UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"

Pós-Graduação em Ciência da Computação

Danilo Samuel Jodas

Desenvolvimento de um sistema de navegação baseado em máquina de vetores de suporte para dirigibilidade de um robô móvel por caminhos em plantações

UNESP

2012

Danilo Samuel Jodas

Desenvolvimento de um sistema de navegação baseado em máquina de vetores de suporte para dirigibilidade de um robô móvel por caminhos em plantações

Orientador: Prof. Dr. Norian Marranghello

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação – Área de Concentração em Sistemas de Computação, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

UNESP

2012

Jodas, Danilo Samuel.

Desenvolvimento de um sistema de navegação baseado em máquina de vetores de suporte para dirigibilidade de um robô móvel por caminhos em plantações / Danilo Samuel Jodas. - São José do Rio Preto : [s.n.], 2012.

101 f. : 65 il. ; 30 cm.

Orientador: Norian Marranghello

Dissertação (mestrado) - Universidade Estadual Paulista, Instituto de Biociências, Letras e Ciências Exatas

1. Robótica móvel. 2. Processamento de imagens. 3. Máquinas de vetores de suporte. I. Marranghello, Norian. II. Universidade Estadual Paulista, Instituto de Biociências, Letras e Ciências Exatas. III. Desenvolvimento de um sistema de navegação baseado em máquina de vetores de suporte para dirigibilidade de um robô móvel por caminhos em plantações.

CDU – 004.932

Ficha catalográfica elaborada pela Biblioteca do IBILCE
Campus de São José do Rio Preto - UNESP

DANILO SAMUEL JODAS

Desenvolvimento de um sistema de navegação baseado em máquina de vetores de suporte para dirigibilidade de um robô móvel por caminhos em plantações

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação – Área de Concentração em Sistemas de Computação, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

BANCA EXAMINADORA

Prof. Dr. Norian Marranghello
Professor Titular Doutor
IBILCE/UNESP – São José do Rio Preto
Orientador

Prof. Dr. Evandro Luís Linhari Rodrigues
Professor Doutor
EESC/USP – São Carlos

Prof. Dr. Alexandre César Rodrigues da Silva
Professor Titular Doutor
UNESP – Ilha Solteira

São José do Rio Preto, 02 de agosto de 2012

AGRADECIMENTOS

Agradeço aos meus pais, Antonio e Cleonice, pelo apoio oferecido durante o meu período de estudos e em todos os momentos da minha vida. Agradeço à eles também pelos ensinamentos que me ofereceram e que contribuíram para minha formação como ser humano.

Aos meus tios, Evandro, Silmara, Valdete e Carlos, que ofereceram apoio em um período difícil pelo qual passei durante os estudos.

Aos meus tios, Ademir, Ana, Cleuza e Cleide, que se fizerem presentes em um momento de angústia e tristeza pelo qual passei.

A todos os familiares que me apoiaram durante o período de estudo.

A Capes, pelo auxílio financeiro oferecido para o desenvolvimento deste trabalho.

Ao meu orientador, professor Norian Marranghello, por confiar na minha capacidade de elaborar este trabalho. Agradeço também pelos ensinamentos que contribuíram tanto para minha formação profissional, quanto para a formação como ser humano.

Ao professor Aledir Silveira Pereira, pela oportunidade oferecida de participar do programa de pós-graduação como aluno especial e também pelas contribuições oferecidas à este projeto.

Ao professor Rodrigo Capobianco Guido, pelo auxílio e sugestões oferecidas durante a elaboração deste trabalho.

Aos membros da banca examinadora, Dr. Evandro Luís Linhari Rodrigues e Dr. Alexandre César Rodrigues da Silva , por aceitarem avaliar o meu trabalho.

Aos meus companheiros de estudo, Tiago, Alexandre, Jonathan, Alessandro, Fábio e Roberta, que proporcionaram bons momentos durante esta etapa da minha vida.

Agradeço as pessoas que contribuíram, direta ou indiretamente, neste etapa da minha vida.

Sumário

Lista de figuras

Lista de tabelas

Lista de abreviaturas e siglas

ASIC: *Application Specific Integrated Circuits*

FPGA: *Field Programable Gate Arrays*

HFK: *Hardware Friendly Kernel*

HIK: *Histogram Intersection Kernel*

HSI: *Hue, Saturation, Intensity*

IEEE: *Institute of Electrical and Electronic Engineers*

MEF: Máquina de estados finitos

MLP: *Multi Layer Perceptron*

RBF: *Radial Basis Function*

RGB: *Red, Green, Blue*

RNA: Redes Neurais Artificiais

SVM: *Support Vector Machine*

VHDL: *Very High Speed Integrated Circuits Hardware Description Language*

RESUMO

A utilização de robôs móveis mostra-se importante em atividades onde a atuação do ser humano é difícil ou perigosa. A exploração em locais de difícil acesso, como por exemplo em operações de resgate e em missões espaciais, é uma situação onde robôs móveis são frequentemente utilizados para evitar exposição dos especialistas humanos a situações de riscos. Na agricultura, robôs móveis são utilizados em tarefas de cultivo e na aplicação de agrotóxicos em quantidades mínimas para reduzir a poluição do meio ambiente. Neste trabalho é apresentado o desenvolvimento de um sistema para controlar a navegação de um robô móvel autônomo por caminhos em plantações. O controle da direção do robô é realizado com base em imagens das trilhas as quais, após um processamento prévio, para extração de características, são submetidas a máquinas de vetores de suporte, para a definição da rota a ser seguida. O objetivo do projeto no qual este trabalho se insere é o controle do robô em tempo real, para tanto, o sistema foi implementado em *hardware* para mostrar que o ganho de desempenho pode ser melhor em relação à execução em *software*. Neste trabalho, relata-se a implementação de uma máquina de vetores de suporte a qual apresentou uma precisão em torno de 93% da rota adequada.

Palavras chave: Robótica móvel, processamento de imagens, máquinas de vetores de suporte

ABSTRACT

The use of mobile robots turns out to be interesting in activities where the actions of human beings is difficult or dangerous. The exploration in areas of difficult access, such as in rescue operations and in space missions, is a situation where mobile robots are often used to avoid exposure of human experts to risky situations. In agriculture, mobile robots are used in tasks of cultivation and application of pesticides in minute quantities to reduce environmental pollution. In this paper we present the development of a system to control an autonomous mobile robot navigation through tracks in plantations. Track images are used to control robot direction by preprocessing them to extract image features, and then submitting such characteristic features to a support vector machine to find out the most appropriate route. As the overall goal of the project to which this work is connected is the robot control in real time, the system was embedded onto a hardware platform to show that the performance gain can be better if compared to execution in software. However, in this paper we report the software implementation of a support vector machine, which presented around 93% accuracy in predicting the appropriate route.

Keywords: Mobile robotic, image processing, support vector machines

1 Introdução

Nos últimos anos houve um aumento significativo da utilização de robôs móveis em diversas áreas, tais como exploração espacial e operações de resgate. Esse aumento se deve à execução de atividades em locais que são de difícil acesso ou em situações que podem ocasionar riscos aos seres humanos.

O surgimento dos algoritmos inteligentes foi um fator que contribuiu significativamente para o avanço da robótica móvel devido a possibilidade de criação de agentes inteligentes que atuem de forma autônoma e confiável na execução de atividades para os quais foram projetados, sem a intervenção de um especialista humano. Grande parte dos trabalhos desenvolvidos em robótica móvel abrange a utilização de redes neurais artificiais como sistemas inteligentes, as quais determinam uma saída baseando-se nos dados de entrada do ambiente externo, sendo estes capturados por algum tipo de sensor. Esses dados de saída são utilizados para realizar algum tipo de controle do agente móvel.

A combinação de algoritmos inteligentes com visão computacional tem proporcionado bons resultados em aplicações onde robôs móveis necessitam de um mapeamento do ambiente de atuação, com o intuito de evitar colisão com outros objetos ou determinar a direção para um local. Neste caso, as imagens são capturadas por uma câmera de vídeo e são utilizadas por algoritmos de pré-processamento e segmentação para extração das características relevantes da imagem, sendo estas utilizadas como dados de entrada pelos algoritmos inteligentes para a determinação de uma saída.

O surgimento de arquiteturas de processadores com maior capacidade computacional também teve importância nesse cenário, já que a execução dessas tarefas necessitam de respostas em tempo real para garantir a viabilidade da implementação em *hardware* dos algoritmos computacionais de controle do robô móvel. Dispositivos FPGA (*Field Programmable Gate Array*) também são utilizados na implementação de robôs móveis devido à possibilidade de reconfiguração da sua estrutura interna de dispositivos lógicos, podendo isto pode melhorar significativamente o desempenho na execução das atividades do robô móvel devido à adaptação do *hardware* para um algoritmo específico.

1.1 Objetivo

Neste trabalho propõe-se um sistema de navegação baseado em algoritmos de processamento de imagens e máquinas de vetores de suporte implementado em *hardware*, para garantir a dirigibilidade de um robô móvel por caminhos de plantações. O sistema de navegação é utilizado para manter o robô na trilha da plantação e controlar sua direção baseando-se em imagens do ambiente, as quais são utilizadas por algoritmos de processamento de imagens que são aplicados para a melhoria da qualidade da imagem e na extração das características do caminho. O caminho identificado é utilizado por uma máquina de vetores de suporte para a determinação do ângulo de direção do robô móvel.

1.2 Justificativa

Soluções baseadas em sistema de *hardware* são necessárias em situações onde as baseadas em *softwares* são inviáveis ou ineficazes para determinar uma resposta em tempo adequado.

O desenvolvimento do sistema de navegação é direcionado para a agricultura. Além desta atividade, outras serão aplicadas neste cenário, tais como a detecção de pragas em plantações e controle da aplicação de agrotóxicos para erradicá-las. Portanto, o controle de navegação é um elemento essencial para o sucesso das outras atividades.

1.3 Organização do trabalho

O texto está estruturado da seguinte maneira:

- **Capítulo 2:** aborda o cenário atual da robótica móvel e os trabalhos desenvolvidos nesta área;
- **Capítulo 3:** aborda os conceitos fundamentais de processamento de imagens, as técnicas de melhoria da qualidade de imagens e os filtros de segmentação utilizados para extração de características. Além disso, aborda os fundamentos de redes neurais artificiais, tais como o modelo matemático de um neurônio biológico, as topologias utilizadas no desenvolvimento de redes neurais artificiais e as técnicas de aprendizagem para a classificação de padrões. São apresentados os conceitos

fundamentais de máquinas de vetores de suporte e as funções de *kernel* utilizadas em sua construção. Neste capítulo, também são apresentados os conceitos sobre dispositivos FPGA e linguagens de descrição de *hardware*;

- **Capítulo 4:** é apresentado o sistema de navegação desenvolvido em *software* e as etapas utilizadas para processamento da imagem e reconhecimento do ângulo de direção;
- **Capítulo 5:** são apresentados os testes realizados com máquinas de vetores de suporte, a implementação e execução do algoritmo no dispositivo FPGA e a comparação dos resultados obtidos nos testes em *software* e em *hardware.* Além disso, é apresentada uma análise de desempenho em relação ao tempo de execução do algoritmo da máquina de vetores de supote em *software* e em *hardware*;
- **Capítulo 6:** são apresentadas as considerações finais.

2 Robótica móvel

Um robô móvel é um dispositivo eletromecânico, formado de sensores e atuadores, que é disposto em um ambiente para que execute uma determinada tarefa. Os sensores são dispositivos que podem fornecer a habilidade de percepção do ambiente e obstáculos ao redor do robô, podendo ser utilizados na recuperação de informações para prover um mecanismo de ação para os dispositivos atuadores. Alguns sensores utilizados em robôs móveis são sonares e câmeras de vídeos. Já os atuadores são dispositivos que fornecem uma ação para o robô móvel e a possibilidade de interação com o ambiente. Portanto, o mecanismo de percepção-ação de um robô compreende a obtenção de informações do ambiente, o processamento dessas informações para gerar a ação para os atuadores e a aplicação dos resultados do processamento no controle dos atuadores, que pode consistir no acionamento do motor para movimentação de um braço manipulador ou movimento das rodas de um robô móvel.

A utilização de robôs móveis na realização de atividades de riscos e em locais de difícil acesso para especialistas humanos é composta de inúmeros desafios, principalmente os relacionados ao desenvolvimento de algoritmos inteligentes que controlem o agente móvel de maneira autônoma, eficiente e confiável, evitando o máximo possível a intervenção de um controlador humano. Além disso, situações imprevisíveis e difíceis de serem tratadas podem ocorrer durante a atuação do robô móvel em um determinado ambiente, como, por exemplo, quando é utilizado o mecanismo de captura de imagens por meio de uma câmera de vídeo em um terreno irregular. Essa situação envolve o estudo adequado de técnicas de processamento de imagens que diminuam o máximo possível a sobreposição de imagens causada pela trepidação do robô em sua navegação. A utilização de sensores do tipo sonar também pode causar alterações na maneira como o robô navega pelo ambiente, principalmente quando utilizados para a detecção de obstáculos, se submetidos a interferências indesejadas geradas por fatores naturais ou quando há perda de funcionalidade de um dos sensores.

O desenvolvimento de algoritmos de navegação em ambientes estáticos, os quais possuem objetos que não alteram de posição, é relativamente mais fácil quando comparado a ambientes dinâmicos ou àqueles que não possuem marcações para servir de guia para a navegação do robô móvel.

Outra questão importante que deve ser levada em consideração no desenvolvimento de robôs móveis é a otimização, pois o *hardware* utilizado pode ter limitações que restrinjam a execução de algoritmos de controle que consomem muito tempo de processamento e recursos

do *hardware*, o que pode prejudicar o desempenho na execução das tarefas, sendo inaceitável em missões críticas. Quando a navegação envolve visão computacional, a perda de desempenho pode ser significativa dependendo da quantidade de quadros que são capturados, sendo que uma taxa muito alta de leitura pode agravar a execução do algoritmo.

2.1 Trabalhos relatados

Diversos trabalhos na área de robótica móvel foram desenvolvidos (1-10), principalmente relacionados à utilização de algoritmos inteligentes para desvios de obstáculos e navegação em estradas. Na agricultura, a utilização de robôs móveis na realização de tarefas de cultivo e pulverização tem sido foco de muitas pesquisas, principalmente no controle da aplicação de produtos agrotóxicos em quantidades mínimas para reduzir a poluição do meio ambiente e os riscos relacionados à saúde humana causados pelo excesso de produtos tóxicos nos alimentos (1), (2).

No trabalho de Mandow e outros (1) é apresentado o desenvolvimento de um robô móvel denominado AURORA, o qual navega de forma autônoma nos corredores de uma estufa para realizar a pulverização de plantas. A informação necessária para a navegação do AURORA é capturada por sensores ultra-sônicos, os quais são utilizados pelo algoritmo de navegação para manter o robô alinhado aos corredores ou para determinar o ângulo de direção quando o robô atinge o final do corredor. Além disso, o robô pode ser remotamente manipulado por um especialista baseando-se em imagens fornecidas por uma câmera de vídeo, sendo que estas imagens não são utilizadas no controle de navegação. O principal objetivo deste trabalho foi minimizar os danos à saúde dos trabalhadores causados por produtos tóxicos, já que o ambiente fechado de uma estufa é um agravante devido à pouca ventilação e às altas temperaturas.

No trabalho de Åstrand e Baerveldt (2) é apresentado um robô móvel que é utilizado no combate a ervas daninhas em plantações de beterraba. Duas câmeras de vídeo são utilizadas para a detecção da trilha da plantação e distinção entre a erva daninha e a planta normal. A câmera de vídeo utilizada para detecção das trilhas é posicionada no topo do robô e realiza a captura das imagens em níveis de cinza. Um algoritmo modificado de detecção de linhas é utilizado para determinar as características do caminho, o qual é baseado nas plantações de ambos os lados. Uma segunda câmera de vídeo, posicionada perpendicularmente ao solo, é utilizada para capturar imagens das plantas. As imagens são

segmentadas e utilizadas por um algoritmo de detecção de ervas daninhas e, sendo elas localizadas, uma ferramenta mecânica de corte é utilizada para eliminá-las da plantação. Uma característica importante citada no trabalho é que todos os subsistemas do robô móvel trabalham simultaneamente. Este trabalho é um exemplo do uso de mecanização na erradicação de pragas em plantações sem o uso de agrotóxicos.

No trabalho de Van Henten e outros (3) é apresentado um robô autônomo utilizado em uma estufa com o objetivo de remover as folhas da extremidade inferior de cultivos de pepino. O robô é equipado com uma câmera de vídeo que captura as imagens para a detecção das hastes das plantas, as quais são utilizadas como informação para uma ferramenta de corte. O objetivo da remoção das folhas é evitar o surgimento de pragas nas plantas que estão em fase de crescimento.

No trabalho de Gonzales e Taha (4) é apresentado um sistema de navegação baseado em visão computacional e lógica *fuzzy*. O trabalho consiste na utilização de técnicas de processamento de imagens utilizadas para a detecção do ponto de destino do robô móvel. As imagens são capturadas por uma câmera de vídeo, a qual é colocada no topo do robô para obter uma visão aérea do ambiente, e são utilizadas na determinação da posição, obstáculos no ambiente e o destino do robô. O destino é representado por um objeto no ambiente. A imagem é pré-processada para a eliminação de ruídos e segmentada para extração do objeto de destino, sendo este representado por uma cor que o diferencie dos outros objetos. Há dois sistemas de inferência *fuzzy,* sendo cada um utilizado para determinar a velocidade das rodas direita e esquerda de maneira independente. As entradas para os sistemas de inferência *fuzzy* são: a diferença entre o ângulo de orientação do robô com o ângulo do objeto de destino e a distância entre o robô e o objeto de destino, sendo esta última informação baseada no centro do robô. As saídas dos controladores *fuzzy* são as velocidades das rodas direita e esquerda. Caso haja necessidade de um movimento para a direita, então a velocidade na roda esquerda precisa ser maior que a da direita ou o inverso se o movimento for para a esquerda. Se o movimento for em linha reta, então as saídas dos sistemas de inferência *fuzzy* serão valores iguais de velocidade para as duas rodas.

No trabalho de Achmad e Karsiti (5) é apresentado um sistema de navegação baseado em lógica *fuzzy* e visão computacional, o qual foi testado em um ambiente de simulação desenvolvido utilizando a biblioteca OpenGL, para manter um robô móvel em um corredor baseando-se na identificação de paredes. Algoritmos de processamento de imagens são utilizados para extrair as características das paredes do corredor, desprezando as regiões que

não são utilizadas. O sistema de inferência *fuzzy* é composto de 40 regras e as saídas são o ângulo de direção e a velocidade do robô.

No trabalho de Ismail e outros (6) é apresentado um algoritmo de navegação que se baseia em uma linha fixa no solo. Uma *webcam* é utilizada para a captura da imagem do solo, a qual é utilizada por um algoritmo de processamento de imagens para a extração das características da linha, sendo estas utilizadas pelo algoritmo de controle para determinar a velocidade das rodas, para movimentar o robô no ângulo apropriado ou em linha reta. O algoritmo foi projetado para tratar as variações de luminosidade da imagem, sendo que é realizada uma equalização de histograma antes da extração das características da linha para aumentar o contraste da imagem.

Pesquisas em robótica bio-inspirada também tiveram avanços, principalmente em otimizações de navegação. No trabalho de Folgheraiter e outros (7) é apresentada uma rede neural artificial para um robô móvel bio-inspirado, a qual é utilizada para determinar as velocidades das rodas para auxiliar a navegação do robô móvel até uma fonte de som. A rede neural é formada por duas camadas, sendo que a primeira camada recebe as entradas dos sensores e a segunda camada determina as saídas para os motores de controle das rodas. Há sete sensores instalados no robô móvel, sendo dois sensores de contato, dois sensores de captura de sons, dois sensores de detecção das plataformas de recarga de energia e um sensor indicando o nível de energia. Dependendo das entradas apresentadas à primeira camada, a saída para a camada seguinte pode ser inibida ou estimulada. Por exemplo, os sensores de contato são utilizados para detectar obstáculos à esquerda e à direita do robô móvel. Quando o sensor de contato à direita detecta um obstáculo, a saída para o primeiro neurônio da camada seguinte será estimulada, sendo inibitória no segundo neurônio. A saída do primeiro neurônio controla a velocidade da roda direita e a saída do segundo neurônio controla a velocidade da roda esquerda. Portanto, quando o primeiro neurônio é estimulado, a velocidade da roda direita aumenta, fazendo com que o robô dirija-se para a esquerda. O funcionamento dos neurônios que recebem as informações dos sensores de som é análogo ao dos sensores de contato, ocasionando estímulo ou inibição aos neurônios da segunda camada para determinar a direção até a fonte de som. Os sensores de energia são utilizados para detectar uma fonte de energia disponível para recarregar a bateria do robô móvel, sendo que a rede neural artificial tem o comportamento análogo aos casos anteriores. O sensor de nível de energia é utilizado para controlar as atividades de navegação para as fonte de som e energia, sendo que ambas não podem ocorrer simultaneamente.

No trabalho de Li, Jiang e Wang (8) é apresentado um algoritmo de navegação de um veículo baseado nas marcações das estradas. A técnica de extração das características das marcações laterais e centrais da estrada é baseada em lógica *fuzzy*. A diferença de iluminação das regiões da imagem é essencial para o correto funcionamento do algoritmo, sendo que se um pixel pertence à uma região uniforme, então este se torna branco, caso contrário torna-se preto. Há também algumas regras no sistema de inferência *fuzzy* que são utilizadas para o tratamento da alta incidência de iluminação solar. Após esse processamento, a imagem é binarizada e utilizada pelos algoritmos de navegação para determinar o ângulo de direção do veículo.

A exploração espacial representa um grande desafio para a pesquisa em robótica móvel devido à incapacidade de recuperação dos robôs em caso de falha, sendo que tal situação representa perdas altíssimas de investimentos. Portanto, a aplicação de sistemas inteligentes nessas missões é essencial para garantir um grau máximo de autonomia e capacidade de tratar situações inesperadas as quais tem extremo potencial de ocorrer principalmente em ambientes espaciais, já que o conhecimento do terreno de outros planetas pode ser incerto. No trabalho de Howard e Seraji (9) são relatadas diversas características que devem ser levadas em consideração no desenvolvimento de algoritmos de navegação para robôs móveis utilizados em missões espaciais, tais como análise de irregularidades no terreno, detecção de declives, descontinuidades e análise da textura do solo por onde o robô navegará. Em seu trabalho, os algoritmos desenvolvidos utilizam imagens capturadas por câmeras de vídeo e são baseados em lógica *fuzzy* e redes neurais artificiais para a análise das características citadas. A avaliação da irregularidade do solo consiste na identificação de rochas e na distância entre elas, a qual determina se há possibilidade de passagem do robô. A análise da descontinuidade do terreno permite identificar penhascos, valas ou pequenas crateras. A análise da textura do solo é relevante para determinar riscos que podem ocorrer na navegação. Solos com uma quantidade grande de areia, por exemplo, podem fazer com que o robô fique preso. Quando há uma quantidade grande de cascalho no solo, o risco das rodas do robô deslizarem é alto. Os dados de análise dessas características são essenciais para avaliar o nível de segurança de navegação pelo terreno, o qual é resultante de um sistema de inferência *fuzzy*. Os algoritmos foram testados no robô móvel *Pionner AT,* o qual é composto por sete sensores do tipo sonar e seis câmeras de vídeo posicionadas no topo do robô. O robô móvel pode ser visto na figura 2.1.

Figura 2.1: Robô Pionner AT (9)

Young-Jae Ryoo (10) apresentou um sistema de navegação visual para robôs móveis baseado em processamento de imagem e rede neural artificial. O sistema recebe dados de oito pontos das linhas da estrada, os quais são passados para uma rede neural artificial que apresenta o ângulo de direção do robô móvel. As entradas para a rede neural artificial são oito pontos que cruzam horizontalmente com as bordas da estrada, conforme mostrado na figura 2.2. O algoritmo de aprendizado utilizado no trabalho foi o de retropropagação (*backpropagation*).

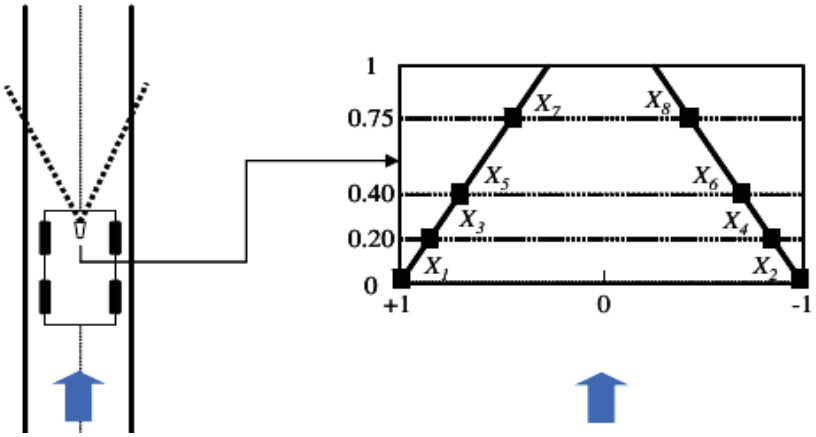


Figura 2.2: Representação dos oito pontos extraídos das bordas da estrada

3 Fundamentação teórica

O objetivo deste capítulo é abordar os fundamentos de imagens digitais e as técnicas utilizadas para a melhoria de sua qualidade e extração de características que foram utilizadas no desenvolvimento do sistema de navegação proposto. Além disso, será abordado uma fundamentação teórica sobre redes neurais artificiais e máquinas de vetores de suporte.

A utilização de sistemas de visão computacional na área de robótica móvel têm possibilitado avanços em pesquisas que têm o objetivo de desenvolver algoritmos de controle para robôs móveis os quais resultem em maior autonomia e execução mais confiável das atividades. A utilização de sensores do tipo sonar oferece uma boa navegação do robô móvel sem a necessidade do conhecimento do ambiente. No entanto, há atividades onde é necessário o reconhecimento de objetos do terreno em que o robô móvel se situa, como por exemplo a determinação de estradas ou trilhas para permitir a sua navegação. A realização de um mapeamento do ambiente permite, além de um conhecimento mais preciso do local, a aplicação de algoritmos de reconhecimento de padrões para a extração de características relevantes do terreno para determinar as ações do sistema de controle. No entanto, a irregularidade do terreno e a iluminação do ambiente podem prejudicar a qualidade das imagens capturadas e tornar o sistema de controle ineficiente. As técnicas de processamento de imagens têm grande relevância em sistemas de visão computacional devido à possibilidade de se melhorar a qualidade da imagem e do reconhecimento de padrões, para percepção de máquinas (11). A melhoria da qualidade da imagem permite uma melhor identificação da sua representação, sendo esta etapa importante para o sucesso das técnicas de reconhecimento de padrões.

O reconhecimento de padrões é uma etapa importante na determinação dos objetos presentes na imagem e o seu significado, podendo nesta etapa serem utilizadas técnicas de inteligência artificial. A inteligência artificial é uma representação da cognição humana aplicada à computação para a solução de problemas. Segundo Fernandes (12), a palavra inteligência vem do latim *inter* (entre) e *legere* (escolher) e representa a ação do ser humano de escolher entre uma coisa e outra. Segundo o mesmo autor, a manifestação inteligente é composta pela aquisição, pelo armazenamento e pela inferência de conhecimento. Portanto, a inteligência artificial é a implementação computacional da aproximação da ação de escolha do ser humano. O termo aproximação é utilizado para retratar o não conhecimento completo sobre as funções de conhecimento e aprendizado do cérebro humano, sendo os métodos de

inteligência artificial existentes, principalmente os pertencentes à abordagem conexionista, modelos matemáticos que simulam a atividade cognitiva. São muitas as aplicações que utilizam inteligência artificial, sendo que há grande esforço de sua utilização no desenvolvimento de robôs que desempenhem determinadas tarefas de maneira eficiente e que tenham a habilidade de adquirir e representar o conhecimento para permitir maior autonomia em suas atividades.

3.1 Processamento de imagens

Uma imagem é formada mediante a incidência de uma fonte de iluminação sobre um objeto, o qual reflete parte da luz incidida que é capturada por um sensor sensível ao espectro de luz refletido. Os componentes que compõem uma imagem f(x,y) são a iluminação e a reflectância. A iluminação, descrita como i(x,y), refere-se à quantidade luz que incide sobre o objeto, sendo a reflectância, descrita por r(x,y), o componente que corresponde a quantidade de luz refletida pelo objeto (11). O produto de i(x,y) por r(x,y) forma a imagem f(x,y) e é representado pela equação 3.1.

$$f(x,y)=i(x,y)\cdot r(x,y) \qquad 3.1$$

Uma imagem digital é uma função f(x,y), sendo (x,y) as coordenadas espaciais da imagem e f(x,y) a intensidade do brilho da imagem nesta coordenada.

Uma imagem digital é digitalizada tanto espacialmente quanto em amplitude. A representação da imagem em coordenadas espaciais (x,y) é denominada amostragem da imagem e é representada por uma matriz de duas dimensões MxN, onde os índices de linha e coluna representam a posição de um ponto da imagem. Este ponto também é denominado como pixel. A digitalização em amplitude é denominada de quantização dos níveis de cinza da imagem e é representada pela variação dos níveis de cinza da imagem, os quais são aplicados nas coordenadas (x,y) da matriz da imagem. Em uma imagem de 8 bits, a variação da intensidade de cinza é representada digitalmente por valores no intervalo de 0 à 255, sendo que o valor 0 representa a cor preta, o valor 255 representa a cor branca e os valores intermediários representam as variações de cinza. Cada ponto da matriz MxN possui um valor neste intervalo e o resultado é uma imagem formada por níveis de cinza. Na figura 3.1 é mostrada uma representação matricial de uma imagem, onde cada elemento representa o valor de cinza na coordenada (x,y), e na figura 3.2 é possível visualizar uma imagem digital em nível de cinza e sua representação matricial.

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ & & \dots & \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$

Figura 3.1: Representação matricial de uma imagem (11)

243	252	253	247	215	202	201	204	210	202
248	252	254	240	208	199	199	204	209	200
250	251	255	229	202	200	196	202	207	203
251	253	252	222	199	203	199	199	205	201
246	238	233	214	201	201	201	201	204	197
228	208	202	203	201	196	199	207	207	197
211	206	204	198	184	197	204	208	209	197
197	200	196	184	189	199	201	209	298	201
190	192	194	185	190	187	192	211	208	195
175	184	192	190	187	183	193	198	192	193

Figura 3.2: Imagem digital e sua representação matricial

3.1.1 Etapas do processamento de imagens

O processamento de imagens envolve cinco etapas:

- Aquisição da imagem
- Pré-processamento
- Segmentação
- Representação e descrição
- Reconhecimento e interpretação

Segundo Gonzalez e Woods (11), a aquisição da imagem é realizada por um equipamento físico sensível à uma banda do espectro de energia eletromagnética, tais como raios X, ultravioleta, visível ou banda infravermelha, e que produza um sinal elétrico de saída proporcional a um nível de energia percebida. Câmeras e *scanners* são equipamentos típicos utilizados na aquisição de imagens.

Após a aquisição da imagem, vem a etapa de pré-processamento. Esta etapa é importante devido aos vários fatores que podem prejudicar a qualidade da imagem capturada, tais como a baixa luminosidade do ambiente ou o tipo de equipamento e resolução utilizados no momento da aquisição. O pré-processamento consiste na utilização de algoritmos para a

melhoria da qualidade da imagem, tais como aumento do brilho ou contraste e remoção de ruídos, sendo estas operações importantes para determinar o sucesso das etapas seguintes.

A etapa de segmentação consiste em dividir a imagem em suas partes ou objetos constituintes. A segmentação de uma imagem utilizada em algoritmos de navegação de robôs móveis, por exemplo, consiste em extrair as características que representam as estradas ou trilhas. Além disso, a extração dos objetos contidos dentro da trilha é importante para a tarefa de desvio de obstáculos.

A etapa de representação e descrição utiliza os dados da imagem segmentada e consiste em representá-la em um formato adequado para o processamento seguinte. O resultado desta etapa pode ser uma imagem representada apenas por suas fronteiras ou pela região interna. A representação das fronteiras é útil em operações que determinam as formas da imagem e a representação da região interna é utilizada para determinação da textura. A descrição consiste em caracterizar a imagem representada em um dos padrões anteriores. Uma imagem representada por uma fronteira pode ser descrita por características como tamanho, orientação da linha ou o número de concavidades na fronteira (11).

