UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

EGG EVALUATION: OVOSCOPIA VIA SMARTPHONE

GUILHERME MURILO DA ROSA

BLUMENAU
2017

GUILHERME MURILO DA ROSA

EGG EVALUATION: OVOSCOPIA VIA SMARTPHONE

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. Dalton Solano dos Reis, Mestre - Orientador

BLUMENAU
2017

EGG EVALUATION: OVOSCOPIA VIA SMARTPHONE

Por

GUILHERME MURILO DA ROSA

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Prof. Dalton Solano dos Reis, Mestre - Orientador, FURB
Membro:	Prof. Marcel Hugo, Mestre – FURB
Membro:	Prof. Gilvan Justino, Mestre – FURB

Blumenau, 12 de dezembro de 2017

Dedico este trabalho ao meu pai Marcelo Murilo da Rosa, minha mãe Julice Virginia Spegiorin Maciel, minha avó Delba Sebastiana da Rosa e meu falecido avô Vicente da Rosa.

AGRADECIMENTOS

À minha família que me apoiou durante todos estes anos.

Aos meus amigos que compreenderam a minha ausência durante anos e sempre me deram apoio emocional.

Agradecimento especial para a minha amiga Jamile Sbardelotto Freitas, pela a sua ajuda na correção nos erros de português.

Ao meu orientador que participou na etapa da proposta Aurélio Faustino Hoppe, pelo seu bom humor, disposição e atenção em todas as nossas conversas, pela troca de experiência e conselhos.

Ao meu orientador Dalton Solano dos Reis, pela paixão pelo o que você faz, tornando suas aulas fruto de inspiração.

A todos os meus colegas que participaram comigo durante a minha graduação e que de alguma forma contribuíram na minha formação acadêmica.

“A mente que se abre a uma nova ideia jamais volta ao seu tamanho original.”

Albert Einstein

RESUMO

Com o crescimento do mercado agrícola, tem levantado a necessidade de investimentos em tecnologias de automação. A ovoscopia automatizada já está presente como produto fabril e também tem incentivado estudos acadêmicos para o aperfeiçoamento desta técnica. Este trabalho apresenta o desenvolvimento de um aplicativo para a plataforma Android que realiza processamento de imagens em ovos de galinha da cor branca, buscando extrair características como a sujeira, rachaduras e a classificação da qualidade da casca. As aplicações das técnicas de processamento de imagem foram realizados através da biblioteca JavaCV, que é uma adaptação da biblioteca OpenCV para Java. O aplicativo faz o uso de imagens digitais capturadas através de dispositivos móveis, tais como: fotos e fotos da galeria. Os resultados mostraram positivos, porém, demonstraram também pontos que devem ser melhorados, principalmente na funcionalidade de tempo real e na precisão. Por fim, foram sugeridas algumas extensões que buscam melhorar e expandir as funcionalidades do aplicativo criado.

Palavras-chave: Ovoscopia. OpenCV. JavaCV. Processamento de imagens. Android.

ABSTRACT

With the growth of the agricultural market, it has raised the need for investments in automation technologies. The automated cuboid is already present as a manufacturing product and has also encouraged academic studies to improve this technique. This work presents the development of an application for the Android platform that performs image processing on white chicken eggs, seeking to extract characteristics such as dirt, cracking and eggshell quality classification. The applications of the image processing techniques were through the JavaCV library, which is an adaptation of the OpenCV library for Java. The application makes use of digital images captured through mobile devices, such as: photos and photos from the gallery. The results were positive, however, they also showed points that should be improved, especially in real-time functionality and accuracy. Finally, some extensions were suggested that seek to improve and expand the functionality of the created application.

Key-words: Ovoscopy. OpenCV. JavaCV. Image processing. Android.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT – Associação Brasileira de Normas Técnicas

API – Application Programming Interface

FR – Free Range

HSV – Hue, Saturation and Value

IBGE – Instituto Brasileiro de Geografia e Estatística

JavaCV – Java interface to OpenCV

KGF – Kilogram-force

NDK – Native Development Kit

OpenCV – Open Source Vision Library

RF – Requisito Funcional

RNF – Requisito Não-Funcional

ROI – Region of interest

SDK – Software development kit

UML – Unified Modeling Language

SUMÁRIO

1 INTRODUÇÃO

A avicultura brasileira vem apresentando índices de crescimentos altos nas últimas três décadas (BELUSSO; HESPANHOL, 2010). Segundo a Secretaria de Estado da Agricultura e do Abastecimento (2012) o Brasil é o sétimo maior produtor mundial de ovos. Segundo o IBGE (2016), o Brasil registrou recorde na produção de ovos em 2016, a produção cresceu 5,8%, totalizando em 3,10 bilhões de dúzias.

Segundo Santini e Souza Filho (2005), a avicultura ainda é um mercado em expansão. Porém, precisa passar por investimentos e melhorias tecnológicas em toda sua cadeia produtiva, visando maximização de lucros, aumento de qualidade de produto e reduzindo o custo de produção, assim como, fornecendo diferenciais competitivos no mercado.

Para Heemann (2013), por ser um produto de origem animal, o ovo deve obrigatoriamente passar pelo processo de inspeção. Primeiramente, é realizada a inspeção das embalagens dos ovos, onde observam as características e condições da embalagem, como sua limpeza e cheiro. Depois disso, é verificado o estado de limpeza e integridade da casca. Esses ovos são higienizados e podem ser encaminhados para ovoscopia. A ovoscopia consiste na observação interna do ovo. O processo consiste basicamente em colocar o ovo contra um foco de luz em um ambiente escuro. A luz proveniente do ovoscópio permite a observação de anormalidades no ovo. Primeiro observa-se a casca, buscando rachadura e irregularidades, após observa-se as condições da gema, principalmente sua mobilidade e então inspeção da clara, onde é procurada a presença de corpo estranho no ovo, manchas de sangue, desenvolvimento de embrião e o natural aumento da câmara de ar. Com base nos parâmetros citados acima o ovo é destinado para o consumo "in natura", ovo encontrado no comércio, ou ainda pode ser destinado para processos industriais que permitam seu consumo (BRASIL, 1952).

Visto que o mercado continua em expansão e que a inspeção do ovo tem sua importância para garantir a qualidade do ovo, foi desenvolvido um aplicativo capaz de avaliar a qualidade, bem como a classificação das características que podem ser observadas na parte externa.

1.1 OBJETIVOS

O objetivo deste trabalho é desenvolver um aplicativo para smartphones que possa ser uma etapa inicial para o desenvolvimento de trabalhos sucessores para se tornar um produto para o usuário final, que seria o consumidor do varejo.

Os objetivos específicos do trabalho são:

a) capturar imagens de ovos através de um dispositivo móvel;
b) efetuar a segmentação dos ovos existentes nas imagens;
c) extrair e analisar as características morfológicas das imperfeições externas existentes nos ovos, para avaliação da qualidade dos mesmos.

1.2 ESTRUTURA

Este trabalho está dividido em quatro capítulos: introdução, fundamentação teórica, desenvolvimento e conclusões. O segundo capítulo apresenta a fundamentação teórica, onde será abordada a inspeção de ovos, sendo ela manualmente e através de software. No terceiro capítulo é demonstrada a especificação do aplicativo e o detalhamento da implementação. Por fim, no quarto capítulo são descritas as conclusões, os resultados obtidos e algumas extensões.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo está organizado em três seções. A seção 2.1 detalha o processo de inspeção de ovos. A seção 2.2 apresenta conceitos de segmentação de imagens. Na seção 2.3 são listados três trabalhos correlatos sobre diagnóstico das imperfeições em ovos.

2.1 INSPEÇÃO DE OVOS

Resumidamente, a inspeção e classificação dos ovos consiste em seis etapas, sendo elas recepção e seleção por tamanho, higienização para remoção das fezes, ovoscopia, óleo mineral/ácido peracético para reposição da cutícula de mucina, classificação e embalagem dos ovos (BRASIL, 1952). Os ovos que não se aproximam das características mínimas exigidas para as classes e tipos estabelecidos pela legislação brasileira são considerados impróprios para o consumo in natura, sendo apenas permitida sua utilização na indústria (BRASIL, 1991).

A ovoscopia é realizada em duas etapas, avaliação interna e externa, por meio do ovoscópio (BRASIL, 1952). No ovoscópio o ovo é exposto contra um feixe de luz, sendo que esta pode ser natural ou artificial. A Figura 1 apresenta um ovoscópio comercial. O modelo abaixo permite avaliação de um ovo. Porém, há ovoscópios que possibilitam avaliação de mais de um ovo simultaneamente, onde o objetivo é que a luz incida sobre o ovo para avaliação externa e através do ovo para avaliação interna.

Figura 1 - Ovoscópio

Fonte: Chocadeiras Golden (2016).

Ao ser colocado no ovoscópio, com o feixe de luz acesso, primeiramente é realizada a avaliação externa e em seguida a avaliação interna. A avaliação externa tem como objetivo determinar a qualidade da casca do ovo, destacando pontos como: limpeza, textura, formato e integridade da casca, apontando assim, quaisquer alterações como: ovos defeituosos, ovos trincados e vazados, sujos com sangue e/ou fezes e ovos com casca fina.

Quanto à integridade da casca, ovos com as cascas trincadas (observa-se uma fratura da casca, mas as membranas da casca estão íntegras) ou quebradas (quando as membranas da casca estão rompidas com ou sem extravasamento de conteúdo) não devem ser destinados ao consumo (COUTTS; WILSON, 2007). Os ovos que possuem cascas sujas por fezes prejudicam a avaliação do produto e até aumentam a probabilidade de contaminação bacteriana do ovo. Em relação à textura da casca, a mesma deve ser lisa, sem qualquer deformação ou manchas, pois as áreas ásperas e ou enrugadas são pontos mais frágeis, possibilitando a quebra do ovo (GUEDES, 1961). O tamanho do ovo pode ser definido por alguns fatores como idade do lote, precocidade das aves no início de produção, manejo alimentar e níveis nutricionais, consumo de água e ração e até pela temperatura ambiente ao qual as aves foram expostas (SESTI; ITO, 2009). O ovo deve ser ovoide, ovos que apresentam formato irregular, como alongamento e achatamento, são mais frágeis e consequentemente mais sujeitos a quebra no transporte (GUEDES, 1961).

A luz proveniente do ovoscópio permite também a observação de anormalidades internas no ovo. Podem ser encontrados corpo estranho no ovo, manchas de sangue, desenvolvimento de embrião e o aumento da câmara de ar. Na Figura 2 há alguns exemplos de alterações em ovos durante a avaliação externa.

Figura 2 - Avaliação externa

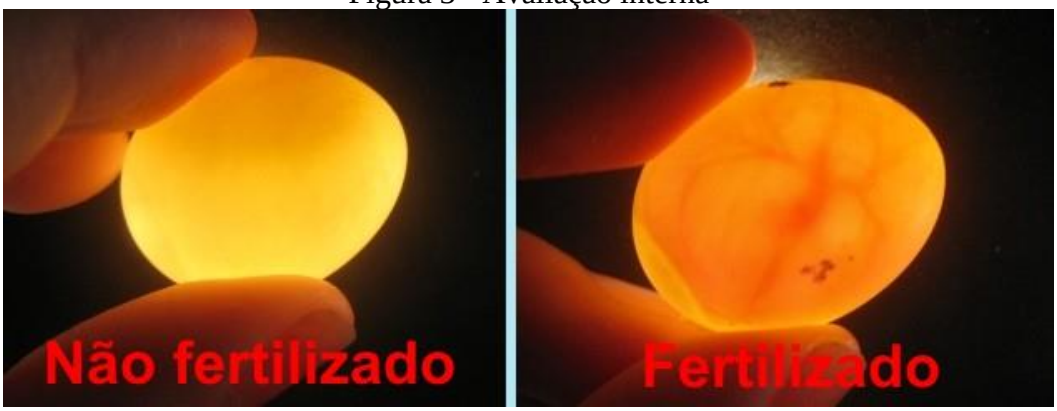
Fonte: Cobb (2013).

