INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CÂMPUS FLORIANÓPOLIS

DEPARTAMENTO ACADÊMICO DE METAL MECÂNICA

BACHARELADO EM ENGENHARIA MECATRÔNICA

ALAN DEIVIS VALMORBIDA

DESENVOLVIMENTO DE UM SISTEMA DE PROCESSAMENTO DE IMAGENS PARA INSPEÇÃO AUTOMATIZADA UTILIZANDO FPGA

FLORIANÓPOLIS, 2018.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA

CÂMPUS FLORIANÓPOLIS

DEPARTAMENTO ACADÊMICO DE METAL MECÂNICA

BACHARELADO EM ENGENHARIA MECATRÔNICA

ALAN DEIVIS VALMORBIDA

DESENVOLVIMENTO DE UM SISTEMA DE PROCESSAMENTO DE IMAGENS PARA INSPEÇÃO AUTOMATIZADA UTILIZANDO FPGA

Trabalho de Conclusão de Curso submetido ao Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina como parte dos requisitos para obtenção do título de Bacharel em Engenharia Mecatrônica.

Professor Orientador: Maurício Edgar Stivanello, Dr. Eng.

FLORIANÓPOLIS, 2018.

Ficha de identificação de obra elaborada pelo autor.

Valmorbida, Alan Deivis
Desenvolvimento de um Sistema de Imagens para Inspeção Automatizada Utilizando FPGA / Alan Deivis Valmorbida; Orientação de Maurício Edgar Stivanello. – Florianópolis, SC, 2018.
89 p.

Trabalho de Conclusão de Curso (TCC) – Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina, Câmpus Florianópolis. Bacharelado em Engenharia Mecatrônica.
Departamento Acadêmico de Metal Mecânica.
Inclui Referências.

1. Sistema de visão computacional. 2. Inspeção automatizada. 3. FPGA. 4. Rotulação de componentes conexos.

I. Stivanello, Maurício Edgar. II. Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina. III. Desenvolvimento de um Sistema de Imagens para Inspeção Automatizada Utilizando FPGA

DESENVOLVIMENTO DE UM SISTEMA DE PROCESSAMENTO DE IMAGENS PARA INSPEÇÃO AUTOMATIZADA UTILIZANDO FPGA

ALAN DEIVIS VALMORBIDA

Este trabalho foi julgado adequado para a obtenção do Título de Bacharel em Engenharia Mecatrônica e aprovado na sua forma final pela banca examinadora do Curso de Bacharelado em Engenharia Mecatrônica do Instituto Federal de Educação, Ciência e Tecnologia de Santa Catarina.

Florianópolis, 12 de Dezembro de 2018

Banca Examinadora:

_____

Maurício Edgar Stivanello, Dr. Eng. (orientador)

_____

Francisco Edson Nogueira de Melo, Me. Eng.

_____

Francisco Rafael Moreira da Mota, Dr. Eng.

INSTITUTO FEDERAL
SANTA CATARINA

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS

DECLARAÇÃO DE FINALIZAÇÃO DE TRABALHO DE CURSO

Declaro que o(a) estudante **ALAN DEIVIS VALMORBIDA**, matrícula n° **141000064-8**, do Curso de Engenharia Mecatrônica, defendeu o trabalho intitulado **DESENVOLVIMENTO DE UM SISTEMA DE PROCESSAMENTO DE IMAGENS PARA INSPEÇÃO AUTOMATIZADA UTILIZANDO FPGA**, o qual está apto a fazer parte do banco de dados da Biblioteca Hercílio Luz do Instituto Federal de Santa Catarina, Campus Florianópolis.

Florianópolis, 12 de dezembro de 2018.

Prof. Orientador do TCC: Maurício Edgar Stivanello

AGRADECIMENTOS

Aos meus pais Gilmar e Lucia Valmorbida, por me criarem e educarem, muito do que sou hoje devo a eles.

Ao professor Mauricio Stivanello, por toda a orientação fornecidas para o desenvolvimento do presente trabalho e também pelos ensinamentos durante toda a graduação e bolsa de pesquisa.

Aos professores membros da banca Francisco Edson Nogueira de Melo e Francisco Rafael Moreira da Mota, pelas contribuições e melhoramentos sugeridos ao trabalho.

Aos companheiros de laboratório, pela agradável convivência e em especial a Lucas Martins pelo auxílio no desenvolvimento de algumas funcionalidades.

A minha namorada Ana Maria Navarro Barbosa, pelo companheirismo, apoio e incentivo em todos os momentos.

Aos meus amigos Ezamilton Pazzette e Elcio Pazzette, pela convivência, apoio e conselhos.

Aos amigos formados durante a curso, especialmente a Gabriel Dal Ponte e Arthur Kretzer, pelos momentos de distração e alegria.

A todos os professores e funcionários do IFSC que contribuíram com a minha formação.

“O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis. ”

(José de Alencar)

RESUMO

O setor industrial tem buscado cada vez mais a utilização de sistemas de inspeção automatizada, seja para a substituição da inspeção realizada por operários ou para implantações em linhas ainda não inspecionadas. Esta busca é explicada pelo fato da inspeção automatizada conseguir conferir uma maior qualidade aos produtos, uma vez que tem maiores taxas de acerto e por permitir maiores velocidades de inspeção do que as alcançadas por inspeção realizada por operadores humanos. Normalmente os sistemas desenvolvidos para esse fim são baseados em plataformas de propósito geral (computadores), que muitas vezes não são adequadas ao ambiente fabril por questões da agressividade do ambiente. Outro fator limitante na utilização desta plataforma está relacionado a problemas de performance, visto que não foi desenvolvido especificamente para esta aplicação. Como alternativa para estas limitações o presente trabalho propôs a implementação de um sistema de inspeção de produtos manufaturas empregando FPGA, para o qual foi sintetizado um *hardware* especializado em processamento de imagem. Para cumprir o seu objetivo o sistema é capaz de realizar as seguintes funções: aquisição de imagens, conversão entre sistema de cores, segmentação, rotulação e extração de descritores de objetos presentes em imagem. Cada uma das funções e algoritmos desenvolvidos foram validados através de inspeções realizadas em aplicações industriais reais, comprovando a efetividade do sistema.

Palavras-chave: Sistema de visão computacional. Inspeção automatizada. FPGA. Rotulação de componentes conexos.

ABSTRACT

The industrial sector has increasingly sought the use of automated inspection systems, either for the replacement of the inspection performed by workers or for deployments in lines not yet inspected. This search is explained by the fact that the automated inspection is able to confer a higher quality to the products, since it has higher rates of accuracy and for allowing higher inspection speeds than those achieved by inspection performed by human operators. Normally the systems developed for this purpose are based on general purpose platforms (computers), which are often not suitable for the industrial environment because of the aggressiveness of the environment. Another limiting factor in the use of this platform is related to performance problems, since it was not developed specifically for this application. As an alternative to these limitations, the present work proposed the implementation of a system of inspection of manufacturing products employing FPGA, for which a hardware specialized in image processing was synthesized. To fulfill its objective the system is able to perform the following functions: image acquisition, color system conversion, segmentation, labeling and extraction of descriptors of objects present in image. Each of the functions and algorithms developed were validated through inspections carried out in real industrial applications, proving the effectiveness of the system.

Keywords: *Computer vision system. Automated inspection. FPGA. Labeling connected components.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ASIC – *Application Specific Integrated Circuits*

CCD – *Charge-Coupled Device*

CI – Circuito Integrado

CMOS – *Complementary Metal Oxide Semiconductor*

CMYK – *Cyan Magenta Yellow Key*

FIFO – *First in, First out*

FPGA – *Field Programmable Gate Array*

HDL – *Hardware Description Language*

HSL – *Hue Lightness Saturation*

HSV – *Hue Saturation Value*

I2C – *Inter-Integrated Circuit*

RAM – *Random Access Memory*

RGB – *Red Green Blue*

SoC – *System-on-a-chip*

SRAM – *Static Random Access Memory*

UART – *Universal Asynchronous Receiver/Transmitter*

VGA – *Video Graphics Array*

VHDL – VHSIC *Hardware Description Language*

VHSIC – *Very High Speed Integrated Circuits*

SUMÁRIO

1 INTRODUÇÃO

No atual cenário mundial, consumidores buscam por maior qualidade nos produtos consumidos. Este fato, somado ao mercado cada vez mais competitivo, faz com que as indústrias tenham que inspecionar 100% dos produtos manufaturados, a fim de evitar que produtos defeituosos cheguem ao consumidor. Além disso, é importante observar que a inspeção realizada ao longo dos processos produtivos pode evitar prejuízos associados a mão de obra e matéria prima, evitando que produtos comprometidos detectados nas fases iniciais sejam repassados para as etapas seguintes.

Entre as alternativas para a inspeção de produtos tem-se duas que mais se destacam: inspeção visual por operários e inspeção utilizando visão computacional. A inspeção visual torna-se atraente devido ao baixo custo inicial, no entanto, há problemas relacionados ao cansaço visual do operador que faz com que o mesmo, após longos períodos de tempo olhando o mesmo produto repetitivamente a altas velocidades não preste mais atenção nos detalhes e aprove produtos defeituosos (DAVIES, 2012). A visão computacional apresenta um custo inicial mais elevado, porém quando integrada com sucesso a uma linha de produção oferece alta velocidade de inspeção e níveis muito altos de acerto (DAVIES, 2012).

No mercado existem produtos proprietários de inspeção compostos por câmeras inteligentes e módulos de iluminação. Estas câmeras possuem software embarcado e *hardware* de tamanho reduzido porte (OMRON MICROSCAN SYSTEMS, 2018). Porém, a maioria destes produtos são importados, e o alto custo de tais soluções dificulta a utilização da tecnologia pela indústria, principalmente as de médio e pequeno.

Iniciativas de desenvolvimento de sistemas de inspeção por visão computacional abertos, customizados e de baixo custo são geralmente baseadas em computadores de propósito geral. Computadores deste tipo possibilitam o uso em grande gama de funções diferentes, porém não apresentando desempenho comparável a outras plataformas com *hardware* e *software* dedicado a tarefas específicas.

A razão da utilização de computadores de propósito geral é o fato de existirem diversas bibliotecas e ferramentas disponíveis para tal plataforma, tanto proprietárias quanto livres, facilitando assim a implementação de tais sistemas.

Porém, uma série de problemas são enfrentados na iniciativa de utilizar computadores de propósito geral em aplicações industriais. Um primeiro problema diz respeito ao ambiente agressivo, caracterizado por vibrações, interferências eletromagnéticas, altas temperaturas, choques, poeira, dentre outros fatores. Esta dificuldade pode ser contornada parcialmente com o uso de computadores especificamente construídos para o ambiente industrial. Porém, mesmo computadores industriais apresentam desafios relacionados às dimensões, significativamente maiores do que soluções proprietárias e que muitas vezes dificulta a integração. Ainda, por se tratar de uma arquitetura de propósito geral, problemas de performance podem ser enfrentados visto que são executados diversos processos concorrentes do próprio sistema operacional.

Uma possível solução para o problema de performance e tamanho é o desenvolvimento de um *hardware* e/ou software dedicado para processamento de imagens. Uma alternativa é a utilização de microcontroladores de uso geral com software dedicado, porém está também pode esbarrar no problema de desempenho, especialmente em algoritmos altamente interativos, como é o caso do algoritmo de componentes conexos.

Outra opção é o desenvolvimento de circuitos integrados especifico de aplicação (ASIC - *Application specific Integrated Circuit*), que são *hardwares* dedicados para a execução de alguma tarefa, onde a eficiência e velocidade conseguida é a maior possível. Porém o seu desenvolvimento não é viável para baixa e média produção, visto que exige tempo e custo de projeto extremamente elevado (MERTES, 2012).

Neste contexto, a tecnologia de matriz de portas programáveis em Campo (FPGA - *Field Programmable Gate Array*) se torna uma opção. Dentre as vantagens do uso de um FPGA, temos a possibilidade de ser reconfigurado de acordo com a necessidade, o ganho em desempenho uma vez que várias tarefas podem ser realizadas de forma paralela, menor consumo de energia e tamanho reduzido.

Neste trabalho é apresentado o desenvolvimento de um sistema de processamento de imagens utilizando FPGA, para o qual é sintetizado um *hardware* dedicado como alternativa para os problemas levantados.

2 OBJETIVOS

Os objetivos propostos para o desenvolvimento do trabalho são descritos na sequência.

2.1 Objetivos Gerais

Desenvolver um sistema de inspeção por detecção, contagem e medição de peças através de imagens empregando FPGA.

2.2 Objetivos Específicos

Os objetivos específicos são:

a) estudo teórico de técnicas e trabalhos correlatos;
b) definição da arquitetura do sistema: plataforma FPGA e periféricos necessários;
c) projeto e implementação dos algoritmos de processamento e análise que compõem o sistema de inspeção por imagens;
d) validação do sistema utilizando aplicações industriais;
e) documentação dos resultados obtidos.

3 FUNDAMENTAÇÃO TEÓRICA

Nesta seção estão descritos alguns tópicos básicos que possibilitam um melhor entendimento do trabalho realizado. Na primeira parte são descritos conceitos básicos sobre imagens digitais e alguns processos aos quais uma imagem pode ser submetida. Na sequência são expostas as possíveis etapas e configurações de um sistema de visão computacional. Na terceira parte são descritas a forma construtiva e de funcionamento de um FPGA, bem com expostos fabricantes e algumas ferramentas disponíveis para o desenvolvimento nessa plataforma. Por fim são discutidos trabalhos semelhantes disponíveis na literatura.

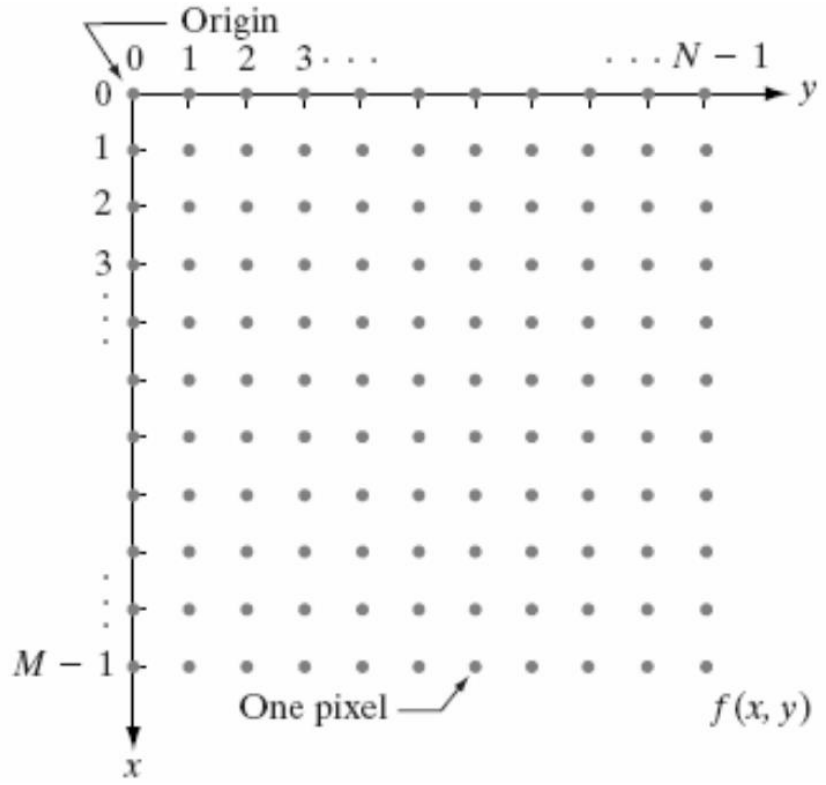
3.1 Processamento Digital De Imagens

Para que seja possível classificar objetos presentes em uma imagem, utilizando alguma métrica é necessário que a imagem seja submetida a uma série de processamentos. Estas operações permitem que atributos úteis possam ser extraídos e comparados a valores de referência. Algumas das operações mais utilizadas no processamento de imagens são abaixo abordadas.

3.1.1 Imagem Digital

Uma imagem digital pode ser entendida como uma função bidimensional f(x, y), onde para cada coordenada espacial (x, y) temos um valor proporcional a luminosidade da imagem naquele ponto. Desta forma, uma imagem digital monocromática pode ser entendida como uma matriz bidimensional onde os valores de cada posição, denominados pixels, recebem um valor proporcional ao seu brilho (ou nível de cinza) dentro de determinada amplitude (FILHO; NETO, 1999; GONZALEZ; WOODS, 2008; MERTES, 2012). A Figura 1 ilustra a representação de uma imagem digital.

Figura 1 - Representação de uma imagem digital



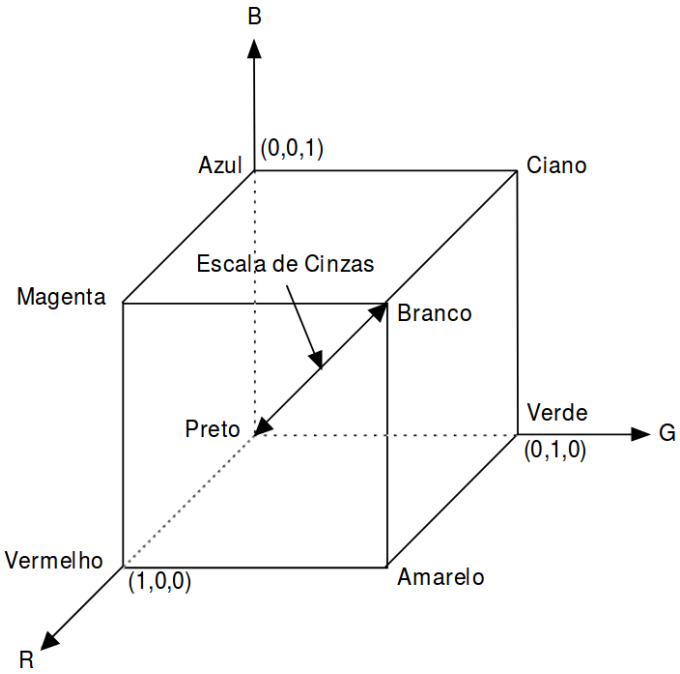
Fonte: Santiago (2009)

Normalmente, em uma imagem digital o valor de cada pixel é expresso por um *byte* (oito *bits*), podendo assim ter 256 níveis de cinza. Nesta representação, o nível de valor 0 é o preto, representando a ausência de luz, e o de valor 255 é o branco, que representando o brilho mais intenso possível de ser percebido.

3.1.1.1 Modelos de cores

Além da representação em escala de cinzas, descrita na seção anterior, temos outros modelos. Uma imagem digital também pode ser colorida, onde cada um dos pixels terá mais de um valor para representar a diferença entre cores, existindo assim diversos sistemas de cores disponíveis para tal objetivo. Um dos sistemas mais utilizados é o RGB que é um sistema de cores aditivas (baseado na emissão de luz, no qual a mistura das cores primárias forma branco), onde as três cores primárias aditivas são: vermelha (R - *Red*), verde (G - *Green*) e azul (B - *Blue*). Neste sistema os componentes RGB são combinadas para formar uma grande quantidade do espectro de cores visíveis. A Figura 2 representa o sistema de cores RGB.

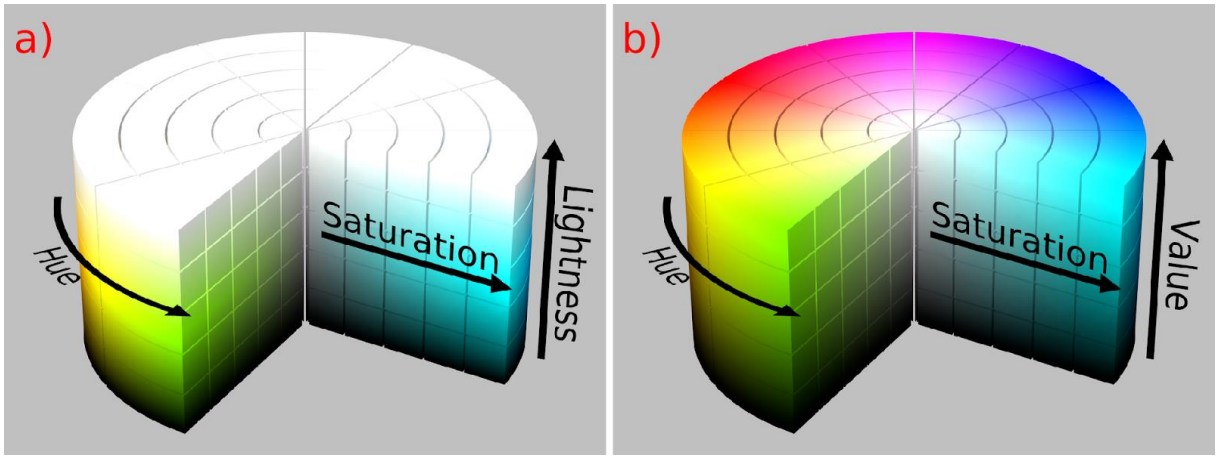
Figura 2 - Sistemas de cores RGB



Fonte: Filho, Neto (1999)

Alguns sistemas de cor foram especialmente desenvolvidos para se assemelhar melhor ao modo como o olho humano percebe as cores e a luz. Este é o caso dos sistemas de cores HSV e HSL. O modelo HSV é baseado pelas componentes matiz (H - *Hue*), saturação (S - *Saturation*) e valor (V - *Value*), e no sistema HSL a componente valor é substituído pela componente luminosidade (L - *Lightness*). Os sistemas HSV e HSL podem ser representados por cilindros como os mostrados na Figura 3.

Figura 3 - Sistemas de cores: a) HSL; b) HSV



Fonte: Wikipedia (2018a)

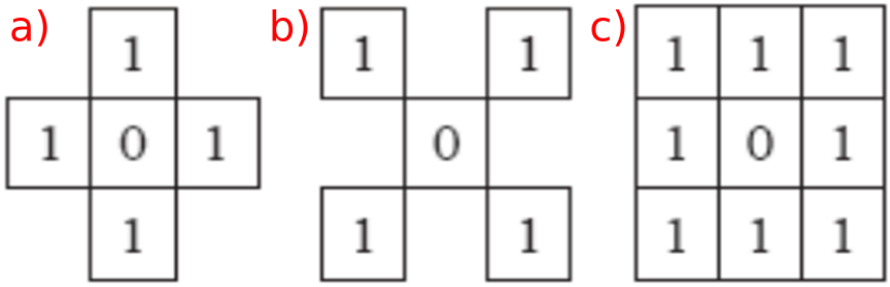
Outro modelo de cor bastante utilizado é o CMYK, que é usado por impressoras, sendo um sistema de cores subtrativas (baseado na absorção de luz,

onde a mistura das cores primárias resulta em preto) que utiliza as cores primárias pigmento. As cores primárias subtrativas são opostas às utilizadas no modelo RGB, sendo elas: ciano (C - Cyan), magenta (M - Magenta), amarelo (Y - Yellow) além do preto (Black (K - Key)).

3.1.1.2 Conectividade de pixels

Em processamentos digitais de imagem o conceito de vizinhança (ou conectividade) de pixels é muito importante pois várias operações são nele baseados. Um pixel tem quatro pixels vizinhos horizontais e verticais e quatro vizinhos diagonais. Em alguns algoritmos de processamento de imagens são considerados a 4-vizinhança, como ilustrado na Figura 4.a, onde são considerados apenas vizinhos horizontais e verticais. Em outros são considerados somente a vizinhança diagonal, como ilustrado na Figura 4.b. Ainda há aqueles em que são considerados tanto os vizinhos horizontais, verticais e diagonais, como ilustrado na Figura 4.c, utilizando o conceito de 8-vizinhança (FILHO; NETO, 1999).

Figura 4 - Vizinhança do pixel representado com 0: a) 4-vizinhança; b) vizinhança diagonal; c) 8-vizinhança



Fonte: Mestes (2012)

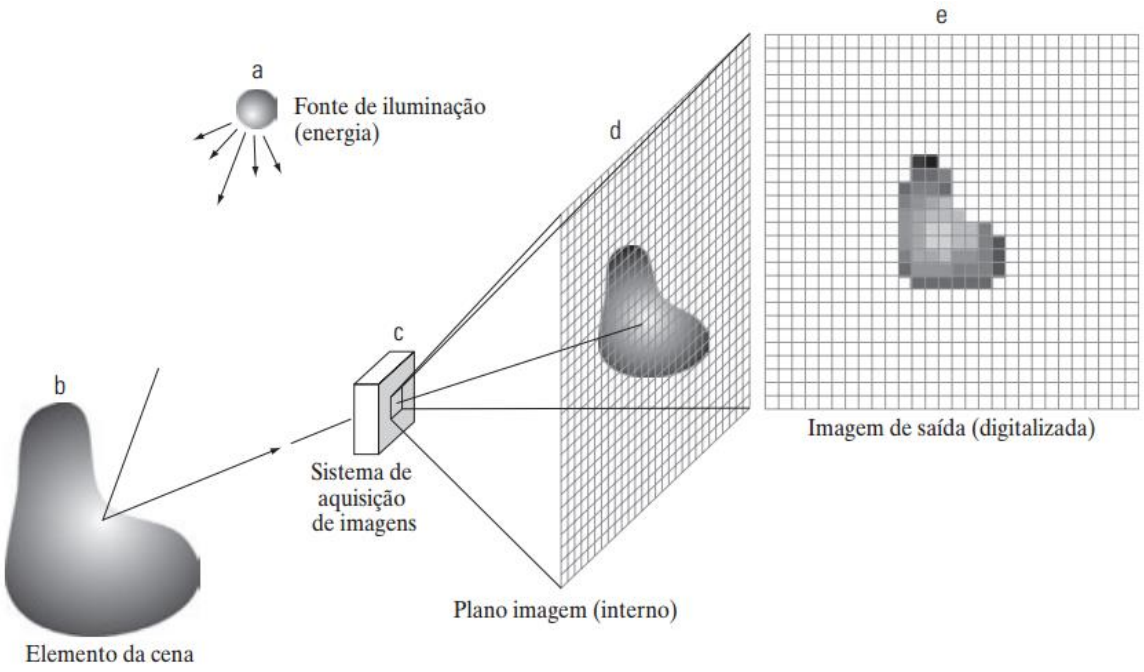
De forma matemática, um pixel p de coordenadas (x,y) está conectado ao vizinho superior (x, y-1), ao inferior (x, y-1), aos laterais (x-1, y) e (x+1, y) e aos diagonais de coordenadas (x-1, y-1), (x+1, y-1), (x-1, y+1) e (x+1, y+1).

