

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E
INFORMÁTICA INDUSTRIAL – CPGEI**

RICARDO UMBRIA PEDRONI

**SISTEMA AUTÔNOMO EM FPGA PARA CAPTURA E PROCESSAMENTO EM
TEMPO REAL DE IMAGENS DA PUPILA**

DISSERTAÇÃO

CURITIBA

2011

RICARDO UMBRIA PEDRONI

**SISTEMA AUTÔNOMO EM FPGA PARA CAPTURA E PROCESSAMENTO EM
TEMPO REAL DE IMAGENS DA PUPILA**

Dissertação de mestrado apresentado ao Programa de Pós Graduação em Engenharia Elétrica e Informática Industrial da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do grau de “Mestre em Ciências”. Área de Concentração: Informática Industrial.

Orientador: Prof. Dr. Volnei A. Pedroni

CURITIBA

2011

Dados Internacionais de Catalogação na Publicação

P372 Pedroni, Ricardo Umbria
Sistema autônomo em FPGA para captura e processamento em tempo real de
imagens da pupila / Ricardo Umbria Pedroni. — 2011.
84 f. : il. ; 30 cm

Orientador: Volnei A. Pedroni.

Dissertação (Mestrado) – Universidade Tecnológica Federal do Paraná. Programa
de Pós-graduação em Engenharia Elétrica e Informática Industrial. Área de
concentração: Informática industrial, Curitiba, 2011.

Bibliografia: p. 83-84.

1. Biometria. 2. Pupilometria. 3. Hardware. 4. Arranjos de lógica programável em
campo – FPGA. 5. Robôs autônomos. I. Pedroni, Volnei A., orient. II. Universidade
Tecnológica Federal do Paraná. Programa de Pós-graduação em Engenharia Elétrica e
Informática Industrial. III. Título.

CDD (22. ed.) 570.15195

Biblioteca Central da UTFPR, Campus Curitiba

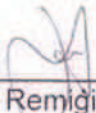
Título da Dissertação Nº 563:

“Sistema Autônomo em FPGA para Captura e Processamento em Tempo Real de Imagens da Pupila”

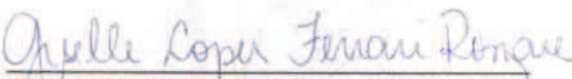
por

Ricardo Umbria Pedroni

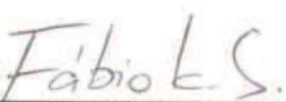
Esta dissertação foi apresentada como requisito parcial à obtenção do grau de MESTRE EM CIÊNCIAS – Área de Concentração: Engenharia de Automação e Sistemas, pelo Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial – CPGEI – da Universidade Tecnológica Federal do Paraná – UTFPR – Campus Curitiba, às 10h30 do dia 28 de junho de 2011. O trabalho foi aprovado pela Banca Examinadora, composta pelos professores:



Prof. Humberto Remigio Gamba, Dr.
(Presidente – UTFPR)

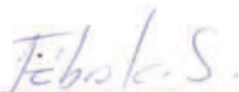


Prof. Giselle Lopes Ferrari Ronque, Dr.
(UFPR)



Prof. Fábio Kurt Schneider, Dr.
(UTFPR)

Visto da coordenação:



Prof. Fábio Kurt Schneider, Dr.
(Coordenador do CPGEI)

Aos dois que, por amor, criaram.

Aos dois que, por amor, se juntaram.

A uma que, por amor, será.

Aos cinco que, por amor e nada mais, serão sempre minha família.

AGRADECIMENTOS

À UTFPR, pela sua estrutura e, especialmente, pelos seus professores, cujos ensinamentos muitas vezes transcenderam as paredes da sala de aula.

À minha família, cujo agradecimento não pode ser limitado apenas a esse período da minha vida, mas sim, a toda ela.

À minha namorada, por simplesmente ser quem ela é.

E ao meu orientador, especialmente, cujas lições acadêmicas só podem ser equiparadas em valor com suas lições de vida. Agradeço a Deus pela honra de poder chamá-lo, além de mestre, especialmente de pai.

RESUMO

PEDRONI, Ricardo U. Sistema autônomo em FPGA para captura e processamento em tempo real de imagens da pupila. 2011. 84 f. Dissertação. Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial. Universidade Tecnológica Federal do Paraná. 2011.

Essa dissertação propõe um algoritmo e um equipamento (*hardware*) para captação de imagens da pupila do olho humano e processamento das mesmas a fim de obter, de forma portátil, autônoma, segura, não invasiva e em tempo real, informações sobre a pupila. Mais especificamente, o objetivo é obter informações que permitam determinar o diâmetro da pupila, tanto de forma estática (pupila com tamanho estável, sem a incidência intencional de luz) quanto dinâmica (pupila variando devido à aplicação de luz com intensidade variável). Tal sistema pode ser utilizado no setor da saúde, por exemplo, para realização da pupilometria, exame feito na área de oftalmologia, ou para medição da velocidade de expansão da pupila, exame auxiliar no diagnóstico de uma série de doenças que afetam o sistema nervoso.

O equipamento desenvolvido é composto por três partes principais: câmera digital controlável, responsável pela obtenção das imagens; display LCD (*Liquid Crystal Display*), usado para visualização das imagens e resultados durante os testes; e FPGA (*Field Programmable Gate Array*), contendo todos os circuitos responsáveis pelo controle da câmera, processamento das imagens e controle do display. A escolha de cada componente do sistema foi feita procurando atender às especificações do projeto com o menor custo possível, fato este que também foi determinante no desenvolvimento do algoritmo, o qual procura utilizar o mínimo de recursos.

É fundamental salientar que o sistema foi desenvolvido para operar de forma completamente autônoma, ou seja, sem necessidade de ligação a um computador ou a qualquer outro equipamento ou *software* externo. Como se desejou que nem mesmo uma memória externa fosse utilizada (nada além dos recursos internos à FPGA), um dos grandes desafios foi a obtenção de um algoritmo final que demandasse uma quantidade de memória (para os *pixels* das imagens, por exemplo, tanto pré quanto pós processamento) tão reduzida quanto possível, a fim de ser possível implementar o sistema completo numa única FPGA e com baixo custo. O equipamento desenvolvido é capaz de processar (isto é, capturar as imagens, analisá-las, detectar a pupila e fornecer os resultados no display) à taxa de 24 quadros por segundo.

Todos os circuitos, incluindo o algoritmo principal proposto e todas as demais partes (controlador da câmera, controlador do display, memória de *pixels*, etc.) necessários ao funcionamento do sistema foram codificados utilizando uma HDL (*Hardware Description Language*). Mais especificamente, VHDL foi empregado na quase totalidade dos blocos, com Verilog utilizado em alguns casos. O uso de uma HDL é de suma importância pois permite obter um projeto moderno e eficiente, independente da tecnologia ou fabricante da FPGA utilizada, facilmente simulável e sintetizável. De particular interesse é que assim os circuitos podem ser facilmente parametrizados, de forma que testes para vários valores de parâmetros tornam-se mais simples, o que é fundamental na fase de estudos. Além disso, o projeto fica melhor documentado, e o código pode ser utilizado diretamente para a fabricação de um ASIC (*Application Specific Integrated Circuit*), isto é, um chip dedicado, se assim for desejado.

O algoritmo desenvolvido procura o centro e o raio da pupila através de estimativas iniciais extraídas do histograma e centróide da imagem. Paralelamente, são obtidas as bordas da imagem capturada, utilizando uma versão em hardware do detector de bordas

de Canny. Uma vez estimado o centro da pupila, obtém-se 24 pontos sobre a borda ao redor do centro estimado, separados por 15 graus um do outro. A obtenção desses pontos se dá através de um método criado nessa dissertação, chamado *explosão radial*. Com esses pontos, estima-se, utilizando o método RANSAC (*Random Sample Consensus*), o centro e o raio verdadeiros da pupila do indivíduo sob teste. Tudo ocorre de forma automática, em tempo real e sem contato físico, bastando o indivíduo posicionar seu olho em frente e próximo à câmera.

Esse método de análise da pupila é o resultado de um grande número de investigações, incluindo simulações e implementações físicas de diversas alternativas, até chegar-se ao resultado desejável com um mínimo de recursos, o qual poderá servir como ponto de partida para um sistema profissional de mensuração da pupila.

Palavras-chave: Biometria, pupilometria, *hardware*, FPGA, sistemas autônomos

ABSTRACT

PEDRONI, Ricardo U. Autonomous system implemented in FPGA for real-time pupil's image capture and processing. 2011. 84 f. Dissertation. Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial. Universidade Tecnológica Federal do Paraná. 2011.

This dissertation proposes an algorithm and a corresponding hardware implementation capable of capturing images from the human eye and processing these images to obtain, in a portable, autonomous, secure, and non-invasive way, in real time, information regarding the pupil. More specifically, the objective is to obtain information that allows the equipment to determine the pupil's diameter, both in static form (i.e., with constant light intensity) and in dynamic form (pupil under varying light intensity). Such a system can be used in the health sector, for example, in exams such as pupillometry, a test done by ophthalmologists, or for measuring the pupil's expansion rate, a test used in the diagnosis of a series of diseases that affect the nervous system.

The developed equipment is composed of three main parts: a controllable digital camera, responsible for obtaining the images; an LCD (Liquid Crystal Display), used to view the images and results during the tests; and an FPGA (Field Programmable Gate Array) device, containing all the circuits responsible for controlling the camera, processing the image, and controlling the digital display. Each component was chosen to fulfill the required specifications with a cost as low as possible. This attempt to reduce the cost also affected the final algorithm, developed with the idea of minimal resources requirement in mind.

The system was developed to operate in a completely autonomous form, without the necessity of a computer or any other equipment or external software. Because the use of even an external memory was undesirable (no other resources besides the internal FPGA's resources), one of the biggest challenges was to reach an algorithm that required as little logic (memory, specially) as possible, so it could fit completely in a single, low-cost FPGA. The final prototype, successfully demonstrated, is capable of producing the desired results (capturing the image, analyzing it, extracting the pupil parameters, and showing results on the display) at a rate of approximately 24 frames per second.

All the circuits, including the main proposed algorithm and the other parts (camera controller, display controller, pixel memory, etc.) were coded using an HDL (Hardware Description Language). More specifically, VHDL was employed in most hardware blocks, with Verilog used in some of them. The use of an HDL is of great importance because it leads to a modern and efficient project, which can be more easily simulated and synthesized, and is independent from the FPGA's technology or manufacturer. This is also important because the hardware blocks can be easily parameterized, so tests for different algorithm parameters are simpler to execute. Additionally, the project is better documented and can be reused, and the code can be used directly for the fabrication of an ASIC (Application Specific Integrated Circuit – a dedicated chip) if so is desirable.

The developed algorithm seeks the pupil's center and radius through initial estimates extracted from the image's histogram and centroid. In parallel, the captured image's borders are detected using a hardware version of Canny's algorithm. Once the initial pupil center has been estimated, 24 points are obtained on the surrounding borders are obtained using a method called *radial explosion*, developed in this dissertation. With these points, and using the RANSAC (Random Sample Consensus) algorithm, the best estimates for the real pupil's center and radius are obtained. All of this occurs in an automatic manner, in real time and with no physical contact, by simply having the person under test with his/her eye positioned in front and near the camera.

This method of analysis of the pupil is the result of a large number of investigations, including simulations and physical implementations of various alternatives, until a

satisfactory result was attained with a small amount of resources. This project can be the starting point for the development of a professional pupil measurement system.

Key words: Biometry, pupillometry, hardware, FPGA, autonomous systems

SUMÁRIO

1 INTRODUÇÃO	11
1.1 MOTIVAÇÃO	11
1.2 OBJETIVO DO TRABALHO	12
1.3 ESTRUTURA DA DISSERTAÇÃO	13
2 FUNDAMENTAÇÃO TEÓRICA	15
2.1 INTRODUÇÃO	15
2.2 O OLHO HUMANO	15
2.3 SENSORES DE LUZ TIPO CCD E CMOS	16
2.4 IMAGENS DIGITAIS	18
2.5 PROCESSAMENTO DIGITAL DE IMAGENS (PDI)	19
2.6 HISTOGRAMA DE UMA IMAGEM	20
2.7 CENTRÓIDE DE UMA IMAGEM BINÁRIA	21
2.8 CONVOLUÇÃO DE SINAIS	22
2.9 FILTRAGEM ESPACIAL (MATRIZ DE COEFICIENTES)	23
2.10 EFEITO DE BORDAS	24
2.11 FILTRO GAUSSIANO	26
2.12 FILTRO DE PREWITT	27
2.13 DETECTOR DE BORDAS DE CANNY	28
2.14 ALGORITMO RANSAC	30
2.15 ESTIMATIVA DE CÍRCULOS	31
2.16 FIFO	33
3 REVISÃO BIBLIOGRÁFICA	35
3.1 HISTÓRIA DA PUPILOMETRIA	35
3.2 MÉTODOS DE LOCALIZAÇÃO E MENSURAÇÃO DA PUPILA	36
4 DESENVOLVIMENTO E IMPLEMENTAÇÃO	41
4.1 INTRODUÇÃO	41
4.2 AQUISIÇÃO DA IMAGEM DIGITAL	41
4.3 APRESENTAÇÃO DA IMAGEM	44
4.4 UNIDADE CENTRAL DE PROCESSAMENTO	44
4.5 ESTRUTURA DO ALGORITMO PROPOSTO	46
4.6 IMAGEM ORIGINAL E CONVERSÃO EM TONS DE CINZA	48
4.7 PROCESSAMENTO SERIAL E REDUÇÃO DE RECURSOS NECESSÁRIOS	49
4.8 DETECTOR DE BORDAS DE CANNY	52
4.9 HISTOGRAMA	55
4.10 CENTRÓIDE E ESTIMATIVA DO CENTRO	58
4.11 OBTENÇÃO DAS AMOSTRAS ATRAVÉS DO MÉTODO “EXPLOSÃO RADIAL”	59
4.12 ESTIMATIVA DE MODELOS DE CÍRCULO	63
4.13 ESCOLHA DO MELHOR CÍRCULO CANDIDATO USANDO RANSAC	68
4.14 RESUMO DO MAPEAMENTO DO ALGORITMO NO HARDWARE	69
5 RESULTADOS	71
5.1 INTRODUÇÃO	71
5.2 DEFINIÇÕES GERAIS DO PROJETO	71
5.3 SISTEMA AUTÔNOMO <i>VERSUS</i> SISTEMA OPERADO POR COMPUTADOR	72
5.4 RECURSOS NECESSÁRIOS E CUSTOS DO PROJETO	73
5.5 IMPORTÂNCIA DE UTILIZAR-SE UMA HDL	76
5.6 TESTES	77
5.7 DIFICULDADES ENCONTRADAS	78

6 CONCLUSÕES E TRABALHOS FUTUROS	81
REFERÊNCIAS	83

CAPÍTULO 1

INTRODUÇÃO

1.1 MOTIVAÇÃO

A biometria e a visão computacional são recursos úteis em diversos segmentos da atividade humana, como a indústria e a área da saúde. Neste último, a tecnologia de captura e processamento digital de imagens é usada, por exemplo, para elaboração de diagnósticos médicos, normalmente complexos, os quais seriam simplesmente impossíveis de obter sem essa tecnologia.

O desenvolvimento de equipamentos para diagnóstico de uso particular e automático (isto é, que não precisam de um profissional para operar) tem recebido especial atenção, pois os exames podem então ser feitos pelo próprio paciente ou por pessoas não especializadas, permitindo assim diagnósticos mais frequentes e de menor custo.

Um dos possíveis usos médicos para esse tipo de processamento digital de imagens é a biometria óptica (BO). A BO consiste em capturar uma imagem digital (ou seqüência de imagens) do olho do paciente e aplicar a esta um algoritmo (ou um conjunto de algoritmos) para extrair dela os parâmetros de interesse (por exemplo, tamanho da pupila, tamanho da íris, e variação do tamanho da pupila com o tempo para diferentes intensidades luminosas). Esses algoritmos são desenvolvidos utilizando uma combinação de ferramentas matemáticas, como a detecção de bordas (CANNY, 1986) e a média de *pixels* vizinhos. Ressalta-se aqui que todo o exame ocorre de uma forma não invasiva.

Um exemplo de situação médica onde a pupilometria é útil é na análise de doenças que afetam o sistema nervoso. Uma dessas doenças é a Neuropatia Autonômica Diabética (NAD) (FERRARI, 2008), que é uma doença que se manifesta muitas vezes em pacientes diabéticos. É sabido que a pupila se dilata e contrai de acordo com a intensidade da luz incidente sobre ela, mudança de tamanho esta produzida pelos sistemas nervosos simpático e parassimpático. Porém, verificou-se que pacientes que sofrem de NAD apresentam taxas de variação do tamanho da pupila diferentes daquelas das pessoas que não sofrem desta doença, de modo que a pupilometria pode ser um importante aliado no diagnóstico e acompanhamento da evolução da doença.

Outra situação em que a pupilometria mostra-se útil é na detecção de pessoas sob influência de drogas (JACKSON, 2000), pois o consumo de drogas modifica a reação da

pupila à luz. Certas drogas, como o álcool, causam redução do tamanho da pupila, enquanto que outras, como o LSD e a cocaína, causam uma dilatação anormal. Conseqüentemente, policiais e fiscais podem utilizar a mensuração da pupila como exame auxiliar para verificar se um indivíduo está ou não sob efeito de alguma droga. Novamente, o exame ocorre de forma simples, rápida e não invasiva.

As aplicações descritas acima foram os principais motivadores para o desenvolvimento da técnica para pupilometria apresentada nessa dissertação. Além disso, a escassez de material acadêmico sobre a pupilometria e sobre o processamento digital de imagens especificamente para este fim também se constitui num desafio motivador. Vale citar ainda a dificuldade de encontrar ou mesmo a inexistência de um produto (pupilômetro) similar nacional e a importância do desenvolvimento de equipamentos eletrônicos para o bem-estar das pessoas, particularmente no acompanhamento e prevenção de doenças e progresso geral da medicina.

1.2 OBJETIVO DO TRABALHO

O objetivo deste trabalho é estudar, desenvolver e testar um sistema autônomo e simples para análise da pupila humana, com a finalidade de obter informações que permitam estimar o seu diâmetro, tanto de forma estática (pupila sujeita a luz com intensidade constante) quanto dinâmica (pupila sujeita a luz com intensidade variável). Tal sistema deve ser portátil e não invasivo, capaz de operar em tempo real e de forma autônoma, e deve ser implementado utilizando técnicas digitais modernas (HDL e FPGA), inteiramente em *hardware*, com custo tão baixo quanto possível.

Os principais componentes do sistema podem ser vistos no diagrama da Figura 1. Como pode-se observar, eles consistem em uma câmera digital para obtenção das imagens, uma FPGA contendo todos os circuitos para processamento das imagens e controle da câmera e do display, um conjunto de botões de comando a serem acionados pelo operador do sistema, e um display LCD para visualização das imagens e resultados durante os testes.

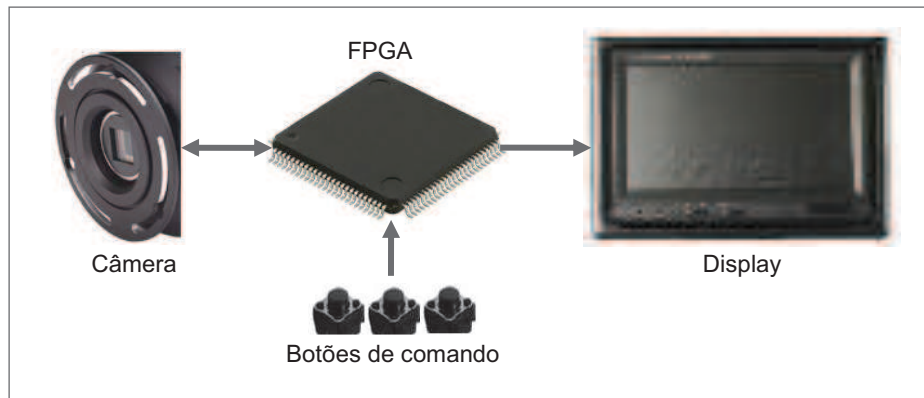


Figura 1 - Diagrama do sistema, mostrando seus principais componentes.

1.3 ESTRUTURA DA DISSERTAÇÃO

Esse trabalho está dividido em seis capítulos. No primeiro capítulo, consta a introdução. No segundo capítulo, a fundamentação teórica é realizada, abordando os aspectos essenciais para o pleno entendimento da dissertação. No terceiro capítulo, uma revisão bibliográfica é feita, na qual os principais artigos e demais materiais bibliográficos estudados ao longo do desenvolvimento da dissertação e os métodos e algoritmos usados para produzir a versão final do sistema são comentados. No quarto capítulo, explica-se a estrutura final do projeto, tanto em termos de algoritmo quanto de hardware, e como esses dois elementos interagem. No quinto capítulo, os resultados obtidos são apresentados. Finalmente, no último capítulo, constam um resumo das principais conclusões e sugestões para possíveis trabalhos futuros.

CAPÍTULO 2 FUNDAMENTAÇÃO TEÓRICA

2.1 INTRODUÇÃO

Este capítulo aborda aspectos considerados fundamentais para a plena leitura e entendimento da dissertação, tanto para esclarecer a proposta do trabalho quanto para a apresentação dos métodos utilizados. O leitor será contextualizado nos fundamentos teóricos essenciais, tanto na área biológica, a qual abrange o objeto sob estudo (conjunto olho-pupila), quanto na área de engenharia e matemática, utilizada no desenvolvimento do projeto.

2.2 O OLHO HUMANO

O olho, órgão presente em quase todos os animais (NILSSON, 2001), tem por finalidade capturar ondas luminosas e transformá-las em impulsos eletro-químicos, os quais são enviados ao cérebro, onde são processados, formando aquilo que interpretamos como imagem. Alguns detalhes do olho humano podem ser observados na Figura 2, que mostra uma secção do mesmo.