Como pode ser observado na Figura 2, os ovos podem apresentar diversas alterações como estarem sujos mesmo após o processo de higienização, pode haver alterações estruturais como cascas fina e áspera, enrugada, trincada pela unha da galinha, face plana, ovo arredondado e até com um tamanho anormal, que é indicativo de gema dupla (BRASIL, 1991).

Em relação ao aspecto interno, podem ocorrer pequenas manchas de carne ou sangue na gema ou na clara, o que é um fato normal e não prejudica o valor dos ovos para o consumo. Em contrapartida, ovos com desenvolvimento embrionário, também chamados de galados, ou deteriorados, não são aceitáveis para a comercialização, sendo considerados impróprios para o consumo. É possível constatar ainda ruptura de gema, infecção por fungos entre outras (OLIVEIRA, 2000).

Na Figura 3 é possível observar a diferença entre um ovo não fertilizado, próprio para consumo in natura e, um ovo fertilizado, chamado também de ovo galado, que pela legislação em vigor é impróprio para consumo in natura e também industrial.

Figura 3 - Avaliação interna



Fonte: Chocadeiras Faisão (2016).

Como pode ser observado na Figura 3, é nítida a diferença entre um ovo não fertilizado, onde seu interior é livre de qualquer sombra ou mancha. Já um ovo fertilizado, ou galado, o padrão de imagem pode variar de acordo com o desenvolvimento do embrião. No caso da Figura 3, trata-se de um embrião de cerca de 9 (nove) dias. Porém, essa constatação só é possível com auxílio da ovoscopia, pois externamente ambos os ovos são semelhantes (ALMEIDA, 1999).

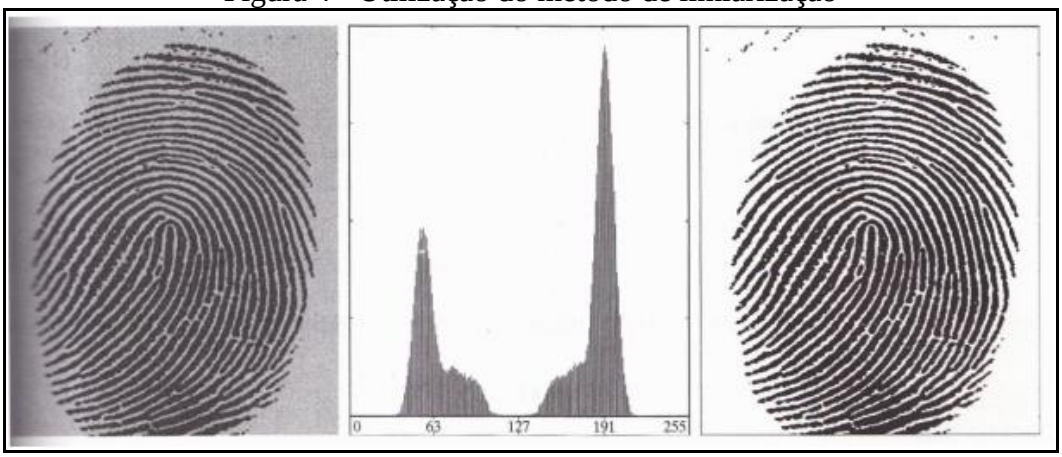
2.2 SEGMENTAÇÃO DE IMAGENS

Uma imagem digital a ser processada pode ser representada de forma bidimensional, em escalas de cinza e possui como seus atributos linhas, colunas e a intensidade de cada pixel da imagem (GONZALES; WOODS, 2008, p. 23).

A segmentação da imagem é o processo que subdivide a imagem em elementos, a qual pode ser distinguida em objetos relevantes e o plano de fundo. Sua capacidade de precisão é determinante para o sucesso ou falha dos procedimentos de análise computadorizada. É importante nesta etapa aplicar estes procedimentos somente o suficiente para a distinção dos elementos, não havendo necessidade de segmentar uma imagem acima do nível de detalhe necessário para a identificação dos mesmos (GONZALES; WOODS, 2008, p. 711).

Segundo Gonzales e Woods (2008, p. 760), a limiarização (thresholding) é uma das técnicas mais utilizadas na segmentação de uma imagem em tons de cinza. A Figura 4 representa o resultado da aplicação do algoritmo.

Figura 4 - Utilização do método de limiarização



Fonte: adaptado de Gonzales e Woods (2008, p. 765).

A segmentação é realizada percorrendo a imagem pixel a pixel, rotulando com 1 o que corresponde como parte relevante e 0 os pixels que correspondem ao fundo. O resultado deste método pode variar conforme o limiar. Na segunda imagem da Figura 4 que pode ser visto o histograma agrupado em duas concentrações de coloração, que por sua vez, foi considerado como plano de fundo os tons de cinza com escala acima de 127.

2.3 TRABALHOS CORRELATOS

A seguir estão relacionados três trabalhos correlatos ao proposto. O item 2.3.1 descreve o Sistema de Inspeção Visual Automática Aplicado ao Controle de Qualidade de Ovos em Linhas de Produção (MACHADO, 2009). No item 2.3.2 é apresentado um estudo sobre a espessura das cascas dos ovos e as consequências de um ovo malformado (GALVÃO, 2013). O item 2.3.3 trata da detecção automática dos defeitos nas cascas dos ovos (GARCÍA, 2000).

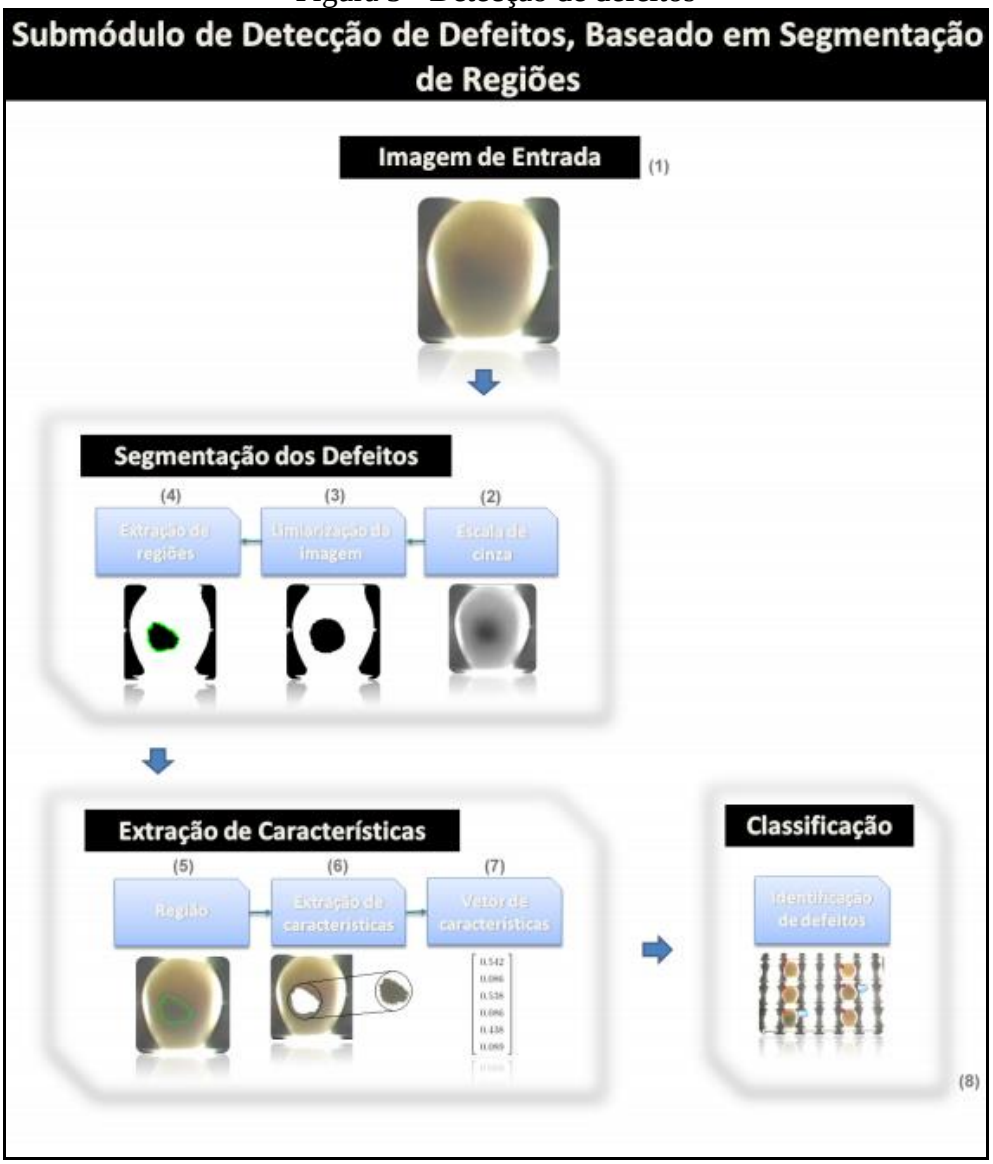
2.3.1 Sistema de Inspeção Visual Automática Aplicado ao Controle de Qualidade de Ovos em Linhas de Produção

Segundo Machado (2009), as técnicas de processamento de imagem podem ser utilizadas para simular a visão humana, evitando problemas de cansaço na visão. Este sistema busca substituir a mão de obra humana nas tarefas de inspeção do controle de qualidade dos ovos, fornecendo o aumento da produção, por ser uma tarefa automatizada.

A solução proposta por Machado (2009) permite realizar contagem dos ovos com 100% de acerto e a tipificação dos defeitos encontrados em amostras com defeitos previamente conhecidos, sejam eles a detecção com precisão de 75.6% de sujeiras, trincas de 73.3%, manchas de sangue de 78.5%, vazamentos de gema de 62.5% e pontos de sangue de 50%.

O algoritmo aplicado no processo de extração de características dos defeitos internos é idêntico ao de extração de defeitos externos. A detecção dos defeitos é realizada em etapas a partir de imagens geradas de uma fotografia original, conforme mostra a Figura 5.

Figura 5 - Detecção de defeitos



Fonte: Machado (2009 p. 90).

Na Figura 5 pode-se observar o processo de segmentação da imagem. Na primeira etapa realiza-se a captura da imagem. Na segunda etapa é gerada uma imagem em tons de cinza. Na terceira etapa é realizada a segmentação da imagem. Na quarta etapa é realizada a identificação da mancha encontrada dentro do ovo. Na quinta etapa são realçadas as bordas da mancha na imagem original. Na sexta, sétima e oitava respectivamente, é realizada a extração de características e a partir do vetor de características é realizada a tipificação do defeito, sendo que o resultado final será uma imagem com o destaque somente nos ovos onde foram encontrados estes defeitos.

2.3.2 Ovos produzidos em diferentes sistemas de alojamento: qualidade e segurança microbiológica, parâmetros físicos, validação e utilização de método multiresíduo para detecção de antimicrobianos e pesticidas

O objetivo do estudo, realizado por Galvão (2013), foi comparar parâmetros de qualidade e segurança alimentar de ovos produzidos em dois diferentes sistemas de alojamento para galinhas poedeiras, Free Range (FR), onde as galinhas não vivem em gaiolas e tem acesso à área externa, e bateria de gaiolas, sistema clássico de granja. Ao término do estudo foi constatado, por meio de testes validados, que os ovos do sistema FR apresentaram menor peso total, maior espessura de casca e concentração de albúmen. Segundo Galvão (2013), a espessura das cascas afeta diretamente as propriedades internas dos ovos, uma casca muito fina torna o ovo mais sensível à quebra e também à proliferação bacterianas, afetando na durabilidade do ovo como produto final.

