UNIVERSIDADE FEDERAL DO RIO GRANDE
CENTRO DE CIÊNCIAS COMPUTACIONAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO
CURSO DE MESTRADO EM ENGENHARIA DA COMPUTAÇÃO

Dissertação de Mestrado

Desenvolvimento de uma Ferramenta Para a Classificação Semi-Supervisionada de Moscas-das-Frutas Usando Smartphones

Geison Quevedo

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande, como requisito parcial para a obtenção do grau de Mestre em Engenharia de Computação

Orientador: Prof. Dr. Vagner Santos da Rosa

Rio Grande, Março de 2017

Ficha catalográfica

Q58d Quevedo, Geison.
Desenvolvimento de uma ferramenta para a classificação semi-supervisionada de moscas-das-frutas usando smartphones / Geison Quevedo. – 2017.
88 p.

Dissertação (mestrado) – Universidade Federal do Rio Grande – FURG, Programa de Pós-graduação em Engenharia da Computação. Rio Grande/RS, 2017.
Orientador: Drª. Vagner Santos da Rosa.

1. Computação 2. Análise de imagens 3. Moscas 4. Classificação semi-supervisionada 5. Smartphones I. Rosa, Vagner Santos da II. Título.

CDU 004

Catalogação na Fonte: Bibliotecário Me. João Paulo Borges da Silveira CRB 10/2130

UNIVERSIDADE FEDERAL DO RIO GRANDE
Centro de Ciências Computacionais
Programa da Pós-Graduação em Computação
Curso de Mestrado em Engenharia de Computação

ATA DE SESSÃO DE DEFESA DE DISSERTAÇÃO DE MESTRADO

Ata No. 11 /2017

Na data de 05 de abril de 2017, às 15:00 horas, ocorreu a Sessão de Defesa de Dissertação de Mestrado de Geison Quevedo, que apresentou a dissertação intitulada *Desenvolvimento de uma Ferramenta Para a Classificação Semi-Supervisionada de Moscas-das-Frutas Usando Smartphones*, realizada sob a orientação do Prof. Vagner Santos da Rosa. A banca examinadora foi constituída pelos Dr. Alessandro de Lima Bicho (FURG), Dr. Rafael da Silva Gonçalves (Doutorado em Fitossanidade pela Universidade Federal de Pelotas em 2016. Atualmente sou PosDoc da Capes/Embrapa) e Dr. Ricardo Nagel Rodrigues (FURG), sob a presidência do orientador. Após a apresentação do trabalho, a banca arguiu o candidato e, a seguir, deliberou pela

( ) aprovação da Dissertação
(X) aprovação da Dissertação, sugerindo modificações no texto
( ) reprovação da Dissertação

Rio Grande, 05 de abril de 2017

Prof. Dr. Alessandro de Lima Bicho

Dr. Rafael da Silva Gonçalves

Prof. Dr. Ricardo Nagel Rodrigues

Prof. Dr. Vagner Santos da Rosa
Orientador(a)

AGRADECIMENTOS

Gostaria de agradecer ao meu orientador Vagner Santos da Rosa por toda sua tutoria, disponibilidade, pela sua grande capacidade de motivar e enxergar o potencial de cada aluno. A todos os meus amigos e familiares que me acompanharam estes anos e que de alguma forma me deram apoio durante esta fase completada.

A menos que modifiquemos a nossa maneira de pensar, não seremos capazes de resolver os problemas causados pela forma como nos acostumamos a ver o mundo.
— Albert Einstein

Resumo

QUEVEDO, Geison. **Desenvolvimento de uma Ferramenta Para a Classificação Semi-Supervisionada de Moscas-das-Frutas Usando Smartphones**. 2017. 88 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande, Rio Grande.

O cultivo de árvores frutíferas desempenha um papel importante na economia da região sul do estado do Rio Grande do Sul, porém pragas são uma constante ameaça que causam perdas econômicas significativas. Um dos principais danos encontrados nas frutíferas da região sul do estado do Rio Grande do Sul, são danos causados pela moscas-das-frutas *Ceratitis capitata* e *Anastrepha fraterculus*, no qual o dano causado é exclusivamente no fruto. Uma das maneiras de monitoramento da mosca-das-frutas é ter conhecimento da existência das moscas na plantação através do uso de armadilhas, onde a mosca é atraída por atrativos e posteriormente identificada. Logo após, os insetos capturados são identificados (espécie) e contados manualmente. O conteúdo da armadilha é descartado ao final do processo. Neste trabalho é proposto um método para registrar o conteúdo das armadilhas e automatizar a contagem das moscas-das-frutas com a identificação por meio da análise de imagens digitais. Este processo é realizado através da aquisição de imagens das moscas coletadas.

Palavras-chave: mosca-das-frutas, contagem, visão computacional, identificação.

Abstract

QUEVEDO, Geison. **Developing a Tool for a Semi-Supervised Network of Fruit Flies Using Smartphones.** 2017. 88 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande, Rio Grande.

The cultivation of fruit trees plays an important role in the economy of the southern region of the state of Rio Grande do Sul, however a constant threat that causes significant economic losses. One of the main damages found in the regions of the state of Rio Grande do Sul is caused by fruit flies of the species *Ceratitis capitata* and *Anastrepha fraterculus*, in which the damage caused is exclusively on the fruit. One of the ways of monitoring fruit fly is a knowledge of the existence of flies in the plantation through the use of traps, where a fly is attracted by attractiveness and subsequently identified. Soon after, the captured insects are identified (species) and counted manually. The contents of the trap are discarded at the end of the process. This work is proposed as a method to record the content of traps and to automate a fruit flies count with an identification through the analysis of digital images. This process is accomplished by the acquisition of contented sieves images with the collected flies.

Keywords: fruit fly, counting, computer vision, identification

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
BSD	*Berkeley Software Distribution*
DL	*Deep Learning*
HSV	*Hue (matiz), Saturation (saturação) e Value (valor)*
IDE	*Ambiente de Desenvolvimento Integrado*
OpenCL	*Open Computing Language*
OpenCV	*Open Source Computer Vision Library*
PDI	*Processamento Digital de Imagens*
RGB	*Red, Green and Blue*
SDK	*Software Development Kit*
SoC	*System on Chip*
VC	*Visão Computacional*
XML	*Extensible Markup Language*

1 INTRODUÇÃO

A busca na utilização de novas maneiras e de métodos de visão computacional já ocorre há muito tempo. Uma das primeiras aplicações foi idealizada no início do século XX, no qual consistiu em criar um sistema de transmissão de imagens via cabo submarino, através do oceano Atlântico, de Londres para New York. Este sistema foi desenvolvido por Bartlane que na época reduziu o tempo de envio da imagem de mais de uma semana para menos de três horas (GONZALEZ; WOOD, 2000).

A partir deste marco potencializou-se a busca por métodos de visão computacional no qual cresceu consideravelmente, assim, como a inovação e o surgimento de novos métodos e algoritmos. Tais recursos podem ser aplicados nas mais diversas áreas, como a médica, robótica e agrícolas. A visão computacional aplicada em pesquisas na área da engenharia agrícola teve seu início no final de 1970. Suas aplicações estão voltadas principalmente à inspeção e classificação de qualidade de produtos, ao monitoramento no crescimento de cultivares, à identificação de doenças e pragas, entre outros (WEIZHENG et al., 2008).

Os sistemas produtivos agrícolas têm sido beneficiados pela incorporação de avanços tecnológicos primeiramente destinados a setores industriais trazendo assim a mecanização, juntamente com esse avanço surgiram novos fertilizantes sintetizados agregando valor para a agricultura. A era da tecnologia contribuiu largamente para a automação e engenharia genética. A era da informação agregou novas maneiras de potencializar os avanços tecnológicos integrando tecnologia com à agricultura de precisão (LECHETA, 2013).

Com o avanço da tecnologia os profissionais, principalmente, ligados a atividades de campo não podem mais realizar trabalhos triviais ou que uma máquina poderia realizar, devem eles seguir o avanço da tecnologia e fazer uso dos dispositivos moveis no meio do campo.

A partir de desta perspectiva surgiu o interesse em estabelecer contatos e buscar pelo conhecimento que vai além da área da computação, na procura por profissionais de outras áreas. Surgiu-se então o interesse de dar início ao desenvolvimento de uma aplicação para smartphones para auxiliar produtores rurais na gestão do agronegócio e através do monitoramento da praga das moscas-das-

frutas. Sendo assim, este trabalho tem por objetivo auxiliar no processo de monitoramento da pra das moscas-das-frutas, atuando na identificação e contagem das moscas-das-frutas através das armadilhas McPhail.

Considerando que as moscas-das-frutas capturadas nas armadilhas McPhail serão expostas em peneiras para realizar sua contagem, é importante compreender que o processo atual de contagem das moscas-das-frutas será adaptado. Na *Figura 1* podemos ver o processo atual e suas modificações durante o escopo do projeto. No primeiro bloco da Figura 1 pode-se ver o processo atual de contagem das moscas-das-frutas (detalhado na seção 2.4). Na sequência, podemos ver no segundo bloco a primeira alteração proposta neste trabalho, no qual optou-se em não realizar contagem das moscas em peneiras convencionais e sim em uma peneira com revestida com uma malha de algodão. No terceiro bloco da Figura 1, podemos ver no final do processo de contagem das moscas a peneira revestida com a malha de algodão juntamente com tela do aplicativo no qual fará a contagem das moscas e sua identificação automaticamente. Este trabalho contempla explicar o processo atual de contagem das moscas-das-frutas e apresentar uma proposta de alternativa para contagem das moscas-das-frutas utilizado um aplicativo e algoritmos de visão computacional.

Figura 1 Processo atual e suas adaptações.

Para melhor compressão deste trabalho é importante saber sobre o processo de monitoramento utilizado e em qual etapa do processo o aplicativo e algoritmos serão aplicados.

1.1 Objetivo

Este trabalho tem como objetivo identificar algoritmos de processamento de imagens e desenvolver um software para smartphones para realizar a contagem e identificação das moscas-das-frutas das espécies Ceratitis capitata e Anastrepha. Fraterculus.

Objetivos Específicos:

- Definir a amplitude do espectro de cor no espaço HSV com os tons de cor do corpo das moscas *C. capitata* e *A. fraterculus*;
- Realizar a contagem das moscas-das-frutas
- Realizar a identificação das moscas-das-frutas de forma semi-supervisionada;
- Desenvolver uma aplicação para smartphone para realizar os processos de contagem das moscas-das-frutas e identificação das espécies *C. capitata* e *A. fraterculus* através da visão computacional.

1.2 Estrutura Do Texto

O Capítulo 1 introduz o problema da praga das moscas-das-frutas e a importância do controle desta praga, assim como objetivos e recursos utilizados, contextualizando assim, o problema tratado neste trabalho. Na sequência, o Capítulo 2 apresenta uma revisão sobre a praga das moscas-das-frutas para melhor compreensão deste trabalho, assim como uma revisão teórica sobre visão computacional necessária para o entendimento das metodologias adotadas neste trabalho na busca de resolver o problema de visão. Em continuação, o Capitulo 3 são apresentados trabalhos relacionados de forma objetiva contemplando seus resultados e metodologias para detecção de moscas-das-frutas segundo sua espécie e contagem de insetos e outros objetos de estudo importantes para compressão e comparação podendo assim fundamentar as escolhas tomadas durante o projeto.

