UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

Proposta de um Sistema de Tomada de Decisão para Detecção de Veículos em Movimento para FPGA

Egídio Ieno Júnior

Outubro de 2018

Itajubá M.G.

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

Egídio Ieno Júnior

Proposta de um Sistema de Tomada de Decisão para Detecção de Veículos em Movimento para FPGA

Tese submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do título de **Doutor em Ciências em Engenharia Elétrica**.

Área de concentração: Microeletrônica

Orientador: Prof. Dr. Tales Cleber Pimenta
Coorientador: Prof. Dr. Luis Manuel Garcés Socarrás

Outubro de 2018

Itajubá M.G.

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

Egídio Ieno Júnior

Proposta de um Sistema de Tomada de Decisão para Detecção de Veículos em Movimento para FPGA

Tese aprovada por banca examinadora em 25 de outubro de 2018, conferindo ao autor o título de **Doutor em Ciências em Engenharia Elétrica**.

Banca examinadora:

Prof. Dr. Tales Cleber Pimenta, UNIFEI (Orientador)
Prof. Dr. Evaldo Renó Faria Cintra, FEPI
Prof. Dr. Joel José Puga Coelho Rodrigues, INATEL
Prof. Dr. Mateus Augusto Faustino Chaib Junqueira, UNIFEI
Prof. Dr. Robson Luiz Moreno, UNIFEI

Itajubá M.G.

2018

À minha família, em especial aos meus filhos Anthony e Benício.

Agradecimentos

Primeiramente, agradeço ao criador por me manter saudável durante esta caminhada. E a minha família, pelo apoio, em especial durante o período de estudos em Cuba.

Aos professores do programa de pós-graduação em engenharia elétrica da Universidade Federal de Itajubá (UNIFEI), especialmente ao professor Dr. Tales Cleber Pimenta pela orientação e ao professor Dr. Robson Luiz Moreno pela dedicação em ensinar.

Aos colegas de curso, que através da apresentação de seminários compartilharam suas ideias e encaminharam soluções através de críticas sempre construtivas ao longo do desenvolvimento desta tese.

Aos professores da Universidade Tecnológica de Havana “José Antonio Echeverría” (CUJAE), especialmente ao professor Dr. Luis Manuel Garcés Socarrás pela sua coorientação e acolhida em Cuba, disponibilizando informações imprescindíveis ao desenvolvimento da proposta desta tese durante o período de Doutorado Sanduíche financiado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) no projeto de cooperação internacional entre o Brasil e Cuba no ano de 2015.

À Universidade Federal de Itajubá, que juntamente com o apoio financeiro da CAPES permitiu a continuidade deste trabalho no Brasil. E ao Centro Federal de Educação Tecnológica (CEFET) de Varginha MG pela liberação das atividades acadêmicas, proporcionando a dedicação integral ao doutorado.

Resumo

Os métodos pesquisados para detecção de objetos em movimento através do processamento de imagens em processadores de uso geral (*General Purpose Processors - GPPs*) apresentam, em sua maioria, uma abordagem que não permite uma implementação com bons resultados em matriz de portas programável em campo (*Field Programmable Gate Array - FPGA*). Isso ocorre devido à classificação correta dos pixels estar diretamente relacionada à implementação de técnicas mais complexas para modelar a imagem de referência e que requerem muitos recursos em termos de memória. Além disso, quase todos os métodos analisados realizam apenas o processamento da tomada de decisão clássica, sendo poucas as propostas que baseiam sua tomada de decisão na integral fuzzy. Assim, visando melhorar a classificação dos pixels durante o processo de detecção de veículos em movimento é proposta uma abordagem que realiza a fusão das tomadas de decisão fuzzy e clássica combinando técnicas convencionais de processamento digital de imagens. Dessa forma, o sistema de tomada de decisão proposto para detectar os veículos em movimento busca não comprometer os resultados em termos de classificação dos pixels mesmo utilizando uma técnica de modelagem simples para obter a imagem de referência. Essa imagem é obtida através da estimativa do valor mediano e possibilita que o sistema de detecção de veículos em movimento proposto não precise do armazenamento de várias imagens para obter a imagem de referência. Os resultados são verificados em termos de recursos ocupados, frequência máxima de operação e classificação dos pixels em FPGAs de baixo custo. Além disso, os resultados em termos de classificação dos pixels são comparados através de várias medidas com outros métodos, apresentando resultados promissores no processamento de imagens em tempo real em FPGAs de baixo custo.

Palavras-chaves: Processamento de imagens em tempo real, detecção de veículos em movimento, integral fuzzy, modelagem da imagem de referência, FPGA de baixo custo.

Abstract

Most of the methods surveyed for the detection of moving objects through of image processing in General Purpose Processors (GPPs) present an approach that does not allow successful implementation in Field Programmable Gate Array (FPGA). This is because the correct pixel classification is directly related to the implementation of more complex techniques to model the background image and that require many resources in terms of memory. Moreover, almost all of the methods analyzed perform only the classic decision-making process, with few proposals that base their decision-making on the fuzzy integral. Thus, in order to improve the pixel classification during the process of detecting moving vehicles, an approach is proposed to merge the results of the fuzzy and classic decision-making process combining conventional techniques of digital image processing. Therefore, the proposed decision-making system for detecting moving vehicles seeks to not compromise results in terms of pixel classification even using a simple technique to model the background image. This image is obtained by estimating the median value, so that the moving vehicle detection system does not require the storage of multiple images to obtain the reference image. Results in terms of occupied resource, maximum operating frequency, and pixel classification are verified in low-cost FPGA-based implementation. In addition, the results in terms of pixel classification are compared across various measures with other methods, presenting promising results in low-cost FPGA-based real-time implementations.

Keywords: Real-time image processing, moving vehicles detection, fuzzy integral, background image modeling, low-cost FPGA.

Lista de figuras

Lista de tabelas

Sumário

1 Introdução

Nesta tese é apresentada a proposta de um sistema de tomada de decisão para FPGA de baixo custo que combina técnicas convencionais de processamento digital de imagens com a integral fuzzy para detecção de veículos em movimento. Inicialmente são apresentadas neste capítulo as considerações necessárias ao entendimento da abordagem proposta. Os problemas encontrados na implementação dos sistemas de detecção de objetos existentes em FPGA são descritos. A hipótese, a metodologia de pesquisa, os objetivos, a estrutura do trabalho e as principais contribuições para o avanço do estado da arte também são apresentadas.

1.1 Visão geral

Através do processamento digital de imagens (*Digital Image Processing - DIP*) são realizadas operações matemáticas que visam melhorar as informações extraídas das imagens. Assim, é possível corrigir defeitos de aquisição e realçar informações de interesse, como as bordas de objetos. De forma geral, a imagem é modificada para sua visualização ou para fornecer informações à etapa de análise digital de imagens [1,2]. Essa etapa consiste na extração e tratamento dessas informações quantitativamente. Embora o ser humano seja muito competente em realizar tarefas de reconhecimento, nessa etapa pode-se obter medições mais rápidas e precisas. Os algoritmos desenvolvidos em visão computacional (*Computer Vision - CV*) aplicados em sistemas inteligentes buscam aproximar os resultados dessa etapa aos da percepção da visão humana. A geometria e a velocidade de objetos que existam no ambiente são alguns dos parâmetros obtidos por esses algoritmos. Além disso, o rastreamento de objetos, reconhecimento e a restauração de imagens são áreas também relacionadas [2].

Há vários métodos para detecção de objetos em movimento desenvolvidos ao longo dos anos baseados no processamento de imagens, como os apresentados em [3–37]. No entanto, a maioria das propostas pesquisadas são implementadas em software através de processadores de uso geral (*General Purpose Processors - GPPs*). Assim, o desenvolvimento de muitos desses algoritmos não considera a possibilidade de implementação em hardware. Todavia, aplicações com a necessidade de operação remota e equipamentos portáteis criaram uma nova abordagem ao processamento inerentemente sequencial, o qual apresentava, inicialmente, bons resultados com a frequência de relógio da ordem de giga-hertz, mas consumia muita energia, reduzindo a autonomia dos equipamentos desenvolvidos. Contudo, devido ao paralelismo inerente à implementação em hardware, pode-se trabalhar com uma

frequência de relógio inferior, da ordem de mega-hertz, reduzindo o consumo de energia e ainda permitindo alcançar um tempo de resposta menor.

Com a evolução contínua da capacidade de portas lógicas por unidade de área da matriz de portas programável em campo (*Field Programmable Gate Array - FPGA*) há muitas propostas baseadas no processamento de imagens sendo implementadas através de adaptações de um processamento sequencial em um processamento inerentemente paralelo [38,39]. Além disso, a evolução das arquiteturas das FPGAs nos últimos anos tem permitido a implementação em hardware de sistemas completos baseados no processamento de imagens para várias aplicações. Entre essas aplicações estão os equipamentos de inspeção visual, monitoramento e rastreamento de veículos, aquisição e análise de imagens médicas, biometria, segurança e vídeo games baseados em realidade aumentada, sendo algumas exemplificadas em [40–55]. Muitas dessas aplicações devem ser executadas em tempo real, o que envolve a análise e o processamento de pelo menos 30 imagens por segundo. Isso representa um desafio para aplicações em sistemas embarcados, portáteis e de baixo consumo de energia. Assim, nessas aplicações, nas quais o processamento de grande volume de dados em curto intervalo de tempo é necessário, há um considerável aumento na área de pesquisas em FPGA em relação à outras tecnologias já consagradas, tais como a computação sequencial, processador digital de sinal (*Digital Signal Processor - DSP*) e em circuitos integrados de aplicação específica (*Application Specific Integrated Circuit - ASIC*) [56].

Na literatura são encontradas várias pesquisas sobre implementações em FPGA realizadas através de adaptações de algoritmos para detecção de objetos em movimento desenvolvidos para GPPs, como as apresentadas em [57–60]. No entanto, nem sempre essas implementações são eficientes no uso de recursos dos dispositivos programáveis, sobretudo quando é necessário incorporar novas funcionalidades. Isso ocorre devido ao fato do uso da FPGA como plataforma de implementação apresentar uma quantidade limitada de recursos disponíveis, restringindo a quantidade de memória e as implementações com operações aritméticas em ponto fixo a um determinado número de bits. Nesses dispositivos também é possível realizar operações aritméticas com resultados próximos aos GPPs através da representação em ponto flutuante. Contudo, o aumento no consumo de recursos pode ser considerável, limitando a quantidade de etapas do processamento de imagens implementadas na FPGA [56]. Além disso, a frequência máxima de operação pode ser reduzida, tornando inviável o processamento em tempo real das imagens. Assim, para que a proposta de um sistema de tomada de decisão para detecção de veículos em movimento para FPGA apresente bons resultados deve ser obtida uma relação de compromisso aceitável entre o desempenho

em termos de classificação dos pixels, do tempo necessário para o processamento de cada imagem e dos recursos ocupados.

Algumas tarefas realizadas no processamento de imagens podem requerer a implementação de algoritmos complexos, exigindo do sistema um alto desempenho e requisitos rigorosos de flexibilidade, custo, confiabilidade e tempo de desenvolvimento [61–63]. Essas tarefas geralmente são divididas em quatro etapas conhecidas como pré-processamento, segmentação, extração e interpretação das informações [64,65]. Essas etapas são divididas em três níveis: baixo, intermediário e alto. Os processos de baixo nível envolvem a aquisição da imagem e operações primitivas, tais como a redução de ruído, melhoria no contraste de uma imagem e transformação do padrão de cor. Os processos de nível intermediário são realizados através da operação de segmentação, a qual é responsável por separar os objetos a serem analisados do resto da imagem. Além disso, a extração de características desses objetos é realizada no nível intermediário para descrevê-los. Já os processos de alto nível estão relacionados com as tarefas de interpretação das informações, nas quais são consideradas a análise, o reconhecimento e a classificação dos objetos, além do desenvolvimento de funções cognitivas associadas à visão humana [2,40,66]. A complexidade envolvida na implementação dos algoritmos em cada etapa depende dos objetivos a serem alcançados pela aplicação proposta. A Figura 1.1 apresenta as principais etapas e suas categorias baseadas no processamento de imagens.

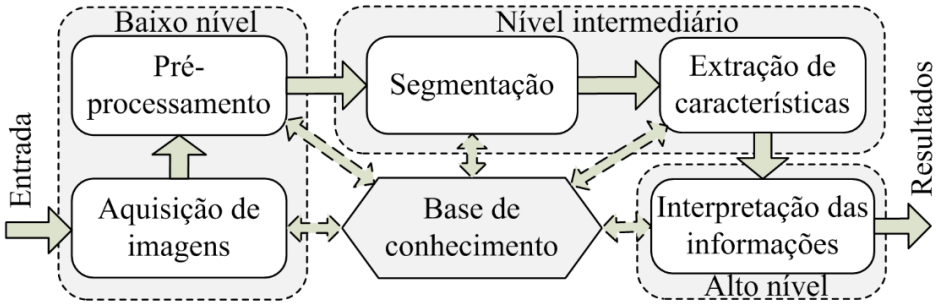


Figura 1.1 - Etapas do processamento de imagens [67].

Verifica-se através da Figura 1.1 a relação existente entre as etapas do processamento de imagens representada em cada bloco. Embora as aplicações sejam definidas na etapa implementada em alto nível, o sucesso do sistema de processamento depende das informações processadas nos níveis anteriores. O sistema proposto nesta tese atua diretamente no processo de segmentação, ou seja, na separação dos veículos em movimento da parte estática obtida através de uma imagem de referência. Esse processo de segmentação de veículos em movimento em uma sequência de imagens é considerado uma etapa crítica [68]. Dessa forma, a obtenção de informações mais confiáveis durante a etapa de segmentação reduz a

complexidade dos algoritmos implementados na etapa de alto nível, os quais são também responsáveis por corrigi-las. Sendo assim, etapas de transformação do padrão de cor *RGB* e extração de descritores de textura são propostas nesta tese. Além disso, a implementação de operações de filtragem é proposta tanto em pré-processamento, como em pós-processamento. Contudo, a base de conhecimento deve envolver técnicas de processamento de imagens que vão além das convencionais para implementar um sistema com resultados melhores em termos de classificação dos pixels. O desenvolvimento de etapas do processamento através de lógica fuzzy tem se tornado atrativo quando, por exemplo, são combinadas informações de textura e cor para detecção de objetos em movimento, como proposto em [15–18]. Do exposto, a maior dificuldade em desenvolver sistemas de detecção de veículos em movimento confiáveis em termos de classificação dos pixels é a necessidade de conhecimento nas áreas correlatas e suas fronteiras.

O sistema de visão humano é capaz de considerar informações imprecisas em termos dos dados fornecidos e do conhecimento para sua análise. De maneira geral, as pessoas utilizam uma linguagem descritiva própria para definir características que estão sujeitas à várias interpretações. As etapas baseadas no processamento de imagens implementadas através de lógica fuzzy permitem aproximar essa habilidade humana de resolver problemas através de informações capturadas pela sua visão [69].

Os conjuntos clássicos contêm elementos ou objetos que satisfazem condições precisas, sendo assim, os seus elementos pertencem ou não ao conjunto. Ao contrário do que ocorre com a teoria clássica de conjuntos, a teoria de conjuntos fuzzy é utilizada para quantificar ideias vagas e ambíguas [69]. As incertezas e a falta de precisão no processamento de imagens podem acontecer em todas as etapas, indo desde a ambiguidade em relação à informação obtida pelos valores dos pixels até a falta de informações nos processos de interpretação [69–71]. No entanto, propor soluções completas baseadas na teoria de conjuntos fuzzy em hardware apresenta como principal desvantagem a maior utilização de recursos quando comparadas as baseadas na teoria clássica de conjuntos. Esse fato ocorre, por exemplo, devido à necessidade da implementação de funções de pertinência complexas em alguns sistemas, algoritmos de agrupamentos e integrais fuzzy.

A implementação da integral fuzzy no processamento da tomada de decisão multicritério para detecção de objetos em movimento apresenta resultados melhores em termos de classificação dos pixels quando comparada a alguns métodos baseados na tomada de decisão clássica em [15–18]. Isso ocorre devido à combinação ponderada de várias informações extraídas das imagens, tais como a textura e a cor. Além disso, a função de

pertinência é considerada simples, já que pode ser implementada através das medidas de similaridade realizadas entre imagens. Embora a tomada de decisão multicritério implementada através da integral fuzzy apresente resultados promissores, ainda há muitos pixels classificados incorretamente, como os pixels pertencentes às áreas de sombras dos veículos em movimento. Essa classificação incorreta ocorre quando os pixels referentes às áreas dos veículos em movimento e de suas respectivas áreas sombreadas possuem semelhanças em cor e textura. Sendo assim, o método proposto para detectar essas sombras em [13,14] apresenta-se como um processamento capaz de auxiliar na tomada de decisão final. Nessa abordagem são combinadas através da operação de lógica clássica as informações de tonalidade, saturação e brilho extraídas do padrão de cor *HSV* para identificar os pixels referentes às áreas de sombras dos objetos em movimento. A fusão das tomadas de decisão fuzzy e clássica, as quais processam informações independentes, torna-se adequada à implementação em hardware. Contudo, um sistema capaz de melhorar a classificação dos pixels referentes às áreas dos veículos em movimento e às áreas estáticas das imagens capturadas em ambiente sem controle de iluminação pode ser proposto para FPGA.

1.2 Descrição do problema

Nesta tese são apresentados os resultados de parte do projeto de cooperação internacional na área de pesquisa sobre “Implementação de Algoritmos em Hardware Reconfigurável para o Desenvolvimento de Sistemas Embarcados” entre Brasil e Cuba financiado pela CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior). Durante o desenvolvimento de bibliotecas através do fluxo de projetos baseado em modelos utilizando os softwares MATLAB®, Simulink® e System Generator® em FPGAs da Xilinx® para implementação de etapas de monitoramento e rastreamento de veículos verificou-se que as propostas com resultados melhores, além de utilizarem técnicas complexas no processamento em alto nível, precisam corrigir as informações recebidas da etapa de segmentação. Esses sistemas normalmente apresentam a utilização de mais recursos computacionais para corrigir, por exemplo, as coordenadas utilizadas no rastreamento de veículos [54]. Assim, o sistema de tomada de decisão para detecção de veículos em movimento proposto na etapa de segmentação deve apresentar resultados confiáveis em termos de classificação dos pixels. Do exposto, iniciou-se a busca na literatura das técnicas mais simples existentes e que apresentassem possíveis soluções para os problemas envolvendo a classificação incorreta dos pixels referentes às áreas dos veículos em movimento e às áreas estáticas da imagem.

A técnica mais simples para implementação em FPGA da etapa de detecção de veículos em movimento em termos de operações matemáticas e que utilize poucos recursos em termos de memória realiza a subtração entre imagens consecutivas em escala de cinza. Embora o processamento da imagem em escala de cinza reduza a utilização de recursos comparado ao processamento de informações extraídas de algum padrão de cor, essa técnica não apresenta bons resultados em termos de classificação dos pixels. Dessa forma, mais informações são necessárias para detectar os pixels referentes às áreas dos veículos em movimento e de suas respectivas sombras devido aos problemas encontrados durante a tomada de decisão relacionados à variação de iluminação, ruído e cores dos veículos. Além disso, essa técnica apresenta o problema da duplicidade de veículos em movimento. Para resolver esse problema a subtração pode ser realizada entre a imagem atual e uma imagem de referência sem objetos em movimento, o que requer uma técnica de modelagem. No entanto, a implementação de técnicas complexas para modelar a imagem de referência nos métodos de detecção de objetos em movimento existentes para GPPs e que requerem muitos recursos em termos de memória aumenta a classificação incorreta dos pixels em FPGA. Isso ocorre devido às simplificações realizadas nas operações matemáticas e redução na quantidade de memória que é requerida pela técnica utilizada para modelar a imagem de referência.

Entre os métodos de detecção de objetos em movimento analisados em [72–74] através de implementações em GPPs, não há nenhum capaz de obter resultados ótimos na classificação dos pixels referentes às áreas dos objetos em movimento e às áreas estáticas da imagem. Desses métodos, os que utilizam técnicas mais complexas de modelagem para obter a imagem de referência apresentam resultados melhores em termos de classificação dos pixels, mas requerem muito mais recursos em termos de memória para modelar essa imagem. Outros métodos de detecção de objetos em movimento que utilizam técnicas mais simples conseguem obter a imagem de referência utilizando relativamente poucos recursos em termos de memória. Todavia, a classificação incorreta dos pixels nesses métodos de detecção de objetos em movimento aumenta. Assim, a implementação em FPGA dos melhores métodos existentes para GPPs em termos de classificação dos pixels requer muitos recursos em termos de memória.

1.3 Hipótese

Além dos problemas encontrados na implementação das técnicas mais simples de detecção de objetos em movimento pesquisadas durante a revisão bibliográfica, verificou-se que as versões dos melhores métodos em termos de classificação dos pixels propostos

originalmente para GPPs em FPGAs resultam em um aumento da classificação incorreta dos pixels. Além disso, nenhum dos métodos de detecção de objetos em movimento analisados combina o processamento paralelo de uma tomada de decisão multicritério baseada na integral fuzzy com a operação de lógica clássica para detectar objetos em movimento. Assim, o presente trabalho busca comprovar a hipótese de que é possível implementar em FPGA de baixo custo um sistema que combine a tomada de decisão fuzzy e clássica para detecção de veículos em movimento melhorando a classificação dos pixels e realizando a tomada de decisão em tempo real sem utilizar muitos recursos em termos de memória. A hipótese levantada relaciona-se diretamente com a viabilidade da implementação utilizando apenas os recursos internos da FPGA para o processamento da tomada de decisão sobre os pixels que pertencem à região dos veículos em movimento e à parte estática em imagens capturadas através de uma câmera fixa em ambiente sem controle de iluminação.

1.4 Objetivos

De forma geral, o objetivo nesta tese é desenvolver um sistema de tomada de decisão para FPGA de baixo custo capaz de melhorar a classificação dos pixels durante a detecção de veículos em movimento com resposta em tempo real utilizando uma técnica simples para obter a imagem de referência com poucos recursos em termos de memória. Para tanto, busca-se cumprir os seguintes objetivos específicos:

- combinar técnicas convencionais de processamento digital de imagens;
- combinar a tomada de decisão fuzzy e clássica em um processamento paralelo;
- realizar a tomada de decisão entre a imagem atual e uma imagem de referência modelada com apenas uma imagem armazenada;
- testar o sistema de tomada de decisão em FPGA de baixo custo;
- comprovar se os resultados em termos de classificação dos pixels são melhores em relação à implementação apenas da tomada de decisão fuzzy e comparar com outros métodos;
- verificar se há capacidade de incorporar novas funcionalidades;
- verificar se as imagens são processadas em tempo real.

1.5 Metodologia

Durante o desenvolvimento inicial do tema abordado nesta tese, a pesquisa bibliográfica é utilizada para revisar as técnicas convencionais de processamento digital de

imagens, os métodos de detecção de objetos em movimento, a teoria fuzzy aplicada ao processamento de imagens e as medidas para validar o sistema de tomada de decisão proposto. Para comprovar a hipótese levantada, o sistema de tomada de decisão proposto é implementado através do fluxo de projetos baseado em modelos utilizando os softwares MATLAB®, Simulink® e System Generator® em FPGAs de baixo custo da Xilinx®. Dessa forma, os resultados do processamento de uma sequência de imagens são obtidos de forma experimental em termos de classificação dos pixels, recursos ocupados e frequência máxima de operação. Os resultados em termos de classificação dos pixels são apresentados de forma quantitativa devido à subjetividade da análise visual realizada sobre as imagens processadas. Assim, os resultados obtidos através de várias medidas são comparados com outros métodos pesquisados na literatura e analisados.

1.6 Contribuições

O sistema de tomada de decisão proposto é realizado utilizando uma imagem de referência modelada pela estimativa do valor mediano ao invés da média aritmética entre várias imagens armazenadas no início da sequência implementada em [15,75]. Dessa forma, a necessidade de recursos em termos de memória é de apenas uma imagem armazenada. No entanto, para que o sistema de tomada de decisão apresente resultados melhores em termos de classificação dos pixels quando comparado à implementação baseada apenas na tomada de decisão fuzzy proposta em [15–17], várias informações são combinadas através de uma tomada de decisão clássica em um processamento paralelo. Enquanto na tomada de decisão fuzzy são combinadas informações de luminância, textura e cor extraídas do padrão de cor *YCbCr*, na tomada de decisão clássica são combinadas informações de tonalidade, saturação e brilho extraídas do padrão de cor *HSV.* Assim, na proposta desta tese é realizada a extração independente de várias informações que são combinadas através da integral fuzzy parametrizada para detectar os pixels referentes às áreas dos veículos em movimento e da operação de lógica clássica parametrizada para detectar os pixels referentes às áreas de sombras desses veículos. Na Figura 1.2 são apresentadas através de um diagrama de blocos as contribuições realizadas no sistema de tomada de decisão proposto no estado da arte. Os blocos da proposta original são identificados na legenda com contorno tracejado, os blocos que representam pequenas modificações com pontilhados e os blocos que são implementados na proposta atual com traço e ponto.

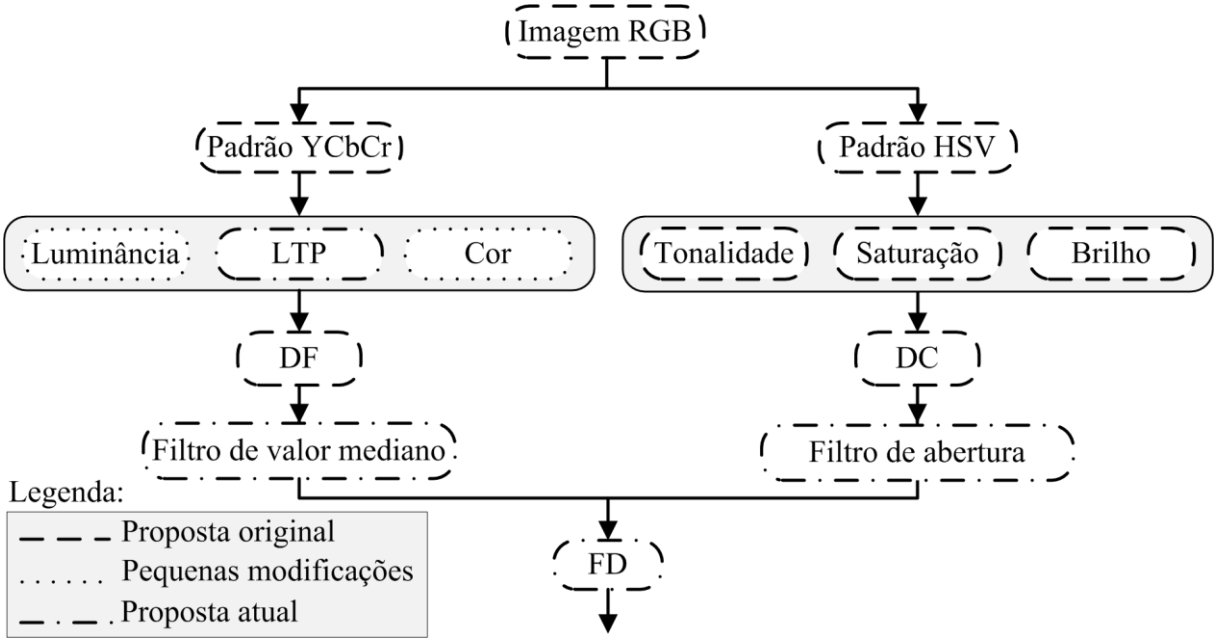


Figura 1.2 - Contribuições do sistema de tomada de decisão proposto.

Os blocos referentes ao padrão de cor *YCbCr* e *HSV* são implementados com base na proposta original para detecção dos pixels referentes às áreas dos objetos em movimento apresentada em [15–17] e de suas sombras apresentada em [13,14], respectivamente. Na proposta apresentada em [75] a remoção das áreas de sombras dos objetos em movimento é realizada após a detecção desses objetos baseada na proposta original apresentada em [15–17]. Já na proposta atual as informações de tonalidade, saturação e brilho são combinadas paralelamente ao processamento da tomada de decisão fuzzy para remoção dessas áreas. As informações de luminância e cor apresentam pequenas modificações em relação à proposta original. Assim, um filtro de valor mediano em janela 3x3 busca tornar homogênea a luminância em pequenas regiões da imagem e através de uma média aritmética entre os componentes de cor *Cb* e *Cr* é reduzida a quantidade de critérios de quatro para três na tomada de decisão fuzzy. A arquitetura do bloco para extração do descritor de textura implementa o padrão local ternário (*Local Ternary Pattern - LTP*) ao invés do padrão local binário (*Local Binary Pattern - LBP*) implementado em [15–17]. A arquitetura do bloco *LTP* é desenvolvida e implementada na proposta atual para tornar a tomada de decisão fuzzy menos sensível à variação de iluminação e ao ruído. Os blocos de *Decisão Fuzzy* (*DF*) e de *Decisão Clássica* (*DC*) são implementados com base nas propostas originais. As etapas de pós-processamento através dos filtros de valor mediano e morfológico de abertura são implementadas na proposta atual. Além disso, o bloco de *Fusão das Decisões* (*FD*) é desenvolvido e implementado para combinar os resultados das tomadas de decisão fuzzy e clássica.

Do exposto, as principais contribuições do sistema de tomada de decisão para FPGA proposto são:

- um sistema de tomada de decisão multicritério combinando a integral fuzzy com a operação de lógica clássica;
- desenvolvimento da arquitetura do bloco de medida de similaridade;
- desenvolvimento e implementação da arquitetura do bloco *LTP* na tomada de decisão fuzzy;
- implementação dos blocos dos filtros de valor mediano na decisão fuzzy;
- desenvolvimento da arquitetura do bloco de decisão fuzzy;
- implementação do filtro morfológico de abertura na decisão clássica;
- desenvolvimento da arquitetura do bloco de decisão clássica;
- desenvolvimento e implementação do bloco de fusão das decisões.

1.7 Estrutura do trabalho

O presente trabalho está dividido em seis capítulos. Neste primeiro capítulo é apresentado, de forma geral, os assuntos abordados durante o desenvolvimento da proposta desta tese. Os problemas encontrados na literatura relacionados à detecção de objetos em movimento são descritos. Além disso, a hipótese, a metodologia, os objetivos, as contribuições e a estrutura do trabalho são detalhadas.

No segundo capítulo é detalhada a teoria clássica sobre vários tipos de filtros de operação linear aplicados na detecção de veículos em movimento com áreas de sombras. As etapas de filtragem são analisadas através de implementações no software MATLAB®. Dessa forma, busca-se verificar a sua viabilidade em termos funcional para extração de características e redução na classificação incorreta dos pixels no sistema de tomada de decisão proposto nesta tese. Estudos sobre a implementação dos filtros de operação não linear e de operação morfológica no pós-processamento da tomada de decisão fuzzy e clássica são realizados. Os padrões de cor, as medidas de similaridade e de dissimilaridade são detalhadas buscando informações que melhorem a classificação dos pixels referentes às áreas dos veículos em movimento e às áreas estáticas da imagem.

No terceiro capítulo são detalhadas as técnicas de detecção de objetos em movimento consideradas mais simples e analisados os problemas encontrados nessas técnicas através de implementações no software MATLAB®. Além disso, é apresentado um estudo sobre os métodos de detecção de objetos em movimento considerados relevantes por apresentarem

bons resultados em termos de classificação dos pixels durante a revisão bibliográfica. Nesse estudo buscam-se soluções que possam ser utilizadas na implementação em FPGA proposta nesta tese e características para justificar os resultados da análise comparativa realizada no quinto capítulo.

No quarto capítulo são detalhados os conceitos sobre a teoria de conjuntos fuzzy utilizados como base para tomada de decisão fuzzy proposta no quinto capítulo. As soluções baseadas em agrupamentos e sistemas fuzzy são analisadas para verificar a viabilidade dessas técnicas em FPGA. Os conceitos envolvidos na tomada de decisão multicritério são detalhados através da integral fuzzy.

No quinto capítulo é detalhado o sistema de processamento proposto para tomada de decisão durante a detecção de veículos em movimento. A técnica para modelar a imagem de referência, as medidas, a sequência de imagens e a metodologia de implementação utilizadas para validar o sistema proposto em FPGA são detalhadas. O desempenho do sistema de tomada de decisão proposto em termos de classificação dos pixels é avaliado através da comparação de várias medidas com outros métodos de detecção de objetos em movimento. Além desses resultados, são analisados os recursos ocupados e a frequência máxima de operação obtidos através da implementação em FPGA de baixo custo.

As conclusões e os trabalhos futuros são apresentados no sexto capítulo.

Por fim, uma lista contendo os artigos publicados durante o desenvolvimento da proposta desta tese é apresentada no apêndice.

2 Técnicas convencionais de processamento digital de imagens

Neste capítulo é apresentada a teoria clássica sobre os tipos de filtros mais utilizados no processamento digital de imagens. As etapas de filtragem analisadas são apresentadas particularmente na detecção de veículos em movimento através do software MATLAB®. Os resultados visuais da implementação dos filtros de operação não linear e de operação morfológica no pós-processamento da tomada de decisão fuzzy e clássica são apresentados. Além disso, os estudos realizados sobre os padrões de cor, as medidas de similaridade e de dissimilaridade pesquisados durante a revisão bibliográfica são apresentados.

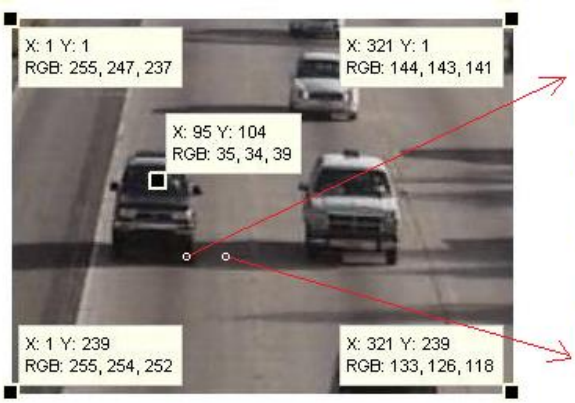
2.1 Definição de imagem

Uma imagem digital é definida como uma função bidimensional, $I(x,y)$, que para valores quantificados em cada canal de 8 bits representa a intensidade luminosa variando de 0 a 255. Em muitas aplicações a imagem é processada utilizando apenas as informações em escala de cinza obtidas através da transformação realizada na imagem composta pelas três cores primárias vermelha, verde e azul (*Red Green Blue* - *RGB*). Assim, para valores quantificados em 8 bits para cada cor primária há 24 bits representando os três componentes de cor. A definição para imagem digital permite a representação através de uma matriz com as coordenadas espaciais (x,y), onde cada ponto dessa matriz digital é denominado elemento da imagem (*Picture Element* - *Pixel*). A resolução de uma imagem está relacionada com o nível de detalhes de suas informações. A resolução pode ser representada pela quantidade de linhas (m) e colunas (n) ou ainda pela quantidade total de pixels [66]. O tamanho necessário para armazenar uma imagem depende também da quantidade de bits utilizada para representar cada pixel. Assim, o espaço de memória necessário para armazenar uma imagem em alta definição (*High Definition* - *HD*) 1280 x 720 pixels em escala de cinza é de 900 kB, já para a imagem no padrão de cor *RGB* é de 2700 kB. Os pixels que formam uma imagem são representados através da matriz (2.1).

$$I(x,y)=\begin{bmatrix} i_{1,1} & i_{1,2} & \cdots & i_{1,n} \\ i_{2,1} & i_{2,2} & \cdots & i_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ i_{m,1} & i_{m,2} & \cdots & i_{m,n} \end{bmatrix} \tag{2.1}$$

A representação de uma sequência de imagens de vídeo requer uma variável capaz de identificar as mudanças temporais, sendo assim, a função é dada como $I(x,y,t)$. A transmissão da sequência de imagens entre dispositivos é realizada com um pixel a cada ciclo de relógio. Assim, a informação é convertida da forma matricial para a vetorial. Dessa forma, quando informações referentes a mais de um pixel são necessárias simultaneamente na FPGA, a imagem é convertida da forma serial para paralela. Para as imagens coloridas, cada componente do padrão de cor é processado como um vetor de informações individuais.

