que compreendem os intervalos de 350° a 360° e 0° a 10°, e a porta lógica OU realiza a concatenação entre esses conjuntos.

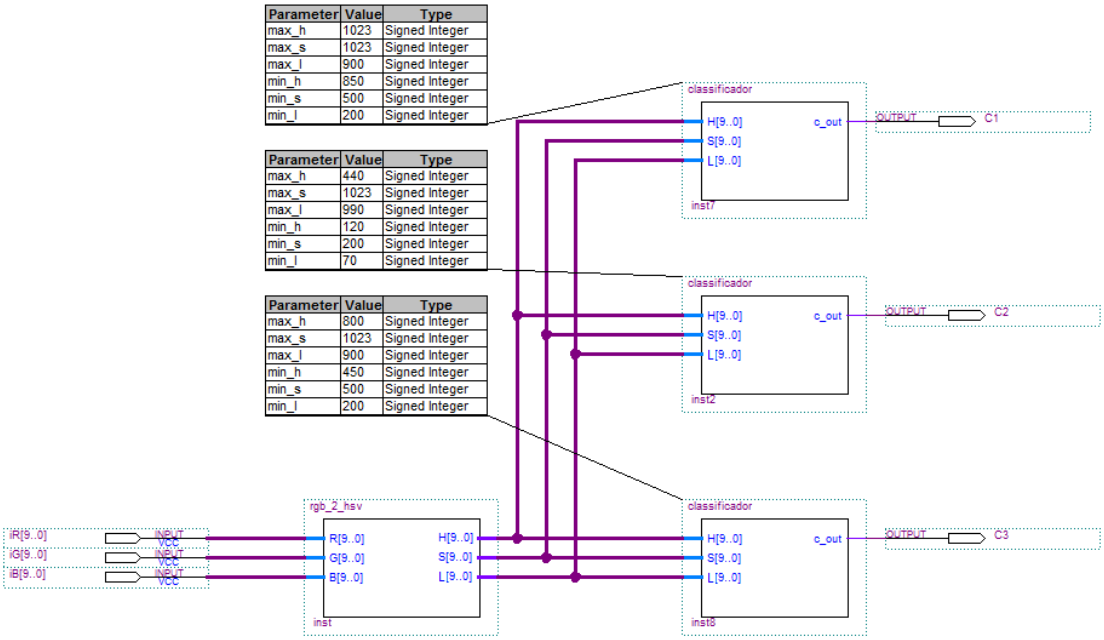


Figura 4.2.2: Diagrama do circuito de classificação dos pixels.

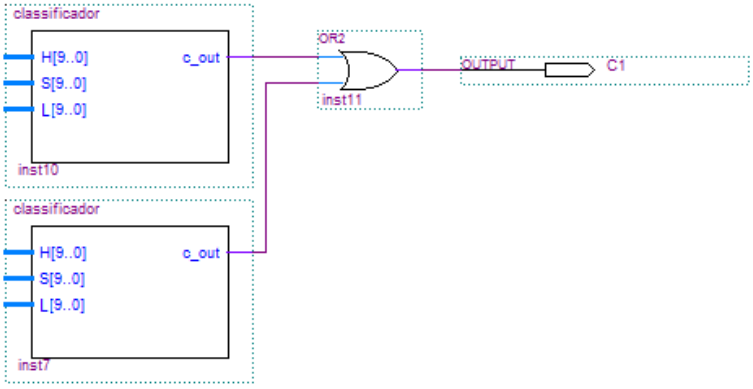


Figura 4.2.3: Arranjo proposto para a classificação de cores na faixa descontínua em torno de $H = 0°$.

No Capítulo 2 foram apresentadas as equações para a conversão de espaço de cores RGB para o HSL. No entanto, em termos práticos, essas equações apresentam limitações devido aos operadores trigonométricos e exponenciais envolvidos. Uma alternativa comumente adotada para a conversão dos espaços de cores com um custo computacional reduzido é a utilização da aproximação dada pelas equações:

$$H = \begin{cases} 60\dfrac{G-B}{máx(R,G,B)-mín(R,G,B)} & se\ máx(R,G,B) = R\ e\ G \geq B \\ 60\dfrac{G-B}{máx(R,G,B)-mín(R,G,B)} + 360 & se\ máx(R,G,B) = R\ e\ G < B \\ 60\dfrac{B-R}{máx(R,G,B)-mín(R,G,B)} + 120 & se\ máx(R,G,B) = G \\ 60\dfrac{R-G}{máx(R,G,B)-mín(R,G,B)} + 240 & se\ máx(R,G,B) = B \end{cases} \tag{4.2.1}$$

$$S = \frac{\{máx(R,G,B) - mín(R,G,B)\}}{máx(R,G,B)} \tag{4.2.2}$$

$$L = \frac{1}{2}[máx(R,G,B) + mín(R,G,B)] \tag{4.2.3}$$

A classificação dos *pixels* da imagem no espaço de cores HSL ocorre por meio de um processo de limiarização global. O fundamento desse processo consiste em separar uma imagem $f(x,y)$ por meio de um limiar T em dois modos, conforme a equação:

$$g(x,y) = \begin{cases} 1 & se\ f(x,y) > T \\ 0 & se\ f(x,y) \leq T \end{cases} \tag{4.2.4}$$

A função $g(x,y)$ é a imagem binária limiarizada por T. Os pontos (x,y) da imagem em que $f(x,y) > T$ são chamados de ponto do objeto e os demais de ponto de fundo.

No processo de classificação de imagens coloridas no modelo HSL foi utilizado um limiar mínimo e um limiar máximo para cada componente de cor, conforme as equações:

$$g_H(x,y) = \begin{cases} 1 & se\ T_{mím,H} \leq H(x,y) \leq T_{máx,H} \\ 0 & caso\ contrário \end{cases} \tag{4.2.5}$$

$$g_S(x,y) = \begin{cases} 1 & se\ T_{mím,S} \leq S(x,y) \leq T_{máx,S} \\ 0 & caso\ contrário \end{cases} \tag{4.2.6}$$

$$g_L(x,y) = \begin{cases} 1 & se\ T_{mím,L} \leq L(x,y) \leq T_{máx,L} \\ 0 & caso\ contrário \end{cases} \tag{4.2.7}$$

O conjunto dos *pixels* que correspondem à classe estabelecida pelo usuário consiste nos *pixels* cujas funções limiarizadas g_h, g_s e g_l são iguais a 1.

Os circuitos propostos para a conversão dos espaços de cores e limiarização dos *pixels* foram desenvolvidos por meio da linguagem de descrição de *hardware* VHDL. Os respectivos códigos são apresentados nos Apêndices A e B.

4.3 PROCESSAMENTO MORFOLÓGICO

O circuito para o processamento morfológico apresentado no presente trabalho foi baseado na arquitetura proposta por Pedrino (PEDRINO, 2003). A arquitetura consiste em um arranjo de múltiplos estágios em uma arquitetura pipeline serial para o processamento de imagens em tempo real por operações básicas de morfologia matemática, tais como erosão e dilatação.

As operações são realizadas por meio dos blocos “processador_morfologico”. Cada bloco possui um conjunto de entradas para a imagem binária, obtida a partir do circuito de classificação, e sinais de sincronismo de vídeo. As saídas correspondem à imagem binária processada e seus respectivos sinais de sincronismo. Os sinais de sincronismo, nas saídas dos blocos, são atrasados em relação aos sinais de entrada em um período que corresponde à latência necessária para o processamento dos *pixels*. Dessa forma, essas unidades de processamento podem ser conectadas em arranjos do tipo cascata, formando uma sequência de operações morfológicas básicas que correspondem ao algoritmo morfológico proposto. A simbologia adotada para a unidade de processamento “processador_morfologico” é ilustrada na Figura 4.3.1. A operação

realizada pela unidade de processamento e o elemento estruturante utilizado são estabelecidos por meio dos parâmetros *Op* e *St*, respectivamente.

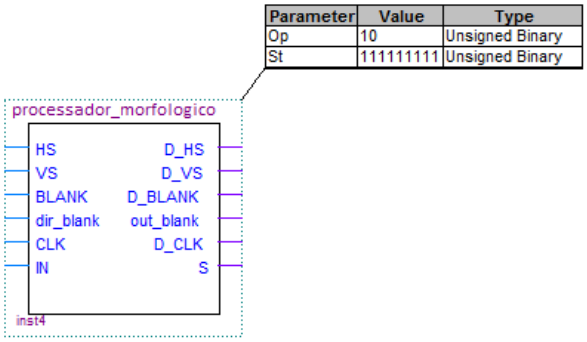


Figura 4.3.1: Simbologia da unidade de processamento morfológico.

O circuito da unidade de processamento é constituído por três blocos, conforme ilustrado na Figura 4.3.2. São esses: “memoria_de_deslocamento”, “operador_logico” e “delay_sincronismo”.

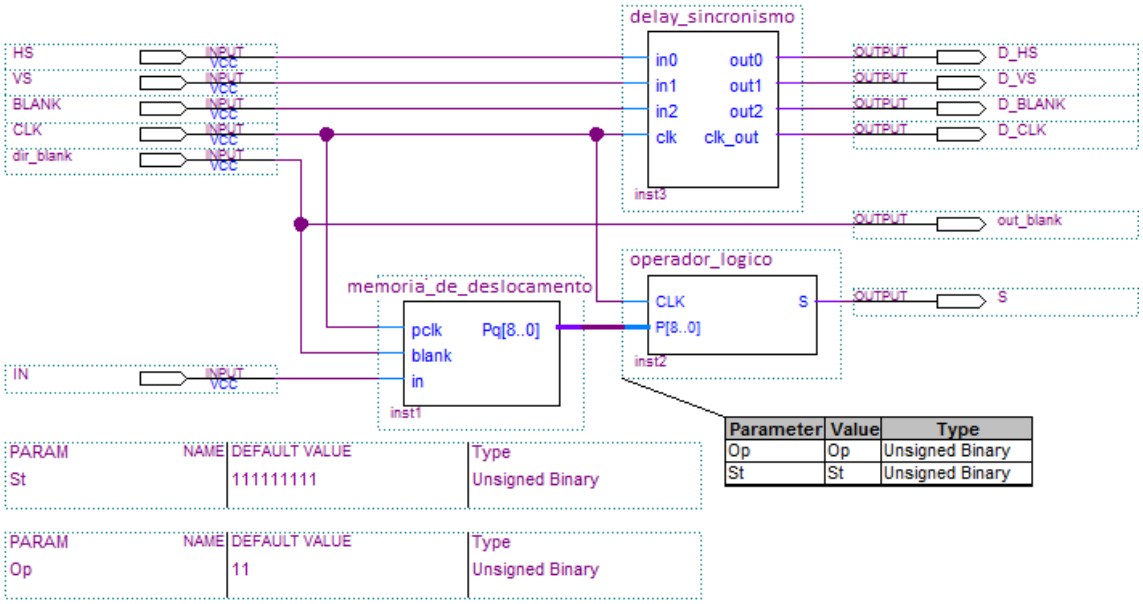


Figura 4.3.2: Circuito da unidade de processamento morfológico.

O bloco “memoria_de_deslocamento” é um arranjo de memórias sequenciais que permite o “operador_logico” acessar simultaneamente os *pixels* de uma janela de dimensão 3 × 3. Essa janela é deslocada sobre a imagem à medida que os *pixels* são

encaminhados ao processador. Essa estratégia permite o acesso simultâneo dos *pixels* que compõem a janela, permitindo o processamento da imagem em um fluxo contínuo. Na Figura 4.3.3 é ilustrado o seu princípio de funcionamento.

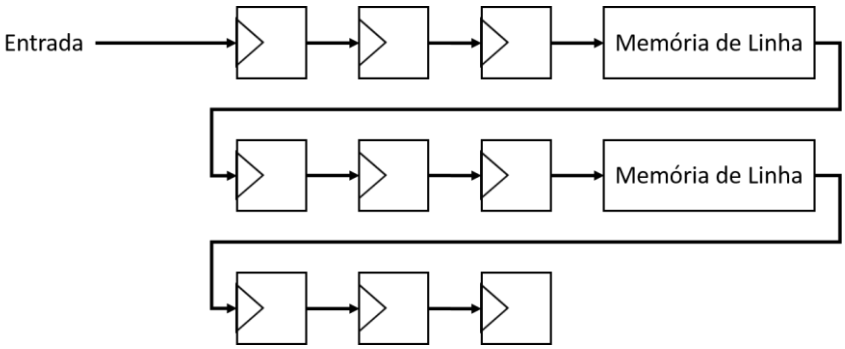


Figura 4.3.3: Princípio de funcionamento do circuito de acesso à janela de pixels.

O circuito do bloco "memoria_de_deslocamento" foi implementado por meio de um arranjo de blocos gerados automaticamente pela ferramenta MegaWizard Plug-in Manager do *software* Quartus II.

O bloco "operador_logico" é um circuito lógico combinacional desenvolvido em VHDL que realiza a operação morfológica no *pixel* central da janela de *pixels*. A operação é realizada conforme os parâmetros estabelecidos para a unidade de processamento. O código em VHDL do circuito é apresentado no Apêndice C.

O bloco "delay_sincronismo" é o circuito responsável por realizar o retardo dos sinais de sincronismo de vídeo. O seu circuito também foi implementado por meio de um arranjo de blocos gerados automaticamente pela ferramenta MegaWizard Plug-in Manager. Os circuitos dos blocos "memoria_de_deslocamento" e "delay_sincronismo" são apresentados nos Apêndices D e E.

A fim de atender as necessidades de algoritmos diversos, as operações lógicas também podem ser combinadas em conjunto com as operações morfológicas. Na Figura 4.3.4 é ilustrado o arranjo proposto para a extração de bordas. Esse arranjo é composto por duas unidades de processamento morfológico e uma porta lógica XOR. A extração de bordas é um requisito fundamental para o propósito de extração das linhas de navegação realizada na próxima fase do sistema. O fluxograma do algoritmo é ilustrado

na Figura 4.3.5. Para que a operação lógica XOR seja realizada entre imagens sincronizadas, a operação “nulo” foi realizada em paralelo com a segunda erosão.