A última etapa do processamento de imagens é o reconhecimento e a interpretação. O reconhecimento baseia-se nos descritores da imagem para classificá-la em um determinado grupo. A interpretação é utilizada para dar um significado ao objeto reconhecido. Por exemplo, o reconhecimento de um penhasco utiliza os descritores gerados na etapa de representação e descrição. Já a interpretação consiste em determinar o grau de profundidade deste penhasco. Algoritmos de inteligência artificial, tais como redes neurais artificiais, podem ser utilizados nesta etapa.

3.1.2 Métodos de realce de imagens

As técnicas de realce de imagens são utilizadas para alterar a representação original da imagem com o objetivo de melhorar sua qualidade e torná-la adequada para processamento nas etapas seguintes. A escolha e combinação adequadas das técnicas de realce são essenciais para o sucesso das etapas seguintes do processamento da imagem.

Há dois tipos de processamento de realce de imagens: métodos de realce no domínio espacial e métodos de realce no domínio da freqüência. O primeiro refere-se ao conjunto de *pixels* com seus respectivos valores de níveis de cinza, enquanto que o segundo refere-se ao sinais da imagem no domínio da transformada de Fourier (11).

No domínio espacial, representa-se a imagem como um agregado de *pixels*. Os algoritmos utilizados nesta representação atuam diretamente sobre esses *pixels*. As operações de realce da imagem podem ser realizadas sobre um único pixel ou sobre sua vizinhança. No primeiro caso, a operação é realizada pixel à pixel, independentemente dos *pixels* vizinhos, e no segundo caso um pixel receberá o valor da operação sobre os valores de cinza de todos os *pixels* que estão em sua vizinhança. As operações de realce no domínio espacial são representadas conforme a equação 3.2.

$$g(x, y) = T[f(x, y)] \qquad 3.2$$

Na equação 3.2, f(x,y) é valor do nível de cinza da imagem na coordenada (x,y), T representa um operador aplicado sobre esse nível de cinza e g(x,y) é valor do nível de cinza resultante da operação. Técnicas de realce que envolvem vizinhança de *pixels* utilizam operadores denominados filtros espaciais ou máscaras, as quais são pequenas matrizes convoluídas sobre a imagem. Neste caso, os valores de níveis de cinza da vizinhança de um pixel (x,y) da imagem são utilizados para determinar o novo nível de cinza do pixel (x,y). A matriz é então deslocada para o próximo pixel (x,y) da imagem de entrada e a operação é repetida até o final da imagem. Na figura 3.3 é ilustrada uma representação de uma máscara de ordem ímpar sobre a matriz da imagem.

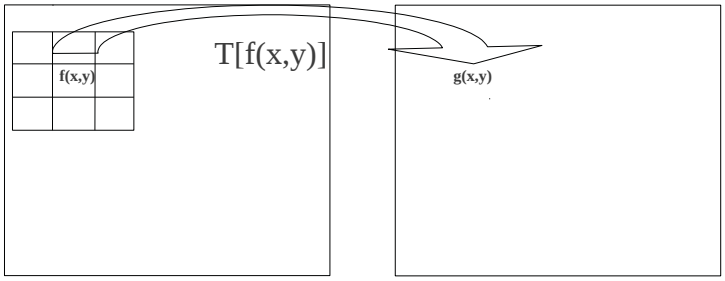


Figura 3.3: Representação de uma máscara

3.1.2.1 Convolução com máscaras

Seja I a matriz completa de uma imagem e M uma matriz de dimensões menores do que I. A convolução entre I e M é descrita como I * M e consiste na soma dos produtos dos valores de I e M.

A matriz M é denominada máscara ou janela e é sobreposta sobre uma região da matriz I da imagem para que cada elemento de M seja multiplicado pelo valor correspondente

de I dentro da máscara, obtendo assim uma soma ponderada. A máscara é então deslocada horizontal e verticalmente, da esquerda para a direita, até que o último elemento de I seja processado. O resultado será armazenado em uma matriz R com as mesmas dimensões de I (13). Portanto, a convolução basicamente consiste na multiplicação, soma e deslocamento da máscara sobre a imagem e é uma abordagem utilizada para calcular o valor de um pixel baseando-se nos valores de cinza dos seus vizinhos. Na figura 3.4 são ilustradas algumas máscaras utilizadas em convoluções.

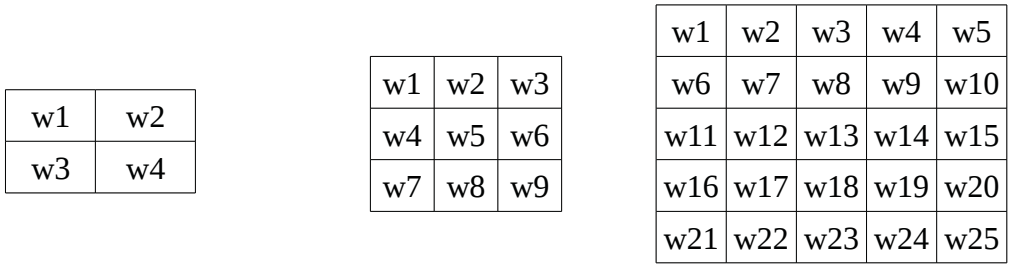


Figura 3.4: Máscaras utilizadas em convoluções (11)

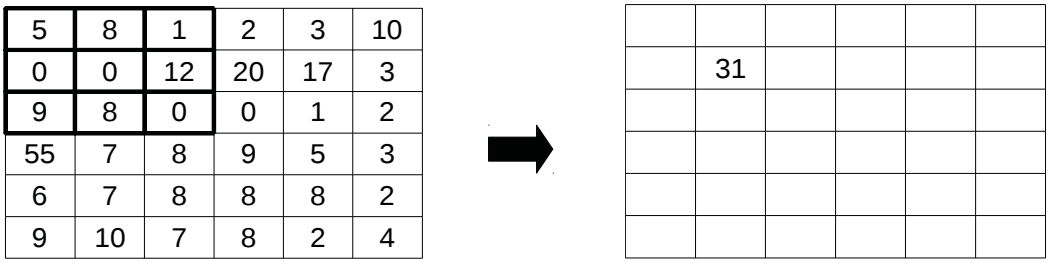
A convolução é representada pela equação 3.3.

$$R = \sum_{i=1}^{k} w_i * z_i \qquad 3.3$$

Na equação 3.3, k é o número de *pixels* dentro da máscara e R é a matriz resultante da convolução. O resultado da convolução é aplicado a um pixel de R, o qual depende da máscara sendo utilizada na operação. Para máscaras com dimensões ímpares, o resultado da convolução é aplicado na matriz R na posição onde ficou sobreposto o pixel central da máscara na matriz de origem. Em máscaras de dimensões pares, o resultado é aplicado na matriz R na posição onde ficou sobreposto o primeiro pixel da máscara na matriz de origem. Na figura 3.5 é ilustrado o procedimento de convolução usando uma máscara de dimensão ímpar. Na figura 3.5 (a) é apresentada uma máscara de dimensões 3x3 e na figura 3.5 (b) é ilustrado o procedimento de convolução desta máscara sobre os *pixels* vizinhos do pixel (1,1) e o resultado sendo armazenado no pixel (1,1) da matriz resultante. Após isso, a máscara é deslocada para a direita e o pixel (1,2) é o novo centro da máscara. Esse processo é realizado até o pixel (4,4), sendo o resultado final apresentado na figura 3.5 (c).

1	1	1
0	0	0
1	1	1

a)



f(1,1) = 1 x 5 + 1 x 8 + 1 x 1 + 0 x 0 + 0 x 0 + 0 x 12 + 1 x 9 + 1 x 8 + 1 x 0 = 31

b)

	31	19	7	18	
	82	56	71	57	
	38	31	25	21	
	96	49	39	31	

c)

Figura 3.5: a) máscara b) imagem com a máscara sobreposta sobre o pixel central (1,1)
c) resultado da convolução

No exemplo da figura 3.5 (c), os *pixels* da borda foram desconsiderados devido a falta de elementos que necessitam estar presentes para completar a sobreposição da máscara. Portanto, os *pixels* próximos a borda necessitam de um tratamento especial e as seguintes estratégias podem ser utilizadas para solucionar o problema:

1. Atribuir o valor zero aos *pixels* da borda da imagem resultante, os quais não foram possíveis de serem calculados;
2. Manter os mesmos valores da imagem original nos *pixels* da borda da imagem resultante;
3. Utilizar apenas os *pixels* da imagem original que foram sobrepostos pela máscara.

Na figura 3.6 são apresentados os resultados das três estratégias citadas.

0	0	0	0	0	0
0	31	19	7	18	0
0	82	56	71	57	0
0	38	31	25	21	0
0	96	49	39	31	0
0	0	0	0	0	0

a)

5	8	1	2	3	10
0	31	19	7	18	3
9	82	56	71	57	2
55	38	31	25	21	3
6	96	49	39	31	2
9	10	7	8	2	4

b)

0	12	32	49	40	20
20	31	19	7	18	16
62	82	56	71	57	28
30	38	31	25	21	13
81	96	49	39	31	14
13	21	23	24	18	10

c)

Figura 3.6: a) resultado da primeira estratégia b) resultado da segunda estratégia c) resultado da terceira estratégia

3.1.2.2 Filtragem espacial

A filtragem espacial consiste em realçar uma imagem utilizando o conceito de convolução com máscaras. As técnicas pertencentes à esta categoria são as filtragens passa baixas, passa altas e mediana. O uso de filtros espaciais possibilita a definição de uma vizinhança de *pixels* que são utilizados para obter o novo nível de cinza de um pixel (x,y) contido na imagem.

As filtragens passa baixas e mediana são técnicas de suavização de imagens e a aplicação destes filtros provoca o borramento da imagem e a redução dos ruídos. A filtragem passa altas é uma técnica de aguçamento, a qual consiste em enfatizar detalhes finos da imagem ou realçar regiões suavizadas (11).

A filtragem passa baixas é utilizada para minimizar ou remover da imagem os elementos de alta freqüência, deixando inalteradas as baixas freqüências. Os componentes de alta freqüência são caracterizados por bordas e detalhes finos da imagem, de modo que o resultado desta filtragem é o “borramento” da imagem (11). A aplicação deste filtro consiste na soma dos produtos entre os coeficientes da máscara, sendo estes positivos, e os níveis de cinza da imagem contidos na máscara, assim como apresentado na equação 3.3. Entretanto, o uso desta equação pode resultar em um valor de R que não esteja no intervalo de níveis de

cinza válido para a imagem. Neste caso, é realizada uma normalização dividindo-se o valor de R pelo número de *pixels* com valores positivos da máscara, conforme a equação 3.4.

$$R = \frac{1}{N} * \sum_{i=1}^{k} wi * zi \qquad 3.4$$

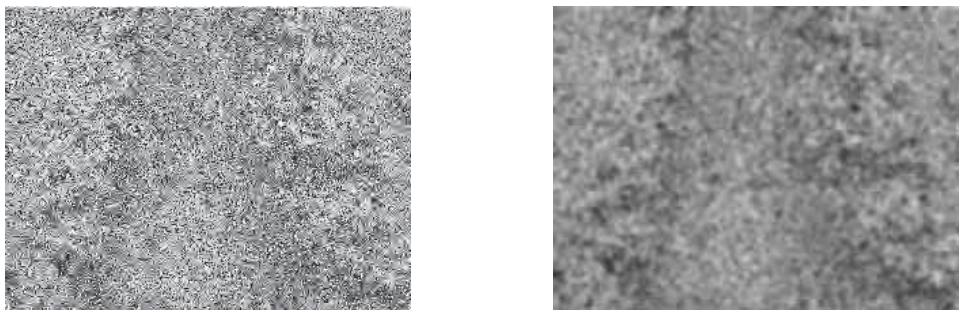
O filtro espacial passa baixas pode ser implementado por matrizes quadradas, sendo as dimensões 3x3, 5x5 ou 7x7 as mais utilizadas. Entretanto, matrizes com dimensões pares também podem ser utilizadas. Na figura 3.7 são exibidas três matrizes frequentemente utilizadas na filtragem espacial. A utilização desse tipo de máscara resulta na média dos níveis de cinza vizinhos a um pixel central (x,y), sendo o resultado aplicado ao pixel central (x,y).

$$\frac{1}{9} x \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \frac{1}{25} x \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad \frac{1}{49} x \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

a) b) c)

Figura 3.7: Filtros espaciais 3x3, 5x5 e 7x7, respectivamente (11)

Na figura 3.8 (b) é exibido o resultado da filtragem passa baixas de média da vizinhança em uma imagem em nível de cinza utilizando uma máscara 5x5.



a) b)

Figura 3.8: a) Imagem com ruído b) imagem após aplicação da filtragem passa baixas de média da vizinhança

A filtragem mediana, considerada também uma técnica de suavização, é utilizada para eliminar ruídos “granulados” da imagem. Na filtragem mediana, o valor de um determinado pixel é substituído pelo valor mediano dos níveis de cinza dos *pixels* vizinhos, ao contrário da filtragem passa baixas onde o pixel recebe a soma ponderada dos elementos da máscara e dos níveis de cinza dos *pixels* da imagem sobrepostos pela máscara. O valor mediano *m* de um

conjunto de valores é tal que metade dos valores do conjunto são menores do que *m* e a outra metade dos valores são maiores do que *m*. A implementação da filtragem mediana é feita com a seleção dos valores do pixel central (x,y) e de seus vizinhos, a determinação do valor mediano e a aplicação do resultado ao pixel central (x,y) (11). Uma matriz de dimensões 3x3, 5x5 ou 7x7 é utilizada como filtro espacial para esta operação, sendo que esta consiste em selecionar os valores de nível de cinza dos *pixels* contidos no filtro. Os valores extraídos são ordenados de maneira crescente e o valor mediano é aplicado na matriz resultante na posição onde ficou sobreposto o pixel central da máscara na matriz de origem. Em um filtro espacial 3x3, por exemplo, são extraídos nove elementos, sendo o quinto o valor mediano. Isso permite que valores de *pixels* que representam ruídos isolados da imagem se assemelhem a seus vizinhos, efetivamente eliminando os ruídos granulados contidos isoladamente na área do filtro espacial (11). Na figura 3.9 é ilustrado o funcionamento da filtragem mediana.

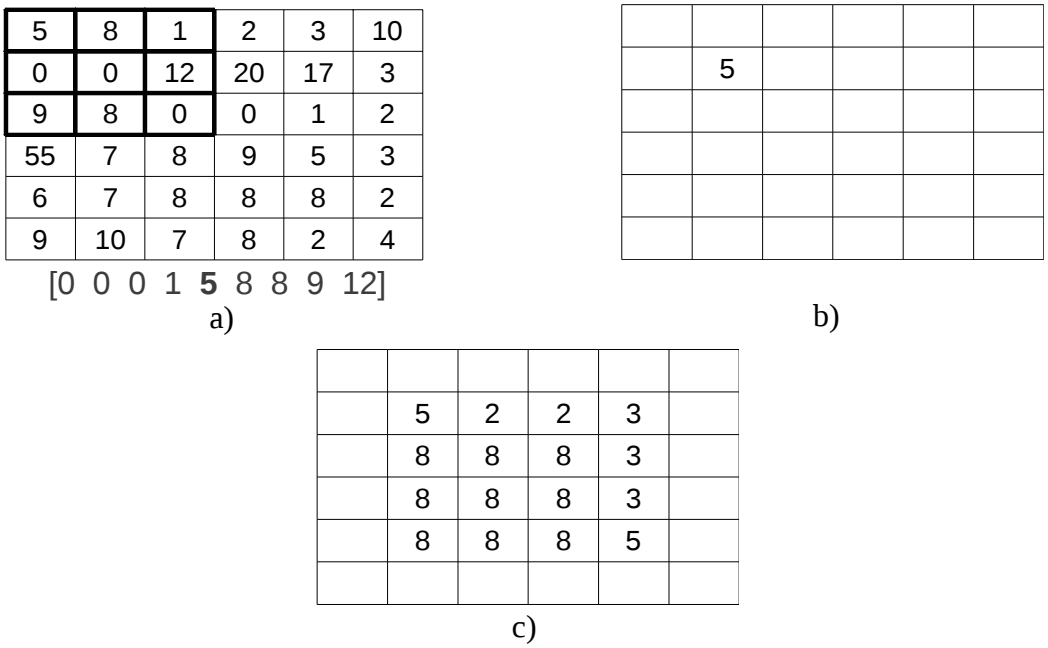
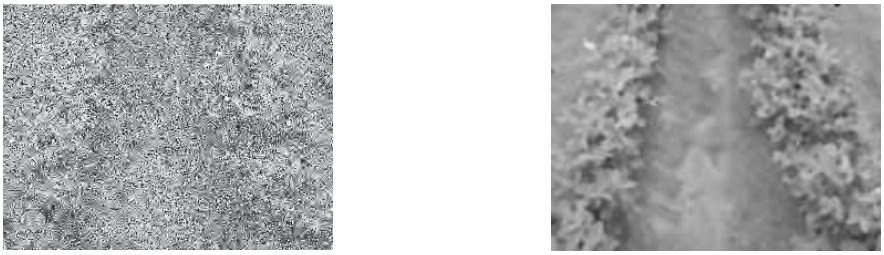


Figura 3.9: Representação da filtragem mediana a) imagem b) resultado da filtragem mediana para um pixel c) resultado da filtragem para a imagem

Na figura 3.10 é ilustrado a aplicação da filtragem mediana sobre uma imagem em níveis de cinza submetida a ruídos.



a) b)

Figura 3.10: a) Imagem submetida a ruído b) Imagem processada com a filtragem mediana utilizando uma máscara 5x5

3.1.3 Processamento de imagens coloridas

Segundo Marques Filho e Vieira Neto (13), o uso de cores em processamento de imagens decorre de dois fatores motivantes:

- No reconhecimento de padrões, a cor é um poderoso descritor de propriedades que pode simplificar sua identificação e segmentação;
- Na análise de imagens com intervenção humana, o olho humano pode discernir variações de cores de diferentes matizes e intensidades, enquanto sua capacidade de distinguir diferentes tons de cinza não passa de algumas poucas dezenas de tons diferentes.

Em 1666, Isaac Newton descobriu que quando um feixe de luz branca atravessa um prisma de vidro, este produz um novo conjunto de cores que variam do vermelho até violeta. O espectro de luz é medido em comprimentos de onda, sendo que a luz visível a olho nu ocupa uma faixa pequena a qual é composta pelas seguinte cores: vermelha, laranja, amarela, verde, azul e violeta. Na figura 3.11 é ilustrado o conjunto de espectros de luz e seus comprimentos de onda.

A percepção de cores no espectro de luz visível acontece quando o olho humano recebe uma energia eletromagnética. A visualização de um objeto é feita pelo luz que ele reflete (13). A luz incidente em um objeto é refletida em um intervalo de comprimento de onda do espectro de luz, sendo que se ela for refletida de maneira balanceada em todos os comprimento de onda de luz visível é vista como luz branca pelo observador. Entretanto, um corpo que reflete a luz em um intervalo limitado de comprimento de onda exibirá alguns tons de cores. Por exemplo, objetos com cor verde refletem a luz com comprimentos de onda no intervalo de 500 a 570 nm, enquanto absorve a maior parte da energia dos outros comprimentos (11).

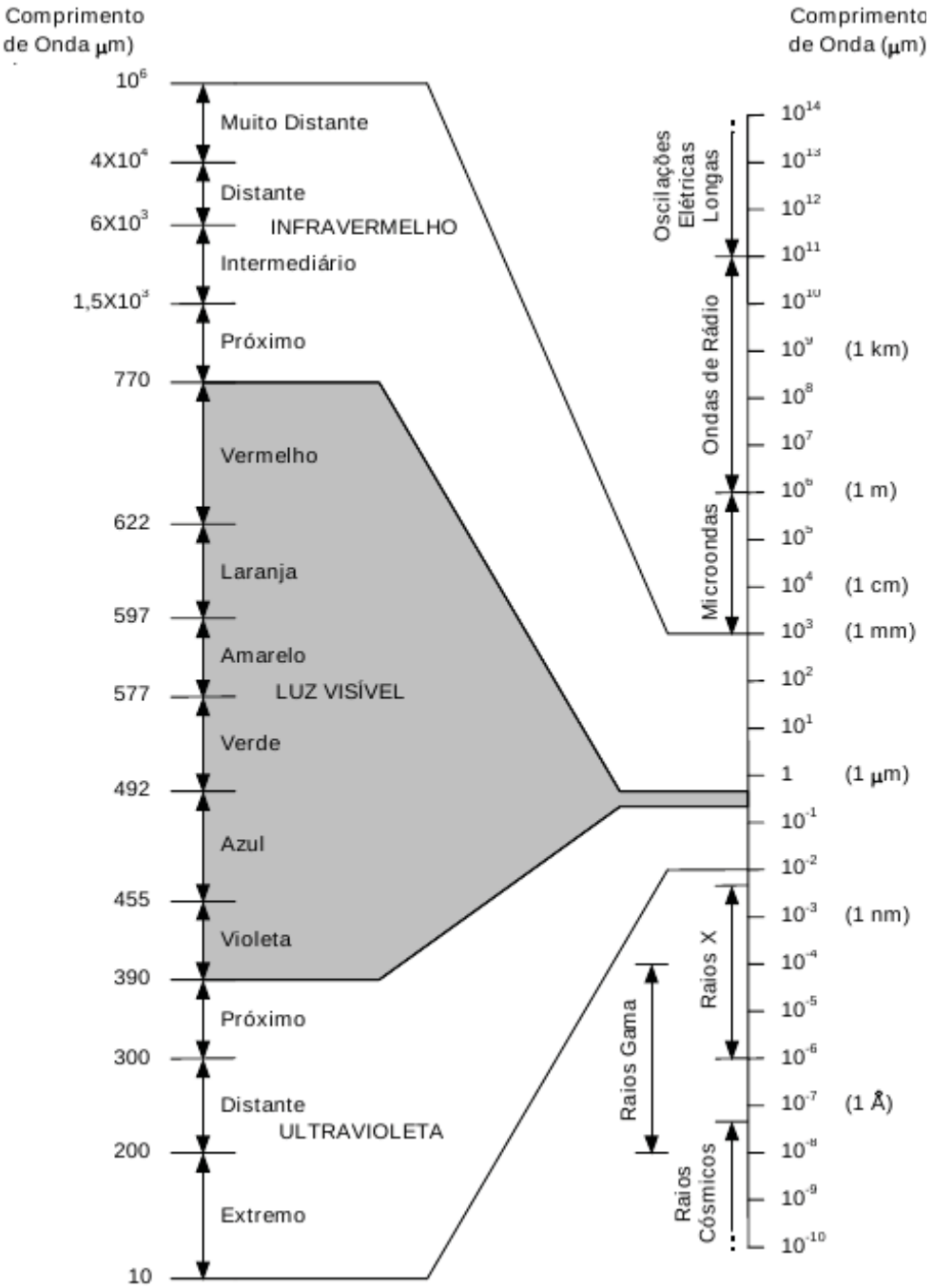


Figura 3.11: Espectros de luz e seus comprimentos de onda (13)

A formação de cores é basicamente realizada por meio da combinação de três cores primárias: vermelha, verde e azul. Essas três cores são denominadas cores primárias aditivas, pois para se obter uma cor é necessária a combinação aditiva de uma ou mais delas, em diferentes proporções. A combinação das cores primárias, duas a duas, produz as chamadas cores secundárias, sendo elas: Ciano (Verde + Azul), Magenta (Vermelha + Azul) e Amarela (Vermelha + Verde). A combinação das três cores primárias ou de uma secundária com sua primária oposta produz a cor branca. As cores secundárias são denominadas subtrativas, pois a

combinação das cores subtrai a luz branca incidente, refletindo apenas a cor correspondente ao pigmento (13). A combinação das cores secundárias, duas a duas, produz uma cor primária, sendo elas: Vermelha (Magenta + Amarela), Verde (Ciano + Amarela) e Azul (Ciano + Magenta).

As características utilizadas para distinguir uma cor da outra são: brilho, matiz e saturação. O brilho é o componente proveniente da intensidade luminosa incidente sobre o objeto. O matiz é o componente que descreve a cor pura da imagem. A saturação corresponde a quantidade de luz branca combinada com a cor pura. Por exemplo, as cores rosa e vermelha apresentam o mesmo matiz, mas diferentes graus de saturação (13).

3.1.3.1 Modelo de cor RGB

No modelo de cor RGB (*Red, Green* e *Blue*) as cores são formadas mediante a combinação das cores primárias vermelha, verde e azul. A representação do modelo RGB é feita em um sistema de coordenadas cartesiano R^3, onde cada eixo representa uma cor primária e a origem deste plano representa a cor preta. Na figura 3.12 é ilustrada a representação do modelo de cor RGB.

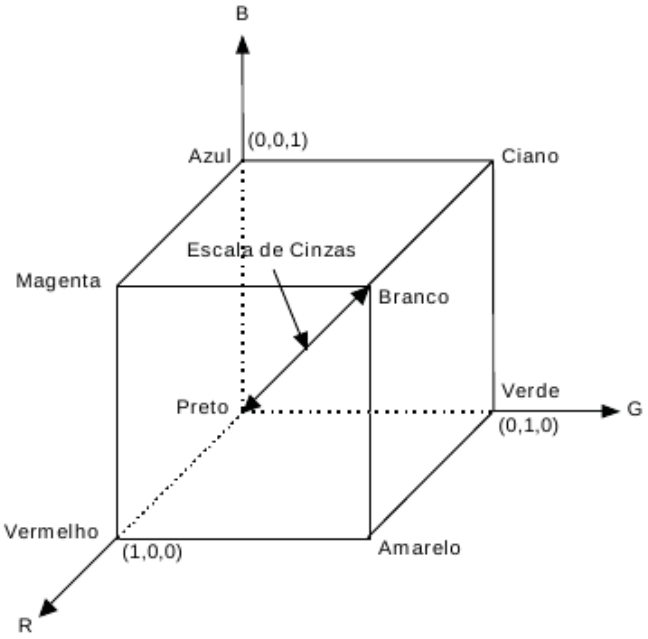


Figura 3.12: Representação do modelo de cores RGB (13)

Cada ponto no sistema de coordenadas representa uma cor primária ou secundária, sendo que pontos sobre os eixos x,y,z representam as cores primárias, pontos que são uniões

de duas cores primárias representam as cores secundárias ciano, magenta e amarela e o ponto mais distante da origem representa a cor branca. A união destes pontos forma um cubo onde as cores estão definidas por pontos sobre ou dentro do cubo, definidas por vetores que se estendem a partir da origem (11).

Imagens no modelo RGB são formadas utilizando três planos, um para cada cor primária, os quais são matrizes com valores de intensidades que são combinadas para formar a imagem cromática. Todo processamento realizado em uma imagem colorida deve ser feito sobre as três matrizes e o resultado do processamento deve ser combinado para formar uma nova imagem cromática (11). Entretanto, isso pode ocasionar sérias alterações nas cores da imagem devido aos valores de intensidade de cada matriz serem alterados de maneira independente, podendo isto gerar cores diferentes quando combinadas e prejudicar o processamento baseado em extração de cor. Na figura 3.13 é ilustrado o processo de equalização de histograma em uma imagem com modelo de cor RGB que provocou alterações nas cores originais da imagem. A alteração notável está no caminho da plantação, onde a cor original marrom foi modificada para uma cor próxima do vermelho. É possível perceber na área correspondente a plantação algumas regiões que ficaram com cores próximas ao azul, principalmente no topo à esquerda da plantação.

a) b)

Figura 3.13: a) Imagem cromática original no modelo RGB b) imagem após a equalização de histograma, com a cor do caminho modificada

3.1.3.2 Modelo de cor HSI

A identificação de cores no modelo RGB torna-se difícil quando há muitas variações para uma mesma cor, já que é necessário utilizar a combinação de três valores para se chegar a uma cor. Além disso, o processamento de imagens neste modelo pode causar alterações de cores que podem prejudicar a identificação de padrões.

O modelo de cor HSI (*Hue, Saturation, Intensity*) é muito utilizado em sistemas de visão computacional devido a capacidade de identificação de cores utilizando apenas uma

informação. No modelo HSI a informação da cor é separada da informação de intensidade, sendo que as transformações de intensidade, tais como equalização de histograma e alterações de brilho e contraste, não afetam a informação de cor. No modelo HSI, a cor é representada por três componentes: matiz, saturação e intensidade. O matiz é o componente que descreve a cor pura de um objeto. Quando um objeto é denominado vermelho, laranja ou azul, por exemplo, está se fazendo uma referência ao componente matiz. A saturação é o componente que representa a quantidade de luz branca diluída com o matiz. O grau de saturação é inversamente proporcional a quantidade de luz branca no matiz, ou seja, cores mais saturadas tem pouca luz branca diluída. Por exemplo, cores como o rosa (vermelha e branca) e lilás (violeta e branca) são menos saturadas (11). A intensidade é o componente que descreve o brilho da imagem. Os componentes matiz e a saturação são denominados como a característica cromática da imagem. Portanto, no modelo HSI a imagem é composta pela intensidade e cromaticidade (11).

O modelo de cor HSI é descrito por um triângulo onde os vértices representam as cores primárias e as arestas correspondem as cores secundárias, totalmente saturadas, como ilustrado na figura 3.14.

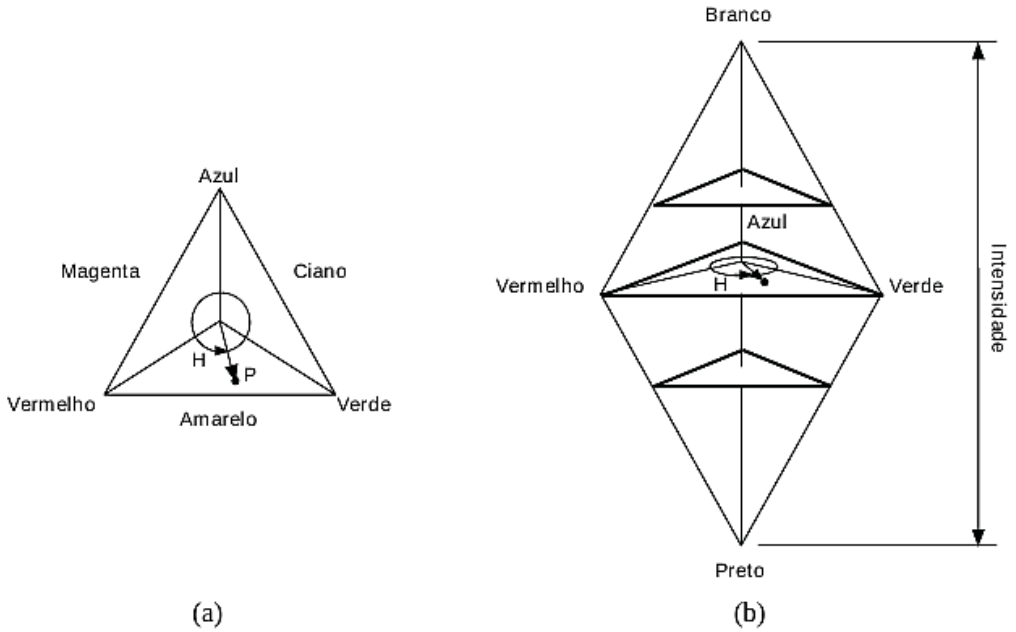


Figura 3.14: Representação do modelo de cor HSI (13)

O matiz H de um ponto P é descrito como a rotação do triângulo com origem na cor vermelha. Quando H=0º, então a cor é vermelha, quando H=60º a cor é amarela. A saturação é representada pela distância do ponto P em relação ao centro do triângulo, sendo o grau de

diluição de luz branca no matiz diretamente proporcional a distância do vértice do matiz ao centro do triângulo. Quanto mais distante do centro o ponto P estiver, mais saturada será a cor. A intensidade é obtida mediante o deslocamento vertical do triângulo sobre uma linha perpendicular que passa pelo seu centro, sendo mais escuras as intensidades abaixo do centro da linha e as intensidades mais claras acima do centro da linha, conforme pode ser visto na figura 3.14 (b).

O processamento de imagens utilizando técnicas de transformação de intensidade no modelo de cor HSI proporciona melhores resultados em relação ao modelo RGB, pois o processamento se limita apenas ao componente de intensidade sem afetar as características cromáticas da imagem. Na figura 3.15 é ilustrado o procedimento de equalização de histograma em uma imagem convertida para o modelo HSI.

a) b)

Figura 3.15: a) Imagem original b) resultado da equalização de histograma no modelo HSI

Na figura 3.15 (b) é possível perceber o aumento de intensidade da imagem sem perder as características de cores. O resultado da equalização em relação ao modelo RGB é mais adequado para processamento posterior devido a cromaticidade da imagem ter sido mantida.