Foram realizados testes microbiológicos nos ovos e no ambiente das aves. Os testes físicos foram realizados somente nos ovos. Os testes microbiológicos realizados foram pesquisa de Salmonella em ambiente de produção, água, ração e nas cascas dos ovos. Essa bactéria foi pesquisada pelo seu interesse na saúde pública, uma vez que a salmonelose é uma grave doença em seres humanos, além de afetar a produtividade da ave quando em desequilíbrio. Também foi realizada enumeração de enterobactérias.

Dentre os testes físicos realizados merecem destaque os testes de gravidade específica, resistência da casca do ovo à quebra e espessura da casca. O teste de gravidade específica foi realizado mergulhando os ovos em soluções salinas com densidades variando de 1,060 a 1,100, sendo que a gravidade específica de cada ovo foi determinada pela menor solução em que o ovo flutuou. A resistência da casca do ovo é um teste realizado usando uma sonda de ruptura, que está ilustrada na Figura 6.

Figura 6 - Sonda de ruptura

Fonte: Galvão (2013).

Essa sonda exerce força sobre o ovo até que a casca se rompa. Esta força exercida pela sonda é medida em quilograma-força (KGF) e tem como objetivo determinar a resistência do ovo a quebras. Para determinar a espessura da casca pode ser utilizado um paquímetro. Porém, primeiramente deve ser feita a lavagem da casca e sua secagem em estufa. Foram realizadas três medidas na região equatorial do ovo usando paquímetro, então foi calculada a média dessas mensurações e o resultado foi expresso em milímetros (Figura 7).

Figura 7 - Procedimento de determinação da espessura da casca

Fonte: Galvão (2013).

A imagem do lado esquerdo da Figura 7 mostra a lavagem da casca do ovo. Após essa etapa, a casca é colocada em uma estufa e então é medida com um paquímetro como pode-se observar na figura do lado direito.

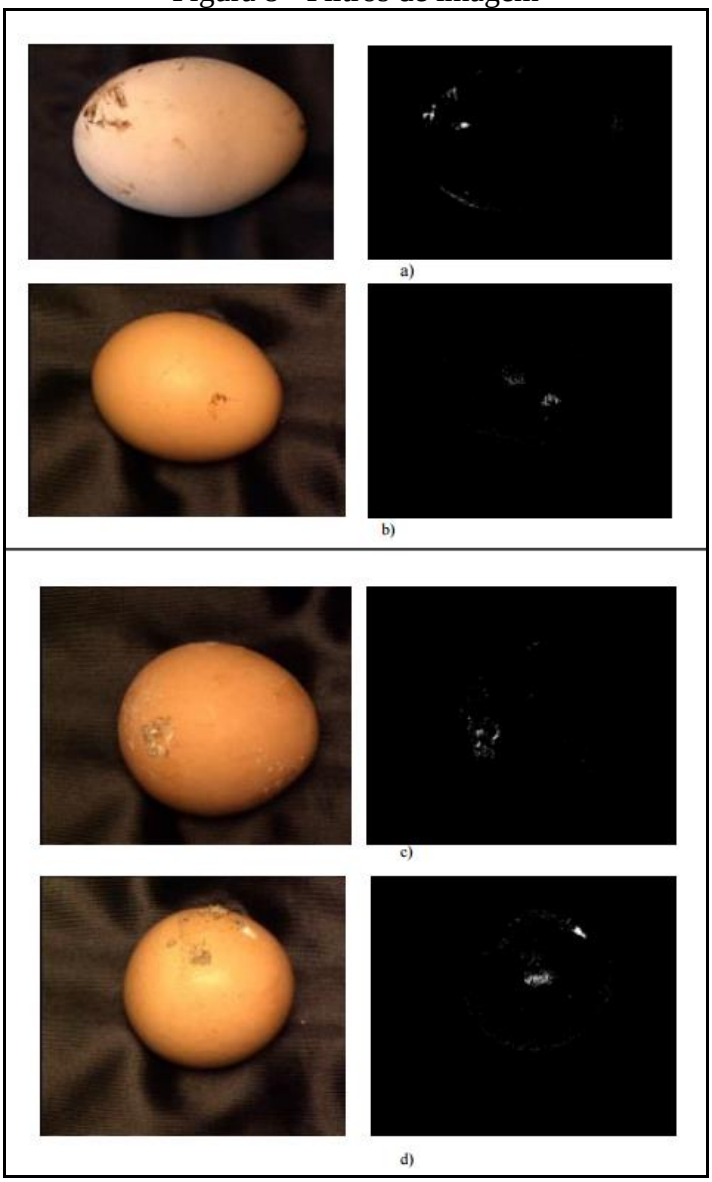
Em relação à qualidade física dos ovos, o que é de maior interesse para o presente estudo, a autora concluiu que mesmo observando diferenças estatísticas em alguns dos parâmetros físicos avaliados, estas diferenças não atribuem aos ovos produzidos pelo sistema Free Range (FR) um perfil especial que justifique sua maior escolha pelo consumidor. No entanto para o produtor existe o benefício de produzir ovos com cascas mais resistentes, que é interessante principalmente ao levar em consideração resistência maior durante o transporte e também contra infecções bacterianas.

2.3.3 Análise do índice de cor para detecção automática de defeitos na casca de ovos

O trabalho de García (2000) apresenta um sistema de visão artificial dedicado à rejeição automática de ovos sujos. Obteve-se alto sucesso na classificação de inconformidades nos ovos, tendo sua precisão de 80% nas sujeiras, 80% nas manchas de sangue e 90% nas rachaduras, com processamento executado em tempo real.

García (2000) encontrou dificuldades com relação à baixa diferença de intensidade entre a cor do ovo e dos defeitos, e também devido a iluminação que causa um ponto de brilho no centro da casca do ovo. Na Figura 8 tem-se a aplicação dos filtros de imagem para detecção de sujeiras nos ovos.

Figura 8 - Filtros de imagem



Fonte: García (2000).

A Figura 8 apresenta nas imagens à esquerda os ovos sujos e nas imagens à direita apresentam representações (em cor branca) das sujeiras a partir de filtros.

García (2000) propôs como solução às dificuldades encontradas, a utilização de uma fonte luminosa, fazendo com que a avaliação não confunda o reflexo da iluminação com um possível defeito no ovo.

3 DESENVOLVIMENTO

Este capítulo apresentará todo o ciclo de desenvolvimento do aplicativo desenvolvido, contendo os requisitos trabalhados, a especificação, a implementação e por fim, os resultados obtidos.

3.1 REQUISITOS

A seguir estão listados os requisitos do aplicativo, categorizados em Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF). O aplicativo deve:

a) permitir ao usuário tirar fotos de ovos a partir da câmera de um dispositivo móvel (RF);
b) permitir o usuário escolher uma foto da galeria de imagens de um dispositivo móvel (RF);
c) utilizar técnicas de realce e segmentação a fim de gerar uma nova imagem, a partir da imagem original, que mostre de forma realçada quais são os ovos defeituosos (RF);
d) extrair características externas dos ovos que sendo elas: presença de rachaduras, sujeiras e qualidade da casca (RF);
e) ser implementado utilizando a linguagem de programação Java (RNF);
f) ser implementada utilizando a biblioteca JavaCV (RNF);
g) ser desenvolvido para a plataforma Android (RNF).

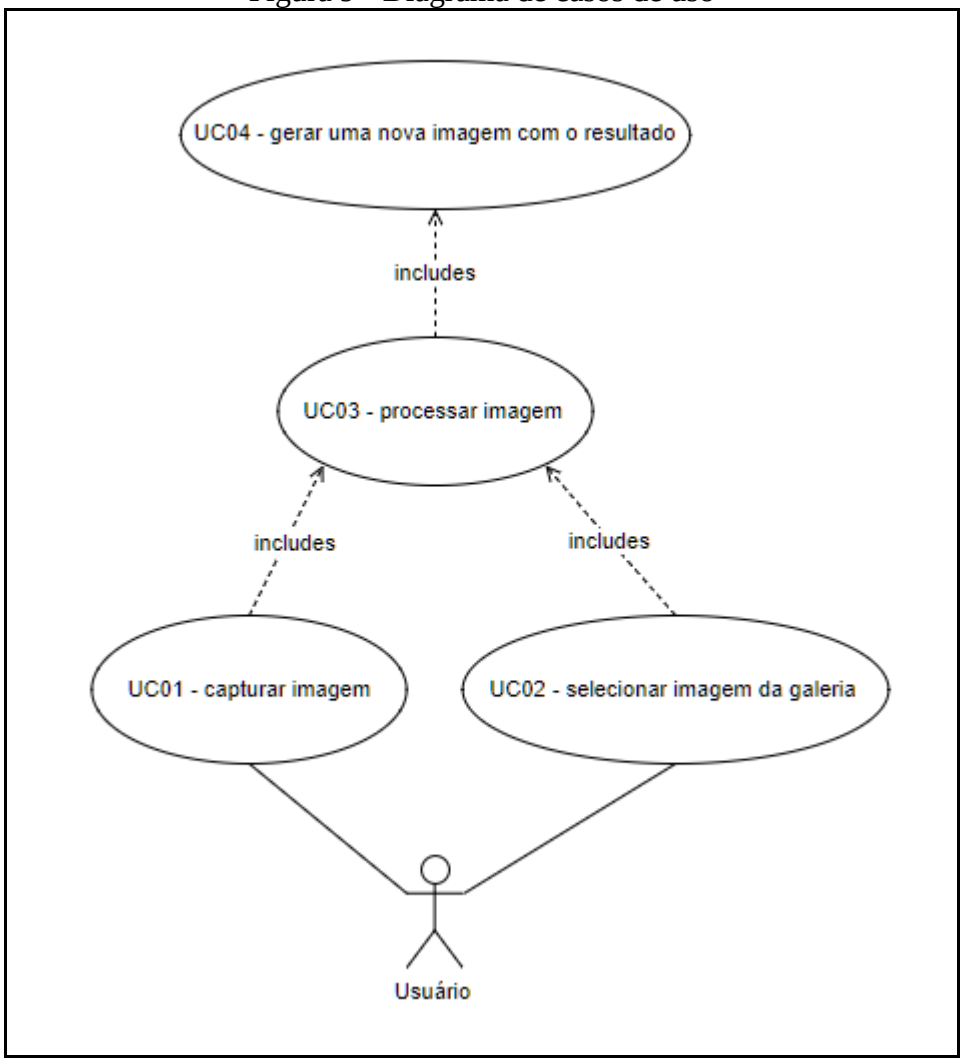
3.2 ESPECIFICAÇÃO

A especificação do aplicativo foi representada através de diagramas da Unified Modeling Language (UML), utilizando a ferramenta Draw.io. A Seção 3.2.1 apresenta o diagrama de casos de uso, em seguida na seção 3.2.2, apresenta o diagrama de classes.

3.2.1 Diagrama de casos de uso

A partir do levantamento dos requisitos da aplicação, foi desenvolvido o diagrama de casos de uso conforme ilustrado na Figura 9, que representam as funcionalidades disponíveis para o ator `Usuário`.

