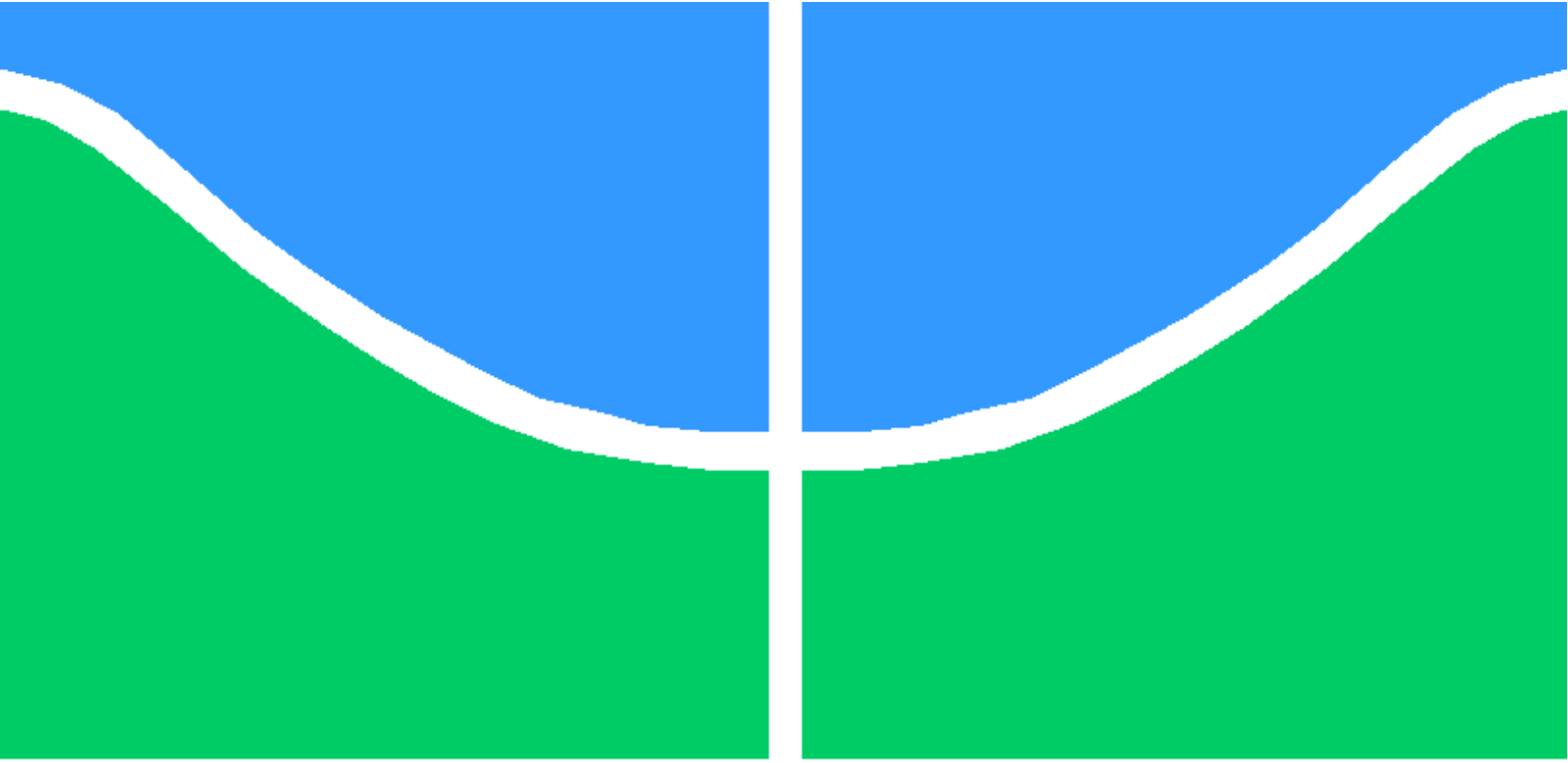

TRABALHO DE GRADUAÇÃO

CONTROLE DE SISTEMA DE DOIS GRAUS DE LIBERDADE COM REALIMENTAÇÃO VISUAL POR MEIO DE SEGMENTAÇÃO POR CORES EM PLATAFORMA RECONFIGURÁVEL

Antônio Pedro Cardillo Bittencourt 07/30271

Brasília, Dezembro de 2013

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

TRABALHO DE GRADUAÇÃO

CONTROLE DE SISTEMA DE DOIS GRAUS DE LIBERDADE COM REALIMENTAÇÃO VISUAL POR MEIO DE SEGMENTAÇÃO POR CORES EM PLATAFORMA RECONFIGURÁVEL

Antônio Pedro Cardillo Bittencourt 07/30271

Relatório submetido como requisito parcial para a obtenção do grau de Engenheiro de Controle e Automação

Banca Examinadora

Prof. Jones Yudi Mori, ENM/UnB _____
Orientador

Prof. Carlos Humberto Llanos Quintero, ENM/UnB _____
Co-orientador

Janier Arias Garcia, GRACO/UnB _____

Daniel Maurício Muñoz Arboleda, FGA/UnB _____

FICHA CATALOGRÁFICA

BITTENCOURT, A.P.C.
CONTROLE DE SISTEMA DE DOIS GRAUS DE LIBERDADE COM REALIMENTAÇÃO VISUAL POR MEIO DE SEGMENTAÇÃO POR CORES EM PLATAFORMA RECONFIGURÁVEL

[Distrito Federal] 2013.

xviii, 44p., 297 mm (FT/UnB, Engenheiro, Controle e Automação, 2013). Trabalho de Graduação – Universidade de Brasília. Faculdade de Tecnologia.

1.Visão Computacional 2.Sistemas Reconfiguráveis
3.Controle e Automação 4.Segmentação por Cores

I. Mecatrônica/FT/UnB

REFERÊNCIA BIBLIOGRÁFICA

BITTENCOURT, A.P.C. (2013). Controle de Sistema de Dois Graus de Liberdade com Realimentação Visual por Meio de Segmentação por Cores em Plataforma Reconfigurável. Trabalho de Graduação em Engenharia de Controle e Automação, Publicação FT.TG-nº17/2013, Faculdade de Tecnologia, Universidade de Brasília, Brasília, DF, 44p.

CESSÃO DE DIREITOS

AUTOR: A.P.C. BITTENCOURT.

TÍTULO DO TRABALHO DE GRADUAÇÃO: Controle de Sistema de Dois Graus de Liberdade com Realimentação Visual por Meio de Segmentação por Cores em Plataforma Reconfigurável

GRAU: Engenheiro de Controle e Automação ANO: 2013

Dedicatória

Dedico este trabalho a família, amigos e orientadores. Agradecendo ainda aos mesmos pelo suporte e paciência no período em que este trabalho foi realizado.

Antônio Pedro Cardillo Bittencourt 07/30271

RESUMO

O presente trabalho apresenta uma implementação reconfigurável de um sistema de segmentação por cores em espaço HSV, composto de tonalidade, saturação e brilho. Implementação esta que é comparada à de um sistema de segmentação por distância euclidiana no espaço RGB, composto de intensidades de vermelho, verde e azul. Com as informações da diferença da distância entre o centro da imagem e o centro do objeto obtidas a partir da segmentação, é feita uma interface entre a FPGA(arranjo de portas programável em campo) e um sistema mecânico e eletrônico capaz de alinhar a câmera extratora da imagem e o objeto de interesse.

ABSTRACT

This paper presents a HSV(hue, saturation and value) color segmentation application implemented in a reconfigurable system. That strategy is compared to another responsible for a RGB(red, green and blue) color segmentation based on the euclidean distance computation. Analysing the the distance between the image center and object center it is possible to build an interface between the FPGA(field-programmable gate array) and the mechanical and electronic system capable of align the camera used to extract the images and object of interest.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Símbolos Latinos

B	Componente azul do píxel
d	Distância entre o centro da imagem e o centro observado do objeto
$d\underline{euclidiana}$	Distância euclidiana
$d\underline{Mahalanobis}$	Distância de Mahalanobis
dc	Ciclo de trabalho
K	Constante de proporcionalidade
G	Componente verde do píxel
H	Componente de tonalidade do píxel
N	Número de processadores
p	Proporção do algoritmo a ser executada em paralelo
R	Componente vermelho do píxel
s	Proporção do algoritmo a ser executada em série
S	Componente de saturação do píxel
$speedup$	Velocidade de processamento
V	Componente de brilho do píxel

Símbolos Gregos

μ	Valor típico da variável a qual se deseja identificar
σ	Raiz quadrada do elemento de covariância

Siglas

ASIC	Application Specific Integrated Circuit
CD	Compact Disc
CPLD	Complex Programmable Logic Device
EEPROM	Electrically-Erasable Programmable Read-Only Memory
FPGA	Field-programmable gate array
HDL	Hardware Description Language
HSV	Hue, Saturation and Value
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
LUT	Look-up Table
PAL	Programmable array logic
PROM	Programmable read-only memory
RGB	Red, Green and Blue
ULA	Unidade lógica Aritmética
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuits
VLSI	Very Large Scale of Integration

Capítulo 1

Introdução

Observa-se uma evolução significativa nas áreas de processamento e análise de imagens. Tal evolução não se deve somente ao maior interesse de profissionais oriundos de domínios de conhecimento diversos, como medicina, biologia, artes, sensoriamento remoto, microscopia e engenharia de controle e automação, este último o domínio em que é enquadrado este projeto, mas se deve também à evolução tecnológica que permite recursos computacionais mais robustos para operações e manipulações em imagens(Pedrini e Schwartz,2008). Pode-se citar como exemplo de tecnologia que permite o avanço do estudo de processamento e análise de imagens, os FPGAs, instrumentos que permitem uma implementação reconfigurável em Hardware.

1.1 Visão Computacional

É razoável argumentar que, dentre todos os sentidos humanos, a visão é o mais importante (Bailey,2011). A complexidade da capacidade humana de captura, processamento e interceptação de informação visual é um grande motivador do desenvolvimento de novas técnicas de visão computacional e algoritmos de análise e processamento de imagens(Pedrini e Schwartz,2008). Entende-se por processamento digital de imagens as técnicas de extração, representação e modificação de imagens com o auxílio de sistemas computacionais(Pedrini e Schwartz,2008),sendo uma imagem uma representação espacial de um objeto, cena ou outro fenômeno(Harliack e Saphiro,1991). Representação esta que é útil a uma quantização espacial por amostrar uma função contínua. Tais técnicas visam facilitar a percepção humana e a interpretação por meio de máquinas da imagem processada, ainda que utilizem pouco conhecimento sobre o conteúdo e semântica, estes interessantes à análise de imagens. Exemplos de métodos de processamento incluem redução de ruído, aumento de contraste, extração de bordas e compressão de imagens. Como etapa anterior ao processamento, a imagem

deve ser digitalizada e convertida para um formato adequado. Inicialmente, para este projeto, a imagem é representada por uma matriz tridimensional com valores de intensidade ou quantidade de tons primários de vermelho, verde e azul. Tal definição será detalhada posteriormente. De forma resumida, diz-se que o processamento pode ser subdividido em aquisição, pré-processamento, segmentação, representação, descrição, reconhecimento e interpretação. Aquisição é a captura da imagem por meio de dispositivo ou sensor(Pedrini e Schwartz,2008). Pré-processamento consiste da atenuação de ruído, correção de contraste ou brilho, suavização de propriedades presentes na imagem, entre outros métodos(Pedrini e Schwartz,2008). Segmentação é a extração de áreas de interesse contidas na imagem, com base em detecção de descontinuidades, bordas, similaridades regiões(Pedrini e Schwartz,2008). Representação é o uso de estruturas para armazenamento e manipulação dos objetos de interesse contidos na imagem(Pedrini e Schwartz,2008). Descrição é a extração de características ou propriedades que possam ser utilizadas na separação em classes de objetos(Pedrini e Schwartz,2008). No reconhecimento atribui-se um identificador ou rótulo aos objetos da imagem e, em seguida, é feita a interpretação, quando se atribui um significado aos objetos reconhecidos(Pedrini e Schwartz,2008). Análise digital de imagens é baseada em formas, texturas, cores ou níveis de cinza, na qual geralmente se aplicam conhecimentos de diversos domínios como geometria computacional, visualização científica, psicofísica, estatística, teoria da informação entre outros(Pedrini e Schwartz,2008).Envlovem, consequentemente tarefas como segmentaçào da imagem em regiões de interesse, reconhecimento ou classificação de objetos e redução dos mesmos a formas que melhor representam o conteúdo da imagem. A análise de imagens é a abstração de alto nível da visão computacional, enquanto o processamento é a abstração de baixo nível. A divisão entre as duas facilita o entendimento e o estudo de visão computacional. Processamento de imagens pode ser também definido como a ação de uma série de operações matemáticas em uma imagem para obter o resultado desejado e análise como o uso de um computador para extrair dados de imagens(Bailey,2011). Com isto define-se visão de máquina como o uso de processamento e análise de imagens como parte de um sistema de controle para uma máquina(Schaffer,1984). Conforme foi dito, a imagem se caracteriza por quantizar uma área ou volume (para imagens tridimensionais) do espaço. A imagem é uma amostra de uma função contínua arbitrária,em que de outra forma seria difícil representar computacionalmente. Esta função contínua é porém armazenada em um array discreto, mais intuitivamente este array é de duas dimensões, mas a extração de imagens com duas câmeras seguida de uma rotina de operação de triangulação permite o armazenamento de uma imagem tridimensional em um array. Arrays de mais dimensões, embora não intuitivos, também são possíveis. O processo de amostragem resulta em uma série de elementos de imagem discretos que são os pixéis para imagens bidimensionais e os voxels para imagens tridimensionais, a quantização se refere a atribuição dos valores 1 ou 0 a cada um destes elementos(Bailey,2011).

1.2 Aspectos interessantes de implementações em hardware

Um sistema embarcado é um sistema computacional que é embarcado em um produto ou componente. Tal sistema é usualmente destinado à performance de uma tarefa específica(Catsoulis,2005).

Sistemas embarcados podem funcionar como um sistema de tempo real. Definem-se estes

sistemas como aqueles cuja resposta deve ser dada em um tempo específico, caso contrário o sistema é considerado como falho(Dougherty e Laplante,1985). Um sistema de visão computacional é tido como de tempo real se todos os seus procedimentos de captura, processamento, análise e uma tomada de decisão a partir dos dados da análise estão restritos a um tempo específico, que é comumente, mas não obrigatoriamente, a taxa de frames enviados pela câmera. A definição anterior de sistemas de tempo real é atribuida para sistemas de tempo real do tipo hard em Bailey(Bailey, 2011), sendo feita uma distinção entre este e um sistema em tempo real do tipo soft, no qual o sistema não é considerado falho caso não sejam executadas todas as tarefas durante o período especificado, mas existe uma deterioração na performance. Um "streaming online"de vídeo é do tipo soft, pois são geradas saídas, ainda que o frame de entrada não seja recebido ou decodificado. Sistemas que não são de tempo real podem possuir componentes que sejam. Um sistema de visão computacional que não é de tempo real pode fazer algum processamento no tempo entre o envio de dois píxeis ou pelo menos armazenar o pixel de entrada. É importante observar que estes tempos de captura são da ordem de dezenas de nanossegundos, por isto é necessária a implementação destes componentes em hardware (mais vantagens da implementação em hardware serão descritas ao longo deste texto). Por último um sistema de tempo real não é necessáriamente um sistema de alto desempenho. Tal associação nem sempre correta é todavia comum (Dougherty e Laplane,1985). Isto não ocorre quando o tempo de resposta especificado for bem grande em determinadas aplicações, ou ainda quando o algorítimo do sistema for simples não requerendo assim alto desempenho(Bailey,2011).