No Capítulo 4 é detalhado as metodologias e métodos utilizados para o desenvolvimento da criação de um modelo matemático capaz de realizar a identificação da praga de forma semi-supervisionado e supervisionado.

O Capítulo 5 apresenta os principais resultados do trabalho, abordando eficiência dos modelos testados e principais problemas encontrados durante a fase de desenvolvimento do aplicativo. Por fim, o Capítulo 6, apresenta uma conclusão com base nos resultados obtidos. Para finalizar este capitulo fica registrado algumas melhorias e avanços como trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será descrito a importância da praga moscas-das-frutas, bem como a fundamentação teórica de Visão Computacional e Processamento Digital de Imagens necessária para a compreensão deste trabalho.

2.1 Praga mosca-das-frutas

Entre as principais pragas que atingem a fruticultura em nível mundial encontramos a praga das moscas-das-frutas, Estas, causam grandes danos à produção quando medidas de controle populacional não são tomadas.

Um dos principais danos ocorridos pelas moscas-das-frutas é provocado devido à oviposição, no qual a mosca perfura os frutos causando o desenvolvimento das larvas danificando o interior, onde inicialmente não pode ser visto nenhum sinal na epiderme do fruto. Outro dano comum é que após esse processo, a epiderme fique danificada, possibilitando a entrada de microrganismos, causando o apodrecimento do mesmo (CARVALHO, 2005).

2.2 Custo da Praga da Mosca-das-frutas

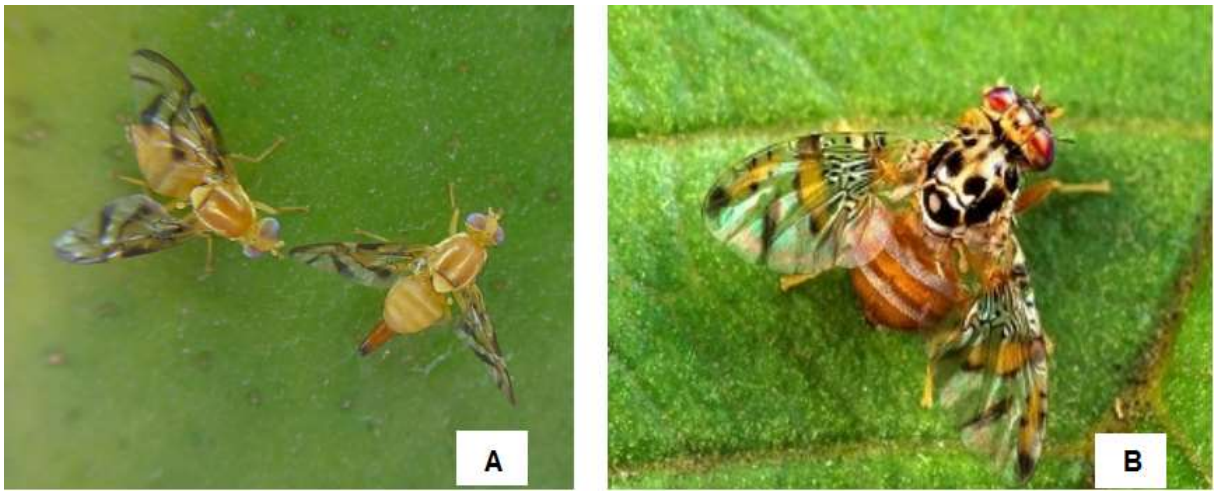
Os danos causados pelas moscas-das-frutas não se dão apenas pelo apodrecimento e ao gasto envolvido no controle da praga. O principal custo está diretamente ligado às barreiras quarentenárias na exportação da fruta para Japão e Estados Unidos, no qual impões medidas de controle e tratamento para os países que apresentam ocorrência desta praga.

A Organização das Nações Unidas para a Alimentação e a Agricultura (FAO) contabiliza uma perda de mais de US$ 1,7 bilhão por ano, dos quais 10% ocorrem no Brasil (FAO, 2016). A fruticultura brasileira vem apresentando aumento em sua produção para o mercado interno e externo. Em 2012 o valor das exportações de fruta foi de U$ 910 milhões, sendo o Brasil o terceiro maior produtor de frutas, produzindo 42 milhões de toneladas.

2.3 Principais pragas na fruticultura no Sul do Brasil

A mosca *A. Fraterculus* é considerada uma das principais pragas para a fruticultura na região Sul do Brasil, encontrada frequentemente no estado do Rio Grande do Sul onde juntamente com a *C. capitata* tornam-se um dos principais problemas relacionados ao uso de agrotóxicos. Ao longo de dez anos pode-se verificar que as maiores incidências de captura de moscas-das-frutas ocorreram nos meses de novembro ao final de janeiro principalmente em pomares de macieira e pessegueiro (SALLES; KOVALESKI, 1990).

Figura 2 *Anastrepha fraterculus* (A) e *Ceratitis capitata* (B).



Fonte: PARANHOS, B. A. J, 2008.

2.3.1 Anastrepha fraterculus

A mosca-das-frutas *A. fraterculus* também conhecida como sul-americana, desenvolve-se em 109 plantas hospedeiras de 265 espécies pertencentes a 59 famílias (ZUCCHI, 2008). Os adultos da *A. fraterculus* medem aproximadamente 6,5 mm de comprimento, e possuem coloração amarela, tórax marrom.

2.3.2 Ceratitis capitata

A mosca *C. capitata* também conhecida como mosca-do-mediterrâneo é nativa da África equatorial, e já pode ser encontrada em todos os continentes (METCALF et al., 1962), sendo considerada a espécie de moscas-das-frutas mais nociva. Em

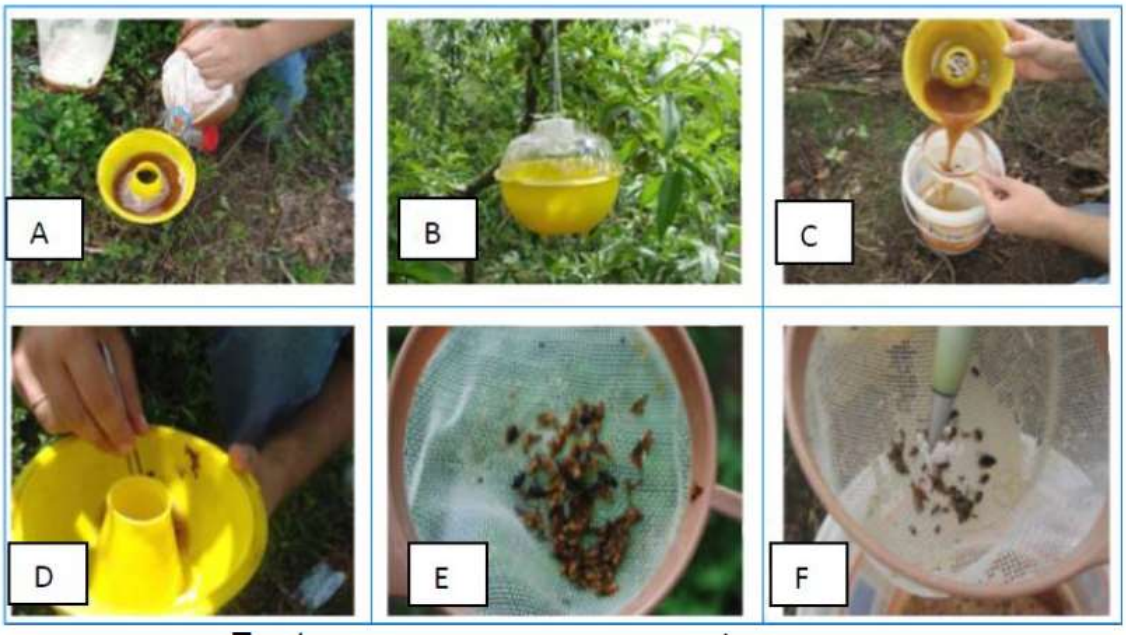
Pelotas RS, sua presença em níveis significativos foi registrada pela primeira vez por Nunes et al. (2012) infestando frutos de caquizeiro, goiabeira e araçazeiro.

A mosca em fase adulta mede entre 4 a 5 mm de comprimento e chega de 10 a 12 mm de envergadura; tem coloração predominantemente amarelo escuro, olhos castanho-violáceos, tórax preto e abdome amarelo escuro.

2.4 Controle de Pragas na Agricultura

Tanto nos locais onde não existe o registro da praga moscas-das-frutas como em regiões em que se realizam monitoramento para controle, são necessários métodos confiáveis para medição e mensurar as populações das moscas, para se conhecer sua densidade populacional, tornando possível verificar sua ausência ou baixa prevalência. Desta maneira o controle e monitoramento de pragas têm agregado experiências de diferentes países e vem demonstrando que é mais eficiente e econômico determinar a presença da mosca-das-frutas como inseto adulto (BRAGA SOBRINHO; MALAVASI; OMETO, 2001). Para garantir um controle eficiente de uma determinada espécie de praga, deve-se ter o conhecimento da sua flutuação populacional, pois com o conhecimento das épocas de maior ocorrência de uma determinada praga é possível realizar o planejamento de estratégias mais eficazes para seu controle. Um dos métodos para conhecer a flutuação populacional das moscas-das-frutas se dá através de contagem das moscas-das-frutas em armadilhas do tipo MCPhail, ocorre como demostrado na Figura 3, onde o responsável pela contagem vai até o local onde está a armadilha e leva com ele um balde e peneira. Este processo é iniciado adicionando um atrativo na armadilha e fixando a armadilha à árvore como pode ser visto na Figura 2 (A, B). Para verificar a existência da praga, deve ser feito a contagem que consiste em abrir a armadilha e filtrar os insetos capturados em uma peneira, derramando o atrativo no balde Figura 2 (C), caso alguma mosca fique presa na armadilha esta é removida e colocada junto às demais moscas na peneira Figura 2 (D), tendo realizado esta etapa, a contagem e identificação podem ser feita com ajuda de uma pinça caso haja necessidade.

Figura 3 Método manual da contagem das moscas-das-frutas.



Fonte: Marcos Botton e Dori Edson Nava[1]

Sabe-se que a armadilha McPhail é eficaz para controle das mosca-das-frutas para região sul do Rio Grande do Sul. Este tipo de armadilha é feito de plástico transparente com uma abertura circular em sua base para a entrada das moscas (Figura 4). Na parte basal é adicionado o líquido contendo proteína hidrolisada, vinagre ou outro atraente com base de suco de frutas. A captura se baseia no fato de que a mosca é atraída pelo odor da proteína e ao entrar na armadilha cai no líquido e morre. O seu raio de atração é mais limitado do que o das armadilhas contendo feromônio (BRAGA SOBRINHO; MALAVASI; OMETO, 2001).

[1] Imagens obtidas do folder *Recomendações para o monitoramento e controle da mosca-das-frutas sul-americana em pessegueiro* através do site https://ainfo.cnptia.embrapa.br/digital/bitstream/item/54689/1/folder-monitoramento-e-controle-da-mosca.pdf

Figura 4 Armadilha McPhail.

Fonte: Elaborado pelo autor

2.5 Processamento de Imagens

Nesta sessão será descrito o referencial teórico base referente ao PDI e suas subáreas necessários para compreensão do atual trabalho.