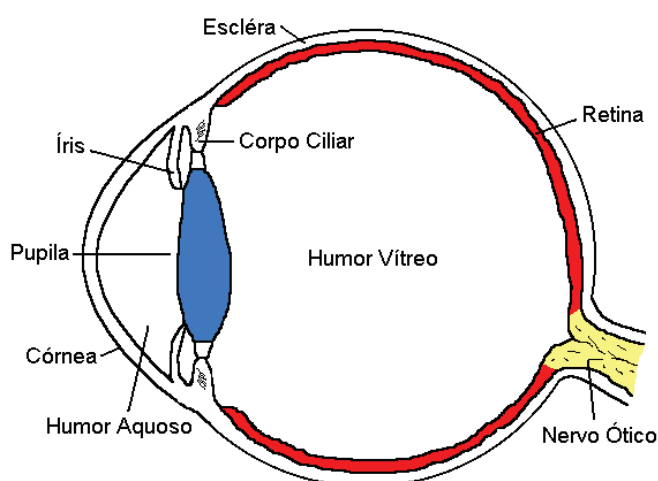


Figura 2 - Secção do olho humano (modificada de <http://www.odec.ca>).

A luz entra através de um tecido fino, chamado de córnea, passa por um líquido transparente, chamado humor aquoso, localizado na parte dianteira do olho, atravessa uma abertura na íris, conhecida como pupila, a qual se regula em abertura de acordo com a intensidade da luz entrante, e é focalizada pelo cristalino sobre a retina, na qual se localizam os fotorreceptores responsáveis em transformar a luz em impulsos eletroquímicos que são enviados ao cérebro através do nervo óptico. Outras partes fundamentais do olho são a esclera, que é a parte branca do olho, e o humor vítreo, que é outro líquido que preenche a cavidade posterior do olho, anterior à córnea.

O controle do tamanho da pupila (i.e., sua dilatação ou contração) ocorre através de um par de músculos chamados esfíncter e músculo dilatador, respectivamente responsáveis por contrair e dilatar a pupila. Esses músculos recebem comandos dos sistemas nervosos simpático e parassimpático.

2.3 SENSORES DE LUZ TIPO CCD E CMOS

Charge-Coupled Device (CCD) é um componente que capta intensidades luminosas e as converte em sinais elétricos (tensões) analógicos. Seu uso mais comum é em máquinas fotográficas e filmadoras, onde o CCD é exposto à luz (imagem), convertendo a intensidade luminosa incidente nos seus vários pontos em uma imagem correspondente.

Um CCD é ilustrado na Figura 3(a). Ele é composto por vários blocos menores, chamados *pixels*, que fazem a captura da luz em cada ponto. Seu princípio de funcionamento é simples: trata-se de um material semicondutor, de modo que, quando exposto à luz, cargas livres são geradas, as quais crescem em quantidade de acordo com a intensidade da luz e o tempo de exposição; as cargas em cada *pixel* são coletadas em cada leitura, dando origem a uma tensão analógica proporcional à quantidade de cargas, a qual é posteriormente convertida em uma tensão digital por um conversor analógico-digital, instalado no próprio CCD ou fora dele.

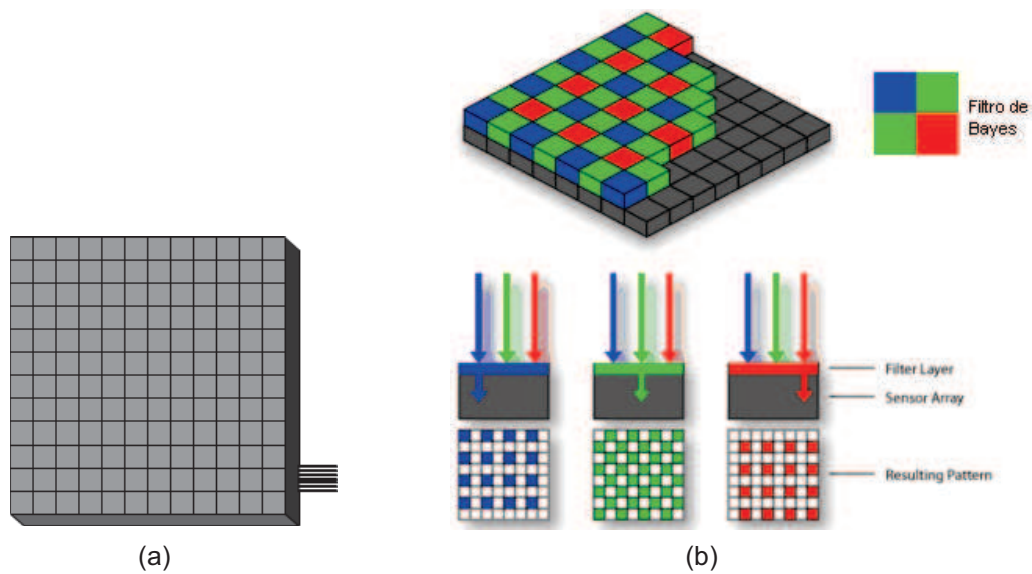


Figura 3 – Estrutura típica de CCDs.

A captura de cores é ilustrada no CCD da Figura 3(b). Como a formação de cargas é praticamente independente da cor (comprimento de onda) da luz incidente, as informações sobre cores precisam ser obtidas por algum mecanismo adicional. Uma abordagem comum é o filtro de Bayes (mostrado na parte superior direita da Figura 3(b)), que consiste em colocar sobre cada *pixel* um filtro de quatro partes, onde uma é transparente à luz azul, outra é transparente à luz vermelha, e duas são transparentes à luz verde (o olho humano é mais sensível à luz verde, daí o cuidado maior com essa cor). Quanto maior for o número de *pixels*, maior será a resolução da imagem.

Em geral, uma lente é acoplada ao CCD para focalizar a imagem corretamente sobre sua superfície. Esse processo assemelha-se muito ao processo que ocorre no olho, onde o cristalino é utilizado para focalizar a imagem corretamente sobre a retina. A qualidade e correto posicionamento da lente afetam enormemente a imagem capturada.

Outro tipo de sensor de intensidade luminosa para produção de imagens, criado depois do CCD, é o sensor CMOS (*Complementary Metal Oxide Semiconductor*). Neste sensor, são utilizados *pixels* ativos, os quais têm acesso individual, cada qual com seu próprio amplificador sensor.

Tanto o CCD quanto o sensor CMOS apresentam vantagens em determinados aspectos e ambos podem ser encontrados abundantemente no meio comercial. Resumidamente, os CCDs podem apresentar qualidade de imagem ligeiramente superior,

enquanto os sensores CMOS tendem a ser mais rápidos, consumir menos energia, ser mais fáceis de operar e ter custos menores. No presente projeto, um sensor do tipo CMOS foi utilizado.

2.4 IMAGENS DIGITAIS

Uma imagem digital é especificada por duas dimensões, uma sendo a largura (W) e outra a altura (H), como ilustra a Figura 4. Conforme já vimos, o menor elemento que compõe uma imagem digital chama-se *pixel*, havendo assim $W \times H$ *pixels* por imagem. Quanto maior for o número de *pixels*, maior será a resolução da imagem, porém maior também será a quantidade de recursos necessários para armazená-la e processá-la.

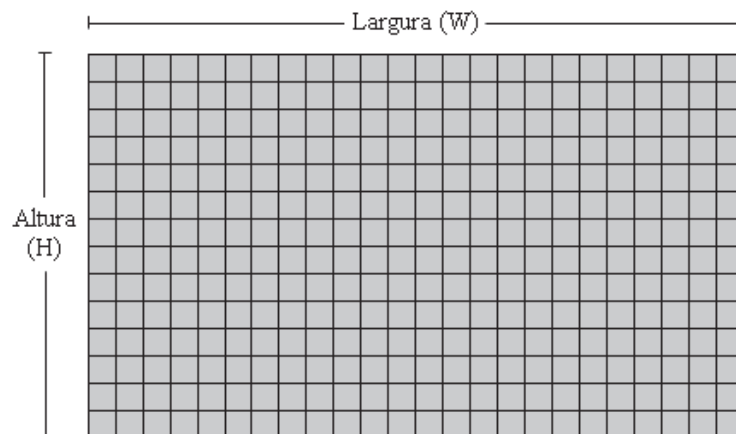


Figura 4 - Estrutura de uma imagem digital.

Ao trabalhar com uma imagem digital, é usual tratá-la como uma matriz com H linhas e W colunas de *pixels*, onde cada *pixel* é representado por um número pré-estabelecido (N) de *bits*. Valores usuais para N são 1, 8, 16, 24 e 32 *bits* por *pixel*. Quanto maior for esse número, maior será a possível diversidade de intensidades de cada *pixel*; por exemplo, um *pixel* representado por 1 *bit* pode ter somente duas cores ou intensidades (valores '0' e '1'), ao passo que com 8 *bits* pode-se ter 256 diferentes níveis para cada *pixel* da imagem, aumentando a possibilidade de representações distintas. Portanto, o tamanho de uma imagem monocromática em termos de armazenagem é calculado por $W \times H \times N$ *bits*, ou seja, seu tamanho computacional será determinado pelas dimensões da imagem vezes quantos *bits* existem por *pixel*.

Na descrição acima, considerou-se que a imagem é monocromática, contendo, portanto, somente diferentes intensidades da mesma cor (geralmente, cinza). Porém, em muitas aplicações, informações sobre cores da imagem são também necessárias.

Conforme já mostrado na Figura 3(b), uma imagem em cores pode ser decomposta em três cores fundamentais: vermelho (*Red*, R), verde (*Green*, G) e azul (*Blue*, B), formando o sistema RGB. Portanto, uma imagem colorida necessita das intensidades relativas a cada cor para cada *pixel*, o que resulta em três matrizes de dimensões $W \times H$ com N bits por *pixel*. Dessa forma, são necessários $3 \times W \times H \times N$ bits para armazenar uma imagem colorida.

Em algoritmos utilizados para sistemas de identificação através da biometria (DAUGMANN, 1992), é usual utilizar as cores da imagem para auxiliar na correta identificação do indivíduo sob teste, de modo que, nesses casos, as informações relativas às cores devem ser mantidas. Um aspecto fundamental do algoritmo proposto é que as informações sobre cores não são necessárias, podendo-se então trabalhar apenas com tons de cinza. Nesse caso, os extremos são o preto (ausência completa de luz) e o branco (saturação máxima de luz), entre os quais existem $2^N - 2$ pontos com diferentes tonalidades de cinza.

Como no protótipo da presente dissertação uma câmera em cores foi utilizada, seus *pixels* foram convertidas em tons de cinza através do seguinte cálculo: 30% da intensidade do vermelho + 59% da intensidade do verde + 11% da intensidade do azul. Esses valores são empíricos, determinados em função da sensibilidade do olho humano a cada uma dessas cores (PRATT, 1991).

2.5 PROCESSAMENTO DIGITAL DE IMAGENS (PDI)

A área de Processamento Digital de Sinais (PDS) estuda formas de processamento de sinais que, embora possam ser inerentemente analógicos, foram antes convertidos a um formato digital, possibilitando que sejam armazenados e manipulados livremente por um computador ou qualquer outro dispositivo processador, tal como um processador digital de sinais (PDS) ou um circuito instalado em uma FPGA, dentre outros.

Processamento Digital de Imagens (PDI) é um caso particular de PDS, no qual os sinais a serem processados provêm de uma ou mais imagens digitalizadas. Como citado anteriormente, a imagem digital é tratada como uma matriz de duas dimensões.

No processamento de imagens é possível utilizar algoritmos que extraíam uma informação da imagem, classifiquem algum parâmetro da mesma, ou até mesmo que encontrem algum objeto nela. Formalmente, algumas técnicas clássicas de PDI são: projeção, extração de atributos, reconhecimento, classificação e segmentação, dentre outros (GONZALEZ, 2007).

Serão descritos, nas próximas seções, os principais processos de PDI utilizados para o desenvolvimento do projeto dessa dissertação.

2.6 HISTOGRAMA DE UMA IMAGEM

Um histograma de uma imagem digital é uma representação gráfica da distribuição de tonalidades dessa imagem. São considerados todos os *pixels* presentes na imagem e suas respectivas intensidades. Para uma intensidade específica, são contados todos os *pixels* que possuem aquele valor. Uma vez que isso é feito para todas as intensidades, tem-se o histograma completo. Nessa dissertação, serão utilizados somente histogramas de tons de cinza.

Um exemplo é mostrado na Figura 5(a). Trata-se de uma imagem de tamanho 10×10 (portanto, com 100 *pixels*), onde 2 bits foram utilizados para codificar cada *pixel* (portanto, com 4 intensidades por *pixel*: preto, cinza escuro, cinza claro e branco). Nessa imagem, existem 12 *pixels* pretos (intensidade “00”), 40 cinza escuros (intensidade “01”), 19 cinza claros (intensidade “10”) e 29 brancos (intensidade “11”). O histograma correspondente consta na Figura 5(b). Somando todas as quantidades, o valor resultante tem que ser igual ao número total de *pixels* presentes na imagem (nesse exemplo, 100 *pixels*).

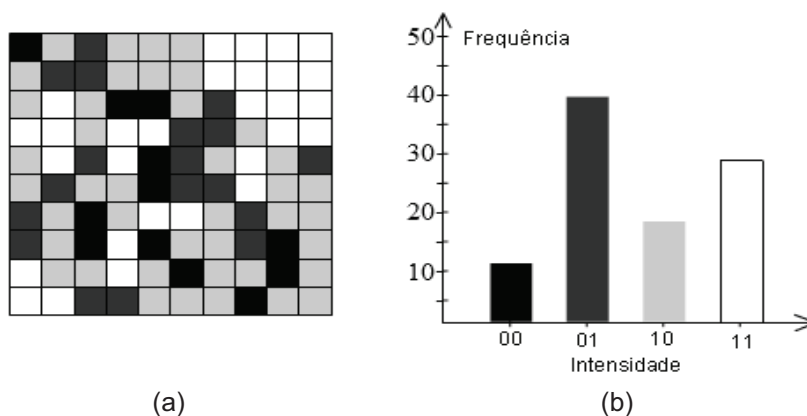


Figura 5 – (a) Exemplo de imagem digital e (b) seu histograma.

O histograma de uma imagem pode evidenciar algumas características importantes da mesma, como o contraste e a saturação geral. Diversas técnicas existem para modificar a aparência de uma imagem modificando o formato do seu histograma, como é o caso da equalização de histograma, que visa distribuir as intensidades dos *pixels* ao longo de todo o histograma de forma igual. Será visto adiante que imagens normais da pupila possuem um histograma típico, característica esta que será usada em uma das etapas do algoritmo.

2.7 CENTRÓIDE DE UMA IMAGEM BINÁRIA

Por definição, o centróide de um objeto de duas dimensões (num par de eixos x - y) é o ponto onde se localiza o seu centro de gravidade. Para calcular esse ponto, basta tomar a média de todas as abscissas e a média de todas as ordenadas dos pontos constituintes desse objeto. O ponto (c_x, c_y) resultante será seu centróide.

Esse conceito pode ser facilmente estendido às imagens. Por exemplo, para obter o centróide de uma imagem binária (isto é, em preto e branco), considera-se como objeto o conjunto de todos os pontos pretos da mesma. Um exemplo é apresentado na Figura 6, a qual mostra uma imagem com 15 *pixels* pretos e 85 *pixels* brancos. Fazendo a média aritmética das abscissas e das ordenadas dos pretos, obtém-se $c_x = 3,47$ e $c_y = 5$ (este ponto foi marcado na figura).

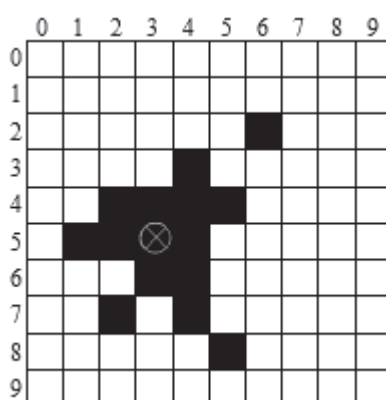


Figura 6 - Cálculo do centróide de uma imagem binária.

É importante destacar o truncamento ou arredondamento normalmente usado quando se trabalha com imagens digitais. Os *pixels* são sempre endereçados com

números inteiros, o mesmo valendo em geral para os resultados dos cálculos envolvendo esses *pixels*. Nessa dissertação, optou-se por truncamento, pela facilidade que essa opção apresenta sobre o arredondamento, visto que este último exige mais recursos para realizar no âmbito de *hardware*. Então, o valor final de c_x no exemplo da Figura 6 seria 3.

2.8 CONVOLUÇÃO DE SINAIS

Na área de PDS, uma operação encontrada com frequência é a operação de convolução, utilizada principalmente para filtragem de sinais. Como esta operação será necessária no desenvolvimento do algoritmo, uma breve revisão da mesma é apresentada nesta seção.

No âmbito de filtragem de sinais, pode-se convoluir um sinal com outro, resultando, por exemplo, na remoção de uma faixa específica de frequências do sinal filtrado ou suavização das bordas da imagem. Indiferentemente do objetivo desejado, a convolução possui um formato genérico, expresso pela Equação (1).

$$s(t) = \int_{-\infty}^{\infty} e(\tau)h(t - \tau)d\tau \quad (1)$$

Onde:

- s : sinal de saída
- e : sinal de entrada
- h : função de convolução ou função de filtragem
- t : variável no domínio padrão (tempo)
- τ : variável auxiliar na convolução

Diz-se que o sinal $s(t)$ resultante conterà o sinal filtrado pela convolução.

Uma imagem digital é um sinal. Se uma imagem for disposta como um vetor, terá uma forma de onda digital, com valores para cada amostra (i.e., para cada *pixel*). Na sua forma usual (i.e., duas dimensões, ou 2D), a imagem digital é uma matriz contendo $W \times H$ amostras, com N bits por amostra, à qual é aplicável a operação de convolução previamente descrita. Note, porém, que a Equação (1) precisa ser ajustada para o sinal 2D, resultando a Equação (2) abaixo.

$$s(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e(\tau, \vartheta) h(x - \tau, y - \vartheta) d\tau d\vartheta \quad (2)$$

Onde:

- s : sinal de saída 2D
- e : sinal de entrada 2D
- h : função de convolução ou função de filtragem 2D
- x, y : variáveis no domínio padrão
- τ, ϑ : variáveis auxiliares na convolução

As Equações (1) e (2) foram apresentadas na sua forma para variáveis contínuas (i.e., podem ter qualquer valor real). Em uma imagem digital, os valores das amostras são discretizados, assim como o são as posições espaciais de cada amostra (e.g., posição [0,1], posição [6,15]), o que é típico quando se trabalha com sistemas digitais, como na FPGA utilizada. Conseqüentemente, as variáveis utilizadas nas equações anteriores devem ser substituídas por variáveis discretas (valores inteiros). A expressão equivalente à Equação (2), para variáveis discretas, consta na Equação (3).

$$s[x,y] = \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} e[\tau, \vartheta] h[x - \tau, y - \vartheta] \quad (3)$$

A convolução no domínio discreto é mais simples de calcular, pois utiliza somatórios no lugar de integrais, reduzindo enormemente a complexidade computacional envolvida.

2.9 FILTRAGEM ESPACIAL (MATRIZ DE COEFICIENTES)

Para filtrar informações de imagens discretas, normalmente utilizam-se filtros espaciais. Tais filtros consistem na aplicação de máscaras, que são pequenas matrizes com um coeficiente em cada posição, as quais são deslocadas progressivamente sobre a matriz da imagem original (a ser filtrada), procedendo-se então à convolução entre a máscara e a imagem.

Para efeitos práticos, são usadas normalmente máscaras (matrizes) quadradas, de dimensões ímpares e não muito grandes (e.g. 3×3, 5×5). Isso se deve ao fato de que, para esses tamanhos, existe um coeficiente central bem definido e o esforço computacional não é demasiadamente grande. Adicionalmente, deve-se considerar o efeito de borda, que é a

eliminação de resultados quando a matriz de coeficientes está na região inicial ou final das linhas e colunas. O efeito de bordas será visto em mais detalhes na próxima seção.

A filtragem espacial é ilustrada na Figura 7, onde tem-se um filtro $h[x,y]$ de tamanho 3×3 deslizando sobre uma imagem de tamanho 10×10 .

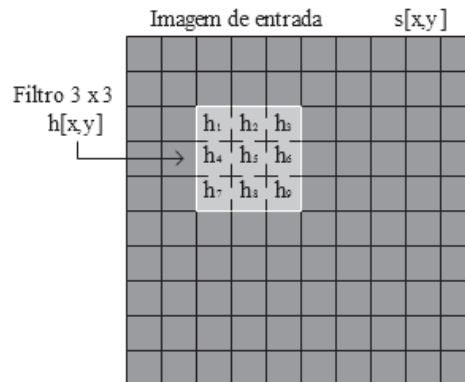


Figura 7 - Convolução de uma imagem com um filtro 3×3 .

Cada posição do filtro tem um valor de coeficiente. A primeira filtragem (convolução) acontece com a máscara posicionada sobre o primeiro *pixel* válido da imagem. Cada coeficiente de h é multiplicado pelo coeficiente da imagem que ele sobrepõe. Após realizar as nove (3×3) multiplicações, soma-se todos os valores obtidos, substituindo-se então o valor da imagem sob a posição central da máscara por este novo valor (ou, equivalentemente, pela média aritmética dos valores multiplicados). A filtragem encerra-se quando este procedimento tiver sido aplicado a todos os *pixels* válidos da imagem. Diz-se então que a imagem foi filtrada pela máscara (ou filtro).

2.10 EFEITO DE BORDAS

Um problema encontrado quando se trabalha com processamento digital de imagens é o efeito de borda. Quando se deseja realizar a convolução de um *pixel* de uma imagem com uma máscara, o processo usual é multiplicar o coeficiente de cada posição da máscara com o *pixel* correspondente da imagem abaixo dele. Para um dado *pixel*, seu valor processado será dado quando o centro da máscara estiver sobre ele, como visto na Figura 8.

⋮	⋮	⋮	⋮	⋮	⋮	⋮	
⋮	5,2	5,3	5,4	5,5	5,6	5,7	⋮
⋮	6,2	6,3	6,4	6,5	6,6	6,7	⋮
⋮	7,2	7,3	7,4	7,5	7,6	7,7	⋮
⋮	8,2	8,3	8,4	8,5	8,6	8,7	⋮
⋮	9,2	9,3	9,4	9,5	9,6	9,7	⋮
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Figura 8 - Convolução de uma máscara 3×3 com uma imagem digital.

No exemplo da Figura 8, pode-se ver a convolução entre uma máscara 3×3 e os *pixels* sob essa máscara. No entanto, o valor produzido nesse instante da convolução diz respeito ao *pixel* central, ou seja, aquele que está diretamente abaixo do coeficiente central da máscara, nesse caso representado pelo *pixel* de posição (7,5) (ou seja, com coordenadas $x=7$ e $y=5$). Portanto, é possível ver que para obter o valor processado de um *pixel* qualquer, usam-se valores de *pixels* adjacentes e a quantidade de *pixels* vizinhos que serão usados depende diretamente do tamanho da máscara escolhida.

