UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

THALES LEVI AZEVEDO VALENTE

DIAGNÓSTICO DE ESTRABISMO EM IMAGENS DIGITAIS UTILIZANDO DISPOSITIVOS MÓVEIS

São Luís
2015

THALES LEVI AZEVEDO VALENTE

DIAGNÓSTICO DE ESTRABISMO EM IMAGENS DIGITAIS UTILIZANDO DISPOSITIVOS MÓVEIS

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. João Dallyson Sousa de Almeida

São Luís
2015

Valente, Thales Levi Azevedo

Diagnóstico de estrabismo em imagens digitais utilizando dispositivos móveis/ Thales Levi Azevedo Valente. – São Luís, 2015.

77 f.
Orientador: Prof. Dr. João Dallyson Sousa de Almeida.

Monografia (Graduação) – Universidade Federal do Maranhão, Curso de Ciência da Computação, 2015.

1. Processamento de imagens 2. Estrabismo 3. Teste de Hirschberg 4. Android

CDU 004.932.2

THALES LEVI AZEVEDO VALENTE

DIAGNÓSTICO DE ESTRABISMO EM IMAGENS DIGITAIS UTILIZANDO DISPOSITIVOS MÓVEIS

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão para obtenção do grau de Bacharel em Ciência da Computação.

Aprovada em: 16 / 01 / 2015

BANCA EXAMINADORA

Prof. Dr. João Dallyson Sousa de Almeida (Orientador)
Doutor em Engenharia da Eletricidade
Universidade Federal do Maranhão

Prof. Dr. Aristófanes Correa Silva
Doutor em Informática
Universidade Federal do Maranhão

Prof. Dr. Geraldo Braz Junior
Doutor em Engenharia da Eletricidade
Universidade Federal do Maranhão

À Deus, minha família, professores e amigos.

AGRADECIMENTOS

Primeiramente a Deus que me permitiu todas as alegrias, conquistas e que me deu forças para lutar mesmo nos momentos mais difíceis.

Minha família pelo suporte durante todos os anos da minha vida, especialmente aos meus pais Aquiles Valente e Leonilde Valente, pelo apoio e incentivo ao estudo, e aos meus irmãos Thalita Valente e Thiago Valente, pelo apoio e acompanhamento fraterno.

Minha namorada Lillian Garcês, pelo amor, apoio, compreensão e companheirismo, nos 2 anos mais significativos da minha graduação.

Todos os meus professores, pelo conhecimento que me proporcionaram. Em especial ao amigo, professor e orientador João Dallyson, pela orientação, paciência e conselhos, nessa monografia e em outros trabalhos realizados no decorrer da minha vida acadêmica. Ao professor Geraldo Braz, que me deu apoio e incentivo desde o início da minha jornada acadêmica. Ao professor Aristófanes Corrêa, pelo acompanhamento e, junto com o professor Geraldo Braz, pela oportunidade de participar do NCA, fato que marcou a minha vida acadêmica. Ao professor Jorge Antonio, pelos conselhos no decorrer do desenvolvimento deste trabalho.

Meus amigos e companheiros na jornada de graduação que estiveram sempre dispostos a ajudar, compartilhando da luta e colaborando com essa conquista. E para a todos que contribuíram direta ou indiretamente para a conclusão deste trabalho.

"O que adquire conhecimento ama a sua alma. O que conserva a inteligência faz o bem." (Salomão)

RESUMO

Com o avanço da tecnologia *mobile*, o aumento da capacidade de processamento dos dispositivos móveis bem como o surgimento de sistemas mais robustos para os mesmos, foi possível o surgimento de novas linhas de pesquisas visando criar novas ferramentas e aplicações. A maioria dessas novas linhas de pesquisa tem por finalidade automatizar alguma tarefa, consequentemente deixando-a mais veloz, confiável, segura e com a vantagem da portabilidade e praticidade que estes dispositivos oferecem. O processamento de imagens médicas tem contribuído para a detecção e o diagnóstico de anomalias no corpo humano, sendo uma importante ferramenta na redução do grau de incerteza do diagnóstico, provendo uma fonte adicional de informação ao especialista. Uma dessas anomalias é o estrabismo, que afeta aproximadamente 4% da população provocando problemas estéticos, reversíveis a qualquer idade, modificando o mecanismo de visão. Conquanto, a utilização de recursos de alta tecnologia *mobile* no auxílio diagnóstico e terapêutico na subespecialidade estrabismo ainda não é uma realidade. Sendo assim, este trabalho tem como objetivo portabilizar a metodologia proposta por Almeida (2013), baseada no Teste de Hirschberg, para o ambiente móvel Android, afim de comparar os resultados obtidos e de servir de auxílio para o especialista. Para tanto, o este trabalho está organizado em oito fases de execução: aquisição de imagens, segmentação da face, localização da região dos olhos, localização dos olhos, localização do limbo, localização do brilho, detecção de estrabismo e diagnóstico de estrabismo.

Palavras-chave: Processamento de imagens. Estrabismo. Teste de Hirschberg. Android

ABSTRACT

With the advancement of mobile technology, the increase in mobile devices processing power and the appearance of more robust systems for them, was possible the appearance of new lines of research to create new tools and applications. Most of these new lines of research aims to automate some task, thus making it more rapid, reliable, secure and with the advantage of portability and convenience that these devices offer. The medical image processing has contributed to the detection and diagnosis of anomalies in the human body, is an important tool in reducing the degree of uncertainty of the diagnosis, providing an additional source of information to the specialist. One of these anomalies is the Strabismus that affects about 4% of the population causing aesthetic problems, reversible at any age, changing the viewing mechanism. However, the application of the high technology mobile features in the diagnosis and therapeutic aid subspecialty strabismus is not yet a reality. Thus, this study aims make portable the methodology proposed by Almeida (2013), based on the Hirschberg test for the mobile environment Android, in order to compare the results and be a help to the expert. Therefore, the application is organized into eight stages: image acquisition, face segmentation, detecting the eye region, detecting the eyes, limbus location, brightness location, strabismus detection and diagnosis of strabismus.

Keywords: Image processing. Strabismus. Hirschberg Test. Android

LISTA DE FIGURAS

LISTA DE SIGLAS

ADB	- Android Debug Bridge
API	- Application Programming Interface
CAD	- Computer-Aided Diagnosis
DDMS	- Dalvik Debug Monitor de Server
DEXTRO	- Posição do olhar direcionado para direita
DH	- Desvio horizontal
DV	- Desvio vertical
DHD	- Desvio horizontal direito (lado do olho)
DHE	- Desvio horizontal esquerdo (lado do olho)
DVD	- Desvio vertical direito (lado do olho)
DVE	- Desvio vertical esquerdo (lado do olho)
ET	- Esotropia
FN	- Falso Negativo
FP	- Falso Positivo
GUI	- Graphical User Interface
HT	- Hipertropia
HoT	- Hipotropia
IDE	- Integrated Development Environment
IPC	- Inter-process Communication
INFRA	- Posição do olhar direcionado para baixo
JDK	- Java Development Kit
JRE	- Java Runtime Environment
JVM	- Java Virtual Machine
LEVO	- Posição do olhar direcionado para esquerda
MVS	- Máquina de Vetores de Suporte
NDK	- Native Development Kit
OD	- Olho direito
OE	- Olho esquerdo
OOM	– Out-of-Memory Handler
ORL	- Olivetti Research Lab
ORTO	- Olhos alinhados horizontal e/ou vertical
PPO	- Posição Primária do Olhar

QEMU	- Quick emulator
RP	- Reconhecimento de Padrões
SDK	- Software Development Kit
SMS	- Short Message Service
SUPRA	- Posição do olhar direcionado para cima
VC	- Vetor de características
VN	- Verdadeiro Negativo
VP	- Verdadeiro Positivo
TH	- Transformada de Hough
UI	- User Interface
XML	- Extensible Markup Language
XT	- Exotropia

SUMÁRIO

1 INTRODUÇÃO

O estrabismo é uma anomalia dos olhos que afeta cerca de 4% da população provocando problemas estéticos, reversíveis a qualquer idade, e alterações sensoriais irreversíveis, modificando o mecanismo de visão. É uma das alterações oftalmológicas mais comuns na infância. Pode ser definido como uma interação binocular anormal entre os olhos na qual uma mesma imagem não chega a fóvea[1] de ambos os olhos no mesmo instante de tempo, ou seja, os olhos não fixam a mesma imagem (ALMEIDA, 2013).

Os sintomas e as consequências do estrabismo diferem conforme a idade em que aparecem e a maneira como se manifestam. Ao surgir antes dos 6 anos de idade, como consequência do estrabismo, temos que geralmente a pessoa não apresentará diplopia[2], mas sim ambliopia[3]. Já nos casos após os 6 anos de idade a pessoa apresentará diplopia e, em qualquer idade, os estrábicos podem apresentar dor de cabeça, causada pelo esforço para manter os olhos alinhados. Pode apresentar também torcicolos oculares que se caracterizam por a pessoa girar a cabeça para uma determinada posição com a finalidade de usar melhor os dois olhos.

Sistemas de auxílio ao diagnóstico (Computer-Aided Diagnosis – CAD) são ferramentas computacionais desenvolvidas por vários grupos de pesquisa com a finalidade de auxiliar, geralmente especialistas, na tomada de decisão a respeito de um diagnóstico. Os sistemas CAD servem como uma segunda opinião para os especialistas e estão muito relacionados com a área de processamento de imagens digitais, que apresenta um conjunto de técnicas computacionais que visam a extração e seleção de informações de imagens pertinentes a um problema levantado.

O Teste de Hirschberg é um tipo de exame utilizado para diagnosticar o estrabismo, que consiste em incidir um foco de luz nos olhos do paciente, a fim de verificar o alinhamento do reflexo em ambos os olhos. Neste trabalho é apresentado um aplicativo Android desenvolvido para auxílio ao diagnóstico que se faz valer de imagens de pacientes em que foram aplicados o teste de Hirschberg para aferição do diagnóstico do estrabismo.

[1] Fóvea ou mácula é a região de concentração de células cônicas (onde a visão é mais nítida) (JUNQUEIRA; CARNEIRO, 1999)

[2] Diplopia consiste na percepção do mesmo objeto em duas localizações espaciais diferentes (na retina). Na infância, pode haver eliminação da imagem captada pelo olho desviado

[3] Ambliopia caracteriza-se por uma diminuição da acuidade visual de um olho em relação ao outro.

1.1 Justificativa

Na determinação do estrabismo atuam forças ativas (tônus e contração voluntária do próprio músculo) e forças passivas (conjuntiva, cápsula de Tenon[4], gordura orbitária, fáscias[5] e tendões musculares, conformação óssea da órbita, etc.). Do equilíbrio entre esses dois grupos de forças é que surge o estrabismo ou paralelismo ocular.

O ideal durante uma cirurgia, considerando-se apenas o ajuste pré-operatório do plano cirúrgico, é que todo paciente fosse operado acordado, sob anestesia tópica; porém isso requer boa cooperação por parte do paciente. Como o estrabismo é uma patologia de predomínio nas crianças, a maioria não está indicado para fazer a cirurgia sob anestesia tópica. Há ainda outras contraindicações relativas, tais como: reoperações, cirurgias, pacientes muito ansiosos ou com retardo mental.

Existem casos em que não se pode usar anestesia tópica, e nesta situação uma das alternativas mais viáveis é usar a anestesia geral. Nesse caso, considera-se que os músculos do paciente – após anestesiado – só estarão sob o efeito das forças passivas, podendo ocorrer três situações. Na primeira, o desvio não sofre alteração. Na segunda situação, o desvio torna-se maior com o paciente em vigília, o que significa que as forças passivas contribuem mais para o desvio do que as forças ativas e, neste caso, o plano cirúrgico tem que ser alterado um pouco, ou seja, opera-se os mesmos músculos já pré-determinados, mas em uma quantidade menor de cirurgia. No terceiro caso, o desvio torna-se menor anestesiado do que era de vigília, sendo assim, a cirurgia precisa ser aumentada.

Para avaliar o desvio durante a cirurgia, o médico não dispõe de quase nenhum recurso, diferentemente do que ocorre no consultório, durante a avaliação pré-operatória. Por isso, a avaliação pré-operatória acaba sendo feita apenas pelo método de Hirschberg, de forma empírica. Sendo assim, precisa-se da avaliação pré-operatório para balizar os ajustes do plano cirúrgico, mas suas medidas são pouco precisas. Consequentemente, têm-se a necessidade de usar uma ferramenta computacional móvel e portátil para melhorar a confiabilidade das medidas pré-operatórias.

Após finalizar a cirurgia de cada músculo previsto para ser operado, o cirurgião volta a se valer do método de Hirschberg, para quantificar quanto foi o efeito daquele músculo que acabou de ser operado no sentido de diminuir o desvio que está sendo tratado, e assim fazer

[4] Cápsula de Tenon envolve a superfície externa da esclera consistindo em uma membrana fibroelástica que serve de apoio para os movimentos oculares.
[5] Fáscias são tecidos conjuntivos que revestem, separam músculos, e transmitem tensões mecânicas geradas pela atividade muscular reduzindo a fricção entre eles.

uma eventual reprogramação no plano cirúrgico dos músculos que ainda faltam ser operados. Este é outro precioso momento do ato cirúrgico, e que mais uma vez é baseado em um método de medição pouco confiável. Aqui também, o uso de uma ferramenta computacional seria de grande valia.

Além dos cenários citados, há uma grande necessidade também em regiões onde geralmente não conta com um serviço de especialistas da área da oftalmologia. Com uma ferramenta computacional para dispositivos móveis, uma pessoa treinada, seguindo o protocolo básico (Seção 3.2) de utilização poderia fazer o diagnóstico e, posteriormente, os resultados, com as respectivas imagens, seriam levados para a análise do especialista, desta forma auxiliando-o na obtenção de um diagnóstico mais rápido e preciso.

1.2 Objetivos

Este trabalho tem por objetivo portabilizar a metodologia computacional proposta por Almeida (2013), até a etapa de diagnóstico de estrabismo, para o ambiente móvel Android e comparar os diagnósticos obtidos em entre estes e os fornecidos pelo especialista.

1.2.1 Objetivos Específicos

Para concretizar este objetivo geral, alguns objetivos específicos devem ser alcançados:

- Estudar os conceitos do estrabismo;
- Estudar o método de Hirschberg utilizado pelos especialistas na detecção do estrabismo;
- Estudar o sistema Android;
- Analisar e estudar as bibliotecas que trabalham com imagens no ambiente Android;
- Estudar os recursos da *Application Programming Interface (API)* do Android, conforme a necessidade para o desenvolvimento do aplicativo, como detecção de gestos, câmera, entre outros;
- Estudar e desenvolver a metodologia proposta por Almeida (2013) para detectar e diagnosticar o estrabismo por meio do teste de Hirschberg;
- Comparar os resultados obtidos pelo aplicativo desenvolvido com os diagnósticos do especialista e os obtidos por Almeida (2013).

1.3 Trabalhos Relacionados

Mesmo que o uso de ferramentas computacionais para auxílio de especialistas na área da oftalmologia ainda sejam recentes, podemos encontrar algumas ferramentas que foram ou estão sendo desenvolvidas para que estes possam tomar decisões confiáveis a respeito de patologias da visão. Esta seção apresenta pesquisa, equipamentos e trabalhos relacionados à detecção e diagnóstico do estrabismo.

Como equipamentos relacionados à detecção e diagnósticos de patologias do olho, podemos citar os rastreadores oculares, utilizados em laboratórios de pesquisa de motilidade[6] para mensuração de desvios e movimentações oculares, e os sinoptóforos eletrônicos[7], para aferição de estrabismo. No entanto, na prática, ambos são difíceis de se aplicar por pessoas não especializadas em motilidade ocular, além de serem pouco acessíveis.

Helventon et al. (2001) propuseram um método utilizando telemedicina para diagnóstico e plano de tratamento de estrabismo em locais sem especialistas do mesmo. Neste, as imagens são capturadas através de câmeras fotográficas e enviadas por e-mail para estrabólogos[8], que fazem análise e retornam resposta para os oftalmologistas.

Medeiros (2008) desenvolveu um *software* para computadores com o objetivo de diagnosticar de forma semiautomática possíveis casos de estrabismo através de imagens digitais capturadas por uma câmera fotográfica. Ele combinou o método de Canny para realce de bordas e a Transformada de Hough para detecção do limbo e do brilho. Nele, o usuário informa a região dos olhos através de uma interface, podendo também informar os parâmetros do Canny e do Hough. Porém, ainda é necessário fazer testes com vários pacientes estrábicos e obter medidas mais precisas de correlação entre *pixel* e milímetros. Já no trabalho aqui proposto, a localização da região dos olhos é feita de forma automática.

Chandna *et al.* (2009) utilizou redes neurais do tipo *Backpropagation* para realização de diagnóstico diferencial[9] de estrabismos verticais (desvios para cima e para baixo). Utilizou o Cover Test[10] juntamente com prismas para medir o desvio, para este, posteriormente, ser fornecido manualmente pelo especialista. O *software* Master (SINGH, 2012), foi desenvolvido para sugerir o diagnóstico e o tratamento cirúrgico do estrabismo, sendo

[6] Neste caso, pesquisa de motilidade refere-se ao exame de motilidade ocular, no qual realiza-se a avaliação do alinhamento dos olhos nas diversas posições do olhar.
[7] O sinaptóforo é um equipamento destinado para investigação diagnóstica e tratamento ortóptico de estrabismos.
[8] Estrabólogo é o médico oftalmologista especializado em estrabismo.
[9] Diagnóstico diferencial é o método para distinguir dois distúrbios de aparência semelhante.
[10] Cover Test ou Teste de Oclusão é realizado com a oclusão de um dos olhos: se ele tem tendência a se desviar, entra em desvio, atrás da cobertura. Quando o descobrimos, ou ele aparece desviado e assim fica, ou se move para a posição correta, recuperando o alinhamento.

necessário, também, que as medidas do desvio sejam informadas manualmente. Já no trabalho aqui proposto, a magnitude do desvio é determinada automaticamente por meio da aplicação de técnicas de processamento de imagens e reconhecimento de padrões na foto do paciente.

Lorenzi (2014) desenvolveu um aplicativo para dispositivos móveis baseados em Android para detecção de estrabismo, utilizando a câmera deste para a aquisição das imagens. Em sua metodologia, foi utilizado Classificadores em Cascata de Haar para localização da face e dos olhos. Para extração da região dos olhos, foi recortado 25% das regiões inferior e superior da imagem. Para as etapas de localização do limbo e do brilho, foram utilizados o método de Canny para realce de bordas e a transformada de Hough para detecção de círculos. Foram utilizadas 52 imagens de 21 pessoas, sendo que destas, apenas 3 possuíam estrabismo. Obteve-se 40,38% de taxa de acerto na realização do exame, sendo que, como foram capturadas mais de uma amostra por pessoa, esse percentual de 40,48% de sucesso cobriu 85,71% ou 18 pacientes. No entanto, no diagnóstico não é informado o tipo de estrabismo que o paciente possui e nem a medida em dioptrias prismáticas (Δ), medida utilizada pelos estrabólogos para mensurar o desvio. Já no trabalho aqui proposto, ao final do diagnóstico, tanto se informa o tipo do desvio quanto o valor dos desvios vertical e horizontal, em dioptrias, para ambos os olhos.

Almeida (2013) propôs uma metodologia separada em 7 etapas para detecção, diagnóstico e plano cirúrgico do estrabismo: (1) segmentação da face; (2) detecção da região dos olhos; (3) localização dos olhos; (4) detecção do limbo e do brilho; (5) detecção; (6) diagnóstico e (7) plano cirúrgico do estrabismo. Ele utilizou 200 imagens de pacientes, com e sem, estrabismo e lentes corretivas, e em 5 posições do olhar: PPO, SUPRA, INFRA, LEVO e DEXTRO. Restringindo-se à sexta etapa e à posição primária do olhar (PPO), posição de interesse para este trabalho, Almeida (2013) obteve 96,97% de taxa de acerto na detecção de estrabismo, em 33 imagens de pacientes. Para o diagnóstico, obteve as taxas de acerto de 92,85% em ET (esotropia), 100% em XT (exotropia), 66,67% em hipertropia e hiotropia e 6,66% para os que não apresentavam desvios (ORTO).

Por Almeida (2013) obter os melhores resultados no que diz respeito a detecção e diagnóstico automáticos de estrabismo, o trabalho aqui proposto se baseará no mesmo, tanto em metodologia, quanto nos valores para conversão de medidas (*pixel*/Δ) em que o mesmo obteve os melhores resultados.

1.4 Organização do Trabalho

Além do capítulo inicial, este trabalho contém outros quatro capítulos que visam contemplar todos os objetivos e os resultados do desenvolvimento deste aplicativo.

No **Capítulo 2**, apresenta-se a fundamentação teórica necessária para a compreensão do presente trabalho. São descritos conceitos relacionados e aplicados ao estudo do estrabismo como a anatomia dos olhos e o método de Hirschberg. São expostas, também, as técnicas de processamento de imagens (filtragem homomórfica, equalização do histograma, método de Canny e a transformada de Hough), as funções geoestatísticas (semivariograma, semimadograma, correlograma e covariograma) além da máquina de vetor de suporte (SVM) para classificação.

No **Capítulo 3**, descrevem-se o protocolo de aquisição das imagens, a base de imagens utilizadas e as oito etapas (aquisição de imagens, segmentação da face, detecção de região dos olhos, localização dos olhos, localização do limbo, localização do brilho; detecção e diagnóstico do estrabismo) que compõem as etapas de execução do aplicativo, necessárias para diagnosticar o estrabismo em imagens digitais por meio do teste de Hirschberg.

No **Capítulo 4** apresentam-se os resultados e as discussões obtidos na detecção e diagnóstico do estrabismo utilizando o aplicativo desenvolvido.