3.1.3.3 Conversão de RGB para HSI

A conversão de imagens do modelo RGB para o HSI é feita mediante as equações 3.5, 3.6 e 3.7.

$$I = \frac{1}{3}(R+G+B) \tag{3.5}$$

$$S = 1 - \frac{3}{R+G+B}[min(R,G,B)] \tag{3.6}$$

$$H = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{(1/2)}}\right\} \tag{3.7}$$

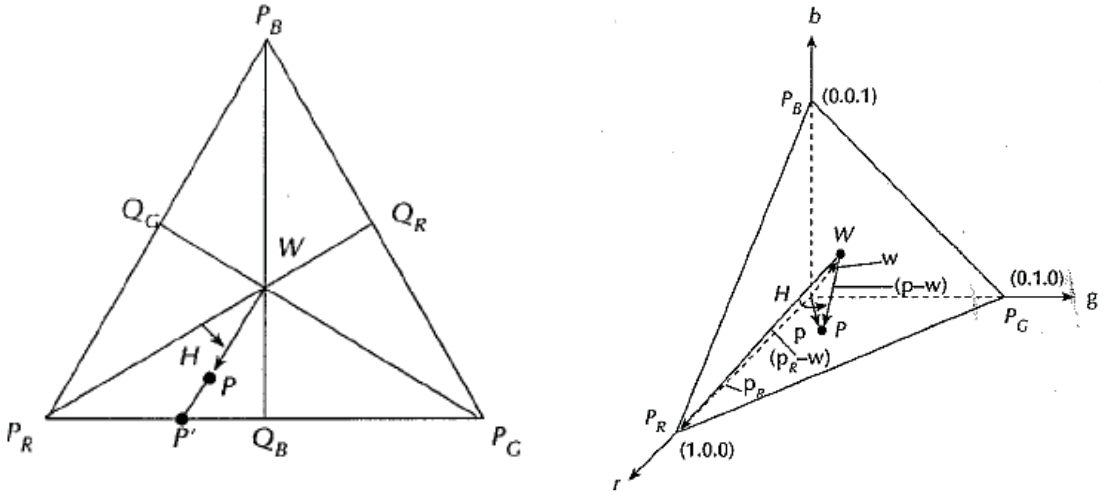
O componente de intensidade do modelo HSI é obtido de uma imagem RGB pela média dos valores R, G e B. Os valores R, G e B também podem ser normalizados no intervalo [0,1], sendo as equações 3.8, 3.9 e 3.10 utilizadas para representar essa normalização.

$$r = \frac{R}{R+G+B} \qquad 3.8$$

$$g = \frac{G}{R+G+B} \qquad 3.9$$

$$b = \frac{B}{R+G+B} \qquad 3.10$$

Para compreender a equação 3.7, é necessária a análise da construção geométrica do triângulo HSI mostrada na figura 3.16 (b) (11).



a) b)

Figura 3.16: a) representação do modelo de cor HSI b) construção geométrica do triângulo HSI (11)

A obtenção do componente matiz tem como base as seguintes condições (11):

a) O ponto W tem coordenadas (1/3, 1/3, 1/3), o qual corresponde o centro do triângulo;
b) Um ponto de cor P arbitrário tem coordenadas (r,g,b);
c) O vetor estendendo-se da origem até W é denotado por $\vec{w}$. Do mesmo modo, os vetores estendendo-se da origem até P_R e até P são denominados $\vec{p}_r$ e $\vec{p}$, respectivamente. P_R é o ponto que representa a cor vermelha;

Os segmentos de reta WP_R e WP podem ser descritos em termos vetoriais como (p_R – w) e (p – w), respectivamente. Sendo H o ângulo entre os segmentos de reta WP_R e WP, a projeção de WP sobre WP_R é obtida pelo produto escalar entre os dois segmentos de reta e é

descrita pela equação 3.11.

$$WP \cdot WP_R = ||WP||\ ||WP_R||\ cos(H) \quad 3.11$$

Na equação 3.11, $||WP||$ e $||WP_R||$ representam as normas (comprimento) dos vetores WP e WP_R, respectivamente. Substituindo pelos termos vetoriais, podemos reescrever a equação 3.11 da seguinte maneira:

$$(p - w) \cdot (p_R - w) = ||(p - w)||\ ||(p_R - w)||\ cos(H) \quad 3.12$$

O ângulo H pode ser obtido mediante a alteração da equação 3.12 da seguinte maneira:

$$H = \arccos\left\{ \frac{(p-w)\cdot(p_r-w)}{\|p-w\|\cdot\|p_r-w\|} \right\} \quad 3.13$$

A norma de um vetor v com elementos a_1, a_2 e a_3 é definida conforme a equação 3.14.

$$\|v\| = \sqrt{a_1^2 + a_2^2 + a_3^2} \quad 3.14$$

Sendo p = (*r,g,b*) e w = (1/3,1/3,1/3), então a norma do vetor *(p – w)* é dada pelas equações 3.15 e 3.16.

$$\|p-w\| = \left[\left(r-\frac{1}{3}\right)^2 + \left(g-\frac{1}{3}\right)^2 + \left(b-\frac{1}{3}\right)^2 \right]^{(1/2)} \quad 3.15$$

$$\|p-w\| = \left[\frac{9(R^2+G^2+B^2) - 3(R+G+B)^2}{9(R+G+B)^2} \right]^{(1/2)} \quad 3.16$$

Sendo p_R = (1,0,0) e w = (1/3,1/3,1/3), então a norma do vetor (p_R – w) é dado pela equação 3.17.

$$\|p_r-w\| = \left[\left(1-\frac{1}{3}\right)^2 + \left(0-\frac{1}{3}\right)^2 + \left(0-\frac{1}{3}\right)^2 \right]^{(1/2)} = \left[\frac{2}{3} \right]^{(1/2)} \quad 3.17$$

Os vetores (p – w) e (p_R – w) são descritos pelas equações 3.18 e 3.19 respectivamente.

$$(p-w) = \left(\left(r-\frac{1}{3}\right), \left(g-\frac{1}{3}\right), \left(b-\frac{1}{3}\right) \right) \quad 3.18$$

$$(p_r-w) = \left(\left(1-\frac{1}{3}\right), \left(0-\frac{1}{3}\right), \left(0-\frac{1}{3}\right) \right) \quad 3.19$$

O produto escalar *(p – w)* · *(p_R – w)* pode ser descrito por meio da equação 3.20.

$$(p-w)\cdot(p_R-w) = \left(\frac{2}{3}\left(r-\frac{1}{3}\right) - \frac{1}{3}\left(g-\frac{1}{3}\right) + \frac{1}{3}\left(b-\frac{1}{3}\right) \right) = \frac{2\mathrm{R}-G-B}{3(R+G+B)} \quad 3.20$$

Substituindo o produto escalar, produzido pela equação 3.20, e o resultado das equações 3.15 e 3.16 na equação 3.13, teremos a equação 3.21 do componente matiz, já

simplificada.

$$H = \arccos\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{(1/2)}}\right\} \qquad 3.21$$

Portanto, o componente matiz é descrito como o produto escalar entre os segmentos de reta WP_R e WP e a equação 3.21 produz valores no intervalo 0º ≤ H ≤ 180º.

Seja WP o segmento de reta com origem em W e término em P. A saturação S de um ponto P é dada pelo raio da reta |WP'|, onde P' é a projeção do ponto P sobre o plano mais próximo do triângulo, como pode ser observado na figura 3.17.

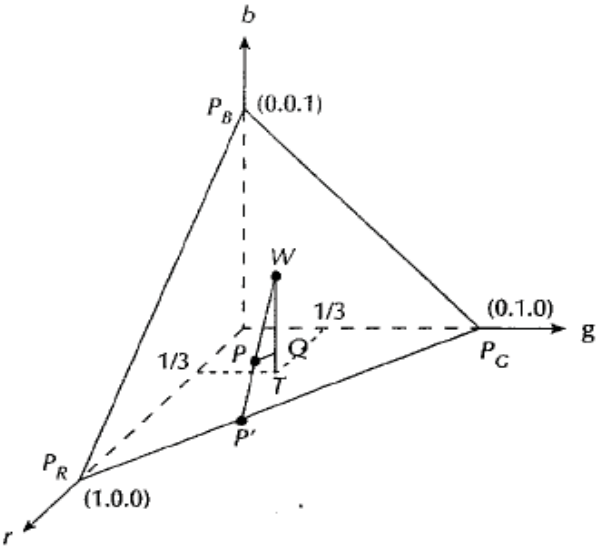


Figura 3.17: Representação geométrica do triângulo HSI para o componente S (11)

Seja T a projeção do ponto W sobre o plano *rg,* paralela ao eixo b, e seja Q a projeção do ponto P sobre a reta WT, paralela ao eixo *b.* A saturação S pode ser representada mediante a equação 3.22.

$$S = \frac{|WP|}{|WP'|} = \frac{|WQ|}{|WT|} = \frac{|WT|-|QT|}{|WT|} \qquad 3.22$$

Sendo |WT| = 1/3 e |QT| = b, então pode-se atribuir esses valores na equação 3.22. para obter a equação 3.23.

$$\begin{aligned} S &= 3\left(\frac{1}{3}-b\right) \\ &= 1-3\mathrm{b} \\ &= 1-b_0 \end{aligned} \qquad 3.23$$

Onde b_0 = B/I, sendo I dado pela equação 3.5. Sendo b0 o valor mínimo no plano rg,

então a obtenção do componente S é feita pela equação 3.24.

$$S = 1 - min(r_{0,} g_{0,} b_0) = 1 - \frac{3}{R+G+B}[min(R, G, B)] \quad 3.24$$

3.1.4 Morfologia matemática

A palavra morfologia se refere ao estudo da estrutura dos animais e plantas. A morfologia matemática é o estudo da estrutura geométrica dos objetos contidos em uma imagem. A morfologia matemática pode ser aplicada em várias áreas de processamento e análise de imagens, com objetivos tão distintos como por exemplo realce, filtragem, segmentação, detecção de bordas, esqueletização e afinamento (13).

O principio básico da morfologia matemática consiste em extrair as informações relativas à geometria e à topologia de um conjunto desconhecido (uma imagem), pela transformação por meio de outro conjunto completamente definido, chamado elemento estruturante. Portanto, a base da morfologia matemática é a teoria de conjuntos. Por exemplo, o conjunto de todos os *pixels* pretos em uma imagem binária descreve completamente a imagem. Em imagens binárias, os conjuntos em questão são membros do espaço inteiro bidimensional Z^2, onde cada elemento do conjunto é um vetor 2-D cujas coordenadas são as coordenadas (x,y) do pixel preto (por convenção) na imagem. Imagens com mais níveis de cinza podem ser representadas por conjuntos cujos elementos estão no espaço Z^3. Neste caso, os vetores têm três elementos, sendo os dois primeiros as coordenadas do pixel e o terceiro seu nível de cinza (13).

3.1.4.1 Teoria dos conjuntos

Sejam A e B conjuntos do espaço Z^2, cujos componentes são a = (a_1, a_2) e b = (b_1, b_2), respectivamente. A translação de A por x = (x_1, x_2) é descrita como:

$$A_x = \{c \mid c = a + x, para\, a \in A\} \quad 3.25$$

Diante disso, conclui-se que a translação é a soma dos componentes de A pelos componentes de X.

A reflexão do conjunto B é realizada tornando negativo os valores de b, sendo denotada por B_r e definida como:

$$B_r = \{x | x = -b, \, para \, b \in B\} \quad 3.26$$

O complemento do conjunto A é o conjunto de valores que não são pertencentes a A, sendo definido como:

$$A^c = \{x | x \notin A\} \quad 3.27$$

A diferença entre dois conjuntos A e B é definida como os valores que estão apenas no conjunto A, mas não em A e B simultaneamente, e é descrita como:

$$A - B = \{x | x \in A, x \notin B\} = A \cap B^c \quad 3.28$$

Na figura 3.18 é ilustrado o resultado da aplicação de cada operação de conjunto sobre uma imagem (13). O ponto preto em destaque na imagem de cada figura representa a origem de cada conjunto. Na figura 3.18 (a) é apresentada uma imagem com os pontos do objeto pertencentes ao conjunto A e na figura 3.18 (b) é ilustrado o processo de translação de A por x. Na figura 3.18 (c) é apresentada uma imagem com os pontos do objeto pertencentes ao conjunto B e na figura 3.18 (d) é ilustrado o processo de reflexão de B em torno do ponto preto em destaque na figura 3.18 (c). Na figura 3.18 (e) é apresentada uma imagem cujos valores de cinza estão no conjunto A e na figura 3.18 (f) é apresentado o resultado da diferença do conjunto A pelo conjunto B, o qual é representado pela imagem do quadrado sem os pontos sobrepostos da imagem do triângulo.

3.1.4.2 Dilatação

A operação morfológica de dilatação é realizada mediante a utilização de um elemento estruturante B sobre um conjunto A, ambos pertencentes a Z^2, e é definida como:

$$A \oplus B = \{x | (B_r) \cap A \neq \emptyset\} \quad 3.29$$

O processo de dilatação é a sobreposição da reflexão do elemento estruturante B sobre o conjunto A, sendo que essa sobreposição deve ter pelo menos um elemento não nulo. Essa interpretação permite reescrever a equação 3.29 da seguinte maneira:

$$A \oplus B = \{x | [(B_r)_x \cap A] \subseteq A\} \quad 3.30$$

O processo de dilatação pode ser visto como uma convolução do elemento estruturante B sobre o conjunto A. Desta maneira, o elemento estruturante é deslocado sobre o conjunto A, de maneira que sempre que houver uma sobreposição não nula da origem do elemento estruturante sobre o conjunto haverá uma dilatação deste conjunto de modo que ela complemente a sobreposição. Na figura 3.19 é ilustrado o processo de dilatação do conjunto A

utilizando a reflexão do elemento estruturante B.

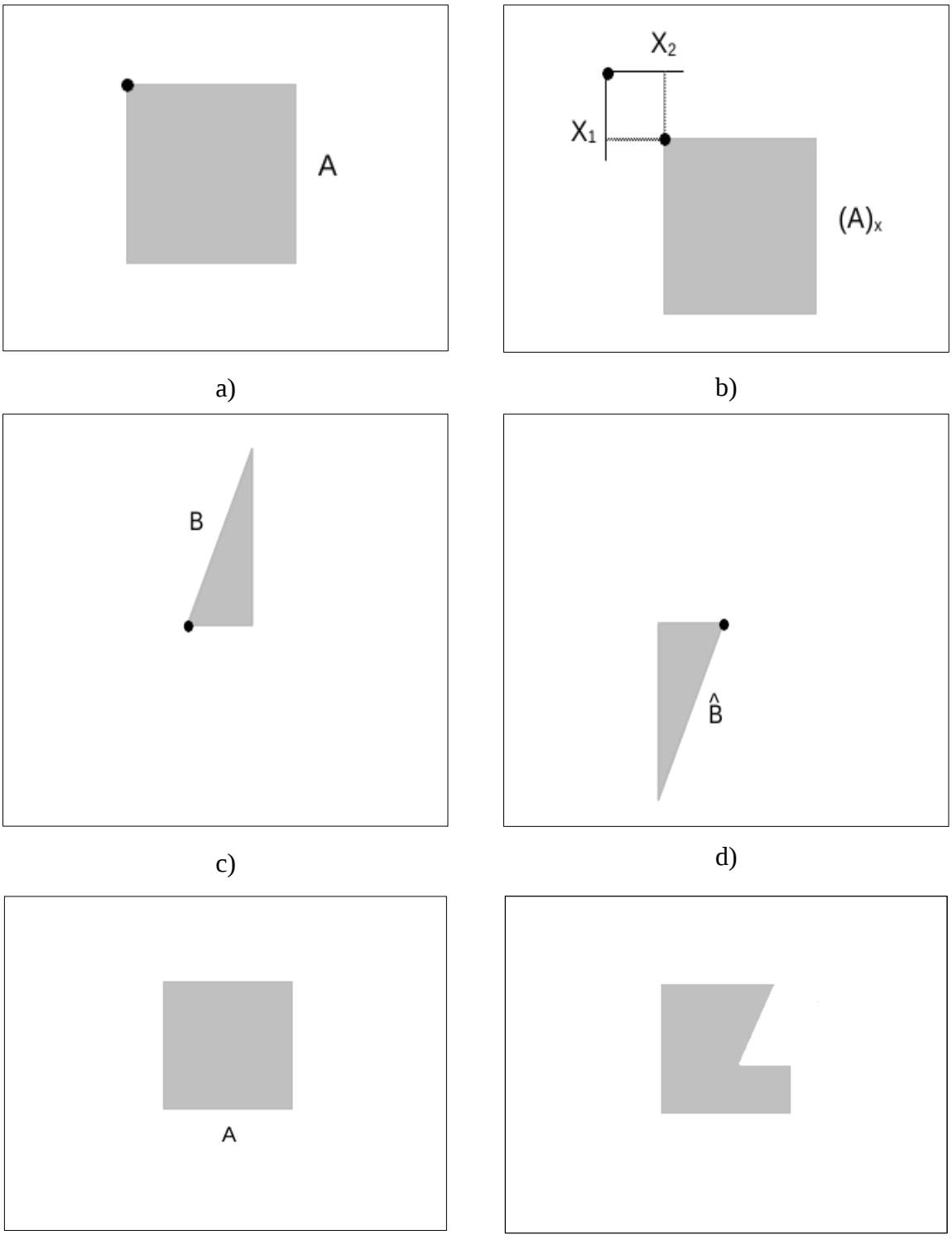
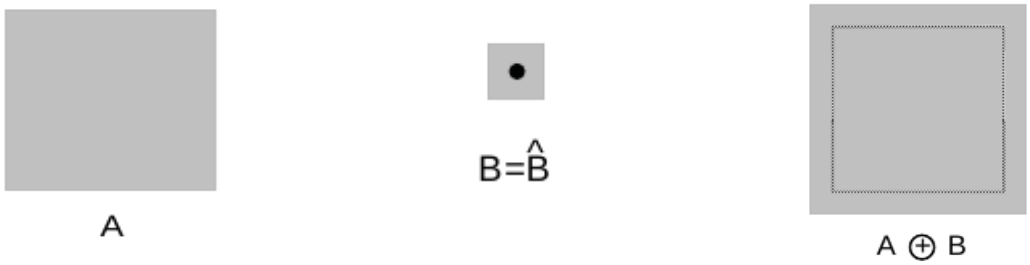


Figura 3.18: a) Conjunto A b) Conjunto A após a translação por X c) Conjunto B d) Conjunto B após a reflexão e) Conjunto A f) Resultado da diferença entre os conjuntos A e B (13)



a) b) c)

Figura 3.19: a) Conjunto A b) Reflexão do conjunto B c) Dilatação de A por B (13)

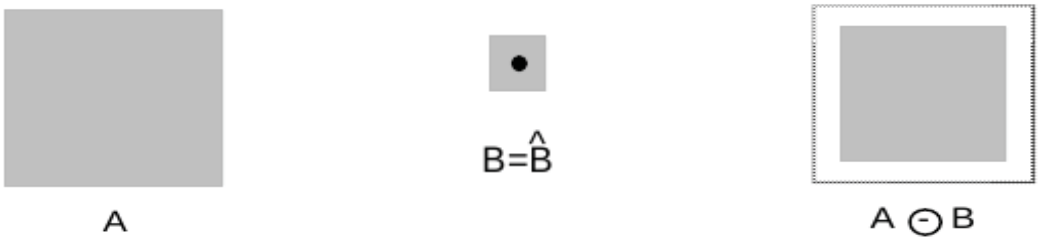
A linha pontilhada da imagem na figura 3.19 (c) indica os limites do conjunto original A, sendo que a região fora desta fronteira é o resultado da dilatação quando a origem do elemento estruturante B sobrepõem esta fronteira. A dilatação expande uma imagem.

3.1.4.3 Erosão

A erosão de um conjunto A pelo conjunto B, ambos pertencente ao Z^2, é descrita como $A \ominus B$ e definida como:

$$A \ominus B = \left\{ x \middle| (B_r) \subseteq A \right\} \qquad 3.31$$

A operação morfológica de erosão é a sobreposição do elemento estruturante B sobre o conjunto A, de maneira que essa sobreposição faça com que B esteja contido em A. Quando a origem de B estiver sobre A e a condição de contenção for satisfeita, então o conjunto A será erodido por B de modo que este processo complemente a sobreposição. Na figura 3.20 (c) é ilustrado o procedimento de erosão do conjunto A pelo elemento estruturante B.



a) b) c)

Figura 3.20: a) Conjunto A b) Elemento estruturante c) Resultado da erosão (13)

O contorno pontilhado na figura 3.20 (c) é a fronteira original do conjunto A, sendo a região não preenchida entre esta fronteira e a borda do conjunto o resultado da erosão. A erosão diminui uma imagem.

3.1.4.4 Abertura e fechamento

As operações de dilatação e erosão podem ser utilizadas como filtro de pré-processamento de imagens, principalmente na redução de ruídos e suavização de bordas. A abertura é um procedimento que tende a suavizar as bordas de uma imagem, quebrar istmos estreitos e eliminar proeminências. O fechamento também tende a suavizar as bordas, mas também pode fundir as quebras em golfos finos, eliminar pequenas aberturas e preencher fendas em um contorno (11).

A operação de abertura de um conjunto A por um elemento estruturante B é descrita como A ∘ B e definida como:

$$A \circ B = (A \ominus B) \oplus B \qquad 3.32$$

A abertura é a erosão do conjunto A pelo elemento estruturante B, seguida de uma dilatação. A erosão tende a eliminar as pontes estreitas que ligam duas regiões do conjunto e as pequenas proeminências da imagem. A dilatação tende a suavizar as bordas do conjunto erodido. Na figura 3.21 é ilustrado o procedimento de abertura de um conjunto por um elemento estruturante em forma de disco.

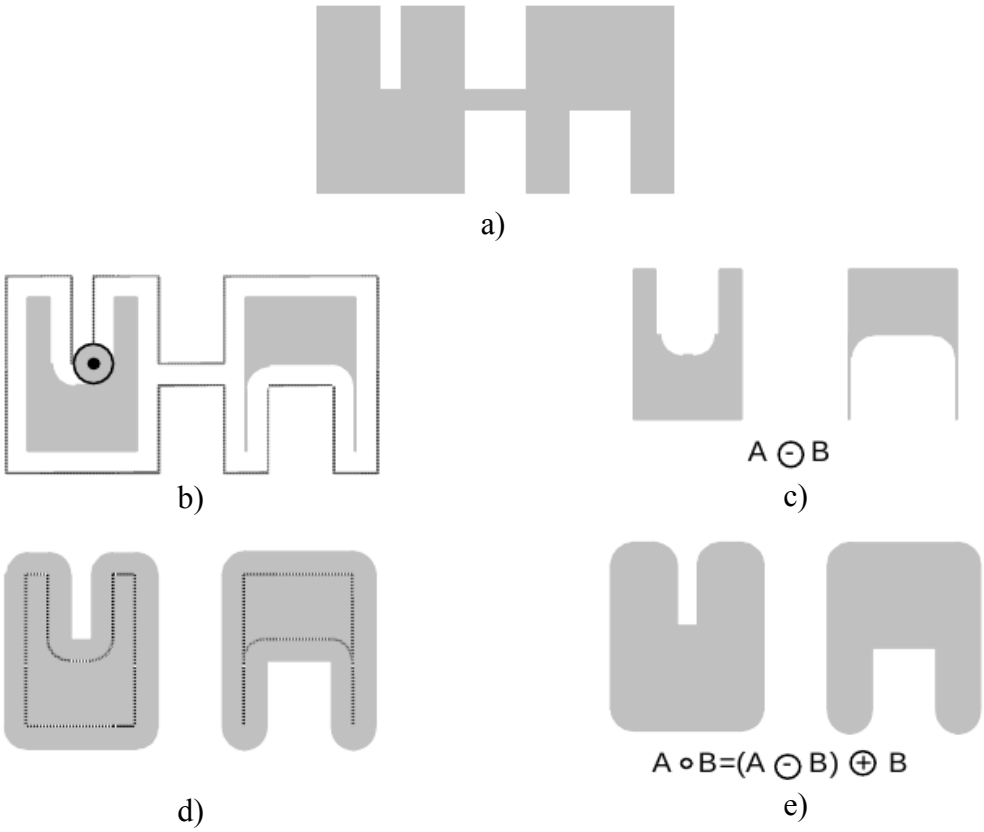


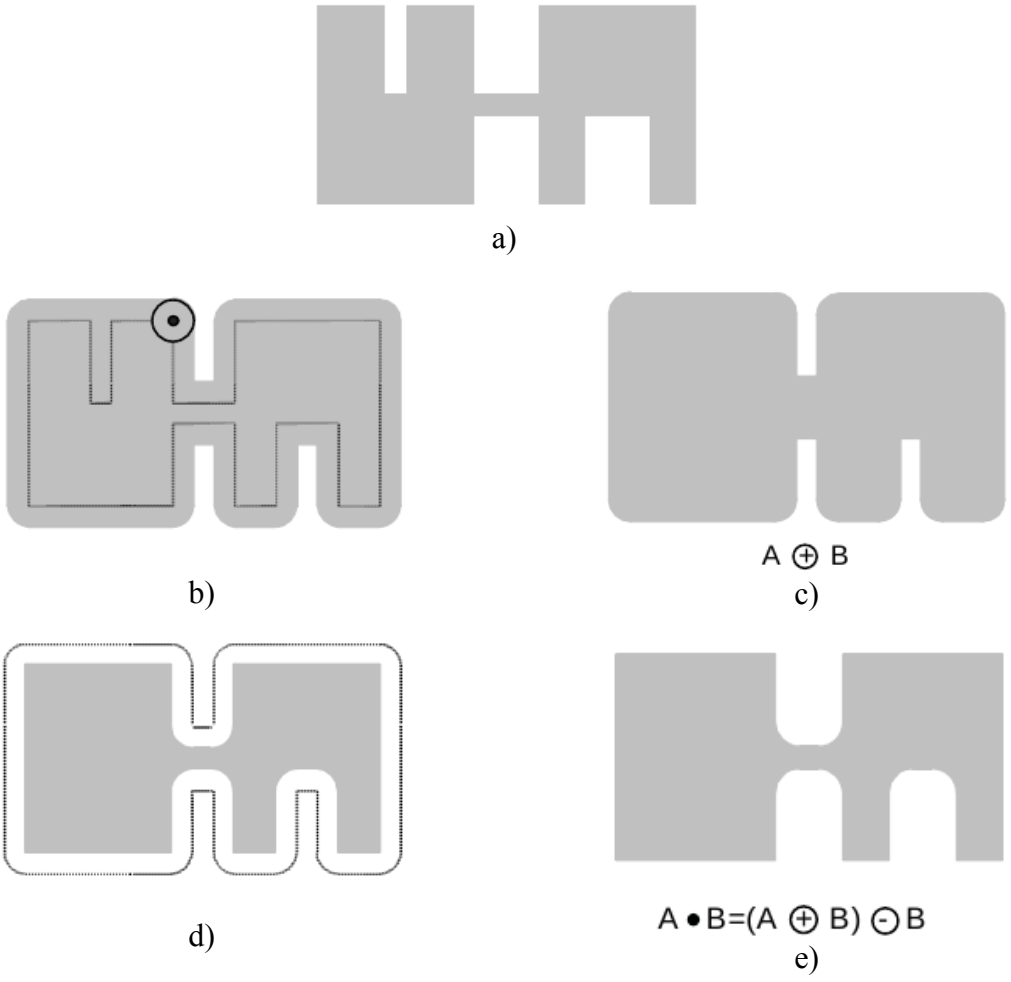
Figura 3.21: a) Conjunto A b) processo de erosão c) resultado da erosão d) processo de dilatação e) resultado da dilatação (13)

Na figura 3.21 (c) é exibido o resultado da operação de erosão, onde a ponte que une as duas regiões do conjunto foi eliminada devido o tamanho do elemento estruturante ser maior que ponte. O resultado da dilatação exibido na figura 3.21 (e) é uma imagem com as bordas suavizadas e as concavidades expandidas.

A operação de fechamento é descrita como A • B e definida como:

$$A \bullet B = (A \oplus B) \ominus B \qquad 3.33$$

O fechamento é a dilatação do conjunto A pelo elemento estruturante B, seguida de uma erosão. A dilatação tende a eliminar pequenos buracos na imagem, fundir istmos estreitos ou expandí-los e eliminar pequenas concavidades. A erosão tende a eliminar pequenas saliências remanescente da dilatação, aumentar as concavidades e suavizar as bordas. Na figura 3.22 é ilustrada a operação de fechamento em um conjunto.



a)

b)

c)

d)

e)

Figura 3.22: a) conjunto b) processo de dilatação c) resultado da dilatação d) processo de erosão e) resultado da erosão (13)

3.1.5 Segmentação de imagens

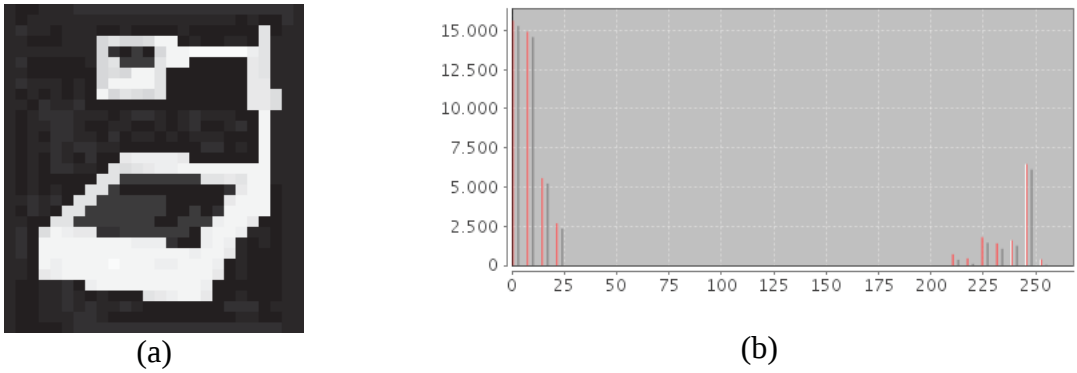
A segmentação consiste em dividir uma imagem em suas partes ou objetos constituintes e extrair apenas as partes da imagem que são relevantes para a análise. A segmentação envolve duas características: descontinuidades e similaridade. A primeira característica se baseia em mudanças abruptas de níveis de cinza ou de cor, tais como bordas, pontos e linhas. A segunda característica se baseia na extração de regiões similares da imagem e as técnicas desta categoria são limiarização, crescimento de regiões e divisão e fusão de regiões.

3.1.5.1 Binarização

A binarização é uma técnica que consiste em estabelecer um valor limiar T de maneira que

$$g(x,y)=\begin{cases}1 \, se \, f(x,y)>T\\0 \, se \, f(x,y)\leq T\end{cases} \qquad 3.34$$

O objetivo da utilização desta técnica é criar uma imagem binária, isto é, com apenas dois níveis de intensidade: 0 e 1. Uma imagem composta de objetos iluminados sobre um fundo escuro pode ser separada em dois grupos dominantes (11). O histograma de uma imagem com essa característica é dividido em dois ou mais grupos de níveis de intensidade, sendo que os grupos com níveis de intensidade baixo representam o fundo da imagem e os com níveis de intensidade alto representam os objetos. Assim, no processo de binarização os objetos presentes na imagem têm intensidade igual à 1 e a parte que corresponde o fundo tem intensidade igual à 0, ou vice-versa. Na figura 3.23 é exibida uma imagem e seu histograma, no qual é possível verificar dois grupos de intensidade de cinza.



(a) (b)

Figura 3.23: a) Imagem original b) Histograma

A maior dificuldade do processo de binarização de uma imagem é a escolha do valor do limiar. Um valor inadequado pode resultar em perda de informações dos objetos, as quais podem ser interpretadas como fundo. A escolha pode ser feita de maneira visual por um especialista. Na figura 3.23, por exemplo, a análise do histograma indica que um valor de limiar entre 50 e 200 resultará em uma boa separação dos dois grupos de intensidade de cinza. No entanto, em diversas aplicações é necessário o estabelecimento automático do valor do limiar, sem a intervenção de um especialista. Há vários métodos que podem ser utilizados para determinar o valor do limiar, tais como o método de Otsu, o qual se baseia no histograma da imagem, e a técnica do método do vale, na qual identifica-se regiões côncavas no histograma que representam a transição dos níveis de cinza entre o objeto e o fundo.