Quando a máscara de convolução se aproxima das bordas da imagem, suas extremidades se posicionarão fora da imagem, sobre posições onde não existem *pixels*. Um exemplo desse caso pode ser visto na Figura 9.

	0,0	0,1	0,2	0,3	⋮
	1,0	1,1	1,2	1,3	⋮
	2,0	2,1	2,2	2,3	⋮
	3,0	3,1	3,2	3,3	⋮
	⋮	⋮	⋮	⋮	⋮

Figura 9 - Convolução próximo às bordas da imagem.

Essa tentativa de processamento com posições inexistentes cria o chamado efeito de borda da convolução de imagens.

Algumas formas de tentar diminuir esse efeito são a repetição de valores da borda e o espelhamento dos valores da borda. Na repetição, as posições inexistentes assumem o

valor de intensidade igual ao *pixel* existente mais próximo. Já no espelhamento, as posições inexistentes recebem valores de acordo com os *pixels* em posições simétricas e espelhadas de si. Nesse projeto, optou-se por utilizar o espelhamento e um exemplo disso pode ser visto na Figura 10, onde se tem uma máscara 1×7 e os *pixels* das posições (0,1), (0,2) e (0,3) foram espelhados.

0,3	0,2	0,1	0,0	0,1	0,2	0,3	...
			1,0	1,1	1,2	1,3	...
			2,0	2,1	2,2	2,3	...
			⋮	⋮	⋮	⋮	

Figura 10 - Exemplo de espelhamento de *pixels* para reduzir o efeito de bordas.

Por mais que o método do espelhamento ou repetição possa reduzir o efeito de bordas, os valores finais produzidos para os *pixels* de borda se baseiam em valores artificiais, uma vez que essas posições não existem originalmente na imagem digital. Esse é outro motivo pelo qual se evita utilizar máscaras que tenham dimensões muito grandes.

2.11 FILTRO GAUSSIANO

A curva Gaussiana (ou ainda curva ou distribuição normal) é uma distribuição matemática bastante encontrada em aplicações de engenharia, especialmente quando envolvem processamento de sinais. A curva Gaussiana corresponde à seguinte equação matemática:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad (4)$$

Onde:

- $f(x)$: Distribuição Gaussiana
- x : Variável no domínio padrão
- μ : Média da distribuição
- σ^2 : Variância da distribuição

Grande parte da utilidade da distribuição normal provém do fato que essa curva representa a distribuição de probabilidade de diversos processos de interesse à engenharia e outras áreas. Por exemplo, o teorema do limite central diz que, para um número grande de amostras aleatórias, a distribuição é uma curva Gaussiana.

Para a área de processamento digital de imagens, um dos usos da distribuição Gaussiana se encontra na filtragem de imagens. Como as imagens estão digitalizadas, é necessário também ter-se um filtro em forma digital, com coeficientes discretos, os quais, conforme já visto, são usualmente quadrados e de tamanho ímpar. Para manter a máxima semelhança com o filtro Gaussiano contínuo, considera-se a posição central do filtro como sendo o valor máximo da curva, enquanto que, para as outras posições, são adotados valores simétricos, resultando um filtro com simetria radial, ou seja, qualquer posição que tenha a mesma distância da posição central terá necessariamente o mesmo valor. Um exemplo de filtro Gaussiano, de tamanho 5x5, pode ser visto na Figura 11.

$$\frac{1}{237} \times \begin{array}{|c|c|c|c|c|} \hline 1 & 4 & 7 & 4 & 1 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 7 & 26 & 41 & 26 & 7 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 1 & 4 & 7 & 4 & 1 \\ \hline \end{array}$$

Figura 11 - Exemplo de filtro Gaussiano.

A forma de executar a filtragem segue a explicação da seção anterior. Após esse processo, a imagem será uma versão borrada da imagem original. Essa filtragem é útil principalmente para reduzir ruído contido na imagem, suavizando seu conteúdo.

O filtro Gaussiano é um dos componentes do detector de bordas de Canny, que será descrito na seção 2.13. Detalhes do filtro Gaussiano serão vistos também nos outros capítulos.

2.12 FILTRO DE PREWITT

O filtro de Prewitt (ou operador de Prewitt) é um operador de diferenciação discreto. Ele funciona baseado em dois filtros, um que calcula o diferencial horizontal e outro que calcula o diferencial vertical. Exemplos podem ser vistos na Figura 12.

-1	0	1	-1	-1	-1
-1	0	1	0	0	0
-1	0	1	1	1	1

Figura 12 - Filtros de Prewitt.

Uma borda em uma imagem digital é normalmente a junção de duas regiões com diferenças de intensidade significativas. Com um filtro como o de Prewitt, é possível detectar certas bordas. Dada sua natureza, a aplicação de cada filtro em separado permite determinar o mapa do gradiente da imagem, destacando os pontos que têm as maiores diferenças de intensidade.

Devido a essas características, o filtro de Prewitt é mais um dos componentes do detector de bordas de Canny, descrito a seguir.

2.13 DETECTOR DE BORDAS DE CANNY

Detectores de borda são algoritmos que visam extrair de uma imagem digital as bordas dos objetos nela presentes, sejam essas imagens coloridas ou em tons de cinza. O algoritmo criado por Canny (CANNY, 1986) serve como um robusto detector de bordas que funciona relativamente bem mesmo com dados ruidosos, como é o caso de imagens obtidas com câmeras de custo reduzido (a menor qualidade dos sensores implica em uma maior presença de ruído na imagem final).

O funcionamento do detector de bordas de Canny é ilustrado na Figura 13. Na Figura 13(a), uma imagem digital original de um olho humano é mostrada, ao passo que a Figura 3(b) mostra a mesma imagem após ter sido processada pelo detector.

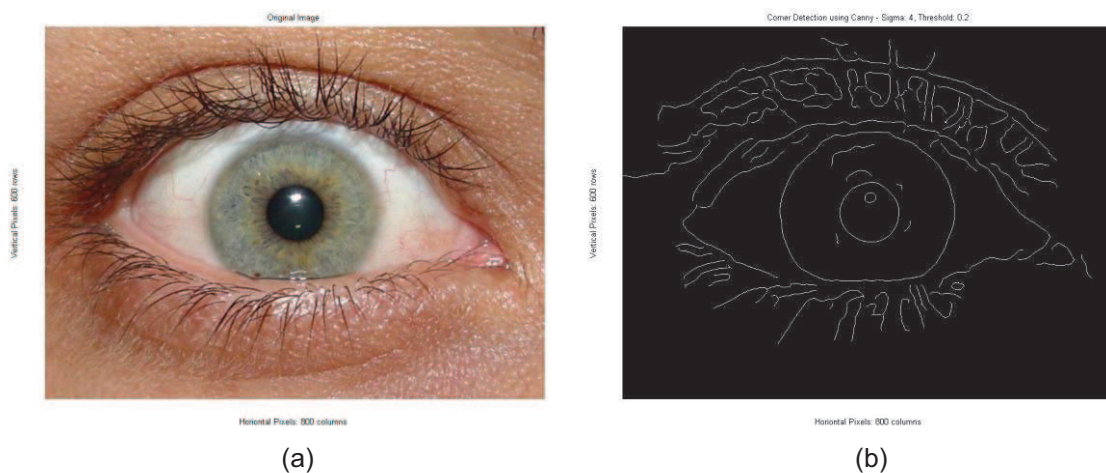


Figura 13 – (a) Imagem original; (b) Imagem após passar pelo detetor de bordas de Canny.

A primeira fase do detetor de bordas de Canny visa suavizar a imagem de entrada, com o propósito de amenizar o ruído presente ao longo da mesma. Isso pode ser realizado por um filtro passa-baixas, como o filtro Gaussiano, onde é utilizada a convolução, tendo como resultado uma imagem suavizada.

A segunda etapa se dedica a encontrar o gradiente da imagem, que é a mudança direcional de intensidade ao longo dessa mesma. Uma mudança absoluta maior da intensidade da imagem produz um valor de gradiente maior. Um filtro como o filtro de Prewitt pode alcançar esse objetivo. O propósito de achar-se o gradiente é que bordas normalmente representam mudanças altas nos valores de intensidade ao longo da imagem, de forma que o gradiente fornece uma boa estimativa inicial para as bordas presentes na imagem.

Após o cálculo do gradiente, a fase de supressão dos valores não máximos é aplicada. Nesse passo, verifica-se se a magnitude do gradiente assume um máximo local na direção do gradiente, com base na intensidade dos vizinhos que seguem aquela direção. Se a magnitude for maior do que os valores vizinhos, então o valor é um máximo local e é marcado como tal. Valores que não preenchem esse requisito são marcados como não sendo de borda e têm suas intensidades zeradas.

Finalmente, esse mapa de valores resultante é avaliado. Utilizam-se dois limiares (*thresholds*), um alto e um baixo. Toda intensidade que não foi zerada na fase anterior é comparada com esses valores. Os valores situados acima do limiar alto (T_H) são considerados bordas verdadeiras, enquanto os valores inferiores ao limiar baixo (T_L) são zerados. Já os pontos situados entre T_L e T_H serão reavaliados. Averigua-se se algum

vizinho desses pontos é uma borda verdadeira; em caso positivo, esse ponto também é considerado como tal.

2.14 ALGORITMO RANSAC

O RANSAC (*Random Sample Consensus*) é um algoritmo iterativo para estimar parâmetros de um modelo matemático a partir de um grupo de amostras. É um algoritmo eficiente para grupos que possam ter amostras erradas devido à má estimação dessas ou à presença significativa de ruído.

O algoritmo baseia-se na ideia de que, dentre as amostras presentes, devem existir algumas que preencham corretamente algum modelo matemático e seus parâmetros (e.g., um círculo), embora possam existir no conjunto de amostras casos que não obedecem àquele modelo. Cabe ao modelo determinar quais amostras serão aceitas como corretas e quais parâmetros se encaixam naquele modelo.

O método RANSAC seleciona iterativamente um subgrupo de amostras. Consideram-se essas amostras temporariamente como corretas e essa hipótese é testada da seguinte maneira:

1. Um modelo é criado com base em um grupo de amostras consideradas corretas (indiferentemente de serem realmente corretas ou não). Determina-se então os parâmetros do modelo. No presente caso, o objetivo é determinar o centro de um círculo;
2. Todas as outras amostras são testadas considerando o modelo gerado como verdadeiro. Se uma amostra se encaixar no modelo, contabiliza-se isso como um acerto;
3. O modelo estimado recebe uma avaliação baseada no número de amostras que com sucesso satisfizeram os parâmetros do modelo gerado, ou seja, de acordo com o número de acertos;
4. No final do processo, volta-se ao passo 1 e repete-se todas as instruções, até que todos os grupos tenham sido testados.

Esse processo é repetido por um número determinado de vezes. No exemplo dessa dissertação, são gerados 32 círculos modelos e todos são testados. O modelo de círculo que tiver o maior número de acertos após o teste de todos os modelos será considerado o melhor representante da pupila na imagem.

Na Figura 14, é possível ver a simulação de um ambiente com ruído e amostras errôneas. No entanto, através de um subgrupo dessas amostras, gera-se o círculo desenhado em azul. Como esse modelo terá um número de acertos (amostras que condizem com seu modelo e estão posicionadas próximas ao seu raio) maior do que os outros modelos, ele será escolhido como o melhor candidato, mesmo que existam diversas amostras que não correspondem ao modelo escolhido.

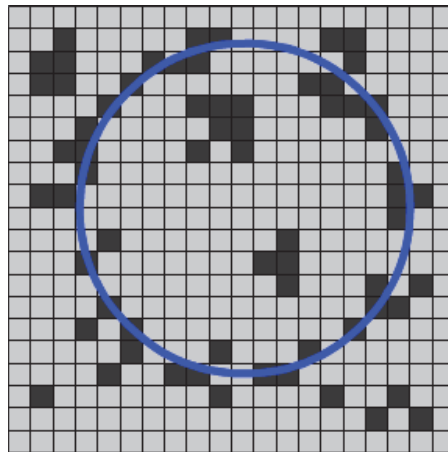


Figura 14 – Aplicação do algoritmo RANSAC para encontrar uma circunferência em um ambiente ruidoso.

2.15 ESTIMATIVA DE CÍRCULOS

Dado um conjunto de três pontos não colineares, é possível traçar uma circunferência única de raio R e centro (x_C, y_C) que passe pelos três pontos. A Figura 15 mostra um exemplo.

Uma das técnicas para determinar o centro do círculo gerado é descrita a seguir. Primeiramente, são obtidas as equações de duas retas, chamadas AB e BC, que passam, respectivamente, pelos pontos A-B e B-C da circunferência, conforme consta nas Equações (5) e (6).

$$y_{AB} = m_{AB}(x - x_A) + y_A \quad (5)$$

$$y_{BC} = m_{BC}(x - x_B) + y_B \quad (6)$$

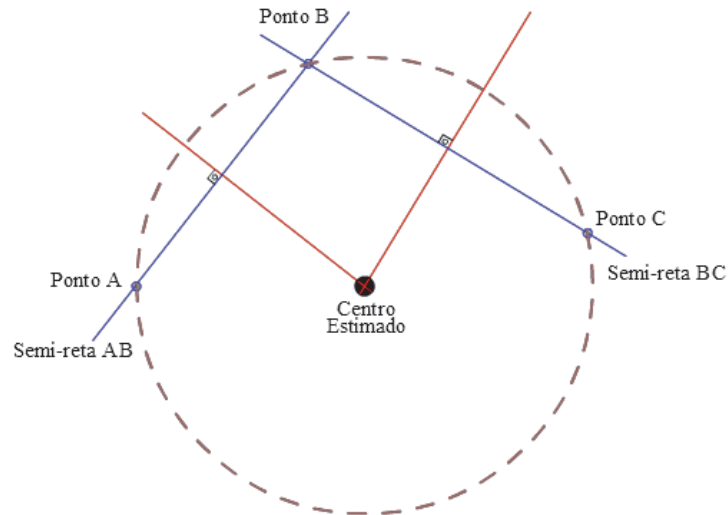


Figura 15 - Geração de círculo a partir de três pontos não colineares.

Onde:

- y_{AB}, y_{BC} – coordenadas das retas AB e BC
- x, y – variáveis de entrada das equações
- x_A, y_A – coordenadas do ponto A
- x_B, y_B – coordenadas do ponto B
- m_{AB}, m_{BC} – coeficientes angulares das retas AB e BC

Por sua vez, o coeficiente angular (m) dessas retas pode ser determinado pelas Equações (7) e (8).

$$m_{AB} = \frac{y_B - y_A}{x_B - x_A} \quad (7)$$

$$m_{BC} = \frac{y_C - y_B}{x_C - x_B} \quad (8)$$

O centro do círculo será o ponto de encontro das mediatrizes das retas AB e BC. Para encontrar essas retas, sabemos que o coeficiente angular de qualquer reta perpendicular à outra reta que tenha coeficiente angular m é $-1/m$. Então as equações das retas que cortam perpendicularmente ao meio os segmentos AB e BC são determinadas pelas Equações (9) e (10).

$$y'_{AB} = -\frac{1}{m_{AB}} \cdot \left(x - \frac{x_A + x_B}{2} \right) + \frac{y_A + y_B}{2} \quad (9)$$

$$y'_{BC} = -\frac{1}{m_{BC}} \cdot \left(x - \frac{x_B + x_C}{2} \right) + \frac{y_B + y_C}{2} \quad (10)$$

Como as duas retas se encontram no centro do círculo, para encontrar esse ponto basta isolar a coordenada x na equação, como realizado na Equação (11).

$$x = \frac{m_{AB}m_{BC}(y_A - y_C) + m_{BC}(x_A + x_B) - m_{AB}(x_B + x_C)}{2(m_{BC} - m_{AB})} \quad (11)$$

Para encontrar a coordenada y do centro, insere-se o valor de x encontrado em uma das equações das retas normais.

Finalmente, para obter o raio, é preciso apenas calcular a distância euclidiana do centro do círculo determinado até qualquer um dos três pontos geradores, dado que esses pontos são constituintes do perímetro do círculo.

2.16 FIFO

Este é o último dos aspectos fundamentais a serem comentados nesta parte introdutória da dissertação.

Em eletrônica e em *hardware*, FIFO (*First-In First-Out*) é uma espécie de fila ou série de blocos de memória, com tamanho constante, onde, a cada ciclo de funcionamento, um novo valor é armazenado no fim da fila, causando assim o movimento de todos os demais blocos uma posição adiante e, por conseguinte, levando à eliminação do bloco que era anteriormente o primeiro da fila.

A FIFO utilizada em *hardware* puro, como é o caso da FIFO usada nessa dissertação e quando se trabalha com FPGAs, não deve ser confundida com o conceito de FIFO da computação. Ambos são tipos de filas, mas a FIFO da computação (em *software*) pode possuir tamanho virtualmente ilimitado e pode modificar esse tamanho de forma dinâmica, através do conceito de alocação dinâmica e, na entrada de um novo dado no final da fila, não precisa eliminar o dado que está no começo da mesma. Como citado, as FIFOs utilizadas nesse projeto são de tamanho fixo e efetuam o deslocamento de seu conteúdo uma vez por ciclo de clock.

FIFOs são especialmente úteis em eletrônica quando é necessário sincronizar dois sinais digitais mas um está atrasado em relação ao outro; pode-se usar uma FIFO para atrasar o sinal digital que está adiantado por um número específico de ciclos e assim ter os dois sinais sincronizados. Outra utilidade comum é quando se quer apenas armazenar alguns valores mas não se deseja utilizar uma memória inteira para tal; com uma FIFO, é possível armazenar apenas blocos de informação por vez e utilizá-los somente no ciclo desejado. Essas duas abordagens de FIFO serão bastante utilizadas no hardware desenvolvido neste projeto.

Com o término desta seção, tem-se o material básico necessário para melhor compreensão dos principais princípios adotados no desenvolvimento da dissertação.

CAPÍTULO 3

REVISÃO BIBLIOGRÁFICA

Este capítulo objetiva apresentar as principais referências bibliográficas utilizadas para o desenvolvimento da dissertação. Primeiramente, será apresentada uma breve história da pupilometria. Em seguida, serão apresentados os métodos usados na versão final do projeto. Todos os itens estão listados por ordem cronológica de estudo.

3.1 HISTÓRIA DA PUPILOMETRIA

O estudo da pupila, do seu tamanho e as formas de como se dilata e se contrai pode ser datado aos dias de Charles Darwin, no século XIX. Já era sabido na época que a pupila mudava de tamanho de acordo com a intensidade da luz incidente sobre a mesma, porém estudos relatavam também que existia uma relação entre as sensações ou emoções de um indivíduo e a contração ou expansão da pupila. Um experimento contido em um desses estudos (DARWIN, 1871) apontava que a pupila de um indivíduo se expandia ao ouvir o som de um tiro de revólver, dessa forma sugerindo haver uma conexão entre a ação da pupila e uma resposta ao espanto.

Houve diferentes maneiras de tentar mensurar a pupila ao longo do tempo. Antes da década de 60 do século XX, a medida da pupila era comumente feita através de métodos que incidiam luz sobre os olhos do paciente e se utilizavam da luz refletida na íris, ou através do uso de luz infravermelha (GONZALEZ, 2007). No entanto, esses equipamentos eram massivos e a precisão dos exames era muito baixa.

Em meados dos anos 60, o interesse em medir a pupila já se refletia nos avanços das tecnologias disponíveis para tal. Um pupilômetro criado por Otto Lowenstein (THOMPSON, 2005) visualizava o olho através de filmes, mensurava as duas pupilas e informava o operador sobre a taxa de mudança do tamanho das mesmas. Contudo, novamente a precisão dos resultados era baixa e o tamanho do equipamento o tornava inviável para uso generalizado.

Já nos anos 70, Eckhard Hess (WAITE, 1999) usava uma câmera de vídeo, um projetor e espelhos em seu pupilômetro. Usando um filme infravermelho, o aparelho

obtinha uma média de 20 quadros por segundo durante a mensuração. No entanto, embora o aparelho fosse economicamente viável, precisava de uma luminosidade muito bem controlada, a mensuração era demorada e o resultado também era impreciso.

Atualmente, os pupilômetros são equipamentos tecnologicamente avançados e têm níveis de precisão cada vez maiores. Alguns dos mais modernos adotam algum tipo de câmera e processamento de imagens, utilizando-se de processadores de sinais para tal. O tamanho e o custo dos aparelhos foram reduzidos significativamente, ao mesmo tempo que a precisão dos mesmos aumentou. Atualmente, já existem pupilômetros comerciais de uso doméstico, porém, pelo que foi possível obter de informação pela pesquisa realizada, não existe uma versão nacional de um pupilômetro digital.

3.2 MÉTODOS DE LOCALIZAÇÃO E MENSURAÇÃO DA PUPILA

Existem várias formas propostas para a correta detecção da pupila. Todos os métodos possuem algum consenso em relação ao tipo e à qualidade da imagem que está sendo avaliada. Por exemplo, o olho do indivíduo a ser avaliado precisa estar de forma relativamente clara e focada na imagem para que o teste possa ocorrer corretamente. Como é o caso para a maioria dos problemas de PDI, o algoritmo é suscetível a algum tipo de erro, mas medidas são tomadas para reduzir esses casos.

Uma forma de detectar a pupila em uma imagem, elaborada por Wildes et al. (WILDE, 1994), baseia-se na detecção de bordas da imagem e utilização da transformada de Hough. A detecção de bordas ocorre utilizando-se um detector de bordas de Canny. Já a transformada de Hough (SHAPIRO, 2001) é uma técnica de extração de atributos de uma imagem digital, a qual procura elementos que seguem algum modelo matemático conhecido, como, por exemplo, um círculo, utilizando-se de um processo de votos. Com base na imagem original e aplicando a transformada de Hough, obtém-se o chamado “espaço do acumulador”. Dentro desse espaço, o objeto que mais se assemelhar ao modelo matemático proposto terá um índice de acúmulo alto, assim determinando qual é o objeto procurado e seus parâmetros.

A transformada de Hough, em sua forma básica, procura apenas por linhas dentro de uma imagem. No entanto, no começo da década de 70, Peter Hart et al. desenvolveram a versão genérica da transformada, permitindo assim a busca por elementos mais complexos, como círculos, nas imagens (DUDA, 1972).