Para algoiritimos predominantemente sequenciais não existem grandes vantagens em uma implementação que explora o paralelismo. Porém, para outros algoritimos com várias partes que podem ser executadas em paralelo, tal implementação proporciona uma maior velocidade possível prevista pela lei de Amdahl(Amdahl,1967)(equação 1.1). Na mesma,a variável **s** é a proporção do algorítimo que deve ser executada em série enquanto a variável **p** é a fração do algoritmo que pode ser executada em paralelo em **n** processadores(definindo inicialmente paralelismo para sistemas multiprocessadores). E a igualdade na relação de menor ou igual é atingida quando não existem custos extras de processamento provocados pelo paralelismo(como a comunicação entre processadores).

$$Speedup \leq \frac{s+p}{s+\frac{p}{N}} = \frac{N}{1+(N-1)s} \tag{1.1}$$

A velocidade é, conforme já exposto, limitada pela proporção do algoritmo que deve ser executada em série:

$$speedup = \lim_{N\to\infty} \frac{1}{s} \tag{1.2}$$

Felizmente, componentes das etapas de processamento são excencialmente paralelos, especialmente nas primeiras etapas(Bailey,2011). Mas estas etapas são executadas de forma sequencial. Para que cada etapa seja executada em um processador separado, pode ser feita uma organização em pipleining. Tal organização funciona como uma linha de montagem, em que os dados são transmitidos entre as etapas durante o processo. Este procedimento não minimiza o tempo de resposta, para um único dado o processamento demora até um pouco mais devido à inserção de registra-

dores de pipelining, mas aumenta o throughtput, pois o primeiro processador pode processar o segundo dado quando o segundo processador já recebeu o primeiro dado. Define-se latência como o tempo entre o primeiro dado de entrada ser inserido e o útlimo dado de saída estar disponível. Sistemas nos quais toda a imagem deve ser processada para gerar uma saída são de alta latência. Sistemas de baixa latência possuem maior benefício em uma implementação em pipeline. Em uma implementação em hardware, a latência do sistema é dada pela soma das latências de cada um dos estados somado ao tempo de inserção da imagem completa. Com pequenas latências, a varíavel speedup tem um valor alto próximo a N(número de processadores no exemplo multiprocessador).

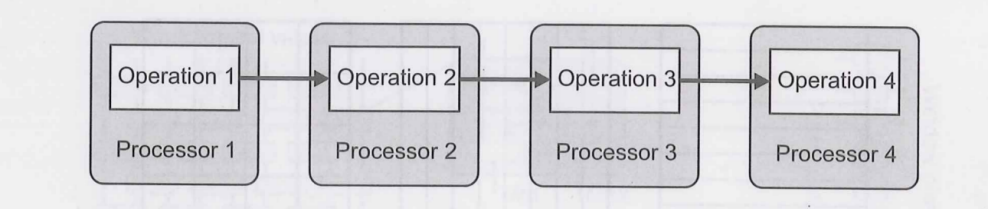

Figura 1.1: Organização em Pipeline explorando o paralelismo temporal

Uma das duas dificuldades que podem ser encontradas no uso de pieplining para algorítimos de maior complexidade (Duff, 2000) é a sincronização do caminhos de dados. Por exemplo, na figura 1.1 o processador do estágio 4 poderia receber também dados do processador do estágio 1, por isto usar um sistema síncrono em vez de um assíncrono garante uma maior robustez. Em um sistema síncrono é possivel passar estes dados pelos registradores de pipelining em cada estágio criando uma linha de atraso. A outra e maior dificuldade é trabalhar com realimentação do sistema, isto pode ocorrer tanto para algoritmos iterativos quanto para sistemas em que os estágios iniciais adequam sua execução aos resultados dos estágios finais. Procedimentos para resolver tal dificuldade não são abordados neste trabalho. Paralelismos do tipo espacial são aqueles que quando particionada a imagem, geralmente em blocos de linhas, colunas ou blocos retangulares,como mostrado na figura 1.2, são aplicadas as operações em paralelo para cada partição. No código estas operações agem desde os loops internos até o loop mais externo que contém a ordem de iteração das instruções. Neste paralelismo, o desempenho é limitado pela comunicação entre processadores que interpretam cada partição. Dividir a informação em memórias locais acessadas individualmente por cada processador torna a comunicação mais eficiente do que usar uma única memória global. Para os estágios posteriores, as operações não são mais feitas diretamente na imagem, porém podem ser feitas partições delimitando estruturas de dados, regiões ou objetos.

Além do paralelismo espacial tem-se o paralelismo lógico que separa cada instância do bloco funcional em paralelo usando o bloco funcional mais de uma vez para cada operação. Um exemplo de aplicacão em processamento de imagens do paralelismo lógico é a implementação do filtro linear de convolução(Bailey,2011). Este filtro executa operações de multiplicação em cada pixel da imagem por um peso ou coeficiente e no mesmo os blocos de multiplicação e adição são repetidos diversas vezes.

As definições anteriores para paralelismo foram feitas para paralelismo em sistemas multiprocessadores, porém há pouca diferença teórica entre estas definições em um sistema implementado

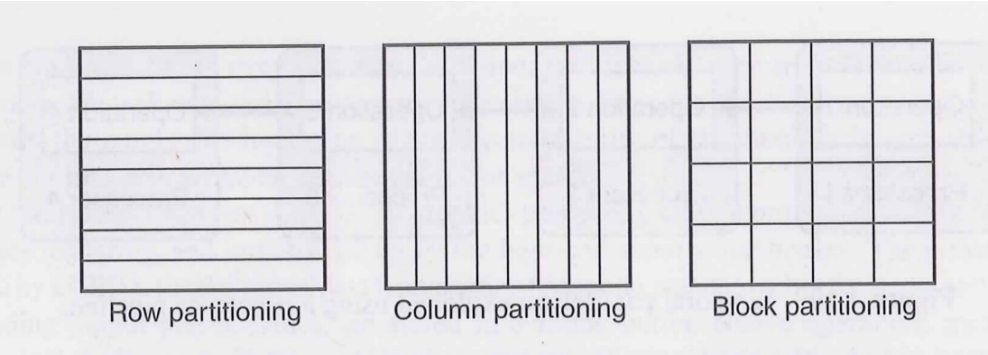

Figura 1.2: Paralelismo espacial ao particionar a imagem em linhas, colunas ou blocos

em hardware dedicado, este último presente no escopo deste trabalho. O que foi afirmado para cada processador em um sistema multiprocessadores pode ser traduzido para uma área separada do hardware implementado. Inicialmente, sistemas computadorizados eram bastante lentos para processar imagem de qualquer tamanho ou em qualquer volume (Bailey, 2011). A estrutura regular de imagens levou ao design de sistemas de hardware que explorassem o paralelismo, pois sistemas de hardware são intrinscecamente paralelos. Após implementações em hardware de pequena e média escala, o advento da VLSI(integração em larga escala),integração que consiste de circuitos integrados fabricados com muitos transistores por encapsulamento, diminuiu custos de hardware e melhorou a velocidade e a performance dos sistemas de hardware. Uma maior variedade de arquiteturas de hardware se tornou avaliável sendo implementadas nestas o processamento de imagens (Offen,1985). A desvantagem dos sistemas em hardware, apesar da sua alta velocidade, era sua inflexibilidade. Sendo difícil, senão impossível, reconfigurar este tipo de sistema quando existiam mudanças na tarefa executada. Nos anos 1980 houve a introdução dos FPGAs(Arranjo de portas programável em campo) que combinavam o paralelismo intrínsceco do hardware com a flexibilidade de um software, uma vez que as suas funcionalidades podem ser reprogramadas. Na época, os FPGAs possuiam pouca densidade de portas sendo usados para prover uma lógica de interfaceamento flexível entre componentes e para implementação de controladores com máquinas de estado finitas. Posteriormente FPGAs mais poderosos foram usados como coprocessadores de visão computacional ou aceleradores operando com um computador hospedeiro(por exemplo Splash-2 system; Athanas e Abbot,1995). FPGAs modernos possuem atualmente recursos suficientes para permitir que aplicações inteiras sejam implementadas em um único FPGA, tornando-os a escolha ideal para sistemas embarcados em tempo real de visão computacional(Bailey,2011). Computadores se baseam na ideia de ter um circuito genérico no qual a funcionalidade pode ser programada para uma diferente aplicação. A ULA(uniladade lógica aritmética) é encarregada de executar diversas operações a partir de um conjunto de sinais de entrada e de controle. Porém a ULA está limitada a executar uma instrução por vez e qualquer aplicação a ser nela executada deve ser quebrada em uma sequência de sinais de controle que devem ser armazenados em uma memória. Ao contrário da ULA, uma lógica programável é uma representação funcional de um circuito que é implementado como um sistema paralelo, não sequencial. A lógica programável também poderia ser programada para atingir os requisitos da aplicação. Qualquer função lógica pode ser implementada em função dos termos OR e AND, portanto os primeiros dispositivos de lógica programável consistiam de

uma arquitetura em arrays usando estes dois termos. Cada entrada, e também a sua inversa, poderiam ser conectadas a um set de portões OR ou AND. Os círculos na figura 2 representam as conexões programáveis entre a linha vertical de entradas com a horizontal. Estes arrays de ORs e ANDs podem ser implementados apenas com NANDs segundo o teorema de De Morgan. Uma lógica sequencial, como contadores e máquinas de estado finito, poderia ser implementada com estes dispositivos ao se acresentar latches ou flip-flops com a realimentação adequada.

Os primeiros dispositivos lógico-programáveis tinham o objetivo de simplificar circuitos que continham muitos dispositivos lógicos. Dentre estes dispositivos, os programáveis em campo poderiam ser implementados com diversos fusíveis que, ao aplicar uma corrente muita alta, teriam as conexões indesejadas rompidas. Porém eram mais usualmente implementados com antifusíveis que, ao aplicar uma alta voltagem, eram rompidos permitindo a condutância da linha desejada. As duas metodologias eram destrutivas e dispositivos com esta implementação não podiam ser reprogramados. Posteriormente, os fusiveis foram substituídos por EEPROMs(memória de apenas leitura elétricamente apagável e programável) que podia ser programada, permitindo modificações remotas e posteriores em produtos lógico-programáveis. Outro paradigma surgido foi o do PAL(Array lógico programável) que mantinha uma conexão prévia da saída das portas AND e entrada das portas OR. Embora isto limitasse a flexibilidade, problema que poderia ser resolvido pois os circuitos mais complexos que excediam estas limitações podiam ser implementados a partir de uma combinação das saídas, isto diminuia a área do circuito e aumentava a velocidade da resposta das portas OR. Estes PALs foram posteriormente simplificados, mantendo a seção AND fixa e fazendo a OR programável(veja estas implementações na figura 1.3). Cada porta AND era implementada como uma linha da tabela verdade das funções de entrada. Havendo portanto 2N portas AND para N entradas. Estas estruturas resultavam PROMs(memórias programáveis de somente leitura) e LUTs(look-up table, traduzida aqui livremente como tabela de checagem) e tinham consideravel flexibilidade, apesar do crescimento exponencial da área do circuito com um maior número de entradas. PROMs mais largos sãom muito mais lentos que que circuitos lógicos dedicados a uma tarefa específica e consomem muito mais energia. Mas por sua flexibilidade o PROM é o bloco mais usado nos FPGAs, sendo efetivo quando há um pequeno número de entradas. A figura 1.4 mostra a implementacao real de uma PROM no FPGA em que a linha OR é substituída por uma célula de memória conectada a um transistor. Para um pequeno número de entradas, a célula de memória pode ser conectada a uma árvore de multiplexação.

Ao se integrarem vários blocos lógicos em um único chip, surgem duas classes de dispositivos: Os CPLDs(dispositivos lógicos complexos programáveis), com uma maior área de blocos lógicos comparada às interconexões, e os já citados FPGAs, com menores blocos lógicos e uma arquitetura dominada por interconexões. CPLDs sao usualmente programáveis em memória flash, enquanto FPGAs, em células de memória estáticas que precisam ser programadas toda vez que o dispositivo é ligado(Bailey, 2011). Tanto os FPGAs como os CPLDs foram introduzidos nos anos 1980 e eram utilizados inicialmente para prover interfaceamento flexível entre placas de circuito impresso e outros componentes (Bailey,2011). Os FPGAs dominaram o mercado por sua estrutura de interconexão mais flexível. Os FPGAs atuais já possuem uma grande densidade de blocos lógicos, assim como sua interconexão associada, para implementar aplicacões complexas em um único chip.

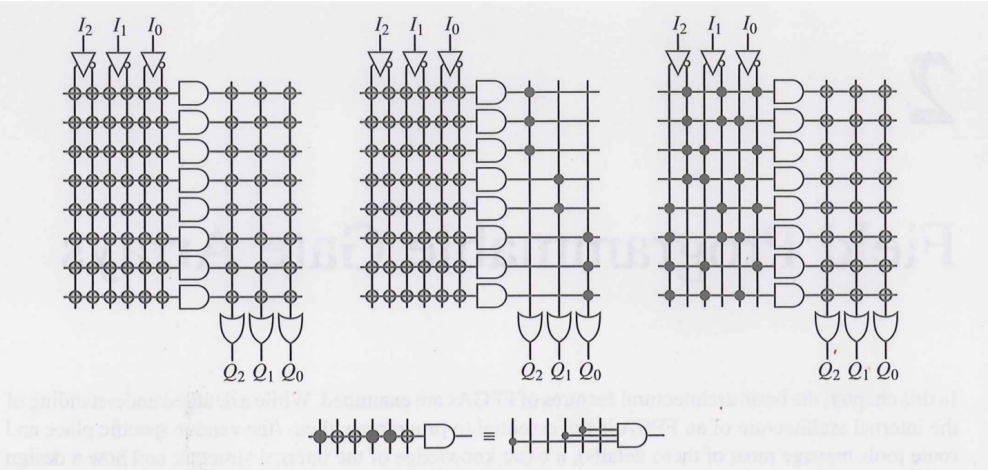

Figura 1.3: Arranjos diferentes para o PLA. O arranjo mais à esquerda possui tanto as seções OR e AND programáveis, enquanto o central possui apenas a seção AND programável, tendo seção OR fixa. E o arranjo da direita possui a seção OR programável e a seção AND fixa.

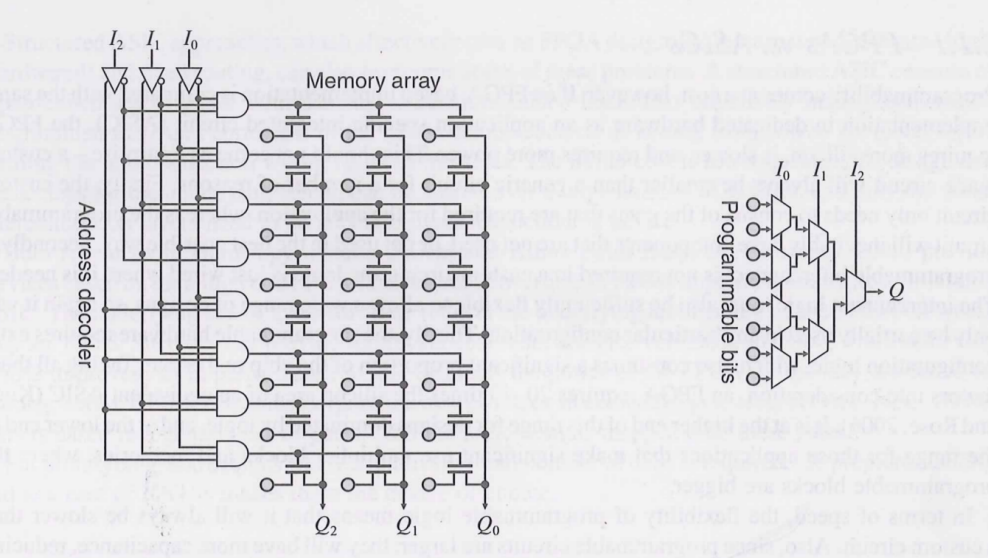

Figura 1.4: Arquiteturas PROM à esquerda implementadas como memória e à direita como árvore de multiplexação

São exemplos destas aplicações memórias, interfaces de entradas e saídas de grande velolcidade e até mesmo núcleos de processadores (Leong,2008). Implementar em FPGAs estes blocos, que seriam comumente implementados em software, economiza áreas de circuito lógico, eliminando o uso desnecessário, melhora a velocidade de resposta e economiza energia (Bailey,2011). Como se espera, existe um custo na programabilidade. Uma implementação em FPGA requer mais área de silício, é mais lenta e requer mais energia que um ASIC (circuito integrado específico para a aplicação). O fato de o circuito programável inevitavelmente constituir de circuitos que não serão usados na aplicação e provavelmente circuitos não utilizados da melhor forma possível é a primeira razão para isso(Bailey,2011). Circuitos programáveis precisam também de uma área dedicada à interconexão programável e ainda de uma uma área dedicada à lógica de configuração. Com isto,

a mesma aplicação implementada em FPGA possui de vinte a quarenta vezes a mesma área de silício do que uma implementada em um ASIC(Kuon e Rose, 2006). A maior área do circuito também indica que ele possui maior capacitância implicando uma resposta mais lenta. Outro fator que reduz a velocidade também provém de características físicas provocadas pela área de interconexão, pois esta afasta os conjuntos de blocos fazendo com que a informação trafegue por um caminho mais longo. A interconexão ainda introduz um atraso para cada switch que está conectada. Resumidamente, um FPGA consome dez a quinze vezes mais energia dinâmica do que a mesma implementação em um ASIC e é de três a quatro vezes mais lenta (Kuon e Rose,2006). Porém, além da programabilidade, um FPGA possui outras vantagens em relação a um ASIC. A máscara e o custo de desenho de uma ASIC são maiores apesar de possuir um custo por chip menor, tornando um FPGA mais barata quando é requerido menor volume de chips. Com as capabilidades crescentes dos FPGAs, o ponto de cruzamento dos custos é deslocado tendendo a desfavorescer os ASICs ao longo dos anos(Bailey,2011). FPGAs também são encomendados mais rapidamente pois requerem menor tempo de montagem e sua reconfigurabilidade permite a readaptação do sistema à uma nova necessidade sem a necessidade de outra encomenda(Bailey,2011). O comparativo entre os custos de um FPGA, um ASIC e um ASIC estruturado é mostrado na figura 1.5.