2.5.1 Visão Computacional

Para Hutchinson, Hager e Corke (1996), a visão computacional é definida como uma ciência, onde é possível as máquinas enxergar, possibilitando-as a realização de tarefas como, por exemplo, o reconhecimento de imagens. Na visão de Gonzalez e Woods (1992), a visão computacional é definida como um processo que produz através de uma imagem de algo real, algum tipo de informação. É importante considerar que a principal característica que um sistema de visão computacional deve possuir é o recurso e a capacidade de identificação a partir de uma imagem (GONZALEZ; WOODS, 1992).

2.5.2 Espaço de Cor

As cores possuem características importantes às quais permitem que uma imagem possa ser segmentada. Os seres humanos podem perceber algumas cores que procedem da natureza da luz refletida pelos objetos, devido à luz visível ser

composta por uma banda de frequência estreita no espectro de energia eletromagnética. Um objeto que reflete a luz com certo balanço entre os comprimentos de onda visíveis ao olho humano percebido como branco, porém, se caso este objeto reflita apenas uma faixa limitada do espectro visível, no qual será visto com diferentes tons de cores (GONZALEZ; WOODS, 2000).

A superfície da retina do olho humano possui receptores discretos de luz, chamados de bastonetes e cones. O número de cones que existe em cada um dos olhos é de aproximadamente de 6 a 7 milhões de cones, que possuem maior concentração na região central da retina e são sensíveis a cor.

A existência de bastonetes é muito maior, chegando aproximadamente entre 75 e 150 milhões, e se encontram espalhados sobre a superfície da retina e permite dar uma visão geral do campo, os bastonetes não estão relacionados a visualização de cores, porem são muito sensíveis a níveis de luminosidade.

Os cones possuem em três categorias de sensoriamento, na qual correspondem aproximadamente ao vermelho, verde e azul. Da totalidade dos cones, 65% deles são sensíveis aos comprimentos de onda caracterizados como vermelho, 33% são sensíveis aos de comprimentos de onda de cor verde e apenas 2% são sensíveis à cor azul, sendo estes últimos mais sensíveis. Para que seja possível trabalhar de maneira matemática com as cores visualizadas, devem-se utilizar sistemas que trabalham de forma semelhante com o modo de captação do olho humano, sistemas estes chamados de espaços de cor. A visão computacional frequentemente utiliza a identificação de cores em conjunto com outras propriedades para mapear áreas específicas da imagem (SHIVAKUMAR et al., 2002).

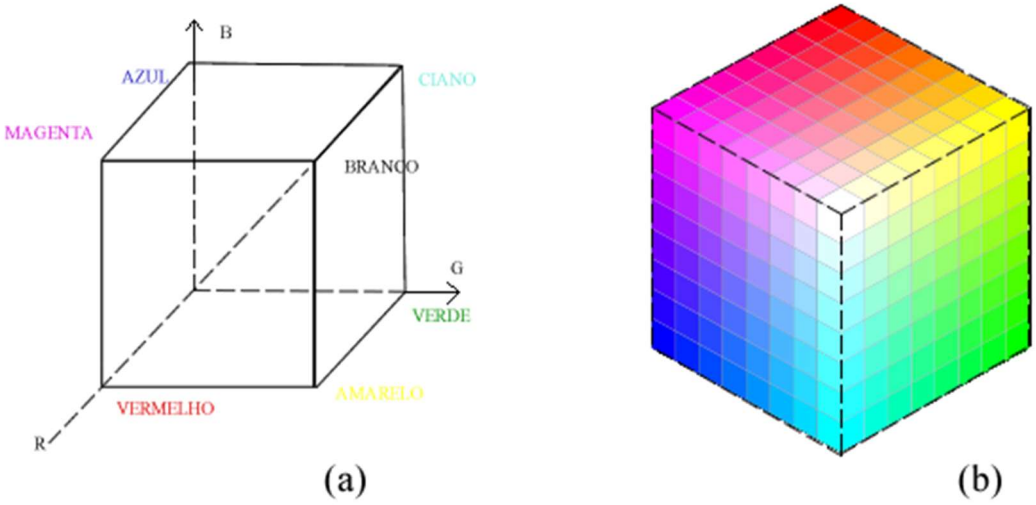
Cada pixel de uma imagem digital colorida pode representar um tom dentre milhares de tons e intensidades de inúmeras cores perceptíveis aos olhos humanos. Um dos modelos mais utilizados está o RGB. Através deste, é possível transformá-lo em qualquer outro espaço de cor. Nas subseções a seguir, serão apresentados os espaços de cores utilizados durante o processo de reconhecimento das moscas-das-frutas: RGB e HSV.

2.5.2.1 Modelo de Cor RGB

O modelo de cor RGB é representado por coordenadas 3D. Na *Figura 5* (a), é exibido o modelo RGB e na *Figura 5* (b), pode-se ver o cubo de cores RGB. Nas coordenadas cartesianas estão às cores primárias (vermelha, verde e azul) e nos

outros vértices estão às cores secundárias (ciana, magenta e amarela). Um ponto entre o plano azul e o plano verde, que seja equidistante, define a cor ciana. O ponto quanto mais próximo da origem mais escuro se torna. Nos planos entre as cores azul e vermelha, um ponto, que seja equidistante, define a cor magenta. Um ponto nos planos entre as cores vermelha e verde, que seja equidistante, define a cor amarela. Na origem, localiza a cor preta e, no seu vértice oposto, a cor branca. Os níveis de cinza ficam na diagonal do cubo entre a cor preta e branca. Uma imagem colorida, representada pelo modelo de cor RGB, constitui-se de três matrizes de M x N, onde N e M referem-se à resolução horizontal (linha) e vertical (coluna) da imagem. Cada matriz correspondente os componentes espectrais, vermelha (R - Red), verde (G - Green) e azul (B - Blue).

Figura 5 (a) Modelo de cor RGB, (b) Cubo de cores RGB.



Fonte: Elaborado pelo autor.

Apesar de o modelo RGB ser frequentemente utilizado, ele não representa o melhor modelo quando se faz necessário a análise das cores para a seleção de áreas de interesse dado que não é um sistema onde o espaço de cor é uniforme (LAGANIÈRE, 2011). Para essas aplicações existem melhores resultados ao se utilizar modelo de cores que separem os componentes de iluminação, matiz e saturação, como o HSV.

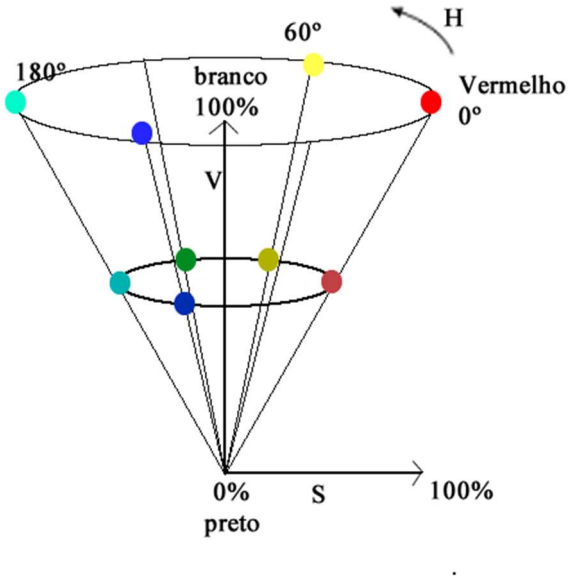
2.5.2.2 Modelo de Cor HSV

O modelo de cor HSV é baseado na percepção da cor pelos seres humanos, e é representado em forma de cone, como pode ser visto na Figura 6.

No modelo HSV, a cor pode ser separada em três componentes distintos: matiz (H-*Hue*), saturação (S-*Saturation*) e valor (V-*Value*). A matiz é o componente que corresponde a cor distinguindo-se uma cor de outra através do seu valor e podem variar em um eixo de 0º a 360º graus, onde esses valores são representados pelos ângulos ao redor do eixo vertical do cone.

A saturação determina o quanto a cor é pura, ou seja, quanto menor esse valor, mais com tom de cinza aparecerá a cor e quanto maior o valor, mais pura é a cor podendo variar seus valores de 0% a 100%. As cores mais apagadas possuem a saturação com valores mais próximos ao 0%. A componente valor (*Value*) define a intensidade do brilho da cor. Os valores do brilho estão no eixo vertical do cone, e podem variar de 0% a 100% (CANDEIAS e SILVA, 2004).

Figura 6 Representação cônica do espaço HSV.



Fonte: Elaborado pelo autor.

2.5.3 Segmentação

Segundo Szeliski (2011), o processo de segmentação de uma imagem compreende em dividi-la em regiões com atributos ou características semelhantes com o objetivo principal de tornar possível serem rotuladas cada região encontrada.

Para Forsyth e Ponce (2002), busca-se obter sub-imagens agrupando, normalmente, pixeis com valores próximos na escala de cinza.

Atualmente a literatura apresenta vários métodos de segmentação, apresentando métodos mais simples no qual realizam apenas operações pixel-a-pixel, e apresentando métodos mais complexos cujos algoritmos procuram compreender a imagem por modelos matemáticos. Apesar da contínua evolução nas técnicas de segmentação, ainda não há um consenso da melhor técnica para esta tarefa, principalmente tratando-se do uso de imagens de ambiente externo sem controle de iluminação (PAL; PAL, 1993; TOYAMA et al., 1999). Segundo Pal e Pal (1993) não existe um único método que seja ideal para lidar com todos os tipos de imagens.

2.5.4 Operadores morfológicos

A palavra morfologia vem da área da biologia significa estudo forma, no entanto, na visão computacional é utilizada a morfologia matemática. Para realizar o processamento morfológico em imagens, é necessário previamente realizar a aplicação da binarização na imagem. Os principais operadores morfológicos utilizados são a erosão e a dilatação, com o uso destes operadores é possível alterar a forma das imagens (FACON, 1996).

Segundo Ghods (2016), a erosão é utilizada no processo de contagem da mosca branca, a fim de remover pequenos objetos e reduzir o número de falsas moscas, o mesmo operador de erosão é utilizado neste trabalho, com a finalidade de tirar pequenas imperfeições da imagem e remover pequenos objetos encontrados nas imagens.

Esse processo pode ser considerado uma transformação morfológica que combina dois conjuntos A e B usando vetores de subtração. A erosão de “A” por “B”, indicada por $A \ominus$ B é definida por:

$$A \ominus B = \{Z \vee (B)z \subseteq A\} \quad (1)$$

Assim, a equação indica que a erosão de A por B é o conjunto de todos os pontos Z de forma que B, transladado por Z, está contido em A. Com o uso da erosão, objetos pequenos tendem a sumir. Outro processo comumente utilizado é o de dilatação. Kumara (2012), demostra o uso da dilação em seu experimento, após remover os objetos indesejados com o operador morfológico de erosão é então aplicado o operador de dilatação para manter o tamanho original dos objetos que não foram removidos. A dilatação combina dois

conjuntos A e B usando a adição vetorial, ampliando os objetos da imagem. A dilatação de A por B é definida como:

$$A \oplus B = \{Z \mid (B')_z \cap A\} \quad (2)$$

O símbolo $\oplus$ indica a operação de dilatação. A equação é baseada na reflexão de B em torno da sua origem, seguida da translação dessa reflexão por Z. Desta forma, a dilatação de A por B é o conjunto de todos os deslocamentos, Z, de forma que B' e A se sobreponham no mínimo por um elemento. Com esta operação, objetos pequenos tendem a ser unidos, fazendo que a área dos objetos aumente. Esse processo pode ser utilizado para restaurar imagens, como por exemplo, linhas de uma faixa de pedestres apagadas parcialmente (GONZALEZ, 2010).

