UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

PUPILOMETRIA DINÂMICA: UMA PROPOSTA DE RASTREAMENTO DA POSIÇÃO E TAMANHO DA PUPILA HUMANA EM TEMPO REAL

Alessandro Gontijo da Costa Dias

Uberlândia, MG

2014

PUPILOMETRIA DINÂMICA: UMA PROPOSTA DE RASTREAMENTO DA POSIÇÃO E TAMANHO DA PUPILA HUMANA EM TEMPO REAL

Alessandro Gontijo da Costa Dias

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal de Uberlândia, como requisito parcial para obtenção do título de Mestre em Ciências.

Área de concentração: Processamento da Informação.

Prof. Dr. Antônio C. P. Veiga Orientador	Prof. Dr. Edgard A. Lamounier Júnior Coordenador do Programa de Pós-Graduação

Uberlândia, MG

2014

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

D541p 2014

Dias, Alessandro Gontijo da Costa, 1984-

Pupilometria dinâmica : uma proposta de rastreamento da posição e tamanho da pupila humana em tempo real / Alessandro Gontijo da Costa Dias. - 2014.

116 f. : il.

Orientador: Antônio Cláudio Paschoarelli Veiga.

Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Elétrica.

Inclui bibliografia.

1. Engenharia elétrica - Teses. 2. Pupilometria - Teses. 3. Olhos - Exame - Teses. 4. Processamento de imagens - Teses. I. Veiga, Antônio Cláudio Paschoarelli, 1963-. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Engenharia Elétrica. III. Título.

CDU: 621.3

PUPILOMETRIA DINÂMICA: UMA PROPOSTA DE RASTREAMENTO DA POSIÇÃO E TAMANHO DA PUPILA HUMANA EM TEMPO REAL

Alessandro Gontijo da Costa Dias

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal de Uberlândia, como requisito parcial para obtenção do título de Mestre em Ciências.

Área de concentração: Processamento da Informação.

Aprovada em 17 de Janeiro de 2014.

Banca examinadora:

Antônio. C. P. Veiga, Dr - Orientador (UFU)

Alexandre Coutinho Mateus, Dr (UFU)

Luciano Xavier Medeiros, Dr (UFTM)

Milena Bueno Pereira Carneiro, Dr (UFU)

À todos que acreditaram
e tornaram este Trabalho possível.

Agradecimentos

Aos meus pais, Christiane e Eduardo que juntamente com Valcir e Sandra, me criaram e me deram a base para construir o que sou hoje.

Aos meus irmãos, Taciana, Cristiano e Geraldo, muito amigos com quem posso sempre contar.

À minha avó Solange por estar sempre presente e auxiliando em tudo.

À Ana Paula por fazer com que essa Dissertação se desenvolvesse, pela ajuda, amizade e companheirismo.

Ao meu orientador Prof. Dr. Paschoarelli, pela paciência e apoio.

Aos meus colegas da pós-graduação, em especial ao Cláriton Bernadelli, por ceder imagens de seu estudo, pelas conversas e dicas que contribuíram para a construção deste trabalho.

À todos os amigos do Instituto de Física, Agrenor, Marlon e Guilherme, que me acompanharam e tiveram paciência enquanto este trabalho era desenvolvido.

Aos meus amigos e todas as pessoas que estiveram indiretamente ligadas a este trabalho, saibam que vocês também o tornou possível.

À FAPEMIG pelo apoio financeiro aos projetos relacionados a este trabalho, que foram e serão de grande ajuda para a continuação do mesmo.

A Deus, que fez com que todos aqui citados influenciassem para a conclusão do mesmo.

Muito Obrigado.

“Mas pra quem tem pensamento forte,

o impossível é só questão de opinião”

Chorão E Thiago Castanho

Resumo

O estudo dos movimentos (contração e dilatação) da pupila tem interesse clínico relevante, pois é utilizado como método de avaliação tanto do sistema visual humano quanto do sistema nervoso.

Este trabalho apresenta uma técnica para efetuar o rastreamento tanto da posição quanto do diâmetro da pupila humana em tempo real. A técnica sugerida utiliza as informações da pupila encontrada no frame anterior, tamanho e posição, que são utilizadas em um algoritmo que objetiva a redução do tempo gasto no rastreamento do próximo frame. Utilizando este algoritmo é possível efetuar o rastreamento da posição e tamanho da pupila humana em tempo real e em alguns casos com uma média de até 140 frames por segundo.

Foram realizadas comparações com outros trabalhos tanto em relação a tempo de processamento de cada frame, quanto em precisão na localização e diâmetro da pupila encontrada em cada imagem.

O rastreamento do diâmetro da pupila é fundamental na extração de parâmetros relacionados aos movimentos pupilares, e estes auxiliam no diagnósticos de diversas doenças, entre outras aplicações. O presente trabalho faz não só o rastreamento do diâmetro mas também da posição da pupila em tempo real.

Palavras-chave

Rastreamento, Processamento em Tempo Real, Transformada de Hough, Processamento de Imagens, Pupila, Vídeo *Tracking*.

Abstract

The study of the pupil's movements (contraction and dilatation) has a significant clinical interest because it is used as a method of evaluating both, the human visual system and nervous system.

This paper presents a technique that can make the human pupil tracking in real time, both the position and the diameter. That is crucial in extracting parameters related to pupillary movements, and these help in diagnosis of various diseases, among other applications. The suggested technique uses the information of the pupil found in the previous frame, size and position, that are used in an algorithm that aims to reduce the time spent on the next frame tracking. Using this procedure it is possible to make the human pupil tracking in real time, in some cases with an average of up to 140 frames per second.

Comparisons with other studies were performed both, for the processing time of each frame and for accuracy in pupil's location and diameter found in each image .

The pupil's position tracking is crucial in extracting parameters related to pupillary movements, and these help in diagnosis of various diseases, among other applications. The present work does not track only the diameter but also the position of the pupil in real time.

Palavras-chave

Tracking, Real Time Processing, Hough Transform, Image Processing, Pupil, Video Tracking.

Sumário

Lista de Figuras

Lista de Tabelas

1 INTRODUÇÃO

1.1 Introdução

A pupila realiza movimentos em diversas situações como, por exemplo, durante o ajuste de foco, devido ao reflexo à luz, em situações de medo, mudança no nível de atenção, entre outros [1]. Esses movimentos correspondem a contrações ou mioses e dilatações ou midríases.

Mais precisamente a pupila dilata pela ativação do sistema nervoso autônomo simpático (SNS) e contrai por ativação do sistema nervoso autônomo parassimpático (SNP) [1][2].

O sistema nervoso autônomo (SNA) que deriva do sistema nervoso periférico possui a função de regular as atividades involuntárias do nosso organismo, como por exemplo, pressão, temperatura e batimentos cardíacos. Ele, por sua vez, pode ser dividido em sistema nervoso autônomo simpático e parassimpático. O primeiro é utilizado quando o organismo requer respostas rápidas e o último nos casos opostos [3].

O estudo do movimento da pupila tem interesse clínico relevante, pois esta atua como um indicador objetivo da sensibilidade da retina à luz e, por consequência, do nervo óptico. A sua oscilação permanente é resultado do equilíbrio entre fluxos opostos dos sistemas nervosos simpático e parassimpático. O estudo dos movimentos da pupila de um indivíduo permite o diagnóstico de várias doenças, entre as quais distúrbios do sono (narcolepsia), esquizofrenia (reação a fármacos), Síndrome de Adie, Alzheimer, dependência de narcóticos,

entre outras [4].

Além disso, a função pupilar tem sido reconhecida como uma importante porta de acesso não invasiva para avaliação do sistema nervoso autônomo [5].

Porque o estudo dos movimentos da pupila é importante? Para responder a essa pergunta vários métodos têm sido usados para medir as funções pupilares, principalmente aqueles que envolvem o registro do reflexo da função pupilar a um estímulo de luz (ou, simplesmente, reflexo pupilar), também conhecidos como pupilometria dinâmica, por fornecer informação da inervação simpática e parassimpática [5].

Recentemente, alguns trabalhos têm indicado parâmetros e valores padrões da resposta temporal da pupila para indivíduos sadios [6][7].

A pupila dilata e contrai por ativação dos sistemas nervosos simpático e parassimpático, respectivamente. Este mecanismo é muito importante na avaliação do estado geral do sistema nervoso autônomo. Surge, portanto, a necessidade de calcular com precisão o tamanho da pupila e da sua evolução no tempo, para fazer tal avaliação [8]. É importante salientar que os métodos de pupilometria são métodos não invasivos, confortáveis para quem os realiza e que podem ser utilizados em diversas áreas, tais como a oftalmologia, a psiquiatria, a psicologia, a biométrica, entre outras [4].

Conforme descrito em [9], vários fatores podem alterar ou iniciar o processo de movimentação da pupila. Basicamente esses movimentos são: (i) a acomodação, que ocorre devido ao processo de foco, (ii) o reflexo pupilar a luz *(Pupil Light Reflex* - PLR), que é o reflexo da pupila para adaptar o sistema visual a iluminação ambiente e (iii) o *hippus*, uma oscilação que ocorre constantemente como parte de um processo de convergência disparado pela acomodação e pelo PLR.

Além disso, encontra-se na literatura vários fatores que podem afetar a dinâmica dos movimentos pupilares, como por exemplo, o uso de drogas, doenças, fatores emocionais, respiração e batimentos cardíacos, interesse e curiosidade, idade, cor da íris, nível de consciência, o local onde um raio de luz atinge o cristalino, na borda ou no centro, comprimento de onda da luz incidente e os padrões espaciais.

A pupila controla a quantidade de luz que entra nos olhos protegendo assim, a retina

de uma carga muito intensa de luz. O PLR é um processo ligado à intensidade de luz incidente na retina e pode ser influenciado por fatores como: o uso de medicamentos, o estado emocional do indivíduo, o contraste da imagem percebida, além de produzir resultados diferentes de acordo com o comprimento de onda.

Uma evidência percebida pelo PLR é que o reflexo é síncrono entre as duas pupilas. A falta desse sincronismo é devido a algum mau funcionamento do sistema visual humano. Na literatura esse sincronismo é chamado de reflexo consensual [10].

O PLR é um exemplo claro de homeostase em seres humanos, ou seja, um mecanismo dinâmico de equilíbrio controlado por sistemas de regulação inter-relacionados. Esse processo visa estabelecer o equilíbrio do sistema contrariando qualquer mudança com a finalidade de manter a capacidade do ser humano de adaptação ao meio. Ele modela as ações do corpo em dois passos distintos: percepção e ajuste. Na íris, a percepção ocorre quando a luz chega à retina e a informação é enviada ao cérebro, e o ajuste quando o cérebro envia sinais para fechar ou abrir a pupila [11].

As pupilas estão em constante movimento de contração e relaxamento, o movimento pulsátil denominado *hippus*, é determinado pelo balanço entre o sistema nervoso simpático e o parassimpático [2]. Ao longo da adaptação ao estímulo e mesmo em uma iluminação estável ou na escuridão total ocorre uma variação de cerca de $0,2\ Hz$ [4]. Estas variações possuem uma sincronia com a respiração e com o ritmo regular circulatório e neural. O ritmo da respiração (*High Frequency* - HF) é a marcação para o nervo parassimpático, enquanto os batimentos cardíacos possuem uma relação maior com o nervo simpático (*Low Frequency* - LF) [9].

Uma variação anormal na acomodação pode ser resultado de algum problema no sistema nervoso, podendo ser ocasionado por alcoolismo, diabetes, AIDS, síndrome de down, depressão, mal de Alzheimer, hipertensão intercranial, medo e assim por diante [9]. O PLR e o hippus são processos automáticos, uma reação reflexa, ou seja, diferente da acomodação, eles não são comandados pelo cérebro, mas sim pelo sistema nervoso autônomo (SNA) [9].

1.2 Pupilometria

Um ciclo pupilar pode ser alcançado por meio do movimento de contração da pupila seguido por uma expansão e finalmente retornando ao estado inicial. Existem outros tipos de simulações, por exemplo, o *pupil escape* é o nome considerado quando uma pupila dilatada sofre a ação de um estímulo de luz e contrai e, após alguns segundos ou minutos, retorna ao seu estado inicial. Quando a intensidade de luz é muito grande e a pupila não retorna ao seu estado inicial, denomina-se *pupil capture* [9].

A pupilometria mede diversos componentes: amplitude máxima, latência, velocidade de dilatação e contração e tamanho máximo e mínimo. A amplitude corresponde à diferença entre o tamanho inicial e o mínimo da pupila durante o PLR.

Sabe-se que o cristalino deixa de ser totalmente transparente com a idade, reduzindo a quantidade de luz que chega à retina. Como a amplitude é proporcional ao logaritmo de todo o fluxo de luz que chega na retina, ela acaba tornando-se, também, dependente da idade do indivíduo. Essa perda é estimada em cerca de $0,4$ mm por década a partir dos 20 anos [9]. Smith e Dewhirst [12] estudaram a relação da idade com as dimensões pupilares e realizaram testes em 163 indivíduos com idades entre 15 e 92 anos. A Tabela 1.1, adaptada de [12], mostra os resultados obtidos.

Latência é o tempo que a pupila leva para reagir a um estímulo de luz. Em observações clínicas, a latência pode identificar objetivamente os atrasos no processamento visual, que são proporcionais ao dano causado ao sistema visual. A latência comparada a amplitude é menos volátil e é menos afetada pelas propriedades mecânicas da íris, no entanto ela ainda é afetada. Para definir matematicamente o instante que a latência termina, utiliza-se a segunda derivada da função de tamanho da pupila em relação ao tempo. Durante a maior inclinação no gráfico de velocidade (1^a derivada) existe um canal na função de aceleração (2^a derivada). O pico desse canal é o ponto onde a latência termina. Existem evidências que a latência e a amplitude são governadas pela ativação parassimpática junto com a velocidade de contração. A velocidade de dilatação é governada pela ativação simpática.

Assim como a amplitude e a latência, o tamanho da pupila, em estado de descanso,

é maior para pessoas mais velhas. A velocidade máxima de dilatação e de contração é menor nessas pessoas (o que indica perda da força de atuação dos nervos simpáticos e parassimpáticos). A ciência ainda não determinou o que faz essas alterações ocorrerem, mas como o cristalino torna-se menos transparente com a idade, obviamente, ele deixa passar menos luz para a retina, e assim, menos contração. Uma das hipóteses seria o aumento da influência parassimpática ou a perda da influência simpática. É importante salientar também, que o tamanho máximo da pupila não depende do tamanho da íris, mas sim da sua inervação [9].

Atualmente, existem vários modelos que simulam a dinâmica da pupila e vários estudos que determinam seus parâmetros para o propósito de pesquisa e da prática clínica.

1.3 Motivação

Conhecendo as possibilidades da pupilometria descritas na seção 1.1, e analisando alguns trabalhos que envolvem o estudo de pupilometria, [7] e [5], é possível analisar os movimentos pupilares, porém essas análises dependem de uma pré-gravação dos movimen-

Tabela 1.1: *Variação do diâmetro da pupila na ausência de luz.*

Variação de idade	Média	Diâmetro da pupila(mm)			Diâmetro da pupila(%)		
(Idade)	(Idade)	Mín	Esperado	Máx	Mín	Esperado	Máx
15-19	17	6,0	7,4	8,8	52,0	63,3	74,6
20-24	22	5,8	7,2	8,6	50,5	61,8	73,1
25-29	27	5,6	7,0	8,4	49,0	60,3	71,5
30-34	32	5,4	6,8	8,2	47,5	58,8	70,0
35-39	37	5,2	6,6	8,0	46,0	57,3	68,5
40-44	42	5,0	6,4	7,8	44,5	55,8	67,0
45-49	47	4,8	6,2	7,6	43,0	54,3	65,5
50-54	52	4,6	6,0	7,4	41,5	52,7	64,0
55-59	57	4,4	5,8	7,2	40,0	51,2	62,5
60-64	62	4,2	5,6	7,0	38,5	49,7	61,0
65-69	67	4,0	5,4	6,8	37,0	48,2	59,5
70-74	72	3,8	5,2	6,6	35,4	46,7	58,0

tos pupilares da pessoa em questão, o que torna a pupilometria dinâmica um processo que exige tempo para ser realizado. Normalmente, no processo da pupilometria são adquiridas as imagens com o olho submetido a diversas variações de iluminação, dependendo do objetivo do sistema e do estudo proposto, após capturadas, essas imagens são transferidas para um computador que efetua o processamento, por meio do rastreamento da pupila e da associação com a intensidade da luz no olho do indivíduo naquele momento. A partir daí tem-se os dados que serão analisados.

Analisando outros trabalhos que também efetuam o rastreamento da pupila, [13] e [14], porém não possuem como objetivo a pupilometria com rastreamento de todas as características da pupila, somente da posição, observa-se que os mesmos possuem poucos resultados. Dessa forma, não é possível utilizá-los como referência na realização da pupilometria dinâmica.

Um estudo que merece destaque no ramo é o de Bernadelli [11], que conseguiu realizar a pupilometria dinâmica utilizando materiais de baixo custo, com boa resolução temporal e espacial. A partir dos resultados obtidos por ele, objetiva-se, neste trabalho, realizar o mesmo tipo de processamento, porém, em um tempo suficiente para captar a variação do tamanho da pupila de um frame para o outro.

1.4 Objetivos

O objetivo desse trabalho é realizar a pupilometria dinâmica em tempo real. Ou seja, a imagem será captada e processada simultaneamente, gerando os resultados de forma mais rápida.

Para a realização desse processo, será necessário a criação de um software que seja capaz de analisar as imagens de forma tão rápida quanto o desejado, com melhor precisão quanto a posição e tamanho da pupila encontrada.

Para efeito de comparação, o Sistema Proposto será submetido a testes de processamento em tempo real, tempo gasto no processamento, comparando com a referência [11],

e precisão de posição e tamanho da pupila encontrada, utilizando o Banco de Imagens CASIA [15] e outro trabalho também conhecido que efetua o rastreamento da posição e do tamanho da pupila em uma imagem [16].

1.5 Estrutura desta Dissertação

Esta Dissertação está dividida em seis capítulos que foram distribuídos para que o assunto proposto seja tratado de forma clara e objetiva, sendo estes:

O Capítulo 1 é a introdução do trabalho, que explica o que é Pupilometria, quais as motivações que incentivaram o mesmo e os objetivos.

O Capítulo 2 apresenta algumas técnicas de processamento de imagens digitais que possibilitam o processamento realizado posteriormente no desenvolvimento do software resultante deste trabalho.

O Capítulo 3 demonstra algumas técnicas de processamento de vídeos digitais que serviram como base para que o objetivo deste trabalho fosse alcançado.

O Capítulo 4 descreve o Sistema proposto, expondo como foi desenvolvido o Sistema, detalhando as bibliotecas e técnicas envolvidas para se chegar ao resultado desejado.

O Capítulo 5 detalha as aplicações e resultados, mostrando os testes realizados, as comparações com outros trabalhos do mesmo tipo e os resultados do presente trabalho.

E, finalmente, o Capítulo 6 que faz uma conclusão e cita as contribuições e possibilidades de trabalhos futuros que podem contribuir ainda mais com a Pupilometria Dinâmica.

2 PROCESSAMENTO DE IMAGENS DIGITAIS

2.1 Introdução

Para os seres humanos é fácil distinguir algumas informações de uma imagem, definida por uma função bidimensional, $f(x, y)$, mas já para uma máquina, são apenas números organizados em uma matriz bidimensional, que não possuem nenhuma lógica ou organização.

Com o objetivo de extrair uma informação mais útil de uma imagem digital na forma $f(x, y)$, é utilizado um sistema de processamento de imagens explicado na próxima seção, que engloba desde a aquisição da imagem até a informação adquirida após todas as etapas do processamento.

2.2 Etapas de um Sistema de Processamento de Imagens

Um sistema de Processamento de Imagens é utilizado para se obter de uma imagem digital, na forma $f(x, y)$, informações mais detalhadas, ou precisas, e legíveis para um computador, de acordo com as etapas ilustradas na Figura 2.1, adaptada de [17].

As escolhas feitas para se chegar ao *resultado* desejado, ou seja, encontrar um objeto com o formato circular, dependem de um bom *domínio do problema* em questão e de uma

base de conhecimento das ferramentas que serão utilizadas.

Algumas ferramentas usadas nas diversas etapas de um sistema de processamento de imagens serão explicadas nas próximas subseções, dando maiores detalhes à algumas etapas conforme as necessidades do presente trabalho.

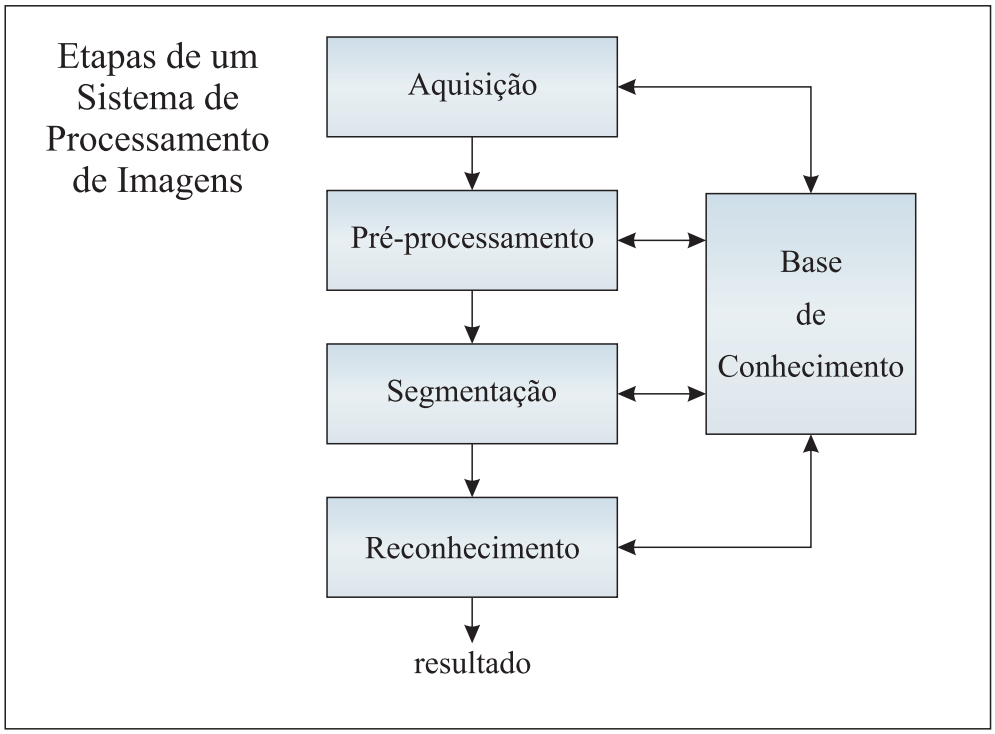


Figura 2.1: *Etapas de um sistema de processamento de imagens.*

2.2.1 Etapa de Aquisição

Também de acordo com [17], na etapa de *aquisição* a imagem é capturada por um dispositivo e convertida de forma adequada para o processamento que se deseja fazer.

Conforme citado anteriormente, a fim de obter um bom *resultado*, encontrar a posição e o diâmetro da pupila, e possuindo um bom *domínio do problema*, para a etapa de *aquisição* deve-se escolher um dispositivo que satisfaça as necessidades do seu problema, quanto às características de: espectro, resolução, tempo de exposição, entre outros fatores que influenciam diretamente em um bom resultado.

Após a aquisição da imagem (normalmente na forma analógica) o dispositivo ou o próprio computador a transforma para o formato digital, envolvendo dois processos, *amostragem* e *quantização* conforme [18].

2.2.2 Etapa de Pré-processamento

A imagem no formato digital resultante do processo de aquisição pode não estar adequada a uma aplicação específica [18], daí a necessidade de um pré-processamento, visando melhorar a qualidade da imagem e/ou alterar algumas propriedades, como cores, contraste, brilho.

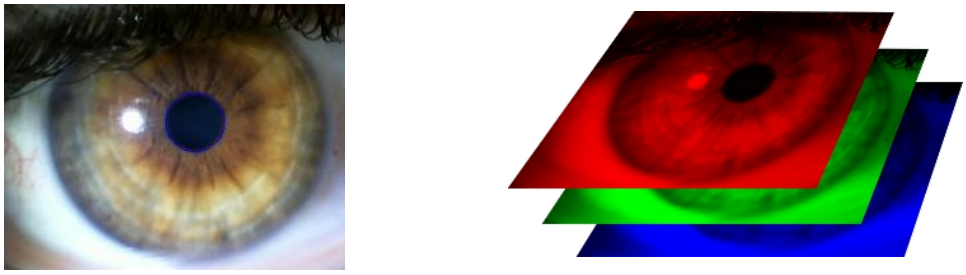
No presente trabalho, com o objetivo de diminuir o tempo de processamento em cada imagem esta seção do pré-processamento foi idealizada de forma a diminuir informações desnecessárias da imagem e realçar detalhes importantes, de acordo com os seguintes itens:

2.2.2.1 Conversão para Escala de Cinza

Com um objetivo semelhante ao deste trabalho, no Reconhecimento de Iris [19], utiliza-se uma câmera monocromática com iluminação infra vermelha de forma que a aquisição das imagens não seja invasiva para os humanos. Porém, como a imagem adquirida pelo dispositivo utilizado é colorida, utiliza-se a conversão para escala de cinza.

A imagem obtida pelo dispositivo utilizado neste trabalho é uma matriz de três camadas, cada camada sendo uma cor do modelo RGB, como representado na Figura 2.2.

Como o objetivo deste trabalho é utilizar o menor tempo possível no processamento de cada imagem, essa conversão também tem um papel importante, de forma que não sejam perdidas informações relevantes da imagem original para a imagem em escala de cinza, ver Figura 2.3, e ainda proporciona que o tempo utilizado nos processamentos seguintes seja menor que quando se utiliza uma imagem colorida (original).



(a) *Imagem colorida no modelo RGB.*

(b) *Camadas Vermelho, Verde e Azul (RGB).*

Figura 2.2: *Representação de uma imagem colorida no modelo RGB.*

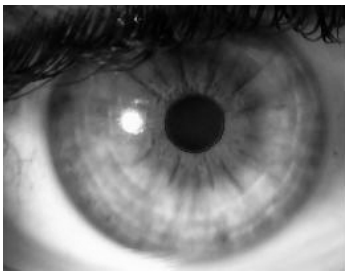


Figura 2.3: *Imagem da Figura 2.2 em Escala de Cinza.*

2.2.2.2 Equalização de Histograma

De acordo com [17], o *histograma* de uma imagem digital é a distribuição dos níveis de cinza desta imagem, normalmente representado por um gráfico, como na Figura 2.4, que mostra o número de pixels da imagem para cada nível de cinza.

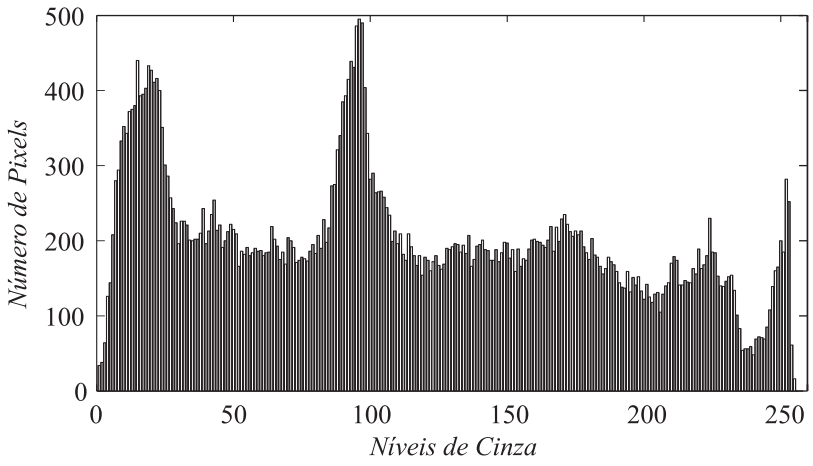


Figura 2.4: *Histograma da imagem da Figura 2.3.*

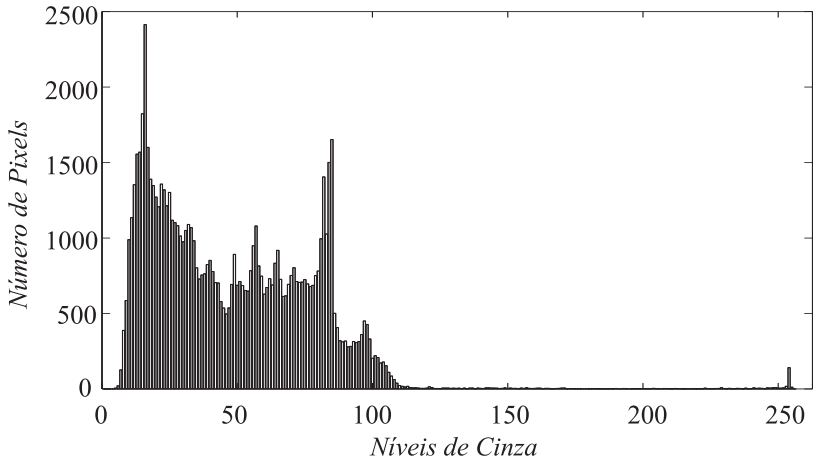
Desta forma, uma imagem clara apresenta um histograma onde os níveis de cinza estão mais afastados da origem, ou seja, com maior concentração de níveis de cinza com valores elevados (normalmente acima de 128). E uma imagem escura apresenta um histograma com valores baixos de níveis de cinza, próximos à origem.