A conversão de uma imagem em níveis de cinza para uma representação binária melhora o desempenho de algoritmos nas etapas de representação e reconhecimento, já que apenas dois valores de intensidade são utilizados na análise.

3.1.5.2 Segmentação orientada a regiões

A segmentação orientada a regiões consiste em particionar uma imagem em regiões conforme um critério de similaridade entre *pixels* adjacentes. Seja R a completa região de uma imagem, o processo de particionamento da região R em n regiões menores R_1, R_2, … R_n segue as seguintes condições (11):

1. $\bigcup_{i=1}^{n} R_i = R$
2. R_i é uma região conexa, i = 1, 2, …, n
3. $R_i \cap R_j = \phi$ para todo i e j, $i \neq j$
4. $P(R_i)$ = VERDADEIRO para i = 1, 2, …, n
5. $P(R_i \cup R_j)$ = FALSO para $i \neq j$

A primeira condição indica que a segmentação deve ser completa, ou seja, cada pixel deve pertencer a uma região. A segunda condição indica que os *pixels* de uma região deve seguir uma conectividade. A terceira condição indica que as regiões são disjuntas. A quarta condição define o critério que deve ser estabelecido para que os *pixels* pertençam a uma região. Por exemplo, pode-se definir que uma região será composta somente de *pixels* adjacentes de mesma intensidade. A quinta condição estabelece que as regiões R_i e R_j são diferentes em relação ao predicado P (11).

O crescimento de regiões é um procedimento que agrega *pixels* para formar regiões maiores. A agregação de *pixels* é um dos procedimentos de crescimento de regiões e consiste na escolha de *pixels* denominados “sementes”, onde a partir deles os *pixels* adjacentes que obedeçam um critério de similaridade – tais como cor ou textura - são agregados para formar uma região (11). Na figura 3.24 é apresentado um exemplo do conceito de crescimento de regiões por agregação de *pixels*.

0	50	3	100	70
8	255	255	255	255
0	34	255	255	67
40	21	255	255	0
90	87	255	255	80

(a)

0	50	3	100	70
8	a	a	a	a
0	34	a	a	67
40	21	a	a	0
90	87	a	a	80

(b)

Figura 3.24: a) matriz da imagem original b) resultado do crescimento por agregação de pixel em b) (Adaptado de (11))

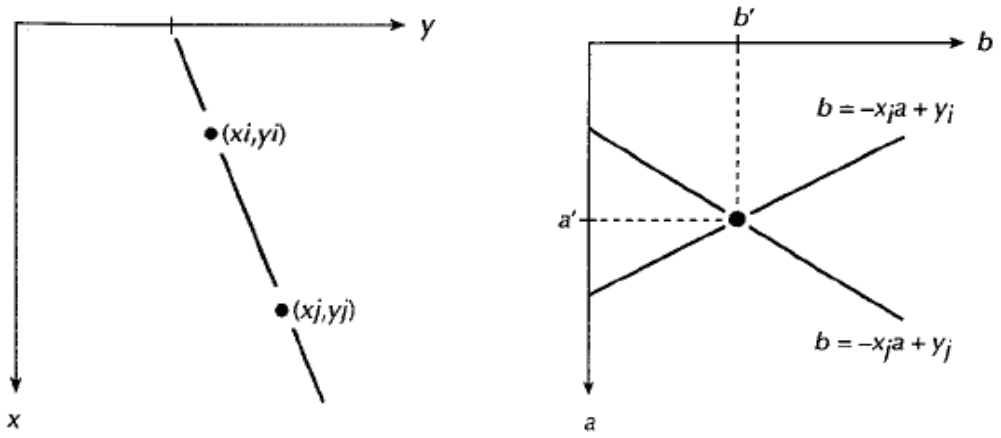
A figura 3.24 (a) representa uma matriz de uma imagem de nível de cinza. O pixel (2,2) será utilizado como semente para formar uma região com *pixels* que possuam a mesma intensidade deste pixel semente. Na figura 3.24 (b) é apresentado o resultado da segmentação, onde os *pixels* pertencentes a região do pixel semente estão rotulado com a letra “a”.

3.1.5.3 Transformada de Hough

Suponha que para n pontos em uma imagem desejamos encontrar subconjuntos desses pontos que estejam contidos em retas. Uma solução possível é primeiro encontrar todas as linhas contidas na imagem, seguida da busca dos pontos contidos nas linhas. O problema com esse procedimento é que ele envolve a busca de $n(n-1)/2 \approx n^2$ linhas, seguido de $(n)n(n-1)/2 \approx n^3$ comparações de cada ponto com todas as linhas (11).

A transformada de Hough é uma técnica utilizada para transformar os pontos de uma imagem do espaço de coordenadas (x,y) para o espaço de parâmetros (a,b), onde a é o coeficiente de inclinação da reta e b é o deslocamento da reta em relação ao eixo y (11). A equação da reta, descrita como $y = ax + b$, pode ser reescrita como $b = -ax + y$. No primeiro caso, os valores (a,b) são fixos e formam uma reta que contenha um conjunto de pontos (x,y). A equação do segundo caso é utilizada para formar uma reta que contenha um conjunto de

pontos (a,b) com valores fixos para (x,y). Portanto, um ponto (x,y) no espaço da imagem corresponde a uma reta no espaço de parâmetros, como pode ser observado na figura 3.25.



a) b)

Figura 3.25: a) Pontos no espaço (x,y) b) Pontos no espaço de parâmetros (a,b) (11)

Os pontos (x_i,y_i) e (x_j,y_j) na figura 3.25 (a) correspondem a duas retas no espaço de parâmetros, como mostrado na figura 3.25 (b), quando utilizados na equação da reta do espaço de parâmetros. Os pontos a' e b' representam, respectivamente, a inclinação da reta e o ponto de intersecção com o eixo y da linha que passa pelos pontos (x_i,y_i) e (x_j,y_j) no plano (x,y). Portanto, os pontos contidos na reta mostrada na figura 3.25 (a) possuem retas que se interceptam no ponto (a',b') do espaço de parâmetros.

Outra maneira de representar a reta no espaço (x,y) é utilizar as coordenadas polares (ρ, θ), onde ρ é a distância da reta em relação a origem e θ representa o ângulo de inclinação do segmento $\overline{AB}$ em relação ao eixo x, conforme mostrado na figura 3.26.

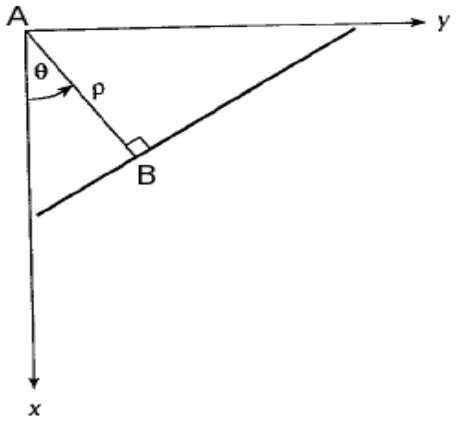


Figura 3.26: Representação da reta em coordenadas polares (Adaptado de Gonzalez e Woods (11))

A construção de uma reta utilizando estes atributos é feita mediante a equação 3.35.

$$x\cos(\theta) + y\,sen(\theta) = \rho \qquad 3.35$$

Os pontos colineares do espaço (x,y) são representados por meio de senóides no espaço de parâmetros (ρ, θ). M pontos colineares no espaço (x,y) geram M senóides no espaço de parâmetros (ρ, θ) que se interceptam em um ponto (ρ_i, θ_j). O domínio do ângulo é ±90º em relação ao eixo x. Assim, uma reta horizontal tem θ=0º e ρ sendo igual a intersecção com o eixo positivo x. Uma reta vertical tem θ=90º e ρ igual a intersecção com o eixo positivo y ou θ=-90º com ρ sendo a intersecção com o eixo negativo y (11).

O objetivo é encontrar os valores (ρ, θ) para a construção das retas no espaço (x,y), sendo a abordagem utilizada para esta tarefa a construção das chamadas células acumuladoras, as quais são mostradas na figura 3.27.

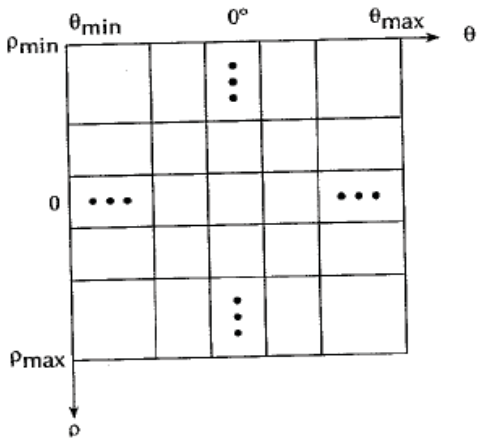
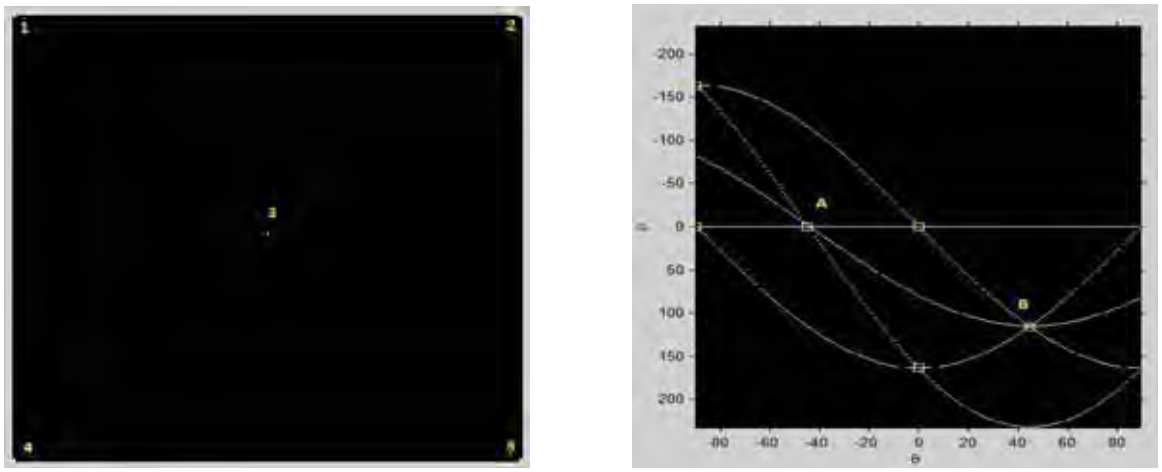


Figura 3.27: Células acumuladoras (11)

Cada célula acumuladora representa a quantidade de pontos colineares em uma reta com coordenadas polares (ρ_i, θ_j). Cada célula do arranjo acumulador é inicializada com o valor zero e as seguintes etapas são realizadas para se obter os valores de (ρ, θ):

- Utilizar a magnitude do gradiente na imagem de entrada para obter suas bordas;
- Aplicar cada valor de pixel (x,y) da borda à equação 3.35 e incrementar a célula C(ρ, θ) do acumulador caso a equação seja satisfeita;
- Procurar pelas células (ρ, θ) com valores mais altos no arranjo acumulador, os quais representam a quantidade de pontos colineares em uma reta de coordenadas polares (ρ, θ), e substituir na equação 3.35 para encontrar a equação da reta na forma y = ax + y.

Na figura 3.28 (b) é exibido o resultado da transformada de Hough sobre a imagem ilustrada na figura 3.28 (a). Na figura 3.28 (b) a interceptação de três senóides no ponto A, referentes aos ponto (1,3,5) do espaço (x,y), indica a existência de colinearidade entre esses pontos no espaço (x,y). Da mesma maneira, a existência da colinearidade entre os pontos (2,3,4) pode ser vista pela interceptação de três senóides no ponto B.



a) b)

Figura 3.28: a) pontos no espaço (x,y) b) senóides no espaço de parâmetros

3.1.6 Esqueleto de uma região

Segundo Gonzalez e Woods (11), uma abordagem importante para a representação de uma região planar consiste em reduzí-la a um grafo. Essa redução é realizada obtendo-se o esqueleto da região por meio de um algoritmo de afinamento. O processo de obtenção do esqueleto de uma imagem é útil em muitas aplicações, tais como biometria e reconhecimento de caracteres. A procedimento de extração de um esqueleto pode ser realizado por meio de técnicas tais como a transformação do eixo médio ou por meio de procedimentos de dilatação e erosão da morfologia matemática (11).

Algoritmos de afinamento são frequentemente utilizados para eliminar pontos da fronteira de uma região obedecendo algumas restrições, tais como: não remover pontos extremos, não comprometer a conectividade e não causar erosão excessiva na imagem (11). Uma técnica de afinamento de implementação simples foi proposta por Zhang e Suen (14), a qual é utilizada em imagens binárias. Seja uma região composta de *pixels* de cor branca sobre um fundo preto. O algoritmo é executado em dois passos sobre um pixel da região que possua pelo menos um vizinho de 8 com mesma intensidade de cinza. Essa restrição indica que o

pixel deve pertencer a borda da região. Na figura 3.29 é ilustrada uma definição de vizinhança de 8.

p9	p2	p3
p8	p1	p4
p7	p6	p5

Figura 3.29: Definição de uma vizinhança de 8

A aplicação do algoritmo no primeiro passo elimina um ponto p da borda se as seguintes restrições forem satisfeitas:

a) $2 \leq N(p1) \leq 6$;
b) $S(p1) = 1$;
c) $p2 * p4 * p6 = 0$;
d) $p4 * p6 * p8 = 0$.

Nas condições acima, N(p1) representa o número de vizinhos não nulos de p1, S(p1) é o número de transições 0-1 na sequência p2, p3, …, p2, onde 0 representa a cor preta e 1 representa a cor branca e p2, p3, …, p2 representam os *pixels* da imagem original sobrepostos pela máscara da figura 3.29. Na figura 3.30 é ilustrada uma região sobre uma vizinhança de 8, sendo que neste caso N(p1) =5 e S(p1) = 2.

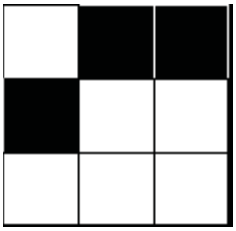


Figura 3.30: Região de uma imagem sobre uma vizinhança de 8 com N(p1) = 5 e S(p1) = 2

O segundo passo do algoritmo mantém as condições a) e b) do primeiro passo. Entretanto, as condições c) e d) são alteradas da seguinte maneira:

c') $p2*p4*p8 = 0$
d') $p2*p6*p8 = 0$

A condição a) não é satisfeita caso o número de vizinhos brancos do pixel p1 for 1 ou 8. Se o pixel p1 tiver apenas um vizinho branco, então significa que o pixel é o final de um segmento do esqueleto e este não pode ser removido. Se o pixel p1 tiver oito vizinhos com a cor branca, a sua eliminação pode provocar erosão na região. A condição b) não é satisfeita se

o pixel central p1 tiver mais que uma transição 0-1 na sequência p2, p3, …, p2 e o pixel p1 não pode ser removido para evitar desconexão de segmentos do esqueleto. Por exemplo, na figura 3.30 a remoção do pixel p1 provocaria uma desconexão do pixel (0,0) da região. As condições c) e d) são satisfeitas simultaneamente quando, no mínimo, p4 = 0 ou p6 = 0, indicando que é um ponto de borda do leste e sul, respectivamente, ou quando p2 = 0 e p8 = 0, indicando que é um ponto de borda noroeste. De maneira similar, as condições c') e d') são satisfeitas simultaneamente quando p2 = 0 ou p8 = 0, indicando um ponto de borda norte e oeste respectivamente, ou quando p4 =0 e p6 = 0, indicando um ponto de borda sudeste. Pontos de borda do canto nordeste possuem p2 = 0 e p 4 = 0, assim como pontos do canto sudoeste possuem p6 = 0 e p8 = 0 (11). Na figura 3.31 são ilustradas imagens com diferentes pontos de borda, sendo que todas satisfazem as condições do primeiro e do segundo passo.

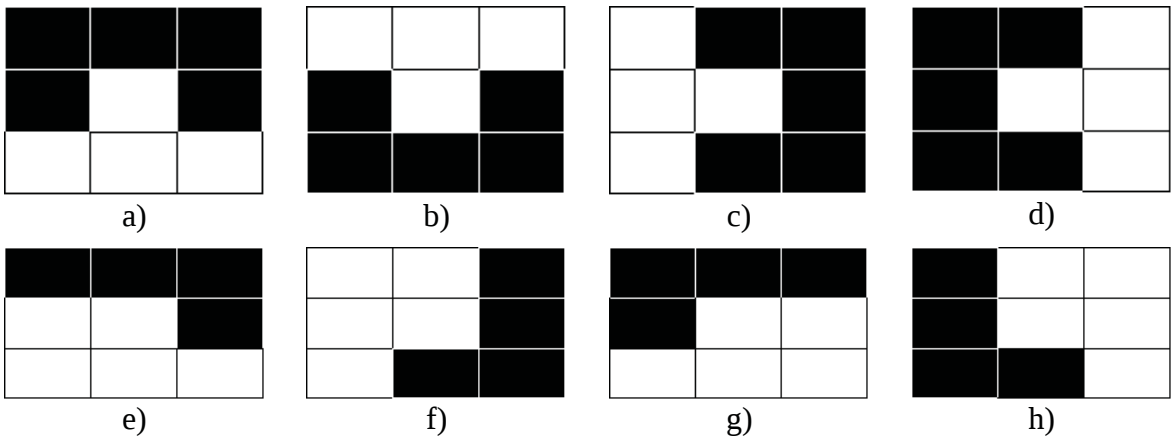


Figura 3.31: Ponto de borda p1 na região a) norte b) sul c) leste d) oeste e) nordeste f) sudeste g) noroeste h) sudoeste

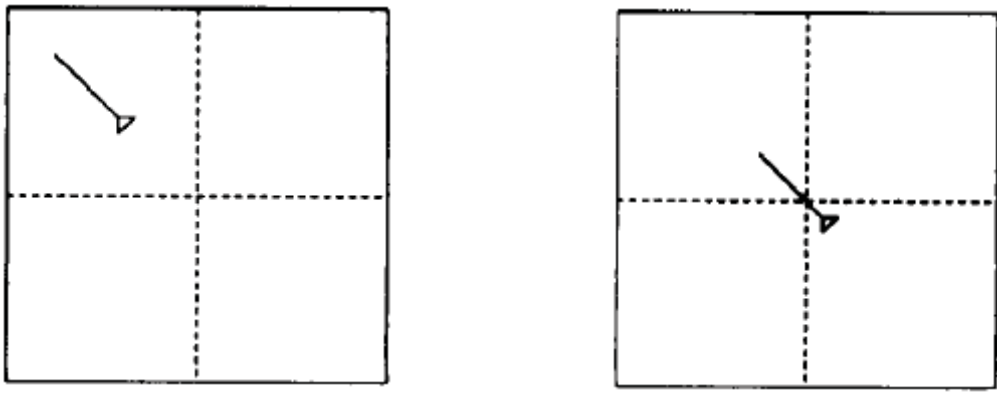
As imagens contidas na figura 3.31 (a-d) possuem N(p1) = 3, e as imagens contidas nas figuras 3.31 (e-h) contém N(p1) = 4. Além disso, todas as imagem possuem S(p1) = 1.

3.1.7 Translação de um objeto na imagem

Segundo Yüccer e Oflazer (15) o reconhecimento de padrões é altamente sensível à transformações de rotação, escala e translação do objeto de interesse na imagem. Eles relatam que na etapa de treinamento a área de interesse concentra-se na localização em que o padrão se encontra. Diante desta característica, eles desenvolveram um sistema de classificação de objetos baseado em redes neurais artificiais o qual é invariante a rotação, translação e escala do objeto na imagem. No trabalho deles são descritos três módulos que compõem o sistema:

módulo de pré-processamento, módulo de classificação e módulo de interpretação. O módulo de pré-processamento é utilizado para realizar transformações em imagens, de modo que o resultado são imagens com translação, rotação e escala uniformes. O módulo de classificação utiliza uma rede neural artificial para o reconhecimento das imagens resultantes do módulo de pré-processamento. O módulo de interpretação é utilizado para avaliar o resultado da rede neural artificial e determinar a qual classe uma imagem pertence (15).

O módulo de pré-processamento é composto de três sub-módulos: módulo de translação, módulo de rotação e módulo de escala. No módulo de translação, é feito o cálculo do centro do objeto para fazer com que ele coincida com o centro da imagem. Na figura 3.32 é ilustrado um exemplo deste processo.



a) b)

Figura 3.32: a) Objeto fora do centro da imagem b) objeto deslocado para o centro da imagem (15)

O centro do objeto é obtido pela média das coordenadas (x,y) que contêm *pixels* brancos e é representada pelas equações 3.36 e 3.37 .

$$x_{av} = \frac{1}{\sum_{i=1}^{N}\sum_{j=1}^{N} f(x_i, y_j)} \sum_{i=1}^{N}\sum_{j=1}^{N} f(x_i, y_j)\cdot x_i \qquad 3.36$$

$$x_{av} = \frac{1}{\sum_{i=1}^{N}\sum_{j=1}^{N} f(x_i, y_j)} \sum_{i=1}^{N}\sum_{j=1}^{N} f(x_i, y_j)\cdot y_j \qquad 3.37$$

onde f(x, y) é a intensidade de cinza nas coordenadas (x,y). A função de mapeamento dos *pixels* do objeto para o centro da imagem é realizada mediante a equação 3.38.

$$f_t(x_i, y_i) = ((M/2) - Xav, (N/2) - Yav) \qquad 3.38$$

onde M e N representam a quantidade de colunas e linhas da imagem, respectivamente. O procedimento consiste em calcular a diferença entre o centro (x_{av}, y_{av}) do objeto e o centro (x,y) da imagem e utilizar esta diferença para deslocar os *pixels* do objeto para o centro da imagem.

3.2 Redes Neurais Artificiais

Redes neurais artificiais representam uma abordagem conexionista da inteligência artificial e são estruturas baseadas no modelo do sistema nervoso dos seres vivos. O sistema nervoso é uma estrutura composta por células denominadas neurônios, as quais reagem a um determinado estímulo do ambiente. A estrutura de um neurônio biológico é composta por dendritos, corpo celular e axônio. Os dendritos são ligações por onde são recebidos os sinais de entrada, provenientes do ambiente externo ou de outros neurônios, os quais são transmitidos para o corpo celular. O corpo celular é o núcleo de processamento do neurônio, no qual ocorre a transformação dos sinais de entrada em novos impulsos a serem transmitidos para os próximos neurônios. Os axônios são ligações de saída do neurônio, os quais transmitem os sinais processados pelo corpo celular aos neurônios conectados à sua extremidade. O final do axônio é composto de ramificações que se conectam aos dendritos de entrada do próximo neurônio por meio de uma região de contato denominada sinapse. A sinapse é ponto de transmissão de sinais entre os neurônios. Quando os sinais de saída atingem a extremidade de um axônio, na região pré-sináptica, ocorre a liberação de substâncias químicas transmissoras, as quais são recebidas pela região pós-sináptica, provocando o impulso nervoso que será recebido pelo próximo neurônio.

O modelo matemático de um neurônio biológico foi proposto por McCulloch e Pitts em 1943 [14] e sua estrutura pode ser vista na figura 3.33.

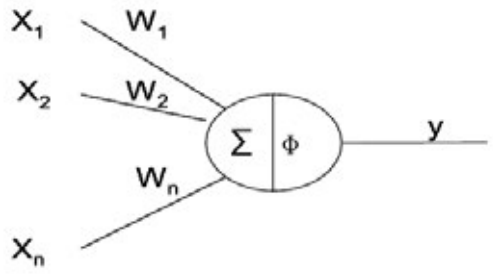


Figura 3.33: Modelo matemático do neurônio de McCulloch e Pitts (16)

O neurônio artificial possui conexões de entrada semelhantes aos dendritos de um neurônio biológico. Essas conexões recebem os valores de sinais do ambiente externo, como por exemplo os valores capturados por sensores, ou os valores de saída de outro neurônio. A cada conexão é atribuído um peso *w*, o qual é multiplicado pelo seu valor de entrada *x*. Os valores ponderados de cada conexão são utilizados em uma função de somatório para produzir uma soma ponderada, a qual é utilizada por uma função de ativação $\varphi(.)$ para produzir a saída do neurônio. Se o valor de saída for superior à um valor limiar estabelecido, então a saída do neurônio é estimuladora. Caso contrário, a saída do neurônio é inibitória. A equação 3.39 expressa o funcionamento do neurônio artificial.

$$v_i = \sum_{i=1}^{n} x_i w_i \qquad 3.39$$

Na equação 3.39, n é o número de entradas e w_i é o peso da conexão associada ao neurônio x_i.

A função de ativação φ é utilizada para estabelecer se a saída do neurônio será estimuladora ou inibitória e pode ser expressa mediante a equação 3.40.

$$y_i = \varphi(v_i) \geq \theta \qquad 3.40$$

A saída do neurônio será ativada se o somatório dos produtos entre as entradas e seus pesos for maior ou igual ao valor limiar estabelecido. As funções de ativação mais utilizadas são a função linear, a função rampa, a função degrau e a função sigmóide, mostradas na figura 3.34 e comentadas a seguir.

A equação 3.41 é utilizada para representar a função linear, mostrada na figura 3.34 (a).

$$\varphi(v_i) = \alpha \cdot v_i \qquad 3.41$$

onde α é o coeficiente que determina a inclinação da reta que passa pela origem. A função linear pode ser reescrita para gerar valores entre (-1,1), sendo a equação 3.42 utilizada para representar essa restrição, dando origem a função rampa.

$$\varphi(v_i) = \begin{cases} 1 \ se \ v_i \geqslant +c \\ v_i \ se -c < v_i < +c \\ -1 \ v_i \leqslant -c \end{cases} \qquad 3.42$$

onde *c* é um valor constante. Se a equação 3.42 for modificada para gerar valores positivos ou negativos, sem um intervalo constante, isto dá origem a função degrau.

$$\varphi(v_i) = \begin{cases} 1 \ se \ v_i > +c \\ -1 \ v_i \leqslant -c \end{cases} \qquad 3.43$$

Uma das funções de ativação mais utilizadas em redes neurais artificiais é a função sigmóide. Uma característica deste tipo de função é que ela é contínua no intervalo [-1,1] e diferenciável (17). Uma função de ativação do tipo sigmóide é a função logística, mostrada na equação 3.44.

$$\varphi(v_i)=\frac{1}{1+\exp(-av_i)} \qquad 3.44$$

onde *a* é o parâmetro de inclinação da curva.

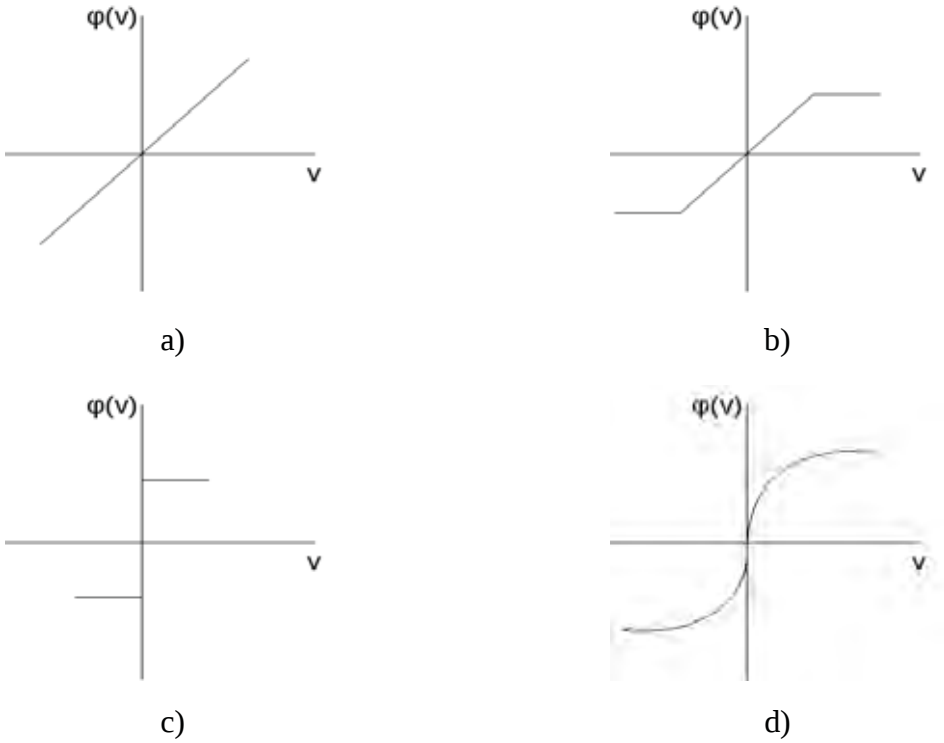


Figura 3.34: a) Função de ativação linear b) função de ativação rampa c) função de ativação degrau d) função de ativação sigmóide (16)

3.2.1 Estrutura de uma rede neural artificial

Uma rede neural artificial é uma estrutura formada por unidades de processamento interconectadas que simulam o comportamento de uma rede neural biológica. Braga (16) define uma rede neural artificial como um sistema de processamento paralelo, devido à topologia da rede e a distribuição de processamento das informações de entrada para cada neurônio. Há três características que definem a rede neural artificial:

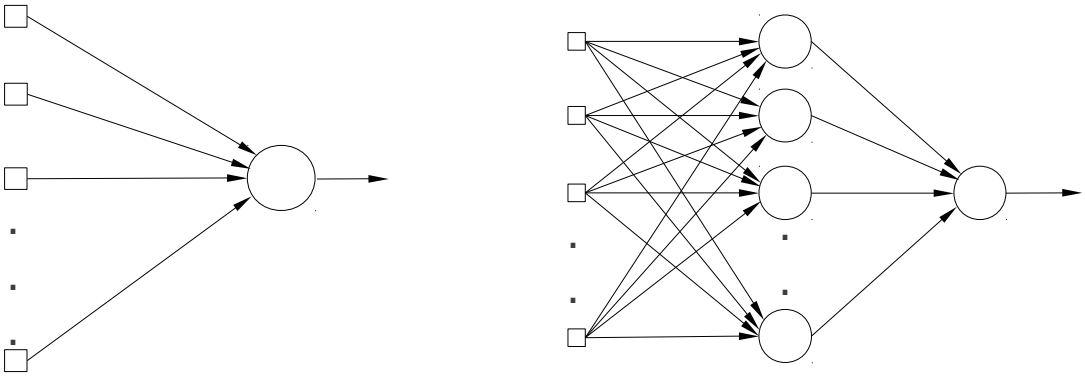
- Número de camadas;
- Tipo de conexões entre os neurônios;

- Tipo de conectividade.

Os neurônios em uma rede neural artificial são dispostos em camadas. As camadas que compõem uma rede neural são:

- camada de entrada: recebe as informações do ambiente externo;
- camada oculta: onde ocorre o processamento da função de ativação. O resultado da função de ativação é transmitido para as próximas camadas;
- camada de saída: onde são apresentados os resultados do processamento das camadas ocultas.

Na figura 3.35 são apresentadas duas redes neurais artificiais, sendo a figura 3.35 (a) representando uma rede neural artificial com duas camadas e a figura 3.35 (b) uma rede neural artificial com três camadas.



a) b)

Figura 3.35: a) Rede neural artificial de duas camadas b) rede neural artificial com três camadas

Uma rede neural artificial pode ser formada apenas por uma camada de entrada e outra de saída. No entanto, a maioria das implementações de redes neurais artificiais utilizam no mínimo uma camada oculta.

O tipo de conexão da rede neural artificial determina a maneira como os sinais são transmitidos entre as camadas, sendo *feedforward* e *feedbackward* os dois tipos de conexões. No primeiro tipo, o sinal é propagado desde a camada de entrada até a camada de saída, sendo que o sinal de saída de um neurônio não é utilizado como entrada para os neurônios das camadas anteriores. Já no segundo tipo de conexão, o sinal de saída de um neurônio pode ser utilizado como sinal de entrada para um neurônio em uma camada anterior.

O tipo de conectividade determina a maneira como ocorrem as conexões entre os neurônios de cada camada e estas podem ser conexões completas ou parciais. Nas conexões completas, cada neurônio de uma camada está conectado a todos os neurônios da camada

posterior, o que não ocorre na conexão parcial.

O estabelecimento de qual topologia utilizar, a quantidade de camadas e o número de neurônios em cada camada depende muito do problema. A escolha inadequada dos parâmetros da rede pode torná-la ineficiente. Além disso, o tempo de treinamento da rede neural artificial pode ser extenso dependendo do número de camadas e da quantidade de informações a serem processadas. No entanto, um pequeno número de camadas e neurônios pode fazer com que o treinamento convirja rapidamente para uma solução, sendo que para novos padrões o resultado da rede pode ser insatisfatório.