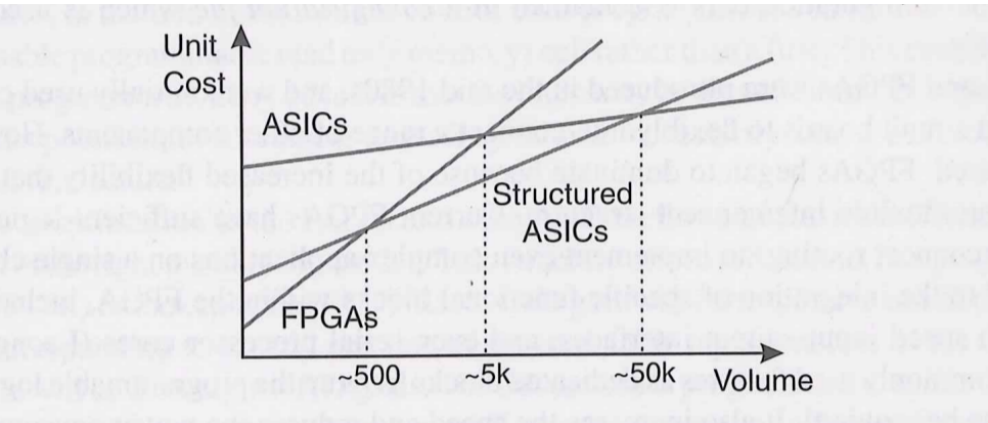

Figura 1.5: Comparativo entre os custos relativos de um FPGA, um ASIC e um ASIC estruturado.

Concluindo, FPGAs fornecem uma área de hardware separada para cada função sendo inerentemente paralelas. Isto os torna apropriados ao uso no processamento de imagens que é inerentemente paralelo, especialmente em suas primeiras etapas. O paralelismo espacial é explorado ao fazer cópias do hardware implementado e designar cada uma delas à uma partição da imagem. Outra maneira de impor um paralelismo é através da lógica ao se "desenrolar"os loops e transformar uma série de operações em várias operações em paralelo. Em uma arquitetura em pipeline, uma área de hardware seprarada é usada para cada operação, elas são organizadas em série e funcionam melhor em caminhos de dados síncronos. Para caminhos assíncronos, buffers podem ser incorporados para lidar com variações nos fluxos de dados ou padrões de acessos(Baiely,2011). Pielines são conhecidos por sua eficiência e são geralmente necessários para lidar com o stream de dados atingindo o throughtput necessário (Baiey,2011). Dados em stream ocorrem usualmente ao interfacear o FPGA com uma câmera ou display conforme ocorre neste trabalho.

1.3 Linguagens de descrição de hardware

A maneira mais tradicional de descrever um circuito eletrônico é a partir de um diagrama esquemático. Este é melhor aplicado a projetos de pequeno porte. Para projetos de grande porte, seu uso ainda é viável ao se organizar o circuito esquematizado em um conjunto de blocos funcionais. Ferramentas modernas também são capazes de integrar um projeto contendo um circuito esquematizado e um FPGA ou outro hardware programável. Porém, a desvantagem de tais sistemas é o fato de sua funcionalidade não ser transparente na sua representação estrutural dificultando a interpretação (Bailey,2011). Com isto, as linguagens de descrição de hardware(HDLs) evoluiram desde os anos 1980 para contornar as limitações dos diagramas estruturais e prover uma representação padrão do comportamento de um circuito. Pela natureza paralela do hardware, uma HDL deve ser capaz de representar um comportamento concorrente para o nível de portas lógicas. Isto as diferencia das linguagens de programação que organizam suas instruções de modo puramente sequencial. As duas principais HDLs são o VHDL e o Verilog, ambos seguindo um padrão IEEE. Ambas são linguagem hierárquicas e estruturais que descrevem estruturalmente os elementos lógicos do hardware e suas interconexões. Mas permitem também uma descrição funcional que é mais intuitiva. Nesta abstração deve ser implementado como o circuito deve se comportar ou responder a uma mudança nas entradas, a ferramenta de síntese é responsável por estabelecer a lógica necessária para implementar a funcionalidade desejada.

1.3.1 Verilog

Verilog em comparação ao VHDL é considerada uma linguagem mais compacta(Bailey,2011). Isto se deve ao fato de essa linguagem ter sido modelada com o objetivo de ser similar à linguagem C de programação, enquanto a última possui raízes na linguagem ADA. O verilog se organiza em módulos, cada qual designando um bloco funcional específico, com suas portas de entradas e saídas associadas. A descrição funciona de forma a referenciar os módulos internos e estabelecer a interconexão entre os módulos. Em cada módulo, existem as variáveis *wire*(fio comum) e *reg* que mantêm o seu valor até que seja mudado (tal instrução é geralmente, mas não obrigatoriamente, associada a um flip-flop). Variáveis podem ser criadas a partir de arranjos de bits e a instrução *assign* pode atribuir o valor de uma variável de destino a partir de uma variável ou expressão contendo uma variável de origem. Estas designações se organizam de forma concorrente e não sequencial. A expressão *initial* pode ser usada para atribuir um conjunto de instruções sequenciais implementadas uma vez durante a inicialização e a expressão *always* é capaz de sempre executar suas instruções internas sequencialmente. Contudo, *always* é mais comumente associado a um ciclo de relógio implementando suas instruções internas paralelamente ou concorrentemente. Assim como outras linguagens, o Verilog permite a implementação de diversas aplicações ao permitir ser abstraída em uma variedade de estilos de programação. Um estudo mais aprofundado desta linguagem foge por isto ao escopo deste trabalho.

1.4 Estratégias de rastreamento de objetos

Para solucionar o problema de rastreamento de objetos, é necessária uma análise na imagem do que o objeto difere do resto da imagem. Isto é feito através de um procedimento de segmentação. Segmentação é o processo de separar a imagem entre seus componentes significativos(Bailey,2011). A segmentação foi na literatura dividida em dois tipos, baseadas em contornos, que possuem a estratégia de detecção dos limites entre as regiões. Algoritimos comuns para isso são os de detecção de bordas ou filtros de gradiente. O outro grupo é o baseado em regiões, este classifica os píxeis como pertencentes a uma classe de objetos ou outra através de uma propriedade local. Rigorosamente, a segmentação consiste de por uma filtragem ,que destaca ou detecta a propriedade usada pela segmentação (Randen e Husoy, 1999), seguida por um Thresholding, classificação de um pixel, o designando a uma região. Estes métodos podem ser globais, associados à imagem inteira ou incrementais, quando acontecem operações locais. Segmentações podem analisar cor, intensidade, textura ou qualquer outra propriedade. O fato de o objeto rastreado neste ser de uma cor chamativa, sugere a implementação de um sistema de segmentação por cores. Este sistema é baseado em regiões. Comparado à segmentação por subtração de fundos, cuja literatura está presente na bibliografia(Ferreira,2012) da qual foram aproveitados entre outros blocos funcionais, o bloco para cálculo de centro de massa da imagem binária, segmentar por cores apresenta a vantagem de possibilitar a segmentação em diferentes espaços e apresenta uma limitação ao poder somente segmentar aquele objeto específico ou algum objeto da mesma cor. Utiliza-se a subtração de fundos geralmente em uma câmera estática na qual se desejam detectar variações na imagem, e a segmentação por cores para rastrear um objeto cuja cor não se confunda com a de outros objetos ou elementos do fundo não interessantes ao rastreamento.

1.4.1 Espaços de Cores

Antes de discutir segmentação por cores, serão apresentados alguns modelos de cores, ou espaços em que essas cores são representadas. No Espaço RGB, cada cor aparece representada por seus componentes espectrais primários vermelhos, verdes ou azuis. Os componentes são dispostos através de um espaço cartesiano tridimensional. No cubo mostrado na figura 1.6, é possível ainda observar os componentes secundários: amarelo, ciano e magenta como arestas. Preto é a origem do eixo cartesiano e branco o ponto mais afastado.

O outro espaço interessante a este projeto é o chamado HSV, descrito através dos componentes intensidade, saturação e brilho. Este espaço está mais associado à maneira psicológica pela qual as cores são interpretadas. Este espaço é geralmente representado como um cone conforme representado na figura 1.7. A intensidade é o ângulo ao redor do cone, a saturação a sua distância ao centro e o brilho sua componente vertical. Uma segmentação feita no HSV através apenas da componente de intensidade é vantajosa em relação a uma no espaço RGB por, entre outros motivos, ser menos sucetível à interferencia das variações da iluminação ambiente na imagem. As componentes de brilho e saturação podem ser usadas para detectar ou excluir cores preto,cinza e branco. No decorrer deste trabalho, foi feita uma segmentação no espaço HSV por meio de um

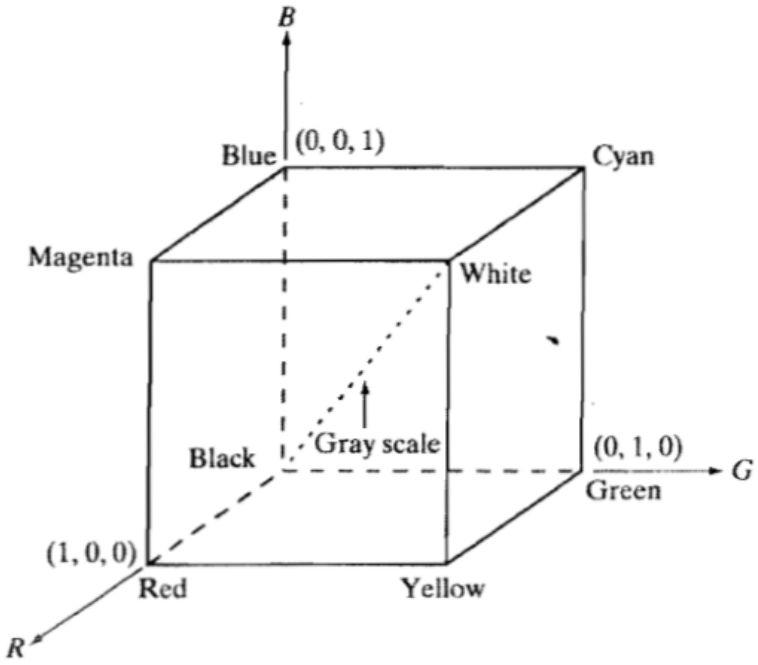

Figura 1.6: Representação RGB

simples thresholding, porém é apresentada também uma implementação de segmentação por distância euclidiana no espaço RGB e a definição da distância de Mahalanobis(distância euclidiana generalizada).

As equações para conversão de um pixel no espaço RGB para outro no espaço HSV são as seguintes, com **Max** e **Min** correspondendo aos valores máximo e mínimo das componentes **R**,**G** e **B**:

$$H = 64\frac{G-B}{Max-Min} \tag{1.3}$$

se Max=R e G $\geq$ B

$$64\frac{G-B}{Max-Min} + 384 \tag{1.4}$$

se Max=R eG $<$ B

$$64\frac{B-R}{Max-Min} + 128 \tag{1.5}$$

se Max=G

$$64\frac{B-R}{Max-Min} + 256 \tag{1.6}$$

se Max=B

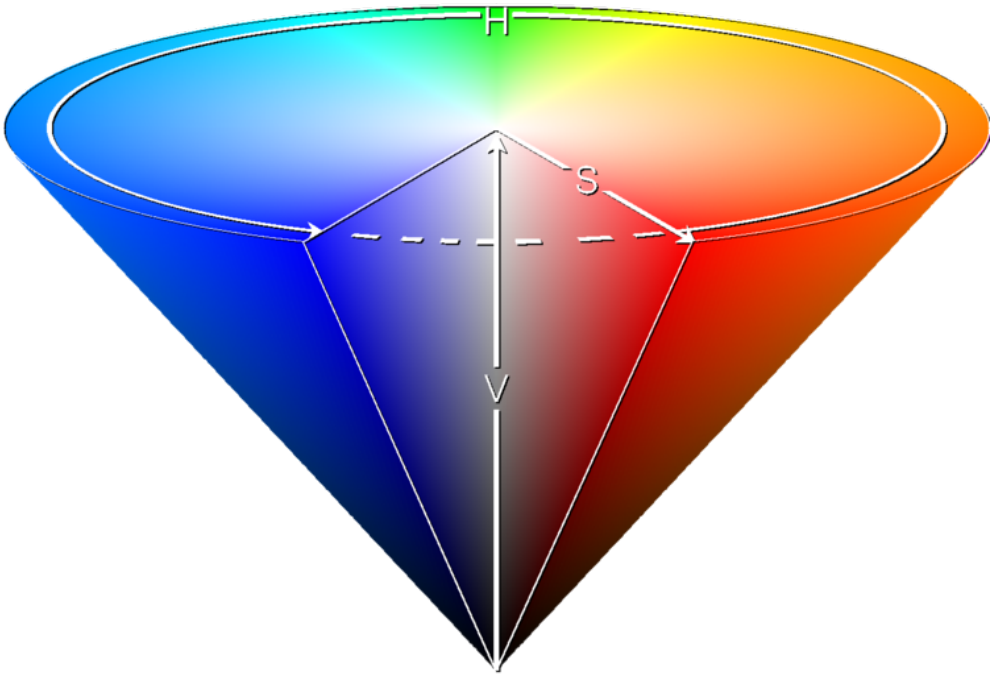

Figura 1.7: Representação em cone do espaço HSV

$$S = 100\frac{Max - Min}{Max} \tag{1.7}$$

$$V = 100Max \tag{1.8}$$

Estas equações retornam valores de saturação(**S**) e Brilho(**V**) em uma escala de zero a cem. A intensidade(**H**) é retornada em uma escala de zero a trezentos e oitenta e quatro. Este valor foi escolhido em vez da escala de zero a trezentos e sessenta graus, que representaria a angulação do cone, pois assim foi possível substituir a operação de multiplicação pela constante de valor sessenta, por outra de valor sessenta e quatro que equivale a um deslocamento lógico em hardware de 6 casas para a esquerda, acelerando a operação e diminuindo o tempo de resposta do sistema.

São apresentadas neste trabalho as seguintes estratégias de segmentação no espaço RGB, através da distância euclidiana e da distância de Mahalanobis.

A distância euclidiana é definida por:

$$\underline{deuclidiana} = \sqrt{(x-\mu)^T(x-\mu)} \tag{1.9}$$

Da estatística, para o vetor x, μ seria o vetor de seus respectivos valores médios. Porém, nesta aplicação temos μ como sendo valores típicos da cor a qual se deseja indetificar na imagem. Observa-se que estes vetores possuem três dados que representam as componentes vermelho,verde e azul da cor.