Assim, como nem sempre existe um controle da iluminação no momento da aquisição da imagem, é desejado que exista uma certa simetria na distribuição dos níveis de cinza, ou seja, no histograma, de forma que seja mais fácil a percepção de informações na imagem. Para isso aplica-se a *Equalização de Histograma*, a Figura 2.5 apresenta um exemplo da aplicação desta técnica, mostrando uma imagem original e o resultado da sua equalização de histograma. A Figura 2.6 apresenta os respectivos histogramas.

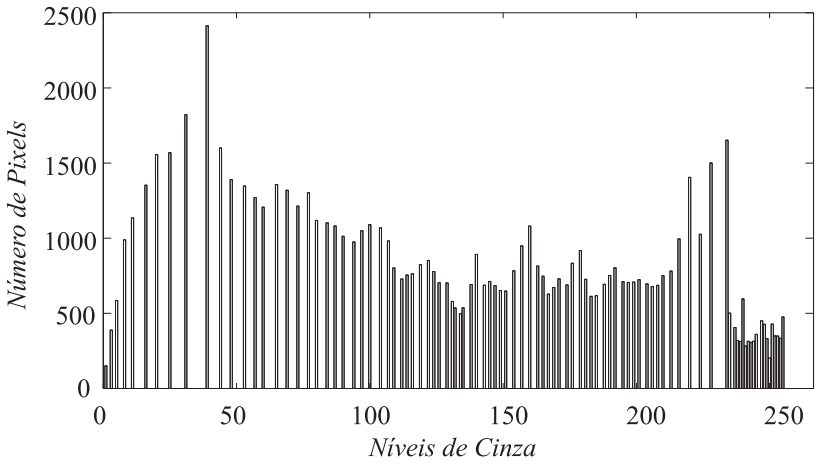
(a) *Original.* (b) *Histograma Equalizado.*

Figura 2.5: *Comparação entre uma imagem original e a imagem resultante da equalização de histograma.*

Em termos computacionais a aplicação da equalização do histograma na imagem faz com que seja mais fácil e eficiente a realização de uma parte da segmentação descrita na próxima seção, que é a detecção de bordas.



(a) *Original.*



(b) *Histograma Equalizado.*

Figura 2.6: *Comparação entre o histograma da imagem original e da mesma imagem com equalização de histograma.*

2.2.3 Etapa de Segmentação

Segundo [17] na etapa de *segmentação* é realizada a extração e identificação de áreas de interesse contidas na imagem, geralmente baseada na detecção de bordas ou regiões na imagem.

E de acordo com [18] uma segmentação bem sucedida aumenta as chances de sucesso na solução de problemas que requerem que os objetos sejam individualmente identificados, ou seja, quanto mais precisa for a segmentação, maiores são as chances de sucesso no

reconhecimento dos objetos.

Assim como em [11], este trabalho realiza a busca por um objeto circular, para tanto é necessário a utilização da detecção de bordas, explicada na próxima subseção.

2.2.3.1 Detecção de Bordas

Uma borda é o limite de duas regiões com níveis de cinza diferentes. Existem diversas técnicas para detecção de bordas, onde a maioria delas utiliza um operador local diferencial, quanto maior o valor absoluto do operador no local maior a chance de ser uma borda ou descontinuidade [17].

A derivada primeira é positiva nas transições da região escura para a clara, negativa nas transições da região clara para a escura e nula nas áreas de nível de cinza constante. Portanto o valor da derivada primeira pode ser utilizada na detecção de uma borda em uma imagem.

Como uma imagem é uma função bidimensional, a primeira derivada em qualquer ponto da imagem é obtida usando-se o valor do gradiente naquele ponto. Dessa forma o vetor gradiente $\nabla f(x, y)$ de uma imagem na posição (x, y) pode ser calculado pelas derivadas parciais, representado pela Equação 2.1 [17].

$$\nabla f(x, y) = \frac{\partial f(x, y)}{\partial x} i + \frac{\partial f(x, y)}{\partial y} j \tag{2.1}$$

Ou na forma matricial expresso pela Equação 2.2.

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \\ \dfrac{\partial f}{\partial y} \end{bmatrix} \tag{2.2}$$

Em detecção de bordas, o valor absoluto do vetor gradiente equivale à maior taxa de variação de $f(x, y)$, calculado de acordo com a Equação 2.3.

$$\nabla f = \sqrt{G_x^2 + G_y^2} = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{2.3}$$

Devido ao custo computacional necessário para resolver a Equação 2.3, o valor do gradiente pode ser aproximado pelo uso dos valores absolutos

$$\nabla f \approx |G_x| + |G_y| \tag{2.4}$$

ou do valor máximo entre os gradientes da direção x e y

$$\nabla f \approx max(|G_x|, |G_y|) \tag{2.5}$$

Uma mudança em intensidade pode ser detectada pela diferença entre os valores dos pixels vizinhos. Bordas *verticais* podem ser encontradas pela diferença *horizontal* entre os pontos vizinhos e vice-versa. Desta forma, seja uma região de uma imagem mostrada na Figura 2.7, onde os valores indicam os níveis de cinza dos pixels, o valor do gradiente, mostrado na Equação 2.3, pode ser aproximado no ponto (x, y) de várias maneiras, em que as mais usuais são *operador de Prewitt*, *operador de Sobel* e *operador de Kirsch*, em que todos utilizam máscaras 3×3 específicas em cada direção e determinam o valor do gradiente de forma aproximada, representado pelas Equações 2.4 e 2.5.

f(x-1,y-1)	*f(x,y-1)*	*f(x+1,y-1)*
f(x-1,y)	*f(x,y)*	*f(x+1,y)*
f(x-1,y+1)	*f(x,y+1)*	*f(x+1,y+1)*

Figura 2.7: *Região da imagem formada por* 3×3 *pixels.*

Além dos operadores para detecção de bordas citados acima existe um em especial, o detector de *Bordas de Canny*, que procura otimizar a localização de pontos de borda na presença de ruido. Embora o algoritmo seja mais complexo, o desempenho deste detector

é, em geral, superior aos acima citados, e é baseado em três objetivos básicos [17]:

- *Baixa taxa de erro.* As bordas detectadas devem ser o mais próximas possível das bordas verdadeiras.
- *Os pontos de borda devem estar bem localizados.* A distância entre um ponto marcado como uma borda e o centro da borda verdadeira deve ser mínima.
- *Resposta de um único ponto de borda.* O número de máximos locais em torno da borda verdadeira deve ser mínimo.

Para que esses objetivos sejam satisfeitos, inicialmente, a imagem é suavizada por meio de um filtro Gaussiano, em seguida o valor e a direção do gradiente são calculados, após este cálculo a borda é localizada tomando-se apenas os pontos cuja magnitude seja localmente máxima na direção do gradiente.

A Figura 2.8 representa uma imagem de bordas utilizando o método de Canny.

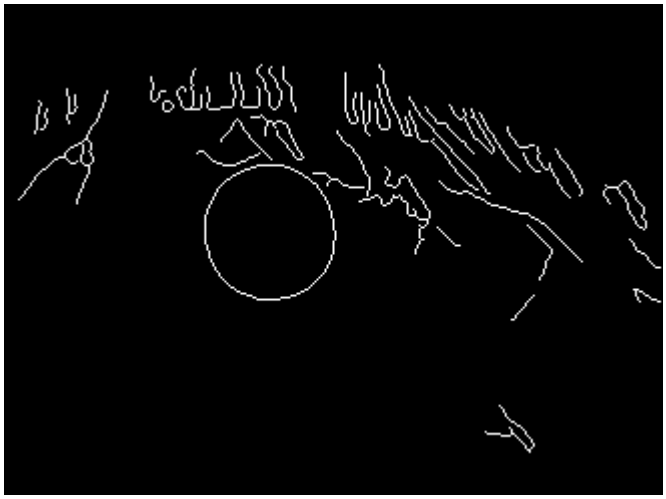


Figura 2.8: *Imagem de borda da Figura 2.5 utilizando o método Canny.*

Dessa forma, é conquistado os objetivos acima citados, alcançando uma detecção de bordas precisa para passar para o próximo passo, o reconhecimento.

2.2.4 Etapa de Reconhecimento

De acordo com a Figura 2.1, a etapa de *reconhecimento* é a última etapa para se chegar ao *resultado* desejado. E é nesta etapa que se vê a necessidade de todas as etapas anteriores, em especial da segmentação, pois se o objeto que se deseja fazer o reconhecimento não estiver devidamente destacado, esta etapa de reconhecimento, não obterá êxito ou irá gastar muito tempo de processamento para encontrar o objeto desejado.

O reconhecimento pode ser realizado através de várias propriedades da imagem, tais como forma, posição, cor, textura e assim por diante [11]. No presente trabalho o objetivo é rastrear a pupila que pode ser aproximada por um formato circular, e possui uma coloração mais escura, próxima do preto, portanto o reconhecimento pode ser realizado tanto por correspondência de forma quanto por cor, porém na imagem podem aparecer outros objetos com a coloração escura, como sombras ou os cílios da pessoa, o que torna o reconhecimento pela forma mais preciso.

Deste modo, se o objetivo é encontrar um objeto através da forma, é possível utilizar várias ferramentas para isto, porém a que resulta em uma maior precisão e possibilidade de escolha dos resultados é a Transformada de Hough, detalhada adiante.

2.2.4.1 Transformada de Hough

A Transformada de Hough (TH) (Hough, 1962) é uma técnica que localiza formas em imagens digitais. Em particular ela é usada na extração de linhas, círculos e elipses. No caso de linhas sua definição matemática é equivalente a Transformada de Radon. A TH foi introduzida por Hough e usada para encontrar sinais de bolhas. Todavia, Rosenfeld notou seu potencial no processamento de imagens (Rosenfeld, 1969).

A principal vantagem da TH está no fato dela ser bem mais rápida que o *template matching*. Sua implementação define um mapeamento dos pontos da imagem em um espaço acumulador (*Hough space*) ou *Matriz de votos*. O mapeamento é alcançado de maneira eficiente baseado na função que descreve a forma procurada. Esse mapeamento requer muito menos esforço computacional que o *template matching*. Todavia, a TH ainda

requer considerável capacidade de armazenamento e processamento [20].

2.2.4.2 Transformada de Hough para linhas

Em uma parametrização cartesiana, pontos colineares em uma imagem com coordenadas (x, y) estão relacionados por sua inclinação m e seu ponto de intersecção c com o eixo y de acordo com a Equação (2.6).

$$y = mx + c \tag{2.6}$$

A Equação (2.6) pode ser escrita na forma homogênea conforme a Equação (2.7).

$$Ay + Bx + 1 = 0 \tag{2.7}$$

onde:
$A = \frac{-1}{c}$;
$B = \frac{m}{c}$.
Então uma linha pode ser definida considerando um par de valores (A, B).

A Equação (2.7) pode ser vista como um sistema de equações simplesmente reescritas em termos de parametrização cartesiana como mostra a Equação (2.8).

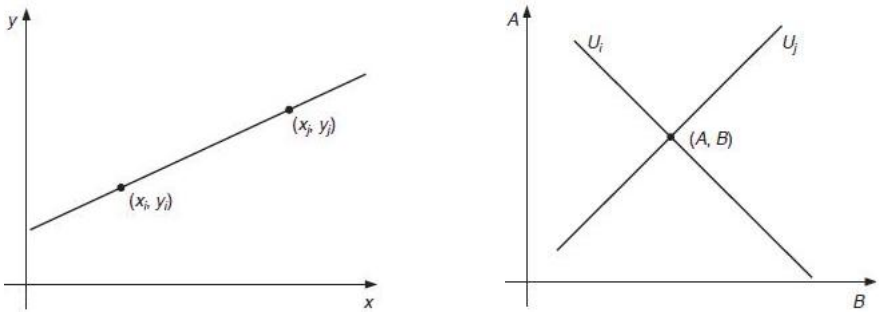
$$c = -x_i m + y_i \tag{2.8}$$

Logo, para determinar uma linha deve-se encontrar os parâmetros (A, B) ou (m, c) que satisfaça as Equações (2.7) e (2.8) respectivamente. Pode-se observar que o sistema é geralmente sobredeterminado, isto é, existem mais equações do que incógnitas. Então deve-se encontrar a solução mais próxima que satisfaça todas as equações simultaneamente.

A relação entre um ponto (x_i, y_i) em uma imagem e a linha da Equação (2.8) é ilustrada na Figura 2.9. Os pontos (x_i, y_i) e (x_j, y_j) na Figura 2.9(a) definem as linhas U_i e U_j na Figura 2.9(b), respectivamente. O ponto (A, B) no espaço de parâmetros é comum a essas

duas retas associadas aos pontos (x_i, y_i) e (x_j, y_j). Na verdade, todos os pontos colineares em uma imagem interceptam-se no mesmo ponto concorrente (A, B) independente da parametrização da linha utilizada.

A TH resolve esse sistema de uma forma muito eficiente simplesmente armazenando as soluções potenciais em uma matriz acumuladora (matriz de votos). O contador é feito traçando todas as linhas para cada ponto (x_i, y_i). Cada traço incrementa um elemento na matriz e então o problema de extração de uma linha torna-se um problema de localização de um máximo no espaço acumulador. Essa estratégia é robusta e tem demonstrado ser eficiente na presença de ruído e oclusões.



(a) *Imagem contendo uma linha.* (b) *Linhas no espaço dual.*

Figura 2.9: *Transformada de Hough para linhas.*

Os eixos no espaço dual representam os parâmetros da linha. No caso de parametrização cartesiana m pode assumir uma gama infinita de valores uma vez que as linhas podem variar de horizontal para vertical. Uma vez que os votos são armazenados na matriz é possível considerar as técnicas de *antialiasing* e melhorar a estratégia de armazenamento.

A magnitude do pico é proporcional ao número de pixeis na linha gerada, de acordo com a Figura 2.10, adaptada de [21]. As bordas da chave nas Figuras 2.10(b) e 2.10(c) definem duas linhas principais. A imagem da Figura 2.10(c), por conter mais ruído, apresenta um número maior de votos, porém o valor de pico encontra-se no mesmo lugar.

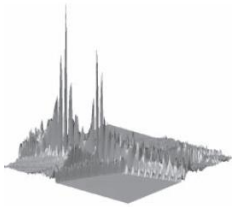
Pode-se verificar nas Figuras 2.10(g), 2.10(h) e 2.10(i) que a TH fornece uma estimativa correta dos parâmetros que especificam as linhas. Todavia, o problema de não linearidade dos parâmetros e a discretização produzem ruído no acumulador.

(a) *Linha.*

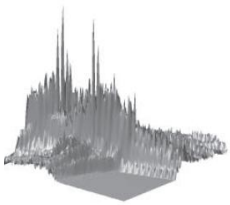
(b) *Chave.*

(c) *Chave e ruído.*

(d) *Acumulador da linha.*

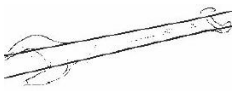


(e) *Acumulador da chave.*



(f) *Acumulador da chave com ruído.*

(g) *Linha do acumulador.*



(h) *Linhas do acumulador.*

(i) *Linhas do acumulador.*

Figura 2.10: *Aplicação da TH para linhas.*

Um dos principais problemas na execução da TH é a definição de um espaço acumulador adequado. Por outro lado seu custo computacional depende do número de pontos de bordas (n_e) e do comprimento das linhas formadas no espaço de parâmetros (l), com um custo de $O(n_e l)$, consideravelmente menor do que o custo do *Template matching* [20].

Uma alternativa para evitar os problemas da parametrização cartesiana na TH é basear a função de mapeamento em uma parametrização alternativa. Uma das técnicas mais comprovadas é a chamada "Parametrização Pé Normal" (*foot-of-normal*) que determina uma linha considerando um ponto (x, y) como uma função perpendicular à linha que passa pela origem da imagem. Esta forma também é conhecida como TH Polar.

O ponto onde essa linha intercepta a linha na imagem é obtido pela Equação (2.9).

$$\rho = x\cos(\theta) + y\,\mathrm{sen}(\theta) \tag{2.9}$$

onde:

θ - ângulo da linha normal à linha da imagem;

ρ - distância entre a origem e o ponto onde as linhas se cruzam, como ilustrado na Figura 2.11, adaptada de [21].

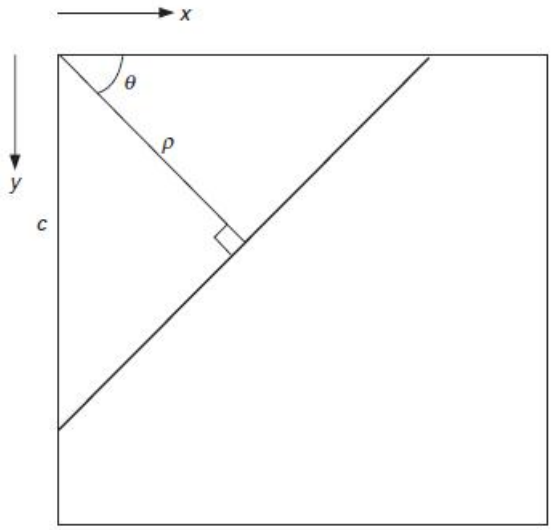


Figura 2.11: *Linha na forma polar.*

Lembrando que duas linhas são perpendiculares se o produto de suas inclinações m é -1, e considerando a geometria do arranjo da Figura 2.11, obtem-se a Equação (2.10).

$$c = \frac{\rho}{\mathrm{sen}(\theta)} \tag{2.10a}$$

$$m = -\frac{1}{\tan(\theta)} \tag{2.10b}$$

Assim, tem-se uma função de mapeamento diferente onde os votos são expressos de

forma senoidal em um acumulador 2D em termos de ρ e θ.

A vantagem desse mapeamento alternativo é que os valores dos parâmetros ρ e θ são limitados. A variação de ρ está dentro de 180° e seus possíveis valores são dados pelo tamanho da imagem, pois o comprimento máximo da linha é de $\sqrt{M^2 + N^2}$, onde M e N são os lados da imagem.

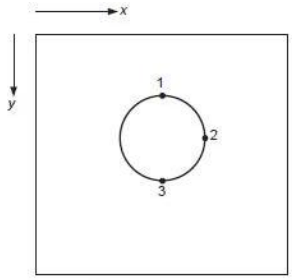
2.2.4.3 Transformada de Hough para Círculos

A TH pode ser estendida introduzindo a equação da curva no processo de detecção. Esta equação pode ser considerada na forma explicita ou paramétrica. De forma explicita a TH pode ser definida considerando a equação de um circulo conforme a Equação (2.11).

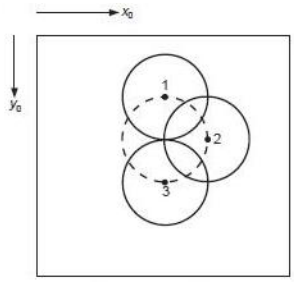
$$(x - x_0)^2 + (y - y_0)^2 = r^2 \tag{2.11}$$

A Equação (2.11) define um conjunto de pontos (x, y), centrados na região (x_0, y_0) e raio r. Essa equação, novamente, pode ser vista de duas maneiras: como um locus de pontos (x, y) em uma imagem, ou como um conjunto de pontos (x_0, y_0) centrados em (x, y) com raio r .

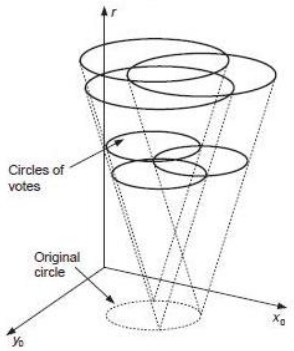
A Figura 2.12, adaptada de [21], ilustra as duas definições. Cada ponto da borda define um conjunto de círculos no espaço do acumulador. Estes círculos são definidos para todos os valores possíveis do raio e são centrados nas coordenadas do ponto de borda. A Figura 2.12(b) mostra três círculos definidos por três pontos localizados na borda. Esses círculos são dados para um determinado valor de r. Na verdade, cada ponto da borda define círculos para os outros valores do raio. Isto implica que o espaço acumulador é tridimensional e que os pontos de borda são mapeados em um cone no espaço acumulador. A Figura 2.12(c) ilustra esse acumulador. Depois de reunir todos os pontos de borda em evidência o máximo no espaço acumulador corresponde aos parâmetros do círculo original da imagem. O procedimento é o mesmo descrito para a TH de linha, porém os votos são gerados pela Equação (2.11).



(a) *Imagem com um círculo.*



(b) *Espaço acumulador.*



(c) *Espaço acumulador 3D.*

Figura 2.12: *Transformada de Hough para círculos.*

A Equação (2.11) pode ser definida na forma paramétrica, como mostrado na Equação (2.12).

$$x = x_0 + r\cos(\theta) \tag{2.12a}$$

$$y = y_0 + r\,\mathrm{sen}(\theta) \tag{2.12b}$$

Observe que θ não é um parâmetro livre, mas define o traço da curva. O traço da curva (ou superfície) é comumente referenciado como função de espalhamento pontual.

Essas equações definem pontos no espaço acumulador dependente do raio r. A vantagem dessa representação é que ela permite ser resolvida por parâmetros. Então a TH mapeada

é definida pela Equação (2.13).

$$x_0 = x - r\cos(\theta) \tag{2.13a}$$

$$y_0 = y - r\,\mathrm{sen}(\theta) \tag{2.13b}$$

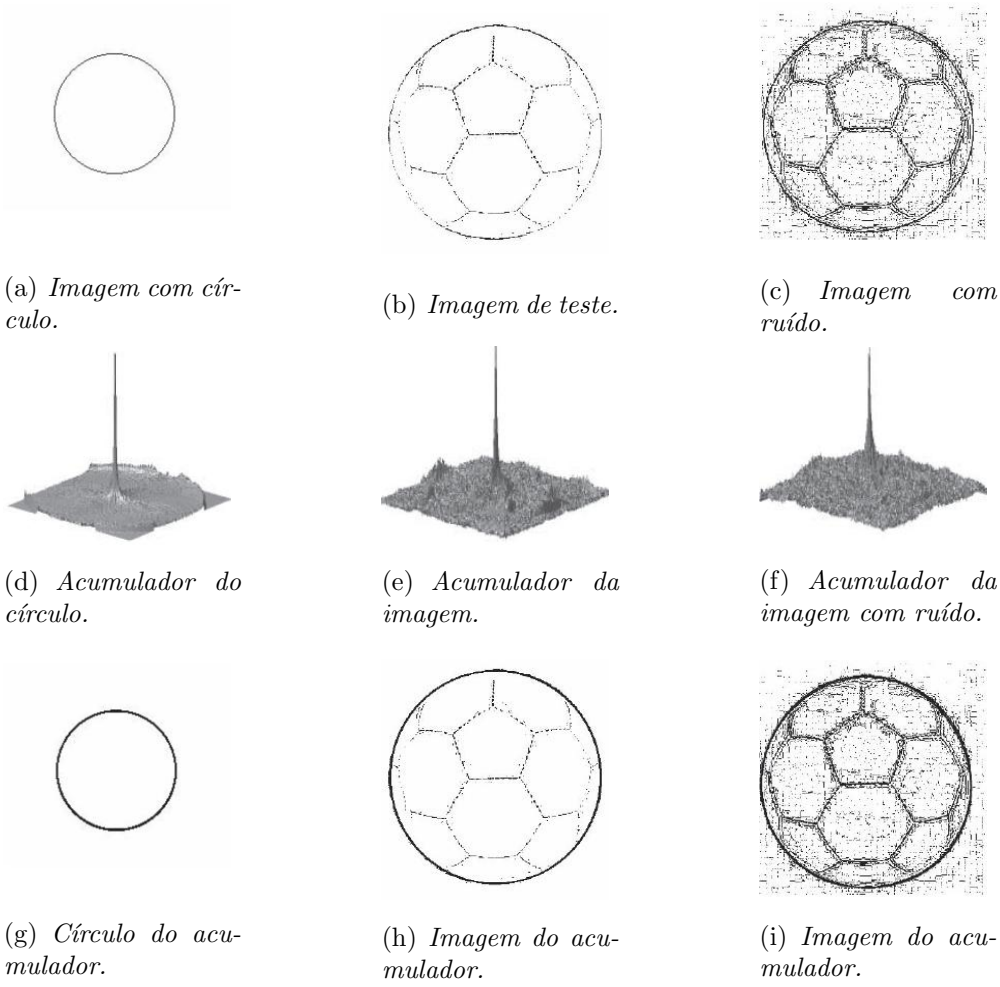
O acumulador é na verdade 2D, em termos dos parâmetros do centro e para um raio fornecido como argumento fixo da função. Um círculo é formado variando θ de 0° a 360°. A discretização de θ controla o efeito da granulação. Um incremento pequeno cobre o espaço de parâmetros de forma mais suave, muito grande resulta em uma cobertura esparsa. O espaço acumulador, inicialmente zero, é incrementado somente para pontos cujas coordenadas situam-se dentro do intervalo especificado.

A aplicação para a TH do círculo é ilustrada na Figura 2.13, adaptada de [21]. Nesta figura as bordas são completamente bem definidas. O pico do acumulador está no centro do círculo, além disso, nota-se que o algoritmo é capaz de resolver problemas de oclusão e ruído.

Outro exemplo de extração de círculos pela TH é mostrado na Figura 2.14, adaptada de [21]. Esta figura corresponde a uma imagem real de baixa resolução pré-processada pelo detector de bordas de Sobel. O raio, parâmetro de entrada do algoritmo, foi fixado em 5 pixeis. Pode-se verificar que a TH encontra círculos com um número máximo de pontos; é possível incluir outras constantes de controle, como por exemplo, gradiente de direção para objetos com um perfil de iluminação.

2.2.4.4 Como Desenhar o Círculo no Espaço Acumulador

Um círculo pode ser desenhado no espaço acumulador diretamente das equações em (2.13), neste caso, existe um problema em relação a como selecionar os valores discretos ou a resolução apropriada de θ. Uma solução consiste em utilizar uma alta resolução de



(a) *Imagem com círculo.*

(b) *Imagem de teste.*

(c) *Imagem com ruído.*

(d) *Acumulador do círculo.*

(e) *Acumulador da imagem.*

(f) *Acumulador da imagem com ruído.*

(g) *Círculo do acumulador.*

(h) *Imagem do acumulador.*

(i) *Imagem do acumulador.*

Figura 2.13: *Aplicação da TH para círculos.*

θ e, em seguida, arredondar os valores de *sen* e *cos*, o que pode resultar em uma grande sobreposição de valores (um pixel da borda sendo desenhado mais de uma vez) ou em uma falta de pixels, se o raio for grande [22].

Para garantir que todos os pixeis de borda sejam desenhados, pode ser selecionado uma resolução de θ muito alta, porém isto também irá aumentar o custo computacional. Uma forma de reduzir o custo computacional seria pré-calcular os valores de *sen* e *cos* e usar uma tabela de pesquisa.

Embora a solução acima funcione, existe uma outra forma de solucionar este problema. Fazer a substituição das equações em (2.13) pelo Algoritmo de Bresenham para desenhar

(a) *Imagem do olho.* (b) *Bordas de Sobel.* (c) *TH para círculos.*

Figura 2.14: *Uso da TH para círculos.*

os círculos no *Espaço Acumulador* [23]. O algoritmo de Bresenham foi projetado para desenhar linhas e círculos em monitores digitais, sem sobreposição de valores e sem espaços vazios, sem preenchimento, e é, portanto, ideal também para este contexto, diminuindo o custo computacional por somente utilizar comparações e somas algébricas de números inteiros.