3.1.2 Aquisição de Imagens Digitais

A aquisição de imagens digitais corresponde ao processo no qual é convertida uma cena real tridimensional para uma imagem digital bidimensional. A aquisição pode ser subdividida em duas etapas: a etapa de transdução optométrica e a etapa de digitalização da imagem. O processo de aquisição é representado pela Figura 5, onde é ilustrado um elemento da cena (b) que está sendo iluminado por uma fonte de luz (a), que após a passagem da luz refletida pelo objeto por um sistema de

lentes de aquisição (c) a imagem é amostrada por um transdutor (d) sendo digitalizada na sequência (e) (GONZALEZ; WOODS, 2008).

Figura 5 - Aquisição de imagem Digital



Fonte: Gonzalez (2008)

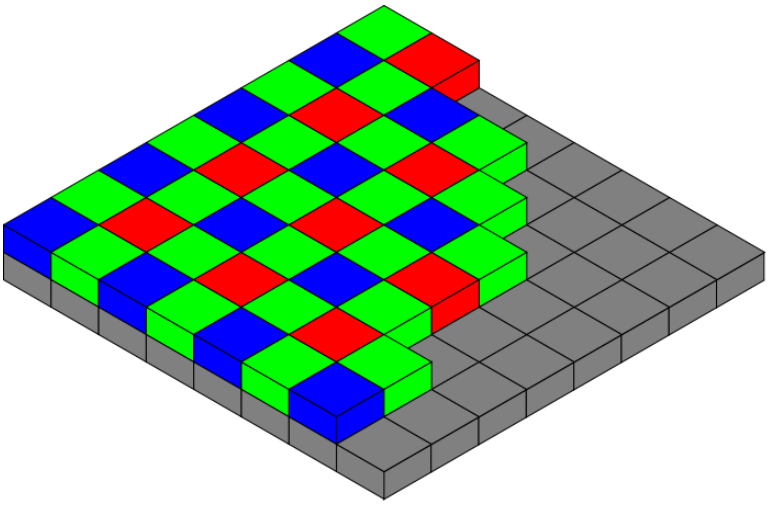
A etapa de transdução depende de um dispositivo físico que seja sensível a espectros de energia eletromagnética - como por exemplo a luz visível - e que converta a energia recebida em sinais elétricos analógicos. Os dispositivos mais utilizados atualmente são os CMOSs (*Complementary Metal-Oxide Semiconductor*) seguido pelos CCDs (*Charge Coupled Device*). As vantagens como tamanho reduzido, maior velocidade de aquisição e menor consumo fazem dos dispositivos CMOS os mais utilizados, porém dispositivos CCDs podem ser encontrados em produtos de topo de mercado de fotografias onde a qualidade final da imagem estática é o aspecto mais importante do produto (TIM MOYNIHAN, 2011).

Tanto dispositivos CCDs como CMOS são de forma básica formados de arranjos matriciais de células fotossensíveis. Para a aquisição de imagens coloridas é utilizado um conjunto de prismas e filtros que são encarregados de decompor a luz nos componentes R, G e B de forma a serem captadas por células independentes.

Na maioria dos dispositivos a divisão de cores é realizada por um filtro Bayer, no qual metade dos filtros permitem a passagem somente de luz verde, um quarto de vermelho e um quarto de azul. O dobro de elementos verdes é utilizado para

imitar a maior sensibilidade do olho humano a cor verde (SAVVAS, 2006). A Figura 6 representa um filtro Bayer aplicado sobre um sensor de imagem.

Figura 6 - Filtro Bayer



Fonte: Wikipedia (2018b)

Para se construir uma imagem RGB a partir de um sensor com filtro Bayer é necessário calcular os valores dos componentes vermelho e azul para um pixel verde, azul e verde para um pixel vermelho e verde e vermelho para um pixel azul por operações de interpolação entre pixels vizinhos.

Outro fator importante na etapa de transdução é a de amostragem, a qual se refere a perda de resolução da cena real para a imagem adquirida devido a quantidade limitada de células fotossensíveis no dispositivo de aquisição. No dispositivo é realizada uma média do valor de luminosidade da parte da imagem que está sobre cada um dos elementos transdutores. A amostragem pode ser facilmente entendida observando-se a Figura 5 onde o elemento da cena (b) é dividido pelos vários elementos que formam o plano de aquisição (d).

Na etapa de digitalização da imagem, os sensores de imagem são lidos e suas saídas analógicas são convertidas para valores digitais quantificados proporcionalmente entre os níveis máximos e mínimos possíveis para um pixel. A quantificação tem como objetivo permitir que após a digitalização ainda seja possível observar a diferenças de tons e cores da cena real da qual a imagem foi adquirida. A digitalização pode ser observada na Figura 5, onde, a partir da amostragem realizada (d) é assumido um nível de cor para o pixel (e).

Geralmente, é utilizado 1 *byte* (8 *bits*) para armazenar o valor de um pixel no caso de imagens em tons de cinza e 1 *byte* para cada componente RGB em uma imagem colorida, sendo 0 e 255 os valores mínimos e máximos possíveis para elemento de cor de um pixel (MERTES, 2012). Quanto menor for o número de níveis possíveis, mais abrupta será a mudança entre tons em uma imagem digital.

3.1.3 Conversão Entre Espaços de Cor

Como foi descrito anteriormente, uma imagem pode ser descrita em diversos modelos de cores, podendo assim em alguns casos ser necessário a conversão entre espaços para destacar alguma característica ou aplicar algum algoritmo na imagem. Um exemplo simples é a conversão de uma imagem RGB para HSL para diminuir a luminosidade da imagem mexendo somente no componente L ao invés dos três componentes RGB.

A conversão de RGB para tons de cinza também é muito utilizada, especialmente em processamentos em que só será levado em conta a intensidade e não as cores propriamente ditas. Outra vantagem em se trabalhar somente com tons de cinza é a menor quantidade de informação, pois ao invés de três componentes só haverá um, diminuindo assim o tempo e memória necessária para processamento.

Para a conversão de RGB para tons de cinza existem dois principais métodos, o primeiro é o método da média onde o valor de cinza de um pixel é dado pela média entre os três canais RGB, sendo expresso pela Equação 1.

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

O outro método é o da luminosidade que busca rebalancear a média de forma que o nível de cinza fique mais semelhante ao percebido pelo olho humano (NOGUEIRA, 2016). A conversão para tons de cinza pelo método da luminosidade é calculado a partir da Equação 2.

$$C = 0{,}29 * R + 0{,}59 * G + 0{,}11 * B \tag{2}$$

Como pode ser visto na equação acima, o componente verde possui um peso superior devido à maior sensibilidade do olho humano a tal cor. A diferença entre os dois métodos pode ser visualizada na Figura 7.

Figura 7 - Diferença entre métodos de conversão para tons de cinza: a) Imagem original colorida; b) Método da média; c) Método luminosidade

Fonte: Tutorialspoint (2018)

Como pode ser visto na Figura 7, apesar do método da média ser mais simples, este método resulta em uma imagem não muito semelhante visualmente a original, pois o olho humano tem percepção diferente entre cada uma das cores.

3.1.4 Limiarização

Uma das etapas mais importantes no processamento de imagem é a segmentação. Nesta etapa são separados os objetos de interesse do restante da imagem. A limiarização está entre as técnicas mais utilizadas de segmentação existentes devido a sua facilidade, simplicidade de aplicação e seu custo computacional relativamente baixo (FILHO; NETO, 1999; MERTES, 2012).

A limiarização consiste em fixar um limiar dentro dos níveis de cinza possíveis para um pixel de uma imagem. Após o limiar ser fixado a imagem é varrida e todos os pixels com valor abaixo do limiar fixado tem seu valor ajustado para 0 e todos os pixels com valores acima do limiar são ajustados para o valor binário 1.

Desta forma, a imagem produzida pela limiarização é uma imagem binária, isto é, com somente dois níveis, motivo pelo qual a limiarização é conhecida também de binarização. Em uma imagem binária usualmente os pixels pretos (com valor 0) são considerados pertencentes ao fundo da imagem e os pixels brancos (com valor 1) são considerados formadores de objetos de interesse da imagem.

O processo de limiarização e a diferença entre valores de limiar pode ser observado na Figura 8.

Figura 8 - Limiarização: a) Imagem original; b) Imagem binarizada com limiar = 100; c) Imagem binarizada com limiar = 150

Fonte: Próprio autor

Como pode ser visto na Figura 8 diferentes valores de limiar produzem diferentes imagens segmentadas. Desta forma, o correto ajuste de limiar é uma das etapas mais críticas para a correta identificação de objetos em uma cena.

3.1.5 Operações Morfológicas

As operações morfológicas são normalmente empregadas para corrigir imperfeições na etapa de segmentação de uma imagem. Seu uso permite, por exemplo: eliminar pequenos elementos oriundos de ruídos e fechar buracos nos objetos da imagem.

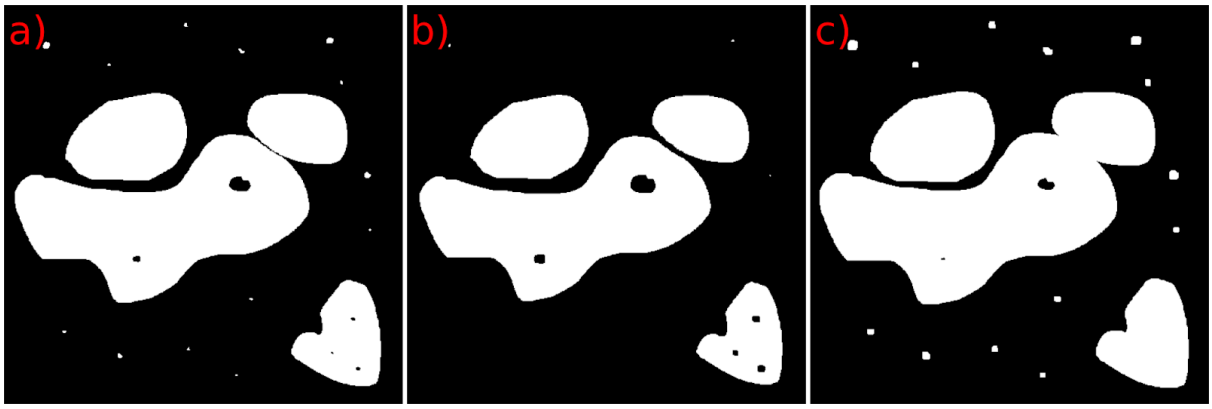
As operações morfológicas utilizam um elemento estruturante normalmente formado pelos conjuntos de 4-vizinhança ou 8-vizinhança, explicados anteriormente, mas podem ser utilizados diversos outros elementos de acordo com a necessidade e objetivo das operações.

As operações morfológicas mais básicas são as de erosão e dilatação de imagens. A erosão permite separar objetos que se tocam e remover pequenos objetos da imagem. Considerando uma imagem binarizada onde o fundo tem valor 0 (preto) e os objetos 1 (branco), na erosão a imagem é varrida pelo elemento estruturante de forma que o pixel analisado receberá o valor 1 somente se todos os pixels contidos no elemento estruturante forem 1, caso contrário o pixel receber o valor 0.

A dilatação pode ser entendida como a operação contrária a erosão e permite unir objetos e preencher buracos em seus interiores. Desta forma um pixel analisado receberá o valor 0 somente se todos os pixels do elemento estruturante

forem 0, caso contrário é atribuído o valor 1. Na Figura 9 podem ser observados os resultados das operações de erosão e dilatação.

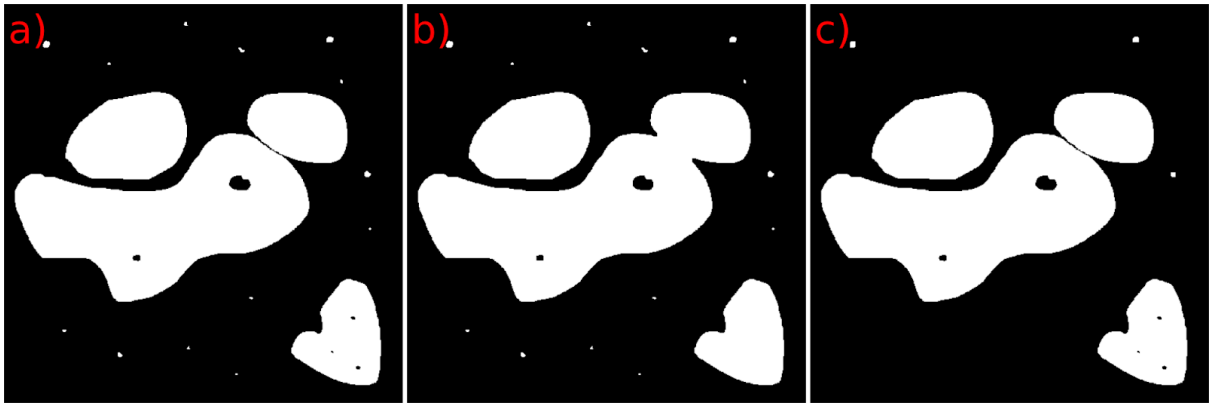
Figura 9 - Erosão e dilatação: a) Imagem original; b) Imagem erodida: c) imagem dilatada



Fonte: Próprio autor

Outras operações muito utilizadas são as de abertura e fechamento, que são derivadas das operações de erosão e dilatação. A abertura é constituída por erosão seguida de uma dilatação e permite eliminar pequenos objetos da imagem e suavizar bordas, o fechamento é uma dilatação seguida de uma erosão que permite o fechamento de pequenos separações e furos em objetos (ESQUEF; ALBUQUERQUE; ALBUQUERQUE, 2003). Na Figura 10 podem ser observadas as operações de abertura e fechamento de uma imagem.

Figura 10 - Abertura e fechamento: a) Imagem original; b) Imagem após abertura; c) Imagem após fechamento



Fonte: Próprio autor

As operações de abertura e fechamento acabam sendo mais utilizadas que as operações de erosão e dilatação isoladas pois além de realizarem a eliminação de ruídos e fechamento de furos, acabam retornando os objetos da imagem ao tamanho original permitindo assim a coleta de seus atributos de forma mais fiel.

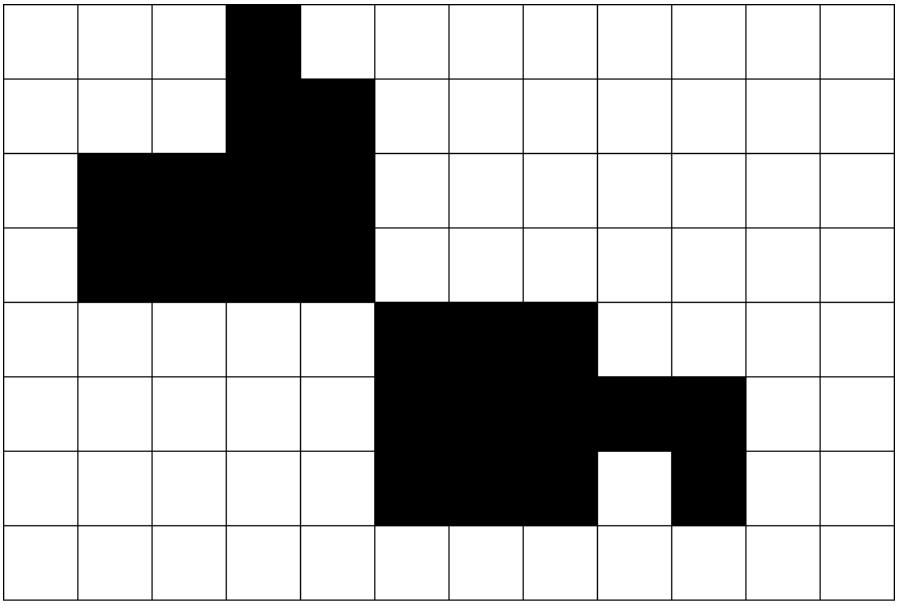
3.1.6 Rotulação de Componentes Conexos

Após a segmentação de uma imagem é possível a distinção entre objetos presentes na imagem e fundo, porém não há a distinção entre os diversos possíveis objetos presentes. Um dos métodos mais utilizados para a diferenciação entre os objetos é o algoritmo de rotulação por componentes conexos.

Além do conceito de conectividade já explicado na seção 3.1.1.2, outro conceito importante para o entendimento do algoritmo de componentes conexos é o de similaridade entre pixels. Em imagens em tons de cinza, pixels similares são aqueles que apresentam tons de cinza semelhantes, já em imagens binarizadas um pixel é similar a outro se ambos apresentam o mesmo valor.

Portanto para que dois pixels sejam considerados conectados é necessário que ambos sejam considerados semelhantes e que sejam vizinhos. Observando a Figura 11 é possível observar a diferença entre a utilização da 4 e 8 vizinhança.

Figura 11 - Diferença entre 4 e 8 vizinhança



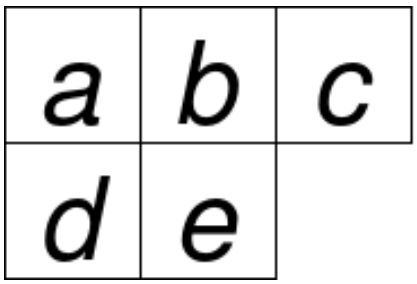
Fonte: Próprio autor

Como pode ser visto na figura acima, no caso da utilização da 4-vizinhança é considerada a existência de dois componentes conexos distintos. No entanto com 8-vizinhança somente um componente conexo é considerado, devido aos grupos de pixels pretos compartilharem somente uma vizinhança diagonal. Portanto, a escolha da vizinhança influencia diretamente a quantidade de objetos detectados afetando a extração de suas características subsequentemente.

Existem diversas implementações distintas para a rotulação de componentes conexos. Algumas delas necessitam de diversas varreduras na imagem até que todos os pixels de um componente tenham o mesmo rótulo, já algumas implementações necessitam de somente uma. Basicamente, em todas as implementações a imagem é varrida, normalmente em sentido da esquerda para a direita e de cima para baixo, atribuindo rótulos provisórios a todos os pixels que são considerados pertencentes a um objeto. Toda vez que um novo pixel é analisado, seus vizinhos são analisados para que seja atribuído um pixel apropriado caso alguns dos mesmos já tenham sido rotulados.

Além dos algoritmos *multi-pass* e *one-pass* que varrem a imagem várias e uma única vez respectivamente, existem implementações chamadas de *two-pass* que varrem a imagem duas vezes, sendo este o mais clássico e de relativamente fácil implementação. Para um melhor entendimento do algoritmo, a Figura 12 representa o pixel examinado *e* seus vizinhos *a*, *b*, *c* e *d* (considerando 8-vizinhança) que já foram examinados.

Figura 12 - Máscara de pixels analisados na rotulação de componentes conexos



Fonte: Adaptado de Santiago (2009)

Inicialmente, o pixel *e* é analisado e caso seu valor indique que é pertencente ao fundo e não faz parte de um objeto (normalmente valor 0 em imagens binárias) seu valor continua sendo 0, indicando que não tem rótulos atribuídos. Caso o valor do pixel indique que faz parte de um objeto, os pixels já analisados *a*, *b*, *c* e *d* são examinados à procura de rótulos já atribuídos e, caso nenhum deles tenha um rótulo, o pixel *e* recebe um novo rótulo. Caso somente um dos pixels contenha um rótulo ou dois ou mais pixels tenham o mesmo rótulo, o pixel *e* recebe esse mesmo rótulo, caso existam rótulos diferentes, *e* recebe o menor dos rótulos dos vizinhos e a relação de equivalência entre rótulos encontrados é guardada em um vetor.

Após todos os pixels serem analisados pelo algoritmo e todos os pixels pertencentes a objetos rotulados com rótulos temporários, é realizada uma segunda passada na imagem onde para cada rótulo atribuído é consultado o vetor de equivalência e atribuído um rótulo definitivo ao pixel.

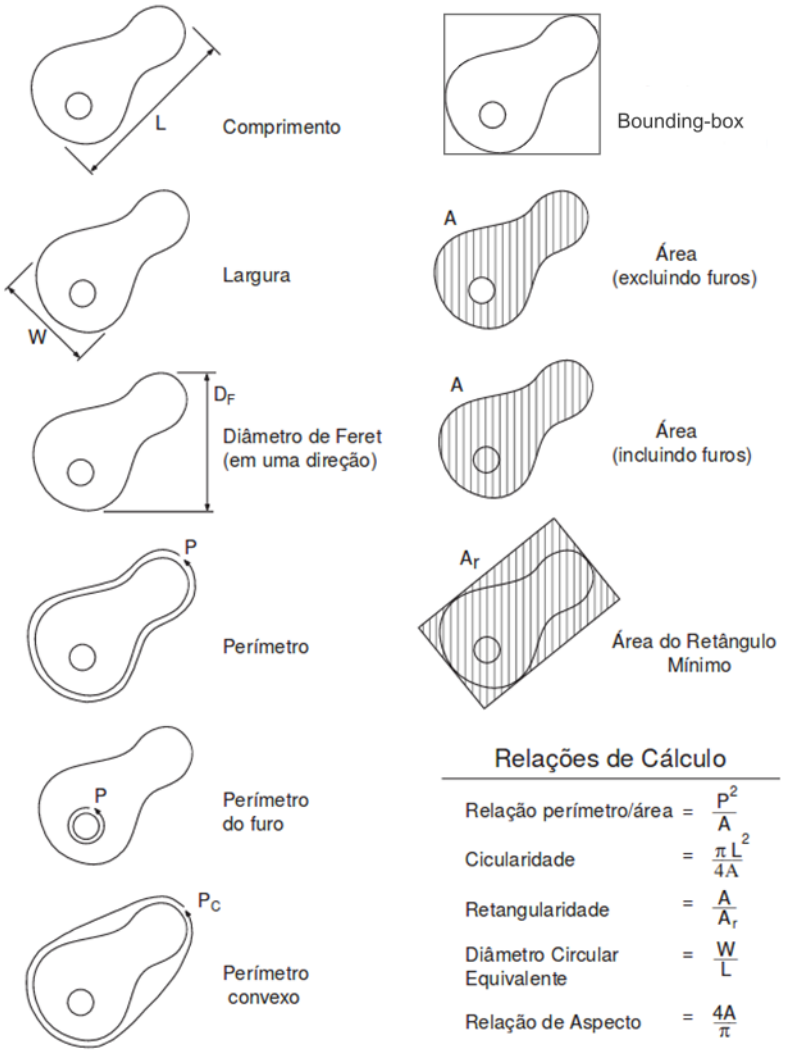
Uma vez que todos os objetos foram devidamente rotulados, é possível extrair descritores que permitam a análise e classificação dos objetos da imagem. Descritores podem ser definidos como qualquer parâmetro que descreve alguma característica de um elemento presente em uma imagem.

Abaixo são listados e brevemente descritos alguns dos descritores mais utilizados:

a) área: quantidade total de pixels de um objeto. Uma vez que se saiba a relação de área em pixels de uma imagem para alguma unidade do mundo real pode ser feita a conversão de forma a exibir ou utilizar o valor em algum cálculo.
b) perímetro: soma das distâncias entre todos os pixels formadores da borda de um componente, onde para dois vizinhos superiores e laterais a distância é de 1 pixel e para vizinhos diagonais a distância é igual a $\sqrt{2}$ pixels.
c) *bounding-box*: descritor que representa uma caixa ao redor do objeto. As posições dos vértices da caixa são dadas pelas posições mínimas e máximas dos pixels formadores do componente, desta forma o descritor pode ser subdividido em valores *x* mínimo e máximo e valores *y* mínimo e máximo.
d) centroide: posição central do descritor *bounding-box*, é calculada a partir da média dos valores máximos e mínimos dos eixos X e Y.
e) comprimento: maior distância linear entre dois pixels de um mesmo objeto.
f) circularidade: índice que através de uma relação entre os descritores comprimento e área indica o quão próximo de um círculo o objeto é.

Na Figura 13 podem ser vistos alguns dos descritores citados.

Figura 13 - Exemplo de descritores



Fonte: Adaptado de Esquef (2003)

Na Figura 13 pode ser observado que existem diversos descritores e também que os mesmos podem ser combinados de forma a produzir outros novos descritores que sejam relevantes para a análise de alguma característica.

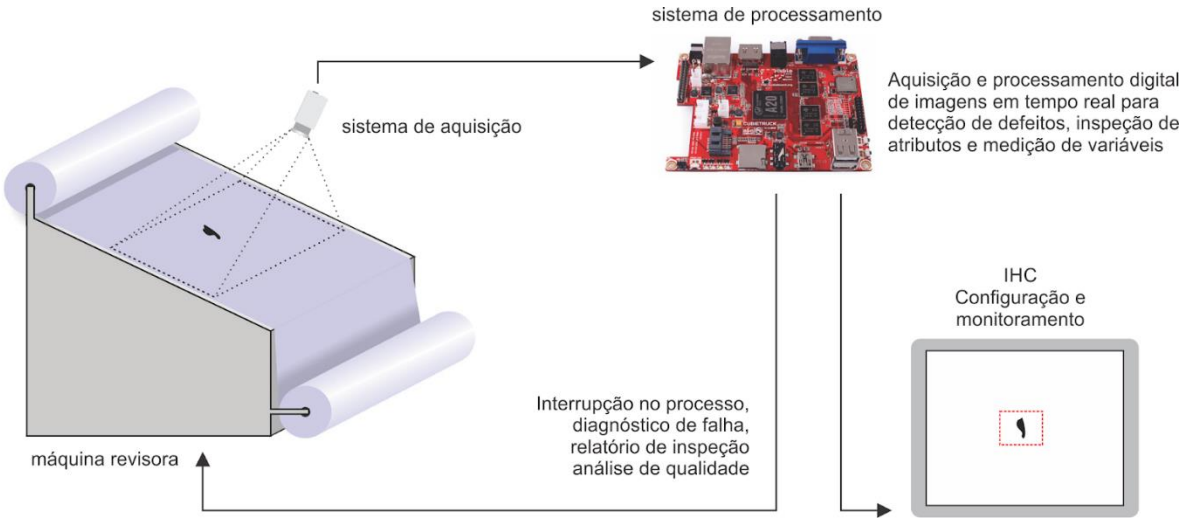
3.2 Visão Computacional

Uma das ferramentas mais utilizadas para automatizar processos de inspeção em indústrias é a visão computacional. Esta ferramenta utiliza imagens adquiridas de câmeras e algoritmos de processamento de imagens para analisar características relevantes do produto inspecionado a fim de identificar, de forma automática, defeitos (STEGER; ULRICH; WIEDEMANN, 2008; SZELISKI, 2010).