As coordenadas cartesianas utilizadas no processamento de imagens apresentam sua origem no canto superior esquerdo da imagem. Na Figura 2.1 observa-se uma imagem no padrão de cor *RGB* relacionada a uma rodovia em monitoramento. Através das coordenadas de seus pontos extremos obtidas através do software MATLAB® verifica-se a resolução da imagem a ser processada.



	(114,150)									(123,150)
R:	19	21	28	36	42	46	48	51	51	53
G:	15	17	24	32	38	42	44	45	45	47
B:	12	14	21	31	37	41	43	45	45	49
Gray:	16	18	25	33	39	43	45	47	47	49
H:	26	26	12	12	12	12	12	0	0	340
S:	37	33	25	14	12	11	10	12	12	11
V:	7	8	11	14	17	18	19	20	20	21
Y:	30	31	37	44	50	53	55	56	56	58
Cb:	126	126	126	127	127	127	127	127	127	128
Cr:	130	130	130	130	130	130	130	131	131	130
Y:	16	18	25	33	39	43	45	47	47	49
I:	3	3	3	3	3	3	3	4	4	3
Q:	0	0	0	1	1	1	1	1	1	2

Figura 2.1 - Representação matricial de uma imagem em alguns padrões de cor.

Observa-se na região expandida da Figura 2.1 os valores críticos dos pixels devido à oclusão de parte do veículo da esquerda na sombra do veículo da direita. Para isso, são apresentados 10 valores referentes aos padrões *RGB, Gray* (escala de cinza), *HSV, YCbCr* e *YIQ* entre os pontos (114,150) e (123,150) na região de sombra. Nessa situação torna-se difícil segmentar o veículo mais escuro, representado pelo ponto (95,104), utilizando um limiar global no padrão de cor *RGB*. Isso ocorre porque os valores dos pixels na região de sombra apresentada são similares aos encontrados na região delimitada por esse veículo.

Na Figura 2.2 é representada em três dimensões a mesma imagem da Figura 2.1 através do software MATLAB®. Nessa imagem é possível verificar as variações devido à mudança no valor de cada pixel correspondente às posições na imagem. Além disso, verifica-se a semelhança entre as informações contidas na área de sombra comum aos dois veículos trafegando lateralmente.

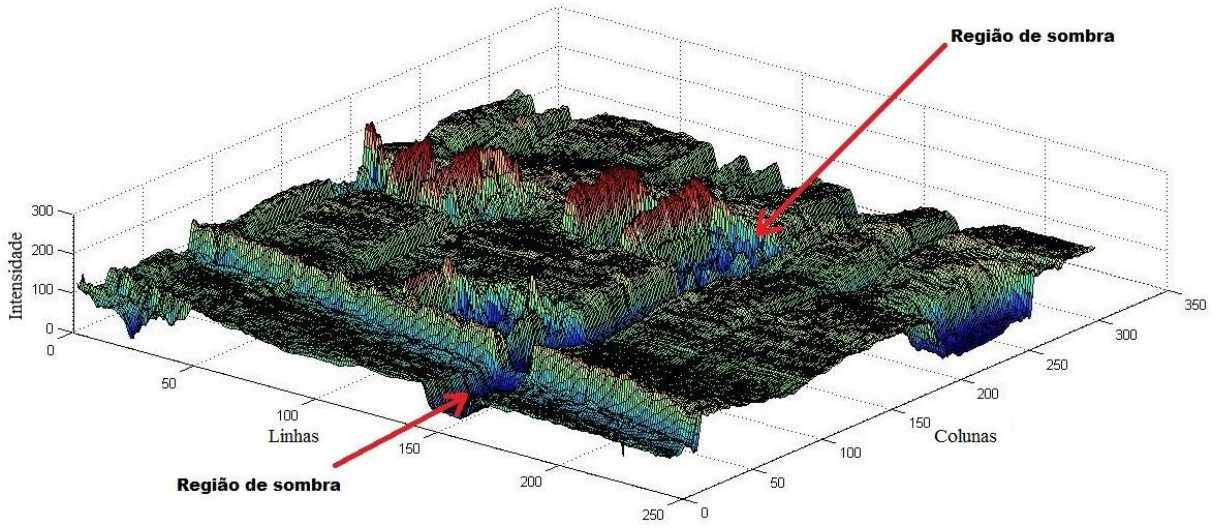


Figura 2.2 - Imagem representada em três dimensões.

2.2 Operações em imagens

A representação da imagem através de uma matriz facilita o processamento em software através de ferramentas, tais como o MATLAB®. Assim, todas as operações matriciais são válidas, podendo ser implementadas facilmente tanto as operações aritméticas (adição, subtração, multiplicação e divisão), bem como as lógicas (*E, Ou, XOu* e *NOu*). No entanto, para que seja possível a realização da operação de lógica clássica, a imagem deve ser binarizada. Assim, os pixels referentes à matriz binária da imagem apresentam valores lógicos 0 (preto) e 1 (branco), permitindo por exemplo, que as posições de veículos em movimento sejam mapeadas para a imagem original. Dessa forma, é possível realizar operações entre duas imagens em sequência ou entre a imagem atual e uma imagem de referência, buscando informações em comum. Uma operação muito empregada no processo de filtragem é a convolução em imagens. Essa operação é realizada entre os pixels de uma região da imagem e um núcleo de convolução, o qual define as mudanças a serem obtidas na imagem resultante.

Uma grande variedade de filtros digitais pode ser implementada através da convolução espacial em imagens. Para isso, basta alterar os coeficientes do núcleo de convolução. Esses filtros são os mais utilizados no processamento digital de imagens, atuando na etapa de pré-processamento para detectar bordas, suavizar contornos e reduzir o efeito dos ruídos. Para reduzir a classificação incorreta dos pixels que formam os objetos em movimento podem ser implementados filtros de valor mediano ou morfológico em etapa de pós-processamento.

2.2.1 Filtros de operação linear

As operações de multiplicação e adição realizadas em uma imagem são consideradas uma transformação linear. A convolução de duas dimensões é uma operação característica desse processo. Essa técnica permite obter, por exemplo, a detecção de bordas, realce e a suavização dos contornos dos objetos presentes nas imagens [1].

O processamento através de filtros de operações lineares inclui operações bem conhecidas como a convolução e a correlação representadas em (2.2) e (2.3), respectivamente. As operações matemáticas de convolução ($*$) e correlação ($\circ$) são realizadas entre a imagem digital $I(x,y)$ e o núcleo de convolução $C(i,j)$.

$$I(x,y)*C(i,j) = \sum_{i=-\frac{p}{2}}^{i=\frac{p}{2}} \sum_{j=-\frac{q}{2}}^{j=\frac{q}{2}} C(i,j)I(x-i,y-j) \tag{2.2}$$

$$I(x,y)\circ C(i,j) = \sum_{i=-\frac{p}{2}}^{i=\frac{p}{2}} \sum_{j=-\frac{q}{2}}^{j=\frac{q}{2}} C(i,j)I(x+i,y+j) \tag{2.3}$$

O tamanho da janela de processamento da imagem é definido em (2.4), sendo que na maioria das aplicações essas janelas são implementadas com tamanho de 3x3 ou 5x5. Assim, os valores de p e q são usualmente definidos em 2 ou 4 [76].

$$\left(x-\frac{p}{2} \le i \le x+\frac{p}{2}, y-\frac{q}{2} \le j \le y+\frac{q}{2} \right) \tag{2.4}$$

O núcleo de convolução do filtro definido pelos coeficientes C_{ij} é apresentado na matriz (2.5).

$$C(i,j) = \begin{bmatrix} C_{-\frac{p}{2},-\frac{q}{2}} & C_{-\frac{p}{2},-\frac{q}{2}+1} & \cdots & C_{-\frac{p}{2},\frac{q}{2}} \\ C_{-\frac{p}{2}+1,-\frac{q}{2}} & C_{-\frac{p}{2}+1,-\frac{q}{2}+1} & \cdots & C_{-\frac{p}{2}+1,\frac{q}{2}} \\ \vdots & \vdots & \ddots & \vdots \\ C_{\frac{p}{2},-\frac{q}{2}} & C_{\frac{p}{2},-\frac{q}{2}+1} & \cdots & C_{\frac{p}{2},\frac{q}{2}} \end{bmatrix} \tag{2.5}$$

A diferença entre as operações de convolução e correlação é a rotação do núcleo em 180^0. Dessa forma, os resultados da operação de convolução e correlação são os mesmos quando o núcleo é simétrico.

Considerando que a imagem é armazenada como uma matriz com informações discretas, é necessário produzir também uma forma discreta de distribuição dos pixels para obter o núcleo de convolução. Para o filtro com núcleo Gaussiano, por exemplo, uma forma discreta da função de distribuição gaussiana é utilizada. Já para o filtro com núcleo

Laplaciano é utilizada uma forma discreta da derivada segunda. Os cálculos para obtenção dos coeficientes dos núcleos de convolução dos filtros derivativos de primeira e segunda ordem são apresentados em [1].

Na Figura 2.3 são apresentados os núcleos de convolução dos filtros: Edge, Sobel, Laplaciano e Prewitt. Os núcleos que possuem uma coluna com todos os elementos repetidos em outra permitem a implementação de um número menor de operações matemáticas durante o processo de convolução [2,77,78].

$$\underset{\text{Edge}}{\begin{bmatrix} -1 & -1 & -1 \\ -1 & +8 & -1 \\ -1 & -1 & -1 \end{bmatrix}} \quad \underset{\text{Sobel X}}{\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}} \quad \underset{\text{Sobel Y}}{\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}} \quad \underset{\text{Sobel X-Y}}{\begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & +1 \\ 0 & +1 & +1 \end{bmatrix}}$$

$$\underset{\text{Laplaciano}}{\begin{bmatrix} 0 & -1 & 0 \\ -1 & +4 & -1 \\ 0 & -1 & 0 \end{bmatrix}} \quad \underset{\text{Prewitt X}}{\begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix}} \quad \underset{\text{Prewitt Y}}{\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix}} \quad \underset{\text{Prewitt X-Y}}{\begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & +1 \\ 0 & +1 & +2 \end{bmatrix}}$$

Figura 2.3 - Núcleos de convolução para detecção de bordas.

Na Figura 2.4 são apresentados os núcleos de convolução Blur, Smooth e Gaussiano para a suavização do valor dos pixels, reduzindo o ruído. Esses núcleos de convolução, além de serem os mais utilizados, apresentam implementações eficientes em FPGA através de uma biblioteca desenvolvida utilizando os softwares MATLAB®, Simulink® e System Generator® em [76].

$$\underset{\text{Blur}}{\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}} \quad \underset{\text{Smooth}}{\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 5 & 5 & 5 & 1 \\ 1 & 5 & 44 & 5 & 1 \\ 1 & 5 & 5 & 5 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}} \quad \underset{\text{Gaussiano}}{\begin{bmatrix} 1 & 1 & 2 & 1 & 1 \\ 1 & 2 & 4 & 2 & 1 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \\ 1 & 1 & 2 & 1 & 1 \end{bmatrix}}$$

Figura 2.4 - Núcleos de convolução para suavização.

Na Figura 2.5 exemplifica-se a operação de convolução realizada pelo filtro de operação linear.

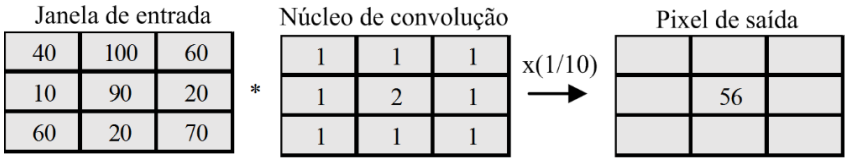


Figura 2.5 - Exemplo do filtro de convolução.

Para o núcleo de convolução utilizado, o valor normalizado do pixel central é de 56. Os valores dos demais pixels são obtidos com o deslocamento do núcleo de convolução por toda a imagem. Dessa forma, para um núcleo de convolução de tamanho 3x3 uma nova

operação é realizada a cada entrada de três novos pixels simultaneamente [76].

A detecção de borda de objetos é conseguida porque ocorre uma mudança no valor do pixel nesse local. Isso reflete o contraste entre os pixels dos objetos de uma imagem e o fundo ou entre dois objetos. No entanto, essa diferenciação é sensível ao ruído. Assim, a orientação vertical pode suavizar o ruído enquanto a horizontal detecta as bordas verticais. Nessa situação, as bordas são detectadas apenas em um sentido [1]. Alternativamente, desde que as bordas contenham informação de alta frequência, poderá ser utilizado um filtro passa-alta para detectar as bordas em todas as direções. Durante a etapa de filtragem de ruído passa-baixa deve ser verificado se as informações sobre as bordas não são perdidas [2].

Os filtros de detecção de bordas Sobel e Prewitt são compostos de três orientações em diferentes ângulos: X, Y e X-Y. A aplicação desses filtros, assim como o Edge e o Laplaciano, visa obter as bordas em preto e branco dos objetos em uma imagem após o processo de binarização, como o proposto em [79] para FPGA. O desenvolvimento matemático para obtenção do núcleo desses filtros e seu funcionamento são apresentados em [1,2,76,80]. Devido à dificuldade em distinguir mudanças no valor de um pixel contaminado pela interferência de ruído, a operação desses filtros perde informações das bordas tentando remover o ruído [2,81].

Os filtros que implementam os núcleos de convolução Blur, Smooth e Gaussiano são utilizados para diminuir o ruído em imagens. Sendo assim, esses filtros são utilizados na etapa de pré-processamento para remover pequenos detalhes em linhas e curvas [1]. Contudo, a aplicação desses filtros modifica todos os pixels da imagem [38,82].

A imagem processada pelo filtro Laplaciano é muito sensível ao ruído. Assim, geralmente deve ser processada anteriormente por um filtro Gaussiano. Como essas duas operações são lineares há a possibilidade de implementar apenas um núcleo de convolução, reduzindo o tempo de processamento e os recursos ocupados na implementação em FPGA. Esse filtro é denominado Laplaciano do Gaussiano (*Laplacian of Gaussian - LoG*) [2,83,84].

No trabalho apresentado em [85], a detecção das sombras de objetos se movendo tem seu aspecto físico bem detalhado. Nesse trabalho é proposto a implementação de um filtro que utiliza o núcleo de convolução Laplaciano e o filtro de valor mediano para eliminar pequenos fragmentos de borda. Após realizar o cálculo de variância em paralelo com a filtragem de valor mediano é implementada a operação de binarização e a detecção de cruzamento por zero, sendo essas informações combinadas através da operação de lógica clássica *E*. Na Figura 2.6 são apresentados os resultados do processo de filtragem dos núcleos de convolução mostrados na Figura 2.3. A operação de filtragem é realizada na imagem original

transformada do padrão de cor *RGB* para escala de cinza buscando reduzir a área de sombras dos veículos em movimento.

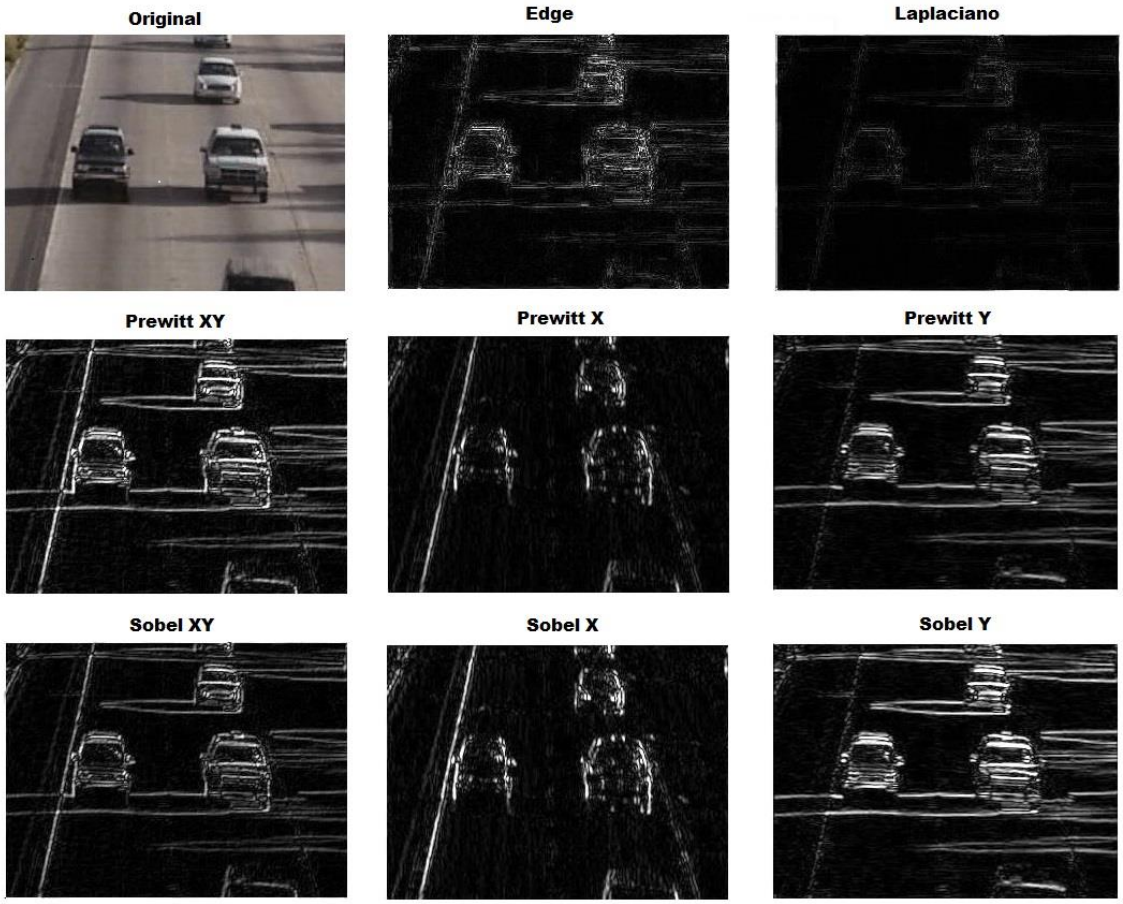


Figura 2.6 - Aplicação dos filtros de convolução em regiões de sombras com oclusão.

Na Figura 2.6 verifica-se que a implementação dos operadores orientados em X, Sobel e Prewitt, permite reduzir a área de sombras dos veículos em movimento em regiões com oclusão. Contudo, a segmentação desses veículos combinando a informação de borda extraída com esses operadores é dificultada pela similaridade entre os valores dos pixels referentes às áreas dos veículos em movimento e da imagem de referência. Assim, embora o sistema de tomada de decisão proposto nesta tese na Seção 5.3 combine várias informações para reduzir a classificação incorreta dos pixels referentes às áreas dos veículos em movimento, a medida de similaridade implementada com a informação obtida através desses filtros não permite melhorar os resultados.

2.2.2 Filtros de operação não linear

A obtenção dos valores mínimo, máximo e mediano em uma imagem é uma operação não linear. Essa operação ocorre utilizando o ordenamento espacial e é baseada na substituição do pixel central de uma área determinada pelo tamanho de uma janela por uma

posição requerida entre os valores ordenados. Um caso específico é o filtro de valor mediano, que requer a posição central dos pixels ordenados [1,2,82,86]. O desempenho desse filtro para remoção de ruído impulsivo é avaliado em [87,88].

Na Figura 2.7 exemplifica-se a operação de ordenamento e a obtenção do valor mediano em uma janela 3x3.

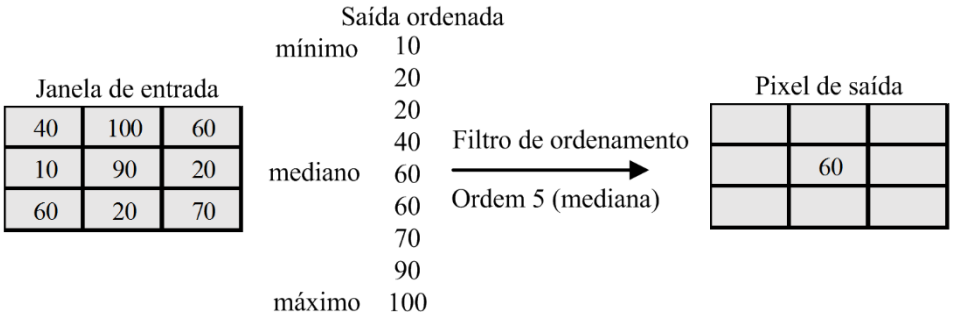


Figura 2.7 - Exemplo de operação do filtro de valor mediano.

Observe na Figura 2.7 que o valor do pixel central é de 60, diferente do calculado para o filtro de convolução apresentado na Figura 2.5, que é de 56.

A sequência de imagens para testes apresentada na Figura 5.7 é utilizada para exemplificar a implementação da operação do filtro de valor mediano após a detecção dos veículos em movimento na tomada de decisão fuzzy proposta na Seção 5.3.2. Através das imagens apresentadas na Figura 2.8 é possível analisar o resultado dessa operação. Após o processamento da imagem original na tomada de decisão fuzzy, observa-se vários pixels referentes às áreas estáticas da imagem classificados incorretamente. A redução desses pixels é observada após a operação do filtro de valor mediano.

Figura 2.8 - Exemplo de operação do filtro de valor mediano na tomada de decisão fuzzy.

O filtro de valor mediano é muito popular porque possui uma excelente capacidade de redução do ruído impulsivo [88]. Além disso, perde menos informações que os filtros lineares de suavização com janela de mesmo tamanho [1]. Esse filtro é implementado em [89] para auxiliar a remoção de ruído na etapa de pré-processamento de um sistema de detecção de objetos em movimento através da diferença entre imagens consecutivas.

2.2.3 Filtros de operação morfológica

A palavra morfologia foi empregada pela primeira vez no ramo da biologia que estuda as formas das células e de outras estruturas. No processamento de imagens, a morfologia matemática torna-se um instrumento para a extração dos componentes de imagens que são úteis na representação e descrição da forma de uma região, como o seu contorno [66]. A operação implementada nesses filtros também é considerada não linear, já que utiliza a operação de mínimo ou de máximo. Esses filtros implementam uma janela contendo os elementos estruturantes. Para modificar as propriedades geométricas dos objetos é utilizada uma operação de lógica clássica entre os elementos estruturantes e os pixels contidos na região determinada por esses elementos na imagem binária.

A linguagem da morfologia matemática é a teoria de conjuntos. Os conjuntos são utilizados para representação dos objetos em uma imagem. Dentro da morfologia matemática existem algumas operações importantes como a dilatação, erosão, extração de borda, preenchimento e extração de componentes conectados.

A dilatação e a erosão são operações fundamentais na morfologia matemática e muitos algoritmos são baseados nessas operações. A operação de dilatação de *A*, objeto da imagem, pelo elemento estruturante (*Structuring Element - SE*) *B* é representada em (2.6), em que $\hat{B}$ representa a reflexão do conjunto *B* em relação a sua origem e $\varnothing$ representa o conjunto vazio. O conjunto resultante da operação de dilatação corresponde ao conjunto de todos os pontos em *Z* tais que *B*, quando transladado por *Z*, se sobreponham em pelo menos um elemento não nulo. A implementação da operação de dilatação é realizada através da obtenção do valor máximo entre os pixels em uma janela determinada pelos elementos estruturantes. Os elementos nulos de cada coluna do *SE* modificam os pixels unitários correspondentes na imagem através da operação de lógica clássica *E*, assim, os pixels referentes aos elementos unitários mantém o valor original [1,76,90].

$$A \oplus B = \left\{ Z \,|\, (\hat{B})_Z \cap A \neq \varnothing \right\} \tag{2.6}$$

Na Figura 2.9 é exemplificada uma operação de dilatação com dois elementos estruturantes de tamanhos diferentes, 3x3 e 5x3. Nesse exemplo é preciso que o centro do elemento estruturante satisfaça a condição de dilatação. Assim, toda região do elemento estruturante é marcada. A cor cinza no objeto representa as posições nas quais o elemento estruturante é marcado.

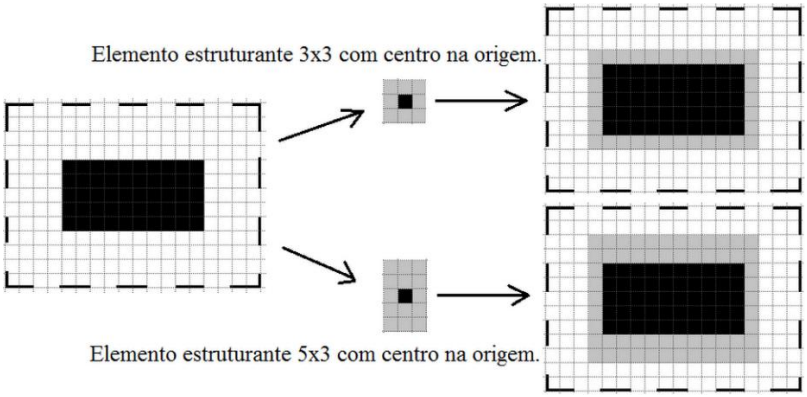


Figura 2.9 - Ilustração da operação de dilatação com elemento estruturante 3x3 e 5x3.

O conjunto resultante da operação de erosão corresponde ao conjunto de todos os pontos em *Z* tais que *B*, quando transladado por *Z*, esteja contido em *A*. Assim, não há elementos sobrepostos. A implementação da operação de erosão é realizada através da obtenção do valor mínimo entre os pixels em uma janela determinada pelos elementos estruturantes. Os elementos unitários de cada coluna do *SE* modificam os pixels nulos correspondentes na imagem através da operação de lógica clássica *Ou*, assim, os pixels referentes aos elementos nulos mantém o valor original [1,76,90]. A operação de erosão de *A* pelo elemento estruturante *B*, é representada em (2.7).

$$A \ominus B = \left\{ Z \,|\, (\hat{B})_Z \cap A = \varnothing \right\} \tag{2.7}$$

Na operação de erosão ocorre justamente o oposto da dilatação, ao invés dos objetos aumentarem de tamanho, esses são reduzidos. Na Figura 2.10 é exemplificada uma operação de erosão com dois elementos estruturantes de tamanhos diferentes, 3x3 e 5x3. A cor cinza no objeto representa as posições nas quais o elemento estruturante é marcado.

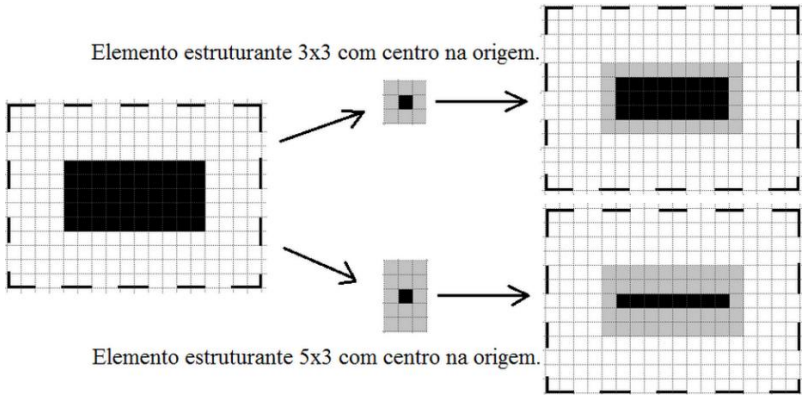


Figura 2.10 - Ilustração da operação de erosão com elemento estruturante 3x3 e 5x3.

As operações de abertura e fechamento são compostas por operações de dilatação e erosão, sendo opostas. Enquanto a operação de abertura tende a abrir fendas e desconectar objetos na imagem, a operação de fechamento conecta objetos e fecha fendas.

Na operação de abertura é realizada primeiramente a operação de erosão e depois a dilatação, conforme definida em (2.8) [1,76,90].

$$A \circ B = (A \ominus B) \oplus B \tag{2.8}$$

Na operação de fechamento é realizada primeiramente a operação de dilatação e depois a erosão, conforme definida em (2.9) [1,76,90].

$$A \bullet B = (A \oplus B) \ominus B \tag{2.9}$$

As operações morfológicas sobre a imagem são desenvolvidas através de algoritmos que analisam a sua estrutura geométrica. Essas operações podem ser utilizadas tanto em imagem binária, em escala de cinza ou colorida [91,92]. Como o operador morfológico utiliza um elemento estruturante para processar a imagem, a alteração do *SE* permite a obtenção de diferentes resultados para a mesma operação [66,91]. Na Figura 2.11 são apresentados os elementos estruturantes básicos utilizados para a janela 3x3 e na Figura 2.12 os elementos estruturantes básicos utilizados para a janela 5x5.

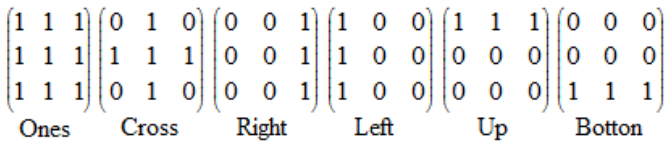


Figura 2.11 - Elementos estruturantes básicos para janela 3x3.

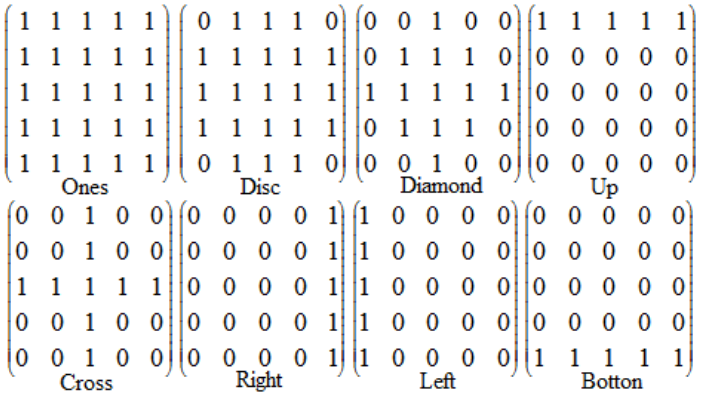


Figura 2.12 - Elementos estruturantes básicos para janela 5x5.

Uma das principais potencialidades dos operadores morfológicos é a possibilidade de obter vários resultados com as operações básicas implementadas em ordem trocada. Essas operações são muito utilizadas em restauração de imagem [66,93]. Na literatura são encontradas várias propostas para melhorar os resultados em termos de classificação dos pixels através de operações morfológicas. Alguns métodos de detecção de objetos em movimento, como os apresentados em [55,68,94] empregam as operações morfológicas em

etapas de pós-processamento, alcançando resultados em termos de classificação dos pixels próximos a de algoritmos mais complexos.

A sequência de imagens para testes apresentada na Figura 5.7 é utilizada para exemplificar a implementação da operação morfológica de abertura após a detecção de áreas de sombras dos veículos em movimento na tomada de decisão clássica proposta na Seção 5.3.1. Através das imagens apresentadas na Figura 2.13 é possível analisar o resultado dessa operação. Após o processamento da imagem original na tomada de decisão clássica, observa-se vários pixels referentes às áreas dos veículos em movimento classificados incorretamente como pertencentes às áreas de sombras. Nesse exemplo as pequenas regiões de pixels isoladas são eliminadas na operação de erosão e as regiões maiores são recuperadas pela dilatação após o processamento do filtro morfológico de abertura.

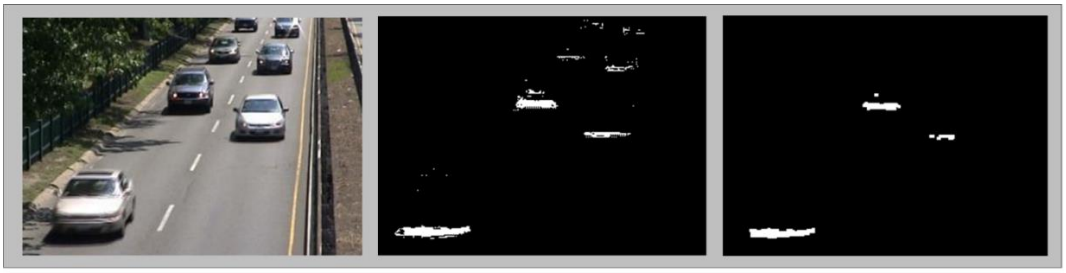


Figura 2.13 - Exemplo de operação do filtro morfológico de abertura na tomada de decisão clássica.

A operação de erosão não elimina apenas a informação da imagem que não possua elemento estruturante, mas também reduz os demais objetos. Além disso, a operação de dilatação pode unir objetos reduzindo a quantidade detectada. Dessa forma, o tamanho do elemento estruturante e a quantidade de vezes que a operação é realizada devem ser cuidadosamente considerados.

2.3 Padrões de cor

A informação inicial a ser processada em uma imagem está relacionada ao padrão de cor, que normalmente é o *RGB*. Há várias propostas na área de detecção de objetos em movimento que implementam a transformação desse padrão para a escala de cinza. O processamento em escala de cinza reduz a quantidade de recursos computacionais, mas dependendo da aplicação pode comprometer os resultados. Dessa forma, existem algoritmos desenvolvidos com base no processamento de informações de cores, já que, por exemplo, não há como separar os pixels referentes às áreas de sombras dos respectivos objetos em movimento se esses possuem valores próximos em escala de cinza. Dentre os padrões de cor mais implementados em trabalhos relacionados à detecção de objetos em movimento estão o

YIQ, *YCbCr* e o *HSV*.

A transformação do padrão de cor *RGB* para escala de cinza [95] é realizada através da equação (2.10). Essa equação é proposta para aproximar a percepção da visão humana do brilho através de pesos diferentes aplicados em cada componente do padrão de cor *RGB* e é utilizada na função *rgb2gray* (*RGB*) do software MATLAB® [96]. Uma implementação otimizada em FPGA em termos de recursos para essa conversão é apresentada em [48]. Essa implementação utiliza registradores de deslocamento e adição ao invés de multiplicadores.

$$Escala\ de\ cinza\ = \ 0,299R+0,587G+0,114B \tag{2.10}$$

Do ponto de vista de recursos computacionais, uma maior simplificação pode ser realizada através da média aritmética entre os componentes do padrão de cor *RGB*, ou seja, atribuindo um peso igual. No entanto, os valores dos pixels obtidos ficam mais distantes da percepção da visão humana [96].

O padrão de cor *YIQ* é o padrão definido pelo comitê nacional do sistema de televisão (*National Television Systems Commitee - NTSC*). O componente *Y* desse padrão de cor representa a luminância, enquanto os componentes *I* e *Q* as informações da cor, as quais indicam a tonalidade e a saturação, respectivamente. A vantagem desse padrão de cor é a disponibilidade direta da informação da imagem em escala de cinza através do componente *Y* [97].

O padrão de cor *YIQ* é convertido do padrão *RGB* através das equações definidas em (2.11), (2.12) e (2.13) [98].

$$Y \ = \ 0,299R+0,587G+0,114B \tag{2.11}$$

$$I \ = \ 0,596R-0,274G-0,321B \tag{2.12}$$

$$Q \ = \ 0,212R-0,523G+0,311B \tag{2.13}$$

Observa-se em relação às equações matemáticas para conversão do padrão de cor *RGB* para *YIQ* que a soma dos coeficientes da equação (2.11) é igual a 1, enquanto a soma dos coeficientes das equações (2.12) e (2.13) é 0. Dessa forma, para uma imagem convertida em escala de cinza, em que todos os componentes *R*, *G* e *B* são iguais, os componentes *I* e *Q* correspondem a 0.