O Algoritmo de Bresenham para círculos utiliza a simetria de uma circunferência para diminuir o tempo de processamento, conforme visto na Figura 2.15 adaptada de [24], um círculo pode ser dividido em octantes e a partir das coordenadas de um ponto de um octante é possível encontrar as coordenadas dos pontos dos outros octantes, conforme mostrado também na Figura 2.15.

Limitando a imagem ao primeiro octante e pegando um ponto conhecido A, conforme mostrado na Figura 2.16, adaptada de [24], com certeza o próximo ponto B estaria um pixel acima do ponto A e poderia estar no mesmo alinhamento vertical ou um pixel à esquerda.

A escolha correta deste próximo ponto, B_1 ou B_2, é feita através da equação da circunferência definida pela Equação (2.14),

$$F(x, y) = x^2 + y^2 - r^2 \tag{2.14}$$

em que se $F(x, y) > 0$ (x, y) é um ponto fora da circunferência, se $F(x, y) < 0$ (x, y) é um ponto dentro da circunferência e se $F(x, y) = 0$ (x, y) é um ponto em cima da

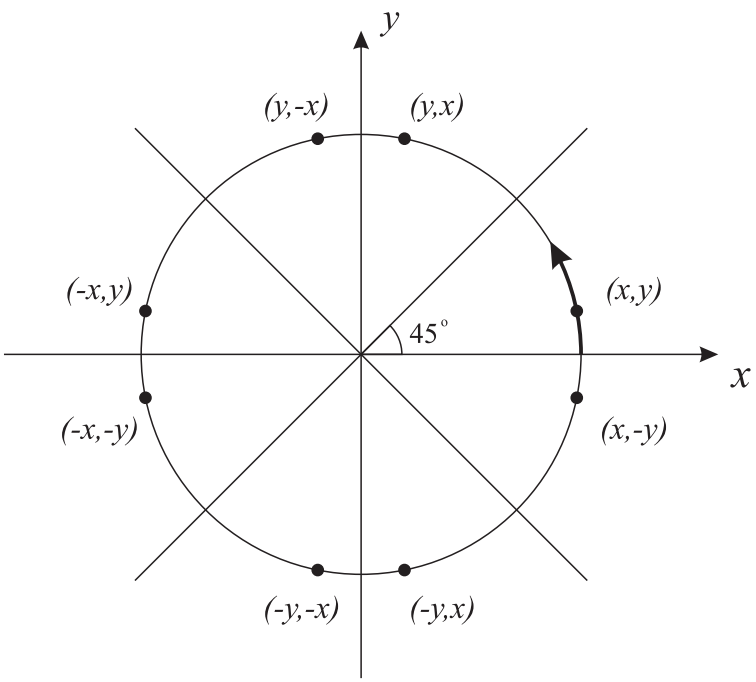


Figura 2.15: *Simetria de pontos em uma circunferência.*

circunferência, desta forma o próximo ponto é aquele que está mais próximo de $F(x, y) = 0$, ou seja, se $|F(x_{B_1}, y_{B_1})| < |F(x_{B_2}, y_{B_2})|$ então o próximo ponto da circunferência será B_1 cujas coordenadas são:

$x_{B_1} = x_A$ e

$y_{B_1} = y_A + 1$

Já caso $|F(x_{B_2}, y_{B_2})| < |F(x_{B_1}, y_{B_1})|$ o próximo ponto da circunferência será B_2 cujas coordenadas são:

$x_{B_2} = x_A - 1$ e

$y_{B_2} = y_A + 1$

Todo este processo é repetido até que se chegue ao ângulo de 45 graus, ou seja, as coordenadas do ponto são iguais ($x_B = y_B$).

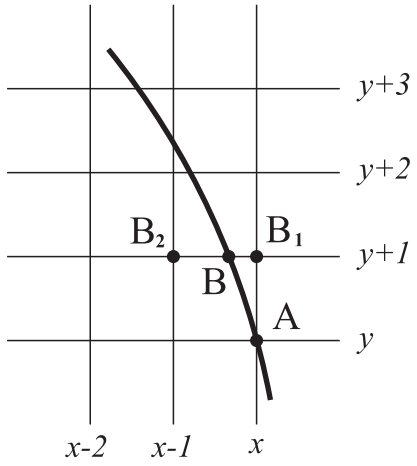


Figura 2.16: *Possíveis pixels para a escolha do próximo ponto da circunferência.*

2.3 Considerações Finais do Capítulo

Este Capítulo apresentou técnicas utilizadas em um sistema de processamento de imagens, desde a aquisição da imagem até o reconhecimento de algum objeto da mesma, salientando que todas essas técnicas foram utilizadas neste trabalho.

De forma que este capítulo é um pré requisito para o próximo que será de Processamento de Vídeos Digitais, que nada mais é do que um processamento de Imagens Digitais sequenciais.

3 PROCESSAMENTO DE VÍDEOS DIGITAIS

3.1 Introdução

Processamento de video digital é o estudo de algoritmos para processamento de imagens em movimento, que estão representadas no formato digital, discutidas no Capítulo 2. Lembrando que há uma distinção entre *video digital* e *imagens digitais*, que são imagens que não se alteram ao longo do tempo [25].

Um video digital é uma imagem em movimento, desta forma, sendo um video dinâmico, o conteúdo visual depende do tempo, e geralmente contém movimentação ou alteração de objetos. Portanto é importante perceber que vídeos digitais são sinais de três dimensões, duas no espaço e uma no tempo, representado pela Figura 3.1 extraída de [25].

A disponibilidade de uma sequência de imagens permite a medida de algumas grandezas, tais como variação da intensidade de pixels, movimento da câmera e profundidade, e detecção e rastreamento de objetos em movimento. Por sua vez, o processamento de imagens sequenciais requer o desenvolvimento de técnicas sofisticadas para extrair essa informação. Com a recente disponibilidade de computadores poderosos e acessíveis, sistemas de armazenamento de dados, e dispositivos de aquisição de imagem, análise de imagens sequenciais fez a transição para uma área de prática com aplicações significativas em diversos segmentos.

São várias aplicações que utilizam o processamento de vídeos digitais [25], inclusive

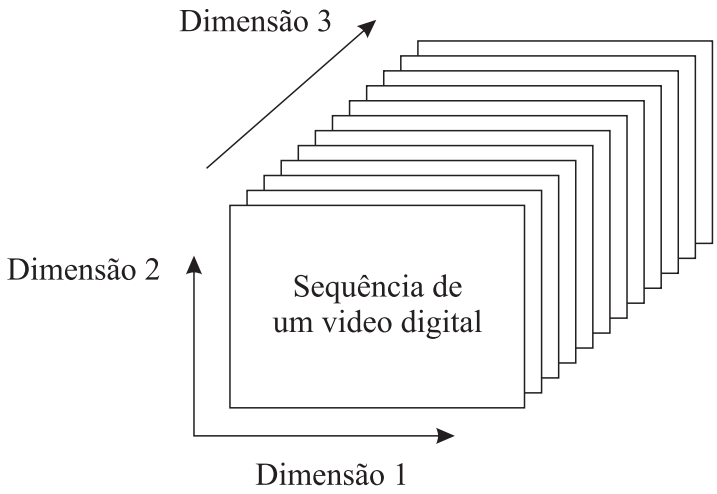


Figura 3.1: *Dimensões de um vídeo digital.*

alguns exigem o processamento em tempo real, tais como:

- Vídeoconferência utilizando softwares modernos em um computador pessoal.
- Vídeo de Medicina, onde os processos dinâmicos do corpo humano, tais como o movimento do coração, pode ser visto em tempo real.
- Comunicação em vídeo, que podem ser feitas através do aparelho celular.
- Vídeo multimídia, em que o vídeo é associado a gráficos, palavras entre outros, possibilitando uma experiência diferenciada.
- Cinema digital, não apenas na elaboração de filmes animados, mas para a entrega dos filmes aos cinemas em formato digital.
- Codificação e compressão de vídeo para minimizar o espaço de armazenamento e agilizar a transmissão [26].
- Rastreamento de objetos e/ou pessoas para sistemas internos de TV [26]

Após uma breve explicação das utilidades de um processamento de vídeo, voltando para o objetivo do presente trabalho, várias são as técnicas e possibilidades para utilização do rastreamento em vídeo, e serão discutidas nas seções seguintes.

3.2 Rastreamento em Vídeo

O rastreamento consiste em estabelecer relações temporais entre objetos alvo de quadros de vídeo consecutivos, isto é, identificar, no quadro corrente de um vídeo, a localização dos objetos alvo detectados no quadro anterior.

Video tracking têm recebido grande atenção nos últimos anos, principalmente devido a variedade de aplicações, como por exemplo, interfaces homem máquina, sistemas de segurança, realidade virtual e codificação de vídeo. Qualquer que seja a aplicação o resultado final depende da representação utilizada para descrever o alvo rastreado, podendo nesse caso ser um contorno do objeto, um conjunto de coordenadas 2D do seu centro de massa, sua posição 3D e assim por diante [25].

Alguns sistemas de *tracking* utilizam sensores ou transmissores para marcar o objeto a ser rastreado e são chamados de *Active Object Tracking*. No entanto, estes são considerados intrusivos e adequados para ambientes controlados o que faz do rastreamento passivo mais desejável, apesar deste ser bem mais difícil de promover.

No *tracking* passivo, objeto deste trabalho, alguns critérios de representação podem ser utilizados para a detecção - forma, posição, cor, textura e assim por diante.

O desafio torna-se ainda maior nos casos em que o objeto a ser rastreado possui estrutura deformável e padrões de cor variáveis, como no caso da íris e pupila. Além disso, o movimento da pupila pode ser afetado por fatores inesperados que usualmente impossibilitam qualquer tipo de predição a partir de imagens. A pupila reage muito rapidamente a variações na iluminação (na ordem de milissegundos para a contração).

3.2.1 Rastreamento de Objetos Baseado em Regiões

O rastreamento de objetos baseado em regiões é geralmente uma maneira eficiente de interpretar e analisar o movimento observado em uma sequência de vídeo. Uma região da imagem pode ser definida como um conjunto de pixels com características homogêneas. Ela pode ser obtida por segmentação de imagens, podendo ser baseada em diferentes caracte-

rísticas do objeto (cor, textura, bordas) e/ou sobre o movimento observado nos quadros da sequência de vídeo.

A informação de cor mostra ser uma alternativa efetiva para o rastreamento de objetos baseado em região de interesse por tornar o processamento mais rápido em tempo real, isto é, 20 a 30 *frames* por segundo (FPS) [27].

O maior problema da segmentação e do *tracking* baseados em cor refere-se à sua baixa eficiência contra variações de iluminação principalmente em cenas ao ar livre [27].

Um modo de se obter certo grau de invariabilidade de iluminação é usando apenas os valores de cromaticidade do espaço de cor HSV (*Hue-Saturation-Value*), ou seja, utilizar apenas as componentes H e S [27].

A Figura 3.2, adaptada de [28], ilustra uma segmentação a partir da modelagem eficiente da cor do objeto. A modelagem correta permite que o objeto seja diferenciado de outros objetos presentes na cena e também do fundo da imagem.

(a) *Imagem original.* (b) *Imagem segmentada.*

Figura 3.2: *Exemplo de rastreamento baseado em espaço de cor.*

Um algoritmo de segmentação de cor pode ser concebido em três etapas: a escolha de um espaço de cor adequado, a modelagem do objeto sobre o espaço de cor selecionado, e o método utilizado para classificar os pixeis individuais do objeto e do não-objeto. Os métodos de modelagem de cor podem ser grosseiramente classificados como paramétricos (usando uma única Gaussiana ou uma mistura de Gaussianas) e não paramétrico (baseado em histograma, tais como as tabelas de pesquisa e os mapas de probabilidade de Bayes) [25], [27], [21].

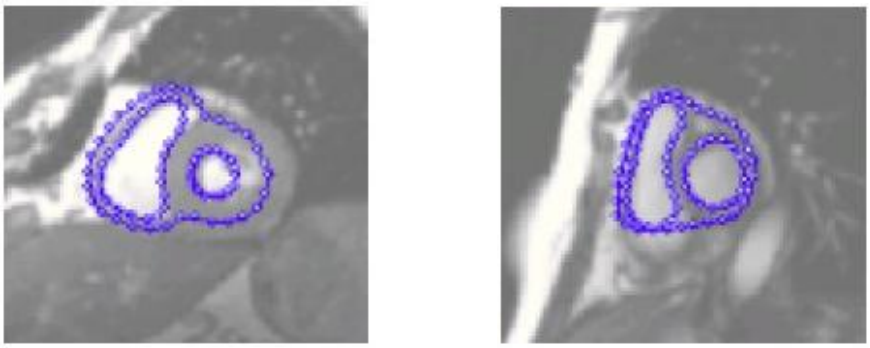
3.2.2 Rastreamento de Objetos Baseado em Contorno

Diversos trabalhos utilizam as informações de contorno para rastrear objetos ao longo do tempo e recuperar as informações de posição e forma do objeto ao logo da sequência de vídeo [29], [30].

Geralmente, as técnicas baseadas em contorno envolvem cálculos de distâncias, logo tais métodos de modelagem são mais complicados do que a modelagem por regiões, por exemplo, usando cores. No entanto, o rastreamento por contorno geralmente é mais robusto do que algoritmos baseados em rastreamento por região, pois ele pode ser adaptado para lidar com oclusão parcial e informações de contorno e é, na maioria dos casos, mais insensível à variação de iluminação [27].

Contornos ativos, também conhecido como *snakes*, têm sido amplamente utilizados por pesquisadores para realizar a segmentação de objetos e monitoramento. Um algoritmo de contorno ativo deforma dinamicamente o contorno para "casar"com as características da imagem tais como linhas, bordas, limites, e assim por diante. *Snakes* consistem em curvas paramétricas elásticas cuja deformação é sujeita à forças internas (contorno de forças elásticas) e forças externas (devido ao conteúdo da imagem e outras restrições) [25].

A Figura 3.3, adaptada de [31], ilustra a utilização de uma *Snake* em um procedimento de segmentação em ressonância magnética cardíaca.



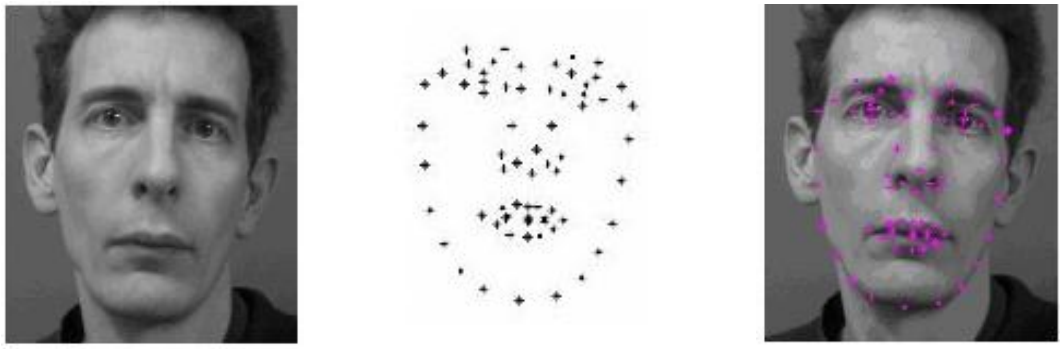
(a) *Segmentação em ressonância magnética cardíaca.*

(b) *Ajuste do contorno (snake) à borda.*

Figura 3.3: *Exemplo de rastreamento baseado em contorno ativo.*

3.2.3 Rastreamento de Objetos Baseado em Pontos Característicos

Conforme [25], o rastreamento baseado em pontos característicos pode ser definido como a tentativa de recuperar os parâmetros de movimento de um ponto em uma sequência de vídeo, mais precisamente os parâmetros associados com a translação do ponto no plano, pois pontos no espaço 2D não giram nem transladam com respeito à profundidade. Mais formalmente, $A = A_0, A_1, \ldots, A_{n-1}$ denota os N quadros em uma sequência de vídeo e $\mathbf{m}_i(x_i, y_i), i = 0 \ldots N-1$ é a posição do ponto característico nesses quadros. A tarefa é determinar um vetor de movimento $\mathbf{d}_i(d_{x,i}, d_{y,i})$ que melhor determina a posição do ponto característico no próximo *frame*, $\mathbf{m}_{i+1}(x_{i+1}, y_{i+1})$ que é $\mathbf{m}_{i+1} = \mathbf{m}_i + \mathbf{d}_i$. O objeto a ser rastreado é geralmente definido por uma "caixa"delimitadora ou um envoltório convexo dos pontos rastreados, conforme a Figura 3.4, adaptada de [32].



(a) *Imagem original.* (b) *Pontos chave.* (c) *Imagem etiquetada.*

Figura 3.4: *Exemplo de rastreamento baseado em pontos característicos.*

Um dos problemas típicos encontrado em rastreamentos baseado em pontos característicos pode ser a detecção de pontos atípicos individuais, contudo algoritmos para essa abordagem podem ser construídos de forma muito eficiente se for considerado pontos com características salientes [33].

Esses pontos podem ser, por exemplo, aqueles associados com informações de alta curvatura local, isto é, cantos e bordas. A invariância da vizinhança do ponto (ou descritor

correspondente) às variações de iluminação e alterações de ponto de vista corresponde a outra característica desejável. Muitos pontos característicos (também chamados de pontos chave) têm sido propostos na literatura e para maiores detalhes [25], [32].

3.2.4 Rastreamento de Objetos Baseado em Modelos

Vários pesquisadores utilizam técnicas baseadas em modelos para rastreamento de objetos em 2D [25], [21], [34]. Esse tipo abordagem está relacionado com o rastreamento baseado em região porque um modelo é essencialmente uma forma da região a ser rastreada na imagem.

Por razões práticas é necessário conhecer o modelo a ser buscado nas imagens (etapa de inicialização). Por exemplo, os modelos podem ser específicos, ou seja, uma instância de uma classe de objetos ou podem ser um modelo estatístico extraído de uma base de dados.

A correspondência de modelo *Template Matching* pode ser definida como o processo de busca da imagem alvo para determinar a região da imagem que se assemelha ao modelo, baseado em uma medida de similaridade ou distância. Essencialmente, a região do modelo deve passar por uma transformação geométrica que minimizaria a distância da medida usada no processo de comparação. O objetivo de um algoritmo de correspondência de modelo é estimar os parâmetros de tal transformação *off-line* [25], [34].

3.3 Método de Rastreamento Utilizado

Os conhecimentos prévios sobre o objeto podem ser utilizados como informação para simplificar o rastreamento [35].

Como o objetivo do presente trabalho é efetuar o rastreamento de informações do objeto, no caso tamanho e posição da pupila, em tempo real, é necessário reduzir ao máximo informações desnecessárias na imagem. Por isso, ao encontrar a posição e o tamanho da pupila em um quadro, no próximo quadro tem grande chance de a pupila estar bem próxima

do local encontrado anteriormente, portanto não há a necessidade de efetuar a busca por toda a imagem, mas somente próximo ao local encontrado no quadro anterior.

O mesmo ocorre para o tamanho, ou seja, a diferença de tamanho para a pupila encontrada em dois quadros consecutivos é muito pequena, portanto seria perda de tempo efetuar a busca por pupilas com tamanhos muitos diferentes do que o tamanho encontrado no quadro anterior.

Assim, para economizar tempo no processamento e obter as informações desejadas em tempo real, posição e tamanho da pupila, a imagem de um quadro é cortada em um tamanho próximo ao tamanho da pupila encontrada no quadro anterior, com a mesma posição, e o parâmetro utilizado para busca, raio da pupila a ser encontrada, também próximo ao encontrado no quadro anterior.

Conforme será mostrado no Capítulo 5 este método reduz drasticamente o tempo de processamento de cada frame, e conforme será visto, o espaço de busca também é reduzido, auxiliando para um melhor tempo de processamento.

3.4 Tratamento de Oclusões

Um dos maiores desafios dos algoritmos de rastreamento de objetos em sequências de imagens é a oclusão ou auto-oclusão do objeto [36]. Uma oclusão ocorre quando parte ou até mesmo todo o objeto rastreado não é visível em todos os *frames* da sequência. Por exemplo, essa situação pode ocorrer devido a presença de um ou mais objetos estáticos na cena ou devido a objetos com movimentos relativos opostos.

Dependendo da aplicação um sistema de rastreamento pode ignorar o tratamento de oclusões ou simplesmente tratá-lo como ruído no processo correspondente. No entanto, uma série de abordagens tenta lidar com oclusões de forma mais avançada.

Existem várias maneiras de lidar com as oclusões. Por exemplo, a utilização de sistemas de rastreamento redundantes que reinicia o processo de busca do objeto sempre que este estiver ocluído. Outra estratégia pode ser a divisão do objeto de interesse em partes

menores e rastreá-las. Obviamente, nesses sistemas o esforço computacional é maior.

Outros métodos podem ser empregados, como por exemplo, os métodos de filtragem estocástica, métodos baseados em pontos característicos como o *Corner finding* e métodos baseados em estimativas de velocidades como o *Optical flow* [37].

Com relação a sistemas de rastreamento com múltiplas câmeras o problema da oclusão se torna mais simples, pois um objeto ocluído em uma visão pode ser totalmente visível em outra. Esses sistemas combinam as informações de todas as câmeras ao determinar a "melhor vista". Quando uma câmera perde o alvo devido à oclusão, a informação é obtida a partir de outras câmeras de monitoramento que também estão rastreando o objeto [25].

No caso do presente trabalho uma oclusão ocorre quando a pessoa pisca, ocultando completa ou parcialmente a pupila, desta forma, para que o sistema não retorne ao estado inicial, de realizar o processamento em toda a imagem para buscar a pupila, gastando mais tempo, o sistema salva por até cinco imagens consecutivas os dados da pupila encontrada anteriormente.

3.5 Considerações Finais do Capítulo

Este capítulo apresenta importantes conceitos do rastreamento de objetos e tratamento de oclusões. Detalha-se, de forma sucinta, como o sistema criado realiza este rastreamento e como ele trata as possíveis oclusões.

O próximo capítulo faz a descrição do sistema proposto, colocando em prática um software que realiza o processamento de imagens e de vídeos descritos no Capítulo anterior e neste.

4 DESCRIÇÃO DO SISTEMA

4.1 Introdução

O Software foi desenvolvido no Microsoft Visual C# 2010 Express [38], desde a aquisição das imagens, passando por todo o processamento, até as informações adquiridas após o processamento do sistema.

Para verificar a eficiência e analisar precisão do sistema, foram efetuadas diversos testes envolvendo diferentes formas de aquisições de imagens, inclusive em imagens estáticas com o objetivo de comparar a precisão com outros softwares já existentes, conforme será detalhado no próximo Capítulo.

Já o processamento em si, que inclui o Pré-Processamento, a Segmentação e o Reconhecimento que foram explicados no Capítulo 2, seção 2.2, e ainda o Método de Rastreamento Utilizado no presente trabalho que foi explicado no Capítulo 3, seção 3.3, é comum a todos os testes. Sendo assim, essas partes do software serão explicadas nas seções subsequentes.

4.2 Descrição do Sistema

Para efetuar o processamento das imagens foi utilizada a biblioteca de processamento de imagens do conjunto de bibliotecas do AForge.NET versão 2.2.4 [39] e para que o sistema

seja capaz de efetuar todo este processamento é necessário que a imagem de entrada seja do tipo *bitmap*, classe pertencente à biblioteca pré-carregada do Visual C# .NET Framework 4 [40].

A seguir será explicado como foi efetuado o processamento, quais as classes e bibliotecas necessárias, dividido conforme as próximas subseções, primeiro o *Pré-Processamento e Segmentação*, seguido do *Reconhecimento* e por fim o *Método de Rastreamento Utilizado*.

4.2.1 Pré-Processamento e Segmentação

Para efetuar o Pré-Processamento e a Segmentação utilizou-se a classe *Filters*, da biblioteca *Imaging* da AForge.NET [41], para efetuar uma sequencia de filtros na imagem:

- *Grayscale.CommonAlgorithms.BT709* que faz a conversão da imagem de RGB para escala de cinza, seção 2.2.2;
- *HistogramEqualization*, que efetua a equalização do histograma, seção 2.2.2;
- *Threshold* que efetua a binarização da imagem de acordo com um limiar estabelecido empiricamente;
- *CannyEdgeDetector*, que efetua a busca de bordas pelo algoritmo de Canny, Figura 2.8.

4.2.2 Reconhecimento

Na fase de *Reconhecimento* foi efetuada a busca pela Pupila, e assumindo que esta possui um formato circular, foi efetuada a busca por um círculo, e para encontrar os círculos a classe *HoughCircleTransformation* da biblioteca *Imaging* da AForge.NET [41] foi utilizada como base para criar uma outra classe que utiliza menos tempo de processamento.

A principal diferença entre a classe original e a classe criada é que na primeira podem ser encontrados quantos círculos existirem, porém com um raio específico, e não um intervalo

definido de raios, ou seja, nesse caso a matriz de votos da TCH terá apenas duas dimensões, e não três, como descrito no Capítulo 2.

Além do espaço de busca, a classe retorna também as informações dos círculos encontrados, como Posição do Centro (X e Y), Raio, Intensidade e Intensidade Relativa, sendo que neste caso a Intensidade é o número de pixels que estão na borda deste círculo, ou seja, quanto mais pontos de borda tiver o círculo encontrado maior será a Intensidade, e a Intensidade Relativa é o número de pixels que estão na borda do círculo encontrado dividido pela Intensidade máxima encontrada.

Já na classe criada são efetuadas buscas por círculos com um intervalo definido de raios, de acordo com o parâmetro de entrada. A saída da classe possui apenas um círculo, já que o objetivo do sistema é encontrar a pupila, ou seja, somente um círculo deve ser encontrado, sendo que, no caso, foi considerado o que mais se aproxima de um círculo completo. Ainda nesta classe não é armazenada a matriz de votos, somente as informações do círculo mais provável encontrado, como Posição do Centro, Raio, Intensidade e Intensidade Relativa ao Raio, este último é a divisão da Intensidade pelo Raio, desta forma quanto maior for o valor mais completo está o círculo, que será exemplificado através da Figura 4.1.

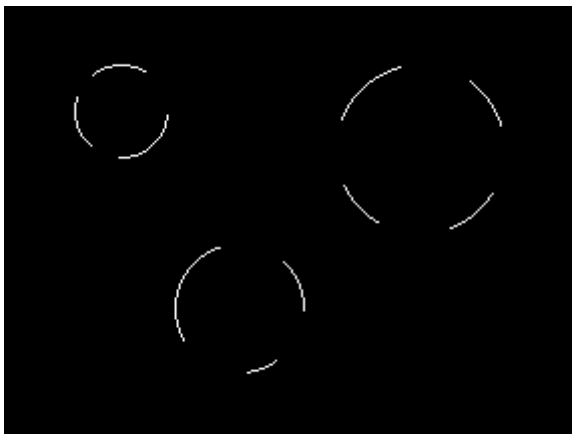


Figura 4.1: *Figura criada com o objetivo de demonstrar a diferença entre as classes (algoritmo) criada e original.*

A Figura 4.1 foi criada com o objetivo de mostrar a saída da classe original e explicar como ocorre a escolha do melhor círculo na classe criada. Esta figura contém três círculos

incompletos com tamanhos 23, 32 e 42 pixels de raio e centro em posições diferentes.

Utilizando a classe original da TCH da AForge.NET, e assumindo que os raios dos círculos são conhecidos, foi necessário utilizar a classe três vezes, uma para cada raio conhecido, as saídas da classe original para cada raio são explicadas abaixo:

- Saída para o raio de 23 pixels.

  A Figura 4.2 mostra o espaço de busca para um círculo de 23 pixels de raio, em que, assim como nos outros casos, o ponto mais brilhante é o centro do círculo encontrado com o raio de 23 pixels e em vermelho é a imagem original em que foi efetuada a busca.