3.2.1 Arquitetura Simplificada e Suas Etapas Principais

Existem diversas configurações para um sistema de inspeção com visão computacional dependendo do ambiente de aplicação do mesmo e qual objeto se deseja inspecionar. De forma básica, todos os sistemas necessitam de sistemas de iluminação, sistema de aquisição, um meio de transmissão, um computador ou outra plataforma para o processamento e um meio de saída para a informação extraída, seja um monitor ou um sinal de atuação para que o produto seja descartado de forma automática. A Figura 14 ilustra a configuração básica de um sistema de visão computacional, exemplificando a sua utilização na detecção de defeitos em tecidos.

Figura 14 - Configuração básica de um Sistema de Visão

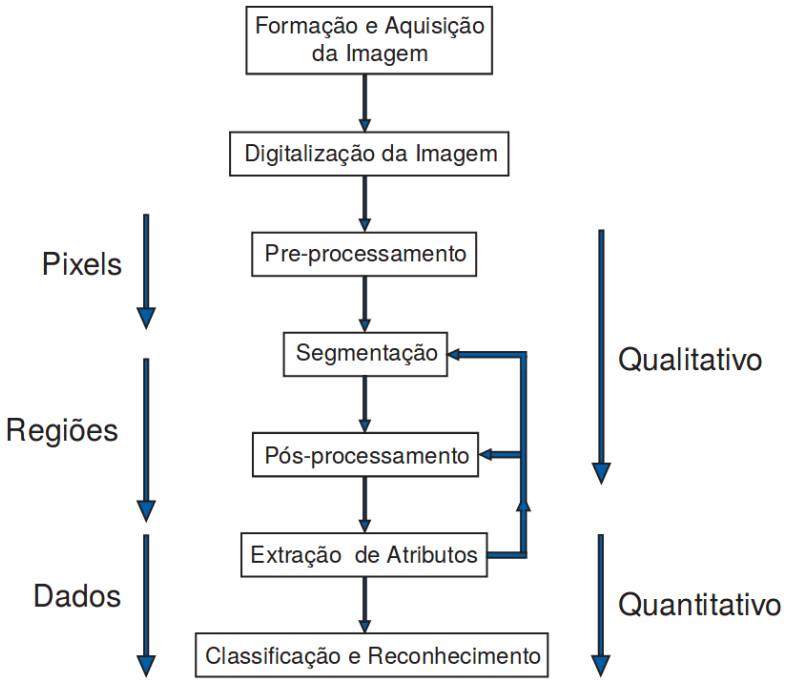


Fonte: Vargas (2016)

Um sistema de processamento de imagens é constituído de diversas etapas, entre elas: aquisição de imagem, pré-processamento, segmentação, extração de características e interpretação (ESQUEF; ALBUQUERQUE; ALBUQUERQUE, 2003). Essas etapas e o fluxo entre elas é ilustrada na Figura 15. A seguir é dada uma breve descrição de cada uma das etapas.

Aquisição: esta etapa como já descrito na seção 3.1.2 depende de um dispositivo capaz de transformar energia eletromagnética em sinais digitais, de forma que possam ser processados. A energia captada pode ser tanto do espectro de luz visível, como infravermelho, ultravioleta, raios X entre outros.

Figura 15 - Etapas do processamento de imagens



Fonte: Esquef (2003)

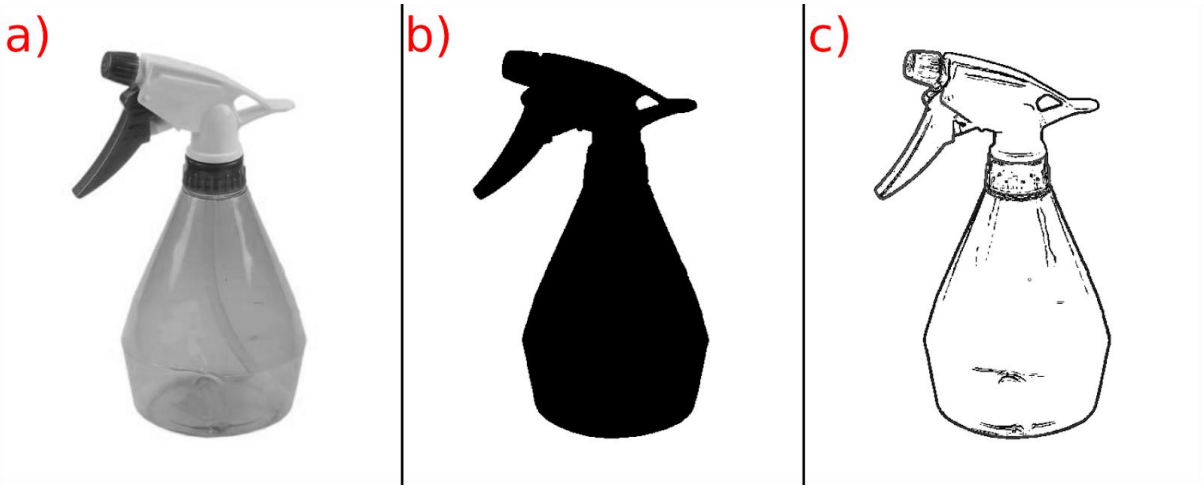
Pré-processamento: tem como objetivo melhorar a qualidade da imagem oriunda do dispositivo de aquisição, de forma a deixá-la mais adequada para as etapas decorrentes de processamento. Entre os processos mais aplicados nesta etapa estão filtros de mediana, que tem como objetivo reduzir ruídos na imagem, filtros de passa alta que realçam contornos e bordas dos objetos na imagem e conversão para tons de cinza de forma a diminuir a quantidade de informações da imagem a ser processada. A Figura 16 ilustra algumas das operações realizadas na etapa de pré-processamento.

Figura 16 - Exemplo pré-processamento: a) Imagem original; b) Filtro Mediana; c) Imagem (b) convertida para tons de cinza

Fonte: Próprio autor

Segmentação: nesta etapa são destacados os objetos de interesse do restante da imagem. Para esta etapa existem duas técnicas bem difundidas e a escolha entre uma ou outra depende das características que se pretende analisar nas imagens. Uma das técnicas é a binarização de imagens que é baseada na similaridade entre pixels e já foi descrita na seção 3.1.4. A outra técnica é a detecção de bordas que se baseia na descontinuidade entre pixels, ou seja: o algoritmo é sensível a grandes variações na tonalidade entre pixels vizinhos o que permite a detecção de bordas e contornos de objetos. A Figura 17 ilustra os dois métodos mais utilizados na etapa de segmentação.

Figura 17 - Exemplo de Segmentação: a) Original; b) Binarização; c) Detecção de bordas



Fonte: Próprio autor

Pós-processamento: tem como papel a correção de imperfeições da segmentação da imagem. Nesta etapa, normalmente, são utilizadas técnicas de morfologia como os processos de abertura e fechamento de imagens.

Extração de características: etapa em que são extraídas informações úteis da imagem processada. Um dos algoritmos mais utilizados nesta etapa é o de rotulação por componentes conexos que tem como objetivo produzir a distinção entre cada um dos objetos.

Interpretação: etapa em que os descritores extraídos da imagem são analisados e os objetos são classificados. No caso de inspeção de produtos, esta é a etapa em que o objeto inspecionado é considerado aprovado ou rejeitado.

Para se atingir resultados satisfatórios na inspeção de produtos é necessário que em cada uma das etapas acima descritas, se escolha entre os

diversos algoritmos e que os mesmos sejam ajustados de forma a se destacar e corretamente analisar as características relevantes do produto.

3.2.2 Aplicações de Sistemas de Visão

A visão computacional vem sendo utilizada em indústrias dos mais diversos segmentos (DAVIES, 2012), sendo utilizada, por exemplo: na produção de tecidos para identificar defeitos que podem surgir no processo de fiação e tecelagem (VARGAS et al., 2016), em indústrias madeireiras onde tábuas cortadas frequentemente podem apresentar defeitos como furos, trincas e nós (STIVANELLO et al., 2016) e na indústria de envase e enlatados onde podem ocorrer defeitos nos rótulos, lacres ou a presença de contaminantes (STIVANELLO; GOMES, 2006).

Mesmo com a enorme gama de aplicações possíveis para um sistema de inspeção por visão computacional, estes são geralmente implementados para resolver os quatro tipos mais comuns de funções:

a) analisar a presença ou ausência de um componente: casos onde a simples existência ou falta de um componente é suficiente para a aprovação ou rejeição do objeto inspecionado. Neste caso, a extração de atributos dos objetos não é necessária para que um resultado seja obtido. Um exemplo deste caso é a existência de um buracos no tecido no processo de tecelagem que inviabiliza a produção de qualquer produto (VARGAS et al., 2016).

b) contar objetos em uma cena: casos onde a simples presença de um objeto não é suficiente, pois é necessário descobrir o número total de componentes de forma a poder comparar com um objetivo.

c) medir tamanhos e distâncias entre componentes: inspeções onde é necessário a extração de descritores de componentes e/ou suas combinações com descritores de outros componentes presentes na cena. Variáveis como área, altura e largura de um componente e distância entre dois ou mais componentes podem ser utilizados para a avaliação. Alguns exemplos de aplicação são: rejeição de tábuas em madeireiras caso as mesmas apresentem um percentual acima do tolerável de nós (STIVANELLO et al., 2016) ou caso a tábua tenha largura ou comprimento fora das dimensões nominais.

d) decodificar informação: aplicações onde é necessário entender uma informação organizada segundo um padrão conhecido, seja um texto escrito ou uma

representação gráfica como *QRCode* ou códigos de barra. Um exemplo de aplicação é a utilização de *QRcodes* em peças para rastreabilidade em cada um dos processos ao qual a mesma é submetida em uma industrial.

3.3 Descrição Geral do Fpga

FPGAs são circuitos integrados (CI) que são projetados e construídos de maneira que sejam reconfiguráveis após a suas fabricações. Ao contrário dos CIs ASICs que tem todas suas funções definidas na fabricação, FPGAs podem ser reconfigurados de acordo com a necessidade do usuário utilizado para isso uma linguagem de descrição de *hardware* (HDL - *Hardware Description Language*) e ferramentas de desenvolvimento que permitam a síntese e configuração do circuito projetado. Por exemplo, um FPGA pode ser configurado para ser um processador, um contador ou *hardware* dedicado a outro tipo de processamento digital.

Os dispositivos FPGA foram criados em 1985 por Ross Freeman e Bernard Vonderschmitt, co-fundadores da empresa Xilinx que é atualmente o maior fabricante de FPGAs do mundo (ROELANDTS WIM, 1999; WOODS, 2008). Embora tenham evoluído muito desde sua criação, um FPGA é basicamente formado por três elementos básicos, sendo eles:

a) blocos de entradas e saídas: são responsáveis pela interface com componentes externos ao FPGA;

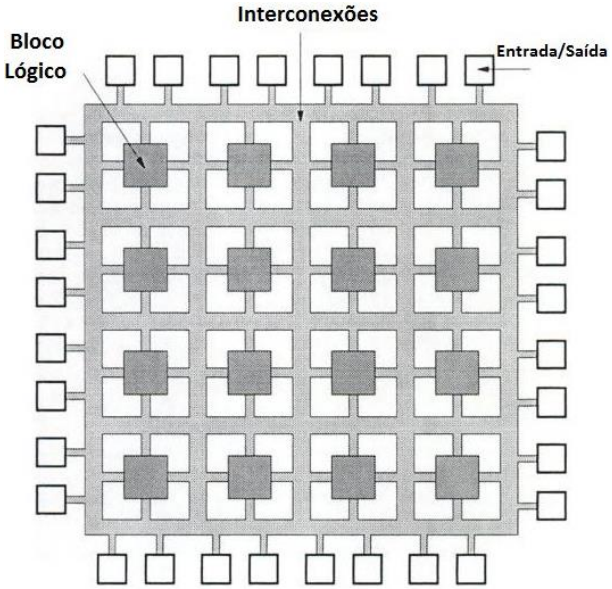
b) blocos lógicos configuráveis: circuitos construídos normalmente com *flips-flops*, LUTs e MUXs que são utilizados para implementar pequenas funções lógicas;

c) barramentos e chaves de interconexão: trilhas que tem como função a conexão entre blocos de entradas e saídas e blocos lógicos.

Na Figura 18 podem ser observados a forma do arranjo entre cada um dos componentes, onde os blocos lógicos formam uma matriz bidimensional e as chaves de interconexão formam canais de roteamento entre os blocos lógicos.

Embora os três elementos sejam os elementos básicos formadores de um FPGA, hoje é comum encontrar FPGA que contam com processadores, elementos de memória e outros circuitos dedicados no mesmo *chip* formando um SoC (*System on Chip*). Um exemplo destes componentes são os que integram no mesmo CI um FPGA e um microprocessador ARM.

Figura 18 - Arquitetura do FPGA



Fonte: Junior (2016)

Devido ao grande avanço na tecnologia desde sua criação, é possível encontrar no mercado FPGAs que utilizam diferente tecnologias em suas construções (BOB ZEIDMAN, 2006; WOODS, 2008). Abaixo são descritos os três mais utilizados:

a) SRAM: a configuração dos elementos comutadores é guardada em células de memória SRAM, onde cada comutador tem um bit associado. É a tecnologia mais utilizada atualmente e pode ser facilmente regravado, tantas vezes quanto necessário. Tem como desvantagem suas características voláteis, necessitando serem regravados após terem sua alimentação interrompida. Esse problema é contornado com o uso de memórias *Flash* ou EEPROM externas que inicializam o FPGA toda vez que o dispositivo é alimentado.

b) *Antifuse*: pode ser gravado somente uma vez. Antes de ser gravado, se comporta como um circuito aberto, mas após ter uma certa tensão aplicada são curto circuitadas as conexões projetadas. Dentre suas vantagens, não necessitam ser regravados a cada inicialização, são mais confiáveis e tendem a ser mais rápidos que o os FPGAs baseados em SRAM. Dentre suas desvantagens, tem um custo mais elevado devido a fabricação ser mais complexa e a necessidade de descartar o dispositivo cada vez que se mude o *hardware* projetado.

c) *Flash*: sua forma construtiva é semelhante ao baseado em SRAM, no entanto as células SRAM são substituídas por *Flash*. Esta tecnologia une as vantagens dos FPGAs baseados em SRAM e *Antifuse* um a vez que podem ser

regravados e de não serem voláteis, já estando pronto para utilização na alimentação do circuito.

Para a configuração de um FPGA é utilizada uma linguagem HDL, como por exemplo: VHDL e Verilog. Essas linguagens permitem a descrição do circuito a ser montado que após ser analisado e sintetizado pode ser gravado em um dispositivo. Porém, antes da gravação de um circuito é indicado que seja feita a simulação do mesmo utilizando uma das diversas ferramentas disponíveis para esse fim.

3.3.1 Arquiteturas Disponíveis Empregando FPGA

Um FPGA, devido a sua grande versatilidade, permite a criação de soluções de diferentes formas. Uma delas é a da síntese de *hardwares* dedicados para os processamentos que se deseja realizar, embora esta abordagem seja menos eficiente que dispositivos projetados e construídos com um único fim (ASICs), tem custo e tempo de projeto bem menores (MERTES, 2012).

Outra é a utilização de *Soft-Processors* que utilizam a capacidade do FPGA de forma similar a processadores de propósito geral. A utilização destes componentes fornece ainda mais versatilidade ao uso dos FPGAs, pois permite a programação de funções através de linguagens normalmente utilizadas em outras plataformas, como por exemplo em C. O uso de *Soft-Processors* pode ser aliado a outros componentes construídos em *hardware* ou mesmo ao uso de vários destes processadores no mesmo FPGA. Esta arquitetura ainda possibilita a execução de um sistema operacional (em alguns casos), que pode ser utilizado para criar tarefas de mais alto nível do ponto de vista do usuário, como interfaces gráficas.

Outra arquitetura possível é o uso de FPGAs SoCs que já contam com um *Hard-Processor* (processador que foi projetado e construído em *hardware* que não permite sua reconfiguração) no mesmo CI. Esta arquitetura além de ter as mesmas vantagens que um *Soft-Processor*, possibilita a execução de tarefas de maneira mais eficiente que um *Soft-Processor,* uma vez que é construído em hardware dedicado. Porém estes componentes não podem ser customizados como os construídos em FPGA.

3.3.2 *Kits* e Ferramentas

Existem diversos fornecedores de FPGAs no mundo, mas o mercado é dominado por duas empresas que juntas detêm cerca de 89% do mercado. Em

primeiro lugar está a empresa Xilinx, seguida da Intel Altera com 53% e 36% de domínio, respectivamente (PAUL DILLIEN, 2017).

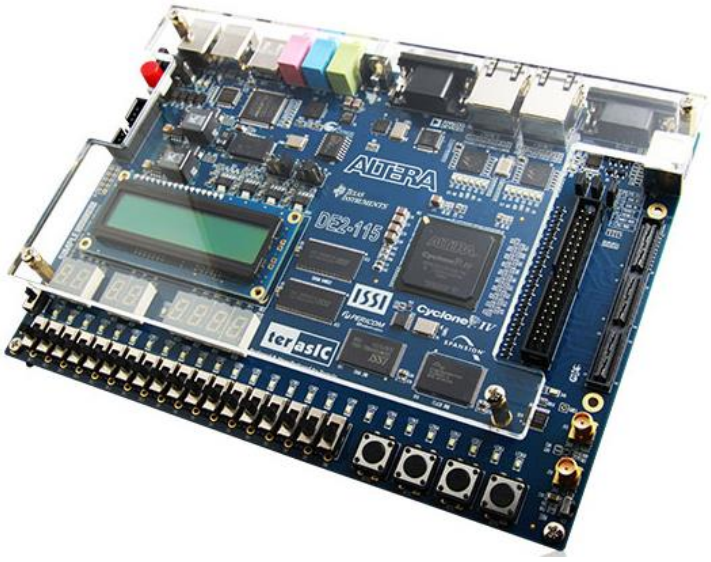
Mesmo entre os dispositivos do mesmo fabricante existe grande variedade de famílias de dispositivos e estas famílias, por sua vez, abrangem várias gerações e vários modelos de FPGAs. Existindo desde dispositivos mais básicos com um número menor de células lógicas e custo menor voltados a projetos pequenos de baixa complexidade a dispositivos voltados a projetos de grande complexidade que necessitem de grande capacidade de processamento.

Diversas empresas produzem *kits* de desenvolvimento, de forma que reúnem em uma única placa um FPGA, dispositivos de memória, LEDs, botões entre outros componentes e facilidade de acesso a pinos de entradas e saídas de forma a facilitar o desenvolvimento de sistemas que usem um FPGA.

Um *kit* amplamente utilizado é o DE2-115, que corresponde a uma plataforma de desenvolvimento baseada em FPGA da Altera e desenvolvida pela empresa Terasic que pode ser utilizada tanto para atividades de pesquisa, educação e desenvolvimento de produtos. O FPGA utilizado é da família *Cyclone* IV de modelo EP4CE115 construído com 114,480 elementos lógicos, 3,9Mb de RAM interna, 4 PLLs de uso geral e 266 multiplicadores de 18bits (ALTERA, 2016; DALGLEISH et al., 2017; TERASIC INC., 2013).

Na Figura 19 é mostrada uma imagem do *kit* de desenvolvimento.

Figura 19 - Placa de desenvolvimento DE2-115



Fonte: Terasic Inc. (2013)

Por se tratar de uma placa de desenvolvimento além do FPGA, esta conta com diversos outros recursos na mesma placa de forma a facilitar a utilização e agilizar o desenvolvimento de sistemas com FPGA. Entre os componentes disponíveis na placa para utilização, vale a pena destacar:

a) 18 switches e 4 *push-buttons*;
b) 18 LEDs vermelhos e 9 verdes;
c) 8 displays de 7 segmentos;
d) display LCD de 16X2;
g) dispositivos de memória: SDRAM 128MB, SRAM 2MB, Flash 8MB e 32Kbit EEPROM;
e) saída VGA com conversores digitais analógicos de alta velocidade;
f) socket para SD *Card*;
g) porta RS-232;
h) porta de expansão de 40 pinos de I/Os;
i) duas portas Ethernet;
j) codec de áudio de 24 bits;
k) portas USB A e USB B.

Além dos *kits* FPGA propriamente ditos, é comum que os fabricantes disponibilizem outros periféricos na forma de módulos compatíveis, como câmeras, visores, dentre outros. Um destes periféricos é o módulo D8M-GPIO, exibido na Figura 20, que corresponde a uma câmera, produzida pela Terasic, que permite a aquisição de imagens através de um conector 2X20 que é a interface padrão nas plataformas de desenvolvimento da fabricante, como o caso da DE2-115.

Figura 20 - D8M-GPIO

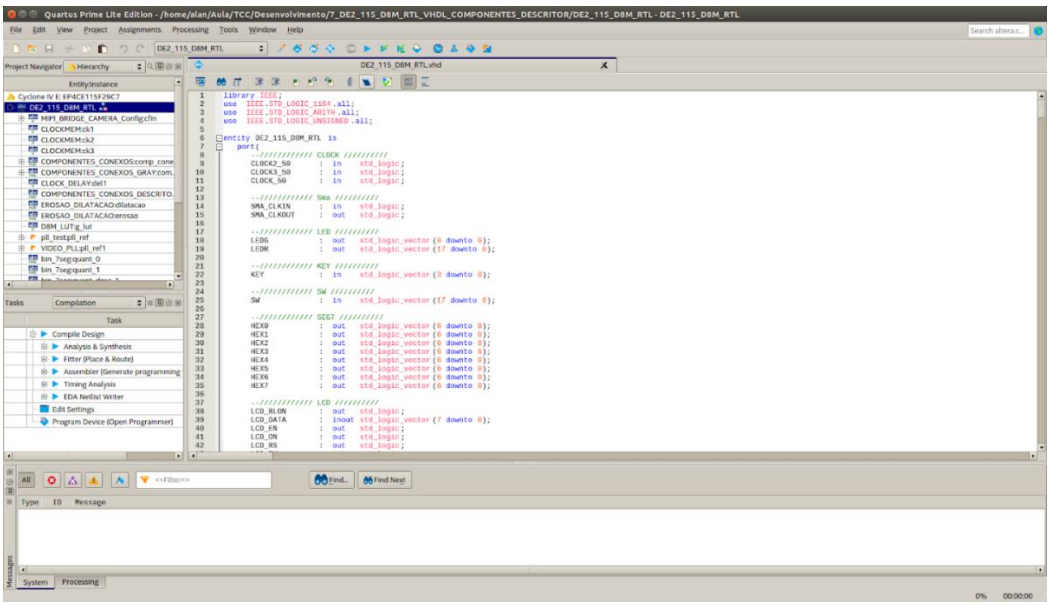
Fonte: Terasic (2018)

O módulo é formado por uma câmera com resolução de 8MP (3264x2448 pixels) com pixels arranjados em um filtro Bayer, interface MIPI e um decodificador MIPI para interface paralela pela qual é lida a informação digital da imagem (TERASIC INC., 2018b). A representação por blocos dos componentes da D8M-GPIO pode ser observada no Anexo A.

Como pode ser visto no diagrama em anexo, tanto a câmera como o conversor MIPI podem ser configurados via I2C. Algumas configurações possíveis são: resolução, qualidade e taxa de aquisição de imagens e formato de saída na interface paralela. A câmera também conta com ajuste de foco que pode ser controlado via I2C. Uma vez que foram configurados a câmera e conversor é possível adquirir frames na interface paralela fornecendo um *clock* de referência e ajustando alguns pinos de controle.

O desenvolvimento de soluções para os FPGAs normalmente é realizado fazendo uso de ambientes integrados de desenvolvimento. Um exemplo deste tipo de ferramenta é o Quartus, que corresponde a um *software* de desenvolvimento para dispositivos lógicos programáveis (PLD - *Programmable Logic Device*) produzido pela Intel. Esta ferramenta disponibiliza uma versão gratuita que tem suporte a famílias de dispositivos de baixo custo, como o caso da família *Cyclone IV* integrante do *kit* de desenvolvimento DE2-115. Na Figura 21 é exibido uma captura de tela da janela principal da ferramenta.

Figura 21 – Captura de tela do s*oftware* Quartus Prime



Fonte: Próprio autor

A ferramenta permite: verificação, síntese de linguagens de descrição de *hardware*, otimização, simulação do projeto e configuração de dispositivos a partir do hardware sintetizado. É possível o desenvolvimento de sistemas utilizando esquemáticos gráficos ou HDLs como: VHDL, Verilog e SystemVerilog.

Para simulação do sistema projetado o Quartus oferece conexão com a ferramenta *ModelSim* desenvolvido pela empresa *Mentor Graphics*. Esta ferramenta permite a simulação de linguagens HDL antes de serem sintetizadas em um dispositivo físico. Através do uso de *testbenchs* é possível produzir sinais de estímulo de modo a verificar o correto funcionamento do projeto.

3.4 Trabalhos Correlatos

Para o estudo da viabilidade do trabalho, para ter material de base e também para ter uma comparação com os resultados do trabalho, buscou-se na literatura trabalhos de processamento de imagem empregando FPGA. Abaixo são discutidos alguns dos trabalhos disponíveis na literatura.

No trabalho de Mcbader (2003) foi projetado uma arquitetura flexível e paralela voltada a processamento de imagens. Nessa arquitetura foram implementados algoritmos de pré-processamento e segmentação de imagem, tais como: filtragens, correlação e binarização. A imagem tinha como fonte uma câmera com resolução de 256x256 pixels e a escolha e configuração de processamentos era realizado através de um computador, onde também pode ser acessada a imagem resultante do processamento. Como resultados do trabalho, obteve-se um aumento de desempenho de alguns algoritmos de mais de 250% quando comparado ao mesmo algoritmo realizado em um computador. Em algoritmos mais simples como o de binarização, foi alcançado uma taxa de 666.67 *frames* por segundo (FPS).

No trabalho de Ribeiro (2007) foi desenvolvido um *hardware* capaz de realizar operações como: limiarização, aplicação de máscaras e filtros de suavização. Tanto a entrada e saída da imagem processada, além de escolha e parametrização dos processamentos é realizada por uma porta paralela ao qual o FPGA é conectado ao computador. Porém o processamento é realizado em imagens somente de 72x72 pixels em tons de cinza.

No sistema desenvolvido por Mertes (2012) além de filtros de suavização também foram implementados detecção de bordas, equalização de histograma,

normalização de cores e de luminância. Apesar da baixa resolução de 60x40 pixels, a implementação obteve bons resultados dos processamentos. A origem das imagens era uma memória externa Flash. Uma das falhas do trabalho é de exibir resultados somente simulados e não realizados no software gravado no FPGA.