No padrão de cor *YCbCr*, amplamente empregado em vídeo digital, o componente *Y* corresponde a luminância, enquanto os componentes *Cb* e *Cr* representam as informações de crominância [99]. A conversão do padrão de cor *RGB* para *YCbCr* é realizada através da equação (2.11) apresentada anteriormente e das equações (2.14) e (2.15) [100].

$$C_b \ = \ -0,169R-0,331G+0,500B+128 \tag{2.14}$$

$$C_r = +0,500R - 0,419G - 0,081B + 128 \tag{2.15}$$

O padrão de cor *RGB* possui valores entre 0 e 255 para representação de cada pixel em 24 bits. Já o padrão *HSV* possui o componente *H* variando de 0^0 a 360^0 e os componentes *S* e *V* variando de 0% a 100% [101]. A cada valor do componente do padrão *RGB*, haverá a necessidade de obter o maior e o menor valor. As equações (2.16), (2.17) e (2.18) apresentam os cálculos iniciais para a conversão do padrão *RGB* em *HSV*.

$$C_{\max} = \max\ R,G,B \tag{2.16}$$

$$C_{\min} = \min\ R,G,B \tag{2.17}$$

$$\Delta = C_{\max} - C_{\min} \tag{2.18}$$

A obtenção do componente que define a tonalidade ou matiz (*Hue* - *H*) é condicional, sendo a sua conversão dependente do componente de cor predominante no padrão *RGB* (C_{max}). O cálculo desse componente é apresentado em (2.19) [101,102].

$$H = \begin{cases} 0 & , \text{se } C_{\max} = C_{\min} \\ \left(\frac{G - B}{\Delta}\right)\times 60° & , \text{se } C_{\max} = R \text{ e } G \geq B \\ \left(\frac{G - B}{\Delta}\right)\times 60° + 360°, & \text{se } C_{\max} = R \text{ e } G < B \\ \left(\frac{B - R}{\Delta} + 2\right)\times 60° & , \text{se } C_{\max} = G \\ \left(\frac{R - G}{\Delta} + 4\right)\times 60° & , \text{se } C_{\max} = B \end{cases} \tag{2.19}$$

A obtenção da pureza ou saturação (*Saturation* - *S*) também é dependente do valor obtido em C_{max}. Quanto menor esse valor, mais a imagem se aproxima de uma imagem em escala de cinza. Em caso contrário, mais a imagem se apresenta como uma cor pura. O cálculo desse componente é apresentado em (2.20) [101].

$$S = \begin{cases} 0 & , \text{se } C_{\max} = 0 \\ \frac{\Delta}{C_{\max}}\times 100, & \text{se } C_{\max} \neq 0 \end{cases} \tag{2.20}$$

O cálculo do brilho ou valor (*Value* - *V*) é apresentado em (2.21) [101].

$$V = \frac{C_{\max}}{255}\times 100 \tag{2.21}$$

2.4 Medidas de similaridade

A detecção de objetos em movimento através da comparação entre a imagem atual e uma imagem de referência é implementada em muitos trabalhos através de uma simples subtração entre essas imagens. Outra possibilidade para estabelecer essa comparação consiste em definir medidas de similaridade entre os pixels da imagem atual e da imagem de referência. Nesse caso, os pixels correspondentes à imagem de referência devem ser semelhantes nas duas imagens, enquanto os pixels dos objetos em movimento devem ser diferentes. O cálculo das medidas de similaridade pode ser realizado através de componentes dos padrões de cor que mais se aproximam da percepção da visão humana. Além disso, para maior robustez contra as variações de iluminação, a combinação de informações pode considerar medidas de similaridade entre os descritores de textura [17].

Uma forma para obter os resultados da medida de similaridade entre 0 e 1 é através da operação de divisão entre mínimos e máximos do conjunto formado pelo valor em cada posição (*x*,*y*) na imagem atual ($I^a(x,y)$) e na imagem de referência ($I^r(x,y)$) [103]. Essa forma de representação para medida de similaridade é apresentada em (2.22).

$$S(x,y) = \frac{\min\left\{I^a(x,y), I^r(x,y)\right\}}{\max\left\{I^a(x,y), I^r(x,y)\right\}} \tag{2.22}$$

Os valores correspondentes a cada pixel na imagem atual e na imagem de referência podem ser obtidos diretamente da transformação entre os padrões de cor. Essa representação também pode ser adotada para medida de similaridade em textura.

2.4.1 Medida de similaridade em padrão de cor

A medida de similaridade implementada em [15,18], de forma geral, é descrita para qualquer padrão de cor com três componentes C_1, C_2 e C_3. Essa medida de similaridade representada por $S_k^C(x,y)$ na posição (*x*,*y*) é apresentada em (2.23).

$$S_k^C(x,y) = \begin{cases} \dfrac{I_k^a(x,y)}{I_k^r(x,y)}, \text{ se } I_k^a(x,y) < I_k^r(x,y) \\ 1 \quad\quad , \text{ se } I_k^a(x,y) = I_k^r(x,y) \\ \dfrac{I_k^r(x,y)}{I_k^a(x,y)}, \text{ se } I_k^a(x,y) > I_k^r(x,y) \end{cases} \tag{2.23}$$

O índice *k* pertencente ao conjunto {1,2,3} representa os três componentes do padrão de cor, *a* e *r* indicam respectivamente a imagem atual e a de referência. O resultado da medida

de similaridade $S_k^C(x,y)$ está entre 0 e 1, assim, $S_k^C(x,y)$ é próximo de 1 quando $I^a(x,y)$ e $I^r(x,y)$ possuem informações semelhantes de cor. Dessa forma, os resultados das medidas de similaridade apresentam números no domínio fuzzy.

2.4.2 Medida de similaridade em textura

Um dos métodos utilizados para obter a textura entre um pixel central e seus vizinhos define um padrão local binário (*Local Binary Pattern - LBP*). Como os valores dos pixels vizinhos são trocados por 0 ou 1 de acordo com a comparação realizada com o valor do pixel central, a análise da imagem através dos descritores *LBP* torna-se menos sensível às alterações nesses valores. Consequentemente, a tomada de decisão realizada entre os descritores extraídos de uma imagem atual e de uma de referência torna-se mais robusta à variação da iluminação [17]. No operador *LBP* apresentado em [104,105] é calculado um descritor entre uma quantidade de pixels (P) de uma vizinhança circular de raio (R). O operador *LBP* é definido em (2.24), sendo que (x_c,y_c) representa a posição do pixel central.

$$LBP_{P,R}(x_c, y_c) = \sum_{i=0}^{P-1} s(g_p - g_c) 2^P \tag{2.24}$$

O valor de g_c corresponde ao pixel central e g_p representa os pixels contidos em P vizinhos. A função $s(g_p - g_c)$ é definida em (2.25).

$$s(g_p - g_c) = \begin{cases} 1, \text{ se } g_p \geq g_c \\ 0, \text{ se } g_p < g_c \end{cases} \tag{2.25}$$

Esses descritores de textura podem ser comparados através da implementação de histogramas. No entanto, a medida de similaridade entre imagens através do cálculo de histogramas dos descritores utiliza uma informação geral sobre as imagens. Sendo assim, não há nenhuma relação entre os descritores e sua posição na imagem. Do exposto, a medida de similaridade em textura implementada nesta tese na Seção 5.3.2 baseia-se na proposta desenvolvida em [15,18]. Essa medida é apresentada em (2.26), sendo calculada para cada posição da imagem.

$$S^T(x,y) = \begin{cases} \dfrac{T^a(x,y)}{T^r(x,y)}, \text{ se } T^a(x,y) < T^r(x,y) \\ 1 \quad , \text{ se } T^a(x,y) = T^r(x,y) \\ \dfrac{T^r(x,y)}{T^a(x,y)}, \text{ se } T^a(x,y) > T^r(x,y) \end{cases} \tag{2.26}$$

Os termos $T^a(x,y)$ e $T^r(x,y)$ representam a textura do pixel localizado na posição (x,y) da imagem atual e da imagem de referência, respectivamente. O resultado da medida de

similaridade $S^T(x,y)$ está entre 0 e 1, assim, $S^T(x,y)$ é próximo de 1 quando $T^a(x,y)$ e $T^r(x,y)$ apresentam informações de textura semelhantes.

No descritor *LBP* o limiar de comparação utilizado para obter o código binário é exatamente o pixel central, portanto, o processamento tende a ser sensível ao ruído em regiões das imagens nas quais há valores de pixels próximos e padrão uniforme. Para reduzir esse problema, em [106] é apresentado um padrão local ternário (*Local Ternary Pattern - LTP*) que quantiza a diferença em três níveis utilizando uma faixa de tolerância (τ). Assim, o padrão ternário é dividido em dois descritores *LBP*s, sendo um correspondente à parte positiva e outro à negativa. Dessa forma, para obter apenas um valor decimal como no descritor *LBP*, esses dois padrões são concatenados. As condições para a codificação do descritor *LTP* proposto em [106] são definidas em (2.27).

$$s\left(g_p, g_c, \tau\right) = \begin{cases} 1, \text{ se } g_p \geq (1+\tau)g_c \\ 0, \text{ se } \left|g_p - g_c\right| < \tau g_c \\ -1, \text{ se } g_p \leq (1-\tau)g_c \end{cases} \tag{2.27}$$

Na Figura 2.14 é apresentado um exemplo de codificação baseado no descritor *LTP* para uma faixa de tolerância de 10% do valor do pixel central.

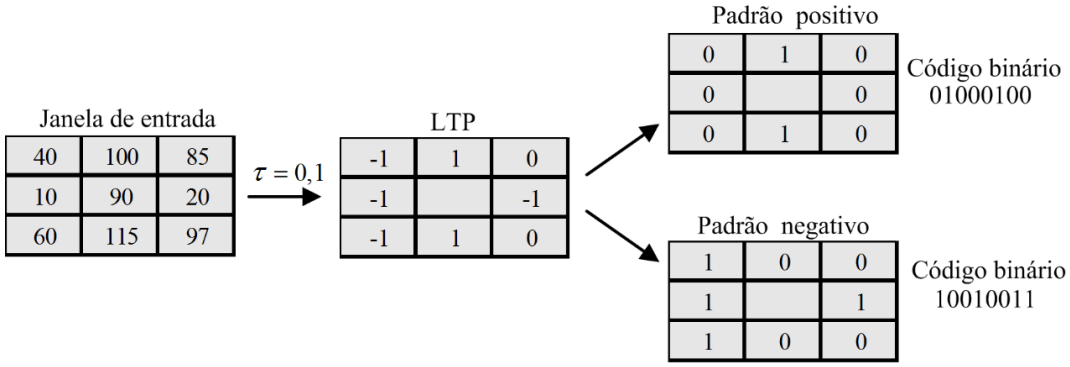


Figura 2.14 - Exemplo de codificação *LTP*.

Vários pesquisadores estão buscando melhorar os resultados de detecção de objetos em movimento através de informações de textura. O trabalho apresentado em [107], por exemplo, utiliza o conceito baseado no *LBP* para calcular os descritores não apenas para os pixels em uma imagem, como também entre imagens no intuito de detectar os objetos em movimento. Nessa proposta os objetos considerados em movimento são classificados utilizando um pixel central na imagem de referência e seus pixels vizinhos na imagem atual. Ao invés de utilizar a distância Euclidiana para medir a dissimilaridade entre duas amostras conforme proposto no algoritmo original, em [108] é utilizada a medida de distância de Manhattan. Segundo os autores, além da simplificação em termos de recursos

computacionais, essa medida de distância apresenta um desempenho superior.

Um padrão local ternário invariante à variação da iluminação (*Scale Invariant Local Ternary Pattern* - *SILTP*), proposto inicialmente em [109], pode ser considerado computacionalmente eficiente, já que implementa apenas uma comparação a mais que o *LBP* para cada pixel vizinho. Além disso, acrescenta também um limiar que aumenta a tolerância ao ruído. Há três valores possíveis para cada comparação do *SILTP* que são codificados em dois bits. Isso faz com que o descritor se torne invariante, mesmo quando há mudança repentina na iluminação [109–111]. No entanto, a codificação ocorre com 16 bits ao invés dos 8 bits do descritor *LBP*. A forma de obter o descritor *SILTP* é definida em (2.28).

$$SILTP^{\tau}_{P,R}\left(x_c, y_c\right) = \bigoplus_{i=0}^{P-1} s\left(g_p, g_c, \tau\right) \tag{2.28}$$

O operador $\oplus$ representa em (2.28) a concatenação de cada um dos dois bits codificados na vizinhança determinada pelo raio. A codificação é definida em (2.29).

$$s\left(g_p, g_c, \tau\right) = \begin{cases} 01, \text{ se } g_p \geq (1+\tau) g_c \\ 00, \text{ se } \left|g_p - g_c\right| < \tau g_c \\ 10, \text{ se } g_p \leq (1-\tau) g_c \end{cases} \tag{2.29}$$

De acordo com a proposta apresentada em [112], o descritor *SILTP* é modificado para reduzir a quantidade de padrões binários possíveis. Essa modificação é implementada através de um método que verifica se o padrão é uniforme. Assim, para transições de 0 para 1 ou de 1 para 0 o padrão pode ter no máximo duas alterações. Segundo os autores, o descritor denominado *USILTP* (*Uniform SILTP*) apresenta desempenho superior ao *SILTP* e *LBP* em termos de classificação dos pixels correspondentes às áreas dos veículos em movimento.

2.4.3 Medidas de dissimilaridade

Muitos autores consideram a medida de distância como uma medida de similaridade. Assim, uma medida de distância inferior a um determinado limiar estabelece a semelhança entre os pixels [1]. No entanto, há autores que consideram os cálculos de distância como medidas de dissimilaridade ou proximidade entre pontos [113,114]. Dessa forma, ao contrário das medidas de similaridade apresentadas, quanto maior for a medida de dissimilaridade, menor será a semelhança entre os pontos. Embora os resultados dos cálculos de algumas medidas representem diretamente a similaridade e outras a dissimilaridade, essas medidas interpretadas adequadamente trazem a mesma informação.

As medidas de distância são muito implementadas na classificação de objetos em grupos [115], como os pixels correspondentes aos grupos de objetos em movimento e

estáticos. Uma das medidas mais utilizadas é a de Minkowski, apresentada na equação (2.30) em três dimensões. Nessa equação, $d(p_1,p_2)$ representa a distância entre as coordenadas (x_1,y_1,z_1) e (x_2,y_2,z_2) dos pontos p_1 e p_2, respectivamente.

$$d(p_1,p_2) = \sqrt[q]{|x_1-x_2|^q+|y_1-y_2|^q+|z_1-z_2|^q} \tag{2.30}$$

Um caso especial da medida apresentada na equação (2.30) é quando $q = 2$. Nessa situação, obtém-se a distância Euclidiana em duas dimensões, dada pela equação (2.31).

$$d(p_1,p_2) = \sqrt{|x_1-x_2|^2+|y_1-y_2|^2} \tag{2.31}$$

Através da medida de Minkowski ainda é possível derivar mais duas medidas de distância muito utilizadas. A medida de distância de Manhattan ocorre para $q = 1$, sendo apresentada na equação (2.32).

$$d(p_1,p_2) = |x_1-x_2|+|y_1-y_2| \tag{2.32}$$

Quando q tende ao ∞, tem-se a distância de Chebyshev. Assim, a distância de Chebyshev em duas dimensões é dada em (2.33).

$$d(p_1,p_2) = \max(|x_1-x_2|,|y_1-y_2|) \tag{2.33}$$

A definição do valor de q depende do peso considerado para distâncias maiores. Assim, quanto maior o valor de q maior a sensibilidade da medida a distâncias maiores.

Na Figura 2.15 são exemplificadas as medidas apresentadas anteriormente.

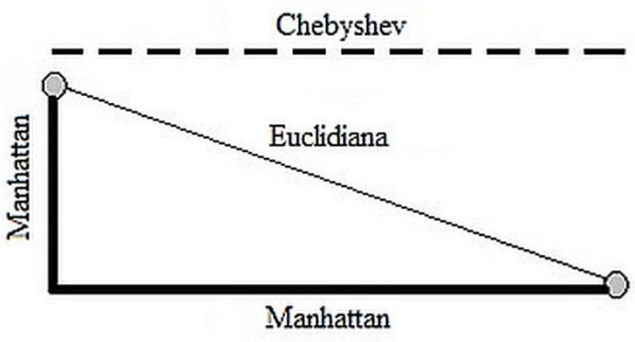


Figura 2.15 - Exemplificando as medidas de distância Euclidiana, Manhattan e Chebyshev.

2.5 Considerações finais

Um dos objetivos do sistema de detecção de veículos em movimento proposto é melhorar a classificação dos pixels durante a tomada de decisão. Assim, apenas os pixels referentes às áreas dos veículos em movimento devem ser considerados em movimento. Dessa forma, embora as operações de filtragem implementadas através de filtros lineares orientados em X permitam reduzir as áreas de sombras dos veículos em movimento, não contribuem para

a classificação dos pixels no sistema de tomada de decisão proposto na Seção 5.3. Já a operação de filtragem não linear implementada através do valor mediano permite reduzir os efeitos do ruído impulsivo sobre as imagens. Contudo, a classificação incorreta dos pixels após o processamento da tomada de decisão fuzzy também pode ser reduzida. Além disso, a implementação da operação morfológica pode reduzir a classificação incorreta dos pixels referentes às áreas de sombras dos veículos em movimento após o processamento da tomada de decisão clássica.

A implementação da transformação do padrão *RGB* para padrões de cor que apresentam informações mais próximas das interpretadas pelo sistema de visão humana pode melhorar a classificação dos pixels durante a tomada de decisão. As medidas de similaridade analisadas permitem o processamento de informações diretamente no domínio fuzzy. Além disso, a medida de similaridade realizada entre os descritores de textura *LTP* fornece informação mais robusta à variação da iluminação e ao ruído com uma implementação mais simples comparada ao *SILTP* e ao *USILTP*.

3 Detecção de objetos em movimento

Neste capítulo é apresentado um estudo sobre as técnicas de detecção de objetos em movimento consideradas mais simples e analisados os problemas encontrados nessas técnicas através de implementações no software MATLAB®. Além disso, é apresentado um estudo sobre os métodos de detecção de objetos em movimento mais promissores analisados durante a revisão bibliográfica.

3.1 A subtração entre imagens

O processamento mais simples para detectar movimento utilizando uma câmera convencional fixa é a subtração entre imagens. Para que isso ocorra, uma primeira imagem sem objetos em movimento deve ser capturada e mantida como referência. No entanto, a obtenção da imagem de referência sem veículos em movimento em rodovias com tráfego intenso requer a implementação de uma técnica de modelagem. Para cada nova imagem capturada, a sua diferença absoluta para a imagem de referência é calculada [116]. Assim, para todas as posições da imagem que essa diferença for superior a um determinado limiar, o pixel é classificado como parte de um objeto em movimento. Um problema dessa técnica de detecção de objetos em movimento é a necessidade de atualização da imagem de referência para que, por exemplo, um veículo recentemente estacionado em frente à câmera, seja incorporado a essa imagem. Dos trabalhos apresentados em [117–120], os problemas mais críticos encontrados na subtração pela imagem de referência são relacionados a seguir.

- Os objetos que constituem a imagem de referência podem iniciar movimento e novos objetos podem ser considerados estáticos. Essas mudanças devem ser consideradas para que não ocorra detecção incorreta indefinidamente.
- As variações na iluminação modificam a imagem de referência. Esse problema torna-se crítico em dias nublados, alterando os valores dos pixels da imagem em poucos segundos.
- O modelo da imagem de referência nem sempre é completamente estático. Assim, podem existir, por exemplo, árvores e arbustos balançando com o vento ou ainda as ondulações na água.

- Em muitas situações práticas, como em local de tráfego intenso de veículos, um período de treinamento no qual apenas informações estáticas são observadas durante a obtenção da imagem de referência não é possível.
- A textura de um objeto em movimento pode ser semelhante à textura da imagem de referência.
- As alterações no interior de objetos em movimento uniformemente coloridos são de difícil detecção.
- As sombras não fazem parte dos objetos em movimento. Um sistema de detecção deve classificar os pixels referentes às áreas de sombras desses objetos como sendo estáticos.

Os quatro primeiros problemas apresentados podem ser minimizados com soluções aplicadas nas etapas de inicialização, treinamento e atualização do modelo da imagem utilizado como referência, o qual deve acompanhar as alterações de variação da iluminação ao longo do dia. Para os demais problemas, a combinação de mais informações durante a tomada de decisão pode melhorar a classificação dos pixels referentes às áreas dos veículos em movimento.

Ao invés da utilização de um modelo da imagem de referência para detecção de objetos em movimento, a subtração entre imagens consecutivas pode ser realizada. Assim, o problema de atualização de imagens é resolvido, reduzindo o custo em termos de memória e recursos para o processamento das fases de inicialização e treinamento necessárias para obtenção do modelo da imagem de referência. No entanto, os resultados da detecção de objetos em movimento através da subtração entre imagens consecutivas podem apresentar um aumento ou uma redução no tamanho desses objetos, o que aumenta a classificação incorreta dos pixels.

3.1.1 Modelagem da imagem de referência

Existem vários métodos de detecção de objetos em movimento utilizando a modelagem da imagem de referência e pesquisas mais recentes apresentam um comparativo entre o desempenho das propostas mais promissoras em [118,120–123]. Esses métodos normalmente são classificados conforme a técnica utilizada na etapa de modelagem da imagem de referência.

A maioria das técnicas utilizadas para obter a imagem de referência modela estatisticamente o comportamento de cada um de seus pixels ao longo do tempo. Assim, busca-se estimar a provável imagem de referência. No trabalho apresentado em [124] as

técnicas de modelagem são classificadas nas categorias não recursiva e recursiva.

A categoria não recursiva utiliza uma abordagem de janela deslizante através do armazenamento de algumas imagens. A imagem de referência é estimada baseando-se na variação temporal de cada pixel armazenado. A manutenção do histórico de imagens utilizadas exige uma quantidade de recursos em termos de memória que pode limitar a implementação em FPGA de técnicas baseadas nessa categoria. Além disso, o tempo necessário para obter uma imagem de referência válida pode tornar o processo de inicialização do sistema inviável, já que os cálculos da estimativa devem percorrer todas as imagens armazenadas. Entre as técnicas não recursivas destacam-se a diferença entre os valores máximos e mínimos em imagens [11], filtro temporal de valor mediano [14], filtro linear preditivo [117] e os modelos não paramétricos [3].

Na categoria recursiva não há necessidade do armazenamento de imagens, já que o modelo é atualizado a cada nova imagem. Devido a essa característica, as técnicas recursivas tendem a serem mais eficientes [124]. A diferença entre imagens consecutivas [35], o filtro temporal de valor mediano estimado [125] e a mistura de gaussianas [19] são exemplos dessas técnicas.

Os métodos utilizados para detectar objetos em movimento também diferem na técnica de atualização, que pode ser conservadora ou liberal. Na técnica conservadora apenas as posições dos pixels consideradas estáticas são atualizadas, enquanto que na liberal, todas as posições são atualizadas. A principal desvantagem da abordagem liberal é a inclusão rápida de pixels referentes às áreas dos veículos em movimento na imagem de referência. Isso aumenta os erros de detecção para a situação na qual os veículos se movem lentamente. A técnica conservadora evita essa situação, entretanto apresenta como desvantagem a possibilidade da classificação incorreta dos pixels referentes às áreas dos veículos em movimento que inicialmente eram considerados como estáticos [1,44].

Na proposta apresentada em [126], a imagem de referência é modelada pela distribuição gaussiana buscando a rápida atualização e estabilidade do modelo. Esse algoritmo modela a imagem de referência com uma única média e variância para cada pixel. Contudo, a classificação incorreta dos pixels aumenta quando uma câmera externa filma uma área que apresenta árvores cobrindo parcialmente um prédio. O valor de um determinado pixel dessa imagem pode representar por vezes uma folha de árvore, o galho de árvore ou a parede do prédio que a árvore encobre. Esse pixel, apesar da variação, deve ser sempre considerado parte do modelo da imagem de referência e uma única distribuição gaussiana não é capaz de

modelá-lo. Para minimizar os problemas anteriores, em [19] é proposta a modelagem da imagem de referência utilizando uma mistura de funções de distribuição gaussiana. Assim, o algoritmo melhora a classificação dos pixels em relação à variação repentina de iluminação. Para reduzir a carga computacional dos sistemas de detecção de objetos em movimento, a modelagem da imagem de referência pode ser realizada através da operação de média das últimas imagens ou ainda o valor mediano. A principal desvantagem desse tipo de modelagem é a necessidade de armazenar os valores dos pixels das últimas imagens. Esse elevado custo em termos de memória pode inviabilizar a sua implementação em FPGA. Na abordagem proposta em [127–129], a imagem de referência é modelada através da estimativa do valor mediano dos pixels em várias imagens observadas. Essa solução não apresenta a necessidade de armazenar as imagens, tornando-a interessante em plataformas nas quais os recursos em termos de memória são reduzidos.

3.1.2 Classificação dos objetos em movimento

Para detecção de objetos em movimento utilizando a subtração pelo modelo da imagem de referência é construída uma única imagem durante um tempo *t*. Assim, a forma mais simples de classificar os pixels referentes aos objetos em movimento é aplicar um limiar à diferença entre a imagem atual $I(x,y,t)$ e a estimativa de uma imagem de referência $R(x,y)$. Dessa forma, um pixel é classificado como parte do objeto em movimento se atender à inequação (3.1).

$$\left|I(x,y,t)-R(x,y)\right| > Lim \tag{3.1}$$

O valor do limiar determinado empiricamente deve ser atualizado durante o processo de detecção [130]. As posições *x* e *y* são as coordenadas dos pixels na matriz da imagem.

Uma outra forma utilizada para detectar os objetos em movimento é apresentada em [124]. Nessa proposta é calculada a média (μ) e o desvio padrão (σ) sobre todas as posições da diferença $|I(x,y,t)-R(x,y)|$. Assim, um pixel é considerado parte do objeto em movimento se atender à inequação (3.2).

$$\frac{\left|I(x,y,t)-R(x,y)-\mu\right|}{\sigma} > Lim \tag{3.2}$$

Idealmente o valor do limiar deve ser obtido em função da localização espacial. Esse limiar deve ser menor para regiões da imagem nas quais o contraste é baixo. Na proposta apresentada em [131], a diferença relativa ao invés da diferença absoluta é implementada. Assim, busca-se enfatizar o contraste em áreas mais escuras, como as regiões de sombras. Um

pixel é considerado parte do objeto em movimento se a condição apresentada na inequação (3.3) é satisfeita.

$$\frac{\left|I(x,y,t)-R(x,y)\right|}{R(x,y)} > Lim \tag{3.3}$$

Em alguns trabalhos são utilizadas informações extraídas de um padrão de cor, como o *RGB* em [132] para melhorar a classificação dos pixels. No entanto, há a necessidade de aplicar um limiar diferente para cada componente do padrão de cor, conforme as inequações apresentadas em (3.4).

$$\begin{aligned} \left|I_R(x,y,t)-R_R(x,y)\right| &> Lim_R \\ \left|I_G(x,y,t)-R_G(x,y)\right| &> Lim_G \\ \left|I_B(x,y,t)-R_B(x,y)\right| &> Lim_B \end{aligned} \tag{3.4}$$

Caso a independência entre os limiares não seja válida, pode-se utilizar a distância Euclidiana [133]. Todavia, para o padrão de cor *HSV*, o qual apresenta as coordenadas em escalas diferentes, a distância de Mahalanobis se torna uma alternativa [134]. Contudo, para normalizar as diferentes escalas e variâncias entre os componentes dos vetores é calculada a matriz inversa, o que aumenta a carga computacional.

Ao detectar objetos em movimento em uma sequência de imagens é possível realizar a diferença temporal entre os pixels pertencentes a duas ou mais imagens consecutivas. Essa técnica tem a vantagem de utilizar a imagem de referência atualizada. No entanto, perde a consistência fornecida pela modelagem da imagem de referência. A operação realizada entre imagens consecutivas é apresentada na inequação (3.5).

$$\left|I(x,y,t)-I(x,y,t-1)\right| > Lim \tag{3.5}$$

Um dos grandes problemas da subtração entre imagens consecutivas, conforme apresentado na Seção 3.1.3 a seguir, é o aumento da classificação incorreta dos pixels quando a imagem anterior apresenta objetos em movimento. Assim, embora a técnica seja apropriada para ambientes dinâmicos, possui eficiência comprometida na classificação dos pixels. Consequentemente, a classificação incorreta de pixels na área correspondente aos objetos em movimento exige um processamento adicional como proposto em [35,55,94,135] através de operações morfológicas.

3.1.3 Problemas na detecção de veículos em movimento

Alguns problemas relacionados à detecção de veículos em movimento são apresentados através de uma rotina implementada no software MATLAB®. Assim, através desses problemas verifica-se a influência dos limiares durante o processo de binarização. No exemplo apresentado na Figura 3.1, uma imagem sem veículos em movimento é considerada como imagem de referência (*R*). Dessa forma, para cada nova imagem *I*(*t*) capturada pela câmera é realizada uma operação simples de subtração (*Sub*) com a imagem de referência. Como há a possibilidade de resultados com valores negativos, a imagem resultante é formada pelo valor absoluto (*Abs*) da diferença calculada em cada posição. Na Figura 3.1 são apresentados os resultados dessa abordagem implementada no software MATLAB®.

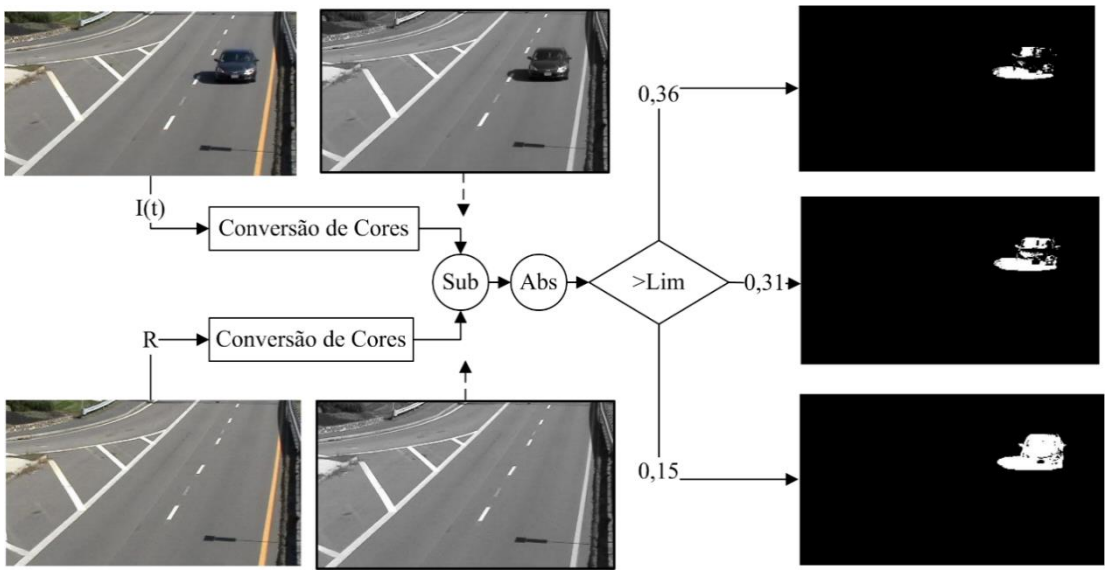


Figura 3.1 - Subtração pela imagem de referência sem veículos em movimento.

As imagens são incialmente convertidas do padrão de cor *RGB* para a escala de cinza, reduzindo a quantidade de recursos necessários. Posteriormente, essas imagens são subtraídas e o valor absoluto de cada posição é testado no processo de binarização. Observa-se na Figura 3.1 que as imagens binárias resultantes são altamente dependentes dos limiares (*Lim*) parametrizados: 0,36; 0,31 e 0,15. Para o limiar de 0,36 o veículo em movimento apresenta vários pixels classificados incorretamente como estáticos, ou seja, fazem parte da imagem de referência. Para o limiar de 0,31 mais pixels fazem parte do veículo em movimento, mas ainda há pixels classificados incorretamente. No limiar de 0,15 quase todos os pixels formando o veículo em movimento são classificados corretamente. No entanto, para todos os limiares testados o veículo em movimento apresenta sua área aumentada devido aos pixels referentes a sua sombra. Como o exemplo mostra um veículo em movimento com os valores dos pixels em escala de cinza próximos dos valores dos pixels de sua própria sombra, há a

necessidade de implementação de um processo de detecção que utilize mais informações extraídas de um padrão de cor.

Na abordagem apresentada na Figura 3.1, os veículos que entram em cena e ficam estacionados continuam a serem detectados como objetos em movimento, já que a imagem de referência é estática. Uma solução para atualizar dinamicamente a imagem de referência é proposta em [43] utilizando tanto a imagem atual como a imagem anterior. Dessa forma, a atualização da imagem de referência é dada em (3.6).

$$R(t) = \alpha I(t) + (1-\alpha) R(t-1) \tag{3.6}$$

Observa-se em (3.6) duas situações extremas. Para $\alpha = 0$, a imagem de referência será a primeira imagem ou um modelo da imagem. Para $\alpha = 1$, a imagem de referência será a imagem anterior. Dessa forma, para valores de α variando entre 0 e 1, é possível obter uma imagem de referência que se adapte às condições de iluminação ao longo do dia. Na Figura 3.2 é apresentada uma abordagem adaptativa baseada na proposta realizada em [43] para atualização da imagem referência.

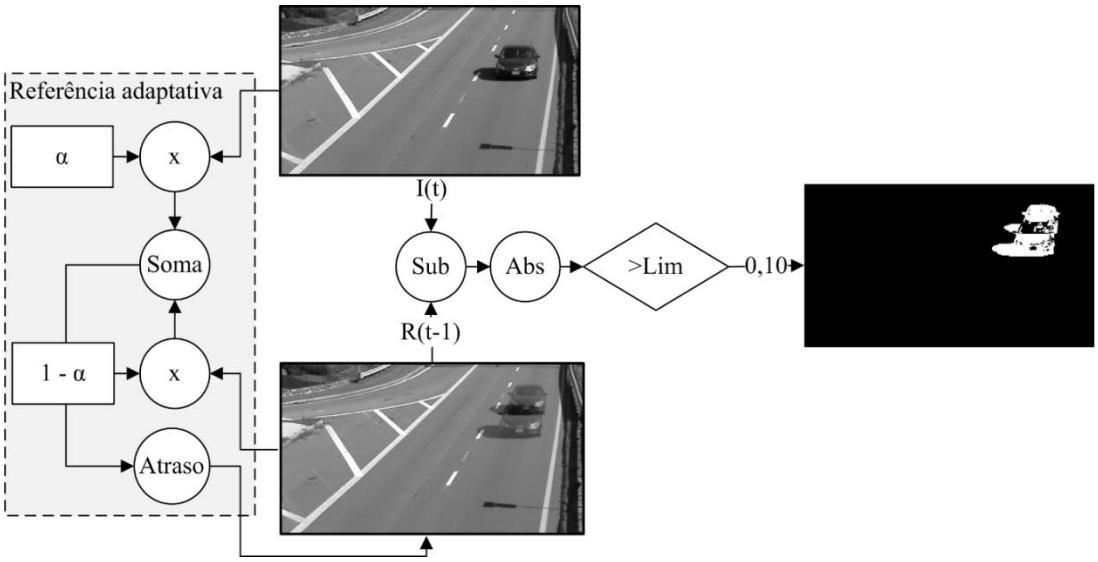


Figura 3.2 - Subtração pela imagem de referência atualizada.