No **Capítulo 5** apresentam-se as conclusões do estudo, mostrando a viabilidade do aplicativo e apresentando as contribuições e sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

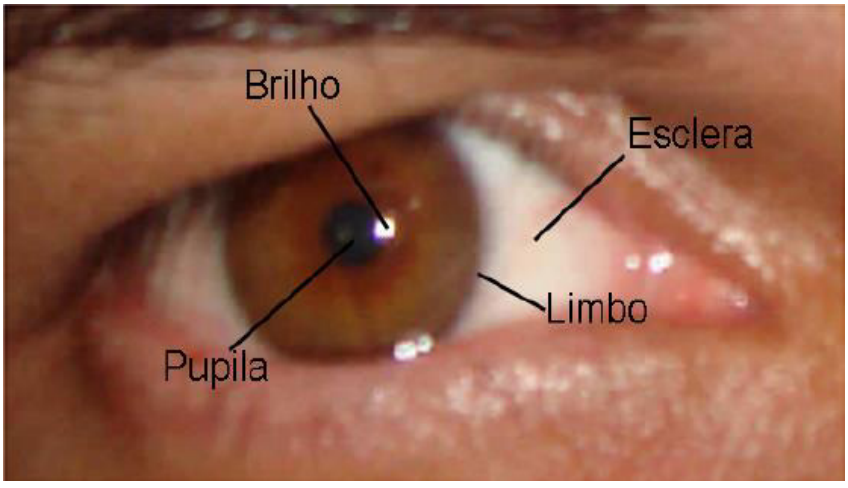
Este capítulo aborda os conceitos necessários para entender o funcionamento das etapas executadas para que o aplicativo possa alcançar o seu objetivo. São abordados tópicos da medicina oftalmológica (anatomia ocular, estrabismo, método de Hirschberg), do sistema Android e das principais técnicas computacionais presentes na metodologia utilizada no aplicativo.

2.1 O Olho

Os olhos são órgãos fotossensíveis complexos que atingem alto grau de evolução, permitindo uma análise minuciosa quanto à forma dos objetos, cor e a intensidade de luz refletida. Cada olho fica dentro de uma caixa óssea protetora - a órbita – e apresenta basicamente uma câmara escura, uma camada de células receptoras sensoriais, um sistema de lentes para focalizar a imagem e um sistema de células e nervos para conduzir o estímulo ao córtex cerebral (JUNQUEIRA; CARNEIRO, 1999).

A Figura 1 mostra as principais partes da estrutura externa do olho humano utilizadas nesse trabalho. A esclera, região opaca e esbranquiçada, é uma membrana rija e inelástica que mantém o tamanho e a forma do bulbo. Sua superfície externa é branca e absolutamente lisa, exceto nos pontos em que se inserem os retos e oblíquos.

Figura 1- Estrutura externa do olho humano.



Fonte: ALMEIDA, 2013

O limbo é a região de contorno entre a esclera e a próxima região mais ao centro, a íris, que se define por um diafragma circular pigmentado. A pupila, abertura circular escura, é responsável pela transmissão da luz. O brilho, elemento que não faz parte da estrutura do

olho, é o reflexo luminoso gerado pelo *flash* da câmera do dispositivo móvel na aquisição da imagem, dentro da região limbar representado na Figura 1 pelo círculo branco.

2.2 O Estrabismo

O estrabismo é a diferença entre os alinhamentos esperados dos olhos. Enquanto um dos olhos fixa para um ponto central, o outro fixa para esquerda ou para direita, para cima ou para baixo. A Figura 2 ilustra a presença de estrabismo no olho esquerdo de uma pessoa, onde este aponta para o lado esquerdo, enquanto o outro aponto para um ponto fixo central.

Figura 2- Exemplo de estrabismo

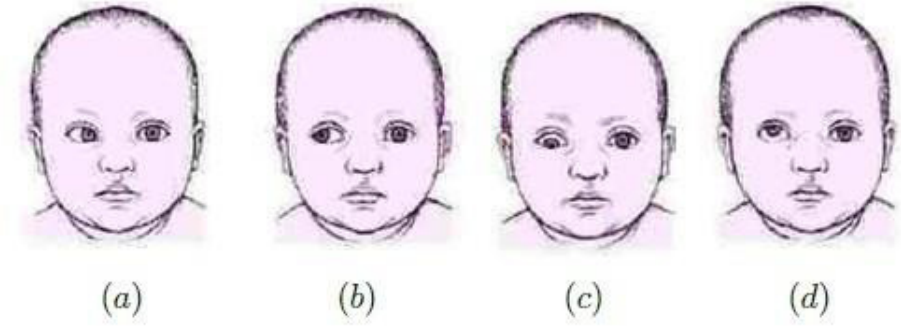
Fonte: WRIGHT *et al*., 2007.

A causa do estrabismo é o desequilíbrio dos músculos oculares, que são responsáveis pela iniciação, coordenação e conclusão dos movimentos oculares. Em condições normais, estes músculos trabalham de forma coordenada permitindo uma visão binocular única, uma vez que o cérebro funde as imagens dos dois olhos e as interpreta como uma só. Sabe-se, ainda, que pode estar associado a distúrbios neurológicos causados por patologias ou acidentes que alteram o funcionamento destes músculos ou até a falta do uso de lentes corretivas no momento adequado por crianças que iniciam os estudos fazendo uso da visão de forma mais acentuada.

O estrabismo pode ser classificado em horizontais: convergente (esotropia) e divergente (exotropia); e verticais: hipertropia e hipotropia. É convergente, quando um olho fixa a imagem e o outro vira para dentro, e divergente, quando se desloca para fora. Na hipertropia um dos olhos pode girar para cima e na hipotropia o olho gira para baixo. Em ambos os casos, cada um dos olhos focaliza imagens diferentes e a pessoa tem o que chamamos de diplopia, ou visão dupla (ALMEIDA, 2013). Nas Figuras 3a, 3b, 3c e 3d temos exemplos dos tipos de estrabismo nas quais o olho afetado é o direito.

Existem inúmeras técnicas que podem ser aplicadas na correção do estrabismo, estabelecendo o equilíbrio dos músculos oculares e resolvendo o problema da ambliopia. O tratamento do estrabismo possui por metas preservar a visão, alinhar os olhos e restaurar a visão binocular e é caracterizado por prescrição de óculos, realização de exercícios ortópticos e obstrução do olho fixador alternando com o outro olho. Caso o tratamento não seja o suficiente, aplica-se o procedimento cirúrgico realizando retrocesso ou ressecção dos músculos oculares debilitados. Vale ressaltar que a melhor fase para o tratamento é até os sete primeiros anos de vida, período em que fundamentalmente a visão se desenvolve. Após esta fase, torna-se mais difícil atingir o sucesso esperado no tratamento do estrabismo.

Figura 3- Tipos de estrabismo. (a) Esotropia (ET), (b) Exotropia (XT), (c) Hipotropia (HoT), e (d) Hipertropia (H)



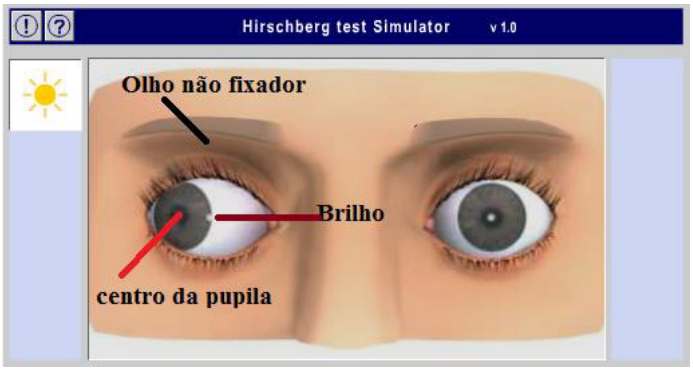
Fonte: MERCK, 2009

2.2.1 O método de Hirschberg

O método de Hirschbeg foi desenvolvido em 1886 pelo oftalmologista alemão Julius Hirschberg. Este método consiste em avaliar o ângulo de desvio de um estrabismo, com o uso de foco luminoso, calculando a magnitude aproximada desse desvio de acordo com o deslocamento do reflexo luminoso, no olho não fixador, em relação ao centro do seu globo ocular. Quando se vai avaliar uma pessoa utilizando este método considera-se que um dos olhos é o olho fixador, com o brilho ou primeira imagem de Purkinje, alinhado com o centro óptico deste. E o outro olho, não fixador, é onde o desvio será observado. Conquanto, existem fatores que interferem no posicionamento do reflexo luminoso do olho não fixador em relação à posição que ele – o reflexo – se encontra no olho fixador, como a curvatura corneana, tamanho da córnea e do próprio olho e sua refração, e ângulo Kappa, que é o ângulo formado entre a linha visual e o eixo da pupila.

Na Figura 4, temos uma imagem que ilustra a simulação do Teste de Hirschberg em um paciente que apresenta exotropia de aproximadamente 90Δ no olho direito onde a imagem não está sendo formada no centro óptico.

Figura 4– Simulação do Método de Hirschberg para avaliação da magnitude do desvio através da localização do centro do olho não fixador e o brilho gerado no mesmo



Fonte: adaptado de TECHS, 2009

2.3 Métodos Computacionais de Auxílio e Diagnóstico

Sistemas de auxílio ao diagnóstico (computer-aided diagnosis – CAD) são ferramentas computacionais, acoplados a equipamentos médicos ou não, que tem a finalidade de auxiliar na tomada de decisão a respeito de um diagnóstico (BREITMAN; ANIDO, 2006). Aplicações com a finalidades diversas têm sido desenvolvidas por vários grupos de pesquisas, visando contribuir para a detecção e diagnóstico precoce de doenças, como uma segunda opinião para os especialistas.

Em geral, os sistemas CAD fornecem opiniões a partir de informações extraídas de imagens médicas, que podem ser provenientes de diversos tipos de modalidades como: radiografia, ultrassonografia e ressonância magnética nuclear, entre outras. Técnicas de processamento de imagens, inteligência artificial, reconhecimento de padrões, entre outras especificidades computacionais, são aplicadas com o objetivo de melhorar tais imagens e extrair delas informações úteis ao diagnóstico (BREITMAN; ANIDO, 2006).

Os sistemas CAD geralmente utilizam-se de técnicas provenientes de duas áreas do conhecimento: visão computacional, que envolve o processamento de imagem para realce, segmentação e extração de atributos; e inteligência artificial, que inclui métodos para seleção de atributos e reconhecimento de padrões (KENNETH, 1996). Os conceitos utilizados para criar uma metodologia CAD serão abordados a seguir, iniciando-se com os principais conceitos em processamento de imagens digitais.

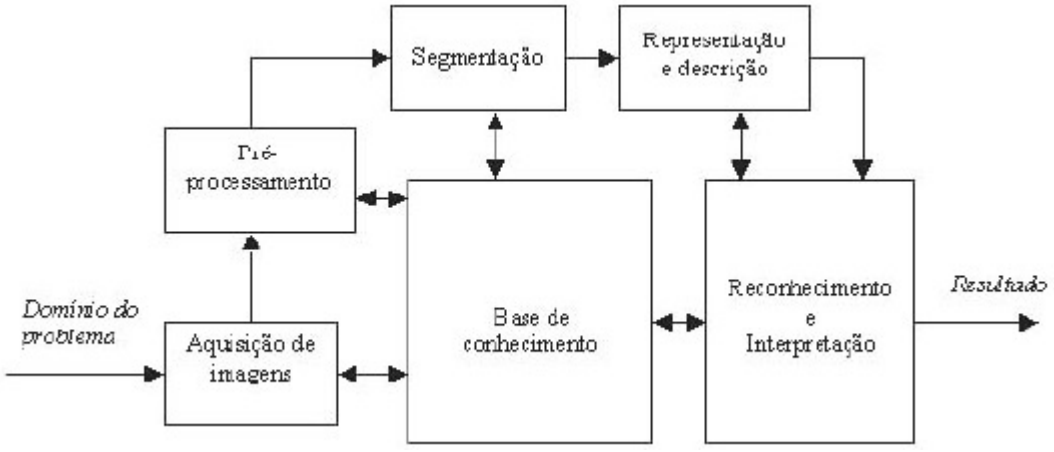
2.4 Processamento de Imagens Digitais

Uma imagem digital pode ser definida como uma função bidimensional f(x,y), onde x e y são coordenadas espaciais e a amplitude de f relacionada a cada par (x,y) é a intensidade da imagem naquele ponto, sendo x,y e f valores discretos e finitos. A representação de uma imagem digital é uma matriz, onde cada um dos pontos ou elementos são chamados de *elementos pictóricos*, *elementos de imagem*, *pels* ou, ainda, *pixels*.

A formação de uma imagem ocorre quando um sensor registra a radiação que interagiu com objetos físicos (BALLARD; BROWN, 1982). Esse registro é a representação desse objeto, tal qual pode ser processado e manipulado. Em imagens médicas, geralmente se utiliza a representação mais clássica de imagens digitais, a imagem em nível de cinza, representando uma função de intensidade luminosa refletida no objeto. Essa intensidade é representada por valores no intervalo de zero, representando a cor preta ou sem intensidade, a um valor máximo M representando a cor mais clara ou de máxima intensidade, a cor branca.

O objetivo de definir matematicamente a imagem é a possibilidade de manipular o seu conteúdo a fim de transformar ou retirar dela informações importantes. Ao vasto conjunto de operações que podemos aplicar em uma matriz que representa uma imagem denominamos de processamento de imagens (BREITMAN; ANIDO, 2006). Este processamento ocorre em etapas, podendo ter por resultado uma nova imagem ou não. Na Figura 5 temos a representação dessas etapas.

Figura 5- Etapas do Processamento de Imagens Digitais.



Fonte: GONZALEZ; WOODS, 2002

A necessidade de processar imagens surge de um problema a ser resolvido com a aplicação do processamento de imagens. Do domínio do problema, se faz a aquisição de imagens, correspondendo a primeira etapa. A definição de Ballard e Brown (1982) citada

explica exatamente o que acontece nessa etapa. No aplicativo desenvolvido neste trabalho a imagens podem ser adquiridas de duas formas: (1) utilizando a câmera do próprio dispositivo móvel como sensor para aquisição das imagens dos pacientes ou (2) buscando na galeria de imagens do dispositivo. Esta última permite que a imagem seja adquirida por outros dispositivos de aquisição de imagens digitais.

A etapa de pré-processamento objetiva melhorar a imagem para aumentar a probabilidade de sucesso na detecção do objeto de interesse nela, através de técnicas que realçam contrastes ou diminuem efeitos espúrios na aquisição da imagem (ruídos, por exemplo).

A etapa de segmentação busca extrair uma região de interesse, objetivando separar o objeto de interesse ou diminuir o espaço de busca. Geralmente esta etapa é baseada em forma, textura ou cor. Neste trabalho, por exemplo, uma das segmentações foi baseada na forma circular, para extração do limbo.

Mais adiante, temos a etapa de representação e descrição, ou ainda, extração de características. Esta etapa visa selecionar características ou atributos que irão resultar em alguma informação quantitativa de interesse, ou seja, que servirá de base para diferenciar uma classe de outras classes de objetos. Neste trabalho, correspondendo a esta etapa, fez-se o uso de funções geoestatísticas para constituir um vetor de características que contém a informação do padrão da região dos olhos.

Por fim temos o reconhecimento e interpretação, que consistem em atribuir rótulos a cada objeto e dar significado a cada conjunto deles. Em outras palavras, busca-se, a partir de uma base de conhecimento previamente construída, através de características extraídas na etapa anterior, classificar cada objeto em algum dos grupos rotulados anteriormente.

2.4.1 Filtragem Homomórfica

A filtragem homomórfica tenta analisar separadamente as informações de iluminação e reflectância na imagem. Melhorando as propriedades de reflectância na imagem, pode-se melhorar o contraste nas áreas com baixo e altos níveis de iluminação (BURGISS; GOODRIDGE, 2000).

Uma imagem é o registro da luz refletida por um objeto que sofreu incidência da mesma, a partir de alguma fonte de iluminação. Portando, um modelo da imagem é:

$$F(x, y) = i(x, y)r(x, y) \quad (1)$$

onde i(x,y) representa a iluminação, principal contribuinte para o limite de intensidade, e r(x,y) representa a reflectância, principal contribuinte do contraste local e detalhes presentes em áreas com pouca iluminação. Sabendo-se disto, o objetivo do filtro é reduzir a componente de iluminação e aumentar a reflectância, objetivando aumentar o contraste local e melhorar os detalhes onde há pouca iluminação.

Neste trabalho, foi usado o filtro homomórfico no domínio espacial da imagem digital baseado no filtro proposto por Melo *et al.* (2005). Este recebe como entrada uma função (x,y) que representa uma imagem em nível de cinza. O valor de cada *pixel* é convertido para logaritmo natural *log*(1+f(x,y)), resultando na separação das componentes de iluminação e reflectância. Em seguida, aplica-se os filtros passa-alta e passa-baixa, obtendo-se *log*(i(x,y)) e *log*(r(x,y)) respectivamente.

Cada imagem resultante é multiplicadas por valores $\alpha < 1$ e $\beta > 1$, respectivamente, para diminuir os limites de intensidade da iluminação e aumentar o contraste local da reflectância. Depois, os resultados são somados e normalizados entre 0 e 1. Os últimos passos são calcular o exponencial da imagem normalizada, normalizar os valores entre 0 e 255 e aplicar equalização do histograma.

Neste trabalho, o filtro homomórfico foi utilizado para diminuir a influência da iluminação intensa pontual do brilho e aumentar o contraste local da região dos olhos, dando mais equilíbrio às diferenças de intensidade, conforme ilustra a Figura 6.

Figura 6- Exemplo de utilização do filtro homomórfico. (a) Imagem original. (b) Imagem após a filtragem homomórfica.

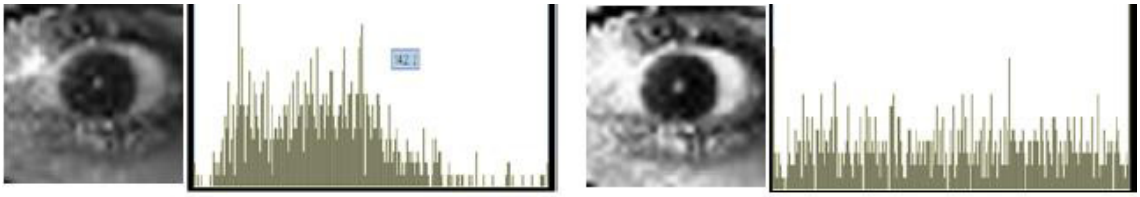
(a) (b)

2.4.2 Equalização de Histograma

Um histograma, em processamento de imagens, é a distribuição de frequência das intensidades de cor, ou seja, a informação da quantidade de vezes que cada intensidade de cor se repete. Matematicamente, o histograma pode ser expresso por H(rk) = nk, onde rk é o k-enésimo nível de intensidade de cor, em um intervalo [0, L-1] (L sendo o número de níveis de intensidade), e nk é o número de *pixels* contendo nível de intensidade de cor igual a rk na imagem.

A equalização de histograma, em processamento de imagens, é uma técnica que consiste em fazer uma distribuição uniforme do histograma da imagem, com a finalidade de realçar contrastes. Na Figura 7 é demonstrado a aplicação da equalização de histograma em uma imagem em nível de cinza, mostrando também seus respectivos gráficos de distribuição de frequências.

Figura 7- Exemplo de utilização de Equalização de Histograma (a) imagem original, (b) imagem após a equalização de histograma



(a) (b)

Fonte: ALMEIDA, 2013

2.4.3 Método de Canny

O propósito da detecção de bordas em uma imagem é reduzir a quantidade de informação em uma imagem, enquanto preserva as propriedades estruturais desta ou, ainda, realçar as bordas presentes na imagem.

Um dos métodos mais conhecidos para detecção de bordas é o método de Canny. Este método foi desenvolvido por John F. Canny em 1986 (CANNY, 1986), tendo por objetivos principais boa detecção (identificar todas as bordas possíveis), boa localização (bordas encontradas devem ser mais próximas possíveis das reais e resposta mínima (retorno de um único ponto de borda). Este método possui 4 etapas básicas, as quais serão brevemente discorridas mais adiante.

2.4.3.1 Suavização da Imagem

Primeiramente, a imagem é suavizada através de uma função gaussiana bidimensional com a finalidade de reduzir ruídos, pois estes podem ser detectados como bordas. Computacionalmente, é mais viável utilizar funções gaussianas unidimensionais, no eixo x e y, para se economizar tempo de processamento.

$$g(x,y) = \frac{1}{2\pi\sigma^2}\, e^{\frac{-(x^2+y^2)}{2\sigma^2}} \qquad (2)$$

onde g(x,y) representa o valor do pixel e x e y representam as posições dos pixels na imagem.

2.4.3.2 Cálculo do Gradiente

São consideradas bordas locais onde a intensidade de nível de cinza sofre maior variação. Variações podem ser descritas através de derivadas. Como uma imagem depende de duas coordenadas espaciais as bordas podem ser expressas através de derivadas parciais ou cálculo do gradiente.

Considerando a etapa anterior, a imagem uniformizada é separada, em duas outras, nas direções x e y utilizando as equações, respectivamente:

$$dx(x,y) = \frac{-x}{2\pi\sigma^4}\, e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{3}$$

$$dy(x,y) = \frac{-y}{2\pi\sigma^4}\, e^{\frac{-(x^2+y^2)}{2\sigma^2}} \tag{4}$$

onde d(x,y) e dy(x,y) representa o valor do pixel em cada imagem e x e y representam as posições dos pixels.

A magnitude do gradiente ou "força das bordas" é obtida através do cálculo das distâncias entre os valores das direções x e y, com a equação:

$$mag(x,y) = \sqrt{dx^2 + dy^2} \tag{5}$$

onde mag(x,y) representa o valor do pixel na imagem resultado e x e y representam as posições dos pixels nas imagens.

Tendo-se o gradiente nas direções x e y, encontra-se a direção da borda por meio da equação:

$$dir(x,y) = arctg\left(\frac{dy(x,y)}{dx(x,y)}\right) \tag{6}$$

onde dir(x,y) é a matriz que guarda os valores da direção de cada borda na posição (x,y) e x e y representam as posições dos pixels na imagem.