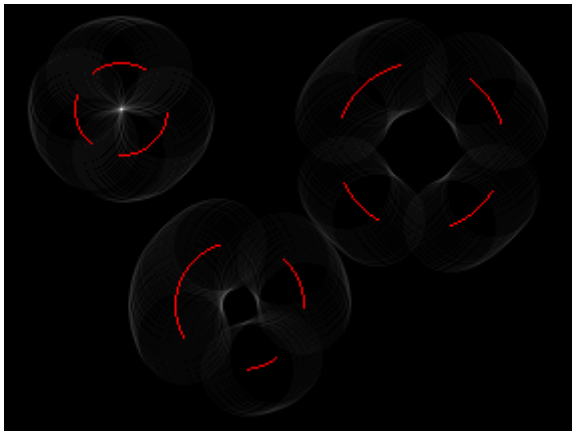


Figura 4.2: *Espaço de busca para o raio de 23 pixels.*

  Neste caso as informações de saída da classe original, pegando apenas o melhor círculo, foram:

  – Posição X: 58

  – Posição Y: 52

  – Intensidade: 84

  – Intensidade Relativa: 1

  Lembrando que, seguindo a teoria da Transformada de Hough foram encontrados outros círculos porém com menores Intensidades e Intensidades Relativas, sendo des-

cartados neste caso pois o Objetivo do presente Trabalho é encontrar apenas um círculo, o que possuir maior número de pontos na borda.

- Saída para o raio de 32 pixels.

  A Figura 4.3 mostra o espaço de busca para um círculo de 32 pixels de raio, em que, assim como nos outros casos, o ponto mais brilhante é o centro do círculo encontrado com o raio de 32 pixels e em vermelho é a imagem original em que foi efetuada a busca.

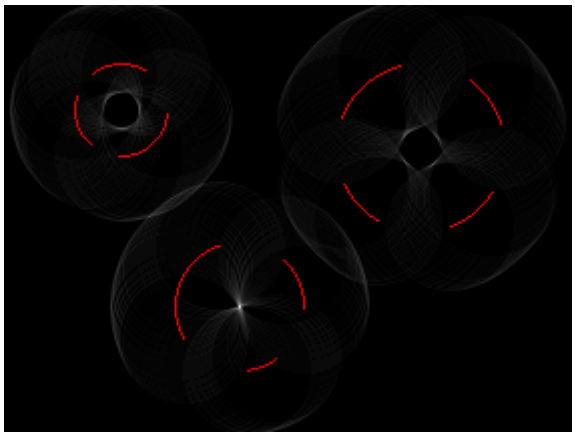


Figura 4.3: *Espaço de busca para o raio de 32 pixels.*

Neste caso as informações de saída da classe original, pegando apenas o melhor círculo, foram:

  - Posição X: 117
  - Posição Y: 150
  - Intensidade: 92
  - Intensidade Relativa: 1

Lembrando que assim como antes os outros círculos encontrados foram descartados por falta de interesse nos mesmos.

- Saída para o raio de 42 pixels.

A Figura 4.4 mostra o espaço de busca para um círculo de 42 pixels de raio, em que, assim como nos outros casos, o ponto mais brilhante é o centro do círculo encontrado com o raio de 42 pixels e em vermelho é a imagem original em que foi efetuada a busca.

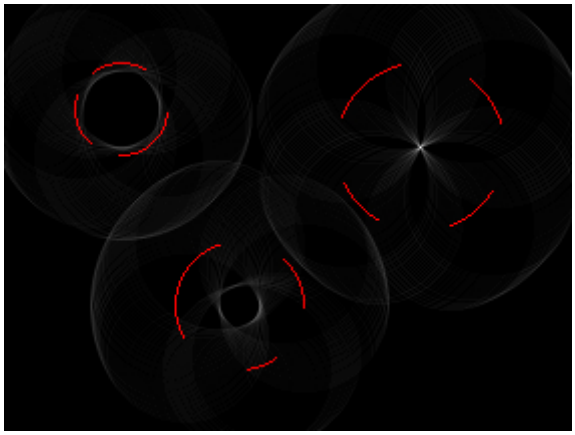


Figura 4.4: *Espaço de busca para o raio de 42 pixels.*

Neste caso as informações de saída da classe original, pegando apenas o melhor círculo, foram:

- Posição X: 207
- Posição Y: 71
- Intensidade: 102
- Intensidade Relativa: 1

Novamente, assim como nos dois casos anteriores houveram outros círculos encontrados porém também descartados.

A saída da classe original é para apenas um círculo, portanto para encontrar um círculo com um raio desconhecido é necessário utilizá-la várias vezes e salvar a informação para cada raio de busca, e apenas após esse processo, utilizar uma função para estabelecer qual o círculo ideal dentre as possibilidades encontradas.

Caso fosse realizada uma análise superficial da saída dessa classe para cada um dos raios acima, não seria suficiente escolher o melhor círculo pela Intensidade Relativa, pois todos possuem o valor 1 (um), daí poderia ser escolhido a intensidade, que no caso acima o círculo com o raio de 42 possui a maior intensidade.

Por isso, foi criada uma classe baseada na classe original da TCH da AForge.NET, em que, ao invés de se ter a necessidade de utilizar a classe original várias vezes, consumindo mais tempo no processamento, é informado previamente um intervalo de raios para ser efetuada a busca. Ainda nessa classe criada, a saída não é mais a mesma da classe original, na qual haveria a necessidade de se guardar um círculo para cada raio com grande consumo de memória e tempo. Nessa fase será armazenado apenas o melhor círculo, dessa forma, sempre que um círculo melhor que o anterior é encontrado, esse último é substituído.

O melhor círculo, no caso da classe criada, é dado pela sua Intensidade Relativa ao Raio, valor dado pela Intensidade dividida pelo Raio, podendo ser feita uma analogia com o comprimento de um semi-círculo relativo ao Raio, que seria o ângulo deste semi-círculo em radianos. Dessa forma, quanto maior a Intensidade Relativa ao Raio, mais completo é o círculo, sendo que o valor máximo desta variável ocorre quando o círculo é completo, ou seja, não possui falhas.

Utilizando a classe criada, a saída para o intervalo de raios de 22 até 43 é um círculo com as características:

- Raio: 23
- Posição X: 58
- Posição Y: 52
- Intensidade: 84
- Intensidade Relativa ao raio: 3,6522 (Intensidade dividida pelo Raio)

Caso fosse realizada a análise dos círculos encontrados para os raios de 32 e 42 pixels, mesmo a Intensidade sendo maior do que para o raio de 23, a Intensidade Relativa ao Raio

foi 2,8750 e 2,4286 respectivamente, mostrando que mesmo com intensidades maiores o melhor círculo deveria realmente ser o que tem o raio de 23 pixels.

4.2.3 Método de Rastreamento Utilizado

Como demonstrado na seção 3.3, com o objetivo de diminuir o tempo de processamento, foram utilizadas informações do objeto encontrado no frame anterior como parâmetro de entrada para o próximo frame, porém, isso só é possível se o objeto foi encontrado de forma correta anteriormente.

Desta forma, durante a execução de todo o sistema, podem ocorrer duas situações. Uma situação é aquela em que a imagem é a primeira a ser processada ou houve mais de cinco imagens consecutivas sem encontrar um círculo. A outra situação ocorre quando foi encontrado círculo nas últimas cinco imagens. As duas situações possuem funções em comum, o Pré-Processamento e Segmentação e o Reconhecimento, explicados nas seções anteriores.

Abaixo serão detalhadas essas duas situações.

Primeira Situação: Primeira Imagem ou círculo não encontrado em mais de cinco imagens consecutivas

Essa parte do processamento é necessária quando o software é iniciado ou quando ocorre algum erro na busca por um círculo. Esse erro pode ser devido à movimentação muito rápida do olho, ou se a pessoa ficou com o olho fechado por um intervalo longo ou ainda alguma outra razão que faça com que o ponto central da pupila saia do raio de busca por mais de cinco quadros consecutivos.

Quando esse erro ocorre o programa efetua o pré-processamento e segmentação na imagem inteira e, então, na imagem de bordas, busca um círculo com um raio que esteja no intervalo mínimo de 15 e máximo de 70 pixels, que é o intervalo possível utilizado para encontrar a pupila, pois o menor raio encontrado foi de 17 e o maior foi de 37 em todas as

imagens analisadas neste trabalho.

Após a busca, a informação do centro e raio do círculo encontrado é utilizada no processamento do quadro seguinte.

Segunda Situação: Círculo encontrado em um intervalo de até cinco imagens anteriores

Essa parte do processamento é realizada quando em até cinco imagens anteriores foi encontrado um círculo com raio de limite especificado.

Nesse caso, antes de passar pelo pré-processamento e pela busca, a imagem é cortada com um tamanho de 2,5 vezes o diâmetro (limite máximo encontrado para um possível movimento da pupila) e posição com centro do círculo encontrado no quadro anterior, dessa forma, o espaço de busca diminui, reduzindo o tempo de processamento.

Ainda nessa situação, a busca é realizada apenas para encontrar círculo de raio num intervalo de dois pixels acima e abaixo do raio do circulo do quadro anterior (limite máximo encontrado para uma possível dilatação ou contração da pupila), fazendo com que o tempo de processamento seja ainda menor.

Após a busca, a informação do centro e raio do círculo encontrado é utilizada no processamento do quadro seguinte.

Depois de cada quadro processado o sistema salva em um documento de texto as informações adquiridas no processamento desse quadro, tais como: tempo de processamento, posição do centro e raio do círculo encontrado.

4.3 Considerações Finais do Capítulo

Neste Capítulo foi descrito todo o sistema, de acordo com os Capítulos 2 e 3, demonstrando quais bibliotecas foram utilizadas em cada parte do processamento.

No próximo capítulo o sistema proposto será colocado em teste em três situações distintas.

5 TESTES E RESULTADOS

5.1 Introdução

O Sistema foi testado em relação a tempo real, tempo de processamento e precisão na localização e tamanho da pupila.

No caso do teste em tempo real, o Sistema foi testado em duas condições: a Primeira Condição refere-se à busca de um círculo desenhado em uma folha de papel, movimentando-a em frente à câmera; a Segunda Condição refere-se à busca pela pupila em um olho humano.

No teste de tempo de processamento foram utilizadas imagens do trabalho [11] submetidas ao software criado por ele, e ao software criado no presente trabalho em condições diferentes, mostrando a diferença do tempo de processamento utilizando o método de rastreamento proposto.

No teste da precisão foram utilizados três softwares diferentes para encontrar a posição e o raio das pupilas de imagens do banco de imagens CASIA [15], comparando esses valores e encontrando a precisão do software proposto.

Todos os testes foram realizados em um computador com processador Core i5 de segunda geração e 4 GB de memória RAM.

5.2 Testes em Tempo Real

Os testes em tempo real foram divididos em duas fases, uma para verificar que o sistema realmente consegue encontrar um círculo e efetuar o processamento necessário em tempo real, e outra em uma situação real em que foi efetuada a busca pela pupila em um olho.

Para captar as imagens dessas duas situações e também da próxima seção, foi utilizada uma câmera Logitech modelo C525 capaz de filmar em HD 720p capturando até 30 FPS [42] conforme mostrada na figura 5.1.

Figura 5.1: Câmera utilizada na captação das imagens.

O detalhamento de cada teste em tempo real será feito nas próximas subseções.

5.2.1 Primeira Condição

Para realizar este teste foi desenhado um círculo preto em uma folha de papel com o objetivo de simular uma pupila, conforme ilustrado na Figura 5.2, esta folha foi então posicionada em frente à câmera e o sistema foi iniciado.

A busca pelo objeto na imagem do primeiro frame, ilustrada pela Figura 5.2, demorou 1340 *ms*, pois o intervalo do raio para a busca era de 5 a 100 pixels. Para a imagem do segundo frame utilizou-se a informação do objeto encontrado no primeiro frame e alterado o intervalo de busca para três pixels a menos e a mais do raio encontrando no primeiro frame,

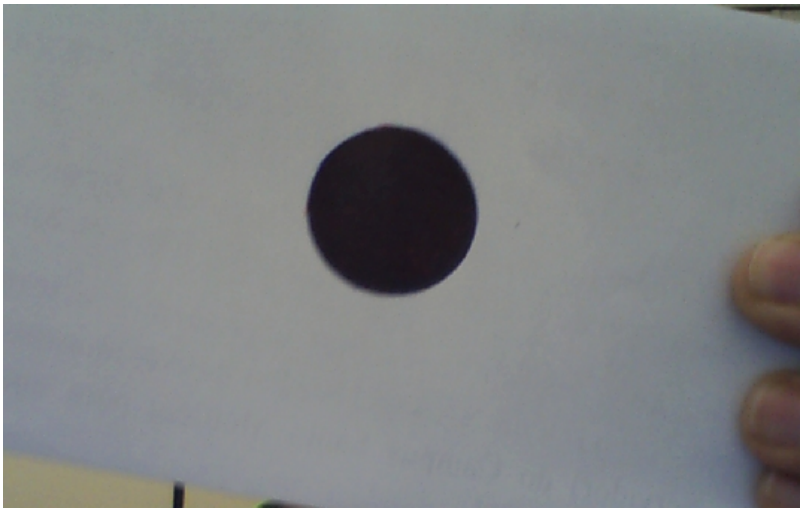


Figura 5.2: Exemplo de imagem utilizada para busca do círculo na primeira condição.

a busca demorou 44 *ms*, lembrando também que o espaço de busca diminuiu, ilustrada pela Figura 5.3.a.

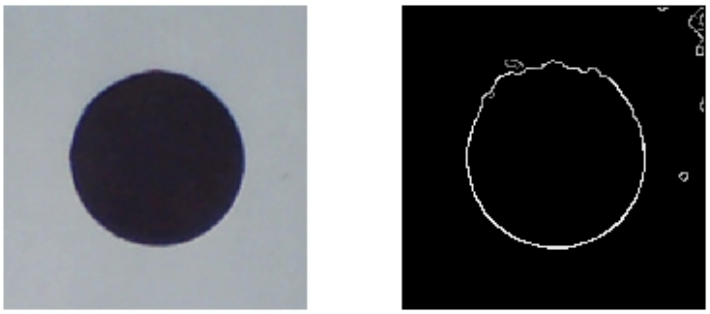


Figura 5.3: a) Imagens cortada e b) pré-processada de acordo com as Etapas descritas no Capítulo 2.

Durante todo o tempo de aquisição das imagens a folha era deslocada para os lados e para cima, para simular a movimentação da pupila de uma pessoa, sendo que o gráfico da Figura 5.4 mostra a alteração da posição x (azul) e y (vermelho) de quadro a quadro.

É importante salientar que pelo gráfico da Figura 5.4 não existe uma grande variação da posição x e y do círculo encontrado de um frame para outro. Este fato colabora para a comprovação da eficácia de utilização desta informação como posição central e limite de corte do próximo frame, bem como na estimativa da posição do centro do círculo no

mesmo.

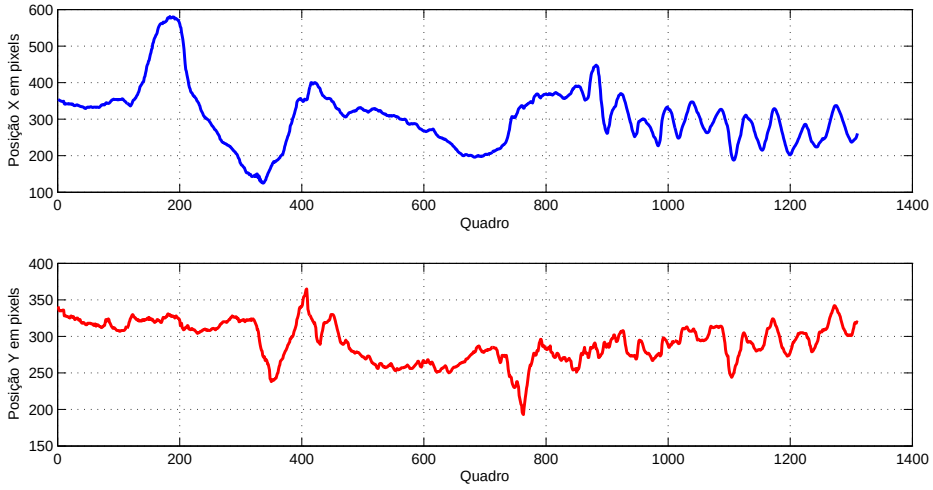


Figura 5.4: Alteração da posição do centro, x (azul) e y (vermelho), de quadro a quadro para o teste na Primeira Condição.

Além do deslocamento lateral a folha também era movimentada para trás e para frente, aumentando e diminuindo o raio do círculo encontrado, com o objetivo de simular a dilatação e contração da pupila pelo reflexo luminoso, sendo que o gráfico da Figura 5.5 mostra a alteração do raio de quadro a quadro.

Desta forma, foram adquiridas 1310 imagens, excluindo o tempo de processamento da primeira imagem, os tempos ficaram em um intervalo de 5 a 64 *ms*, conforme mostra a Figura 5.6.

Realizando ainda uma análise um pouco mais aprofundada em relação ao tempo gasto no processamento de cada quadro, conforme o gráfico na Figura 5.7 pode-se observar que quanto maior o raio encontrado em um quadro, maior o tempo gasto no processamento do mesmo.

O tempo médio de processamento de cada imagem foi de 24,2 *ms*, abaixo de 33 *ms*, que é o tempo para se conseguir um processamento de vídeo em tempo real, considerando a câmera utilizada que gera 30 Frames por Segundo, sendo possível processar um quadro antes do próximo ser adquirido, conforme [25]. Ainda analisando os resultados deste primeiro

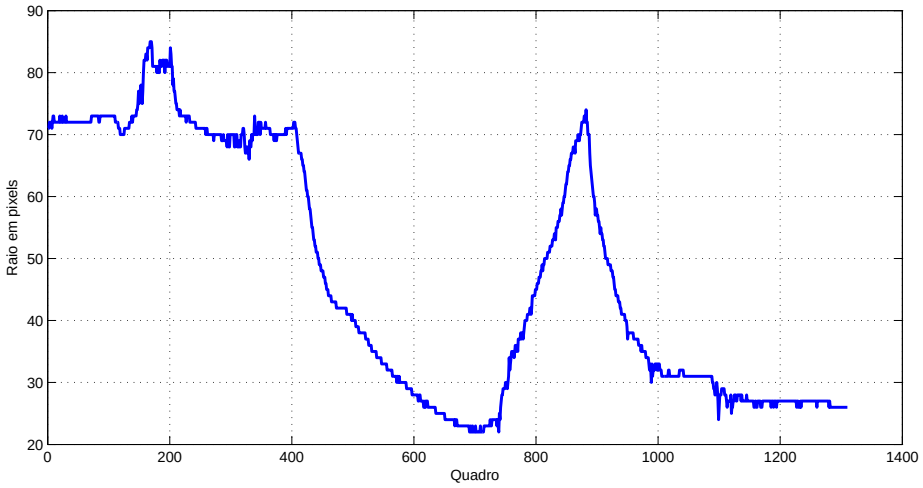


Figura 5.5: Alteração do raio do círculo encontrado de quadro a quadro para o teste na Primeira Condição.

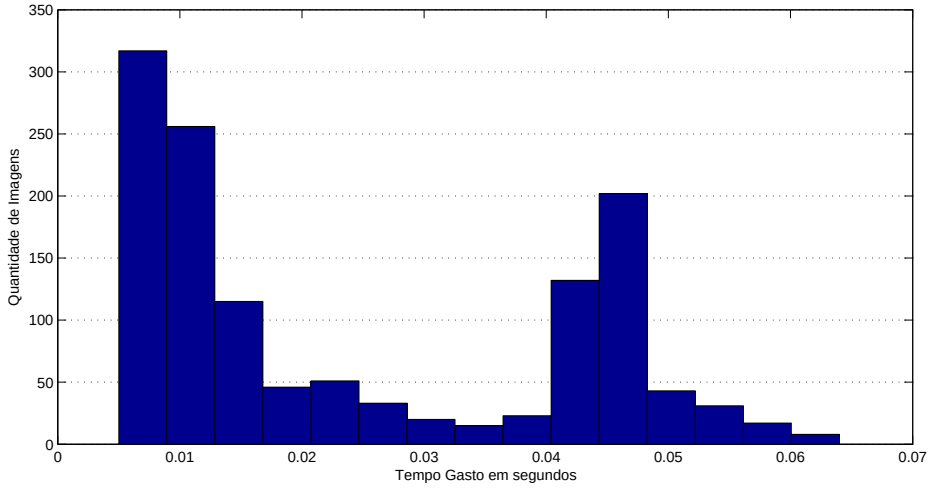


Figura 5.6: Distribuição de frequência do tempo gasto no processamento de cada quadro, excluindo o primeiro, para o teste na Primeira Condição.

teste, pela Figura 5.7 pôde-se observar que, para manter um tempo gasto no processamento de cada quadro inferior aos 33 *ms*, alcançando 30 FPS, o raio encontrado deve ser inferior a 62 pixels.

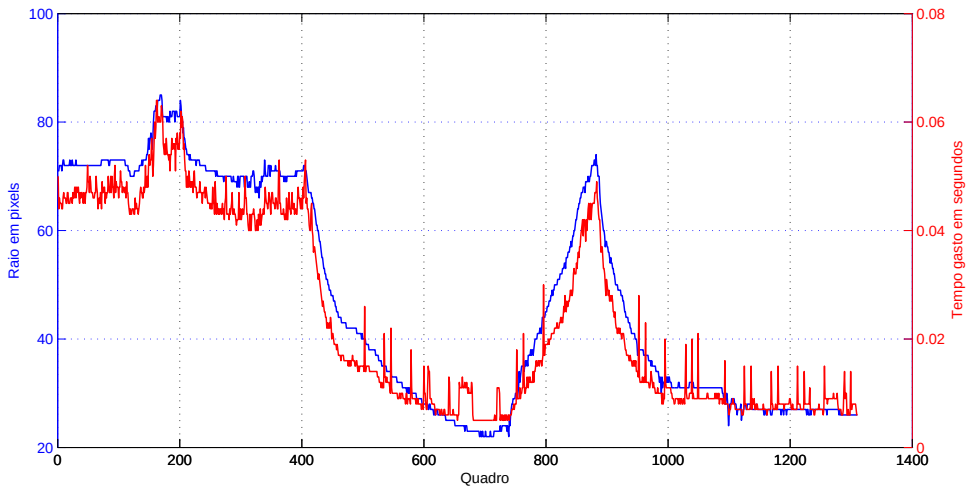


Figura 5.7: Relação entre o raio encontrado (azul) e o tempo gasto no processamento (vermelho) de cada quadro para o teste na Primeira Condição.

5.2.2 Segunda Condição

Para realizar o teste em condição real, utilizou-se uma iluminação infravermelha no olho que seriam captadas as imagens, tomando cuidado para que não houvesse iluminação externa no mesmo. A Figura 5.8 mostra o aparato criado para a captura das imagens do presente trabalho.

Figura 5.8: Sistema utilizado para efetuar o teste em condições reais do olho humano.

Nesta condição, o tempo de busca pelo objeto (pupila) na imagem do primeiro frame, ilustrada pela Figura 5.9, foi de 2518 *ms*, onde foi efetuada uma busca para círculos com raio de 15 a 70 pixels. Para a imagem do segundo frame, onde são utilizadas as informações do círculo encontrado no primeiro frame para efetuar o corte da imagem original, ilustrada pela Figura 5.10.a, e diminuir o limite de raios de busca, o tempo na busca foi de 7 *ms*.

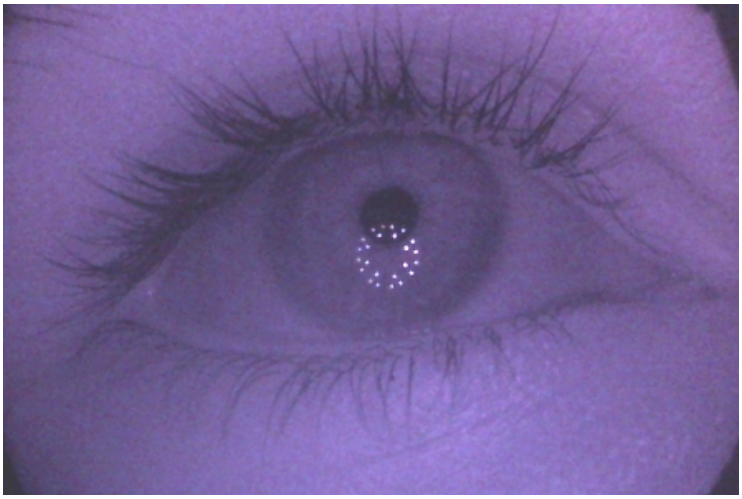


Figura 5.9: Imagem do olho utilizada para busca da pupila na segunda condição.

Figura 5.10: a) Imagem do olho cortada e b) pré processada após o primeiro frame.

A aquisição das imagens do olho foi feita com o olho iluminado somente por Led's infravermelho, e ao longo do tempo a pessoa ia mexendo o olho para poder testar o Sistema, o gráfico da Figura 5.11 mostra a variação da posição X e Y da pupila de quadro a quadro.

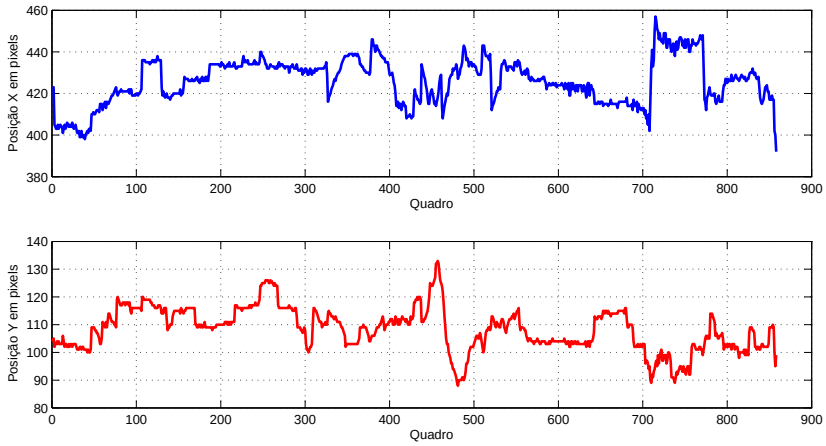


Figura 5.11: Alteração da posição x (azul) e y (vermelho) de quadro a quadro para o teste na Segunda Condição.

É importante observar também o gráfico da Figura 5.12, que mostra a diferença da posição da pupila encontrada entre dois quadros sequenciais. A partir deste gráfico pode-se observar que a diferença máxima foi de 18 pixels, de forma que o centro sempre estará dentro do corte realizado na imagem original.

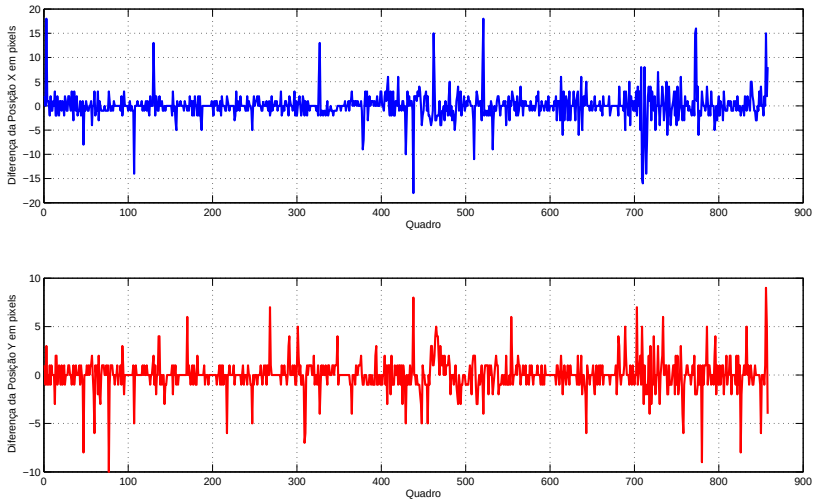


Figura 5.12: Diferança da posição x (azul) e y (vermelho) entre um quadro e o seguinte para o teste na Segunda Condição.

O gráfico da Figura 5.13 ilustra a variação do raio da pupila encontrada em cada quadro, pode-se observar que não houve uma variação tão significativa como do teste da subseção anterior, isso se deve ao fato de que não houve grande variação de iluminação no olho que não estava sendo filmado.

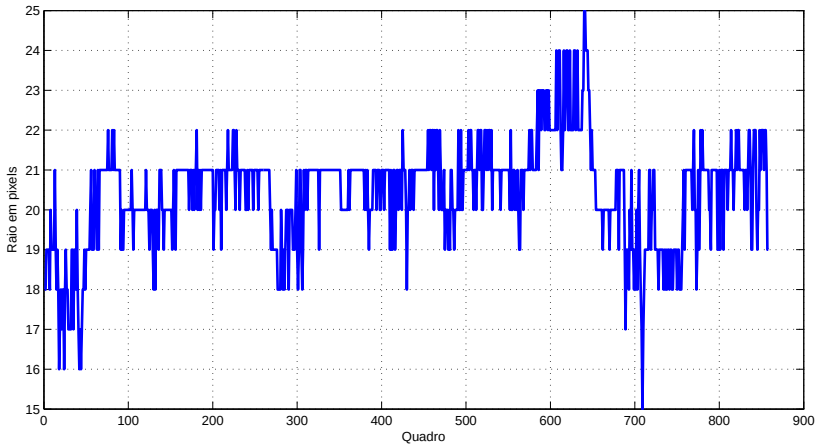


Figura 5.13: Alteração do raio da pupila de quadro a quadro para o teste na Segunda Condição.