A transformada de Hough apresentou-se como uma técnica eficaz para detectar formas matemáticas dentro de uma imagem digital, tolerando certo grau de ruído. Porém, o cálculo da transformada e os acumuladores necessários para obter o resultado final são computacionalmente complexos e extensos. Para contornar isso, Rad et al. (RAD, 2003) propôs um algoritmo da transformada de Hough que exige menos memória e é mais simples de executar, específico para a detecção de círculos, chamado *detecção rápida de círculos (Fast Circle Detection - FCD)*.

O FCD é composto por quatro fases. Primeiramente, calcula-se o gradiente da imagem, de forma similar ao detector de bordas de Canny. A segunda fase consiste em achar os vetores de gradiente que estejam separados por um ângulo de 180° um do outro e que a reta que atravessa os dois pontos esteja na mesma direção do gradiente de cada ponto. Em seguida, para cada par de vetores que atende às especificações da segunda fase, adota-se um círculo candidato, com centro na metade do segmento de reta que liga os pontos e raio igual à metade desse comprimento. Por último, os círculos candidatos são agrupados pela ordem da distância entre eles e os círculos existentes são detectados com base em algum critério, como a média do grupo.

Outro método bastante usado para encontrar círculos é a detecção circular de bordas proposta por Daugman (DAUGMAN, 1992), o qual funciona através da aplicação do operador integral-diferencial descrito na Equação (12).

$$\max(r, x_0, y_0) = \left| G_\sigma(r) * \frac{\partial}{\partial r, x_0, y_0} \oint \frac{I(x, y)}{2\pi r} ds \right| \quad (12)$$

Onde:

- r – raio da pupila
- x_0 – coordenada x do centro da pupila
- y_0 – coordenada y do centro da pupila
- $G_\sigma(r)$ – função de suavização da imagem (em geral, Gaussiana)
- $I(x, y)$ – imagem original

No método de Daugman, a imagem original primeiramente é suavizada por um filtro Gaussiano. Em seguida, procura-se um círculo que tenha a maior mudança de gradiente, similar ao processo explicado para a transformada de Hough, onde a maior variação aponta as bordas da imagem. Esse método é bastante utilizado atualmente, especialmente para casos onde a detecção da pupila e da íris são necessárias.

Pode-se detectar a pupila também através da análise do histograma da imagem digital. A pupila, em uma imagem digital de boa qualidade, é usualmente a parte mais escura presente, significando que os *pixels* que a compõe estarão localizados na parte baixa do histograma (recordando que o histograma é a distribuição de todos os *pixels* de acordo com suas intensidades). Para tanto, usa-se o limiar de segmentação da pupila, o qual é o valor mínimo local entre o primeiro e o segundo máximos do histograma. Todos os *pixels* abaixo desse limiar serão considerados constituintes da pupila. Faz-se então a média das posições dos *pixels* abaixo do limiar, estimando-se assim um centro para a pupila (PAN, 2005).

Para a realização do processo completo de mensuração, não basta apenas localizar o objeto. Com uma aproximação do centro da pupila obtida, pode-se começar a estimar o centro e o raio verdadeiros. Uma proposta para tal é o algoritmo chamado *starburst* (LI, 2005), ou explosão em estrela (algumas iterações do algoritmo podem ser vistas na Figura 16).

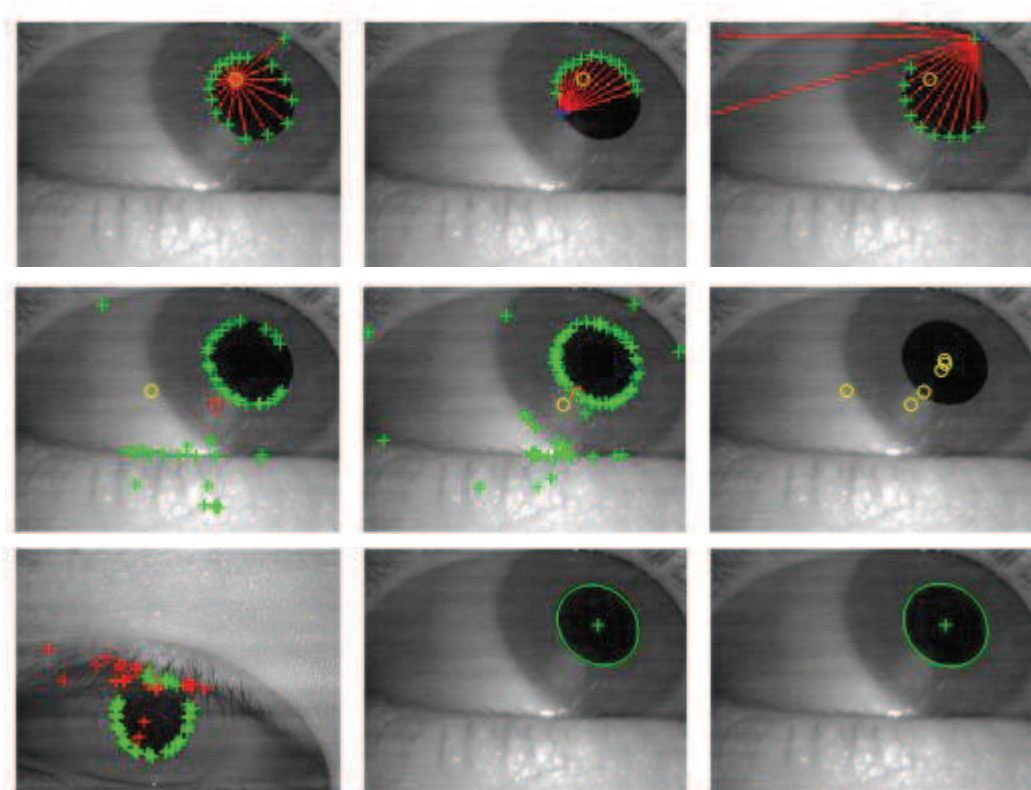


Figura 16 – Diversas iterações do algoritmo *starburst* (modificado de LI, 2005).

O algoritmo estima um centro e , baseado em sua posição, calcula um conjunto de retas que interseccionam o ponto de centro da pupila. O número de retas usado é variável e a precisão do algoritmo aumenta com esse número. Em seguida, calcula-se quais pontos de borda estão sobre essas retas. Para cada ponto encontrado, é refeito o processo anterior. O número de iterações também é variável, mas é limitado para não tornar o processo muito lento. Com um conjunto substancial de pontos, calcula-se algumas elipses candidatas. Em seguida, o método RANSAC é usado para estimar qual elipse tem o maior número de pontos supostamente corretos, a qual é então considerada como sendo o contorno da pupila.

O algoritmo é computacionalmente lento se comparado com outros métodos, porém apresenta uma taxa de erro menor. Para o algoritmo desenvolvido nesse projeto, usam-se diversos conhecimentos dos algoritmos apresentados, levando em consideração as necessidades específicas do presente sistema.

Com esse conjunto de bibliografia explanado, juntamente com o material apresentado no Capítulo 2, têm-se os conhecimentos necessários para introduzir as ferramentas e métodos utilizados na criação do algoritmo dessa dissertação.

CAPÍTULO 4

DESENVOLVIMENTO E IMPLEMENTAÇÃO

4.1 INTRODUÇÃO

Durante os estudos e implementações do projeto da dissertação, mudou-se uma série de vezes a concepção de como seria a sua versão final. Essas mudanças ocorreram tanto em termos de algoritmo e código quanto no *hardware* a ser usado. As mesmas foram necessárias para atender às características do projeto com o menor custo possível.

Um esboço geral dos componentes usados no sistema pode ser visto na Figura 17(a), o qual consta também, na forma de diagrama de blocos, na Figura 17(b). Conforme já havia sido adiantado na Figura 1, os principais componentes são uma câmera digital (com lente), um display LCD, e uma FPGA, esta última contendo todos os circuitos processadores de imagens e controladores da câmera e do display. A Figura 17 mostra também os demais componentes do sistema, quais sejam, cone para bloqueio de luz externa, com iluminação interna através de LEDs infravermelhos, botões de comando, a serem acionados pelo operador do equipamento, e fonte de alimentação.

4.2 AQUISIÇÃO DA IMAGEM DIGITAL

O sistema de aquisição de imagens compõe a primeira parte do projeto. Sua função é capturar imagens do olho do indivíduo a ser testado através de uma câmera digital.

No protótipo do sistema foi utilizada uma câmera da Terasic (www.terasic.com), a qual pode ser vista na Figura 18. Ela é composta por um sensor CMOS, uma lente e um barramento de pinos para interfacear com a FPGA. As imagens saem da câmera já digitalizadas com formato RGB de 12 bits.

A câmera utilizada é fornecida em duas versões: 5 *Mega pixels* (MP), com preço de USD 85,00, e 1,3 *Mega pixels*, ao preço de USD 70,00. Foi utilizada a primeira destas, mas como a resolução máxima não foi necessária, a versão com menor preço poderia ter sido empregada (tal análise será feita no Capítulo 5).

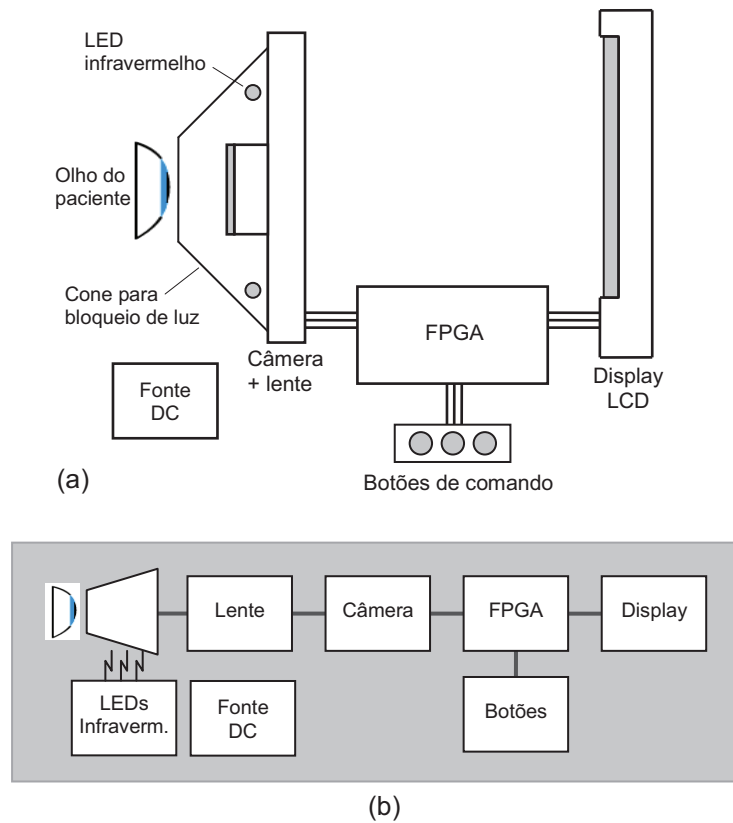


Figura 17 - Modelo do projeto.

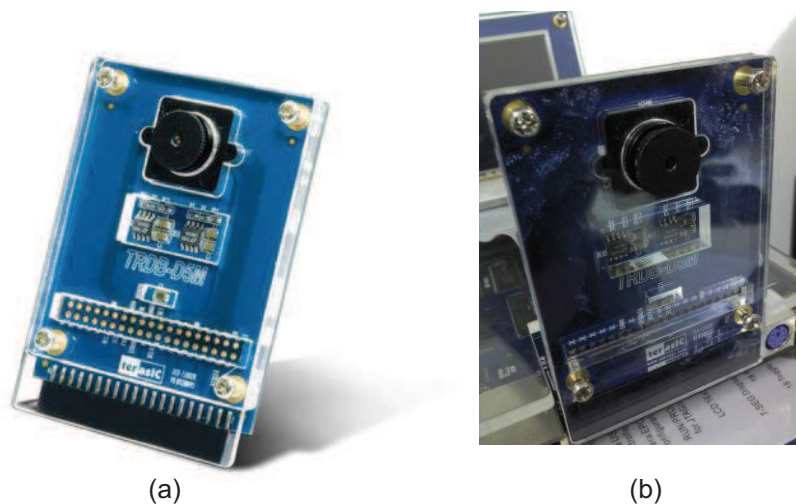


Figura 18 – (a) Câmera Terasic; (b) Câmera sendo usada no protótipo para a dissertação.

Não foi especificada pelo fabricante a lente utilizada para a câmera e dessa forma não foi possível descobrir com exatidão alguns parâmetros, como a distância focal. No

entanto, é possível regular a sua distância focal manualmente, tendo sido então configurada para ter uma imagem otimizada a uma distância de aproximadamente 5 cm.

Um detalhe importante é que foi colocado junto à câmera um conjunto de LEDs (*Light Emitting Diodes*) infravermelhos. Para obter uma imagem digital de qualidade razoável, é necessário haver uma iluminação uniforme sobre o olho do indivíduo a ser testado. Isso poderia ser feito utilizando-se LEDs comuns, com comprimento de onda na faixa do espectro visível ao olho humano. Porém, como é necessário iluminar a imagem sem afetar o tamanho da pupila do indivíduo sendo testado, usa-se LEDs infravermelhos, pois os mesmos não afetam o tamanho da pupila e incidem luz visível apenas para o sensor da câmera. Vale notar que muitas câmeras incluem um filtro infravermelho, para filtrar tais frequências e remover seu efeito sobre a iluminação da imagem. Como a necessidade nessa dissertação era justamente o contrário, foi necessário retirar essa película que ficava sobre a lente para que a câmera pudesse ser sensível a essa faixa de comprimentos de onda.

Para impedir a entrada de luz externa e ajudar o paciente a se posicionar, colocou-se um pequeno cone em frente à câmera, com abertura do tamanho aproximado do olho humano. O cone acoplado à câmera pode ser visto na Figura 19.



Figura 19 - Cone acoplado à câmera (abrigando LEDs infravermelhos) para propiciar uma iluminação mais uniforme e controlada sobre o olho.

4.3 APRESENTAÇÃO DA IMAGEM

Para que seja possível o usuário saber se está posicionado de forma correta em frente à câmera, um display LCD foi empregado para mostrar a imagem que a câmera está recebendo e processando. O indivíduo deve posicionar seu olho tal que a pupila fique aproximadamente no centro do LCD, assim melhorando a imagem a ser tratada.

O display utilizado no projeto pode ser visto na Figura 20. Ele tem resolução de 800 por 480 *pixels* e apresenta funcionalidades de *touch-screen*, com um custo de USD 170,00. Contudo, como as funcionalidades do *touch-screen* não foram usadas, esse LCD pode ser substituído por outro, sem essa função, portanto com menor custo (tal análise será feita no Capítulo 5).

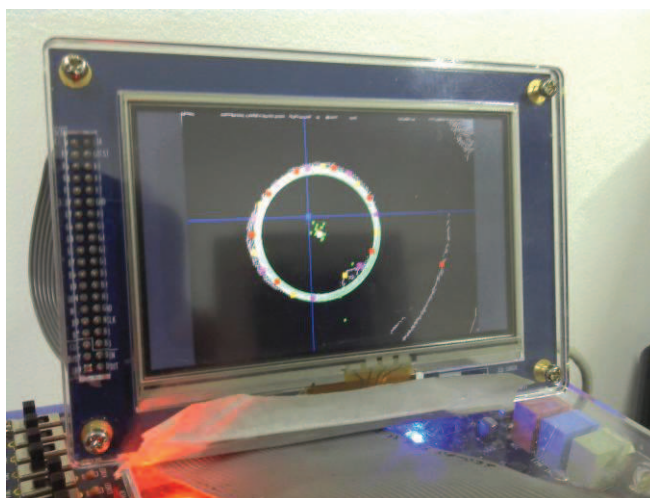


Figura 20 - LCD utilizado no protótipo do sistema.

4.4 UNIDADE CENTRAL DE PROCESSAMENTO

Conforme mencionado anteriormente, todos os circuitos (incluindo a memória) foram embarcados em uma única FPGA. Um diagrama da mesma pode ser visto na Figura 21, o qual foi dividido em quatro blocos:

- Controlador da câmera, responsável pela configuração e leitura da câmera digital;
- Interface com botões de comando, responsável pela leitura dos botões acionados pelo operador do equipamento para configuração da câmera, seleção de parâmetros do algoritmo, congelamento de imagem, etc.;

- Controlador do display, responsável pela configuração e escrita dos dados no display LCD;
- Processador de imagens, que é a parte principal do sistema, responsável pelo processamento das imagens digitais de acordo com o algoritmo proposto e armazenamento dos dados necessários.

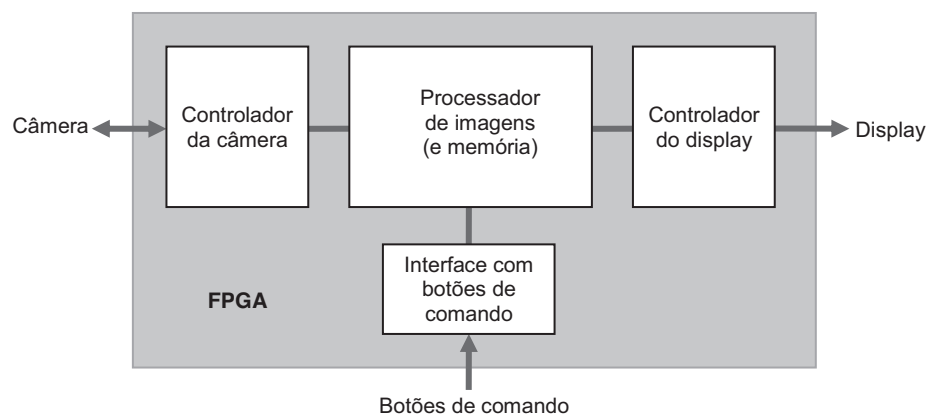


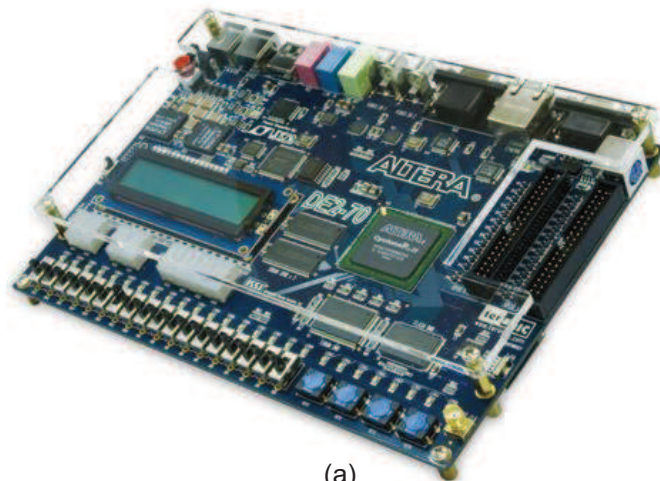
Figura 21 - Diagrama da unidade de processamento (localizada na FPGA).

A FPGA utilizada no protótipo do sistema foi uma Cyclone II (EP2C35F672C6N), da Altera, com preço de USD 149,60 (www.altera.com). A mesma tem 33 mil portas lógicas, com 475 pinos de entrada/saída disponíveis para o usuário, distribuídos em um encapsulamento FBGA (*Fine pitch Ball Grid Array*). Como no caso do LCD, esta FPGA também pode ser trocada por uma de menor custo (tal análise será feita no Capítulo 5).

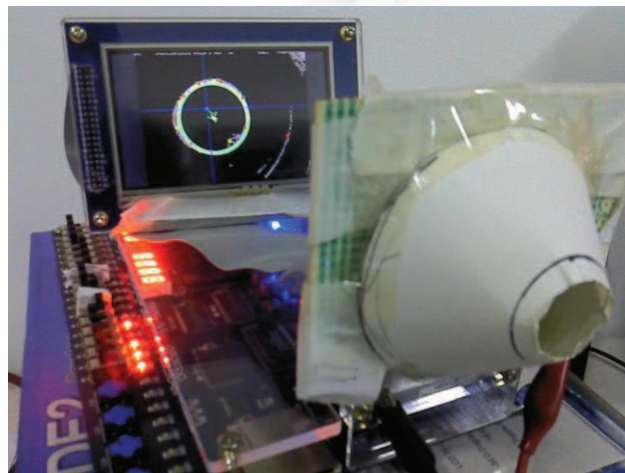
A FPGA contém todos os circuitos necessários ao funcionamento do sistema (listados acima). Ela configura o modo de operação da câmera, regulando aspectos como resolução, quadros por segundo e tempo de exposição, devendo, em seguida, processar as imagens recebidas. A FPGA precisa também monitorar os botões de comando, que podem alterar o tempo de exposição da câmera, selecionar valores de *threshold* do algoritmo (detalhados adiante), parar a recepção de imagens e congelar um quadro no LCD. A FPGA também faz a configuração do LCD, informando que resolução usar e o número de *bits* por *pixel* utilizado. Cabe a ela também fazer o envio da imagem a ser exibida. Por último, a FPGA fará todos os cálculos e processamentos necessários para que o algoritmo funcione de forma adequada.

A FPGA utilizada no protótipo encontra-se em um *kit* de desenvolvimento DE2, também da Terasic (www.terasic.com), o qual pode ser visto na Figura 22(a).

Uma vista do sistema final completo, demonstrado com plena funcionalidade à Banca Examinadora durante a defesa da dissertação, consta na Figura 22(b).



(a)



(b)

Figura 22 – (a) *Kit* de desenvolvimento DE2; (b) Sistema completo (câmera, cone, FPGA, LCD e *kit*) demonstrado à Banca Examinadora.

4.5 ESTRUTURA DO ALGORITMO PROPOSTO

A estrutura do algoritmo proposto é mostrada na Figura 23.