Na Figura 3.2 verifica-se a possibilidade de atualizar a imagem de referência dinamicamente durante o processamento das imagens. Nessa situação é apresentada a subtração com uma atualização lenta, ou seja, correspondente à velocidade reduzida no tráfego de veículos. Assim, o veículo tem sua forma aumentada, podendo dobrar o seu tamanho. Consequentemente, a determinação da taxa de atualização α é fundamental para que a vantagem da adaptação da imagem de referência às condições de iluminação ao longo do tempo de captura melhore a classificação dos pixels.

Ao invés de atualizar o modelo da imagem de referência, outra solução possível é utilizar uma imagem anterior na sequência. Nessa abordagem, denominada subtração entre imagens consecutivas, é possível adaptar as mudanças da iluminação dinamicamente. Os veículos que param não são mais detectados. A diferenciação entre imagens consecutivas detecta bem objetos com os valores de cores uniformemente distribuídos e distantes da imagem anterior. No entanto, a dimensão dos veículos é alterada e dependendo da velocidade de deslocamento, o mesmo objeto pode surgir em duplicidade na imagem. Para resolver essa situação é proposta em [136,137] a subtração entre três imagens. Assim, busca-se através da operação de lógica clássica *E*, representada pelo símbolo ($\wedge$), a separação da imagem duplicada. A dificuldade está em relacionar a taxa de captura da câmera à imagem (*N*) a ser subtraída, já que o tamanho e a velocidade dos veículos são determinantes e podem variar. A operação realizada entre as imagens consecutivas é apresentada em (3.7).

$$D(N) = \left|I(t)-I(t-N)\right| \wedge \left|I(t)-I(t+N)\right| \tag{3.7}$$

Essa solução permite recuperar o tamanho correto do objeto e assim obter coordenadas precisas para o seu rastreamento em etapas posteriores. Uma dessas imagens é obtida através da subtração entre uma imagem atual e uma anterior ($I(t) - I(t - N)$) e a outra entre a imagem atual e uma posterior ($I(t) - I(t + N)$). Embora essa solução apresente inicialmente uma implementação simples, haverá a necessidade do armazenamento das informações pertinentes a essas imagens. Na Figura 3.3 é apresentada a solução implementada no software MATLAB®.

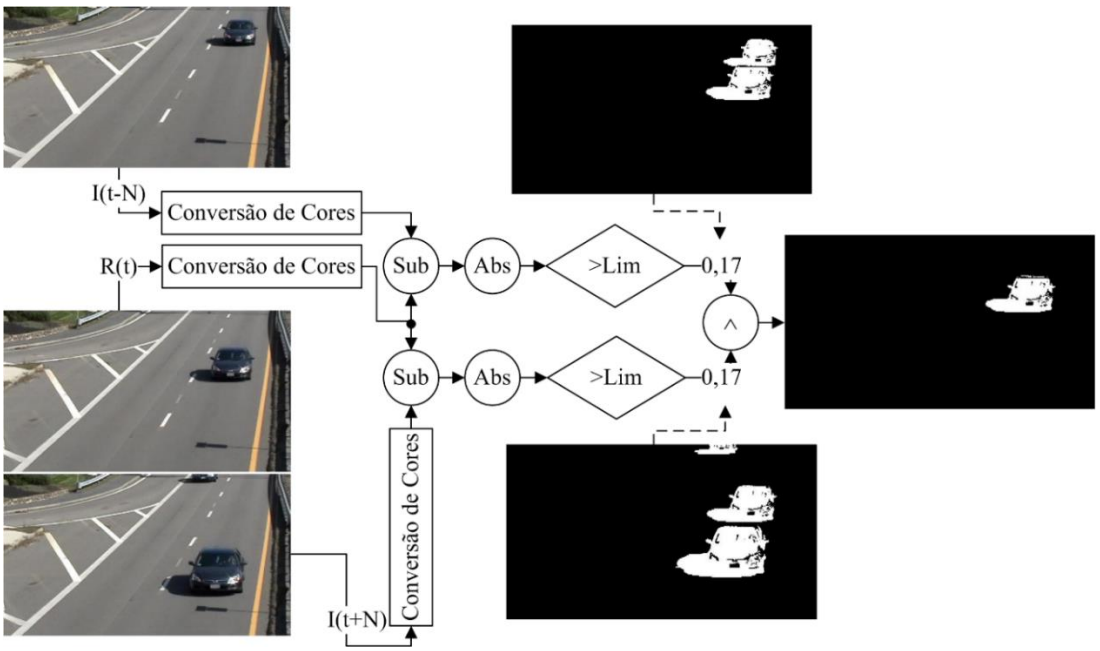


Figura 3.3 - Subtração entre imagens consecutivas.

Além dos problemas apresentados anteriormente, o limiar utilizado na operação de binarização da imagem também deve ser atualizado para se adaptar às condições de iluminação do ambiente. Para uma sequência de imagens na qual a taxa de captura é de 30 fps há um processamento de 1800 imagens por minuto. Assim, pode-se verificar experimentalmente que a forte correlação existente entre os pixels de várias imagens consecutivas capturadas por uma câmera estática permite o cálculo de um novo limiar em um intervalo de tempo maior sem perder a precisão dos resultados [138]. Outra questão relacionada ao limiar é a imagem a ser utilizada para sua obtenção, já que é possível determinar esse valor a partir de uma imagem anterior ou ainda da diferença entre imagens. O cálculo do limiar através da diferença entre imagens reduz informações desnecessárias decorrentes das áreas referentes a sua parte estática.

3.2 Métodos de detecção de objetos em movimento

Para tratar com os problemas apresentados, vários métodos de detecção de objetos em movimento foram estudados durante a revisão bibliográfica. Através da análise de vários desses métodos realizada em [118], verifica-se a utilização de métodos básicos, estatísticos, de agrupamentos e de estimativa que buscam modelar e atualizar o comportamento de cada pixel ao longo do tempo. Vários desses métodos são classificados na avaliação comparativa apresentada em [120]. Nesse trabalho, os métodos são classificados em paramétricos e não paramétricos, relacionando-os com as informações utilizadas na classificação dos pixels. Entre os métodos pesquisados, são apresentados a seguir um resumo sobre suas características principais, destacando as informações envolvidas durante o processamento da tomada de decisão e a necessidade de muitos recursos em termos de memória. Alguns conceitos apresentados nesses métodos são utilizados na implementação em FPGA proposta no quinto capítulo. Além disso, as características principais desses métodos são utilizadas na análise dos resultados em termos de classificação dos pixels.

3.2.1 Modelo de mistura de gaussianas

No método de detecção de objetos em movimento baseado na mistura de gaussianas (*Gaussian Mixture Model - GMM*) são combinadas funções gaussianas com parâmetros diferentes para modelar as variáveis aleatórias correspondentes às observações. Na proposta apresentada em [19,20], o método *GMM* é implementado para modelar a imagem de referência sem precisar armazenar um conjunto de dados de entrada durante o processo de

inicialização. A utilização do valor médio e da covariância permitem que a tomada de decisão em cada posição da imagem não seja limitada por um limiar global. Um pixel corresponde à imagem de referência se o seu valor está até 2,5 vezes o desvio padrão da distribuição. A modelagem multimodal da mistura de gaussianas permite tratar com árvores balançando com o vento e mudanças de iluminação gradual. No entanto, o *GMM* apresenta algumas desvantagens. O número de gaussianas, assim como a média, o desvio padrão e a taxa de aprendizagem devem ser pré-determinados. Além disso, nem sempre a distribuição temporal dos pixels segue uma distribuição gaussiana, o que ocasiona uma recuperação lenta dos erros de segmentação. Outro problema é a necessidade de uma sequência de imagens sem objetos em movimento para o treinamento de uma imagem de referência mais rápido e preciso [120].

A técnica para modelar a imagem de referência utilizando a mistura de gaussianas apresentada em [19,20] utiliza um número fixo de componentes, assim, é proposto em [6] um algoritmo capaz de adaptar automaticamente o número de componentes que modela cada pixel da sequência de imagens (*Adaptive Gaussian Mixture Model - AGMM*). Os parâmetros de média, desvio padrão e peso de cada componente são atualizados recursivamente. Nessa proposta a distância de Mahalanobis é utilizada para verificar se uma amostra pertence à componente da mistura gaussiana.

3.2.2 Estimativa de densidades através de um núcleo

O método de estimativa de densidades através de um núcleo (*Kernel Density Estimator – KDE*) utiliza uma técnica não paramétrica para estimar a função de probabilidade de uma variável aleatória na qual cada observação é ponderada pela distância em relação a um valor central, o núcleo. Assim, ao invés de utilizar uma função de probabilidade conhecida, essa técnica utiliza uma estimativa de funções de probabilidade baseada em um núcleo para modelar a distribuição temporal dos pixels. A proposta para detecção de objetos em movimento apresentada em [3] utiliza o método *KDE* para estimar em algumas amostras recentes os valores que modelam a distribuição dos pixels da imagem de referência. A ideia inicial é que para cada amostra é possível reconstruir a função de probabilidade que a gerou. Dessa forma, a subtração pelo modelo da imagem de referência é realizada entre a imagem atual e a reconstruída. Uma forma simples utilizada para estimar uma função de probabilidade é através da implementação de histogramas sobre a imagem. Sendo assim, a probabilidade com que cada pixel aparece na imagem é obtida através de sua frequência.

O método *KDE* proposto em [3] utiliza informações do padrão de cor *RGB* normalizado para separar regiões de sombras dos objetos em movimento. Além disso, modela

a imagem de referência multimodal com mudanças rápidas, incluindo diretamente no modelo os valores observados recentemente. No entanto, durante todo o processo de modelagem há a necessidade de permanecerem armazenadas várias imagens, o que requer muitos recursos em termos de memória [120].

3.2.3 *Codebook*

No trabalho apresentado em [5] é proposto um método de detecção de objetos em movimento através de uma imagem de referência obtida por uma técnica de agrupamento. Esse método é chamado pelos autores de *Codebook* e modela uma imagem de referência multimodal através de uma sequência longa de imagens. Assim, para cada pixel é construído, com base na sequência de treinamento, uma tabela (*codebook*) para armazenar uma ou várias informações (*codewords*). Os *codewords* consistem de um vetor *RGB* com valores dos pixels e de variáveis temporais, os quais formam o conjunto de informações representativas dos pixels da imagem de referência. O número de *codewords* depende da variação das informações da imagem de referência. Para criar ou atualizar os *codewords* durante o tempo de treinamento inicial é calculada a distorção de cor. Ao contrário do método *GMM*, o *Codebook* consegue processar objetos movendo durante o período de treinamento inicial. Contudo, se a cor dos pixels que formam os objetos em movimento é similar à cor dos pixels na imagem de referência, haverá um aumento na detecção incorreta desses pixels [120]. No método de detecção de objetos em movimento proposto em [22], a modelagem da imagem de referência é implementada através do agrupamento dos pixels em categorias, no entanto a tomada de decisão é realizada apenas com a informação dos valores dos pixels em escala de cinza.

3.2.4 *SACON* (*SAmple CONsensus*)

No método de detecção de objetos em movimento *SACON* proposto em [7] é calculado o consenso entre várias amostras da imagem de referência e estimado o seu modelo estatisticamente. Para isso, são armazenadas várias imagens de referência. Assim, quando há um consenso entre a amostra anterior e a atual, a amostra armazenada é marcada com nível lógico 1. Dessa forma, o pixel é classificado como parte de um objeto em movimento se não houver um consenso entre as amostras. Isso é possível através de um limiar determinado empiricamente. Essa proposta utiliza o padrão de cor *RGB* normalizado para reduzir a sensibilidade à variação da iluminação. Na abordagem proposta em [37], uma estratégia para estimar a imagem de referência muito similar ao método *SACON* é implementada. Nessa

abordagem é utilizada informação extraída através de um descritor de textura para tornar o sistema mais robusto à variação da iluminação.

Uma vantagem apresentada no método *SACON* é que não é necessário assumir uma função de probabilidade para os pixels. No entanto, o esquema de atualização da imagem de referência emprega a política de entrada e saída (*first-in-first-out*), assim, há falhas em não verificar as demais amostras da imagem de referência armazenadas [120]. Além disso, a necessidade de muitos recursos em termos de memória pode representar uma limitação na classificação correta dos pixels.

3.2.5 *Vibe* (*Visual Background Extractor*)

No método de detecção de objetos em movimento *Vibe* proposto em [9], a imagem de referência é obtida utilizando uma estratégia para escolher aleatoriamente entre os pixels contidos em uma determinada região ao redor de cada pixel central, os que formam a imagem de referência. Esse método necessita de capacidade de armazenamento para os valores de amostras correspondentes à imagem de referência. Um pixel é considerado parte da imagem de referência se a interseção entre a região ao redor do pixel central e a coleção de amostras da imagem de referência é maior ou igual a um limiar determinado empiricamente.

O método *Vibe* baseia-se na mesma consideração apresentada em [139], de que os pixels vizinhos possuem distribuição temporal similar. Assim, busca-se justificar a utilização da primeira imagem capturada para inicializar o modelo da imagem de referência. No entanto, os resultados desse método podem piorar muito se a primeira imagem apresentar objetos em movimento. Uma desvantagem desse método é a utilização apenas de informações dos componentes do padrão de cor *RGB* para modelar a imagem de referência, o que o torna sensível à variação da iluminação ao longo do tempo. Além disso, para a sequência de imagens capturadas em outros locais, torna-se necessário o ajuste manual dos parâmetros de forma que se adaptem às mudanças na imagem de referência [120].

A imagem de referência modelada pelo método *Vibe* consiste em um conjunto de valores de pixels observados. Essa é uma diferença importante quando comparado aos métodos *GMM* e *AGMM*, os quais utilizam uma função de probabilidade para modelar a imagem de referência. Assim, como a definição de uma função de probabilidade apropriada é uma tarefa difícil e nem sempre condiz com a distribuição temporal dos pixels, o método *Vibe* pode apresentar resultados superiores ao *GMM* e ao *AGMM*.

3.2.6 *PBAS* (*Pixel Based Adaptive Segmenter*)

No método de detecção de objetos em movimento *PBAS* proposto em [10] a imagem de referência é modelada combinando o *Vibe* com o *SACON* em um método adaptativo baseado na informação do pixel. O *PBAS* utiliza um histórico com o valor dos pixels de várias imagens de referência como modelo e uma técnica de atualização aleatória semelhante ao *Vibe*. No entanto, no método *PBAS* não são utilizados parâmetros fixos, o que permite, por exemplo, a atualização ao longo do tempo do valor do limiar responsável pela classificação do pixel como parte do objeto em movimento ou da imagem de referência [118,120].

As propostas dos métodos *Vibe* e *PBAS* utilizam a técnica conservativa para atualização da imagem de referência, assim, apenas as posições dos pixels considerados parte da imagem de referência são atualizadas. Essa técnica evita que objetos movendo lentamente sejam classificados como parte da imagem de referência, entretanto objetos em movimento classificados incorretamente como estáticos dificilmente voltam a serem classificados corretamente. Uma solução possível é a contagem do número de vezes que uma posição apresenta um pixel classificado como um objeto em movimento. Sendo assim, quando o valor do contador exceder um determinado limite, uma atualização é forçada [44].

3.2.7 *SAKBOT* (*Statistical And Knowledge-Based Object Tracker*)

Embora os métodos *ViBe* e *PBAS* apresentem bons resultados em várias sequências de imagens, a sua taxa de acertos é comprometida em ambientes nos quais há sombras dos objetos considerados em movimento. O método de detecção de objetos em movimento *SAKBOT* proposto em [4] apresenta a vantagem de tratar o problema das sombras de objetos em movimento. Essa abordagem é proposta com a finalidade de detectar quaisquer objetos em movimento através da supressão dos pixels considerados estáticos. O primeiro passo é converter a informação a partir do padrão de cor *RGB* para *HSV*, padrão mais próximo da percepção da visão humana [13]. Assim, considerando que a oclusão da luz na sombra projetada escurece o pixel na imagem de referência e satura a sua cor, os pixels são classificados. Dessa forma, as informações dos componentes do padrão de cor são processadas para diferenciar os pixels dos objetos em movimento de suas respectivas sombras. O modelo da imagem de referência é obtido através do valor mediano entre vários pixels, o que requer muitos recursos em termos de memória para o armazenamento de várias imagens. O método de detecção de objetos em movimento proposto em [8] também utiliza informações extraídas do padrão de cor *HSV* para modelar os pixels da imagem de referência.

Nesse método um vetor de peso para cada pixel é inicializado com os componentes do padrão de cor obtidos da primeira imagem da sequência. A técnica proposta para detectar os pixels referentes às sombras dos veículos em movimento é baseada na proposta apresentada em [13,14].

3.2.8 *W*^{4}

O método *W*^{4} apresentado no sistema proposto em [11] para detectar o movimento de pessoas baseia-se na modelagem da imagem de referência através de valores máximos e mínimos dos pixels em escala de cinza em uma sequência de imagens. Nesse sistema é utilizado um período de 30 segundos para modelar a imagem de referência utilizando o valor mediano dos pixels das imagens armazenadas. O modelo de cada pixel utiliza o valor mínimo, máximo e a diferença entre duas imagens consecutivas durante o período de treinamento. Embora a implementação desse método seja simples, requer considerável quantidade de recursos em termos de memória durante o tempo de treinamento proposto. Outra técnica simples para modelar a imagem de referência é utilizada no método de detectação de objetos em movimento proposto em [21]. Nessa técnica o valor do pixel mais frequente em uma sequência inicial de imagens é classificado como parte da imagem de referência. Esse processo é realizado através de histogramas para cada componente do padrão de cor *RGB*.

3.2.9 Medidas de distorção

No método de detecção de objetos em movimento apresentado inicialmente em [12], a imagem de referência é modelada utilizando medidas de distorção de cor e brilho em uma sequência de imagens. A extração de informações sobre a imagem de referência, objetos em movimento e de suas respectivas sombras são implementadas através de limiares determinados empiricamente no padrão de cor *RGB*.

A medida de distorção do brilho ($\alpha(x,y)$) é definida como o valor escalar que projeta a cor atual $I(x,y)$ na linha de cor que se aproxima o máximo possível do valor de referência. Uma característica importante é o padrão apresentado ao longo dessa linha de cor, alterando apenas a sua intensidade luminosa. Sendo assim, torna-se possível determinar se um pixel possui um valor de cor semelhante a outro, mas com o brilho diferente. A distorção de brilho significa apenas uma mudança na iluminação, permanecendo constante a cor do pixel. Consequentemente, a medida de distorção do brilho permite identificar os pixels na imagem que formam as áreas de sombras. Isso ocorre porque os pixels de áreas sombreadas possuem um valor de cor semelhante à imagem de referência, mas com um brilho bem menor. A

medida de distorção do brilho formulada em [12] é apresentada em (3.8).

$$\alpha_{(x,y)} = \frac{\frac{I_R(x,y)\mu_R(x,y)}{\sigma_R^2(x,y)} + \frac{I_G(x,y)\mu_G(x,y)}{\sigma_G^2(x,y)} + \frac{I_B(x,y)\mu_B(x,y)}{\sigma_B^2(x,y)}}{\left(\frac{\mu_R(x,y)}{\sigma_R(x,y)}\right)^2 + \left(\frac{\mu_G(x,y)}{\sigma_G(x,y)}\right)^2 + \left(\frac{\mu_B(x,y)}{\sigma_B(x,y)}\right)^2} \quad (3.8)$$

Para o brilho do pixel na imagem atual igual ao da imagem de referência a distorção será 1. Assim, tem-se para uma distorção menor do que 1, que o pixel na imagem atual é mais escuro e maior do que 1, mais claro.

A medida de distorção da cor ($CD(x,y)$) é definida como sendo a distância ortogonal entre o valor da cor atual e a linha de cor no modelo. Essa medida permite verificar se um pixel da imagem atual pertence a um objeto em movimento ou à imagem de referência. A medida de distorção de cor formulada em [12] é apresentada em (3.9).

$$CD_{(x,y)} = \sqrt{\left(\frac{I_R(x,y) - \alpha(x,y)\mu_R(x,y)}{\sigma_R(x,y)}\right)^2 + \left(\frac{I_G(x,y) - \alpha(x,y)\mu_G(x,y)}{\sigma_G(x,y)}\right)^2 + \left(\frac{I_B(x,y) - \alpha(x,y)\mu_B(x,y)}{\sigma_B(x,y)}\right)^2} \quad (3.9)$$

Os resultados das fórmulas (3.8) e (3.9) podem apresentar valores elevados para as medidas de distorção quando o desvio padrão for baixo. Para solucionar esse problema é necessário determinar um valor mínimo para cada desvio padrão. Na proposta apresentada em [12], esses valores mínimos não foram determinados. A implementação dessa proposta requer recursos em termos de memória proporcionais ao número e à resolução das imagens utilizadas na fase de treinamento para obter a imagem de referência.

A abordagem para detectar os pixels das áreas de sombras dos objetos em movimento proposta em [13,14] também utiliza informações extraídas de um padrão de cor. No entanto, ao invés de implementar as medidas de distorção no padrão de cor *RGB*, a tomada de decisão é desenvolvida através de informações obtidas do padrão de cor *HSV*. Esse padrão já fornece informações de tonalidade, saturação e brilho separadamente. Dessa forma, a implementação do sistema de tomada de decisão clássica proposto na Seção 5.3.1 pode reduzir a sua carga computacional.

3.2.10 Multicritério

No método de detecção de objetos em movimento proposto em [15–17], a tomada de decisão baseia-se na análise multicritério. Esse método implementa a tomada de decisão através da combinação de informações de cor e textura utilizando o descritor *LBP*. A implementação da combinação dessas informações extraídas do padrão de cor *YCbCr* através da integral fuzzy proposta por *Choquet* apresenta resultados melhores que o padrão de cor

HSV [16]. O processo de modelagem da imagem de referência implementado em [15] é realizado através da média aritmética entre várias imagens armazenadas no início da sequência. A proposta desenvolvida em [18] é muito similar a anterior, no entanto é implementado um descritor de textura *LBP* modificado para reduzir a sensibilidade do sistema ao movimento de rotação dos objetos. Os resultados desses métodos apresentam uma classificação melhor que o método *GMM* proposto inicialmente em [19,20]. Embora esses métodos considerem informações temporal, espacial e de cor, ainda há muitos pixels das áreas de sombras dos objetos em movimento classificados incorretamente. Assim, esses pixels acabam sendo classificados como parte dos objetos em movimento. Devido à extração de cada característica do padrão de cor ser independente, o processamento da tomada de decisão pode ser implementado em paralelo.

No sistema proposto em [75] é implementado após a tomada de decisão fuzzy para detecção dos pixels referentes às áreas dos veículos em movimento desenvolvida com base na proposta apresentada em [15], uma etapa de pós-processamento para remoção de áreas de sombras dos veículos em movimento. A remoção das áreas de sombras é baseada na proposta apresentada em [13,14].

3.3 Considerações finais

Um dos objetivos nesta tese é obter um sistema de tomada de decisão para detecção de veículos em movimento para FPGA sem utilizar muitos recursos em termos de memória. Esse objetivo é fácil de atender com a implementação de métodos mais simples, como os baseados na diferença entre imagens consecutivas. No entanto, os resultados em termos de classificação dos pixels são comprometidos pela existência de veículos em movimento na imagem anterior. Assim, para melhorar a classificação dos pixels torna-se necessária que a detecção de veículos em movimento seja realizada entre a imagem atual e uma imagem de referência sem veículos em movimento. A estimativa do valor mediano de cada posição da imagem permite uma modelagem simples da imagem de referência armazenando apenas uma imagem. Entretanto, a tomada de decisão implementada para classificar os pixels referentes aos veículos em movimento deve combinar mais informações extraídas das imagens para compensar os resultados da medida de similaridade realizada com uma imagem de referência modelada por essa técnica. Contudo, o método de detecção de objetos em movimento baseado na integral fuzzy permite a realização de um processo de tomada de decisão que combina diretamente várias informações, como a luminância, cor e textura extraídas paralelamente. Embora esse método apresente bons resultados, as informações combinadas através da integral fuzzy não

conseguem separar precisamente os pixels referentes às áreas dos veículos em movimento de suas respectivas sombras. Sendo assim, informações de tonalidade, saturação e brilho extraídas do padrão de cor *HSV* podem ser combinadas em um processamento paralelo através de uma tomada de decisão clássica para melhorar a classificação dos pixels do sistema, o que justifica a sua implementação no sistema de tomada de decisão para FPGA proposto na Seção 5.3.1.

4 Teoria fuzzy aplicada ao processamento de imagens

Neste capítulo são apresentadas a teoria de conjuntos, agrupamentos, sistemas e integral fuzzy capazes de tratar as incertezas inerentes à falta de informações extraídas de uma sequência de imagens. Os conceitos sobre sistemas e agrupamentos fuzzy são apresentados através do software MATLAB® e alguns aspectos sobre a implementação em FPGA são analisados. Um estudo sobre a tomada de decisão multicritério através da integral fuzzy é apresentado.

4.1 Conjuntos fuzzy

Inicialmente introduzida por *Zadeh* em [140], a teoria de conjuntos fuzzy busca tratar matematicamente conceitos vagos e subjetivos existentes na comunicação humana. Assim, termos linguísticos subjetivos como aproximadamente, maior que, entre outros, definem conceitos imprecisos que não podem ser tratados pelos conjuntos clássicos. Portanto, através do formalismo desenvolvido torna-se possível realizar a implementação de operações de cálculos com informações imprecisas em computadores e em hardware dedicado.

A teoria fuzzy aplicada no processamento de imagens é bastante extensa. É um assunto atual e de interesse em publicações que abrangem desde a etapa de transformação do padrão de cor, passando pela segmentação de objetos até a sua classificação.

Os conjuntos clássicos contêm elementos ou objetos que satisfazem propriedades precisas, sendo assim, os seus elementos são classificados em categorias bem definidas. O conjunto de números naturais entre 0 e 255 representando, por exemplo, a escala de cinza de uma imagem, é apresentado em (4.1).

$$A = \{x \in N \mid 0 \leq x \leq 255\} \tag{4.1}$$

A sua função de pertinência $\mu_A(x)$ é definida em (4.2).

$$\mu_A(x) = \begin{cases} 1, \text{ se } x \in A \\ 0, \text{ caso contrário} \end{cases} \tag{4.2}$$

A função de pertinência de um conjunto clássico define que todo número inteiro está ou não no conjunto *A*. Devido ao fato da função de pertinência mapear todos os números inteiros entre 0 e 255 em dois elementos do conjunto $\{0,1\}$, as operações com os conjuntos clássicos correspondem às operações com lógica booleana. O resultado dessas operações lógicas 0 ou 1 pode representar, por exemplo, preto ou branco, falso ou verdadeiro, negativo

ou positivo. Em situações nas quais a pertinência entre os elementos e os conjuntos não é precisa, pode-se dizer que um elemento pertença em menor ou maior grau a um conjunto. Assim, a operação através de conjuntos fuzzy permite tratar com o conceito de verdade parcial, ou seja, entre completamente verdadeiro e completamente falso. Essa verdade parcial, denominada lógica fuzzy, reflete a incerteza inerente ao problema a ser resolvido [1].

4.1.1 Definições para o conjunto fuzzy

Com base nas referências [1,69,140–142], as principais definições aplicadas aos conjuntos fuzzy são apresentadas a seguir.

Conjunto fuzzy. Seja *X* o conjunto universo e *A* um subconjunto fuzzy associado com a função de pertinência $\mu_A(x): X \rightarrow [0,1]$ em que os elementos podem pertencer parcialmente ao conjunto *A*. Um conjunto fuzzy *A* é um subconjunto de *X*, definido como um conjunto de pares ordenados, como representado em (4.3).

$$A = \left\{\left(x, \mu_A(x)\right) \mid x \in X\right\} \tag{4.3}$$

A cada elemento de *X* é atribuído um certo grau de pertinência que varia entre 0 e 1. O conjunto universo *X* pode ser composto por elementos discretos ou ser um espaço contínuo. Como consequência, o subconjunto fuzzy *A* também pode ser discreto ou contínuo [69].

Conjunto fuzzy normalizado. Um conjunto fuzzy *A* é normalizado se existe pelo menos um elemento *x* pertencente a *X* tal que $\mu_A(x) = 1$.

Altura. A altura (*h*) de um conjunto fuzzy *A* é o seu maior grau de pertinência. A altura é representada em (4.4), na qual s*up* representa a abreviatura de *supremum* [140,141].

$$h(A) = \sup_{x \in X} \mu_A(x) \tag{4.4}$$

Suporte. O Suporte (*Supp*) de *A* é o subconjunto de *X* no qual os elementos têm grau de pertinência não-nulo em relação ao subconjunto *A*, sendo representado em (4.5).

$$Supp(A) = \left\{x \in X : \mu_A(x) > 0\right\} \tag{4.5}$$

Cardinalidade. A cardinalidade (|*A*|) de um conjunto fuzzy *A* é a soma dos graus de pertinência de todos os elementos do conjunto *A* que pertencem a um conjunto universo *X*, representada em (4.6).

$$|A| = \sum_{x \in X} \mu_A(x) \tag{4.6}$$

Corte-α. O corte-α é um conjunto fuzzy *A* especificado por um conjunto ordinário contendo todos os elementos de *A* que possuem grau de pertinência maior ou igual a α. O

corte-α em um conjunto fuzzy *A* é representado em (4.7).

$$A_\alpha = \left\{ x \in X : \mu_A(x) \geq \alpha \right\} \tag{4.7}$$

Para a situação em que α = 1, A_α corresponde aos elementos de *X* que pertencem a *A*, com o conceito de pertinência clássica.

O teorema da representação afirma que qualquer conjunto fuzzy *A* pode ser decomposto em uma série de cortes-α. Esse teorema é representado em (4.8).

$$A = \bigcup_{\alpha \in [0,1]} \left(\alpha A_\alpha \right) \tag{4.8}$$

4.1.2 Operações com conjuntos fuzzy

Nos sistemas que implementam a lógica fuzzy, o processamento de informações fuzzy consiste normalmente de operações que são realizadas entre seus conjuntos fuzzy. As operações básicas de união, intersecção e complemento em conjuntos fuzzy são normalmente definidas em função de operadores de máximo e mínimo. Esses operadores são análogos aos operadores produto e soma das operações aritméticas [1]. As definições seguintes são baseadas nesses operadores.

Conjunto união. O conjunto união entre dois conjuntos fuzzy *A* e *B*, pertencentes a um mesmo conjunto universo *X*, dado por $A \cup B$, é representado em (4.9).

$$\begin{aligned} A \cup B &= \max\left[\mu_A(x), \mu_B(x) \right] \text{ ou} \\ A \cup B &= \mu_A(x) \vee \mu_B(x) \end{aligned} \tag{4.9}$$

Conjunto intersecção. O conjunto intersecção de *A* e *B*, dado por $A \cap B$, é representado em (4.10).

$$\begin{aligned} A \cap B &= \min\left[\mu_A(x), \mu_B(x) \right] \text{ ou} \\ A \cap B &= \mu_A(x) \wedge \mu_B(x) \end{aligned} \tag{4.10}$$

Conjunto complemento. O conjunto complemento de um conjunto fuzzy *A* normalizado é formado pela subtração de $\mu_A(x)$ do valor unitário. O complemento do conjunto *A* é representado em (4.11).

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{4.11}$$

As operações de união e intersecção através de conjuntos fuzzy também são encontradas na literatura com os termos utilizados em operações de lógica clássica *Ou* e *E*, respectivamente.

4.1.3 Números fuzzy

Os números fuzzy representam um caso especial de conjunto fuzzy que definem um intervalo fuzzy nos números reais. Para um número real cujo valor preciso não é conhecido com exatidão, esse número é definido através de um intervalo fuzzy. Um intervalo fuzzy é geralmente representado por dois pontos extremos, sendo um valor mínimo (a_1), um valor máximo (a_3) e um valor médio (a_2). Os números fuzzy mais comuns são os triangulares e os trapezoidais, já que os graus de pertinência formam funções que requerem cálculos mais simples.

Para um conjunto fuzzy *A*, um corte-α em *A* pode ser representado por um par de valores que determinam o seu domínio. Observa-se na Figura 4.1 que um corte-α em *A* produz o conjunto A_α. Além dos intervalos utilizando os pontos extremos, é possível estabelecer qualquer outro intervalo dentro de um número fuzzy associado a um corte-α, conforme apresentado na Figura 4.1.

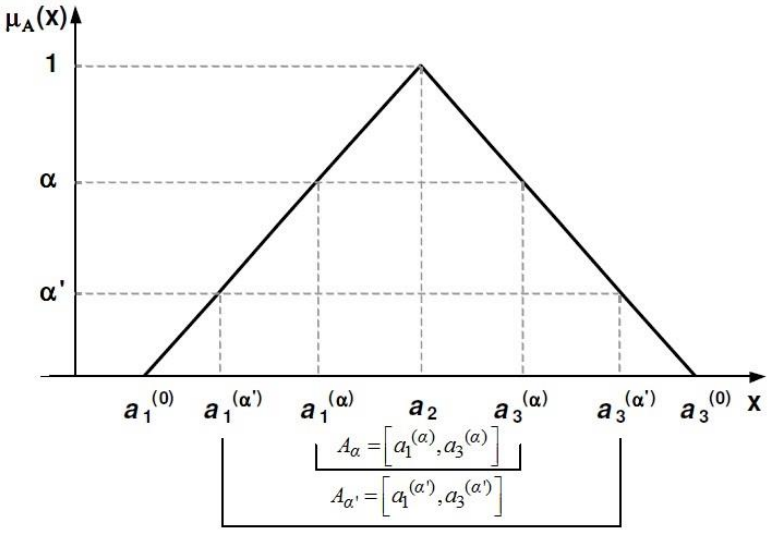


Figura 4.1 - Corte-α de um número fuzzy.

Na Figura 4.2 são apresentadas as funções de pertinência mais utilizadas considerando os valores dos pixels entre 0 e 255. É possível notar que cada função tem uma inclinação diferente, o que a torna mais adequada em algumas aplicações. As funções foram geradas no software MATLAB® e cada uma apresenta seu nome e os parâmetros utilizados no seu respectivo eixo das abscissas.

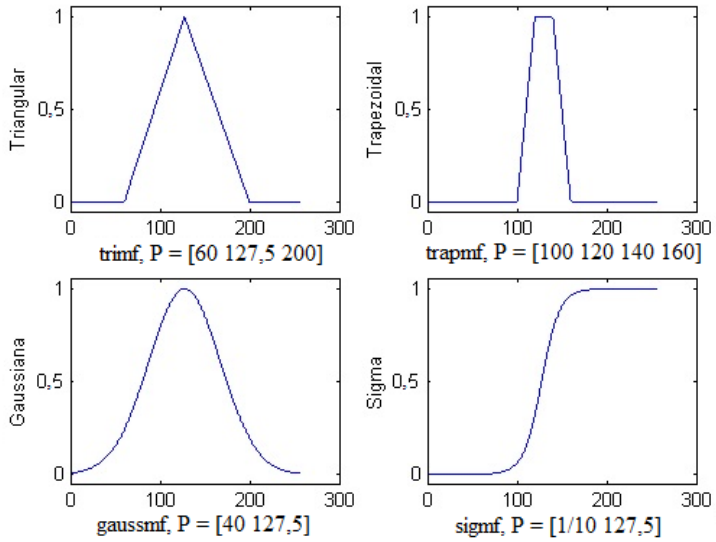


Figura 4.2 - Funções de pertinência triangular, trapezoidal, gaussiana e sigma.