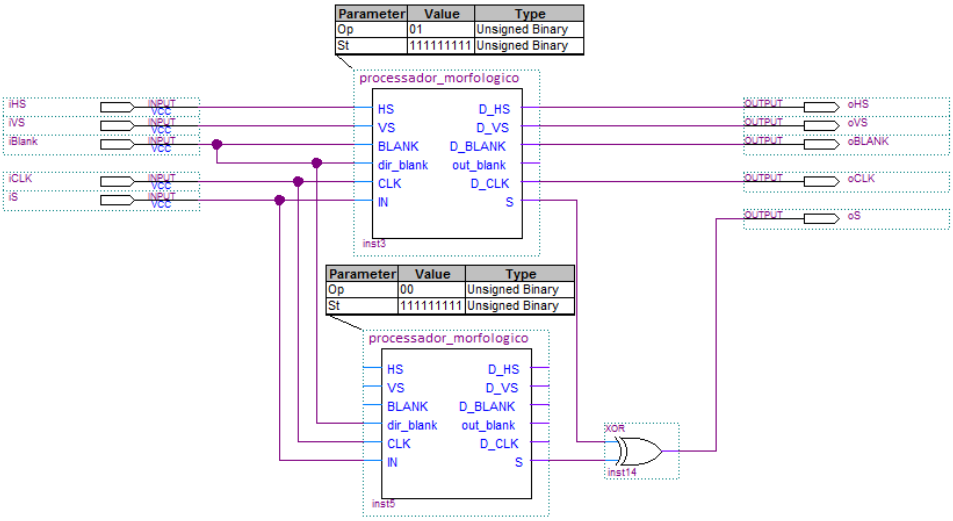


Figura 4.3.4: Arranjo para a extração de bordas.

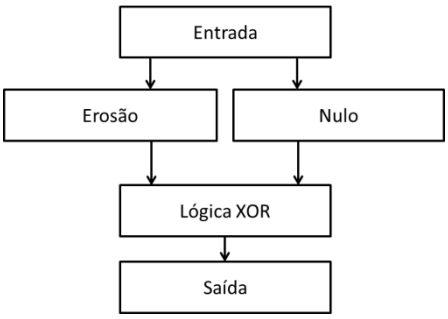


Figura 4.3.5: Fluxograma do algoritmo para extração de bordas.

4.4 EXTRAÇÃO DAS LINHAS DE NAVEGAÇÃO

Para a extração das linhas de navegação, no presente trabalho foi desenvolvido um circuito dedicado baseado no método de regressão por mínimos quadrados. O método consiste em uma análise estatística com o objetivo de verificar a existência de

uma relação funcional entre uma variável dependente com uma ou mais variáveis independentes.

Os parâmetros da regressão são os fatores α e β, conforme as equações:

$$\alpha = \frac{\sum X^2 \sum Y - \sum(XY) \sum X}{n \sum X^2 - (\sum X)^2} \tag{4.4.1}$$

$$\beta = \frac{n \sum(XY) - \sum X \sum Y}{n \sum X^2 - (\sum X)^2} \tag{4.4.2}$$

Os fatores α e β representam a inclinação da reta e a interceptação da reta com o eixo vertical, respectivamente. As variáveis X e Y são as coordenadas dos *pixels* que compõe os objetos na imagem binária e n é o número de amostras.

No contexto do presente trabalho, a análise consiste no cálculo dos fatores que definem as retas limite entre a região navegável e não navegável. Posteriormente, esses parâmetros são utilizados para a estimativa da linha central para a orientação do veículo. O processo de regressão ocorre em duas regiões distintas da imagem que, de acordo com as características locais e do posicionamento da câmera, são definidas pelo usuário por meio de parâmetros de configuração. A restrição das regiões para o cálculo de regressão permite minimizar os erros no processo de extração das linhas de orientação.

O diagrama do circuito desenvolvido para o cálculo da estimativa das linhas de navegação é ilustrado na Figura 4.4.1. O circuito é composto pelos blocos “regressao”, “filtro” e “linha_media”.

O cálculo dos parâmetros das retas é realizado por meio dos blocos “regressao”, sendo um bloco responsável pela reta que define a área limite navegável à direita da imagem e o outro à esquerda. O circuito foi implementado de forma que α mede a tangente do ângulo θ formado entre a linha de orientação e o eixo vertical da imagem e β o valor da coordenada horizontal na intersecção da linha de orientação com o eixo horizontal, conforme ilustrado na Figura 4.4.2. Os parâmetros x_0, x_1, y_0 e y_1 são as coordenadas que definem a área para o cálculo da regressão. Os subscritos A e B nos

parâmetros da Figura 4.4.2 indicam a região esquerda e direita da imagem, respectivamente, para o cálculo de regressão.

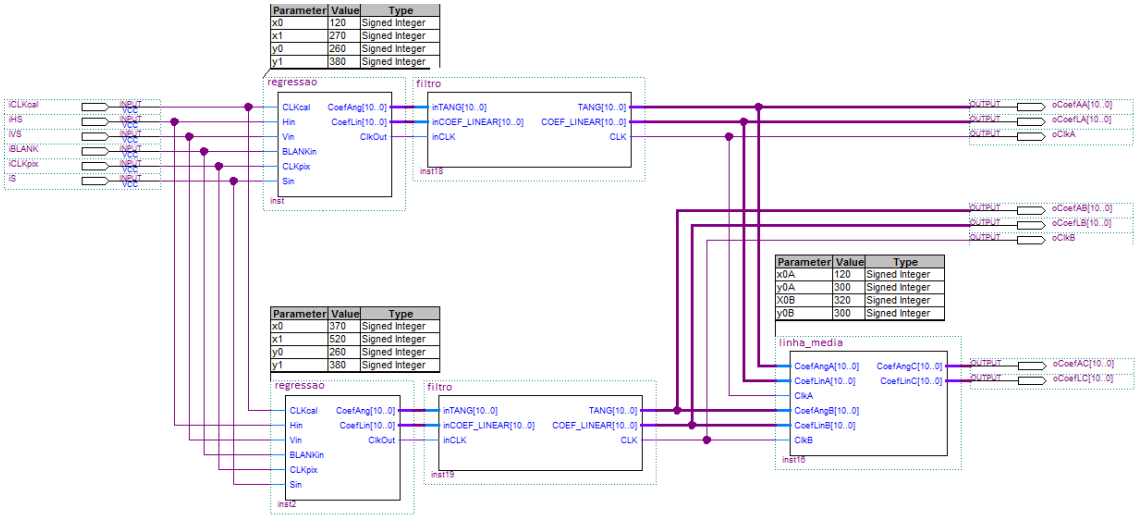


Figura 4.4.1: Circuito para a extração das linhas de navegação.

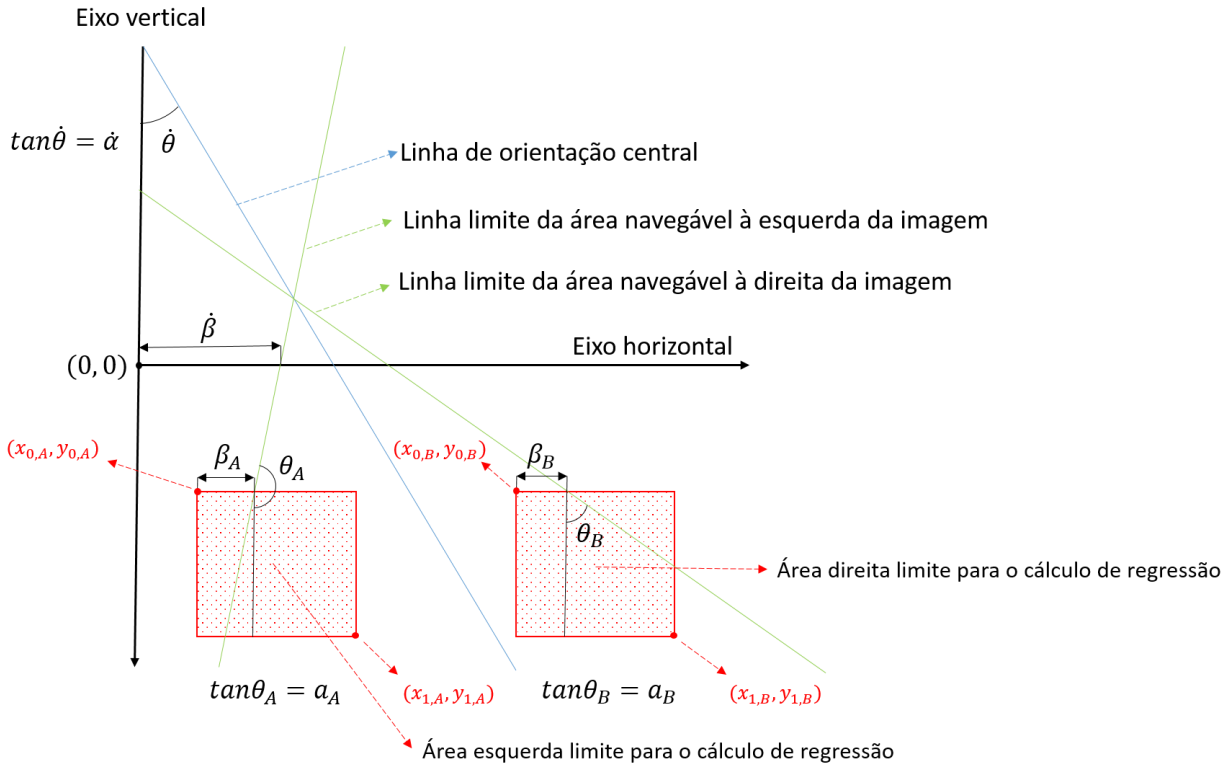


Figura 4.4.2: Parâmetros de orientação.

No exemplo ilustrado na Figura 4.4.3, a linha de orientação, destacado em azul, corresponde à reta intermediária entre as retas que contornam a região passível de navegação. A linha de orientação central é calculada pelo bloco “linha_media” por meio das Equações 4.4.3 e 4.4.4.

Figura 4.4.3: Representação das linhas de orientação em uma imagem típica de cena agrícola (Adaptado de SHARIFI & CHEN, 2015).

$$\dot{\alpha} = \frac{\alpha_A + \alpha_B}{2} \tag{4.4.3}$$

$$\dot{\beta} = \frac{(\beta_A + x_{0,A} - y_{0,A} \times \alpha_A) + (\beta_B + x_{0,B} - y_{0,B} \times \alpha_B)}{2} \tag{4.4.4}$$

Os fatores $\dot{\alpha}$ e $\dot{\beta}$ são os coeficientes angular e linear, respectivamente, da função que define a linha de orientação central.

Os blocos “filtro” realizam uma média aritmética em cada parâmetro obtido no processo de regressão, minimizando os efeitos causados por resultados dispersos. A média é realizada em conjuntos de vinte amostras, embora, de acordo com a aplicação, esse parâmetro possa ser modificado no projeto. Os circuitos dos blocos “regressao”, “filtro” e “linha_media” foram desenvolvidos em VHDL e os seus respectivos códigos são apresentados nos Apêndices F, G e H.

Os resultados dos coeficientes obtidos nessa etapa do sistema podem ser exportados para o sistema de controle de um robô por meio de um barramento paralelo de informações ou por meio de um protocolo de comunicação serial que pode ser implementado no próprio FPGA.

4.5 RESULTADOS

A seguir são apresentados os resultados obtidos em ensaios realizados na fase de desenvolvimento do sistema de visão proposto no presente trabalho. Os testes foram realizados nas etapas de classificação, processamento morfológico e extração das linhas de navegação. As imagens utilizadas para os experimentos iniciais foram, em sua maioria, extraídas de filmagens realizadas durante pesquisas anteriores da equipe do Núcleo de Ensino e Pesquisa em Automação e Simulação (NepAS). Posteriormente, um conjunto de imagens foi capturado em ambiente agrícola para o teste de validação final do projeto.

Para a realização dos ensaios foi utilizado um computador pessoal convencional com uma saída de vídeo analógica conectada à entrada de vídeo do DE2-70 e um monitor colorido no padrão VGA conectado à saída do sistema. As imagens de entrada foram reproduzidas pelo computador e as imagens processadas nas diferentes etapas do sistema foram capturadas por meio de fotografias do monitor.

Para os testes do circuito de classificação, os *pixels* das imagens foram classificados em três classes correspondentes ao céu, solo e vegetação. As configurações do circuito de classificação são realizadas por meio dos parâmetros que correspondem aos limiares máximos e mínimos dos componentes H, S e L. Os parâmetros estabelecidos empiricamente para o propósito dos testes são apresentados na Tabela 4.5.1.

Conforme mencionado anteriormente, os *pixels* com valores extremos de intensidade ou baixo nível de saturação apresentam limitações quanto a sua classificação. Dessa forma, os parâmetros limiares dos componentes S e L foram ajustados de forma a ignorar tais *pixels*. Na região correspondente ao céu, especificamente, uma parte significativa dos *pixels* não pode ser classificada como tons de azul devido ao seu alto nível de intensidade. Para a classificação dessa região foi utilizado um componente extra para classificar os *pixels* brancos e, posteriormente, realizado a concatenação desse conjunto com o conjunto dos *pixels* classificados em azul por meio de uma porta lógica OU. Os resultados obtidos nos testes do circuito de classificação dos *pixels* são ilustrados na Figura 4.5.1.

			Mínimo (10 bit)	Máximo (10 bit)
Céu	Azul	H	480	680
		S	120	1023
		L	160	840
	Branco	H	0	1023
		S	0	1023
		L	840	1023
Solo	Vermelho (H ≥ 0)	H	0	140
		S	160	1023
		L	160	840
	Vermelho (H < 0)	H	880	1023
		S	160	1023
		L	160	840
Vegetação	Verde	H	180	400
		S	120	1023
		L	160	840

Tabela 4.5.1: Parâmetros do circuito de classificação de cores.