2.5.5 Processamento de Histograma

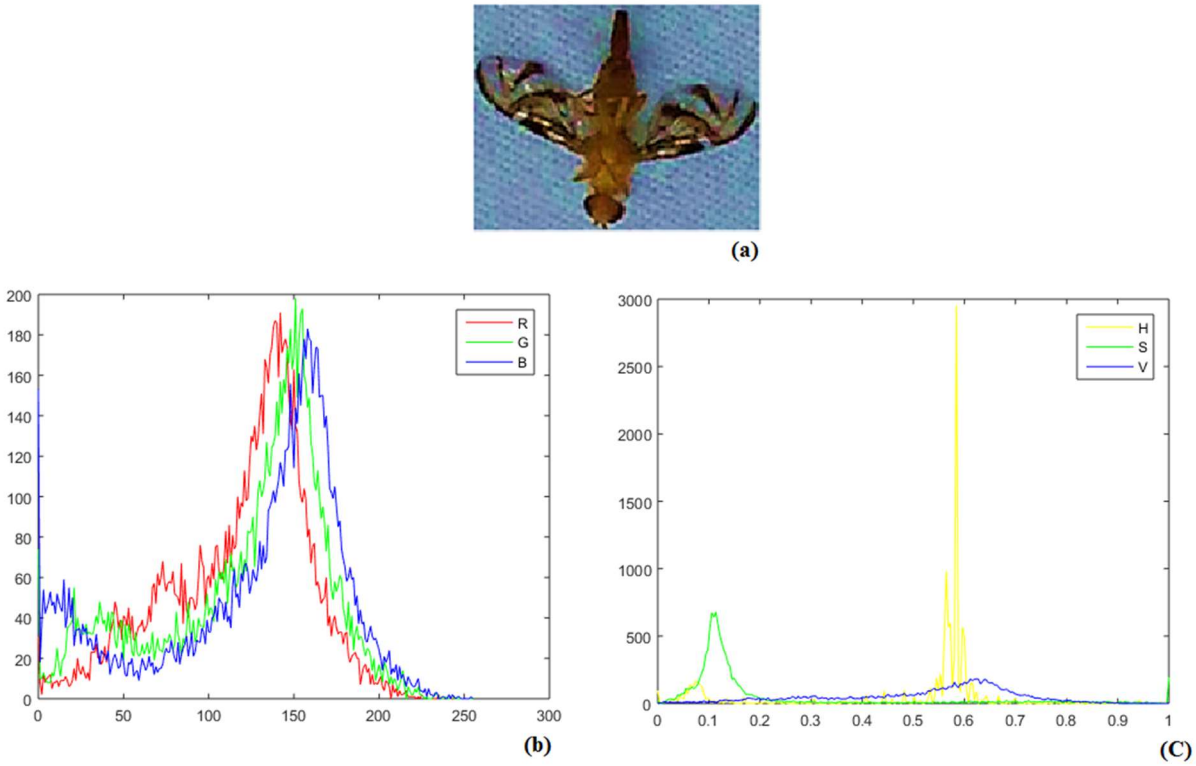
O histograma é uma ferramenta que permite exibir como está distribuída a frequência das intensidades luminosas, permitindo assim representar a quantidade que casa intensidade de cor se repete na imagem. Desta forma é possível proporcionar uma melhor compreensão da imagem, permitindo analisar os parâmetros como luminosidade e contraste através da imagem (SALOMON, 2013).

Em uma imagem em níveis de cinza, o histograma é capaz de identificar a percentagem da intensidade dos pixeis em um determinado espaço de cor. A intensidade e qualidade dos níveis de contraste e a intensidade de brilho podem ser visualizados em um gráfico de barras.

Outro recurso derivado do histograma é a comparação por meio de histogramas. Sua eficiência é descrita por Zhu e Zhang (2010), no qual demostraram em seu trabalho a possibilidade de realizar a identificação de insetos através da comparação de histogramas, realizando a transformação de uma imagem colorida representada no espaço de cor RGB para HSV, para minimizar a influência das variações de iluminação, foram utilizados apenas a tonalidade e saturação do componente.

A Figura 7 ilustra uma imagem colorida e seu respectivo histograma em três níveis de cores, vermelho, verde e azul.

Figura 7 Imagem original (a), histograma de cada canal RGB (b) e histograma em HSV (c).



Fonte: Elaborado pelo autor

2.6 Deep Learning

Houve recentemente o ressurgimento do modelo de *Deep Learning* no qual foi desencadeado pelos trabalhos sobre as representações de aprendizagem e outros modelos mais tradicionais (HINTON et al., 2012).

As arquiteturas de *Deep Learning* resolvem problemas que exigem funções extremamente complexas e geralmente envolvem um conjunto de dados que em grande parte dos casos não é rotulada, onde por si só a rede é capaz de aprender sozinha.

Segundo Bengio (2009), para lidar com isso, os métodos de *Deep Learning* (DL) aprendem hierarquias de recursos com características de níveis mais altos de hierarquia formados por uma composição de características de nível inferior.

O DL então é capaz de aprende comportamentos complexos com conjuntos de informações expansivas no qual pode selecionar características efetivas automaticamente por estruturas de redes neurais em várias camadas profundas.

Esses modelos alcançam tais metas, adotando camadas não supervisionadas, bem-sucedidas por atualizações, aplicando o ensino de aprendizagem aos dados de sinal (KIM, CHOI, LEE, 2015).

Ding e Taylor (2016), em seu experimento puderam realizar o monitoramento do número de mariposas em um sistema de manejo de pragas baseados em feromônio. Em seu trabalho, foi proposto um sistema de detecção automática baseado na aprendizagem profunda para a identificação e contagem de pragas em pontos de captura de imagem dentro de armadilhas em campo. Aplicado a um conjunto de dados de mariposas, o método proposto apresenta um desempenho promissor tanto qualitativo como quantitativamente.

Toda via observando os resultados dos testes, seus melhores resultados obtiveram uma taxa de acerto de 82% a 92% em seus métodos. Apesar do resultado não chegar a 100% de acerto, este é um método promissor para certas áreas de monitoramento de pragas.

2.7 Aprendizado Supervisionado e Não Supervisionado

Dado um classificador e um modelo adequado para um determinado problema, um conjunto de treinamento é realizado a fim de estimar os melhores parâmetros para o classificador. Uma das principais tarefas após encontrar os melhores parâmetros e características de uma imagem é realizar a classificação do objeto atribuindo ao objeto uma categoria (JAIN, 1989).

Para Duda et al. (2000), a aprendizagem é a aplicação do algoritmo com a finalidade de reduzir o erro do conjunto de treinamento, no qual podem ser classificados em dois tipos básicos de aprendizado: supervisionado e o não supervisionado.

2.7.1 Aprendizado Supervisionado

Para o aprendizado supervisionado utiliza-se um rótulo para cada conjunto de amostra a fim de ajustar os parâmetros de classificação podendo assim reduzir os erros de treinamento. Para que ocorra uma classificação supervisionada, necessita-se de uma amostra representativa de cada classe a ser identificada. Nestas amostras devem estar contidas as informações de comportamento médio das classes, estas amostras podem ser denominadas como pixel de treinamento do sistema. Para que ocorra um treinamento eficaz,

é essencial uma boa seleção de amostras e que garantam a homogeneidade (COSTA, CESAR Jr., 2001).

Dentre as técnicas de classificação supervisionadas podemos ter dois métodos, distribuição livre e estatística.

Distribuição livre: neste método não é preciso nenhum tipo de conhecimento, visto que funções de probabilidade são autossuficientes para este método. Pode ser entendida quando existem K diferentes classes de padrões definidas por S^1, S^1, ..., S_k, cada classe caracterizada por apresentar M_k diferentes protótipos em uma imagem. Uma das funções essenciais de reconhecimento de padrões é chamada função discriminante, no qual dividirá o espaço em k diferentes regiões, onde cada região apresentará padrões similares.

Classificação estatística: este método utiliza modelos de probabilidades no qual podem ser ou não parametrizados. Sua principal característica é que, para cada classe de padrões, é atribuído uma função de densidade de probabilidade, que quando associada a um grupo, define a probabilidade existente de uma classe e seus objetos estarem contidos nesta imagem. Esta classificação é caracterizada por agregar um fundamento explícito do modelo de probabilidade, por exemplo, a teoria Bayesiana, que é matematicamente rigorosa e apresenta uma aplicabilidade probabilística por inferência. Levando em conta o domínio matemático, a classificação estatística foi provada com sucesso em aplicações de visão computacional.

2.7.2 Aprendizado Não Supervisionado

Durante o aprendizado não supervisionado, conhecido como agrupamento ou “clusterização”, não se faz uso de nenhuma informação sobre a classe do conjunto de treinamento. Neste tipo de aprendizado, a quantidade de classes e suas condições iniciais são passadas ao algoritmo de treino, no qual faz uma busca por grupos representativos no conjunto de dados, desta forma pode-se verificar que este tipo de classificação é adequado quando o objeto de estudo a ser investigado é desconhecido ou não possui características definidas.

Para Jain (1989), o objetivo é a identificação de agrupamentos em uma imagem, onde um agrupamento é definido por um grupo de pontos na imagem

que possui uma densidade local elevada quando comparada com a densidade de outras áreas da imagem (JAIN, 1989).

É possível encontrar outro tipo de aprendizado na literatura, como por exemplo o aprendizado semi-supervisionado, neste tipo de aprendizado, uma parte das amostras de treinamento não possuem rótulo de classe, desta forma parte do conjunto de treino sem rótulo é utilizada para melhorar o classificador final de acordo com o algoritmo de agrupamento utilizado (THEODORIDIS, KOUTROUMBAS, 2009).

2.8 Classificação Bayesiana

A teoria de decisão Bayesiana é considerada uma abordagem estatística fundamental para classificação de padrões, onde assume que cada classe de dados por características distintas entre si (HASTIE, 2008). Ela se baseia em quantificar a troca entre as diversas decisões de classificação utilizando probabilidade e os custos que acompanham essas decisões. Segundo Duda et al. (2000), um classificador Bayesiano classifica um objeto na classe a que é mais provável pertencer baseado nas características observadas. Cada classe de padrões tem sua probabilidade conhecida, porém seus padrões podem ser completamente conhecidos ou não, muitas vezes devem-se estimar os atributos dos padrões para análise. Os estados das naturezas possíveis são chamados de classes ω, sendo que ω = ω^1 para uma primeira classe,$\omega = \omega^2$ para uma segunda classe é $\omega = \omega_n$ para uma classe *n*.

A probabilidade a *priori* ω_j é o conhecimento anterior ao problema e é denotado por $P\ (\omega_j)$. Como esta informação ainda não seja suficiente para decidir em favor entre duas classes, é necessário l9evar em consideração os atributos das classes em questão.

Um dado vetor de atributos X cujo domínio é uma variável aleatória com d- componentes, a função densidade de probabilidade condicional $p(X \mid \omega_j)$, é a probabilidade de ocorrer X sendo ω_j a classe verdadeira. Pela regra de Bayes temos:

$$P(\omega_j \mid X) = \frac{p(x|\omega_j)P(\omega_j)}{p(X)} \tag{3}$$

onde

$$P(X) = \sum_{j=1}^{N} p(X|\omega_j)\, P(\omega_j) \tag{4}$$

e temos *N definido como* o número de classes assim podemos definir o critério de decisão como:

$$Decisão \begin{cases} \omega_1, \ CASO \ P(\omega_1|x) > P(\omega_2 \mid x) \\ \omega_2, \ CASO \ P(\omega_2|x) > P(\omega_1 \mid x) \end{cases} \tag{5}$$

A tomada de decisão de Bayes e dada pela máxima probabilidade a posteriori, no qual a decisão será dada em favor da probabilidade visa minimizar o erro.