Figura 9 - Diagrama de casos de uso



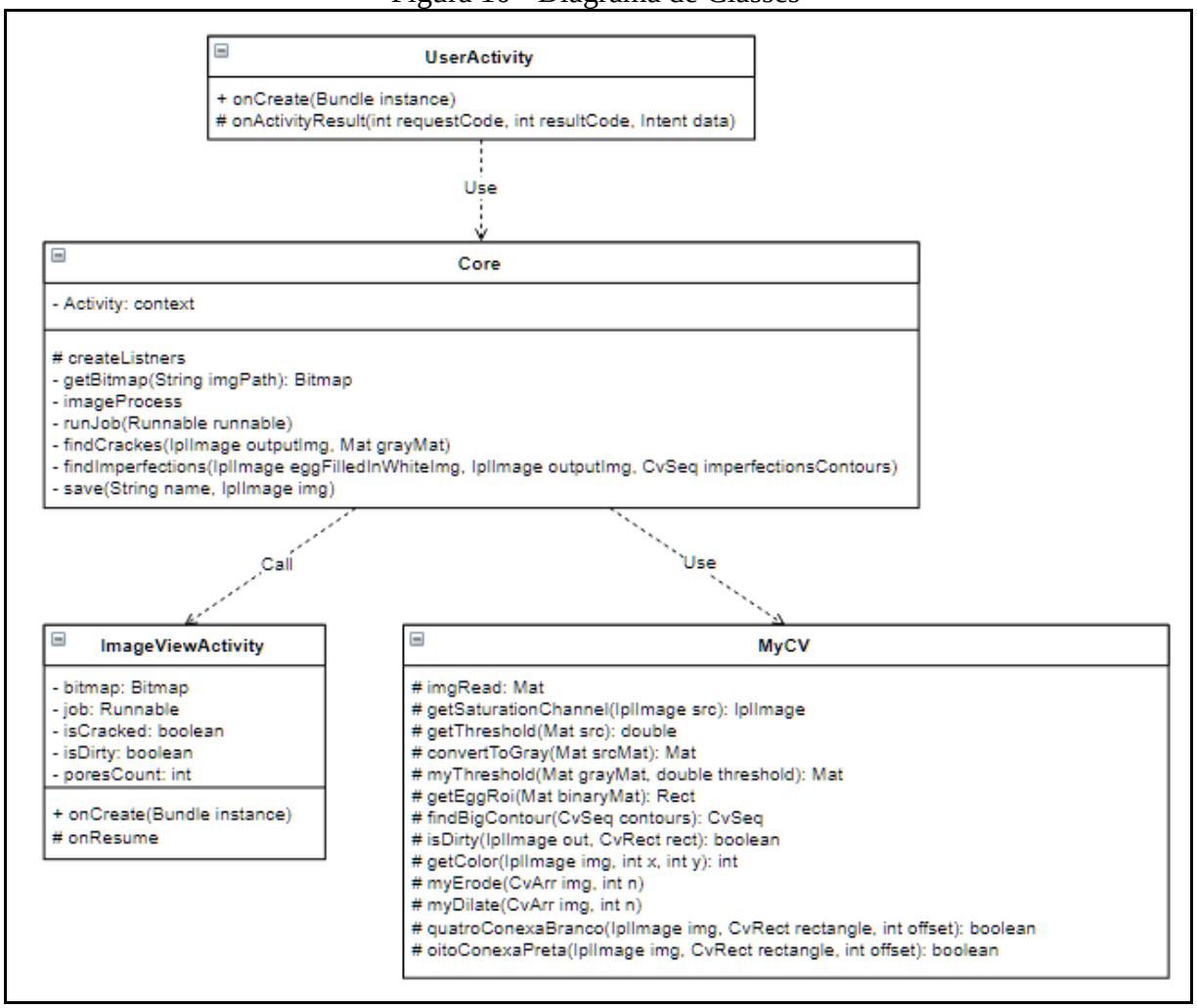
Fonte: Elaborado pelo autor.

O ator `Usuário` inicializa diretamente com dois casos de uso. O caso de uso `UC01 – capturar imagem` permite ao `Usuário` tirar uma foto a partir do dispositivo. O caso de uso `UC02 – selecionar imagem da galeria` descreve a funcionalidade que permite ao `Usuário` selecionar uma foto da galeria de imagens do dispositivo. O caso de uso `UC03 – processar imagem` que se refere a execução do algoritmo de processamento de imagens. E, por fim, o caso de uso `UC04 – gerar uma nova imagem com o resultado` que apresenta ao usuário a imagem processada com as informações qualitativas do ovo.

3.2.2 Diagrama de classes

O diagrama de classes apresenta uma visão de como as classes estão estruturadas e relacionadas. Nesta seção são descritas as classes que compõem o aplicativo, conforme apresenta a Figura 10.

Figura 10 - Diagrama de Classes



Fonte: Elaborado pelo autor.

A classe `UserActivity` fornece as opções de interação com usuário, onde serão executadas rotinas a partir dos eventos de toque de botão. A classe `Core` é o núcleo da implementação, é onde se encontram as implementações relacionadas à visão computacional no ovo. Esta classe é composta pelo seguinte ciclo de vida: ler a imagem, processar e salvar. A classe `MyCV`, trata-se de um utilitário para facilitar o uso da biblioteca JavaCV e permitir que o código fique mais legível. A classe `ImageViewActivity` é responsável por exibir ao usuário o resultado do processamento de imagem.

3.3 IMPLEMENTAÇÃO

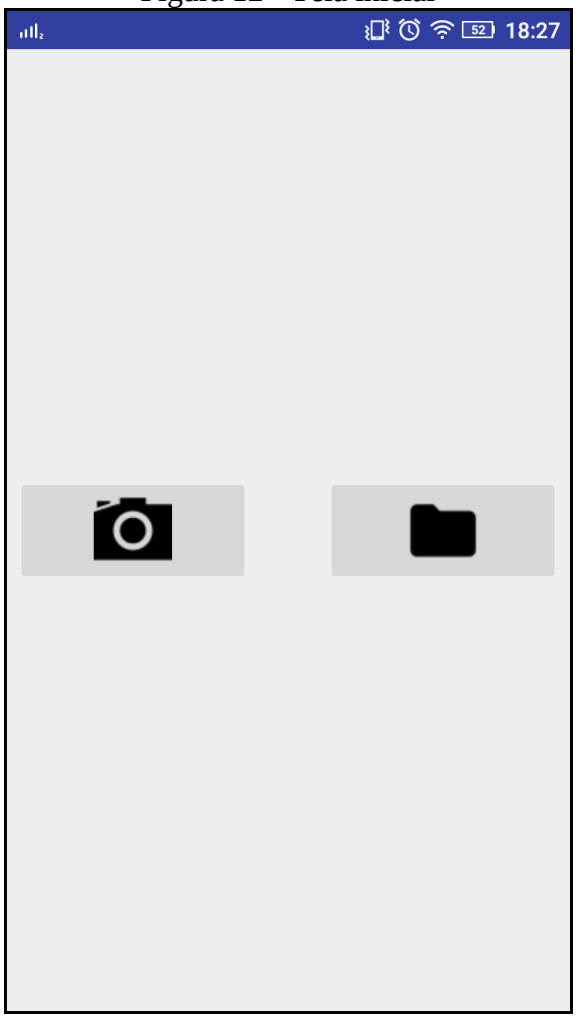
A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

Para o desenvolvimento do aplicativo foi utilizado a linguagem de programação Java na versão 8. O ambiente de desenvolvimento foi o Android Studio 3.0, ferramenta oficial para criação de aplicativos Android disponibilizada pela Google. Além de todo o ambiente de desenvolvimento, são disponibilizadas várias API's pelo SDK do Android, como a de câmera, galeria de imagens e manipulação de imagens, também oferece um simulador para testes sem dispositivos físicos. Foi utilizado Gradle como ferramenta de *build* e injeção de dependências. Foi utilizado a biblioteca JavaCV específica para desenvolvimento em dispositivos Android. Esta biblioteca utiliza o recurso NDK (Native Development Kit), interface que por sua vez permite chamadas de sub-rotinas escritas em C da biblioteca OpenCV. A biblioteca JavaCV é apenas uma camada de abstração em Java que por fim executa métodos da biblioteca OpenCV.

Ao abrir o aplicativo é exibida a tela principal, na qual o usuário opta pela captura de uma imagem ou selecionar uma já existente da galeria. A Figura 12 demonstra a tela inicial do sistema.

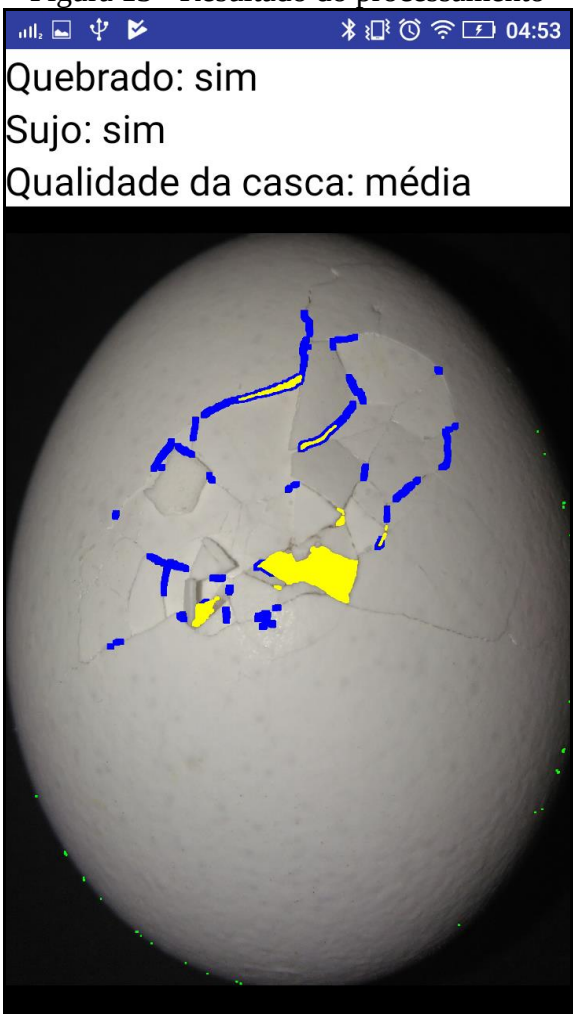
Figura 12 - Tela inicial



Fonte: Elaborado pelo autor.

O primeiro botão, retratado pelo ícone de uma câmera, seleciona a opção de captura de imagem. O segundo botão, retratado pelo ícone de uma pasta, seleciona a opção de uma imagem a partir da galeria de fotos do dispositivo. Após a seleção de uma imagem é realizado o processamento, em seguida é exibido a imagem gerada com o resultado do processamento, como pode ser observado na Figura 13.