Na fase de teste da etapa de processamento morfológico foram estabelecidas algumas sequências de operações para o processamento das imagens binárias obtidas pelo circuito de classificação.

Na Figura 4.5.2 são ilustrados os resultados obtidos no processamento morfológico das imagens correspondentes a classe da vegetação. A primeira coluna corresponde às imagens de entrada no circuito de classificação. A segunda coluna corresponde aos resultados da aplicação de uma operação de erosão e a terceira coluna aos resultados da aplicação de duas operações de erosão consecutivas. O elemento estruturante utilizado foi um quadrado de dimensão 3 × 3.

O próximo teste realizado, embora as imagens utilizadas não foram de um ambiente agrícola, permitiu verificar o comportamento do circuito nessa etapa do desenvolvimento do projeto. O teste consistiu no processamento de imagens de um corredor de terra em duas situações. Na primeira situação foi realizada uma sequência de duas operações de abertura, três operações de dilatação e oito operações de fechamento. Na segunda situação, a sequência é complementada por uma operação de extração de bordas. Os resultados do teste são ilustrados na Figura 4.5.3 e os parâmetros utilizados para classificar os *pixels* correspondentes ao solo são apresentados na Tabela 4.5.2. As operações morfológicas e os parâmetros limiares utilizados no teste foram obtidos empiricamente.

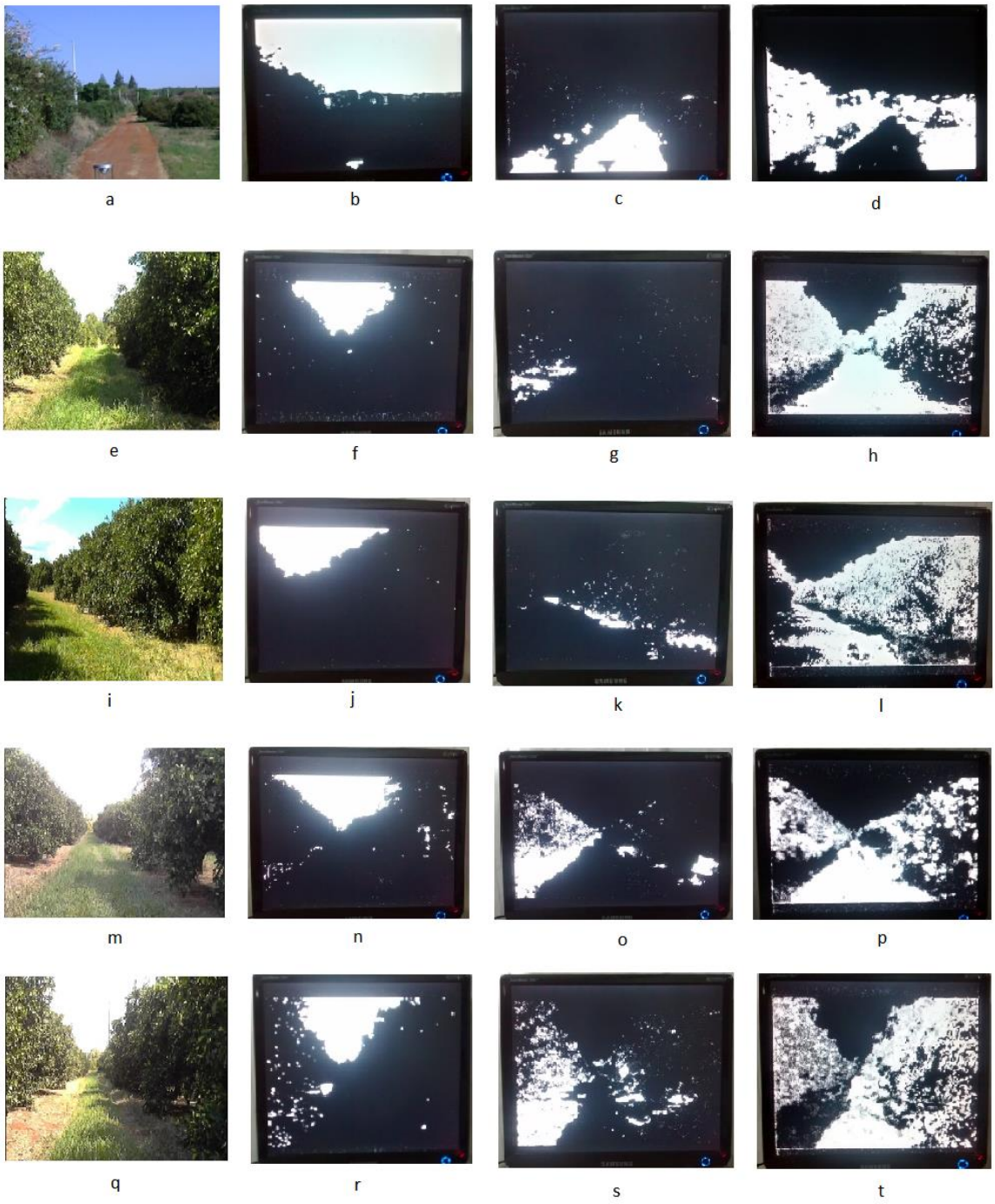


Figura 4.5.1: Resultados do processo de classificação dos pixels: (a), (e), (i), (m) e (q) imagens originais; (b), (f), (j), (n) e (r) imagens binárias da classe correspondente à região do céu; (c), (g), (k), (o) e (s) imagens binárias da classe correspondente à região do solo; (d), (h), (l), (p) e (t) imagens binárias da classe correspondente à região da vegetação.

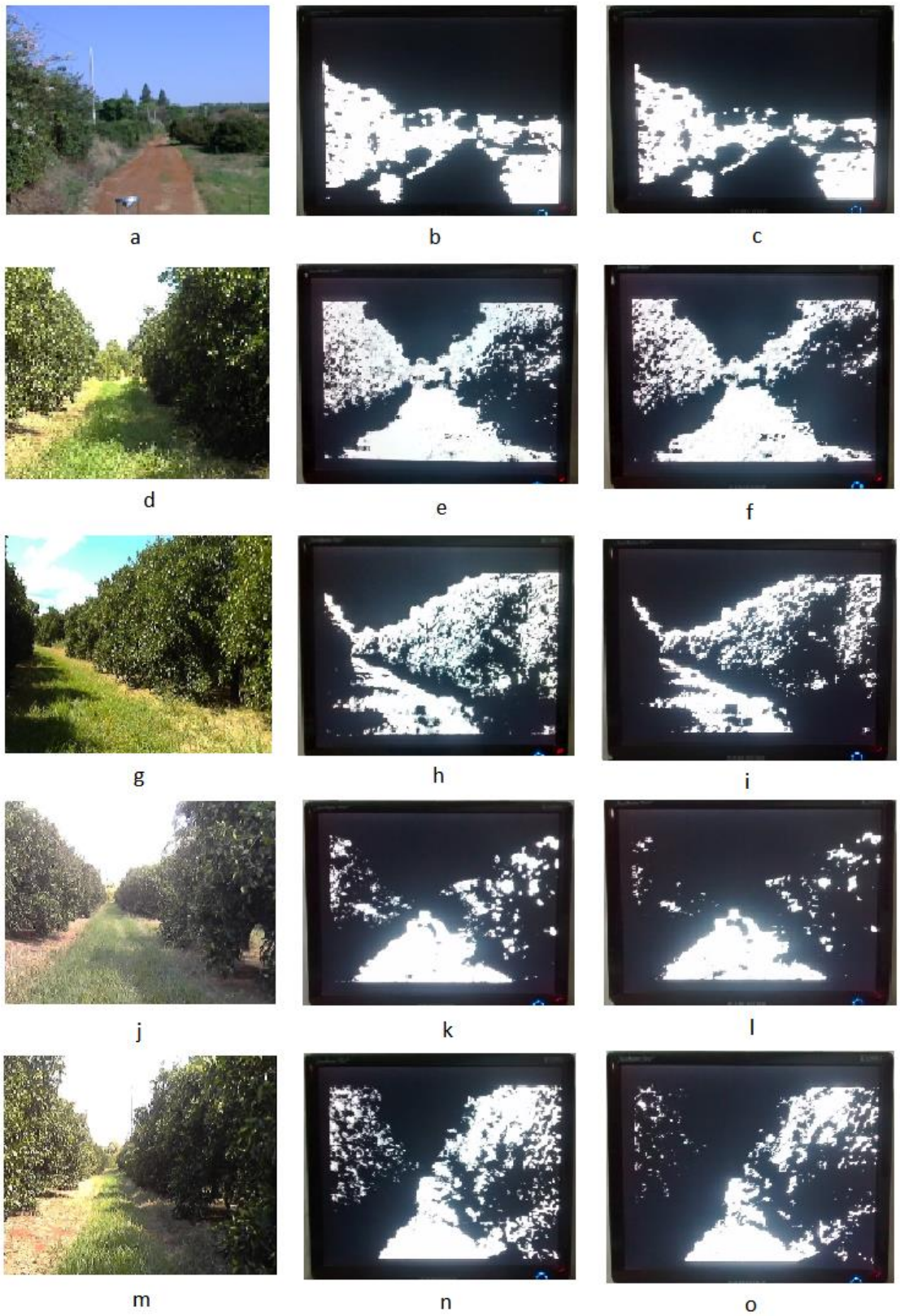


Figura 4.5.2: Resultados do processamento morfológico: (a), (d), (g), (j) e (m) imagens originais; (b), (e), (h), (k) e (n) operação de erosão; (c), (f), (i), (l) e (o) duas operações de erosão.

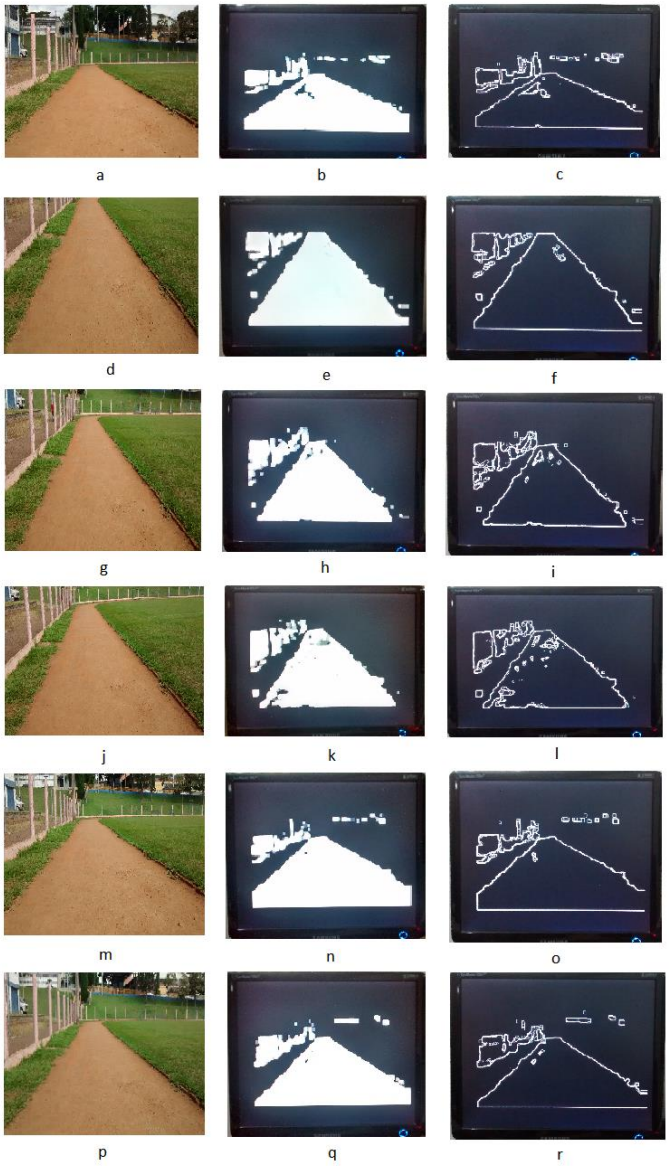


Figura 4.5.3: Resultados do processamento morfológico em duas sequências de operações: (a), (d), (g), (j), (m) e (p) imagens originais; (b), (e), (h), (k), (n) e (q) sequência de duas operações de abertura, três operações de dilatação e oito operações de fechamento; (c), (f), (i), (l), (o) e (r) sequência de duas operações de abertura, três operações de dilatação, oito operações de fechamento e extração de bordas.

		Mínimo (10 bit)	Máximo (10 bit)
Solo	H	85	110
	S	160	1023
	L	160	840

Tabela 4.5.2: Parâmetros utilizados para classificar os pixels correspondentes ao solo nas imagens do corredor de terra.

Os resultados dos testes na etapa de extração das linhas de navegação são ilustrados na Figura 4.5.4. As retas destacadas em azul correspondem aos resultados das operações de regressão e a linha central para a orientação do veículo. Os retângulos destacados em vermelho correspondem às áreas de aplicação da regressão e os *pixels* destacados em verde correspondem à imagem binária obtida após a etapa de extração de bordas. Na Figura 4.5.5 são ilustrados os resultados da extração das linhas de navegação em imagens de carreadores de um pomar de laranja.