A formularização da distância euclidiana poderia ser generalizada atrvés da equação de Mahalanobis que é da forma:

$$d\underline{Mahalanobis} = \sqrt{(x-\mu)^T(\sigma^2)^-1(x-\mu)} \tag{1.10}$$

Onde σ^2 era a matriz de covariância para quando μ tinha significado estatístico. Mesmo com a outra interpretação de μ , mantem-se a mesma definição de σ^2 que é para três componentes de cores:

$$\sigma^2 = \begin{bmatrix} \sigma^2(I1,I1) & \sigma^2(I1,I2) & \sigma^2(I1,I3) \\ \sigma^2(I2,I1) & \sigma^2(I2,I2) & \sigma^2(I2,I3) \\ \sigma^2(I3,I1) & \sigma^2(I3,I2) & \sigma^2(I3,I3) \end{bmatrix} \tag{1.11}$$

O elemento de covariância $\sigma^2(Ii,Ij)$ para esta aplicação de segmentação por cores tem uma definição simplificada, oriunda da teoria estatística, em que ele é definido como o valor esperado da expressão a seguir:

$$\sigma^2(Ii,Ij) = (X(i)-\mu(i))(X(j)-\mu(j)) \tag{1.12}$$

Com i e j variando de um a três, representando cada elemento do vetor.

Para a segmentação é feito uma simples operação de Thresholding classificando os píxeis como zero ou um em uma imagem binária. O threshold corresponde a uma distância, euclidiana ou de mahalanobis, arbitrária. É sugerida para implementações em hardware a retirada da raiz quadrada das expressões para o cálculo desta distância e uma escolha de threshold equivalente ao quadrado do threshold anteriormente escolhido. Isto é feito para simplificar a implementação, pois uma raiz quadrada implementada em linguagem de descrição de hardware corresponde a um número razoavelmente grande de blocos funcionais.

1.5 Definição do problema

Espera-se com este trabalho poder realizar o "tracking"de um objeto (no caso uma bola de frescobol) ao se distinguir a sua cor do fundo. Um pan-tilt, mecanismo de dois graus de liberdade, será usado como uma estrutura mecânica de suporte à câmera. A informação da câmera será repassada a um FPGA onde podemos aproveitar a maior eficiência com o paralelismo descrevendo o algoritmo de tracking em hardware.

1.6 Objetivos do projeto

Para resolver o problema proposto serão realizadas as seguintes sequências de atividades:

-Teste de algorítimos de visão computacional no software Matlab e análise de resultados.

-Descrição em Hardware em Verilog, cuja funcionalidade é equivalente a do algorítimo escolhido.

-Simulação do código Verilog.

- Implementação em FPGA

-Projeto de um pan-tilt com auxílio do Solidworks.

-Construção e montagem do Pan-Tilt com chapas de alumínio e conexão a motores de corrente contíinua.

-Implementação de circuito eletrônico como interface entre o FPGA e os motores de corrente contínua.

Capítulo 2

Desenvolvimento

Nesta seção serão apresentados detalhes da plataforma utilizada. Isto inclui os componentes eletrônicos presentes nos Kits Altera e Terasic utilizados e também a estrutura mecânica, motores e a interface eletrônica pensada para o funcionamento do sistema. Foi usado neste trabalho o FPGA da Altera e um kit de Câmera e Display da Terasic compatível com o FPGA escolhido. A câmera é sustentada por um Pan-Tilt, mecanismo de dois graus de liberdade. O projeto desta estrutura foi feita para este trabalho e sua construção será detalhada mais a frente.

2.1 Plataforma utilizada

O fato de o objeto rastreado ser de uma cor chamativa sugere a implementação de um sistema de segmentação por cores. Foram propostas estratégias para isto. Conforme foi dito, a partir da camera usada é possível acessar os canais de cores vermelho, verde e azul. Como estratégias de segmentação por cores no espaço RGB.

2.1.1 Kit Altera e Terasic

2.1.1.1 Camera

Foi utilizada a câmera Terasic TRDB_D5M Digital Camera Package. Do seu datasheet, pode ser extraída a seguinte tabela na figura 2.2:

A partir da tabela é possível entender que os píxeis da imagem vêm em uma grade quadrada formada a partir da imagem RGB conforme extraidas pelo filtro de Bayer. Na resolução VGA(640x480), que será utilizada para tratamento das imagens, podem ser transmitidos até 70 frames por segundo e a máxima transferência de dados é de 96MP/s com 96MHz de máximo ciclo de relógio. A alimetação da câmera é de 3,3 V e tal alimentação pode ser fornescida pelo FPGA.

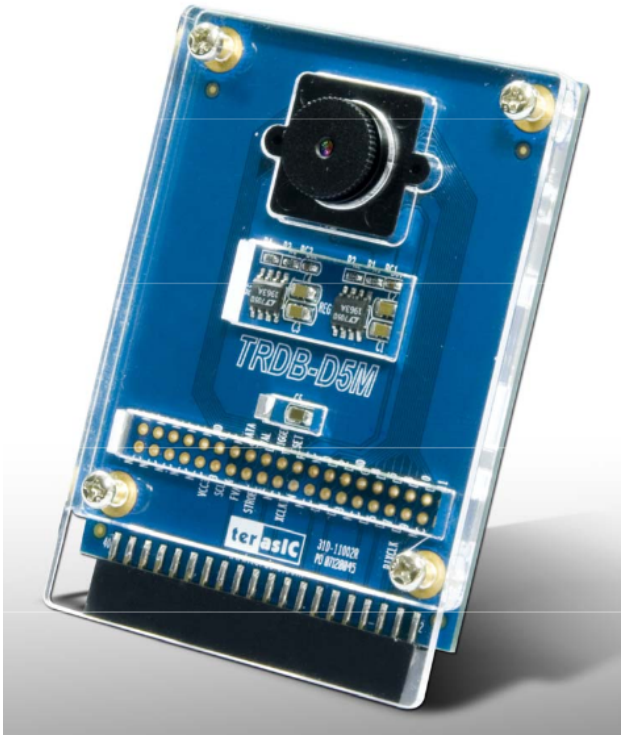

Figura 2.1: Câmera Usada

Parameter		Value
Active pixels		2,592H x 1,944V
Pixel size		2.2μ m x 2.2μ m
Color filter array		RGB Bayer pattern
Shutter type		Global reset release (GRR),
Maximum data rate/master clock		96 Mp/s at 96 MHz
Frame rate	Full resolution	Programmable up to 15 fps
	VGA (640 x 480)	Programmable up to 70 fps
ADC resolution		12-bit
Responsivity		1.4 V/lux-sec (550nm)
Pixel dynamic range		70.1dB
SNRMAX		38.1dB
Supply Voltage	Power	3.3V
	I/O	1.7V ~ 3.1V

Figura 2.2: Características da Câmera

2.1.1.2 Display

Utilizou-se o Display Terasic TRDB_LTM 4.3"LCD Touch Panel Package e, ao conferir o seu datasheet, é possível garantir a compatibilidade do display com o restante do sistema.

Figura 2.3: Display Usado

Figura 2.4: FPGA usado

2.1.1.3 FPGA

Finalmente, o FPGA utilizada foi o Altera DE2 e a informação de descrição de Hardware foi enviada ao chip Altera Cyclone II 2C35 FPGA device. Segundo seu datasheet podemos extrair as seguintes informações úteis acerca das capacidades de memória e relógio do FPGA:

- 512-Kbyte SRAM

- 8-Mbyte SDRAM

- 50-MHz oscillator and 27-MHz

- Oscillator for clock sources

E as seguintes informações sobre entradas e saídas usadas:

- VGA DAC (10-bit high-speed triple DACs) with VGA-out connector

- USB Host/Slave Controller with USB type A and type B connectors

- RS-232 transceiver and 9-pin connector

- PS/2 mouse/keyboard connector

- Two 40-pin Expansion Headers with diode protection

Nota-se que um dos Headers de expansão foi usado para conexão da câmera e o outro para conexão do Display, existem na câmera pinos não conectados conforme especificados em seu datasheet, estes foram usados para os sinais de PWM que serão explicados na seção de interface.

2.1.2 Pan-tilt

O pan-tilt foi construído com chapas de alumínio. Alguns processos de fabricação foram necessários para atingir o resultado final. As chapas foram dobradas com auxílio de uma dobradeira e foram executados ainda processos de furação. A esquematização do pan-tilt no SolidWorks é representada na figura 2.5.

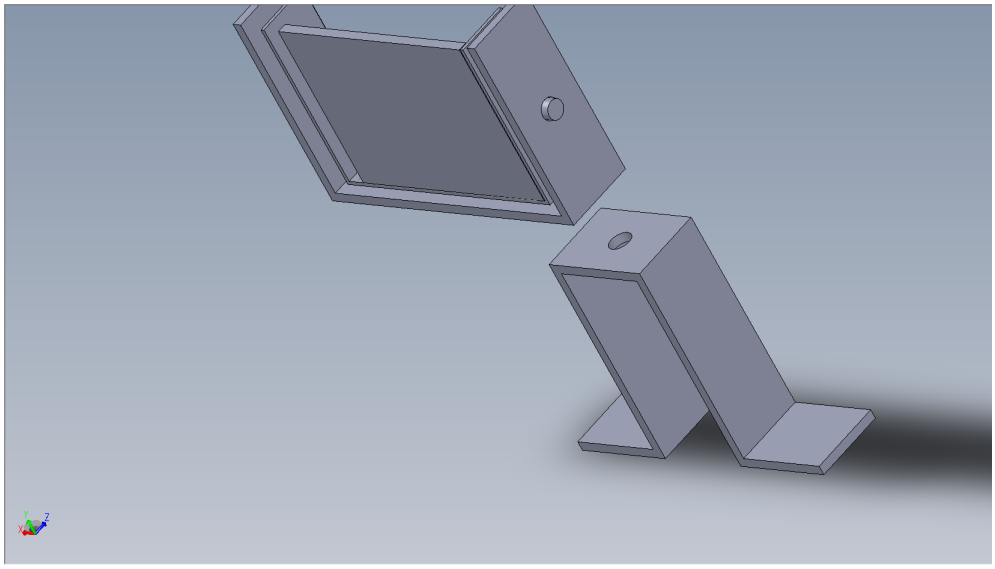

Figura 2.5: Desenho no SolidWorks do Pan-Tilt

Nesta figura se pode observar que o pan-tilt é composto de uma estrutura em forma de U na parte inferior sem liberdade de rotação. Esta estrutura possui um furo para a passagem do primeiro

motor responsável por girar a estrutura em forma de U superior em torno do eixo Y(vertical). A estrutura em forma de U em cima suporta um segundo motor acomodado horizontalmente na mesma unidade que aciona um sistema de duas engrenagens e uma esteira na lateral da estrutura responsável por girar o eixo que passa pelos furos superiores ainda desta estrutura. Este eixo provoca uma rotação em torno do eixo X(horizontal) da última estrutura que possui superfície retangular na qual podem ser acopladas duas câmeras, embora neste trabalho só tenha sido usada uma. No anexo de fotos adicionais teremos as fotos da pan-tilt já construída.

2.2 Descrição em hardware

A estrutura do sistema foi aproveitada do trabalho referenciado(Ferreira,2012) e o bloco de processamento de imagens feito substitui um outro que usava como estratégia a subtração de fundos. Este novo bloco, que usa segmentação por cores, está hachurado em vermelho na figura 2.6. A figura mostra também um bloco responsável por aplicar um filtro de erosão retirando as bordas da imagem extraída, que estão mais sujeitas a erros. E também consiste de um bloco que faz o cálculo do centro de massa armazenando a imagem inteira nos dispositivos de memória presentes no FPGA. Estes blocos foram aproveitados do trabalho original. Este último sistema é um bloco

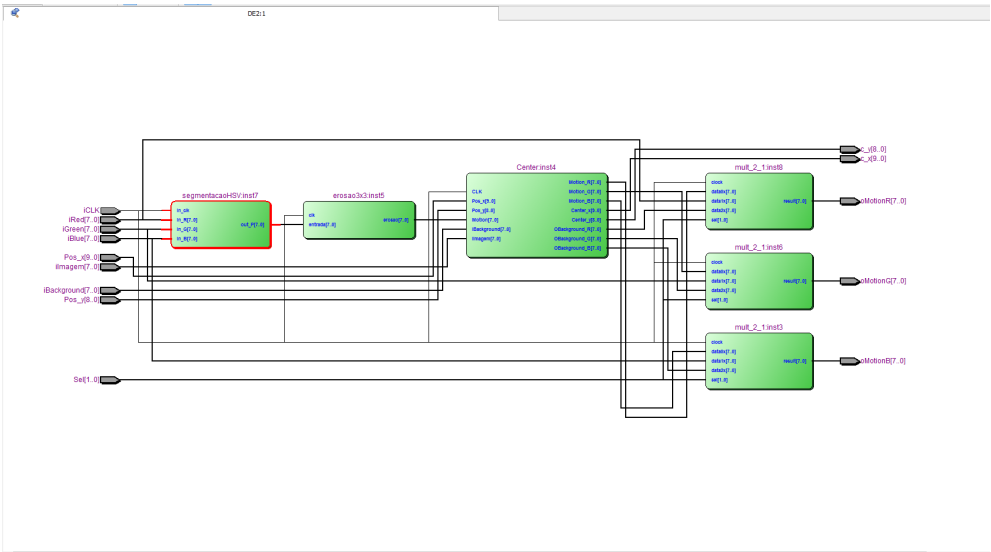

Figura 2.6: Sistema 1

de um sistema ainda maior que possui a extração de informações da câmera e do touchscreen do display, este não usado neste trabalho, extração de informações de um teclado conectado a FPGA pela porta PS2/2, usado como dispositivo, entre outros blocos, todos aproveitados do sistema original com exceção do bloco de PWM, que também foi desenvolvido neste trabalho. O sistema maior está na figura 2.7.

Nota-se no sistema desenvolvido que não foi usado o paralelismo espacial, pois a imagem é extraída pixel a pixel da câmera.

O novo bloco de segmentação foi desenvolvido usando as estratégias de segmentação por dis-

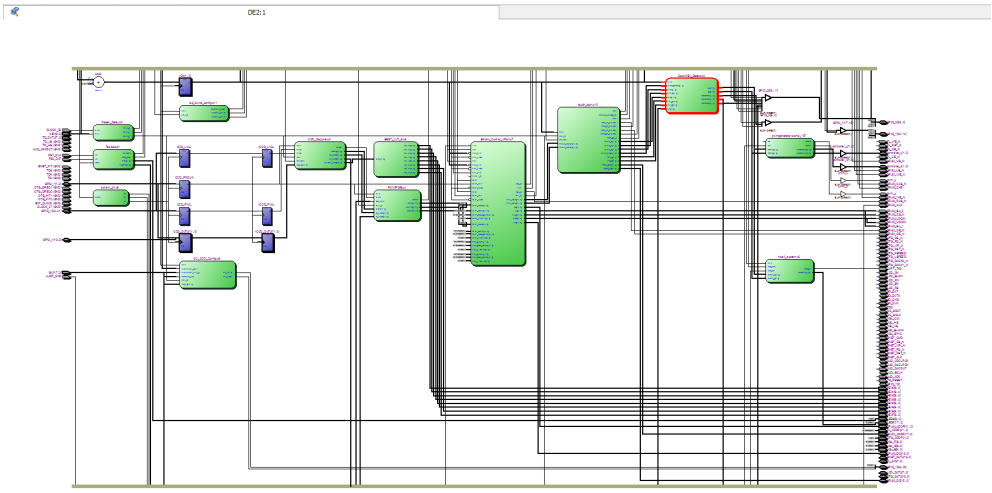

Figura 2.7: Sistema 2

tância euclidiana e por thresholding no espaço HSV.

Os esquemáticos disponíveis no anexo foram gerados através do Altera Quartus para as implementações feitas em Verilog e permitem a visualização em baixo nível de cada bloco individual. Porém, pela dificuldade em interpretar os blocos mais complexos, é recomendada a consulta aos códigos presentes no conteúdo do CD. A figura 2.8 mostra os blocos da segmentação por distância euclidiana e a figura 2.9 os blocos da segmentação no espaço HSV.