No trabalho de Gupta (2014) foi desenvolvido um algoritmo de rotulação de componentes conexos de duas passadas baseada em árvores de decisão. O diferencial do trabalho é a implementação de varreduras em diferentes faixas da imagem de forma paralela, ao final das varreduras os resultados são fundidos. Com essa abordagem em paralelo foi possível diminuir o tempo de processamento consideravelmente.

No sistema implementado por Walczyk (2010) foi desenvolvido um *hardware* de rotulação de componentes conexos de somente uma passada que utiliza como fonte da imagem uma câmera infravermelha com resolução de 640x480 pixels e tem capacidade de processamento de 30 FPS. Após o processamento, a imagem é enviada para um computador via *bluetooth* e também é exibida em um monitor. Porém, a saída do módulo desenvolvido não é adequada para um grande número de inspeções pois suas saídas são somente o *bounding-box* dos componentes encontrados não fornecendo diversos outros descritores que são muito úteis.

4 DESENVOLVIMENTO DO SISTEMA PROPOSTO

Nesta seção será discutido o desenvolvimento do trabalho proposto, sendo também expostos os requisitos e uma visão geral do sistema implementado.

4.1 Requisitos Funcionais e Não Funcionais

O sistema desenvolvido é voltado a aplicações de inspeção automatizada comumente encontrados em ambientes industriais tais como verificação da presença de peças e partes, avaliação da dimensão de componentes e também contagem de elementos. Além disso, a utilização de um sistema de inspeção automatizada na indústria normalmente requer operação de forma integrada à linha de produção. Neste sentido, o início de cada inspeção é realizado por algum evento, ou gatilho, como a passagem de uma peça por uma esteira. Da mesma forma, após a análise, muitas vezes será necessário enviar um sinal associado ao resultado da avaliação automatizada, como por exemplo para acionamento de um sistema de rejeito que descartará uma peça defeituosa.

Em função disso foi realizada uma análise de recursos e funcionalidades necessários no sistema. Os requisitos definidos foram divididos entre requisitos funcionais e não funcionais e orientaram o desenvolvimento do presente trabalho. Os requisitos funcionais que são aqueles que definem as funções que o sistema deve ter, foram estabelecidos como os seguintes:

a) verificar a presença e ausência de elementos na imagem;
b) realizar contagem de objetos em uma imagem;
c) possibilitar a obtenção da posição de objetos presentes na imagem;
d) realizar medição de objetos presente em uma imagem.

Os requisitos não funcionais que por sua vez dizem respeito a como o sistema cumprira suas funções, foram definidos como os seguintes:

a) sistema compacto: ter dimensões menores que um computador de propósito geral.

b) possibilidade de integração com componentes externos: contar com *trigger* para processamento e saída sinalizadora de aprovação ou reprovação de inspeção.

c) exibição em um monitor VGA da imagem original, de cada uma das etapas e do resultado final do processamento.

Uma vez que os requisitos foram definidos, pode-se dar início no projeto do sistema, buscando-se atingir os todos os requisitos fixados.

4.2 Projeto e Implementação do Sistema

A primeira etapa do projeto do sistema foi a escolha dos componentes que seriam utilizados de forma que os requisitos pudessem ser atingidos. A escolha pela plataforma de desenvolvimento DE2-115 se fez por a mesma estar disponível no laboratório CSI (laboratório no qual o trabalho foi desenvolvido), ter capacidade lógica necessária, contar com componentes de memória e interface VGA úteis para processamento de imagem e exibição dos resultados e possibilitar a interface com o usuário e componentes externo por meio dos botões, chaves, display e pinos de I/O.

A escolha da câmera D8M-GPIO se deu pelo fato de ser compatível com a plataforma de desenvolvimento DE2-115, apresentar qualidade de imagem suficiente (resolução de 640x480 pixels a 60 FPS) para um software de inspeção e estar disponível no laboratório CSI.

Para a implementação utilizou-se o *software* Quartus Prime e linguagem de descrição de *hardware* VHDL. A escolha do *software* se fez por ser a ferramenta do próprio fabricante do FPGA e ser um pacote com diversas ferramentas e recursos úteis para o desenvolvimento de sistemas com FPGA. A linguagem VHDL foi escolhida devido a familiaridade e conhecimento do autor do presente trabalho.

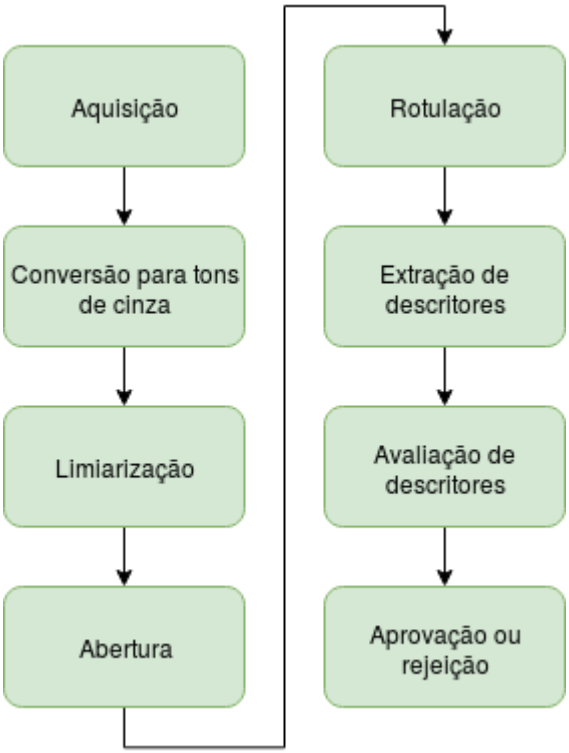
4.2.1 Descrição Geral do Fluxo de Processamento

Uma vez definidos os componentes que seriam utilizados, fizeram-se uma análise dos processamentos e algoritmos que seriam necessários para que os requisitos fossem alcançados. Ao final desta análise, concluiu-se que o processamento das imagens deveria seguir o fluxo de operações ilustrado pela Figura 22.

A aquisição da imagem compreende os processos de digitalização de imagens pela câmera, a transferência da informação formadora da imagem entre módulo D8M-GPIO e DE2-115 e a conversão entre o formato disponibilizado pela câmera (RAW) e o formato utilizado nos algoritmos de processamento (RGB).

Uma vez que a imagem já se encontra no formato correto, é realizado uma conversão de imagem colorida para tons de cinza de forma a diminuir a quantidade de informação por pixel da imagem.

Figura 22 - Fluxo do processamento de imagens definido



Fonte: Próprio autor

Na sequência, a imagem em tons de cinza sofre o processo de limiarização, com o objetivo de destacar os objetos de interesse do restante da imagem. De forma a corrigir defeitos na segmentação por limiarização, é realizada a operação morfológica de abertura.

Com a imagem devidamente segmentada, a mesma é submetida ao processo de rotulação por componentes conexos que tem como resultado a imagem com todos seus componentes discretizados de forma que possam ser extraídos os descritores de cada objeto sem a interferência de outros objetos presentes na imagem.

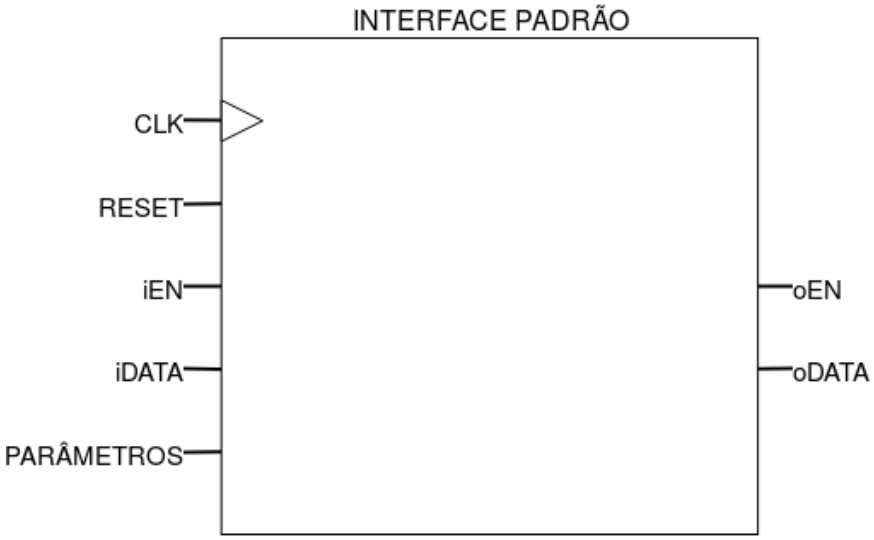
Após a extração dos descritores, estes podem ser analisados e comparados com os resultados que são esperados de forma a produzir um resultado de aprovação ou rejeição do objeto inspecionado.

4.2.2 Projeto Geral do Sistema

Como pode ser visto pelo fluxo de processamentos do sistema na Figura 22, para serem alcançados os resultados finais da inspeção o sistema pode ser dividido em diversos módulos diferentes específicos para cada processo. Por sua vez, estes módulos também podem ser subdivididos em outros módulos internos caso seja necessário. Esta divisão de um sistema é chamada de projeto hierárquico e tem como objetivo facilitar o desenvolvimento e entendimento do sistema como um todo.

O trabalho desenvolvido utilizou os conceitos de projeto hierárquico e de forma a facilitar o desenvolvimento, integração e possibilitar o uso em outros projetos, definiu-se uma interface padrão para os componentes de processamento de imagem criados. A interface de componentes definida pode ser visualizada na Figura 23.

Figura 23 - Interface padrão de componentes



Fonte: Próprio autor

Como pode ser visto na imagem, os componentes de processamento desenvolvidos contam entrada de *clock* (CLK), reset, habilitação de processamento (iEN), entrada de dados (iDATA) e parâmetros de ajuste. A interface também fornece uma saída de dados (oDATA) e uma habilitação de saída (oEN) que sinaliza para o próximo bloco a saída de dados processados validados.

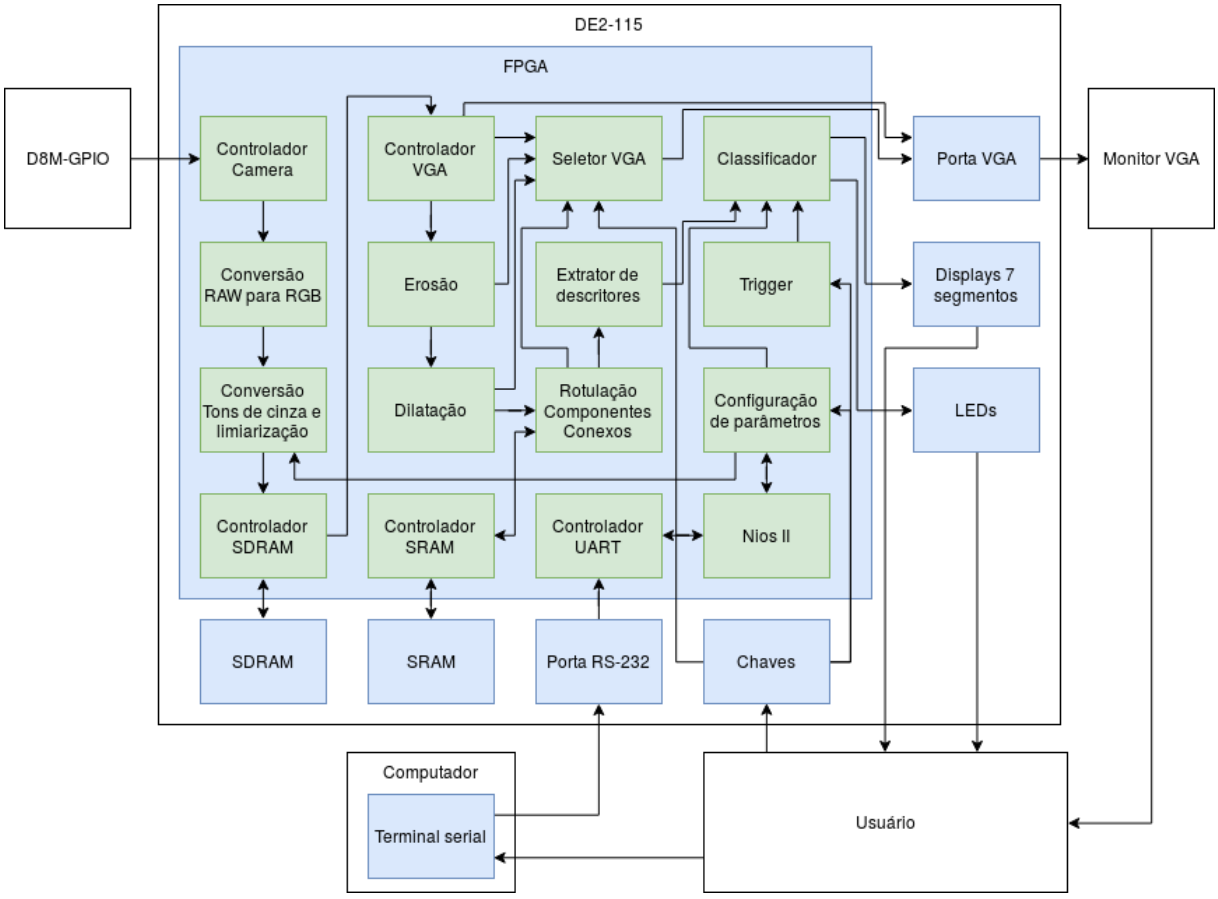
Os pinos contidos na interface foram pensados para utilização na forma de *streaming* onde a entrada de dados de um bloco é ligada diretamente na saída de dados de outro. Da mesma forma que a habilitação de saída é conectado a habilitação de processamento de outro componente indicando o momento que este pode considerar os dados válidos para processamento.

4.2.3 Implementação do Sistema

Após ser realizada a análise de requisitos, de serem definidos os recursos necessários e do projeto do sistema como um todo, iniciou-se a etapa de implementação do sistema proposto.

O desenvolvimento foi iniciado por um projeto de referência disponibilizado pelo fabricante do módulo D8M-GPIO. A demonstração do fabricante realizava a aquisição de imagens e exibição em um monitor, sem nenhum processamento sobre a imagem. Após a desenvolvimento dos algoritmos necessários o resultado final da implementação pode ser simplificado na forma de diagrama de blocos exibido na Figura 24.

Figura 24 - Diagrama de blocos do trabalho desenvolvido



Fonte: Próprio autor

Como pode ser visto na Figura 24, o sistema foi dividido em diversos módulos que interagem entre si e com componentes externos como memórias e chaves ajustadas pelo usuário presentes na plataforma de desenvolvimento. Os blocos de *hardware* sintetizados no FPGA também se comunicam com a câmera D8M-

GPIO, com um monitor VGA e com um computador via interface serial para configuração de parâmetros. Os módulos formadores do sistema serão melhor explicados na sequência.

4.2.4 Aquisição de Imagem

Para a etapa de aquisição de imagem, utilizou-se a implementação disponibilizado no projeto de referência do fabricante da câmera, porém foram necessárias algumas alterações. Na demonstração utilizada, além da aquisição da imagem, já era realizada sua exibição em um monitor VGA com resolução de 640x480 pixels a uma taxa de 60 *frames* por segundo. O Anexo B mostra o diagrama de blocos do projeto usado como base para o desenvolvimento.

Como pode ser visto na imagem, disponível em anexo, a demonstração realiza a configuração da câmera via I2C e envia um *clock* de referência, recebe os valores dos pixels via interface paralela, sinais de *clock* e de sincronização horizontal e vertical. Os valores dos pixels recebidos são guardados em uma memória SDRAM externa ao FPGA, que é lida de acordo com a necessidade de interface VGA. Porém, antes de ser enviada para a saída VGA, a imagem necessita de uma conversão do formato RAW fornecido pela câmera para o formato RGB que é utilizado no padrão VGA.

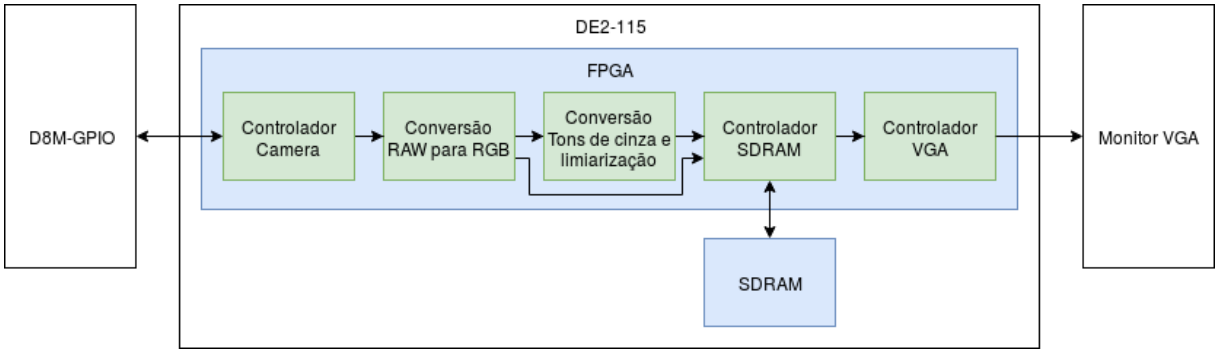
A conversão entre RAW e RGB se deve pela saída da câmera fornecer os valores lidos de cada um dos pixels do sensor de captura, que é formado por um padrão Bayer, de forma bruta. No formato RAW, como já explicado na seção 3.1.2, cada pixel contém somente o valor de um dos componentes RGB e a conversão de formatos é realizado pela interpolação dos componentes entre pixels vizinhos.

O exemplo também conta com um controlador VGA (*Video Graphics Array*) no qual além de controlar as solicitações de pixels a memória SDRAM, também gera sinais de controle da interface VGA, como os sinais de sincronização vertical e horizontal.

A demonstração descrita foi modificada de forma a facilitar os processamentos da imagem. Uma das principais alterações realizadas foi a alteração da ordem entre os blocos de conversão RAW para RGB e controlador SDRAM, de forma a guardar-se a imagem já no formato RGB. Outra foi a inserção de um módulo de conversão de RGB para tons de cinza entre a conversão RAW para RGB e

controlador SDRAM, este componente será melhor detalhado na próxima seção. O diagrama de blocos do exemplo após as alterações pode ser visualizado na Figura 25.

Figura 25 - Diagrama de blocos da demonstração modificada



Fonte: Próprio autor

Comparando-se os diagramas original e com as alterações realizadas é possível ver que entre as demais alterações realizadas está a remoção do componente responsável pelo autofoco. Esta função foi julgada como não necessária para o sistema e por conta disso removida de forma a liberar recursos para outros elementos necessários.

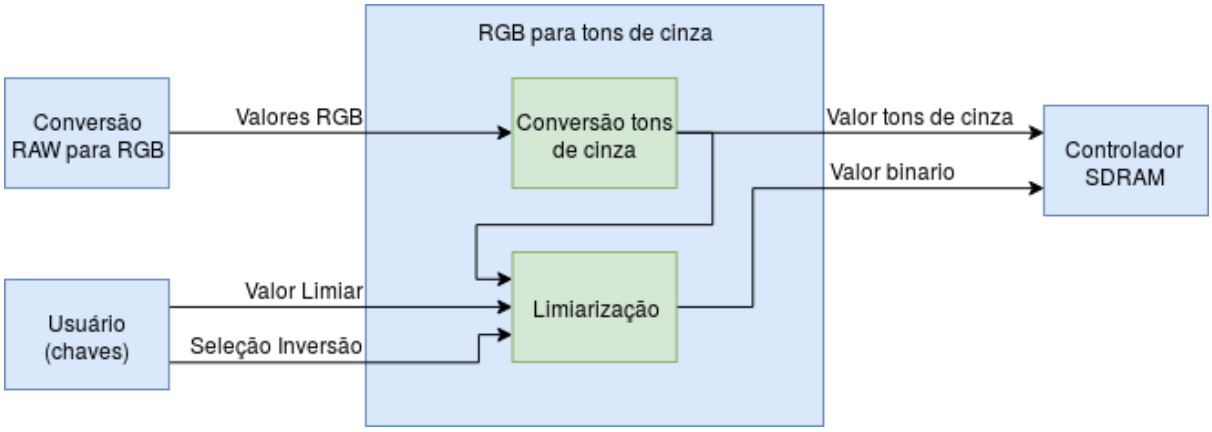
4.2.5 Conversão RGB para Tons de Cinza

Um componente de conversão de cor foi criado para realizar a conversão do sistema de cor RGB para tons de cinza. Para a conversão foi utilizado o método da luminosidade expresso pela Equação 2.

No bloco de conversão para tons de cinza também foi inserido a limiarização da imagem. O limiar pode ser ajustado pelo usuário do sistema utilizando 8 dos switches disponíveis na DE2-115 ou via interface serial. A saída da limiarização pode ser invertida (1 torna-se em 0 e 0 torna-se 1) pelo ajuste de uma chave ou pela interface serial. A inversão se torna útil para que seja possível a segmentação de objetos tanto para elementos claros em fundos escuros quanto para elementos escuros em fundos claros. Na Figura 26 pode ser visto o diagrama de blocos da conversão RGB para tons de cinza.

Como já descrito a conversão para tons de cinza foi inserida no exemplo base, entre os módulos de conversão RAW e o controlador SDRAM para que fosse guardada a imagem nos formatos RGB, tons de cinza e binarizada.

Figura 26 - Diagrama de blocos conversão RGB para tons de cinza



Fonte: Próprio autor

A SDRAM disponível na DE2-115 é formada por dois dispositivos físicos que compartilham o mesmo barramento de controle e endereços. Cada um dos elementos de memória isolados conta com 64MB organizados em endereços com 16 *bits* de dados. O controlador SDRAM gera os sinais de controle e endereços e por meio de FIFOs (*First In, First Out* - Primeiro a entrar, primeiro a sair) controla a informação escrita e lida na memória.

Desta forma, a utilização do controlador permite que sejam escritos e lidos 32 *bits* das duas SDRAMs. Os 32 *bits* foram utilizados da seguinte forma: 8 *bits* para cada um dos componentes RGB, 7 para o valor em tons de cinza (7 *bits* mais significativos dos 8 totais da conversão) e 1 para o valor do pixel após limiarização. A Figura 27 representa a distribuição dos valores nos 32 *bits* formadores de um quadro de memória.

Figura 27 - Formação de um quadro da memória SDRAM

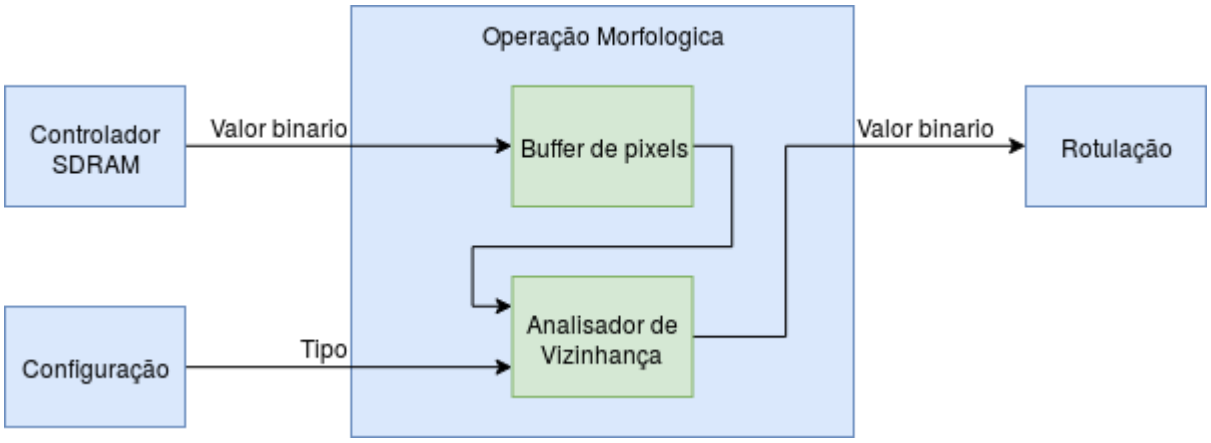
Fonte: Próprio autor

Uma vez que a imagem foi salva na SDRAM, o controlador VGA pode solicitar a sua leitura a exibição em um monitor.

4.2.6 Operações Morfológicas

Ao ser lido um endereço da SDRAM, em solicitação do controlador VGA, o valor do pixel binarizado é enviado para as operaçde pós-processamento. Para a etapa de pós-processamento, desenvolveu-se um módulo para operações morfológicas que pode ser configurado para dilatação ou erosão. Utilizando-se duas instâncias em sequência é possível realizar operações de abertura e fechamento. O diagrama de blocos do componente criado é mostrado na Figura 28.

Figura 28 - Diagrama de blocos operação morfológica



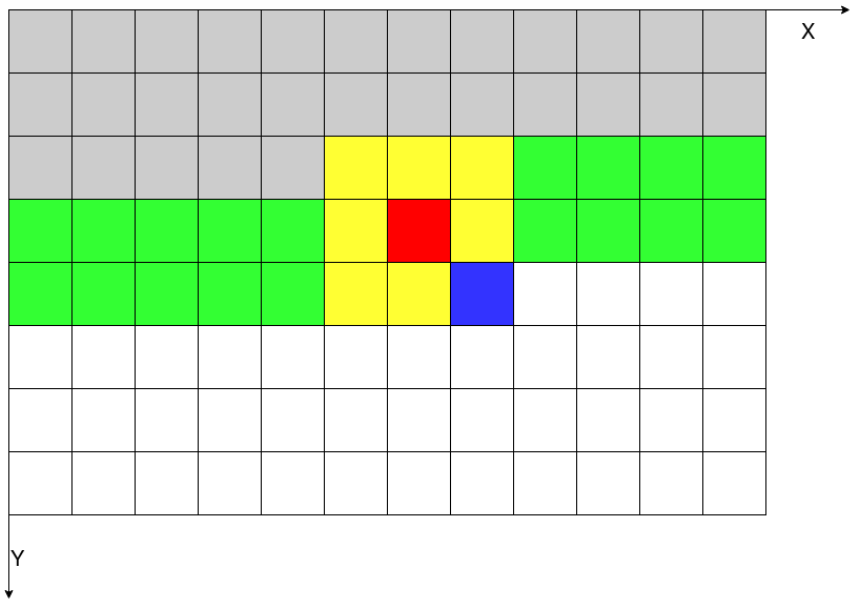
Fonte: Próprio autor

Devido ao fato das operações morfológicas utilizarem a 8-vizinhança para analisar o valor que um pixel assumirá, é necessário que se tenha acesso a todos os pixels da vizinhança no momento da análise. Dessa maneira cada operação morfológica provoca um atraso de uma linha e um pixel no streaming de pixels, até o enchimento de um buffer e todos os vizinhos estarem disponíveis.