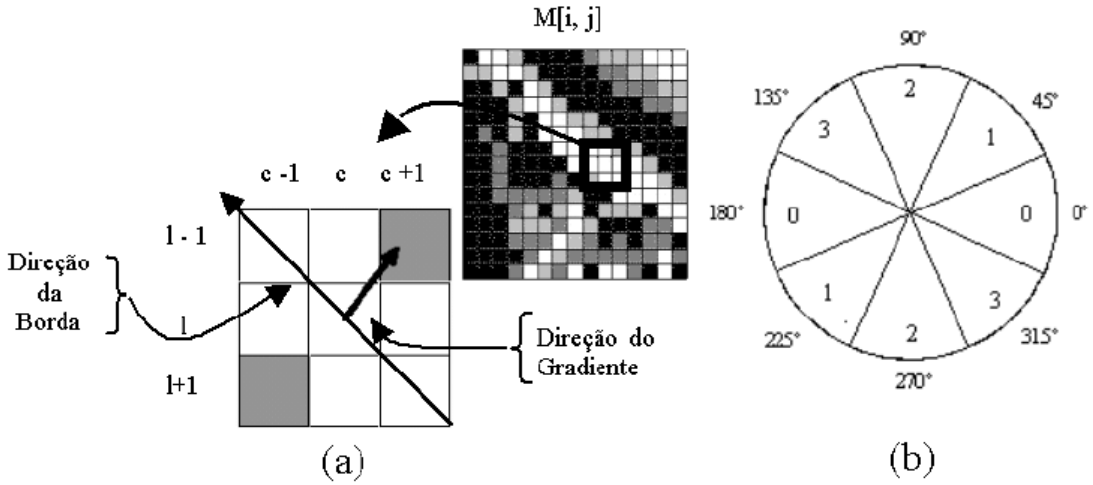
2.4.3.3 Supressão Não-Máxima

Consiste em eliminar os pixels não máximos locais, na direção do gradiente da imagem, visando minimizar a distância entre a borda encontrada e a borda real e, também, reduzir a espessura da borda. Sendo assim, simplificadamente, o algoritmo de supressão numa janela 3x3 pode ser definido por:

1. Definição da direção do gradiente do pixel central;
2. Comparação do valor desse pixel com o valor de seus vizinhos nas direções positiva e negativa do gradiente;
3. Caso o valor desse pixel não seja o maior dos 3, suprima esse valor;

A Figura 8 demonstra como ocorre a supressão.

Figura 8– Supressão não máxima. (a) Esquema de supressão não máxima para Ø= 45º. (b) Setores considerados para a supressão não máxima



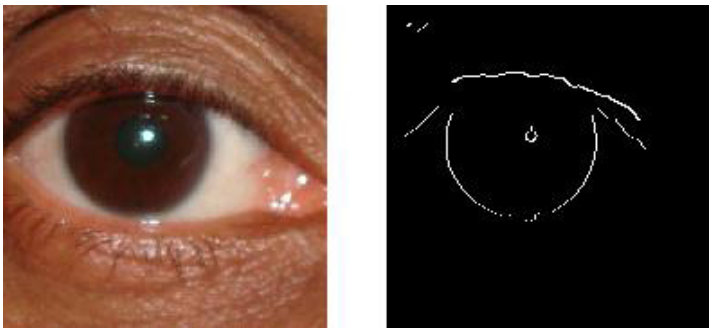
Fonte: VALE e DAL POZ, 2002

2.4.3.4 Limiariação por histerise

A limiarização por histerese é um método de limiarização proposto por Canny (1986) que consiste em utilizar dois limiares, um inferior e outro superior. Os pixels com intensidade abaixo do limiar inferior são considerados não bordas, os com intensidade acima do superior são considerados bordas e os intermediários são analisados de acordo com a vizinhança. Nessa análise se, seguindo uma cadeia de pixels (sua vizinhança e seus vizinhos em 3x3 ou 5x5) for descoberto que essa cadeia está conectada a um "pixel borda", ela é considerada borda. Senão, ela é considerada não borda.

A Figura 9 demonstra o resultado final do método de Canny.

Figura 9– Aplicação do Método de Canny numa imagem de olho. (a) imagem original. (b) imagem pós detecção de bordas



(a) (b)

2.4.4 Transformada de Hough

A Transformada de Hough (TH) é um dos métodos matemáticos mais utilizados para detecção de objetos que podem ser parametrizados (linhas, círculos, elipses, entre outros). Ela foi proposta por Paul Hough em 1962, patenteada pela IBM e reformulada computacionalmente por Duda e Hart (1972).

O princípio da TH é mapear cada ponto da imagem em um plano de parâmetros (espaço de Hough). A ideia é aplicar na imagem uma transformação tal que todos os pontos pertencentes a uma mesma curva sejam mapeados num único ponto de um novo espaço de parametrização da curva procurada, onde geralmente se utiliza uma matriz de inteiros para representar tal espaço.

Neste trabalho, a TH foi aplicada para detectar o limbo dos olhos e o brilho gerado neles, que correspondem às regiões circulares. Para encontrar tais regiões, simplificando o processamento computacional, foram utilizadas as equações paramétricas do círculo:

$$x = a + rcos(\theta) \tag{7}$$

$$y = b + rsin(\theta) \tag{8}$$

A aplicação da TH inicia-se com a detecção de bordas. Depois, aplica-se o mapeamento dos *pixels* para o espaço parametrizado, representando-os em um vetor de acumulação. E, por fim, localiza-se os pontos de máximo do vetor e converte-se os índices dos cumes para as coordenadas x, y.

2.5 Análise de Textura

Embora não exista uma definição universalmente aceita para conceituar textura, pode-se defini-la como sendo um atributo que representa o arranjo espacial dos níveis de cinza dos pixels de uma região (GERACI, 1990), ou ainda como uma medida de propriedades tal qual como suavidade, asperidade e regularidade (GONZALEZ; WOODS, 2002).

Este trabalho utiliza a análise de texturas por meio de funções geoestatísticas propostas também por Almeid*a* (2013), para reconhecimento do padrão identificado por regiões de olhos e não-olhos.

2.5.1 Funções Geoestatísticas para Extração de Textura

Os conceitos básicos de geoestatística surgiram quando Krige (1951), em seus trabalhos com dados de mineração da África do Sul, concluiu que somente a informação da

variância dos dados é insuficiente para explicar um fenômeno em estudo, tendo que se levar em consideração também a distância entre as informações.

Os fundamentos teóricos da geoestatística podem ser encontrados nos trabalhos desenvolvidos por Matheron (1963, 1971), os quais foram baseados nas observações de Krige. Através destes trabalhos, surgiu a Teoria das Variáveis Regionalizadas. Matheron (1963) definiu Variável Regionalizada como uma função espacial numérica, que varia de um local para outro, com uma continuidade aparente e cuja variação não pode ser representada por uma função matemática simples, mas sim por um variograma. Segundo essa teoria, a diferença nos valores de uma dada variável tomada em dois pontos no campo depende da distância entre eles. Com isso, a diferença entre os valores do atributo tomados em dois pontos mais próximos no espaço deve ser menor do que a diferença entre valores tomados em dois pontos mais distantes. Logo, cada valor carrega consigo uma forte interferência dos valores de sua vizinhança, ilustrando uma continuidade espacial (ISAAKS; SRIVASTAVA, 1989).

Podemos aplicar os conceitos de geoestatística no processamento de imagens digitais ao considerarmos que as texturas podem ser descritas como variáveis regionalizadas, em termos dos dois componentes principais associados aos seus pixels (ou outra unidade): variabilidade e autocorrelação. Com isso, obtemos medidas através destes componentes simultaneamente e, posteriormente, descrevemos a textura obtida de uma determinada imagem ou região dela, através do grau de associação espacial presente dentro dos elementos geograficamente referenciados da mesma.

Neste contexto, foram utilizadas quatro funções geoestatísticas – semivariograma, covariograma, semimadograma e correlograma – na extração de características para descrever o padrão da região dos olhos. Tais medidas serão brevemente abordadas nas seções a seguir.

2.5.2 Semivariograma

O gráfico da semivariância como uma função da distância de um ponto é denominado semivariograma, sendo que a distância entre as amostras é diretamente proporcional à semivariância (SILVA et al., 2004).

O semivariograma é definido por:

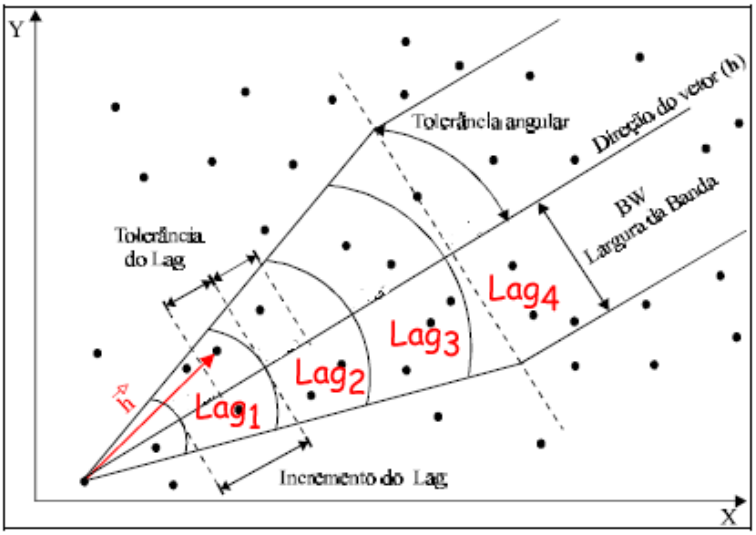
$$\gamma(h) = \frac{1}{2N(h)} \sum_{i=1}^{N(h)} (x_i - y_i)^2 \tag{9}$$

onde h é o vetor distância (*lag distance*) entre os valores de origens, x_i, e os valores de extremidades, y_i, *e N(h)* é o número de pares de distância h.

Os outros parâmetros usados para calcular o semivariograma, como incremento do lag (*lag spacing*), tolerância do lag (*lag tolerance*), direção (*direction*), tolerância angular (*angular tolerance*) e largura máxima de banda (*maximum bandwidth*) são ilustrados na Figura 10.

Tomando-se como referência o Lag2, um incremento de lag igual a 1 pixel com tolerância de 0,5. Considera-se ainda a direção de medida de 45° com tolerância angular 22,5º. Então, qualquer par de observações cuja distância esteja compreendida entre 1,5 e 2,5 pixels e 22,5º e 67,5º será incluído no cálculo do semivariograma de Lag2. Este processo é repetido para todos os lags.

Figura 10- Parâmetros utilizados para cálculo das funções geoestatísticas.



Fonte: CAMARGO, 1997.

2.5.3 Semimadograma

Segundo Silva *et al.* (2004) o semimadograma é a média da diferença absoluta medida nos pares da amostra, como uma função de distância e direção. A função é definida por:

$$m(h) = \frac{1}{2N(h)} \sum_{i=1}^{N(h)} |x_i - y_i| \qquad (10)$$

onde h é o vetor distância (*lag distance*) entre os valores de origens, x_i, e os valores de extremidades, y_i, e N(h) é o número de pares na distância h.

2.5.4 Covariograma

A função covariograma é a relação da covariância de um diagrama de dispersão e uma distância *h*. Esta função apresenta resultados altos para distâncias pequenas, sendo que os mesmos tendem a decrescer a medida que a distância aumenta, ou seja, a função tende a ser alta quando h = 0, e tende para zero para pontos que são separados por distâncias grandes ou iguais ao limite (SOUZA JUNIOR 2006). O covariograma é definido por:

$$C(h) = \frac{1}{N(h)} \sum_{i=1}^{N(h)} x_i y_i - m_{-k} m_{+k} \tag{11}$$

onde m_{-k} é a média dos valores das origens dos vetores,

$$m_{-h} = \frac{1}{N(h)} \sum_{i=1}^{N(h)} x_i \tag{12}$$

e m_{-h}é a média dos valores das extremidades dos vetores,

$$m_{=h} = \frac{1}{N(h)} \sum_{i=1}^{N(h)} y_i \tag{13}$$

2.5.5 Correlograma

Podendo ser chamada também de função de correlação, é a função normalizada da função covariograma tendo seus coeficientes nas faixas de valores de -1 a 1. Espera-se que, quanto mais próximo de 1, mais as variáveis estejam próximas umas das outras e distantes caso contrário.

A correlação é definida por:

$$\rho(h) = \frac{C(h)}{\sigma_{-h} \sigma_{+h}} \tag{14}$$

onde σ_{-h} é o desvio padrão dos valores das origens dos vetores,

$$\sigma_{-h} = \left[\frac{1}{N(h)} \sum_{i=1}^{N(h)} x^2 - m_{-h}^2 \right]^{\frac{1}{2}} \tag{15}$$

e σ_{-h} é o desvio padrão dos valores das extremidades dos vetores,

$$\sigma_{+h} = \left[\frac{1}{N(h)} \sum_{i=1}^{N(h)} y^2 - m_{+h}^2 \right]^{\frac{1}{2}} \tag{16}$$

2.6 Reconhecimento de Padrões

Entende-se por padrão as propriedades que possibilitam o agrupamento de objetos semelhantes dentro de uma determinada classe ou categoria, mediante a interpretação de dados de entrada, que permitam a extração das características relevantes desses objetos, (TOU; GONZALES, 1981). Entende-se por classe ou categoria de um padrão um conjunto de características comuns aos objetos de estudo. Assim, reconhecimento de padrões (RP) pode ser definido como o agrupamento de objetos, a partir de características relevantes extraídas destes, de modo que em cada grupo os objetos tenham alto grau de características semelhantes entre si e que cada grupo tenha alto grau de características discriminatórias entre si.

Um sistema completo de reconhecimento de padrões consiste em três componentes principais: um sensor, que obtém as informações a serem classificadas; um mecanismo de extração de características, afim de discretizá-las e formar um vetor de características e um mecanismo de classificação. Este último pode ser supervisionado e não supervisionado. Na classificação supervisionada, que é utilizada neste trabalho, o sistema é "ensinado" a reconhecer padrões representativos de cada classe já disponível, através de uma base de conhecimento treinada e disponibilizada a priori.

As características selecionadas para formar classes devem ter o máximo de relevância possível, isto é, devem ter alta taxa de discriminação de grupos e não devem ser correlatas entre si, com a intenção de diminuir a dimensionalidade do vetor de características. O objetivo disto é diminuir a carga do classificador, propiciando uma maior agilidade e menor probabilidade de erros.

Após a seleção de características de cada objeto da população, a próxima etapa é atribuir um rótulo a cada vetor de características, formando as amostras. Os objetos são rotulados a partir do conhecimento humano. Na fase de treinamento, o classificador tenta gerar assinaturas que melhor distinguem as classes para cada rótulo pertencente ao conjunto de amostras. Esse processo é importante na fase de reconhecimento, que fará uso da assinatura para identificar se, novas amostras não treinadas, fazem parte de uma população específica (ALMEIDA, 2013).

Neste trabalho, foi utilizada a Máquina de Vetor de Suporte (MVS) para realizar o reconhecimento da região dos olhos.

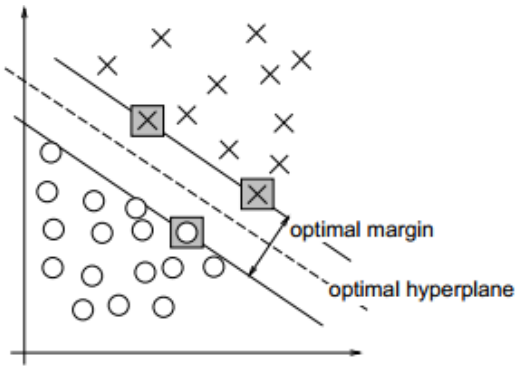
2.6.1 Máquina de Vetores de Suporte

As Máquinas de Vetores de Suporte constituem uma técnica embasada na teoria de aprendizado estatístico (VAPNIK, 1995) e vem recebendo grande atenção nos últimos anos (HEARST et al., 1998; CRISTIANINI; SHAWE-TAYLOR, 2000). Elas têm a capacidade de resolver problemas de classificação e regressão, adquirindo com o aprendizado na etapa de treinamento a capacidade de generalização.

As MVS possuem muitas características atrativas. Uma delas é a boa capacidade de generalização, ou melhor, ele é eficiente na classificação de dados que não pertençam ao conjunto utilizado em seu treinamento. Outras características a serem mencionadas são a robustez com objetos de grandes dimensões, como imagens, e sua base teórica bem estabelecida dentro da Matemática e Estatística.

O funcionamento das MVS é definido da seguinte maneira: dadas duas classes e um conjunto de objetos que pertencem a essas, irá ser determinado um hiperplano que os separa, da melhor forma possível, de forma a colocar o máximo desses objetos em suas respectivas classes. Este tipo de classificação é chamado de classificação binária e tal hiperplano é chamado de hiperplano ótimo, podendo este possuir outros hiperplanos como suporte denominados vetores suporte. A Figura 11 demonstra estes conceitos.

Figura 11– Hiperplano ótimo, objetos classificados e vetores de suporte (linhas com quadrados sobre elas).



Fonte: VAPNIK, 1995.

Porém, esta abordagem é restrita a problemas que são linearmente separáveis. Para problemas não linearmente separáveis, mapeia-se as características para uma dimensão maior utilizando uma função *kernel*. Isto é feito, pois a medida que se aumenta a dimensionalidade de um problema também aumenta-se a probabilidade desse problema se tornar linearmente separável. Abaixo temos as funções básicas de *kernel*:

- Linear: $K(x, y) = x^T y$;
- Polinomial: $K(x, y) = (\gamma x^T y + r)^d, \gamma > 0$;

- Sigmoidal: $K(x, y) = \tanh(\gamma x^T y + r)$;
- Função Básica Radial (RBF): $K(x, y) = e^{-\gamma||x-y||^2}$.

Neste trabalho, é utilizado um algoritmo supervisionado. Logo, assume-se a existência de um conjunto de amostras padrões cujas classes são conhecidas e um classificador já treinado. Utiliza-se também um *kernel* RBF com parâmetro definido pelo usuário.

2.7 Android

O Android é um sistema operacional para dispositivos móveis *open source* baseado em Linux e desenvolvido pelo grupo *Open Handset Alliance*, liderada pelo Google (BRAHLER, 2010). O grupo, liderado pelo Google, inclui operadoras de telefonia móvel, fabricantes de aparelhos portáteis, fabricantes de componentes, provedores de plataformas e soluções de software e empresas de marketing. Foi criado com o objetivo de ser uma plataforma moderna, única, aberta e flexível que oferece um ambiente de desenvolvimento poderoso.

As aplicações no Android são desenvolvidas em linguagem de programação Java, utilizando o Android *Software Development Kit* (SDK), e executam na Máquina Virtual Dalvik. O Android também possui suporte para o desenvolvimento de aplicações nativas, escritas em C e C++, através do Android *Native Development Kit* (NDK) (BRAHLER, 2010).

O Android foi baseado no núcleo 2.6 do Linux. Sendo assim, toda a segurança dele é baseada no Linux. Cada aplicação é executada em um único processo e, para cada uma delas, é criado um usuário no sistema operacional para ter acesso a sua estrutura de diretórios. Dessa forma nenhuma outra aplicação pode ter acesso a esta aplicação. Porém, o Android permite a integração entre aplicações. Essa integração se dá por mensagens enviadas ao sistema, chamadas *intents* ou intenções. Formalmente, as *intents* podem ser definidas como mensagens enviadas por um componente da sua aplicação para o Android, informando a intenção de inicializar outro componente, da mesma aplicação ou de outra. Assim sendo, gera funcionalidades que podem ser reutilizadas sem a necessidade de importar códigos ou dependências de outra aplicação.

Por ser completamente livre e de código aberto, diversos programadores do mundo contribuem para melhorar a plataforma. Além disso, a licença do Android é flexível, permitindo assim que cada fabricante possa realizar alterações no código-fonte para customizar seus produtos, sem precisar compartilhar essas alterações com ninguém. Outra característica deste sistema é o fato dele ser um sistema livre, portanto, pode ser copiado,

usado, modificado e redistribuído de acordo com as necessidades de cada usuário. Por essas razões, desenvolvedores de aplicações, fabricantes de dispositivos móveis e os próprios usuários são beneficiados.

O sistema Android bateu recorde de participação em embarques globais de smartphones para o mercado, com cerca de 81% de acordo com o site de análise de mercado Global Canalys (2014), para o primeiro trimestre de 2014. São 279.4 milhões de smartphones embarcados no total, sendo que os que possuem o sistema Android lideram, equivalendo a cerca de 5 vezes mais do que o segundo lugar, com 16%. Foi feito também uma projeção em que, dos 1.5 bilhões de smartphones que serão enviados para o mercado entre 2013 e 2017, 73% possuirá esse sistema.

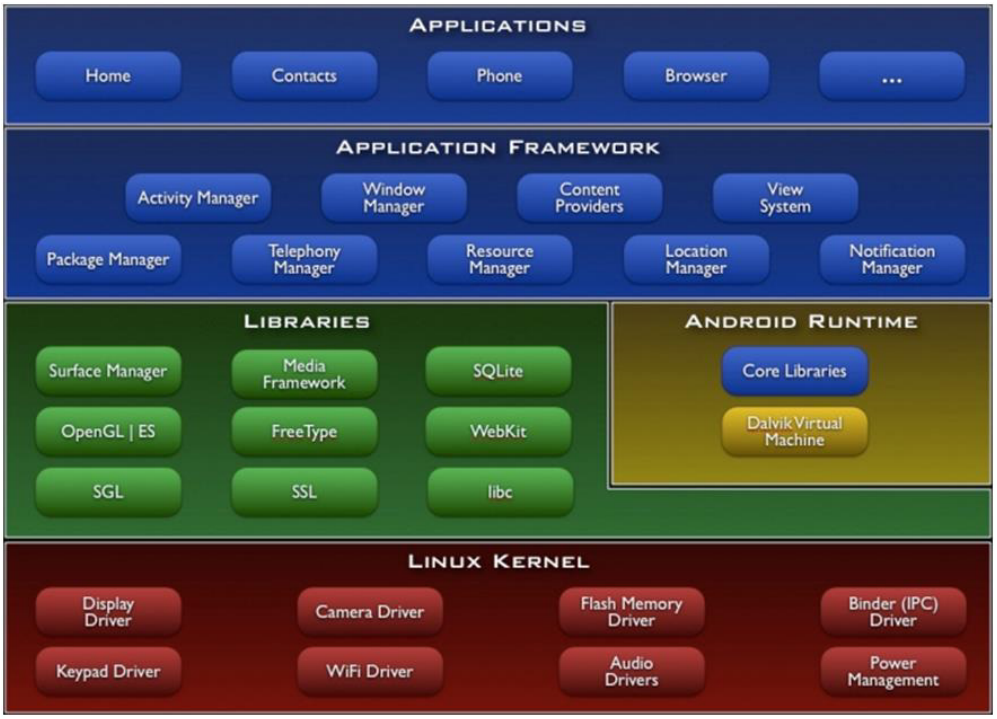
Ainda, segundo levantamento feito por IDC (2014) para o terceiro trimestre de 2014, cerca de 84% dos smartphones enviados ao mercado correspondiam a smartphones com Android. Levantamentos também da própria Google apontam para o sucesso da plataforma. De acordo com a Google, em sua conferência anual realizada no início de 2014, a Google I/O (2014), foi analisado durante um período de 30 dias quantos dispositivos Android foram conectados à internet: cerca de 1 bilhão de dispositivos foram conectados. Por isso, é importante conhecermos também um pouco da arquitetura e do ambiente de desenvolvimento Android.