As funções de pertinência triangular e trapezoidal apresentam transição linear. No entanto, a função de pertinência trapezoidal possui uma região com grau de pertinência constante para os valores dos pixels de 120 a 140. Já as funções de pertinência gaussiana e sigma apresentam uma transição mais suave quando comparadas com as funções de pertinência triangular e trapezoidal. De acordo com seus parâmetros, as funções de pertinência triangular e trapezoidal podem apresentar simetria. Um exemplo de números fuzzy triangulares simétricos é apresentado em [71] para calcular estatísticas fuzzy em imagens digitais. Nesse trabalho os números fuzzy triangulares simétricos representam os valores dos pixels em escala de cinza ao redor de um pixel central, o qual apresenta o maior grau de pertinência.

4.2 A imprecisão das informações extraídas das imagens

O conceito da informação está intimamente ligado ao da incerteza. A deficiência da informação pode resultar em incerteza na solução de um determinado problema. No nível empírico a incerteza está presente em qualquer medição, sendo resultante de uma combinação de erros de medição, da resolução dos instrumentos de medição e de ruídos. No nível cognitivo, a incerteza aparece na ambiguidade inerente da linguagem natural. A informação é perfeita quando é precisa e certa, sendo imperfeita quando é inconsistente, imprecisa e incerta. A imprecisão e a inconsistência são propriedades relacionadas ao conteúdo da informação. Já a incerteza é uma propriedade que resulta da falta de informação sobre o ambiente para decidir se a afirmação é verdadeira ou falsa [143]. Assim, a melhora na classificação correta dos pixels do sistema de tomada de decisão proposto requer além da medida de similaridade

envolvendo os valores dos pixels em escala de cinza, informações de cor e espacial.

Na Figura 4.3 são apresentadas algumas situações nas quais a lógica fuzzy pode ser implementada nas etapas do processamento de imagens para tratar as incertezas ao invés da teoria clássica.

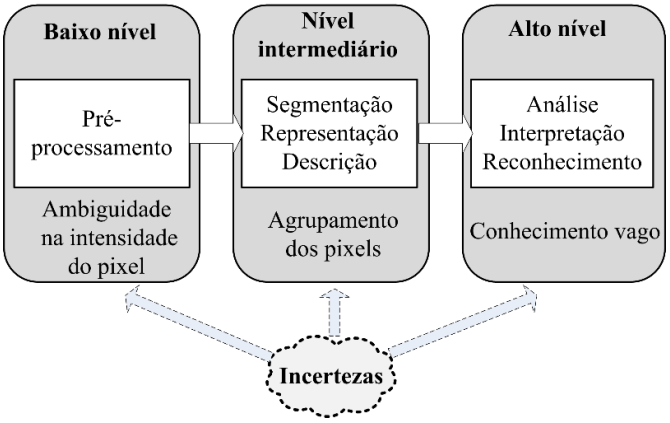


Figura 4.3 - Implementação de lógica fuzzy em processamento de imagens [69].

Verifica-se no processamento de baixo nível a ambiguidade na informação extraída dos valores dos pixels. Assim, por exemplo, na melhoria do contraste fica a incerteza sobre qual pixel deve ser transformado para mais escuro ou mais claro. No nível intermediário, a incerteza é relacionada, por exemplo, ao limite sobre as regiões entre os pixels classificados como veículos em movimento e estáticos. No processamento de alto nível a incerteza pode estar relacionada, por exemplo, em como identificar e rastrear separadamente veículos de passeio e de carga em movimento em uma mesma sequência de imagens.

Algumas técnicas fuzzy podem ser utilizadas na classificação de informações. Entre elas, destacam-se as abordagens sintáticas, como regras *if-then* e gramática fuzzy e as abordagens numéricas, como a integral fuzzy e os algoritmos de agrupamentos fuzzy. Assim, considerando os valores da função de pertinência, a classificação pode ser um tipo de decisão baseada, por exemplo, na distância para os centros de agrupamentos ou ainda em medidas de similaridade combinadas através da integral fuzzy [69].

4.3 Sistemas fuzzy

Os sistemas fuzzy são sistemas baseados em regras que utilizam a lógica fuzzy para tomar decisões. No processamento de imagem essas decisões são realizadas sobre as informações extraídas das imagens. Assim, de forma geral, um sistema fuzzy faz corresponder uma saída fuzzy para cada entrada fuzzy. No entanto, considerando o processamento de imagens com cada pixel correspondendo aos valores de 0 a 255, um número real deve corresponder a um valor nessa faixa na saída. Como os números fuzzy correspondem aos

valores entre 0 e 1, torna-se necessária a implementação de módulos de conversão. De forma resumida, um módulo de *fuzzificação* implementado na entrada deve ser capaz de transformar as variáveis de entrada em variáveis do conjunto fuzzy através das funções de pertinência. Um módulo contendo a base de regras constitui o núcleo do sistema. Os conectivos lógicos para estabelecer a relação fuzzy que modela a base de regras são definidos no módulo de inferência. Um módulo de *defuzzificação* é utilizado para converter a variável de saída para um valor numérico entre 0 e 255 [144].

Para realizar as operações com os conjuntos fuzzy existem os métodos de inferência fuzzy. Há diferentes métodos de inferência fuzzy com propriedades distintas. Entre esses métodos, os propostos por *Mamdani* [145,146] e por *Sugeno* [147,148] são os mais comuns. Para exemplificar o processamento utilizando o sistema fuzzy é implementado para inferência o método de *Mamdani*, já que além de ser considerado simples e eficiente, é bastante condizente com a intuição humana. Na Figura 4.4 é apresentado um sistema de inferência fuzzy baseado em *Mamdani* e implementado no software MATLAB® para converter o padrão de cor *RGB* 24 bits em uma imagem monocromática. Para as funções de pertinência da entrada do sistema proposto são utilizadas funções gaussianas representando os conjuntos com os valores baixo, médio e alto dos pixels de cada um dos três componentes do padrão de cor *RGB*. A função de pertinência na saída é composta por três funções trapezoidais representando também os conjuntos com os valores baixo, médio e alto dos pixels. A operação de lógica fuzzy *E* ($\wedge$) é implementada para relacionar as variáveis em cada regra. A função de lógica fuzzy *Ou* ($\vee$) é utilizada para agregar os resultados, e assim, obter através da medida do centro geométrico ou centroide, o pixel correspondente à imagem monocromática resultante.

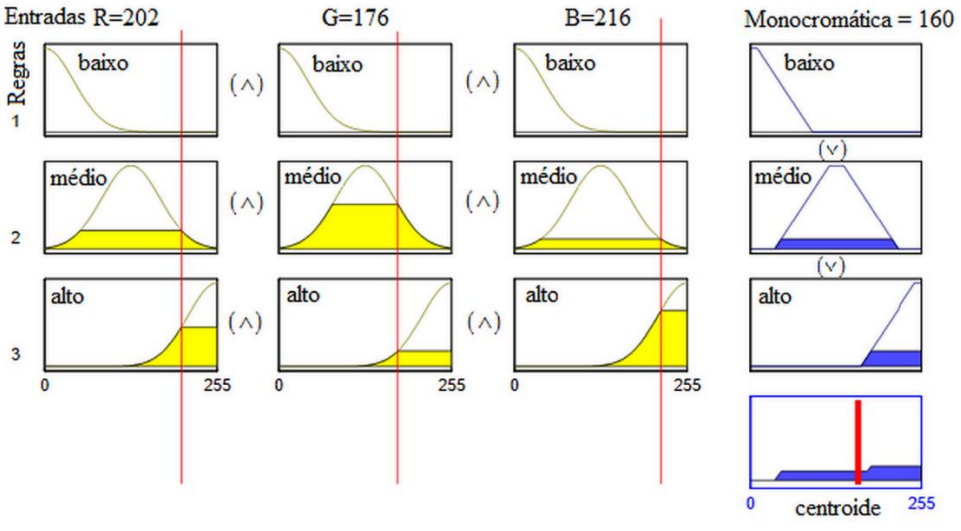


Figura 4.4 - Sistema de inferência fuzzy baseado em *Mamdani*.

O levantamento das funções de pertinência depende de cada aplicação, sendo que nesse caso, a máxima pertinência será obtida para o conjunto baixo com o valor em 0 e para o alto em 255. O maior grau de pertinência para o conjunto médio é dado para parâmetro de média da curva gaussiana utilizada, que é 127,5.

4.4 Integral fuzzy

Uma das áreas na qual se tem abordado muito a teoria fuzzy é na tomada de decisão multicritério [149–153]. Nesse tipo de decisão, várias informações são combinadas através do grau de importância de cada conjunto de medidas fuzzy. Cada informação extraída de uma imagem é considerada um critério durante o processo de decisão. Como a própria decisão em determinar se um pixel pertence ou não a um objeto pode tornar-se incerta devido à falta de informação, a combinação desses critérios através do cálculo da integral fuzzy pode melhorar a classificação dos pixels. A integral fuzzy é utilizada na decisão multicritério de diferentes formas. A combinação dos resultados de diferentes algoritmos de segmentação ou a segmentação através da combinação de critérios diferentes como cor, brilho e textura são alguns dos exemplos.

4.4.1 A integral fuzzy proposta por *Sugeno*

Há várias definições para integral fuzzy, entre elas as que mais se destacam são as propostas inicialmente por *Sugeno* em [154] e por *Choquet* em [155].

A integral fuzzy proposta por *Sugeno* (S_g) de uma função $h: X \rightarrow [0,1]$ em relação à medida fuzzy $g(A_i)$ é definida em (4.12) [69,149]. Sendo $g(A_i)$ uma medida fuzzy no conjunto finito X de n elementos $x_1, x_2, x_3,...x_n$ e $h(x_i)$ um subconjunto de X. A função $h(x_i)$ deve ser decrescente em X tal que $h(x_1) \geq h(x_2) \geq h(x_3) \geq ... \geq h(x_n)$.

$$S_g = \bigvee_{i=1}^{n} \left[h\left(x_i\right) \wedge g\left(A_i\right) \right] \tag{4.12}$$

O subconjunto $A_i = \{x_1,x_2,x_3,...x_i\}$ representa as combinações de elementos possíveis. Os símbolos $\vee$ e $\wedge$ representam, nesse caso, os operadores de máximo e mínimo, respectivamente. Essa formulação da integral fuzzy reduz o custo computacional de 2^n para n cálculos, no entanto a função $h(x_i)$ deve ser ordenada antes [69].

A interpretação da integral fuzzy em relação à combinação de informações para tomada de decisão multicritério considera $g(A_i)$ como sendo o grau de importância de cada conjunto de critérios adotados e $h(x_i)$ como sendo uma função fuzzy obtida para cada critério.

O cálculo da integral fuzzy apresentado em (4.12) pode ser considerado como uma

combinação pessimista da evidência objetiva ($h(x_i)$) e importância subjetiva da fonte de informação ($g(A_i)$). Sendo assim, pode-se desenvolver uma combinação mais otimista através da inversão da ordem dos operadores de máximo e mínimo [69].

4.4.2 A integral fuzzy proposta por *Choquet*

A integral de *Sugeno* é mais adequada para combinar critérios em aplicações nas quais a ordem dos elementos é importante. Uma integral adequada à combinação cardinal, ou seja, quando a distância entre os elementos tem significado, é proposta por *Choquet*. A integral de *Choquet* é definida em (4.13) [150]. Embora os trabalhos propostos por *Sugeno* e *Choquet* apresentem notações diferentes em suas definições, adota-se nesta tese uma notação comum e $h(x_o) = 0$. A função fuzzy $h(x_i)$ deve ser colocada em ordem crescente e os conjuntos de medidas fuzzy utilizados de acordo com sua posição.

$$C_g = \sum_{i=1}^{n} \left(h\left(x_i\right) - h\left(x_{i-1}\right) \right) g\left(A_i\right) \tag{4.13}$$

4.4.3 Função e medida para integral fuzzy

A função fuzzy $h(x_i)$ pode ser considerada uma medida de similaridade entre os critérios obtidos através de uma imagem de referência e da imagem atual conforme apresentada na Seção 2.4. A definição da função fuzzy obtida em [103] é apresentada em (4.14).

$$h\left(x_i\right) = \frac{\min\left(x_i^a, x_i^r\right)}{\max\left(x_i^a, x_i^r\right)} \tag{4.14}$$

Os resultados obtidos da função apresentada em (4.14) representam uma função que varia entre 0 e 1. Isso ocorre porque o valor correspondente ao numerador é sempre menor que o denominador para o mesmo critério, independente de pertencer à imagem de referência (x_i^r) ou à imagem atual (x_i^a).

Para calcular a medida fuzzy $g(A_i)$ da união de quaisquer conjuntos disjuntos dos quais as medidas fuzzy são dadas, pode-se utilizar uma versão operacional proposta por *Sugeno* [154]. Essa formulação é apresentada em (4.15).

$$g\left(A_i\right) = \frac{\left[\prod_{i=1}^{n} \left(1 + \lambda g\left(x_i\right)\right) - 1 \right]}{\lambda}, \ \lambda \neq 0 \tag{4.15}$$

Para a situação na qual $g(A_i) = 1$, tem-se o cálculo do parâmetro λ dado em (4.16)

[103,149]. Esse parâmetro da medida fuzzy representa a interação entre os critérios que são combinados [15,75].

$$\lambda + 1 = \prod_{i=1}^{n} \left(1 + \lambda g\left(x_i\right)\right) \tag{4.16}$$

A medida fuzzy para a união de dois conjuntos pode ser calculada em (4.17), na qual $g(A)$ e $g(B)$ representam os pesos da importância de cada subconjunto de elementos A e B, respectivamente. Sendo A, $B \subset X$ e $A \cap B = \emptyset$ [149].

$$g(A \cup B) = g(A) + g(B) + \lambda g(A) g(B) \tag{4.17}$$

Uma propriedade interessante da integral de *Choquet* é que se $g(A_i)$ é uma medida de probabilidade, a integral fuzzy é equivalente a integral de *Lebesgue* clássica [151,156]. Sendo assim, calcula-se a expectativa de $h(x_i)$ no que diz respeito a $g(A_i)$ em uma abordagem comum de probabilidade. A integral fuzzy é uma forma do operador de média no sentido de que o seu resultado está entre os valores mínimo e máximo da função $h(x_i)$ a ser integrada. Assim, uma série de operadores de combinação utilizados são casos especiais da integral fuzzy, como os operadores de mínimo e máximo, a soma ponderada e a média ponderada ordenada. Aplicando uma medida fuzzy apropriada, os pesos representam não apenas a importância das fontes de informações individuais, mas também as interações entre qualquer subconjunto das fontes [157].

Para uma medida fuzzy aditiva, ou seja, $g(A \cup B) = g(A) + g(B)$, quando $A \cap B = \emptyset$, não é considerada a interação entre os critérios. Nessa situação, a integral de *Choquet* coincide com a média aritmética ponderada ordenada (*Ordered Weighted Averaging - OWA*), representando um caso particular [158,159]. Um aspecto fundamental do operador *OWA* é a reordenação dos critérios, sendo os pesos associados com a posição. Como são operadores mais simples, sua aplicação é amplamente realizada em problemas de decisão. Algumas generalizações mais conhecidas do operador *OWA* são analisadas no trabalho apresentado em [159].

4.5 Algoritmo de agrupamento *k-means* em lógica fuzzy

A análise de agrupamento é uma ferramenta que pode ser útil para a classificação dos pixels referentes às áreas dos veículos em movimento em uma sequência de imagens. Essa técnica pode ser empregada para reduzir a dimensão de um conjunto de dados, reduzindo uma gama ampla de objetos à informação do centro de seu grupo. O método de agrupamento é uma técnica de aprendizagem não supervisionada, ou seja, não requer um modelo para o seu

treinamento. Assim, torna-se necessário recorrer a medidas obtidas com as informações das imagens para realizar os agrupamentos lógicos. Dessa forma, com os padrões definidos, é possível que o processo de formação dos agrupamentos consiga inferir a que grupo pertence o pixel de acordo com alguma informação [160], como o valor do pixel em escala de cinza. A medida a ser implementada depende das informações disponíveis e da aplicação [142].

O algoritmo *k-means* é uma heurística de agrupamento não hierárquico que busca minimizar a distância dos elementos a um conjunto de *k* centros dado por $X = \{x_1,x_2,x_3,...x_k\}$ de forma iterativa. A distância entre um ponto P_i e um conjunto de grupos é dada por *d(P,X)*. Essa é a distância de um ponto ao centro mais próximo dele [161]. A função a ser minimizada é apresentada em (4.18).

$$d\left(P,X\right) = \frac{1}{n}\sum_{i=1}^{n} d\left(P_i,X\right)^2 \tag{4.18}$$

De forma simplificada, o algoritmo *k-means* realiza a escolha de k elementos aleatoriamente ou através de alguma regra estabelecida como sendo a base de cada grupo, denominados centroides. O centroide é dado pela média de todos os *n* elementos dentro de um agrupamento, sendo os demais objetos associados ao centroide mais próximo. A cada passo os centroides são recalculados dentre os objetos de seu próprio grupo e os objetos são realocados para o centroide mais próximo. Esse processo é realizado repetidas vezes até que o nível de convergência seja satisfatório. Para isso, algum critério de parada deve ser estabelecido. O algoritmo depende do parâmetro *k*, o qual define o número de grupos. A definição desse parâmetro pode comprometer os resultados, já que em muitas situações não se conhece o número de agrupamentos necessários. Consequentemente, a utilização de um número pequeno de grupos pode causar a junção de pixels que deveriam ser classificados separadamente. Assim, embora o processamento desse algoritmo seja veloz, convergindo em poucas iterações, a escolha aleatória dos centros durante a inicialização pode comprometer os resultados do processo de segmentação [162].

O algoritmo *FCM* (*Fuzzy C-Means*) é introduzido em [163]. É um dos algoritmos mais simples, bem documentado e utilizado na literatura para o processo de agrupamento [163,164]. Esse algoritmo consiste na versão fuzzy do algoritmo de agrupamento *k-means* [165]. Como o algoritmo *FCM* realiza o agrupamento através do conceito fuzzy, cada ponto no conjunto de dados mais próximo ao centro de um grupo terá um grau de pertinência maior e ao mais distante, um grau menor [162]. Consequentemente, todos os pontos pertencem a todos os grupos em maior ou menor grau. Um estudo apresentado em [166] avalia algumas medidas de distância implementadas para definir a formação dos agrupamentos. Nesse

trabalho, a distância Euclidiana apresenta resultados melhores que a de Manhattan.

O exemplo a seguir apresenta uma implementação do algoritmo *FCM* no software MATLAB® para segmentar objetos em uma imagem. O agrupamento é realizado de acordo com a medida de distância Euclidiana entre os pixels e uma posição mais central possível em três grupos. A cada iteração, o centroide é atualizado até que a diferença para o anteriormente calculado seja menor ou igual a 10^{-6}. Dessa forma, atingido o critério de parada, considera-se que houve a convergência do algoritmo. No final do processamento é estabelecido o valor do pixel central para cada agrupamento com intensidade variando de 0 a 255. Na Figura 4.5 são apresentadas as funções de pertinência estabelecidas na entrada do processo de agrupamento fuzzy.

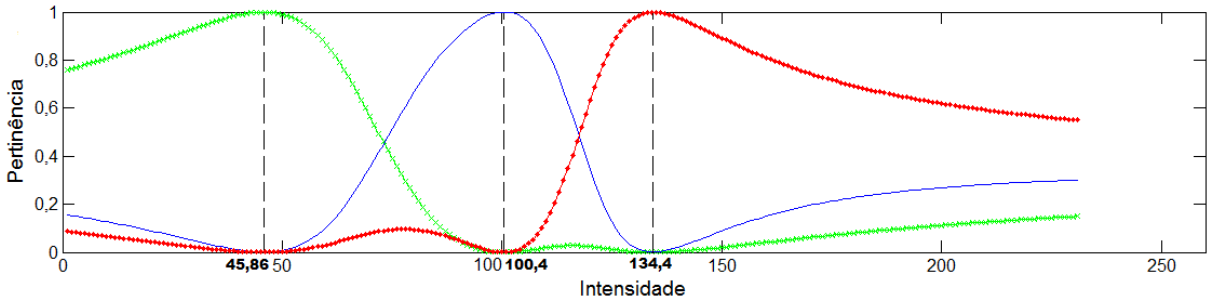


Figura 4.5 - Funções de pertinência.

Nas funções apresentadas na Figura 4.5 o grau de pertinência maior é referente aos valores 45,86; 100,4 e 134,4 representados em negrito no eixo das abscissas.

O algoritmo implementado em software através do MATLAB® apresenta como vantagem a utilização do histograma da imagem para atualizar os valores dos centroides. Assim, a segmentação dos objetos da imagem em três grupos de pixels é realizada utilizando uma implementação mais eficiente em termos de memória e tempo de processamento. Isso ocorre porque após um atraso inicial para a realização dos cálculos do histograma, o tempo do processamento para verificar a que grupo pertence o pixel é baseado na quantidade de bits que representa a escala de cinza e não na quantidade de pixels da imagem.

Considerando a atribuição em três grupos, observa-se na Figura 4.6 as regras estabelecidas para mapear os pixels em três imagens de saída, denominado processo de *defuzzificação*. De acordo com a Figura 4.6 os agrupamentos são definidos nos intervalos apresentados em (4.19).

$$\begin{aligned} 0 \leq \text{intensidade} \leq 73 &\Rightarrow \text{Agrupamento 1} \\ 73 < \text{intensidade} \leq 117 &\Rightarrow \text{Agrupamento 2} \\ 117 < \text{intensidade} \leq 230 &\Rightarrow \text{Agrupamento 3} \end{aligned} \tag{4.19}$$

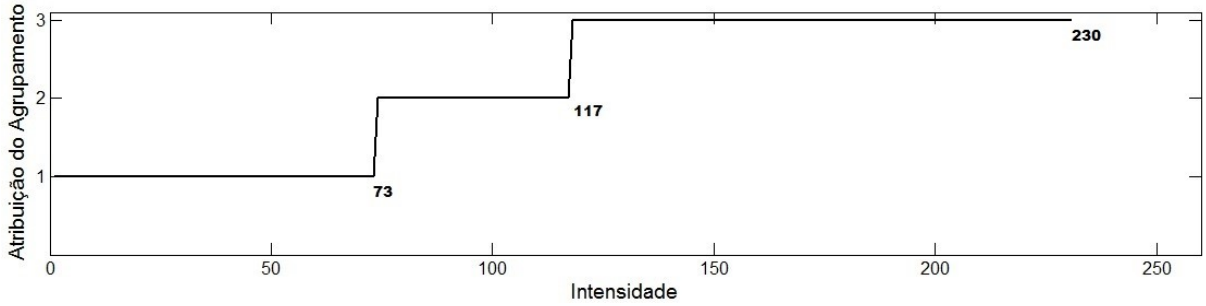


Figura 4.6 - Atribuição dos agrupamentos.

De posse do mapa de pertinência de cada agrupamento é apresentada na Figura 4.7 a imagem resultante em cada grupo. Devido à similaridade dos valores dos pixels em escala de cinza das áreas dos veículos em movimento e de suas respectivas sombras, não há como separá-las.

Figura 4.7 - Pixels mapeados conforme a definição inicial de cada agrupamento.

Uma outra possibilidade de visualização é mapear cada grupo através dos componentes de um padrão de cor formando apenas uma imagem. Sendo assim, o agrupamento utilizando o padrão de cor *RGB* é realizado conforme apresentado na imagem segmentada em escala de cinza na Figura 4.8. Embora o padrão de cor *RGB* tenha sido utilizado para mapear os pixels na imagem de saída, as informações iniciais utilizadas no processo de agrupamento foram obtidas a partir dos três agrupamentos em escala de cinza. Dessa forma, o problema encontrado ao tentar separar um pixel da área sombreada do veículo em movimento persiste. Para melhorar a classificação correta dos pixels, o algoritmo *FCM* é modificado para realizar o agrupamento através de informações do padrão de cor, que nesse exemplo, é o *RGB*. O agrupamento dos pixels referentes às áreas dos veículos em movimento é mapeado em vermelho, os referentes às áreas sombreadas em verde e os referentes às áreas estáticas da imagem em azul. Assim, observa-se através da imagem segmentada em padrão de cor *RGB* apresentada na Figura 4.8 uma distinção maior obtida entre os pixels referentes às áreas dos veículos em movimento, de suas respectivas sombras e da área estática da imagem.

Figura 4.8 - Segmentação através de agrupamentos em escala de cinza e cor.

Embora os resultados do processo de agrupamento para o padrão de cor *RGB* apresentem uma possível solução para a separação entre os pixels dos veículos em movimento e de suas respectivas sombras, ainda há vários pixels classificados incorretamente. Assim, a implementação de outros padrões de cor que apresentam a informação de luminância e cor separadas podem ajudar na classificação dos pixels. Como as cores dos veículos em movimento variam durante a sequência de imagens, o processo de agrupamento deve considerar a informação temporal. Consequentemente, a solução requer além da extração de mais informações da imagem, uma forma de combiná-las. Além disso, uma análise criteriosa deve ser realizada na implementação desses algoritmos de agrupamento em sua forma original, já que requerem inicialmente a definição do número de grupos, os critérios para o agrupamento e processam uma mesma imagem várias vezes.

Uma análise comparativa realizada entre os algoritmos *k-means* e *FCM* em [162] apresenta resultados próximos no processo de agrupamento, entretanto o algoritmo *FCM* requer mais tempo de processamento devido aos cálculos das funções fuzzy.

4.6 Considerações finais

A implementação em FPGA de sistemas fuzzy para a transformação do padrão de cor para uma representação da imagem em escala monocromática não é viável em termos de recursos computacionais. Isso ocorre porque dependendo da etapa a ser implementada, os resultados nem sempre são melhores que as implementações através de técnicas convencionais de processamento digital de imagens. Assim, a implementação de etapas do processamento de imagens utilizando a abordagem através de sistemas fuzzy deve ser avaliada criteriosamente.

Os conceitos envolvendo os agrupamentos fuzzy apresentam um papel importante no desenvolvimento de algoritmos capazes de tratar com informações inerentemente imprecisas. No entanto, a tomada de decisão nesses algoritmos em sua forma original é realizada

utilizando apenas um critério referente ao centro de cada agrupamento. Dessa forma, a tomada de decisão fica comprometida na detecção de veículos em movimento. Além disso, devido ao processamento iterativo realizado em cada imagem, a frequência de operação da implementação em FPGA de sistemas com resposta em tempo real deve permitir o processamento em uma taxa superior a da aquisição das imagens.

A tomada de decisão multicritério utilizando a integral fuzzy pode ser implementada de forma a melhorar a classificação dos pixels. Isso ocorre devido à possibilidade de tratar a incerteza utilizando uma combinação de informações extraídas independentemente da sequência de imagens. Assim, as medidas de similaridade podem ser processadas paralelamente, o que justifica a sua implementação no sistema de tomada de decisão proposto para FPGA na Seção 5.3.2.

5 O sistema de tomada de decisão proposto

Neste capítulo é apresentado o sistema de tomada de decisão para detecção de veículos em movimento para FPGA proposto. A técnica para modelar a imagem de referência, as medidas, a sequência de imagens e a metodologia de implementação em FPGA utilizadas para validar o sistema proposto são apresentadas. Uma análise sobre o desempenho em termos de classificação dos pixels do sistema de tomada de decisão proposto em FPGA de baixo custo é apresentada. Os recursos ocupados e a frequência máxima de operação também são apresentados e analisados.

5.1 Considerações iniciais

Para que o sistema de tomada de decisão para detecção de veículos em movimento proposto utilize poucos recursos em termos de memória é considerada uma técnica de modelagem simples para obter a imagem de referência sem veículos em movimento. Dessa forma, busca-se a classificação correta dos pixels através de um sistema de tomada de decisão que combina várias informações referentes às áreas dos veículos em movimento através da decisão fuzzy e de suas respectivas sombras através da decisão clássica. No entanto, durante o processamento da tomada de decisão fuzzy devem ser extraídas apenas as informações das imagens que melhorem o desempenho em termos de classificação dos pixels. Isso ocorre porque a quantidade de combinações para calcular a integral fuzzy aumenta exponencialmente com a quantidade de critérios adotados.

A tomada de decisão clássica é combinada paralelamente no sistema proposto buscando melhorar a classificação dos pixels em relação à implementação apenas da tomada de decisão fuzzy. Além disso, esse sistema deve apresentar um bom desempenho em termos de classificação dos pixels em relação aos métodos de detecção de objetos em movimento existentes, processar as imagens em tempo real em FPGA de baixo custo e utilizar poucos recursos, já que sistemas completos de monitoramento de tráfego inteligentes possuem várias outras etapas, como o rastreamento automático de veículos sob suspeita de algum tipo de infração de trânsito.

5.2 A imagem de referência

Uma forma simples para obter a imagem de referência em um sistema de monitoramento baseado em vídeo é considerar apenas a primeira imagem capturada na sequência. Assim, caso ocorra uma reinicialização do sistema, a mudança de iluminação da imagem capturada é rapidamente considerada. Nos trabalhos apresentados em [9,167,168] é proposta uma abordagem utilizando um modelo da imagem de referência baseado na escolha aleatória entre os pixels vizinhos em pequenas regiões na primeira imagem. Essa estratégia apresenta como principal vantagem o tempo de inicialização, já que o sistema de tomada de decisão para detecção de veículos em movimento já é válido a partir da segunda imagem. O problema ocorre quando veículos em movimento estão presentes na primeira imagem, embora tendem a desaparecer ao longo do tempo, há um aumento na classificação incorreta dos pixels [169].

Um dos objetivos nesta tese é realizar a tomada de decisão entre a imagem atual e uma imagem de referência modelada com apenas uma imagem armazenada. Assim, a imagem de referência utilizada para validar o sistema proposto baseia-se nos trabalhos apresentados em [127–129]. Nesses trabalhos é realizada uma operação para estimar o valor mediano entre os pixels de várias imagens. Dessa forma, é possível reduzir a quantidade de recursos em termos de memória ao invés da implementação da média aritmética entre várias imagens armazenadas no início da sequência em [15,75], já que para estimar o valor mediano dos pixels não há necessidade do armazenamento das imagens processadas durante a inicialização do sistema.

Nos trabalhos apresentados em [127–129] é utilizada a função sinal sgn(x) para estimar o valor mediano dos pixels em várias imagens. Essa função é apresentada na Figura 5.1.

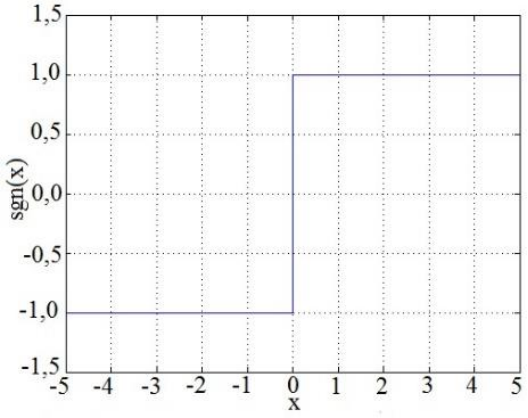


Figura 5.1 - Função sinal.

A estimativa do valor mediano é obtida comparando-se o valor do pixel em cada posição da imagem de referência (I_r) com o valor do pixel na respectiva posição da imagem atual (I_a). Sendo assim, o valor mediano estimado é obtido incrementando por um se o valor do pixel da imagem de referência é menor que na imagem atual ou decrementado por um, se é maior. À medida em que as imagens são processadas há uma atualização dinâmica da primeira imagem até obter a imagem de referência livre de veículos em movimento. Considerando a aleatoriedade das informações obtidas da imagem atual, haverá a convergência dos valores de cada pixel para o seu valor mediano [127]. A função apresentada na Figura 5.1 é definida em (5.1) para as variáveis utilizadas.

$$\operatorname{sgn}\left(I_a - I_r\right) = \begin{cases} -1, \text{ se } I_a - I_r < 0 \\ 0, \text{ se } I_a - I_r = 0 \\ +1, \text{ se } I_a - I_r > 0 \end{cases} \tag{5.1}$$

Para testar o sistema de tomada de decisão proposto, o valor mediano é estimado processando as 469 imagens do início da sequência. Isso ocorre porque todas as medidas apresentadas na Tabela 5.4 para os outros métodos de detecção de objetos em movimento são obtidas a partir da imagem 470. Dessa forma, haverá uma única imagem no padrão de cor *RGB* armazenada como referência no final desse processo.

Os pixels referentes às áreas de sombras dos veículos em movimento não representam um problema durante a modelagem da imagem de referência. Isso ocorre porque a cor desses pixels difere da cor da pavimentação asfáltica da imagem de referência. Além disso, para a situação na qual a cor do veículo é similar à cor de sua sombra, ambos os pixels são descartados.

Vale ressaltar que esta proposta modela bem imagem unimodal, como ocorre na maioria das imagens capturadas em rodovias. Dessa forma, a imagem de referência pode ser obtida por um processo de modelagem mais simples. Contudo, a classificação correta dos pixels referentes às áreas dos veículos em movimento deve ser compensada pelo sistema de tomada de decisão proposto na Seção 5.3 a seguir.

5.3 O sistema de tomada de decisão

A tomada de decisão multicritério através da integral fuzzy possibilita um aumento na classificação correta dos pixels. Isso ocorre devido à utilização de mais informações durante a tomada de decisão. Assim, além de informações individuais sobre cada pixel, são combinadas informações contidas nas regiões de sua vizinhança. No entanto, vários pixels pertencentes às áreas de sombras dos veículos em movimento ainda são classificados incorretamente. Dessa

forma, é proposto um processamento paralelo para identificar os pixels referentes às áreas de sombras dos veículos em movimento através da tomada de decisão clássica. Na Figura 5.2 é apresentado o sistema de tomada de decisão proposto para detectar os veículos em movimento através da fusão da tomada de decisão fuzzy e da tomada de decisão clássica. Nessa proposta são implementadas medidas em três componentes de dois padrões de cor. Para que isso ocorra, as informações inicialmente são transformadas do padrão de cor *RGB* para o *YCbCr* e para o *HSV*, paralelamente. As informações utilizadas para detectar os pixels referentes às áreas dos veículos em movimento são extraídas do padrão de cor *YCbCr*. Já as informações utilizadas para detectar os pixels referentes às áreas de sombras desses veículos são extraídas do padrão de cor *HSV*. Esses padrões de cor são escolhidos por apresentarem resultados experimentais que comprovam sua eficiência na segmentação de objetos em movimento em [16] e de suas respectivas áreas de sombras em [13].