O atraso provocado pela operação morfológica pode ser mais facilmente entendido observando-se a Figura 29, onde a imagem é preenchida da esquerda para a direita (eixo X) e de cima para baixo (eixo Y), o pixel a ser analisado é o representado pela cor vermelha, o último pixel vindo de processos anteriores é o destacado em cor azul e os pixels amarelos em conjunto com o azul formam a vizinhança do pixel analisado.

Desta forma, para que os pixels da vizinhança mais o pixel alvo estejam acessíveis no momento da análise é necessário que todos os pixels representados pelas cores: amarelo, verde e vermelho tenham sido armazenados em um buffer no momento que ficaram ativos na entrada do bloco morfológico.

Figura 29 - Atraso operação morfológica

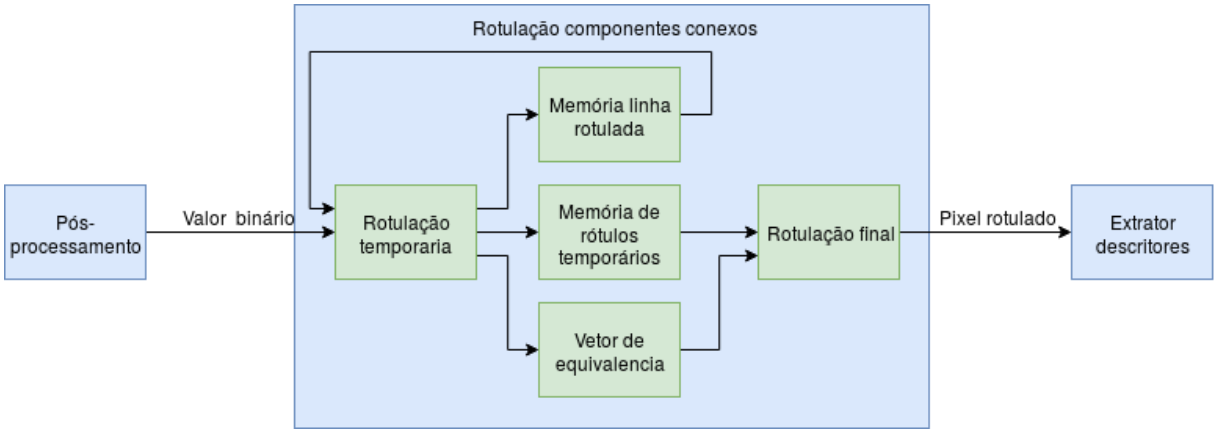


Fonte: Próprio autor

4.2.7 Rotulação de Componentes Conexos

Uma vez realizada a etapa de pós-processamento, o fluxo de pixels segue para a operação de rotulação de componentes conexos. A implementação seguiu o algoritmo de duas passadas descrito na seção 3.1.6, com o conceito de 4-vizinhança e pode ser simplificada para o diagrama de blocos exibido na Figura 30.

Figura 30 - Diagrama de blocos componentes conexos



Fonte: Próprio autor

Como pode ser visto no diagrama de blocos, cada novo pixel oriundo de um bloco externo (pós-processamento) é analisado e tem um rótulo temporário atribuído de acordo com os rótulos de seu vizinho superior e vizinho lateral esquerdo já rotulados e armazenados. Após ser atribuído um rótulo temporário, este valor é

guardado na memória de linha rotulada de forma que quando um pixel for analisado seu vizinho superior e lateral já rotulados estejam acessíveis.

O rótulo temporário atribuído também é salvo em uma memória de rótulos temporários que contém toda a imagem com rótulos temporários. A memória de rótulos temporários é lida pela rotulação final que utiliza o vetor de equivalência para gerar uma imagem onde cada componente seja rotulado sem ambiguidades.

O uso de duas memórias se faz necessário para que os processos de rotulação temporário e final ocorram em paralelo, pois cada memória utilizada tem somente uma porta de leitura e uma de escrita.

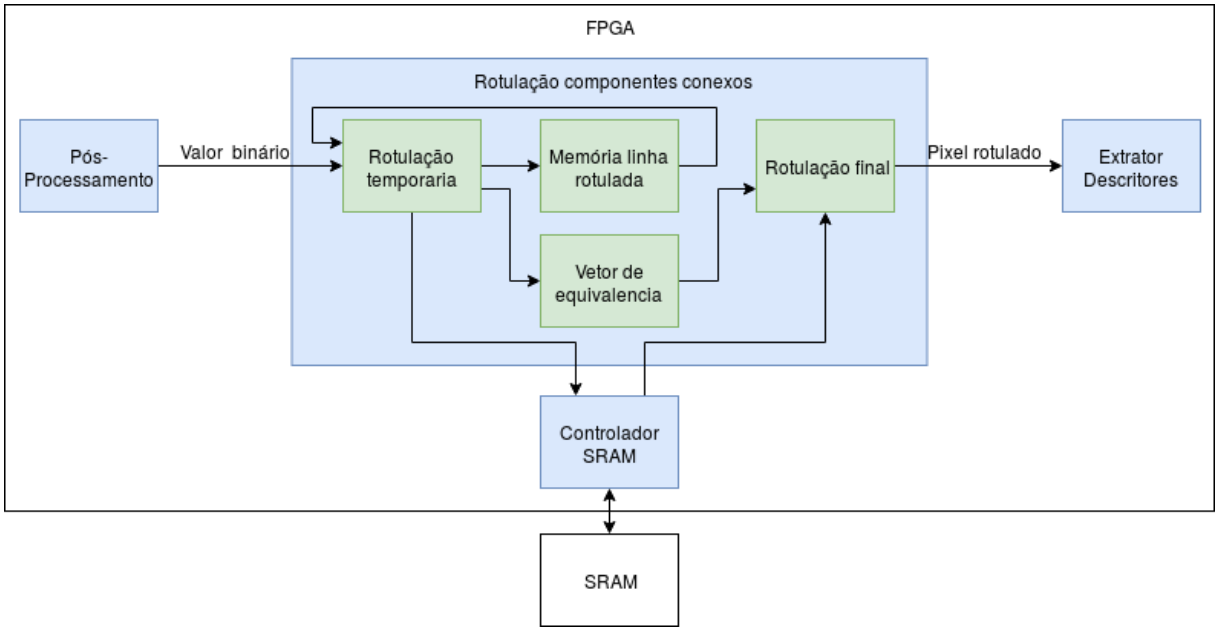
O vetor de equivalência é o responsável por guardar a informação de que dois ou mais rótulos são pertencentes ao mesmo objeto. O vetor é preenchido a cada vez que um pixel é considerado pertencente a um objeto e seus dois vizinhos foram rotulados com valores diferentes. O vetor de equivalência é lido na etapa de rotulação final de forma que um objeto seja rotulado por inteiro por somente um único valor.

Devido a limitações de memória interna do FPGA na qual foi implementada a memória dos rótulos temporários e a memória de linha rotulada, utilizou-se somente 8 *bits* para o valor de um rótulo. Desta forma, somente são permitidos 255 rótulos diferentes em uma imagem. Este valor pode ser baixo para imagens com geometria muito complexas, como espirais com muitas voltas, porém para a grande maioria das aplicações é suficiente.

Após a integração do módulo de rotulação ao sistema, o mesmo necessitava da alocação de uma grande parte da memória (cerca de 60%), acarretando em grande demora na compilação do projeto e inviabilizando a implementação de outros componentes que necessitem grande quantidade de memória.

Como uma forma de contornar este problema, o módulo foi reprojetado de forma a utilizar a memória externa SRAM disponível na DE2-115. O diagrama de blocos desta implementação é mostrado na Figura 31.

Figura 31 - Diagrama de blocos da rotulação com memória SRAM



Fonte: Próprio autor

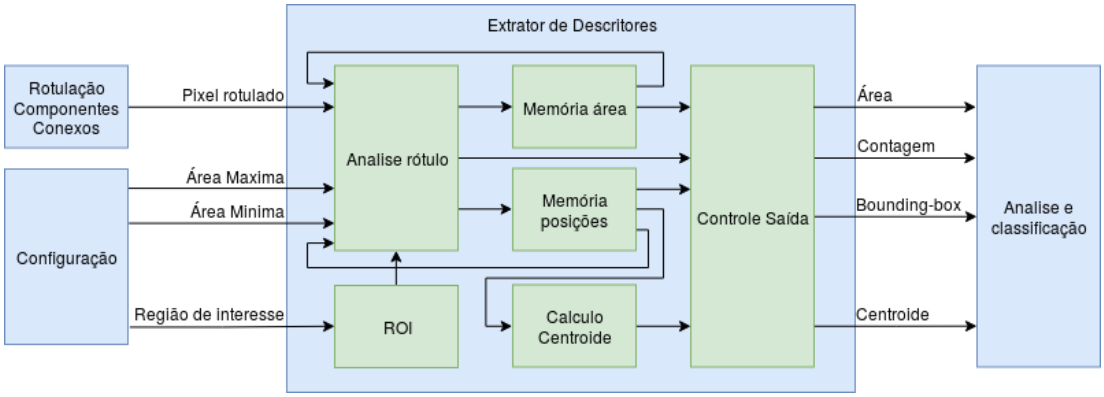
Como pode ser visto, a memória de rótulos temporários foi substituído pela memória externa SRAM. Porém, essa mudança implicou na necessidade da criação de um controlador para possibilitar a leitura e escrita na SRAM. Devido à grande redução de recursos de *hardware* em comparação com a utilização somente de memória interna, a rotulação com uso da SRAM foi utilizada no sistema.

Ambas as implementações se basearam no algoritmo de rotulação de duas passadas. Portanto, uma imagem é corretamente rotulada somente na segunda passada sobre a mesma, após todos os rótulos temporários serem atribuídos e o vetor de equivalência ser preenchido. Desta forma, a rotulação da imagem provoca um atraso de um *frame* no processamento da imagem.

4.2.8 Extração de Descritores

A implementação da rotulação tem como saída os pixels rotulados e uma contagem dos componentes presentes na imagem. Embora a quantidade de componentes já possa ser utilizada para avaliar a imagem, não se extraiu nenhum descritor dos objetos. Para a extração dos atributos de cada um dos componentes presentes na imagem, desenvolveu-se um componente do sistema capaz de obter os seguintes descritores: área, *bounding-box* e centroide. A Figura 32 representa o diagrama de blocos do extrator de descritores.

Figura 32 - Diagrama de blocos extrator de descritores



Fonte: Próprio autor

Como pode ser visto no diagrama, implementou-se também um filtro por área de componentes de forma que a contagem e extração de características de objetos somente seja aplicado aos que tem área dentro da faixa definida. Este filtro permite, por exemplo, ignorar pequenos elementos produzidos por ruídos que não foram eliminados na etapa de segmentação e pós-processamento.

Também foi implementado uma função de região de interesse (ROI), pela qual é possível definir uma região que contém os objetos de interesse, ignorando todos os componentes e seus descritores fora da zona de interesse. Tanto o filtro por área como posição e dimensão da região de interesse podem ser ajustados de acordo com a necessidade.

4.2.9 Análise de Descritores

Uma vez que os descritores foram extraídos dos objetos presentes na imagem, pode ser realizada análise sobre os mesmos de forma a classificar o objeto como aprovado ou rejeitado. Devido à grande quantidade de modos de classificação dos atributos, buscou-se desenvolver um componente que pudesse atender um grande número destas formas e que ao mesmo tempo pudesse ser reconfigurado. A reconfiguração se faz necessária para possibilitar inspeção de diferentes objetos ou mudança de padrão do que é considerado aceitável.

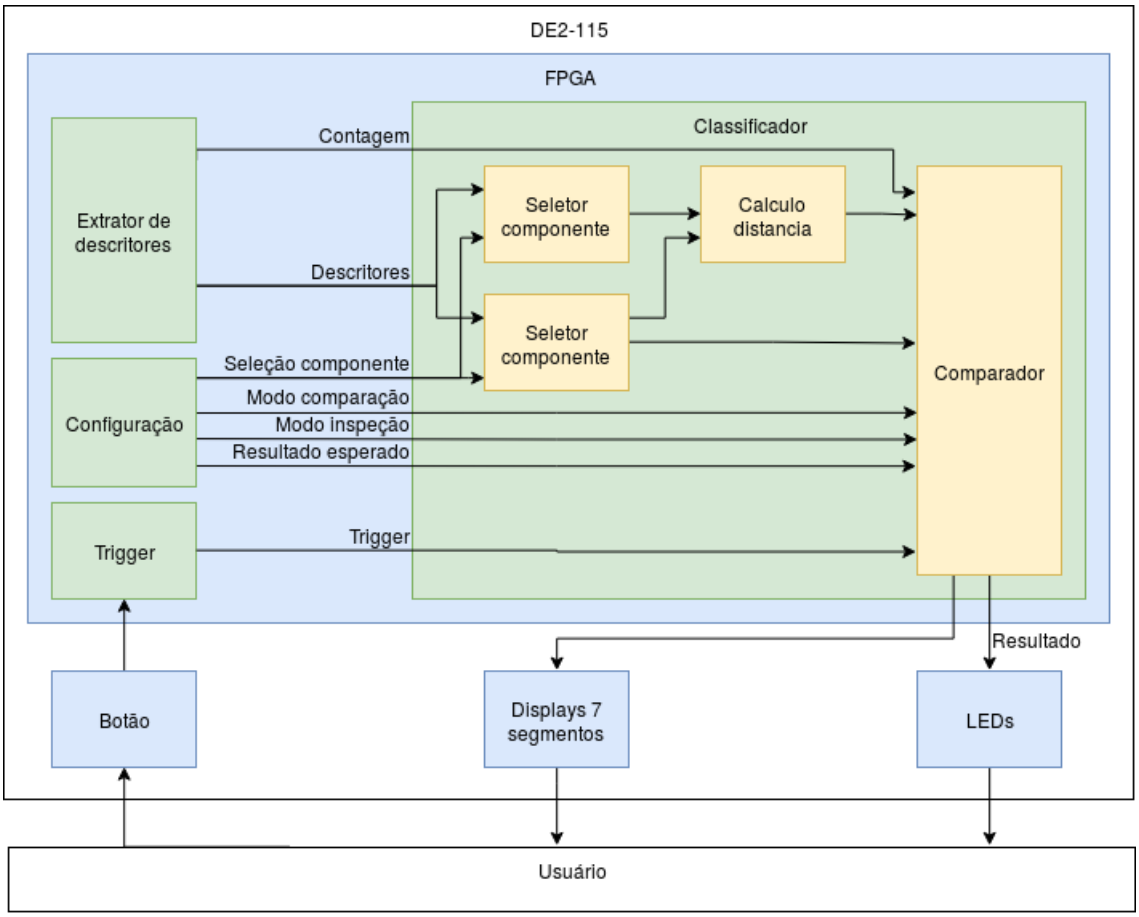
Outro elemento implementado, responsável pela obtenção do resultado de inspeção e que faz parte dos requisitos é um *trigger* (gatilho) que permite uma interface com o mundo externo, pelo qual é possível informar o momento que é necessário realizar a inspeção na imagem. Ao fim do processamento da imagem

obtida após o gatilho, a contagem e descritores extraídos dos objetos podem ser submetidos a análise.

A classificação da imagem pode ser enviada ao mundo externo por meio de um pino de saída da DE2-115 ou sinalizado através de um dos LEDs disponíveis na plataforma. Este sinal de saída poderia ser enviado a um atuador ou informar o operador da existência do problema.

Estes componentes responsáveis pelo resultado de uma inspeção podem ser representados pelo diagrama de blocos apresentado na Figura 33.

Figura 33 - Diagrama de blocos de componentes responsáveis pela obtenção do resultado de uma inspeção



Fonte: Próprio autor

Como pode ser visto no diagrama de blocos, o classificador é composto por dois blocos de seleção de componente que tem como objetivo selecionar os descritores de um componente específico, um bloco de cálculo de distância entre dois pontos da imagem e um comparador que pode ser ajustado para diferentes tipos de análise. A configuração do modo do bloco de classificação pode ser ajustada por mudar o sinal modo de inspeção de acordo com a Tabela 1.

Tabela 1 - Ajuste modo de inspeção

Descritor analisado	Quantidade	Área	Posição	Distância
Valor modo de inspeção	0	1	2	3

Fonte: Próprio autor

Uma vez ajustado o modo de inspeção, o descritor selecionado é comparado com o valor configurado no sinal de resultado esperado. O modo de comparação é ajustado pelo valor do modo de comparação de acordo com a Tabela 2.

Tabela 2 - Ajuste modo de comparação

Comparação com resultado	Igual	Diferente	Maior	Menor
Valor modo de comparação	0	1	2	3

Fonte: Próprio autor

Desta forma, ajustando os parâmetros necessários, é possível ajustar o sistema para diversos tipos de inspeções. O sistema também pode ser implantado em uma linha de produção pela utilização do *trigger* e sinal de resultado de análise.

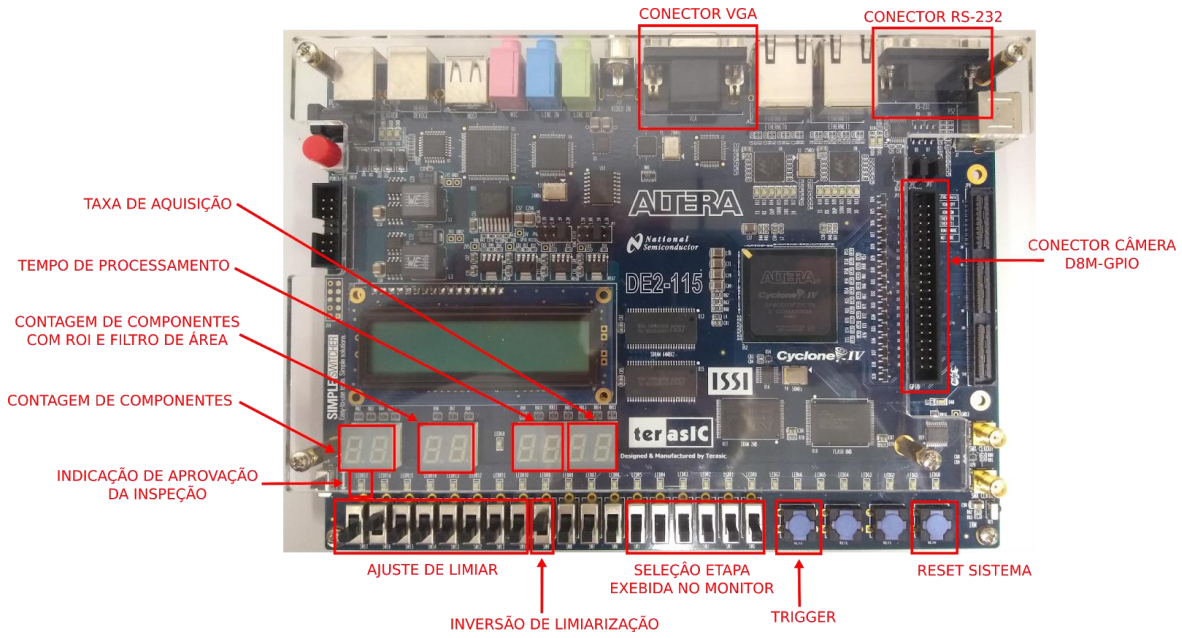
4.2.10 Interação com o Usuário

Como foi descrito, o sistema foi desenvolvido de forma a permitir a interação com o usuário por meio dos recursos disponíveis na plataforma de desenvolvimento. Na Figura 34 é possível observar a localização na DE2-115 dos componentes responsáveis por essa interação.

Como pode ser visto, o usuário pode ajustar o valor do limiar de segmentação pelo uso de oito chaves da placa, com outra chave é possível inverter a limiarização. Em dois displays de sete segmentos são exibidos os valores das contagens realizadas em representação hexadecimal.

Também é possível simular a interação do sistema por meio de um botão de *trigger* que poderia ser um sensor indicando a passagem de uma peça por exemplo. Uma vez que o *trigger* é pressionado e os processamentos na imagem são concluídos, um LED indica se o conteúdo da imagem foi considerado aprovado ou não. O LED poderia ser facilmente trocado por um atuador que faria o descarte automático de itens com defeito.

Figura 34 - Interface com o usuário

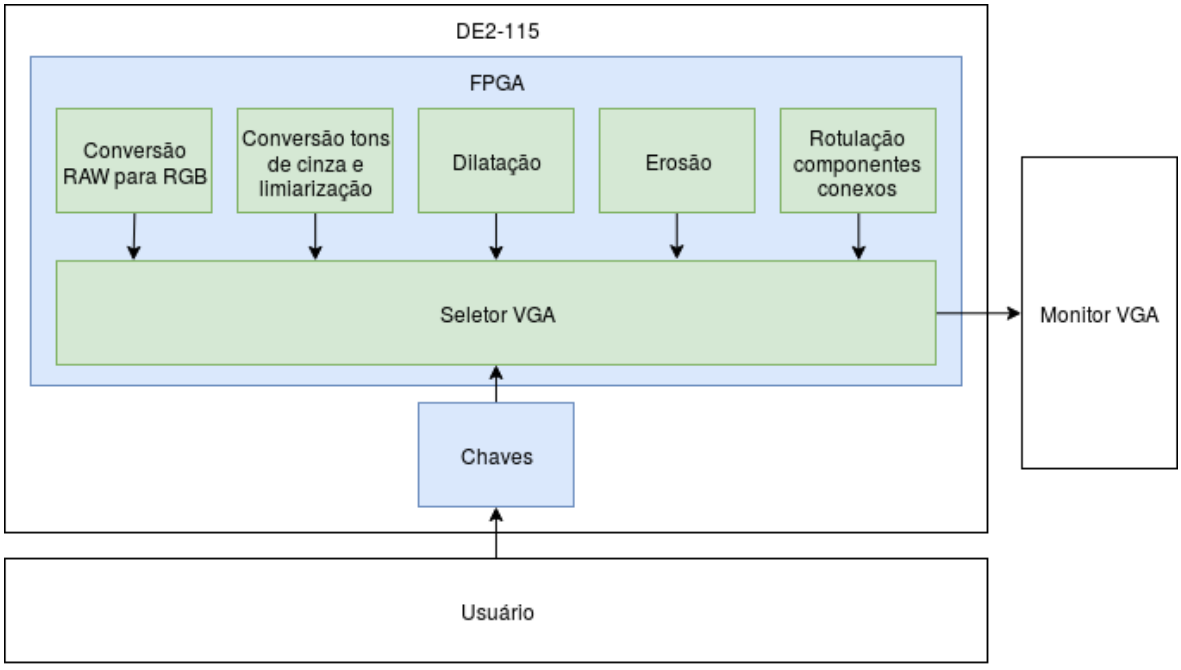


Fonte: Próprio autor

4.2.10.1 Interface para exibição das etapas de processamento

Para que fosse satisfeito o requisito de exibir todas as etapas do processamento em um monitor, implementou-se um seletor de fonte de vídeo, que permite que o usuário alterne entre a exibição de cada uma das etapas, desde a imagem original vinda da aquisição até a etapa de rotulação onde os descritores são extraídos. O Bloco pode ser representado pelo diagrama apresentado na Figura 35.

Figura 35 - Diagrama de blocos seletor VGA



Fonte: Próprio autor

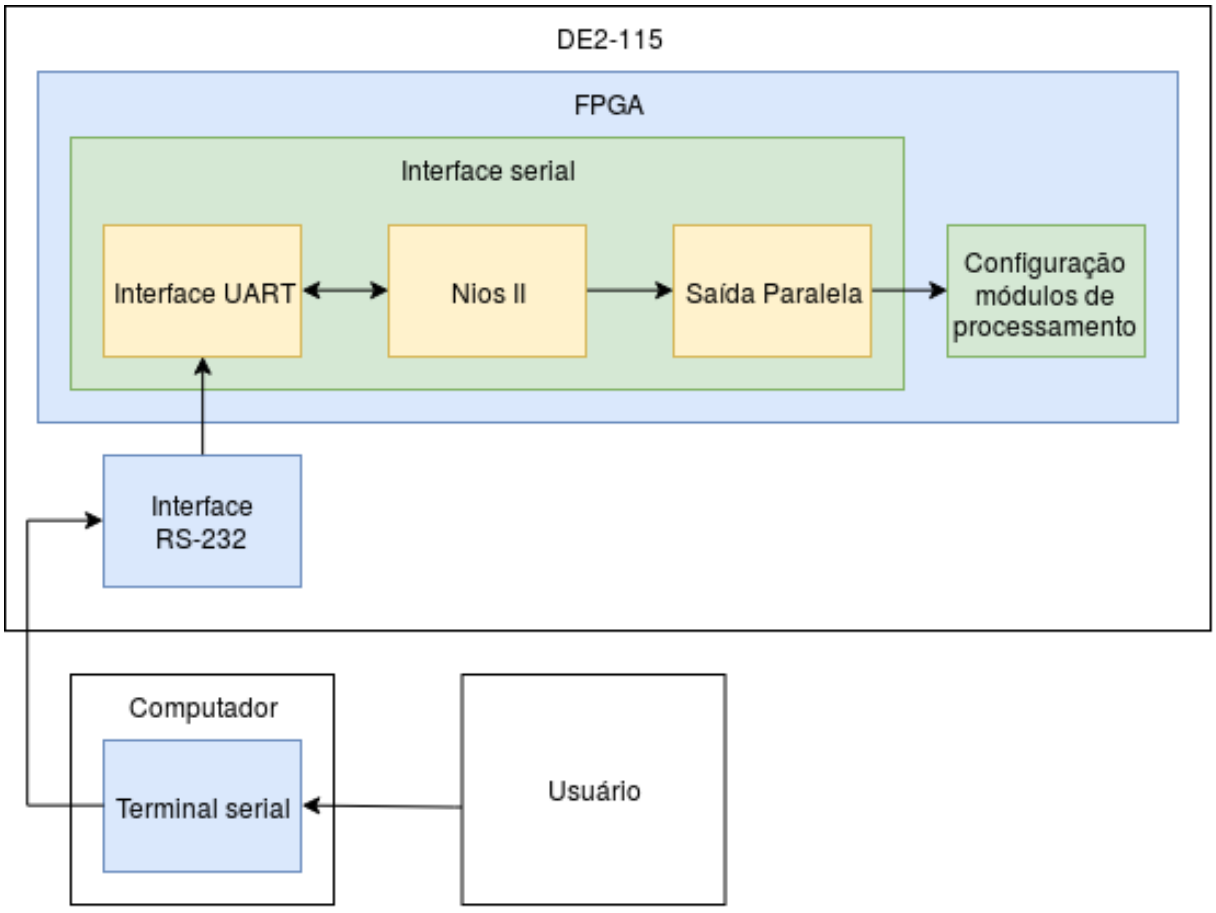
Como pode ser visto na Figura 35, a fonte da imagem exibida no monitor pode ser facilmente configurada pelo usuário através de ajuste de cinco chaves presentes na placa de desenvolvimento DE2-115.