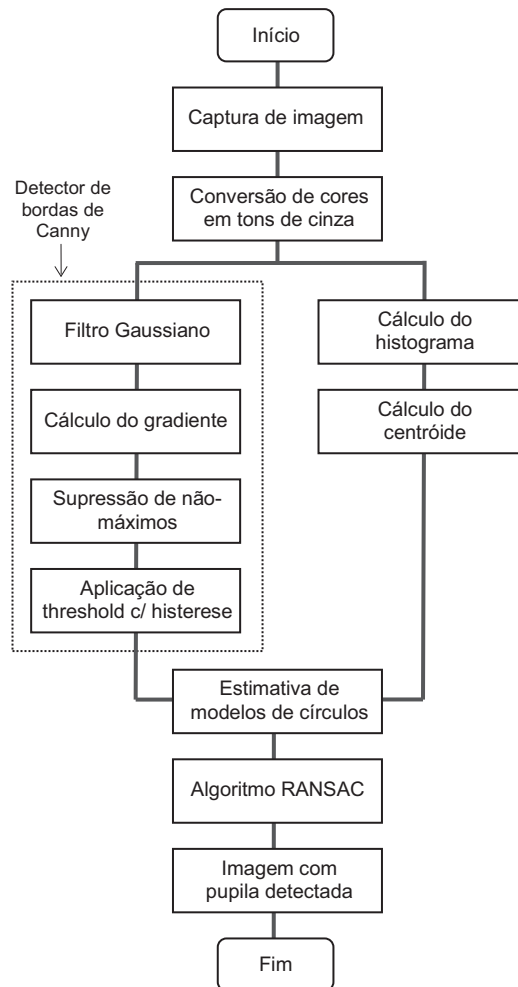


Figura 23 - Estrutura do algoritmo final.

Como pode-se observar, o procedimento começa com a obtenção da imagem original (formato digital RGB de 12 bits), através da câmera CMOS. Em seguida, ocorre a conversão dos *pixels* coloridos para tons de cinza (o que permite reduzir o número de bits a serem processados sem afetar a qualidade dos resultados). Na fase seguinte, duas sequências ocorrem em paralelo: de um lado, o detector de bordas de Canny é aplicado, ao passo que no outro lado ocorrem os cálculos do histograma e do centróide.

O detector de bordas de Canny é composto por quatro etapas: filtro Gaussiano, calculador de gradiente, supressor de não-máximos e, finalmente, operador com *threshold* e histerese. O primeiro passo dentro do algoritmo é suavizar a imagem. Para atingir esse objetivo, a imagem é processada por um filtro Gaussiano, que é passa-baixas, suavizando bordas e detalhes abruptos da mesma. Depois, calcula-se o gradiente ao longo da imagem. Dentro desse bloco, encontra-se um filtro de Prewitt. São realizadas então as outras duas

etapas do algoritmo de Canny, com supressão de não-máximos e aplicação de *threshold* com histerese. O resultado final é uma imagem binária (preto e branco) que contém apenas as bordas da imagem original (um exemplo foi mostrado na Figura 13).

Em paralelo ao bloco do detector de bordas de Canny, é calculado o histograma da imagem. Com ele, obtém-se o limiar da pupila e, com esse valor, pode-se calcular o centróide.

As saídas das duas sequências descritas acima são processadas simultaneamente (por isso a importância do sincronismo e o uso de FIFOs) no próximo bloco e usadas para estimar os modelos de círculos possíveis. Esses modelos são testados pelo método RANSAC no bloco seguinte e o modelo que melhor representar a pupila será considerado como verdadeiro na última etapa do algoritmo proposto.

Nas seções a seguir, detalhes adicionais sobre a implementação deste algoritmo em hardware são apresentados.

4.6 IMAGEM ORIGINAL E CONVERSÃO EM TONS DE CINZA

A imagem digital provém da câmera em um formato RGB de 12 bits por *pixel* e cada dois *pixels* adjacentes são de duas cores distintas (padrão de Bayer, descrito na Seção 2.3). Na Figura 24 é possível visualizar a ordem de leitura dos *pixels* da câmera CMOS utilizada no projeto.

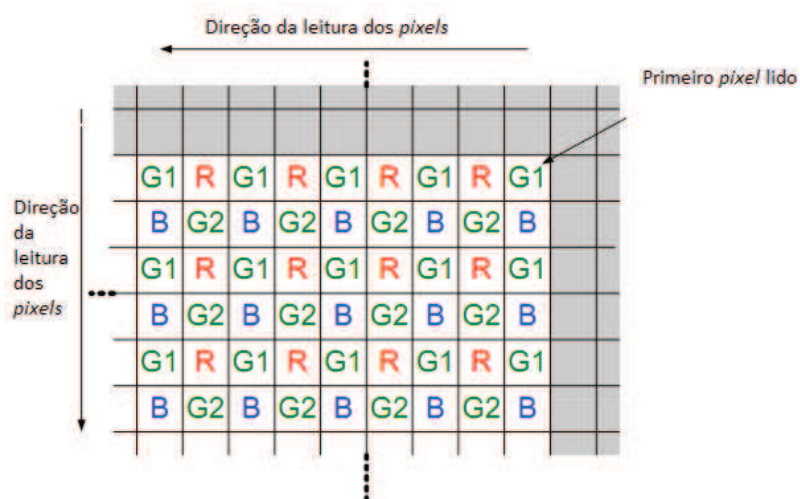


Figura 24 - Saída dos *pixels* da câmera (modificado de www.terasic.com).

Os *pixels* da imagem original contêm informações sobre cores, atributo que pode ser ignorado para o algoritmo proposto. Para não processar dados de forma desnecessária e retardar o processo, trabalha-se com imagens em tons de cinza, cujos *pixels* têm valor de intensidade calculado pela Equação (13).

$$I(x,y) = 0,3 \cdot R(x,y) + 0,59 \cdot G(x,y) + 0,11 \cdot B(x,y) \quad (13)$$

Onde:

- $I(x,y)$ – Intensidade do *pixel* em tom de cinza
- $R(x,y)$ – Intensidade do componente vermelho do *pixel* da imagem original
- $G(x,y)$ – Intensidade do componente verde do *pixel* da imagem original
- $B(x,y)$ – Intensidade do componente azul do *pixel* da imagem original

Conforme mencionado anteriormente, essa composição de valores deve-se ao fato do componente de luz verde ser aquele ao qual o olho humano tem maior sensibilidade.

4.7 PROCESSAMENTO SERIAL E REDUÇÃO DE RECURSOS NECESSÁRIOS

Quando é necessário realizar operações de filtragem com máscaras, como descrito nas Seções 2.9 e 2.10, é preciso guardar os valores a serem processados. Existem dois conjuntos distintos de valores: os valores das intensidades da imagem digital sendo processada e os valores dos coeficientes de cada posição da máscara a ser aplicada. A quantidade de valores a serem guardados depende também do tamanho da máscara a se aplicar.

Em PDI, é comum encontrar algoritmos que recebem uma imagem de uma câmera ou arquivo, armazenam essa imagem por inteiro e depois fazem o devido processamento (recordando que o tamanho de uma imagem é igual a $W \times H \times N \times C$, onde W é a largura da imagem, em *pixels*, H é a altura, igualmente em *pixels*, N é o número de *bits* usados para cada intensidade de cor e C é o número de componentes de cores diferentes usados). No presente projeto, memória é um recurso limitado e dos mais preciosos. Como não deseja-se utilizar memória externa, armazenar um quadro da imagem por inteiro seria inviável.

Para contornar essa limitação de *hardware*, algumas medidas foram tomadas. Primeiramente, apenas filtros com dimensões pequenas foram utilizados, procurando atender às necessidades com o mínimo de processamento e de memória. Por exemplo, o filtro Gaussiano escolhido foi de tamanho 5×5 , ao passo que para o gradiente o filtro escolhido foi o de Prewitt, cujo tamanho é 3×3 . Essas duas máscaras são ilustradas na Figura 25(a).

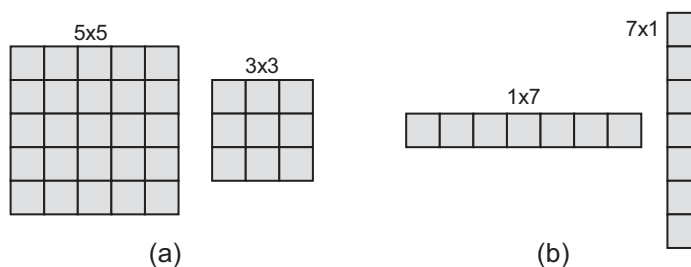


Figura 25 – (a) Máscaras utilizadas no filtro Gaussiano (5×5) e no filtro de Prewitt (3×3);
 (b) Máscaras 1D de tamanho 1×7 e 7×1 utilizadas na decomposição da máscara 2D de tamanho 5×5 do filtro Gaussiano.

Outra medida foi tirar vantagem da simetria radial do filtro Gaussiano (ver Figura 11). Devido a tal simetria, sua máscara pode ser decomposta em dois kernels, cada um de dimensão 1D, ou seja, simples vetores (FISHER, 2003). Desta forma, o filtro utilizado (5×5) pode ser computado com duas máscaras vetoriais, sendo uma 1×7 e a outra, 7×1 . Essas máscaras são mostradas na Figura 25(b).

A medida mais importante para redução dos recursos necessários ao sistema consistiu na utilização de processamento serial, o qual apresenta duas vantagens operacionais fundamentais:

- O processamento dos *pixels* pode ocorrer em paralelo com a leitura de outros *pixels*. Em algoritmos que aguardam a leitura do quadro completo da imagem, o processamento só pode ser feito uma vez que essa leitura seja concluída. No processamento serial, é utilizado o conceito de *pipelining*, onde o resultado final desejado é produzido parcialmente em paralelo com outra atividade, que, nesse caso, é a leitura dos *pixels*. Isso implica em uma redução do tempo de processamento total necessário e conseqüente aumento da taxa de quadros processados por segundo;
- Não existe a necessidade de armazenar quadros inteiros da imagem, pois uma vez que um *pixel* tenha sido processado, ele pode ser descartado, liberando a

memória na qual ele estava guardado, reduzindo assim a quantidade total de memória necessária.

Para que fosse possível utilizar processamento serial, procurou-se uma câmera que escrevesse serialmente em seu barramento os *pixels* da imagem captada. Enquanto a leitura da imagem da câmera é feita, um *pixel* é recebido a cada ciclo de *clock*.

O uso de processamento serial é ilustrado na operação do filtro Gaussiano do projeto, o qual, conforme mencionado, utiliza duas máscaras vetoriais (1×7 e 7×1). O processamento dos *pixels* horizontais (máscara 1×7) é mostrado na Figura 26, contendo os sete valores de *pixels* necessários em um dado momento para poder executar a convolução. Os sete valores são os três valores à esquerda do *pixel* central, o valor do próprio *pixel* central e os outros três valores à direita. Para essa máscara, é necessário guardar sete valores (de 8 *bits* cada, resultando 7 *bytes*) em memória, que se traduz no uso de $7 \times 8 = 56$ registradores (*flip-flops* do tipo D). A Figura 26 ilustra duas etapas consecutivas dessa convolução.

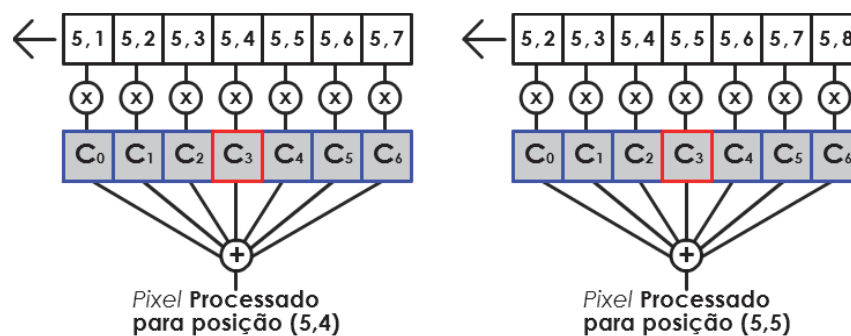


Figura 26 - Processamento serial dos valores de dois pontos consecutivos (horizontais) da imagem.

Já para o processamento vertical (máscara 7×1), é necessário armazenar um número maior de valores. Veja na Figura 27 que agora a máscara utiliza sete posições ainda, porém todas de linhas diferentes (observe que os valores nos registradores na parte superior da figura à esquerda são todos de posições diferentes daqueles nos registradores na parte superior da figura à direita).

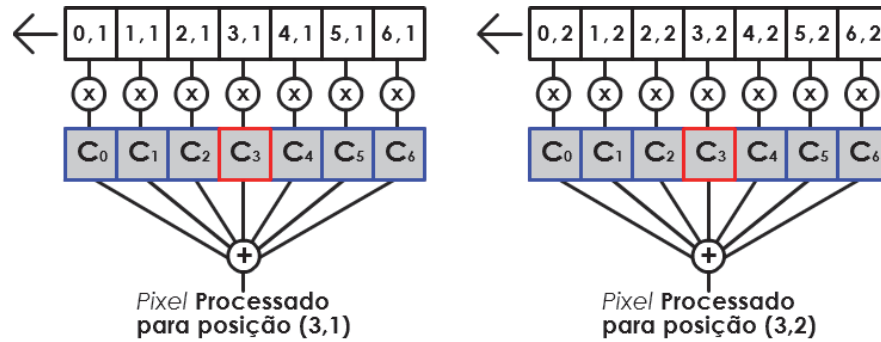


Figura 27 - Processamento serial dos valores de duas etapas consecutivas (verticais) da imagem.

A câmera envia os valores dos *pixels* sempre no sentido horizontal, lendo coluna por coluna, até chegar à última coluna, para então reiniciar na linha abaixo. Para realizar o processamento serial utilizando-se de valores que estão verticalmente abaixo do valor central é necessário então guardar todas as linhas entre a linha do mais recente *pixel* sendo processado até a linha do último *pixel* sendo processado, ou seja, um total de seis linhas mais um *pixel*. Esse armazenamento maior se deve justamente à natureza do problema, que apresenta um filtro vertical convoluindo com uma série de *pixels* que são lidos horizontalmente. Vale comentar que, no caso dessa implementação, a convolução horizontal ocorre antes da convolução vertical, de forma que os *pixels* na saída da operação horizontal são a entrada para a convolução vertical. No final da operação na vertical, tem-se a imagem completamente convoluída com o filtro Gaussiano.

A forma de processamento proposta e descrita acima foi fundamental à viabilização do projeto, pois permitiu minimizar a quantidade de recursos necessários (é importante lembrar que trata-se de PDI implementado inteiramente em *hardware*).

4.8 DETECTOR DE BORDAS DE CANNY

Conforme visto na Figura 23, o algoritmo do detector de bordas de Canny é composto por quatro etapas: filtro Gaussiano, cálculo do gradiente, supressão de não-máximos e aplicação de *threshold* com histerese.

Filtro Gaussiano

Conforme visto na Seção 2.13, o algoritmo proposto por Canny começa com um filtro Gaussiano, cujo objetivo é suavizar a imagem, diminuindo assim as distorções

provocadas principalmente pelo ruído impulsivo. Tal filtragem consiste em substituir cada pixel pela média ponderada de todos os valores sob a máscara do filtro. Tal máscara é usualmente quadrada e com tamanho ímpar, como aquela adotada no presente projeto, cujas dimensões são 5×5 , com média e variância ajustadas de forma empírica, otimizadas para o processo.

Um exemplo de uma imagem processada pelo filtro Gaussiano do projeto é mostrada na Figura 28. Na Figura 28(a), consta a imagem original, ao passo que a Figura 28(b) mostra a mesma imagem após ter sido convolucionada pelo filtro. Repare como a imagem filtrada foi suavizada em relação à original e a intensidade ao longo da imagem está mais uniforme. Na Figura 28(b), o efeito de bordas também pode ser observado.

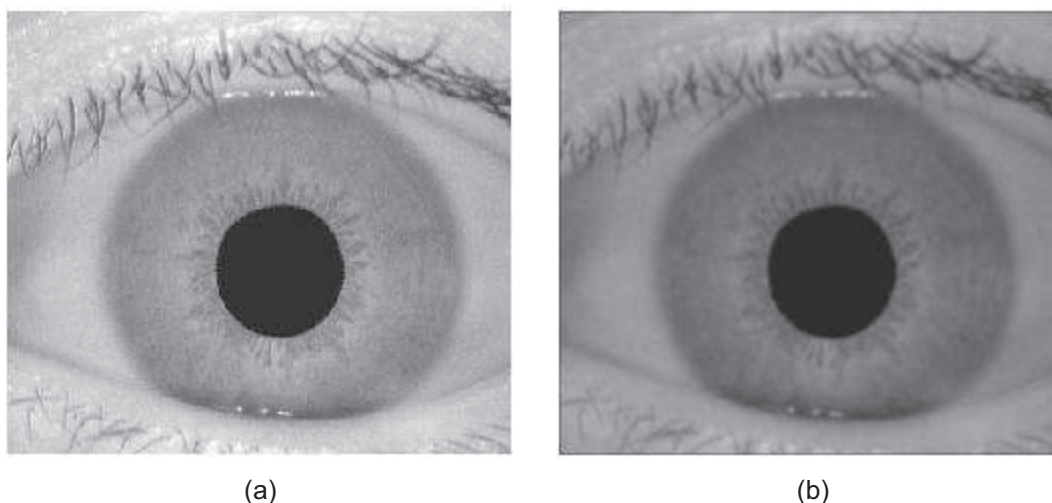


Figura 28 – (a) Imagem monocromática original; (b) Imagem processada pelo filtro Gaussiano 5×5 utilizado no projeto.

Cálculo do Gradiente

Como vimos, o gradiente de uma imagem é a forma direcional de como as intensidades variam dentro da mesma. Se uma parte de uma imagem contem uma superfície lisa e uniforme, o valor absoluto do gradiente será pequeno, pois não existe uma variação de intensidade significativa. Porém, se outro pedaço da imagem contiver uma paisagem ou objeto, variações abruptas de intensidade serão normalmente observadas, o que é típico quando há bordas ou cantos na imagem, resultando um módulo elevado de gradiente. Portanto, o gradiente é um bom indicador da existência de bordas ao longo de um imagem.

Existem alguns operadores que podem ser aplicados para se obter o gradiente de uma imagem. Esses operadores são representados em forma de máscaras e variam em tamanho e em coeficientes. O operador utilizado nesse projeto é o de Prewitt, como mostrado na Figura 12. Outros possíveis operadores são os de Roberts e Sobel (GONZALES, 2007), dentre outros. No entanto, o operador de Prewitt foi escolhido pelo seu tamanho reduzido e simplicidade de uso.

Após a filtragem, o resultado é um mapa de gradientes. Porém, em vez da matriz conter a intensidade de cada *pixel*, tem-se agora o módulo do gradiente para cada posição, com módulos maiores tendendo ao branco e módulos menores tendendo ao preto. Portanto, esse mapa assemelha-se a uma imagem que mostra apenas as bordas dos objetos contidos na imagem original.

Embora a imagem resultante tenha as bordas razoavelmente detectadas, a mesma pode ser refinada através da aplicação de duas etapas adicionais, que consistem na supressão de não-máximos e aplicação de *threshold* com histerese.

Supressão dos Não-Máximos

Nessa fase, são feitas supressões de todos os valores que não são máximos locais relativos ao sentido do gradiente (recordando que, ao calcular o gradiente, obtém-se um módulo e um ângulo). Por exemplo, se uma posição qualquer tem seu ângulo de gradiente próximo de 0 grau, o módulo do gradiente dessa posição precisa ser maior ou igual ao módulo do gradiente da posição da esquerda e da direita dele. Caso contrário, ele será considerado um não-máximo e será suprimido. O teste ocorre para todas as posições e os ângulos são arredondados ao valor mais próximo de um dos quatro valores de ângulo 0, 45, 90 e 135 graus, e então comparados com os valores vizinhos correspondentes.

Após essa etapa, as bordas são afinadas e várias falsas bordas presentes no começo do processo são suprimidas.

Threshold com Histerese

Após a etapa anterior, assume-se que os valores de gradiente que possuem módulo maior são mais prováveis de representar bordas verdadeiras do que os de módulo menor. Contudo, devido a ruído e outros problemas na qualidade da imagem, alguns pontos ainda existentes podem representar falsas bordas. Para aprimorar ainda mais a detecção e tentar processar apenas as bordas verdadeiras da imagem, a quarta fase do algoritmo de Canny

é realizada. Porém, realizar essa filtragem com base em um único valor de limiar (*threshold*), onde todos os valores menores que o limiar são eliminados, produz uma imagem com bordas de pouca qualidade. Canny então optou por fazer essa filtragem usando dois limiares, de forma a haver uma histerese.

Estipulam-se dois limiares, um alto (T_H) e um baixo (T_L), e assume-se que bordas verdadeiras formam curvas contínuas ao longo da imagem. Todos os pontos com gradiente acima de T_H são automaticamente considerados bordas verdadeiras. Igualmente, todo ponto que estiver conectado (i.e., que seja um dos oito *pixels* que tangenciam esse *pixel*) a um ponto que tenha módulo de gradiente acima de T_H e módulo próprio acima de T_L também é considerado verdadeiro. Finalmente, qualquer ponto com gradiente abaixo de T_L é marcado como falso e tem seu valor zerado.

Ao final dessa etapa, o algoritmo de detecção de bordas de Canny estará completo, resultando um mapa fiel das bordas dos objetos presentes na imagem original (conforme ilustrado na Figura 13).

4.9 HISTOGRAMA

Através do histograma de uma imagem digital do olho, algumas informações importantes podem ser retiradas, úteis para o processamento da imagem. Em imagens de boa qualidade (iluminação adequada e alto contraste), o histograma da imagem do olho tem um padrão regular, com dois máximos locais e um mínimo local no lado inferior (ROVANI, 2006). Um exemplo disso pode ser visto na Figura 29. A porção inicial do histograma (até o primeiro mínimo local) representa os *pixels* constituintes da pupila, uma vez que ela tende a ser a parte mais escura da imagem.

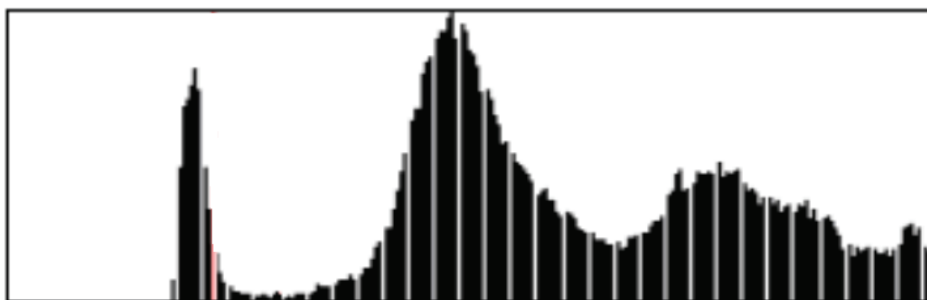


Figura 29 - Histograma típico de uma imagem digital do olho humano (modificado de ROVANI, 2006).