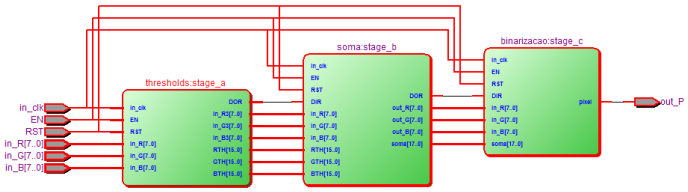

Figura 2.8: Estrutura em Pipeline da implementacão por segmentação euclidiana

A segmentação euclidiana consiste de uma etapa denominada *thresholds*, uma *soma* e outra *binarização*. Na etapa *thresholds*, a partir dos valores típicos das componentes da cor a qual se deseja segmentar, inserida por meio do código e das entradas **in_R**, **in_G** e **In_B**, com os componentes vermelho, azul e verde, são gerados as saídas **RTH**, **GTH** e **BTH** correspondentes às distâncias nos três eixos dos pixeis de entrada aos valores típicos. Também estão presentes as entradas **in_clk**, **EN** e **RST**, que se referem ao ciclo de relógio o habilitador e o reset, e a saída **DOR** que habilita o próximo estágio. Nota-se que as entradas são retrasmitidas para os próximos estágios para facilitar a verificação. Na etapa *soma*, é feito o cálculo da soma do quadrado das

distâncias que é armazenado na variável **outsoma**. A entrada **DIR** é a habilitação do estágio de pipeline ativa quando a saída **DOR** do estágio anterior estiver ativa. Na última etapa, *binarização*, é gerada a saída **pixel**, correspondente a um pixel da imagem binária gerada para uso dos blocos posteriores, que é um quando o quadrado da distância euclidiana calculado no bloco posterior é menor que um threshold específico.

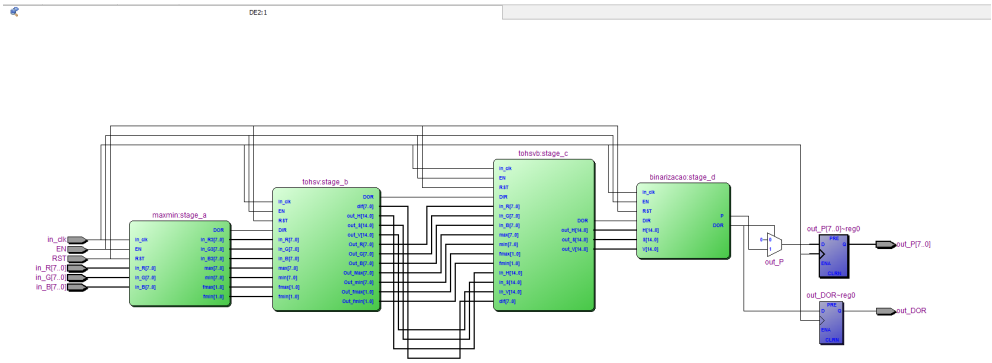

Figura 2.9: Estrutura em Pipeline da implementação no espaço HSV

Na segmentação feita no espaço HSV temos os estágios denominados *maxmin*, *tohsv*,*tohsvb* e *binarização*. Em *maxmin*, são gerados o maior e o menor valor das componentes vermelho,verde e azul nas saídas **max** correspondente ao máximo e **min** correspondente ao mínimo. As saídas **fmax** e **fmin** carregam a codificação indicando em qual componente ocorre o máximo e o mínimo. Esta codificação é 00 para vermelho, 01 para o verde e 10 para o azul. No estágio *tohsv* são gerados o numerador e o denominador do cálculo da componente H na operação de divisão presente na equação de transformação para o espaço HSV, presente na seção de introdução. O numerador é armazenado na variável **out_H** enquanto o denominador é armazenado na variável **dif**. São calculados os valores da componente S e V em **out_S** e **out_V**. No estágio tohsv_b é gerado o valor da componente H na nova saída **out_H** de acordo ainda com a equação apresentada na introdução. Na etapa *binarização*, é gerado o pixel da imagem binária **P**, se os valores de tonalidade, brilho e saturação estão nos valores desejados. Esta segmentação é feita principalmente na componente de tonalidade, eliminando ainda baixos valores de saturação que indicam cor acinzentada e valores muito altos ou baixos de brilho que indicariam cor branca ou preta.

2.3 Interface

Além da Pan-tilt e do kit eletrônico, foram utilizados dois motores de corrente contínua, cuja relação entre a tensão efetiva fornecida através de um sinal de PWM, vindo do FPGA, e a sua velocidade angular é proporcional. Esta relação pode ser obtida através do estudo da conversão de energia. Foram usados ainda pontes-H L298N que possibilitam o giro do motor nos dois sentidos, escolhidas adequadamente as variáveis de controle. Como as pontes-H precisam de entradas de 5V para seus pinos de controle e da tensão de entrada, tiveram de ser usados optoacopladores

4N25 que habilitariam uma tensão de 5V quando recebida do FPGA a tensão de 3,3V que ela pode fornecer. Esta estrutura que contêm ainda os resistores necessários ao seu funcionamento é conhecida e mostrada no anexo de esquemáticos

Esta implementação contudo não funcionou na prática durante o período em que foi feito este trabalho, impossibilitando a interface entre o FPGA e os motores do Pan-Tilt.

Capítulo 3

Resultados Experimentais

Este capítulo contém as implementações em MATLAB das estratégias de segmentação por cores, além dos resultados obtidos ao implementar o código Verilog no software Altera Quartus e em seguida o simular no software Modelsim. Os resultados da consequente implementação em FPGA, assim como estratégias para controle, também são abordados.

3.1 MATLAB

Antes de serem implementadas no FPGA, a segmentação por cores implementada em HSV, as segmentações por distância euclidiana e distância de Mahalanobis em RGB, todas gerando posteriormente uma imagem binária, foram simuladas no MATLAB e os resultrados obtidos foram os mostrados nas figuras 3.1 e 3.2 e 3.3 comparando as imagens originais e as obtidas pelas operações de segmentação descritas anteriormente:

Analisando as imagens, é observado que para a primeira imagem original, bola amarela, a segmentação no espaço HSV classifica uma região perto do contorno erroneamente como sendo parte do objeto de interesse. Porém, para cálculos posteriores do centro de imagem, a segmentação está vantajosa pelo fato de ter um menor "buraco"na área interior da imagem, ou seja um menor número de píxeis que foram classificados incorretamente como não pertencendo à imagem. Para a segunda imagem original, bola azul, afirma-se que a segmentação no espaço HSV ficou bem fiel à imagem original, enquanto as demais segmentações feitas no espaço RGB apresentaram uma disformidade deixando de classificar parte do objeto de estudo. Finalmente, na terceira segmentação, almofada com tachas, há apenas um caso em que a separação das cores foi feita com sucesso, intencionava-se separar o verde, este caso acontece no espaço HSV. Conclui-se esta seção afirmando que a segmentação no espaço HSV apresenta para os casos testados um melhor resultado e portanto este espaço será posteriormente priorizado na implementação. Estes resultados são subjetivos e dependem dos thresholds escolhidos, porém foram testados alguns thresholds e aqui se afirma que foram feitas boas escolhas de thresholds para a segmentação. Os thresholds são apresentados nas tabelas 3.1 e 3.2 e estão em uma escala de zero a duzentos e cinquenta e cinco.

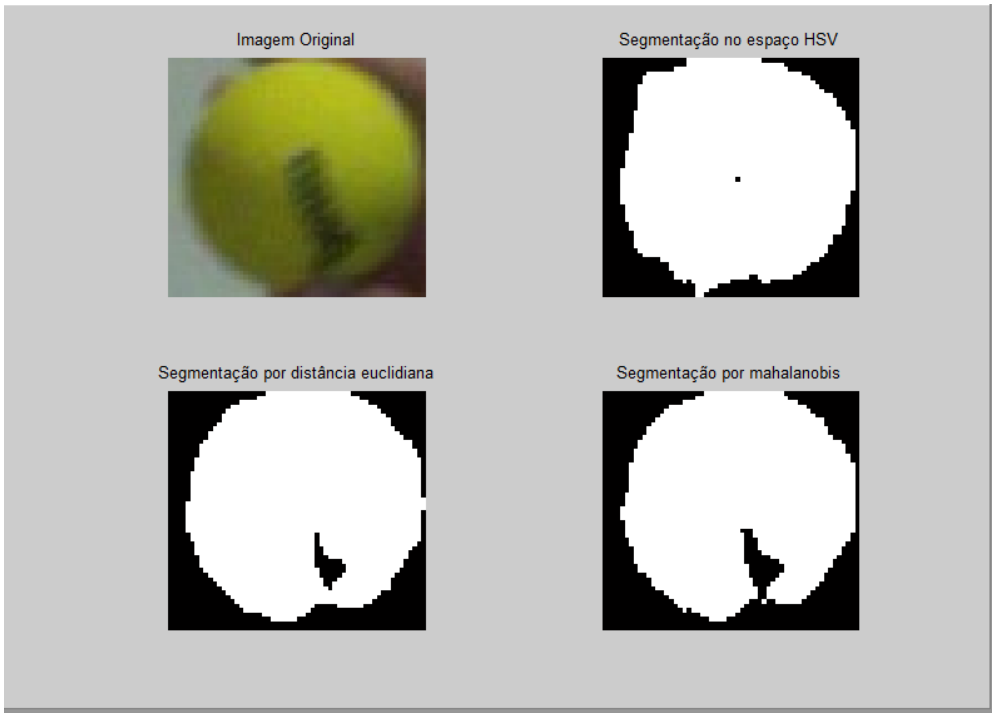

Figura 3.1: Segmentação por cores no MATLAB para o primeiro objeto

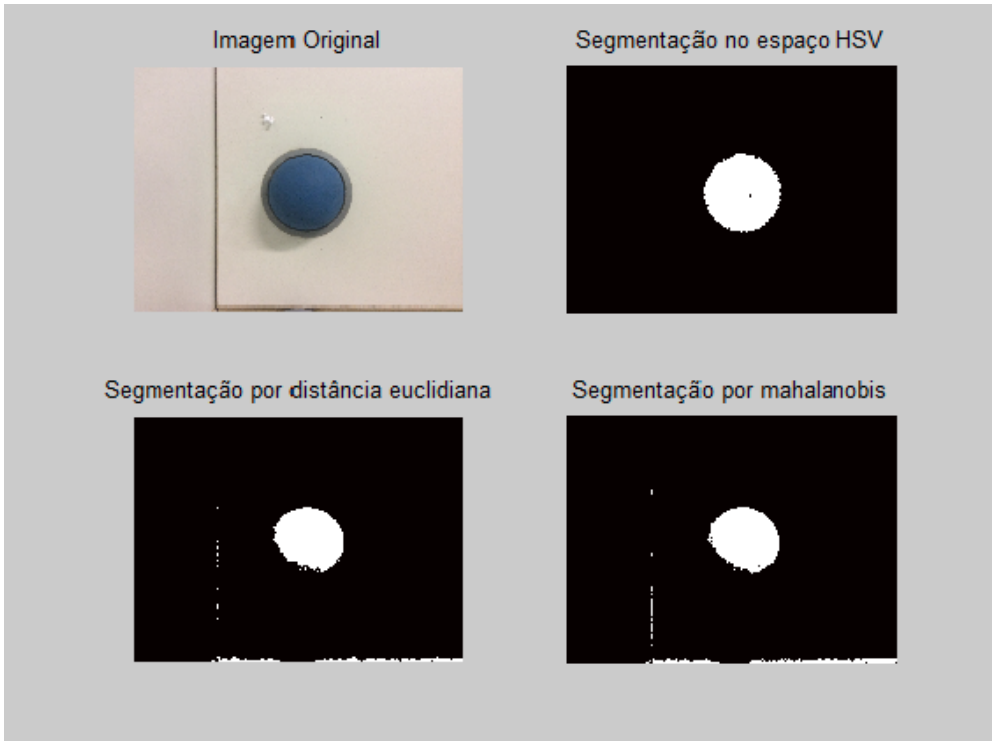

Figura 3.2: Segmentação por cores no MATLAB para o segundo objeto

	Bola amarela	Bola azul
Mínimo valor da componente de tonalidade no espaço HSV	35	135
Máximo valor da componente tonalidade no espaço HSV	55	155
Valor típico da componente Vermelho	111	59
Valor típico da componente Verde	140	84
Valor típico da componente Azul	22	124
Máxima distância euclidiana para classificação	5	2
Máxima distância de Mahalanobis para classificação	0.5	0.002

Tabela 3.1: Valor das variáveis na simulação

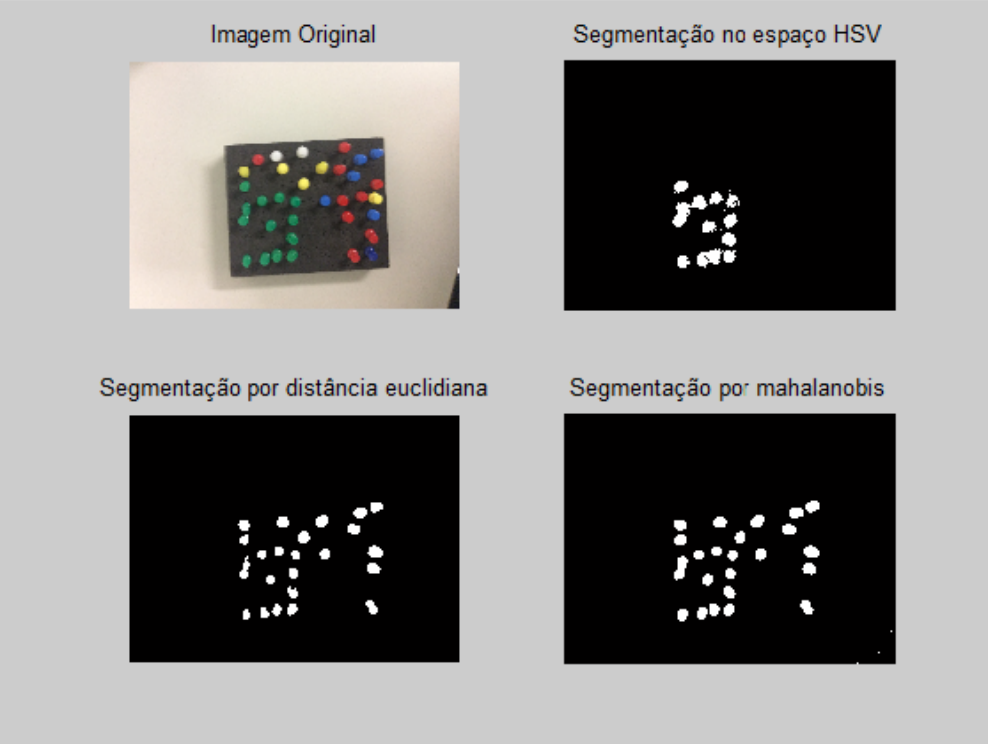

Figura 3.3: Segmentação por cores no MATLAB para o terceiro objeto

3.2 Implementação em Verilog

Em seguida, foram implementadas tanto a segmentação euclidiana quanto a HSV em verilog para simulação.

A análise funcional obtida com o auxílio do software Modelsim, a partir da compilação gerada pelo software Altera Quartus, foi a mostrada na imagem 3.4 para a segmentação por distância euclidiana. A seguinte, na figura 3.5, foi obtida para a segmentação HSV.