2.7.1 Arquitetura Android

O Android é um *software stack* composto de várias camadas: Núcleo Linux (*Kernel Linux*), Bibliotecas (*Libraries*) e Android Runtime, *Framework* de Aplicações (*Application Framework*) e Aplicativos (*Applications*) como mostra a Figura 12. Em sua base, temos uma versão modificada do Núcleo Linux 2.6, que provê vários serviços essenciais, tais como: segurança, rede e gerenciamento de memória e processos, além de uma camada de abstração de *hardware* para as outras camadas de *software* (ANDROID DEVELOPERS, 2014). A Figura 12 mostra a arquitetura do Android e suas camadas hierárquicas.

Esta versão do Núcleo Linux foi modificada com o objetivo de melhor atender as necessidades existentes nos dispositivos móveis, como as de gerenciamento: de memória, de energia e do ambiente de execução. Entre essas modificações podemos citar o Binder, um novo mecanismo para a comunicação entre processos (IPC - *Inter-process Communication*) e chamada remota de métodos, que permite que um processo possa chamar uma rotina em outro processo.

Figura 12- Arquitetura Android.



Fonte: ANDORID DEVELOPERS, 2014

Com relação a memória, o Android introduz um mecanismo (OOM – *Out-of-Memory Handler*) para terminar processos quando há falta de memória, além do *ashmem* e *pmem*. O primeiro deles é um alocador de memória compartilhada, com melhor suporte à dispositivos com pouca capacidade de memória. Já o segundo é um alocador de memória de processo, e é utilizado para o gerenciamento de grandes regiões contíguas de memória física compartilhadas entre os espaços dos usuários e *drivers* do núcleo (ANDROID DEVELOPERS, 2014).

Acima do núcleo ficam as bibliotecas C/C++ utilizadas por diversos componentes do sistema como: uma implementação da biblioteca padrão do C (libc), otimizada para dispositivos embarcados; bibliotecas para suporte a formatos de áudio, vídeo e imagens; um gerenciador que intermedia o acesso ao display e compõe as camadas de imagem 2D e 3D; uma *engine* para navegadores WebKit; bibliotecas para gráficos 2D (SGL) e 3D (OpenGL ES); um renderizador de fontes bitmap e vetoriais; e o banco de dados relacional SQLite (ANDROID DEVELOPERS, 2014).

Já na camada Android Runtime se encontra um conjunto de bibliotecas do núcleo Java (*core libraries*) e a Máquina Virtual Dalvik, onde as aplicações são executadas. No Android, aplicações escritas em Java são executadas em sua própria instância de máquina virtual, que por sua vez é executada em seu próprio processo no Linux, isolando-a de outras aplicações e

facilitando o controle de recursos. Cada instância de máquina virtual possui sua própria pilha de memória e seu próprio *garbage collector.*

A Dalvik é frequentemente referenciada como uma Máquina Virtual Java, mas isso não é estritamente exato, devido ao bytecode que ela opera não ser o bytecode da JVM (*Java Virtual Machine*). Ao invés, uma ferramenta chamada dx, incluída no Android SDK, transforma os arquivos .class de uma classe compilada por um compilador java, comum para a JVM, em outro formato especifico de classe; o formato .dex.

A camada de *Framework* de Aplicações é a camada de um conjunto de ferramentas que fornecem vários serviços para os aplicativos visando o reuso de componentes.

Aplicações normalmente são compostas por atividades (*Activity*). No Android uma atividade é uma ação específica que um usuário pode realizar dentro de um aplicativo. Estas ações podem incluir a inicialização de outras atividades, tanto dentro como fora da aplicação a qual ela pertence, através de intenções (*Intent*), criando o que se chama de tarefa. Uma tarefa pode conter várias atividades, que são organizadas em uma pilha na ordem em que foram criadas, permitindo assim que o usuário volte para atividades em que estava anteriormente (ANDROID DEVELOPERS, 2014).

Quando uma nova tarefa é iniciada ou quando o usuário retorna a tela inicial, a tarefa anteriormente em primeiro plano passa para o segundo plano. Uma tarefa em segundo plano pode retornar ao primeiro plano e outras várias tarefas podem estar em segundo plano ao mesmo tempo. Entretanto, se muitas tarefas estiverem em segundo plano, o sistema pode começar a finalizar atividades em execução para recuperar memória, fazendo com que as atividades percam seus estados, o que não significa que o usuário não possa mais navegar de volta a esta atividade.

Todo o ciclo de vida das atividades, bem como a organização das mesmas em pilhas e tarefas é de responsabilidade do Gerenciador de Atividades (*Activity Manager*). Além dele, ainda existem: Gerenciador de Pacotes (*Package Manager*), responsável por manter o registro de todas as aplicações instaladas no dispositivo; Gerenciador de Janelas (*Windows Manager*), que gerencia a tela do dispositivo e cria superfícies para as aplicações em execução; Gerenciador de Telefone (*Telephony Manager*), que fornece a outras aplicações serviços relacionados com telefonia; Provedor de Conteúdo (*Content Provider*), para o compartilhamento de dados entre aplicações; Gerenciador de Recursos (*Resource Manager*), que prove acesso a tudo aquilo que faz parte de um aplicativo e que não seja código (recursos), como arquivos XML, bitmaps ou outros arquivos, como sons.

Existe ainda o Sistema de Visualização (*View System*), que fornece um conjunto básico de *views* que podem ser utilizadas e estendida por aplicações, incluindo: listas, caixas de texto, tabelas e botões, e o Gerenciador de Notificações (*Notification Manager*), que permite que uma aplicação notifique o usuário sobre algum evento através de vibração, piscar de *leds*, tocar algum som ou mostrar um ícone na barra de status (ELINUX, 2014).

No nível mais ao topo se encontra a Camada de Aplicações (*Applications*). Esta é a camada de interação entre o usuário e o dispositivo móvel. É onde está localizado uma lista de aplicações padrão, como cliente de e-mail, programa de SMS (*Short Message Service*), calendário, mapas, navegador, gerenciador de contatos e outros que são desenvolvidos.

2.7.2 Android SDK

O Android SDK (*Software Development Kit*) ou Kit de Desenvolvimento de Programas Android é um conjunto de ferramentas para desenvolver, testar e depurar aplicativos Android. Ele inclui várias ferramentas, tais como: um *debugger,* para testar e depurar programas, um conjunto de bibliotecas da API Android, documentação, códigos de amostra e tutoriais. Além destes componentes, o Android SDK também possui um emulador baseado em QEMU (*quick emulator* – emulador rápido), que é um hipervisor livre e de código aberto que gerencia e executa múltiplos sistemas operacionais num mesmo *hardware*.

Os pacotes modulares que compõem o Android SDK podem ser baixados separadamente usando o Gerenciador SDK (SDK *Manager*). Assim, quando uma nova versão da plataforma Android ou as ferramentas do SDK são atualizadas, este gerenciador é usado para baixá-los separadamente. Entre exemplos de pacotes disponíveis, podemos encontrar:

- Ferramentas SDK, contém ferramentas de depuração e teste. Como exemplo de ferramentas temos o *traceview,* para visualizar graficamente os *logs* criados usando o *debug,* o *Dalvik Debug Monitor de Server* (DDMS), que fornece depuração de recursos como captura de tela e informações de *thread* e *heap*, e o *Android Debug Bridge* (ADB), que fornece acesso a um dispositivo do sistema de desenvolvimento;
- Ferramentas de Plataforma SDK, normalmente são atualizadas assim que uma nova plataforma Android é lançada a fim de suportar os mais recentes recursos. Usado para depuração e desenvolvimento da aplicação e é dependente de plataforma;

- Documentação, uma cópia *off-line* da documentação mais recente da API da plataforma Android;
- Plataforma SDK, há uma plataforma SDK disponível para cada versão do Android. Ela inclui um arquivo chamado android.jar com uma biblioteca Android totalmente compatível. A fim de criar um aplicativo Android, deve ser especificada uma plataforma SDK como o seu destino de compilação;
- Códigos de amostra, uma coleção de aplicativos de exemplo que demonstram uma variedade de APIs da plataforma. Estes são um grande recurso para navegar código do aplicativo Android. Demos do aplicativo API em particular fornece um grande número de pequenas demos que você deve explorar.

Atualmente, o Android SDK oferece suporte ao Linux, Windows e Mac OS X. O Ambiente de Desenvolvimento Integrado (IDE - *Integrated Development Environment*) mais usado é o Eclipse associado com o *Android Development Tools* (ADT) *plugin,* embora se possa utilizar também as IDEs Netbeans e MotoDev. Juntamente com o *Java Development Kit* (JDK) e *o Java Runtime Environment* (JRE), estes recursos formam o ambiente de desenvolvimento para Android. Apesar de suas constantes melhorias e surgimento de novas versões da plataforma Android, o SDK também suporta versões antigas deste sistema. Isto é muito útil para o caso em que desenvolvedores desejem direcionar os seus aplicativos também para dispositivos mais antigos.

2.7.3 Android Development Tools (ADT) Plugin

O ADT é um *plugin* associado ao Eclipse (FOUNDATION, 2014) que fornece um conjunto de ferramentas que são integrados com o Eclipse IDE, formando o Eclipse ADT (ANDROID DEVELOPERS, 2014). Ele oferece acesso a diversos recursos que ajudam a desenvolver aplicações Android rapidamente. A *Graphical User Interface* (GUI) permite o acesso a muitas das ferramentas de linha de comando do SDK, bem como uma ferramenta de design de UI (*User Interface*) para prototipagem rápida, concepção e construção de interface de usuário do aplicativo (ANDROID DEVELOPERS, 2014).

Pode-se destacar como principais características desse *plugin* a integração das ferramentas do Android SDK, bem como a criação, construção, empacotamento, instalação e depuração de projetos Android com o menu do Eclipse. Além destas, também podemos destacar editores XML (*Extensible Markup Language*) personalizados que permitem editar arquivos XML específicos do Android (recursos, manifesto, menus, layouts) em uma

interface de usuário baseada em formulário e um editor de layout gráfico que permite criar interfaces de usuário com uma interface de arrastar e soltar.

Para o desenvolvimento deste trabalho, este *plugin* foi utilizado para integrar o Eclipse IDE e o Android SDK.

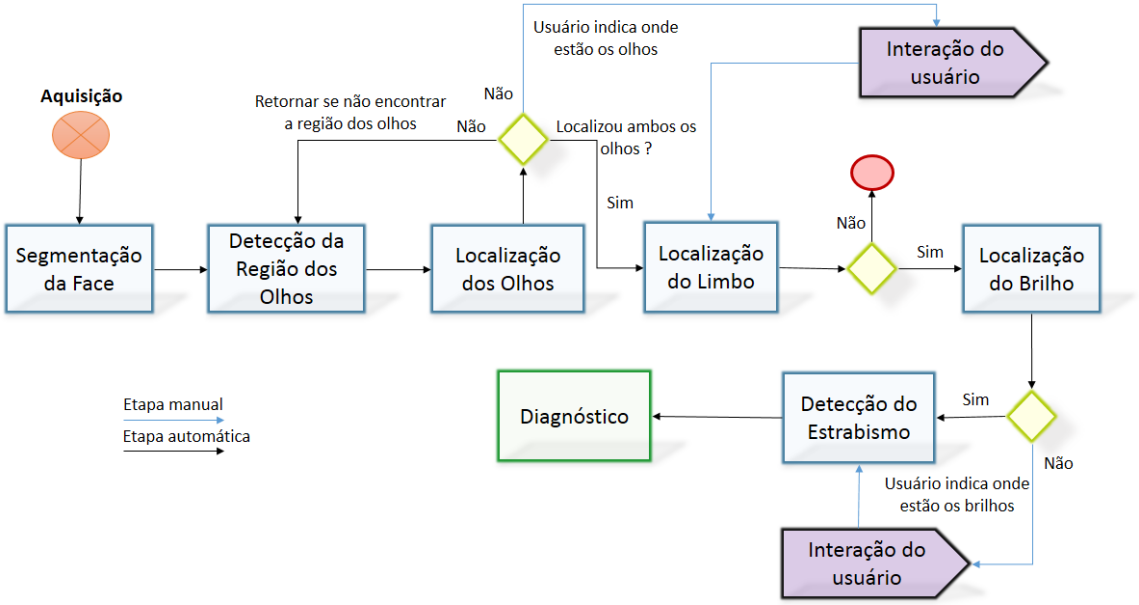
3 APLICATIVO DESENVOLVIDO

A portabilidade da metodologia proposta por Almeida (2013) para dispositivos móveis baseado em Android teve como consequência a produção de um Aplicativo. Este foi desenvolvido para versões iguais ou superiores ao Android 4.1. A escolha desta foi feita mediante ao fato de que celulares com uma versão inferior a escolhida não têm capacidade de *hardware* suficiente para um bom desempenho de execução, fora o fato de extinção destes do mercado. Aconselha-se que se utilize dispositivos com *hardware* próximo ou superior ao especificado para teste na seção 3.1.1. A Figura 13 ilustra a sequência das etapas de execução.

O aplicativo é multilíngue, tendo como idioma padrão o inglês. Caso o dispositivo em que o aplicativo se encontra instalado esteja configurado com idioma português brasileiro, o aplicativo automaticamente mudará para o idioma português brasileiro. Todas as palavras do aplicativo foram escritas no arquivo em seus respectivos arquivos string.xml. O sistema se encarrega de carregar o arquivo correto de acordo com a configuração de idioma do dispositivo. Como tal, algumas figuras presentes neste trabalho podem aparecer em inglês ou em português brasileiro. Outros idiomas são facilmente adicionados, e poderão ser adicionados futuramente.

Nas seções a seguir, serão descritos os materiais utilizados na portabilização da metodologia (*hardware e software*), o protocolo de aquisição de imagens, a base utilizada para testes e validação do mesmo e cada etapa de execução do aplicativo. A execução do aplicativo está organizada em 7 etapas automáticas, sendo que em 2 delas pode haver interação do usuário, e a inicial, onde com certeza há interação do usuário.

Figura 13- Etapas de execução do aplicativo



3.1.1 Software e Hardware Utilizados

Para o desenvolvimento do aplicativo, foi utilizada a linguagem Java. A escolha dessa linguagem se deu, principalmente, por ser a linguagem utilizada pelo SDK do Android, fornecida pela Google. Ao todo, as ferramentas utilizadas foram:

- Eclipse ADT (ANDROID DEVELOPERS, 2014);
- Android SDK (ANDROID DEVELOPERS, 2014);
- JRE 8 (*Java Runtime Environment*) (ORACLE, 2014);
- JDK 8 (*Java Development Kit*) (ORACLE, 2014).

O Eclipse ADT foi utilizado como IDE (*Integrated Development Environment*) para o desenvolvimento. O *plugin* ADT (Seção 2.7.3) foi usado para *linkar* a IDE com o Android SDK, que é o *kit* de desenvolvimento para Android (Seção 2.7.2). O JRE composto pela *Java Virtual Machine* (JVM), pelas classes de núcleo da plataforma Java e bibliotecas da plataforma Java para suporte. Já o JDK foi utilizado por ser necessário para o desenvolvimento utilizando a linguagem Java, com recursos e bibliotecas.

Além destes, foram utilizadas ainda as bibliotecas:

- Jhlabs para Android (MORDONEZ-ME, 2014): contém vários filtros de processamento de imagem. Desta, foi utilizado o filtro *SmartBlur* para suavizar a imagem com objetivo de reduzir ruídos e diminuir detalhes na textura da pele. Isto foi feito para que melhorasse os resultados da etapa de detecção de bordas, pois ruídos poderiam ser detectados como bordas e detalhes na pele do paciente não interessam;
- SubSampling scale image view (DAVEMORRISSEY, 2014): contém janelas personalizadas com detecção de gestos para *zoom*, *pan*, rotação, animação e eventos de toque. Essa biblioteca foi modificada para carregar imagens diretamente de objetos bitmaps, pois ela carregava apenas de arquivo. Ela está sob licença *Apache License Version* 2.0, permitindo assim seu uso e modificação;
- OpenCV para Android 2.4.6 (OPENCV, 2014): OpenCV é uma biblioteca de visão computacional, possuindo módulos de processamento de imagens e reconhecimento de padrões, além de várias estruturas de dados. Ela é liberada sob a licença BSD e, portanto, é gratuita tanto para uso acadêmico e comercial. Foi utilizada para desenhar formas simples na imagem, como círculo e quadrado.
- LIBSVM (CHANG; LIN, 2014): biblioteca para utilização da MVS.

Relativo ao *hardware*, fora utilizado um Notebook ASUS Vivobook com processador Core i5-3317U 1.7 Ghz, 4GB de memória RAM DDR3 com sistema operacional Windows para a programação. Na execução e testes do aplicativo foi utilizado um *smartphone* Sony Xperia ZQ com processador 1,5 GHz Qualcomm APQ8064 Quad Core, 2GB de memória RAM com sistema operacional Android 4.3.

3.1.2 Base de Imagens e Protocolo de Aquisição

Utiliza-se a base de imagens de pacientes elaborada por Almeida (2013). A esta base foram adicionadas 5 imagens de pacientes na posição primária do olhar (PPO), adquiridas com resolução de 2048 x 1536 *pixels,* totalizando 45 imagens.

Almeida (2013) definiu um protocolo mínimo a ser seguido. Segundo ele, a aquisição deve ser realizada pelo especialista a cerca de 40-50cm do paciente e com a câmera configurada com *zoom* óptico 3x. Além disso, o especialista deve estar posicionado à frente do paciente que, por sua vez, deve estar sentado numa cadeira com o rosto centralizado e olhando para a objetiva da câmera. O ambiente deve estar com a luz ligada, sem foco de luz complementar que não seja o *flash* da câmera, que deve estar ativado. Caso o paciente utilize lentes corretivas, a foto será adquirida com as mesmas. Esse protocolo deve ser seguido, também, na utilização do aplicativo desenvolvido neste trabalho para aquisição e diagnósticos em novas imagens.

É importante frisar que o dispositivo, no momento da aquisição de imagens, deve estar na posição horizontal. Isto é necessário para que a imagem seja adquirida de tal forma que a largura seja maior do que a altura.

3.2 Aquisição de Imagens

Nesta etapa, o usuário carrega a imagem no aplicativo. O sistema Android permite que aplicativos iniciem outros aplicativos através de mensagens enviadas ao sistema. O sistema os inicia, para que estes executem alguma tarefa e retorne dados ao aplicativo que o invocou, conforme descrito na seção 2.7.1.

Aproveitando-se disto, o aplicativo fornece a opção para que o usuário inicie um aplicativo de galeria, para que seja selecionada uma imagem dentre as armazenadas no dispositivo. Há ainda a opção de iniciar um aplicativo de câmera, para que o usuário faça a captura da imagem. Vale lembrar que, em ambos os casos, o usuário deve obedecer a resolução mínima exigida pelo aplicativo, que é de 2048 x 1536 *pixels,* e obedecer, por

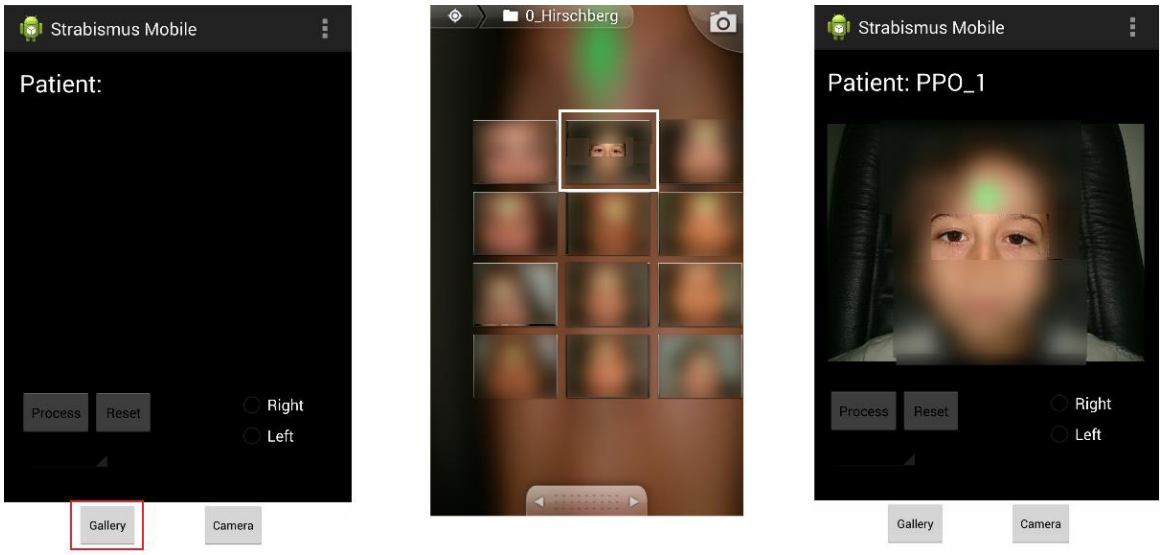
conseguinte, a proporção da mesma, que é de 4:3, além de configurar o *zoom* óptico da câmera para 3x. Caso a resolução seja maior, a imagem será automaticamente redimensionada para a resolução mínima. No entanto, caso a proporção seja diferente, a largura da imagem poderá estender-se ou comprimir-se, o que poderá ocasionar erros no diagnóstico.