Neste teste real foram adquiridas 858 imagens, excluindo o primeiro frame, o tempo de processamento ficou entre 4 e 20 *ms*, conforme o gráfico da Figura 5.14.

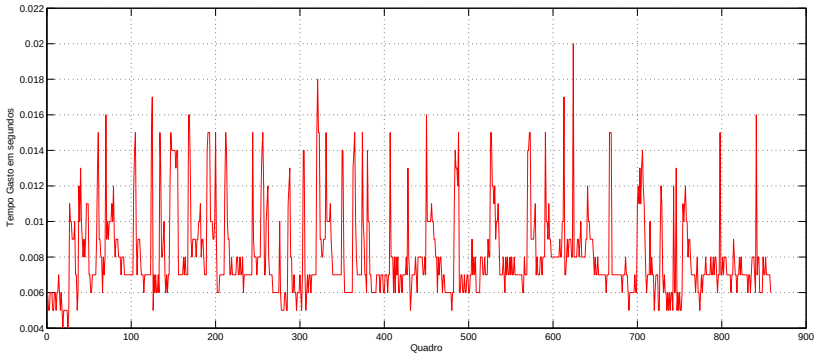


Figura 5.14: Tempo gasto no processamento de cada quadro para o teste na Segunda Condição.

Através do gráfico da Figura 5.15 que ilustra a distribuição de frequência do tempo gasto no processamento de cada imagem do olho, é possível perceber que o sistema é capaz

de efetuar rastreamento da pupila em até 250 FPS utilizando o menor tempo gasto (0,004 *ms*), ou uma média de 123 FPS, sendo que o tempo médio foi de 8,1 *ms*, o que supera o tempo necessário para se alcançar um processamento de vídeo em tempo real.

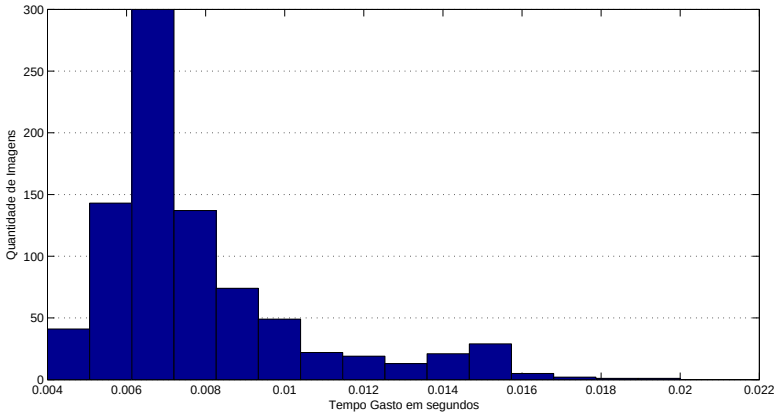


Figura 5.15: Distribuição de frequência do tempo gasto no processamento de cada imagem do olho para o teste na Segunda Condição.

Essa diferença em tempo de processamento do círculo no papel para o olho humano se deve inteiramente pelo tamanho do círculo encontrado. Utilizando a folha de papel como teste, os círculos encontrados ficaram com raios de 22 a 85 pixels. Já utilizando o olho humano os círculos encontrados tinham raios entre 15 e 25 pixels. Como o tamanho é menor o sistema encontra menos pontos de borda e, portanto, leva menos tempo para efetuar a busca pelo círculo através da TCH.

5.2.3 Comparação com Trabalhos Existentes

Em [43] é efetuado o rastreamento da pupila com o objetivo de obter imagens de alta resolução da retina, juntamente com a posição é encontrado também o tamanho, porém este tamanho não é utilizado no trabalho, e o resultado é de até 85 imagens processadas por segundo.

Em [13] não foi informado o tempo gasto no processamento de cada imagem, somente

quantos FPS possui a câmera utilizada, sendo que foi realizado apenas o rastreamento da posição da pupila.

Comparando os resultados obtidos com os trabalhos acima citados observa-se que em relação ao quesito tempo, o presente trabalho alcança o menor tempo gasto, incluindo o rastreamento do tamanho da pupila. Comparando com o trabalho que efetuou o rastreamento tanto da posição quanto do tamanho [43] o sistema proposto possui um resultado melhor, que chegou à possibilidade de rastrear a posição e tamanho da pupila em até 250 FPS.

5.3 Teste no Tempo de Processamento

Para realizar um Teste em relação ao Tempo de Processamento foram utilizadas imagens do trabalho [11], no qual foi utilizada a mesma Câmera do trabalho atual, Figura 5.1 [42], porém a iluminação do olho que estavam sendo capturadas as imagens não era com luz infravermelha, e sim da cor que estava sendo realizado o teste, ou seja, se o objetivo era ver como o olho reagia com a luz vermelha, o olho era iluminado diretamente com um Led de cor vermelha, alterando apenas a intensidade do brilho do mesmo, conforme o desejado. O mesmo foi feito para as cores branca, azul e verde.

Para os testes do tempo de processamento foram adquiridos os dados do círculo encontrado em cada imagem, raio e posição, assim como o tempo gasto no processamento de cada uma. Esses conjuntos de imagens foram submetidos ao sistema do trabalho [11], desenvolvido em Matlab, e ao sistema proposto, desenvolvido em C#, com diferentes parâmetros, demonstrando a diferença no tempo de processamento.

O Sistema em Matlab foi desenvolvido no trabalho [11], em que em cada imagem de uma pasta, assim como no sistema proposto, é efetuado um pré-processamento, descrito no Capítulo 2, e depois realizada a busca por um círculo com raio entre 15 e 70 pixels. Os dados adquiridos no processamento de cada imagem foram armazenados em um documento de texto para posterior análise.

Para efetuar o processamento das imagens que estão em uma determinada pasta foi criado um Sistema em C# em que é possível escolher os parâmetros que serão aplicados a cada imagem, conforme Figura 5.16, que mostra uma imagem do sistema desenvolvido para o processamento das imagens de uma pasta, sendo que este sistema utiliza o pré-processamento descrito no Capítulo 2 e a busca pelo círculo é realizada pela função proposta, descrita no Capítulo 4, e os dados adquiridos no processamento de cada imagem também são armazenados em um documento de texto para posterior análise.

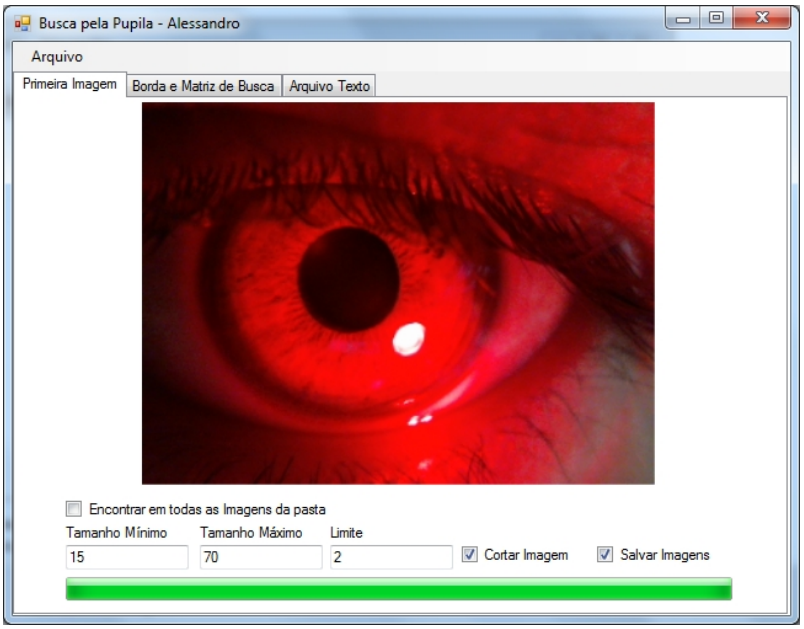


Figura 5.16: Imagem do Sistema desenvolvido para o processamento das imagens pertencentes a uma pasta.

Os testes foram realizados para dois conjuntos de imagens, um com a cor Branca e outro com a cor Vermelha, ambos foram adquiridos do olho da mesma pessoa, o da cor Branca com a intensidade da luz variando de 10% a 100% e retornando a 10%, e o da cor Vermelha com intensidade variando de 02% a 40%. Os resultados e análises destes testes estão descritos nas próximas Subseções.

5.3.1 Primeiro Conjunto de Imagens: Iluminação do Olho com LED da Cor Branca

Este primeiro conjunto possui 553 imagens do olho iluminado por um Led da cor Branca, sendo que, conforme dito anteriormente, no começo da captação das imagens a intensidade do Led era de 10%, no quadro 154 a intensidade passou a ser 100%, e depois no quadro 332 a intensidade voltou para 10%. As dimensões das Imagens são 329 por 245 pixels, e a Figura 5.17 mostra uma das imagens deste conjunto.

Figura 5.17: Exemplo de Imagem do Primeiro Conjunto.

Esse conjunto de imagens foi submetido aos dois sistemas conforme descrito anteriormente e os resultados de cada teste estão descritos abaixo juntamente com explicações e comparações entre os resultados.

Conjunto Submetido ao Sistema em Matlab

A Tabela 5.1 mostra os dados obtidos das trinta primeiras imagens, deste primeiro conjunto, submetidas ao sistema em Matlab, em que o hífen (-) representa imagens em que não foram encontrados círculos.

Imagem	X (px)	Y (px)	Raio (px)	Tempo (s)
1	-	-	-	1,216492
2	-	-	-	1,245612
3	-	-	-	1,112621
4	126	223	16	1,319663
5	139	225	16	1,414747
6	96	219	16	1,567809
7	97	216	17	1,587915
8	95	216	16	1,380674
9	127	223	16	1,403074
10	128	223	16	1,365319
11	-	-	-	1,346968
12	127	225	16	1,279828
13	127	227	16	1,320553
14	128	227	16	1,317278
15	128	229	16	1,273226
16	-	-	-	1,290348
17	-	-	-	1,218788
18	-	-	-	1,23408
19	-	-	-	1,161826
20	-	-	-	1,102935
21	-	-	-	1,19515
22	-	-	-	1,285982
23	-	-	-	1,506887
24	-	-	-	1,370749
25	170	40	26	2,14053
26	167	38	25	2,05221
27	77	183	17	1,932103
28	-	-	-	1,423824
29	175	158	16	1,53533
30	16-	16	25	1,365861

Tabela 5.1: Dados obtidos do processamento das primeiras 30 imagens do Primeiro Conjunto, utilizando o Sistema em Matlab.

Utilizando os dados obtidos do processamento de todas as 553 imagens observa-se que de todas as imagens, foi encontrado círculos em 62% ou 341 imagens, sendo que alguns desses círculos são falsos positivos pois de uma imagem para a outra tanto a distância quanto o raio possuem uma grande diferença, conforme é possível observar nos Gráficos

das Figuras 5.18 e 5.19, em que os pontos vermelhos representam as imagens em que não foi encontrado nenhum círculo, existem pontos em que há essa diferença de posição ou raio de uma imagem para outra, representando um falso positivo.

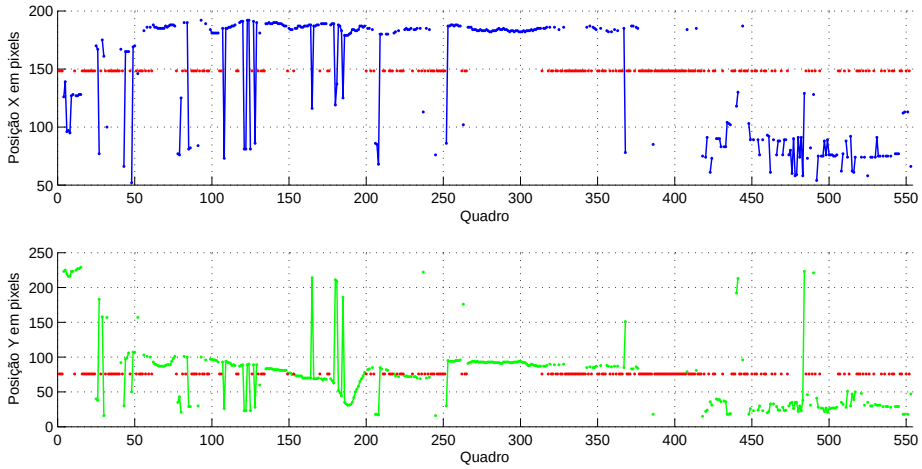


Figura 5.18: Variação da posição x (azul) e y (verde) do círculo encontrado em cada imagem do Primeiro Conjunto, utilizando o Sistema em Matlab.

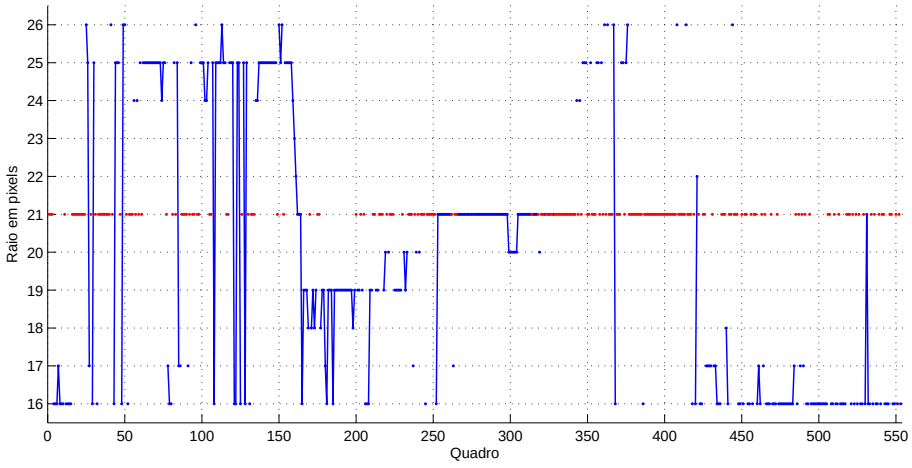


Figura 5.19: Variação do raio do círculo encontrado em cada imagem do Primeiro Conjunto, utilizando o Sistema em Matlab.

O Gráfico da Figura 5.20, mostra a distribuição de frequência do tempo gasto no processamento de cada imagem do olho, e é possível perceber que o tempo de processamento

de cada imagem está entre 250 *ms* e 4,25 *segundos* ou 4250 *ms*, com uma média de 1460 *ms*.

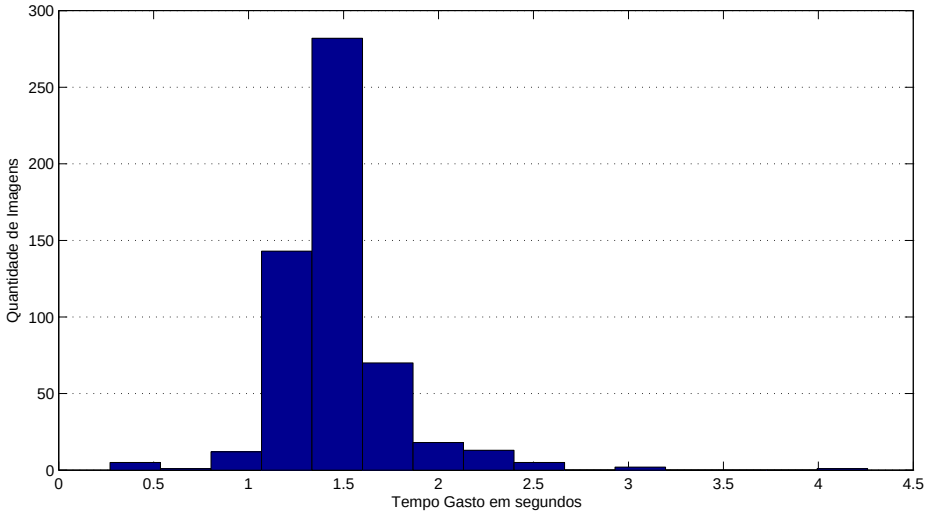


Figura 5.20: Distribuição de frequência do tempo gasto no processamento de cada imagem do Primeiro Conjunto, utilizando o Sistema em Matlab.

Conjunto Submetido ao Sistema Proposto

As Imagens deste primeiro conjunto foram submetido ao sistema proposto em diversas situações: a mesma situação do sistema do trabalho [11], em que não há o corte da imagem nem um limite de busca de raios; depois foi testado de forma que houvesse um corte da próxima imagem porém ainda sem um limite na busca de raios; outro teste foi sem o corte da imagem porém com um limite de busca pelo raio de dois pixels acima e abaixo do raio encontrado na imagem anterior; mais um teste foi com o sistema completo, com o corte da imagem e com um limite de busca pelo raio de dois pixels acima e abaixo do raio encontrado na imagem anterior; e por fim um teste com corte e com um limite de busca pelo raio de cinco pixels acima e abaixo do raio encontrado na imagem anterior.

- **Sem Corte na Imagem e Sem Limite de Busca no Raio:**

Utilizando os dados obtidos do processamento de todas as 553 imagens observa-se que de todas as imagens, foi encontrado círculos em 549 imagens, sendo que alguns desses círculos são falsos positivos, conforme é possível observar nos Gráficos das Figuras 5.21 e 5.22, em que os pontos vermelhos representam as imagens em que não foi encontrado nenhum círculo, existem pontos em que há essa diferença de posição ou raio de uma imagem para outro, representando um falso positivo.

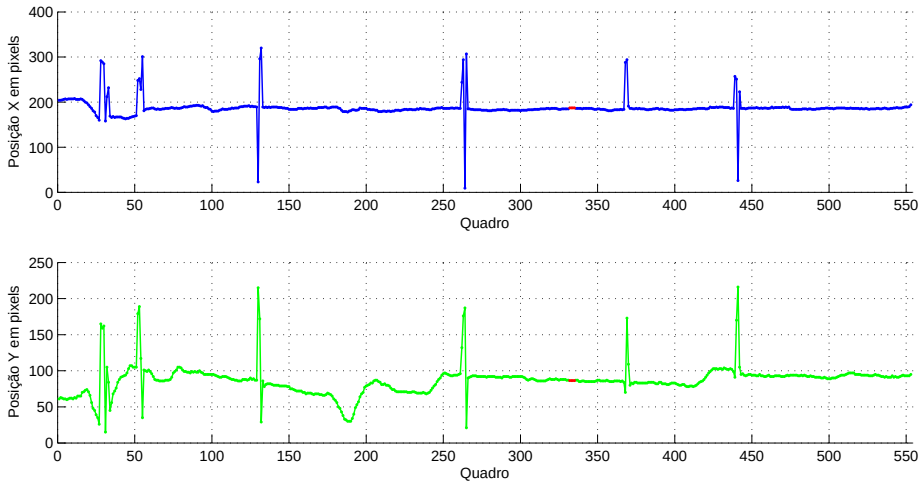


Figura 5.21: Variação da posição x (azul) e y (verde) do círculo encontrado em cada imagem do Primeiro Conjunto, utilizando o Sistema Proposto na Situação 1.

O Gráfico da Figura 5.23, mostra a distribuição de frequência do tempo gasto no processamento de cada imagem do olho, e é possível perceber que o tempo de processamento de cada imagem está entre 100 *ms* e 750 *ms*, com uma média de 560 *ms*.

- **Com Corte na Imagem e Sem Limite de Busca no Raio:**

  Utilizando os dados obtidos do processamento de todas as 553 imagens observa-se que de todas as imagens, foi encontrado círculos em 524 imagens, com menos falsos positivos, conforme é possível observar nos Gráficos das Figuras 5.24 e 5.25, em que os pontos vermelhos representam as imagens em que não foi encontrado nenhum círculo, existem menos pontos em que há essa diferença de posição ou raio de uma

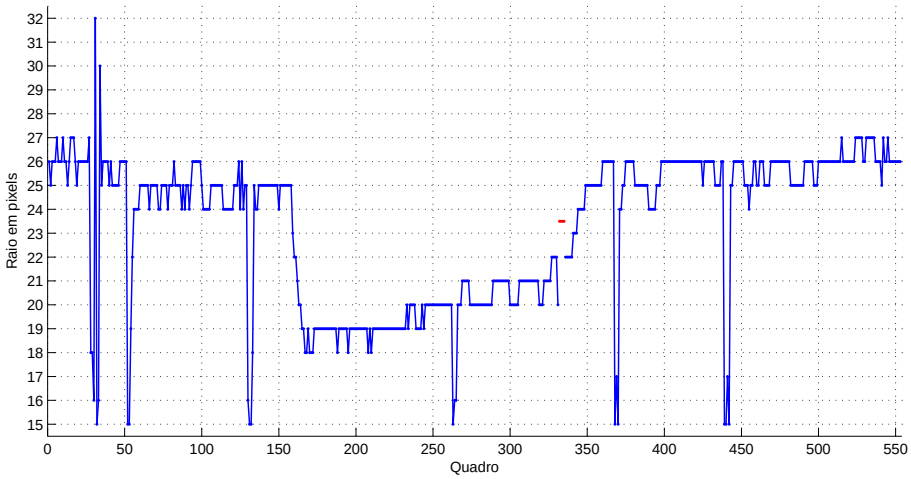


Figura 5.22: Variação do raio do círculo encontrado em cada imagem do Primeiro Conjunto, utilizando o Sistema Proposto na Situação 1.

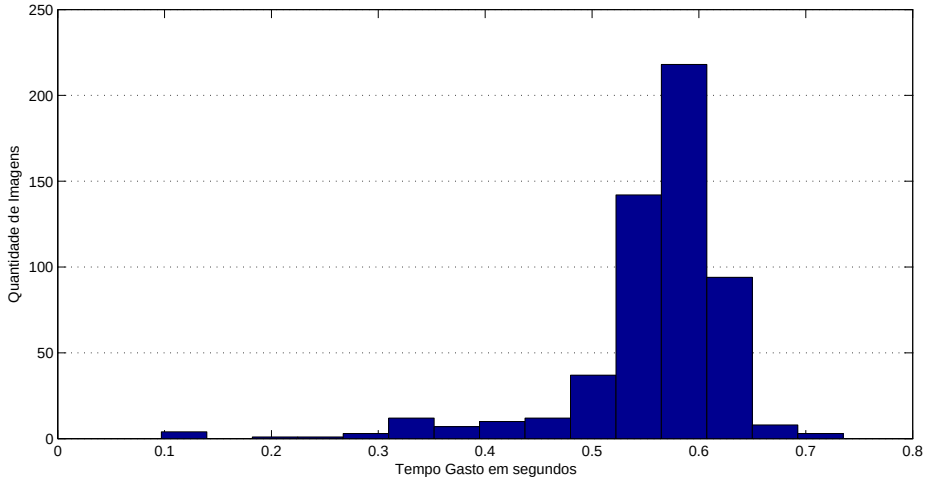


Figura 5.23: Distribuição de frequência do tempo gasto no processamento de cada imagem do Primeiro Conjunto, utilizando o Sistema Proposto na Situação 1.

imagem para outra, representando um falso positivo, se comparado com os Gráficos das Figuras 5.21 e 5.22.

O Gráfico da Figura 5.26, mostra a distribuição de frequência do tempo gasto no processamento de cada imagem do olho, e é possível perceber que o tempo de processamento de cada imagem está entre 8 *ms* e 500 *ms*, com uma média de 53,1 *ms*.

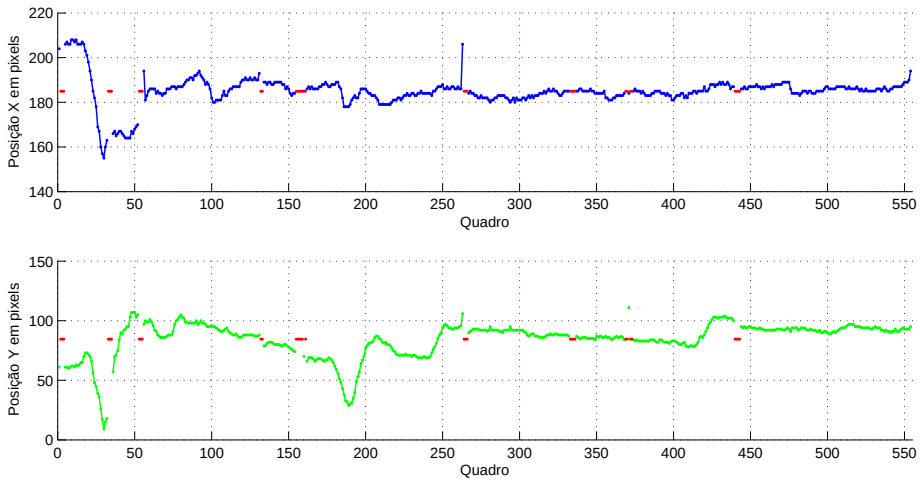


Figura 5.24: Variação da posição x (azul) e y (verde) do círculo encontrado em cada imagem do Primeiro Conjunto, utilizando o Sistema Proposto na Situação 2.

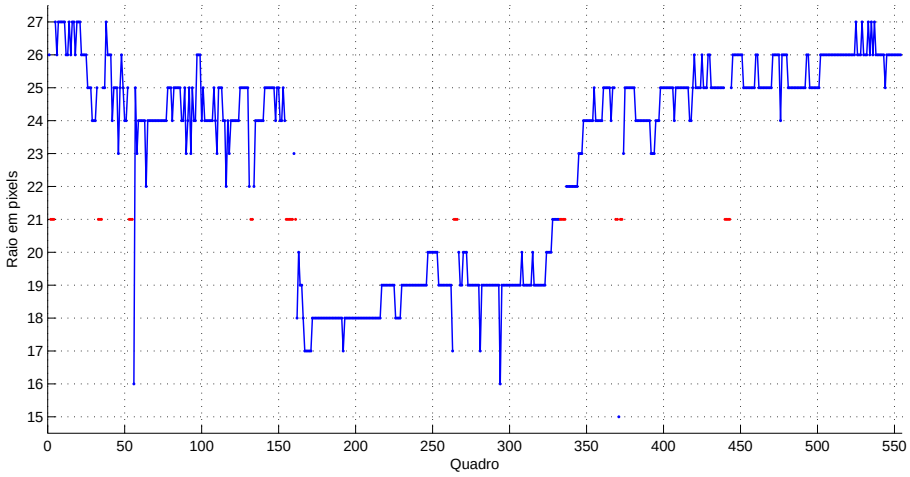


Figura 5.25: Variação do raio do círculo encontrado em cada imagem do Primeiro Conjunto, utilizando o Sistema Proposto na Situação 2.

O que deve ser levado em consideração no gráfico da Figura 5.26 é que existem apenas 2 imagens com tempo de processamento em torno de 500 *ms*, que são justamente onde houve mais de 5 imagens consecutivas sem encontrar a pupila, e por isso houve o processamento em toda a imagem, conforme explicado no Capítulo 4 o que leva mais tempo de processamento. Excluindo o tempo dessas imagens obtém-se o Gráfico

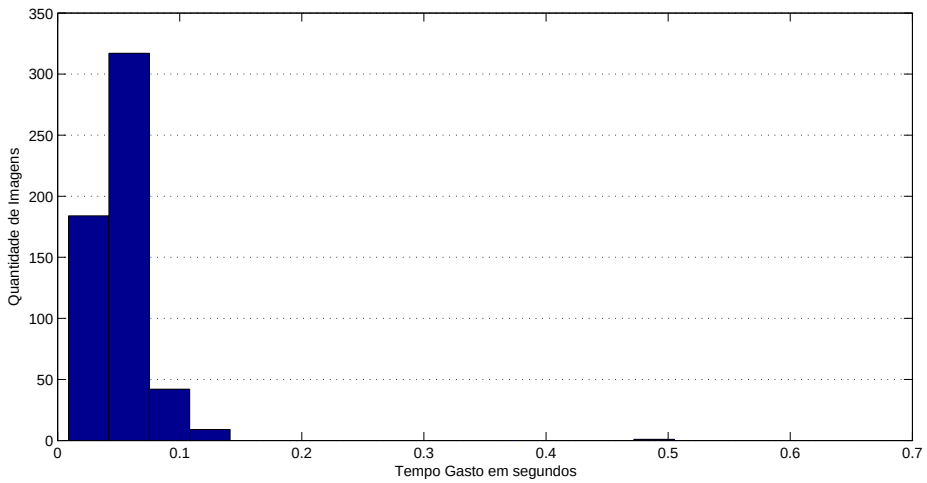


Figura 5.26: Distribuição de frequência do tempo gasto no processamento de cada imagem do Primeiro Conjunto, utilizando o Sistema Proposto na Situação 2.

da Figura 5.27, em que o tempo mínimo é de 8 *ms* o máximo de 51,5 *ms*, com uma média de 12,7 *ms*.