Infelizmente, para ambas as simulações, não houve ocorrências de uma saída um para o pixel da

	Almofada com tachas
Mínimo valor da componente de tonalidade no espaço HSV	93
Máximo valor da componente tonalidade no espaço HSV	113
Valor típico da componente Vermelho	42
Valor típico da componente Verde	143
Valor típico da componente Azul	101
Máxima distância euclidiana para classificação	5
Máxima distância de Mahalanobis para classificação	0.005

Tabela 3.2: Valor das variáveis na simulação:continuação

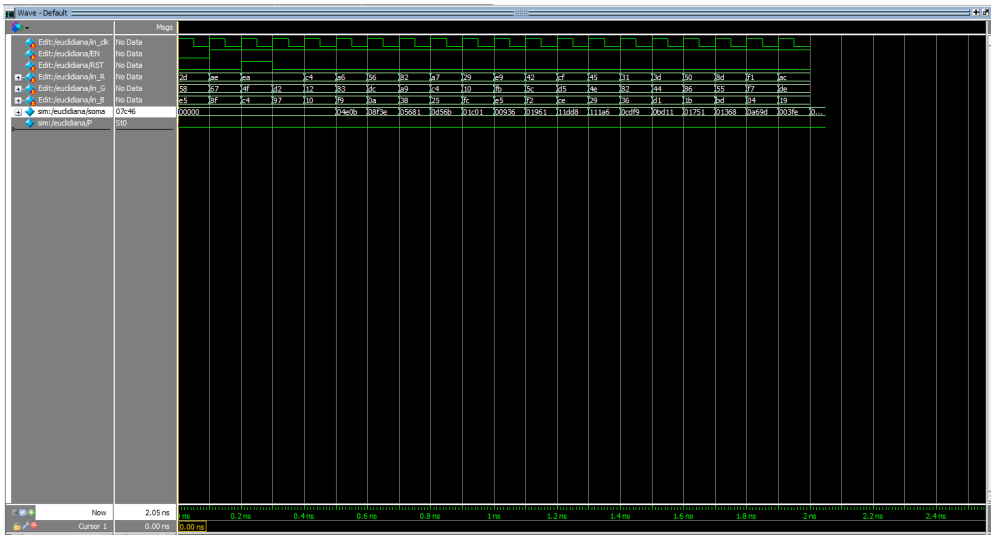

Figura 3.4: Simulação da segmentação por distância euclidiana

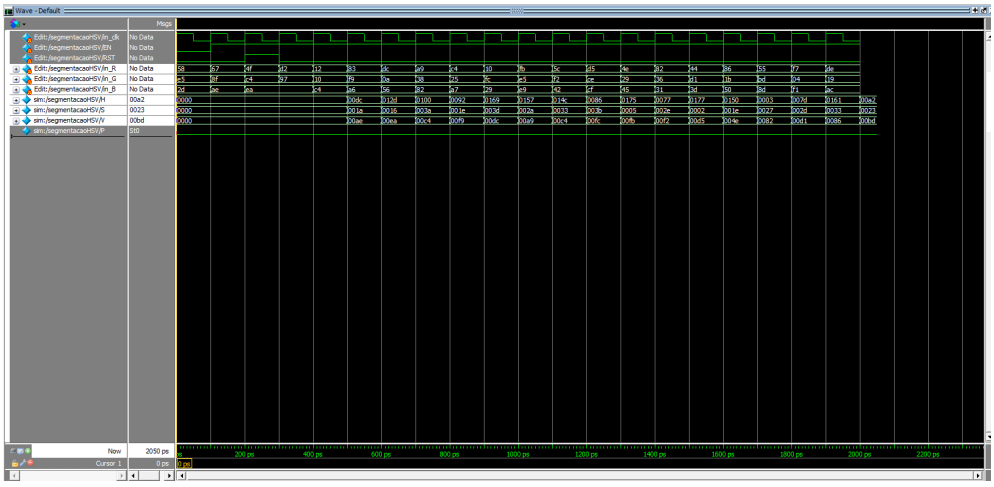

Figura 3.5: Simulação da segmentação no espaço HSV

imagem binária abreviado nestas por **P**. Isto ocorreu devido às variáveis de entradas,representadas através das abreviações **in_R**, **in_G** e **in_B**, geradas aleatoriamente, não serem aquelas que produziriam píxeis classificados como um. Porém, é possível verificar o funcionamento através da

checagem das saídas **H**, **S** e **V** para a simulação HSV e da saída soma para a euclidiana que corresponde à soma dos quadrados das componentes de cor.

As demais siglas utilizadas fora **in_clk** para o ciclo de relógio, **EN** para o habilitador e **RST** para o reset.

Os circuitos lógicos equivalentes à implementação em Matlab poderam ser gerados através do software MATLAB e encontram-se no capítulo de esquemáticos.

3.3 Resultados

Verificado o funcionamento das implementações em Verilog na simulação, o bloco foi inserido no conjunto de blocos funcionais do trabalho referenciado na bibliografia(Ferreira,2012). Não foram apresentados problemas de compatibilidade e a partir da imagem binária gerada pelos códigos em verilog, o resto do sistema não teve problemas em encontrar o centro de massa. Com isto é possível verificar o funcionamento do código a partir da imagem fotograda do Display mostrada na figura 3.6:

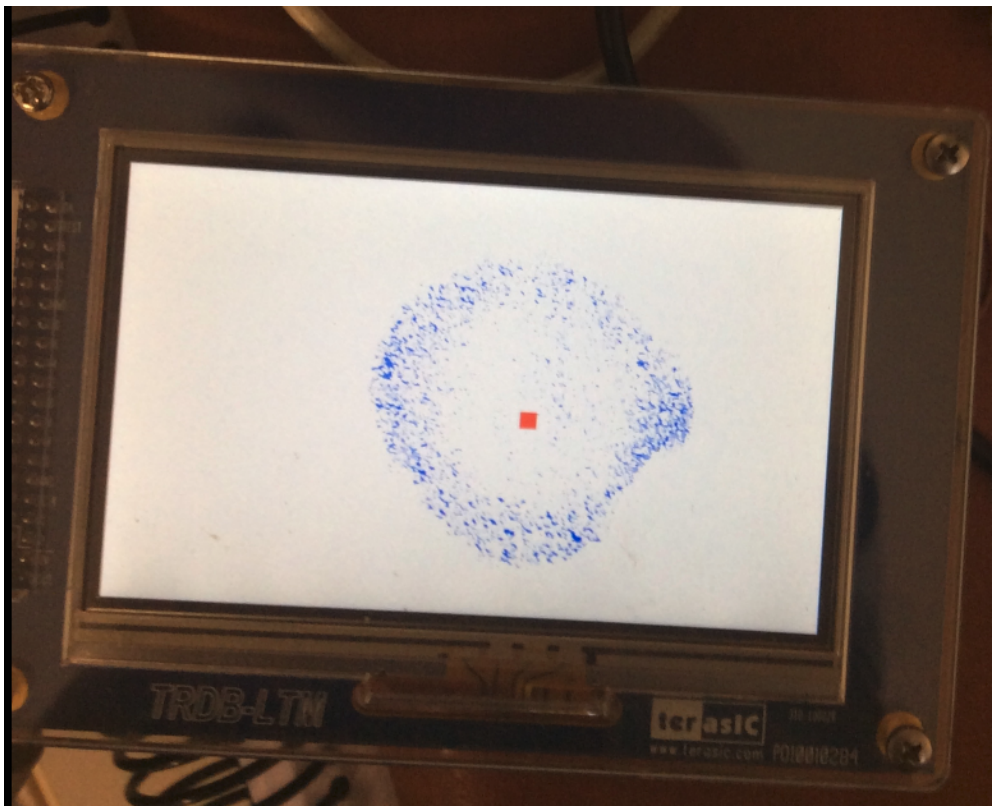

Figura 3.6: Segmentação por cores exibida no display

E na figura 3.7 temos a foto do objeto segmentado, neste caso verde claro.

Outras informações importantes puderam ser extraídas da compilação no software Altera Quartus a tabela 3.3 mostra as caractristicas práticas do bloco de segmentação no espaço HSV implementado isoladamente, sendo possível fazer um comparativo com as caracteristicas práticas do

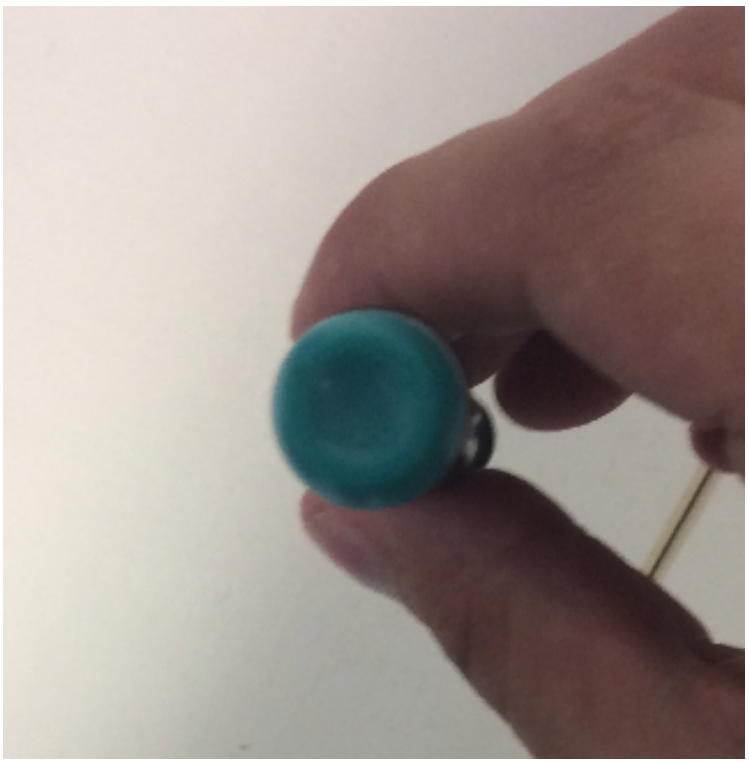

Figura 3.7: Objeto usado para segmentação

total logic elements	13(<1%)
total combinational functions	3(<1%)
dedicated logic registers	13(<1%)
Total Memory Bits	0(0%)
Total Thermal Power Dissipation	33.1 mW
Fmax	1242.24 MHz
Restricted Fmax	420.17 MHZ

Tabela 3.3: Caracteriticas práticas do bloco de segmentação no espaço HSV

sistema como um todo na tabela 3.4.

Os parâmetros **Fmax** e **Restricted Fmax** são diferentes no bloco de segmentação, pois a frequência máxima permitida pela lógica do circuito supera a permitida pela transição lógica dos elementos físicos do FPGA. Portanto o circuito pode trabalhar no limite físico do FPGA. É interessante observar ainda que, conforme apresentado no capítulo dois, o máximo ciclo de relógio gerado pela Altera DE2 é de 50 MHZ, ciclo este que pode ser usado para o projeto, pois a implementação que trabalha na menor frequência, no caso a do sistema como um todo pode apresentar Fmax de 77MHz. Analisando as informações relacionadas ao número de elementos e

total logic elements	7873(24%)
total combinational functions	6556(20%)
dedicated logic registers	4755(11%)
Total Memory Bits	363,448(75%)
Total Thermal Power Dissipation	431.03 mW
Fmax	77 MHz
Restricted Fmax	77 MHZ

Tabela 3.4: Caracteriticas práticas do sistema como um todo

aos memory bits, é possível observar que o sistema pode ser implementado sem exceder os limites do chip ciclone II do FPGA usado para o projeto.

3.4 Medidas de erro

É interessante avaliar qual é o erro obtido no cálculo do centro de massa. Por isto, foi feito um objeto retangular de cartolina, com o centro assinalado pela intersecção das diagonais que foram hachuradas na cartolina de forma que pudessem ser discriminadas na imagem presente no display. Foram feitas quinze medidas do erro médio e do desvio padrão do erro em três situações de iluminação. Embora tenham sido variadas as posições, tanto da câmera que produz a imagem enviada ao display, quanto da câmera digital usada para fotografar o display, tal variação não compromete as medições, uma vez que os valores extraídos de erro foram normalizados ao se dividir pelo contorno medido, se tornando assim uma varável adimensional. A tabela 3.5 mostra os dados obtidos para cada situação e os dados finais; a figura 3.8 mostra o objeto usado e as figura 3.9, 3.10 e 3.11 expõe uma destas medições em cada situação. As fotografias feitas de todas as medições podem ser encontradas no conteúdo do CD.

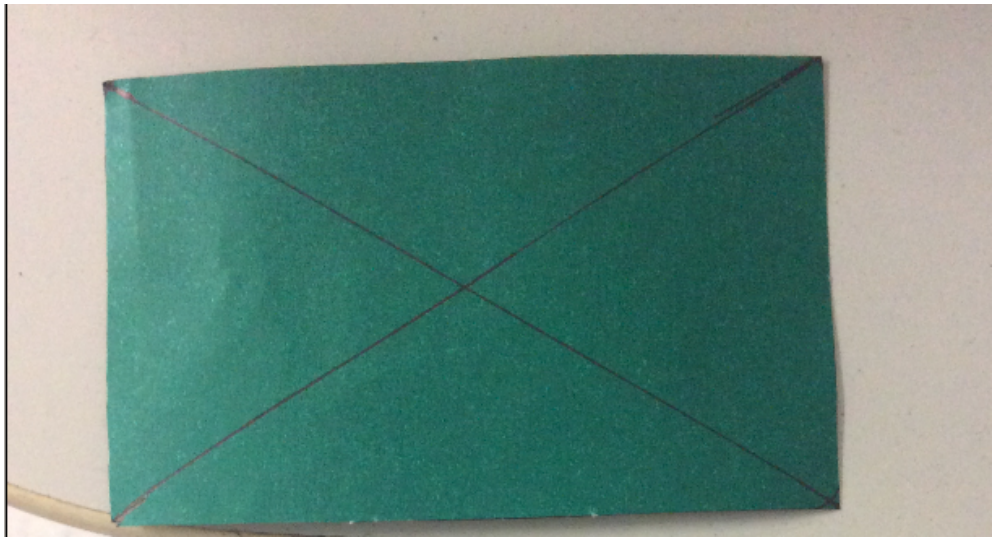

Figura 3.8: Objeto de cartolina usado para medição do erro

Situação de iluminação	erro médio	mediana	desvio padrão
primeira	0.03265	0.03497	0.00552
segunda	0.05532	0.05556	0.00408
terceira	0.08561	0.08763	0.0108
total	0.05786	0.05556	0.0231

Tabela 3.5: Extração de informações realacionadas ao erro

Situação de iluminação	erro médio	mediana	desvio padrão
primeira	0.02867	0.02941	0.01505

Tabela 3.6: Extração de informações realacionadas ao erro em uma distância mais afastada

As medidas de erro apresentaram uma grande variação entre três situações de iluminação. Observa-se que a maior discrepância entre o centro do objeto e o centro medido se deu na situação um pouco mais escura. Em seguida, foi feito outro experimento de medição do erro, desta vez a câmera conectada ao FPGA foi presa afastada da imagem, para que pudessem ser feitas mais variações tanto na posição quanto na rotação do objeto. Novamente foram extraídas quinze imagens, porém apenas uma situação de iluminação foi avaliada. A tabela 3.6 coleta os dados deste último experimento e a figura 3.12 mostra uma das imagens extraídas para o cálculo.

3.5 Estratégias de Controle

A partir do funcionamento do cálculo do centro de massa, foi implementado um bloco de PWM com maior ciclo de trabalho, razão entre o tempo de duração da onda, ou seja o tempo em que a saída está ativa, e o período da onda conforme uma maior distância entre o centro do objeto de interesse e o centro da imagem. é sugerido um ciclo proporcional a uma exponencial suavizando a resposta. Com isto a relação para o ciclo de trabalho é da forma:

$$dc = Ke^{d} \tag{3.1}$$

dc representa o ciclo de trabalho e d a distância entre o centro da imagem e o centro do objeto.

K é ajustado empiricamente para ser pequeno, porém suficiente para vencer a inércia do sistema.