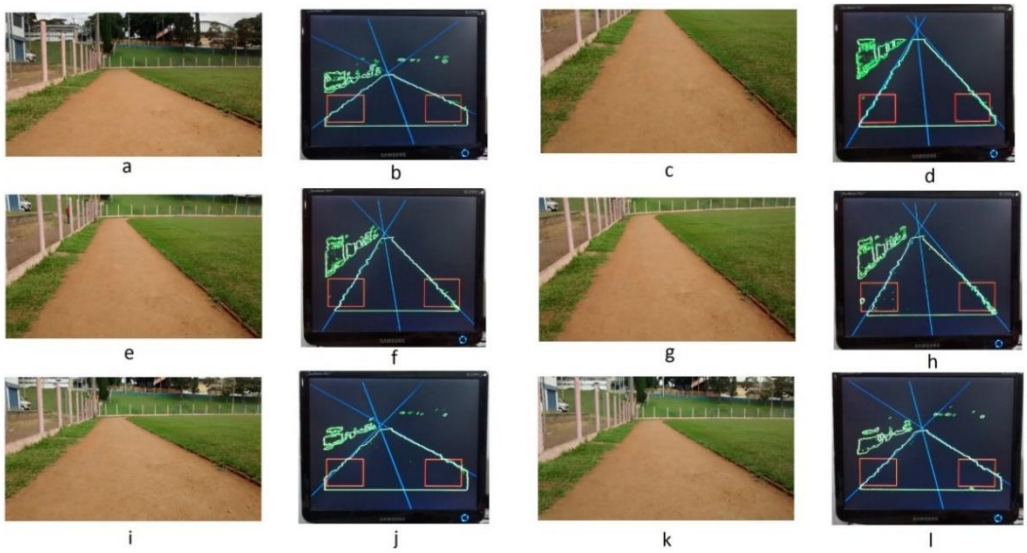


Figura 4.5.4: Resultados do processo de extração das linhas de navegação: (a), (c), (e), (g), (i) e (k) imagens originais; (b), (d), (f), (h), (j) e (l) imagens de saída do sistema.

No último teste para a validação do projeto foram processadas 142 imagens de estradas de terra internas a uma fazenda. Os resultados foram classificados em verdadeiro positivo (VP) para as imagens cujas linhas de orientação foram identificadas de acordo com os propósitos do projeto, ou falso positivo (FP) caso contrário. Os resultados obtidos são apresentados na Tabela 4.5.3. A acurácia obtida no teste foi de aproximadamente 87%. Na primeira etapa do circuito desenvolvido para a estimativa das linhas de orientação são calculados os coeficientes que representam as retas que melhor contornam os limites da área passível de navegação. O circuito desenvolvido para essa finalidade possui a característica em que o resultado do processamento de uma imagem qualquer corresponde sempre aos coeficientes das restas mesmo quando essas não estão presentes na imagem e, portanto, o circuito não apresenta resultados verdadeiros negativos e falsos negativos.

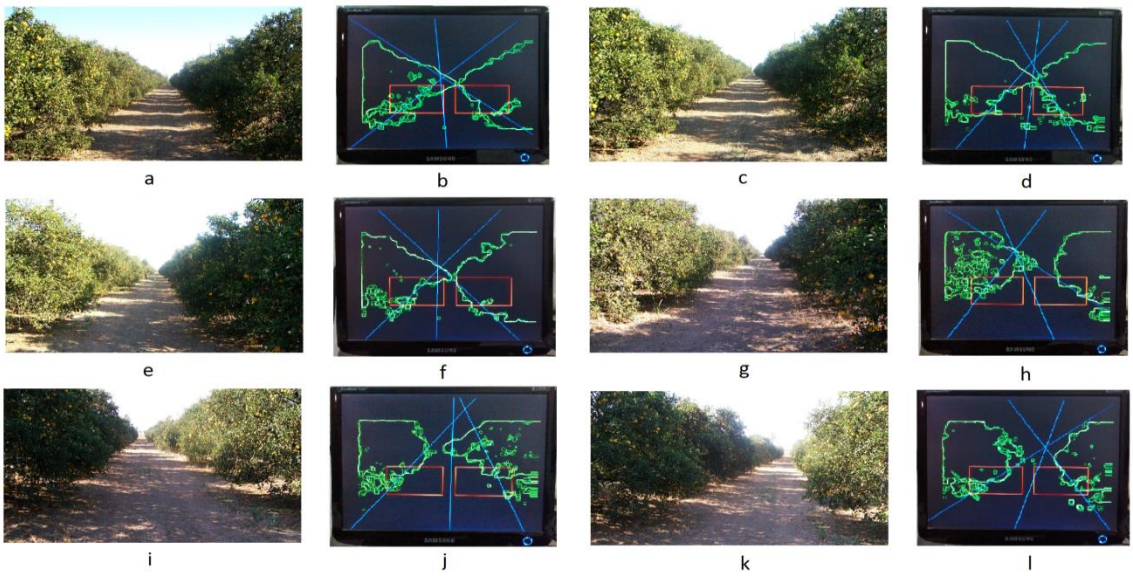


Figura 4.5.5: Resultados do processo de extração das linhas de navegação de corredores em um pomar de laranja: (a), (c), (e), (g), (i) e (k) imagens originais; (b), (d), (f), (h), (j) e (l) imagens de saída do sistema.

Classificação	Número de Imagens
VP	124
FP	18
Total	142

Tabela 4.5.3: Classificação das imagens.

Para o processamento das imagens, o circuito de classificação foi configurado para estabelecer duas classes de *pixels* correspondentes à região do solo e região da vegetação, denominadas de C1 e C2, respectivamente. Os parâmetros de configuração do circuito de classificação são apresentados na Tabela 4.5.4. Na etapa de processamento morfológico, primeiramente, para fins de filtragem, foram estabelecidas as sequências de operações de duas erosões, uma abertura e oito fechamentos para a classe C1 e, uma abertura, dez dilatações e quatro fechamentos para a classe C2, totalizando vinte unidades de processamento morfológico para cada classe. Posteriormente foi estabelecida a operação lógica AND entre a classe C1 e o complemento da classe C2. Os parâmetros de calibração utilizados no teste foram obtidos de forma empírica. A estratégia adotada consiste em classificar a região passível de navegação excluindo a região composta majoritariamente por vegetação da região

correspondente ao solo. O resultado obtido nessa etapa foi complementado pelo algoritmo de extração de bordas apresentado no Item 4.3.

		Mínimo (10 bit)	Máximo (10 bit)
C1	H	0	80
	S	160	1023
	L	160	940
C2	H	120	420
	S	110	1023
	L	110	1023

Tabela 4.5.4: Parâmetros estabelecidos para as classes C1 e C2.

Nas Figuras 4.5.6, 4.5.7, 4.5.8 e 4.5.9 são ilustrados alguns exemplos de resultado verdadeiro positivo.

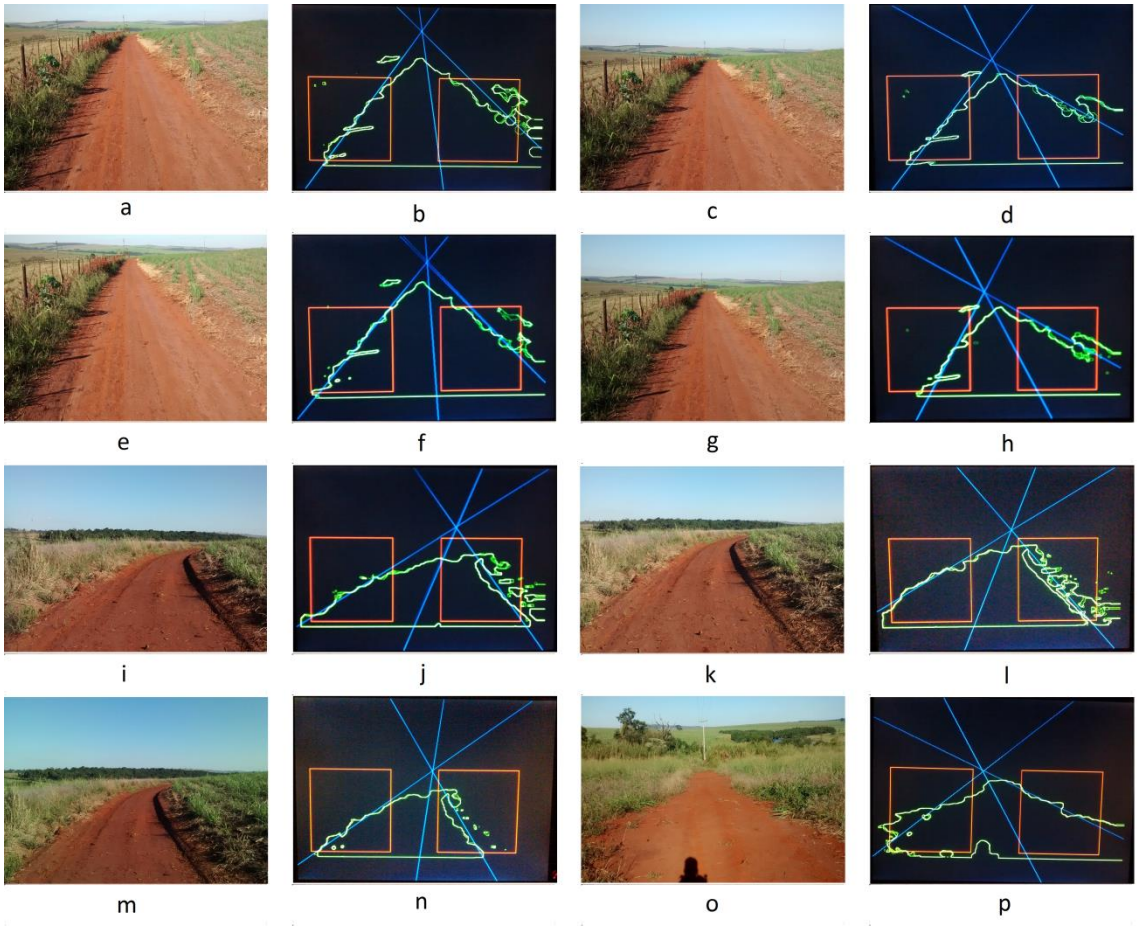


Figura 4.5.6: Exemplo de imagens com resultados verdadeiros positivos: (colunas 1 e 3) imagens originais; (colunas 2 e 4) imagens de saída do sistema.

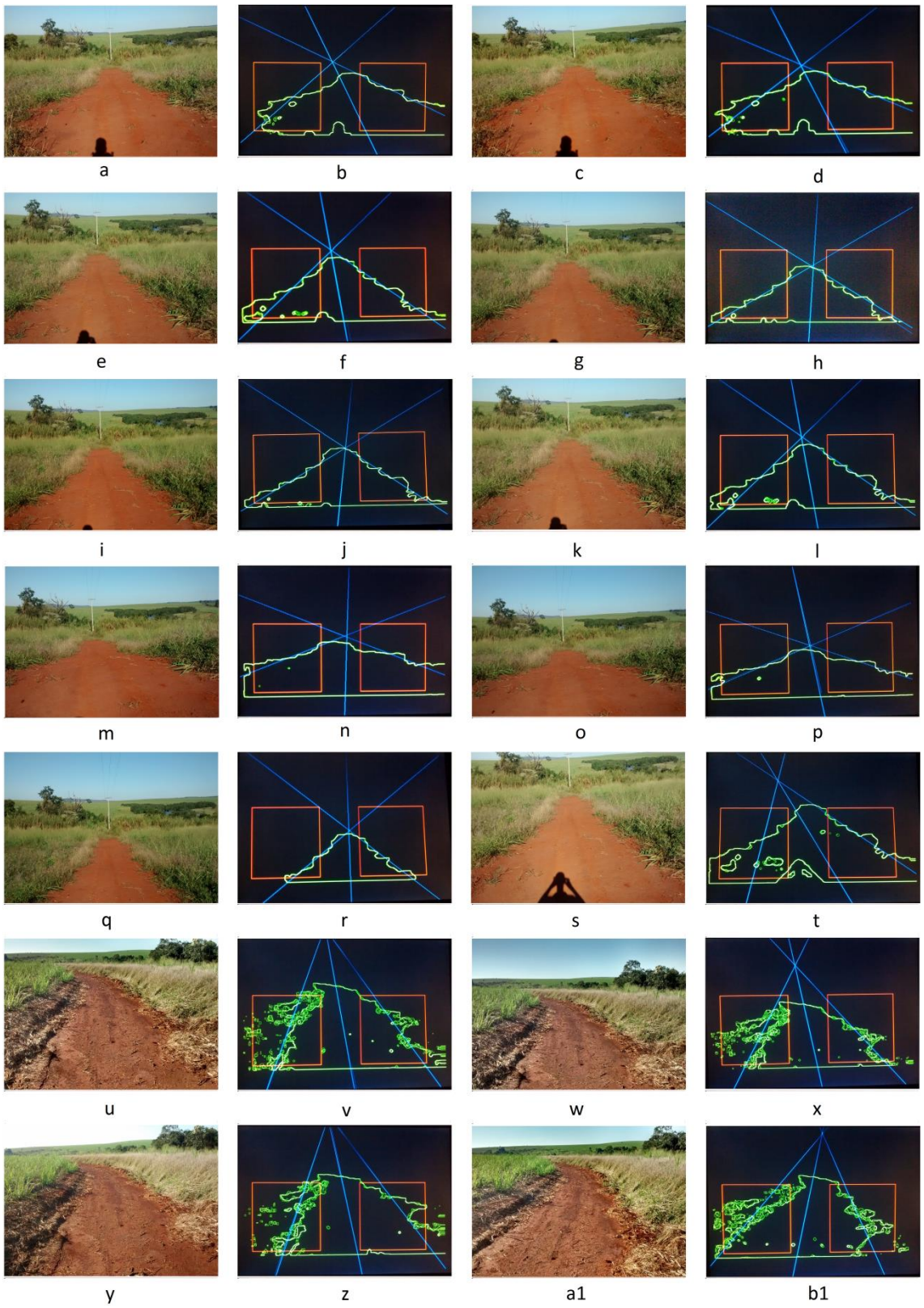


Figura 4.5.7: Exemplo de imagens com resultados verdadeiros positivos: (colunas 1 e 3) imagens originais; (colunas 2 e 4) imagens de saída do sistema.