Para a captura de imagens pelo dispositivo, este deve estar na posição horizontal. Esta posição garante que a largura da imagem será maior do que a altura. Outro ponto é que não é necessário que o usuário preocupe-se se o dispositivo está invertido em quaisquer direção ou não. Isto porque quando um dispositivo captura uma imagem, este salva sua orientação nos dados exif[12] da mesma. Aproveitando-se disto, o aplicativo recupera a informação de orientação, se houver, e gira a imagem em 180º, caso seja necessário. As Figuras 14 (a), (b) e (c) mostram uma aquisição realizada da galeria de imagens.

3.3 Segmentação da Face

Esta etapa tem por objetivo delimitar a área da face visando eliminar o fundo da imagem com a intenção de diminuir o espaço de busca. A Figura 15 apresenta o fluxograma do algoritmo utilizado para a detecção da face. Nesta Figura, são apresentadas as imagens resultantes de cada uma das etapas do algoritmo, onde a imagem resultado de cada etapa é a entrada da etapa seguinte. O resultado final é utilizado na próxima etapa da execução do programa.

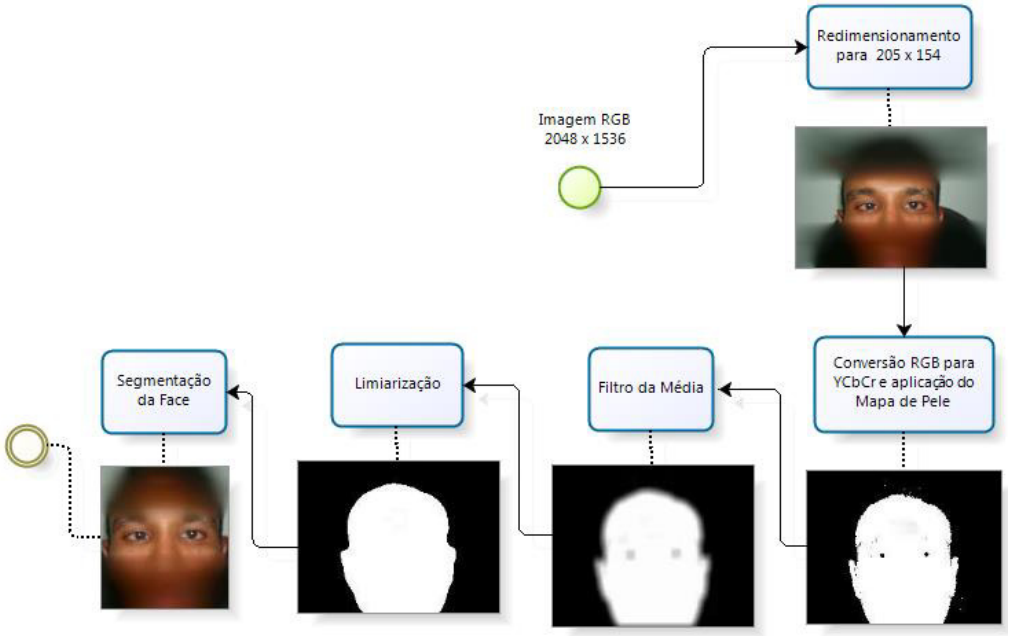
Figura 14- (a) Tela inicial do aplicativo. (b) Imagem sendo selecionada na galeria.



(a) (b) (c)

[12] Exif é uma especificação seguida por fabricantes de câmeras digitais que gravam informações sobre as condições técnicas de captura da imagem junto ao arquivo da imagem propriamente dita na forma de metadados etiquetados

Figura 15- Fluxograma da fase de Segmentação da Face.



Fonte: ALMEIDA, 2013.

Primeiramente a imagem é redimensionada 10x menor para reduzir o custo computacional. Depois, a imagem é convertida do modelo de cor RGB para YCbCr, através da seguinte operação linear:

$$\begin{vmatrix} Y \\ Cb \\ Cr \end{vmatrix} = \begin{vmatrix} 16 \\ 128 \\ 128 \end{vmatrix} + \begin{vmatrix} 0{,}257 & 0{,}504 & 0{,}098 \\ -0{,}148 & -0{,}291 & 0{,}439 \\ 0{,}439 & -0{,}368 & -0{,}071 \end{vmatrix} + \begin{vmatrix} R \\ G \\ B \end{vmatrix} \quad (17)$$

Para segmentar a face utiliza-se o mesmo algoritmo desenvolvido por Chai e Ngan (1999), de segmentação da pele. Neste algoritmo, o mapa da cor da pele é obtido através dos canais de crominância do modelo YCbCr, o Cb e o Cr, utilizando o seguintes limiares:

$$77 \leq Cb \leq 127 \; e \; 133 \leq Cr \leq 173 \quad (18)$$

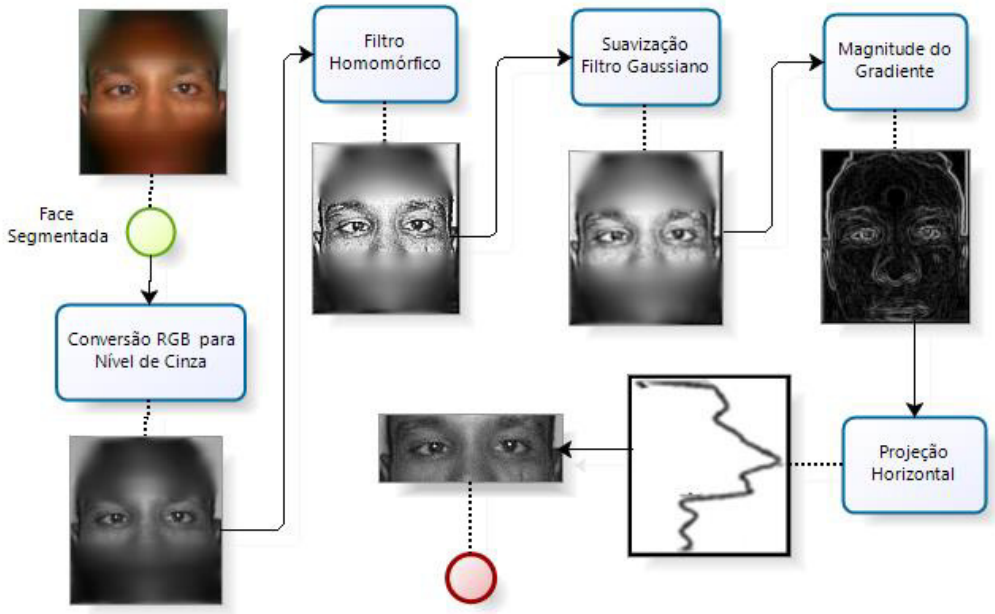
Em seguida, aplica-se o filtro da média (GONZALEZ; WOODS, 2002) usando uma máscara 9x9 para redução de ruídos e, em seguida, aplica-se a limiarização da imagem, tomando como limiar o valor médio da intensidade dos *pixels* da imagem. Por último, as coordenadas e dimensões da região da face são extraídas verificando os limites dos pixels brancos existentes nas extremidades horizontal e vertical da imagem. Essas coordenadas são utilizadas para criar uma sub imagem a partir da imagem original, delimitando a face.

3.4 Detecção da Região dos Olhos

O objetivo desta etapa é gerar uma sub imagem com a possível região dos olhos, reduzindo, ainda mais, o espaço de busca na imagem. Para isso, excluíram-se as regiões que

não têm interesse, como boca, nariz e testa. A Figura 16 representa os passos realizados na detecção automática da região dos olhos.

Figura 16– Fluxograma da Detecção da Região dos Olhos.



Fonte: ALMEIDA, 2013.

Essa fase tem por entrada a imagem resultante da etapa anterior. Assim, primeiramente a imagem é convertida do modelo de cor RGB para níveis de cinza com o objetivo de, novamente, reduzir o custo computacional. Para tal conversão foi utilizada a equação:

$$NC = 0{,}299 * R + 0{,}587 * G + 0{,}114 * B \qquad (19)$$

onde NC é o valor do *pixel* em nível de cinza, R é o canal da primitiva vermelho, G da primitiva verde e B da primitiva azul. Em seguida, aplica-se o filtro homomórfico, para diminuir a influência da iluminação intensa pontual do brilho e aumentar o contraste local da região dos olhos, dando mais equilíbrio às diferenças de intensidade. Os fatores multiplicativos utilizados no filtro foram os mesmos de Almeida (2013), $\alpha = 0{,}9$ *e* $\beta = 1{,}1$. Para os filtros de passa-alta e passa-baixa foram utilizadas as mesmas máscaras utilizadas por Melo *et al.* (2005) em aplicações homomórficas.

Dando continuidade, aplica-se o filtro gaussiano de máscara 3x3 para suavizar a imagem. Depois, calcula-se o gradiente da imagem de entrada, gerada na fase anterior, utilizando filtro de Sobel (GONZALEZ; WOODS, 2002). E, por último, aplica-se a projeção horizontal desse gradiente. Levando em conta que os olhos se localizam na região superior da face e analisando a saída, percebe-se que, dentre os três maiores picos, os olhos estão entre os dois picos mais próximos. Tais informações são utilizadas para identificar a área de interesse.

Ainda, para eliminar interferências da região da boca e da região superior dos cabelos, a projeção é aplicada de 1 /5 a 1 /2.5 da altura da imagem da face.

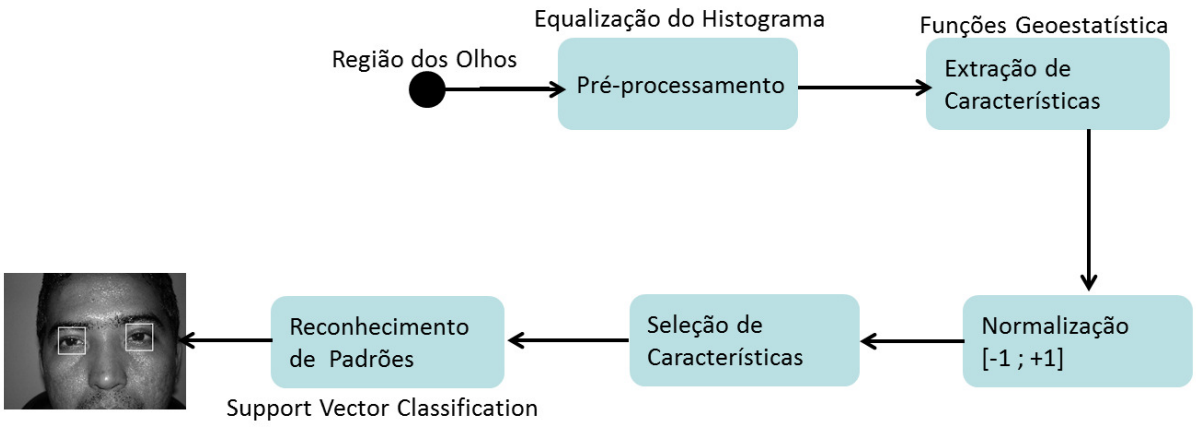
3.5 Localização dos Olhos

Esta etapa é crucial para a continuidade da execução automática do aplicativo. Porém, caso haja falha, o usuário poderá indicar manualmente a localização dos olhos esquerdo e direito, para que posteriormente o aplicativo continue a execução. Nesta seção, serão abordadas tanto a detecção dos olhos realizada de forma automática quanto a realizada manualmente.

3.5.1 Abordagem Automática

Esta abordagem se caracteriza por não precisar da intervenção do usuário para ser executada. Nela, utilizam-se técnicas de processamento de imagens e reconhecimento de padrões. A Figura 17 apresenta as etapas seguindo esta abordagem.

Figura 17– Etapas da Localização dos Olhos.



Fonte: Adaptado de ALMEIDA, 2013.

Por utilizar técnicas de reconhecimento de padrões, ela é realizada em duas fases: treinamento e teste. As etapas serão explicadas a seguir.

3.5.1.1 Base de Treino e Teste

Foi utilizado o classificador treinado por Almeida (2013), onde foram selecionadas 1050 amostras de 35 imagens de pacientes, sendo 18 regiões de olho, 9 de cada olho, e 12 regiões de outras áreas da face. Ou seja, para cada imagem de posição do olhar utilizada no treinamento foram extraídas, manualmente, 30 amostras formadas por uma janela de 30x30

pixels. Porém, este trabalho trata apenas situações em que o paciente está na posição primária do olhar (PPO).

Para teste, os candidatos a olhos foram detectados através da aplicação da transformada de Hough, descrita na seção 2.4.4, baseada em Annulus Orientado (D'ORAZIO *et al*., 2007). O intervalo de raio usado foi de 4 a 10 *pixels*, determinado a partir da análise no banco de imagens utilizado no teste.

Para a aplicação do Annulus, primeiramente calculou-se as derivadas parciais nas componentes X e Y através da convolução dos núcleos do operador de Sobel (GONZALES; WOODS, 2002) sobre a imagem. Após isto, aplica-se a convolução do núcleo da TH no domínio discreto, em X e Y, nas saídas das componentes X e Y da etapa anterior. A soma dos resultados representará o acumulador da TH e, como tal, quanto maior o valor em uma posição, maior a chance dessa posição representar o centro de um círculo (um candidato a olho).

Para finalizar esta sub etapa de teste, as coordenadas são organizadas em ordem decrescente de acumulação. Delas, são extraídas seis coordenadas das que possuem a coordenada X menor ou igual à metade do comprimento da imagem, para cada olho.

3.5.1.2 Pré-processamento e Extração de Características

Para o pré-processamento, aplica-se a equalização do histograma para melhorar o contraste, como descrito na seção 2.4.2. Depois, aplica-se as funções geoestatísticas correlograma, covariograma, semivariograma e semimadograma, apresentadas na seção 2.5.1, para descrever a textura dos objetos.

Os parâmetros usados pelas funções geoestatísticas (seção 2.5.1) para extração das características em cada amostra foram as direções 0º, 45º, 90º e 135º com tolerância angular de 22,5º, incremento de *lag* (distância) igual a 1, 2 e 3 correspondendo a 29, 14 e 9 *lags* e tolerância de cada distância de *lag* igual a 0,5, 1,0 e 1,5, respectivamente. As direções adotadas são as mais utilizadas na literatura para análise de imagens; já para escolher a tolerância de *lag* segundo Isaacs e Srivastava (1989), a escolha mais comum é adotar a metade do incremento de *lag*. (ALMEIDA, 2013). A janela (imagem) utilizada tem tamanho 30x30.

São extraídas 208 características por amostra, correspondentes às quatro direções de 52 *lags* (29 + 14 + 9) para cada função geoestatística, totalizando 832 características para formar o vetor de características (VC). No entanto, foram utilizadas as 29 características

selecionadas por Almeida (2013) através da análise discriminante *stepwise* (LACHENBRUCH; GOLDSTEIN, 1979).

3.5.1.3 Normalização

O objetivo desta sub etapa é a padronização da distribuição dos valores no intervalo de -1 a 1 e facilitar o classificador a convergir com maior facilidade na etapa de treinamento. Para normalizar os dados utiliza-se a equação definida por:

$$Norm = \frac{2 * (num - min)}{\max - min} - 1 \quad (20)$$

Onde *Norm* é o valor normalizado, *min* o menor valor e *max* o maior valor da caraterística, respectivamente.

3.5.1.4 Reconhecimento de Padrões

A etapa final consiste em classificar cada objeto em olho ou não-olho utilizando técnicas de reconhecimento de padrões (seção 2.6) de acordo com a análise de textura obtida pelas funções geoestatísticas (seção 2.5.1).

Para esta etapa, foi utilizado o MVS através da biblioteca LIBSVM (CHANG; LIN, 2003). O núcleo do MVS utilizado é do tipo radial (RBF) com os mesmos parâmetros C e γ estimados por Almeida (2013). Depois, utiliza-se a técnica *v-fold* para a validação cruzada dos dados. Esta técnica consiste em dividir o conjunto de treinamento em *v* subconjuntos de tamanhos iguais e, com o classificador treinado, um subconjunto é testado no restante do subconjunto *v-1*. Foi utilizado 10 subconjuntos, número recomendado por Winter e Frank (2005) para se obter estimativas confiáveis. Cada experimento foi repetido 10 vezes.

Para a construção do modelo, foi descrito um conjunto de classes pré-determinadas, cada qual com seu rótulo de atributo-classe, construído a partir do conjunto de treinamento. Então, cada exemplo é considerado pertencente a uma dessas classes.

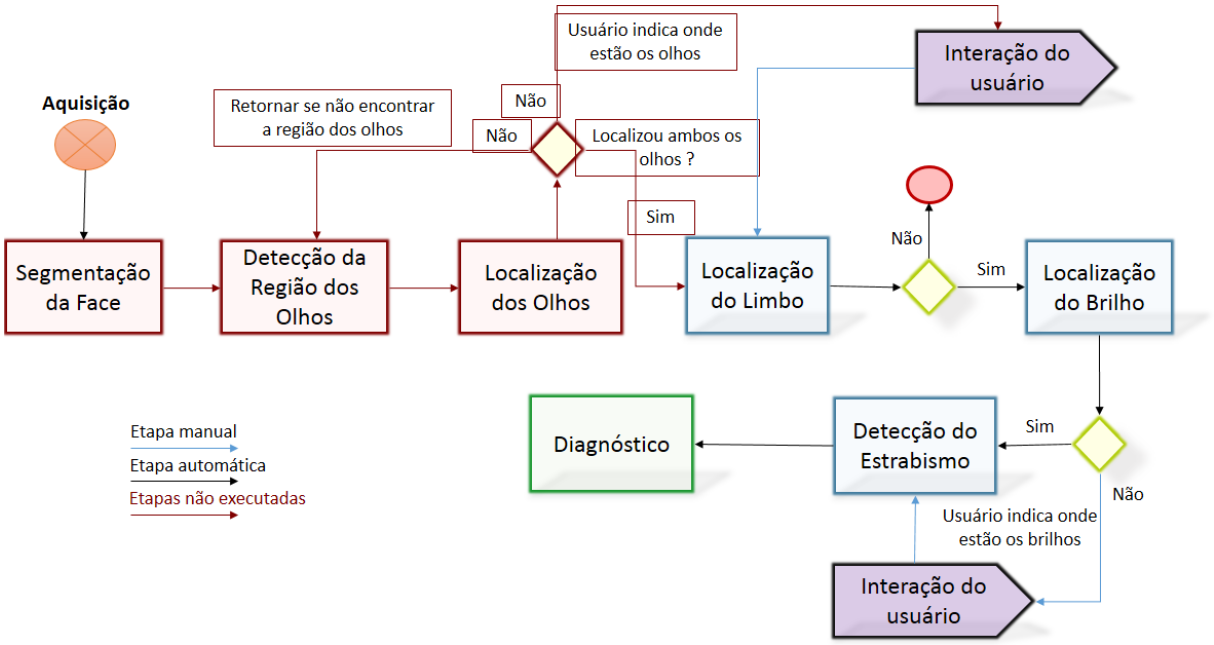
O reconhecimento encerra-se com a classificação da MVS gerando a localização automática dos olhos. Caso nenhum dos olhos sejam detectados, considera-se que os olhos não estão na imagem segmentada na fase anterior (seção 3.4). Assim, a fase anterior é repetida ajustando-se a coordenada Y através do valor atual somado à metade da altura da segmentação anterior, ou seja, a imagem é percorrida de cima para baixo. Este processo é repetido até que se ache os olhos, ou até que não se tenha mais *pixels* a serem verificados onde, neste caso, a metodologia é encerrada. Para determinar se um olho é direito ou esquerdo, verifica-se se a coordenada X está mais à direita ou esquerda da imagem.

3.5.2 Abordagem Manual

Esta abordagem pode ser executada em duas situações: (1) falha na execução da abordagem automática ou (2) por preferência do usuário em executar manualmente. O usuário pode preferir usar esta abordagem quando quiser respostas mais rápidas, visto que não são executadas as etapas de segmentação da face (seção 3.3), detecção da região dos olhos (seção 3.4) e localização automática dos olhos (seção 3.5.1), que demandam tempo e custo computacional. O tempo médio de processamento, a partir da etapa de segmentação da face (abordagem automática) no dispositivo móvel de teste, foi de 60 segundos podendo, em poucos casos, demorar mais. Já na abordagem manual, depois que o usuário já indicou onde estão os olhos e mandou processar, foi de aproximadamente 20 segundos, independentemente de grandes diferenças entre as imagens dos pacientes (lentes corretivas, manchas na pele, excesso de reflexo, imagens não muito bem focadas, diferenças de distância entre o paciente e a câmera).

A Figura 18 apresenta as etapas da execução da metodologia proposta a partir da interação do usuário na etapa de localização dos olhos, independentemente de qual dos dois casos ocorra**.**

Figura 18– Metodologia seguindo a abordagem manual de localização dos olhos



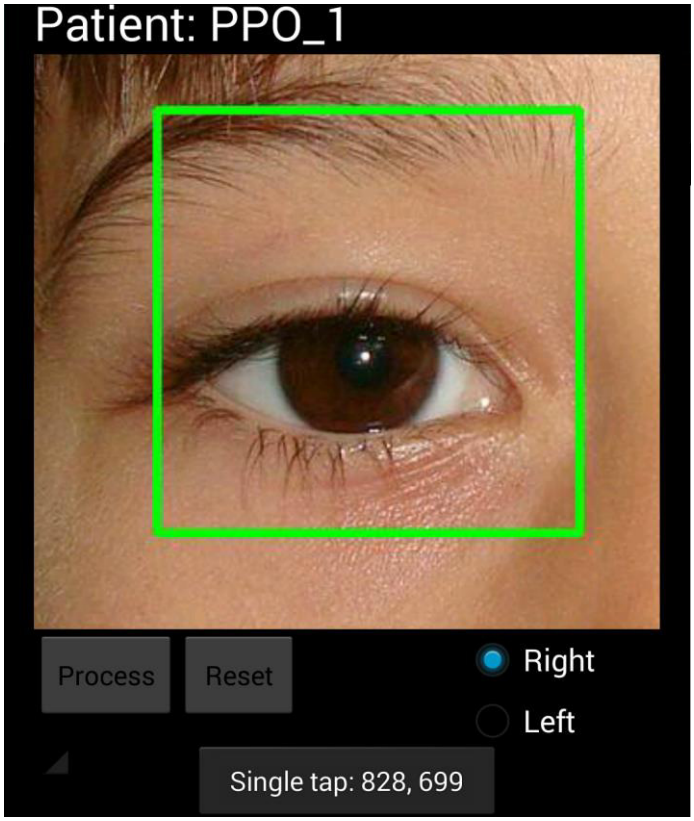
Carregada a imagem no aplicativo, o usuário poderá selecionar as coordenadas onde estão cada um dos olhos. Esta seleção é feita através de toque simples na região da imagem,

onde o usuário primeiramente seleciona a opção correspondente indicando qual o lado do olho irá trabalhar e toca na localização do olho.