3.2.2 Aprendizado

O conhecimento da rede neural artificial está concentrado nos valores dos pesos das conexões sinápticas entre os neurônios. O aprendizado de uma rede neural artificial é o processo de reajuste dos pesos de suas conexões sinápticas de maneira a melhorar o desempenho de classificação para que se possa obter a saída desejada.

Um dos primeiros métodos de aprendizado de RNAs foi proposto por Hebb em 1949 e sua teoria diz que o peso de uma conexão sináptica deve ser reajustado se os neurônios pré e pós sinápticos desta conexão forem ativados sincronicamente, ou seja, simultaneamente . A formulação matemática do aprendizado hebbiano é apresentada na equação 3.45.

$$\Delta w_{ij}(t) = \eta x_i(t) x_j(t) \qquad 3.45$$

Na equação 3.45, η é a taxa de aprendizado, $x_i(t)$ e $x_j(t)$ são os níveis de ativação dos neurônios pré pós sinápticos, respectivamente, no instante de tempo t e w_{ij} é o peso da conexão entre os neurônios x_i e x_j. Portanto, quanto menor o nível de ativação de um neurônio, seja este pré ou pós-sináptico, menor será a influência do reajuste do peso da conexão sináptica.

Outra abordagem utilizada para o aprendizado de redes neurais artificiais consiste na aproximação entre a saída calculada pela rede e a saída desejada informada por um supervisor. A abordagem utilizada para esta aproximação é a aprendizagem por correção de erro. Sendo $d_i(t)$ e $r_j(t)$ a saída desejada e o resultado calculado pela rede neural artificial no instante *t*, respectivamente, para o neurônio de saída j, o sinal de erro será representado mediante a equação 3.46.

$$e_j(t) = d_j(t) - r_j(t) \qquad 3.46$$

O sinal de erro é aplicado no ajuste dos pesos das conexões que incidem sobre o

neurônio de saída j para aproximar sua saída à resposta desejada. Seja $w_{ij}(t)$ o peso sináptico entre um neurônio de entrada x_i e o neurônio de saída x_j. A correção do valor do peso w_{ij} no instante de tempo t pode ser feita mediante equação 3.47, referenciada como regra de aprendizagem delta.

$$w_{ij}(t+1) = w_{ij}(t) + \eta\, e_j(t)\, x_i(t) \qquad 3.47$$

Na equação 3.47, *n* é a taxa de aprendizado, $x_i(t)$ é o valor de ativação do neurônio da camada anterior no instante t e $w_{ij}(t)$ é o peso atual no instante t associado à conexão sináptica entre os neurônios x_i e x_j. Esta última abordagem é uma das mais utilizadas no aprendizado das redes neurais artificiais.

3.2.3 Redes neurais artificiais Perceptron de múltiplas camadas

Uma das topologias de redes neurais artificiais mais utilizadas na solução de problemas complexos é a de múltiplas camadas com conexões *feedforward*. Redes neurais artificiais com essa característica são denominadas perceptrons de múltiplas camadas (MLP - *Multilayer Perceptron*) e representam uma variação das redes neurais perceptron de uma única camada. O *Perceptron* de múltiplas camadas é um modelo projetado para solucionar os problemas não separáveis linearmente, isto é, problemas os quais não são possíveis de serem representados em duas classes separadas por uma reta em um plano. Este problema ocorre quando há padrões de entrada similares que resultam em diferentes valores. O exemplo clássico de problema não separável linearmente é a operação lógica XOR (ou exclusivo), a qual não é solucionada por uma rede neural artificial *Perceptron* composta de uma camada.

Redes neurais artificias MLP utilizam uma abordagem de aprendizado a qual é baseada na regra delta e conhecida como algoritmo de retropropagação de erro (algoritmo *backpropagation*). O aprendizado por retropropagação é realizado em dois passos: o primeiro passo consiste em propagar os sinais de entrada por meio da rede camada por camada, iniciando na camada de entrada e finalizando na camada de saída. O segundo passo consiste em reajustar os pesos de todas as conexões sinápticas por meio da propagação do sinal de erro calculado entre as saídas calculadas e desejadas para cada neurônio da camada de saída. O sinal de erro é propagado da camada de saída até a camada de entrada – daí o nome retropropagação – sendo o ajuste dos pesos realizado em cada camada. Na figura 3.36 é mostrado um perceptron de múltiplas camadas, sendo composto pela camada de entrada, duas camadas ocultas e uma camada de saída.

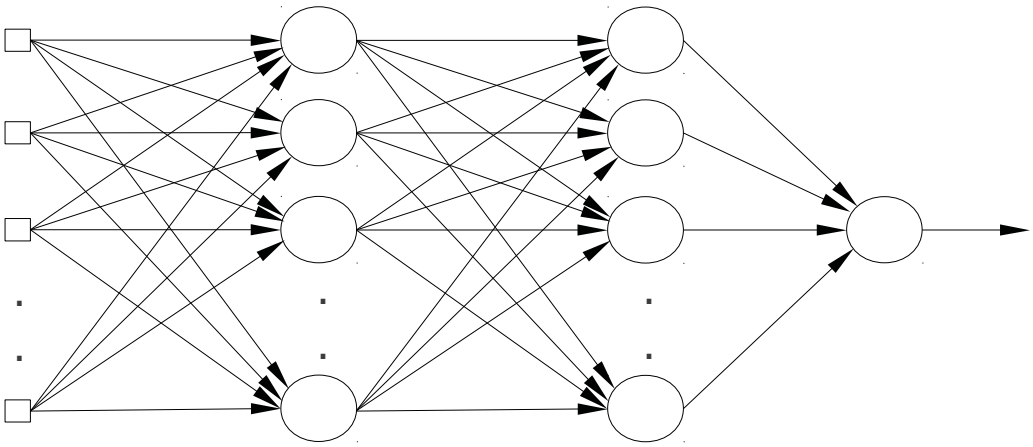


Figura 3.36: Perceptron de múltiplas camadas

A forma geral de representação do algoritmo por retropropagação para a correção do peso da conexão sináptica entre o neurônio x_i e x_j é descrita na equação 3.48 (18).

$$\begin{pmatrix} Novo\ de\ peso \\ \Delta w_{ij}(t+1) \end{pmatrix} = \begin{pmatrix} Taxa\ de\ aprendizagem \\ \eta \end{pmatrix} \cdot \begin{pmatrix} Gradiente \\ \delta_j(t) \end{pmatrix} \cdot \begin{pmatrix} Sinal\ de\ entrada \\ y_i(t) \end{pmatrix} \quad 3.48$$

Na equação 3.48, o sinal de entrada $y_i(t)$ representa o resultado da função de ativação sobre o neurônio x_i, no instante de tempo t, da camada anterior ao do neurônio x_j, conforme foi demonstrado na equação 3.40. O gradiente do neurônio j aponta para as modificações necessárias nos pesos das conexões e é dependente da função de ativação e da camada onde é aplicado o ajuste dos pesos. Para um neurônio na camada de saída, o gradiente é representado pelo produto do erro médio com a derivada da função de ativação, conforme demonstrado na equação 3.49 (18).

$$\delta_j = e_j(t)\varphi'(v_j(t)) \quad 3.49$$

Na equação 3.50 é demonstrado o resultado da substituição do gradiente na equação 3.49, sendo esta equação utilizada para atualização dos pesos das conexões sinápticas dos neurônios na camada de saída.

$$\Delta w_{ij}(t+1) = \eta \cdot e_j(t) \cdot \varphi'(v_j(t)) \cdot y_i(t) \quad 3.50$$

Para um neurônio na camada oculta, o gradiente é definido como o produto da derivada da função de ativação pela soma ponderada dos gradientes gerados nas camadas sucessoras, sejam estas outras camadas ocultas ou uma camada saída. A equação 3.51 demostra o gradiente para os neurônios pertencentes às camadas intermediárias.

$$\delta_j = \varphi'(v_j(t)) \cdot \sum_{1}^{k} \delta_k(t) w_{jk}(t) \quad 3.51$$

Na equação 3.51, k representa o número de gradientes calculados na camada posterior. Substituindo esse resultado na equação 3.48, obtemos a descrição do ajuste dos pesos na

camada oculta mediante a equação 3.52.

$$\Delta w_{ij}(t+1) = \eta \cdot \varphi'(v_j(t)) \cdot \sum_{1}^{k} \delta_k(t) w_{jk}(t) \cdot y_i(t) \qquad 3.52$$

Para a definição final do gradiente, é necessário o conhecimento da função de ativação para descrever a sua derivada φ'. A função de ativação precisa ser contínua para que sua derivada possa ser descrita. Um exemplo de função contínua e diferenciável, amplamente utilizada nos perceptrons de múltiplas camadas, é a função sigmóide logística, descrita na equação 3.44. A derivada desta função é descrita na equação 3.53.

$$\varphi'(v_j(t)) = a y_j(t)[1 - y_j(t)] \qquad 3.53$$

Na equação 3.53, $y_j(t)$ representa o resultado da função de ativação, ou seja, $y_j(n) = \varphi(v_j(n))$. Sendo $y_j(n)$ o sinal de ativação de um neurônio da camada de saída, então $y_j(n) = o_j(n)$. Substituindo a derivada φ' em 3.50, para um neurônio na camada de saída, resulta na equação 3.54.

$$\Delta w_{ij}(t+1) = \eta \cdot a[d_j(t) - o_j(t)] \cdot o_j(t) \cdot [1 - o_j(t)] \cdot y_i(t) \qquad 3.54$$

Na equação 3.54, $y_i(n)$ é o sinal de ativação do neurônio da camada anterior a da saída. Para o caso do neurônio pertencer à camada oculta, a derivada da função de ativação φ', substituída na equação 3.52, resulta na equação 3.55.

$$\Delta w_{ij}(t+1) = \eta \cdot a y_j(t)[1 - y_j(t)] \cdot \sum_{1}^{k} \delta_k(t) w_{jk}(t) \cdot y_i(t) \qquad 3.55$$

O treinamento de uma rede perceptron de múltiplas camadas pode ser representado por um gráfico de superfície de erro, onde os pontos altos do gráfico representam um maior erro produzido pela rede. O ponto mais baixo no gráfico representa o mínimo global, o qual é o ponto que representa a solução ótima de pesos para MLP (16). Na figura 3.37 é ilustrada a superfície de erro do treinamento de redes MLP.

A taxa de aprendizado é um parâmetro que influencia na velocidade de treinamento da rede neural artificial, sendo um fator determinante para o ajuste correto dos pesos das conexões sinápticas. Seu propósito é estabelecer o caminho partindo de um ponto inicial na superfície de erro até o mínimo global. Entretanto, um valor baixo para a taxa de aprendizado pode ocasionar um tempo alto de treinamento, além da possibilidade da solução encontrada pertencer a pontos denominados mínimos locais. Um mínimo local é um ponto que representa uma solução parcial, sendo representado no gráfico de superfície de erro por pequenos vales. Com uma taxa de aprendizado baixa, o treinamento pode convergir rapidamente para um mínimo local, o qual não representa a solução ideal para rede.

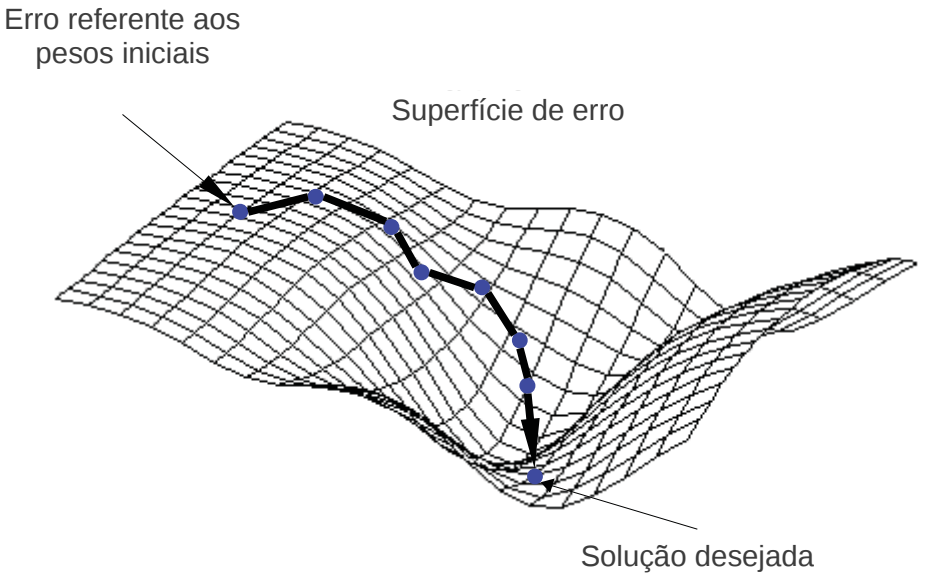


Figura 3.37: Superfície de erro do treinamento da rede MLP (16)

Entretanto, uma taxa de aprendizado muito alta pode ocasionar grandes oscilações no caminho seguido até o mínimo global, podendo isto fazer com que a solução ideal nunca seja alcançada. Uma abordagem utilizada para solucionar esta dificuldade consiste na determinação de um parâmetro denominado taxa de momento, o qual permite aumentar a velocidade do aprendizado da rede e evitar oscilações e instabilidade (16). A equação 3.56 representa a regra delta modificada com a inclusão da taxa de momento.

$$\Delta w_{ij}(t+1) = \alpha \Delta w_{ij}(t-1) + \eta \delta_j(t) y_i(t) \qquad 3.56$$

Na equação 3.56, α é a taxa de momento e $\Delta w_{ij}(t-1)$ é o valor de ajuste de peso na propagação anterior. Portanto, a taxa de momento baseia-se no ajuste do peso na propagação anterior para manter o caminho até o mínimo global sem grandes oscilações, mesmo com uma taxa de aprendizado alta. O parâmetro α é o que determina o tamanho do passo entre cada ponto encontrado no caminho até o mínimo global. Na figura 3.38 é mostrada a superfície do erro e dois caminhos até o mínimo global, sendo um sem a constante de momento α e outro com a constante de momento α. É possível notar na figura 3.38 que o caminho seguido utilizando a constante de momento não possui oscilações em sua trajetória e se mantém uniforme até a solução desejada (mínimo global).

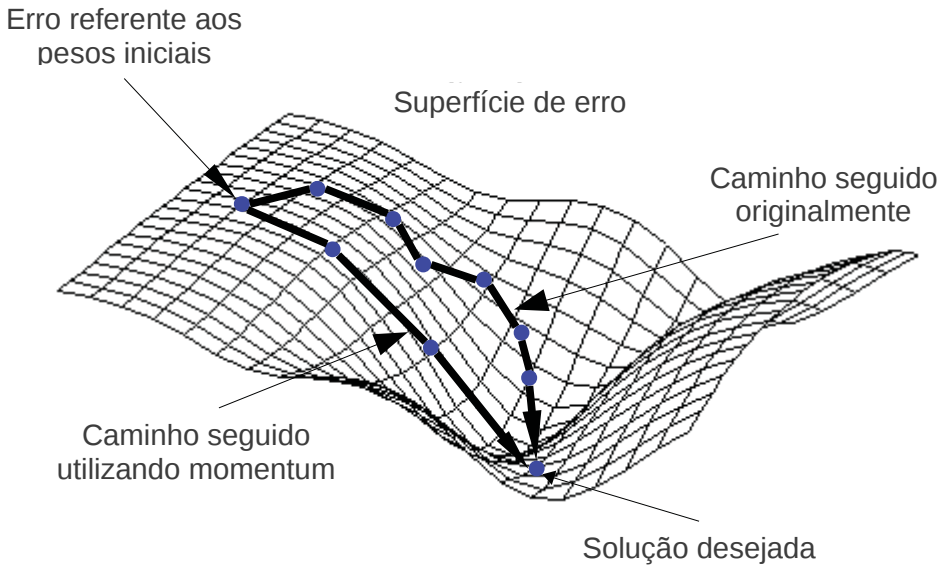


Figura 3.38: Superfície de erro mostrando um caminho até o mínimo global com a constante de momento (16)

3.3 Máquinas de Vetores de Suporte

Máquinas de Vetores de Suporte (SVM – *Support Vector Machine*) são redes neurais artificiais que permitem a classificação de padrões em duas classes separadas por um hiperplano de decisão. O hiperplano de decisão é uma superfície que separa os dados de entrada em duas classes, como mostrado na figura 3.39.

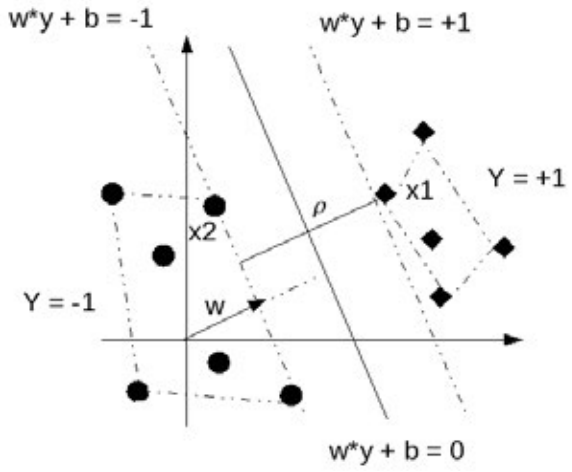


Figura 3.39: Hiperplano de separação de dados em duas classes (Adaptado de (19))

O objetivo de uma máquina de vetores de suporte é encontrar um hiperplano de decisão ótimo o qual mantém uma separação máxima entre os exemplos (18). A construção de

um hiperplano ótimo como superfície de decisão é fundamental para que a separação entre os exemplos seja máxima, sendo esta tarefa realizada na etapa de treinamento da SVM.

Seja uma amostra de treinamento representada por $\{(x_i, d_i)\}_{(i=1)}^{N}$, onde x_i representa o i-ésimo padrão de entrada e d_i representa a saída desejada $\in$ {-1,1} para o i-ésimo padrão de entrada. Considerando que a amostra de treinamento é linearmente separável, a superfície de decisão é representada mediante a equação 3.57.

$$w^T x + b = 0 \tag{3.57}$$

Na equação 3.57, x representa um vetor de entrada, w um vetor de pesos e b representa o *bias*. O uso desta equação gera uma reta que representa a separação entre as duas classes , sendo expressada mediante a equação 3.58.

$$\begin{aligned} w^T x + b \geqslant 0 \; para \; d_i = +1 \\ w^T x + b < 0 \; para \; d_i = -1 \end{aligned} \tag{3.58}$$

O objetivo do uso da SVM é encontrar os valores w_o e b_o que representam os valores ótimos do vetor de peso e do *bias*, respectivamente, para gerar um hiperplano de decisão ótimo onde a separação ρ entre as classes é máxima. Substituindo w_o e b_o na equação 3.57, obtém-se a equação 3.59.

$$w_o^T x_i + b_o = 0 \tag{3.59}$$

Na equação 3.59, b_o é o deslocamento do hiperplano ótimo sobre o eixo y, sendo que se $b_o > 0$ então a origem do hiperplano está no lado positivo; se $b_o < 0$, a origem está no lado negativo do eixo y; se $b_o = 0$, o hiperplano ótimo passa pela origem do plano. Diante disso, o hiperplano ótimo deve satisfazer as seguintes restrições para (w_o, b_o) (18):

$$\begin{aligned} w_o^T x + b_o \geqslant 1 \; para \; d_i = +1 \\ w_o^T x + b_o \leqslant -1 \; para \; d_i = -1 \end{aligned} \tag{3.60}$$

A restrição imposta pela equação 3.60 garante que não há dados entre os hiperplanos $w_o^T x_i + b_o = 0$ e $w_o^T x_i + b_o = \pm 1$. Os pontos que satisfazem a igualdade para a primeira ou segunda linha da equação 3.60 são denominados vetores de suporte. Os vetores de suporte são os pontos que se encontram próximos da superfície de decisão e têm grande influência na localização desta superfície (18). Na figura 3.39, os vetores de suporte são os dois pontos da classe círculo que estão sobre o hiperplano $w^T x + b = -1$ e o ponto da classe losango sobre o hiperplano $w^T x + b = +1$. Combinando as linhas da equação 3.60 e substituindo w_o por w, obtém-se a equação 3.61.

$$d_i (w^T x_i + b) \geqslant 1 \tag{3.61}$$

O valor ótimo da margem de separação ρ entre as classes é dado pela equação 3.62 e maximizá-lo é equivalente a minimizar a norma euclidiana do vetor w, representado na figura 3.39 pelo vetor com origem no ponto (0,0) do plano.

$$\rho = \frac{2}{\|w_o\|} \tag{3.62}$$

O problema consiste em determinar os valores ótimos para w_o e b_o de maneira que satisfaçam a restrição imposta pela equação 3.61 e, ao mesmo tempo, minimize a norma do vetor w. Isto é um problema de otimização quadrática que pode ser resolvido pela teoria dos multiplicadores de Lagrange e consiste em minimizar w e b na equação 3.63.

$$J(w,b,\alpha) = \frac{1}{2} w^T w + \sum_{i=1}^{N} \alpha_i [d_i (w^T x_i + b) - 1] \tag{3.63}$$

Na equação 3.63, o parâmetro α_i representa o multiplicador de Lagrange e o problema passa a ser a minimização de w e b e a maximização dos parâmetros α_i (18). A maximização consiste em encontrar os multiplicadores de Lagrange $\{a_i\}_{(i=1)}^{N}$ que maximizam a função objetivo dada pela equação 3.64.

$$Q(\alpha) = \sum_{i=1}^{N} \alpha - \frac{1}{2} \left(\sum_{i=1}^{N} \sum_{i=1}^{N} \alpha_i \alpha_j d_i d_j x_i^T x_j \right) \tag{3.64}$$

As seguintes restrições são impostas para a equação 3.64:

1. $\sum_{i=1}^{N} \alpha_i d_i = 0$

2. $\alpha \geqslant 0 \; para \; i = 1, 2, \ldots, N$

Tendo encontrado os multiplicadores de Lagrange, o cálculo dos valores ótimos de w_o e b_o são realizados mediante as equações 3.65 e 3.66.

$$w_o = \sum_{i=1}^{N} \alpha_{o,i} d_i x_i \tag{3.65}$$

$$b_o = 1 - w_o^T x^{(s)} \quad para \; d^{(s)} = 1 \tag{3.66}$$

No entanto, é comum nos depararmos com problemas não separáveis linearmente e nesta situação é necessário o mapeamento do conjunto de treinamento, não linear, para um espaço de separação linear, de maneira que seja possível determinar um hiperplano de decisão ótimo. O espaço linearmente separável é denominado espaço de características.

O mapeamento não linear é uma função Φ que transforma os exemplos de treinamento x de um conjunto $S = \{(x_i, d_i)\}_{(i=1)}^{N}$ para o espaço de características, sendo esta transformação

representada por $\{\Phi_j(x)\}_{(j=1)}^{(m_1)}$, onde o parâmetro m_1 representa a dimensão do espaço de características. Uma função *Kernel* mapeia os vetores de entrada (x,y) do espaço não linear para o espaço de características e é representada pela equação 3.67.

$$K(x, y)=\Phi(x)\cdot\Phi(y) \qquad 3.67$$

Na tabela 3.1 são exibidas algumas funções de *Kernel* utilizadas na construção de SVMs.

Tabela 3.1: Núcleos do produto interno utilizados em SVM

Tipo	**Função do núcleo**
Núcleo de função de base radial	$\exp\left(\frac{-1}{2\sigma^2}\|x-y\|^2\right)$
Histogram Intersection Kernel	$\sum min(x, y)$
Hardware Friendly Kernel	$2^{-\Upsilon\|x-y\|_1}$

A última função de *kernel* apresentada na tabela 3.1, denominada *Hardware Friendly Kernel* (HFK), foi proposta por Anguita et al. (20) para reduzir a complexidade de implementação em *hardware* de funções de *kern*el de SVM. O uso deste *kernel* é mais apropriado para implementação em *hardware* do que o *kernel* RBF, pois evita o cálculo de divisões e exponenciais. Desta maneira, o *kernel Hardware Friendly* torna-se mais rápido no treinamento e no reconhecimento de padrões. O parâmetro Υ é um valor inteiro definido como uma potência de dois, isto é, $\Upsilon= 2^{\pm p}$, com p = 0, 1, 2,...,Z. A norma da distância entre os exemplos de entrada x e os vetores de suporte y é representada como $||x – y||$. O uso da norma ao invés da distância euclidiana evita o cálculo da raiz quadrada e da exponencial, tornando a implementação em *hardware* menos complexa.

O funcionamento da máquina de vetores de suporte pode ser visto na figura 3.40. As etapas de funcionamento da máquina de vetores de suporte são descritas a seguir:

- O vetor de entrada x e os vetores de suporte x_i são mapeados para o espaço de características linearmente separável;
- O núcleo $k(x,x_i)$ realiza o produto interno de cada valor do vetor de entrada pelos valores do vetor de suporte, ambos mapeados no espaço de características;
- O resultado do produto interno é multiplicado pelo seu correspondente peso v_i, sendo a saída da SVM a soma ponderada dos pesos v_i pelos produtos internos $k(x,x_i)$.

Os pesos internos v_i são resultados do processo de otimização quadrática, sendo a equação 3.68 utilizada para calculá-los.

$$v_i = \sum_{i=1}^{N} \alpha_{o,i} d_i \Phi(x_i) \qquad 3.68$$

onde α é o valor ótimo do multiplicador de Lagrange.

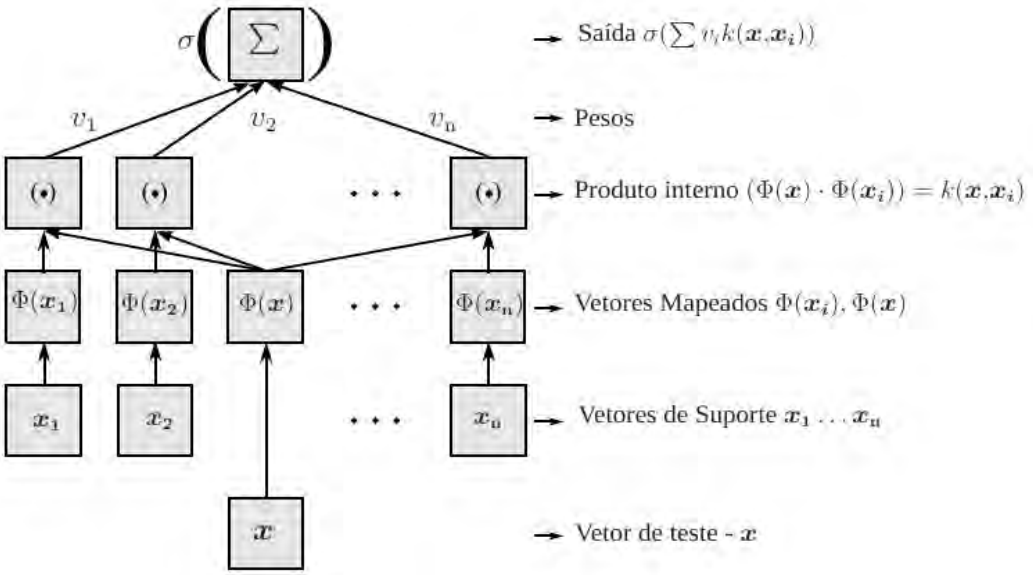


Figura 3.40: Exemplo de uma máquina de vetor de suporte (Adaptado de (19))

3.4 Tópicos de sistemas digitais

Nesta seção são apresentados os conceitos sobre dispositivos FPGA, linguagens de descrição de *hardware* e máquinas de estados finitos, os quais foram utilizados para o desenvolvimento e síntese do algoritmo da SVM em *hardware*.

3.4.1 FPGA

Os processadores atuais são utilizados para executar uma ampla variedade de aplicações com um bom desempenho e um custo razoavelmente baixo. No entanto, esses processadores de propósitos gerais possuem uma arquitetura de *hardware* já pré-definida pelo fabricante, a qual é utilizada para executar variados tipos de aplicações. No entanto, a otimização da estrutura de circuitos lógicos para um algoritmo de propósito específico pode melhorar o desempenho de execução do programa.

Soluções alternativas podem ser utilizadas para contornar esse problema e uma delas é o uso de ASIC *(Application Specific Integrated Circuit),* ou seja, circuitos que são fabricados para a execução de aplicações específicas. Com isso, o desempenho na execução de um determinado algoritmo pode melhorar significativamente, já que as unidades lógicas funcionais são projetadas para este tipo de algoritmo. No entanto, os custos de projeto e implementação desses circuitos são elevados e só podem ser justificados no caso de produções em grandes quantidades. Outro problema relacionado a esses circuitos, o qual ainda permanece, é a inflexibilidade na reconfiguração do *hardware* para permitir sua adaptação a novas aplicações.

A computação reconfigurável surge como alternativa para o desenvolvimento de *hardware* dedicado a um determinado tipo de aplicação e que possa também ser reconfigurado com o objetivo de se adaptar às novas aplicações. A computação reconfigurável é a habilidade de adaptar a estrutura do *hardware* a um determinado tipo de aplicação. Essa flexibilidade proporciona um aumento significativo de desempenho e permite que a computação reconfigurável possa ser utilizada em aplicações de processamento de vídeos e imagens, pesquisas e ordenação de dados e aplicações de tempo real. A implementação de um algoritmo específico em *hardware* possibilita mais paralelismo do que em processadores convencionais, maior velocidade de processamento, menor área para implementação de circuitos lógicos e menor consumo de energia (21).

O desempenho alcançado com a computação reconfigurável tem incentivado a implementação de novas arquiteturas projetadas para otimização de algoritmos de processamento de imagens e melhorias de desempenho na execução de algoritmos para a robótica móvel.

Dentre os dispositivos reconfiguráveis existentes no mercado destacam-se os FPGAs (*Field Programmable Gate Arrays*), os quais são dispositivos semicondutores que podem ser programados após a fabricação. Os FPGAs surgiram em 1985 como uma alternativa para o desenvolvimento de lógica digital. Em vez de estar restrito a uma função de *hardware* pré-determinada, o projetista pode programar no FPGA funções lógicas para se adaptar a novos algoritmos. É possível utilizar um FPGA para implementar qualquer função lógica que um circuito integrado de aplicação específica (ASIC) poderia realizar, porém com a vantagem de poder atualizar a funcionalidade para se adaptar a um novo algoritmo. A descrição das funções lógicas do FPGA pode ser desenvolvida em uma linguagem de descrição de *hardware*, tal como VHDL e Verilog.

Atualmente, os dispositivos FPGAs são compostos por memória SRAM embutida configurável para o armazenamento da configuração do FPGA, transmissores/receptores de alta velocidade, blocos de entrada e saída de alta velocidade e blocos lógicos reconfiguráveis. Especificamente, um FPGA contém componentes lógicos programáveis chamados blocos lógicos e uma hierarquia de interconexões reconfiguráveis que permite que estes blocos lógicos estejam fisicamente conectados. Pode-se configurar elementos lógicos para executar funções combinacionais complexas ou apenas portas lógicas simples, como AND e XOR.

Dispositivos FPGA podem ser utilizados na depuração projetos de *hardware* antes de serem produzidos em silício. De fato, o caráter reprogramável dos FPGAs permite a atualização e a modificação de projetos de *hardware* em menos tempo que o necessário para criar um novo *chip* em silício, o que evita custos adicionais de fabricação em caso de falhas.

3.4.1.1 Arquitetura

Na figura 3.41 é mostrada a arquitetura de um dispositivo FPGA (22).