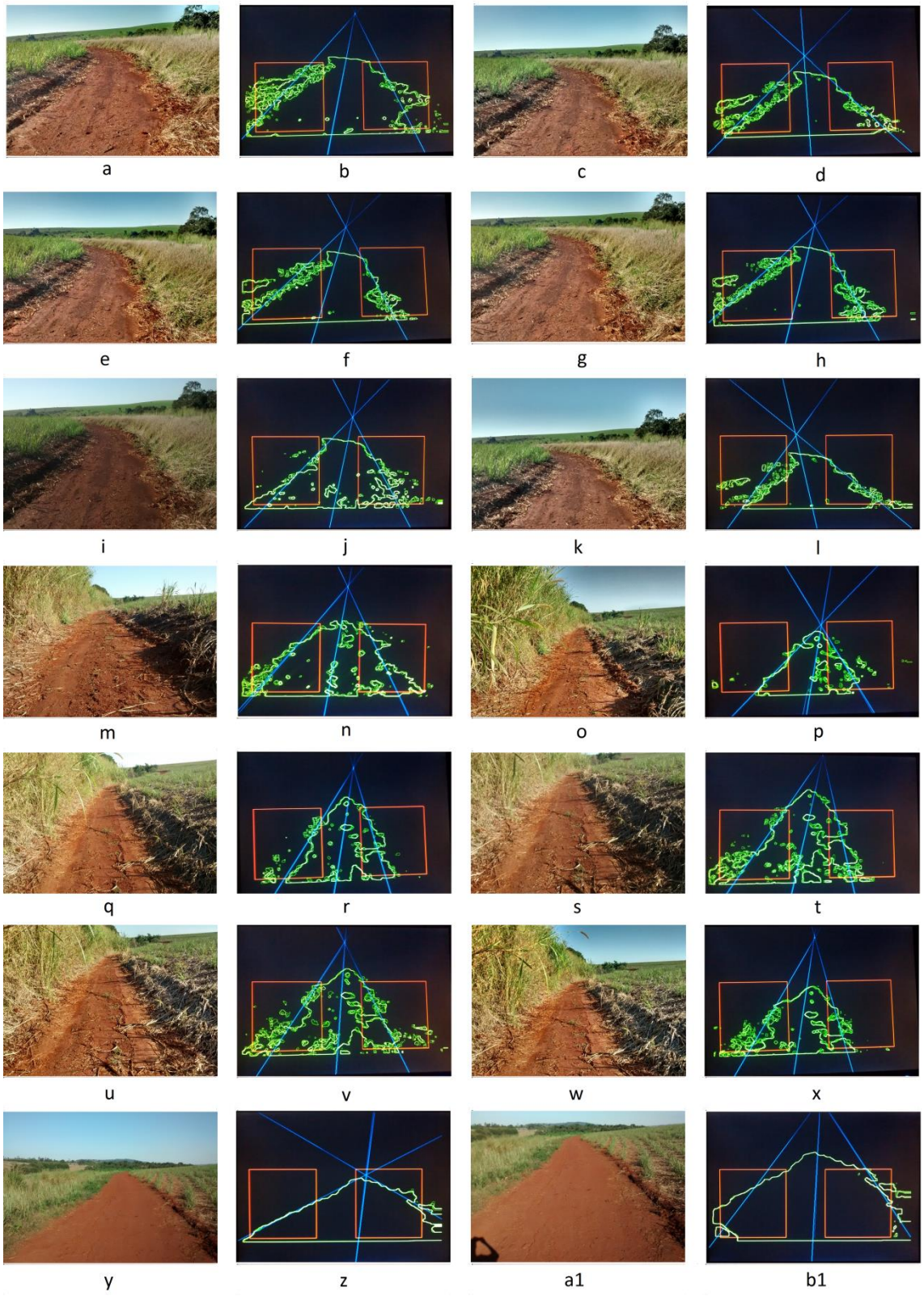


Figura 4.5.8: Exemplo de imagens com resultados verdadeiros positivos: (colunas 1 e 3) imagens originais; (colunas 2 e 4) imagens de saída do sistema.

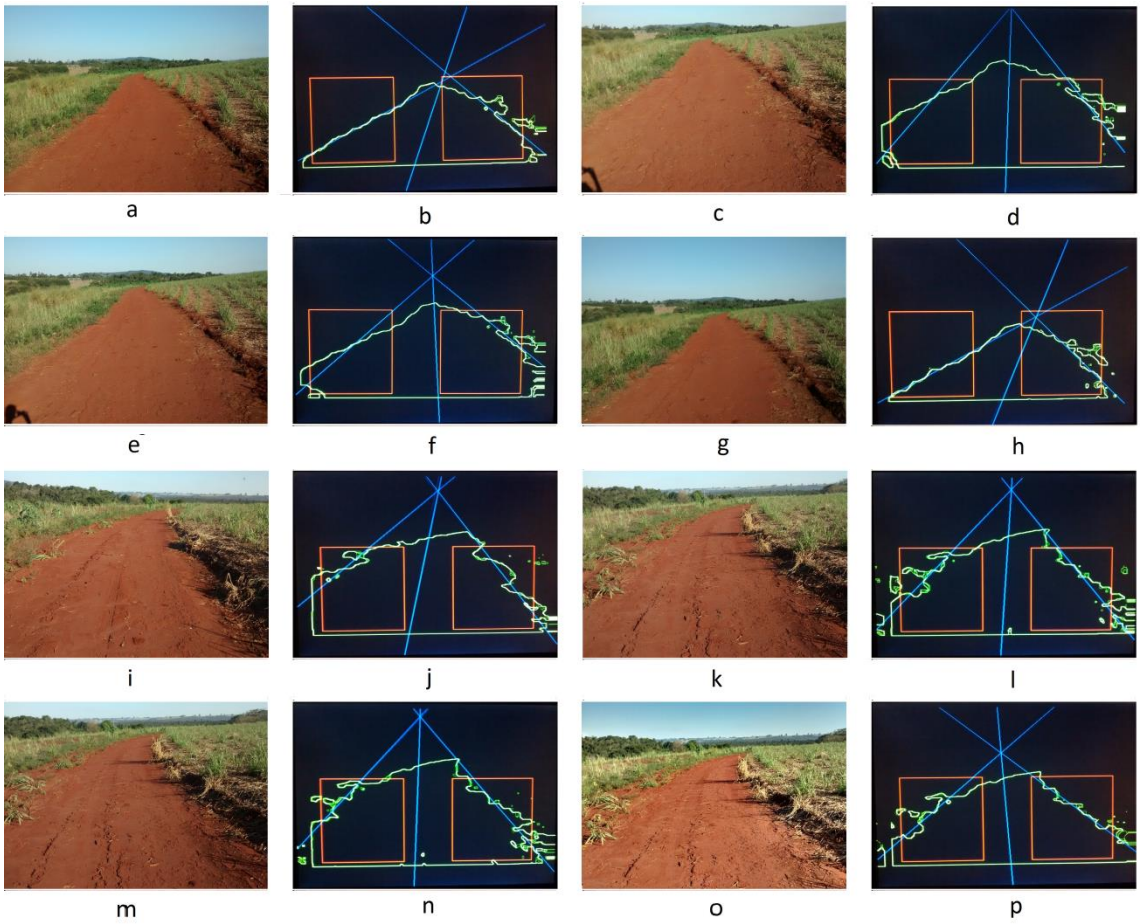


Figura 4.5.9: Exemplo de imagens com resultados verdadeiros positivos: (colunas 1 e 3) imagens originais; (colunas 2 e 4) imagens de saída do sistema.

A seguir, são feitas algumas considerações sobre os resultados apresentados nas Figuras 4.5.6 a 4.5.9.

Nos conjuntos (a, b) e (e, f) da Figura 4.5.6, a extração das linhas que contornam a região passível de navegação é identificada de forma coerente. Embora a região inferior direita da área de plantio tenha uma maior predominância de tonalidades que correspondem a terra e a vegetação seca, as linhas de contorno, nessa região, foram identificadas com o auxílio da metodologia adotada em que a região classificada como vegetação é dilatada e excluída da região classificada como navegável. O sentido de orientação da estrada nas duas imagens originais está levemente inclinado para o lado esquerdo, correspondendo com a linha média calculada a partir das linhas de contorno. Nos conjuntos (i, j) e (k, l), a região do lado direito da estrada apresenta uma sombra formada por um desnível com dimensões relevantes. A região da sombra não é

classificada em ambas as classes C1 e C2 e, portanto, definida pelo sistema como não navegável. O efeito observado nas imagens de saída é um leve deslocamento lateral da linha que identifica o limite direito. A linha média calculada em ambos os conjuntos apresenta uma pequena inclinação à direita. A princípio a estrada nas imagens originais aparenta estar orientada em um ângulo perpendicular ao horizonte. Ao observar com atenção a imagem de saída do sistema fica mais evidente que o trajeto ideal para o deslocamento de um veículo é inclinado à direta. A região do contorno esquerdo da estrada nas imagens dos conjuntos (u, v) e (w, x) da Figura 4.5.7 apresentam uma tonalidade acinzentada. O efeito ocasionado na etapa de classificação é uma aparência de granulação irregular observada nas imagens de saída. A linha do contorno esquerdo foi identificada em uma região mediana à área afetada e com um ângulo coerente ao contorno de estrada. O resultado das linhas médias também foi coerente com a orientação da estrada. Nos conjuntos (e, f) e (g, h) da Figura 4.5.8, as linhas de orientação indicam uma inclinação à direita para o deslocamento de um veículo. Embora na região mais afastada da estrada exista uma curva à esquerda, a orientação do veículo, em uma hipótese, seria corrigida ao longo do trajeto. Nos conjuntos (q, r) até (u, v), a presença de palha sobre a estrada ocasionou em uma concentração de buracos na região classificada como passível de navegação, reduzindo a definição das linhas de contorno da estrada. Os resultados também foram positivos na identificação das linhas de orientação em tal condição. Os conjuntos (y, z) e (a1, b1) da Figura 4.5.8 e os conjuntos (a, b) até (g, h) da Figura 4.5.9 são exemplos de condições onde as linhas de contorno da estrada são apresentadas com maior clareza após os processos de classificação e processamento morfológico.

Nas Figura 4.5.10 e 4.5.11 são ilustrados os resultados falsos positivos. Na análise dos resultados, o conjunto (a, b) da Figura 4.5.10, embora a linha média calculada apresente uma inclinação coerente com a orientação da estrada na imagem, foi classificado com resultado insatisfatório devido à identificação incoerente do contorno direito da estrada. As irregularidades apresentadas após as etapas de classificação e processamento morfológico devido à presença de sombras e a concentração de terra e vegetação seca no canto inferior direito da imagem contribuíram para o erro na identificação da linha de contorno. No conjunto (c, d) a linha média apresentou um ângulo correto, no entanto, ocorreu um deslocamento acentuado da linha para o lado direito da imagem. O contorno esquerdo foi identificado satisfatoriamente e o contorno

direito sofreu os mesmos efeitos do conjunto citado anteriormente. Nos conjuntos (e, f) e (g, h) todas as linhas de orientação foram identificadas incorretamente. Os aspectos observados foram vegetação seca e sombra no contorno da estrada. No conjunto (k, l) a sombra na região central inferior da imagem ocasionou no desvio da linha do contorno direto da estrada. Nos conjuntos (m, n) e (o, p) o mesmo efeito foi observado na região esquerda da imagem. Nos conjuntos (c, d) até (m, n) da Figura 4.5.11, a identificação correta das linhas de contorno não foi possível devido às irregularidades na imagem processada, ocasionadas por uma grande concentração de vegetação seca e terra no contorno da estrada. Embora exista um erro na orientação das linhas de contorno, a linha média calculada apresentou coerência com a orientação da estrada.

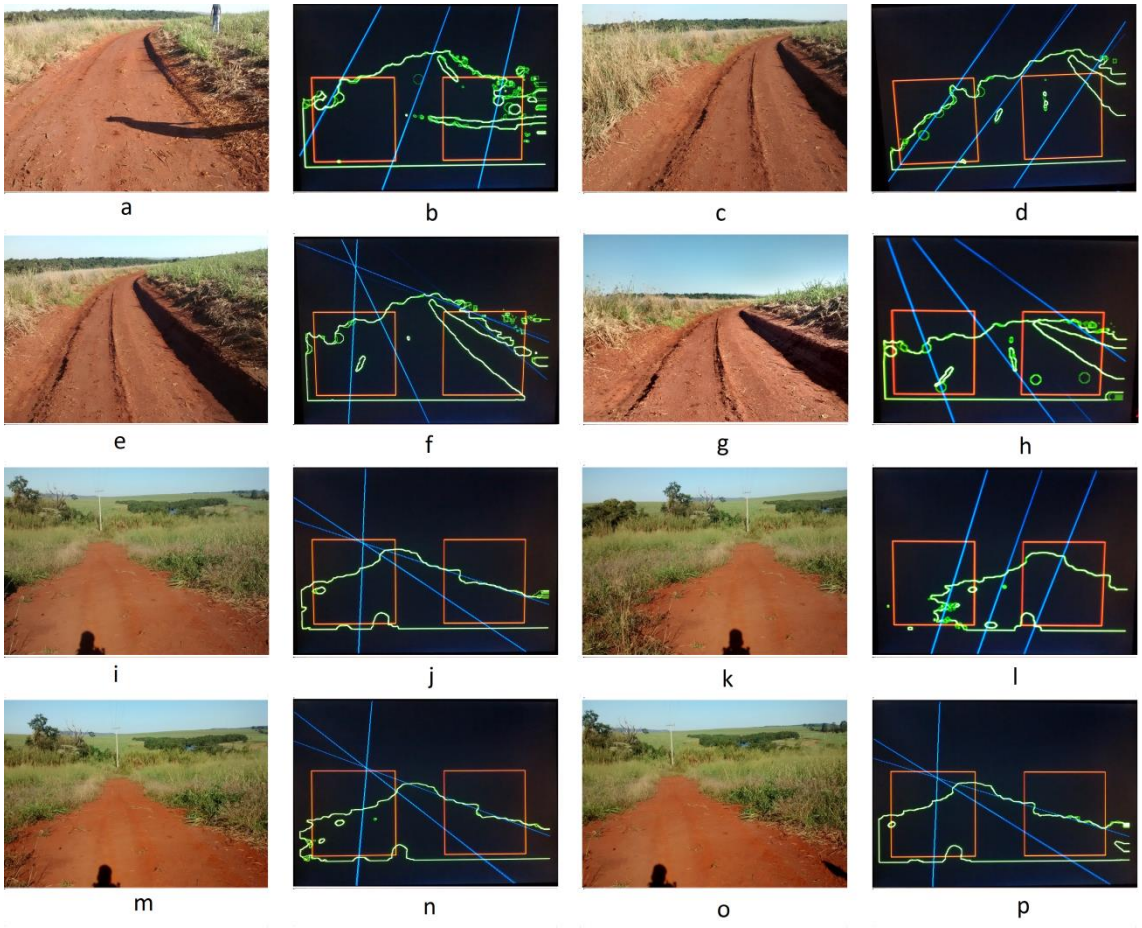


Figura 4.5.10: Imagens com resultados falsos positivos: (colunas 1 e 3) imagens originais; (colunas 2 e 4) imagens de saída do sistema.