Feito isso, será desenhada uma janela verde e quadrada, de dimensões 241x241, tendo as coordenadas do toque como centro da janela. Essas dimensões são calculadas da seguinte maneira: primeiramente definiu-se um raio *r* de valor 120 e, tendo como centro *c(x,y)* da janela o toque do usuário, faz-se *c(x,y)* – 120 e *c(x,y)* + 120 para calcular os pontos extremos superior esquerdo e inferior direito, respectivamente. Este valor do raio foi estipulado empiricamente, através de testes no banco de imagens utilizado no desenvolvimento deste trabalho e levando em conta a resolução da imagem e o protocolo de aquisição que deve ser obedecido (seção 3.1.2).

A Figura 19 mostra a localização realizada de forma manual do olho direito de um paciente. Na região inferior da imagem são mostrados os valores x e y da coordenada onde o usuário tocou.

Figura 19 – Olho direito sendo localizado manualmente pelo usuário



Como se trata de dados fornecidos pelo usuário, a abordagem manual é suscetível a erros. Por isso, caso o usuário erre a localização dos olhos, ele tem a opção de realizar o procedimento novamente, quantas vezes for necessário, por meio do botão *reset*, disponível no aplicativo. Além disso, o usuário poderá dar um duplo toque na região da imagem para

realizar a operação de *zoom* na mesma, com o objetivo de facilitar a seleção do centro da janela e permitir que o usuário possa confirmar que a borda do limbo e o centro do brilho estão detectados precisamente.

3.6 Localização do Limbo

Para localizar o limbo, a imagem original (2048 x 1536 *pixels*) é inicialmente redimensionada para 60% de suas dimensões (1229 x 922 *pixels*), visando reduzir o custo computacional ao passo que não se perca detalhes das bordas do limbo e do brilho.

Na abordagem automática da detecção dos olhos, se trabalhou com resolução 10 vezes menor. Por isso, as coordenadas dos olhos detectados precisam ser aumentadas 6 vezes, visando a correspondência com a resolução que será trabalhada na localização do limbo (60% da original).

Em seguida, é realizado um pré-processamento utilizando o filtro *Smart Blur* implementado para Android (MORDONEZ-ME, 2014) para eliminar ruídos preservando, o máximo possível, as bordas. A imagem é então convertida para nível de cinza e logo aplica-se o método de Canny para detecção das bordas seguido da TH, que utilizará o mapa de bordas gerado. Um bom resultado desta fase depende de que uma porcentagem razoável das bordas do limbo estejam bem definidas, para que a TH possa localizá-lo com maior precisão. Os parâmetros das técnicas foram estabelecidos de modo empírico e são mostrados no Quadro 1:

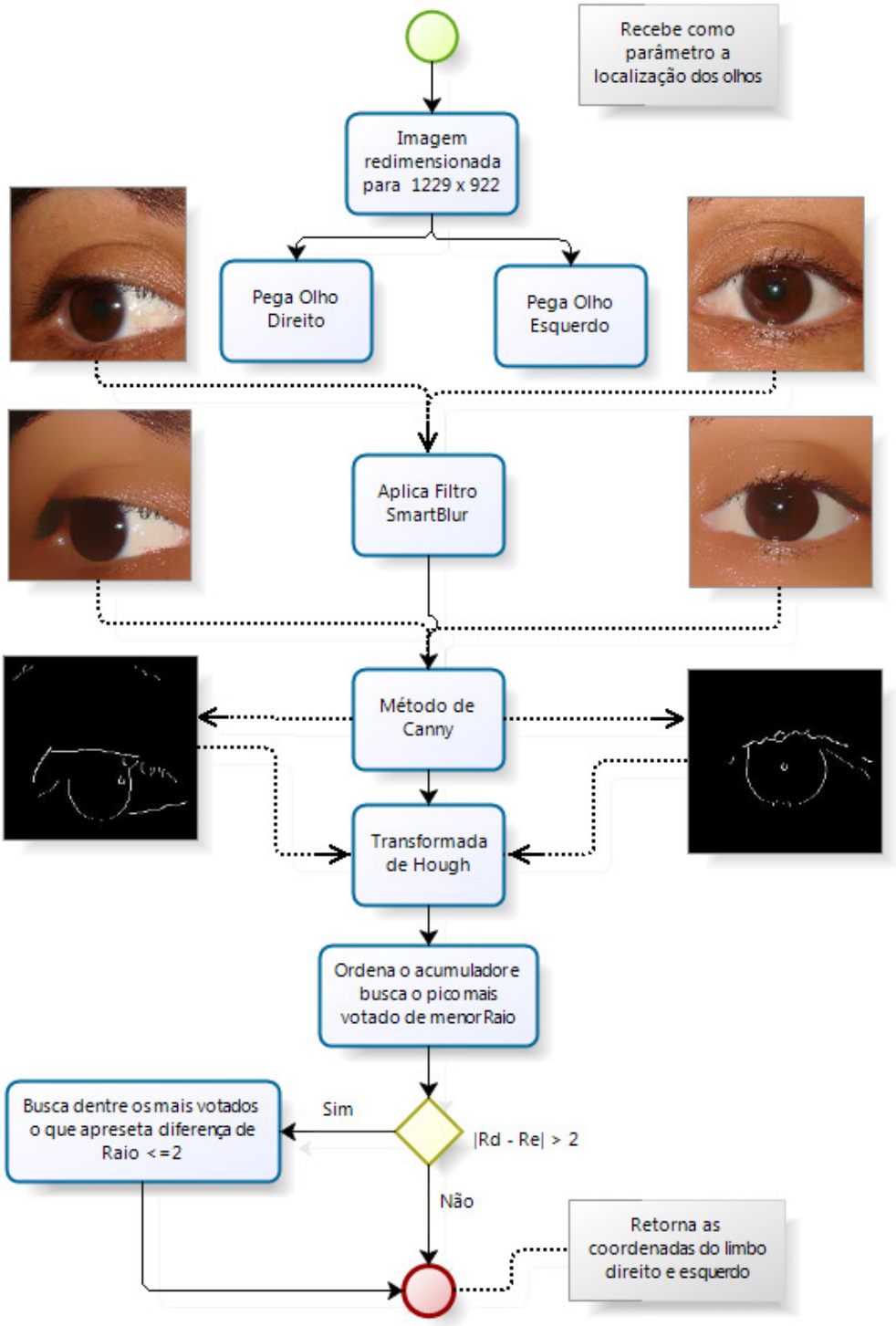
Quadro 1 - Parâmetros do Canny, Hough e Smart Blur, os mesmos utilizados por Almeida (2013)

Canny	**TH – Limbo**	**Smart Blur**
Limite inferior:100	Raio menor:22	Raio horizontal: 50
Limite superior: 136	Raio maior:55	Raio Vertical: 100
Fator de derivação: 1,2	Quantidade de coordenadas: 80	Limiar: 30
Máscarca Gaussiana: 5x5		

A aplicação da TH seguiu a abordagem de Almeida (2013), ou seja, foram considerados os pontos nos intervalos de 0º a 60º, 300º a 360º correspondendo a uma abertura de 120º do lado direito do círculo desenhado nos pontos de bordas e 120º a 240º correspondendo a uma abertura de 120º no lado esquerdo. Os pontos, fora destes intervalos, foram excluídos do vetor de acumulação. Com a realização desse procedimento é excluída a influência das pálpebras na localização do limbo. Os valores de intervalos dos raios e

acumulador foram estabelecidos de modo empírico. A Figura 20 mostra o fluxograma desta etapa.

Figura 20– Fluxograma da etapa de detecção do limbo.



Fonte: ALMEIDA, 2013.

Após a detecção dos possíveis círculos, são selecionadas as 80 coordenadas mais votadas e são ordenadas em ordem decrescente de acordo com os valores no vetor de

acumulação. Depois, é verificado se existem outras coordenadas com a mesma quantidade de votos e com raio menor que o do topo do vetor. Caso existam, considera-se a que tiver menor raio. Com isso, busca-se garantir a escolha de uma região circular. Este processo é realizado para os limbos direito e esquerdo.

Já determinados os dois principais candidatos a limbo, os raios de ambos devem ser próximos. Toma-se o menor deles e verifica-se se a diferença entre eles é menor ou igual a 2 *pixels*. Tomando o raio do olho direito o menor, caso o outro ultrapasse a diferença, seleciona-se o candidato mais votado no seu respectivo vetor que satisfaça o limiar de diferença entre os raios.

3.7 Localização do Brilho

Esta etapa é a última que utiliza técnicas de processamento de imagens e também pode ser realizada de forma automática ou manual. Ambas são apresentadas nas seções a seguir.

3.7.1 Abordagem Automática

Para localizar o brilho, utiliza-se a região do limbo localizada na seção anterior. Nesta, aplica-se um pré-processamento realizando a conversão para nível de cinza e aplicação do filtro da mediana (GONZALEZ; WOODS, 2002). Em seguida, realiza-se a detecção das bordas por meio do método de Canny. Por último, aplica-se a transformada de Hough (TH) para se ter o brilho detectado.

Nesta etapa, os parâmetros do Canny e da TH são os mesmos da etapa anterior, exceto o intervalo de raios do brilho: 1 *pixel* para o raio menor e 4 *pixels* para o maior. Selecionam-se novamente as 80 coordenadas mais votadas e ordena-se decrescentemente. Depois verifica-se, dentre os seis maiores picos no vetor de acumulação, qual deles está mais próximo do centro do limbo, considerando as coordenadas (x,y). Com isso, garante-se que a localização do brilho não caia na região das pálpebras, quando os olhos estiverem entreabertos e nem que seja confundido com reflexos gerados por lentes corretivas.

Caso algum dos brilhos não sejam detectados ou sejam detectados incorretamente, pode-se utilizar a abordagem manual nesta etapa.

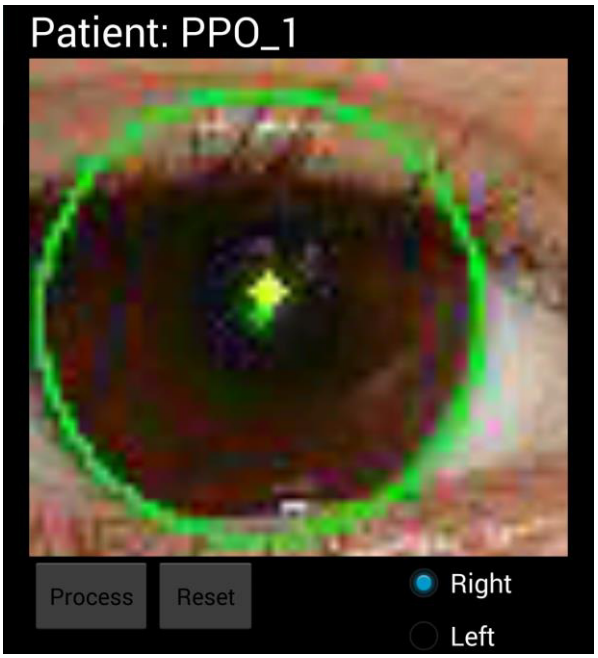
3.7.2 Abordagem Manual

Após a etapa de localização do limbo é executada a abordagem automática da detecção do brilho. Mas, nesta última, pode haver erros na detecção. Estes erros podem ser

ocasionados tanto pelo fato das bordas não terem sido bem detectadas, quanto pelo fato de haver outros pontos luminosos no limbo, decorrentes de reflexos de luz no ambiente ou ainda de reflexo de lentes corretivas. Pode ocorrer ainda do diâmetro do brilho ser maior do que o esperado, ocasionado muitas vezes por imagens não tão bem focadas

Nestes casos, o usuário pode interagir com a região da imagem, selecionando o local do brilho com um pressionamento longo (tempo de no mínimo dois segundos) sobre o local onde se encontra o brilho. Depois de selecionado, o brilho será marcado com um círculo amarelo. Como o brilho é muito pequeno, seria muito difícil acertá-lo com precisão. Por isso, o zoom máximo da janela foi ajustado para 128x, facilitando assim a seleção do local. O usuário seleciona o lado do olho e depois o respectivo brilho, podendo fazer isto para ambos ou apenas para um deles. Após isto, o usuário seleciona a opção de processar e então o diagnóstico será recalculado.

Figura 21– Brilho direito sendo selecionado em uma imagem com zoom de aproximadamente 100x



3.8 Detecção do Estrabismo

A detecção do estrabismo é feita utilizando-se a localização do reflexo luminoso corneano, ou primeira imagem de Purkinje, gerada pelo teste de Hirschberg (seção 2.2.1) e a localização do limbo como parâmetros de verificação do alinhamento de ambos os olhos. Esta etapa tem o objetivo de avaliar a medida do desvio.

A avaliação do posicionamento da primeira imagem de Purkinje é feita através dos 5 passos a seguir:

1. Mede-se a distância, em *pixels,* entre o centro de cada limbo e centro de seu respectivo brilho, nas orientações vertical e horizontal (Desvio Vertical em pixels - DVp e Desvio Horizontal em pixels - DHp;
2. Faz DVp igual à diferença entre os desvios verticais de ambos os olhos, com a finalidade de excluir erros gerados pelo posicionamento da câmera no momento da aquisição;
3. Calculam-se os desvios usando as equações:

$$DH_{\Delta} = DH_p \; x \; pixelMM \; x \; dpMM \quad (21)$$

$$DV_{\Delta} = DV_p \; x \; pixelMM \; x \; dpMM \quad (22)$$

Onde:

- dpMM corresponde ao valor de dioptrias para cada unidade de milímetro,.
- pixelMM corresponde ao valor de *pixel* para cada unidade de milímetro.
- DH_p e DV_p correspondem aos valores dos desvios horizontal e vertical em *pixels*, respectivamente.
- DH_{Δ} e DV_{Δ} correspondem aos valores dos desvios horizontal e vertical em dioptrias prismáticas, respectivamente.

Considerou-se que o diâmetro do limbo de uma pessoa equivale a 11mm e que cada milímetro é equivalente a 15Δ, valores em que Almeida (2013) obteve os melhores resultados. Com a medida do desvio é possível definir se um paciente necessita de tratamento verificando se possui desvio maior ou igual a 10Δ.

A Figura 22 exemplifica o cálculo DV e DH em um paciente com desvios horizontal e vertical. Na figura, percebe-se que o centro do limbo não está alinhado em relação ao brilho em nenhuma das orientações.

Figura 22- Cálculo do alinhamento

3.9 Diagnóstico do Estrabismo

O objetivo desta etapa é identificar o olho fixador e o tipo de desvio. Define-se o olho fixador da seguinte maneira: considerando o olho no qual o brilho esteja mais próximo do centro do limbo, o olho fixador é o olho usado para prestar atenção. Se o outro se desvia é considerado olho desviado, que é usado para fazer o diagnóstico do desvio monocular. Se um olho é considerado sempre fixador, este olho pode ser chamado de olho dominante (SBO, 2012).

Para aferir qual os olhos é o olho fixador, matematicamente, foi calculado a magnitude do desvio entre o centro do limbo de cada olho e o centro de seus respectivos brilhos, através da equação:

$$desvMag = \sqrt{(BO.x - LO.x)^2 + (BO.y - LO.y)^2} \quad (23)$$

onde desvMag é o valor da magnitude do desvio entre o centro do limbo de cada olho e seu respectivo brilho, BO.x e BO.y é a coordenada (x,y) do brilho e LO.x e LO.y é a coordenada (x,y) do centro do limbo. O olho que apresentar a menor magnitude é considerado o olho fixador.

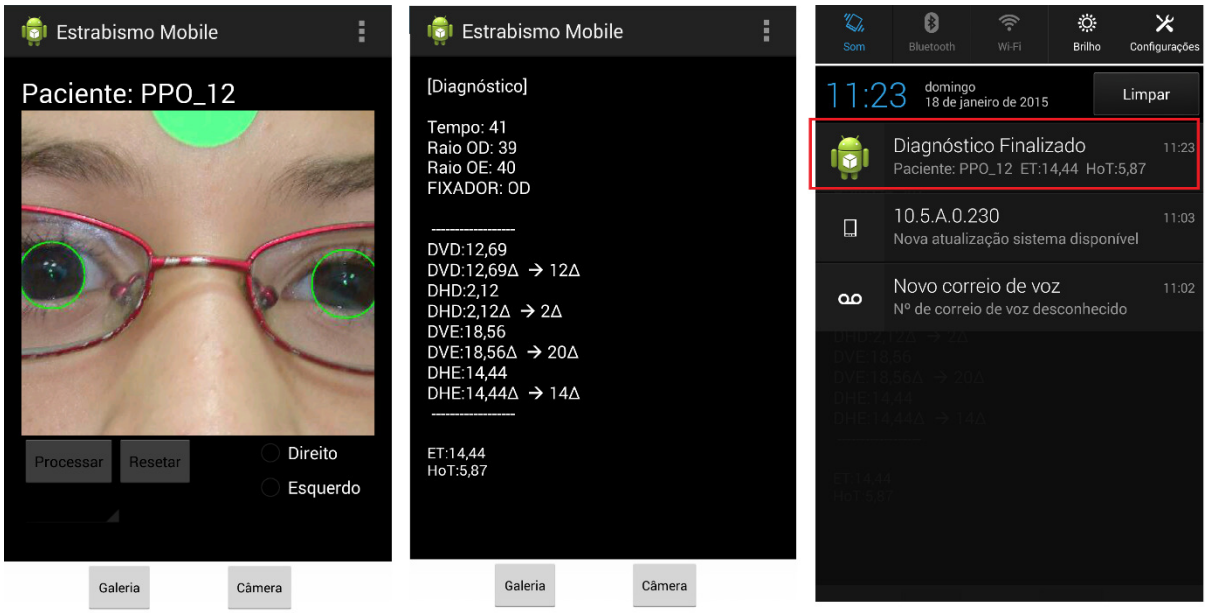
O próximo passo é identificar o tipo do estrabismo no olho não fixador. O estrabismo pode ser horizontal e vertical, cada qual com dois subtipos e a não. Na ordem apresentada, os subtipos definem-se como:

- Convergente ou esotropia (ET): Acontece quando um olho fixa na imagem e o outro fixa para dentro. Neste caso, a coordenada horizontal (eixo x) do brilho é menor que a do centro do limbo;
- Divergente ou exotropia (XT): Acontece quando um olho fixa na imagem e o outro fixa para fora. Neste caso, a coordenada horizontal (eixo x) do brilho é maior que a do centro do limbo;
- Hipertropia (HT): acontece quando um dos olhos fixa para cima, ou seja, a coordenada vertical do brilho é maior do que a do centro do limbo;
- Hipotropia (HoT): acontece quando um dos olhos fixa para baixo, ou seja, a coordenada vertical do brilho é menor do que a do centro do limbo.

Existe também o diagnóstico ORTO, no qual significa que o paciente não possui desvio. Vale ressaltar que em todos os casos em que o paciente apresenta algum desvio, cada um dos olhos focaliza imagens diferentes e a pessoa tem o que se chama de diplopia, ou visão dupla.

Ao finalizar as etapas de execução do aplicativo, o diagnóstico é gerado e uma notificação é lançada na barra de notificações do sistema. Essa notificação contém o nome do arquivo e o tipo de estrabismo vertical e horizontal que o paciente possui. Este recurso é útil pois no meio do diagnóstico e o usuário pode optar por sair do aplicativo enquanto realiza uma ou outra tarefa. Ao receber a notificação, basta que o usuário toque nela e ela retornará a tela do aplicativo. Outra utilidade é que o usuário pode visualizar o diagnóstico do paciente podendo estar executando outros aplicativos. As Figuras 23 (a) e (b) mostram a tela de diagnóstico e a notificação lançada para o usuário.

Figura 23– (a) Tela mostrando a imagem resultado. (b) Tela de diagnóstico do aplicativo. (c) Notificação do aplicativo com diagnóstico na barra de notificação do sistema (demarcado de vermelho)



(a) (b) (c)

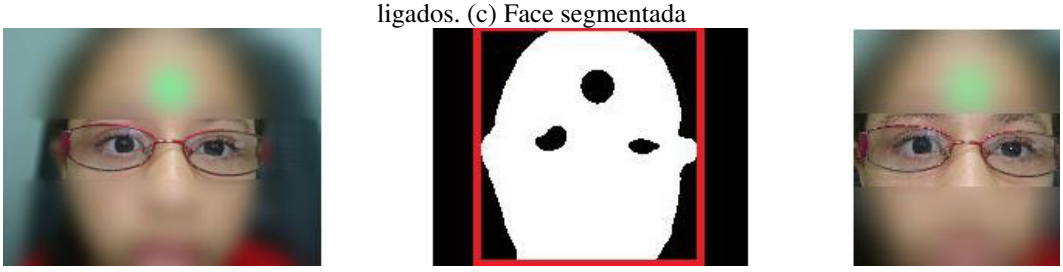
4 RESULTADOS E DISCUSSÕES

Esta seção apresenta e discute os resultados da metodologia portabilizada para dispositivos móveis baseados em Android com o objetivo de diagnosticar o estrabismo por meio de imagens digitais. São discutidos os resultados observados considerando o conjunto de imagens e pacientes em todas as etapas de execução do aplicativo. Para comparações e menções a Almeida (2013), iremos nos restringir às imagens em que os pacientes estão em PPO em todas as etapas de execução, devido a esta ser a posição de interesse para este trabalho.