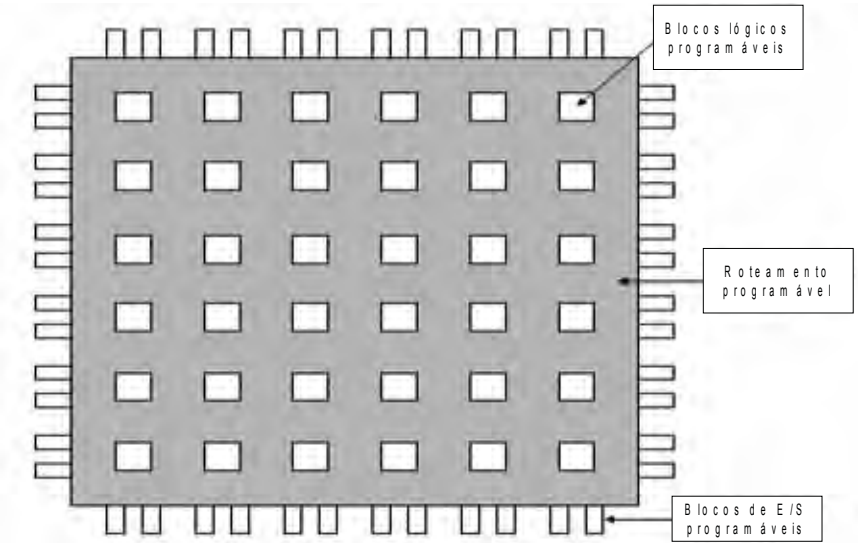


Figura 3.41: Arquitetura de um dispositivo FPGA (Adaptado de Gokhale e Graham (22))

Um FGPA é composto dos seguintes elementos:

- ***Blocos lógicos programáveis:*** Contêm *flip-flops* e lógica combinacional que permite a construção de funções lógicas;
- ***Blocos de entrada e saída:*** São pinos de comunicação que realizam a interface de entrada e saída entre os blocos lógicos e barramentos de transmissão de dados;

- ***Roteamento programável:*** São trilhas de comunicação entre os blocos lógicos reconfiguráveis.

Fisicamente, um FPGA é uma matriz composta de blocos lógicos programáveis que são interligados por vias de comunicação. Os blocos lógicos são circuitos que contêm a representação da função lógica desejada. Isto é feito utilizando LUT (*Look-Up Table*), as quais são tabelas de busca utilizadas para gerar uma saída baseada nos sinais de entrada que são recebidos pelos blocos lógicos. A figura 3.42 ilustra a representação de uma LUT de quatro entradas.

Figura 3.42: Representação de um LUT de quatro entradas

A LUT é uma memória do tipo SRAM que armazena até 2^n células de *bits*, sendo n o número de entradas da LUT. As células armazenam os resultados da função lógica binária sobre os valores de entrada. Na figura 3.43 é ilustrada a representação de uma LUT que implementa a função lógica AND. As 16 células de bits da LUT armazenam os resultados das 4 entradas. A célula lógica 15 contém o resultado para o padrão de entrada “1111”.

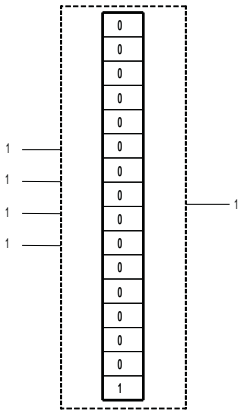


Figura 3.43: *Look-up table* de 4 entradas com 16 células de bits (23)

Para permitir a configuração de arranjos complexos de portas lógicas, são utilizados milhares de blocos lógicos conectados por roteamento programável, os quais definem o fluxo de sinais durante a execução de uma configuração no FPGA. No entanto, um dispositivo FPGA pode ser utilizado para incorporar circuitos mais complexos ou mesmo ser projetado

para ser um processador, como é o caso do FPGA da família Virtex-4 FX, o qual implementa dois núcleos de processadores PowerPC de 32 bits. Os novos dispositivos FPGA vêm com melhorias em várias características, tais como o aumento do número de blocos lógicos e memória RAM, maiores taxas de conectividade serial, menor consumo de energia e conectividade para redes Ethernet. Nos dispositivos FPGA da família Virtex 6 da Xilinx, por exemplo, o número de células lógicas variam entre 74.496 à 758.784. As *look-up tables* podem ser configuradas com 6 entradas e uma saída ou separadas em duas *look-up tables* de 5 entradas cada uma tendo uma saída. Os blocos de memória RAM possuem 36 Kb, sendo que o número varia entre 156 e 1.064. No entanto, segundo a especificação do fabricante, cada bloco de memória pode ser configurado como dois blocos independentes de 18 Kb. Blocos dedicados para processamento digital de sinais estão disponíveis nestes dispositivos, sendo que o número varia entre 288 à 2.016. Blocos de controle de comunicação de redes Ethernet de 10/100/1000 Mb/s também estão disponíveis nestes dispositivos (24).

3.4.2 Descrição do *hardware*

O desenvolvimento dos circuitos lógicos para um dispositivo FPGA pode ser realizado utilizando uma linguagem de descrição de *hardware* (derivada da sigla inglês HDL - *Hardware Description Language*). As linguagens de descrição de *hardware* foram desenvolvidas com o objetivo de auxiliar os projetistas na documentação e descrição do *hardware* utilizando uma linguagem de descrição. A utilização de uma HDL permite ao projetista descrever a seleção, instanciação e a interconexão dos módulos de *hardware*. Além disso, o projetista pode especificar o paralelismo em nível de instrução dentro dos circuitos por meio da criação de unidades de *pipeline* (22).

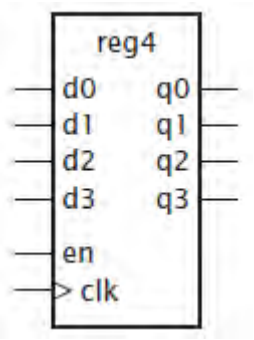
O processo de descrição e síntese de um circuito é dividido em quatro etapas:

- ***Descrição do projeto:*** Nesta etapa o projetista descreve o *hardware* específico para uma aplicação utilizando as linguagens VHDL ou Verilog, ou por meio da utilização de diagramas esquemáticos;
- ***Síntese:*** Nesta etapa, o código em HDL ou o diagrama esquemático é representado por meio de portas lógicas e suas interconexões, sendo que o arquivo utilizado para descrever esta estrutura é denominado *netlist;*
- ***Implementação:*** Nesta etapa, é realizado o mapeamento da estrutura dos circuitos

lógicos do arquivo *netlist* para a estrutura física do dispositivo FPGA alvo. Gokhale e Graham (22) denominaram esta etapa de mapeamento tecnológico. De acordo com os autores, nesta etapa são realizadas otimizações para eliminar redundâncias lógicas e simplificar a estrutura dos circuitos;

- ***Geração do arquivo bitstream:*** A última etapa de configuração do FPGA é a geração do arquivo binário que será utilizado para determinar o fluxo de sinais entre os blocos lógicos do dispositivo. O arquivo é carregado no FPGA e contém a funcionalidade descrita pelo projetista.

Dentre as linguagens de descrição de *hardware* existentes, destaca-se a VHSIC-HDL (*Very High Speed Integrated Circuits Hardware Description Language*), ou simplesmente VHDL, desenvolvida pelo departamento de defesa dos Estados Unidos. A VHDL foi padronizada pela IEEE e atualmente está entre as mais utilizadas para descrição de *hardware.* Na figura 3.44 é apresentado um exemplo de descrição de um registrador de 6 entradas e quatro saídas binárias. Os fabricantes disponibilizam suas próprias ferramentas de desenvolvimento e síntese de circuitos lógicos em FPGAs, como o Quartus II da Altera e o ISE (*Integrated Software Environment*) da Xilinx.



a)

```
entity reg4 is
        port ( d0, d1, d2, d3, en, clk : in bit;
q0, q1, q2, q3 : out bit );
end entity reg4;
```

b)

Figura 3.44: a) Representação do registrador b) Descrição do registrador em VHDL (25)

3.4.3 Dispositivo FPGA Altera Stratix II

Os dispositivos FPGA da série Stratix estão entre os diversos modelos de FPGA comercializados pela Altera Corporation. A série Stratix possui cinco gerações de dispositivos, cada uma tendo suas próprias características como tecnologia de processo, número de pinos de entrada e saída, frequência em Mhz, número de elementos lógicos, número de blocos DSP (*Digital Signal Processing*) e número de bits e de blocos de memória RAM. Dentre as cinco

gerações de dispositivos da série, neste trabalho é descrita apenas a segunda geração, Stratix II, já que este foi o dispositivo utilizado para síntese do circuito implementado em VHDL.

3.4.3.1 Arquitetura

O dispositivo FPGA Stratix II, pertencente à segunda geração da série de dispositivos Stratix, utiliza uma tecnologia de processo de 90 nm. O número de elementos lógicos suportados por esta série varia de 15.600 à 179.400 dependendo do modelo utilizado. O número de bits dos blocos de memória RAM internos varia de 419.328 à 9.383.040. Dispositivos desta geração contêm entre 12 e 96 blocos de processamento digital de sinais (26). Na tabela 3.2 são apresentadas as características de cada modelo desta geração.

Tabela 3.2: Características dos modelos de dispositivos Stratix II (26)

Característica	EP2S15	EP2S30	EP2S60	EP2S90	EP2S130	EP2S180
ALMs	6.240	13.552	24.176	36.384	53.016	71.760
Look-up tables adaptativas	12.480	27.104	48.352	72.768	106.032	143.520
Elementos lógicos	15.600	33.880	60.440	90.960	132.540	179.400
Blocos RAM M512	104	202	329	488	699	930
Blocos RAM M4K	78	144	255	408	609	768
Blocos M-RAM	0	1	2	4	6	9
Total RAM bits	419.328	1.369.728	2.544.192	4.520.488	6.747.840	9.383.040
Blocos DSP	12	16	36	48	63	96
Multiplicadores 18-bit × 18-bit	48	64	144	192	252	384
Enhanced PLLs	2	2	4	4	4	4
Fast PLLs	4	4	8	8	8	8
Pinos de entrada e saída	366	500	718	902	1.126	1.170

O dispositivo Stratix II contém uma arquitetura bidimensional baseada em linhas e colunas para interconexão entre os blocos de arranjos lógicos (LAB), os blocos de memória RAM, blocos de processamento de sinais (DSP) e elementos de entrada e saída (OLE). Na figura 3.45 é mostrada a arquitetura bidimensional do dispositivo Stratix II.

Os blocos de arranjos lógicos (LAB) são, basicamente, compostos por oito módulos

lógicos adaptativos, interconexões locais e sinais de controle entre os módulos lógicos adaptativos. Os módulos lógicos adaptativos (ALM) surgiram na segunda geração de dispositivos da série Stratix e são blocos compostos por *lookup tables* de até oito entradas, dois circuitos somadores e dois registradores de saída. Os dispositivos Stratix da primeira geração implementam elementos lógicos com *lookup tables* de quatros entradas. Os elementos de entrada e saída (IOE), localizados nas extremidades da arquitetura bidimensional, capturam as entradas e saídas dos pinos do dispositivo. Cada elemento de entrada e saída é composto por um *buffer* bidirecional e seis registradores para os sinais de entrada, saída e habilitação de saída.

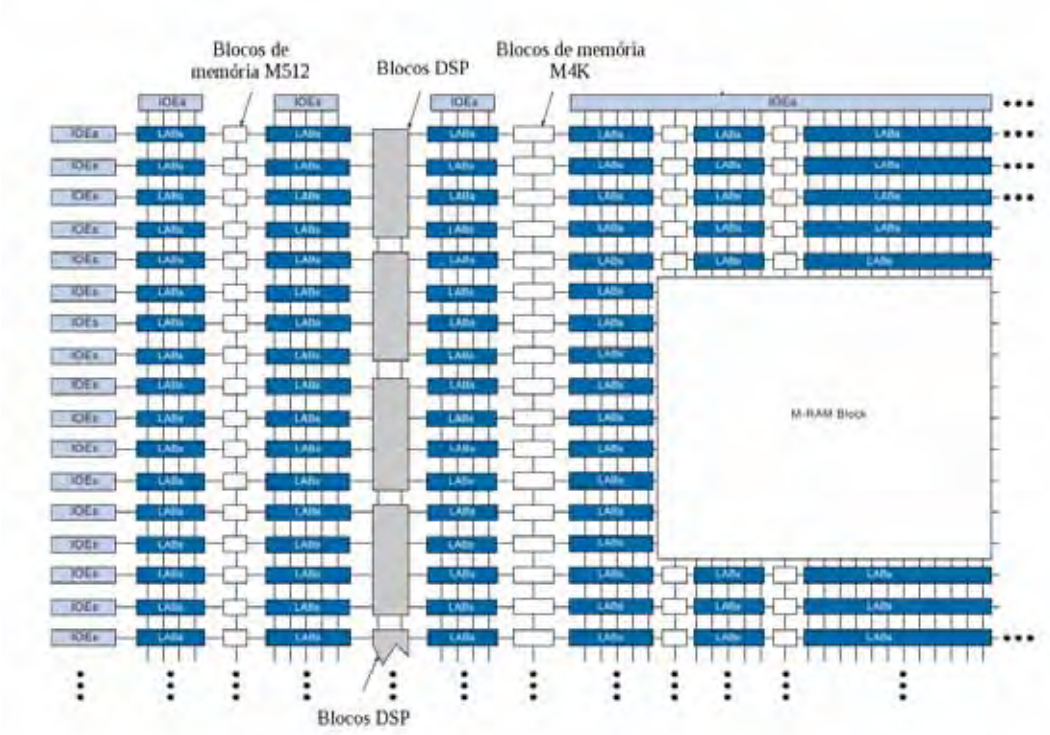


Figura 3.45: Arquitetura do dispositivo Stratix II (26)

A memória interna do dispositivo Stratix II é composta de três tipos de blocos: M512, M4K e M-RAM. Cada bloco oferece uma quantidade de bits e modos de operação diferentes. Dependendo do bloco utilizado, a memória pode ser configurada como RAM (*Random Access Memory*) ou ROM (*Read Only Memory*). Quando configurada como RAM, a memória pode ser utilizada tanto para leitura quando para a escrita de dados. Já no modo ROM, a memória pode ser acessada somente para leitura. Quando os blocos de memória são configurados como RAM ou ROM, um arquivo de inicialização pode ser utilizado para carregar a memória com conteúdo pré-definido na etapa de síntese do circuito. Os blocos de

memória a serem utilizados no FPGA podem ser determinados pela ferramenta de síntese ou na linguagem de descrição de *hardware* pelo projetista do circuito.

A memória interna, quando configurada como RAM ou ROM, pode assumir três modos de operação:

- *single-port*
- *simple dual-port*
- *true dual-port*

No modo de operação *single-port* a memória compartilha o mesmo endereço tanto para leitura quanto para escrita de dados. No modo *simple dual-port* são utilizadas duas portas de endereços, sendo uma utilizada para receber o endereço de leitura e a outra utilizada para receber o endereço de escrita. No modo *true dual-port* a memória possui duas portas para leitura ou escrita de dados. Desta maneira, é possível escrever ou ler dados utilizando dois endereços. Quando a operação é de leitura de dados, o dado fica disponível na porta de saída correspondente ao seu endereço (27). A memória, quando configurada como RAM, pode assumir os modos de operação *single-port, simple dual-port* e *true dual port*. Quando configurada como ROM, os modos de operação válidos são *single-port* e true *dual-port*.

Os blocos de memória M512 são utilizados no armazenamento de pequenas quantidades de dados e cada bloco contém 576 bits. O número de blocos de memória M512 varia de 104 a 930 dependendo do modelo utilizado. Os blocos M512 podem ser configurados nos modos:

- Memória RAM *simple dual-port;*
- Memória RAM *single-port;*
- Memória ROM.

Os blocos de memória M4K suportam o modo de operação *true dual-port* e são utilizados em circuitos que necessitam de grandes quantidades de dados. O número de blocos varia de 78 à 768 dependendo do modelo utilizado. Cada bloco M4K possui 4.608 bits e pode ser configurado nos modos:

- Memória RAM *true dual-port;*
- Memória RAM *simple dual-port;*

- Memória RAM *single-port;*
- Memória ROM.

Os blocos de memória M-RAM são aplicados em circuitos que necessitam de grandes volumes de dados. Cada bloco M-RAM possui 589.824 bits, sendo que o número de blocos é consideravelmente menor em relação aos blocos M512 e M4K, podendo até mesmo não serem suportados por um determinado modelo de dispositivo, como é o caso dos modelos EP2S15. Os modos de operação suportados pelos blocos M-RAM são:

- Memória RAM *true dual-port;*
- Memória RAM *simple dual-port;*
- Memória RAM *single-port.*

3.4.4 Máquinas de estados finitos

Uma máquina de estados finitos (MEF) é matematicamente definida como uma quíntupla, representada na equação 3.69.

$$M = \{Q, I, f, q_{0,} F\} \tag{3.69}$$

Onde:

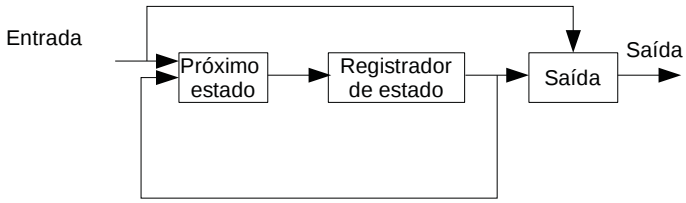
- Q representa o conjunto de estados
- I representa as entradas da MEF
- f representa a função de transição de estados
- q_0 representa o estado inicial da MEF
- F representa o conjunto de estados de parada da MEF

Uma máquina de estado finita começa a execução no estado inicial q0, sendo a transição entre os estados seguintes realizada pela função de transição f até o conjunto de estados de parada F.

Em circuitos digitais, uma máquina de estados finitos começa no estado inicial q0 no reset do sistema. A função de transição de estados é implementada por um circuito combinacional que tem como variáveis o estado atual (variável de estado) e a entrada (variáveis de entrada) para determinar o próximo estado da máquina, o qual pode ser atualizado na borda de subida ou descida do sinal de *clock*. A função de saída é implementada por um circuito combinacional no qual a saída da máquina é determinada pelo seu estado

atual e, possivelmente, pelas suas entradas. As máquinas de estados finitos podem ser divididas em duas classes: Moore e Mealy. Em uma máquina de estados finitos de Moore, a saída é determinada apenas pelo valor do estado atual. Entretanto, quando a saída depende tanto do estado atual quanto das entradas, então temos uma máquina de estados finitos de Mealy. Na figura 3.46 é apresentado um diagrama de blocos de uma máquina de estados finitos de Moore e Mealy.

a)



b)

Figura 3.46: Máquina de estados a) Moore b) Mealy (Adaptado de Grout (28))

4 Implementação do sistema de navegação

Neste capítulo é apresentado o funcionamento do sistema de navegação desenvolvido, bem como as etapas de sua implementação. Na figura 4.1 é mostrado o diagrama de funcionamento do sistema.

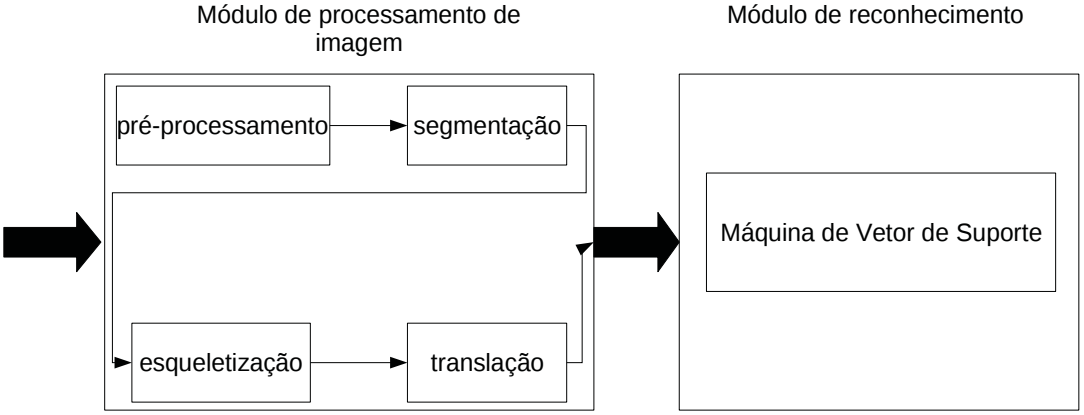


Figura 4.1: Diagrama de funcionamento do sistema

O módulo de processamento de imagens consiste na aplicação de técnicas para melhoria da qualidade da imagem e extração do caminho da plantação. No submódulo de pré-processamento é aplicado um filtro de suavização para a eliminação dos ruídos presentes na imagem. O submódulo de segmentação consiste em separar o caminho da área da plantação e realizar o processo de esqueletização sobre o caminho identificado. O submódulo de translação consiste em deslocar o caminho identificado para o centro da imagem com o objetivo de padronizar a representação dos dados para o módulo de reconhecimento.

O módulo de reconhecimento é composto por uma máquina de vetores de suporte, a qual recebe os *pixels* da imagem resultante do módulo de processamento de imagens e determina uma saída que representa o ângulo de direção para o padrão de *pixels* apresentado. O uso de uma máquina de vetores de suporte se deve ao baixo tempo de treinamento e capacidade de generalização semelhante ao perceptron de múltiplas camadas.

O sistema de navegação foi desenvolvido em *software* para realização de testes iniciais e posteriormente foi feita sua implementação em *hardware*. O desenvolvimento dos algoritmos de pré-processamento das imagens foi realizado na linguagem de programação C em conjunto com a biblioteca OpenCV (*Open Computer Vision Library*). O desenvolvimento do algoritmo da máquina de vetores de suporte também foi realizado na linguagem de programação C.

Para a realização de testes iniciais, formou-se um banco de imagens composto por imagens de plantações de amendoim e soja, as quais foram adquiridas por uma câmera digital Kodak, modelo M531, a uma resolução de 6 *Mega Pixels* e sob diversas condições de iluminação. O banco é composto por 1186 imagens, sendo 570 imagens de amendoim e 616 imagens de soja. Cada imagem foi redimensionada a uma resolução de 300x225 *pixels*, para diminuir o tempo de execução dos algoritmos de pré-processamento e segmentação.

4.1 Módulo de processamento de imagens

Nesta seção são apresentadas as técnicas utilizadas no módulo de processamento de imagens, as quais são utilizadas para a melhoria da qualidade e segmentação da imagem.

4.1.1 Pré-processamento

O pré-processamento da imagem é a etapa que consiste na aplicação de técnicas para a melhoria da qualidade da imagem. As imagens do banco foram submetidas a ruídos para a realização de testes com técnicas de suavização. A filtragem mediana foi utilizada para remover os ruídos presentes na imagem. A escolha deste filtro se deve aos melhores resultados obtidos comparados com a técnica de média da vizinhança. Na figura 4.2 (d) é visualizado o resultado da filtragem mediana com uma máscara 7x7 em uma imagem submetida a ruído. Conforme pode ser vistos nas figuras 4.2 (c) e 4.2 (d), a filtragem mediana foi mais eficiente tanto na remoção de ruídos quanto na preservação das bordas da imagem.

a) b) c) d)

Figura 4.2: a) Imagem sem ruídos b) imagem submetida a ruídos c) resultado da aplicação da média da vizinhança d) resultado da aplicação da filtragem mediana

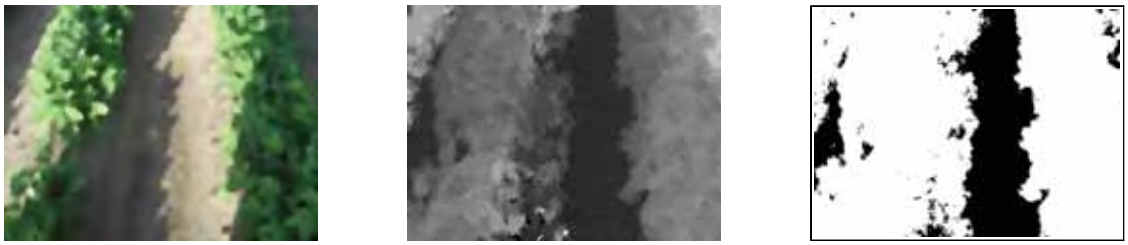
4.1.2 Segmentação

Esta etapa consiste em obter uma imagem binária na qual o caminho é representado em preto e a área correspondente à plantação é representada em branco. As imagens do banco estão representadas no modelo de cor RGB e foram convertidas para o modelo de cor HSI utilizando a abordagem descrita na seção 3.1.3. O modelo de cor HSI foi escolhido para a realização desta tarefa devido a possibilidade de identificação de cores independentemente da intensidade de iluminação ou do grau de saturação das cores da imagem. Conforme discutido na seção 3.1.3, do capítulo 3, o componente matiz representa a cor pura da imagem e possui valores entre 0° e 180º. O intervalo de valores utilizado para a identificação da plantação foi entre 60º e 180º, o qual corresponde ao intervalo de cores entre o amarelo e o azul. O motivo de incluir as variações da cor amarela está no fato de que plantas com tempo de vida mais longo podem apresentar este tom de cor. Valores entre 120º e 180º ainda apresentam alguns tons de verde, motivo este pelo qual este intervalo foi considerado. Valores de *pixels* do componente matiz que estão neste intervalo são representados em branco, sendo os valores fora do intervalo representados em preto. A escolha da área correspondente à plantação como parâmetro de busca pelo caminho da imagem se deve ao fato da cor do solo variar dependendo da região de plantio. Este procedimento de identificação do caminho pode ser visto na figura 4.3.

a) b) c)

Figura 4.3: a) Imagem original b) componente matiz do HSI c) caminho extraído

Uma deficiência do uso desta abordagem pode surgir quando uma imagem possui sombras. Uma sombra é uma região escura na imagem formada pelo bloqueio da iluminação por um objeto. Sombras podem prejudicar a segmentação de uma imagem devido à semelhança com regiões escuras pertencentes às áreas de interesse ou por serem processadas como extensão de um objeto presente na imagem. Na figura 4.4 é ilustrada a incorreta identificação do caminho de uma plantação em uma imagem com sombras.



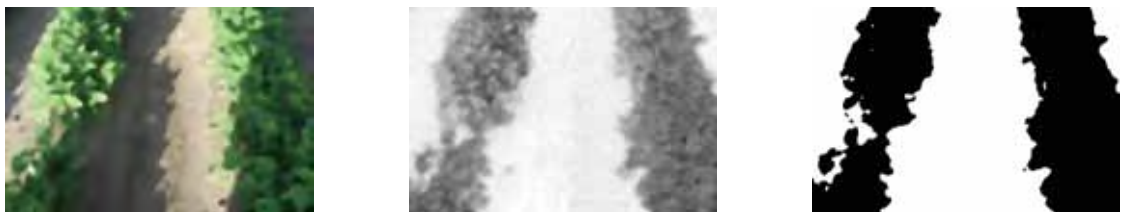
a) b) c)

Figura 4.4: a) Imagem com sombras b) componente matiz da imagem c) caminho identificado incorretamente

É possível notar na figura 4.4 (c) a identificação parcial do caminho da plantação. Isto se deve ao fato da região da sombra ser processada como parte integrante da área da plantação, como pode ser observado na figura 4.4 (b), devido à semelhança com a trilha da plantação esquerda. Portanto, a região correspondente à sombra, quando representada no componente matiz, possui os mesmos valores do intervalo considerado para identificação da área da plantação. Diante desta característica, é necessário que sombras sejam identificadas e desconsideradas no procedimento de extração do caminho. Uma abordagem proposta por Finlayson, Drew e Cheng (29) é a obtenção de uma imagem com valores de *pixels* no espaço logarítmico, denominada imagem do espaço log-cromático, a qual é invariante à iluminação e livre de sombras. Segundo Xu, Qi e Jiang (30), o método proposto por Finlayson, Drew e Cheng (29) pode ser escrito conforme a equação 4.1, a qual é utilizada para obter uma imagem no espaço log-cromático.

$$inv = \cos(\theta)\cdot\ln\left(\frac{r}{g}\right) + \sin(\theta)\cdot\ln\left(\frac{b}{g}\right) \qquad 4.1$$

Na equação 4.1, [r,g,b] são os valores de cores da imagem no modelo RGB e θ é a direção de projeção da sombra. Esta equação segue um princípio que diz que uma imagem no modelo de cor RGB pode ser convertida em uma imagem em nível de cinza a qual representa apenas a propriedade de reflectância (29). Na figura 4.5 (b) é exibido o resultado da aplicação da equação precedente sobre uma imagem com sombra.



a) b)

Figura 4.5: a) imagem com sombra b) imagem invariante a iluminação c) imagem invariante a iluminação binarizada

É possível perceber na figura 4.5 (b) que os efeitos da sombra foram minimizados no caminho da plantação. A imagem invariante a sombras foi submetida ao processo de binarização, conforme é mostrado na figura 4.5 (c), onde foi possível identificar a área da plantação em preto e o caminho em branco. A imagem invariante a iluminação binarizada é utilizada juntamente com o componente matiz para gerar a imagem com o caminho extraído. A condição a ser satisfeita para que o caminho seja identificado corretamente é quando os valores de *pixels* no componente matiz estiverem entre 60º e 180º e os valores dos *pixels* correspondentes na imagem invariante à iluminação binarizada for igual à 0 (preto). Desta maneira, a região do caminho referente à sombra no componente matiz não é identificada como área da plantação, pois nesta situação a condição mencionada resulta em um valor lógico falso. Como a imagem invariante à sombras foi suficiente para a identificação do caminho, não foi necessário eliminar as sombras da imagem original.

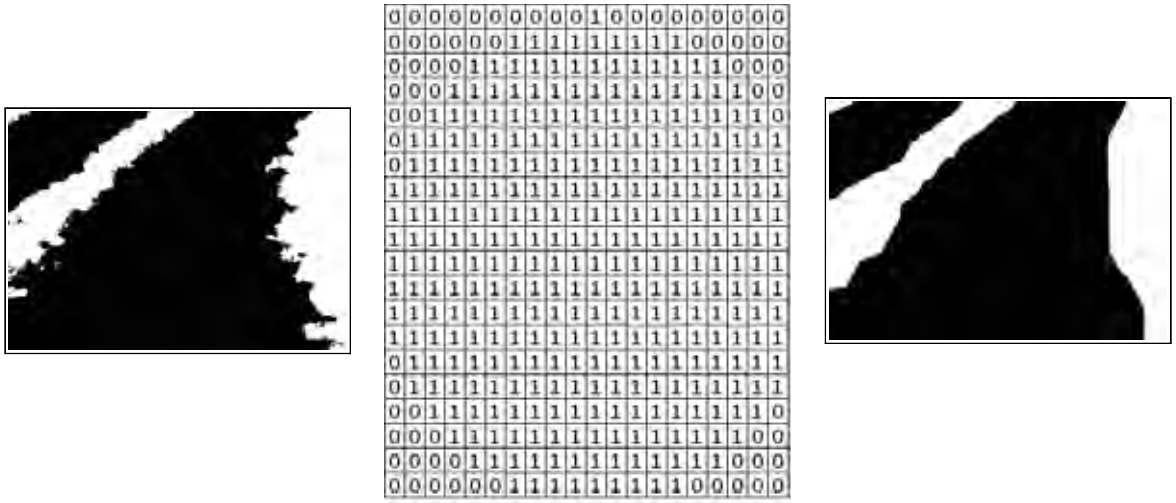
4.1.3 Suavização de bordas por meio de operações morfológicas

A operação de fechamento foi utilizada para suavizar as bordas do caminho identificado, eliminar pequenos buracos na imagem e eliminar pequenos istmos com a trilha adjacente, os quais são gerados devido a falhas na plantação. Este procedimento auxilia na melhor representação do esqueleto do caminho, minimizando os efeitos de proeminências que podem ser gerados devido a irregularidades nas bordas da imagem. Na figura 4.6 (c) é ilustrada o operação de fechamento sobre o resultado da segmentação do caminho da figura 4.6 (a) utilizando um elemento estruturante em forma de elipse exibido na figura 4.6 (b). O elemento estruturante possui dimensões 20x20. Foram utilizadas 20 iterações do fechamento sobre a imagem segmentada para obter mais suavização das bordas. A definição do elemento estruturante e do número de iterações foi feita empiricamente mediante vários testes, sendo que elementos estruturantes em forma de círculo ou elipse deixam as bordas da imagem mais suavizadas devido sua própria borda ser mais suave do que em elementos estruturantes quadrados ou retangulares.