2.9 Plataformas Móveis e Smartphones

Para Weiser (1991), o crescimento significativo no desenvolvimento para dispositivos móveis teve seu início a partir dos anos 90, e foram conhecidos inicialmente como dispositivos de pequeno porte ou dispositivos de mão que possuíam capacidade computacional e funcionalidades semelhantes a um microcomputador, como a popularização destes dispositivos emergiram novas plataformas de desenvolvimento de software voltado para estes dispositivos.

O acréscimo de smartphones foi consideravelmente influenciado pela possibilidade da criação de novas aplicações abrangendo as mais diversas áreas, assim como pelo aumento da capacidade computacional e sensores desses aparelhos (KAUR, 2014).

Para Verkasalo *(2009),* smartphone é um celular com funcionalidades avançadas que podem ser estendidas por meio de programas executados por seu sistema operacional. Os sistemas operacionais dos smartphones permitem que desenvolvedores criem milhares de programas adicionais, com diversas utilidades, tais programas denominados de aplicativos.

No mercado atual existem diversas plataformas para smartphones. Segundo Gartner (2016), as plataformas de execução para os smartphones que mais se destacam são: iOS, Windows Phone e Android, sendo o Android líder no *market-share*, tornando isso um fator crucial na escolha OS a ser implementado o aplicativo móvel.

Tabela *1* Vendas de Smartphones por Sistema operacional no 4º Trimestre de 2015

Sistema Operacional	**4º Trim. 2015 Unidades**	**4º Trim. 2015 Markt-share (%)**	**4º Trim. 2014 Unidades**	**4º Trim. 2015 Markt-share (%)**
Android	325.394.4	80,7	279.057.5	76,0
iOS	71.525.9	17,7	74.831.7	20,4
Windows	4.395	1,1	10.424.5	2,8
Outros	1.794,2	0,5	3.020,8	0,8
Total	403,109.5	100.0	367.334.5	100

Fonte: Gartner (2016)

Devido à pretensão de distribuição de o protótipo desenvolvido alcançar o maior grupo de usuários foi decidido iniciar desenvolvimento do projeto utilizando a plataforma de smartphones Android, podendo abranger assim maior grupo de usuários.

2.9.1 O Sistema Operacional Android

O Google Android é um Sistema plataforma *open-source*, focado em dispositivos móveis como smartphones, tablets e outros dispositivos, sua composição inclui um sistema operacional de *kernel* Linux, middleware, aplicativos e interface de usuário (ROGERS et al., 2009).

A totalidade do Google Android é praticamente distribuído sob a licença Apache 2.0, exceto algumas partes como, por exemplo, o Linux *kernel* patches que estão sob a licença GPLv2, como descreve Lecheta (2009).

O Android teve seu início em novembro de 2007, quando um grupo liderado pela Google e outras 33 empresas anunciaram a formação da Open Handset Alliance (ANDROID, 2017).

Por meio da comunidade *open-source*, a aliança integra softwares e outras propriedades intelectuais fornecidos pelas empresas que a compõem. O

licenciamento do software é feito por meio da licença Apache V2 que permite flexibilidade de uso, alteração do código Android e até mesmo que, após alterado, o código se torne proprietário da empresa que o alterou (ANDROID, 2017).

Entre os principais participantes da aliança podemos listar empresas como as fabricantes de aparelhos telefônicos HTC, LG, Motorola e Samsung, além de operadoras de telefonia móvel como China Mobile Communications, KDDI, Nextel, DoCoMo, T-Mobile, Telecom Italia e Telefônica, as empresas de semicondutores Audience, Broadcom, Intel, Marvell, Nvidia Qualcomm, Synaptics e Texas Instruments e as empresas de software: Ascender, eBay, Google, LivingImage, LiveWire e SONiVOX (ANDROID, 2017).

2.9.2 Arquitetura Android

A *Figura 8* ilustra a arquitetura da plataforma Android no qual pode-se observar uma composição formada por cinco camadas: Linux Kernel, Libraries, Android Runtime, Application Framework e Applications (ANDROID, 2017). Baseada no Kernel Linux e na linguagem de programação Java, estas camadas apresentam as seguintes características (ANDROID, 2017):

- Camada de aplicações (*Applications*): É a camada visível usuário final e contém todos os aplicativos disponíveis que acompanham a versão do Android, tais como o cliente de email, programa de SMS, calendário, mapas, navegador e agenda de contatos, além dos outros aplicativos fornecidos por terceiros ou desenvolvidos pelo próprio usuário.
- Camada Framework da aplicação (*Application Framework*): Os desenvolvedores têm liberdade para usar os recursos do hardware do dispositivo, acessar informação sobre localização, executar serviços em segundo plano (*background services*), configurar alarmes, adicionar notificações na barra de estado (*status bar*) e muito mais. Os desenvolvedores podem acessar sem restrições o mesmo conjunto de bibliotecas usado por aplicações de núcleo. O reuso de componentes é favorecido pela arquitetura da aplicação que permite que qualquer aplicação publique suas funcionalidades permitindo, desta forma, que outras aplicações façam uso destas funcionalidades (sujeito às regras de

segurança imposta pelo framework). Este mesmo mecanismo permite a substituição de componentes pelo usuário;

- Camada das Bibliotecas (*Libraries*): Nesta camada estão disponíveis um conjunto de bibliotecas em C/C++ que são utilizadas pelo sistema, tais recursos estão disponíveis para desenvolvedores por meio do framework da aplicação. Entre estas bibliotecas citamos a biblioteca OpenGL ES 1.0 e a SGL para trabalhar com gráficos, além do SQLite que uma ferramenta poderosa e leve para manipulação base de dados relacional.
- Camada do Ambiente de execução Android (Android *Runtime*): Um conjunto de bibliotecas de núcleo que provê a maioria das funcionalidades disponíveis nas bibliotecas de núcleo a linguagem de programação Java. É importante observar que toda aplicação Android é executada em seu próprio processo, uma instância da máquina virtual (VM – *Virtual Machine*) Dalvik. Esta máquina virtual permite, de modo eficiente, múltiplas instâncias da VM. O arquivo executável da VM Dalvik VM possui extensão "*dex*". A VM Dalvik depende do kernel Linux para funcionalidades como múltiplas linhas de processamento associadas a um único processo (*threading*) e gerenciamento de memória (baixo nível);
- Camada do Kernel Linux: Android conta com o Linux versão 2.6 para serviços de núcleo como segurança, gerenciamento de memória, drivers, etc. O kernel também atua como uma camada de abstração entre o hardware e o resto da pilha de software.

Figura 8- Arquitetura do sistema operacional Android

Fonte: Adaptado de DARWIN (2012)

2.9.3 Versões do Android

O sistema operacional Android sofre melhorias a cada ano, trazendo novas funcionalidades e inovação para o sistema operacional. Na Tabela 2 é possível visualizar as versões lançadas até o momento, juntamente com suas respectivas APIs.

Tabela 2- Versões do sistema Android e suas respectivas APIs.

Versão	***Codinome***	***Nível API***
Android 7.0	*Nougat*	*24*
Android 6.0	*Marshmallow*	*23*
Android 5.1	*Lollipop*	*22*
Android 5.0-5.0.2		*21*
Android 4.4	*KitKat*	*19*
Android 4.3	*Jelly Bean*	*18*
Android 4.2.x		*17*
Android 4.1.x		*16*
Android 4.0.3–4.0.4	*Ice Cream Sandwich*	*15*
Android 2.3.3–2.3.7	*Gingerbread*	*10*
Android 2.2	*Froyo*	*8*
Android 2.1	*Eclair*	*5, 6 e 7*
Android 1.6	*Donut*	*4*
Android 1.5	*Cupcake*	*3*

Fonte: Wikipédia.

2.10 Considerações Finais

Neste capitulo foi exposta a fundamentação teórica após realizar uma pesquisa na literatura e outros trabalhos similares. Durante esta revisão observou-se que a maneira mais segura de iniciar o desenvolvimento do aplicativo é através da identificação semi-supervisionada das moscas-das-frutas utilizando smartphone com o OS Android no qual possui grande abrangência no mercado (Gartner 2016), além de possuir grande de custo benefício para o projeto.

Desde o início do projeto, houve necessidade do uso da visão computacional no qual se pôde facilmente testar no software MatLab (Corker, 20011), devido não ser possível embarcar o MatLab e seus recursos em smartphones, optou-se pelo uso da biblioteca OpenCV 3.2 no qual facilmente pode ser embarcada em grande parte de smartphones.

Um dos questionamentos neste trabalho refere-se à confiança dos dados obtidos através das técnicas de visão computacional, devido às imagens das

moscas não possuir um padrão, qualidade e quantidade necessária para um treinamento mais complexo, foi descartado a uso do aprendizado profundo (DL), uma vez que o tempo para coleta de material e experimentos ficaria com um deadline comprometido para o trabalho, restando assim menos de um ano para realizá-lo. Assim, para a extração dos dados das imagens e gerar informações confiáveis foi adicionado um processo de classificação semi-supervisionada no qual usuário interage com o aplicativo.

3 TRABALHOS RELACIONADOS

Neste capitulo serão vistos trabalhos nos quais fazem uso da visão computacional para a otimização de processos realizados na agricultura.

As técnicas de visão computacional para detecção de pragas são relativamente novas quando comparadas com outras técnicas dentro da visão computacional ligadas a outras áreas de conhecimento. Considerando-se o problema dentro de um escopo maior de aplicações, seja dentro do pequeno agronegócio ou em indústria de grande porte, é fácil perceber a importância de uma detecção e eficiência no controle de pragas. Existe uma necessidade de análise inteligente de conteúdo das matérias disponíveis a fim de identificar as pragas e tomar decisões adequadas para seu controle. Quanto antes uma praga for identificada, maiores são as chances de evitar que a praga se alastre, causando danos e colocando a produção frutífera em risco.

3.1 Rastreamento da Mosca Drosophila Melanogaster

Chao et al.(2015), apresentam um sistema automático de rastreamento de movimentos da mosca *Drosophila melanogaster*. O método proposto lida de forma eficiente com problemas derivados da qualidade limitada da imagem e com interação entre moscas. A aplicação prática da metodologia adotada foi fortemente estimulada pelos resultados obtidos que tiveram como característica inerente à robustez e à variação das condições do ambiente de teste. Para a tarefa de automação do monitoramento as imagens foram convertidas em escala de cinza e processadas através de um filtro Gaussiano. As moscas foram depositadas em uma placa circular onde a detecção de movimento será realizada, neste presente trabalho, a placa é detectada usando um filtro *Canny* seguido de uma detecção de círculo utilizando a transformada de *Hough*. Esta técnica é apropriada para placas em forma de círculo, que é o caso mais comum. Para realizar o monitoramento, primeiro foi necessário identificar o background dos demais elementos e para isso foi o utilizado o método *Simple Gaussian* (SG).