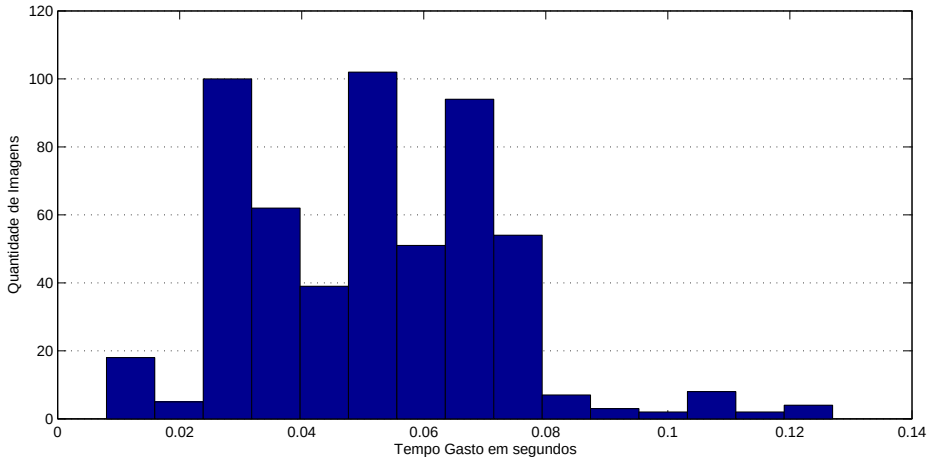


Figura 5.27: Distribuição de frequência do tempo gasto no processamento de cada imagem do Primeiro Conjunto, utilizando o Sistema Proposto na Situação 2, excluindo o tempo de processamento das imagens inteiras.

- **Sem Corte na Imagem e Com Limite de 2 pixels acima e abaixo na Busca pelo Raio:**

Utilizando os dados obtidos do processamento de todas as 553 imagens observa-se que de todas as imagens, foi encontrado círculos em 539 imagens, com alguns falsos positivos, conforme é possível observar nos Gráficos das Figuras 5.28 e 5.29, com os pontos vermelhos representando as imagens em que não foi encontrado nenhum círculo, existem pontos em que há essa diferença de posição ou raio de uma imagem para outra, representando um falso positivo.

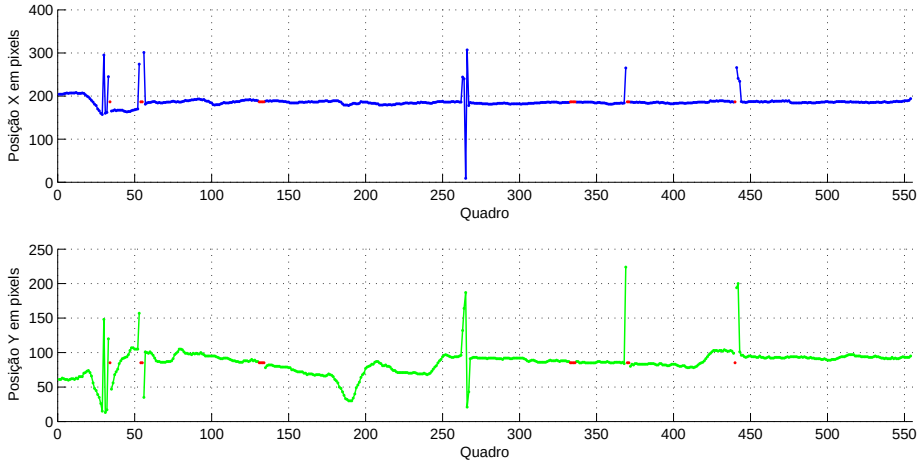


Figura 5.28: Variação da posição x (azul) e y (verde) do círculo encontrado em cada imagem do Primeiro Conjunto, utilizando o Sistema Proposto na Situação 3.

O Gráfico da Figura 5.30, mostra a distribuição de frequência do tempo gasto no processamento de cada imagem do olho, excluindo a primeira imagem em que o tempo de processamento é 501 *ms*, por ele é possível perceber que o tempo de processamento de cada imagem está entre 41 *ms* e 105 *ms*, com uma média de 70 *ms*.

- **Com Corte na Imagem e Com Limite de 2 pixels acima e abaixo na Busca pelo Raio:**

  Utilizando os dados obtidos do processamento de todas as 553 imagens observa-se que de todas as imagens, foi encontrado círculos em 523 imagens, sendo dessa vez poucos desses círculos são falsos positivos, conforme é possível observar nos Gráficos das Figuras 5.31 e 5.32 se comparado com os Gráficos das Figuras 5.21 e 5.22, em

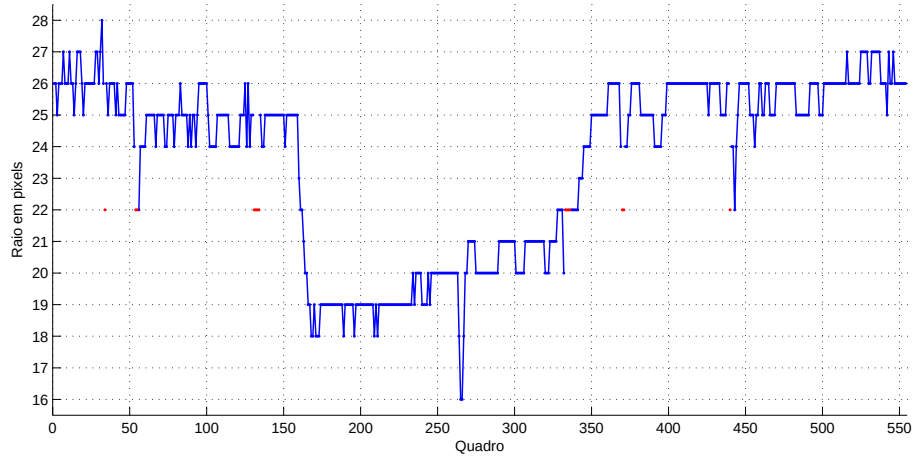


Figura 5.29: Variação do raio do círculo encontrado em cada imagem do Primeiro Conjunto, utilizando o Sistema Proposto na Situação 3.

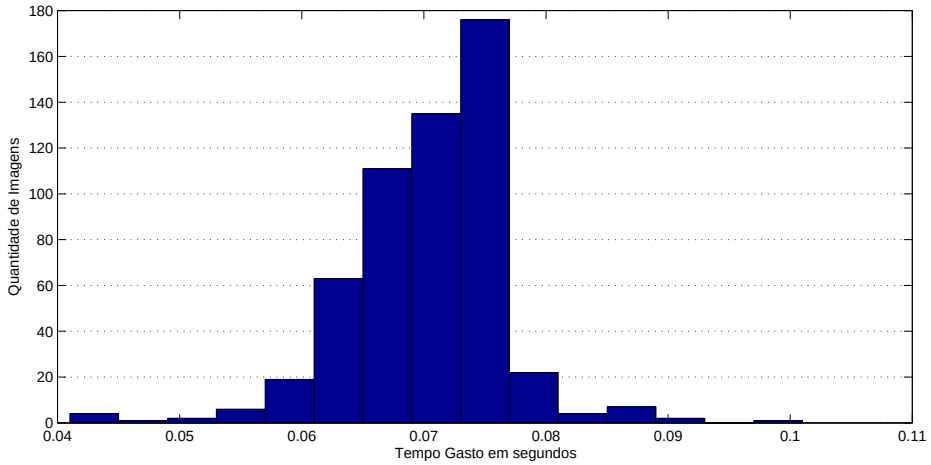


Figura 5.30: Gráfico que mostra a distribuição de frequência do tempo gasto no processamento de cada imagem do Primeiro Conjunto, utilizando o Sistema Proposto na Situação 3, excluindo o tempo de processamento da primeira imagem.

que nos primeiros existem menos pontos em que há essa diferença de posição ou raio de uma imagem para outra, os Gráficos ficaram mais contínuos, com menos picos que representavam um falso positivo.

Pela Figura 5.32 pode-se observar que a máxima variação do raio é de 10 pixels, condizente com o trabalho [11].

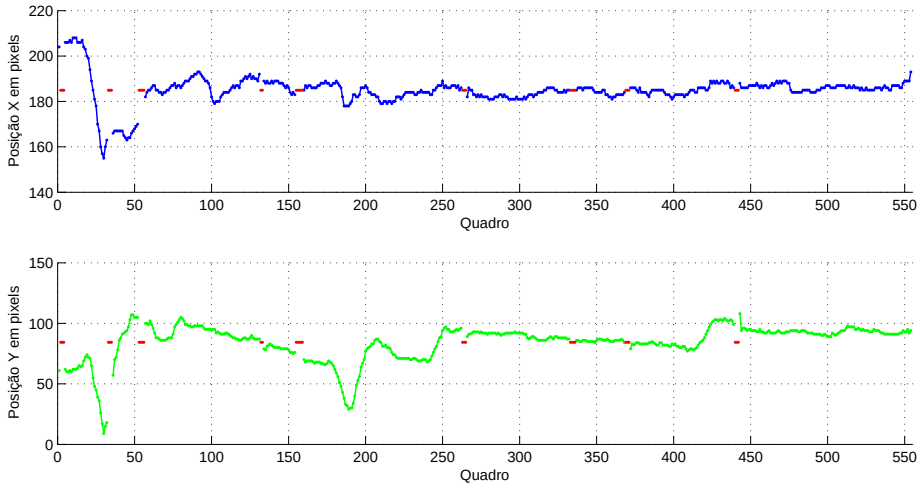


Figura 5.31: Variação da posição x (azul) e y (verde) do círculo encontrado em cada imagem do Primeiro Conjunto, utilizando o Sistema Proposto na Situação 4.

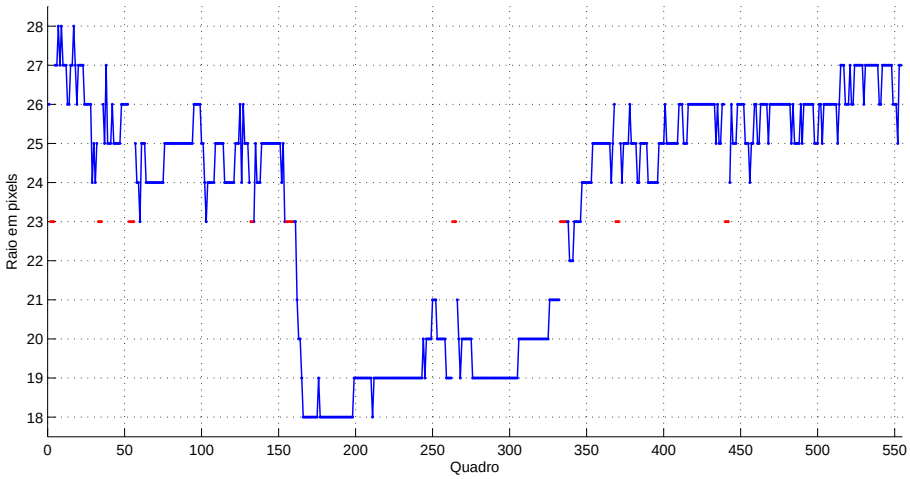


Figura 5.32: Variação do raio do círculo encontrado em cada imagem do Primeiro Conjunto, utilizando o Sistema Proposto na Situação 4.

Excluindo o tempo de processamento de duas imagens que foram processadas por completo, pois não haviam círculos encontrados em cinco imagens anteriores, casos esses que serão justificados posteriormente, o tempo de processamento das outras imagens ficou entre 4 *ms* e 22 *ms*, com uma média de 9,2 *ms* conforme pode ser observado no Gráfico da Figura 5.33, que mostra a distribuição de frequência do

tempo gasto no processamento de cada imagem do olho, excluindo duas imagens cujo tempo foi em torno de 500 *ms*.

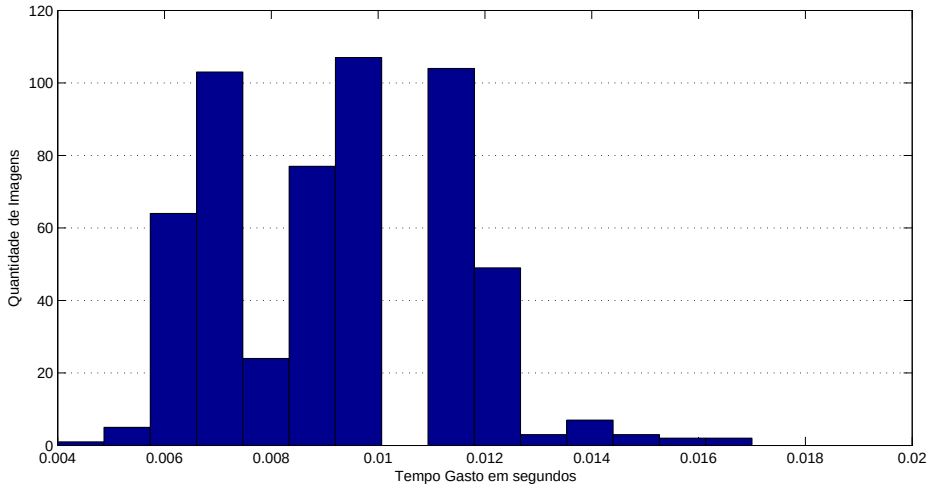


Figura 5.33: Distribuição de frequência do tempo gasto no processamento de cada imagem do Primeiro Conjunto, utilizando o Sistema Proposto na Situação 4, excluindo o tempo de processamento das imagens inteiras.

- **Com Corte na Imagem e Com Limite de 5 pixels acima e abaixo na Busca pelo Raio:**

  Utilizando os dados obtidos do processamento de todas as 553 imagens observa-se que, de todas as imagens, foram encontrados círculos em 525 imagens, sendo que alguns desses círculos são falsos positivos. É possível observar que nos Gráficos das Figuras 5.34 e 5.35 existem mais pontos representando falsos positivos do que nos Gráficos das Figuras 5.31 e 5.32

  que há essa diferença de posição ou raio de uma imagem para outra, representando um falso positivo se comparado com os Gráficos das Figuras 5.31 e 5.32.

  Excluindo o tempo de processamento de duas imagens que foram processadas por completo, pois não haviam círculos encontrados em cinco imagens anteriores, o tempo de processamento das outras imagens ficou entre 4 *ms* e 31 *ms*, com uma média de 12,6 *ms* conforme pode ser observado no Gráfico da Figura 5.36, que mostra a distribuição

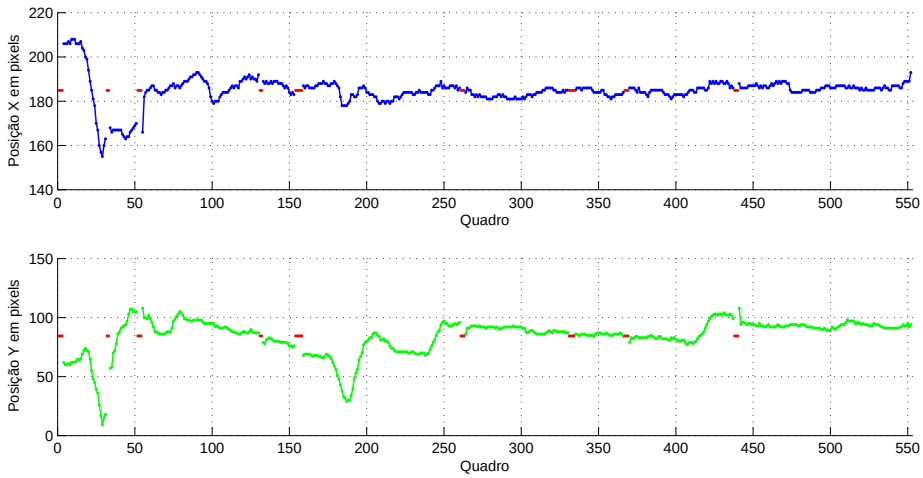


Figura 5.34: Variação da posição x (azul) e y (verde) do círculo encontrado em cada imagem do Primeiro Conjunto, utilizando o Sistema Proposto na Situação 5.

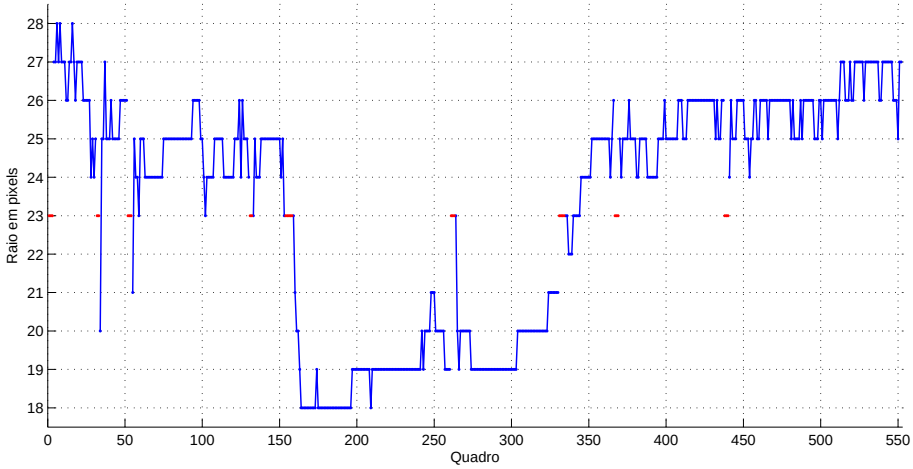


Figura 5.35: Variação do raio do círculo encontrado em cada imagem do Primeiro Conjunto, utilizando o Sistema Proposto na Situação 5.

de frequência do tempo gasto no processamento de cada imagem do olho, excluindo duas imagens cujo tempo foi em torno de 500 *ms*.

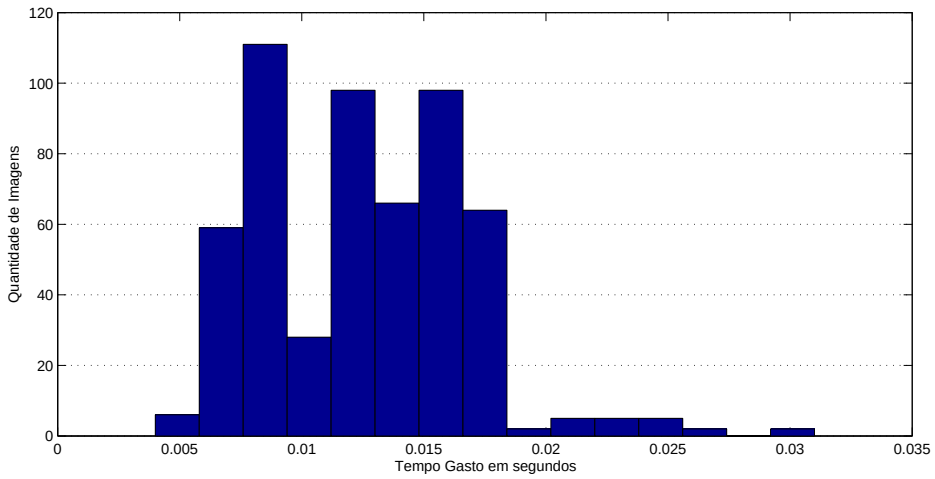


Figura 5.36: Distribuição de frequência do tempo gasto no processamento de cada imagem do Primeiro Conjunto, utilizando o Sistema Proposto na Situação 5, excluindo o tempo de processamento das imagens inteiras.

Discussão dos Resultados

Assim como no Primeiro Conjunto de Imagens, é possível perceber que os quesitos nos quais as diferenças nos resultados são mais realçadas são o tempo de processamento, a precisão de posição e do raio do círculo encontrado:

- **Tempo de Processamento de Cada Imagem:**

  Observando os Gráficos de Distribuição de Frequência do tempo gasto em cada imagem para o Sistema em Matlab e o Sistema em C# com as mesmas configurações, ou seja, sem corte na imagem e sem limite na busca pelo círculo, Figuras 5.20 e 5.23, já é possível observar que o Sistema em C# é muito mais rápido, como já era esperado, enquanto em Matlab o tempo médio gasto em uma imagem é de 1460 *ms* em C# esse tempo cai para 560 *ms*, cerca de 38% do tempo gasto no Sistema anterior. Comparando, então, com a proposta do presente trabalho, efetuando um corte na imagem (diminuindo informações desnecessárias) e limitando a busca por valores de raios mais próximos do encontrado anteriormente, esse tempo diminui ainda mais, 9,2 *ms*, menos do que 2% do tempo gasto no Sistema Proposto sem corte e sem limite

na busca do raio do círculo, comprovando assim a eficiência do Sistema Proposto com o corte e o limite de 2 pixels na busca pelo raio do círculo.

- **Precisão da Posição do Círculo Encontrado:**

  Enquanto a posição dos círculos encontrados para o sistema em Matlab oscilam muito, representando falsas pupilas encontradas, Figura 5.18, no sistema em C# é possível observar que não há essa oscilação, Figura 5.31, principalmente porque o sistema proposto não encontra círculos distantes do círculo encontrado na imagem anterior, porque antes de efetuar a busca ele faz um corte na imagem, possibilitando encontrar apenas aqueles que estão próximos da posição encontrada na imagem anterior.

  O Gráfico da Figura 5.37 mostra bem essa relação de falsas pupilas encontradas, em que para o Sistema proposto (Verde) o gráfico é mais uniforme, sem picos, já para o Sistema em Matlab (Azul) o gráfico possui mais picos que são posições muito diferentes do círculo encontrado de uma imagem para outra. Os pontos vermelhos representam as imagens com círculos não encontrados para o Sistema em Matlab.

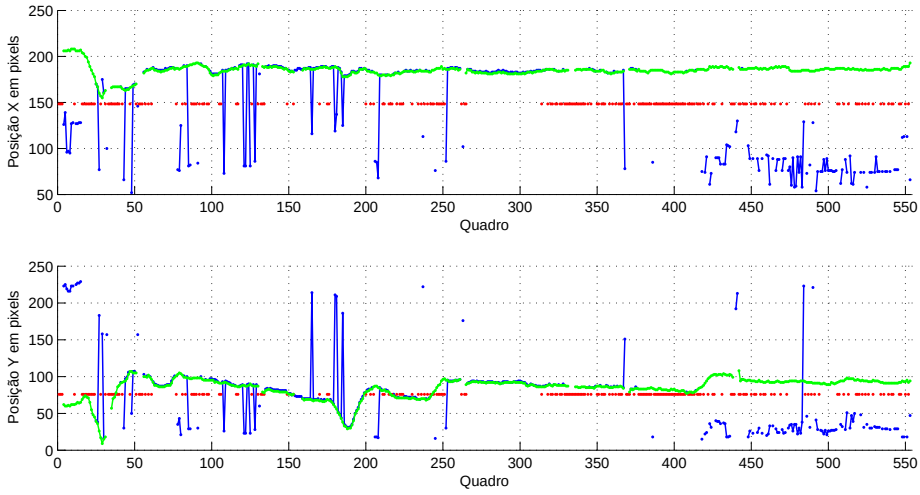


Figura 5.37: Posição X e Y do círculo encontrado em cada imagem do Primeiro Conjunto para o Sistema em Matlab (Azul) e C# (Verde) excluindo os círculos não encontrados.

Também é necessário fazer a comparação entre Sistemas com a mesma linguagem, onde a Figura 5.38 mostra em azul a posição dos círculos encontrados para o Sistema

em C# sem corte e sem limite na busca do raio do círculo, já em verde para o Sistema Proposto com corte na imagem e limite de 2 pixels na busca pelo raio do círculo, e em vermelho os quadros em que não foram encontrados círculos. Nesta imagem é possível verificar que, excluindo os quadros em que não foram encontrados círculos e os falsos positivos, os dois Sistemas encontram a mesma posição dos círculos, e no segundo caso (verde) eliminando os falsos positivos.

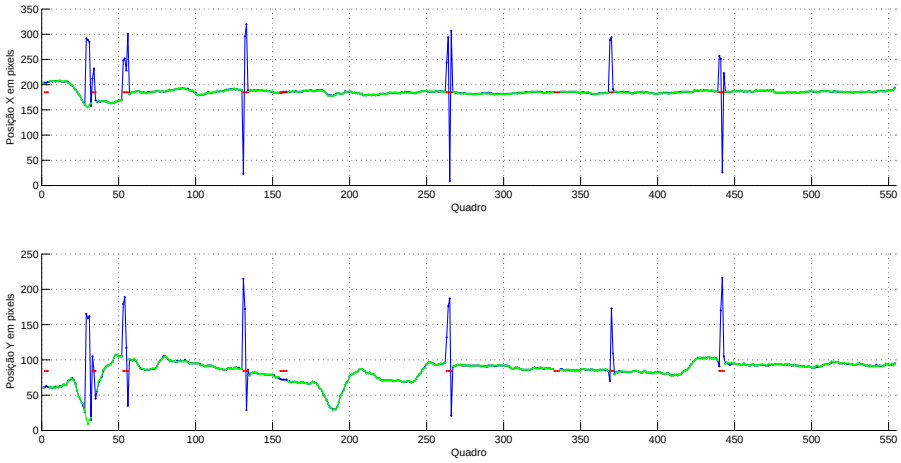


Figura 5.38: Posição X e Y do círculo encontrado em cada imagem do Primeiro Conjunto para o Sistema em C# sem utilizar informações do quadro anterior (Azul) e utilizando essas informações (Verde).

- **Precisão do Raio do Círculo Encontrado:**

  Da mesma forma que a posição dos Círculos encontrados no Sistema em Matlab variavam muito de uma imagem para outra, existe também essa variação para o valor do Raio, Figura 5.19. Já no Sistema proposto, não existe essa grande variação, Figura 5.32, aumentando a precisão do tamanho do círculo encontrando, principalmente pelo limite imposto para o tamanho do círculo que se deseja encontrar.

  O Gráfico da Figura 5.39 mostra bem essa relação de falsas pupilas encontradas, em que para o Sistema proposto (Verde) o gráfico é mais uniforme, sem picos, já para o Sistema em Matlab (Azul) o gráfico possui mais picos que são Tamanhos de

Raios muito diferentes do círculo encontrado de uma imagem para outra. Os pontos vermelhos representam as imagens com círculos não encontrados para o Sistema em Matlab.

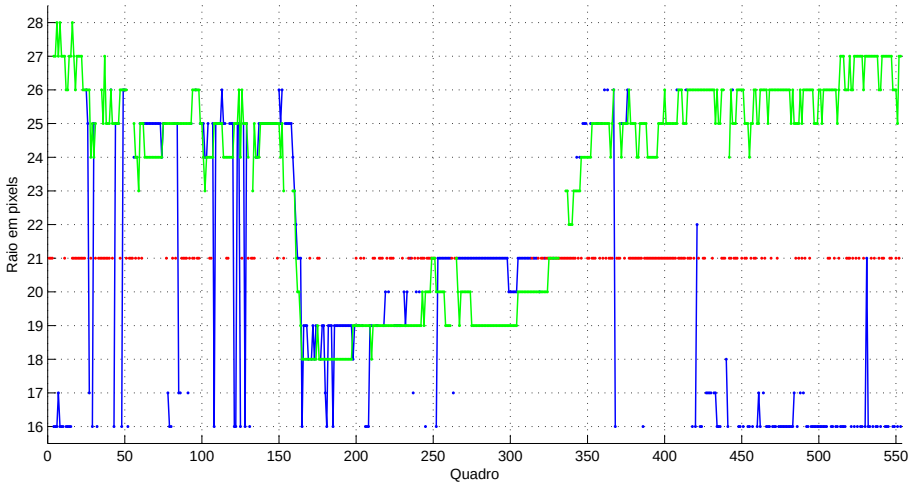
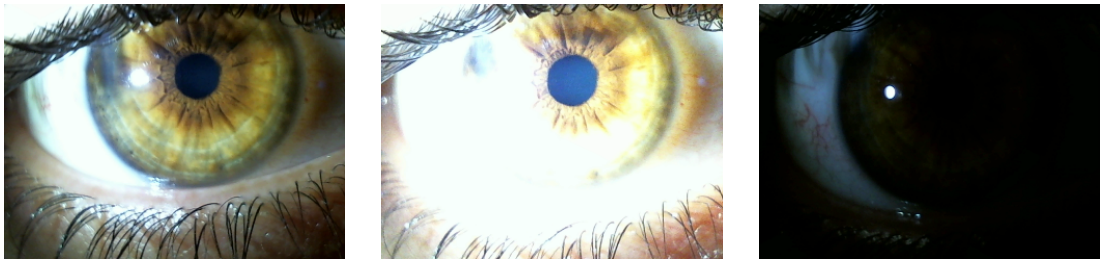


Figura 5.39: Tamanho do Raio do círculo encontrado em cada imagem do Primeiro Conjunto para o Sistema em Matlab (Azul) e C# (Verde) excluindo os círculos não encontrados.