4.2.10.2 Parametrização do fluxo de processamento

Como pode ser visto na descrição dos módulos anteriores, existem muitos parâmetros que o usuário pode alterar de forma a permitir a inspeção de diferentes objetos. Devido ao número limitado de chaves e botões presentes na DE2-115, nem todos os parâmetros poderiam ser alterados de forma simples e intuitiva.

De forma a permitir o ajuste de todas as configurações, desenvolveu-se um módulo para realizar a comunicação com um computador, do qual os parâmetros podem ser enviados. O diagrama de blocos do componente criado é exibido na Figura 36.

Figura 36 - Diagrama de blocos Interface Serial



Fonte: Próprio autor

No componente criado, utilizou-se o *soft-processor* Nios II para gerenciar uma interface serial e realizar a leitura dos dados enviados pelo usuário do sistema

de um computador executando um terminal serial via RS-232. Como existem diversos parâmetros, definiu-se um protocolo de comunicação que permitisse o ajuste de todos. Este protocolo é formado por uma linha onde o primeiro caractere (uma letra) indica o parâmetro a ser alterado e, na sequência, o novo valor da configuração. Na Tabela 3 é exibido o protocolo de comunicação implementado.

Tabela 3 - Protocolo de comunicação serial

Configuração	Caractere	Valor padrão	Valor mínimo	Valor máximo
Habilitação de configuração serial	S	0	0	1
Valor limiar	L	128	0	255
Inversão limiar	I	0	0	1
Área mínima filtro	A	0	0	307200
Área máxima filtro	B	307200	0	307200
ROI X 1	X	0	0	640
ROI Y 1	Y	0	0	480
ROI X 2	U	640	0	640
ROI Y 2	V	480	0	480
Seletor 1	E	1	0	50
Seletor 2	D	2	0	50
Trigger	T	0	0	1
Modo inspeção	M	0	0	3
Modo comparação	C	0	0	3
Valor comparação	R	0	0	307200

Fonte: Próprio autor

Para a decodificação do protocolo, o Nios II é carregado com um programa capaz de realizar tal tarefa. Uma vez que a mensagem foi entendida, o valor definido é enviado para uma saída paralela que é interligada a cada um dos componentes afetados pelo parâmetro modificado.

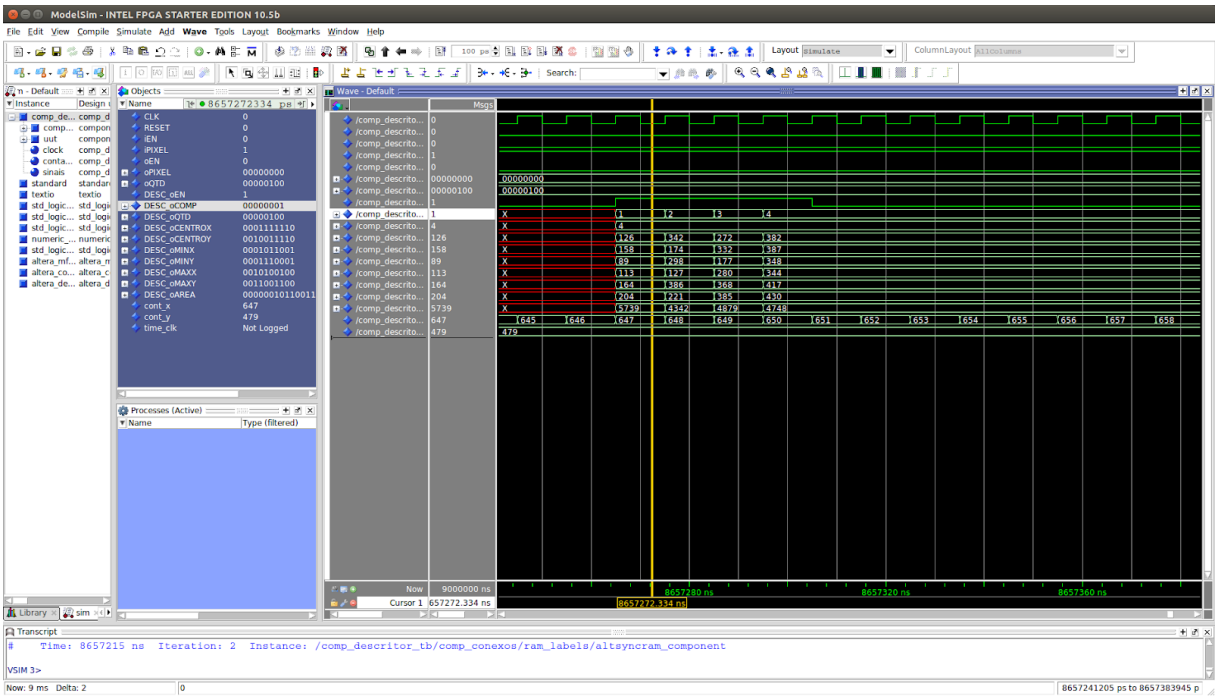
Uma configuração não mencionada anteriormente é a habilitação de configuração serial que tem como objetivo alternar entre a configuração via serial ou

chaves nos parâmetros que podem ser alterados pelos dois modos. Desta forma, o valor e inversão da limiarização, por exemplo, pode ser ajustado pelas chaves ou pela serial.

4.3 Simulação do Sistema

De modo a facilitar o desenvolvimento, diminuir tempos de compilação e encontrar problemas no sistema implementado, foi utilizado a ferramenta *ModelSIM* para simular cada um dos componentes criados e analisar o seu correto funcionamento. Na Figura 37 é mostrada uma das simulações realizadas para os componentes de rotulação e extração de descritores.

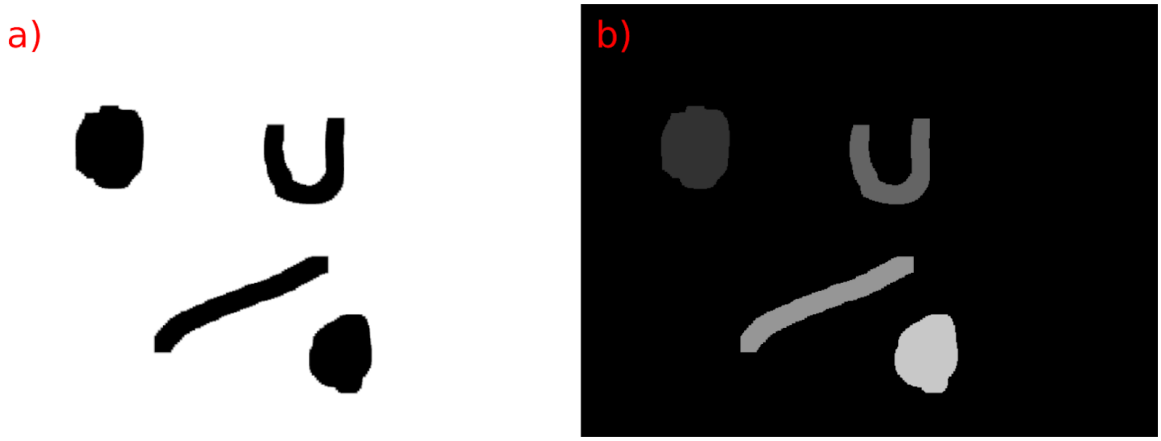
Figura 37 - Simulação dos componentes de rotulação e extração de atributos



Fonte: Próprio autor

Para fornecer os sinais de estímulo para a simulação, adotou-se uma imagem de teste. Esta imagem foi binarizada e o valor de seus pixels escritos em um arquivo de texto. No arquivo de *testbench* implementado, o arquivo de texto é lido e os valores do pixel enviados para a rotulação como se fossem o fluxo de pixel do pós-processamento. Ao fim da rotulação, era salvo, em outro arquivo, os valores de todos os pixels após rotulação. Na Figura 38 é mostrada a imagem de teste e o arquivo de saída da simulação convertido novamente para imagem.

Figura 38 - Imagens Simulação: a) Imagem de teste binarizada; b) Imagem rotulada



Fonte: Próprio autor

Uma vez que os componentes funcionaram em suas simulações, os mesmos foram integrados ao sistema e sintetizou-se um *hardware* que foi gravado no FPGA para testes práticos.

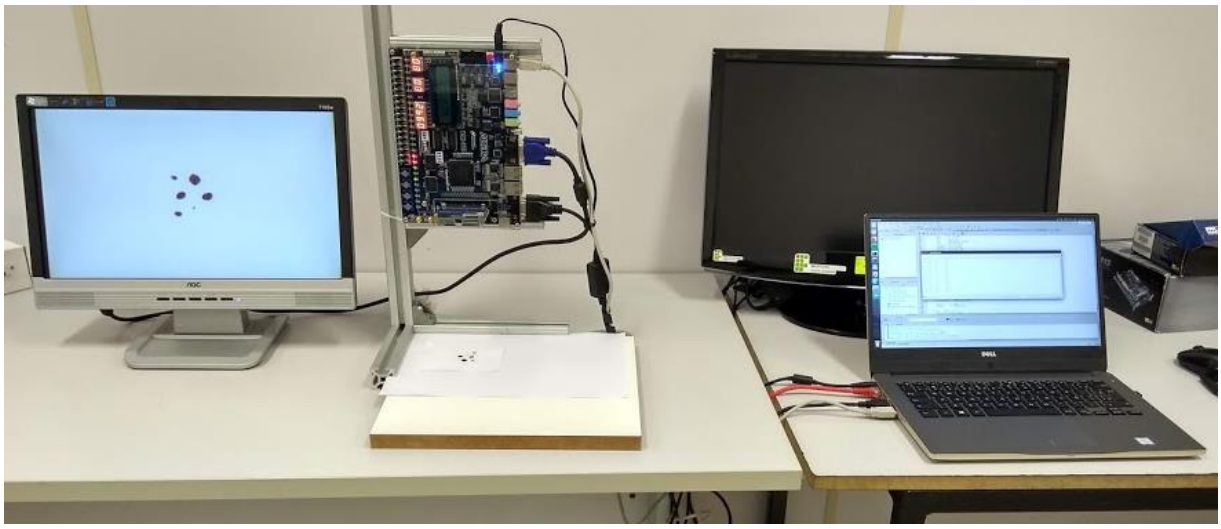
5 RESULTADOS EXPERIMENTAIS

Nesta seção serão apresentados e discutidos os testes realizados e resultados encontrados na etapa de validação do sistema desenvolvido.

5.1 Avaliação Geral

Para avaliar o funcionamento do sistema implementado, gravou-se o *hardware* sintetizado no FPGA da DE2-115 e montou-se o sistema com a câmera D8M-GPIO e saída VGA conectada a um monitor. A montagem do sistema e aparato utilizado durante os testes pode ser observado na Figura 39.

Figura 39 - Sistema montado para testes

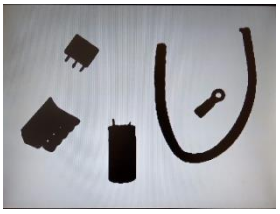
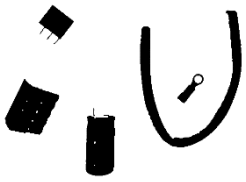


Fonte: Próprio autor

O sistema montado, além da DE2-115 e D8M-GPIO, é formado por um monitor para exibição das imagens adquiridas pela câmera e processadas, por um computador para ajuste dos parâmetros de inspeção, uma estrutura que dá suporte a plataforma de desenvolvimento e uma mesa de inspeção na qual componentes podem ser posicionados para a inspeção.

Após a montagem, submeteu-se o sistema a testes práticos com diversas imagens com diferentes formatos e quantidade de objetos. Nos testes foram observados os resultados de cada um dos processamentos na imagem no monitor VGA e esses resultados foram comparados aos obtidos em um computador, com o uso da biblioteca de processamento de imagens *OpenCV*. Uma das cenas utilizadas nos testes e os resultados de cada uma das etapas de seu processamento são mostrados na Tabela 4, na qual também são apresentados os resultados obtidos pelo computador.

Tabela 4 - Imagens de teste e resultados de processamentos

Processos	Imagem processada no FPGA	Imagem processada pela biblioteca *OpenCV*
Imagens Originais		
Imagem adquirida pela câmera		
Conversão RGB para tons de cinza		
Limiarização		
Erosão		
Dilatação		
Rotulação de componentes conexos		
Componentes contados	5	5

Fonte: Próprio autor

Como pode ser observado na Tabela 4, os resultados de cada uma das etapas de processamentos estão de acordo com o esperado e muito semelhantes ao obtido pelo uso de uma ferramenta já consolidada. As diferenças que podem ser observadas são em grande parte em função da aquisição ter sido realizada com câmeras diferentes. Isto pode ser notado na comparação das imagens adquiridas, onde na imagem adquirida por um celular para o processamento com *OpenCV* é possível observar uma maior reflexão de luz tanto nos objetos como no fundo da cena.

Após a constatação do funcionamento do sistema, foram realizadas medições de tempo gasto em cada uma das etapas do processamento, de forma que pudessem ser comparadas aos tempos de outras plataformas. Para a comparação, realizou-se os mesmos processamentos em um computador, utilizando a biblioteca *OpenCV* e programação em C com compilador GCC 4.9.2.

O computador em que foram realizados os testes conta com um processador Intel *Core* i7-4500U de 1.8 GHz, 8 GB de memória RAM e placa de vídeo GeForce GT 740M. Os tempos obtidos nos processamentos com o FPGA e no computador podem ser observados na Tabela 5.

Tabela 5 - Tempos de cada uma das etapas de processamento

Processo	Tempo FPGA (ms)	Tempo *OpenCV* (ms)
Aquisição	16,7	32,10
Conversão RAW para RGB	15,4	-
Conversão RGB para tons de cinza	15,4	0,96
Limiarização	15,4	0,98
Erosão	15,4	0,98
Dilatação	15,4	0,98
Rotulação de componentes conexos	32,2	1,66
Extração de descritores	15,4	0,02

Fonte: Próprio autor

Como pode ser visto, os tempos de quase todos os processos (no FPGA) foram aproximadamente os mesmos (15,4 ms). Isto já era esperado visto que todos necessitam de somente uma passada sobre a imagem para gerarem os resultados. No caso das conversões RAW para RGB e RGB para tons de cinza, o tempo de

processamento é somente referente a taxa de aquisição de valores de pixels. Nas operações morfológicas de erosão e dilatação, o tempo total é formado pelo tempo de aquisição da imagem mais o tempo necessário para o preenchimento do *buffer* com os vizinhos do pixel que será analisado.

O único tempo discrepante dos demais (32,2 ms) é o necessário para a etapa de rotulação, que se deve a necessidade de varrer a imagem duas vezes como já explicado na seção 3.1.6.

Todos os tempos medidos tem como limitação a taxa de captura imposta pela câmera (conversão RAW para RGB, RGB para tons de cinza e limiarização) e a taxa de exibição da interface VGA (erosão, dilatação, rotulação e extração de descritores) pelo qual os resultados de cada etapa do processamento são exibidos. Os principais fatores que afetam o tempo e são limitantes no processamento, sincronizado com a interface VGA, é a necessidade da utilização de um *clock* de 25 MHz (para formato 640x480 a 60 FPS) e o fator que para a exibição de uma imagem com resolução de 640X480 pixels o quadro total de vídeo formado pelos pixels ativos e sinais de controle serem equivalentes a uma imagem com resolução de 800X525 pixels (SECONS LTD., 2008).

Quando comparado os tempos obtidos no FPGA aos obtidos no computador, pode-se notar que o tempo de aquisição foi quase o dobro. Esta diferença pode ser explicada pelas diferentes câmeras utilizadas: no FPGA, utilizou-se uma câmera com taxa de aquisição de 60 FPS e no computador utilizou-se uma *webcam* com somente 30 FPS. Com a utilização de uma câmera com 60 FPS nos testes com o computador provavelmente seriam alcançados tempos bem próximos aos observados no FPGA. Porém no momento dos testes não foi possível reproduzir tal situação.

Nos demais processos é observado que os tempos obtidos no computador são bem inferiores aos obtidos no FPGA (da ordem de 15 vezes para a maioria). As velocidades de processamento no FPGA poderiam ser ainda maiores caso fossem realizadas de forma dessincronizada com a exibição VGA, como já foi descrito. Porém, mesmo com a diferenças de tempos descrita, o FPGA foi proporcionalmente mais rápido e eficiente que o computador, uma vez que a frequência de processamento no FPGA é de 25 MHz e no computador é de 1,8 GHz (70 vezes maior).

A fim de medir o quão rapidamente uma imagem é processada após uma solicitação por *trigger*, mediram-se também os tempos totais de processamento de uma imagem. Os tempos alcançados são exibidos na Tabela 6, onde também é exibido o tempo dos mesmos processos realizados em um computador. Os tempos medidos no FPGA começam a ser contados após o sinal de *trigger* e finalizados ao ser sinalizado o resultado da inspeção.

Tabela 6 - Tempos de processamento total de uma imagem

Tempo mínimo FPGA (ms)	Tempo máximo FPGA (ms)	Tempo médio FPGA (ms)	Tempo *OpenCV* (ms)
19	43	30,5	35,89

Fonte: Próprio autor

A existência de diferentes valores de tempo se deve ao modo de implementação do *trigger* de processamento. Como a aquisição de imagens é feita de modo contínuo pela câmera, o *trigger* pode ser acionado no início ou fim da aquisição de um quadro. Resultando, desta forma, em diferentes tempos para diferentes momentos em que o *trigger* é acionado durante a aquisição.

O tempo total de processamento (no FPGA) também é influenciado pelo sincronismo entre aquisição e controle de saída VGA com o qual o processamento está sincronizado. A falta de sincronismo pode interferir de modo que a exibição pode estar até um frame atrasado em relação a aquisição, atraso esse refletido no tempo total de processamento.

Outro fato interessante a ser observado entre a Tabela 5 e a Tabela 6 é que o tempo total de processamento no FPGA, diferente do que acontece no computador, não é o resultado da soma dos tempos de cada processo. Isto se deve a forma paralela em que cada um dos processos acontece, de forma que não é necessário que todo um frame seja adquirido para que inicie o seu processamento.

O processamento paralelo pode ser observado comparando-se o tempo total de processamento ao tempo da realização da rotulação por componentes conexos. Como ao fim da aquisição de uma imagem (na qual foi solicitado a inspeção via *trigger*), a primeira varredura da rotulação já está quase completa o tempo total de processamento é inferior ao tempo da rotulação de forma isolada.

O fato dos processamentos ocorrerem em paralelo no sistema implementado, somado a utilização de câmeras com diferentes frequências de aquisição nas duas plataformas comparadas, produziram como resultados tempos médios de processamentos mais rápidos para a implementação do que a comparada em computador.

5.2 Testes com Aplicações Industriais

O sistema também foi submetido a quatro testes semelhantes a aplicações reais em indústrias. Nesses testes, buscou-se submeter o sistema, após seu ajuste, a inspeção tanto de peças consideradas boas, como ruins de forma a se poder avaliar a geração de resultados satisfatórios, de falsos positivos e de falsos negativos.

5.2.1 Aplicação de Medição

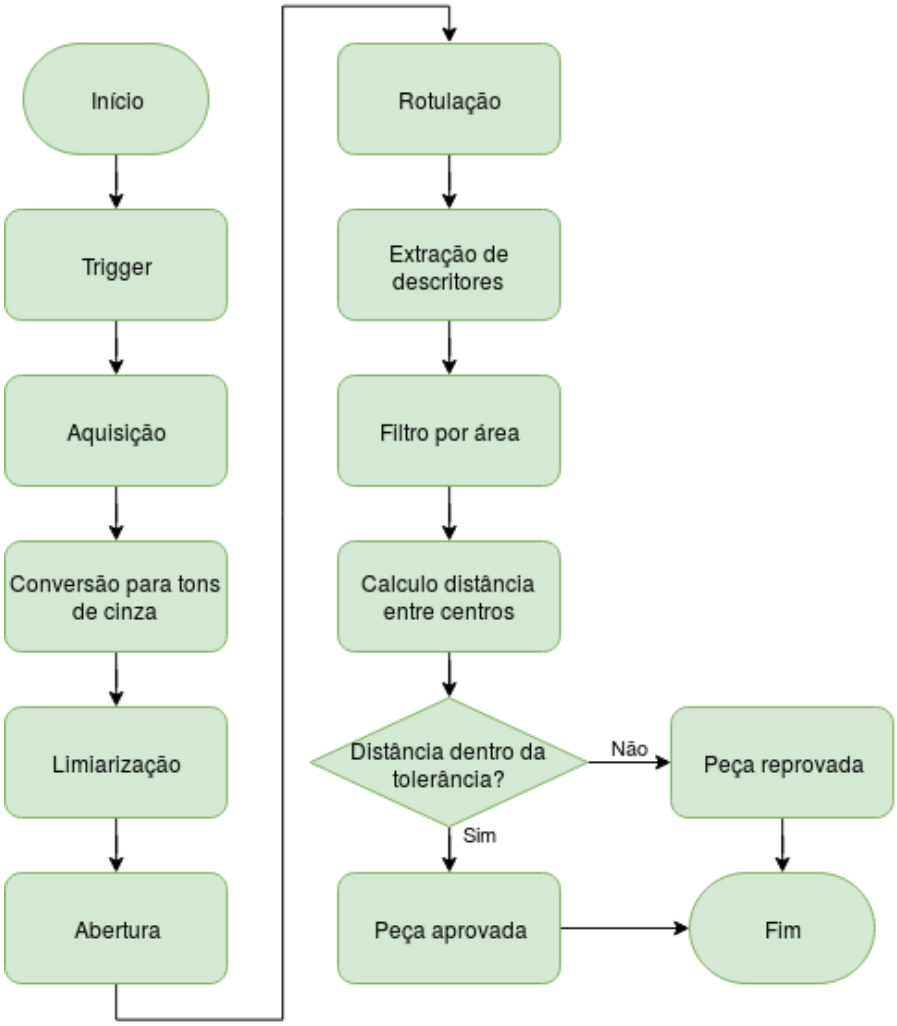
O primeiro caso analisado foi o de juntas de união de componentes que precisam vedação ao serem montados. A junta analisada tem formato elíptico, que idealmente tem três furos em seu corpo e distância entre furos dentro de certa tolerância. Desta forma alguns possíveis problemas são a quantidade de furos errada (para mais ou para menos) e a distância entre furos fora da tolerância. Na Figura 40 pode ser observado a junta analisada.

Figura 40 - Junta inspecionada

Fonte: Próprio autor

Para o teste, da junta o sistema foi configurado de forma a inspecionar a distância entre os centros dos dois menores furos da junta e o fluxograma da inspeção pode ser observado na Figura 41.

Figura 41 - Fluxograma inspeção junta

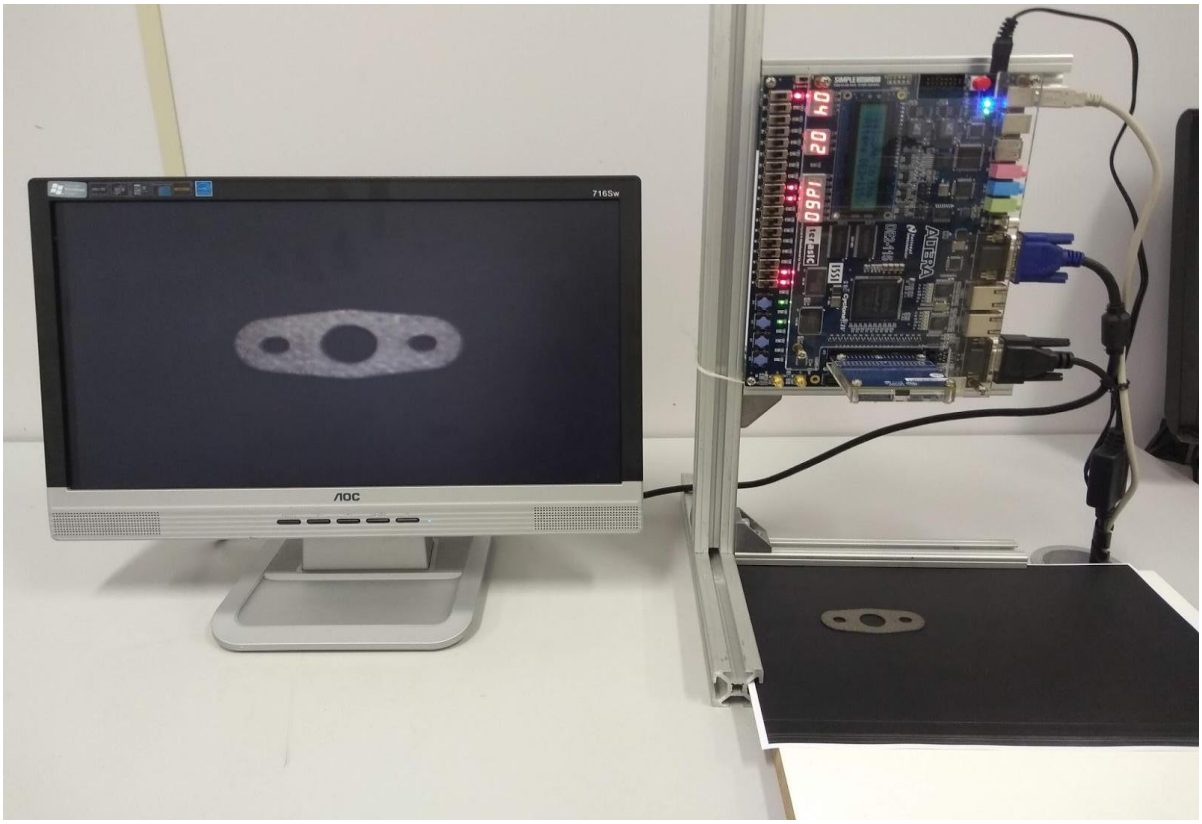


Fonte: Próprio autor

Observando o fluxograma, pode-se notar que quando o gatilho é acionado, uma imagem é adquirida e processada na sequência. Os descritores são filtrados por tamanho de área de forma a serem considerados somente os dois furos menores. Na sequência, a distância entre furos é calculada e comparada a distância padrão. Caso a distância esteja dentro da tolerância aceitável, a peça é considerada aprovada e caso contrário rejeitada.