Outra etapa importante para o monitoramento é calcular a média dos elementos e sua variância. Estes dois valores serão utilizados para determinar se cada elemento corresponde a uma mosca ou não com base num critério probabilístico baseado na área do elemento, evitando falso-positivo. O processo de rastreamento visa manter as identidades das moscas em toda a sequência de imagens. Cada mosca é atribuída uma identidade que deve ser preservada durante a experiência. Um dos problemas detectados foi que em algumas circunstâncias pode haver perda de algumas identidades de mosca (por exemplo, sobreposição de moscas ou fusão). No presente trabalho, abordagem probabilística que emprega um modelo dinâmico para prever a posição de cada voar no tempo *t* a partir das informações atuais em *t-1*, usando um filtro de *Kalman* de correção de predição. Isso resulta em uma maior robustez. A metodologia utilizada permite obter informações detalhadas e precisas sobre as moscas, tanto em relação à sua dinâmica quanto ao seu comportamento. A distância percorrida por cada mosca, as velocidades instantâneas e média, e número de saltos. Além disso, as diferentes moscas puderam ser classificadas pelo seu nível de atividade ou pelo tempo em que permaneceram em um determinado nível de atividade, ou por uma determinada região de placa. Tal experimento mostrou-se satisfatório e com grande eficiência na obtenção dos dados utilizando técnicas de visão computacional.

3.2 Monitoramento da mosca *Bactrocera tryoni* em Armadilhas *Tephri*

Para Liu et al. (2009), a mosca-das-frutas Qeensland (*Bactrocera tryoni*), é um dos insetos mais destrutivos da Austrália e para seu controle foi desenvolvido uma armadilha com visão computacional. Para seu sistema funcionar corretamente foi necessário utilizar uma armadilha do tipo *Tephri*, contudo a mosca é atraída por meio de um atrativo, e as imagens são registradas com as moscas ainda vivas, em muitos casos a mosca se descola durante o processo de captura, provocando imagens borradas outro caso comum é de uma mosca ser registrada em um local da armadilha e posteriormente registrada em outro lugar tornando a contagem imprecisa. Também foi pensado para esse sistema um adaptador de rede no qual disponibiliza as imagens e dados processados em um servidor web podendo ser monitorado via smartphones.

A detecção da mosca neste sistema utiliza a comparação por *template* que é compara com a imagem da mosca onde é então calculado usando uma formula a fim de descobrir a diferença entre as imagens.

Durante o pré-processamento da imagem são corrigidos o brilho e cores automaticamente onde é utilizado a equalização de histograma para salientar as moscas das frutas, facilitando sua segmentação.

As mosca-das-frutas de *Queensland*, possuem características diferentes das demais moscas, na parte superior esquerda e direita são amarelas e apresentam um padrão de simetria, no meio do corpo existe outra parte em amarelo entre as partes do topo esquerdo e direita, estas partes em amarelo são utilizadas para reconhecimento das moscas-das-frutas.

Os autores também analisaram dois espaços de cores, o HSV e CIELab, mas optaram em utilizar o espaço de cor CIELab, devido ao componente H (*Hue*) ser difícil de ser manipulado com a cor amarela. Neste trabalho os autores concluíram que seu experimento se mostrou efetivo podendo ser utilizado para captura de imagens, armazenamento e análise de imagens. Neste trabalho pode-se identificar 8 a cada 10 moscas através de imagens de alta qualidade.

3.3 Detecção e Contagem da Mosca Branca

Boissard, Martin e Moisan (2008), desenvolveram um sistema para detecção e a contagem da mosca branca nas folhas de rosas, isto foi possível por meio da combinação da aprendizagem, técnicas de baseadas em conhecimento e visão computacional, no qual foram realizadas duas classes de testes. A primeira classe corresponde ao uma série de folhas de rosas no qual não continham infestação da mosca branca, já na segunda amostra, foram utilizadas folhas contendo pelo menos uma mosca. Em seu primeiro teste, foram utilizadas 102 imagens, e teve 0% de falsos negativos e 3,1% de falsos positivos. Em outro teste foram utilizadas 60 amostras, tendo atingido 29,6% de falsos negativos e 2,0% de falsos positivos, desta forma, seu trabalho obteve uma média de 11% de falsos negativos e 2,7% de falsos positivos.

Bauch e Rath (2004), criaram um sistema para medir a densidade de uma praga entomológica, a mosca branca (*Trialeurodes vaporariorum* e *Bemesia tabaci*), dentro de estacas de plantas (*Lycopersicon lycopersicum*). O sistema combina extração automática de insetos de pragas com uma unidade de análise

de imagens de computador, e agora atingiu o estado de protótipo. Um mecanismo de aspiração móvel com um sistema de filtro mecânico recentemente desenvolvido é aplicado para coletar moscas brancas adultas. A unidade de coleta pode extrair indivíduos de pragas da população de plantas e alimentá-los a um sistema de reconhecimento óptico. As partículas que estão ligeiramente soltas na parte superior do suporte da planta são aspiradas, filtradas para fora da corrente de ar, colocadas numa correia transportadora e transportadas na frente de uma câmara CCD a cores. A análise de imagem digital é então realizada para classificar os objetos capturados em "organismos-alvo" (mosca branca) e "itens de erro" bióticos ou abióticos (organismos estranhos, poeira, partes de plantas). Os parâmetros de forma e cor servem como critérios de diferenciação.

Os resultados de classificação apresentados, baseiam-se na combinação de parâmetros de forma e cor. A taxa de reconhecimento resultante de 83% para moscas brancas corretamente classificadas ainda não é suficiente. Dado o caso de um baixo grau de infestação e um pequeno número resultante de indivíduos aspirados, uma possível classificação errónea levará a erros grandes desproporcionais na quantificação subsequente da densidade de pragas. Portanto, pesquisas adicionais devem ser realizadas para examinar a adequação dos parâmetros de textura e características morfológicas de moscas brancas adultas, a fim de melhorar o reconhecimento automático de objetos.

4 MATERIAIS E MÉTODOS

Neste capítulo é descrito o método proposto para realizar a contagem e identificação das moscas-das-frutas *A. fraterculus* e *C. capitata* a partir das suas principais caracterizas como tamanho e cor através de imagens digitais.

4.1.1 Ferramentas de Desenvolvimento

Os algoritmos aplicados neste trabalho foram inicialmente escritos em MATLAB e C++ juntamente com o desenvolvimento do protótipo para smartphones Android, os códigos em linguagem em C++ foram migrados para código JAVA. Durante o desenvolvimento em linguagem C++ e JAVA foi utilizado a biblioteca OpenCV. A biblioteca OpenCV é uma ferramenta de código aberto de processamento de imagens, criado inicialmente pela Intel em 1999. Desde então, um número de programadores tem contribuído para a evolução da biblioteca tornando uma das bibliotecas mais utilizadas no mundo para visão computacional (BRADSKI, 2000).

Atualmente a OpenCV possui mais de 2500 algoritmos otimizados, sendo amplamente utilizado em todo o mundo, tendo mais de 2.5 milhões downloads e 40 mil pessoas no grupo de usuários. A OpenCV pode ser utilizada em aplicações acadêmicas e comerciais ou sob uma licença BSD. Atualmente a OpenCV encontra-se na versão 3.2.

Outra ferramenta utilizada para o desenvolvimento da aplicação para smartphone foi a IDE Android Studio, as aplicações Android são desenvolvidas em um ambiente de testes e posteriormente são instaladas nos dispositivos móveis. A plataforma Android, que adotou a linguagem Java para o desenvolvimento de aplicações, fornecendo um SDK composto de várias ferramentas, tais como um depurador, um *plugin* para o ambiente de desenvolvimento Android Studio que possui um emulador de dispositivo permitindo que aplicações Android possam ser escritas sem a necessidade de um dispositivo físico para testes.

Foi escolhido a IDE Android Studio Versão 2.1.1 por possuir uma interface eficiente voltada para a programação na linguagem Java e edição de XML e possuir uma interface com menus de fácil acesso para o código e possuir uma visão clara para criação do *layout* do aplicativo no qual conta com o recurso de *drag-and-drop* dos componentes de interface, simplificando o desenvolvimento.

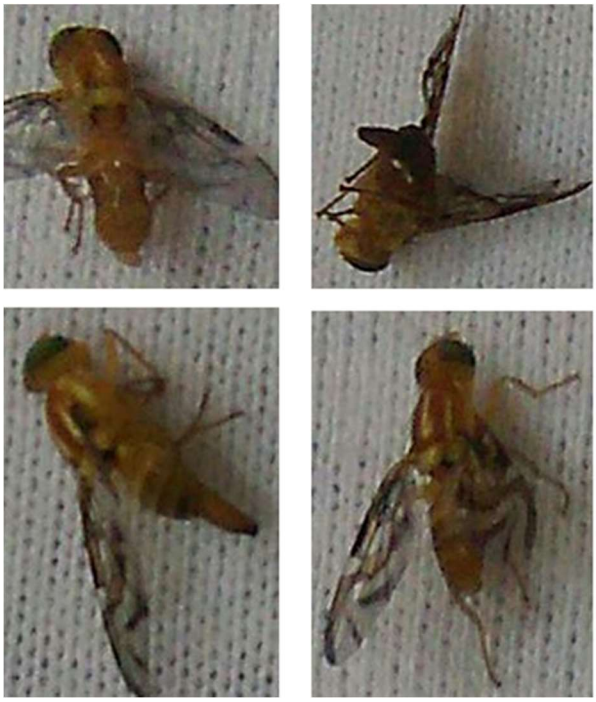
4.1.2 Dataset de imagens

Utilizando-se de um conjunto de amostras das moscas *A. fraterculus* e *C. capitata* foi possível construir um modelo que melhor descreve as características das moscas-das-frutas *A. fraterculus* e *C. capitata*. Para compreender o sistema proposto, foram definidos modelos matemáticos no qual tiveram seus resultados analisados e comparados.

O *dataset* de imagens utilizados neste trabalho foi criado através de amostras de moscas da espécie *A. fraterculus* e *C. capitata* fornecida pela EMBRAPA CLIMA TEMPERADO e foram registradas utilizando a câmera do smartphone Lenovo Vibe K5 em uma resolução de 8MP. As imagens obtidas apresentam brilho, tamanho da peneira e contraste diferentes, constituindo um *dataset* heterogêneo, dificultando o processo de contagem e identificação das moscas-das-fruas.

Nos experimentos foram utilizadas 1000 amostras de moscas recortadas de fotografias contento as moscas-das-frutas, por meio destas imagens foram geradas as imagens das moscas individuais com aproximadamente 150x150 pixel como pode ser visto na *Figura 9*. Destas 1000 amostras foram registradas 350 da espécie *C. capitata* e 650 da espécie *A. fraterculus*. As amostras das moscas foram realizadas sobre um tecido de algodão branco. As moscas utilizadas nestes experimentos não apresentavam sujeiras ou resquícios do atrativo utilizado nas armadilhas McPhail.

Figura 9 Amostra de mosca-das-frutas.



Fonte: Elaborado pelo autor.

4.2 Aplicativo Moscas-das-frutas

Um aplicativo para smartphone foi desenvolvido para o registro de imagens de armadilhas, contagem e identificação de moscas-das-frutas coletadas em peneiras a partir do conteúdo de armadilhas MCPhail.

Para desenvolvimento e testes o aplicativo foi testado em dois smartphones de baixo custo listados na Tabela 2 onde se pode ver as características[2] mais relevantes dos aparelhos utilizados nos testes.

Tabela *2* Modelos de smartphones utilizados para testes.