Contudo, as imagens do olho podem estar desfocadas ou com baixo contraste. Isso pode ocorrer devido principalmente a condições extremas e não favoráveis de iluminação ou mau posicionamento do paciente diante da câmera. O que se produz então é um histograma que tem uma distribuição similar ao de uma imagem nítida e de bom contraste mas com os máximos e mínimos pouco aparentes ou mesmo com o mínimo local inexistente. Na Figura 30, é possível visualizar o histograma de uma imagem com má iluminação.

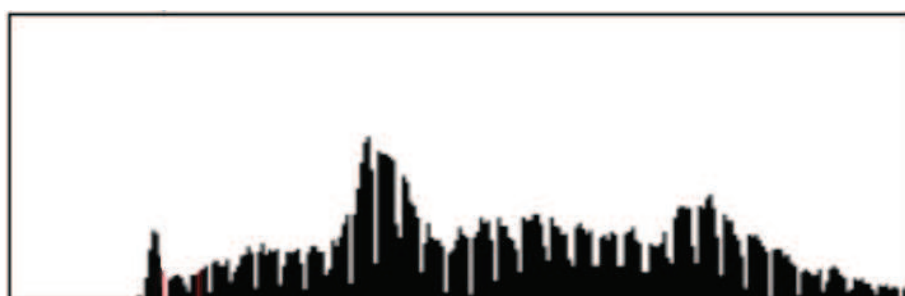


Figura 30 - Histograma de uma imagem do olho fora do padrão (modificado de ROVANI, 2006).

Para contornar esse problema, no algoritmo proposto utiliza-se um limiar variável, cujo valor equivale ao mínimo local encontrado entre os dois máximos locais previamente citados. Dessa forma, considera-se que a pupila é composta por todos os *pixels* com valor igual ou inferior ao limiar selecionado.

Um histograma é alocado no *hardware* como um vetor de 256 posições (dado que cada *pixel* é composto por 8 bits) e cada posição tem 12 bits (podendo contabilizar até 4096 ocorrências de uma determinada intensidade). Como o histograma de uma imagem só está completo quando todos os *pixels* dessa imagem são recebidos e contabilizados, o histograma só estará corretamente calculado e pronto no início do quadro seguinte de imagem.

Existe uma série de maneiras distintas de se obter os máximos e mínimos locais de um determinado vetor de entrada. Nesse projeto, optou-se por utilizar uma máquina de estados para tal objetivo (ROVANI, 2006), mostrada na Figura 31.

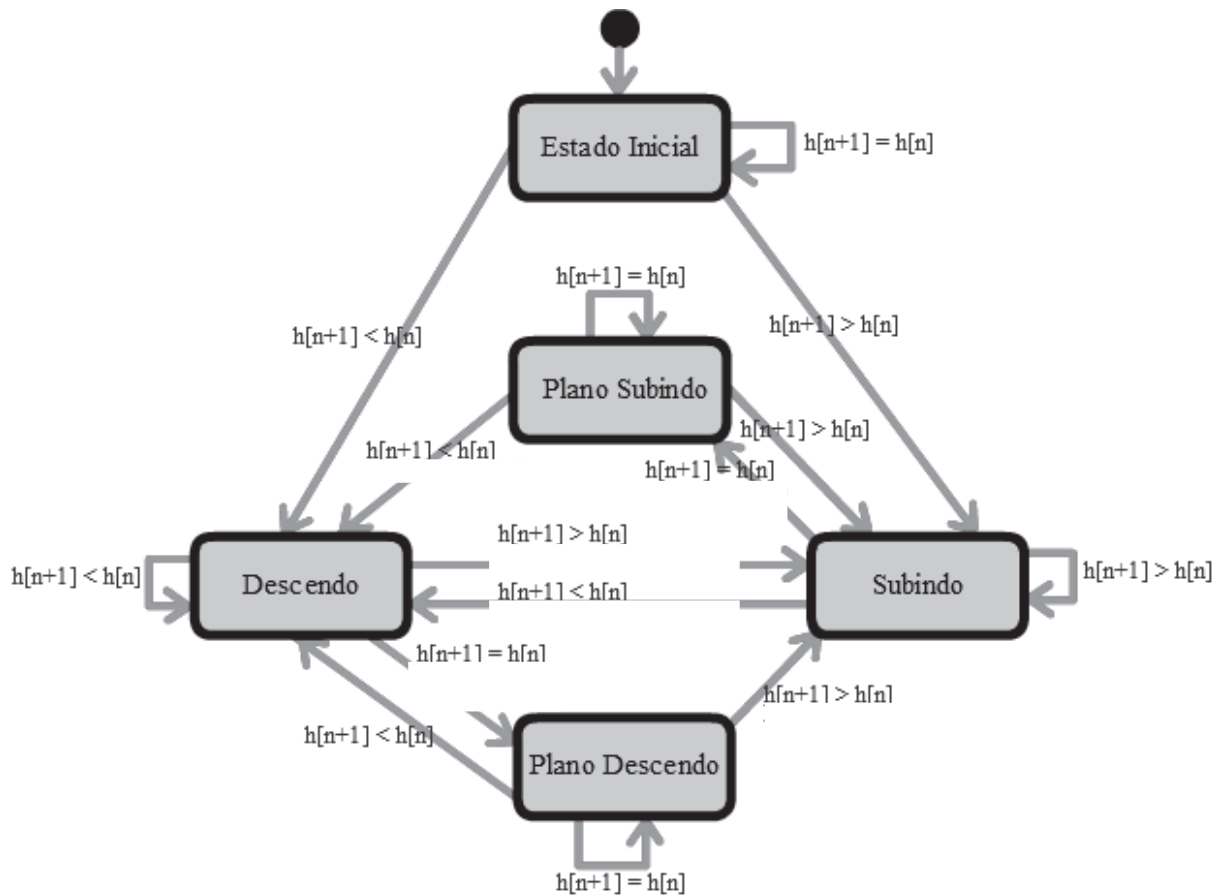


Figura 31 - Máquina de estados para o histograma (modificado de ROVANI, 2006).

A máquina de estados (Figura 31) começa de um estado inicial e lê uma posição do histograma (i.e., uma intensidade de *pixel*) de cada vez. O valor na posição atual é então comparado ao valor da próxima posição, com base no que os valores são guardados em vetores correspondentes, conforme indicado no diagrama de transições de estados. Em resumo, os quatro estados da máquina operam da seguinte maneira:

- Subindo: A máquina vai para esse estado sempre que o valor da próxima posição é maior que o da atual;
- Descendo: A máquina vai para esse estado sempre que o valor da próxima posição é menor que o da atual;
- Plano Subindo: A máquina vai para esse estado sempre que o valor da próxima posição é igual ao valor da atual posição e o estado prévio era "Subindo";
- Plano Descendo: A máquina vai para esse estado sempre que o valor da próxima posição é igual ao valor da atual posição e o estado prévio era "Descendo".

Em suma, a imagem digital do olho possui um histograma padrão, com dois máximos locais e um mínimo local presentes na parte inferior. Uma vez obtidas as listas de máximos e mínimos, é calculado o limiar da pupila, que será o menor mínimo local existente entre o primeiro e o segundo máximos. Consideram-se então todos os *pixels* com intensidade menor ou igual a esse limiar como sendo constituintes da pupila. Se a imagem for binarizada com esse valor, ou seja, se todos os *pixels* com valor menor ou igual ao limiar aparecerem pretos na imagem e todos acima aparecem brancos, a imagem mostrará apenas a pupila. Quando trabalhando com imagens de menor qualidade e contraste, alguns outros elementos podem aparecer na imagem, como partes da sobrancelha e regiões mais escuras.

4.10 CENTRÓIDE E ESTIMATIVA DO CENTRO

A análise do histograma determina um valor de limiar para a pupila. Com esse valor, estipulam-se quais *pixels* compõe a pupila. Em uma imagem digital, todo *pixel* tem uma posição definida por uma coordenada x e uma coordenada y (Figura 32); portanto, é possível pensar na imagem como um sistema de coordenadas discretas x e y e os menores elementos endereçáveis dessa imagem são os *pixels*.

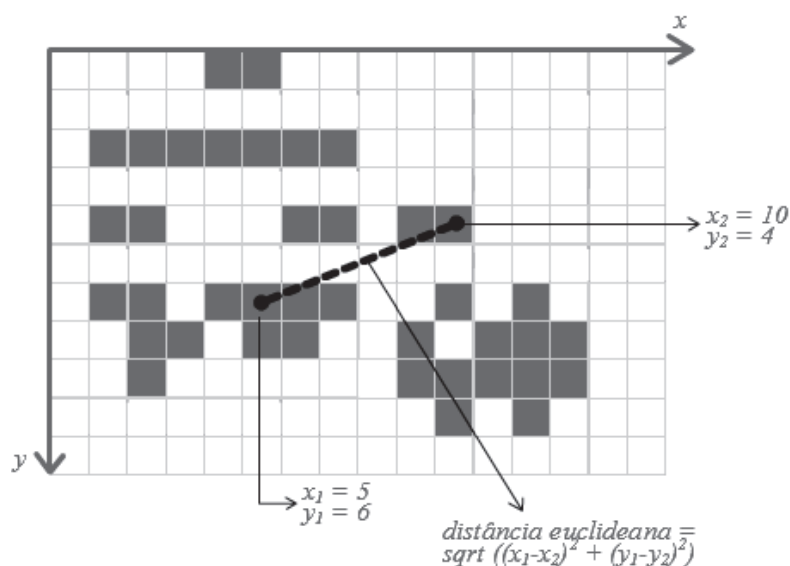


Figura 32 - Sistema de coordenadas de *pixels*.

Como qualquer objeto que é representado em um plano de duas dimensões, pode-se calcular seu centroide, que é o ponto que define seu centro geométrico (que em alguns

casos pode coincidir com o centro de massa e o centro de gravidade). As coordenadas do centroide podem ser calculadas pelas Equações (14) e (15).

$$c_x = \frac{\sum_{j=1}^W \sum_{k=1}^H j \cdot I(j, k)}{\sum_{j=1}^W \sum_{k=1}^H I(j, k)} \quad (14)$$

$$c_y = \frac{\sum_{j=1}^W \sum_{k=1}^H k \cdot I(j, k)}{\sum_{j=1}^W \sum_{k=1}^H I(j, k)} \quad (15)$$

Onde:

- c_x – Coordenada x do centróide
- c_y – Coordenada y do centróide
- j – Coordenada x do *pixel* selecionado naquela iteração
- k – Coordenada y do *pixel* selecionado naquela iteração
- $I(j, k)$ – Valor do *pixel* para a posição (j, k)

Dado que a limiarização utilizada na etapa anterior visa destacar a pupila na imagem e que a pupila, para a maioria das necessidades, pode ter seu formato aproximado a um círculo, o centróide dessa imagem resultará aproximadamente no centro da pupila, com coordenadas (c_x, c_y) . Portanto, ao final dessa etapa, tem-se uma estimativa inicial do centro da pupila. A localização desse centro é essencial para a etapa seguinte do algoritmo.

4.11 OBTENÇÃO DAS AMOSTRAS ATRAVÉS DO MÉTODO “EXPLOSÃO RADIAL”

Diversos métodos de detecção de círculos existem na bibliografia. Cada processo tem suas razões para ser usado. Alguns apresentam tempo de processamento menor, outros visam diminuir a complexidade do sistema, já outros optam por precisão elevada. Mas é comum que, em cada vantagem que um algoritmo apresenta, ele possua uma desvantagem consequente. Isso pode ser dado por uma melhora no tempo de processamento, compensado com um aumento do *hardware* necessário (e consequente aumento de custo) ou uma melhora na precisão do algoritmo, com aumento do tempo de processamento.

No caso dessa dissertação, foi visado usar um *hardware* relativamente simples e com restrições de custo, tendo também em mente a portabilidade e outras características funcionais do equipamento. Uma série de algoritmos diferentes foi testada, tendo-se finalmente chegado a um algoritmo que balanceasse os critérios fundamentais do projeto, como quantidade de *hardware*, portabilidade, tempo de processamento e custo. O algoritmo final escolhido é uma mescla de alguns dos processos descritos anteriormente.

Na seção anterior, chegou-se a uma estimativa para o centro do círculo com base no histograma da imagem. Esta, porém, é apenas uma estimativa simples inicial, que deve receber etapas adicionais a fim de apresentar maior precisão. Tais etapas, desenvolvidas nessa dissertação, são descritas abaixo.

Embora a estimativa inicial para o centro do círculo obtida na seção anterior seja ainda preliminar, o referido ponto situa-se necessariamente dentro do perímetro verdadeiro da pupila. Um exemplo é apresentado na Figura 33, que mostra o ponto onde o centro foi inicialmente estimado para uma dada imagem de entrada. Com esse ponto do centro obtido, o algoritmo se baseia em alguns fundamentos do método *starburst* (introduzido na Seção 3.2). Essa etapa do algoritmo foi criada no desenvolvimento dessa dissertação e foi denominada *explosão radial* (Figura 34). Primeiramente, determinam-se todas as linhas que atravessam o ponto central e que estão separadas de 15° uma da outra, iniciando-se com a linha que o atravessa a um ângulo de 0° (secciona a imagem horizontalmente). O valor de 15° foi determinado de modo empírico, baseado no número suficiente de retas geradas. Com essa diferença angular entre as retas, ter-se-á um total de 12 retas ($360^\circ \div 15^\circ = 24$ semirretas) passando pelo ponto central estimado.

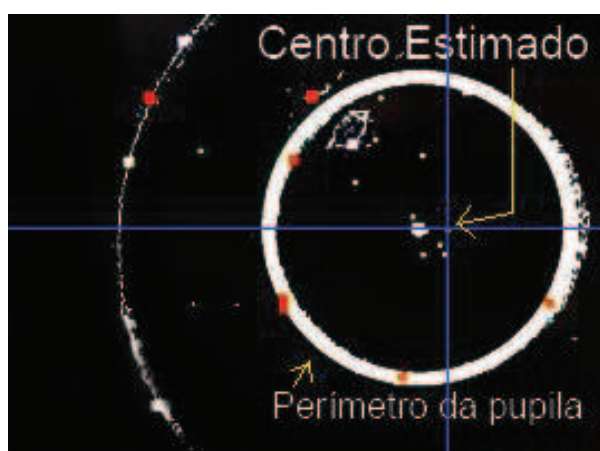


Figura 33 - Estimativa do centro da pupila.

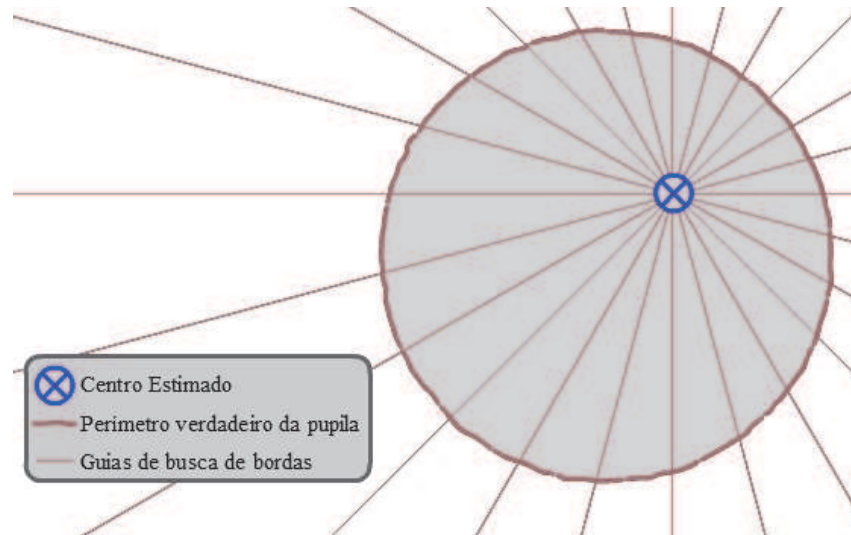


Figura 34 – Explosão Radial: Ponto central e as respectivas guias (semirretas).

A determinação dessas retas pode ser feita como segue. Uma reta que passa por um ponto (com coordenadas x_c e y_c , as quais são as coordenadas do centro estimado) e tem ângulo α com o eixo x , pode ser expressa através da Equação (16).

$$y = y_c + \operatorname{tg}\alpha \cdot (x - x_c) \quad (16)$$

Utilizando essa equação, obtém-se as fórmulas de todas as retas utilizadas no algoritmo, as quais foram listadas na Tabela 1.

Vale notar que, como todas as retas são concorrentes no ponto central, é impossível serem concorrentes em outro ponto, o que garante que não haja um ponto de borda que pertença a duas retas distintas.

Tabela 1 - Fórmulas matemáticas das guias utilizadas na explosão radial.

Número da Reta	Ângulo da Reta (°) Menor / Maior	Fórmula
0	0 / 180	$y = y_c$
1	15 / 195	$y = y_c + \tan(15^\circ) \cdot (x_c - x)$
2	30 / 210	$y = y_c + \tan(30^\circ) \cdot (x_c - x)$
3	45 / 225	$y = (y_c + x_c) - x$
4	60 / 240	$y = y_c + \tan(60^\circ) \cdot (x_c - x)$
5	75 / 255	$y = y_c + \tan(75^\circ) \cdot (x_c - x)$
6	90 / 270	$x = x_c$
7	105 / 285	$y = y_c - \tan(75^\circ) \cdot (x_c - x)$
8	120 / 300	$y = y_c - \tan(60^\circ) \cdot (x_c - x)$
9	135 / 315	$y = (y_c - x_c) + x$
10	150 / 330	$y = y_c - \tan(30^\circ) \cdot (x_c - x)$
11	165 / 345	$y = y_c - \tan(15^\circ) \cdot (x_c - x)$

Com as retas determinadas, utilizam-se em conjunto as bordas detectadas pelo detector de bordas de Canny. Ressalta-se aqui a importância de se usar um algoritmo de detecção de bordas bastante confiável. Nessa etapa, o algoritmo considera todas as retas calculadas e avalia os pontos onde elas passam. Se um dos pontos detectados como borda pelo algoritmo de Canny estiver sobre uma dessas retas e o mesmo ponto for o mais próximo do ponto central estimado, esse ponto será considerado como um ponto de teste da pupila. No total, poderá haver até 24 pontos de teste. Todas as retas se cruzam no ponto de centro estimado. Para cada reta calculada, têm-se duas extremidades. Por exemplo, na reta $0^\circ - 180^\circ$ haverá uma semirreta para a direita do ponto e uma semirreta para a esquerda do ponto. Consequentemente, haverá dois pontos de teste sobre a reta $0^\circ - 180^\circ$, assim como em todas as outras retas. Como a equação da reta é a mesma para as duas semirretas, é preciso determinar em qual das duas extremidades está localizado o ponto de teste detectado. Para isso, basta averiguar em que lado do ponto central se encontra o ponto, isso é, se as coordenadas x ou y são maiores ou menores que a coordenada central (x_c, y_c). A Figura 35 mostra um exemplo do processo da explosão radial para obter amostras nas bordas coincidentes com as retas.

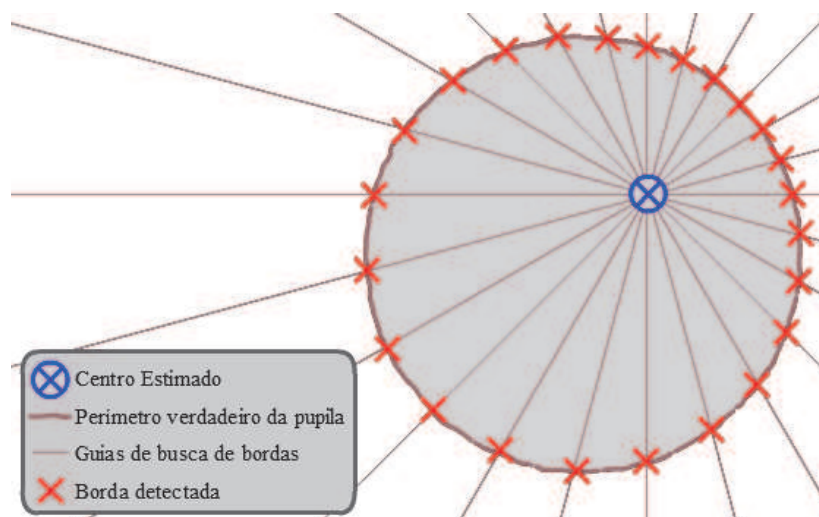


Figura 35 – Explosão Radial: Intersecção das guias com as bordas detectadas.

Após a explosão radial, resultarão 24 amostras de pontos de borda, a partir dos quais serão gerados os círculos candidatos a pupila.

4.12 ESTIMATIVA DE MODELOS DE CÍRCULO

Da geometria, sabe-se que três pontos não colineares determinam um círculo único. Portanto, se todos os pontos de teste estiverem sobre a borda verdadeira da pupila, basta utilizar qualquer grupo de três pontos para determinar o centro e raio verdadeiros da mesma. Idealmente, isso deveria ocorrer, porém nem sempre uma semirreta irá encontrar um ponto de borda correto. Isso pode ocorrer por diversos motivos, dentre eles, como iluminação insuficiente ou excessiva, ruído na imagem, ou bordas falhadas (“buracos” nas bordas verdadeiras).

Nesses casos, a semirreta poderá marcar um ponto de teste errado, cujas coordenadas não estão sobre o perímetro da pupila, introduzindo algumas amostras erradas no grupo total de amostras. Por isso, é inviável trabalhar com todas as amostras de forma igual, dado que algumas provavelmente estarão erradas. Caso essa consideração não seja feita, será gerado um modelo igualmente errado.

Dado que existe a possibilidade de se ter pontos de teste que estão fora do modelo a ser detectado (i.e., círculo), é necessário utilizar um algoritmo que trabalhe com amostras corretas e incorretas de forma robusta, capaz de discernir um modelo correto de um

modelo incorreto e, de alguma forma, beneficiar a apuração do modelo que melhor se encaixar com as amostras corretas.

Considerando essa necessidade, o algoritmo RANSAC (descrito na Seção 2.14) se apresenta como uma alternativa adequada. O princípio do RANSAC é de trabalhar com amostras que se encaixam ao modelo desejado (os chamados *inliers*) e com amostras que não se encaixam ao modelo (*outliers*). É um algoritmo não determinístico, ou seja, ele pode processar uma entrada qualquer de várias maneiras, sem uma especificação prévia de que maneira isso ocorrerá. O método assume que se procura um modelo específico, nesse caso um círculo, e escolhe como resultado aquele grupo de amostras que melhor se ajusta ao modelo desejado.