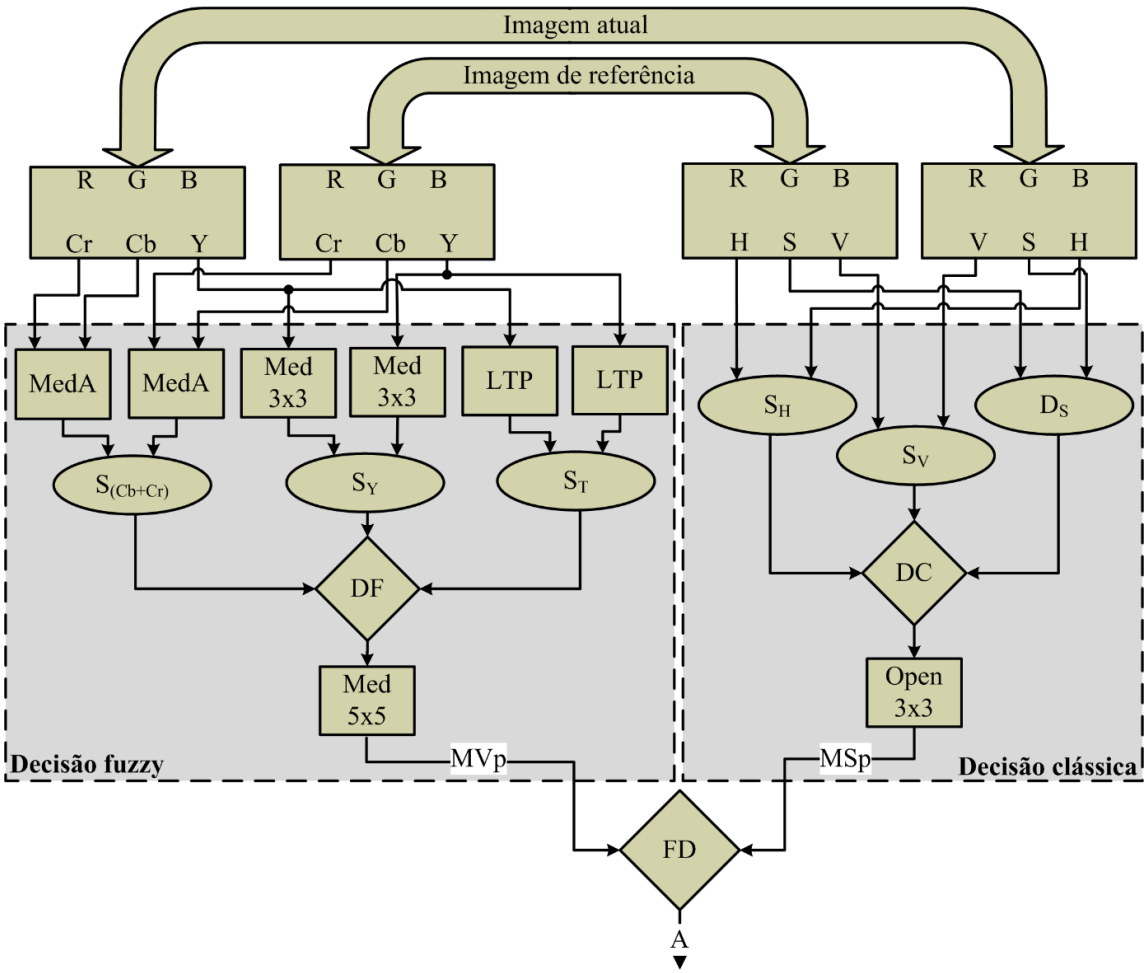


Figura 5.2 - Sistema de tomada de decisão proposto.

Para a tomada de decisão fuzzy é realizada a extração da informação de textura no bloco *LTP*. Essa extração é realizada sobre o componente de luminância (*Y*) do padrão *YCbCr* através do descritor *LTP* apresentado na Seção 2.4.2. O valor da tolerância (τ) ao ruído que

apresenta resultados melhores é de 10% sobre o elemento central. A implementação do descritor *LTP* ao invés do *LBP* utilizado nas propostas apresentadas em [15–18] é motivada por tornar a tomada de decisão fuzzy menos sensível à variação de iluminação e ao ruído. A arquitetura do bloco *LTP* é apresentada na Figura 5.3 através de um diagrama de blocos.

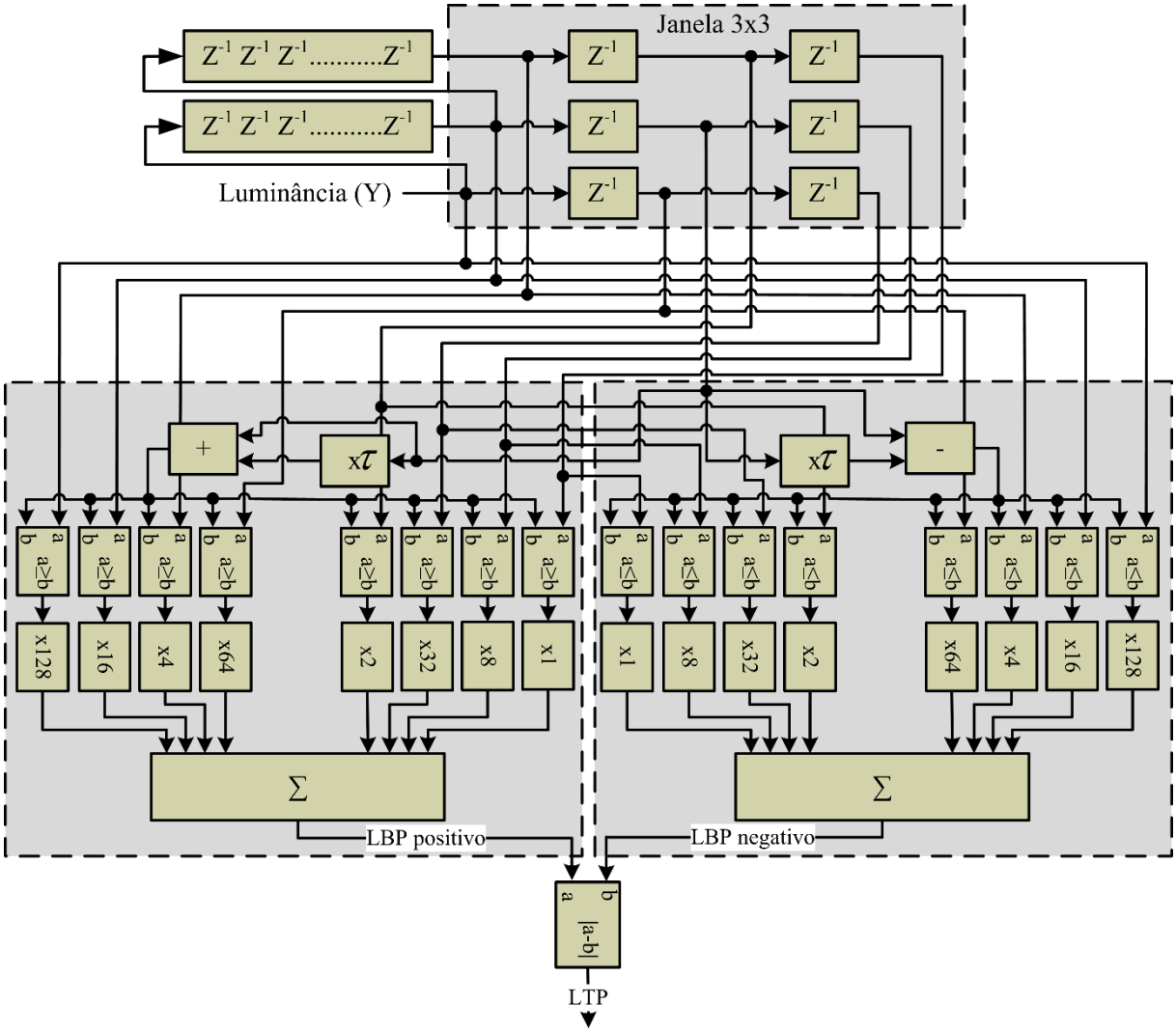


Figura 5.3 - Arquitetura do bloco *LTP*.

Através da arquitetura apresentada na Figura 5.3 verifica-se que o bloco *LTP* realiza testes condicionais em uma janela 3x3. Sendo assim, para que os nove pixels dessa região sejam processados simultaneamente, os pixels referentes ao componente de luminância são inicialmente paralelizados. Esse processo é realizado através de blocos de atrasos (Z^{-1}) configurados de acordo com a quantidade de colunas da imagem. A implementação do bloco *LTP* é resultante do valor absoluto da subtração entre o padrão positivo (*LBP* positivo) e o negativo (*LBP* negativo).

No bloco *Med 3x3* apresentado na Figura 5.2 é realizada a operação de filtragem através do valor mediano em uma janela de tamanho 3x3. Essa operação busca homogeneizar

pequenas regiões na imagem reduzindo o efeito de variação da iluminação nos resultados das medidas de similaridade no componente de luminância. Para processar as informações referentes às cores é realizada a média aritmética entre os componentes *Cb* e *Cr* em cada posição da imagem no bloco *MedA*. Essa simplificação é possível porque a medida de similaridade obtida com o valor médio entre esses componentes apresentam praticamente as mesmas informações que quando processadas separadamente, reduzindo o número de critérios para o cálculo da integral fuzzy.

A tomada de decisão multicritério baseada na integral fuzzy proposta utiliza as informações de três matrizes representando as funções fuzzy. Essas matrizes são obtidas através das medidas de similaridade implementadas nos blocos $S_{(Cb+Cr)}$, S_Y e S_T para os critérios de cor, luminância e textura, respectivamente. De acordo com os estudos apresentados em [15] a integral proposta por Choquet permite obter resultados melhores que Sugeno na classificação dos pixels durante a detecção de objetos em movimento. Assim, a integral fuzzy implementada nesta tese é um caso específico da integral de *Choquet* para a situação na qual não é considerada a interação entre os critérios. Dessa forma, o cálculo da integral é baseado no operador *OWA*, implementado através do bloco *Decisão Fuzzy* (*DF*). Para que esse processo de decisão seja realizado é necessária a determinação dos pesos de acordo com a importância de cada critério utilizado. Esses pesos são obtidos empiricamente e dependem da sequência de imagens. Após a realização da tomada de decisão no bloco *DF* a imagem é processada no bloco *Med 5x5*. Esse bloco realiza a operação de filtragem através do valor mediano em uma janela de tamanho 5x5 para reduzir a classificação incorreta dos pixels. Sendo assim, a máscara binária dos pixels referentes às áreas dos veículos em movimento e às áreas estáticas da imagem (*MVp*) é obtida.

A tomada de decisão clássica é implementada através do bloco *Decisão Clássica* (*DC*). A medida de dissimilaridade é realizada através da diferença entre a imagem atual e a de referência processada no bloco D_S para o componente de saturação (*S*). As medidas de similaridade são implementadas para processar os componentes de tonalidade (*H*) e brilho (*V*) nos blocos S_H e S_V, respectivamente. Após a realização da tomada de decisão no bloco *DC* é aplicada a operação morfológica de abertura no bloco *Open 3x3*. Essa operação busca remover regiões de pixels com tamanho inferior às áreas de sombras dos veículos em movimento na operação de erosão e recuperar essas áreas na operação de dilatação. Os elementos estruturantes apresentados na Seção 2.2.3 são testados e a melhor classificação dos pixels é obtida com o elemento estruturante *Ones* para janela 3x3. Após a operação morfológica é obtida a máscara binária dos pixels referentes às áreas de sombras e às áreas

estáticas da imagem (*MSp*).

No bloco *Fusão das Decisões* (*FD*) é realizada a fusão das duas tomadas de decisão representadas pelas máscaras binárias *MVp* e *MSp*.

As medidas de similaridade são realizadas entre a imagem atual e uma imagem de referência obtida pela técnica de modelagem apresentada na Seção 5.2. O sistema de tomada de decisão para detecção de veículos em movimento proposto é testado com a imagem de referência enviada da área de trabalho do software MATLAB® conforme apresentado na Seção 5.4.3. Sendo assim, o processo de inicialização para obter a imagem de referência não é representado na Figura 5.2.

5.3.1 A tomada de decisão clássica

A tomada de decisão clássica é realizada com base na proposta apresentada em [13,14]. Nessa proposta são mapeados os pixels da área sombreada ao redor dos veículos em movimento através de três condições que envolvem os componentes *H*, *S* e *V* do padrão de cor *HSV*. Os autores determinam a área de sombras considerando que o brilho dessa área é menor do que o brilho dos pixels na imagem de referência. Além disso, os parâmetros referentes à tonalidade em 0,5 e à saturação em 0,1 apresentam os melhores resultados na determinação dessas áreas. A extração de informações do padrão de cor *HSV* permite obter resultados melhores que do padrão de cor *RGB* na discriminação das áreas de sombras. Portanto, justifica-se a utilização de mais recursos em hardware para a transformação entre esses padrões. Com base nessas informações são determinadas as regras utilizadas no processamento da tomada de decisão clássica. Essas regras utilizam a medida de similaridade implementada no bloco S_V apresentada em (5.2), a medida de dissimilaridade implementada no bloco D_S apresentada em (5.3) e a medida de similaridade implementada no bloco S_H apresentada em (5.4). Assim, a matriz binária *MS* é obtida através da combinação dessas medidas utilizando a operação de lógica clássica *E* ($\wedge$) em (5.5).

$$S_V(x,y) = \frac{\min\left(I_V(x,y), R_V(x,y)\right)}{\max\left(I_V(x,y), R_V(x,y)\right)} \tag{5.2}$$

$$D_S(x,y) = \frac{I_S(x,y) - R_S(x,y)}{100} \tag{5.3}$$

$$S_H(x,y) = \frac{\min\left(I_H(x,y), R_H(x,y)\right)}{\max\left(I_H(x,y), R_H(x,y)\right)} \tag{5.4}$$

$$MS(x,y) = \begin{cases} 1, \text{ se } \alpha \le S_V(x,y) \le \beta \ \wedge \ D_S(x,y) \le \tau_S \wedge \ S_H(x,y) \le \tau_H \\ 0, \text{ caso contrário} \end{cases} \quad (5.5)$$

A arquitetura do bloco *DC* é implementada de acordo com o diagrama de blocos apresentado na Figura 5.4.

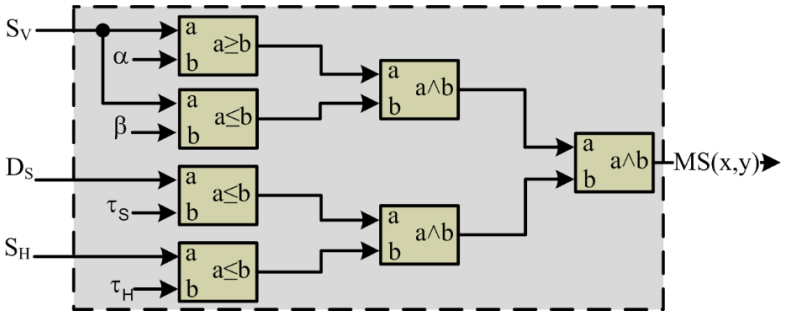


Figura 5.4 - Arquitetura do bloco *DC*.

Os parâmetros α, β, τ_S e τ_H são determinados experimentalmente entre 0 e 1. No entanto, para manter os parâmetros α e β sempre entre 0 e 1, em (5.2) são utilizados os operadores de mínimo (min) e máximo (max) em relação à medida de similaridade apresentada em [13,14]. O parâmetro β adotado é 0,95, o que representa um brilho cerca de 5% menor para a área sombreada. O aumento desse parâmetro torna o processo de detecção de sombras mais sensível ao ruído [4]. Considerando o brilho dos pixels na área sombreada da imagem atual em torno de 15% menor que o dos pixels na imagem de referência, tem-se para o parâmetro α um valor de 0,85.

As áreas sombreadas apresentam uma redução na saturação [13,14]. Assim, considerando que a diferença entre a saturação dos pixels na área sombreada da imagem atual para a imagem de referência apresenta, geralmente, um resultado negativo, em (5.3) o sinal é testado ao invés do resultado de sua diferença. Dessa forma, o parâmetro τ_S determinado experimentalmente em [13] com o valor de 0,10 é implementado com o valor igual a 0. Para o parâmetro τ_H é adotado o valor de 0,35. Além disso, a medida de similaridade implementada em (5.4) apresenta resultados melhores que a medida de dissimilaridade inicialmente proposta em [13,14]. Isso ocorre porque é possível manter um limiar global que satisfaça a condição de similaridade tanto para os valores abaixo de 0,5, bem como os acima.

Os resultados dos blocos que realizam as medidas de similaridade devem apresentar valores entre 0 e 1. Dessa forma, implementa-se através de dois blocos multiplexadores (*Mux_max* e *Mux_min*) uma lógica para selecionar (*Sel*) para o numerador do bloco de divisão (a/b) sempre o menor valor. Devido à possibilidade de ocorrer indeterminação matemática (0/0) nos resultados dessas medidas é implementado um bloco multiplexador (*Mux*) para que

nessa situação, a saída (*S*) apresente máxima similaridade através da seleção de sua segunda entrada. A arquitetura desses blocos é apresentada na Figura 5.5.

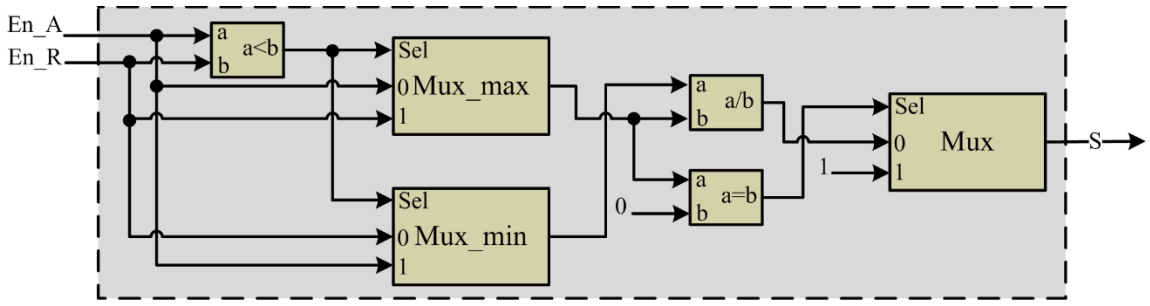


Figura 5.5 - Arquitetura dos blocos de medida de similaridade.

5.3.2 A tomada de decisão fuzzy

A tomada de decisão fuzzy é realizada no bloco *DF* com base nas propostas apresentadas em [15–18]. Nesse bloco é implementada a decisão através da limiarização do resultado da integral de *Choquet*. Assim, os pixels são considerados parte dos veículos em movimento se a integral de *Choquet* for menor ou igual a um determinado limiar (*Lim*). Esse limiar depende das condições do ambiente de captura da sequência de imagens. Como resultado do processo de tomada de decisão fuzzy é obtida a matriz binária *MV* em (5.6).

$$MV(x,y) = \begin{cases} 1, & C_g(x,y) \leq Lim \\ 0, & \text{caso contrário} \end{cases} \tag{5.6}$$

A integral fuzzy é calculada utilizando a função fuzzy $h(x_i)$ apresentada em (4.14) no quarto capítulo. Essa função é obtida através das medidas de similaridade implementadas nos blocos $S_{(Cb+Cr)}$, S_Y e S_T entre os três critérios extraídos de uma imagem de referência e da imagem atual. A integral de *Choquet* é implementada para combinar esses três critétrios, o que resulta em seis possibilidades. Essas combinações são determinadas de acordo com a ordem crescente dos valores obtidos nas medidas de similaridade. Assim, para as condições possíveis tem-se os cálculos da integral discreta de *Choquet* apresentados nas equações (5.7) a (5.12). Na implementação realizada todas as condições são testadas paralelamente selecionando a respectiva entrada de um multiplexador.

a) $S_Y \leq S_T \leq S_{(Cb+Cr)}$

$$C_g(\text{a}) = S_Y.g(\{x_1,x_2,x_3\}) + (S_T - S_Y).g(\{x_2,x_3\}) + (S_{(Cb+Cr)} - S_T).g(\{x_3\}) \tag{5.7}$$

b) $S_Y \leq S_{(Cb+Cr)} \leq S_T$

$$C_g(\text{b}) = S_Y.g(\{x_1,x_2,x_3\}) + (S_{(Cb+Cr)} - S_Y).g(\{x_2,x_3\}) + (S_T - S_{(Cb+Cr)}).g(\{x_2\}) \tag{5.8}$$

c) $S_T \leq S_Y \leq S_{(Cb+Cr)}$

$$C_g(\mathrm{c}) = S_T.g(\{x_1,x_2,x_3\}) + (S_Y - S_T).g(\{x_1,x_3\}) + (S_{(Cb+Cr)} - S_Y).g(\{x_3\}) \quad (5.9)$$

d) $S_T \leq S_{(Cb+Cr)} \leq S_Y$

$$C_g(\mathrm{d}) = S_T.g(\{x_1,x_2,x_3\}) + (S_{(Cb+Cr)} - S_T).g(\{x_1,x_3\}) + (S_Y - S_{(Cb+Cr)})).g(\{x_1\}) \quad (5.10)$$

e) $S_{(Cb+Cr)} \leq S_T \leq S_Y$

$$C_g(\mathrm{e}) = S_{(Cb+Cr)}.g(\{x_1,x_2,x_3\}) + (S_T - S_{(Cb+Cr)}).g(\{x_1,x_2\}) + (S_Y - S_T).g(\{x_1\}) \quad (5.11)$$

f) $S_{(Cb+Cr)} \leq S_Y \leq S_T$

$$C_g(\mathrm{f}) = S_{(Cb+Cr)}.g(\{x_1,x_2,x_3\}) + (S_Y - S_{(Cb+Cr)}).g(\{x_1,x_2\}) + (S_T - S_Y).g(\{x_2\}) \quad (5.12)$$

Para *n* critérios adotados, existem 2^n - 2 possibilidades para ordená-los de forma crescente. Consequentemente, a implementação deve considerar todas as possibilidades para que a integral de *Choquet* seja calculada corretamente. Isso faz com que a escolha de critérios mais adequados possibilite manter uma boa relação entre a classificação correta dos pixels, tempo de processamento e utilização dos recursos da FPGA.

O critério mais importante na tomada de decisão fuzzy representa a maior relevância. Essa relevância é atribuída através da medida fuzzy $g(A_i)$. Na Tabela 5.1 são apresentados alguns valores atribuídos às medidas fuzzy para cada critério durante os testes realizados no software MATLAB®. A atribuição desses valores considera uma medida fuzzy aditiva.

Tabela 5.1 - Medidas fuzzy.

$g(\{x_1\})$	$g(\{x_2\})$	$g(\{x_3\})$	$g(\{x_1,x_2\})$	$g(\{x_1,x_3\})$	$g(\{x_2,x_3\})$
0,8	0,1	0,1	0,9	0,9	0,2
0,7	0,2	0,1	0,9	0,8	0,3
0,6	**0,3**	**0,1**	**0,9**	**0,7**	**0,4**
0,6	0,3	0,2	0,9	0,8	0,5
0,5	0,3	0,2	0,8	0,7	0,5
0,4	0,3	0,3	0,7	0,7	0,6
0,3	0,4	0,3	0,7	0,6	0,7
0,2	0,5	0,3	0,7	0,5	0,8
0,1	0,5	0,4	0,6	0,5	0,9

Devido aos resultados melhores obtidos em termos de classificação dos pixels, o peso de cada critério é determinado com base nos valores apresentados na linha em negrito da Tabela 5.1. Sendo assim, esses valores são refinados, obtendo-se para o critério de luminância ($g(\{x_1\})$), textura ($g(\{x_2\})$) e cor ($g(\{x_3\})$), os pesos de 0,60; 0,32 e 0,08, respectivamente.

A arquitetura interna do bloco *DF* é apresentada na Figura 5.6 através de um diagrama de blocos. De acordo com os testes condicionais ($a \leq b$) iniciais, é verificada a ordem dos valores obtidos através das medidas de similaridade S_Y, S_T e $S_{(Cb+Cr)}$. Essa ordem é

representada por três bits que são concatenados (*Concat*) para selecionar a respectiva entrada de um bloco multiplexador (*Mux*). De acordo com os testes condicionais iniciais, as entradas 2 e 5 não são selecionáveis (*N.s.*), no entanto a implementação requer a atribuição de um valor às referidas entradas. Assim, para as entradas 2 e 5 são atribuídos o valor 1.

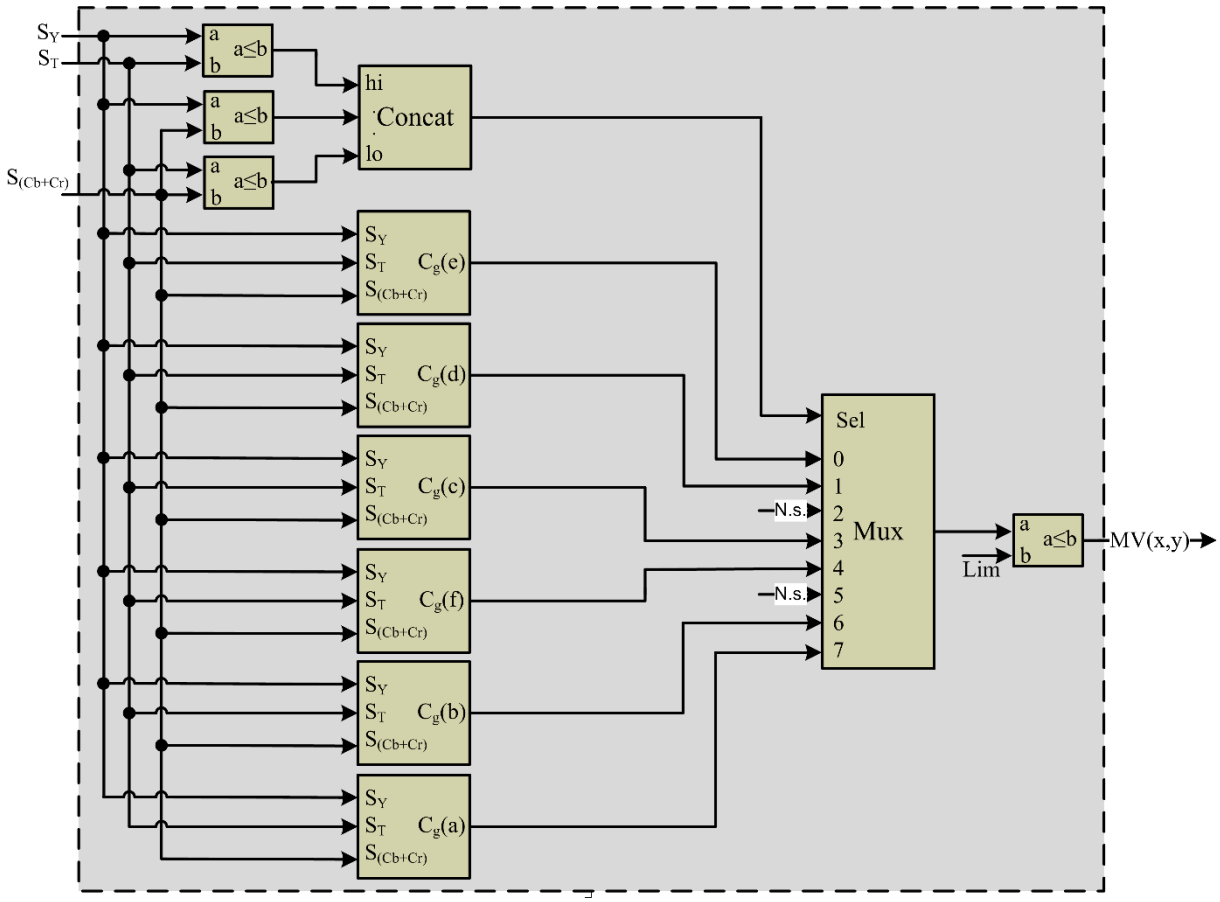


Figura 5.6 - Arquitetura do bloco *DF*.

5.3.3 A fusão das tomadas de decisão fuzzy e clássica

Após a realização do processamento da tomada de decisão fuzzy e clássica sobre cada posição da imagem, ocorre a fusão dos resultados através do bloco *FD*. Essa fusão é implementada através de uma operação de lógica clássica utilizando como informações de entrada as matrizes binárias *MVp* e *MSp*. Essas informações resultam na matriz *A* contendo a imagem binária de saída de acordo com as condições apresentadas em (5.13). As regiões de pixels formadas pelos veículos em movimento são determinadas pelo nível lógico 1, enquanto o nível lógico 0 representa as regiões referentes aos pixels estáticos e as sombras dos veículos em movimento.

$$A(x,y) = \begin{cases} 1, \text{ se } MVp(x,y) = 1 \wedge MSp(x,y) = 0 \\ 0, \text{ caso contrário} \end{cases} \tag{5.13}$$

A operação de lógica clássica *XOu* apresenta uma tabela da verdade próxima da fusão de decisões proposta em (5.13). Essa operação de lógica clássica atende às condições nas quais as regiões das matrizes binárias *MVp* e *MSp* representam os pixels estáticos (*MSp(x,y) = 0*, *MVp(x,y) = 0*); os pixels estáticos da matriz binária das sombras dos veículos em movimento (*MSp(x,y) = 0*) e dos pixels em movimento da matriz binária dos veículos em movimento (*MVp(x,y) = 1*); e os pixels das sombras da matriz binária das sombras dos veículos em movimento (*MSp(x,y) = 1*) e dos pixels em movimento da matriz binária dos veículos em movimento (*MVp(x,y) = 1*). Contudo, para a situação na qual *MSp(x,y)* = 1 e *MVp(x,y)* = 0, a saída *A(x,y)* não pode apresentar nível lógico 1. Isso ocorre porque a região considerada como área das sombras dos veículos em movimento deve ter seus pixels classificados como sendo estáticos. Sendo assim, para realizar a fusão da tomada de decisão fuzzy e da tomada de decisão clássica é implementada uma operação de lógica clássica *E* entre cada posição da matriz binária *MVp* e do resultado da inversão lógica de cada posição da matriz binária *MSp*. Na Tabela 5.2 é apresentada a saída *A(x,y)* para operação de lógica clássica *XOu* testada inicialmente e para operação de lógica clássica *E* implementada no bloco *FD*. As colunas em negrito representam a operação implementada nesse bloco.

Tabela 5.2 - Tabela da verdade para implementação do bloco *FD*.

MVp(x,y)	*MSp(x,y)*	***~MSp(x,y)***	*A(x,y)* (lógica *XOu*)	***A(x,y)*(lógica *E*)**
0	0	**1**	0	**0**
0	1	**0**	1	**0**
1	0	**1**	1	**1**
1	1	**0**	0	**0**

5.4 Resultados experimentais

A matriz de pixels *A* representa a imagem binária na saída do bloco *FD*. Essa matriz binária possui o conjunto de pixels classificados como pertencentes aos veículos em movimento e à região estática de cada imagem na sequência. Assim, cada pixel pode assumir classificação positiva, se pertencer à região do veículo em movimento. Caso pertença à região estática da imagem, a sua classificação é negativa. A região de cada veículo considerado em movimento é extraída manualmente para determinar uma imagem ideal (*ground truth*). Essa imagem é utilizada para comparação com a imagem original. Sendo assim, há quatro situações distintas para a classificação de cada pixel. Se um pixel é parte do veículo em movimento, é classificado como verdadeiro-positivo (*True Positive* - *TP*); se esse pixel é classificado incorretamente como parte do veículo em movimento, é classificado como falso-positivo (*False Positive* - *FP*). Caso o pixel seja classificado corretamente como parte estática da imagem, é classificado como verdadeiro-negativo (*True Negative* - *TN*); se esse

pixel é classificado incorretamente como parte estática da imagem, é classificado como falso-negativo (*False Negative* - *FN*).

5.4.1 As medidas utilizadas

A análise dos resultados do sistema proposto para detecção de objetos em movimento apresentada em [117] é realizada a partir da comparação da imagem processada com uma imagem contendo apenas os objetos em movimento extraídos manualmente. Essa comparação permite uma avaliação quantitativa em termos dos erros de falso-negativos e falso-positivos separadamente. No entanto, uma abordagem mais robusta para avaliação dos resultados proposta em [170] considera as informações de falso-negativos e falso-positivos em uma mesma medida. Essa avaliação quantitativa também é realizada através de medidas de similaridade entre a imagem de saída do sistema proposto e de uma imagem ideal. Nesse contexto, quando os pixels dos objetos em movimento coincidem com os pixels extraídos manualmente, o resultado da medida é 1, e em caso contrário é 0. Essa medida de similaridade é utilizada para avaliar os resultados em vários trabalhos, dentre os quais destacam-se os apresentados em [15,18,171]. A avaliação dos resultados em termos de classificação dos pixels realizada nesta tese baseia-se nesses trabalhos. A medida de similaridade entre os pixels da imagem binária de saída (*A*) e os pixels da região em movimento extraídos manualmente (*B*) é apresentada em (5.14).

$$S(A,B) = \frac{A(x,y) \cap B(x,y)}{A(x,y) \cup B(x,y)} \tag{5.14}$$

Do exposto, tem-se as condições para a classificação dos pixels apresentadas na Tabela 5.3. Através da combinação dos pixels classificados em *TN*, *FN*, *FP* e *TP* há a possibilidade de calcular várias medidas. Entre essas medidas destacam-se a proporção dos pixels classificados incorretamente, taxa de detecção de verdadeiro-positivo, taxa de detecção de verdadeiro-negativo, taxa de detecção de falso-positivo, taxa de detecção de falso-negativo, precisão e F_1 [172–174], as quais são utilizadas para comparar o desempenho do sistema de tomada de decisão proposto nesta tese. Isso ocorre porque esse conjunto de medidas é o mesmo apresentado pelas outras propostas disponibilizadas publicamente em [72–74].

Tabela 5.3 - Condições para classificação dos pixels.

A(x,y)	*B(x,y)*	*S(A,B)*	Classificação dos pixels
0	0	1	*TN*
0	1	0	*FN*
1	0	0	*FP*
1	1	1	*TP*

Através das informações apresentadas na Tabela 5.3 e da sequência de imagens para teste apresentada na Seção 5.4.2, uma função no software MATLAB® é desenvolvida para contabilizar o número de pixels classificados como verdadeiro-negativo, falso-negativo, falso-positivo e verdadeiro-positivo entre as imagens válidas disponíveis. Esses valores são utilizados para calcular as medidas apresentadas a seguir.

As classificações incorretas (*Wrong Classifications* - *WC*) referem-se à proporção dos pixels classificados incorretamente (*FP*+*FN*) em relação a todos os pixels classificados (*TP*+*TN*+*FP*+*FN*). Essa medida considera as classificações incorretas contabilizadas tanto dos pixels dos veículos em movimento, como dos pixels estáticos. No entanto, essa medida é altamente susceptível ao desbalanceamaneto do conjunto de dados, podendo facilmente induzir a uma conclusão errada sobre o desempenho do sistema de tomada de decisão. Assim, torna-se necessária a análise em conjunto de outras medidas. A quantidade de pixels classificados incorretamente é calculada através da equação (5.15).

$$WC = \frac{FP + FN}{TP + TN + FP + FN} \tag{5.15}$$

A taxa de detecção de verdadeiro-positivo (*True Positive Rate* - *TPR*) ou sensibilidade é a proporção de pixels dos veículos em movimento classificados corretamente (*TP*) em relação aos pixels classificados na região de veículos em movimento (*TP*+*FN*). Assim, essa medida calcula a proporção apenas na região contendo os veículos em movimento. Essa taxa de detecção é calculada através da equação (5.16).

$$TPR = \frac{TP}{TP + FN} \tag{5.16}$$

A taxa de detecção de verdadeiro-negativo (*True Negative Rate* - *TNR*) ou especificidade é a proporção entre os pixels da parte estática da imagem classificados corretamente (*TN*) e os pixels classificados na região da parte estática da imagem (*TN*+*FP*). Essa taxa é calculada através da equação (5.17).

$$TNR = \frac{TN}{TN + FP} \tag{5.17}$$

A taxa de detecção de falso-positivo (*False Positive Rate* - *FPR*) é a proporção dos pixels da parte estática da imagem que são classificados incorretamente (*FP*) e os pixels classificados na região da parte estática da imagem (*FP*+*TN*). Essa taxa é calculada através da equação (5.18).

$$FPR = \frac{FP}{FP + TN} \tag{5.18}$$

A taxa de detecção de falso-negativo (*False Negative Rate - FNR*) é a proporção dos pixels dos veículos em movimento que são classificados incorretamente (*FN*) e os pixels classificados na região dos veículos em movimento (*FN+TP*). Essa taxa é calculada através da equação (5.19).

$$FNR = \frac{FN}{FN + TP} \tag{5.19}$$

A precisão (*Precision - Pr*) é a proporção dos pixels dos veículos em movimento classificados corretamente (*TP*) em relação a todos os pixels classificados como parte dos veículos em movimento (*TP+FP*). A precisão é calculada através da equação (5.20).

$$Pr = \frac{TP}{TP + FP} \tag{5.20}$$

Um método de classificação deve apresentar uma taxa elevada de detecção de verdadeiro-positivo, mas sem comprometer a sua precisão [174]. Sendo assim, uma medida mais confiável denominada F_1 envolve a taxa de detecção de verdadeiro-positivo e a precisão. Essa medida é calculada através da equação (5.21).

$$F_1 = \frac{2 \times TPR \times Pr}{TPR + Pr} \tag{5.21}$$

5.4.2 A sequência de imagens para teste

De acordo com a proposta apresentada nesta tese, a detecção de objetos em movimento é implementada através da fusão da tomada de decisão fuzzy e da tomada de decisão clássica baseada no processamento de imagens capturadas em rodovias. Sendo assim, a sequência de imagens analisada considera uma câmera fixa e posicionada frontalmente ao sentido do tráfego de veículos. Dessa forma, através da mesma câmera há a possibilidade de capturar a identificação dos veículos através de sua placa para o processamento em etapas posteriores. Para tornar possível a comparação com outras propostas é utilizada a mesma sequência de imagens apresentada nos trabalhos [72–74]. Esses trabalhos utilizam um banco de imagens bem organizado que apresentam os vários problemas encontrados na detecção de objetos em movimento. Nesse banco de imagens são apresentadas 53 sequências de imagens agrupadas em 11 categorias diferentes, sendo disponibilizadas publicamente a sequência de imagens originais e as consideradas ideais com sombras.