4.1.4 Crescimento de regiões

Devido ao posicionamento da câmera no terreno, é possível que haja na imagem capturada caminhos adjacentes à trilha principal. A eliminação desses caminhos reduz a

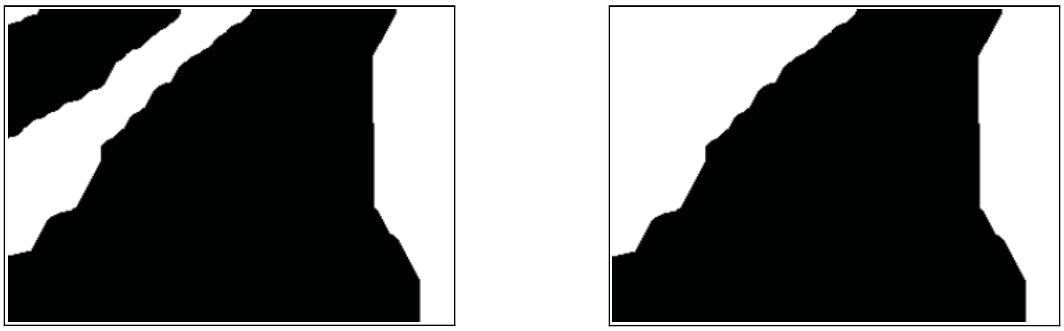
quantidade de informações a serem transmitidas para a máquina de vetores de suporte.



a) b) c)

Figura 4.6: a) Imagem segmentada b) elemento estruturante c) resultado do fechamento

Para a eliminação desses caminhos, foi utilizado o algoritmo de crescimento de regiões por agregação de *pixels*. O algoritmo inicia com a seleção de *pixels* "sementes", sendo que a condição de escolha são os *pixels* com menor nível de cinza, ou seja, em uma imagem binária são *pixels* pretos. Esses níveis de cinza são pertencentes aos caminhos identificados na imagem. Em uma imagem com três caminhos identificados, será possível selecionar três *pixels* sementes, um para cada caminho. O procedimento de crescimento de regiões é realizado e, simultaneamente a este processo, é feita a contagem de *pixels* pertencentes a cada região. Cada região é rotulada com um valor inteiro e ao final do processo é possível determinar qual região – caminho da imagem – possui a maior quantidade de *pixels*, sendo que este caminho é o que prevalece na imagem. Os outros caminhos são eliminados mediante a alteração do seus valores de níveis de cinza para branco. Na figura 4.7 (b) é ilustrado o resultado deste processo em uma imagem com dois caminhos. Na imagem mostrada na figura 4.7 (a), os caminhos são representados pela cor preta, sendo possível notar que o caminho central é o que possui a maior quantidade de *pixels* em relação ao caminho adjacente. Na imagem da figura 4.7 (b) o caminho adjacente foi eliminado.

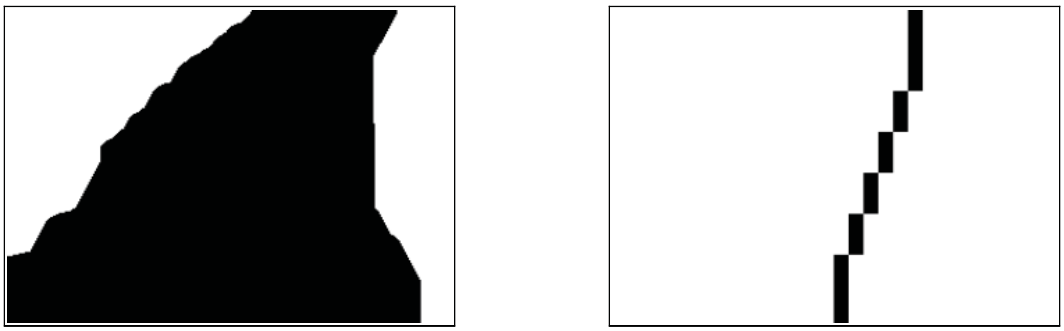


a) b)

Figura 4.7: a) Imagem com dois caminhos, representados pela cor preta b) Imagem com caminhos adjacentes eliminados

4.1.5 Afinamento do caminho

A próxima etapa foi o afinamento do caminho identificado, sendo o algoritmo de afinamento descrito na seção 3.1.6, do capítulo 3, utilizado para realizar este procedimento. O objetivo da obtenção do esqueleto da imagem é reduzir a quantidade de informações do caminho identificado a serem passadas para a máquina de vetores de suporte, reduzindo sua complexidade e melhorando seu desempenho em relação ao tempo de classificação. Além disso, a inclinação do esqueleto na imagem é a informação utilizada para o reconhecimento do ângulo de navegação pela máquina de vetores de suporte. Na figura 4.8 (b) é ilustrado o resultado do processo de afinamento em uma imagem de plantação já segmentada.



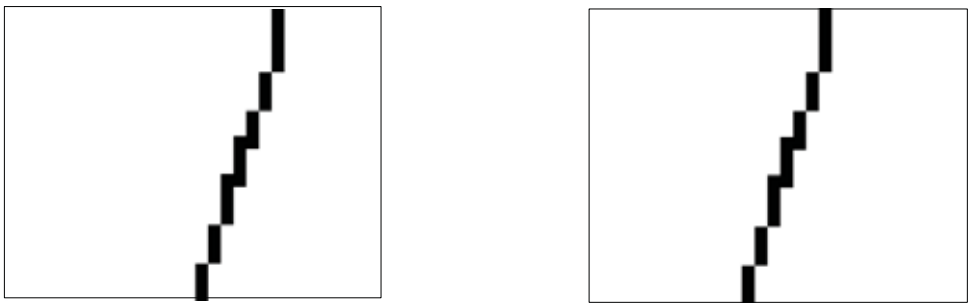
a) b)

Figura 4.8: a) Imagem segmentada b) imagem esqueletizada

4.1.6 Imagem invariante a translação

O caminho extraído da imagem de uma plantação pode estar sujeito a deslocamentos devido ao posicionamento da câmera. Isto pode ocasionar dificuldades no reconhecimento da

direção, pois cada deslocamento é classificado pela máquina de vetores de suporte em um ângulo diferente, mesmo que tenham ângulos idênticos. Portanto, o reconhecimento da direção do esqueleto da imagem deve ser independente do seu deslocamento na imagem. Foi utilizado neste trabalho o método desenvolvido por Yüccer e Oflazer (15), o qual foi discutido na seção 3.1.7, para o tratamento de imagens de modo invariante à translação. Na figura 4.9 (b) é ilustrado o resultado do deslocamento de um esqueleto na imagem. Esta é a imagem a ser transferida para a máquina de vetores de suporte para o reconhecimento do ângulo de direção.



a) b)

Figura 4.9: a) Esqueleto fora do centro imagem b) esqueleto deslocado para o centro da imagem

Para este trabalho, foi utilizado apenas o módulo de translação, pois a rotação do esqueleto na imagem é a informação de interesse para classificação. O módulo de escala não foi utilizado para manter o esqueleto com uma quantidade mínima de *pixels*.

Após o procedimento de translação, a imagem é redimensionada para uma resolução de 30x23, sendo este procedimento necessário para diminuir a quantidade de neurônios na camada de entrada da SVM.

4.1.7 Determinação dos parâmetros de treinamento

Como o aprendizado da SVM é supervisionado, então são necessários os valores de ângulos para cada uma das imagens esqueletizadas para a etapa de treinamento. Os parâmetros de treinamento foram obtidos utilizando a transformada de Hough e o método dos mínimos quadrados. Cada imagem com o caminho esqueletizado foi submetida à transformada de Hough e o valor do ângulo θ do esqueleto da imagem foi o parâmetro associado com saída desejada da SVM para aquela imagem. Como os valores de ângulo de

direção estão no intervalo entre -45º a 45º, com variações de 5º, então o valor gerado pela transformada de Hough foi discretizado utilizando o critério do valor mais próximo. Por exemplo, se o valor θ da transformada de Hough para uma imagem for 42, então a saída desejada para a imagem será 40, pois este é o valor mais próximo de 42.

4.2 Módulo de reconhecimento

Cada imagem foi transformada em um vetor de 690 elementos, os quais correspondem aos valores dos *pixels* da imagem. Esses valores são utilizados como entrada para a máquinas de vetores de suporte para o reconhecimento do ângulo de direção. As imagens são classificadas com ângulos discretizados no intervalo entre -45º a 45º, com variações de 5º, totalizando 19 padrões a serem classificados. Foram utilizadas duas abordagens para a classificação: na primeira abordagem utilizou-se 19 SVMs, cada uma tendo 690 neurônios de entrada, 800 núcleos com função de base radial e 1 saída, sendo a quantidade de neurônios ocultos determinada com base na quantidade de imagens de treinamento. Cada SVM representa um ângulo de direção e sua saída é 1 se a imagem apresentada é classificada como pertencente ao ângulo de direção representado pela SVM, ou -1 caso contrário. Na segunda abordagem utilizou-se apenas uma SVM com o mesmo número de neurônios de entrada e saída das 19 SVMs utilizadas na primeira abordagem, porém foram utilizados 71 núcleos gaussianos e os valores de saída pertencentes ao intervalo entre -45 a 45, ao invés de 1 e -1. A redução da quantidade de neurônios ocultos se deve à redução do número de exemplos de treinamento utilizados nesta abordagem. A segunda abordagem é vantajosa devido ao menor número de neurônios a serem implementados em *hardware*, além de ter proporcionado melhores resultados em relação à primeira abordagem.

4.2.1 Treinamento

Em cada abordagem, foi utilizado um conjunto diferente de imagens de treinamento. Na primeira abordagem, com a utilização de 19 SVMs, foram utilizadas 800 imagens de treinamento, sendo 400 imagens de amendoim e 400 imagens de soja, todas pertencentes ao banco de imagens de plantações. Para cada SVM, foram apresentados todos os exemplos de treinamento e a respectiva saída desejada. Os valores de saída desejados para cada exemplo de

treinamento é 1 ou -1. Se a imagem pertence ao ângulo de direção representado pela SVM, então a saída é 1. Se a saída for -1, então a imagem não pertence ao ângulo representado pela SVM.

Na segunda abordagem, utilizando apenas uma SVM, foram realizados três testes, cada um utilizando diferentes quantidades de imagens para treinamento e verificação. As imagens foram apresentadas a SVM juntamente com as saídas desejadas, as quais nesta abordagem apresentam valores entre -45 a 45. O valor de saída da SVM é um valor real entre -45 a 45.

4.2.2 Verificação

A etapa de verificação consiste na apresentação de imagens que não foram utilizadas na etapa de treinamento para avaliar a capacidade de classificação da SVM. Foram utilizadas 386 imagens de verificação, sendo 170 imagens de amendoim e 216 imagens de soja, todas pertencentes ao banco de imagens de plantações.

Na primeira abordagem, todas as imagens foram apresentadas às 19 SVMs e a direção foi determinada pela SVM que apresentou o valor de saída 1. Uma desvantagem do uso dessa abordagem é a ocorrência de imagens não classificadas quando a saída de todas SVMs forem -1. Na segunda abordagem, as imagens de verificação foram apresentadas a apenas 1 SVM, sendo a direção determinada pelo valor de saída da SVM. O valor de saída foi discretizado utilizando o critério do valor mais próximo, conforme discutido na seção 4.1.7. Esta abordagem é melhor em relação à primeira devido a não ocorrência de imagens sem classificação.

5 Experimentos

Nesta seção são apresentados os testes realizados utilizando as duas abordagens descritas na seção 4.2, além dos resultados obtidos. Os testes foram realizados em duas etapas: etapa de aferição e etapa com imagens reais. A seguir, são descritos os detalhes de cada etapa.

5.1 Teste de aferição

O primeiro teste foi realizado com imagens simuladas e foi uma etapa necessária para a aferição da taxa de reconhecimento para a topologia da SVM, sendo que isto permite verificar se a SVM é capaz de classificar imagens previamente conhecidas. Foram utilizadas 627 imagens simuladas de esqueletos, sendo 342 imagens de treinamento e 285 imagens de verificação, todas submetidas ao algoritmo de translação descrito na seção 4.1.6. No conjunto de imagens de treinamento há 18 imagens em cada ângulo de direção, com diversas variações e deformações em seus esqueletos para garantir a proximidade com situações reais.

No teste de aferição foi utilizada uma rede neural artificial MLP e 19 SVMs. A topologia da MLP é composta de 690 neurônios de entrada, 342 neurônios ocultos e 19 neurônios de saída. Cada neurônio de saída representa um ângulo de direção, sendo o valor de saída de cada neurônio pertencente ao intervalo [0,1]. O ângulo de direção será determinado pelo neurônio de saída que apresentar o maior valor. A função de ativação utilizada em todas as camadas foi a sigmóide logística. O algoritmo *backpropagation* foi utilizado no treinamento com os seguintes parâmetros de aprendizagem:

- Taxa de aprendizado: 0,7;
- Taxa de momento: 0,1;
- Número de épocas: 500.000;
- Erro médio: 0,00001.

Foram utilizadas 19 SVMs para o reconhecimento, sendo cada uma utilizada para classificar um ângulo de direção. Para cada SVM, foram apresentados todos os exemplos de treinamento e a respectiva saída desejada. Os valores de saída desejados para cada exemplo de treinamento é 1 ou -1. Se a imagem pertence ao ângulo de direção representado pela SVM, então a saída é 1. Se a saída for -1, então a imagem não pertence ao ângulo representado pela SVM. Após a etapa de treinamento, foram apresentadas às 19 SVMs as imagens de

verificação e a comparação da saída de cada SVM com a saída desejada de uma imagem. A direção foi determinada pela SVM que apresentou saída 1.

A taxa de acerto de classificação da SVM e da MLP foi equivalente, como pode ser visto na tabela 5.1.

Tabela 5.1: Resultados da etapa de aferição

RNA	Acertos	Erros	Percentual de acerto
MLP	273	12	95,8%
SVM	277	8	97,1%

Na figura 5.1 é mostrado um gráfico onde são apresentados os erros de classificação para cada rede neural artificial. As imagens não classificadas representam a situação onde a saída das 19 SVMs é -1. Nota-se no gráfico da figura 5.1 que esta situação não ocorre no MLP uma vez que este tipo de rede neural sempre apresenta um valor de saída positivo para cada neurônio na camada de saída. Portanto, sempre é possível avaliar o ângulo de direção em uma rede neural artificial MLP pela determinação do neurônio de saída que apresentou maior valor, mesmo que a classificação esteja incorreta.

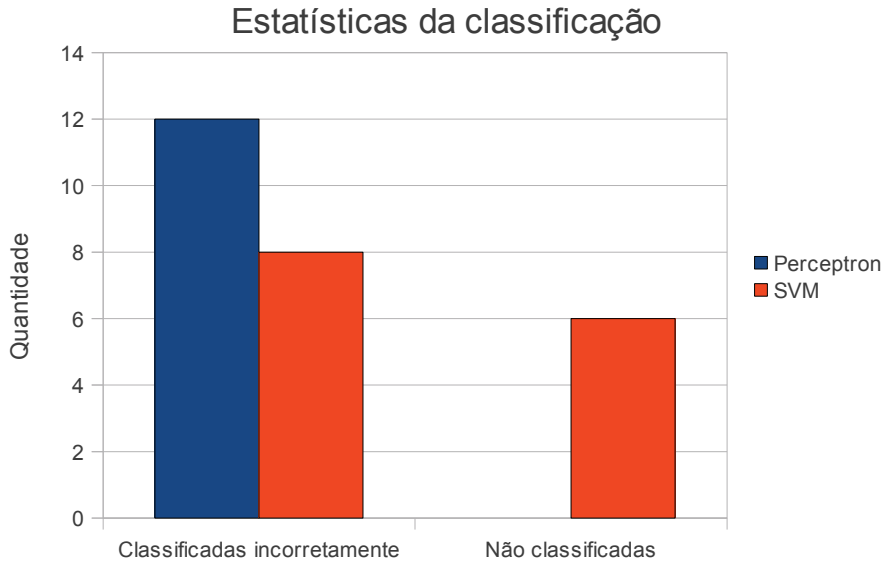


Figura 5.1: Gráfico com as estatísticas da classificação

5.2 Teste com imagens reais

Os testes reais foram realizados com imagens de amendoim e soja, totalizando 1474

imagens para treinamento e verificação. Do conjunto de imagens do banco, foram utilizadas 800 imagens para treinamento, sendo 400 de amendoim e 400 de soja, e 386 imagens para verificação, sendo 170 imagens de amendoim e 216 imagens de soja. Além disso, foram acrescentadas 288 imagens de treinamento simuladas para minimizar o desequilíbrio da quantidade de imagens em cada ângulo, sendo que haviam ângulos com maior quantidade de imagens do que outros. As imagens acrescentadas pertencem ao conjunto de imagens de treinamento da etapa de aferição.

Foram realizados quatro testes com Máquina de Vetores de Suporte, cada um utilizando uma abordagem diferente. As funções de *kernel* utilizadas nas SVMs foram a de base radial, o *hardware-friendly kernel* e o *histogram intersection kernel*, discutidos na seção 3.3. O valor escolhido para o parâmetro ϒ, do *hardware-friendly kernel*, foi 0.0625, o qual foi determinado empiricamente por meio de testes. Em cada teste, foram consideradas as seguintes informações:

- Quantidade de acertos exatos;
- Quantidade de acertos aproximados;
- Erros;
- Imagens não classificadas;
- Percentual de acerto.

Os acertos exatos indicam a quantidade de imagens onde os ângulos de direção desejado e calculado pela SVM coincidiram exatamente. Os acertos aproximados indicam a quantidade de imagens onde os ângulos de direção desejado e calculado pela SVM tiveram uma diferença de 5º. Os erros representam a quantidade de imagens que tiveram uma diferença de 10º ou mais entre o ângulo desejado e o calculado pela SVM. O percentual de acerto no reconhecimento foi calculado considerando os acertos exatos e aproximados. Os acertos aproximados foram considerados devido à semelhança dos esqueletos que possuem ângulos com 5º de diferença, fazendo com que a SVM classifique uma imagem com a informação aproximada do ângulo. A tolerância pode ser aceitável se o tempo de processamento for baixo o suficiente para evitar que a próxima saída seja feita após uma distância de 1 metro percorrida pelo robô móvel, podendo isto minimizar a diferença de distância fora da rota.

Tentou-se utilizar uma MLP nesta etapa, com a mesma topologia e parâmetros de aprendizado da que foi utilizada na etapa de aferição. Entretanto, o tempo de treinamento deste tipo de rede é lento dependendo da quantidade e complexidade dos exemplos e do valor

do erro médio. Devido ao grande número de exemplos de treinamento que foram utilizados nesta etapa e às muitas variações que eles apresentam, a rede MLP não convergiu para o erro médio desejado. Foram realizados vários treinamentos com diferentes topologias, porém em nenhum deles a rede convergiu para uma solução. Como a determinação da topologia de MLP é empírica, concluiu-se que não é adequado utilizá-la para realização de muitos testes, já que cada treinamento pode levar até vários dias.

5.2.1 Primeiro teste com SVM

O primeiro teste foi realizado com 1186 imagens, todas pertencentes ao banco de imagens de plantações. Foram utilizadas 800 imagens para treinamento, sendo 400 imagens de amendoim e 400 de soja, e 386 imagens para verificação, sendo 170 imagens de amendoim e 216 imagens de soja. Foram utilizadas 19 SVMs para classificação, da mesma maneira que foi feita na etapa de aferição. Os exemplos de treinamento foram apresentados para cada SVM, inclusive as saídas desejadas. O ângulo de direção foi determinado pela SVM que apresentou saída 1. Na tabela 5.2 são apresentados os resultados obtidos neste primeiro teste para cada *kernel* usado no treinamento e reconhecimento.

Tabela 5.2: Estatísticas do reconhecimento do primeiro teste

	Radial Basis Function	**Hardware Friendly Kernel**	**Histogram Intersection Kernel**
Acertos exatos	260	262	65
Acertos aproximados	20	22	143
Erros	4	5	178
Imagens não classificadas	102	97	0
Percentual de acerto	72,5%	73,6%	53,9%

Os resultados obtidos dos *kernels Radial Basis Function* e *Hardware-Friendly* forem equivalentes. O *kernel Histogram Intersection* apresentou muitos erros no reconhecimento. O fator prejudicial neste teste foi a quantidade de imagens que não foram classificadas. Essa situação ocorre quando a saída das 19 SVMs é -1, fato que ocorre porque a imagem apresentada não foi reconhecida em nenhum ângulo de direção. A quantidade insuficiente de imagens de treinamento para um determinado ângulo de direção foi um fator que contribuiu para que algumas imagens apresentadas às SVMs não pudessem ser classificadas.

5.2.2 Segundo teste com SVM

A implementação em *hardware* de 19 SVMs pode não ser possível devido à grande quantidade de neurônios que pode exceder a capacidade de armazenamento daquele *hardware*. Cada SVM possui 690 neurônios de entrada, 800 neurônios na camada oculta e 1 de saída, totalizando 1491 neurônios, sendo que a utilização de 19 SVMs totalizará 28329 neurônios a serem implementados em *hardware*. Diante disso, foi estudada a possibilidade de implementação de apenas uma SVM que classifique todos os padrões de direção. Além de diminuir consideravelmente o espaço para síntese da SVM em *hardware*, a classificação pode ser mais precisa devido às imagens não classificadas no teste anterior serem reconhecidas como acertos exatos ou aproximados. Entretanto, a quantidade de erros também pode aumentar devido à saída da SVM estar sujeita a erros.

Neste teste foi utilizada apenas uma SVM, a qual possui saída entre -45 e 45, ao invés de 1 ou -1. Os resultados foram melhores em relação ao teste anterior, conforme pode ser visto na tabela 5.3. Apesar da quantidade de erros ter aumentado, boa parte das imagens que não haviam sido classificadas anteriormente foram classificadas com a tolerância de 5º de diferença, ocasionando um aumento na quantidade de acertos aproximados e consequentemente no percentual de acerto.

Tabela 5.3: Estatísticas do reconhecimento do segundo teste

	Radial Basis Function	**Hardware Friendly Kernel**	**Histogram Intersection Kernel**
Acertos exatos	238	216	0
Acertos aproximados	118	141	0
Erros	30	29	386
Imagens não classificadas	0	0	0
Percentual de acerto	92,2%	92,5%	0,0%

Nota-se que neste teste não foi obtido nenhum acerto com o *Histogram Intersection Kernel*. Isto ocorre porque os valores de entrada para este *kernel* são binários (isto é, 0 e 1) e a probabilidade da ocorrência da soma dos mínimos ser zero ou um valor próximo de zero é alta devido a saturação dos exemplos de treinamento. Como o resultado das soma dos mínimos é o pivô de divisão para a determinação dos pesos da SVM, então os valores dos pesos podem ser altos devido a divisão de um número por um valor próximo a zero resultar em um valor muito alto. Portanto, o resultado do reconhecimento para todos os exemplos de entrada pode estar

fora do intervalo entre -45 e 45, sendo este um fator que pode prejudicar o reconhecimento da SVM quando este *kernel* é usado.

5.2.3 Terceiro teste com SVM

Devido ao desequilíbrio da quantidade de imagens em cada ângulo, notou-se que a classificação da SVM era bem sucedida para ângulos que possuíam quantidades maiores de imagens na etapa de treinamento. Esse fator compromete a SVM por ela estar muito especializada em apenas alguns padrões, prejudicando sua capacidade de generalização e classificação de novos padrões que forem apresentados. Diante disso, foi necessária a inclusão de imagens simuladas na etapa de treinamento para padrões deficientes de imagens, com o objetivo de manter a capacidade de generalização da SVM. Os padrões deficientes de imagens estão entre -10º e -45º e entre 10º e 45º. Foram incluídas 293 imagens simuladas para o treinamento da SVM, todas pertencentes ao conjunto de imagens de treinamento da etapa de aferição. Comparado com o teste anterior, o aumento no número de imagens de treinamento causou um aumento do número de acerto exatos e a redução do número de acertos aproximados e de erros.

Tabela 5.4: Estatísticas do reconhecimento do terceiro teste

	Radial Basis Function	**Hardware Friendly Kernel**	**Histogram Intersection Kernel**
Acertos exatos	291	283	0
Acertos aproximados	74	79	0
Erros	21	24	386
Imagens não classificadas	0	0	0
Percentual de acerto	94,56%	93,78%	0,0%

5.2.4 Quarto teste com SVM

O uso de 1093 imagens de treinamento gera 1093 vetores de suporte e 1093 pesos para compor os parâmetros de reconhecimento da SVM. Sendo cada vetor de suporte representado por uma imagem de treinamento de 690 *pixels* de 1 byte, então são necessários 754.170 bytes ou 736 kbytes de memória para armazenamento de tais parâmetros, os quais não são suportados pela memória RAM interna do modelo de FPGA utilizado neste trabalho. Além

disso, a utilização de um grande número de imagens de treinamento pode diminuir a capacidade de generalização no reconhecimento pela SVM. Diante disso, neste teste foram utilizadas 71 imagens de treinamento e 1057 imagens de verificação, todas pertencentes ao banco de imagens de plantações. A redução do número de imagens de treinamento permite, além de solucionar o problema de limitação do FPGA, aumentar a capacidade de generalização no reconhecimento da SVM. Os resultados podem ser vistos na tabela 5.5.

Tabela 5.5: Estatísticas do reconhecimento do quarto teste

	Radial Basis Function	**Hardware Friendly Kernel**	**Histogram Intersection Kernel**
Acertos exatos	776	810	115
Acertos aproximados	197	175	343
Erros	84	72	599
Imagens não classificadas	0	0	0
Percentual de acerto	92,5%	93,19%	43,33%

Devido ao aumento no número de imagens de teste, o número de acertos aproximados e erros aumentou praticamente na mesma proporção em relação ao terceiro teste. Entretanto, nota-se que houve uma pequena diferença no percentual de acertos em relação ao teste anterior, principalmente no percentual de acertos do segundo *kernel*. Além disso, os resultados obtidos por meio do *hardware-friendly kernel* neste teste foram melhores em relação ao *kernel* de função de base radial. O armazenamento em *hardware* diminui significativamente com a redução de imagens de treinamento, pois são necessários apenas 48 kbytes para armazenamento dos vetores de suporte e dos pesos.

5.3 Implementação e síntese do circuito no FPGA

A topologia da SVM implementada na linguagem C++ foi descrita na linguagem VHDL por meio de um circuito de três entradas e duas saídas, sendo uma entrada reservada para o sinal de *clock*, uma para o sinal de *reset* e uma entrada de oito bits que recebe um *pixel* da imagem de entrada. A primeira saída do circuito é um sinal utilizado para indicar se os *pixels* da imagem devem ser lidos. A segunda saída é um valor de oito bits, o qual representa o ângulo de saída reconhecido pelo circuito da SVM.

Cada imagem de teste foi armazenada na memória interna do FPGA, sendo cada *pixel*

utilizado como entrada para o circuito da SVM. As etapas de execução do circuito foram implementadas por meio de uma máquina de estados finitos de Moore. Máquinas de estados finitos são utilizadas na construção de circuitos sequenciais e são estruturas definidas por um conjunto de entradas, um conjunto de estados, uma função de transição entre os estados e uma função de saída. Os estados representam as sequências de operações desempenhadas pelo processo, sendo cada estado uma situação em que o processo se encontra em um momento de execução. A função de transição determina o próximo estado da máquina baseando-se no estado atual e, possivelmente, nos valores de entrada. A função de saída determina as saídas baseando-se no estado atual e, se necessário, nas entradas da máquina.

Conforme apresentado na seção 3.3, a função de *kernel hardware-friendly* utiliza uma potência de base 2. A implementação do circuito de potência por ser realizada mediante a equação 5.1.

$$x^y = \exp(y * \ln(x)) \tag{5.1}$$

A máquina de estados da SVM implementada em VHDL possui treze estados e é mostrada na figura 5.2.

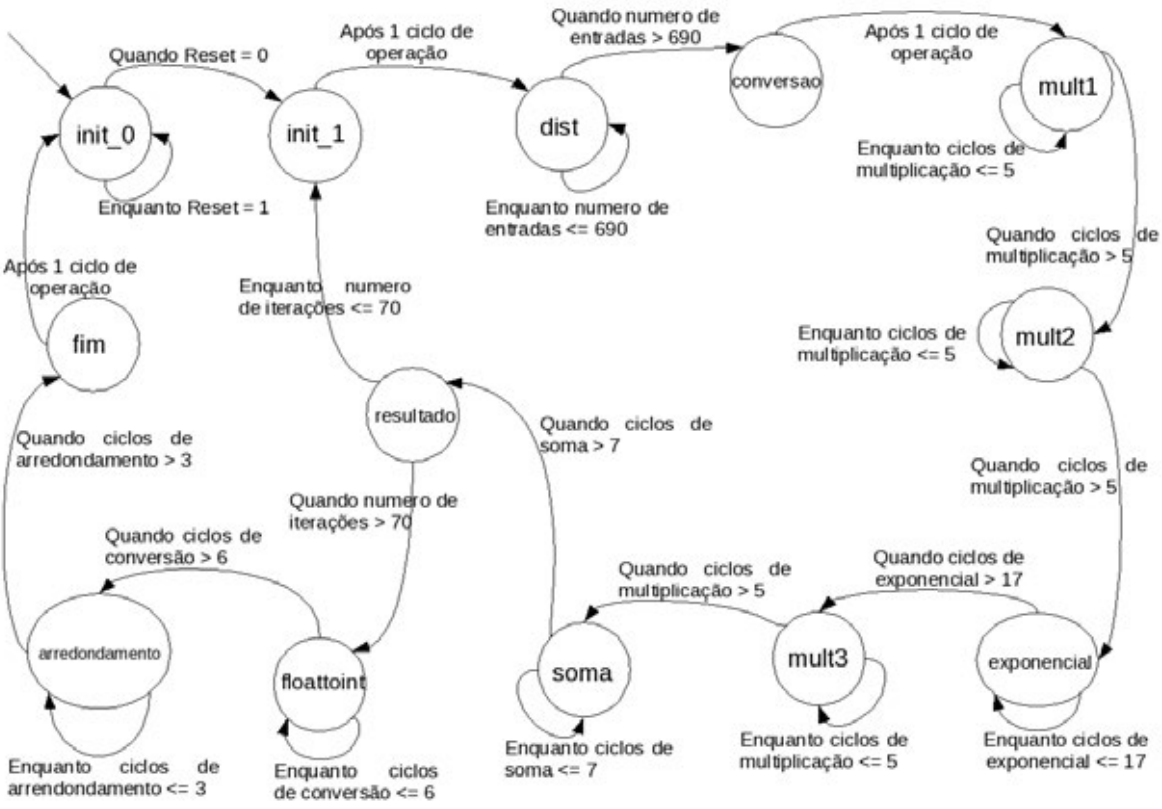


Figura 5.2: Máquina de estados finitos do circuito da SVM

A seguir, uma descrição de cada estado:

- **init_0** e **init_1**: estados iniciais da máquina;
- **dist**: estado onde é calculada a norma ||x – y|| entre os elementos da imagem x e do vetor de suporte y. O estado permanece em recursividade até que a leitura dos 690 elementos da imagem e do vetor de suporte seja realizada;
- **conversão**: estado onde é feita a conversão do peso e do resultado da norma para o padrão IEEE 754 de representação de números de ponto flutuante (31);
- **mutl1, mult2** e **exponencial**: estados que realizam o cálculo da função de *kernel*, conforme apresentado na equação 5.1. A multiplicação é feita em cinco ciclos de clock e o cálculo da exponencial é feito em 17 ciclos de *clock*, fato que justifica a recursividade dos estados. O mesmo se aplica aos estados **mult3**, **soma** e **floattoint;**
- **mult3**: estado onde o valor do peso é multiplicado pelo resultado do *kernel;*
- **soma**: acumula o resultado da função de *kernel*. A soma é feita em sete ciclos de *clock;*
- **floattoint**: converte o resultado, representado no padrão IEEE 754 (31), para número inteiro. A conversão é feita em seis ciclos de *clock;*
- **arredondamento**: estado onde é realizada a discretização do valor inteiro utilizando o método do valor mais próximo discutido na seção 4.1.7. O processo é feito em três ciclos de *clock*;
- **fim**: estado final da máquina, onde o resultado final é apresentado na porta de saída do circuito.

O padrão IEEE 754 (31) foi um modelo criado para obter uma padronização na representação de números de ponto flutuante no sistema binário. Neste padrão, o número de ponto flutuante é composto de três grupos de bits: sinal, expoente e mantissa. Na equação 5.2 é apresentado esse padrão.

$$Número = -1^{s} x\, M\, x\, 2^{E-bias} \tag{5.2}$$

onde:

- s = 0 ou 1
- M = conjuntos de bits significantes que representam a precisão do número
- E = um valor entre -126 e 127
- *bias* = um valor de viés usado para calcular o expoente

O bit de sinal é uma potência de base -1 e expoente s igual a 0 ou 1. Qualquer valor de

base elevado à zero, seja positivo ou negativo, sempre terá 1 como resultado e neste caso o número é positivo. Se o valor do expoente s for igual à 1, então o resultado da potência é -1 e isto significa que o número de ponto flutuante é negativo. A interpretação da mantissa e do expoente será explicada de uma maneira resumida utilizando um número de ponto flutuante de precisão simples (32 bits), o qual possui um bit de sinal, oito bits para o expoente e vinte e três bits para a mantissa, conforme mostrado na figura 5.3.