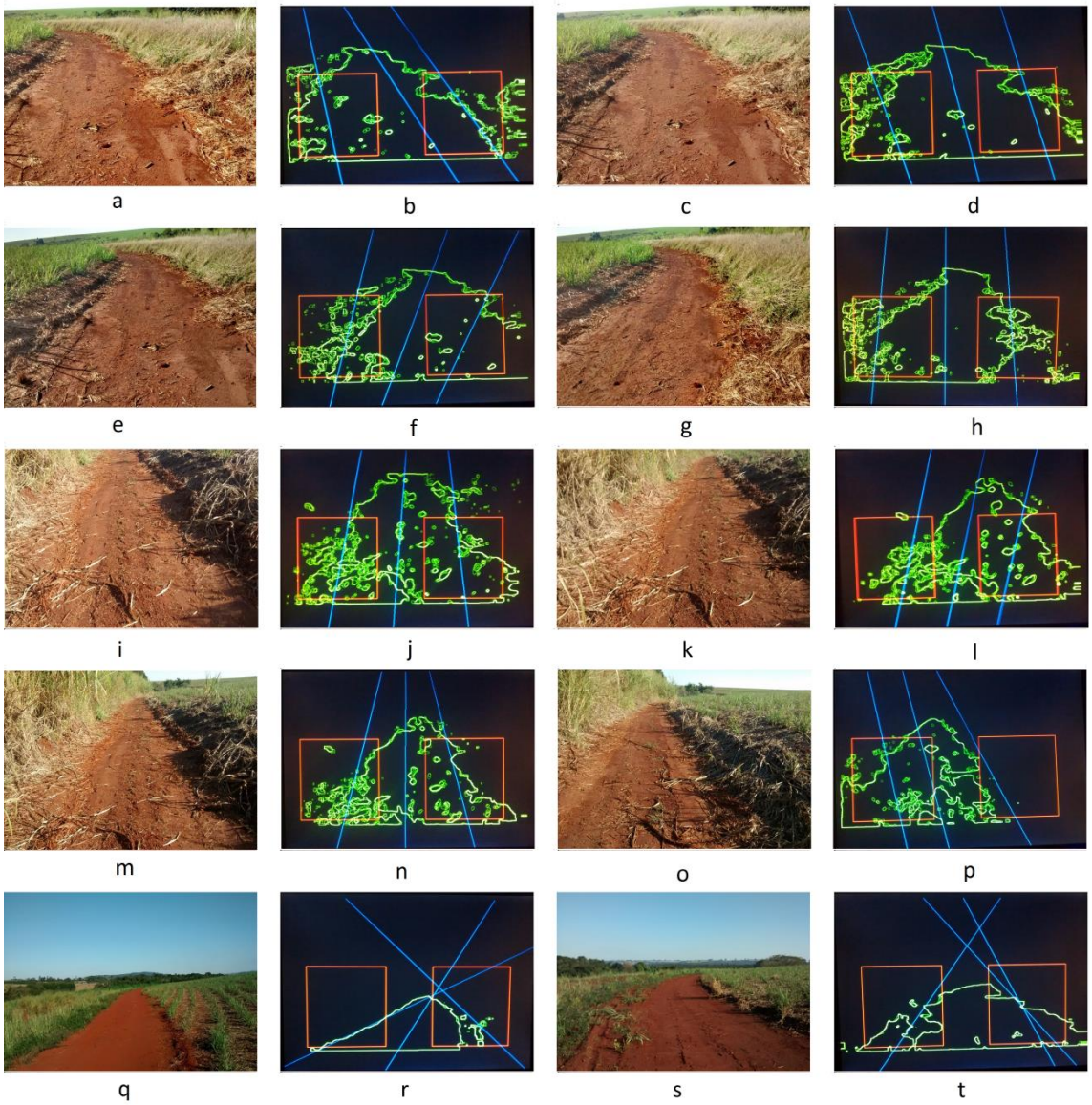


Figura 4.5.11: Imagens com resultados falsos positivos: (colunas 1 e 3) imagens originais; (colunas 2 e 4) imagens de saída do sistema.

4.6 CONSIDERAÇÕES FINAIS

Nesse capítulo foram apresentados os aspectos construtivos do sistema de visão computacional proposto no presente trabalho. O circuito foi implementado por meio da ferramenta de desenvolvimento DE2-70 e pode ser facilmente adaptado para outras plataformas que atendam alguns requisitos mínimos, como, por exemplo, quantidade de LEs. Também foram apresentados os testes realizados no desenvolvimento do projeto para a validação do circuito. Alguns trabalhos apresentados no Capítulo 3 podem fornecer informações relevantes que demonstram a aplicabilidade da metodologia de visão adotada para o desenvolvimento do circuito.

5 CONCLUSÕES

O *hardware* dedicado para a navegação de robôs móveis agrícolas apresentado no presente trabalho foi desenvolvido com base em uma metodologia de visão de baixo custo computacional, ideal para operar em condições onde as características visuais são bem definidas. O sistema consiste primeiramente em um circuito dedicado para a classificação dos *pixels* no modelo de cores HSL. Nessa etapa do projeto os parâmetros do circuito de classificação podem ser ajustados de forma a identificar os *pixels* que pertencem às regiões passíveis de navegação. O circuito desenvolvido para essa finalidade apresentou resultados coerentes com os objetivos do trabalho. A utilização do modelo de cores HSL permitiu melhorar o desempenho do sistema em condições de variação de iluminação. A segunda etapa do sistema consiste em um arranjo de unidades de processamento que pode ser configurado para realizar uma ampla variedade de algoritmos morfológicos para o processamento de imagens binárias. A arquitetura apresentada para o processamento morfológico permitiu desempenho em tempo real na realização de tarefas, como, por exemplo, filtragem e extração de bordas. Nos testes realizados durante o desenvolvimento do projeto foram utilizadas até 40 unidades de processamento morfológico. A etapa final do projeto consiste em um circuito dedicado para a extração das linhas de orientação. O circuito apresentado foi baseado no método de regressão linear por mínimos quadrados. A arquitetura proposta para o circuito de extração permitiu desempenho em tempo real no cálculo dos parâmetros das retas que definem a área passível de navegação e a linha central para a orientação do veículo.

O circuito apresentado neste trabalho consiste em uma arquitetura *pipeline* para o processamento das imagens. Essa arquitetura permite o processamento das imagens na mesma taxa de quadros por intervalo de tempo da câmera de vídeo utilizada, desde que, para a implementação por meio de um FPGA, não ocorram restrições de propagação de sinais no dispositivo. Nos testes realizados neste trabalho as imagens utilizadas foram geradas por meio de um computador configurado para operar em 60 quadros por segundo e, portanto, para a maioria das aplicações agrícolas o processamento pode ser considerado tempo real.

Nos testes realizados o sistema apresentou resultados coerentes no cálculo dos parâmetros de orientação. A utilização de processamento morfológico permitiu melhorar o desempenho do sistema em situações onde as características visuais da cena não permitiram a identificação definida da área passível de navegação por meio do circuito de classificação. No entanto, a configuração dos parâmetros, em tais situações, demandou bastante experiência e tempo para ajustar o sistema. O tempo para compilar o projeto também é outro fator relevante na fase de ajustes do sistema para uma determinada área de cultivo. Dessa forma, a implementação de uma interface para a programação dos parâmetros deve ser considerada para trabalhos futuros.

O emprego de circuitos dedicados para a navegação de robôs móveis agrícolas pode ser uma alternativa para aplicações cujas cenas apresentam características visuais relevantes que podem ser extraídas para determinar a orientação do veículo. A utilização de um *hardware* dedicado elimina a necessidade de embarcar computadores industriais, reduzindo o custo final do projeto. Além disso, o sistema de navegação apresentado permite o processamento de imagens em tempo real, possibilitando o seu emprego em aplicações que exigem velocidade de deslocamento relativamente elevada, como, por exemplo, a pulverização de lavouras.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ALTERA. Nios II Processor: The World's Most Versatile Embedded Processor. Disponível em: https://www.altera.com/products/processors/overview.html. Acesso em: 22 de jun. 2015.

BEN, F.H., ELHOSSINI, A., JUURLINK, B. An efficient and flexible FPGA implementation of a face detection system. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9040, pp. 243-254, 2015.

BOBDA, C. Introduction to Reconfigurable Computing: Architectures, Algorithms, and Applications. Springer Publishing Company, Incorporated, 2007.

BONATO, V. Projeto de um Módulo de Aquisição e Pré-processamento de Imagem Colorida Baseado em Computação Reconfigurável e Aplicado a Robôs Móveis. Dissertação – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos. 2004.

BOYLE, W.; SMITH, G. Charge Coupled Semiconductor Devices. Relatório técnico vol 49 pp.587, Bell Systems Technical Journal, 1970.

BROWN, S.; VRANESIC, Z. Fundamentals Of Digital Logic With Vhdl Design. Toronto: Mc Graw Hill, 2000.

DE MILANO, D., HONORATO, L. B. Visão Computacional. Universidade Estadual de Campinas, Faculdade de Tecnologia. 2010.

CHOI, K. H., HAN, S. K., HAN, S. H., PARK, K., KIM, K., KIM, S. Morphology-based guidance line extraction for an autonomous weeding robot in paddy fields, Computers and Electronics in Agriculture, Volume 113, Pages 266-274, ISSN 0168-1699, 2015.

COMANICIU, D., MEER, P. Mean shift: A robust approach toward feature space analysis. Pattern Analysis and Machine Intelligence, IEEE Transactions on 24.5, pg. 603-619, 2002.

CORREA, D. S. O. Navegação autônoma de Robôs Móveis e Detecção de Intrusos em Ambientes Internos Utilizando Sensores 2D e 3D. Dissertação – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos. 2013.

ENGLISH A., ROSS P., BALL D., CORKE P. Vision based guidance for robot navigation in agriculture (2014) Proceedings - IEEE International Conference on Robotics and Automation, art. no. 6907079, pp. 1693-1698.

FACON, J. A Morfologia Matemática e suas Aplicações em Processamento de Imagens. VII Workshop de Visão Computacional. Universidade Federal do Paraná, Curitiba. 2011.

FAIRCHILD IMAGING. Our History. Disponível em: http://www.fairchildimaging.com/our-history. Acesso em: 10 de jun. 2015.

GOIS, M. M. Representação Nó-profundidade em FPGA para Algoritmos Evolutivos Aplicados ao Projeto de Redes de Larga-escala. Dissertação - Universidade de São Paulo, São Carlos. 2011.

GONÇALES, R. Dispositivo de Varredura Laser 3D Terrestre e Suas Aplicações na Engenharia, com Ênfase em Túneis. Dissertação – Escola Politécnica, Universidade de São Paulo, São Carlos. 2007.

GONZALES, R. C. & WOODS, R. E. Digital Image Processing. 3.ed. New Jersey, EUA: Prentice-Hall Inc. 2007. ISBN 013168728X.

JIANG, G., ZHAO C., SI Y., "A machine vision based crop rows detection for agricultural robots," in Wavelet Analysis and Pattern Recognition (ICWAPR), 2010 International Conference, pg.114-118, 2010.

HAIGANG, Y., JIA, Z., JIABIN, S., LE, Y. Review of Advanced FPGA Architectures and Technologies. *Journal of Electronics*, v. 31, n. 5, 2014.

HASANI, Y., BODENSTORFER, E., BRODERSEN, J., MAYER, K.J. FPGA based image processing for optical surface inspection with real time constraints (2015) Proceedings of SPIE - The International Society for Optical Engineering, 9445, art. no. 94450P.

HIREMATH, S., EVERT, F. K., BRAAK, C. T., STEIN, A, HEIJDEN, G. Image-based particle filtering for navigation in a semi-structured agricultural environment, Biosystems Engineering, Volume 121, May 2014, Pages 85-95, ISSN 1537-5110.

HOSSAIN, Q. D. Design and Characterization of a Current Assisted Photo Mixing Demodulator for TOF Based 3d CMOS Image Sensor. University of Trento, Trento. 2010.

HUI, N. Automatic thresholding for defect. Detection. Pattern Recognition Letters, Vol 27, No. 14, pp. 1644-1649, 2006.

HURWITZ, J.E.D, DENYER, P.B, Baxter, D.J; An 800K-Pixel Colour CMOS Sensor for Consumer Still Cameras; Presented as part of Electronic Imaging at SPIE Photonics West. 1997.

KLASER, R. L. Navegação de Veículos Autônomos em Ambientes Externos não Estruturados Baseada em Visão Computacional. Dissertação – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos. 2014.