Observando os Gráficos é necessário entender qual o motivo de não serem encontrados círculos em determinadas imagens, daí, pelo Gráfico da Figura 5.31 ou 5.32, observa-se que houve 9 conjuntos de imagens (totalizando 30 imagens) em que não foi possível encontrar a pupila.

Desses 9 conjuntos, dois são quando há uma variação muito grande na iluminação e a câmera ainda não se adaptou à ela, dessa forma, algumas imagens ficaram muito claras ou muito escuras, conforme as imagens *b* e *c* da Figura 5.40, além disso, a equalização do histograma não altera significantemente os valores de intensidade da pupila e os parâmetros do Sistema para binarização não estão adaptados a essa variação de iluminação. Daí a importância de utilizar um olho com a iluminação infra-vermelha sendo filmado e o outro olho com a luz variando a intensidade, pois, dessa forma, todas as imagens captadas estariam com o mesmo brilho.

O restante dos 9 conjuntos de imagens com círculos não encontrados se referem a



(a) *Imagem Normal.* (b) *Imagem Clara.* (c) *Imagem Escura.*

Figura 5.40: *Imagens de exemplo do Primeiro Conjunto que a) foi encontrada a pupila, b) e c) não foi encontrada a pupila.*

imagens em que a pessoa está piscando, conforme as imagens da Figura 5.41, que mostra uma sequência de 6 imagens em que 4 delas não é possível encontrar a pupila pois é um momento em que a mesma está completa ou parcialmente coberta pela pálpebra, e as outras duas são imagens que estão antes e depois da piscada.

Figura 5.41: *Imagens de exemplo do Primeiro Conjunto em que a pessoa estava piscando, ocorrendo uma oclusão completa ou parcial da pupila.*

5.3.2 Segundo Conjunto de Imagens: Iluminação do Olho com LED da Cor Vermelha

Este segundo conjunto possui 349 imagens do olho iluminado por um Led da cor Vermelha, sendo que, conforme dito anteriormente, no começo da captação das imagens a intensidade do Led era de 2%, no quadro 237 a intensidade passou a ser 40%. As dimensões das Imagens são 329 por 245 pixels, e a Figura 5.42 mostra uma das imagens deste conjunto.

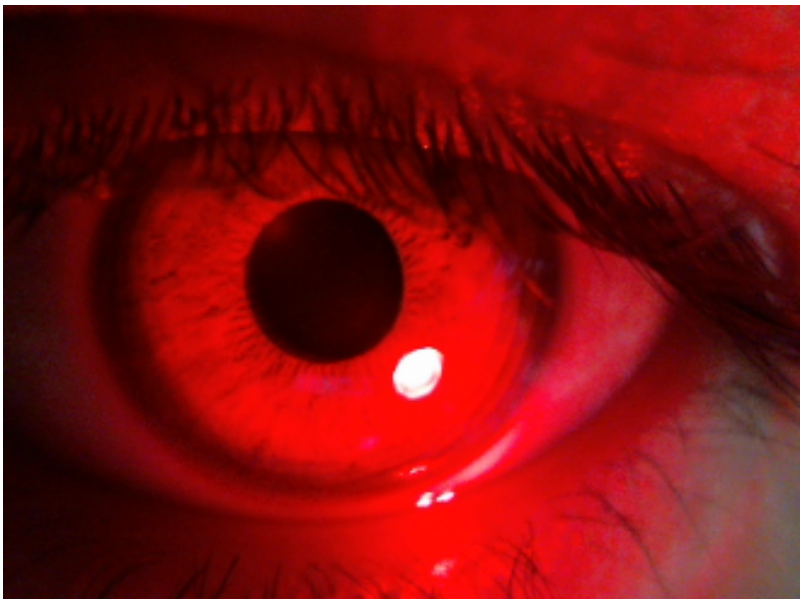


Figura 5.42: Exemplo de Imagem do Segundo Conjunto.

Esse conjunto de imagens foi submetido aos dois sistemas conforme descrito anteriormente e os resultados de cada teste estão descritos abaixo juntamente com explicações e comparações entre os resultados.

Conjunto Submetido ao Sistema em Matlab

A Tabela 5.2 mostra os dados obtidos das trinta primeiras imagens, deste primeiro conjunto, submetidas ao sistema em Matlab, em que o hífen (-) representa imagens em que não foram encontrados círculos.

Imagem	X (px)	Y (px)	Raio (px)	Tempo (s)
1	133	114	33	1,095049
2	147	112	32	0,505917
3	147	111	33	0,480665
4	146	111	33	0,503237
5	149	110	33	0,472627
6	-	-	-	0,480116
7	149	110	34	0,504442
8	150	110	33	0,458612
9	149	109	34	0,570657
10	149	111	33	0,646938
11	149	110	34	0,469127
12	149	111	34	0,464876
13	-	-	-	0,362562
14	131	112	34	0,562853
15	130	109	34	0,598287
16	130	110	35	0,59559
17	131	112	34	0,610428
18	133	114	34	0,551582
19	136	115	34	0,567423
20	135	111	33	0,539121
21	136	112	35	0,543624
22	135	110	33	0,57462
23	135	112	34	0,56705
24	134	109	32	0,537233
25	135	114	32	0,56915
26	134	111	31	0,578258
27	135	117	31	0,548588
28	135	116	32	0,620515
29	135	115	32	0,642445
30	135	113	34	0,755414

Tabela 5.2: Dados obtidos do processamento das primeiras 30 imagens do Segundo Conjunto, utilizando o Sistema em Matlab.

Utilizando os dados obtidos do processamento de todas as 349 imagens observa-se que de todas as imagens, foi encontrado círculos em 96% ou 336 imagens, sendo que, no caso deste segundo conjunto de imagens, foi encontrado menos falsos positivos que no conjunto anterior, conforme é possível observar nos Gráficos das Figuras 5.43 e 5.44, em que os

pontos vermelhos representam as imagens em que não foi encontrado nenhum círculo, existem pontos em que há essa diferença de posição ou raio de uma imagem para outro, representando um falso positivo.

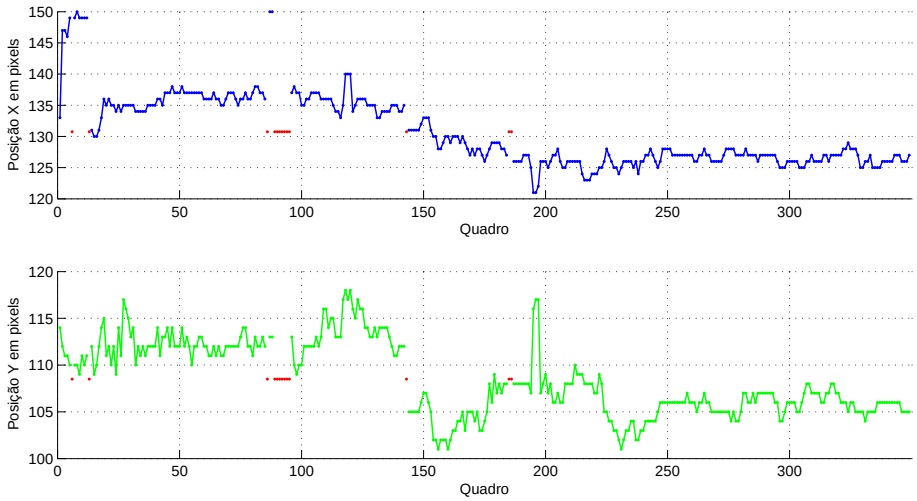


Figura 5.43: Variação da posição x (azul) e y (verde) do círculo encontrado em cada imagem do Segundo Conjunto, utilizando o Sistema em Matlab.

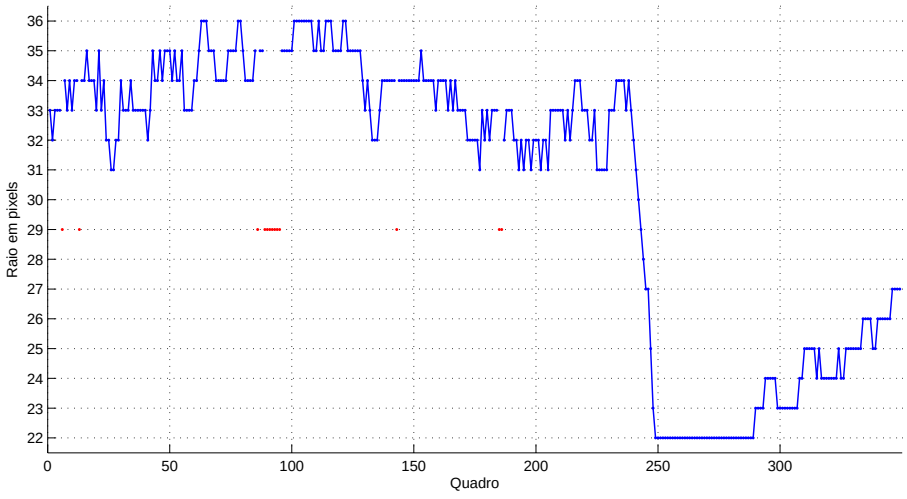


Figura 5.44: Variação do raio do círculo encontrado em cada imagem do Segundo Conjunto, utilizando o Sistema em Matlab.

O Gráfico da Figura 5.45, mostra a distribuição de frequência do tempo gasto no pro-

cessamento de cada imagem do olho, e é possível perceber que o tempo de processamento de cada imagem está entre 300 *ms* e 3,5 *segundos* ou 3500 *ms*, com uma média de 759 *ms*.

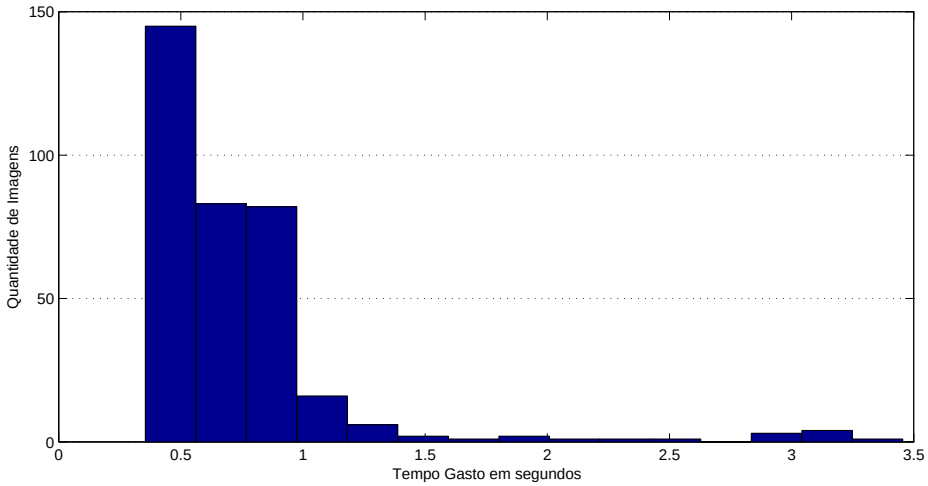


Figura 5.45: Distribuição de frequência do tempo gasto no processamento de cada imagem do Segundo Conjunto, utilizando o Sistema em Matlab.

Conjunto Submetido ao Sistema Proposto

As Imagens deste segundo conjunto foram submetido ao sistema proposto em duas situações: a mesma situação do sistema em [11], em que não há o corte da imagem nem um limite de busca de raios; e mais um teste foi com o sistema completo, com o corte da imagem e com um limite de busca pelo raio de dois pixels acima e abaixo do raio encontrado na imagem anterior.

- **Sem Corte na Imagem e Sem Limite de Busca no Raio:**

  Utilizando os dados obtidos do processamento de todas as 349 imagens observa-se que foi encontrado círculos em todas as imagens, sendo que, assim como no teste em Matlab foram encontrados menos falsos positivos que no conjunto anterior, conforme é possível observar nos Gráficos das Figuras 5.46 e 5.47, que existem pontos em que há essa diferença de posição ou raio de uma imagem para outro, representando um falso positivo.

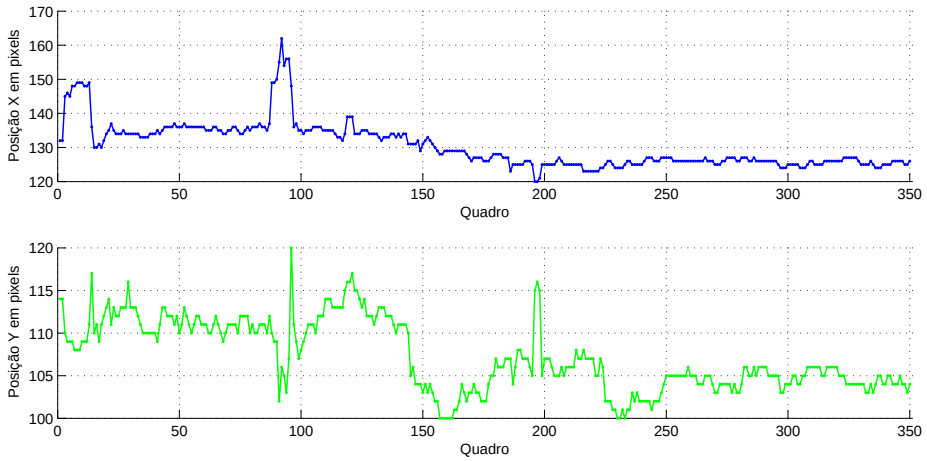


Figura 5.46: Variação da posição x (azul) e y (verde) do círculo encontrado em cada imagem do Segundo Conjunto, utilizando o Sistema Proposto na Situação 1.

Figura 5.47: Variação do raio do círculo encontrado em cada imagem do Segundo Conjunto, utilizando o Sistema Proposto na Situação 1.

O Gráfico da Figura 5.48, mostra a distribuição de frequência do tempo gasto no processamento de cada imagem do olho, e é possível perceber que o tempo de processamento de cada imagem está entre 200 *ms* e 400 *ms*, com uma média de 343,5 *ms*.

- **Com Corte na Imagem e Com Limite de 2 pixels acima e abaixo na Busca**

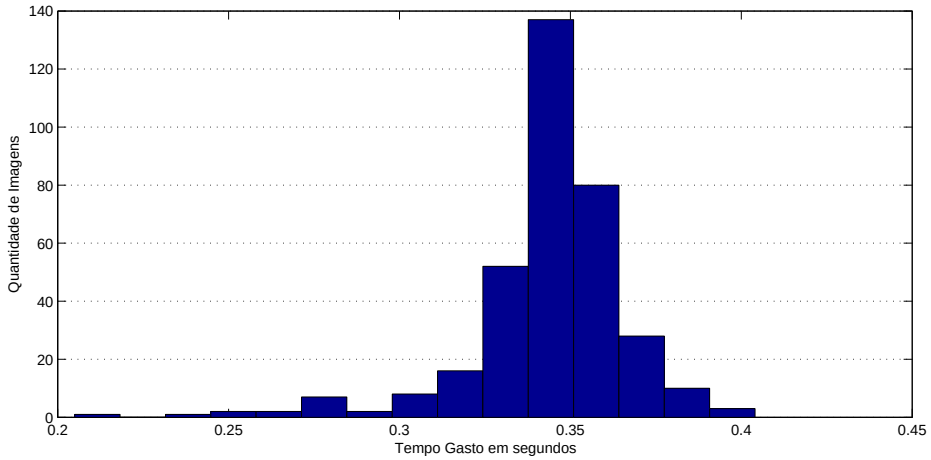


Figura 5.48: Distribuição de frequência do tempo gasto no processamento de cada imagem do Segundo Conjunto, utilizando o Sistema Proposto na Situação 1.

pelo Raio:

Utilizando os dados obtidos do processamento de todas as 349 imagens observa-se que de todas as imagens, foi encontrado círculos em 348 imagens, conforme é possível observar nos Gráficos das Figuras 5.49 e 5.50 existe apenas uma imagem que não foi possível encontrar um círculo, e o gráfico ficou mais contínuo em alguns pontos, principalmente se comparado o Gráfico da Figura 5.47 com o da Figura 5.49

O tempo de processamento das imagens ficou entre 10 *ms* e 35 *ms*, com uma média de 19,5 *ms* conforme pode ser observado no Gráfico da Figura 5.51, que mostra a distribuição de frequência do tempo gasto no processamento de cada imagem do olho.

Discussão dos Resultados

Dos resultados mostrados anteriormente, é possível perceber que a principal diferença nos resultados encontrados se refere ao tempo de processamento. Já nos quesitos posição e raio do círculo encontrado, os resultados obtidos foram semelhantes:

- **Tempo de Processamento de Cada Imagem:**

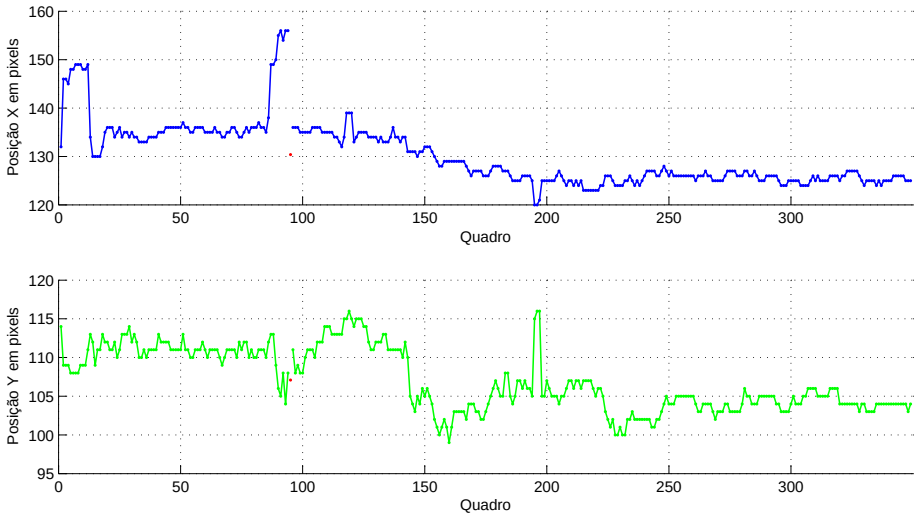


Figura 5.49: Variação da posição x (azul) e y (verde) do círculo encontrado em cada imagem do Segundo Conjunto, utilizando o Sistema Proposto na Situação 2.

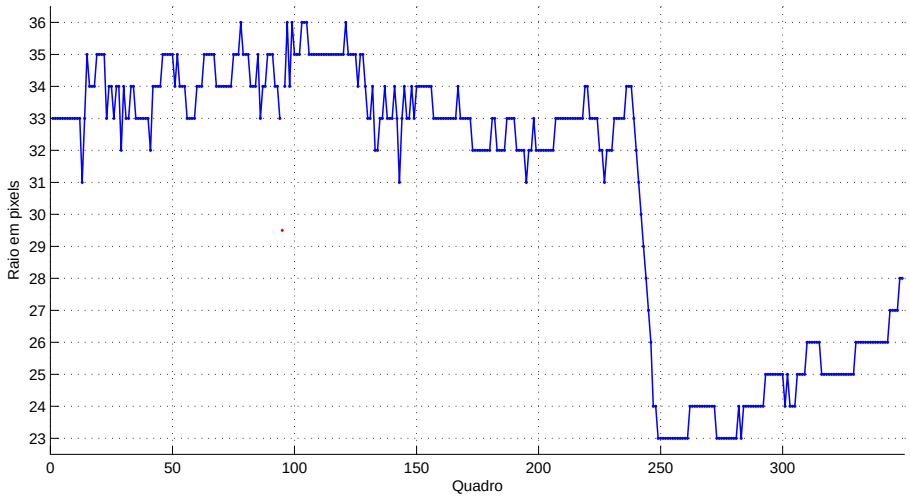


Figura 5.50: Variação do raio do círculo encontrado em cada imagem do Segundo Conjunto, utilizando o Sistema Proposto na Situação 2.

Observando os Gráficos de Distribuição de Frequência do tempo gasto em cada imagem para o Sistema em Matlab e o Sistema em C# com as mesmas configurações, Figuras 5.45 e 5.48, já é possível observar que o Sistema em C# é mais rápido, enquanto em Matlab o tempo médio gasto em uma imagem é de 759 *ms* em C# esse

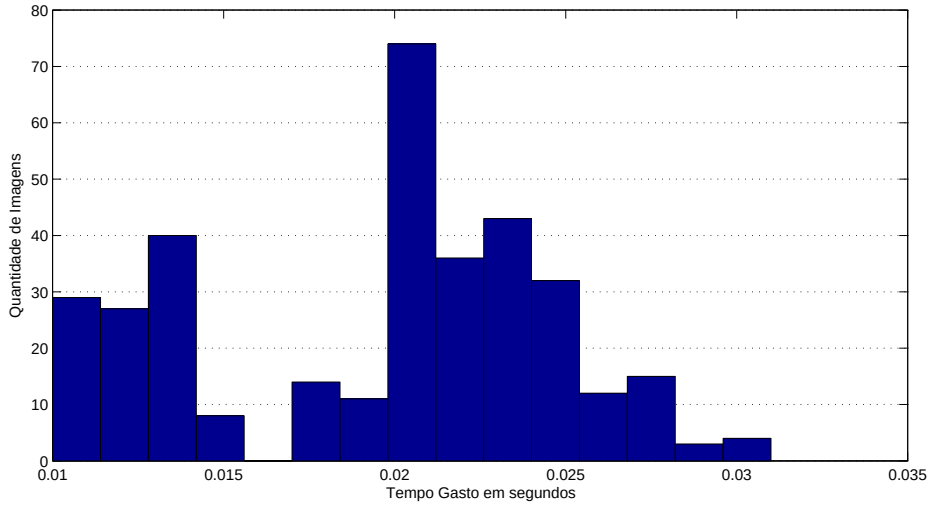


Figura 5.51: Distribuição de frequência do tempo gasto no processamento de cada imagem do Segundo Conjunto, utilizando o Sistema Proposto na Situação 2, excluindo o tempo de processamento das imagens inteiras.

tempo cai para 343,5 *ms*, menos do que a metade do tempo gasto no Sistema anterior. Comparando, então, com a proposta do presente trabalho, efetuando um corte na imagem (diminuindo informações desnecessárias) e limitando a busca por valores de raios mais próximos do encontrado anteriormente, esse tempo diminui ainda mais, 19,5 *ms*, pouco menos do que 7% do tempo gasto no Sistema em Matlab, comprovando assim a eficiência do Sistema proposto.

- **Precisão da Posição e Raio do Círculo Encontrado:**

  Diferente do que ocorreu no Primeiro conjunto de Imagens, a posição dos círculos encontrados tanto para o sistema em Matlab quanto para o Sistema Proposto estão bem próximas, como pode ser observado no Gráfico da Figura 5.52 em que a cor Verde representa o Sistema Proposto e a cor Azul o Sistema em Matlab, e os pontos em vermelho representam as imagens com círculos não encontrados para o Sistema em Matlab.

  Assim como na posição, os raios dos círculos encontrados para o Sistema em Matlab e em C# foram bem próximos, conforme o Gráfico da Figura 5.53 em que a cor Verde

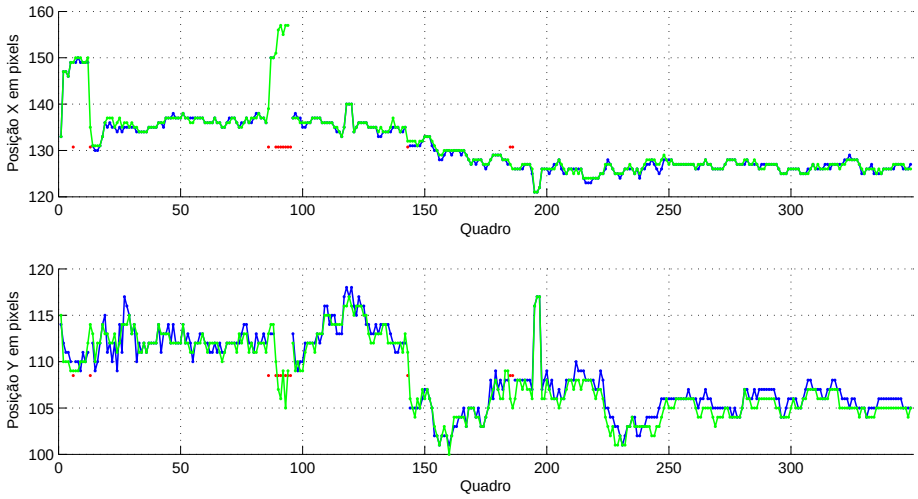


Figura 5.52: Posição X e Y do círculo encontrado em cada imagem do Segundo Conjunto para o Sistema em Matlab (Azul) e C# (Verde) excluindo os círculos não encontrados.

representa o Sistema Proposto e a cor Azul o Sistema em Matlab, e os pontos em vermelho representam as imagens com círculos não encontrados para o Sistema em Matlab.

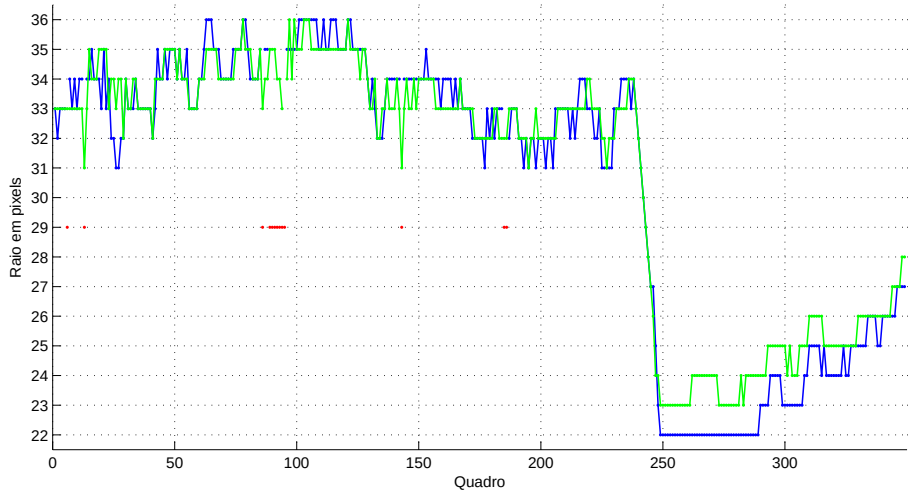


Figura 5.53: Tamanho do Raio do círculo encontrado em cada imagem do Segundo Conjunto para o Sistema em Matlab (Azul) e C# (Verde) excluindo os círculos não encontrados.

A diferença dos resultados entre os Sistemas foi que no Sistema Proposto não foi

encontrado círculo em apenas uma imagem, já no Sistema em Matlab foram em 13, porcentagem bem menor do que no Primeiro Conjunto.

Observando os Gráficos deste segundo conjunto de imagens submetido ao Sistema Proposto é necessário entender qual o motivo de não serem encontrados círculos em determinadas imagens, daí, pelo Gráfico da Figura 5.49 ou 5.50, observa-se que não foi possível encontrar a pupila na imagem de número 95, Figura 5.54, em que houve um movimento rápido do olho em relação ao eixo X, justificando a variação abrupta da posição X encontrado no Gráfico da Figura 5.49 logo após o ponto vermelho.