No algoritmo desenvolvido, existem 24 amostras, obtidas através das 12 retas (ou 24 semirretas) geradas na etapa anterior. Com 24 pontos de amostra, formando grupos de três pontos distintos e sabendo que nenhum dos grupos é colinear, o número total de círculos possíveis a serem gerados é:

$$C_{24}^3 = \frac{24!}{(24-3)!3!} = \frac{24!}{21!3!} = \frac{24 \cdot 23 \cdot 22}{6} = 2024 \text{ círculos} \quad (17)$$

Com essa quantidade de amostras, é possível gerar 2024 círculos candidatos. Pode-se utilizar todos os 2024 círculos (grupos de três pontos) como entradas de teste para o algoritmo RANSAC, mas isso não é feito, pois, além de retardar o algoritmo, afetando o desempenho do sistema e o tempo total de mensuração, nem todo grupo de pontos representa um conjunto de amostras satisfatórias.

Suponhamos inicialmente que um grupo de três pontos vizinhos (pontos vizinhos são aqueles que estão sobre retas suporte separadas de 15°) sejam utilizados, como exemplificado na Figura 36. Se as coordenadas dessas amostras (pontos) estiverem todas corretas, resultará o círculo correto para a pupila (digamos que seja esse o caso na Figura 36). Contudo, para pontos vizinhos, basta que uma das amostras esteja ligeiramente incorreta (ponto central na Figura 37, por exemplo) para que um círculo totalmente diferente resulte (compare os dois círculos na Figura 37).

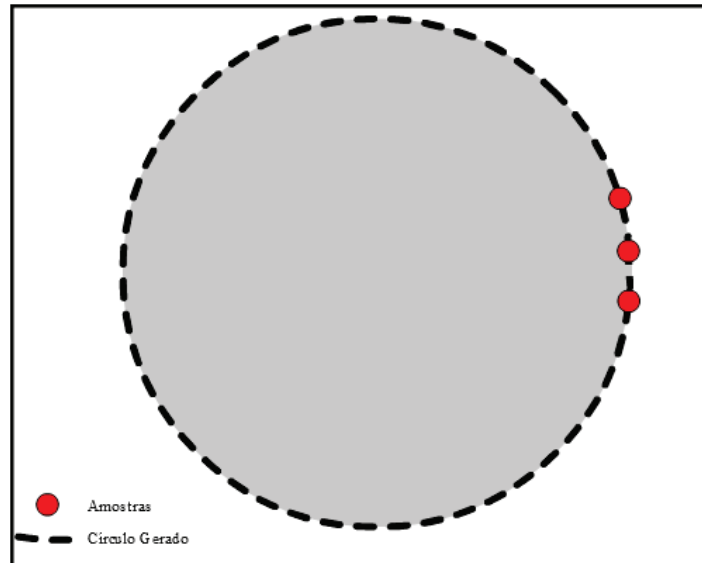


Figura 36 - Círculo gerado por grupo de pontos vizinhos, com as três amostras corretas.

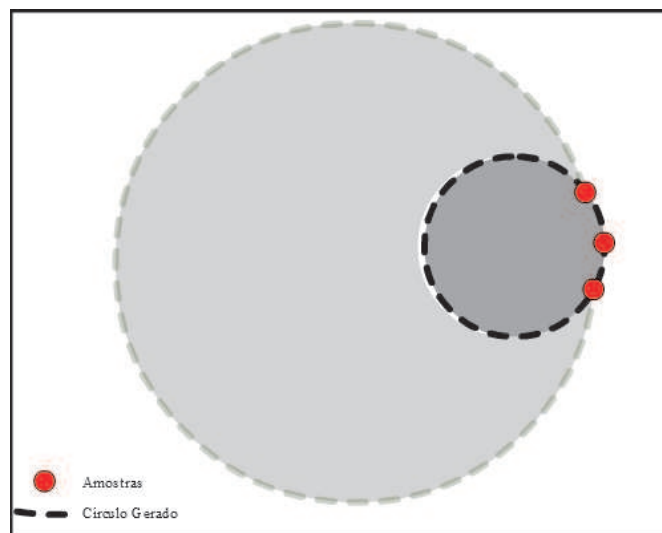


Figura 37 - Outro círculo gerado por grupo de pontos vizinhos, porém com a amostra central ligeiramente incorreta (um pouco mais distante do centro).

Suponhamos agora que, ao invés de pontos vizinhos, pontos equidistantes sejam empregados, conforme ilustra a Figura 38. Se as três amostras forem corretas, resultará novamente o círculo correto para a pupila (digamos que seja esse o caso na Figura 38). Contudo, se uma das amostras for ligeiramente incorreta, como na Figura 39, o resultado não será catastróficamente incorreto, como ocorreu quando pontos vizinhos foram utilizados (Figura 37).

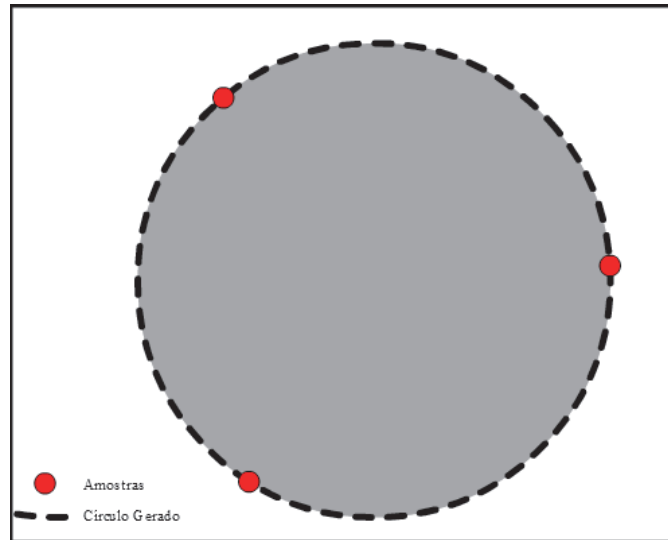


Figura 38 - Círculo gerado por grupo de pontos equidistantes, com as três amostras corretas.

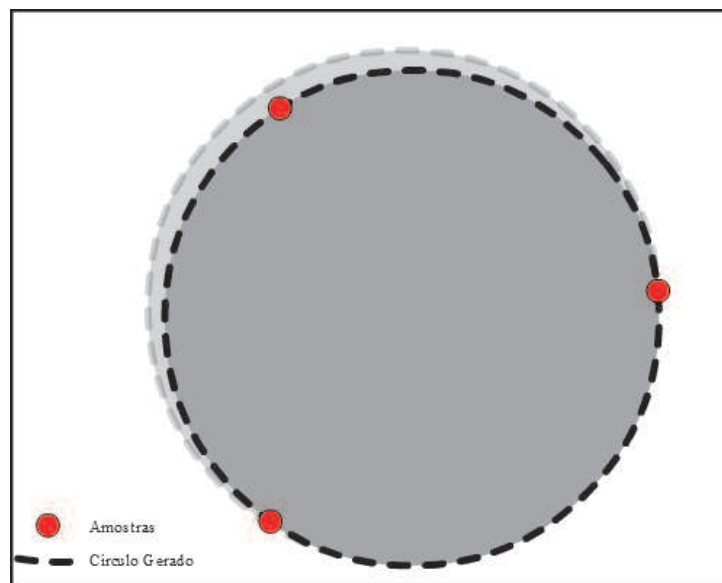


Figura 39 - Outro círculo gerado por grupo de pontos equidistantes, porém com uma das amostras ligeiramente incorreta.

Pela razão exposta acima, pontos equidistantes (sobre retas suporte separadas de 120°) foram empregados nos testes. Para o conjunto de 24 pontos existentes, separados por passos de 15° , existem 8 grupos que atendem a esse quesito ($0^\circ/120^\circ/240^\circ$, $15^\circ/135^\circ/255^\circ$, etc.). Todos esses oito grupos são utilizados no algoritmo proposto, além de outros 24 círculos descritos abaixo.

O uso desses oito círculos reduz o nível de erros no algoritmo. Porém, para dar ao mesmo ainda mais robustez, outros 24 grupos de três pontos (portanto, outros 24 círculos)

são também utilizados, totalizando assim 32 círculos de teste no algoritmo final. Tais grupos são formados pelos pontos separados por 90° , isto é, $0^\circ/90^\circ/180^\circ$, $15^\circ/105^\circ/195^\circ$, $30^\circ/120^\circ/210^\circ$, etc. A razão para escolha desses pontos é que a distância entre eles ainda é elevada o suficiente para prover boas estimativas da pupila.

Seria possível ainda incluir outros conjuntos de pontos para serem testados, mas testes práticos mostraram que, mesmo aumentando o número de grupos utilizados, na maioria dos casos o círculo escolhido como melhor candidato era um desses 32 modelos gerados.

Na Tabela 2 constam os resultados de uma série de testes práticos. Foram realizados cinco testes diferentes, utilizando um número crescente de pontos. As experiências foram realizadas sob condições similares (mesma configuração do equipamento, mesma iluminação, etc.), porém com uma imagem intencionalmente ligeiramente diferente (o paciente se moveu um pouco em relação à câmera). Cada grupo foi avaliado 30 vezes aleatoriamente, e o grupo que foi escolhido em cada observação foi anotado (consta na última coluna da tabela).

Tabela 2 – Testes práticos para determinação dos grupos de maior ocorrência.

Número de Pontos Testados	Número de Vezes Testado	Tripla Mais Escolhida
32	30	$0^\circ/120^\circ/240^\circ$ (70%)
50	30	$15^\circ/135^\circ/255^\circ$ (70%)
64	30	$15^\circ/135^\circ/255^\circ$ (68%)
100	30	$0^\circ/90^\circ/180^\circ$ (65%)
128	30	$30^\circ/150^\circ/270^\circ$ (60%)

É importante observar que, em cada teste da Tabela 2, o círculo obtido com mais frequência foi sempre um daqueles que já constavam no grupo de 32 círculos descritos anteriormente (observe que, nos três primeiros casos, círculos determinados por pontos equidistantes foram escolhidos, ao passo que, nos últimos dois casos, círculos provenientes de pontos a 90° foram selecionados), descartando, para efeitos práticos, a necessidade de trabalhar com um número maior de pontos. Dessa forma, pode-se reduzir o processamento e memória interna total usada no algoritmo.

Esse resultado é muito bem-vindo, uma vez que conseguiu-se que a complexidade do processamento ficasse dentro do limite aceitável para o nível dos componentes

previstos para uso no projeto, o mesmo ocorrendo com a quantidade de memória necessária.

4.13 ESCOLHA DO MELHOR CÍRCULO CANDIDATO USANDO RANSAC

Após a geração de todos os círculos candidatos, é preciso escolher entre eles qual melhor representa a pupila. O mesmo é feito com a ajuda do algoritmo RANSAC (Seção 2.14), adotando-se o círculo gerado que contem sobre o seu perímetro o maior número de amostras como vencedor.

Esse resultado é útil, pois, na fase de geração dos círculos, tomou-se em conta fatores como centro e raio apenas. Agora, no entanto, inclui-se um fator que avalia o quanto o modelo atual se encaixa com as outras amostras presentes. Dessa forma, embora possam existir amostras errôneas, o modelo escolhido será aquele que melhor se ajustar ao conjunto de todas as amostras. Um exemplo da aplicação deste critério pode ser visto na Figura 40.

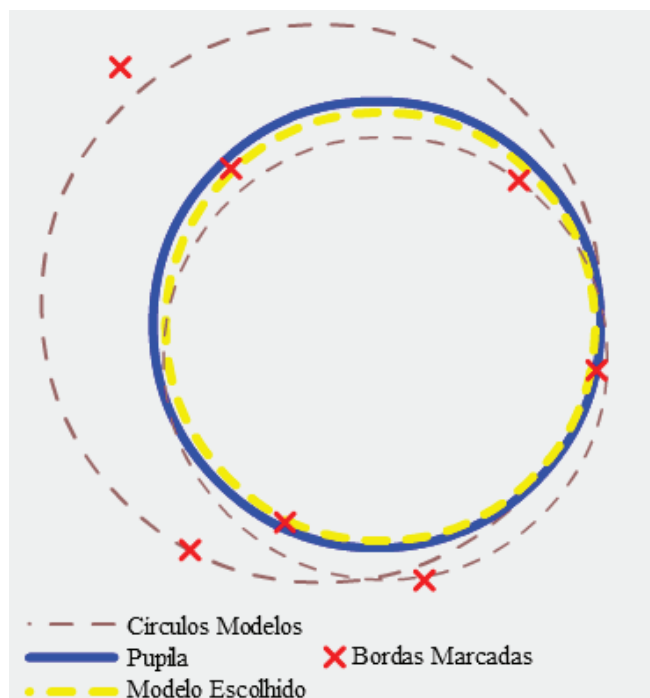


Figura 40 - Exemplo da escolha do melhor círculo como representante da pupila.

4.14 RESUMO DO MAPEAMENTO DO ALGORITMO NO HARDWARE

Em síntese, o mapeamento do algoritmo desenvolvido para *hardware* acontece de acordo com os seguintes passos (ver Figura 23):

- A imagem original é obtida pela câmera CMOS. A câmera possui uma série de parâmetros de configuração, que são ajustados apenas uma vez no início do algoritmo. A leitura dos *pixels* que saem da câmera é feita serialmente, um *pixel* por ciclo de *clock*. Cada *pixel* tem 12 *bits* e representa a intensidade de uma cor primária (vermelho, verde ou azul);
- Em seguida, esses *pixels* coloridos são convertidos em *pixels* em tons de cinza. Esse bloco recebe um conjunto de 3 componentes (RGB) e pondera os valores de cada intensidade, produzindo então um *pixel* monocromático. Esse bloco tem sua saída ligada ao detector de bordas de Canny e ao calculador do histograma;
- O detector de bordas de Canny recebe esses novos *pixels* um por vez. Como as máscaras de filtragem envolvidas podem ser decompostas em vetores unidimensionais, a filtragem ocorre serialmente, reduzindo assim o consumo de memória e acelerando o processo de filtragem;
- O bloco de cálculo do histograma também recebe os *pixels* monocromáticos serialmente, produzindo um histograma completo a cada quadro inteiro de *pixels* recebido (recorde que o histograma só está completo quando todos os *pixels* de um determinado quadro são contados). Quando o histograma é completado, esse bloco fornece os valores de limiar necessários para o bloco da estimativa do centro;
- O bloco do estimador do centro recebe o limiar do histograma e contabiliza todos os *pixels* com intensidade menor que o limiar recebido. Dado o número de *pixels* contados e a posição de cada um, estima-se então um centroide para esses *pixels*, que será considerado o centro estimado para o bloco da estimativa de círculos e o início da “explosão radial”;
- O bloco estimador dos modelos de círculo recebe os *pixels* vindos tanto do bloco da estimativa do centro quanto do detector de bordas de Canny. Como as saídas desses dois blocos não estão sincronizadas, é necessário utilizar uma FIFO para atrasar os *pixels* vindos do detector de Canny. Uma vez feito esse sincronismo, começa-se a leitura de cada bloco para estimar os modelos de

círculos a serem usados no bloco do RANSAC. Com o centro estimado, ocorre a “explosão radial”, onde 24 semirretas, separadas por 15 graus, emergem desse centro e cruzam os *pixels* da imagem (esses *pixels* sobre as semirretas são aqueles produzidos pelo detector de bordas de Canny). Em *hardware*, a “explosão radial” ocorre observando a posição de cada *pixel* que entra serialmente. Como as equações de cada semirreta são sabidas, verifica-se a cada *pixel* entrante se sua posição está ou não sobre uma das semirretas e se o valor desse *pixel* está zerado ou não (i.e., se é uma borda ou não). Com isso feito, esses pontos são considerados como pontos de teste, servindo de base para a geração dos círculos modelos. Com os 24 pontos de teste, são gerados 32 círculos modelos, os quais são subseqüentemente testados no bloco do RANSAC;

- No último bloco, o RANSAC é realizado. Existem 32 círculos modelos, portanto com 32 centros e 32 raios, possivelmente distintos. Além disso, têm-se guardadas em memória as 24 amostras que geraram os círculos candidatos. Então o RANSAC se inicia. Realiza-se a comparação de cada círculo modelo com as 24 amostras. Se uma amostra se encontra sobre a circunferência desse círculo (ou distanciado com um pequeno erro de tolerância), a amostra é considerada como parte desse círculo e incrementa-se a contagem de amostras que atendem corretamente a esse modelo. Essa etapa se realiza 32 vezes, para todos os círculos, e então, dado o círculo que contem o maior número de amostras satisfatórias, ter-se-á a pupila;
- Esse processo se repete aproximadamente 22 a 24 vezes por segundo, a cada vez recalculando a posição e o tamanho da pupila, concluindo assim o algoritmo.

Com o fim do algoritmo, tem-se uma imagem fiel da pupila do indivíduo sob teste. Outros detalhes sobre os resultados obtidos, tanto com relação ao projeto quanto às especificações do algoritmo, serão apresentados no Capítulo 5.

CAPÍTULO 5

RESULTADOS

5.1 INTRODUÇÃO

Conforme mencionado, o presente estudo teve como objetivo desenvolver um sistema simples, compacto e portátil, baseado puramente em *hardware* (FPGA), codificado este em VHDL e/ou Verilog, utilizando a menor quantidade possível de recursos (especialmente memória), para detecção da pupila de um indivíduo posicionado diante de uma câmera, em tempo real. A operação do sistema ocorre de forma autônoma, com seus circuitos completamente embarcados em uma única FPGA, dispensando assim o uso de um computador ou qualquer outro equipamento ou software externo para o seu funcionamento, ao contrário do que ocorre usualmente nesse tipo de sistema.

Os principais resultados e outros aspectos desse estudo são resumidos nas seções a seguir.

5.2 DEFINIÇÕES GERAIS DO PROJETO

Uma das primeiras etapas do projeto foi a elaboração de seus requisitos gerais. Algumas escolhas e estratégias de projeto foram determinadas já nesta fase, enquanto outras foram feitas durante os desenvolvimentos. O uso de uma FPGA e o desenvolvimento baseado inteiramente em *hardware* foi uma das características determinadas no começo. Uma das motivações para tal estava no estudo da viabilidade de utilizar-se *hardware* puro na área de processamento digital de imagens. Essa escolha também se deveu ao fato de sistemas puramente em *hardware* geralmente terem uma taxa de processamento (em número de ciclos de *clock*) mais eficiente do que soluções análogas em *software*, de forma que a opção escolhida poderia viabilizar o uso de um *clock* mais lento (acarretando assim também outras vantagens, como menor consumo de energia e menor custo).

Como consequência (vantagem) de utilizar-se uma FPGA, foi possível descrever o projeto inteiro nas linguagens de descrição de hardware VHDL e Verilog (mais detalhes na seção 5.5). O uso de ambas (VHDL e Verilog são as únicas HDLs padronizadas pelo IEEE e suportadas pelos compiladores) deveu-se simplesmente a motivos educacionais, pois já

tinha amplo conhecimento de VHDL, uma vez que este é ensinado na UTFPR (é a linguagem apresentada na maioria das universidades), porém desejava aprender e praticar também com Verilog.

Diversas definições só foram possíveis durante o desenvolvimento do projeto, pois não era sabido de antemão qual seria o algoritmo final, de forma que não era possível determinar os tipos e quantidades de recursos necessários, nem as formas de otimizar sua utilização. No entanto, sabia-se que seria desejável, por exemplo, que o algoritmo proposto dispensasse o uso de cores das imagens, pois a operação simplesmente com tons de cinza reduziria a quantidade de recursos necessários (principalmente a quantidade de memória).

O algoritmo final foi fruto de etapas contínuas e exaustivas de estudo, tendo-se chegado finalmente a um resultado com operação conforme desejado, com quantidade reduzida de recursos. Uma contribuição importante para tal foi que pode-se de fato obter um algoritmo que não necessita de cores nas imagens, além de efetuar-se processamento serial. É importante mencionar, no entanto, que se num projeto futuro houver uma mudança no formato do algoritmo ou a necessidade de mensurar igualmente a íris (algoritmos para detecção da íris comumente usam informações relativas às cores da imagem), o uso de cores pode ser incluído sem maiores dificuldades.

5.3 SISTEMA AUTÔNOMO *VERSUS* SISTEMA OPERADO POR COMPUTADOR

Uma comparação entre o sistema desenvolvido (portátil, autônomo) com outro baseado em computador é apresentada na Tabela 3.

Tabela 3 - Vantagens e desvantagens do sistema proposto, comparado com outro que usa um computador.

Vantagens	Desvantagens
Dispensa necessidade de computador (operação autônoma)	Processamento sensivelmente mais lento (porem ainda com 24 quadros/segundo, suficientes para a presente aplicação)
Menor custo (FPGA + câmera + LCD tem custo menor do que computador + câmera)	Menor flexibilidade para modificações do sistema
Sem necessidade de <i>software</i> (projeto puramente elaborado em <i>hardware</i> , sem utilização de software livre ou comercial)	Sistema mais complexo para projetar e construir do que usando computador
Compacto e portátil	
Pode ser produzido em <i>ASICs</i> (reduz mais ainda custos – viabilidade depende do volume de produção)	

Um aspecto fundamental nesta comparação é a velocidade de processamento. A FPGA no presente sistema opera com um *clock* de 50 MHz, ou seja, uma instrução de um ciclo leva 20 ns. Computadores modernos operam tipicamente com frequência de *clock* de 2 a 3 GHz. Mesmo considerando os múltiplos ciclos de *clock* necessários para realização de uma instrução num sistema de uso genérico e multitarefa como um computador, o mesmo será mais rápido do que o sistema embarcado operando com a frequência de *clock* acima. Isso afeta diretamente a taxa de quadros por segundo que o sistema é capaz de processar; uma taxa muito baixa impede a mensuração correta da pupila em tempo real, ao passo que uma taxa muito alta (além do necessário para atingir as especificações) representa um desperdício de recursos (energia consumida, tamanho da memória, etc.), além de aumentar o custo. Em termos práticos, equipamentos análogos para testes de pupilometria operam normalmente com taxas em torno de 20 quadros por segundo (TEIKARI, 2007); tendo em vista que o sistema proposto opera com 24 quadros por segundo, verifica-se que a menor velocidade da FPGA, comparada a um computador convencional, não prejudica a qualidade do sistema (ao contrário, é uma taxa justa, pois permite exatamente a operação desejada, sem falta, nem excesso).