Modelo	Arquitetura	SoC Vendor	RAM	Câmera	Preço
Lenovo Vibe K5	ARM Cortex A53 1,50GHz Octa-core	Qualcomm	1892 MB	8 Mp CMOS BSI sensor	R$900,00
Samsung Galaxy E5	ARM Cortex A53 1200 MHz Octa-core	Qualcomm Snapdragon 410	1500 MB	8 Mp	R$1000,00

Fonte: Elaborado pelo autor.

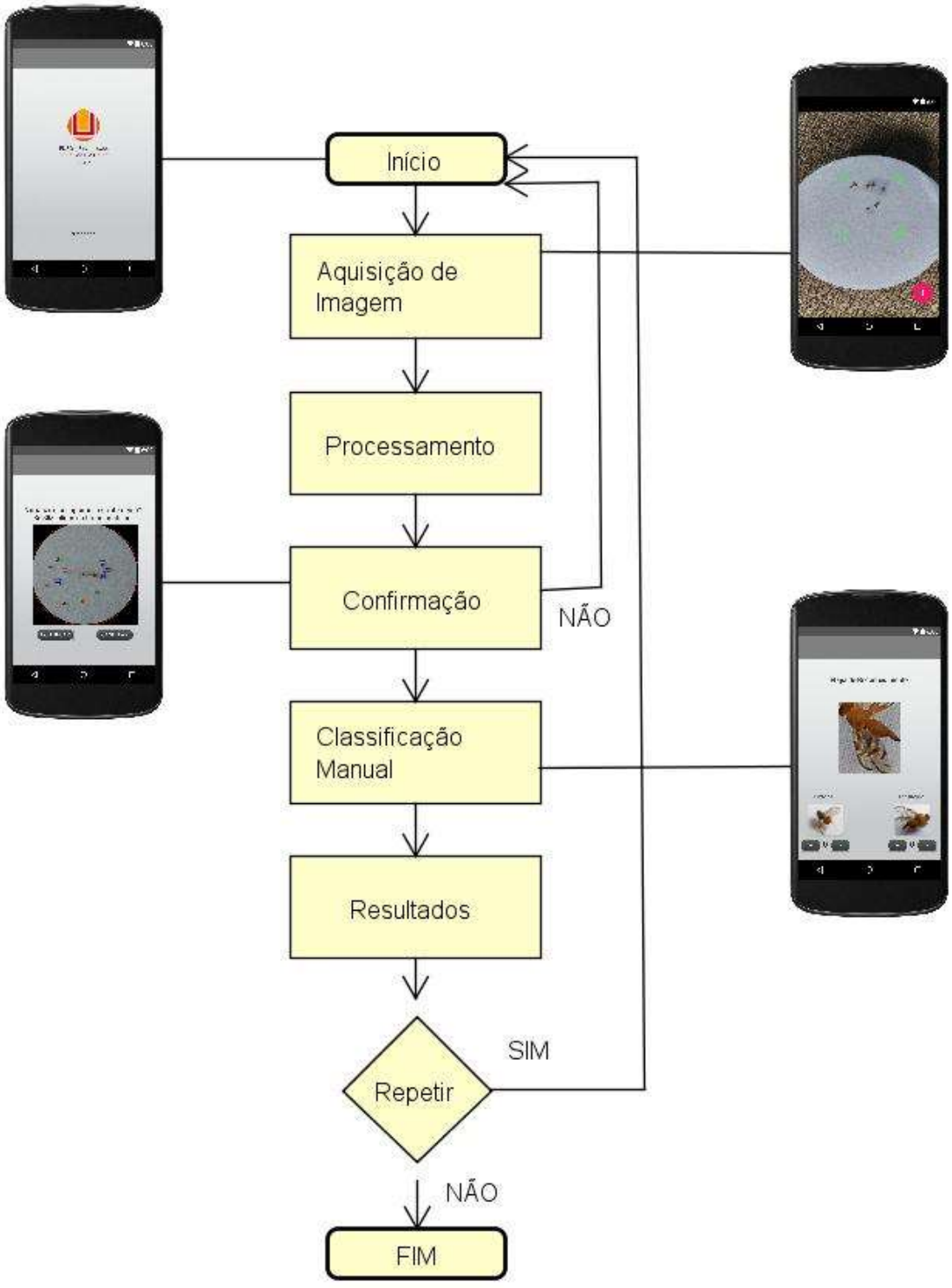
[2] Preço dos smartphones pesquisados no comércio local de Pelotas-RS em 2015 pelo autor.

Este trabalho estrutura-se em uma ferramenta de monitoramento de armadilhas do tipo MCPhail, no qual permitirá o usuário fotografar os objetos contidos na armadilha previamente filtrados em uma peneira em formato circular de 25 cm.

No diagrama de fluxo expresso na Figura 10, é exibido diagrama descrevendo o fluxo do aplicativo que consiste em:

1. Aquisição de imagem: A aquisição de imagem é realizada de forma automática ao detectar a peneira através da câmera do smartphone (Sessão 4.3)
2. Processamento: Nesta etapa a imagem é processada e então realizado a contagem e identificação das moscas
3. Confirmação: Nesta etapa o usuário pode ter uma pré-visualização do resultado
4. Classificação: O usuário deve identificar manualmente as moscas
5. Resultados: É gerado o resultado com a contagem de cada espécie

Figura 10 Diagrama de Fluxo do Processo de Contagem e Identificação.

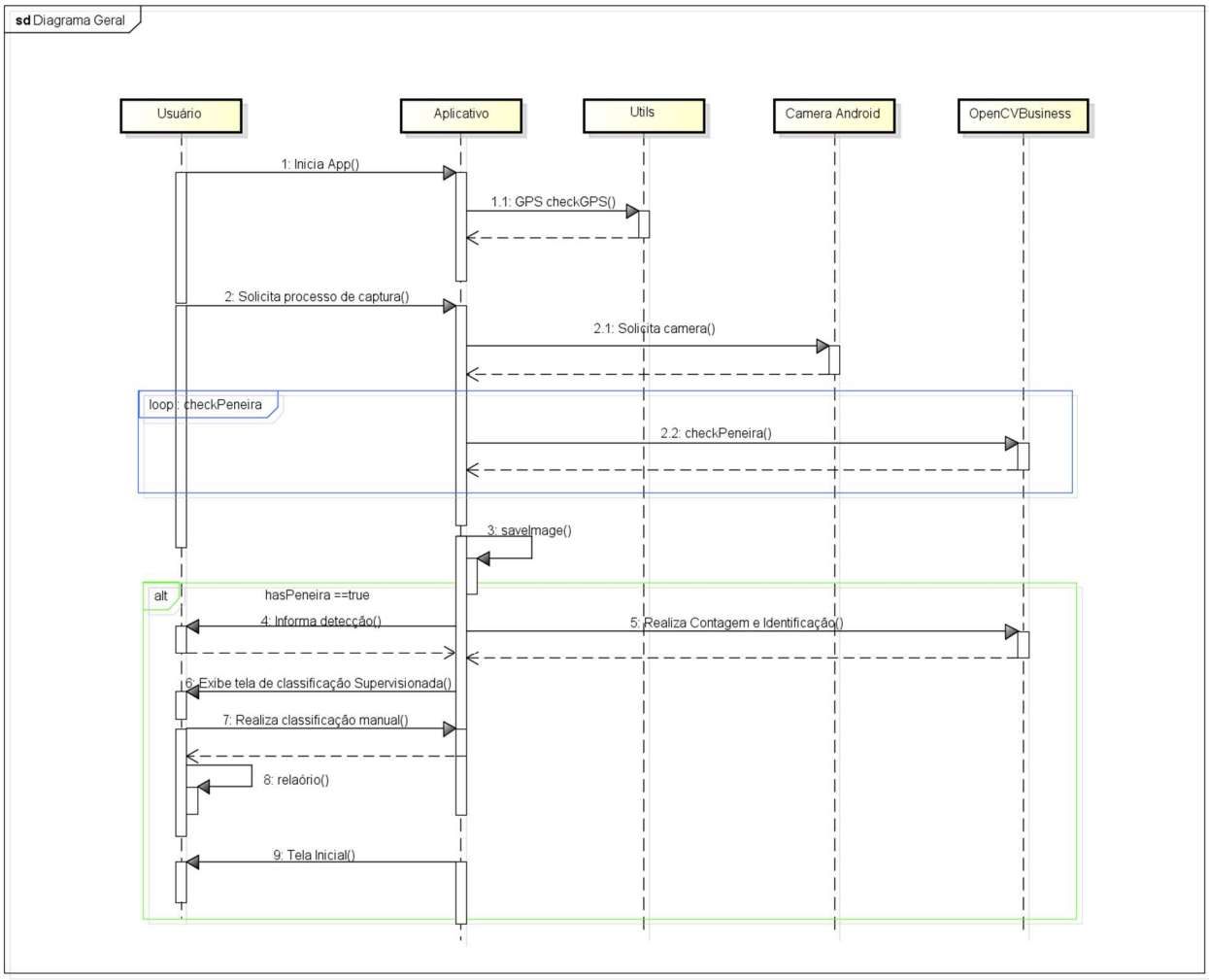


Fonte: Elaborado pelo autor.

Para detalhar o processo de contagem e identificação das moscas-das-frutas, foi criado um diagrama de sequência exibido na *Figura 11*, descrevendo

graficamente as etapas executadas, descrevendo a interação inicial através do usuário com o aplicativo.

Figura 11 Diagrama de Sequência do Sistema de Captura.



Fonte: Elaborado pelo autor.

Durante a inicialização do aplicativo foi proposto uma tela de carregamento do sistema, onde a biblioteca OpenCV é carregada pelo OS Android, assim esse processo é evitado no momento da captura da imagem. Após o carregamento, é feita a verificação do GPS, caso esteja ativo, sendo assim, quando uma imagem for capturada o sistema irá salvar os dados de geolocalização junto às informações da foto.

Depois da verificação do GPS, o usuário terá opção de iniciar a captura da peneira, no momento que ele iniciar este processo, ele deverá registrar a

imagem com o smartphone com aproximadamente 30 cm de distância da peneira, nesta etapa o aplicativo irá iniciar câmera do sistema e realizará a aquisição da imagem a cada três segundos até realizar a detecção da peneira através do método descrito na sessão 4.3. Feito a detecção da peneira, a imagem será salva e processada para realizar a contagem e a identificação das moscas através de um classificador bayesiano.

Após a classificação através do classificador bayesiano o aplicativo exibirá uma tela de confirmação, perguntando se o usuário deseja continuar com uma classificação manual.

A contagem manual é uma etapa para confirmar a classificação bayesiana onde o usuário do sistema pode confirmar a contagem e identificação das moscas. Na etapa de classificação manual, será exibida uma janela onde cada uma das moscas detectadas será exibida para identificação manual.

Ao final da etapa de classificação manual o aplicativo exibirá uma tela com os resultados obtidos.

Caso à peneira não ser encontrada, uma mensagem será exibida para o usuário informando sobre o ocorrido.

4.3 Interface do Aplicativo

O aplicativo para apresenta uma interface e workflow simples e intuitivo, sem muitas possibilidades de interações com usuário. O Aplicativo apresenta oito interfaces principais:

- Tela *splash*
- Tela inicial
- Tela de aquisição de imagem
- Tela de confirmação de aquisição de imagem
- Tela de identificação e contagem das moscas-das-frutas
- Tela de resultados
- Tela de listagem e reprocessamento

4.3.1 Tela de *Splash*

A tela de *Splash* é utilizada apenas ao iniciar o aplicativo e sua principal funcionalidade é fazer as verificações iniciais do sistema, como conexão wifi, GPS, além de ser responsável por carregar a biblioteca OpenCV.