Para a inspeção da junta, utilizou-se um fundo preto de forma a acentuar o contraste entre fundo e peça. A peça colocada no aparato de inspeção pode ser vista na Figura 42.

Figura 42 - Junta posicionada no sistema de inspeção



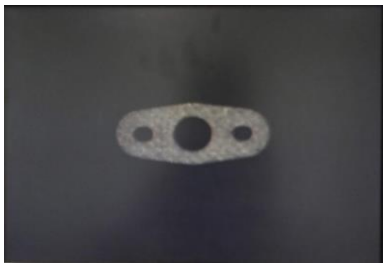
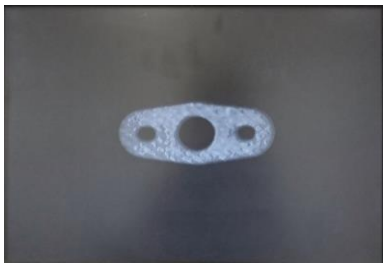
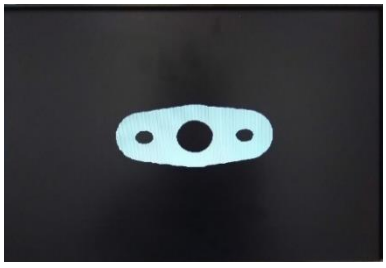
Fonte: Próprio autor

Após o ajuste do sistema de acordo com o fluxograma apresentado na Figura 41, foi verificado o funcionamento de cada uma das etapas do processamento por visualizar os resultados no monitor de vídeo. Cada uma destas etapas pode ser observada na Tabela 7, a partir de fotos do monitor exibindo cada etapa.

Como pode ser visto nas imagens da tabela, o resultado de cada etapa do processamento foi adequado. Na imagem de rotulação também é possível notar que os 3 furos foram corretamente identificados e rotulados. Rotulação esta que pode se visualizada através dos diferentes tons de cinza presentes na imagem.

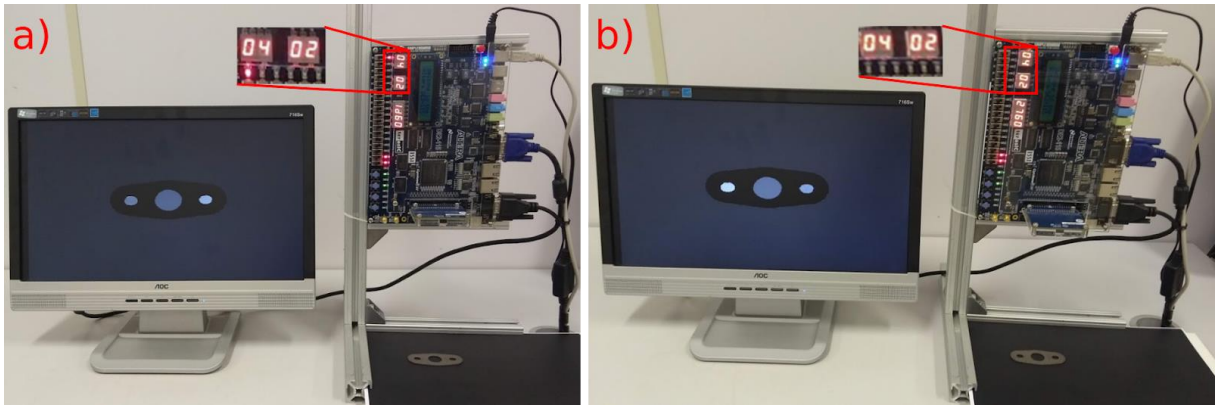
Como só estava disponível uma junta considerada correta, para poder fazer a análise da distância na primeira etapa do teste foi considerada uma dimensão que produziria a aprovação da peça e na segunda etapa uma dimensão diferente da real de modo a produzir um resultado de reprovação. Os resultados de ambas as configurações são mostrados na Figura 43.

Tabela 7 - Imagens da inspeção junta

Etapa	Resultado
Imagem adquirida	
Imagem em tons de cinza	
Imagem binarizada	
Imagem após erosão	
Imagem após dilatação	
Imagem rotulada	

Fonte: Próprio autor

Figura 43 - Resultados da inspeção da junta; a) Junta aprovada; b) Junta reprovada

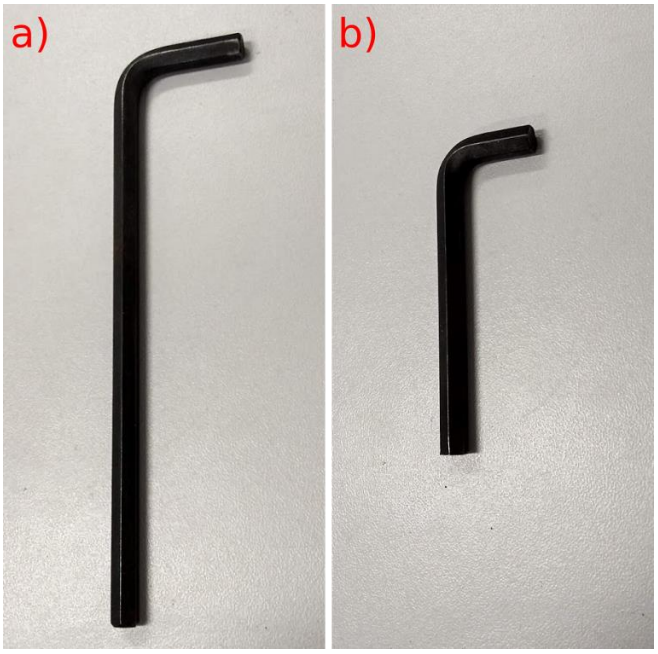


Fonte: Próprio autor

Como pode ser visto, em ambas as configurações foram, contados quatro componentes (fundo mais três furos) e após a aplicação do filtro de área restaram somente dois componentes (dois furos menores). Também pode ser visto pela sinalização do LED que na primeira inspeção a junta foi considerada aprovada (LED acesso) e na segunda reprovada.

O segundo item inspecionado foram chaves Allen, onde o defeito que devia ser detectado pela inspeção era o tamanho incorreto da chave após o processo de fabricação. Na Figura 44 pode ser vista em Figura 44.a uma chave do tamanho esperado e em Figura 44.b uma chave mais curta do que o esperado.

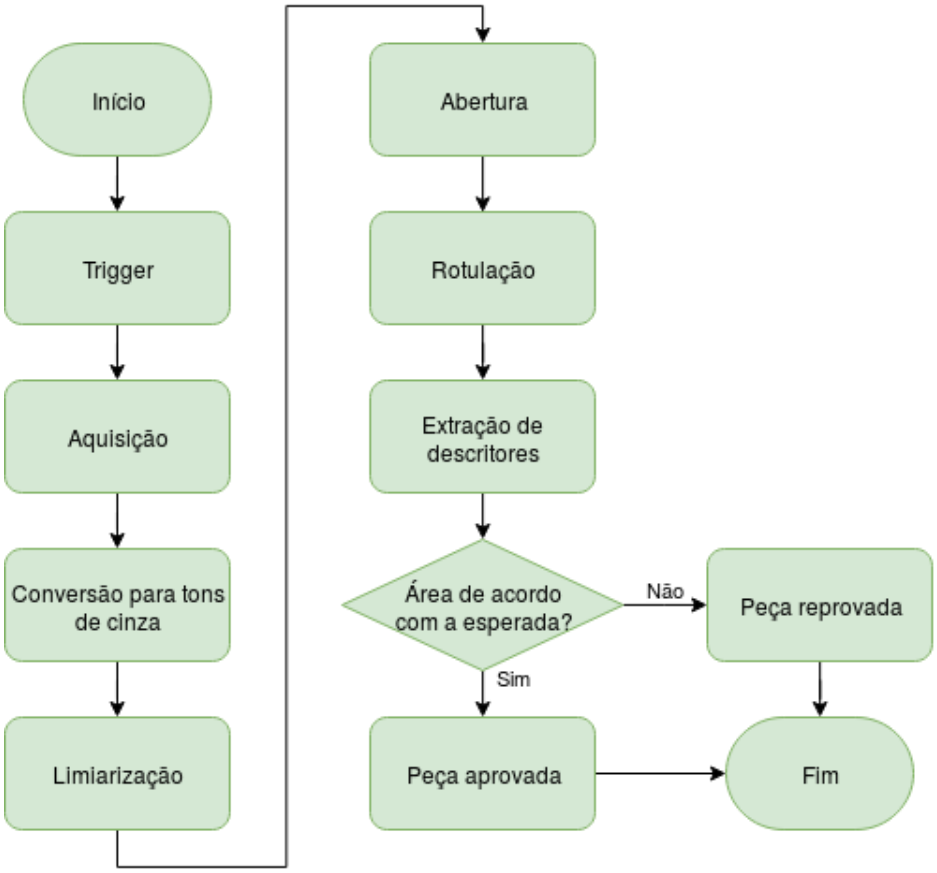
Figura 44 - Chaves inspecionadas a) Chave do tamanho correto; b) Chave com defeito no comprimento



Fonte: Próprio autor

Para a detecção do defeito na chave a inspeção seguiu o fluxograma exposto na Figura 45.

Figura 45 - Fluxograma inspeção chave Allen

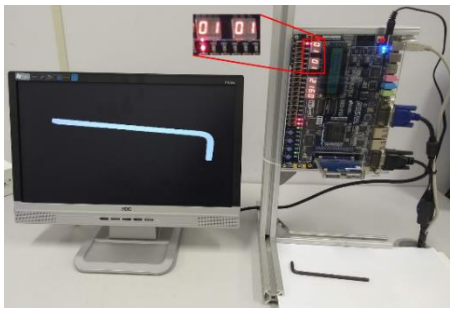
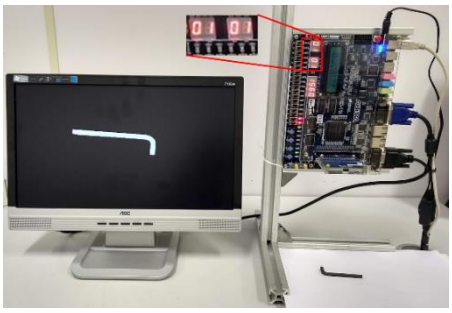


Fonte: Próprio autor

O fluxo de processamento é muito semelhante ao utilizado para a inspeção da junta, mudando somente após a extração dos atributos. Na inspeção realizada, o comprimento da chave será medido de forma indireta através da comparação da área ocupada na imagem pela chave com a área de uma chave considerada adequada.

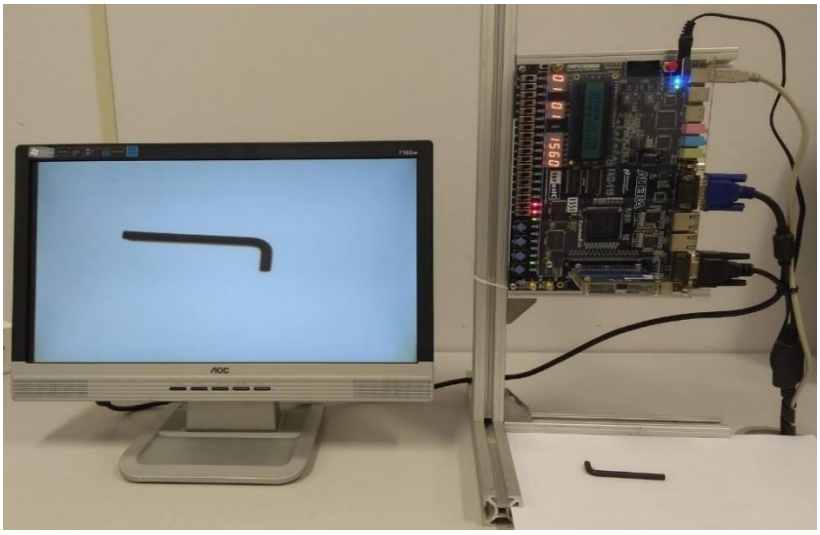
Para a análise, da chave usou-se um fundo branco para que a chave preta ficasse destacada, facilitando a sua segmentação. Este detalhe que pode ser observado na Figura 46, que mostra a chave além posicionada no sistema para inspeção. Após o ajuste do sistema, foram obtidos os resultados apresentados na Tabela 8.

Tabela 8 - Imagens inspeção chave Allen

Etapa	Chave boa	Chave ruim
Imagem adquirida		
Imagem em tons de cinza		
Imagem binarizada		
Imagem após erosão		
Imagem após dilatação		
Imagem rotulada		
Resultado		

Fonte: Próprio autor

Figura 46 - Chave Allen posicionada no sistema de inspeção



Fonte: Próprio autor

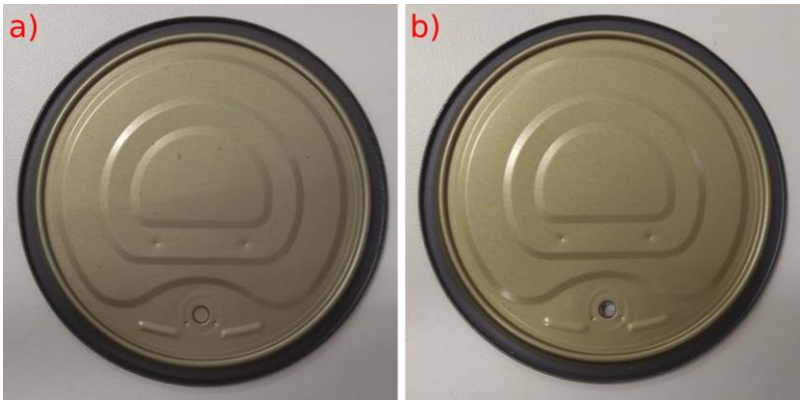
Na Tabela 8 fica evidente o funcionamento de cada etapa do processamento e a correta avaliação do tamanho das chaves baseado na área que as mesmas ocupam na imagem.

5.2.2 Aplicação de Detecção

O terceiro caso de aplicação no qual o sistema foi testado, foi o de tampas de enlatados. A tampa analisada tem formato circular e para que seja possível abrir a lata após o fechamento é inserido uma alça com o uso de um rebite. Durante a fabricação da tampa pode acontecer que o rebite não seja inserido, este defeito deve ser detectado antes do fechamento da lata de forma a minimizar os prejuízos gerados.

Na Figura 47 são mostrados exemplos das tampas inspecionadas, em Figura 47.a é possível observar uma tampa com o rebite corretamente inserido e em Figura 47.b uma tampa em que o rebite não foi inserido.

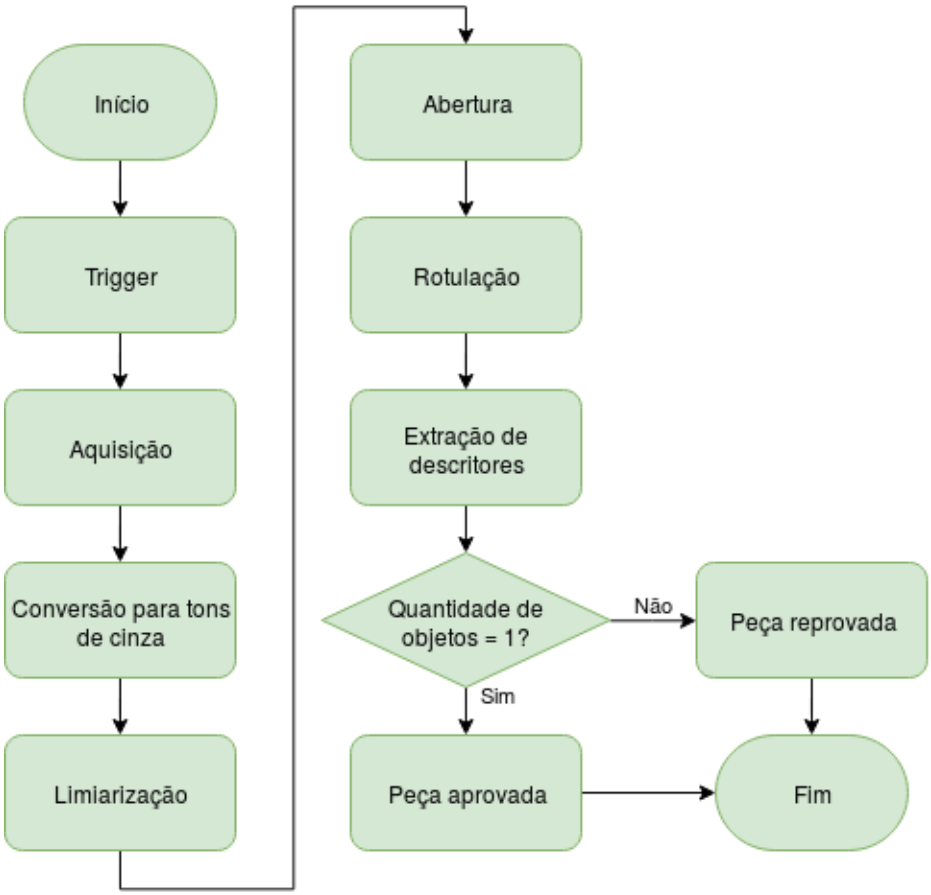
Figura 47 - Tampas inspecionadas: a) Tampa com rebite; b) Tampa sem rebite



Fonte: Próprio autor

Após a definição do defeito a ser detectado elaborou-se o fluxograma de operações apresentado na Figura 48.

Figura 48 - Fluxograma inspeção tampas

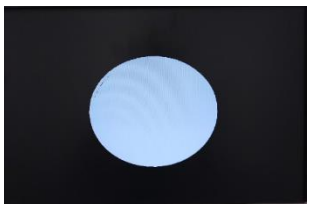
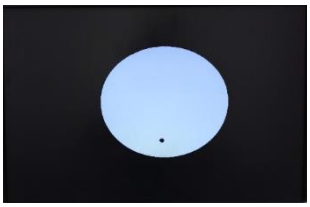
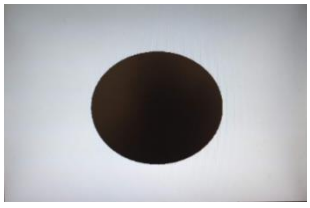
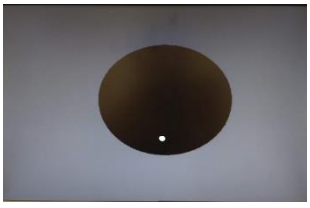
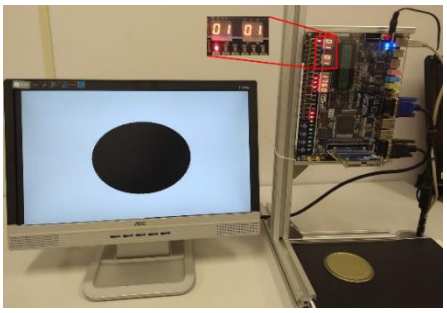
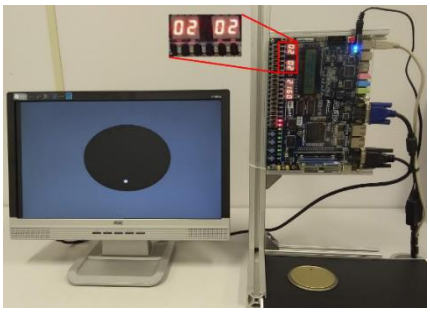


Fonte: Próprio autor

Dos diversos modos possíveis de detectar a presença do furo, optou-se por realizar a contagem dos componentes presentes na cena, considerando o fundo como um objeto. Desta forma caso o rebite não esteja presente serão contados dois componentes conexos: o fundo ao redor da tampa e o fundo visível pelo furo onde o rebite deveria estar presente. A Figura 49 representa uma tampa sendo inspecionada pelo sistema.

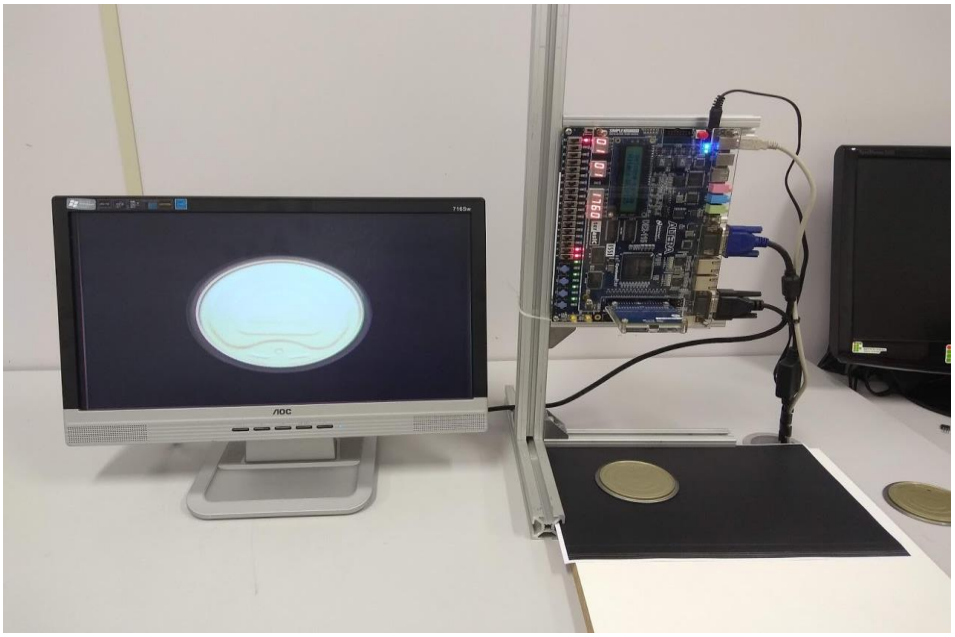
Os resultados da inspeção da tampa e as etapas de processamento podem ser observados na Tabela 9. Nesta tabela também pode ser observado que a tampa onde o rebite está presente teve a contagem de somente um componente e foi considerada aprovada e que na tampa com o furo, foram detectados dois objetos e a tampa foi reprovada.

Tabela 9 - Imagens inspeção tampa de enlatado

Etapa	Tampa boa	Tampa com defeito
Imagem adquirida		
Imagem em tons de cinza		
Imagem binarizada		
Imagem após erosão		
Imagem após dilatação		
Imagem rotulada		
Resultado		

Fonte: Próprio autor

Figura 49 - Tampa de enlatado posicionada no sistema de inspeção

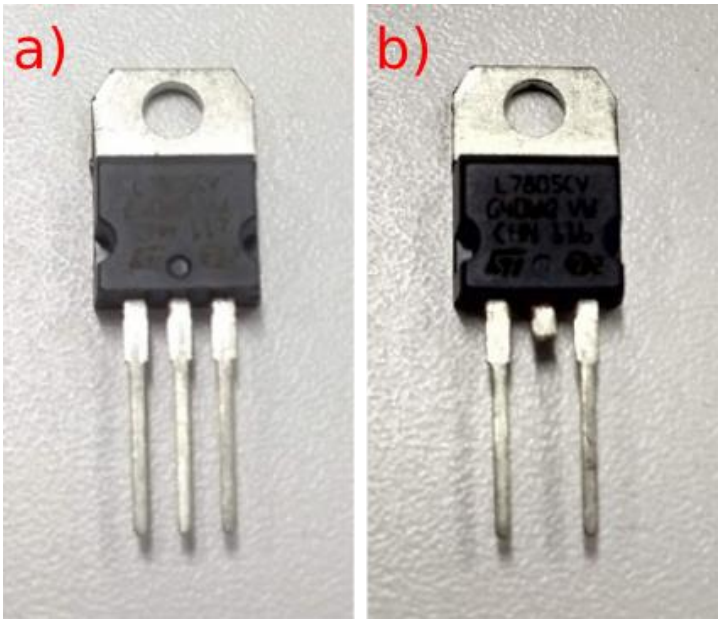


Fonte: Próprio autor

5.2.3 Aplicação de Contagem

O quarto caso de inspeção ao qual o sistema foi submetido foi o de análise de um transistor, no qual o objetivo era a detecção da presença de todos os seus três terminais. Na Figura 50.a pode ser observado um transistor em perfeitas condições e na Figura 50.b um transistor com um terminal defeituoso.

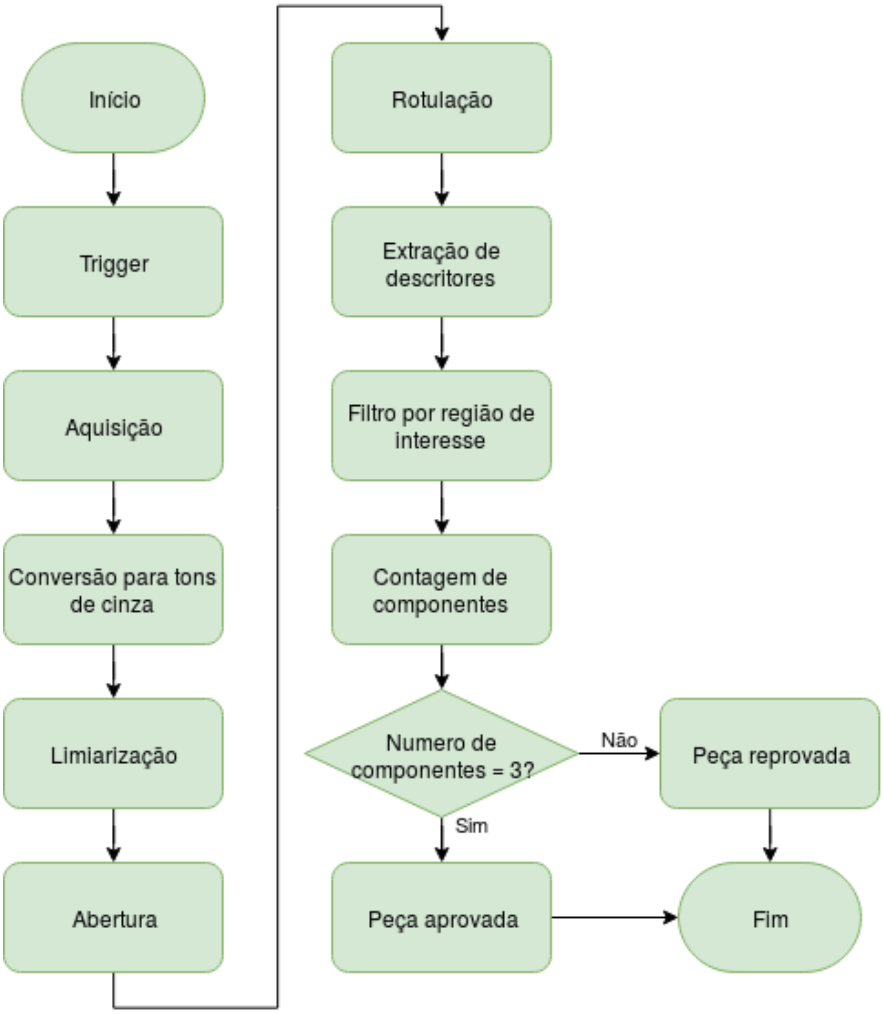
Figura 50 - Transistores inspecionadas a) Transistor bom: b) Transistor com um terminal cortado



Fonte: Próprio autor

O fluxograma de inspeção de um transistor é apresentado na Figura 51.