Figura 13 - Resultado do processamento



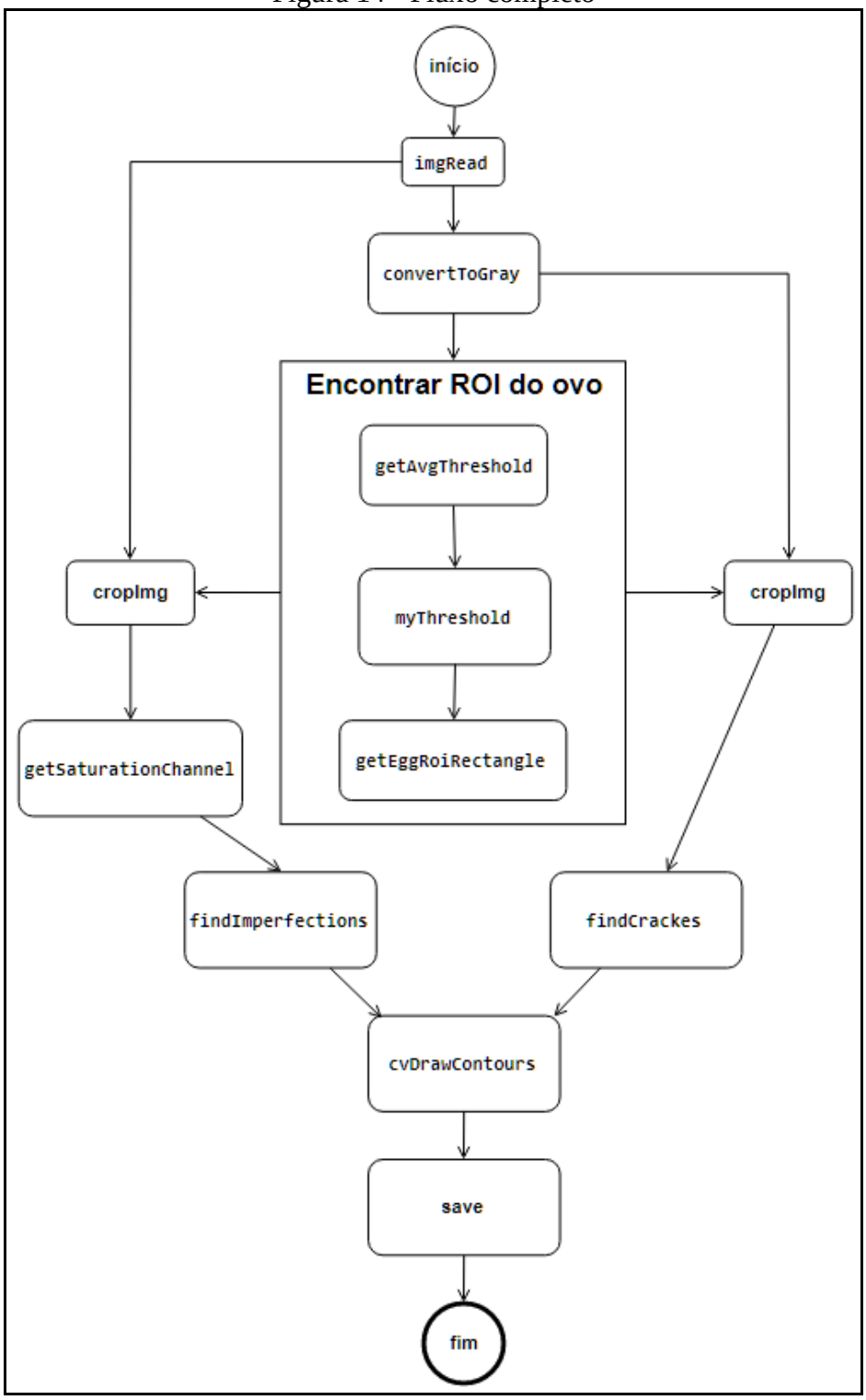
Fonte: Elaborado pelo autor.

O resultado apresenta as características encontradas no ovo. Estas características são respectivamente: se o ovo está quebrado, presença de sujeira e a qualidade da casca. As rachaduras do ovo são representadas da cor azul. A sujeira do ovo, como fezes ou gema, são representadas da cor amarela. E por último a porosidade da casca, que é representada na cor verde. A qualidade da casca é avaliada com a menor presença de poros na casca. Estes poros são percebidos através de pequenas ondulações na casca. Estas ondulações são mais perceptíveis nas extremidades da casca devido ao leve sombreado causado pela protuberância na casca e também pela posição da câmera e luz. O aplicativo analisa a seguinte faixa de poros: até 20 a qualidade é boa, em seguida até 80 a qualidade é média, acima disso a qualidade do ovo é ruim.

3.3.2 Processamento de imagem

O diagrama de atividades da Figura 14, mostra o fluxo completo do aplicativo para encontrar e classificar as anomalias encontradas no ovo.

Figura 14 - Fluxo completo



Fonte: Elaborado pelo autor.

A primeira etapa `imgRead` trata da leitura da imagem. Em seguida no método `convertToGray` é gerado uma nova imagem em escalas de cinza. Com a imagem em escalas de cinza a partir da etapa `getAvgThreshold` é obtido o grau de limiar a partir da média da coloração entre 0 a 255. No Quadro 1 pode ser observado o código para obter o limitar.

Quadro 1 - Código responsável por obter o limiar

```
1  try (CvScalar avg = cvAvg(src)) {
2     return avg.val(0);
3  }
```

Fonte: Elaborado pelo autor.

A classe `cvAgv` é responsável por calcular a média da concentração de cores, como a escala é em cinza, ela tem apenas um canal, o método `val` calcula a média da escala entre 0 e 255 deste canal. Foi percebido que este limiar a partir da média da coloração, é o ideal para identificação do ovo, pois a partir da etapa `myThreshold` é realizado a segmentação da imagem e é possível obter a superfície inteira do ovo. A *boundbox*, também conhecida como região de interesse (ROI) do ovo é obtida partir do método `getEggRoiRectangle`. No Quadro 2 apresenta o código para obtenção da ROI do ovo.

Quadro 2 - Código responsável por obter a ROI do ovo

```
1  try (IplImage contoursImg = new IplImage(binaryMat)) {
2     try (CvMemStorage storageCrop = CvMemStorage.create()) {
3        try (CvSeq contoursCrop = new CvContour()) {
4           cvFindContours(//
5                          contoursImg, //
6                          storageCrop, //
7                          contoursCrop, //
8                          Loader.sizeof(CvContour.class), //
9                          CV_RETR_CCOMP, //
10                         CV_CHAIN_APPROX_NONE, //
11                         new CvPoint(0, 0));
12
13          try (CvSeq eggContour = findBigContour(contoursCrop)) {
14             try (CvRect eggRectangle = cvBoundingRect(eggContour, 0)) {
15                return new Rect(//
16                                eggRectangle.x(), //
17                                eggRectangle.y(), //
18                                eggRectangle.width(), //
19                                eggRectangle.height());
20             }
21          }
22       }
23    }
24 }
```

Fonte: Elaborado pelo autor.

O método `cvFindContours` permite encontrar os contornos da imagem, já o método `findBigContour` encontra o maior contorno. Este maior contorno será a *boundbox* do ovo, que serve para cortar imagem, descartando grande parte do plano de fundo da imagem, que é

desnecessária para o processamento. É possível identificar o que é o ovo a partir do maior contorno encontrado na imagem, conforme pode-se observar no Quadro 3.

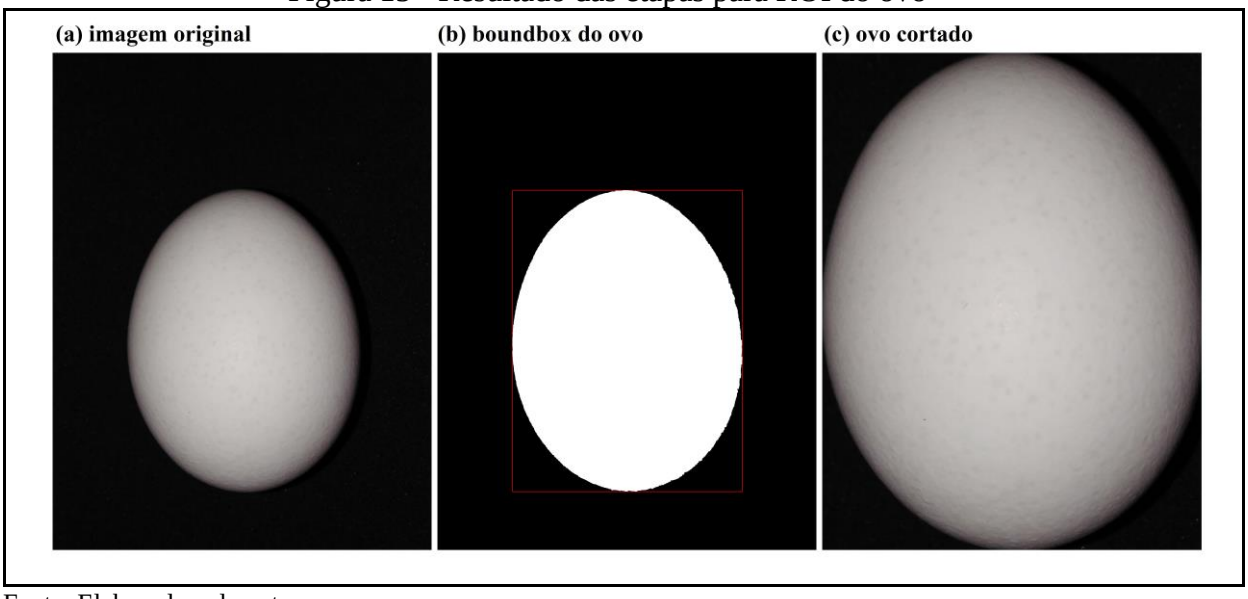
Quadro 3 - Código para encontrar o maior contorno

```
1  double bigArea = 0;
2  CvSeq bigContour = null;
3
4  for (CvSeq ptr = contours; //
5       ptr != null && ptr.address() != 0; //
6       ptr = ptr.h_next()) {
7     double size = cvContourArea(ptr);
8     if (size > bigArea) {
9        bigArea = size;
10       bigContour = ptr;
11    }
12 }
13 return bigContour;
```

Fonte: Elaborado pelo autor.

A etapa `cropImg` por sua vez, é executada duas vezes (vide Figura 14), uma para gerar uma nova imagem cortada a partir da imagem original e a outra para gerar uma imagem cortada a partir da imagem em escalas de cinza. Pode ser observado na Figura 15 o resultado das etapas de *boundbox* e corte da imagem.

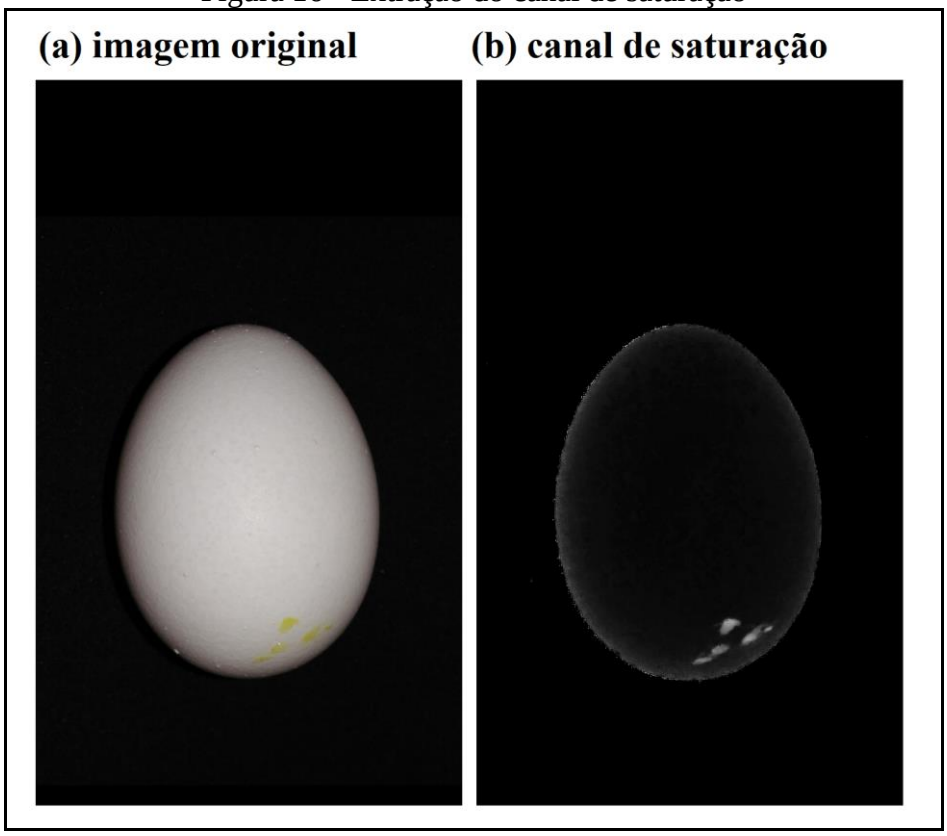
Figura 15 - Resultado das etapas para ROI do ovo



Fonte: Elaborado pelo autor.