5.4 RECURSOS NECESSÁRIOS E CUSTOS DO PROJETO

FPGA

O componente principal do sistema é a FPGA. Devido aos recursos limitados e dificuldades de importação, foi utilizada no projeto uma FPGA que já estava disponível no Laboratório de Microeletrônica da UTFPR (Altera Cyclone II EP2C35F672C6N), a qual tem 33.216 elementos lógicos e custa US\$ 149,60. Após concluído o projeto e efetuadas todas as otimizações, chegou-se à utilização de recursos resumida na Tabela 4.

Tabela 4 - Uso dos recursos da FPGA usada no protótipo do sistema (Cyclone II EP2C35F672C6).

Recurso	Total Utilizado	Total Existente	Uso Total [%]
Elementos lógicos	28.576	33.216	89%
Flip-flops	5.219	33.216	16%
Bits de Memória	297.366	483.840	61%
Pinos	153	475	32%

Há uma série de FPGAs de menor custo capazes de atender às especificações do algoritmo final. Alguns exemplos constam na Tabela 5, a qual mostra, na primeira linha, a FPGA utilizada no protótipo do projeto, e nas demais, algumas alternativas de menor custo (até 65% mais baixo). A margem de recursos a ser deixada livre depende de se se deseja no futuro fazer modificações no sistema e quão grandes tais modificações podem ser.

Tabela 5 – FPGAs com recursos suficientes para atendimento ao projeto. Na primeira linha consta a FPGA utilizada no protótipo (fonte: www.altera.com).

FPGA	Elementos lógicos	Bits de memória	Pinos	Preço (US\$)
Altera Cyclone II EP2C35F672C6N	33K	483.840	475	149,60
Altera Cyclone II EP2C35F672C8N	33K	483.840	475	99,70
Altera Cyclone III EP3C40F324C8N	40K	1.161.216	195	81,30
Altera Cyclone IV EP4CE30F23C7N	28K	608.256	328	49,90

Câmera

O componente seguinte do projeto é a câmera. A câmera utilizada, cujo custo comercial é de US\$85, foi encomendada da Terasic (USA) e também importada com a ajuda do Laboratório de Microeletrônica da UTFPR. Trata-se de uma câmera em cores, com imagens de até 5 Mega *pixels*, geradas à taxa de 15 quadros por segundo na resolução máxima (5 M *pixels*) ou à taxa de até 70 quadros por segundo em resolução VGA (640x480). Tal câmera foi escolhida por possuir recursos um pouco além dos necessários, o que possibilitará o desenvolvimento de outros projetos no futuro, como a inclusão das cores no processamento quando necessário.

Após concluído o projeto e confirmados quais recursos são efetivamente necessários para implementar o algoritmo proposto, concluiu-se que uma câmera em preto e branco, com imagens de 0,4 Mega *pixels* (aproximadamente 640x480 *pixels* – resolução VGA) e taxa de 24 quadros por segundo, é suficiente para tal. Consequentemente, uma câmera de menor custo pode ser utilizada. Alguns exemplos constam na Tabela 6, a qual mostra, na primeira linha, a câmera utilizada no projeto, e nas demais, algumas alternativas de menor custo.

Tabela 6 – Câmeras com recursos suficientes para atendimento ao projeto. Na primeira linha consta a câmera utilizada no protótipo do sistema (fontes: www.terasic.com, www.electronics123.com/s.nl/it.A/id.32/.f, www.alibaba.com).

Câmera	Tipo	Pixels por quadro	Quadros por segundo	Preço (US\$)
Terasic TRDB D5M	Cores	VGA a 5 M	15 a 70	85,00
Terasic TRDB DC2	Cores	1,3 M	30	40,00
C328RS B/W Camera	Preto e Branco	VGA	30	59,90
I-Zone UA16U CMOS Camera	Cores	1,3M	42	22,00

Como no caso das FPGAs, a câmera utilizada também pode ter seu custo reduzido com a compra de outro componente. Apenas nos exemplos orçados, pode-se ver uma diferença de até 75% nos custos envolvidos, apresentando mais uma redução significativa no custo geral do projeto.

Display

O terceiro e último elemento principal do sistema é o display LCD. O display utilizado, cujo custo comercial é de US\$170, também foi encomendado da Terasic (USA) e importado junto com a câmera via Laboratório de Microeletrônica da UTFPR. Trata-se de um display LCD do tipo *touch screen*, com 800 colunas e 480 linhas. Por estar operando no modo VGA (resolução de 640x480 *pixels*), 80 colunas em ambas as extremidades do LCD não são utilizadas, permanecendo pretas e sem imagem. Tal *display* foi escolhido por possuir recursos um pouco além dos necessários, o que possibilitará o desenvolvimento de outros projetos no futuro.

Para o presente projeto, algumas características são dispensáveis, como a opção *touch screen*, de forma que um *display* de menor custo pode ser utilizado. Alguns exemplos constam na Tabela 7, a qual mostra, na primeira linha, o LCD utilizado no projeto, e nas demais, algumas alternativas de custo inferior. Novamente, como nos dois casos anteriores, é possível uma redução de custo significativa, superior a 50% em relação ao componente utilizado no protótipo do sistema.

Tabela 7 – Displays LCD com recursos suficientes para atendimento ao projeto. Na primeira linha consta o display utilizado no protótipo do sistema (fontes: www.terasic.com, www.farnell.com e www.alibaba.com).

Display LCD	Tipo	Tamanho (pixels)	Preço (US\$)
Terasic TRDB LTM	Cores, <i>touch screen</i>	800x480	170,00
Hitachi LMG7520RPFC	Cores, sem <i>touch screen</i>	VGA	152,00
FeelWorld TFT LCD module VGA	Cores, <i>touch screen</i>	VGA	80,00

Custo Total

O custo final dos três principais componentes do sistema consta na Tabela 8. Para os demais elementos do sistema (fonte de alimentação, placa de circuito impresso, encapsulamento, etc.), estima-se um custo da ordem de US\$35, o qual também foi incluído na tabela, resultando um custo total da ordem de US\$187.

Tabela 8 - Custo estimado do sistema.

Componente	Fornecedor/tipo	Preço (US\$)	Participação
FPGA	Altera	49,90	27%
Câmera	I-Zone	22,00	12%
LCD	FeelWorld	80,00	43%
Outros	Diversos	35,00	18%
Total		186,90	100%

5.5 IMPORTÂNCIA DE UTILIZAR-SE UMA HDL

Conforme mencionado, o projeto foi inteiramente desenvolvido utilizando linguagens de descrição de hardware (*Hardware Description Language* – HDL), mais especificamente, VHDL na maioria dos blocos e Verilog em alguns deles. Isso traz vários benefícios ao projeto, tais como:

- Permite desenvolver um projeto moderno e eficiente, independente da tecnologia ou fabricante dos dispositivos utilizados (VHDL e Verilog são linguagens universais, padronizadas pelo IEEE e independentes de tecnologia);

- As HDLs permitem não só a síntese dos circuitos, mas também sua completa simulação, tanto funcional quanto temporal;
- Os circuitos podem ser facilmente parametrizados, de forma que testes para vários valores de parâmetros tornam-se mais simples;
- Modificações e atualizações do projeto são mais simples e rápidas de executar;
- A documentação do projeto também é enormemente facilitada e *bugs* podem ser mais facilmente detectados;
- Os circuitos podem ser reusados, o que tende a baixar enormemente os custos de futuros desenvolvimentos. Por exemplo, os circuitos de interface desenvolvidos (a partir do zero) no presente projeto para controlar a câmera e o display podem ser reusados em outros projetos, pois eles independem do tipo de processamento sendo executado pelos circuitos internos;
- Outro fator importante é o menor *time-to-market*. Uma vez que o projetista tenha adquirido experiência com uma HDL, o tempo de projeto torna-se muito menor (e mais confiável) do que projetos feitos manualmente ou com entradas gráficas (esquemáticos);
- Outro fator relevante é que o mesmo código escrito para implementação em FPGA pode ser enviado a um fabricante de circuitos integrados (*foundry*) para fabricação de um ASIC (*Application Specific Integrated Circuit*), o qual é um circuito integrado fabricado especificamente para executar as mesmas funções da FPGA, porém com custo inferior se o volume de peças for elevado (isso depende de um estudo caso a caso). É também importante mencionar que basta ter-se o código VHDL ou Verilog (ou misto) para que a *foundry* faça o orçamento para fabricação do ASIC, o que é totalmente impossível num outro tipo de projeto.

5.6 TESTES

Além dos inúmeros testes feitos em cada etapa do desenvolvimento desta dissertação com o objetivo de validar os componentes do algoritmo e avaliar a demanda correspondente em termos de *hardware* (como, por exemplo, os testes relatados na Tabela 2, para validação do número de círculos candidatos), e também da demonstração feita à Banca Examinadora, com o protótipo completo e plenamente operacional, foram também efetuados testes com o protótipo em quatro pessoas, em dois dias diferentes, sob

condições de iluminação semelhantes e com a mesma configuração do equipamento. A mensuração é aproximada, uma vez que foi feita manualmente sobre a imagem detectada da pupila (a apresentação automática do resultado, em centímetros, por exemplo, a partir dessa imagem, é sugerida como trabalho futuro). Os resultados obtidos constam na Tabela 9.

Tabela 9 – Resultados de testes utilizando o protótipo com quatro pessoas em dois dias diferentes.

Indivíduo	Idade	Sexo	Valor medido (dia 1, dia 2)	Valor real horizontal aproximado	Valor real vertical aproximado
1	25	Feminino	2,4 cm 2,5 cm	2,6 cm 2,6 cm	2,7 cm 2,75 cm
2	25	Masculino	3,4 cm 3,4 cm	3,4 cm 3,5 cm	3,6 cm 3,65 cm
3	54	Feminino	2,75 cm 2,7 cm	2,8 cm 2,7 cm	2,85 cm 2,8 cm
4	59	Masculino	3,3 cm 3,35 cm	3,4 cm 3,4 cm	3,5 cm 3,55 cm

Como pode-se observar, a faixa de idade dos indivíduos parece não afetar os resultados. Além disso, o erro, que já é pequeno para o propósito deste dispositivo, será provavelmente reduzido quando for incorporada a extração automática dos resultados.

É importante lembrar que no algoritmo utilizado a pupila foi assumida como sendo perfeitamente circular, ao passo que, na prática, a pupila pode ser ligeiramente oval (ADONI, 2007). Além disso, algumas doenças, como catarata, ou operações do olho, como implante de lente, podem modificar o formato da pupila, de forma que é natural esperar-se um pequeno erro adicional em tais casos.

É importante enfatizar que os testes realizados têm apenas como objetivo efetuar uma prova de conceito, podendo assim o algoritmo proposto servir como possível base para o desenvolvimento de um sistema profissional de mensuração da pupila (em um trabalho de doutorado, por exemplo).

5.7 DIFICULDADES ENCONTRADAS

Ao longo do projeto, devido às limitações impostas pelas necessidades do projeto, o rumo do desenvolvimento alterou-se várias vezes. O fato de o sistema ser todo elaborado em *hardware*, sem nenhum uso de *software* e ferramentas de alto nível, que simplificariam

o desenvolvimento, especialmente em relação aos algoritmos como o detector de bordas de Canny e o método RANSAC, impôs um trabalho extra.

Trabalhar com algoritmos em um nível tão baixo de desenvolvimento (*hardware*) apresenta a vantagem de maior controle sobre a elaboração do mesmo e, em geral, traduz-se também em menor tempo de processamento e otimização no uso de recursos, já que várias configurações do algoritmo podem ser ajustadas às necessidades do projeto. Em contrapartida, é exigido do desenvolvedor o pleno entendimento de todos os componentes, e da forma como eles se interconectam para que o sistema funcione adequadamente. Por exemplo, no presente caso, isso implica em compreender plenamente o funcionamento, as configurações e o *hardware* das conexões existentes entre uma câmera CMOS, uma FPGA e um LCD.

Outra desvantagem para o desenvolvedor seguindo um desenvolvimento puro em *hardware* é a difícil comparação da eficiência de um algoritmo com outro. Para o projeto em questão, não há um método simples para comparar tempo de processamento, precisão de resultados e outras características relevantes entre um sistema de pupilometria e outro. A mensuração mais fiel desses parâmetros e o desenvolvimento de um ambiente que propicie maior repetibilidade dos testes serão sugeridos como possíveis avanços para um projeto futuro.

Outras dificuldades que se apresentaram foram devidas à escassez de componentes. O projeto todo é composto por apenas uma câmera CMOS, uma FPGA e um display LCD, com uma placa de circuito impresso para fazer as conexões (barramentos) entre os mesmos, e uma fonte de alimentação. Essa limitação trouxe à tona dificuldades como a impossibilidade de guardar um quadro inteiro da imagem em memória, visto que não havia uma quantidade de memória suficiente para isso. Adicionalmente, a escrita e leitura em uma memória externa reduziriam a velocidade de processamento.

Porém, foi graças a essas dificuldades que as soluções otimizadas surgiram. Por causa da impossibilidade de armazenamento de um quadro inteiro em memória, o processamento serial foi desenvolvido, de forma a reduzir o tempo total de processamento do sistema e introduzir *pipelining* na arquitetura proposta. Mesmo com a dificuldade de se trabalhar com *hardware* puro e VHDL e Verilog, isso permitiu reduzir ainda mais o tempo de processamento e a quantidade de recursos necessários comparados com soluções em *software*, e foi possível compreender, num nível baixo de desenvolvimento, todos os detalhes necessários para a resolução dos problemas apresentados.

CAPÍTULO 6

CONCLUSÕES E TRABALHOS FUTUROS

O estudo, desenvolvimento, implementação e teste de um novo algoritmo e correspondente hardware foram apresentados, os quais visam obter imagens digitais da pupila do olho humano de forma simples, a baixo custo, e em tempo real, à taxa de 24 quadros por segundo. O resultado final é um equipamento completamente autônomo, portátil e capaz de ser operado por qualquer indivíduo, o qual poderá servir como ponto de partida para um sistema profissional de mensuração da pupila.

A implementação foi feita inteiramente em hardware, ilustrando a possibilidade de fazer-se processamento digital de imagens puramente em hardware, sem uso de um computador ou de qualquer outro equipamento ou software externo. A escolha de cada componente do sistema foi feita procurando atender às especificações do projeto com o menor custo possível, fato este que também foi determinante no desenvolvimento do algoritmo final.

Todos os circuitos, incluindo o algoritmo principal proposto e todas as demais partes (controlador da câmera, controlador do display, memória de *pixels*, etc.) foram desenvolvidos utilizando VHDL e Verilog, resultando em um projeto moderno, facilmente simulável e reusável.

Outro fator importante é que o protótipo, construído e testado usando uma única FPGA, pode ser facilmente convertido para implementações usando outros tipos de componentes eletrônicos, como microcontroladores, pois isso não altera a qualidade do algoritmo proposto.

Algumas propostas podem ser feitas para possíveis trabalhos futuros relativos ao projeto desenvolvido. Uma delas seria a inclusão de um sistema operacional embarcado (SOE). Utilizando ainda a FPGA e configurando-a em forma de microprocessador, é possível instalar um SOE, o que possibilitaria o uso de funções e algoritmos já testados e usar linguagens de programação de alto nível (e.g. C, C++) no sistema. Seria interessante comparar o desempenho do sistema puramente em *hardware* com o sistema operando com o SOE a fim de ver se existe uma diferença substancial entre eles. Uma vantagem do SOE seria a possível simplificação de atualizações do sistema.

Outro trabalho natural de esperar-se para o futuro seria a utilização de um processador convencional (ARM, por exemplo) no lugar da FPGA, para comparação de custo, performance, etc. As principais desvantagens seriam a obtenção de um hardware não reconfigurável e a impossibilidade de utilizar VHDL ou Verilog, ficando também impossibilitada uma análise direta sobre a fabricação de um ASIC correspondente.

Finalmente, é importante lembrar que o sistema proposto não é ainda um sistema final, mas sim um protótipo para verificação de um novo algoritmo para obtenção de imagens digitais da pupila humana e análise da viabilidade de implementar-se o mesmo inteiramente em hardware, tendo também por objetivo avaliar a quantidade mínima de recursos de hardware necessários para tal. Para um produto definitivo e pronto para atender ao mercado médico, é necessário dar continuidade a esses estudos, incorporando, por exemplo, a apresentação do valor numérico do perímetro no display a partir das imagens processadas, além de efetuar-se outros testes, envolvendo principalmente precisão e repetibilidade dos resultados. Tal trabalho poderia ser o tema de um doutorado nesta área da pupilometria.

REFERÊNCIAS

ABRANTES, A. J., Marques, J. S., Pattern Recognition Methods for Object Boundary Detection, British Machine Vision Conference, pg. 409-417, 1998.

ADONI, A., McNett, M., The Pupillary Response in Traumatic Brain Injury: A guide for Trauma Nurses, Journal of Trauma Nursing, Vol. 14, pg. 191 – 196, 2007.

ALTERA, Disponível em:
<<http://www.buyaltera.com/scripts/partsearch.dll?Detail&name=544-1694-ND>>. Acesso em: 19/11/2010.

BEATY, J., Lucero-Wagnoer, B., The Pupillary System, Handbook of Psychophysiology, 2ª Edição, Cambridge University Press. pg. 142–162, 2000.

BOURKE, P., Equation of a Circle from 3 Points, 1990, Disponível em:
<<http://paulbourke.net/geometry/circlefrom3/>>. Acesso em: 22/10/2010.

CANNY, J., A Computational Approach to Edge Detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 8, pg. 679-698, 1986.

CANTZLER, H., Random Samples Consensus (RANSAC), British Machine Vision Conference, Cardiff, UK, pg. 458-467, 2002.

DARWIN, C., The Expression of Emotions in Man and Animals, London: John Murray, 1871.

DAUGMAN, J., High Confidence Personal Identification by Rapid Video Analysis of Iris Texture. Proceedings of IEEE, International Carnahan Conference on Security Technology, 50-60, 1992.

DAUGMAN, J., High Confidence Recognition of Persons by Iris Patterns. IEEE 35th International Carnahan Conference on Security Technology, 254-263, 2011.

DUDA, R., Hart, P., Pattern Classification, Wiley-Interscience, 2ª Edição, 1972.

Enciclopédia Britannica, Sensory Reception: Human Vision: Structure and Function of the Eye, 1987.

FERNANDES, T., Identificação de Indivíduos por Biometria da Íris e Densidade de Coincidências de Fase, Dissertação de Mestrado, UTFPR, 2009.

FERRARI, G. L., Pupilometria Dinâmica: Aplicação na Detecção e Avaliação da Neuropatia Autonômica Diabética e Estudo da Correlação entre a Resposta Temporal da Pupila ao Estímulo Visual e a Glicemia, Tese de Doutorado, UTFPR, 2008.

FISHER, R., Perkins, S., Walker, A., Wolfart, E., Hypermedia Image Processing Reference, Image Processing Learning Resources, Disponível em: <<http://homepages.inf.ed.ac.uk/rbf/HIPR2/>>. Acesso em: 19/11/2010.

GABOR, D., Theory of Communication, Journal of The Institute of Electrical Engineers, 93, 429-457, 1946.

GONZALEZ, R.C., Digital Image Processing, 3rd Edition, Prentice Hall, 2007.

GOURAS, P., Color Vision, Disponível em: <<http://webvision.med.utah.edu/book/part-vii-color-vision/color-vision/>>. Acesso em: 19/11/2010.

KAISEMI Group, FPGA to ASIC, ASIC to ASIC, DSP to ASIC Conversions, Disponível em: <<http://www.slideshare.net/kaisemi1/kaisemi-fpga-to-asic-conversions>>. Acesso em: 19/11/2010.

LI, D, Parkhurst, D. J., Starburst: A robust algorithm for video-based eye tracking, Elsevier Science, 2005.

NILSSON, D. E., Evolution of the Eye, PBS.org Article Library, 2010, Disponível em: <http://www.pbs.org/wgbh/evolution/library/01/1/l_011_01.html>. Acesso em: 19/11/2010.

ON Semiconductor, FPGA-to-ASIC Conversion, Disponível em: <<http://www.onsemi.com/PowerSolutions/content.do?id=16788>>. Acesso em: 19/11/2010.

PAN, L., Xie, M., The Algorithm Of Iris Image Preprocessing, Fourth IEEE Workshop on Automatic Identification Advanced Technologies, pg. 134-138, 2005.

PARTALA, T., Surakka, V., Pupil size variation as an indication of affective Processing, International Journal of Human-Computer Studies, 2002.

PEDRONI, V. A., Circuit Design and Simulation with VHDL, MIT Press, 2ª Edição, 2010.

PEDRONI, V. A., Circuit Design with VHDL, MIT Press, 1ª Edição, 2005.

PRATT, W. K., Digital Image Processing. New York, John Wiley & Sons, 1991.

RAD, A. A., Faez, K., Qaragozlou, N., Fast Circle Detection Using Gradient Pair Vectors, Proc. 7th Digital Image Computing: Techniques and Applications, Sydney, 2003.

ROVANI, A. Z., Sistema para Captura Automática de Imagens de Íris, Dissertação de Mestrado, UTFPR, 2006.

SHAPIRO, G., Computer Vision, Prentice-Hall, Inc. 2001.

TEIKAR, P., Automated Pupillometry, 2007, Disponível em: <http://users.tkk.fi/~jteikari/Teikari_AutomatedPupillometry.pdf>. Acesso em: 21/10/2010.

TERASIC1, Disponível em: <<http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=68&No=281>>. Acesso em: 19/11/2010.

TERASIC2, Disponível em: <<http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=68&No=213>>. Acesso em: 19/11/2010.

TERASIC3, Disponível em: <<http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=68&No=50>>. Acesso em: 19/11/2010.

TEXAS INSTRUMENTS, Video and Imaging Guide, Catálogo de Produtos.

THOMPSON, H. S., Otto Lowenstein, Pioneer Pupillographer, Journal of Neuro-Ophthalmology, 25^o Volume, 1^a Edição, pg. 44-49, 2005.

WAITE, J., Eckhard Hess, 1999, Disponível em: <<http://www.muskingum.edu/~psych/psycweb/history/hess.htm>>. Acesso em: 22/10/2010.

YAO, P., LI, J., Ye, X., Zhuang, Z., & Li, B., Iris Recognition Algorithm using Modified Log-Gabor Filters, 18th International Conference on Pattern Recognition, 4, 461-464, 2006.