Figura 51 - Fluxograma inspeção transistor

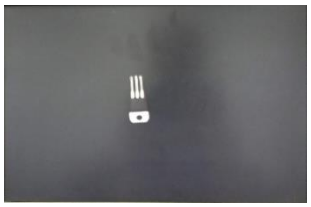
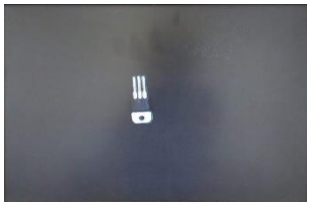
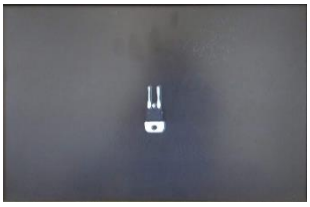
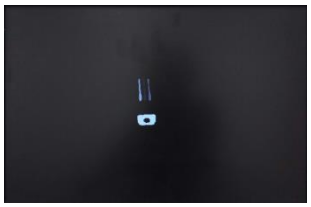
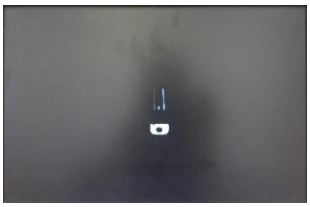
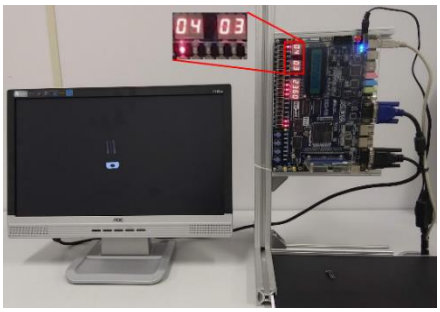
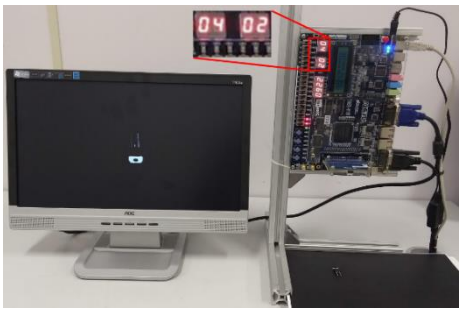


Fonte: Próprio autor

Pode ser observado que o fluxograma de inspeção é igual ao das inspeções de outras peças realizadas até a etapa de extração de descritores, porém na sequência é aplicado um filtro nos componentes encontrados de forma a serem considerados somente os componentes presentes em uma região de interesse da imagem. Este filtro é útil para se contar somente as pernas do transistor sem considerar o restante da imagem. Na Figura 52 é exibido um transistor posicionado para ser inspecionado pelo sistema, na figura também aparece destacada a região de interesse considerada para a inspeção.

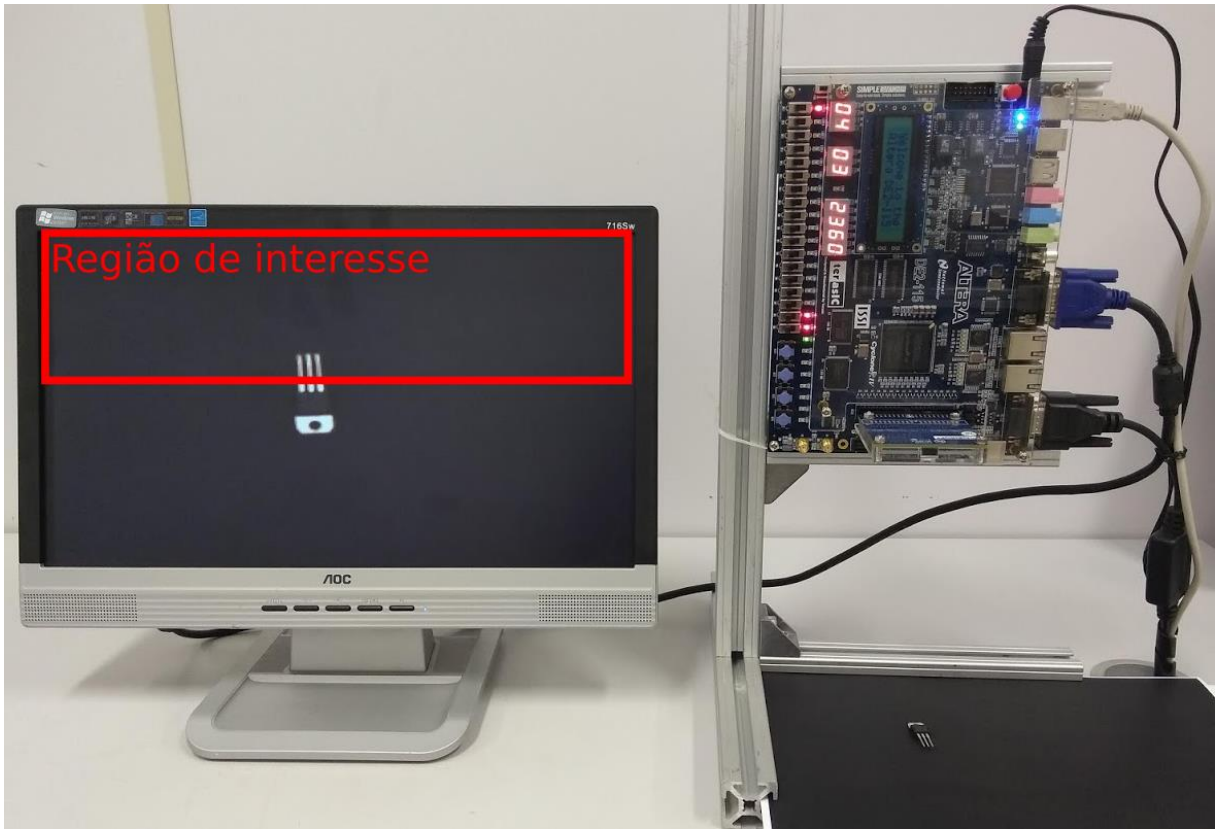
Como pode ser observado, a região de interesse definida é a superior da imagem onde estão presentes somente as pernas do transistor. Uma vez que a região de interesse e demais parâmetros de inspeção foram ajustados, puderam ser colhidas as imagens apresentadas na Tabela 10.

Tabela 10 - Imagens inspeção transistor

Etapa	Tampa boa	Tampa com defeito
Imagem adquirida		
Imagem em tons de cinza		
Imagem binarizada		
Imagem após erosão		
Imagem após dilatação		
Imagem rotulada		
Resultado		

Fonte: Próprio autor

Figura 52 - Transistor posicionada no sistema de inspeção



Fonte: Próprio autor

Pelas imagens apresentadas na Tabela 10 pode ser observado que o sistema contou quatro componentes (três terminais mais a parte metálica para fixação de dissipador) no caso do transistor bom e no ruim, devido a perna estar quebrada, mas ainda presente. Após a aplicação da região de interesse, foram contados três e dois componentes no transistor bom e ruim respectivamente. Pode ser notado também que o transistor em bom estado é sinalizado pelo LED acesso.

5.3 Discussão Geral dos Resultados

Embora o sistema desenvolvido no presente trabalho consiga realizar a análise em diversos casos, como os descritos nesta etapa de resultados, existem algumas outras aplicações que não foram cobertas, como por exemplo o caso de decodificação de códigos de barra.

Desta forma, é importante que a implementação não aloque todos os recursos disponíveis no FPGA utilizado, possibilitando implementações futuras de mais funções. A Tabela 11 mostra a quantidade de recursos necessários na implementação do sistema.

Tabela 11 - Recursos utilizados na implementação

Recurso	Utilizado	Total	Percentual utilizado
Elementos Lógicos	32.196	114.480	28%
Bits de Memória	1.369.600	3.981.312	34%

Fonte: Próprio autor

Observando a Tabela 11, é possível notar que ainda podem ser implementadas mais funções, pois o FPGA presente no *kit* DE2-115 permaneceu com grande parte de seus recursos sem utilização.

Comparando-se o sistema desenvolvido aos disponíveis na literatura, observa-se que foram obtidas velocidades de processamento inferiores aos dos trabalhos de Mcbarder (2003) e de Gupta (2014), isto em parte pelos processamentos estarem sincronizados com a aquisição e exibição. Também se nota que o sistema desenvolvido é capaz de processar imagens com resolução maior (640x480) que a maioria dos trabalhos correlatos encontrados (MCBADER; LEE, 2003; MERTES, 2012; RIBEIRO, 2007). Sendo este outro fator importante para a velocidade em que os processamentos são realizados.

As características do presente trabalho são semelhantes as desenvolvidas no trabalho de Walczyk (2010) no quesitos de resolução de imagem processada e algoritmos utilizados. Entretanto, é importante ressaltar que o trabalho aqui descrito é mais abrangente pois possibilita a extração e comparação de um maior número de descritores.

No sistema implementado também é possível a configuração de parâmetros pelo usuário através de um computador como descrito no trabalho de Ribeiro (2007). Porém, não sendo possível o envio de uma imagem para processamento partir do computador.

Desta forma, o sistema desenvolvido se mostra mais adequado para aplicações reais em inspeções do que os demais trabalhos, pois apresenta qualidade de imagem suficiente para a maioria das aplicações, permite a análise de diferentes descritores de objetos, pode ser facilmente reconfigurado pelo usuário e por apresentar recursos úteis como: região de interesse, filtro de componentes por tamanhos de área e possibilidade de integração com linhas de produção por meio da utilização de *trigger* e sinalização de rejeição.

6 CONCLUSÕES

No presente trabalho é descrito o desenvolvimento de um sistema de inspeção automatizada por visão computacional empregando FPGA. Como demonstrado, foram implementados algoritmos de processamento de imagem importantes e os mesmos são facilmente parametrizável. Desta forma, pode ser utilizado para inspeções existentes em diferentes cenários da indústria.

Com base nos casos reais ao qual o sistema foi testado é possível afirmar que foram cumpridos os requisitos de verificar a presença/ausência, realizar contagem e estimar posições, dimensões e distância entre componentes presentes na cena.

Embora a implementação do sistema tenha se mostrado menos produtiva, quando comparada a realizada em um computador convencional, devido a inexistência de bibliotecas e ferramentas já implementadas, esta se mostrou viável. Além disso, a utilização do FPGA permite o desenvolvimento de sistemas mais adequados ao ambiente industrial, tendo em vista seu tamanho reduzido e capacidade de processamento.

Uma vez que o sistema se mostrou capaz de realizar as inspeções em produtos manufaturados, o mesmo poderia ser utilizado de forma integrada a uma linha de produção, onde para cada produto produzido seria solicitado a sua inspeção, via *trigger*, e um sinal de descarte gerado caso a peça fosse julgada não conforme.

Entre as melhorias possíveis para o sistema, estão o processamento fora de sincronia com a exibição, que resultaria em taxas de processamento maiores. Ainda, implementações que possibilitem uma maior flexibilidade nas comparações com resultados esperados e desenvolvimento de um software supervisório, onde seja possível ajustes de parâmetros de forma gráfica e também a exportação de estatísticas importantes sobre as peças inspecionadas.

REFERÊNCIAS

ALTERA. Cyclone IV FPGA Device Family Overview. [s. l.], v. 1, n. March, 2016. Disponível em: <https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-iv/cyiv-51001.pdf>

BOB ZEIDMAN. **All about FPGAs**. 2006. Disponível em: <https://www.eetimes.com/document.asp?doc_id=1274496&page_number=2>. Acesso em: 3 out. 2018.

DALGLEISH, Tim et al. **DE2-115 User ManualTerasic Inc.**, 2017.

DAVIES, E. R. **Computer and Machine Vision, Fourth Edition: Theory, Algorithms, Practicalities**. 4th. ed. Orlando, FL, USA: Academic Press, Inc., 2012.

ESQUEF, Israel Andrade; ALBUQUERQUE, Márcio Portes De; ALBUQUERQUE, Marcelo Portes De. Processamento Digital de Imagens. **Centro Brasileiro de Pesquisas Fisicas**, [s. l.], 2003.

FILHO, Ogê Marques; NETO, Hugo Vieira. **Processamento Digital de Imagens**. Rio de Janeiro, Brasil: Brasport, 1999.

GONZALEZ, Rafael C.; WOODS, Richard E. **Processamento Digital de Imagens**. 3ª ed. Sao Paulo, SP, Brasil: Pearson Education do Brasil Ltda., 2008.

GUPTA, Siddharth et al. A new parallel algorithm for two-pass connected component labeling. **Proceedings of the International Parallel and Distributed Processing Symposium, IPDPS**, [s. l.], p. 1355–1362, 2014.

JUNIOR, Marcos Antonio da Cunha Oliveira. **Redes Neurais de Hopfield para Roteamento de Redes de Comunicação em FPGA**. 2016. ESCOLA POLITÉCNICA DE PERNAMBUCO UNIVERSIDADE DE PERNAMBUCO, [s. l.], 2016.

MCBADER, Stephanie; LEE, Peter. An FPGA implementation of a flexible, parallel image processing architecture suitable for embedded vision systems. **Proceedings - International Parallel and Distributed Processing Symposium, IPDPS 2003**, [s. l.], v. 00, n. C, 2003.

MERTES, Jacqueline Gomes. **Implementação em FPGA de um sistema para processamento de imagens digitais para aplicações diversificadas**. 2012. UNESP, [s. l.], 2012.

NOGUEIRA, Ricardo César de Almeida. **Análise de Conversão de Imagem Colorida para Tons de Cinza Via Contraste Percebido**. 2016. UFPE, [s. l.], 2016.

OMRON MICROSCAN SYSTEMS, Inc. **Machine Vision Systems**. 2018. Disponível em: <https://www.microscan.com/en-us/products/machine-vision-systems>. Acesso em: 24 set. 2018.

PAUL DILLIEN. **And the Winner of Best FPGA of 2016 is...** 2017. Disponível em: <https://www.eetimes.com/author.asp?doc_id=1331443>. Acesso em: 4 out. 2018.

RIBEIRO, Rodrigo Marcel. **Hardware Para Processamento Digital De Imagens**. 2007. UNICENP, [s. l.], 2007.

ROELANDTS WIM. Cover Story. **Xcell**, San Jose, CA, n. 32, p. 4–5, 1999. Disponível em: <https://www.xilinx.com/publications/archives/xcell/Xcell32.pdf>

SANTIAGO, Diêgo João Costa. **Otimização e eficiência de algoritmos de rotulação de componentes conexos em imagens binárias**. 2009. UFPE, [s. l.], 2009.

SAVVAS, Tsengelidis. **Algebraic Modeling of Transformations from Bayer to RGB Images**. 2006. Technical University of Crete, [s. l.], 2006.

SECONS LTD. **VGA Signal 640 x 480 @ 60 Hz Industry standard timing**. 2008. Disponível em: <http://tinyvga.com/vga-timing/640x480@60Hz>. Acesso em: 16 nov. 2018.

STEGER, C.; ULRICH, M.; WIEDEMANN, C. Machine Vision Algorithms and Applications. **Wiley-Vch**, [s. l.], 2008.

STIVANELLO, M. E. et al. Visage - Sistema De Inspecao De Produtos Por Visao Computacional Baseado Em Diagrama De Blocos. In: 12TH IEEE/IAS INTERNATIONAL CONFERENCE ON INDUSTRY APPLICATIONS 2016, Curitiba, PR, Brasil. **Anais**... Curitiba, PR, Brasil

STIVANELLO, M. E.; GOMES, Paulo Cesar Rodacki. Inspecao Visual Industrial Automatizada Por Analise De Forma Com Descritores De Fourier E Redes Neurais Artificiais. In: SEMINCO - SEMINARIO DE COMPUTACAO 2006, Blumenau. **Anais**... Blumenau

SZELISKI, Richard. **Computer Vision: Algorithms and Applications**. 1st. ed. New York, Ny, Usa: Springer-Verlag, 2010.

TERASIC INC. **Altera DE2-115 Development and Education Board**. 2013. Disponível em: <http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=502>. Acesso em: 8 out. 2018.

TERASIC INC. **8 Mega Pixel Digital Camera Package with GPIO interface**. 2018a. Disponível em: <http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=1011>. Acesso em: 8 out. 2018.

TERASIC INC. **D8M-GPIO User Manual**, 2018. b.

TIM MOYNIHAN. **CMOS Is Winning the Camera Sensor Battle, and Here's Why | TechHive**. 2011. Disponível em: <https://www.techhive.com/article/246931/security-cameras/cmos-is-winning-the-camera-sensor-battle-and-heres-why.html>. Acesso em: 13 set. 2018.

TUTORIALSPOINT. **Grayscale to RGB Conversion**. 2018. Disponível em: <https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm>. Acesso em: 26 set. 2018.

VARGAS, S. et al. Automatic Detection and Classification of Defects in Knitted Fabrics. **Revista Ieee America Latina**, [s. l.], v. 14, p. 3065–3073, 2016.

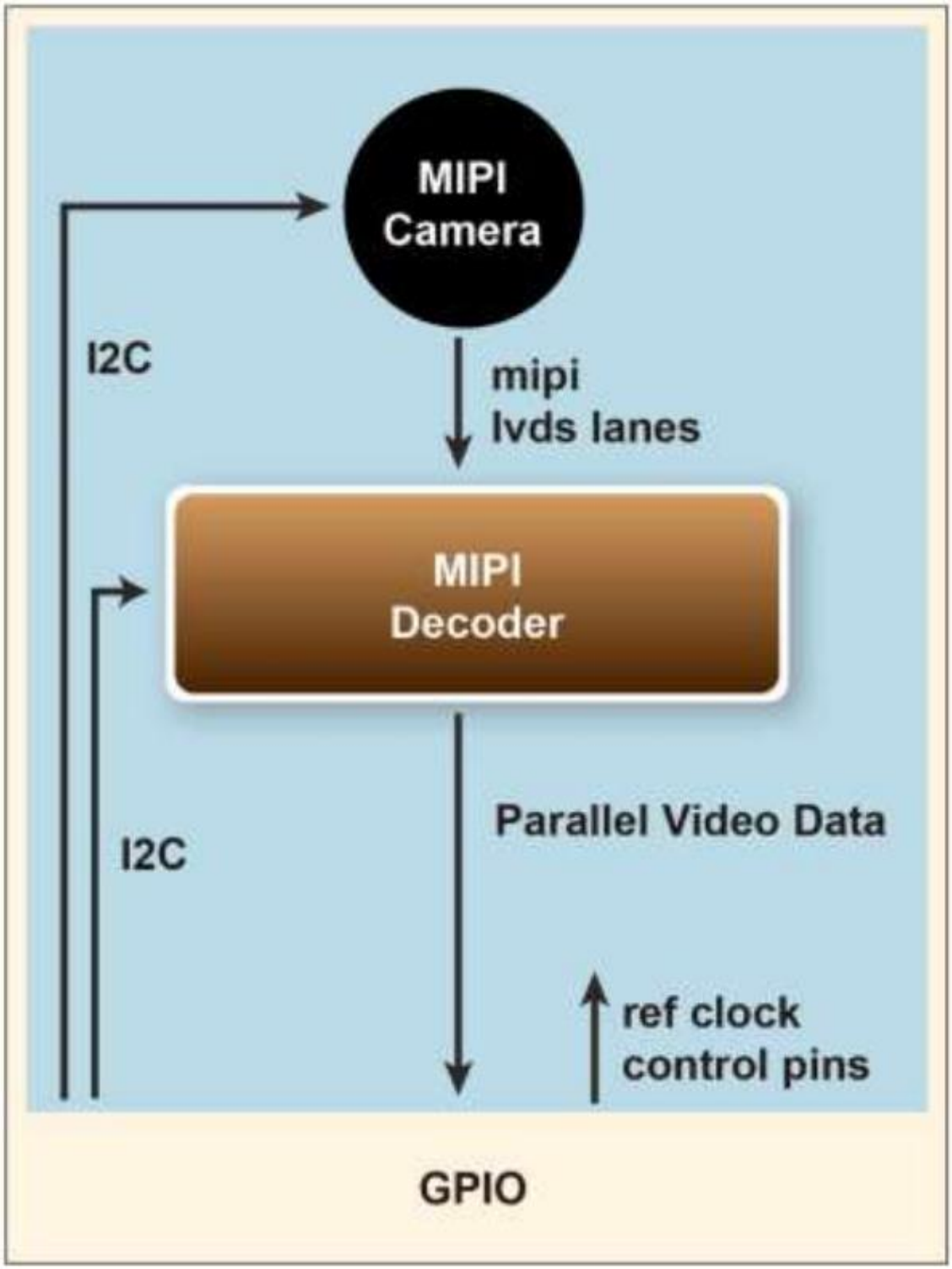
WALCZYK, Robert; ARMITAGE, Alistair; BINNIE, David. Comparative study on connected component labeling algorithms for embedded video processing systems. **IPCV**, [s. l.], 2010.

WIKIPEDIA. **HSL and HSV**. 2018a. Disponível em: <https://en.wikipedia.org/wiki/HSL_and_HSV>. Acesso em: 22 set. 2018.

WIKIPEDIA. **Bayer filter**. 2018b. Disponível em: <https://en.wikipedia.org/wiki/Bayer_filter>. Acesso em: 22 set. 2018.

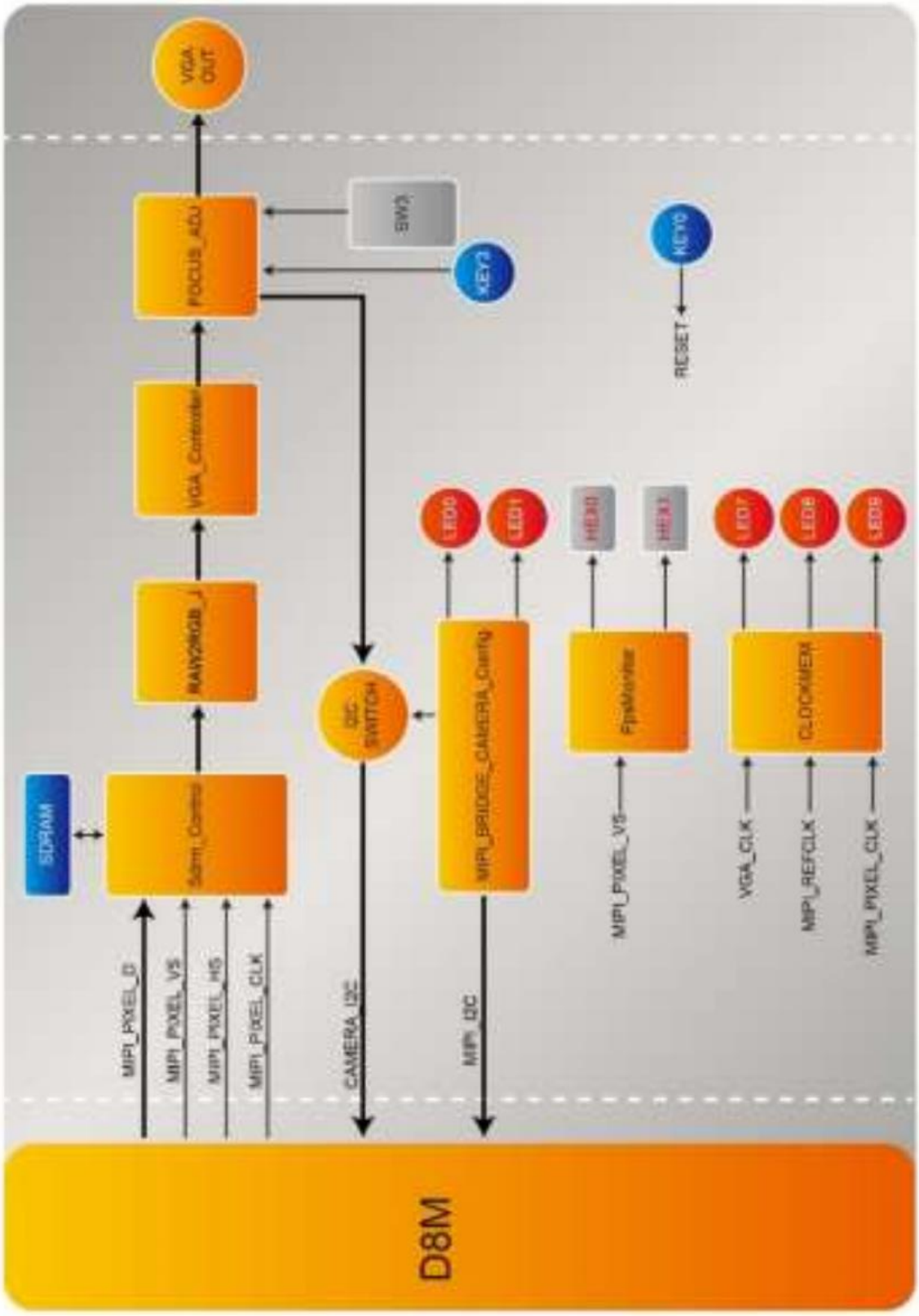
WOODS, Roger. **FPGA-basead Implementation of Signal Processing Systems**. Chichester, West Sussex, United Kingdom: John Wiley & Sons, Ltd Registered, 2008.

ANEXO A - Diagrama de blocos D8M-GPIO



Fonte: Terasic Inc. (2018b)

ANEXO B - Diagrama de blocos do projeto de referência D8M-GPIO



Fonte: Terasic Inc. (2018b)