Na imagem cortada a partir da imagem original é extraído da imagem em HSV o canal de saturação. HSV é a abreviatura para o sistema de cores formadas pelas componentes hue (matiz), saturation (saturação) e value (valor). Após experimentos foi identificado que o canal de saturação é o que mais realça as sujeiras do ovo, como pode ser verificado na Figura 16.

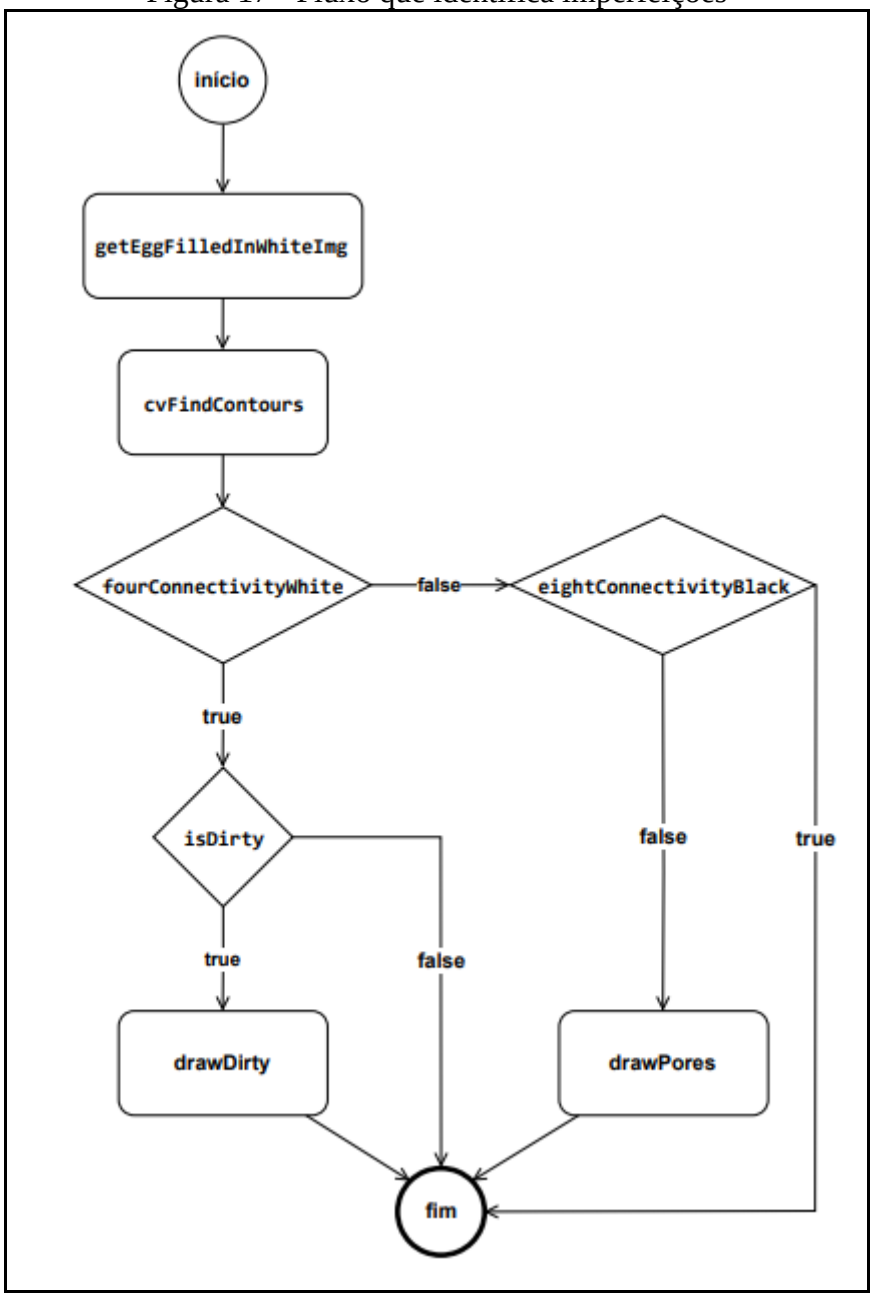
Figura 16 - Extração do canal de saturação



Fonte: Elaborado pelo autor.

A etapa `findImperfections` utiliza como entrada a imagem do canal de saturação. Este método é responsável por identificar a sujeira do ovo, bem como a porosidade da casca. A Figura 17 apresenta o fluxo que identifica as imperfeições do ovo.

Figura 17 - Fluxo que identifica imperfeições



Fonte: Elaborado pelo autor.

Primeiramente na etapa `getEggFilledInWhiteImg` é gerado uma imagem segmentada da imagem cortada do ovo. Essa imagem será utilizada para determinar se os ruídos encontrados estão dentro ou fora do ovo. A imagem segmentada é representada por duas cores, zero para plano de fundo em preto e 255 para o ovo em branco. Na etapa `cvFindContours` serão identificados os contornos. Estes contornos são formados por uma faixa de pontos e para cada ponto encontrado será analisado se ele está dentro ou fora do ovo. A faixa é determinada pelo método `fourConnectivityWhite`, onde serão analisados o ponto original e os quatros pontos adjacentes, sendo eles respectivamente observados na esquerda, em cima, direita e em baixo. Se todos estes pontos estiverem dentro do ovo, significa que é

um ponto identificado em uma região mais interior do ovo. O Quadro 4 apresenta o código fonte.

Quadro 4 - Código fonte do método fourConnectivityWhite

```
1  final int white = 255;
2  final int allWhite = white * 5;
3  final int xLeft = rectangle.x() - offset;
4  final int xRight = rectangle.x() + offset;
5  final int yTop = rectangle.y() - offset;
6  final int yBotton = rectangle.y() + offset;
7  final int middleColor = getColor(img, rectangle.x(), rectangle.y());
8  final int topColor = yTop > 0 ? getColor(img, rectangle.x(), yTop) : 0;
9  final int bottonColor = yBotton <= img.height() ? getColor(img,
   rectangle.x(), yBotton) : 0;
10 final int leftColor = xLeft > 0 ? getColor(img, xLeft, rectangle.y()) :
   0;
11 final int rightColor = xRight <= img.width() ? getColor(img, xRight,
   rectangle.y()) : 0;
12
13 return (middleColor + topColor + bottonColor + leftColor + rightColor
   == allWhite);
```

Fonte: Elaborado pelo autor.

Caso o retorno deste método seja verdadeiro, será verificado por contagens de pixels se a predominância das cores na imagem original cortada são próximas ao amarelo, que é identificado a partir do método `isDirty`, conforme apresentado o Quadro 5.

Quadro 5 - Código fonte que identifica se é uma sujeira

```
1  int dirtyCount = 0;
2  int size = 0;
3  for (int x = 0; x <= rect.width(); x++) {
4     for (int y = 0; y <= rect.height(); y++) {
5        try (CvScalar cvScalar = getCvScalar(out, x + rect.x(), y +
   rect.y())) {
6           double red = cvScalar.get(0);
7           double green = cvScalar.get(1);
8           double blue = cvScalar.get(2);
9           if (red > 130 && green > 130 && blue < 150) {
10             dirtyCount++;
11          }
12          ++size;
13       }
14    }
15 }
16 // Se 1/3 da imagem for suja
17 return dirtyCount > 0 && (size / dirtyCount) > 0.35;
```

Fonte: Elaborado pelo autor.

Caso pelo menos um dos pontos avaliados no método `fourConnectivityWhite` estejam fora do ovo, será chamada a rotina `eightConnectivityBlack`, que analisa o ponto principal e os oito pontos adjacentes a ele, sendo eles respectivamente esquerda, diagonal superior à esquerda, diagonal superior à direita, em cima, direita, diagonal inferior à esquerda, diagonal inferior à direita e em baixo. Se todos estes pontos estiverem no plano de fundo, ou seja, com o valor zero e significa que era um ruído desnecessário, e neste caso será

descartado. O retorno da função neste caso será verdadeira. O Quadro 6 apresenta o código fonte da rotina.

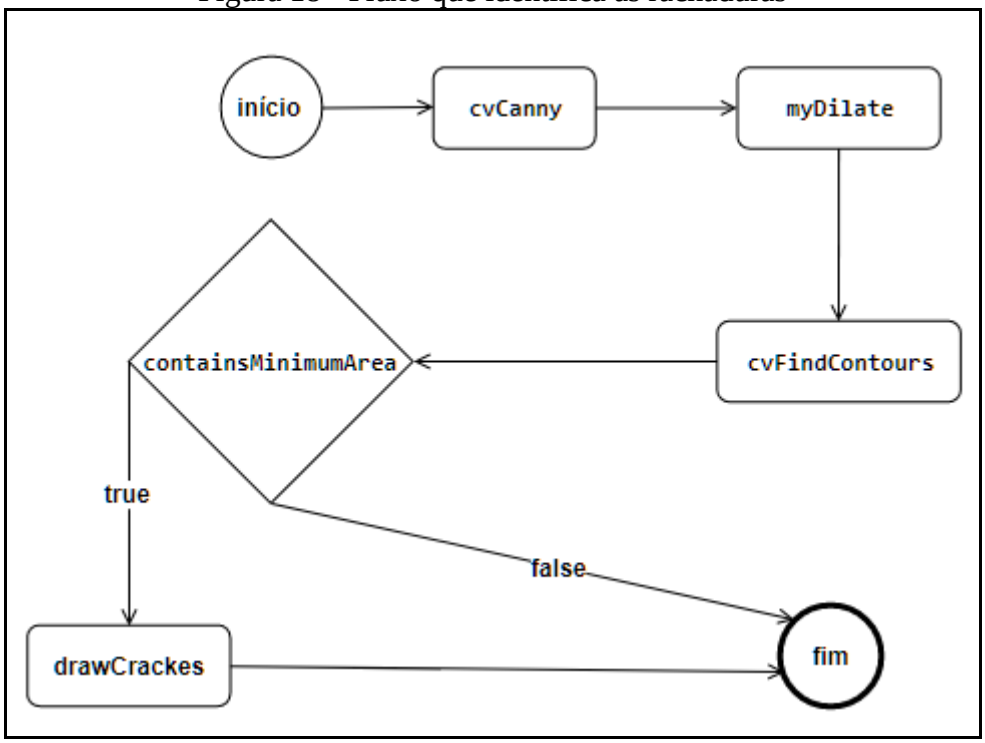
Quadro 6 - Código fonte método eightConnectivityBlack

```
 1  final int height = img.height();
 2  final int width = img.width();
 3  final int xLeft = rectangle.x() - offset;
 4  final int xRight = rectangle.x() + offset;
 5  final int yTop = rectangle.y() - offset;
 6  final int yBotton = rectangle.y() + offset;
 7  final int middleColor = getColor(img, rectangle.x(), rectangle.y());
 8  final int topColor = yTop > 0 ? getColor(img, rectangle.x(), yTop) : 0;
 9  final int leftTopColor = yTop > 0 && xLeft > 0 ? getColor(img, xLeft,
    yTop) : 0;
10  final int rightTopColor = yTop > 0 && xRight <= width ? getColor(img,
    xRight, yTop) : 0;
11  final int bottonColor = yBotton <= height ? getColor(img,
    rectangle.x(), yBotton) : 0;
12  final int leftBottonColor = yBotton <= height && xLeft > 0 ?
    getColor(img, xLeft, yTop) : 0;
13  final int rightBottonColor = yBotton <= height && xRight <= width ?
    getColor(img, xRight, yTop) : 0;
14  final int leftColor = xLeft > 0 ? getColor(img, xLeft, rectangle.y()) :
    0;
15  final int rightColor = xRight <= width ? getColor(img, xRight,
    rectangle.y()) : 0;
16
17  return (middleColor + topColor + leftTopColor + rightTopColor +
    bottonColor + leftBottonColor + rightBottonColor + leftColor +
    rightColor == 0);
```

Fonte: Elaborado pelo autor.