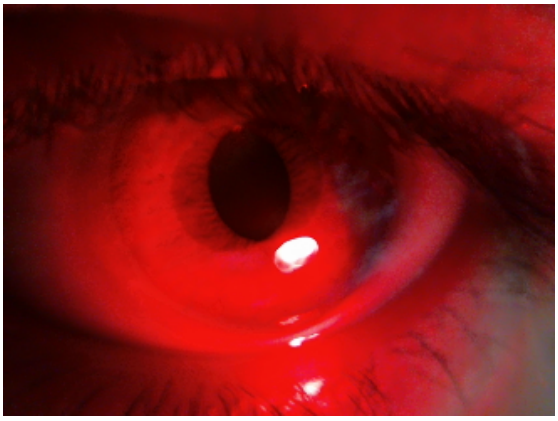


Figura 5.54: Imagem do Segundo Conjunto que não foi possível encontrar um círculo.

Houve também uma variação repentina da posição em relação ao eixo Y, conforme observado também no Gráfico da Figura 5.49, da imagem 194 à 199, em que foi possível encontrar o círculo em todos pois o movimento não foi captado pela câmera como na Figura 5.54, mas conforme as imagens de corte da Figura 5.55 é possível observar que houve movimento de uma imagem para outra e o corte na próxima logo é feito em torno do círculo encontrado na imagem anterior.

Assim como no caso do Primeiro Conjunto de Imagens, neste conjunto também houve um caso em que foi alterada a iluminação, imagem *b* da Figura 5.56, porém neste caso o Sistema Proposto foi capaz de encontrar a pupila mesmo com essa alteração repentina. Daí a importância de utilizar um olho com a iluminação infra-vermelha sendo filmado e o outro olho com a luz variando a intensidade, pois dessa forma todas as imagens captadas

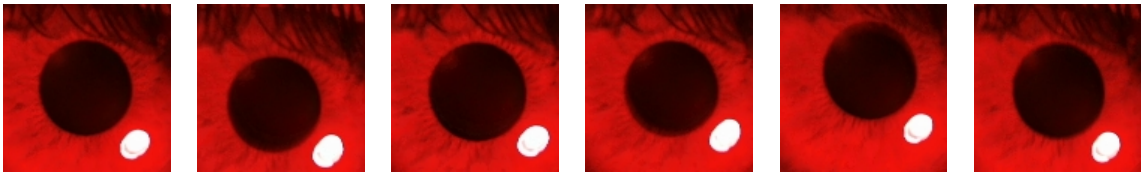
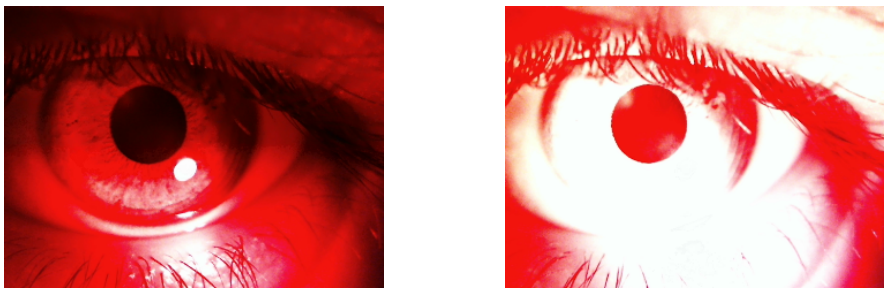


Figura 5.55: *Imagens sequenciais de corte do Segundo conjunto mostrando a movimentação rápida em relação ao eixo Y.*

estariam com o mesmo brilho, reduzindo o risco de não ser encontrado um círculo devido a grande variação de iluminação.



(a) *Imagem Normal.* (b) *Imagem Clara.*

Figura 5.56: *Imagens de exemplo da alteração da iluminação para o Segundo Conjunto.*

5.4 Teste de Precisão

Para o teste de Precisão foi utilizado imagens do banco de dados de Iris CASIA [15] submetidas a um software conhecido que efetua a busca pela pupila em uma imagem do olho [16], ao software de [11] e ao software proposto.

Foram utilizadas imagens pertencentes ao conjunto CASSIA-Iris-Interval, das 10 primeiras pessoas, algumas com imagens do olho esquerdo e direito, da imagem S1001L01.jpg até a imagem S1010R05.jpg, totalizando 122 imagens.

Essas imagens foram submetidas aos 3 softwares, adquirindo os dados da pupila encontrada, posição e raio, e esses dados foram analisados em relação a diferença encontrada de um software para outro, comparando os resultados do software proposto com de Masek [16] e com o do trabalho [11]:

5.4.1 Comparação do Sistema Proposto com o Sistema do Masek

O Gráfico da Figura 5.57 mostra a distribuição de frequência da diferença do raio encontrado para cada imagem em cada sistema.

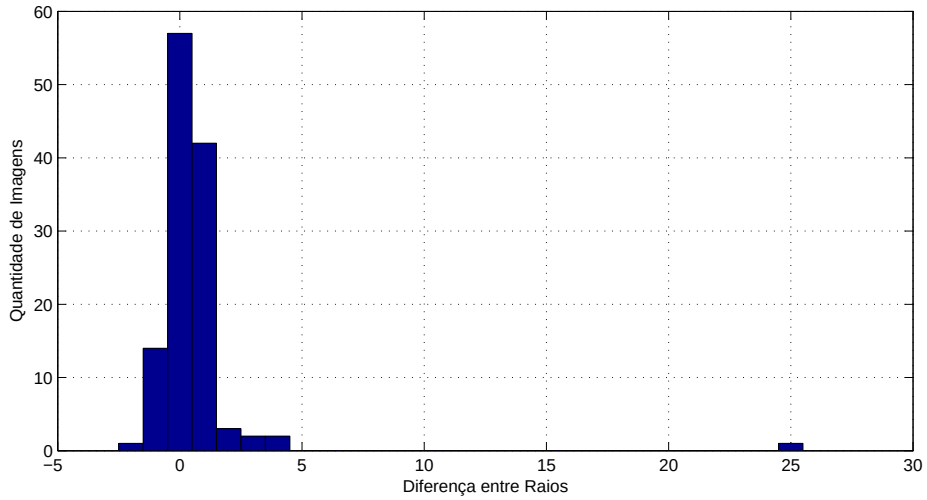


Figura 5.57: Distribuição de Frequência da Diferença do raio encontrado para cada imagem.

Os Gráficos das Figuras 5.58 e 5.59 mostram a distribuição de frequência da distância do centro encontrado para cada imagem em cada sistema, com a diferença que o segundo exclui as distâncias acima de 8 pixels.

Pela Figura 5.57 é possível perceber que o raio encontrado em uma imagem destoa de um sistema para o outro, essa imagem é a da Figura 5.60, S1003R03.jpg, em que o círculo em vermelho foi encontrado pelo sistema do Masek [16] e o círculo em verde pelo Sistema Proposto, essa mesma imagem é uma das que possui uma distância grande (24 pixels) entre os centros encontrados da Figura 5.58.

Duas outras imagens foram selecionadas da Figura 5.58, a que possui a distância de 8 pixels e 99 pixels, que são as Imagens da Figura 5.61, em que, novamente, o círculo em vermelho foi encontrado pelo sistema do Masek e o círculo em verde pelo Sistema Proposto.

Utilizando os dados dos Gráficos das Figuras 5.58 e 5.57 chegou-se a uma média da diferença de raio encontrado de 0,58 pixels e uma média de distância de 3,36 pixels. Excluindo o Raio e as Distâncias que se sobressaíram nos Gráficos obteve-se uma média da

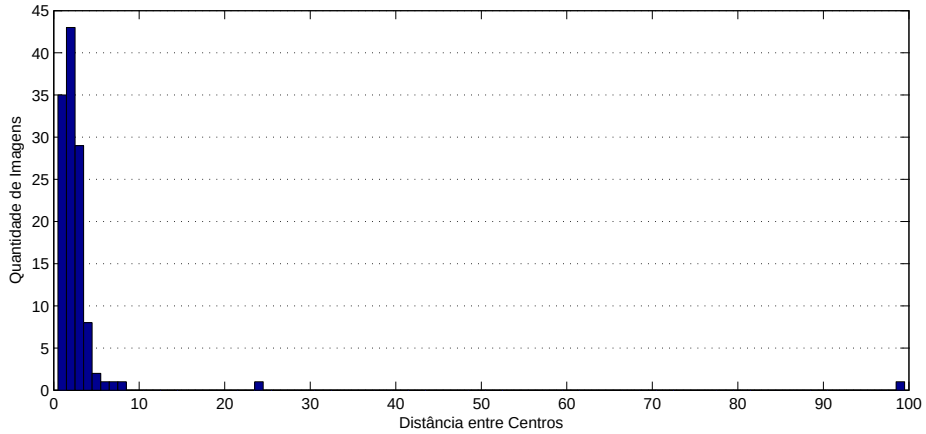


Figura 5.58: Distribuição de frequência da Distância do centro encontrado para cada imagem.

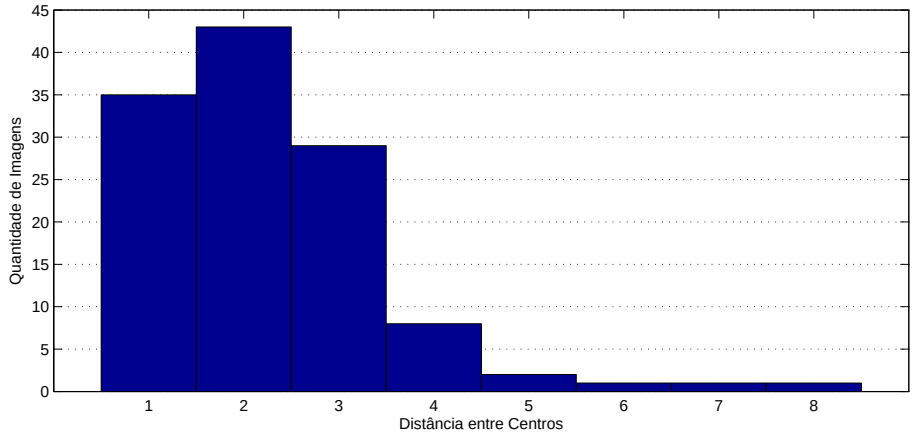


Figura 5.59: Distribuição de frequência da distância do centro encontrado para cada imagem, excluindo distâncias acima de 8 pixels.

diferença de raio encontrado de 0,38 pixels e uma média de distância de 2,39 pixels.

5.4.2 Comparação do Sistema Proposto com o Sistema Matlab

O Gráfico da Figura 5.62 mostra a distribuição de frequência da diferença do raio encontrado para cada imagem em cada sistema.

O Gráfico da Figura 5.63 mostra a distribuição de frequência da distância do centro

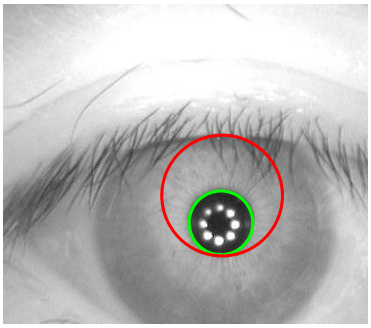
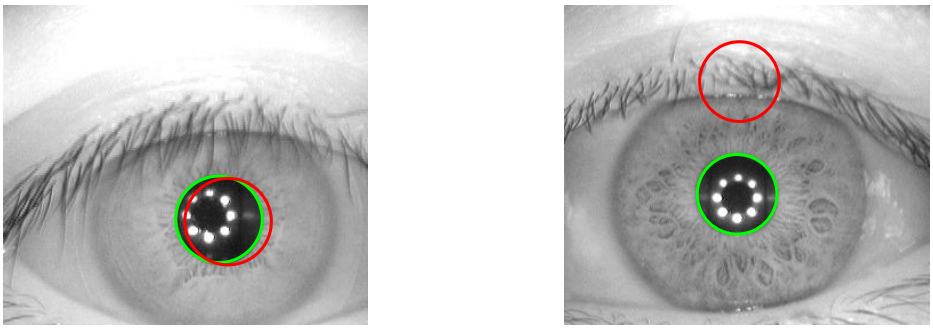


Figura 5.60: Imagem em que a pupila encontrada possui tamanho diferente entre o software do Masek e o proposto.



(a) *S1005R03.jpg* (b) *S1008L03.jpg*

Figura 5.61: Imagem em que a pupila encontrada possui posição diferente entre o software do Masek e o proposto

encontrado para cada imagem em cada sistema.

Utilizando os dados dos Gráficos das Figuras 5.63 e 5.62 chegou-se a uma média da diferença de raio encontrado de 0,16 pixels e uma média de distância de 1,00 pixels, valores menores do que os encontrados no Ítem 5.4.1.

Desta maneira, é possível, verificar que o Sistema Proposto é capaz de encontrar com precisão a posição e o raio da pupila para as imagens do Banco de Dados CASIA.

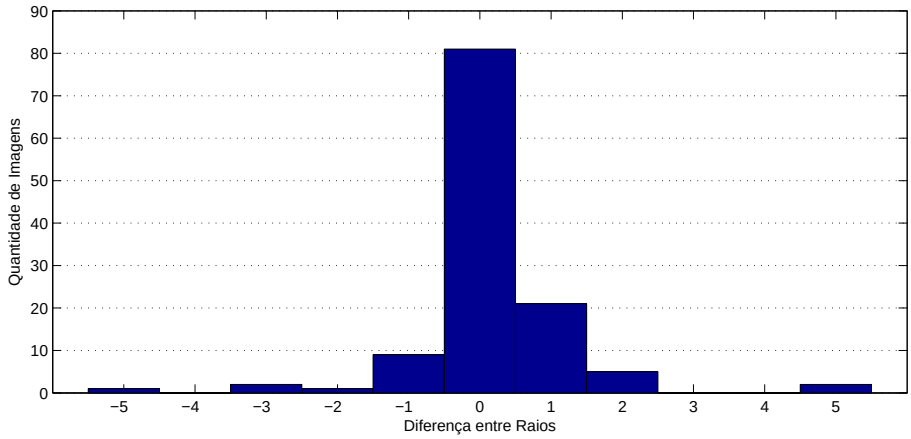


Figura 5.62: Distribuição de Frequência da Diferença do raio encontrado para cada imagem.

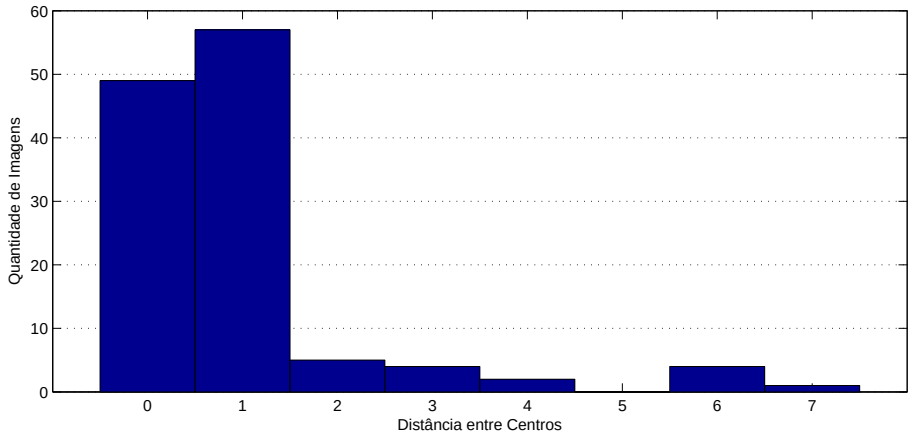


Figura 5.63: Distribuição de frequência da distância do centro encontrado para cada imagem.

5.5 Considerações Finais do Capítulo

Os resultados obtidos e demonstrados aqui, em relação a possibilidade de realizar a pupilometria dinâmica em tempo real, redução do tempo de processamento e precisão na posição e tamanho da pupila encontrada, comprova a utilidade do sistema proposto.

No próximo Capítulo será apresentado as conclusões, associando as principais contribuições, juntamente com sugestões para trabalhos futuros que podem ser desenvolvidos a partir desta Dissertação.

6 CONCLUSÕES, CONTRIBUIÇÕES E TRABALHOS FUTUROS

6.1 Introdução

Essa Dissertação apresenta as etapas necessárias para realizar o rastreamento da pupila de um olho humano em tempo real. Foi desenvolvido um Software capaz de efetuar o processamento em tempo de vídeo, assim que a imagem é adquirida, logo ela já é processada, sendo possível processar no mínimo 50 imagens por segundo ou uma média de 123, conforme mostrado no Capítulo 5.

A utilização do conjunto de Bibliotecas da AForge.NET [39] foi de grande importância pois auxiliou para que o Sistema se tornasse mais eficiente em relação ao tempo, sendo utilizado em todo o Sistema, desde a Aquisição, passando pelo Pré-Processamento, Segmentação até o Reconhecimento, explicados no Capítulo 2, detalhado a sua utilização no Capítulo 4.

A Transformada Circular de Hough se mostrou muito eficiente e precisa na busca de formas circulares em imagens digitais, conforme já tinha sido explorada no trabalho de Bernadelli [11].

Técnicas foram utilizadas para se obter melhor precisão tanto da posição quanto do raio da pupila encontrada em cada imagem subsequente, informações da pupila encontrada no frame anterior foram utilizadas no frame atual, a posição para efetuar o corte na imagem,

e o raio para limitar o raio a ser encontrado.

Utilizando o corte da imagem adquirida na posição do círculo encontrado da imagem anterior favoreceu o Sistema proposto em duas situações distintas: melhor precisão da posição do próximo círculo, não sendo possível encontrar círculos fora do espaço delimitado pelo corte; redução de informações desnecessárias na imagem, reduzindo o tempo de processamento.

Assim como a utilização do corte favoreceu o Sistema em duas situações, a imposição de um limite de raios a ser procurado também favorece e nas mesmas condições: melhor precisão do raio do próximo círculo, reduzindo a chance de se encontrar um falso positivo; redução de buscas desnecessárias pois o tamanho da pupila sempre é próxima da encontrada anteriormente, reduzindo novamente o tempo de processamento.

Desta forma, é possível perceber que as técnicas de processamento de vídeos utilizadas foram de suma importância para alcançar um processamento em tempo real.

Segundo os testes realizados no Capítulo 5 observa-se também a importância de utilizar a iluminação infra-vermelha no olho que se está adquirindo a imagem, pois, dessa maneira, é possível fixar os parâmetros do pré-processamento para diferentes testes e cores de luz incidente, sendo que para os Ítens 5.3.1 e 5.3.2 poderia ser utilizado o mesmo parâmetro para binarização, assim como foi utilizado na Seção 5.4, que foi efetuado a busca pela pupila em imagens de pessoas diferentes e olhos diferentes.

Além da possibilidade de fixação dos parâmetros do pré-processamento, a utilização do infra-vermelho também possibilitaria filmar o olho com o outro em total escuridão, e quando fosse alterada repentinamente a iluminação do outro olho para 100%, o olho filmado continuaria com a mesma claridade (infra-vermelha), não influenciando na busca pela pupila.

6.2 Principais Contribuições

A principal contribuição deste Trabalho é a possibilidade da realização da Pupilometria Dinâmica em Tempo Real, através da utilização da linguagem C#, e principalmente da técnica de adquirir as informações da pupila encontrada na imagem anterior, realizando o corte e limitando o raio da pupila a ser encontrada.

Outra grande contribuição foi o ganho de precisão na posição e no raio das pupilas encontradas em imagens sequenciais, que ocorreu também pela utilização das informações da pupila encontrada na imagem anterior.

Dessa forma, torna-se mais próxima a possibilidade da realização da Pupilometria Dinâmica em dispositivos portáteis, como *Smartphones*, que estão cada vez mais rápidos e acessíveis.

Outra oportunidade que este Trabalho trás é que, através da realização da Pupilometria Dinâmica em Tempo Real, diagnósticos de doenças ou distúrbios relacionados ao sistema nervoso simpático e parassimpático seriam realizados de forma mais rápida.

6.3 Publicações

- A. G. C. Dias, A. C. P. Veiga, M. B. P. Carneiro, C. R. Bernadelli. RASTREAMENTO DAS CARACTERÍSTICAS DA PUPILA HUMANA EM TEMPO REAL. X CEEL - ISSN 2178-8308 (CEEL-2012, CD-ROM). Fevereiro de 2012. Uberlândia, MG – Brasil.

- A. G. C. Dias, A. C. P. Veiga. A REAL TIME HUMAN PUPIL TRACKING PROPOSAL. The 4th IEEE Biosignals & Biorobotics Conference (ISSNIP-2013, CD-ROM). Fevereiro de 2013. Rio de Janeiro, RJ - Brasil.

6.4 Trabalhos Futuros

O Sistema Proposto no presente Trabalho alcançou os Objetivos, porém existem outras possibilidades para explorá-lo ainda mais:

- Utilização de melhores equipamentos tanto de aquisição de imagens quanto de processamento, possibilitando maiores resoluções temporais e espaciais;
- Aperfeiçoamento das Técnicas utilizadas de forma a diminuir ainda mais o tempo de processamento de cada imagem, por exemplo efetuando uma estimativa da posição do círculo a ser encontrado;
- Implementar o sistema completo, com iluminação infra-vermelha, no olho a ser filmado, e iluminação variante de cor e intensidade no outro olho, possibilitando essa alteração de cor e intensidade na interface do usuário.
- Embarcar o Sistema de forma a ficar mais prática a sua utilização, e ampliando as possibilidades para quem o utiliza.
- Comparar a saída do Sistema com modelos pré-existentes de como a pupila reage a determinadas variações de iluminação.
- Utilizar o Sistema Proposto para criar um banco de vídeo de pessoas saudáveis e com doenças ligadas ao sistema nervoso, com o objetivo de aumentar o estudo nesta área.
- Estimar a complexidade computacional entre o Sistema sem utilização dos dados do círculo encontrado no quadro anterior e com a utilização dos mesmos.

Referências Bibliográficas

[1] P. L. Kaufman and A. Alm. *Adler's Physiology of the Eye*. Elsevier, 2011.

[2] P. Schor. *Óptica Fisiológica e Cirurgia Refrativa*. PhD thesis, Universidade Federal de São Paulo – Escola Paulista de Medicina, 2003.

[3] P. Brodal. *The Central Nervous System: Structure and Function*. Oxford University Press, 2010.

[4] G. N. S. M. Leal. Desenvolvimento de um pupilómetro. Master's thesis, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa (FCT-UNL), 2008.

[5] G. L. Ferrari. 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. Master's thesis, Universidade Tecnológica Federal do Paraná, 2010.

[6] A Tales, T Troscianko, D Lush, J Haworth, GK Wilcock, SR Butler, et al. The pupillary light reflex in aging and alzheimer's disease. *Aging (Milan, Italy)*, 13(6):473, 2001.

[7] F. Fotiou, K. N. Fountoulakis, A. Goulas, L. Alexopoulos, and A. Palikaras. Automated standardized pupillometry with optical method for purposes of clinical practice and research. *Clinical Physiology*, 20(5):336 – 347, 2000.

[8] J. Kim, K. Park, and G. Khang. A method for size estimation of amorphous pupil in 3-dimensional geometry. 1:1451 – 1454, 2004.

[9] V. F. Pamplona and M. M. Oliveira. Photorealistic models for pupil light reflex and iridal pattern deformation. Master's thesis, Universidade Federal do Rio Grande do Sul, 2008.

[10] N. Link and L. Stark. Latency of the pupillary response. *Biomedical Engineering, IEEE Transactions on*, 35(3):214–218, 1988.

[11] Cláriton Rodrigues Bernadelli. Rastreamento em vídeo das características da pupila. 2011.

[12] S. A. Smith and R. R. Dewhirst. A simple diagnostic test for pupillary abnormality in diabetic autonomic neuropathy. *Diabetic medicine*, 3(1):38 – 41, 1986.

[13] S. Wibirama, S. Tungjitkusolmun, C. Pintavirooj, and K. Hamamoto. Real time eye tracking using initial centroid and gradient analysis technique. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2009. ECTI-CON 2009. 6th International Conference on*, volume 2, pages 1054–1057. IEEE, 2009.

[14] M. Soltany, S.T. Zadeh, and H.R. Pourreza. Fast and accurate pupil positioning algorithm using circular hough transform and gray projection. In *Proceedings of International Conference on Computer Communication and Management (ICCCM 2011)*, 2011.

[15] AForge.NET. CASIA Iris Image Database. http://biometrics.idealtest.org/. [acessado em 12-Maio-2012].

[16] Libor Masek et al. *Recognition of human iris patterns for biometric identification.* PhD thesis, Master's thesis, University of Western Australia, 2003.

[17] Hélio Pedrini and William Robson Schwartz. *Análise de imagens digitais: princípios, algoritmos e aplicações.* Thomson Learning, 2008.

[18] Richard E Gonzalez, Rafael C e Woods. *Processamento de imagens digitais.* Pearson, 2009.

[19] John Daugman. How iris recognition works. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1):21–30, 2004.

[20] Alan Conrad Bovik. *The essential guide to Image Processing.* Academic Press, 2009.

[21] M. S. Nixon and A. S. Aguado. *Feature Extraction and Image Processing.* Academic Press, 2008.

[22] Simon J.K. Pedersen. Circular hough transform. *Aalborg University, Vision, Graphics, and Interactive Systems*, 2007.

[23] Jack Bresenham. A linear algorithm for incremental digital display of circular arcs. *Communications of the ACM*, 20(2):100–106, 1977.

[24] BO Schneider. Apostila de computação gráfica, 2001.

[25] Alan Conrad Bovik. *The essential guide to video processing.* Academic Press, 2009.

[26] André Hermenegildo Nascimento. Rastreamento de objetos por similaridade de características: Uma aplicação para compressão e indexação de vídeos. Master's thesis, Universidade de Pernambuco, 2011.

[27] C. R. Bernadelli and A. C. P. Veiga. Iris motion tracking using feature extraction by shape matching. *XIII Symposium on Virtual and Augmented Reality - SVR-2011, CD-ROM*, May/2011.

[28] F. Dadgostar and A. Sarrafzadeh. An adaptive real-time skin detector based on hue thresholding: A comparison on two motion tracking methods. *Pattern recognition letters*, 27(12):1342 – 1352, 2006.

[29] R. A. de Bem. Uma abordagem livre de modelo oara rastreamento de objetos em sequências de imagens. Universidade de São Paulo, São Paulo, Brasil, 2007.

[30] C. Y. V. W. da Silva. Extração de carcterísticas de imagens médicas utilizando wavelets para miniração de imagens e auxílio ao diagnóstico. Universidade de São Paulo, São Paulo, Brasil, 2007.

[31] S. Ordas, L. Boisrobert, M. Huguet, and A. F. Frangi. Active shape models with invariant optimal features (iof-asm) application to cardiac mri segmentation. In *Computers in Cardiology, 2003*, pages 633 – 636. IEEE, 2003.

[32] M. J. Vasconcelos and J. M. R. da Silva Tavares. Modelos pontuais de distribuição em visão computacional. In *6º Congresso Nacional Mecânica Experimental*, 2005.

[33] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Training models of shape from sets of examples. In *Proc. British Machine Vision Conference*, volume 9, page 18. Citeseer, 1992.

[34] A. B. V. Graciano. Rastreamento de objetos baseado em reconhecimento estrutural de padrões. Universidade de São Paulo, São Paulo, Brasil, Março 2007.

[35] Simon Medeiros Soares, Thiago de Castro Turino, Mariane Rembold Petraglia, José Gabriel Rodriguez Carneiro Gomes, and Aloysio de Castro Pinto Pedroza. Rastreamento de objetos utilizando processamento de imagem.

[36] A. B. de Oliveira. Filtro de partículas adaptativo para o tratamento de oclusões no rstreamento de objetos em vídeos. Universidade Federal do Rio Grande do Sul, Porto Alegre, Brasil, Maio 2008.

[37] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, 2008.

[38] Microsoft Visual Studio. Visual C# 2010 Express. http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-csharp-express. [acessado em 19-MarÃ§o-2012].

[39] AForge.NET. AForge.NET Framework 2.2.4. http://www.aforgenet.com/news/2012.02.23.releasing_framework_2.2.4.html. [acessado em 20-MarÃ§o-2012].

[40] Biblioteca MSDN. Desenvolvimento .NET. http://msdn.microsoft.com/pt-br/library/aa139615. [acessado em 21-MarÃ§o-2012].

[41] AForge.NET. AForge.NET Framework Documentation. http://www.aforgenet.com/framework/docs/. [acessado em 21-MarÃ§o-2012].

[42] Logitech. Logitech HD Webcam C525. http://www.logitech.com/en-us/webcam-communications/webcams/hd-webcam-c525. [acessado em 23-MarÃ§o-2012].

[43] B. Sahin, B. Lamory, X. Levecq, F. Harms, and C. Dainty. Adaptive optics with pupil tracking for high resolution retinal imaging. *Biomedical optics express*, 3(2):225–239, 2012.