4.1 Segmentação da Face

Nesta fase (seção 3.3), das 45 imagens de pacientes, obteve-se taxa de acerto de 93,34%, considerando-se como erro casos em que foram delimitadas áreas muito maior do que a da face. No entanto, como esta etapa objetiva apenas diminuir o espaço de busca, casos de falha não comprometem a continuidade da execução do aplicativo, visto que a região dos olhos ainda estará presente na imagem. As Figuras 24(a), 24(b) e 24(c) mostram a imagem original, etapa de delimitação da face na etapa de limiarização e a face segmentada respectivamente. Já as Figuras 25(a) e 25(b) mostram o caso em que a face não foi delimitada corretamente, devido a técnica não ter conseguido diferenciar corretamente a cor da camisa do paciente da cor da pele. Para esta etapa, Almeida (2013) obteve 88% para 200 imagens nas 5 posições do olhar.

Figura 24– Face segmentada com sucesso. (a) Imagem original. (b) Face limiarizada com pontos extremos ligados. (c) Face segmentada



(a) (b) (c)

Figura 25– Imagem em que a segmentação da face falhou. (a) Imagem original, (b) Limiarização

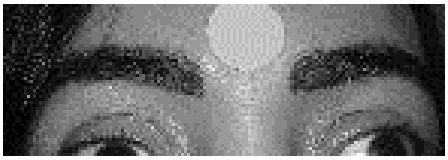
(a) (b)

4.2 Detecção da Região dos Olhos

Na execução desta etapa, apenas uma imagem das 45 imagens não obteve o resultado esperado, correspondendo a 97,78% de taxa de acerto. Neste caso, a região dos olhos foi marcada com as sobrancelhas centralizadas, excluindo parte dos olhos. A Figura 26(a) mostra um caso em que se obteve o sucesso esperado, e a Figura 26(b) mostra o único caso em que não se obteve sucesso.

Figura 26– (a) Região dos olhos segmentada com sucesso. (b) Região dos olhos segmentada com falha

(a)



(b)

Para esta etapa, Almeida (2013) obteve taxa de acerto de 98,5% para 200 imagens de pacientes nas 5 posições do olhar e de 97,5% para 40 imagens em PPO.

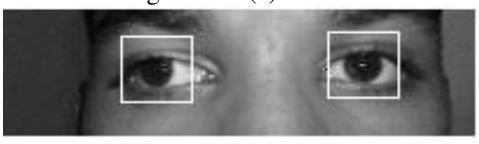
4.3 Localização dos Olhos

Com a redução do espaço de busca alcançada através das segmentações realizadas nas etapas anteriores, segue-se para a etapa de localização dos olhos (seção 3.5).

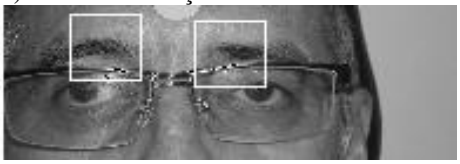
Conforme descrito na seção 3.5.1.1, foi utilizado o mesmo classificador treinado por Almeida (2013). De acordo com Almeida (2013), para 40 imagens de pacientes em PPO, a acurácia do classificador foi de 98,57% para a configuração (treino/teste) de 60%/40% amostras, obtendo ainda taxas de 100% de sensibilidade e 96,20% de especificidade. O desvio padrão (σ) da taxa de acerto, para as configurações treino/teste, foi de 0,88, ou seja, houve pouca variância.

Das 45 imagens utilizadas neste trabalho, apenas uma não obteve o resultado esperado, ou seja, 97,78% de sucesso. São contabilizados como caso de falha as imagens em que o limbo não está dentro da região detectada. A Figura 27(a) mostra um caso em que houve sucesso e a Figura 27(b) o único em que houve falha. Tal falha ocorreu em ambos os olhos.

Figura 27– (a) Olhos detectados com sucesso. (b) Falha na detecção dos olhos.



(a)



(b)

4.4 Localização do Limbo e do Brilho

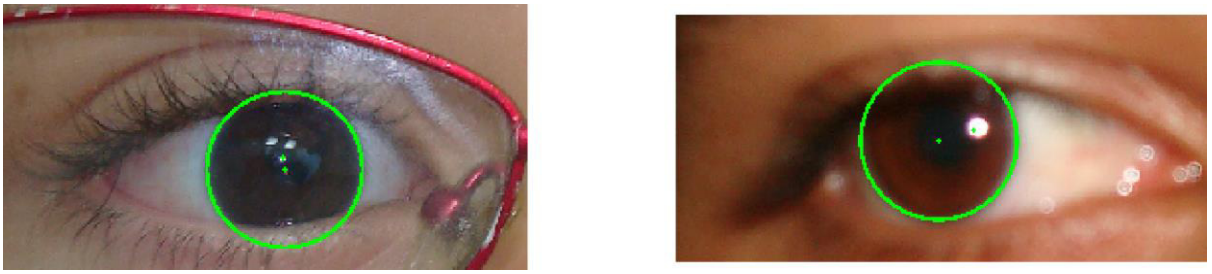
A precisão do diagnóstico depende diretamente do acerto preciso do limbo e do brilho, uma vez que se utilizam as localizações destes para o cálculo do alinhamento (seção 3.8). Nesta etapa, as imagens que não passaram pelas etapas anteriores também serão consideradas, visto que a abordagem manual servirá para estes casos.

Em relação a localização do limbo, ocorreu estouro de pilha de memória no processamento de 5 imagens, não simultaneamente. Isso foi ocasionado pela recursão profunda (ORACLE, 2014) que ocorreu nessas imagens na execução do método de Canny, em situações em que pontos de bordas tinham muitos vizinhos interligados. Nestes casos foram utilizados a abordagem manual de seleção da localização dos olhos, afim de evitar selecionar regiões que aparentam ter muitos detalhes, como a parte superior de lentes corretivas. Tais imagens foram consideradas em etapas mediante sucesso na utilização da abordagem manual e, incluindo estas, a metodologia portada atingiu taxa de acerto de 83,33% utilizando a abordagem manual e 85,55% utilizando a automática. Já para a detecção do brilho, a taxa de acerto foi de 100% para ambas.

Segundo estudos de Choi e Kushner (1998), especialistas experientes estimaram desvios com erro médio de 5 a 10Δ. Por isso, foi considerado erro na localização do limbo os casos em que este não foi demarcado corretamente de forma muito perceptível, a ponto de interferir no resultado do diagnóstico. Para o brilho foi considerado como erro marcações fora de sua região, visto que este é muito pequeno.

Em algumas imagens, apesar do brilho ter sido detectado, mesmo em imagens borradas, o brilho gerado foi maior do que o esperado pela metodologia. Nestes casos, a detecção não marcou o brilho em seu centro. Porém, a abordagem manual pode ser usada nesta situação, caso o usuário queira verificar se houve diferença significativa no resultado em decorrência disto. As Figuras 28 (a) e (b) mostram casos em que o limbo e o brilho foram demarcados precisamente e outro em que o brilho foi demarcado pouco fora do centro.

Figura 28– (a) Limbo e brilho demarcados precisamente. (b) Limbo demarcado precisamente e brilho não.



(a) (b)

A Figura 29 (a) mostra um caso em que o aplicativo não foi 100% preciso, porém interferiu pouco no resultado, ou seja, dentro do erro médio. O especialista diagnosticou 10Δ de DH e 0Δ de DV, enquanto o aplicativo diagnosticou 6,11Δ e 3,06Δ respectivamente. A Figura 29 (b) mostra um caso de erro causado pelo ofuscamento de parte do limbo causado pelo *flash* gerado na lente corretiva do paciente.

Figura 29– (a) Limbo não detectado precisamente, mas que não interferiu muito no diagnóstico. (b) Detecção do limbo com falha ocasionada pelo ofuscamento de parte do limbo

(a)

(b)

4.5 Detecção de Estrabismo

Esta seção apresenta os resultados obtidos pelo aplicativo na detecção de estrabismo para os pacientes em PPO, podendo assim, até esta etapa, ser utilizado na realização de triagem de pacientes. Para as imagens atingirem esta etapa, é necessário que o limbo e o reflexo luminoso gerado pelo teste de Hirschberg tenham sido detectados precisamente nas etapas anteriores. Foram consideradas 38 das 45 imagens de pacientes, ou seja, 84,44% das imagens, a partir desta etapa. As imagens que foram excluídas desta fase foram as que estavam sem foco, sem diagnóstico do especialista ou ainda que o limbo não foi bem detectado.

Analisando a detecção para as imagens consideradas, foram obtidos VP=32, FP=0, VN=5 e FN=1. A Tabela 1, comentada na seção 4.7, apresenta a matriz de decisão que confronta as respostas do aplicativo com as respostas do especialista. A partir dos dados da matriz, verifica-se que a metodologia alcançou 96,96% de sensibilidade, 100% de especificidade e 97,36% de acerto.

Tabela 1 – Matriz de decisão para comparar os resultados do aplicativo com as respostas do especialista.

Resultados	**Estrabismo**	
	Positivo	Negativo
Positivo	32	0
Negativo	1	5

4.6 Diagnóstico do Estrabismo

Esta seção apresenta os resultados obtidos no diagnóstico do estrabismo. A acurácia desta metodologia foi avaliada através de comparações com o diagnóstico apresentado por um especialista. Considerando o erro médio já mencionado na seção 4.4 e devido à subjetividade do teste de Hirschberg, utilizou-se a tolerância de erro de 10Δ para avaliar o valor estimado pelo método em relação à medida do especialista.

Para os tipos de desvios o aplicativo obteve acurácia, em 35 imagens seguindo apenas a abordagem automática, de 93,33% na identificação das esotropias (ET), 100% nas exotropias (XT), 71,43% nas hipertropias (HT), 66,67% nas hipotropias (HoT) e 10% na identificação da ausência de desvios (ORTO). A acurácia na classificação do olho fixador foi de 89,47% e, na aferição dos desvios, atingiu acurácia de 79,48% para o desvio horizontal (DH) e 87,17% para o vertical (DV). Para 38 imagens, utilizando as abordagens manuais quando necessário, foi atingido acurácia 93,75% na identificação das esotropias (ET), 100% nas exotropias (XT), 71,43% nas hipertropias (HT), 66,67% nas hipotropias (HoT) e 8,32% na identificação da ausência de desvios (ORTO). A acurácia na classificação do olho fixador foi de 92,10% e, na aferição dos desvios, atingiu acurácia de 89,74% para o desvio horizontal (DH) e 92,30% para o vertical (DV). A Tabela 2 mostra os resultados do aplicativo, Almeida (2013) e o especialista das aferições dos desvios verticais e horizontais para as imagens de pacientes. São demarcados os pacientes em que se utilizou as abordagens manuais. A Tabela 3 mostra as taxas de acerto do aplicativo e Almeida (2013), em relação ao diagnóstico de estrabismo feito pelo especialista.

O aplicativo obteve baixa acurácia para pacientes sem desvio. Segundo Almeida (2013), a baixa acurácia em diagnóstico de pacientes que não apresentam desvios verticais e/ou horizontais pode ser explicada pelo grau de precisão da metodologia utilizada em relação à avaliação realizada pelo especialista, que normalmente afere desvios em múltiplos de 5Δ desconsiderando desvios pequenos, diagnosticando o paciente como ORTO. Além disso, o método de Hirschberg raramente consegue trabalhar com precisão de 1Δ.

Uma forma encontrada para melhorar a precisão do limbo e, por conseguinte o diagnóstico de estrabismo foi dar a opção de o usuário fazer seleção dentre os círculos obtidos com a TH. No entanto, isto foi feito apenas como teste (não foi incluído nos resultados) e necessita de melhorias na usabilidade. A Figura 30 (a) mostra um caso em que o aplicativo errou no diagnóstico. Segundo o especialista, o paciente possui desvio vertical de 0Δ e o aplicativo aferiu 30Δ. Este erro foi ocasionado devido a imprecisão na detecção do limbo,

onde foi detectado um círculo menor que o limbo do paciente, fazendo o centro se localizar abaixo do que tinha que realmente se localizar e gerando assim um desvio vertical. Já a Figura 30 (b) mostra a lista de círculos no Hough, onde os dois primeiros parâmetros são as coordenada (x,y) do centro, o terceiro é o raio e o quarto o número de votos e o resultado da escolha. A escolha dentre os círculos foi baseada no tamanho do raio, pois o raio do limbo detectado erroneamente tinha 27 *pixels*. Então foi escolhido o círculo mais votado com um raio um pouco maior, ou seja, de 30 pixels. O novo resultado foi de 1,96Δ de DV, assim, seria considerado como acerto.

Tabela 2 – Melhores resultados das aferições dos desvios vertical (DV) e horizontal (DH) entre o aplicativo, Almeida (2013) e o especialista. Nos pacientes demarcados de vermelho foi utilizado a abordagem manual do brilho, nos de azul a abordagem manual do limbo, e nos de lilás, ambas.

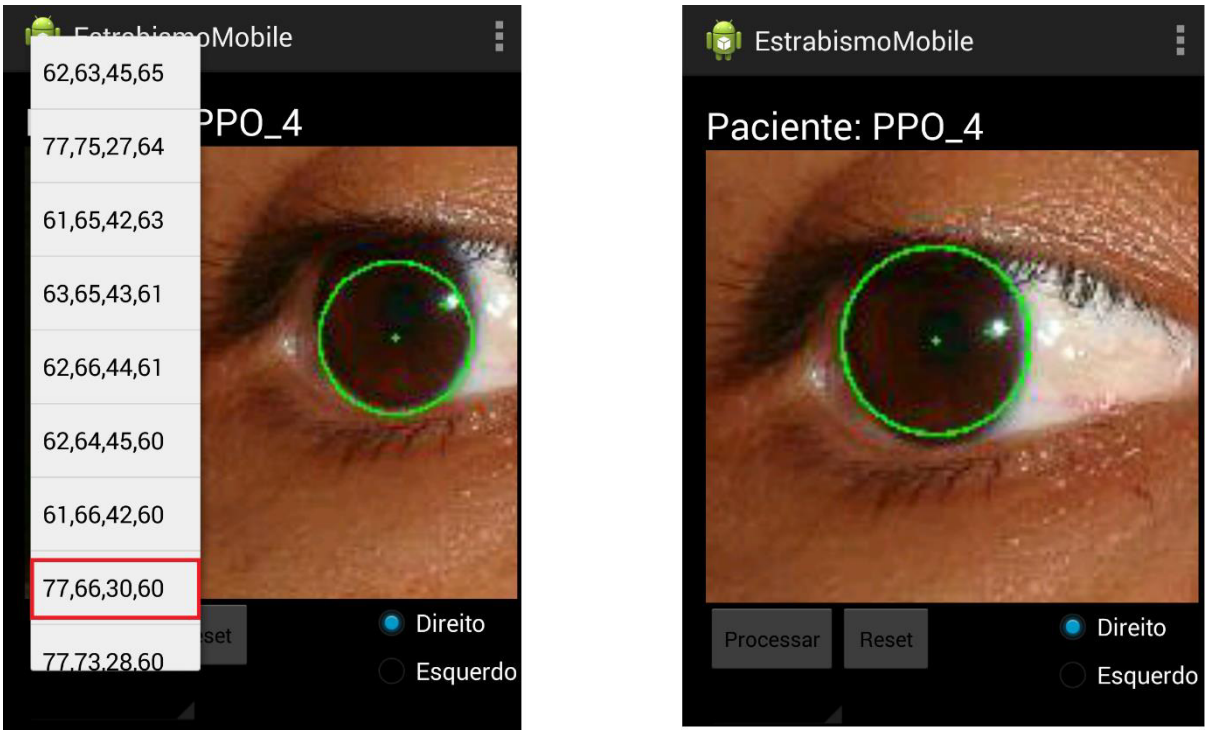
Trabalhos	Aplicativo	Almeida 2013	Espec.	Aplicativo	Almeida 2013	Espec.
PAC.	**DH**	**DH**	**DH**	**DV**	**DV**	**DV**
1	6,11	12,22	10	3,06	3,06	0
2	9,82	7,86	0	1,96	3,93	0
3	27,5	27,5	25	8,42	5,84	0
4	55	55	50	29,98	6,54	0
6	44	40,33	40	0,62	6,63	0
7	26,81	22,69	30	5,84	4,12	15
8	39,72	36,67	55	10,08	0,61	15
10	24,44	21,39	20	18,33	18,33	15
11	48,36	46,75	55	20,53	16,89	10
14	27,5	26,76	25	0,24	0,23	0
15	41,25	42,78	45	3,14	0,63	0
16	30,25	30,25	30	5,67	4,81	0
17	25,21	26,69	35	2,29	2,29	0
18	18,05	15,47	30	0,16	0,15	0
21	24,94	23,02	30	0,09	0,17	0
23	36,67	36,67	45	11,7	13,87	0
25	11,46	10	20	4,58	4,58	0
26	16,99	16,99	25	0,13	0	0
27	22,5	21,84	20	2,79	2,43	6
28	44	45,24	50	1,2	4,74	0
29	12	11,11	15	0	0	0
30	10,76	10,76	20	5,86	3,59	10
31	20,07	17,84	35	6,69	4,46	0
32	20,12	18,11	30	2,39	0,47	5
33	11,79	12,07	20	1,96	3,74	0
37	16,04	13,75	20	0	2,29	0
38	13,75	10,69	20	1,53	1,53	4
39	31,22	31,22	35	0	2,17	0
40	16,92	14,44	20	6,19	4,13	10

41	6,69	8,68	0	2,23	2,29	0
42	6,69	6,69	0	2,23	4,46	0
43	6,69	8,92	0	2,23	0	0
44	7,33	5,5	0	3,67	1,83	0
5	51,56	-	50	6,38	-	0
12	14,44	-	15	5,87	-	0
19	33	-	50	7,33	-	0
22	28,99	-	35	11,15	-	10
34	16,5	-	15	2,17	-	0

Tabela 3 – Resultado do diagnóstico na identificação dos desvios (ET – esotropia, XT – exotropia, HT – hipertropia, HoT – hipotropia e ORTO – sem desvio) e olho fixador de Almeida (2013) e as abordagens propostas (automática e manual)

		%					
Diagnóstico	Nº de Imagens	ET	XT	HT	HoT	ORTO	Fixador
Almeida (2013)	33	92,85	100	66,67	66,67	6,25	92,30
Abordagem Automática	35	93,33	100	71,43	66,67	10	88,57
Abordagem Manual	38	93,75	100	71,43	66,67	8,32	92,10

Figura 30– (a) Limbo detectado com erro e lista gerada pelo Hough. (b) Novo resultado na detecção



(a) (b)

4.7 Discussão dos Resultados

As Classes Java utilizadas neste trabalho para manipulação de imagens não foram as mesmas usadas por Almeida (2013), visto que muitas são incompatíveis com o Android SDK. Por exemplo, Classes como *BufferedImage*, *PlanarImage* e bibliotecas como JAI foram substituídas por recursos existentes na própria API do Android, como a Classe *Bitmap*, que trabalha de forma semelhante a estas. Outro exemplo é o filtro SmartBlur (JHLABS, 2014) que, em Java *desktop* foi implementado utilizando a Classe *BufferedImage*. Já a versão do filtro SmartBlur (MORDOMENEZ-ME, 2014) disponibilizada para Android foi totalmente reescrito utilizando a Classe *Bitmap*. Apesar da semelhança nos resultados, podemos perceber diferenças de contraste e detalhes que podem interferir na detecção de bordas. As Figuras 31 e 32 mostram os resultados após aplicação do filtro SmartBlur (MORDOMENEZ-ME, 2014) para Android e o filtro SmartBlur (JHLABS, 2014) para Java *desktop*, no olho direito do paciente 1. As imagens foram ampliadas para melhor visualização.

Vejamos agora algumas diferenças encontradas na etapa de detecção de bordas para as Figuras 31 e 32, mostradas na Figura 33 (a) e (b). Os limiares foram os mesmos utilizados em todas as etapas e a implementação do Canny foi a mesma, exceto pelas classes que trabalham com imagens que foram substituídas. As imagens foram ampliadas para melhor visualização.

Essas diferenças, mesmo que não muito grandes, interferem na detecção de círculos. Isto pode justificar a pequena diferença entre os resultados deste trabalho com Almeida (2013). A Figura 34 (a) e (b) mostra a diferença entre ambos na detecção do limbo e do brilho e as Tabelas 2 e 3 comprovam a diferença nas aferições dos desvios e diagnóstico.

Figura 31– SmartBlur aplicado em ambiente Android. Imagem com mais contraste, nitidez e detalhes de cílios e brilho. Porém, com menos volume no brilho gerado no centro do limbo.