Como o objetivo desta tese é avaliar o desempenho da classificação dos pixels dos veículos em movimento, não é avaliada a detecção das áreas de sombras desses veículos separadamente. Assim, o método de avaliação é aplicado nos resultados da combinação do

processo de detecção dos pixels referentes aos veículos em movimento e de suas respectivas áreas de sombras, o qual ocorre paralelamente. Na Figura 5.7 é apresentada a sequência de imagens representada através das imagens originais 867, 1195 e 1690, suas respectivas imagens ideais e com as áreas de sombras dos veículos em movimento. Nas imagens ideais com sombras (*B*) as áreas dos veículos em movimento são representadas pelo valor 255 em escala de cinza, as áreas estáticas da imagem por 0 e as áreas de sombras por 50. Nessas imagens utilizadas para teste todos os valores menores ou iguais a 50 são considerados parte estática da imagem, incluindo, assim, as áreas de sombras dos veículos em movimento.

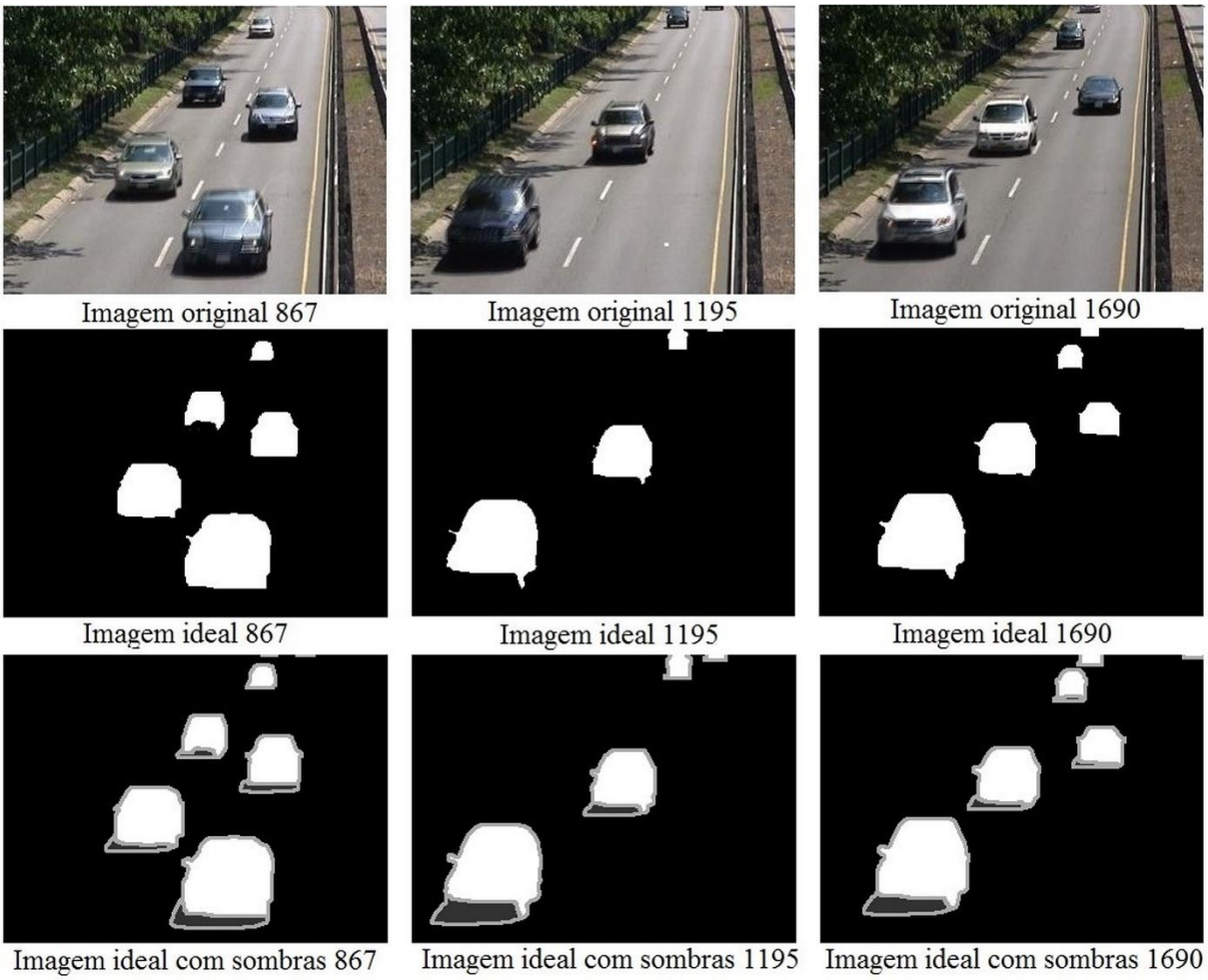


Figura 5.7 - Sequência de imagens para avaliação do desempenho.

A sequência de imagens para teste possui 1700 imagens originais de 320x240 pixels no padrão de cor *RGB* 24 bits. Trata-se de uma sequência de imagens capturada em condição de iluminação variável. Além disso, a sequência de imagens apresenta veículos com os valores dos componentes de cor de vários de seus pixels similares aos da imagem de referência. Sendo assim, de acordo com os critérios adotados, esses pixels podem ser classificados incorretamente, aumentando o número de falso-negativo. Essa sequência

apresenta ainda sombras estáticas sobre a massa asfáltica, podendo ocorrer a classificação incorreta de parte dos pixels da região dos veículos em movimento como sendo parte da imagem de referência. Para a situação na qual os veículos apresentam muitos pixels com os valores dos componentes do padrão de cor similares as suas respectivas áreas sombreadas pode ocorrer um aumento no número de falso-positivo. Dessa forma, tanto o número de falso-positivo como o de falso-negativo na sequência de imagens apresentada dependem das outras informações extraídas das imagens para que o sistema de tomada de decisão apresente resultados melhores em termos de classificação dos pixels.

5.4.3 A metodologia de implementação e teste em FPGA

O fluxo de projetos baseado em modelos é uma abordagem utilizada no desenvolvimento de sistemas em hardware reconfigurável que facilita a reutilização de componentes e a criação de bibliotecas empregando ambientes gráficos para conectar os sinais de blocos diferentes [76]. Os softwares MATLAB® e o Simulink® são ferramentas muito utilizadas no desenvolvimento baseado em modelos e trabalham em conjunto com bibliotecas de componentes dos principais fabricantes de FPGA. A inclusão de bibliotecas de componentes no software Simulink®, como as baseadas nas ferramentas de síntese System Generator® da Xilinx® e DSP Builder da Altera®, permite o desenvolvimento de aplicações complexas em hardware. Com isso, é alcançada a abstração de muitos detalhes sobre as implementações, favorecendo a concentração do projetista no desenvolvimento da proposta [61,67,78,175,176].

A metodologia de implementação em FPGA empregada nesta tese é baseada em modelos desenvolvidos através dos softwares MATLAB®, Simulink® e System Generator®. Assim, através de blocos básicos da biblioteca da Xilinx® é desenvolvido cada bloco do sistema de tomada de decisão para detecção de veículos em movimento proposto. Esses blocos são facilmente reconfiguráveis permitindo a aceleração no tempo de desenvolvimento do sistema proposto. A ferramenta de síntese disponibilizada através do bloco System Generator® permite a geração automática do código, por exemplo, em VHDL [177]. Dessa forma, através do bloco System Generator® da biblioteca da Xilinx® é realizada a verificação da implementação na placa propriamente dita (*HW in loop*). Além disso, o bloco System Generator® permite a compilação dos projetos desenvolvidos para simuladores externos, tais como o ISIM e o ModelSIM. Portanto, além da simulação realizada diretamente com o Simulink®, pode-se verificar o comportamento funcional e temporal da implementação proposta nesses simuladores. Na Figura 5.8 é apresentado o fluxo de projeto através dos

softwares MATLAB®, Simulink®, System Generator®, ISE® Design Suite e ModelSim [67,177,178].

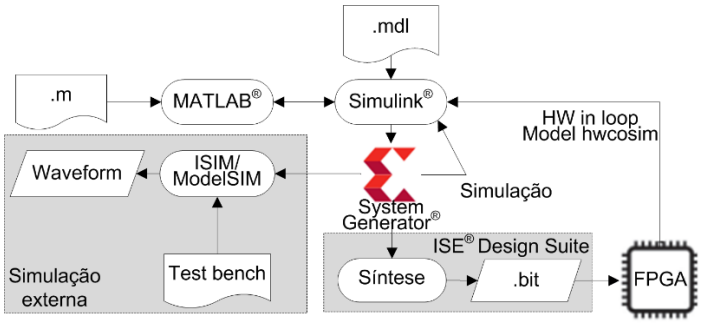


Figura 5.8 - Fluxo do desenvolvimento de projeto com System Generator® [76].

A realização da co-simulação em hardware através do bloco System Generator® permite a síntese em FPGA do sistema de processamento proposto através da ferramenta ISE® Design Suite, gerando o arquivo de configuração (.bit). Para isso é necessária a configuração de parâmetros como o modelo da placa de desenvolvimento, tipo de conexão com o dispositivo (JTAG, USB ou Ethernet), opções de síntese, implementação, temporização e simulação. Assim, concluída a compilação, o sistema cria um novo bloco (*Model hwcosim*) que deverá substituir o sistema de processamento proposto conforme apresentado na Figura 5.9. Ao executar a simulação do modelo em hardware, o bloco System Generator® se encarrega de realizar a síntese, roteamento e a configuração do sistema de processamento na placa conectada. Dessa forma, o sistema de entrada pode enviar a sequência de imagens em teste da área de trabalho do software MATLAB® para a placa de desenvolvimento para ser processada pela FPGA. Com isso, o sistema de saída retorna as imagens binárias ao software MATLAB® para calcular os resultados do sistema de processamento proposto em termos de classificação dos pixels. Essa metodologia de teste facilita a análise dos resultados do sistema de processamento proposto devido ao armazenamento e sincronismo da sequência de imagens serem realizados através do software MATLAB® [40,177,179].

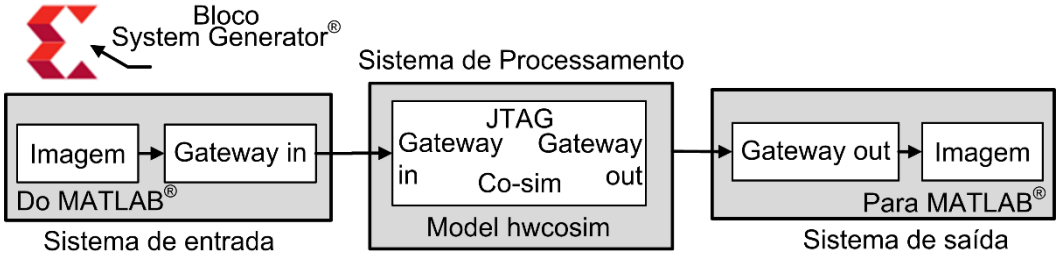


Figura 5.9 - Implementação da co-simulação em hardware [76].

A interface entre os blocos da biblioteca do Simulink® e da Xilinx® é realizada através de portas de entrada (*Gateway in*) e de saída (*Gateway out*). Além disso, essa interface

também realiza a comunicação com a sequência de imagens fornecida pela rotina implementada no software MATLAB® e fornece as imagens processadas para a visualização no software MATLAB® e verificação dos resultados. Essas portas apresentam parâmetros como o tipo de dados e a quantidade de bits, os quais representam os valores dos pixels de entrada.

5.4.4 Análise dos resultados em termos de classificação dos pixels

Através da análise sobre o desempenho de vários métodos de detecção de objetos em movimento apresentados em [118,120] e dos resultados em termos de classificação dos pixels disponíveis em [72–74], alguns métodos para comparação são selecionados. Nessa escolha são considerados os métodos de detecção de objetos em movimento apresentados na Seção 3.2. Vale ressaltar que os resultados apresentados para esses métodos são obtidos através de processadores de uso geral, no entanto, o objetivo não é comparar a implementação em FPGA com GPP, mas posicionar o método proposto no estado da arte em relação à classificação dos pixels e justificá-lo como sendo promissor em FPGA.

Analisando os resultados através da curva de característica de operação do receptor (*Receiver Operating Characteristic* - *ROC*) na Figura 5.10, escolhe-se para a sequência de imagens em teste apresentada na Figura 5.7, o limiar de 0,65 na tomada de decisão fuzzy. Nesse ponto a taxa de detecção de verdadeiro-positivo e a taxa de detecção de falso-positivo obtidas são de 93,84% e 0,41%, respectivamente. Esse limiar é obtido através do índice de *Youden* (*J*) apresentado na equação (5.22) no ponto em que maximiza a relação entre a taxa de detecção de verdadeiro-positivo e de verdadeiro-negativo [180]. O eixo horizontal do gráfico é plotado em escala logarítmica para melhorar a exibição dos valores próximos de zero.

$$J = TPR + TNR - 1 \tag{5.22}$$

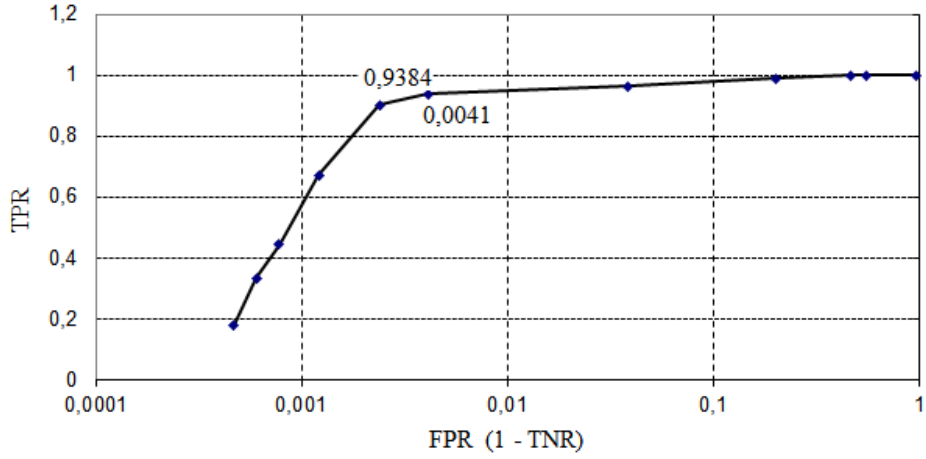


Figura 5.10 - Curva ROC para o sistema de tomada de decisão.

Devido ao desbalanceamento entre a quantidade de pixels pertencentes aos veículos em movimento e a área estática das imagens, a utilização da área sob a curva (*Area Under the Curve - AUC*) ROC apresenta-se como um parâmetro de comparação mais estável para classificar os métodos de detecção de objetos em movimento [181]. Sendo assim, para que trabalhos futuros possam ser comparados através desse parâmetro, apresenta-se para o sistema de tomada de decisão proposto a área sob a curva ROC, que é de 97%. Vale ressaltar que um classificador binário é considerado excelente quando apresenta uma área sob a curva ROC superior a 90% [182].

Na proposta de avaliação apresentada em [72–74], os métodos de detecção de objetos em movimento são classificados através da média aritmética entre a classificação obtida com cada uma das sete medidas definidas na Seção 5.4.1. Como não são disponibilizadas informações suficientes nesses trabalhos para uma análise comparativa através da área sob a curva ROC é utilizado nesta tese o índice de *Youden* para o posicionamento dos métodos. Na Tabela 5.4 são apresentados os resultados das medidas calculadas e o índice de *Youden* para os métodos analisados. Os resultados desses métodos estão organizados em ordem crescente do índice de *Youden*. Todas as medidas são calculadas no software Microsoft Excel® utilizando a classificação dos pixels. Para o método proposto nesta tese e na proposta apresentada em [15–17] é utilizada a classificação dos pixels obtida no software MATLAB®, enquanto que para os demais métodos são utilizados os resultados disponibilizados publicamente pelos trabalhos [72–74].

Tabela 5.4 - Medidas para os métodos analisados.

Referências	*WC*	*TPR*	*TNR*	*FPR*	*FNR*	*Pr*	F_1	*J*
[11]	1,45%	83,05%	99,53%	0,47%	16,95%	91,76%	87,19%	82,58%
[22]	1,43%	83,33%	99,54%	0,46%	16,67%	91,87%	87,39%	82,87%
[21]	1,75%	84,51%	99,12%	0,88%	15,49%	85,79%	85,14%	83,63%
[9]	1,33%	84,52%	99,57%	0,43%	15,48%	92,45%	88,31%	84,09%
[6]	1,13%	89,16%	99,49%	0,51%	10,84%	91,63%	90,38%	88,65%
[15–17]	**1,07%**	**90,50%**	**99,47%**	**0,53%**	**9,50%**	**91,43%**	**90,96%**	**89,97%**
[19,20]	0,90%	91,82%	99,56%	0,44%	8,18%	92,98%	92,40%	91,38%
[8]	0,80%	93,36%	99,57%	0,43%	6,64%	93,18%	93,27%	92,93%
[3]	0,77%	93,79%	99,57%	0,43%	6,21%	93,28%	93,53%	93,36%
Proposto	**0,74%**	**93,84%**	**99,59%**	**0,41%**	**6,16%**	**94,93%**	**94,28%**	**93,43%**
[37]	0,67%	95,23%	99,58%	0,42%	4,77%	93,52%	94,37%	94,81%
[10]	0,66%	95,87%	99,56%	0,44%	4,13%	93,18%	94,51%	95,43%

No sistema de tomada de decisão para detecção de veículos em movimento para FPGA proposto nesta tese e publicado em [183], os resultados dos métodos em termos de classificação dos pixels estão organizados em ordem crescente da medida F_1. A utilização

dessa medida em relação ao índice de *Youden* altera muito pouco o posicionamento dos métodos apresentados na Tabela 5.4, entretanto não são considerados os pixels classificados como verdadeiro-negativos.

A implementação do método proposto em [15–17] é realizada com base apenas na tomada de decisão fuzzy apresentada. Através das medidas apresentadas nas linhas em negrito da Tabela 5.4 verifica-se que o método proposto nesta tese melhora os resultados em termos de classificação dos pixels. Assim, a implementação apenas da tomada de decisão fuzzy proposta em [15–17] apresenta um índice de *Youden* de 89,97%, enquanto que para o sistema proposto, esse índice é de 93,43%.

Observa-se na Tabela 5.4 que os métodos propostos em [10,37] apresentam o índice de *Youden* de 95,43% e 94,81%, respectivamente. Dessa forma, esses métodos são classificados como superiores ao proposto nesta tese. No entanto, esses métodos alcançam uma superioridade máxima de 2% em relação ao método proposto ao custo de uma utilização maior de recursos em termos de memória para armazenar o conjunto de imagens utilizado como referência. A implementação em FPGA desses métodos limitam esse conjunto de imagens a uma quantidade menor de amostras, o que pode comprometer os resultados em termos de classificação dos pixels. De acordo com um estudo comparativo entre vários métodos de detecção de objetos em movimento através da subtração pela imagem de referência apresentado em [184], esses métodos podem ser interessantes em aplicações nas quais a imagem de referência é muito instável e o nível de ruído elevado.

A complexidade computacional dos métodos de detecção de objetos em movimento baseados na modelagem da imagem de referência através da função de probabilidade gaussiana propostos em [6,19,20] é analisada em [185]. Através do método de análise proposto em [185] é observado um tempo de processamento consideravelmente maior que os métodos de detecção de objetos em movimento mais simples apresentados em [9,21,22]. Isso ocorre porque a detecção de objetos em movimento é realizada através da modelagem da imagem de referência utilizando uma mistura de três gaussianas. Todavia, esses métodos quando comparados com os métodos considerados não paramétricos, como o proposto em [3], apresentam vantagens durante o processamento da tomada de decisão. Essas vantagens ocorrem porque parâmetros como a média e o desvio padrão são obtidos por uma função de probabilidade bem definida. Já os métodos não paramétricos apresentam a vantagem de se adaptarem a diferentes distribuições dos pixels, no entanto requerem mais tempo e recursos em termos de memória para obter a imagem de referência [186]. As propostas para detecção

de objetos em movimento apresentadas em [6,19,20] utilizam algoritmos iterativos para classificar os pixels. A implementação desses algoritmos requer que a frequência de operação do sistema de tomada de decisão processe as imagens em tempo real a uma taxa superior a da aquisição.

A utilização de recursos em termos de memória e o tempo de processamento da abordagem proposta em [8] dependem da quantidade dos vetores de peso que modela cada pixel da imagem de referência. Sendo assim, a implementação em FPGA pode comprometer a classificação dos pixels.

5.4.5 Análise dos resultados em termos de recursos utilizados e frequência

Na Tabela 5.5 são apresentados os resultados da implementação do sistema de tomada de decisão total proposto em FPGA *Spartan-6*. Essa FPGA é considerada de baixo custo em relação à FPGA *Virtex-6* também da Xilinx® [187,188]. Nessa tabela também são apresentados o consumo de recursos utilizados na FPGA e a frequência máxima de operação para a implementação individual dos blocos sincronizados de transformação dos padrões de cor (*RGB→YCbCr*, *RGB→HSV*), extração de textura (*LTP*), filtragem (*Med 3x3*, *Med 5x5*, *Open 3x3*), tomada de decisão fuzzy (*DF*) e tomada de decisão clássica (*DC*). Assim, busca-se identificar possíveis pontos críticos no sistema proposto através da análise individual do desempenho de cada bloco. Os resultados gerados pelo software Xilinx® ISE Design Suite 14.7 após o posicionamento e o roteamento na FPGA são apresentados para o sistema de tomada de decisão total processando a sequência de imagens com resolução de 320x240 pixels apresentada na Figura 5.7. Para verificar se o sistema de tomada de decisão proposto processa imagens em alta definição em tempo real são apresentados também os resultados de uma implementação para imagens com resolução de 1366x768 pixels.

As implementações em FPGA dos blocos que realizam as transformações dos padrões de cor e operações de filtragem são obtidas através da biblioteca *XIL XSGImgLib* com arquiteturas apresentadas em [76,189] e distribuída gratuitamente em [67,190,191].

Tabela 5.5 - Consumo de recursos e frequência máxima na FPGA *Spartan-6 LX100*.

Tipo	Disponível	Sistema total 1366 colunas	Sistema total 320 colunas	*RGB YCbCr*	*RGB HSV*	*LTP*	*Med 3x3*	*DF*	*Med 5x5*	*DC*	*Open 3x3*
Registros	126576	19484 (15,39%)	10807 (8,54%)	395	1446	407	360	124	1118	29	90
LUTs	63288	14185 (22,41%)	9910 (15,66%)	861	1138	296	348	247	2042	13	49
Slices	15822	4441 (28,07%)	3237 (20,46%)	331	385	96	103	94	715	9	17
Frequência (MHz)		61,23	63,81	162	154	130	123	184	103	499	325

O sistema de tomada de decisão proposto é capaz de processar um pixel a cada ciclo de relógio após o atraso inicial de 268,71 μs e 61,14 μs para imagens com 1366 colunas e 320 colunas, respectivamente. Para a frequência de operação de 63,81 MHz obtida na FPGA *Spartan-6 LX* é possível processar a sequência de imagens com resolução de 320x240 pixels apresentada na Figura 5.7 em tempo real (830 fps). O padrão de vídeo em alta definição com resolução de 1366x768 pixels requer uma frequência de processamento de pelo menos 63 MHz na frequência de atualização vertical do monitor em 60 Hz para apresentar a sequência de imagens em tempo real [67]. Sendo assim, o sistema de tomada de decisão proposto processa imagens em tempo real até o padrão em alta definição com resolução de 1280x720 pixels. Os resultados da implementação do sistema de tomada de decisão proposto também são verificados na FPGA *Artix-7 A100* da Xilinx®. Essa FPGA é considerada de baixo custo em relação às outras FPGAs *Kintex* e *Virtex* da série 7 da Xilinx® [192]. Os resultados gerados após o posicionamento e o roteamento na FPGA pelo software Xilinx® ISE Design Suite 14.7 são apresentados na Tabela 5.6. Verifica-se através dessa tabela que o sistema proposto consegue processar imagens em alta definição com resolução de 1366x768 pixels com a frequência máxima de operação de 74,86 MHz, sendo possível a tomada de decisão em tempo real. Além disso, é alcançada uma frequência máxima de operação de 77,37 MHz para a sequência de imagens com resolução de 320x240 pixels.

Tabela 5.6 - Consumo de recursos e frequência máxima na FPGA *Artix-7 A100*.

Tipo	Disponível	Sistema total 1366 colunas	Sistema total 320 colunas
Registros	126800	19459 (15,35%)	10774 (8,50%)
LUTs	63400	19658 (31%)	10842 (17,10%)
Slices	15850	5370 (33,88%)	3501 (22,09%)
Frequência (MHz)		74,86	77,37

Na FPGA *Artix-7 A100* o sistema de tomada de decisão proposto é capaz de processar um pixel a cada ciclo de relógio após o atraso inicial de 219,78 μs e 50,42 μs para imagens com 1366 colunas e 320 colunas, respectivamente. Os recursos ocupados considerando a pior situação implementada nas FPGAs em teste não ultrapassam 34%.

6 Conclusões e trabalhos futuros

6.1 Considerações finais

O sistema de tomada de decisão proposto para detecção de veículos em movimento utiliza uma imagem de referência modelada com base na estimativa do valor mediano. Como essa técnica de modelagem da imagem de referência requer o armazenamento de apenas uma única imagem durante o processo de inicialização, utiliza poucos recursos em termos de memória. No entanto, a classificação correta dos pixels depende muito do sistema de tomada de decisão proposto.

Os resultados da fusão da tomada de decisão fuzzy e da tomada de decisão clássica proposta são melhores em termos de classificação dos pixels que a implementação apenas da tomada de decisão fuzzy. Além disso, o sistema proposto apresenta resultados promissores para implementação em FPGA entre os trabalhos analisados. Isso ocorre porque embora o sistema proposto não apresente a melhor classificação dos pixels entre os métodos de detecção de objetos analisados, permite obter medidas próximas de implementações que utilizam mais recursos em termos de memória para modelar a imagem de referência.

O sistema de tomada de decisão proposto apresenta resposta em tempo real processando imagens em alta definição até o padrão de 1280x720 pixels em FPGA de baixo custo *Spartan-6*. Esse sistema também processa em tempo real imagens em alta definição até o padrão 1366x768 pixels em FPGA de baixo custo da série 7. Além disso, a implementação desse sistema permite incorporar novas funcionalidades nessas FPGAs devido à utilização de poucos recursos.

Do exposto, comprova-se a hipótese de que é possível implementar em FPGA de baixo custo um sistema que combine a tomada de decisão fuzzy e clássica para detecção de veículos em movimento melhorando a classificação dos pixels e realizando o sistema de tomada de decisão em tempo real sem utilizar muitos recursos.

6.2 Recomendações para trabalhos futuros

A implementação do sistema de tomada de decisão proposto pode apresentar resultados melhores em termos de classificação dos pixels referentes aos veículos em movimento através da obtenção automática dos parâmetros. Assim, parâmetros como os pesos de cada critério adotados na tomada de decisão fuzzy deverão ser obtidos automaticamente.

Isso ocorre porque a classificação correta dos pixels depende muito da configuração desses parâmetros estar de acordo com a qualidade das imagens capturadas em ambiente sem controle de iluminação.

A técnica de modelagem utilizada para obter a imagem de referência requer poucos recursos em termos de memória. Contudo, devido à possibilidade de variações bruscas da iluminação ao longo do dia, o sistema de tomada de decisão proposto deverá ser combinado com uma técnica de atualização da imagem de referência para melhorar a classificação dos pixels.

APÊNDICE – Lista de publicações

Os artigos resultantes desta pesquisa de doutorado que foram publicados em periódicos e conferência internacionais até o momento são apresentados neste apêndice.

Artigos publicados em periódicos internacionais

- E. Ieno Junior, L. M. Garcés-Socarrás, A. J. Cabrera, e T. C. Pimenta, “Simple generation of threshold for images binarization on FPGA”, Ingeniería e Investigación, vol. 35, no 3, p. 69–75, 2015.
  DOI: 10.15446/ing.investig.v35n3.51750.
- E. Ieno Junior, L. M. Garcés-Socarrás, A. J. Cabrera, e T. C. Pimenta, “Performance analysis of algorithms over FPGA for removing Salt and Pepper noise”, IEEE Latin America Transactions, vol. 14, no 5, p. 2120–2127, 2016.
  DOI: 10.1109/TLA.2016.7530404.
- L. M. Garcés-Socarrás, A. J. C. Sarmiento, S. Sánchez-Solano, P. B. Jiménez, E. Ieno Junior, e T. C. Pimenta, “Modificación automática de arquitecturas de módulos hardware de procesado de imágenes”, Revista de Ingeniería Electrónica, Automática y Comunicaciones (RIELAC), vol. XXXVII, no 3, Havana, p. 21–33, 2016.
- E. Ieno Junior, L. M. Garcés-Socarrás, A. J. Cabrera, e T. C. Pimenta, “FPGA-based EMD Assist for motion detection in critical environments”, IEEE Latin America Transactions, vol. 15, no 10, p. 1856-1863, 2017.
  DOI: 10.1109/10.1109/TLA.2017.8071227.
- E. Ieno Júnior, L. M. Garcés e T. C. Pimenta, “Decision-making system for detection of moving vehicles using a field programmable gate array combining conventional techniques of digital image processing with a fuzzy integral”, Journal of Electronic Imaging, vol. 27, no 4, p. 1-13, 2018.
  DOI: 10.1117/1.JEI.27.4.043001.

Artigo publicado em conferência internacional

- L. M. Garcés-Socarrás, A. J. C. Sarmiento, S. Sánchez-Solano, P. B. Jiménez, E. Ieno Junior, e T. C. Pimenta, “Arquitecturas hardware modificables para bloques de convolución de imágenes sobre FPGA”, XVII Convención y Feira Internacional Informática, Havana, 2018.

Referências

[1] R. C. Gonzalez e R. E. Woods, *Digital Image Processing*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2008.

[2] D. G. Bailey, *Design for Embedded Image Processing on FPGAs*, 1st ed. Solaris South Tower, Singapore: John Wiley & Sons (Asia) Pte Ltd, 2011.

[3] A. Elgammal, D. Harwood, e L. Davis, "Non-parametric model for background subtraction", in *6th European Conference on Computer Vision (ECCV)*, 2000, p. 751–767.

[4] R. Cucchiara, C. Grana, G. Neri, M. Piccardi, e A. Prati, "The Sakbot system for moving object detection and tracking", in *Proceedings of 2nd European Workshop on Advanced Video-Based Surveillance Systems*, 2001, p. 159–162.

[5] K. Kim, T. H. Chalidabhongse, D. Harwood, e L. Davis, "Background modeling and subtraction by codebook construction", in *International Conference on Image Processing (ICIP)*, 2004, p. 3061–3064.

[6] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction", in *17th International Conference on Pattern Recognition (ICPR)*, 2006, p. 28–31.

[7] H. Wang e D. Suter, "Background subtraction based on a robust consensus method", in *18th International Conference on Pattern Recognition (ICPR)*, 2007, p. 223–226.

[8] L. Maddalena e A. Petrosino, "A self-organizing approach to background subtraction for visual surveillance applications", *IEEE Transactions on Image Processing*, vol. 17, nº 7, p. 1168–1177, 2008.

[9] O. Barnich e M. Van Droogenbroeck, "ViBe: a universal background subtraction algorithm for video sequences", *IEEE Transactions on Image Processing*, vol. 20, nº 6, p. 1709–1724, 2011.

[10] M. Hofmann, P. Tiefenbacher, e G. Rigoll, "Background segmentation with feedback: the pixel-based adaptive segmenter", in *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2012, p. 38–43.

[11] I. Haritaoglu, D. Harwood, e L. S. Davis, "W4: real-time surveillance of people and their activities", *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 22, nº 8, p. 809–830, 2000.

[12] T. Horprasert, D. Harwood, e L. S. Davis, "A statistical approach for real-time robust background subtraction and shadow detection", in *7th IEEE International Conference on Computer Vision (ICCV)*, 1999, p. 1–19.

[13] R. Cucchiara, C. Grana, M. Piccardi, A. Prati, e S. Sirotti, "Improving shadow suppression in moving object detection with HSV color information", in *Intelligent Transportation Systems*, 2001, p. 334–339.

[14] R. Cucchiara, C. Grana, M. Piccardi, e A. Prati, "Detecting moving objects, ghosts and shadows in video streams", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, nº 10, p. 1337–1342, 2003.

[15] F. El Baf, T. Bouwmans, e B. Vachon, "Fuzzy integral for moving object detection", in *IEEE World Congress on Computational Intelligence*, 2008, vol. 1, nº 1, p. 1729–1736.

[16] F. El Baf, T. Bouwmans, e B. Vachon, “A fuzzy approach for background subtraction”, in *15th IEEE International Conference on Image Processing (ICIP)*, 2008, p. 2648–2651.

[17] F. El Baf, T. Bouwmans, e B. Vachon, “Foreground detection using the Choquet integral”, in *9th International Workshop on Image Analysis for Multimedia Interactive Services(WIAMIS)*, 2008, p. 187–190.

[18] X. Lu, T. Izumi, T. Takahashi, e L. Wang, “Moving vehicle detection based on fuzzy background subtraction”, in *IEEE International Conference on Fuzzy Systems*, 2014, p. 529–532.

[19] C. Stauffer e W. E. L. Grimson, “Adaptive background mixture models for real-time tracking”, in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1999, p. 246–252.

[20] C. Stauffer e W. E. L. Grimson, “Learning patterns of activity using real-time tracking”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, nº 8, p. 747–757, 2000.

[21] J. Zheng, Y. Wang, N. L. Nihan, e M. E. Hallenbeck, “Extracting roadway background image: a mode-based approach”, *Transportation Research Record Journal of the Transportation Research Board*, vol. 1944, nº 1, p. 82–88, 2006.

[22] S. Yoshinaga, A. Shimada, H. Nagahara, e R. Taniguchi, “Background model based on intensity change similarity among pixels”, in *The 19th Korea-Japan Workshop on Frontiers of Computer Vision*, 2013, p. 276–280.