LI, L., WYRWICZ, A.M. Design of an MR image processing module on an FPGA chip (2015) Journal of Magnetic Resonance, 255, pp. 51-58.

LULIO, L. C. Técnicas de Visão Computacional Aplicadas ao Reconhecimento de Cenas Naturais e Locomoção Autônoma em Robôs Agrícolas Móveis. Dissertação – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos. 2011.

MENDES, C. C. T. Navegação de Robôs Móveis Utilizando Visão Estéreo. Dissertação – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos. 2012.

MENG, Q., LIU, G., ZHANG, M., SI, Y., LI, M. Crop rows detection based on constraint of liner correlation coefficient. Nongye Jixie Xuebao/Transactions of the Chinese Society for Agricultural Machinery, 44 (SUPPL.1), pp. 216-223, 2013

MENG, Q., QIU R., HE J., ZHANG M., MA X., LIU G., Development of agricultural implement system based on machine vision and fuzzy control, Computers and Electronics in Agriculture, Volume 112, p. 128-138, 2015.

MOLZ, R. F. Uma Metodologia para o Desenvolvimento de Aplicações de Visão Computacional utilizando um projeto conjunto de Hardware e Software. Tese – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre. 2001.

MOUSAZADEH, H. A technical review on navigation systems of agricultural autonomous off-road vehicles. Journal of Terramechanics. 2013.

NERIS, L. O. Método de Extração da Posição de Máquinas Agrícolas por Visão Computacional Baseado em Redes Pulsadas e Ponto de Fuga. Tese – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos. 2008.

PEDRINO, E. C. Arquitetura Pipeline para Processamento Morfológico de Imagens Binárias em Tempo Real Utilizando Dispositivos de Lógica Programável Complexa. Dissertação – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos. 2003.

PEDRINO, E. C. Arquitetura Pipeline Reconfigurável Através de Instruções Geradas por Programação Genética para Processamento Morfológico de Imagens Digitais Utilizando FPGAs. Tese – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos. 2008.

SHARIFI M., CHEN X. A novel vision based row guidance approach for navigation of agricultural mobile robots in orchards. Proceedings of the 2015 6th International Conference on Automation, Robotics and Applications, art. no. 7081155, p. 251-255. 2015.

SOUSA, J. S. Sensores de Imagem Digitais CCD e CMOS In: Congresso Norte Nordeste de Pesquisa e Inovação CONNEPI, 7., 2012.

TAYLOR, S. A. CCD and CMOS imaging array technologies: Technology review. Relatório Técnico EPC-1998-106, Cambridge Laboratory, 1998.

TRIEU, D.B.K., MARUYAMA, T. Real-time color image segmentation based on mean shift algorithm using an FPGA (2015) Journal of Real-Time Image Processing, 10 (2), pp. 345-356.

TTOFIS, C., KYRKOU, C., THEOCHARIDES, T. A hardware-efficient architecture for accurate real-time disparity map estimation (2015) ACM Transactions on Embedded Computing Systems, 14 (2), p. 36.

XUE, J., ZHANG, L., GRIFT, T. Variable field-of-view machine vision based row guidance of an agricultural robot. Computers and Electronics in Agriculture, 84, pp. 85-91, 2012

ZHANG, F., YING, Y., SHEN, C., JIANG, H. e ZHANG, Q. (2005). Stereovision-based 3D Field Recognition for Automatic Guidance System of Off-road Vehicle. Proceedings of SPIE – The International Society for Optical Engineering, v 6000, 2005, Two- and Three- Dimensional Methods for Inspection and Metrology III.

APÊNDICE A – Código em VHDL do Circuito para Conversão do Espaço de Cores RGB para HSL

```
ENTITY rgb_2_hsl IS
    PORT(
    R       :in integer range 0 to 1023;   --entrada do componente R
    G       :in integer range 0 to 1023;   --entrada do componente G
    B       :in integer range 0 to 1023;   --entrada do componente B
    H       :out integer range 0 to 1023; --saída do componente H
    S       :out integer range 0 to 1023; --saída do componente S
    L     :out integer range 0 to 1023  --saída do componente L
    );
END rgb_2_hsl;
ARCHITECTURE conversor of rgb_2_hsl IS
signal max    :integer range 0 to 1023; --maior valor entre os componentes RGB
signal min    :integer range 0 to 1023; --menor valor entre os componentes RGB
signal dif   :integer range 0 to 1023; --diferença entre o maior e o menor valor RGB
begin
--Calcular o maior valor RGB
    max     <=  R when (R >= G and R >= B)     else
            G when (G >= B)        else
            B;
--Calcular o menor valor RGB
    min     <=  R when (R <= G and R <= B)     else
            G when (G <= B)        else
            B;
    dif     <= max - min;  --Calcular a diferênça entre o maior e o menor valor RGB
--Calcular o componente H
    H     <=  (G - B) * 170 / dif  when (R >= G and R >= B)  else
            340 + (B - R) * 170 / dif when (G >= B)      else
            680 + (R - G) * 170 / dif;
--Calcular o compomente S
    S   <=  0       when max = min       else
            dif * 1023 / max;
--Calcular o componente L
    L     <= (max + min)/2;
end conversor;
```

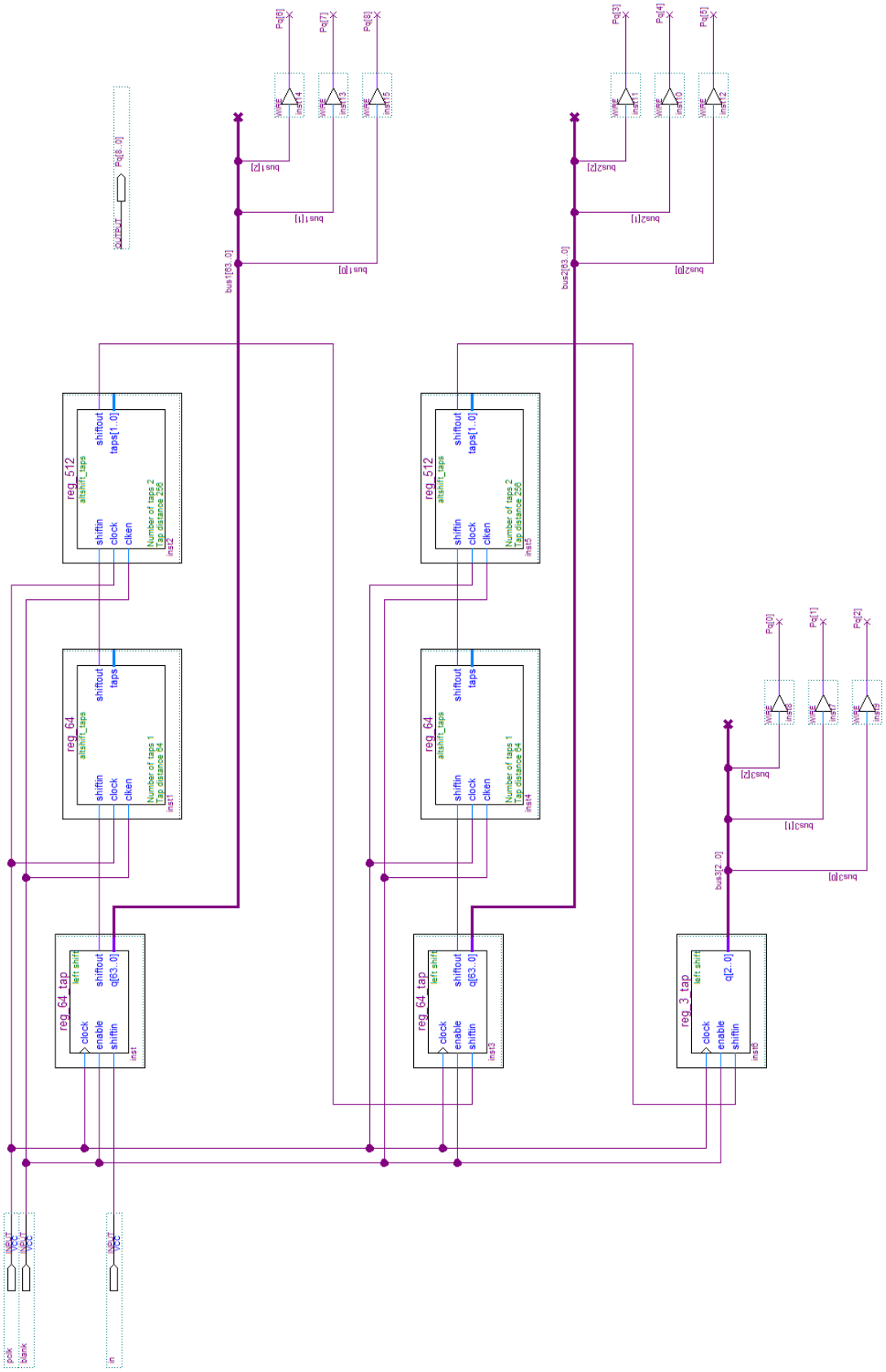
APÊNDICE B – Código em VHDL do Circuito para Limiarização dos Pixels

```
entity classificador is
    generic(
    max_h    :integer := 1023; --parâmetro limiar máximo H
    max_s    :integer := 1023; --parâmetro limiar máximo S
    max_l    :integer := 1023; --parâmetro limiar máximo L
    min_h    :integer := 0;    --parâmetro limiar mínimo H
    min_s    :integer := 0;    --parâmetro limiar mínimo S
    min_l    :integer := 0     --parâmetro limiar mínimo L
    );
    port(
    H      :in integer range 0 to 1023;  -- entrada do componente H
    S      :in integer range 0 to 1023;  -- entrada do componente S
    L      :in integer range 0 to 1023;  -- entrada do componente L
    c_out  :out bit                      --saída imagem limiarizada
    );
end classificador;
architecture classificar of classificador is
begin
--Processo de limiarização
    c_out      <='1'   when H <= max_h and H >= min_h and S <= max_s and S >=
                 min_s and I <= max_i and I >= min_i  else
          '0';
end classificar;
```

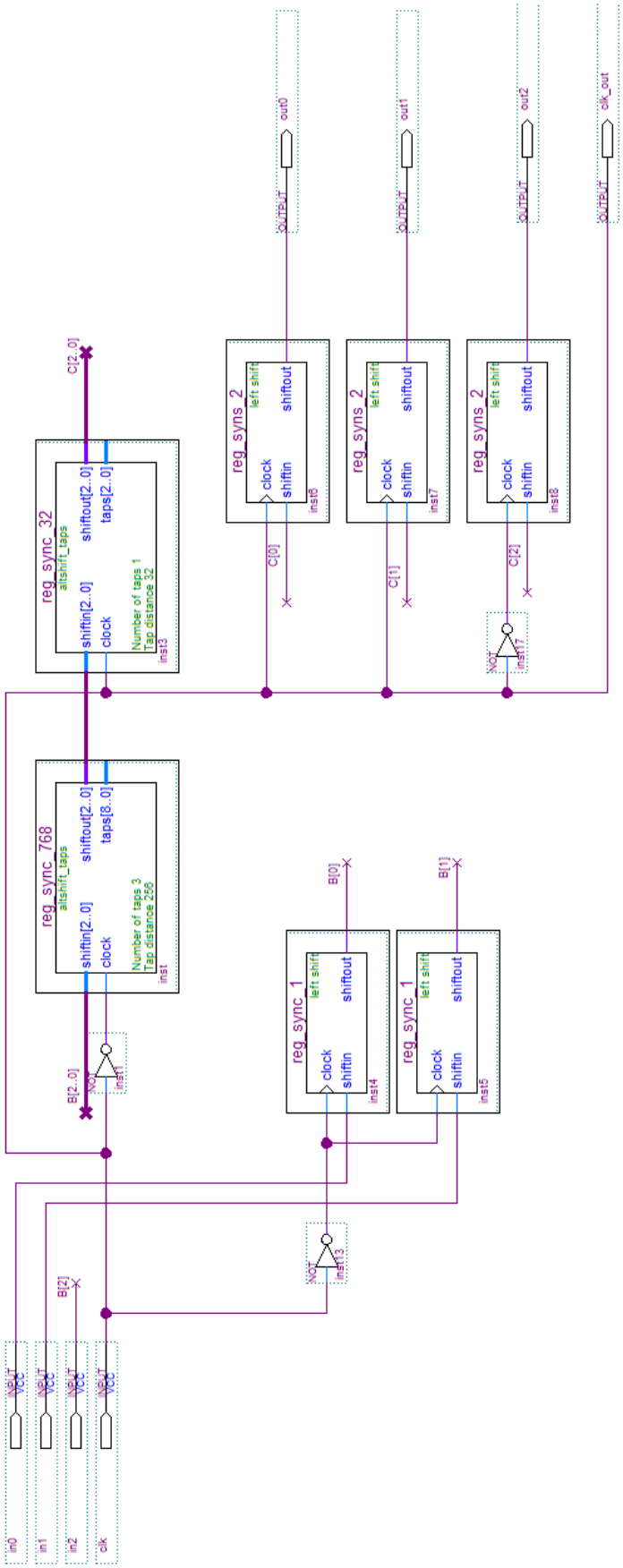
APÊNDICE C – Código em VHDL do Bloco “operador_logico”

```
ENTITY operador IS
    generic(
    Op      :bit_vector(1 downto 0)      := "00";  --parâmetro da operação morfológica
    St      :bit_vector(8 downto 0)      := "111111111"  parâmetro do elemento estruturante
    );
    PORT(
    CLK     :in bit;                     --entrada do clock
    P       :in bit_vector(8 downto 0);  --entrada da janela de pixels
    S       :out bit                     --saída do pixels processado
    );
END operador;
ARCHITECTURE operador_morf of operador IS
signal eros   :bit;  --sinal para a operação de Erosão
signal dil    :bit;  --sinal para a operação de Dilatação
signal nop    :bit;  --sinal para a operação Nulo
begin
--Calculo da operação de erosão
    eros            <= (P(0) or not St(0)) and (P(1) or not St(1)) and (P(2) or not St(2)) and
                    (P(3) or not St(3)) and (P(4) or not St(4)) and (P(5) or not St(5)) and
                    (P(6) or not St(6)) and (P(7) or not St(7)) and (P(8) or not St(8));
--Calculo da operação de dilatação
    dil             <= (P(0) and St(0)) or (P(1) and St(1)) or (P(2) and St(2)) or (P(3) and
                    St(3)) or (P(4) and St(4)) or (P(5) and St(5)) or (P(6) and St(6)) or (P(7)
                    and St(7)) or (P(8) and St(8));
--Calculo da operação Nulo
    nop     <= P(4);
--Processo para atualizar a saída em sincronia com o sinal de clock
    atualizar: process(CLK)
    begin
      if(CLK'event and CLK = '0')then
        if(Op = "01")then
          S <= eros;--saída S = resultado da operação de erosão
        elsif(Op = "10")then
          S <= dil;--saída S = resultado da operação de dilatação
        else
          S <= nop;--saída S = resultado da operação Nulo
        end if;
      end if;
    end process atualizar;
end operador_morf;
```