Caso o retorno do método `eightConnectivityBlack` seja falso, é determinado que o ponto avaliado é referente porosidade da casca, que por sua vez será realçada na cor verde. O método `findCrackes` é o responsável por identificar as rachaduras do ovo. A Figura 18 detalha o fluxo de como as rachaduras são identificadas.

Figura 18 - Fluxo que identifica as rachaduras



Fonte: Elaborado pelo autor.

O filtro de Canny no processamento de imagens é responsável pela detecção de bordas. A partir deste filtro é possível encontrar as rachaduras do ovo, em seguida é realizado a dilatação, para fazer com que os pontos encontrados individualmente se aglutinem, formando uma maior componente conexa. A partir dos contornos encontrados, é avaliado se contém uma área mínima, caso haja serão desenhadas as rachaduras na cor azul.

3.4 ANÁLISE DOS RESULTADOS

Nesta seção são apresentados os experimentos realizados com o aplicativo. Na seção 3.4.1 é feita a análise do resultado do reconhecimento das anomalias. Na seção 3.4.2 é analisado a performance do algoritmo. Por fim, na seção 3.4.3 é feita a comparação desse trabalho com os seus correlatos.

3.4.1 Análise dos resultados

Foram analisados 30 ovos distintos, com o intuito de avaliar as seguintes características: presença de sujeira na casca, presença de rachadura na casca e porosidade da casca. Destes ovos selecionados, 25 (83,33%) apresentavam a presença de sujeira e 14 (46,67%) apresentavam a presença de rachadura. O resumo dos resultados é apresentado na Tabela 1.

Tabela 1 - Resultados

Tarefa	**Precisão**
Diagnóstico da presença de sujeira	96,67%
Diagnóstico da presença de rachadura	56,00%
Havia rachaduras e as encontrou	100,00%
Falso positivo na detecção de rachaduras	44,00%
Diagnóstico da qualidade da casca	76,67%
Falsos poros	6,67%

Fonte: elaborado pelo autor.

Conforme os resultados apresentados, é evidente que o diagnóstico da presença de sujeiras obteve sucesso, embora não foi possível destacar com alta precisão todas as sujeiras percebidas a olho nu. Também foi percebido uma dificuldade no diagnóstico da presença de rachaduras, porém em todos os casos onde havia efetivamente rachaduras, elas foram encontradas. A dificuldade na distinção dá-se em virtude da variedade de cores na sujeira e também devido ao sombreamento sobre as sujeiras. A presença de pontos de sangue também confundiu a classificação. O diagnóstico da qualidade da casca obteve um bom resultado, porém em alguns casos ele deu uma nota acima do esperado, em outros, abaixo. A bateria de testes pode ser observada no Apêndice A, onde serão exibidas as imagens de entrada e as imagens de saída do aplicativo.

3.4.2 Avaliação da performance

Foi utilizado como experimento o celular Lenovo K5. Este celular possui um processador Quad-core 1.5 GHz em conjunto com um processador Quad-core 1.2 GHz, possui 2 gigabytes de memória RAM e uma câmera de 13 megapixels. As imagens que foram utilizadas para o processamento possuem a resolução de 3.120 pixels de largura e 4.160 pixels de altura. Conforme a posição da câmera o corte da imagem do ovo que será utilizada para o processamento. Caso o ovo esteja em pé, a resolução em média é de 2.098 pixels de largura e 2.806 pixels de altura. Caso esteja deitado a resolução em média é de 701 pixels de largura e de 524 pixels de largura. A resolução do corte do ovo varia conforme a posição da fotografia em relação a distância do ovo. Foi percebido que quanto menor a resolução da imagem, maior a performance do algoritmo, porém a precisão do algoritmo cai. Nos testes realizados, o algoritmo consumiu entre 1,69 até 24,62 segundos, estas variações se dão conforme a quantidade de elementos para serem analisados na foto. As imagens que apresentam menos imperfeições levaram menos tempo ao se comparar com as imagens que haviam muitas rachaduras e sujeiras simultaneamente, porém foi percebido que o que mais impacta a performance em si é a resolução da imagem.

3.4.3 Comparativo entre o aplicativo desenvolvido e seus correlatos.

A partir das informações obtidas com os trabalhos descritos, foi montado o Quadro 7, com comparação com os trabalhos correlatos.

Quadro 7 - Comparação com os trabalhos correlatos

Característica / correlatos	**Rosa (2017)**	**Machado (2009)**	**Galvão (2013)**	**García (2000)**
Portável para dispositivos móveis	Sim	Não	Não	Não
Aplica reconhecimento de imagem	Sim	Sim	Não	Sim
Avalia espessura da casca	Não	Não	Sim	Não
Avalia sujidades na superfície da casca	Sim	Sim	Sim	Sim
Avalia rachadura na casca	Sim	Sim	Sim	Sim
Avalia externamente pontos de sangue	Não	Sim	Sim	Sim
Avalia internamente pontos de sangue	Não	Sim	Não	Não
Avalia internamente a presença corpos estranhos	Não	Sim	Não	Não

Fonte: Elaborado pelo autor.

A partir do Quadro 7, conclui-se que o trabalho de Rosa (2017) foi o único no qual utiliza dispositivos móveis no processo de inspeção, se preocupou apenas com os aspectos externos da casca, porém não avaliou os pontos de sangue externos. O trabalho de García (2000) preocupou-se somente com a avaliação externa na superfície da casca. Já o trabalho de Galvão (2013) é o único no qual a avaliação é realizada a olho nu e também é o único que realizou avaliação da espessura da casca. Por fim, o trabalho de Machado (2009) procurou atender a maioria dos aspectos internos e externos do ovo.

4 CONCLUSÕES

Com o crescente interesse público pela qualidade dos alimentos disponíveis para o consumo humano, pode-se sugerir que haja demanda para a aplicação de tecnologias na área alimentícia, visando maior certeza na qualidade do mesmo. O mercado avícola não escapa desta tendência, incluindo-se aí os ovos, considerados alimentos altamente nutritivos, de baixo custo e que possibilitam a fabricação de diversos produtos alimentícios (GALVÃO, 2013).

Em relação aos trabalhos correlatos, a solução de Machado (2009) demonstrou-se mais completa na identificação e classificação das características. O trabalho realiza a captura de imagens a partir de uma câmera fotográfica e em seguida as fotografias são transferidas para o computador para serem feitas as análises para detectar defeitos internos e externos.

A proposta do trabalho era de utilizar a biblioteca JavaCV na plataforma Android e este objetivo foi alcançado. Durante o desenvolvimento do trabalho foi percebido que não há documentação online para versão Android. Também foi percebido a ausência de Javadoc na biblioteca. Sendo assim, como alternativa, foi utilizado a documentação da mesma versão desktop da JavaCV. Em alguns casos os trechos de códigos de exemplos da documentação da mesma versão desktop não eram compatíveis, pois na tradução para a interface Java, foi quebrado a compatibilidade das assinaturas dos métodos. As descobertas por soluções compatíveis foram através de experimentos exploratórios realizados testando as variações de assinaturas dos métodos com o mesmo nome declarado e ou similar, em alguns casos os métodos tinham ou não o sufixo cv. Mesmo assim a documentação da JavaCV não contempla todos os exemplos da documentação da OpenCV, sendo que nestes casos deve-se traduzir o exemplo escrito em C para Java.

Na utilização da biblioteca JavaCV deve se ter muito cuidado no gerenciamento de memória, pois como a biblioteca utiliza sub-rotinas nativas escritas em C, é necessário realizar a chamada dos destrutores para que não haja vazamento de memória. Os destrutores das classes da biblioteca podem ser invocados nas implementações no Android realizado da forma explícita, no caso a chamada do método `close` ou declarará-las dentro de um bloco *Try*. Como todas estas classes implementam a interface `Closeable`, permite que a JVM, realize a invocação do método `close` implicitamente.

4.1 EXTENSÕES

Algumas das possíveis extensões para este trabalho são:

a) realizar inspeção do ovo com plano de fundo dinâmico;
b) realizar inspeção em uma caixa de dúzia de ovos;
c) realizar inspeção em ovos da cor marrom;
d) realizar normalização da iluminação da imagem antes de processar;
e) realizar testes com o processamento de imagem reduzindo a escala da foto original, viabilizado ganho de performance, sem prejudicar a precisão;
f) permitir classificação de pontos de sangue externos;
g) aprimorar o reconhecimento de rachaduras na situação de falso positivo;
h) aprimorar o reconhecimento de poros na situação de falso positivo;
i) aprimorar o diagnóstico da qualidade da casca;
j) tratar a situação de quando o ovo sofrer reflexo na clara de ovo colada na casca;
k) viabilizar o aplicativo para uso voltado a usuário final.

REFERÊNCIAS

ALMEIDA, Jorge Mamede. **Embriologia veterinária comparada**. Rio de Janeiro : Guanabara Koogan. 1999. 176 p.

BELUSSO, Diane; HESPANHOL, Antonio Nivaldo. A evolução da avicultura industrial brasileira e seus efeitos territoriais. **Percurso,** Maringá, v. 2, n. 1. 2010. Disponível em: <http://periodicos.uem.br/ojs/index.php/Percurso/article/view/9855/5801> Acesso em: 5 setembro de 2017.

BRASIL. Ministério da Agricultura, Pecuária e Abastecimento. Regulamento da Inspeção Industrial e Sanitária de Produtos de Origem Animal (RIISPOA). **Inspeção Industrial e Sanitária das Ovos e Derivados**, Titulo IX, C. 1. Brasília,1952.

BRASIL. Resolução CIPOA nº 005 de 19 de novembro de 1991. Trata da aprovação de padrões de identidade e qualidade de produtos lácteos e de ovos do Ministério da Agricultura, Pecuária e Abastecimento. **Diário Oficial República Federativa do Brasil**, nº 78, 1991. Brasília / DF.

COBB. **Guia de Manejo de Incubação**. Outubro de 2013. Disponível em: <http://cobb-vantress.com/languages/guidefiles/e420c01f-a164-4890-9963-60c1e332bf40_pt.pdf> Acesso em: 19 de maio de 2016.

COSTA, João Paulo de Oliveira; ORTOLAN, Lucas Dissenha. **Classificação de imagens de sensoriamento remoto utilizando redes neurais artificiais.** 2014. 78 f. TCC (Graduação) - Curso de Sistemas de Informação, Universidade Tecnológica Federal do Paraná, Curitiba, 2014.

COUTTS, J.A.; WILSON, G.C. **Ovos de ótima qualidade - Uma abordagem rápida**. Reino Unido: 5M Publishing. 2007.65p.

CHOCADEIRAS FAISÃO. **Ovoscopia e Controle de Umidade**. [S.l]. Disponível em: <http://chocadeirasfaisao.blogspot.com.br/p/dicas-de-criacao.html> Acesso em: 18 de maio de 2016.

CHOCADEIRAS GOLDEN. **Comércio de Ovoscópios**. [S.l]. Disponível em: <http://www.chocadeirasgolden.com.br/ovoscopio.html> Acesso em: 10 de maio de 2016.