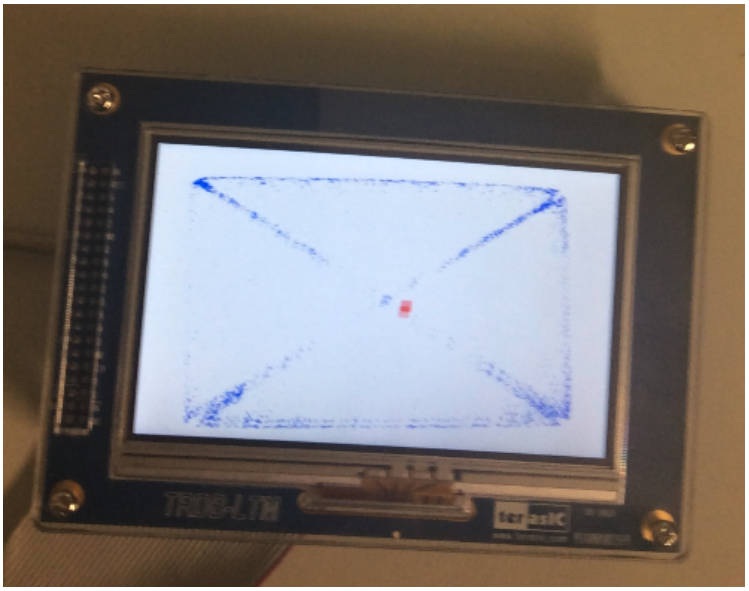

Figura 3.9: Medição do erro na primeira situação de iluminação

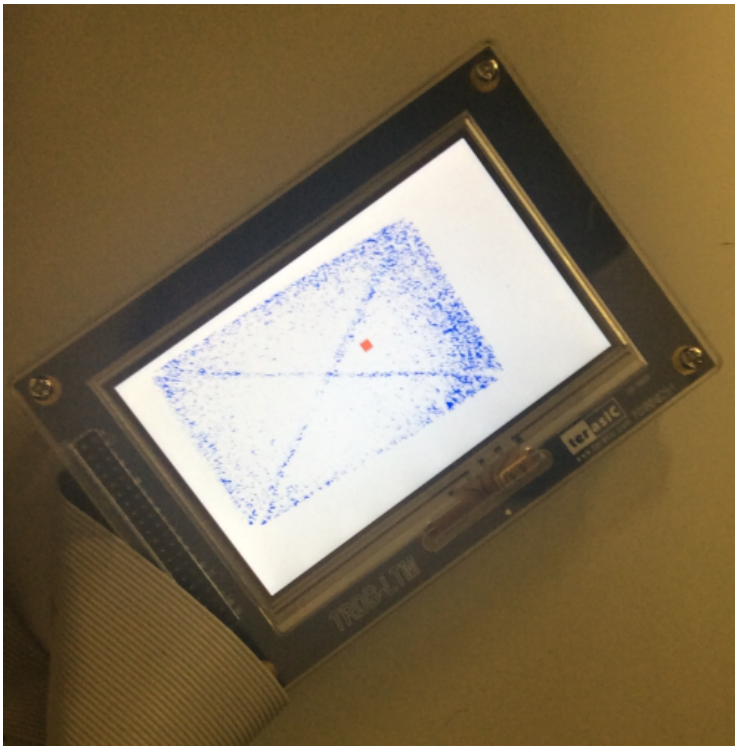

Figura 3.10: Medição do erro na segunda situação de iluminação

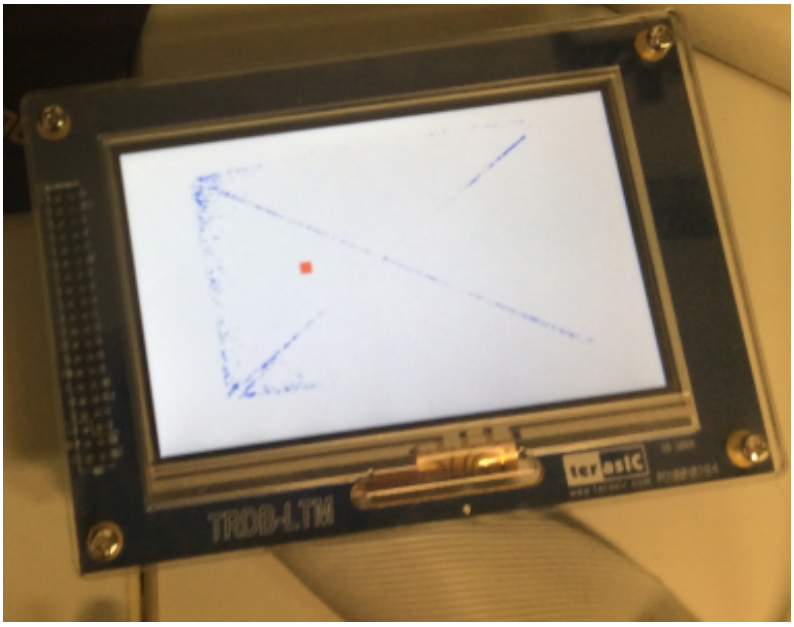

Figura 3.11: Medição do erro na terceira situação de iluminação

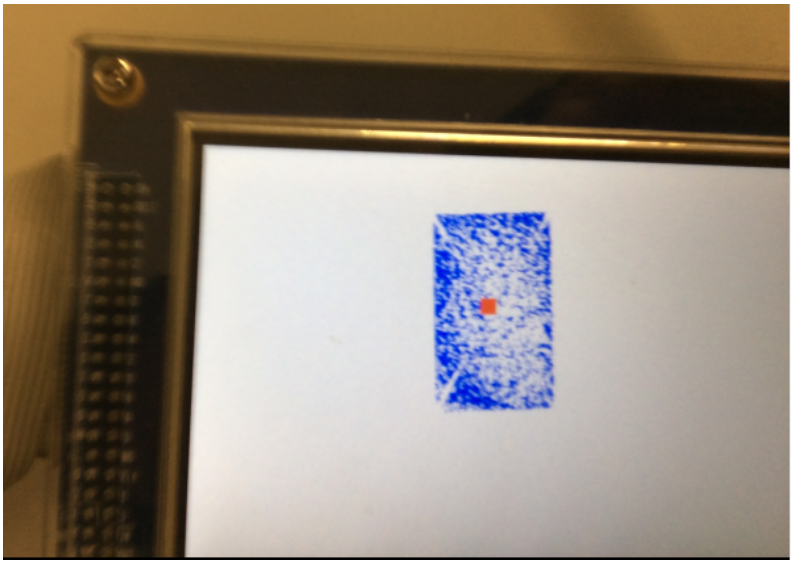

Figura 3.12: Medição do erro em uma distância mais afastada

Capítulo 4

Conclusões e Implementações Futuras

4.1 Conclusões

Primeiramente, é interessante observar as vantagens do uso de segmentação por cores. Em comparação à estratégia de subtração de fundos proposta no outro projeto, ela pode ser extendida a qualquer tipo de fundo. Neste trabalho isto é útil, pois existe mobilidade com dois graus de liberdade para a câmera, e tal utilidade pode ser ainda aproveitada quando o algoritmo for adaptado para outras aplicações em robótica. A desvantagem está na inaptidão em ser extendida a qualquer objeto, se limitando aos objetos da cor escolhida para segmentação. A implementação em hardware explora o paralelismo, ainda que não tenha sido aproveitado o paralelismo espacial da imagem sendo a informação enviada pixel a pixel. Com o código verilog gerado, obteve-se uma frequência de operação máxima de setenta e sete megahertz, maior que a máxima fornecida pela DE2 e usada na aplicação que é de cinquenta megahertz. O modelo de Pan-Tilt apresenta bom funcionamento mecânico, porém sua interface elétrica com o FPGA ainda não está em funcionamento.

4.2 Implementações futuras

Para a melhoria do trabalho apresentado, é necessário primeiramente a conclusão do mesmo atingindo os objetivos propostos. Para isto deverá ser refeito o circuito eletrônico de interface de modo que garanta o seu funcionamento. Em seguida devem ser feitos testes com as estratégias de controle sugeridas. Como implementação futura, é recomendada a extração de informação tridimensional do objeto, sabendo com isto a sua profundidade, ou seja a sua distância para o pan-tilt. O sistema conforme foi projetado não responderá de forma diferente à detecção do objeto uma vez conhecida sua dimensão de profundidade, porém tal informação pode ser útil para a mesma implementação embarcada do código interfaceada a um outro sistema mecânico capaz de interagir com o objeto, como por exemplo um braço robótico. Mesmo assim, o pan-tilt foi projetado para o encaixe de duas câmeras, requisito necessário para extrair informações de profundidade. Um artigo que propõe uma metodologia de triangulação é aqui referenciado(Ferreira, Fortaleza, Llanos e Mori,2012).

Capítulo 5

Referências Bibliográficas

4N25 Datasheet. Disponível em <http://pdf.datasheetcatalog.com/datasheet/motorola/4N26.pdf>. Acesso em 27 dez.2013

Altera DE2 Development and Education Board User Manual. Disponível em <http://www.ee.bgu.ac.il/ adcomplab/ niosII/Boards/DE2_UserManual.pdf>. Acesso em 27 dez.2013

Alvy Ray Smith Bio. Disponível em < http://alvyray.com/Bio/BioCV.htm#Software >. Acesso em 27 dez.2013

Amdahl,G.M.(1967) Validity of the single processor approach to achieving large scale computing capabilities. AFIPS Spring Joint Computer Conference, Atlantic City, New Jersey, USA, vol. 30 (18-20 April, 1967) pp. 483-485. doi: 10.1145/1465482.1465560

Athanas, P.M. and Abbot, A.L. (1995) Real-time image processing on a custom computing platform. IEEE computer, 28(2),16-25. doi:10.1109/2.347995

Bailey, D.G (2011) Design for Embedded Image Processing on FPGAs, 1st edn, John Wiley & Sons

Catsoulis, J. (2005) Designing Embedded Hardware, 2nd edn, O'Reilly

Dougherty, E.R. and Laplante,P.A (1985), Introdution to Real Time imaging, The International Society for Optical Engineering, Washington

Duff, M.J.B (2000) Thirty years of parallel image processing, in Vector and Parallel Processing(VECPAR 2000), Porto, Portugal (21-23 June,2000), Lecture Notes in Computer Science, vol. LNCS 1981, Springer, pp. 419-438. doi:10.1007/3-540-44942-6_35

Ferreira, C. S. (2012) Implementação do Algoritmos de Subtração de Fundo para Detecção de Objetos em Movimento, Usando Sistemas Reconfiguráveis. Dissertação de Mestrado em Sistemas Mecatrônicos, Publicação ENM.DM-48A/12, Departamento de Engenharia Mecânica, Universidade de Brasília, Brasília, DF, 94p.

Ferreira, C.S., Fortaleza, E.,Llanos, C.H. e Mori, J.Y. (2012) Development of a Stereo Vison Measurement Architeture for an Underwater Robot, Mechanical Engineering Department, Unive-

sity of Brasilia, Brasilia, Brazil

Ford, A. and Roberts, A.: HSV: Hue Saturation Value. In: The Colour (color) Equations Document. 1994-1996, acessed on 30. August 2006

Gonzalez, R.C. and Woods (2002) R.E Digital Image Processing, 2nd edn, Prentice Hall

Harliack, R.M. and Shaphiro, L.G. (1991) Glossary of computer vision terms. Pattern Recognition, 24(1),69-93. doi:10.1016/0031-3203(91)90117-N

HSV Colour Space avaliable on <http://de.wikipedia.org/wiki/HSV-Farbraum>, acessed on 28. January, 2014

Kuon, I. and Rose, J. (2006) Measuring the gap FPGAs and ASICs. International Symposium on Field Programmable Gate Arrays, Monterey, California, USA (22-24 Febrary 2006) pp.21-30. doi: 10.1145/1117201.1117205

L298N Datasheet. Disponível em <http://pdf.datasheetcatalog.com/datasheet/SGSThomsonMicroelectronic Acesso em 27 dez.2013

Leong P.H.W. (2008) Recent Trends in FPGA architetures and applications. IEEE International Symposium on Eletronic Design, Test and Applications(DELTA 2008), Hong Kong(23-25 January), pp.137-141. doi:10.1109/DELTA.2008.14

Offen, R.J. (1985) VLSI Image Processing, Collins, London

Pedrini, H. e Schwartz W. R. (2008) Análise de Imagens digitais: Princípios, algoritmos e aplicações – São Paulo: Thomson Learning

Randen, T. and Husoy, J.H (1999) Filtering for texture classification: a comparative study. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21 (4), 291-310. doi: 10.1109/34.76121

Schaffer, G. (1984) Machine Vision: a sense for computer integrated manufacturing. American Machnist, 128(6), 101-129

Terasic TRDB_D5M Manual. Disponível em <http://courses.cs.washington.edu/courses/cse467/08au/labs /Resources/TRDB_D5M_UserGuide.pdf>. Acesso em 27 dez.2013

Terasic TRDB_LTM Manual. Disponível em <http://ebookbrowsee.net/trdb-ltm-userguide-v1-22-110907-pdf-d138134373>. Acesso em 27 dez.2013

ANEXOS

I. FOTOS ADICIONAIS

As fotos presentes neste capítulo mostram a construção prática do pan-tilt, sistema de dois graus de liberdade.