[23] Y. Yang, Q. Zhang, P. Wang, X. Hu, e N. Wu, “Moving object detection for dynamic background scenes based on spatiotemporal model”, *Advances in Multimedia, Hindawi Publishing Corporation*, p. 1–9, 2017.

[24] Y. Zhang, G. Li, X. Xie, e Z. Wang, “A new algorithm for fast and accurate moving object detection based on motion segmentation by clustering”, in *2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, 2017, p. 444–447.

[25] S. Li, P. Liu, e G. Han, “Moving object detection based on Codebook algorithm and Three-frame Difference”, *International Journal of Signal Processing, Image Processing and Pattern Recognition (IJSIP)*, vol. 10, nº 3, p. 23–32, 2017.

[26] M. Prakasha, B. R. Puneeth, e C. Kumar, “Detection and tracking of moving objects using image processing”, *International Journal of Engineering Science Invention & Development*, vol. IV, nº 1, p. 28–39, 2017.

[27] N. S. Ghedia, C. H. Vithalani, e A. Kothari, “Performance evaluation of moving object detection and tracking algorithm for outdoor surveillance using modified GMM”, *International Journal of Engineering Sciences & Research Technology (IJESRT)*, vol. 6, nº 7, p. 215–223, 2017.

[28] R. Chavan e S. R. Gengaje, “Multiple object detection using GMM technique and tracking using Kalman filter”, *International Journal of Computer Applications*, vol. 172, nº 3, p. 20–25, 2017.

[29] A. Keivani, J.-R. Tapamo, e F. Ghayoor, “Motion-based moving object detection and tracking using automatic K-means”, in *IEEE Africon*, 2017, p. 32–37.

[30] S. Singh Sengar e S. Mukhopadhyay, “Moving object detection based on frame difference and W4”, *Signal, Image and Video Processing (SIViP)*, vol. 11, nº 7, p.

1357–1364, 2017.

[31] A. P. Athira, M. Vijayan, e R. Mohan, “Moving object detection using Local Binary Pattern and Gaussian background model”, in *Industry Interactive Innovations in Science, Engineering and Technology. Lecture Notes in Networks and Systems*, vol. 11, Springer, Singapore, 2018, p. 367–376.

[32] K. P. Risha e A. C. Kumar, “Novel method of detecting moving object in video”, *Procedia Technology*, vol. 24, p. 1055–1060, 2016.

[33] J. Dey e N. Praveen, “Moving object detection using genetic algorithm for traffic surveillance”, in *International Conference on Electrical, Electronics and Optimization Techniques (ICEEOT)*, 2016, p. 2289–2293.

[34] P. S. moving object detection and tracking for video surveillanceital Manohar e A. B. Pawar, “Advanced moving object detection and tracking for video surveillance”, *International Journal on Recent and innovation Trends in Computing and Communication*, vol. 5, nº 3, p. 478–483, 2017.

[35] N. Singla, “Motion detection based on frame difference method”, *International Journal of Information & Computation Technology*, vol. 4, nº 15, p. 1559–1565, 2014.

[36] K. Kalirajan e M. Sudha, “Moving object detection for video surveillance”, *The Scientific World Journal, Hindawi Publishing Corporation*, p. 1–10, 2015.

[37] P.-L. St-Charles, G.-A. Bilodeau, e R. Bergevin, “SuBSENSE: a universal change detection method”, *IEEE Transactions on Image Processing*, vol. 24, nº 1, p. 359–373, 2015.

[38] D. G. Bailey e C. T. Johnston, “Algorithm transformation for FPGA implementation”, in *IEEE International Symposium on Electronic Design, Test & Applications*, 2010, p. 77–81.

[39] D. G. Bailey, “Adapting algorithms for hardware implementation”, in *EEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, p. 177–184.

[40] F. Hamdaoui, A. Khalifa, A. Sakly, e A. Mtibaa, “Real time implementation of medical images segmentation based on PSO”, in *International Conference on Control, Decision and Information Technologies (CoDIT)*, 2013, p. 36–42.

[41] T. Jeyaseelan e G. Priyanga, “FPGA implementation of medical image fusion based on NSCT”, *International Journal of Advanced Research Trends in Engineering and Technology*, vol. II, nº I, p. 185–192, 2015.

[42] G. S. Bhargavi, B. P. Kumar, e T. Kalyan, “Fast background subtraction algorithm for moving object detection & tracking in FPGA”, *International Journal of Software & Hardware Research in Engineering*, vol. 2, nº 6, p. 65–70, 2014.

[43] D. Paul e R. Surendran, “Implementation of visual object tracking by adaptive background subtraction on FPGA”, *International Journal of Engineering Research & Technology (IJERT)*, vol. 4, nº 2, p. 133–138, 2015.

[44] T. Kryjak, M. Komorkiewicz, e M. Gorgon, “Real-time foreground object detection combining the PBAS background modelling algorithm and feedback from scene analysis module”, *International Journal of Electronics and Telecommunications*, vol. 60, nº 1, p. 61–72, jan. 2014.

[45] M. Salve, M. Rajput, e S. Shingate, “FPGA based moving object detection algorithm implementation for traffic surveillance”, *Journal of Engineering Research and Applications*, vol. 3, nº 5, p. 1829–1832, 2013.

[46] S. Hema e V. Selvakumar, “Material fault detection using FPGA for Nitrile Butadience Rubber float”, *International Journal of Innovative Research in Computer and Communication Enginering*, vol. 2, nº 11, p. 6987–6993, 2014.

[47] K. Sehairi, C. Benbouchama, e F. Chouireb, “A real time implementation on FPGA of moving objects detection and classification”, *International Journal of Circuits, Systems and Signal Processing*, vol. 9, p. 160–167, 2015.

[48] M. A. Altuncu, T. Guven, Y. Becerikli, e S. Sahin, “Real-time system implementation for image processing with hardware/software co-design on the Xilinx Zynq platform”, *International Journal of Information and Electronics Engineering*, vol. 5, nº 6, p. 473–477, 2015.

[49] M. Prakash, K. Vijaipriya, e S. Nandhini, “FPGA implementation of mass public transit facility for smart security system”, *International Journal of Research in Engineering and Technology (IJRET)*, vol. 4, nº 4, p. 476–481, 2015.

[50] I. Bouganssa, M. Sbihi, e M. Zaim, “Implementation on a FPGA of edge detection algorithm in medical image and tumors Characterization”, in *5th International Conference on Multimedia Computing and Systems (ICMCS)*, 2016, p. 1–6.

[51] L. Jinghong, T. Yanan, e X. Xiujian, “Design and implementation of bus lane video image monitor system based on FPGA”, in *29th Chinese Control And Decision Conference (CCDC)*, 2017, nº 1, p. 4868–4872.

[52] T. M. Khan, D. G. Bailey, M. A. U. Khan, e Y. Kong, “Efficient hardware implementation for fingerprint image enhancement using anisotropic Gaussian filter”, *IEEE Transactions on Image Processing*, vol. 26, nº 5, p. 2116–2126, 2017.

[53] J. Y. Byun e J. W. Jeon, “FPGA based face detection using local ternary pattern under variant illumination condition”, in *Advances in Computer Science and Ubiquitous Computing, Lecture Notes in Electrical Engineering*, vol. 474, Springer, Singapore, 2017, p. 335–370.

[54] E. Ieno Junior, L. M. Garces, A. J. Cabrera, e T. C. Pimenta, “FPGA-based EMD block assist for motion detection in critical environments”, *IEEE Latin America Transactions*, vol. 15, nº 10, p. 1856–1863, 2017.

[55] A. Kondane e B. Patil, “Hardware solution to motion object detection using morphological filtering and FPGA”, *International Journal of Engineering Research and Applications (IJERA)*, vol. 4, nº 4, p. 56–60, 2014.

[56] W. J. MacLean, “An evaluation of the suitability of FPGAs for embedded vision systems”, in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, p. 131–138.

[57] M. A. Veer e P. R. U. Shekokar, “A survey on FPGA implementation of object detection and different techniques and methods in the field of Computer Vision”, *International Journal of Innovative Research in Technology (IJIRT)*, vol. 1, nº 12, p. 138–142, 2015.

[58] S. L. Anusha e C. H. Sowmya, “Review of an efficient object detection using System Generator”, *International Journal of Research & Development Organization (IJRDO)*, vol. 2, nº 4, p. 1–5, 2015.

[59] S. A. Inigo e P. Suresh, “General study on moving object segmentation methods for video”, *International Journal of Advanced Research in Computer Engineering & Tecnology (IJARCET)*, vol. 1, nº 8, 2012.

[60] P. D. Mahamuni, R. P. Patil, e H. S. Thakar, “Moving object detection using background subtraction algorithm using Simulink”, *International Journal of Research in Engineering and Technology (IJRET)*, vol. 3, nº 6, p. 594–598, 2014.

[61] A. Toledo Moreo, J. Suardíaz Muro, S. Cuenca-Asensi, e A. Grediaga, “Novel simulink blockset for image processing codesign”, in *IEEE Mediterranean Electrotechnical Conference (MELECON 2006), Málaga*, 2006, p. 117–120.

[62] Altera, “Video and image processing design using FPGAs”, *Altera White Papers*, nº 1.1, p. 1–6, 2007.

[63] U. Sehgal, “Edge detection technique in digital image processing using fuzzy logic”, *International Journal of Advances in Engineering Research (IJAER)*, vol. 3, nº 1, p. 1–5, 2012.

[64] L. Shapiro e G. Stockman, *Computer Vision*. Upper Saddle River, New Jersey, USA: Pearson Education Inc. - Prentice Hall, 2001.

[65] D. A. Forsyth e J. Once, *Computer Vision: A Modern Approach*, 2nd ed. Upper Saddle River, New Jersey, USA: Pearson Education Inc. - Prentice Hall, 2012.

[66] R. C. Gonzalez, R. E. Woods, e S. L. Eddins, *Digital Image Processing Using Matlab*, 2nd ed. USA: Gatesmark Publishing, 2009.

[67] L. M. Garcés-Socarrás, “Biblioteca basada en modelos de módulos hardware configurables para procesado de imágenes y vídeos”, Tese (Doutorado em Ciências Técnicas), Universidad Tecnológica de La Habana “José Antonio Echeverría”, Havana-Cuba, 2016.

[68] A. S. Mohan e R. R., “Video image processing for moving object detection and segmentation using background subtraction”, in *First International Conference on Computational Systems and Communications (ICCSC)*, 2014, nº 12, p. 288–292.

[69] B. Jähne, H. HauBecker, e P. GeiBler, “Signal Processing and Pattern Recognition”, in *Handbook of Computer Vision and Applications*, Vol. 2., San Jose, CA, USA: Academic Press, 1999, p. 684–722.

[70] S. K. Pal, “Fuzzy image processing and recognition: uncertainty handling and applications”, *International Journal of Image and Graphics*, vol. 1, nº 2, p. 169–195, 2001.

[71] C. V. Jawahar e A. K. Ray, “Fuzzy statistics of digital images”, *IEEE Signal Processing Letters*, vol. 3, nº 8, p. 225–227, 1996.

[72] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, e P. Ishwar, “Changedetection.net: a new change detection benchmark dataset”, in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2012, p. 1–8.

[73] Y. Wang, P. Jodoin, P. Fatih, K. Janusz, Y. Benezeth, e B. Prakash, “CDnet 2014: an expanded change detection benchmark dataset”, in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2014, p. 387–394.

[74] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, e P. Ishwar, “A novel video dataset for change detection benchmarking”, *IEEE Transactions on Image Processing*, vol. 23, nº

11, p. 4663–4679, 2014.

[75] B. Fan e X. Liu, “Foreground detection based on color and texture features”, in *International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, 2015, p. 1940–1944.

[76] L. M. Garcés-Socarrás, S. Sánchez-Solano, P. B. Jiménez, e A. J. C. Sarmiento, “Library for model-based design of image processing algorithms on FPGAs”, *Revista de la Facultad de Ingeniería Universidad Antioquia*, vol. 68, nº 9, p. 36–47, set. 2013.

[77] G. T. Shrivakshan, “A comparison of various edge detection techniques used in image processing”, *International Journal of Computer Science Issues (IJCSI)*, vol. 9, nº 5, p. 269–276, 2012.

[78] A. Toledo, C. Vicente, J. Suardíaz, e S. Cuenca, “Xilinx System Generator based HW components for rapid prototyping of computer vision SW/HW systems”, in *Second Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, 2005, vol. 3522, p. 667–674.

[79] E. Ieno Junior, L. M. Garcés, A. J. Cabrera, e T. C. Pimenta, “Simple generation of threshold for images binarization on FPGA”, *Ingeniería e Investigación*, vol. 35, nº 3, p. 69–75, 2015.

[80] J. M. Blackledge, *Digital Image Processing: Mathematical and Computational Methods*, 1st ed. London: Horwood, 2005.

[81] S. K. Areefabegam e T. Narendrakumar, “FPGA based design and implementation of image edge detection using Xilinx System Generator”, *International Journal of New Trends in Electronics and Comunication*, vol. 2, nº 8, p. 18–21, 2014.

[82] L. M. Garcés-Socarrás, A. J. Cabrera Sarmiento, S. Sánchez-Solano, e P. Brox Jiménez, “Diseño de bloques de convolución para procesado de imágenes con FPGA”, *Revista de Ingeniería Electrónica, Automática y Comunicaciones (RIELAC)*, vol. XXXII, nº 3, La Habana, p. 56–69, 2011.

[83] U. I. Patel e H. Patel, “A review on edge detection techniques based on FPGA”, *International Journal of Modern Trends in Engineering and Research (IJMTER)*, vol. 2, nº 3, p. 12–18, 2015.

[84] S. Ozturk e B. Akdemir, “Comparison of edge detection algorithms for texture analysis on glass production”, in *World Conference on Technology, Innovation and Entrepreneurship*, 2015, vol. 195, p. 2675–2682.

[85] J. Stauder, R. Mech, e O. Jörn, “Detection of moving cast shadows for object segmentation”, *IEEE Transactions on Multimedia*, vol. 1, nº 1, p. 65–76, 1999.

[86] C. Chakrabarti, “High sample rate array architectures for median filters”, *IEEE Transactions on Signal Processing*, vol. 42, nº 3, p. 707–712, 1994.

[87] G. Chinnasamy e M. Gowtham, “Performance comparison of various filters for removing salt & pepper noise”, *International Journal of Multidisciplinary Research and Development*, vol. 2, nº 1, p. 152–155, 2015.

[88] E. Ieno Junior, L. M. Garces, A. J. Cabrera, e T. C. Pimenta, “Performance analysis of algorithms over FPGA for removing Salt and Pepper noise”, *IEEE Latin America Transactions*, vol. 14, nº 5, p. 2120–2127, 2016.

[89] T. Singh, S. Sanju, e B. Vijay, “A new algorithm designing for detection of moving objects in video”, *International Journal of Computer Applications*, vol. 96, nº 2, p. 4–

11, 2014.

[90] H.-Y. Chen e Y.-K. Wang, “Hardware design of moving object detection on reconfigurable system”, *Journal of Computer and Communications*, vol. 4, nº 8, p. 30–43, 2016.

[91] J. C. Tello, “La visión artificial y las operaciones morfológicas en imágenes binarias”, in *Campus Multidisciplinar en Percepción e Inteligencia (CMPI)*, 2006, p. 1–7.

[92] W. J. Torres e R. J. Bello, “Procesamiento de imágenes a color utilizando morfología matemática”, *Sistemas, Cibernética e Informática*, vol. 3, nº 1, p. 79–83, 2006.

[93] A. M. Raid, W. M. Khedr, M. A. El-dosuky, e M. Aoud, “Image restoration based on morphological operations”, *International Journal of Computer Science, Engineering and Information Technology (IJCSEIT)*, vol. 4, nº 3, p. 9–21, 2014.

[94] M. C. Pawaskar, N. S. Narkhede, e S. S. Athalye, “Detection of moving object based on background subtraction”, *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, vol. 3, nº 3, p. 215–218, 2014.

[95] T. Kumar e K. Verma, “A theory based on conversion of RGB image to gray image”, *International Journal of Computer Applications*, vol. 7, nº 2, p. 8–11, 2010.

[96] C. Kanan e G. W. Cottrell, “Color-to-grayscale: does the method matter in image recognition?”, *PLoS ONE*, vol. 7, nº 1, p. 1–7, 2012.

[97] A. A. Hassan, “Coloring of gray-scale image using FPGA”, *Journal of Engineering*, vol. 16, nº 4, p. 5932–5945, 2010.

[98] G. Jeon, “Measuring and comparison of edge detectors in color spaces”, *International Journal of Control and Automation (IJCA)*, vol. 6, nº 5, p. 21–30, 2013.

[99] Y. Said, T. Saidani, e M. Atri, “FPGA-based architectures for image processing using high-level design”, *WSEAS Transactions on Signal Processing*, vol. 11, p. 38–44, 2015.

[100] N. A. Ibraheem, M. M. Hasan, R. Z. Khan, e P. K. Mishra, “Understanding color models: a review”, *ARPN Journal of Science and Technology*, vol. 2, nº 3, p. 265–275, 2012.

[101] Y. Hsu, M. Hsiao-Chun, e T. Ching-Chih, “FPGA implementation of a real-time image tracking system”, in *SICE Annual Conference*, 2010, p. 2878–2884.

[102] L. Touil, A. Ben Abdelali, e A. Mtibaa, “Design and implementation of an RGB to HMMD color conversion module on FPGA”, in *1st International Conference on Information and Communication Technologies (ICTIA)*, 2014, p. 1–7.

[103] P. Chiranjeevi e S. Sengupta, “Interval-valued model level fuzzy subtraction”, *IEEE Transactions on Cybernetics*, vol. PP, nº 99, p. 1–12, 2016.

[104] T. Ojala, M. Pietikäinen, e D. Harwood, “A comparative study of texture measures with classification based on feature distributions”, *Pattern Recognition*, vol. 29, nº l, p. 51–59, 1996.

[105] T. Ojala, M. Pietikainen, e T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, nº 7, p. 971–987, 2002.

[106] X. Tan e B. Triggs, “Enhanced local texture feature sets for face recognition under difficult lighting conditions”, in *International Workshop on Analysis and Modeling of*

Faces and Gestures, 2007, p. 168–182.

[107] G.-A. Bilodeau, J.-P. Jodoin, e N. Saunier, "Change detection in feature space using local binary similarity patterns", in *International Conference on Computer and Robot Vision (CRV)*, 2013, p. 106–112.

[108] P. L. St-Charles e G. A. Bilodeau, "Improving background subtraction using local binary similarity patterns", in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014, p. 509–515.

[109] S. Liao, G. Zhao, V. Kellokumpu, M. Pietikainen, e S. Z. Li, "Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010, p. 1301–1306.

[110] R. Qin, S. Liao, Z. Lei, e S. Z. Li, "Moving cast shadow removal based on local descriptors", in *International Conference on Pattern Recognition*, 2010, p. 10–13.

[111] P. Liu e Y. Zhu, "An adaptive cast shadow detection with combined texture and color models", *International Journal of Future Computer and Communication*, vol. 3, nº 2, p. 113–118, 2014.

[112] F. Li, X. Zhang, C. Zhao, e M. Yan, "Vehicle detection research based on USILTP operator", in *International Conference on Engineering Technology and Application (ICETA)*, 2015, vol. 8, p. 6.

[113] R. O. Duda, P. E. Hart, e D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley-Interscience, 2001.

[114] A. A. Goshtasby, "Similarity and Dissimilarity Measures", in *Image Registration: Principles, Tools and Methods*, London, U.K.: Springer, 2012, p. 7–66.

[115] A. Singh, A. Yadav, e A. Rana, "K-means with three different distance metrics", *International Journal of Computer Applications*, vol. 67, nº 10, p. 13–17, 2013.

[116] S. R. Balaji e S. Karthikeyan, "A survey on moving object tracking using image processing", in *11th International Conference on Intelligent Systems and Control (ISCO)*, 2017, p. 469–474.

[117] K. Toyama, J. Krumm, B. Brumitt, e B. Meyers, "Wallflower: principles and practice of background maintenance", in *International Conference on Computer Vision*, 1999, nº September, p. 255–261.

[118] T. Bouwmans, "Traditional and recent approaches in background modeling for foreground detection: an overview", *Computer Science Review*, vol. 11, nº 12, p. 31–66, 2014.

[119] S. Padmavathi e A. Kumar, "A survey on algorithms of shadow removal in vehicle detection", *International Journal of Computer Technology & Applications*, vol. 5, nº 2, p. 518–521, 2014.

[120] Y. Xu, J. Dong, B. Zhang, e D. Xu, "Background modeling methods in video analysis: a review and comparative evaluation", *CAAI Transactions on Intelligence Technology*, vol. 1, nº 1, p. 43–60, 2016.

[121] S. Jeeva e M. Sivabalakrishnan, "Survey on background modeling and foreground detection for real time video surveillance", in *2nd International Symposium on Big Data and Cloud Computing (ISBCC)*, 2015, vol. 50, p. 566–571.

[122] L. Maddalena e A. Petrosino, “Towards benchmarking scene background initialization”, in *International Conference on Image Analysis and Processing (ICIAP)*, 2015, p. 469–476.

[123] T. Bouwmans, L. Maddalena, e A. Petrosino, “Scene background initialization: a taxonomy”, *Pattern Recognition Letters*, p. 1–10, 2016.

[124] S. S. Cheung e C. Kamath, “Robust Techniques for Background Subtraction in Urban Traffic Video”, in *Symposium on Electronic Imaging*, 2004, p. 1–14.

[125] N. J. B. Mcfarlane e C. P. Schofield, “Segmentation and tracking of piglets in images”, *Machine Vision and Applications*, vol. 8, p. 187–193, 1995.

[126] J. Hiraiwa, E. Vargas, e S. Toral, “An FPGA based embedded vision system for real-time motion segmentation”, in *17th International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2010, vol. 1, nº 3, p. 360–363.

[127] A. Manzanera e J. C. Richefeu, “A new motion detection algorithm based on Sigma-Delta background estimation”, *Pattern Recognition Letters*, vol. 28, p. 320–328, 2007.

[128] L. Lacassagne e A. Manzanera, “Motion detection: fast and robust algorithms for embedded systems”, in *16th IEEE International Conference on Image Processing (ICIP)*, 2009, p. 3265–3268.

[129] M. Vargas, J. M. Milla, S. L. Toral, S. Member, e F. Barrero, “An enhanced background estimation algorithm for vehicle detection in urban traffic scenes”, *IEEE Transactions on Vehicular Technology*, vol. 59, nº 8, p. 3694–3709, 2010.

[130] R. T. Collins, A. J. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, O. Hasegawa, P. Burt, e L. Wixson, “A system for video surveillance and monitoring”. Relatório Técnico, Pittsburgh, PA, p. 1–68, 2000.

[131] L. M. Fuentes e S. A. Velastin, “Vigilancia Avanzada : del tracking a la detección de sucesos”, *IEEE Latin America Transactions*, vol. 2, nº 3, p. 206–211, 2004.

[132] R. Cucchiara, C. Grana, M. Piccardi, e A. Prati, “Detecting objects, shadows and ghosts in video streams by exploiting color and motion information”, in *11th International Conference on Image Analysis and Processing*, 2001, p. 360–365.

[133] A. A. Nascimento, P. C. M. A. Farias, D. M. Matos, e J. L. C. Carvalho, “Cálculo de distâncias euclidianas entre histogramas para sistemas de localização robótica em FPGA”, in *Anais do XX Congresso Brasileiro de Automática*, 2014, p. 93–98.

[134] R. Yan, J. Zhang, J. Yang, e A. G. Hauptmann, “A discriminative learning framework with pairwise constraints for video object classification”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, nº 4, p. 578–593, 2006.

[135] A. S. Philip, “Background subtraction algorithm for moving object detection using denoising architecture in FPGA”, *International Journal of Science and Research (IJSR)*, vol. 2, nº 8, p. 151–157, 2013.

[136] H. Zhang e H. Zhang, “A moving target detection algorithm based on dynamic scenes”, in *8th International Conference on Computer Science & Education (ICCSE)*, 2013, p. 995–998.

[137] S. Alex e A. Wahi, “BSFD: background subtraction frame difference algorithm for moving object”, *Journal of Theoretical and Applied Information Technology (JATIT)*, vol. 60, nº 3, p. 623–628, 2014.

[138] U. Bidarte, J. L. Martin, A. Zuloaga, e J. Ezquerra, “Adaptive image brightness and contrast enhancement circuit for real-time vision systems”, in *IEEE International Conference on Industrial Technology (ICIT)*, 2000, vol. 1, p. 421–426.

[139] P. Jodoin, M. Mignotte, e J. Konrad, “Statistical background subtraction using spatial cues”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, nº 12, p. 1758–1763, 2007.

[140] L. A. Zadeh, “Fuzzy sets”, *Information and Control*, vol. 8, nº 3, p. 338–353, 1965.

[141] C. Wu e C. Wu, “The supremum and infimum of the set of Fuzzy numbers and its application”, *Journal of Mathematical Analysis and Applications*, vol. 210, nº 2, p. 499–511, 1997.

[142] H. J. Zimmermann, *Fuzzy Set Theory and Its Applications*, 3rd ed. Massachusetts: Kluwer Academic Publishers, 1996.

[143] A. Motro e P. Smets, “Imperfect Information: Imprecision - Incertainty”, in *Uncertainty Management in Information Systems: From Needs to Solutions*, Springer Science & Business Media, 2012, p. 225–254.

[144] J. R. Jang, “ANFIS: Adaptive-Network-Based Fuzzy Inference System”, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, nº 3, p. 665–685, 1993.

[145] E. H. Mamdani, “Application of fuzzy algorithms for control of simple dynamic plant”, *Proceedings of the Institution of Electrical Engineers*, vol. 121, nº 12, p. 1585–1588, 1974.

[146] E. H. Mamdani e S. Assilian, “An experiment in linguistic synthesis with a fuzzy logic controller”, *International Journal Man-Machine Studies*, vol. 1, nº 7, p. 1–13, 1975.

[147] T. Takagi e M. Sugeno, “Fuzzy identification of systems and its applications to modeling and control”, *IEEE Transactions on Systems, Man and Cybernetics*, vol. SMC-15, nº 1, p. 116–132, 1985.

[148] K. Ishii e M. Sugeno, “A model of human evaluation process using fuzzy measure”, *International Journal Man-Machine Studies*, vol. 1, nº 22, p. 19–38, 1985.

[149] H. Tahani e J. M. Keller, “Information fusion in computer vision using the fuzzy integral”, *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, nº 3, p. 733–741, 1990.

[150] M. Grabisch, “The application of fuzzy integrals in multicriteria decision making”, *European Journal of Operational Research*, vol. 89, nº 3, p. 445–456, 1996.

[151] M. Grabisch, “Fuzzy Integral for Classification and Feature Extraction”, in *Fuzzy Measures and Integrals: Theory and Applications*, 1 st., M. Grabisch, T. Murofushi, e M. Sugeno, Orgs. Heidelberg, Ger: Physica Verlag, 2000, p. 415–434.

[152] M. Grabisch, S. A. Orlovski, e R. R. Yager, “Fuzzy Aggregation of Numerical Preferences”, in *Fuzzy Sets in Decision Analysis, Operations Research and Statistics*, Norwell, MA, USA: Kluwer Academic, 1998, p. 31–68.

[153] Y. Narukawa e T. Murofushi, “Decision modelling using the Choquet integral”, in *Modeling Decisions for Artificial Intelligence (MDAI)*, 2004, nº 7, p. 183–193.

[154] M. Sugeno, “Theory of fuzzy integrals and its applications”, Tese (Doutorado em Engenharia), Tokyo, Japan, 1974.

[155] G. Choquet, “Theory of capacities”, *Annales de l’institut Fourier*, vol. 5, p. 131–295,

1954.

[156] Q. Zhang, R. Mesiar, J. Li, e P. Struk, “Generalized Lebesgue integral”, *International Journal of Approximate Reasoning*, vol. 52, nº 3, p. 427–443, 2011.

[157] C.-H. Cheng, C.-T. Chen, e S.-F. Huang, “Combining fuzzy integral with order weight average (OWA) method for evaluating financial performance in the semiconductor industry”, *African Journal of Business Management*, vol. 6, nº 21, p. 6358–6368, 2012.

[158] J. Marichal, “An axiomatic approach of the discrete choquet integral as a tool to aggregate interacting criteria”, *IEEE Transactions on Fuzzy Systems*, vol. 8, nº 6, p. 800–807, 2000.

[159] R. Mesiar, A. Stupnanová, e R. R. Yager, “Generalizations of OWA operators”, *IEEE Transactions on Fuzzy Systems*, vol. 23, nº 6, p. 2154–2162, 2015.

[160] A. K. Jain, M. N. Murty, e P. J. Flynn, “Data clustering: a review”, *Journal ACM Computing Surveys*, vol. 31, nº 3, p. 264–323, 1999.

[161] R. Linden, “Técnicas de Agrupamento”, *Revista de Sistemas de Informação da FSMA*, vol. 4, p. 18–36, 2009.

[162] S. Ghosh e S. K. Dubey, “Comparative analysis of K-means and Fuzzy C-means algorithms”, *International Journal of Advanced Computer Science and Applications*, vol. 4, nº 4, p. 35–39, 2013.

[163] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, 1st ed. Springer US, 1981.

[164] J. C. Dunn, “A Fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters”, *Cybernetics and Systems*, vol. 36, nº 7, p. 32–57, 2010.

[165] A. K. Jain, “Data clustering: 50 years beyond K-means”, *Pattern Recognition Letters*, vol. 31, nº 8, p. 651–666, 2010.

[166] S. Das, “Pattern recognition using the fuzzy C-means technique”, *International Journal of Energy, Information and Communications*, vol. 4, nº 1, p. 1–14, 2013.

[167] O. Barnich e V. M. Droogenbroeck, “VIBE: a powerful random technique to estimate the background in video sequences”, in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, p. 945–948.

[168] M. Van Droogenbroeck e O. Paquot, “Background subtraction: experiments and improvements for ViBe”, in *Change Detection Workshop (CDW)*, 2012, nº June, p. 1–6.

[169] T. Kryjak e M. Gorgon, “Real-time implementation of the ViBe foreground object segmentation algorithm”, in *Federated Conference on Computer Science Information Systems (FedCSIS)*, 2013, nº 1, p. 591–596.

[170] L. Li, W. Huang, I. Y. Gu, e Q. Tian, “Statistical modeling of complex backgrounds for foreground object detection”, *IEEE Transactions on Imagem Processing*, vol. 13, nº 11, p. 1459–1472, 2004.

[171] M. M. Azab, H. A. Shedeed, e A. S. Hussein, “A new technique for background modeling and subtraction for motion detection in real-time videos”, in *IEEE International Conference on Image Processing (ICIP)*, 2010, p. 3453–3456.

[172] F. Bashir e F. Porikli, “Performance evaluation of object detection and tracking

systems", in *Proceedings 9th IEEE International Workshop on PETS*, 2006, p. 7–14.

[173] J. Davis e M. Goadrich, "The relationship between precision-recall and ROC curves", in *Proceedings of 23rd International Conference on Machine Learning*, 2006, p. 233–240.

[174] S. Kumar e J. Sen Yadav, "Segmentation of moving objects using background subtraction method in complex environments", *Radioengineering*, vol. 25, nº 2, p. 399–408, 2016.

[175] C. Vicente-Chicote, A. Toledo Moreo, e P. Sánchez-Palma, "Image processing application development: from rapid prototyping to SW/HW co-simulation and automated code generation", in *Second Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, 2005, vol. 3522, p. 659–666.

[176] A. Toledo Moreo, P. Navarro Lorente, F. Soto Valles, J. Suardíaz Muro, e C. Fernández Andrés, "Experiences on developing computer vision hardware algorithms using Xilinx System Generator", *Microprocessors and Microsystems*, vol. 29, nº 8–9, p. 411–419, 2005.

[177] T. Saidani, D. Dia, W. Elhamzi, M. Atri, e R. Tourki, "Hardware co-simulation for video processing using Xilinx System Generator", in *Proceedings of the Word Congress on Engineering*, 2009, vol. I, p. 3–7.

[178] J. Albaladejo, D. de Andrés, L. Lemus, e J. Salvi, "Codesign methodology for computer vision applications", *Microprocessors and Microsystems*, vol. 28, nº 5–6, p. 303–316, 2004.

[179] N. P. Raut e A. V Gokhale, "FPGA implementation for image processing algorithms using Xilinx System Generator", *IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)*, vol. 2, nº 4, p. 26–36, 2013.

[180] E. F. Schisterman, N. J. Perkins, A. Liu, e H. Bondell, "Optimal cut-point and its corresponding Youden index to discriminate individuals using pooled blood samples", *Epidemiology*, vol. 16, nº 1, p. 73–81, 2005.

[181] L. A. Jeni, J. F. Cohn, e F. De La Torre, "Facing imbalanced data: recommendations for the use of performance metrics", in *Humaine Association Conference on Affective Computing Intelligent Interaction*, 2013, p. 245–251.

[182] J. Devi e N. Sehgal, "A technique for improving software quality using support vector machine", *International Journal of Computer Sciences and Engineering*, vol. 5, nº 6, p. 100–105, 2017.

[183] E. Ieno Jr., L. M. G. Socarrás, e T. C. Pimenta, "Decision-making system for detection of moving vehicles using a field programmable gate array combining conventional techniques of digital image processing with a fuzzy integral", *Journal of Electronic Imaging*, vol. 27, nº 4, p. 1–13, 2018.

[184] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, e C. Rosenberger, "Comparative study of background subtraction algorithms", *Journal of Electronic Imaging*, vol. 19, nº 3, p. 033003–033003, 2010.

[185] J. C. Nascimento e J. S. Marques, "Performance evaluation of object detection algorithms for video surveillance", *IEEE Transactions on Multimedia*, vol. 8, nº 4, p. 761–774, 2006.

[186] Y. Liu, X. Chen, e D. Zhao, "Nonparametric background generation", *Journal Visual*

Communication Image Representation, vol. 18, p. 253–259, 2007.

[187] Xilinx, “Spartan-6 Family Overview”, vol. 2.0. Xilinx Inc, San Jose, CA, USA, p. 1–11, 2011.

[188] D. S. August, “Virtex-6 Family Overview”, vol. 2.5. Xilinx Inc, San Jose, CA, USA, p. 1–11, 2015.

[189] L. M. Garcés-Socarrás, “Aceleración de algoritmos mediante hardware reconfigurable: biblioteca de procesamiento de imágenes para System Generator”, Dissertação (Mestre em Sistemas Digitais), Universidad Tecnológica de La Habana “José Antonio Echeverría”, Havana-Cuba, 2011.

[190] L. M. Garcés-Socarrás, A. J. C. Sarmiento, S. Sánchez-Solano, P. B. Jiménez, E. Ieno Junior, e T. C. Pimenta, “Modificación automática de arquitecturas de módulos hardware de procesado de imágenes”, *Revista de Ingeniería Electrónica, Automática y Comunicaciones (RIELAC)*, vol. XXXVII, nº 3, p. 21–33, 2016.

[191] L. M. Garcés-socarrás, D. L. G. Pérez, J. A. C. Sarmiento, S. Sánchez-Solano, P. B. Jiménez, T. C. Pimenta, e E. Ieno, “Arquitecturas hardware modificables para bloques de convolución de imágenes sobre FPGA”, in *VI Simposio Internacional de Electrónica: diseño, aplicaciones, técnicas avanzadas y retos actuales*, 2018, p. 1–9.

[192] Xilinx, “7 Series FPGAs Data Sheet: Overview”, vol. 2.4. Xilinx Inc, San Jose, CA, USA, p. 1–18, 2017.