APÊNDICE D – Diagrama Esquemático do Bloco “memoria_de_deslocamento”



APÊNDICE E – Diagrama Esquemático do Bloco “delay_sincronismo”



APÊNDICE F – Código em VHDL do Bloco “regressão”

```
ENTITY regression IS
    generic(
--Parâmetros de configuração da área estabelecida para o process de regressão
    x0      :integer range 0 to 1023 := 0;
    x1      :integer range 0 to 1023 := 639;
    y0      :integer range 0 to 1023 := 0;
    y1      :integer range 0 to 1023 := 479
    );
PORT(
    CLKcal      :in bit;--entrada de clock do circuito
    Hin         :in bit;--sinal de sincronismo horizontal
    Vin         :in bit;--sinal de sincronismo vertical
    BLANKin     :in bit;--sinal blank
    CLKpix      :in bit;--clock dos pixels
    Sin         :in bit;--Entrada da imagem binária
    CoefAng     :out integer range -1023 to 1023;--saída do coef. angular
    CoefLin     :out integer range -1023 to 1023;--saída do coef. linear
    ClkOut      :out bit --saída de clock para o sincronismo dos coeficientes
);
END regression;
ARCHITECTURE cir_regression of regression IS
type vetor_dados is array(0 to (y1 - y0)) of integer range 0 to 1023;--vetor para inteiros
type vetor_bits is array(0 to (y1 - y0)) of bit;--vetor para binário
signal dados_x    :vetor_dados;--vetor para valores de x
signal dados_y    :vetor_bits;--vetor para os pixels (binário) em uma linha
signal flag_dados   :bit := '0';--sinal de controle
signal e_calculo    :bit := '0';--sinal de controle
signal calculando   :bit := '0';--sinal de controle
signal Hsync    :bit;--sinal de sincronismo
signal Vsync    :bit;--sinal de sincronismo
begin

    --Processo para acumular as variáveis x e y
    detector_horizontal: process(CLKpix)
    variable contador_h     :integer range 0 to 1023;--contador de colunas
    variable sum_x          :integer range 0 to 262143;--soma da coordenada x
    variable media_x        :integer range 0 to 1023;--média de x
    variable n_elementos    :integer range 0 to 1023;--número de pixels “1”
    variable contador_v     :integer range 0 to 1023;--contador de linhas
    variable BlankQP        :bit;--variável do sinal blank
    begin
      if(CLKpix'event and CLKpix = '1')then--novo pixel
        if(Hsync = '1')then--pixel válido (horizontal)
```

```
        if(BLANKin = '1')then--pixel válido (sinal blank)
          if(Contador_h >= x0 and Contador_h <= x1 and contador_v >= y0 and contador_v <
y1)then--pixel dentro da região de regressão (horizontal)
            if(Sin = '1')then--pixel = “1”
              n_elementos := n_elementos + 1;--soma o número de pixels “1”
              sum_x       := sum_x + Contador_h - x0;--soma da coordenada x
            end if;
          end if;
          if(Contador_h = x1 + 1)then--fim da área de regressão (horizontal)
            if(n_elementos = 0)then--linha vazia
              dados_y(contador_v - y0)  <= '0';--dados y vazio para linha “contador_v - y0”
              dados_x(contador_v - y0)  <= 0;--dados x vazio para linha “contador_v - y0”
            else
              media_x := (sum_x / n_elementos);--média de x
              dados_x(contador_v - y0)  <= media_x;--média para linha “contador_v - y0”
              dados_y(contador_v- y0)      <= '1';--flag para linha “contador_v - y0”
            end if;
          end if;
          if(contador_h /= 641)then--final da linha em “contador_h = 641”
            contador_h := contador_h + 1;--incremento de coluna
            end if;
            end if;
          else
            contador_h    := 0;--iniciar contador de coluna
            n_elementos     := 0;--iniciar a variável
            sum_x           := 0;--iniciar a variável
          end if;
        if(Vsync = '0')then--fim do quadro
          contador_v    := 0;--inicia contador de linhas
          flag_dados    <= '0';--iniciar flag
        elsif(BLANKin= '0' and BlankQP = '1')then--detector de borda do sinal blank
          if(contador_v = y1 + 1)then--fim da área de regressão (vertical)
            flag_dados    <= '1';
          end if;
          if(contador_v /= 480)then--última coluna em “contador_v = 480”
            contador_v    := contador_v + 1;--incremento de linha
          end if;
        end if;
        --BlankQP = BLANKin após o clock do pixel
        if(BLANKin = '1')then
          BlankQP := '1';
        else
          BlankQP := '0';
        end if;
      end if;
    end process detector_horizontal;
```

```
--Processo para sincronizar os sinais de sincronismo de video com o clock do pixel
clock_sync: process(CLKpix)
begin
  if(CLKpix'event and CLKpix = '0')then
    Hsync     <= Hin;
    Vsync     <= Vin;
  end if;
end process clock_sync;

--Processo final para o cálculo de regressão
minimo_quadrado: process(CLKcal)
variable passo              :integer range 0 to 511;--variável de controle
variable n_item     :integer range 0 to 511;--número de itens no somatório
variable soma_x             :integer range 0 to 524287;--soma de x
variable soma_y             :integer range 0 to 131071;--soma de y
variable soma_xy            :integer range 0 to 134217727;--soma de x * y
variable soma_yy            :integer range 0 to 67108863;--soma de y²
variable numerador          :integer;--buffer para numerador
variable numeradorb         :integer;--segundo buffer para numerador
variable denominador        :integer;--buffer para denominador
variable calcular           :bit := '0';--flag para iniciar o processo
  begin
    if(CLKcal'event and CLKcal = '0')then
      if(calcular = '1')then
        if((passo <= (y1 - y0)) and dados_y(passo) = '1')then--item válido
          n_item    := n_item + 1;--incremento de itens
          soma_x    := soma_x + dados_x(passo);--somatório de x
          soma_y    := soma_y + passo;--somatório de y
          soma_xy := soma_xy + dados_x(passo) * passo;--somatório de x * y
          soma_yy := soma_yy + passo * passo;--somatório de y²
        else
          case passo is
            --Cálculo parcial (numeradores e denominador)
            when y1 - y0 + 1 => if(n_item < 50)then
                denominador := n_item   * soma_yy - soma_y * soma_y;
                numerador := (n_item  * soma_xy - soma_x * soma_y);
                numeradorb  := soma_yy * soma_x - soma_xy * soma_y;
              else
                denominador := ((n_item) * (soma_yy/10))/10 - (soma_y/10) * (soma_y/10);
                numerador := ((n_item) * (soma_xy/10))/10 - (soma_x/10) * (soma_y/10);
                numeradorb  := (soma_yy/10) * (soma_x/10) - (soma_xy/10) * (soma_y/10);
              end if;
            --Cálculo dos coeficientes angular e linear
            when y1 - y0 + 2    => if(n_elementos > 1 and denominador /= 0)then
                CoefAng    <= numerador/(denominador/100);
                CoefLin     <= numeradorb/denominador;
              end if;
            when y1 - y0 + 4    => ClkOut    <= '0';
```

```
                when y1 - y0 + 6    => ClkOut    <= '1';
                  calcular := '0';
                when others         => null;
              end case;
            end if;
            passo   := passo + 1;
            calculando <= '1';
          else
            --Iniciar as variáveis
            passo      := 0;
            n_item       := 0;
            soma_x      := 0;
            soma_y      := 0;
            soma_xy    := 0;
            soma_yy    := 0;
            calculando <= '0';
            if(e_calculo = '1')then
              calcular := '1';
            end if;
          end if;
        end if;
      end process minimo_quadrado;

      --Processo auxiliar para o controle do início do processo de cálculo de regressão
      controle: process(flag_dados, calculando)
      begin
        if(calculando = '1')then
          e_calculo <= '0';
        elsif(flag_dados'event and flag_dados = '1')then
          e_calculo <= '1';
        end if;
      end process controle;
end cir_regression;
```

APÊNDICE G – Código em VHDL do Bloco “filtro”

```
ENTITY filtro IS
    PORT(
      inTANG          :in integer range -1023 to 1023;--entrada do coef. angular
      inCOEF_LINEAR :in integer range -1023 to 1023;--entrada do coef. linear
      inCLK           :in bit;--entrada de clock do circuito
      TANG            :out integer range -1023 to 1023;--saída do coef. angular
      COEF_LINEAR     :out integer range -1023 to 1023;--saída do coef. linear
      CLK             :out bit --saída de clock do circuito
    );
END filtro;
ARCHITECTURE cir_filtro of filtro IS
begin
    filtrar: process(inCLK)
    variable contador       :integer range 0 to 31; --variável para contar o número de elementos
    variable soma_tang      :integer range -32767 to 32767;--soma dos coef. angulares
    variable soma_coef      :integer range -32767 to 32767;--soma dos coef. lineares
    begin
      if(inCLK'event and inCLK = '1')then
        if(contador = 0)then
          --Iniciar a soma dos elementos
          CLK         <= '1';
          soma_tang   := inTANG;
          soma_coef   := inCOEF_LINEAR;
        elsif(contador = 19)then
          --Soma dos elementos finais
          soma_tang   := soma_tang + inTANG;
          soma_coef   := soma_coef + inCOEF_LINEAR;
          CLK         <= '0';
          --Calculo da média
          TANG        <= soma_tang/20;
          COEF_LINEAR   <= soma_coef/20;
        Else
```

```
            --Soma dos elementos
            soma_tang    := soma_tang + inTANG;
            soma_coef    := soma_coef + inCOEF_LINEAR;
            CLK          <= '0';
          end if;
          if(contador = 19)then
            contador := 0;--iniciar a variável contador
          else
            contador := contador + 1;--incremento da variável contador
          end if;
        end if;
      end process filtrar;
end cir_filtro;
```

APÊNDICE H – Código em VHDL do Bloco “linha_media”

```
ENTITY line_media IS
  generic(
--Coordenadas da área para utilizada para o cálculo da regressão linear
      x0A        :integer range 0 to 1023 := 0;
      y0A        :integer range 0 to 1023 := 0;
      X0B        :integer range 0 to 1023 := 0;
      y0B        :integer range 0 to 1023 := 0
    );
  PORT(
    CoefAngA     :in integer range -1023 to 1023;  --coef. angular lado esquerdo
    CoefLinA     :in integer range -1023 to 1023;  --coef. linear lado esquerdo
    ClkA         :in bit;                          --clock para os parâmetros do lado esquerdo
    CoefAngB     :in integer range -1023 to 1023;  --coef. angular lado direito
    CoefLinB     :in integer range -1023 to 1023;  --coef. linear lado direito
    ClkB         :in bit;                          --clock para os parâmetros do lado direito
    CoefAngC     :out integer range -1023 to 1023;--coef. angular médio
    CoefLinC     :out integer range -1023 to 1023 --coef. linear médio
  );
END line_media;
ARCHITECTURE cir_media of line_media IS
Sinais dos coeficientes angulares e lineares
signal SigCoefAngA :integer range -1023 to 1023;
signal SigCoefLinA  :integer range -1023 to 1023;
signal SigCoefAngB :integer range -1023 to 1023;
signal SigCoefLinB  :integer range -1023 to 1023;
begin
  CoefAngC     <= (SigCoefAngA + SigCoefAngB)/2;--calculo do coef. angular médio
  CoefLinC     <= (SigCoefLinA + SigCoefLinB)/2;--calculo do coef. linear médio
  rechargeA: process(ClkA)
  variable buff1    :integer range -2047 to 2047;
  variable buff2    :integer range -524287 to 524287;
  variable buff3    :integer range -8191 to 8191;
  variable buff4    :integer range -8191 to 8191;
  begin
    if(ClkA'event and ClkA = '0')then
      --Calculo parcial dos coeficientes esquerdos
      SigCoefAngA    <= CoefAngA;
      buff1        := CoefLinA + x0A;
      buff2        := y0A * SigCoefAngA;
      buff3        := buff2 / 100;
      buff4        := buff1 - buff3;
      SigCoefLinA   <= buff4;
    end if;
```

```
  end process rechargeA;

  rechargeB: process(ClkB)
  variable buff11   :integer range -2047 to 2047;
  variable buff21   :integer range -524287 to 524287;
  variable buff31   :integer range -8191 to 8191;
  variable buff41   :integer range -8191 to 8191;
  begin
    if(ClkB'event and ClkB = '0')then
      --Calculo parcial dos coeficientes direitos
      SigCoefAngB     <= CoefAngB;
      buff11          := CoefLinB + x0B;
      buff21          := y0B * SigCoefAngB;
      buff31          := buff21 / 100;
      buff41          := buff11 - buff31;
      SigCoefLinB   <= buff41;
    end if;
  end process rechargeB;
end cir_media;
```