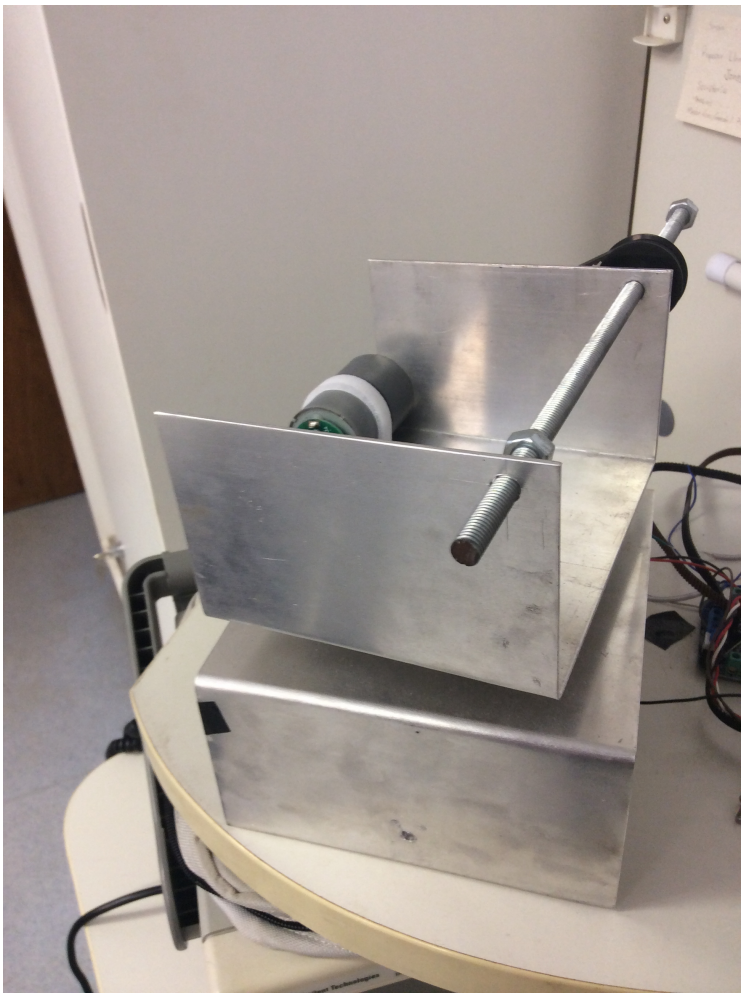

Figura I.1: Construção do Pan-Tilt: Foto 1

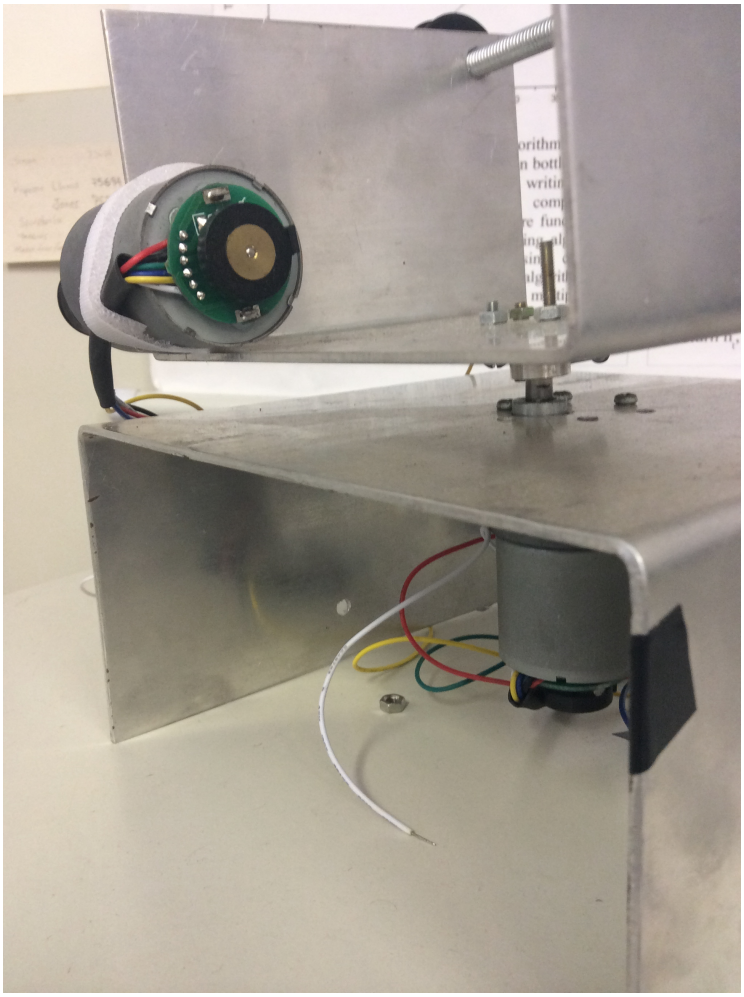

Figura I.2: Construção do Pan-Tilt: Foto 2

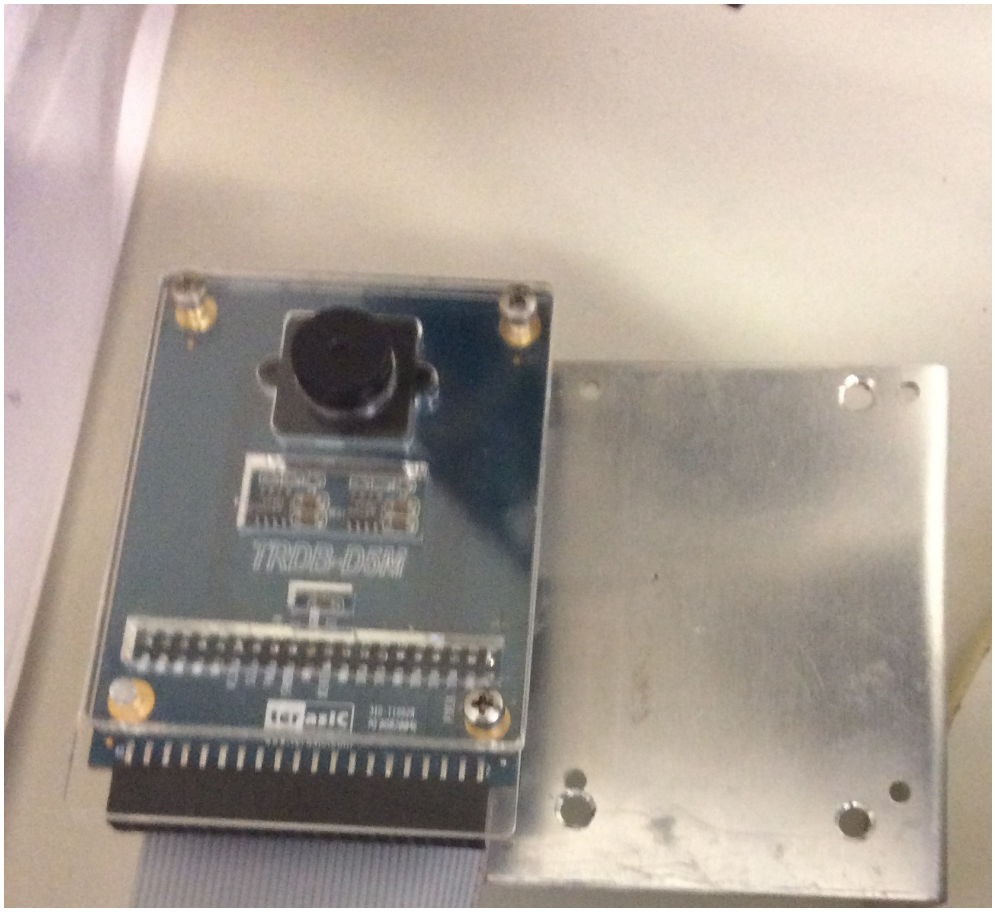

Figura I.3: Construção do Pan-Tilt: Foto 3

II. DIAGRAMAS ESQUEMÁTICOS

A figura a seguir mostra a representação do circuito eletrônico de interface entre o FPGA e o motor. Na implementação prática se tentou usar um shield contendo o CI l298n como ponte-h. Porém, pela ausência deste componente no software CircuitMaker usado para fazer o projeto, foi feita uma construção do mesmo usando um circuito transistorizado.

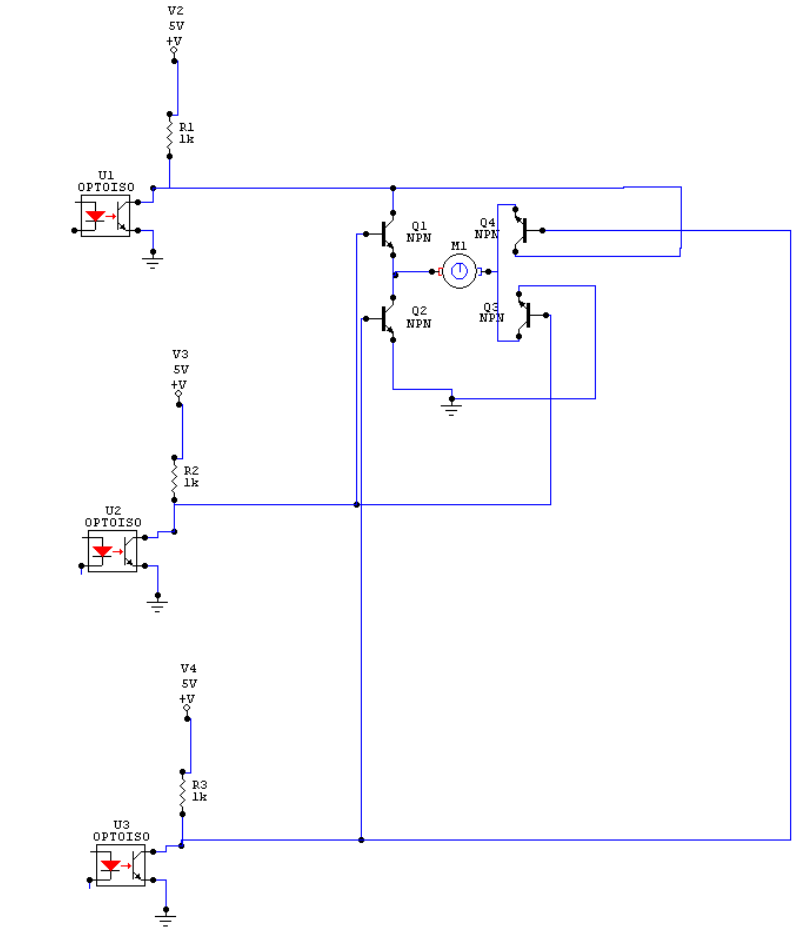

Figura II.1: Circuito Eletrônico

Os esquemáticos nas figuras seguintes foram gerados através do Altera Quartus para as implementações feitas em Verilog.

As figuras II.2, II.3 eII.4 correspondem às implementações por distância euclidiana e as figuras II.5, II.6, II.7 e II.8 correspondem às implementações de segmentação no espaço HSV.

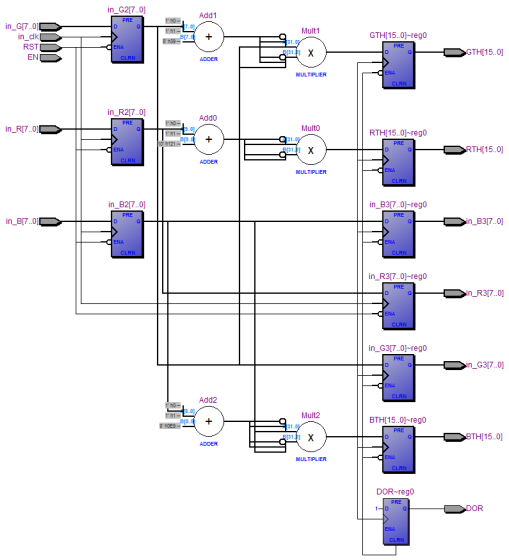

Figura II.2: Elemento thresholds:stage_a da implementacão por segmentação euclidiana

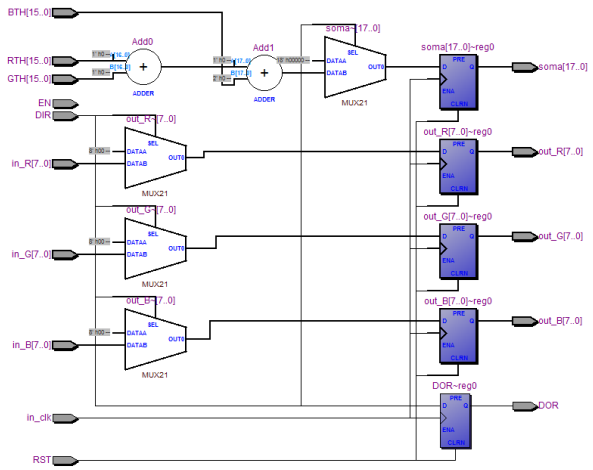

Figura II.3: Elemento soma:stage_b da implementação por segmentação euclidiana

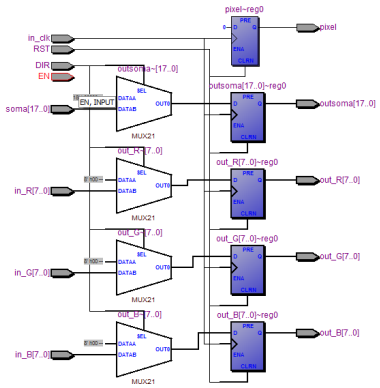

Figura II.4: Elemento biarização:stage_c da implementação por segmentação euclidiana

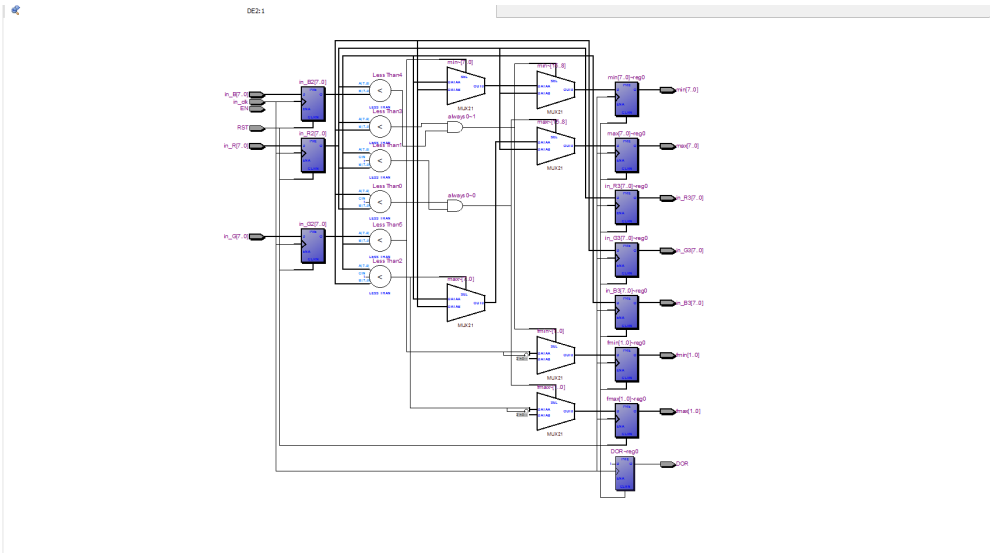

Figura II.5: Elemento maxmin:stage_a da implementação no espaço HSV

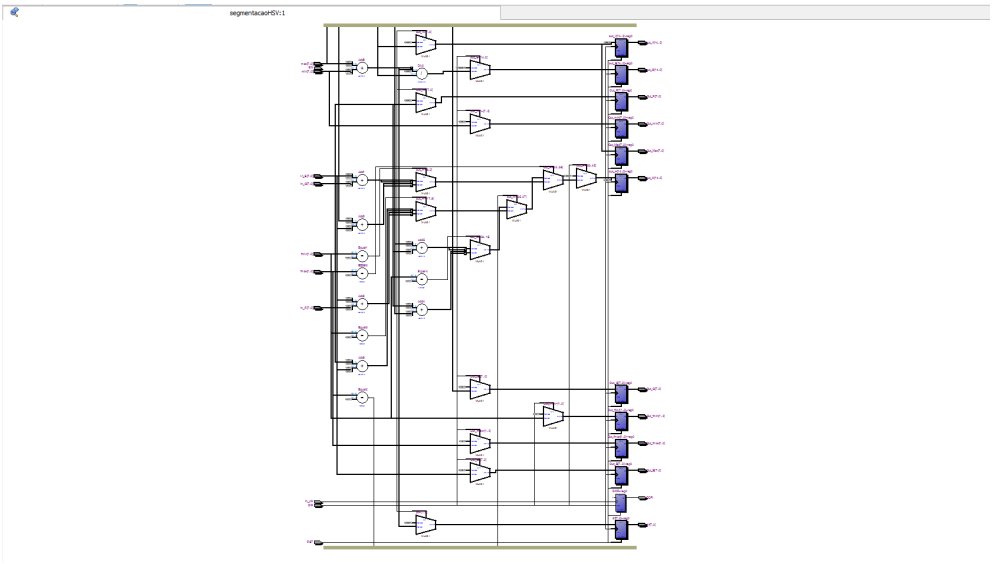

Figura II.6: Elemento tohsv:stage_b da implementação no espaço HSV

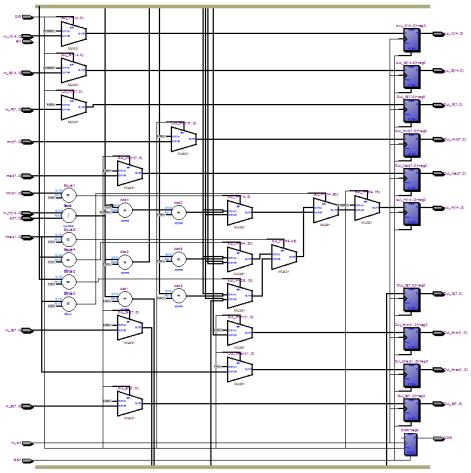

Figura II.7: Elemento tohsvb:stage_c da implementação no espaço HSV

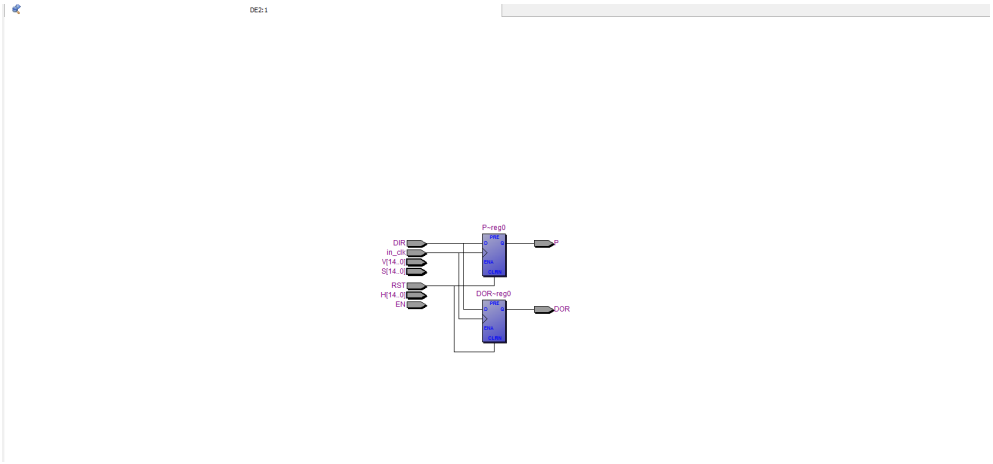

Figura II.8: Elemento binarização:stage_d da implementação no espaço HSV

III. DESCRIÇÃO DO CONTEÚDO DO CD

Como parte do conteúdo do CD, têm-se os códigos Verilog usados para a implementação no FPGA das segmentações no espaço HSV e por distância euclidiana no espaço RGB. O script MATLAB que compara as duas segmentações anteriores, comparando ainda com a segmentação por distância de Mahalanobis no espaço RGB, também está presente. Por último, são acrescentadas as sessenta imagens para os experimentos de medição de erro.