GALVÃO, Julia Arantes. **Ovos produzidos em diferentes sistemas de alojamento: qualidade e segurança microbiológica, parâmetros físicos, validação e utilização de método multiresíduo para detecção de antimicrobianos e pesticidas**. 2013. 105 f. Tese (Doutorado) - Curso de Medicina Veterinária, Universidade Estadual Paulista, Botucatu, Sp, 2013.

GARCÍA, M., C. et al. Automatic rules generation by g.a. for eggshell defect classification. In: EUROPEAN CONGRESS ON COMPUTATIONAL METHODS IN APPLIED SCIENCES AND ENGINEERING, 6, 2000, Barcelona, Spain. **Anais…** Barcelona, Spain: Spanish Association for Numerical Methods in Engineering, 2000. p. 2-6.

GONZALES, Rafael C.; WOODS, Richard E. **Digital image processing**. 3rd ed. Upper Saddle River: Pearson Prentice Hall, 2008.

GUEDES, R. **O ovo e seus aspecto**. Rio de Janeiro: Edições SAI. séries Estudos e Ensaios n° 29. 1961. 156p.

HEEMANN, Samuel. **Agravos à saúde e doenças ocupacionais nos trabalhadores do matadouro-frigorífico de aves de um município do rio grande do sul no ano de 2012.** 2013. 51 f. Tese (Doutorado) - Curso de Especialização em Saúde Pública, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2013.

IBGE - Instituto Brasileiro de Geografia e Estatística. 2016. Disponível em <https://agenciadenoticias.ibge.gov.br/agencia-noticias/2013-agencia-de-noticias/releases/9430-em-2016-producao-de-ovos-e-abate-de-frangos-e-suinos-sao-recordes.html >. Acesso em: 30 de novembro de 2017.

MACHADO, Douglas Silveira. **Sistema de Inspeção Visual Automática Aplicado ao Controle de Qualidade de Ovos em Linhas de Produção**. 2009. 125 f. Tese (Doutorado) - Curso de Modelagem Matemática Computacional, Centro Federal de Educação Tecnológica de Minas Gerais, Belo Horizonte, 2009.

OLIVEIRA, B.L. Processamento e industrialização de ovos. In: SIMPÓSIO GOIANO DE AVICULTURA, 4., 2000, Goiânia - GO. **Anais...** Goiânia - GO, Associação Goiânia de Avicultura, 2000. p. 177-186.

ROSA, Guilherme M. **Egg evaluation: ovoscopia via smartphone**. 2017 Disponível em: <https://bitbucket.org/tecedufurb/guilhermemurilorosa>

SANTINI, G.A.; SOUZA FILHO, H.M. Inovação tecnológica em sistemas agroindustrial: a avicultura de corte no Brasil. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE ECONOMIA, ADMINISTRAÇÃO E SOCIOLOGIA RURAL, 2005, Santa Maria - RS. **Anais...** Brasília - DF: Sociedade Brasileira de Economia, Administração e Sociologia Rural, 2005. p. 1.

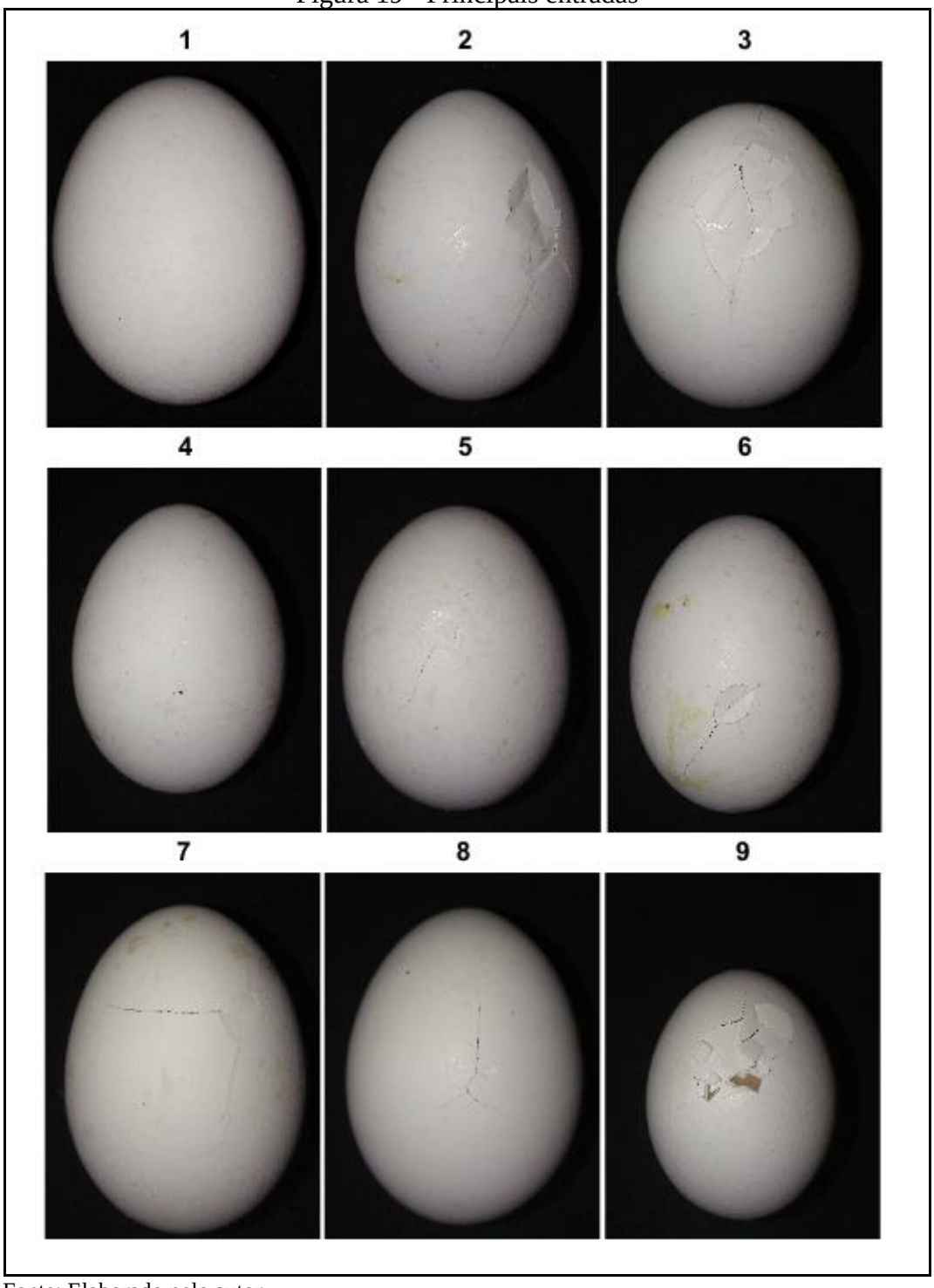
SECRETARIA DE ESTADO DA AGRICULTURA E DO ABASTECIMENTO. **Análise de Conjuntura Agropecuária Avicultura de Postura.** Paraná, 2012. Disponível em: <http://www.agricultura.pr.gov.br/arquivos/File/deral/Prognosticos/avicultura_postura_2012_13.pdf > Acesso em: 5 de setembro de 2017.

SESTI, L.; ITO, N.M.K. **Fisiopatologia do Sistema Reprodutor**. Campinas: FACTA, 2009. p. 315- 80.

APÊNDICE A – Imagens de entrada e de saída na bateria de testes do aplicativo

A seguir serão exibidas algumas entradas, a bateria completa de testes, bem como as imagens em resolução original poderão ser encontradas em Rosa (2017). A Figura 19 apresenta as 9 principais entradas do aplicativo.

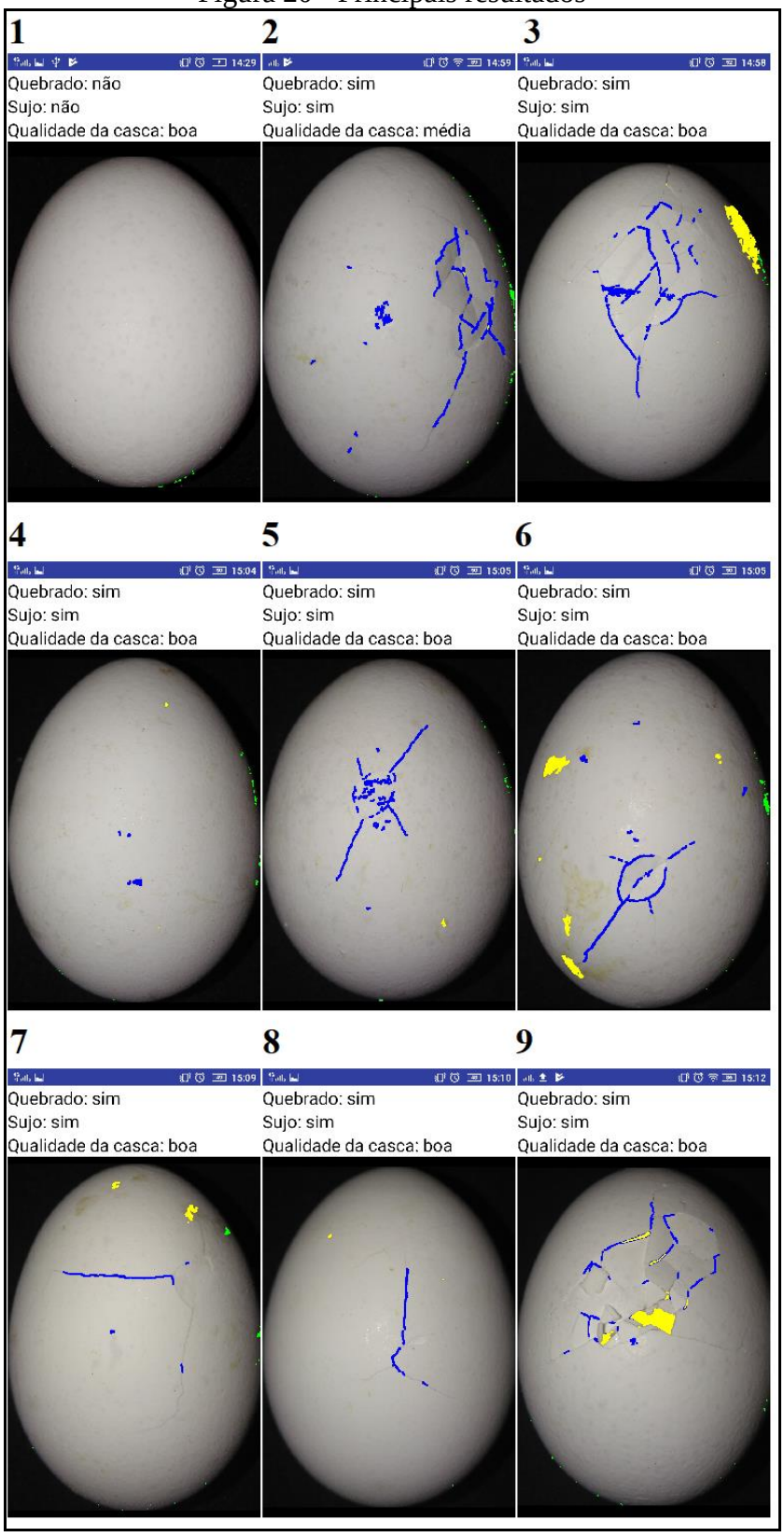
Figura 19 - Principais entradas



Fonte: Elaborado pelo autor.

Por fim serão exibidos os resultados dos aplicativos destas entradas, na mesma ordem, conforme Figura 20.

Figura 20 - Principais resultados



Fonte: Elaborado pelo autor.