S	E	M
1	8	23

Figura 5.3: Representação de um número de ponto flutuante de 32 bits

O cálculo do expoente é realizado subtraindo-se o valor de E de um valor de *bias*, o qual é 127 em um expoente de 8 bits. O valor do expoente E determina o número de deslocamentos da vírgula utilizado para obter o grupo de bits da mantissa. Considere o número 3,75 representado no sistema binário como 11,11. Neste caso, o deslocamento da vírgula é feito, da direita para a esquerda, uma vez até o último bit de valor 1. Isto faz com que o valor da mantissa seja 1,111 (1,875) e o valor de E seja igual a 128, pois o valor do expoente tem que ser igual ao número de deslocamentos da vírgula. Portanto, a representação do número 3,75 no padrão IEEE 754 é:

0	10000000	11100000000000000000000

A mantissa é representada apenas pelo grupo de bits após a vírgula. Substituindo os valores de E e M na equação 5.2 com os valores obtidos no exemplo acima, e S=0 pelo fato do número ser positivo, obtemos o resultado da equação 5.3.

$$Número = -1^{0} x\, 1{,}875_{10} x\, 2^{128-127} = 3{,}75 \qquad (5.3)$$

A fim de verificar o resultado obtido com a execução do circuito, foi realizada uma simulação no *software* ModelSim Starter Edition antes que o circuito fosse sintetizado no FPGA. A simulação foi feita utilizando um arquivo teste (o chamado arquivo *test bench*) no qual foi simulado um ciclo de *clock* de 20 nanossegundos referente a um FPGA de 50 Mhz. As simulações foram feitas utilizando uma amostra de 20 imagens do conjunto de imagens de reconhecimento e a saída do circuito para cada imagem testada foi comparada com a sua correspondente saída em *software*. Os resultados das 20 simulações foram iguais aos obtidos em *software*. Na figura 5.4 é apresentada a saída da simulação realizada em uma imagem, onde o resultado do circuito, com valor 25, correspondeu com o resultado do reconhecimento no *software*.

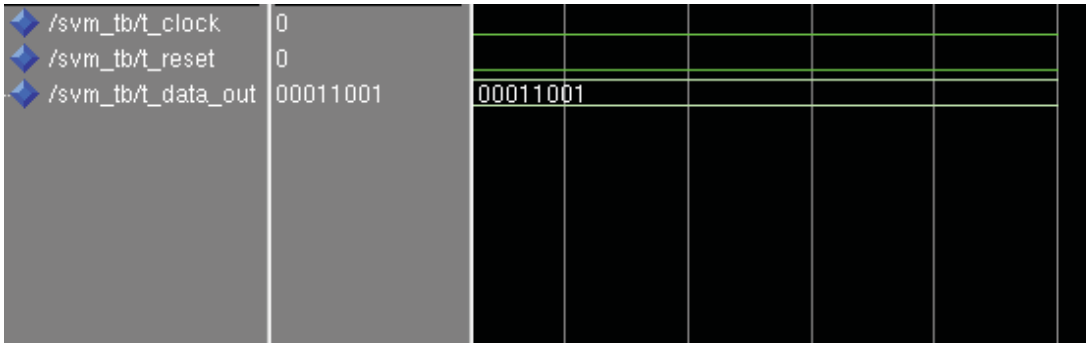


Figura 5.4: Resultado da simulação no *software* ModelSim

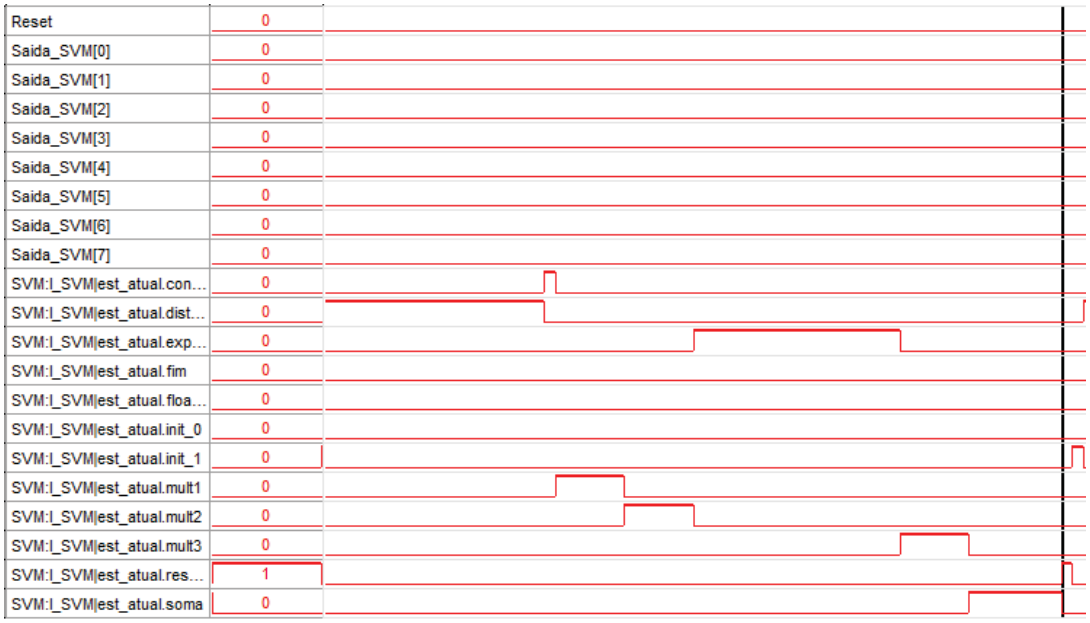
A síntese do circuito foi feita na placa de desenvolvimento NIOS (32), na qual está incorporado um FPGA da família Stratix II, modelo EP2S60F672C5ES de 50 Mhz. O *software* Quartus II foi utilizado para compilação do circuito, comunicação com a placa NIOS e captura dos sinais dos pinos do FPGA, os quais estão ligados à saída do circuito, para verificação dos resultados e do tempo de execução. O *software* Quartus II oferece um relatório de compilação no qual é listado o consumo de recursos do FPGA, tais como número de pinos, elementos lógicos, bits de memória e blocos de processamento de sinais digitais utilizados pelo circuito. O relatório de compilação do circuito da SVM gerado pelo Quartus II é apresentado na tabela 5.6. Embora o circuito utilize operações com números de ponto flutuante, o uso de recursos do FPGA é relativamente baixo.

Tabela 5.6: Relatório de compilação gerado pelo Quartus II

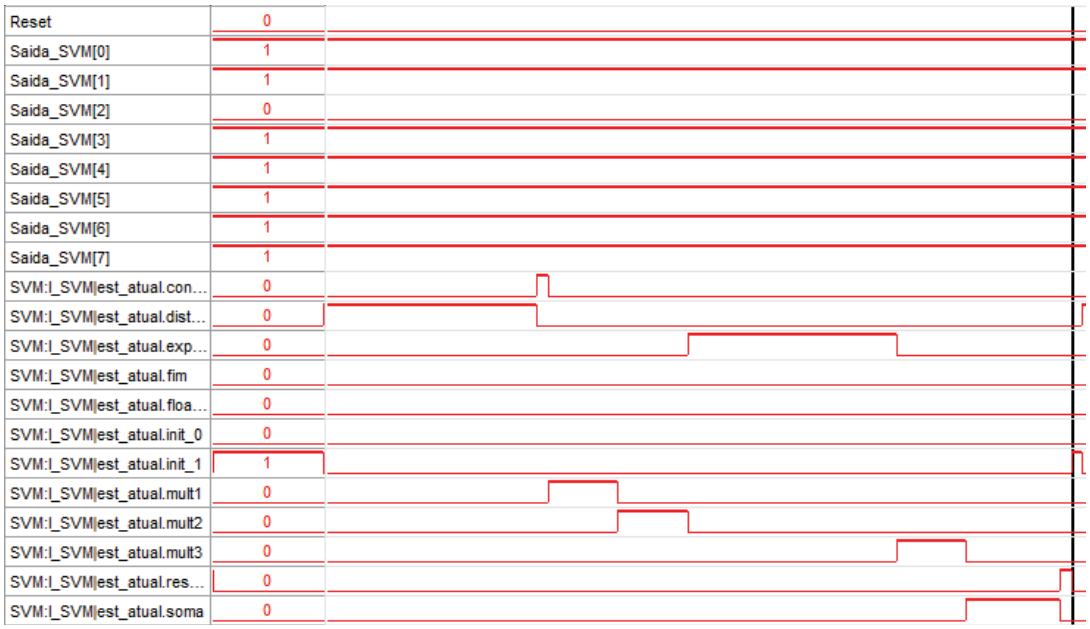
Recurso	Disponível	Consumido	Percentual
Utilização de lógica	48.352	4.937	8%
ALUTs combinacionais		2.509	4%
Registradores lógicos dedicados		2.428	3%
Pinos	493	18	2%
Bits de blocos de memória	2.544.192	398.304	16%
Blocos DSP	288	59	20%

Cada um dos oito bits de saída do circuito da SVM foi conectado a um pino do FPGA. Os sinais de saída desses pinos foram analisados no *software* SignalTap II Logic Analyzer, onde é possível visualizar a execução do circuito no FPGA. Os sinais de saída capturados pelo *software* foram comparados com os resultados obtidos na simulação no *software* ModelSim. A execução foi feita com 20 imagens do conjunto de imagens de reconhecimento, sendo que em todas as execuções os resultados obtidos foram os mesmos das simulações no ModelSim. Na figura 5.5 é mostrada a tela de captura de sinais do *software* SignalTap II de dois dos vinte testes realizados, sendo que a figura 5.5 a) corresponde ao resultado de uma imagem com

valor de saída igual a 0 e a figura 5.5 b) corresponde ao resultado de uma imagem com valor de saída igual a -5. Na tabela 5.7 são mostrados os resultados obtidos na simulação e na execução do circuito no FPGA e a comparação com as saída obtidas no *software* desenvolvido em C++ das 20 imagens testadas.



a)



b)

Figura 5.5: Resultados da execução no FPGA capturados pelo *software* SignalTap II Logic Analyzer a) imagem com valor de saída igual a 0 b) imagem com valor de saída igual a -5

Tabela 5.7: Comparação das saídas geradas pelo *software* desenvolvido em C++, pela simulação e pela execução no FPGA

	Resultado no software (base 10)	**Resultado da simulação (base 2)**	**Resultado do FPGA (base 2)**
Imagem 1	0	00000000	00000000
Imagem 2	-5	11111011	11111011
Imagem 3	0	00000000	00000000
Imagem 4	5	00000101	00000101
Imagem 5	0	00000000	00000000
Imagem 6	0	00000000	00000000
Imagem 7	-5	11111011	11111011
Imagem 8	0	00000000	00000000
Imagem 9	-5	11111011	11111011
Imagem 10	0	00000000	00000000
Imagem 11	0	00000000	00000000
Imagem 12	0	00000000	00000000
Imagem 13	-10	11110110	11110110
Imagem 14	10	00001010	00001010
Imagem 15	15	00001111	00001111
Imagem 16	-15	11110001	11110001
Imagem 17	20	00010100	00010100
Imagem 18	30	00011110	00011110
Imagem 19	-20	11101100	11101100
Imagem 20	25	00011001	00011001

5.4 Análise do desempenho

5.4.1 Análise de desempenho em *software*

Os testes foram realizados em computador com 4 Gibabytes de memória RAM e um processador Intel Dual Core de 2200 Mhz. Com relação ao desempenho do sistema, o custo computacional concentra-se nos algoritmos de processamento de imagens. O tempo médio de execução dos algoritmos do módulo de processamento de imagens, aplicados às 1128 imagens do banco, foi de 3 segundos. Na figura 5.6 é apresentado o tempo de execução de cada algoritmo, aplicados à uma imagem do banco. Nota-se que o algoritmo de esqueletização consume mais tempo em relação aos outros algoritmos, sendo este consumo o décuplo do segundo maior tempo.

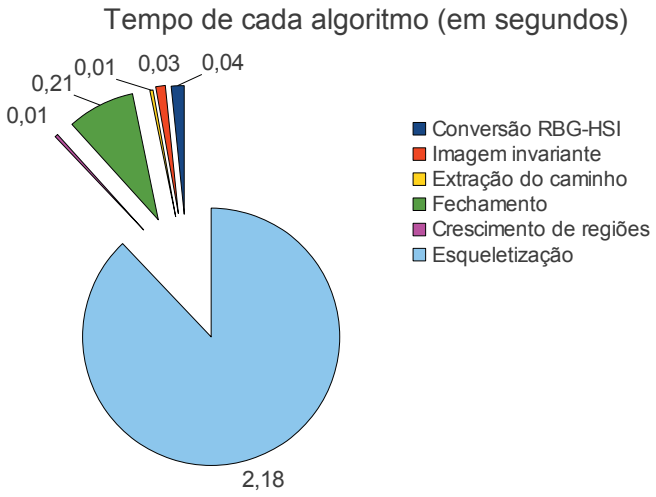
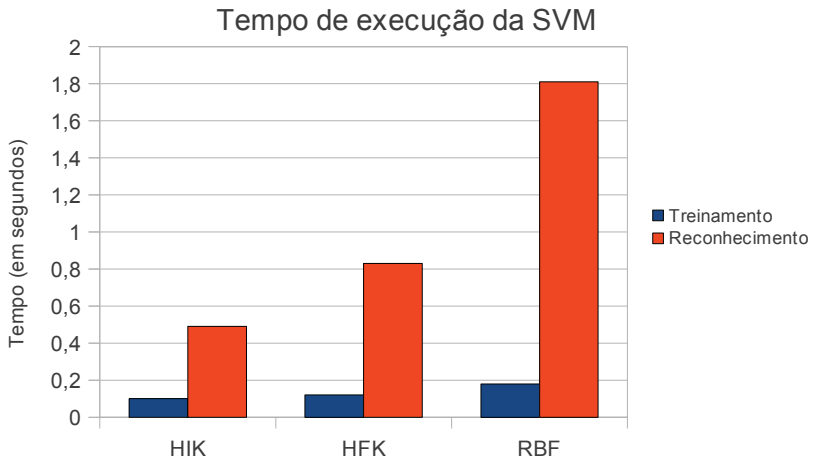


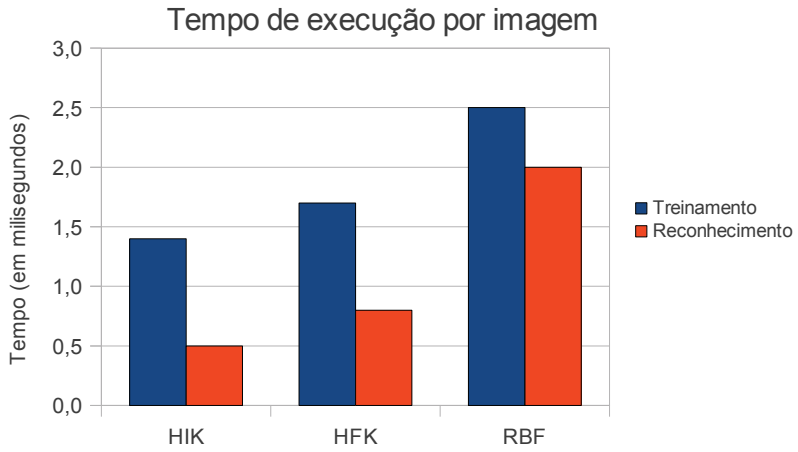
Figura 5.6: Tempo de execução de cada algoritmo, aplicados a uma imagem

O tempo de execução de cada função de *kernel* para o treinamento e reconhecimento da SVM é apresentado no gráfico da figura 5.7. O tempo de execução do *kernel Histogram Intersection* é menor em relação às outras funções de *kernel*, sendo, aproximadamente, duas vezes mais rápido do que o *kernel Hardware-Friendly* e quatro vezes mais rápido que o *kernel* RBF. Entretanto, os resultados apresentados por este *kernel* são inferiores em relação aos *kernels* HFK e RBF, conforme apresentado na seção 5.2. O *kernel* HFK apresentou um tempo de execução duas vezes mais rápido que o *kernel* RBF e, além disso, foi o *kernel* que apresentou melhores resultados no último teste. Independentemente da função de *kernel* utilizada, o uso de SVM mostra-se mais vantajoso, em relação ao tempo de treinamento, do

que o MLP, já que na SVM não há ciclos de retropropagação de erro como ocorre no MLP. Isto possibilita a implementação de treinamento *on-line* para agregar a classificação de novos padrões em diferentes campos de plantações.



a)



b)

Figura 5.7: a) Tempo de execução da SVM para todas imagens b) Tempo médio de execução por imagem

Considere que a atualização da direção será realizada a cada 1 metro e que o ângulo desejado seja 0º. Se a SVM determinar que o ângulo de navegação é 5º, então teremos uma saída fora da rota de 7 cm. Considerando que uma trilha de plantação de amendoim possui 90 cm de largura e que o pneu de um equipamento agrícola possua 13,6 cm de largura, então haverá uma folga de 38,2 cm de cada lado do pneu e uma perda de 7 cm pode ser aceitável nessa situação. Conforme informações obtidas por operadores de equipamentos agrícolas, a velocidade de um implemento agrícola utilizado em tarefas de cultivo e pulverização varia de

6 Km/h a 8 Km/h. Se a velocidade do implemento agrícola for 6 Km/h, ou 1,666 m/s, então a distância de 1 metro será percorrida em aproximadamente 0,6 segundos. Portanto, o tempo de execução do algoritmo deve ser compatível com o tempo necessário para percorrer a distância de 1 metro. É importante destacar que a distância de atualização do ângulo de direção pode ser menor que 1 metro se o tempo de execução do algoritmo for menor que 0,6 segundos.

5.4.2 Análise de desempenho em *hardware*

O tempo de reconhecimento do circuito da SVM no dispositivo FPGA de 50 Mhz é de aproximadamente 1,05 ms, considerando o tempo de execução para uma imagem. Utilizando um processador de 2200 Mhz, o tempo de reconhecimento em *software* é de aproximadamente 0,8 ms, conforme apresentado na figura 5.8, sendo aproximadamente $^{5}/_{4}$ menor em relação ao FPGA.

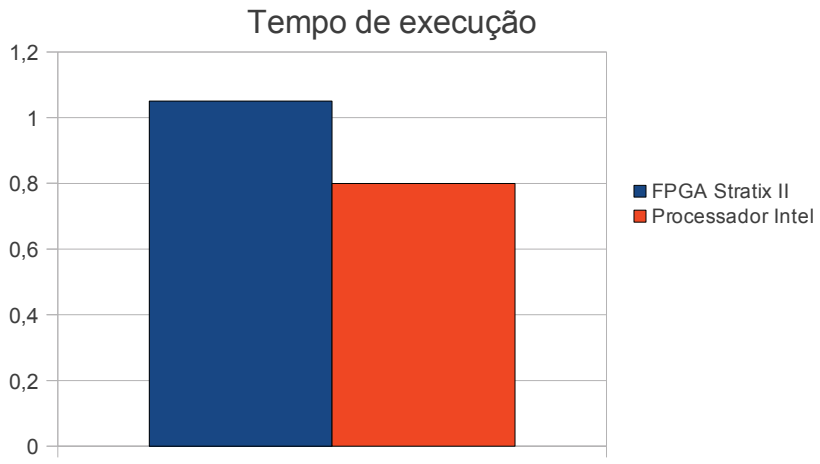


Figura 5.8: Tempo de reconhecimento no FPGA e em *software*

É importante considerar que a análise realizada nesta seção é uma estimativa do que poderia ser alcançado com a execução do algoritmo em *hardware* e que o tempo de execução da SVM em *software* foi obtido em um computador com sistema operacional multi tarefas onde é feito o escalonamento de processos. Além disso, o tempo estimado de execução em uma frequência de *clock* maior não é linear.

Tendo em vista que a execução de um algoritmo em um processador de 2200 Mhz deva ser 44 vezes mais rápida que um FPGA de 50 Mhz, então o tempo de execução estimado da SVM em FPGA 2200 Mhz é de 0,025 milissegundos (25 microssegundos). Em relação ao

processador de propósito geral de mesma frequência utilizado nos testes, o tempo de execução do FPGA seria 32 vezes mais rápido para o mesmo algoritmo. Entretanto, é importante considerar que dispositivos FPGA com frequência de *clock* de 500 Mhz, como o Ciclone III e o Aria II, da Altera, podem melhorar o desempenho de execução do algoritmo em 10 vezes em relação à uma frequência de 50 Mhz, o que leva a tempo de execução estimado de 105 microssegundos do circuito da SVM. Portanto, neste caso o tempo de execução estimado ainda seria 8 vezes mais rápido comparado com o processador de propósito geral. Além disso, conforme discutido no primeiro parágrafo desta seção, a tecnologia de processo utilizada em um processador de 500 Mhz pode ser relativamente mais simples e menos custosa se comparada à de um processador de 2 Ghz.

Considerando as estimativas de tempo mencionadas e a possibilidade de implementação dos algoritmos de processamento de imagens no FPGA, se o ganho estimado de execução de um algoritmo no FPGA de 50 Mhz é $^{5}/_{4}$ em relação à execução no *software* e o tempo médio de execução dos algoritmos de processamento de imagens seja 3 segundos em um processador de 2200 Mhz, então o tempo de execução estimado dos algoritmos de processamento de imagens em um FPGA de 50 Mhz é de aproximadamente 2,5 segundos (2500 missegundos). Considerando que o tempo de navegação em uma distância de 1 metro é 0,6 segundos, então o tempo de execução do algoritmo é incompatível com o tempo de navegação, Entretanto, se considerarmos a possibilidade de implementação dos mesmos algoritmos em um FPGA de 500 Mhz, onde o ganho de desempenho é 10 vezes em relação ao de 50 Mhz, então o tempo de execução estimado seria 240 milissegundos. Dessa maneira, o tempo torna-se compatível com o tempo de navegação (0,6 segundos), sendo ainda possível duas execuções do algoritmo no tempo de navegação.

6 Considerações finais

6.1 Conclusões

Neste trabalho foi apresentado um sistema de navegação para dirigibilidade de robôs móveis por caminhos em plantações, baseado em visão computacional e máquinas de vetores de suporte, além das etapas do desenvolvimento do sistema e os resultados obtidos por meio dos testes em *software* e em *hardware*. O principal objetivo do trabalho foi obter um percentual de acerto superior a 90% utilizando uma pequena quantidade de informações da imagem, isto é, apenas o esqueleto da imagem do caminho da plantação. Além disso, foi desenvolvido um circuito em linguagem VDHL para a execução máquina de vetores de suporte em um dispositivo FPGA, onde foi possível obter resultados idênticos ao do *software*.

A primeira etapa deste trabalho foi o estudo dos algoritmos de processamento de imagens mais adequados para a extração das características do caminho de uma trilha de plantações, sendo tais algoritmos aplicados sobre uma banco de dados de imagem de plantações de amendoim e soja. Durante a execução dos algoritmos de processamento de imagens, foi detectada a necessidade de remoção de sombras da imagem, as quais comprometiam a identificação e extração do caminho da imagem da plantação. A segunda etapa foi a transferência dos *pixels* da imagem segmentada para um algoritmo de reconhecimento do ângulo de direção do caminho. Máquinas de vetores de suporte vêm sendo utilizadas em várias pesquisas direcionadas ao reconhecimento de padrões. A escolha desse tipo de estrutura para a tarefa de reconhecimento do ângulo de navegação se deve à sua semelhança e capacidade de generalização com as redes neurais artificiais. Além disso, o tempo de treinamento de uma SVM é menor do que em redes neurais artificiais multi camadas. O percentual de acerto obtido no reconhecimento do ângulo de direção foi superior à 90%, considerando os acertos exatos e aproximados.

A análise de tempo do reconhecimento da SVM no dispositivo FPGA permite concluir que o desenvolvimento de circuitos dedicados específicos para um algoritmo pode reduzir significativamente o tempo de execução. Conforme os testes realizados no FPGA de 50 Mhz, o tempo de reconhecimento do algoritmo foi aproximadamente $^{5}/_{4}$ maior em relação ao tempo de reconhecimento no *software*. No entanto, segundo estimativas de tempo realizadas em circuitos com frequência de *clock* maior, é possível reduzir este tempo em até 10 vezes

utilizando um dispositivo de 500 Mhz. Neste caso, o tempo de reconhecimento estimado é de aproximadamente 105 microssegundos, sendo menor que o tempo aproximado de 800 microssegundos em *software*. Portanto, o ganho de desempenho em relação ao tempo de execução ainda seria significativo utilizando um dispositivo com uma tecnologia de processamento de baixo custo.

A maior dificuldade enfrentada foi atingir um percentual de acerto exato no reconhecimento, fato este que ocorre devido à semelhança dos esqueletos que possuem valores próximos de ângulos. Entretanto, verificou-se que o reconhecimento para a maioria das imagens diferiu 5º em relação ao ângulo desejado, sendo que isto permitiu verificar a possibilidade de considerar o acerto aproximado por meio da análise da diferença da rota desejada quando esta situação ocorre. Verificou-se que uma diferença de 5º no reconhecimento causa 7 centímetros de desvio em relação à rota correta de navegação. Portanto, o tempo de execução do algoritmo deve ser rápido o suficiente para que essa aproximação seja considerada e permitir taxas de atualização do ângulo de navegação a distâncias menores. De acordo com as análises realizadas na seção 5.4.2, a execução dos algoritmos de processamento de imagens em um dispositivo FPGA de 500 Mhz, por exemplo, permitiria a execução dos algoritmos em um tempo compatível com o de navegação.

Por fim, os testes permitem concluir que a utilização de máquinas de vetores de suporte pode ser adequada para determinar a direção do robô móvel com uma margem de erro pequena em relação ao número de imagens utilizadas na verificação. Além disso, estima-se que o tempo de execução do algoritmo em um circuito dedicado pode melhorar significativamente se comparado com o tempo de execução em *software*, pois tais circuitos são projetados com conexões fixas e arquitetura adequada para execução de um algoritmo específico.

Referências bibliográficas

[1] MANDOW, A. et al. The autonomous mobile robot AURORA for greenhouse operation. **IEEE Robotics & Automation Magazine**, v. 3, n. 4, p. 18-28, dezembro de 1996.

[2] ÅSTRAND, B.; BAERVELDT, A.-J. An Agricultural Mobile Robot with Vision-Based Perception for Mechanical Weed Control. **Journal of Autonomous Robots**, v. 13, n. 1, p. 21-35, julho de 2002.

[3] VAN HENTEN, E. J. et al. An Autonomous Robot for De-leafing Cucumber Plants grown in a High-wire Cultivation. **Elsevier Biosystems Engineering**, v. 94, n. 3, p. 317-323, maio de 2006.

[4] GONZALES, J.; TAHA, Z. Mobile robot navigation using open computer vision with fuzzy controller. **Journal of Advanced Computational Intelligence and Intelligent Informatics**, v. 12, n. 4, p. 336-341, 2008.

[5] ACHMAD, B.; KARSITI, M. N. Visual-Based fuzzy navigation system for mobile robot: wall and corridor follower. In: INTERNATIONAL CONFERENCE ON INTELLIGENT AND ADVANCED SYSTEMS, 2007, Kuala Lumpur. **Proceedings of 2007 International Conference on Intelligent and Advanced Systems**, 2007, p. 244-248.

[6] ISMAIL, A. H. et. al. Vision-based system for line following mobile robot. In: IEEE SYMPOSIUM ON INDUSTRIAL ELECTRONICS & APPLICATIONS, 2009, Kuala Lumpur. **Proceedings of 2009 IEEE Symposium on industrial Electronics & Applications**, 2009, p. 642-645.

[7] FOLGHERAITER, M. et al. BioInspired Neural Controller For a Mobile Robot. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND BIOMIMETICS, 2006, Kunming. **Proceedings of 2006 IEEE International Conference on Robotics and Biomimetics**, 2006, p. 1646-1651.

[8] LI, W.; JIANG, X.; WANG, Y. Road recognition for navigation of an autonomous vehicle by fuzzy reasoning. In: IEEE INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS, 5th, 1996, New Orleans. **Proceedings of the 5th IEEE International Conference on Fuzzy**

Systems, 1996, p. 246-250.

[9] HOWARD, A.; SERAJI, H An intelligent terrain-based navigation system for planetary rovers. **IEEE Robotics & Automation Magazine**, v. 8, n. 4, p. 9-17, dezembro de 2001.

[10] RYOO, YOUNG-JAE. Neural Network Control for Visual Guidance System of Mobile Robot. **Heidelberg: Springer Berlin**, p. 685-693, 2007.

[11] GONZALEZ, R. C.; WOODS, R. E. **Processamento de imagens digitais**. São Paulo: Edgar Blücher, 2000. 528 p.

[12] FERNANDES, A. M. R. **Inteligência artificial: noções gerais**. Florianópolis: Visual Books, 2003. 160 p.

[13] MARQUES FILHO, O.; VIEIRA NETO, H. **Processamento digital de imagens**. Rio de Janeiro: Brasport, 1999. 410 p.

[14] ZHANG, T. Y.; SUEN, C. Y. A fast parallel algorithm for thinning digital patterns. **Communications of the ACM**, v. 27, n. 3, p. 236-239, Março de 1984.

[15] YÜCEER, C. OFLAZER, K. A rotation, scaling, and translation invariant pattern classification system. **Journal of Pattern Recognition**, v. 26, n. 5, p. 687-710, maio de 1993.

[16] BRAGA, A. P.; CARVALHO, A. C. P. L. F.; LUDERMIR, T. B. **Fundamentos de redes neurais artificiais**. Rio de Janeiro: LTC - Livros Técnicos e Científicos, 1998. 262 p.

[17] BABINI, M.; MARRANGHELLO, N. **Introdução às redes neurais artificiais**. São José do Rio Preto: Cultura Acadêmica, 2007. 61 p.

[18] HAYKIN, S. **Redes Neurais: Princípios e prática**. 2. ed. Porto Alegre: Bookman, 2001. 900 p.

[19] HEARST, M. Support Vector Machines. **IEEE Intelligent Systems**, p. 18-21, julho e agosto de 2008.

[20] ANGUITA, D. et al. Feed-Forward Support Vector Machine Without Multipliers. **IEEE Transactions on Neural Networks**, v. 17, n. 5, p. 1328-1331, setembro de 2006.

[21] LEONG, P. H. W.; TSOI, K. H. Field programmable gate array technology for robotics applications. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND BIOMIMETICS, 2005, Shatin. **Proceedings of 2005 IEEE International Conference on Robotics and Biomimetics**, 2005, p. 295-298.

[22] GOKHALE, M. B.; GRAHAM, P. S. **Reconfigurable Computing: Accelerating Computation with Field Programmable Gate Arrays**. Netherlands: Springer, 2005. 238 p.

[23] LOPES, J. J. Estudos e avaliações de compiladores para arquiteturas reconfiguráveis. 2007. 170f. Dissertação (Mestrado)-Universidade de São Paulo, São Carlos, 2007.

[24] XILINX INC. **Virtex-6 Family Overview**. Disponível em http://www.xilinx.com. Acesso em janeiro de 2012.

[25] ASHENDEN, P. J.; LEWIS, J. **The Designer's Guide to VHDL**. Burlington: Elsevier, 2008. 909 p.

[26] ALTERA CORPORATION. **Stratix II Device Handbook**. Disponível em http://www.altera.com. Acesso em janeiro de 2012.

[27] ALTERA CORPORATION. **Internal Memory (RAM and ROM) User Guide**. Disponível em http://www.altera.com. Acesso em janeiro de 2012.

[28] GROUT, I. **Digital Systems Design with FPGAs and CLPDs**. Burlington: Elsevier, 2008. 784 p.

[29] FINLAYSON, G. D.; DREW, M. S.; CHENG, L. Intrinsic Images by Entropy Minimization. In: EUROPEAN CONFERENCE ON COMPUTER VISION, 8th, 2004. **Proceedings of 8th European Conference on Computer Vision**, 2004, p. 582-595.

[30] XU, L.; QI, F.; JIANG, R. Shadow Removal from a Single Image. In: INTERNATIONAL CONFERENCE ON INTELLIGENT SYSTEMS DESIGN AND APPLICATIONS, 6th, 2006, Jinan. **Proceedings of the 6th International Conference on Intelligent Systems Design and Applications**, 2006, p. 1049-1054.

[31] KAHAN, W. **IEEE Standard for Binary Floating-Point Arithmetic**. Disponível em

http://grouper.ieee.org/groups/754/. Acesso em janeiro de 2012.

[32] ALTERA CORPORATION. **Nios Development Board Reference Manual, Stratix II Edition**. Disponível em http://www.altera.com. Acesso em janeiro de 2012.

Autorizo a reprodução xerográfica para fins de pesquisa.

São José do Rio Preto, 02/08/2012

_____

Assinatura