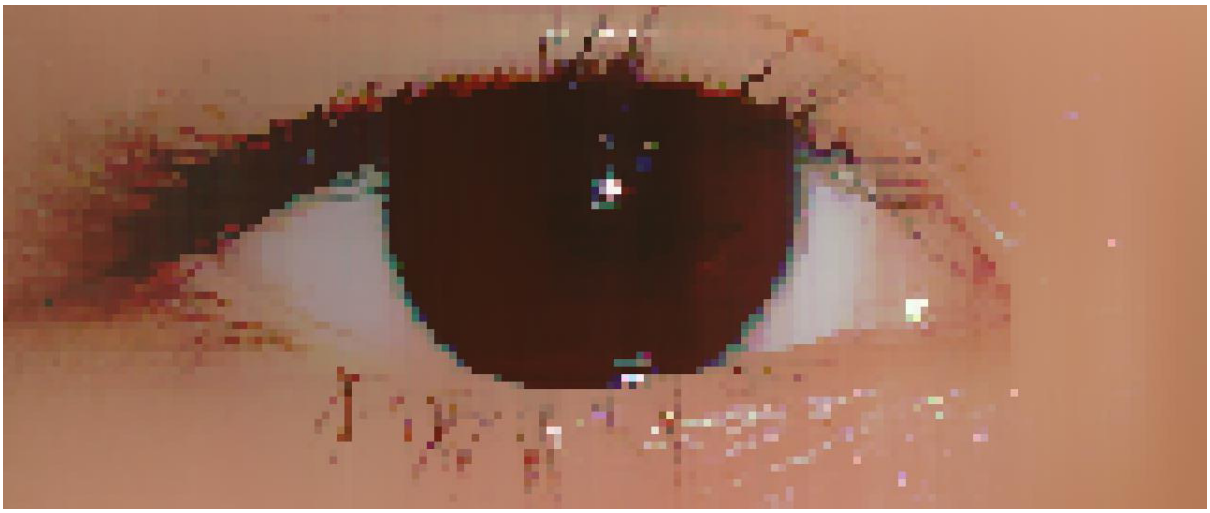


Figura 32- SmartBlur aplicado em Java *desktop*. Imagem com menos contraste, nitidez e menos detalhes de brilho e cílios. Porém, o reflexo luminoso no centro do olho tem mais volume

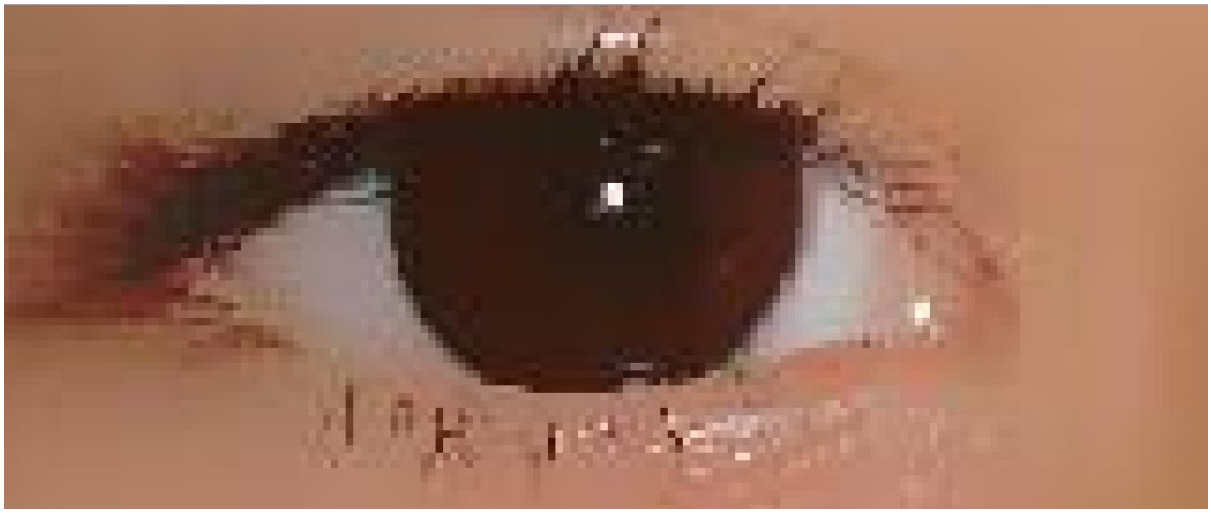
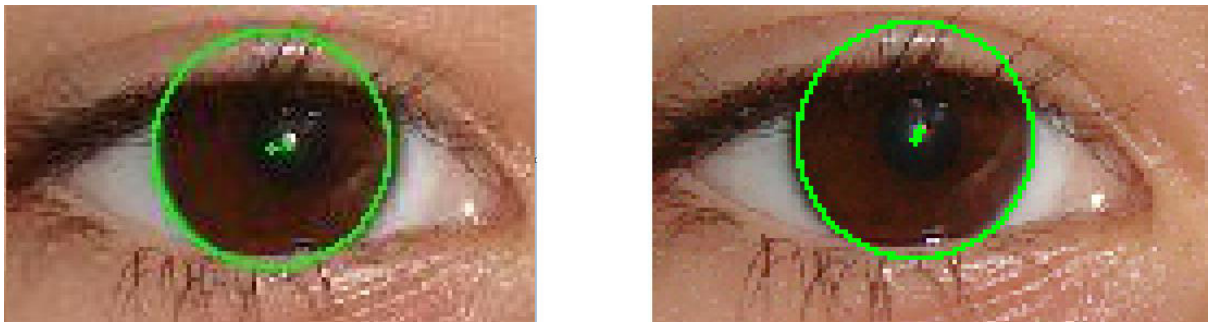


Figura 33– (a) Resultado da implementação da metodologia de Almeida (2013) em Java *desktop*. (b) Resultado no dispositivo móvel

(a) (b)

Figura 34– (a) Limbo detectado mais à esquerda na implementação para Java *desktop*. (b) Limbo detectado mais à direita na implementação para Android



Em relação a detecção de estrabismo, um dos motivos de ambas as abordagens obterem resultados iguais, conforme mostrado na Tabela 1, é devido ao baixo limiar para se considerar um paciente estrábico - maior que 10Δ - descrito na seção 3.8. Tomando por exemplo o paciente 11, o especialista aferiu DH de 55Δ e o aplicativo aferiu 48,36Δ. Caso o aplicativo aferisse, por exemplo, 25Δ, este ainda seria contabilizado como acerto em relação ao paciente ser ou não estrábico.

Outro motivo configura-se no baixo erro médio para aferição de DHs e DVs na abordagem automática, que foram de 6,85 e 5,12, respectivamente. Tais valores foram menores do que a tolerância de erro - 10Δ (seção 4.6). No entanto, um caso em que poderia ocorrer erro na detecção de estrabismo seria se o paciente 12, que foi diagnosticado com 15Δ pelo especialista, fosse diagnosticado pelo aplicativo com 9Δ, tendo por resultado um paciente não estrábico quando este seria estrábico.

Estes motivos também são válidos para a igualdade entre as abordagens em relação ao diagnóstico do tipo do desvio. Caso um paciente fosse diagnosticado com sucesso através da abordagem automática com 15Δ de HT e pela manual com 45Δ de HT, ainda seria considerado acerto em relação ao tipo de desvio - HT.

Em relação a diferença no diagnóstico de olho fixador, entre as abordagens adotadas, temos 1 caso particular. O paciente 1 que, relativo a abordagem automática de localização do brilho, obteve resultados iguais para aferição dos DHs e DVs de ambos os olhos. Fazendo o cálculo para detecção do olho fixador (seção 3.9) temos como resultado que ambos os olhos são fixadores, visto que não há diferencial na magnitude do desvio destes. Tomando os valores dos desvios, em Δ, após a utilização da abordagem manual do brilho, houve um diferencial de magnitude e, por isto, o erro quanto ao diagnóstico do olho fixador deixou de existir. A Tabela 4 mostra, com números, o ocorrido.

Tabela 4 – Caso particular em relação a diferença no diagnóstico do olho fixador entre as abordagens manual e automática da detecção do brilho, onde DVD é o desvio vertical do olho direito, DHD o desvio horizontal do olho esquerdo, magDesvOD a magnitude do desvio do olho direito e as demais variáveis tem os mesmo significados, mas relacionados ao olho esquerdo

	Δ						
	DVD	DHD	magDesvOD (seção 4.8)	DVE	DHE	maDesvOE (seção 4.8)	FIXADOR
Abordagem Automática	6,11	3,06	6,53	3,06	6,11	6,53	AMBOS(erro)
Abordagem Manual	9,17	3,06	9,66	6,11	6,11	8,64	OE (acerto)

Por fim, a Tabela 5 mostra os resultados das etapas de segmentação da face à localização dos olhos da metodologia proposta por Almeida (2013) em Java *desktop* e Android, com suas respectivas quantidades de imagens. Já a Tabela 6 mostra os resultados da localização do limbo e do brilho em ambos os ambientes e em ambas as abordagens propostas no ambiente móvel, também com suas respectivas quantidades de imagens.

Tabela 5 – Taxa de acerto da metodologia proposta por Almeida (2013) em Java d*esktop* e móvel (Android)

	%		
Etapas	Segmentação da Face/Imagens	Detecção da Região dos Olhos/Imagens PPO	Localização dos Olhos/Imagens PPO
Desktop	88% - 200	97,50% - 40	98,57% - 40
Móvel	93,34% - 45	97,78% - 45	97,7% - 45

Tabela 6 **-** Taxa de acerto da metodologia proposta por Almeida (2013) em Java d*esktop* e móvel (Android) nas duas abordagens.

		%	
Etapas	Nº de Imagens	Localização do Limbo	Localização do Brilho
Almeida (2013)	40	94,80	97,40
Abordagem Automática	39	83,33	100
Abordagem Manual	45	85,55	100

4.8 Considerações Finais

O aplicativo será utilizado por um especialista para testes de usabilidade, com o objetivo de obter-se uma avaliação do mesmo. Conforme a necessidade do especialista, farar-se-á mudanças na interface com a finalidade de melhorar a interação do usuário.

Inicialmente, será fornecido um arquivo .apk para que o especialista possa instalar o aplicativo em seu dispositivo móvel. Após testes de usabilidade e estudos feitos para comprovar a viabilidade do aplicativo, pretende-se disponibilizar o aplicativo para *download* na Play Store, loja virtual de aplicativos da Google.

5 CONCLUSÃO

O presente trabalho apresentou um aplicativo para detecção e diagnóstico de estrabismo se fazendo valer da metodologia proposta por Almeida (2013) Java *desktop*. Com a mudança de ambientes, algumas tecnologias tiveram que ter sido substituídas, e mesmo tais trabalhando de forma semelhante, foi obtido pequenas variações no resultados. Com a ampliação da base de dados e o aumento da qualidade dos resultados obtidos a partir deste trabalho e melhorias na interface com o usuário, será possível utilizar o presente aplicativo como uma ferramenta para a área médica.

Para aumentar a confiança nos resultados, há de se ampliar a base de pacientes, além de confrontar o diagnóstico fornecido pelo método com outros especialistas.

5.1 Contribuição

A principal contribuição deste trabalho é a comparação da metodologia proposta por Almeida (2013) aplicada a ambientes diferentes. Outra contribuição deste trabalho, apresentada na seção 1.1, é a utilização do aplicativo nas avaliações pré, durante e pós cirurgia do estrabismo, auxiliando o especialista no diagnóstico do estrabismo melhorando a confiabilidade das medidas fornecendo uma segunda opinião para o especialista.

5.2 Trabalhos Futuros

O aplicativo proposto tem por objetivo servir de ferramenta de auxílio para diagnóstico de estrabismo. No entanto, com o objetivo de aperfeiçoa-lo existem diversas propostas que podem dar continuidade ao trabalho, a saber:

- Com o objetivo de melhorar a aquisição das imagens, pode ser implementado um módulo de câmera para o aplicativo, com a finalidade de dá suporte à centralização do paciente e, ainda, fazer um pré-diagnóstico antes mesmo da captura da imagem e compará-lo ao diagnóstico;
- Com o objetivo de aperfeiçoar a etapa de detecção da face, podem ser feitos testes e análise com a detecção de face fornecida pela API do Android, implementado diretamente na câmera. Ainda relacionado a esta etapa, podem ser feitos testes e análise também utilizando o Classificador em Cascata de Haar afim de diminuir o tempo de resposta do diagnóstico;
- Com o objetivo de melhorar a detecção precisa do limbo, pode ser feita a localização do mesmo mediante a quatro toques feitos pelo usuário na imagem,

e a partir das coordenadas dos mesmos traçar o círculo máximo contido nas quatro coordenadas. Ainda, fornecer a opção para que o usuário possa realizar ajustes dos valores das coordenadas (x y) do centro e raio do círculo;

- Implementar uma maneira alternativa do método de Canny, retirando a recursão presente no algoritmo na tentativa de inibir o estouro de pilha na memória. Ainda para o método de Canny, fazer testes com outras implementações, como o Canny fornecido pela biblioteca OpenCV;
- Com o objetivo de adicionar funcionalidades ao aplicativo: realizar melhorias na interface e usabilidade do aplicativo e implementar um banco de dados no aplicativo para que o usuário possa armazenar as imagens dos pacientes, assim como diagnóstico e outras informações que o mesmo deseja guardar;
- Com o objetivo de melhoria constante do aplicativo, implementar conexão do aplicativo com a internet afim de dar a opção para o usuário reportar erros.
- Utilizar o aplicativo na triagem de pacientes em idade escolar na cidade de Pinheiro-MA, com o objetivo de diagnosticar diagnosticar possíveis casos de estrabismo na região.

Dessa forma, foram relacionadas as principais contribuições e sugestões de melhorias que possam ampliar e melhorar o aplicativo, bem como auxiliar o desenvolvimento de novas pesquisas relacionadas à utilização de ferramentas computacionais no auxílio ao diagnóstico de estrabismo.

REFERÊNCIAS

ALMEIDA, J. D. S. **Metodologia Computacional para Detecção e Diagnóstico Automáticos e Planejamento Cirúrgico de Estrabismo**. Tese (Doutorado) – Doutorado em Engenharia de Eletricidade, Universidade Federal do Maranhão, São Luís, MA, 2013.

ANDROID DEVELOPERS. Disponível em: <http://developer.android.com/develop/index.html.>. Acesso em: junho de 2014.

BALLARD, D.H.; BROWN, C.M. **Computer Vision**. Prentice Hall, NJ, 1982.

BICAS, H. Consideraciones sobre los factores mecánicos en la acción de los músculos oculares. **Acta Estrabologica**, n. XXV, p. 161-178, 1996.

BICAS, H. Interpretação dos mecanismos de ação dos procedimentos cirúrgicos em estrabismo. In: 12º **Congreso del Consejo Latinoamericano de Estrabismo**. Buenos Aires: [s.n.], 1996. p. 297-306.

BICAS, H. Estrabismos: Da teoria à prática, dos conceitos às suas operacionalizações. **Arquivos Brasileiros de Oftalmologia**, v. 72, n. 5, p. 585-615, 2009.

BRAHLER, S. **Analysis of the android architecture. Karlsruher Intitute of Technology**, Germany, 2010.

BREITMAN, K.; ANIDO, R. **Atualizações em Informática.** Rio de Janeiro: PUC-Rio, 2006.

BURGISS, S.; GOODRIDGE, S. Multiframe averaging and homomorphic filtering for clarification of dark and shadowed video scenes [4232-75]. In: **Proceedings-Spie The International Society for Optical Engineering**. [S.l.: s.n.], 2000. p. 480-488.

CAMARGO, E. C. G. **Desenvolvimento, implementação e teste de procedimentos geoestatísticos (KRIGEAGEM) no sistema de processamento de informações georreferenciadas (SPRING)**. Dissertação (Mestrado) - Instituto Nacional de Pesquisas Espaciais, São José dos campos, SP (Brasil), 1997.

CANALYS. **A third of smart phones shipped in Q1 had 5"-plus displays**. Disponível em: <http://www.canalys.com/newsroom/third-smart-phones-shipped-q1-had-5-plus-displays> Acesso em: dezembro de 2014.

CANALYS. **Over 1 billion Android-based smart phones to ship in 2017**. Disponível em: <http://www.canalys.com/newsroom/over-1-billion-android-based-smart-phones-ship-2017> Acesso em: dezembro de 2014

CANNY, J. A computational approach to edge detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE Computer Society Washington, DC, USA, v. 8, n. 6, p. 679-698, 1986.

CHAI, D.; NGAN, K. Face segmentation using skin-color map in videophone applications. **IEEE Transactions on Circuits and Systems for Video Technology**, IEEE, v. 9, n. 4, p. 551-564, 1999.

CHANG, C.-C.; LIN, C.-J. **LIBSVM - A Library for Support Vector Machines**. Disponível em: <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>. Acessado em: junho de 2014.

CHOI, R. Y.; KUSHNER, B. J. The accuracy of experienced strabismologists using the hirschberg and krimsky tests. **Ophthalmology**, Elsevier, v. 105, n. 7, p. 1301-1306, 1998.

CRISTIANINI, N.; SHAWE-TAYLOR, J. **An Introduction to Support Vector Machines and Other Kernel-based Learning Methods**. [S.l.]: Cambridge University Press, Cambridge, UK., 2000.

DAVEMORRISSEY .**Custom image views for Android with pinch to zoom, panning, rotation, and animation support, with easy extension so you can add your own overlays and touch event detection**. Disponível em: <https://github.com/davemorrissey/subsampling-scale-image-view>. Acessado em: novembro de 2014.

D'ORAZIO, T.; LEO, M.; GUARAGNELLA, C.; DISTANTE, A. A visual approach for driver inattention detection. **Pattern Recognition**, Elsevier, v. 40, n. 8, p. 2341-2355, 2007.

DUDA, R. O.; HART, P. E. Use of the hough transformation to detect lines and curves in pictures. **Communications of the ACM**, ACM, v. 15, n. 1, p. 11-15, 1972.

ELINUX. Disponível em: <http://elinux.org/Android_Portal>. Acesso em: junho de 2014.

FOUNDATION, E. Eclipse IDE. 2014. Disponível em: <http://http://www.eclipse.org/downloads/>. Acesso em: fevereiro de 2014.

GERACI, A. K. IEEE Standard Glossary of Image Processing and Pattern Recognition Terminology. **Institute of Electrical and Electronics Engineers (IEEE) Std**, p. 610-4, 1990.

GONZALEZ, R. C.; WOODS, R. E. **Digital image processing**. [S.l.]: Prentice Hall, NJ, 2002.

GOOGLE I /O. Disponível em: <https://www.google.com/events/io>. Acesso em: novembro de 2014.

HAIR, J. F. J.; ANDERSON, R. E.; L., T. R.; BLACK, W. C. **Análise Multivariada de Dados**. [S.l.]: Bookman, 2005.

HAYKIN, S. **Neural networks: a comprehensive fundation.** 2a Edition. Upper Saddle Rever, New Jersey: Prentice Hall, 1999

HEARST, M. A. Trends & Controversies: Support Vector Machines. **IEEE Intelligent Systems**, v. 13, n. 4, p. 18-28, 1998.

IDC. **Smartphone OS Market Share, Q3 2014.** Disponível em <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>. Acesso em: dezembro de 2014.

ISAAKS, E.H.; SRIVASTAVA, R.M. **An introduction to applied geostatistics.** New York, Oxford University Press, 1989. 561p.

JHLABS. **Java Image Filters**. Disponível em: <http://www.jhlabs.com/ip/filters/>. Acesso em: novembro de 2014.

JUNQUEIRA, L. C.; CARNEIRO, J. **Histologia básica**. 8a Edição. [S.l.: s.n.], 1999.

KENNETH, R. **Castleman, Digital image processing**. [S.l.]: Prentice Hall Press, Upper Saddle River, NJ, 1996.

KHAN, Sohail; KHAN, Shahryar; BANURI, Syed Hammad Khalid Banuri; ALAM, Masoom. **Analysis report on android application framework and existing security architecture**. Pakistan: Security Engineering Research Group, 2010 (Technical report).

KRIGE, D. A statistical approach to some mine valuation and allied problems on the Witwatersrand. **Journal of the Chemical, Metallurgical and Mining Society of South Africa**, dec. 1951, p. 119-139.

LACHENBRUCH, P. A.; GOLDSTEIN, M. Discriminant Analysis. **Biometrics**, JSTOR, v. 35, n. 1, p. 69-85, 1979.

MATHERON, G. Principles of geoestatistics. **Economic Geology**, Lancaster, v.58. p. 1246-1266, 1963.

MATHERON, G. The theory of regionalized variables and its application. **EcoleNationale Supérieure des Mines de Paris**, Paris. 1971. v.5. 212 p.

MELO, R. d.; VIEIRA, E. d. A.; CONCI, A. A System to enhance details on partially shadowed images. **by A. Karras, S. Voliotos, M. Rangouse and A. Kokkosis**, p. 309-312, 2005.

MERCK. **Problemas de Saúde na Infância**. 2009. Disponível em <http://www.msd-brazil.com>. Acesso em: novembro de 2014.

MORDONEZ-ME. **Implementation of some Instagram filters using jhlabs library for android**. Disponível em: <https://github.com/mordonez-me/instagram-filters-jhlabs-android>. Acessado em: agosto de 2014.

OPENCV. Disponível em: < http://opencv.org/platforms/android.html>. Acesso em: junho de 2014

ORACLE. **Class StackOverflowError**. Disponível em: <http://docs.oracle.com/javase/7/docs/api/java/lang/StackOverflowError.html>. Acesso em: dezembro de 2014.

SBO. **Glossário da Sociedade Brasileira de Oftalmologia.** [S.l.], 2012. Disponível em: <http://www.sboportal.org.br/site2/glossario.asp>. Acessado em: janeiro de 2014.

SILVA, A.; GATTASS, M.; CARVALHO, P. Analysis of Spatial Variability using Geostatistical Functions for Diagnosis of Lung Nodule in Computerized Tomography Images. **Pattern Analysis and Applications**, Springer, v. 7, n. 3, p. 227-234, 2004.

SINGH, S. **SquintMaster Software**. 2012. Disponível em: <http://www.squintmaster.com>. Acesso em: 08 jun. 2014.

SOLER, L. S. **Uso de Radar de Abertura Sintética na Detecção de Manchas de Óleo na Superfície do Mar a partir de Classificação Textural na Região da Bacia de Campos** - RJ. Dissertação (Mestrado) - Instituto Nacional de Pesquisas Espaciais, São José dos Campos, SP, 2000.

SOUZA JR., O. S. S.; SILVA, A. C.; ABDELOUAH, Z. Personal Identification Based on Iris Texture Analysis Using Semivariogram and Correlogram Functions. **International Journal for Computacional Vision and Biomechanics, Serials Publications** - New Delhi, India, v. 2, n. 1, 2009.

TECHS, E. O. **Hirschberg Simulator**. [S.l.], 2009. Disponível em: <http://eyeontechs.com/>. Acesso em: março de 2014.

TOU, J. T., GONZALEZ, R. C. **Pattern Recognition Principles**. Addison-Wesley Publishing Company, Massachusetts, 1981.

VALE, G. M.; DAL POZ, A. P. Processo de detecção de bordas de Canny. In**: Boletim de Ciências Geodésicas**, Curitiba, v.8, n.2, 2002.

VAPNIK, V. N. **The Nature of Statistical Learning Theory**. [S.l.]: Springer Verlag, New York, 1995.

WITTEN, I. H.; FRANK, E. **Data Mining: Practical machine learning tools and techniques**. [S.l.]: Morgan Kaufmann, 2005.

WRIGHT, K.; FARZAVANDI, S.; THOMPSON, L. **Color atlas of strabismus surgery: strategies and techniques**. [S.l.]: Springer Verlag, 2007.