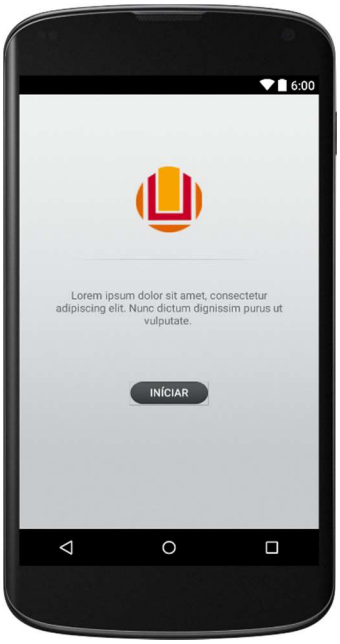
Figura 12 Tela de Splash.

Fonte: Elaborado pelo autor.

4.3.2 Tela Inicial

O intuído da tela é informar ao usuário informações básicas do uso do aplicativo e permitir a partir do botão inferior (INÍCIO) o início da etapa de aquisição de imagem.

Figura 13 Tela Inicial.

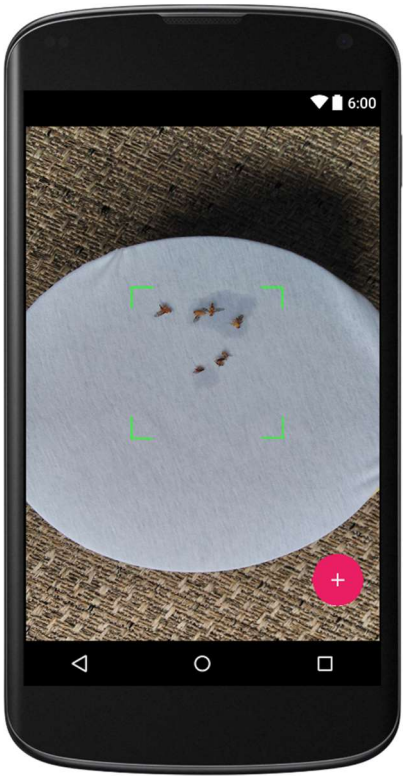


Fonte: Elaborado pelo autor.

4.3.3 Tela de Aquisição de Imagem

O processo para o usuário realizar aquisição da imagem a ser processada é muito simples, o clicar no botão INÍCIAR na tela inicial (*Figura 14*) a tela de aquisição de imagem irá exibir as imagens da câmera do Android e irá marcar no centro da tela um marcador verde no qual serve para indicar que a peneira deve ser centralizada nestes marcadores. Ao detectar a peneira no centro da tela, o sistema irá realizar a aquisição da imagem e em sequência será processada e salva. Após a imagem ser salva, a tela de confirmação será automaticamente carregada.

Figura 14 Tela de Aquisição de Imagem.



Fonte: Elaborado pelo autor.

4.3.4 Tela de Confirmação de Aquisição de Imagem

Na tela de confirmação de aquisição de imagem é exibida uma previa do resultado após o processamento onde é realizado a contagem e identificação das moscas de forma não-supervisionada. Nesta prévia as moscas são identificadas com retângulos de acordo sua espécie ou como clusters de moscas, após verificação desta previa o usuário pode avançar para a tela de contagem e identificação manual ou cancelar o processo utilizando os botões inferiores de Continuar ou Cancelar.

Figura 15 Tela de Confirmação de Aquisição de Imagem.

Fonte: Elaborado pelo autor.

4.3.5 Tela de Identificação e Contagem das Moscas-das-frutas

Nesta tela o usuário tem como finalidade realizar uma nova identificação e contagem de forma manual, a fim de garantir o acerto dos algoritmos já executados. Nesta interface o usuário tem a opção de informar a espécie da mosca encontrada além de informar a quantidade para uma ou ambas as espécies, caso, o usuário incremente o contador de mosca A >=1 e da mosca C para >=1 o usuário poderá clicar em qualquer botão (imagem das moscas) para confirmar a contagem. Este processo será realizado para todas as moscas encontradas.

Figura 16 Tela de Identificação e Contagem das Moscas-das-frutas.

Fonte: Elaborado pelo autor.

4.3.6 Tela de Resultados

A tela de resultados apresentará o total de moscas encontradas de acordo com a classificação manual caso o usuário opte por ela, caso contrário apresentará o resultado da contagem semi-supervisionada. Sendo assim, o usuário poderá saber quantas moscas de cada espécie foi encontrada, agregado a este layout o usuário poderá iniciar o processo de aquisição de imagem novamente.

4.4 Detecção da Peneira

Para realizar a detecção da peneira foi utilizado o detector de círculos baseado em transformadas direcionais de *Hough* no qual é aplicado a uma imagem obtida pela câmera do smartphone em um intervalo de três segundos, para identificar as regiões limitadas por bordas com configuração circular semelhante à que é apresenta na peneira na Figura 17. Antes de iniciar a detecção as imagens foram convertidas para tons de cinza. Como já se conhece o raio da peneira a ser detectada, o algoritmo é aplicado até que a peneira seja

encontrada. Para verificar a configuração ideal e realizar a detecção da peneira, foi utilizado a função *HoughCircles* da biblioteca OpenCV, no qual foram testados os parâmetros de entrada da função referentes a raio mínimo e raio máximo a ser detectado.

Durante os testes utilizou-se uma peneira (Figura 17) com um raio de 3,66 cm, revestida com uma malha de algodão a fim de apresentar um melhor contraste entre as moscas-das-frutas e o background da imagem. Para determinar o raio mínimo, verificou-se o intervalo de 0 a 200 da função *HoughCircles,* onde os valores foram incrementados de 10 em 10. Os testes foram realizados com 60 imagens de peneiras com uma distância de 30cm a 45cm da câmera do smartphone Lenovo Vibe K5. Durante o um pré-processamento as imagens foram modificadas da resolução de 4096X2304 para 256X144.

Figura 17- Peneiras utilizadas durante os testes

Fonte: Elaborado pelo autor.

Foi também aplicado um filtro Gaussiano para suavizar as bordas da imagem. Após a detecção da peneira, o objeto é identificado através de um círculo azul em sua volta e é criada uma máscara para remover o *background* da imagem e recordar deixando apenas a peneira na imagem a ser salva, tornando a imagem menor e otimizando os processos seguintes.

4.5 Definição do Modelo Matemático no Espaço de Cor HSV

No espaço de cores HSV existem importantes propriedades que podemos observar as referentes condições de iluminação, sendo invariantes em determinadas ocorrências:

- Superfícies foscas;
- Fontes de luz branca;
- Luz ambiente;

Devido ao espaço de cor HSV apresentar estas características, ele foi escolhido como objeto de estudo, podendo assim representar imagens coloridas possuindo um grau de independência da luz, uma vez que os testes foram realizados em ambientes com iluminação não controlados.

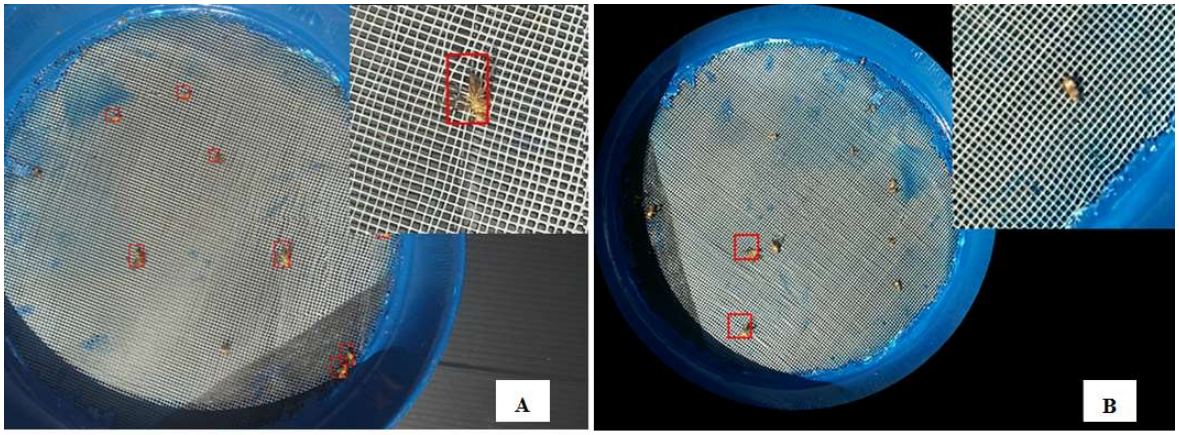
Para definição de um modelo matemático para detecção do corpo da moscas-das-frutas em imagens coloridas representadas no espaço de cores HSV, deve-se ressaltar que os teres componentes matiz (H), saturação (S) e intensidade (V) - possuem diferentes caraterísticas na composição de uma cor e não podem ser manipulados de forma independente. Desta forma, para definição do modelo matemático no espaço de cor HSV, foi definido uma metodologia para observação e seleção das relações entre as variáveis. Este modelo de cor foi utilizado apenas para segmentação e das moscas do restante do material encontrado nas peneiras, toda via durante os testes não foi possível distinguir a cor por espécie, devido a mosca de ambas as espécies apresentarem cores semelhantes durante a decomposição do inseto. Durante os testes foram verificados os intervalos dos valores de cada componente do espaço de cor HSV, e sendo ignorada a espécie das moscas-da-fruta, tendo assim como objetivo principal encontrar o melhor resultado para ambas as espécies.

Para obtenção do modelo cada componente do espaço de cor HSV foi dividida em intervalos de 20 em 20. Neste caso o componente H (matiz), que no espaço de cor varia de 0º a 360º, e foi separado em 18 intervalos primários, sendo eles: (0 º, 20º), (20º, 40º), até (340º, 360º). Para os componentes S (saturação) e V (intensidade), que no espaço de cor foram divididos de 0% a 100%, com um intervalo 20% de uma faixa a outra ficando: (0%, 20%), (20%, 40%), até (80%, 100%). Para criação do modelo foram aplicadas às 55 imagens, sendo estas contendo imagens as moscas-das-frutas de ambas espécies além de outros objetos não moscas-das-frutas, como grama, mosquito e aranha.

Toda via cada um dos canais de cor H, S e V apresenta diferentes funções na composição da cor final, desta forma, foram analisados apenas os intervalos com ocorrência da mosca-das-frutas no primeiro teste, no qual foi utilizado as faixas (0 º, 20 º), (20 º, 40º), até (340 º, 360 º) para o canal H e os valores 0 à 1 nas faixas de faixas (0.0, 0.2), (0.2, 0.4), até (0.8, 1.0) para os canais S e V. A partir dos próximos testes, foram combinados todos os intervalos dos canais S e V com os intervalos do canal H no qual tiveram alguma ocorrência, esta combinação de valores gerou 30 intervalos a serem testados. Ainda neste teste foram utilizadas 15 imagens contendo no total de 210 moscas-das-frutas.

Durante a fase de testes iniciais pode-se perceber uma baixa qualidade das imagens adquiridas através da biblioteca OpenCV quando comparadas com imagens obtidas diretamente pela API da câmera Android como pode ser visto na *Figura 18*. Devido à baixa qualidade das imagens optou-se em utilizar a API de câmera do Android para realizar a aquisição de imagens assim como peneiras revestidas com uma malha de algodão para realçar os objetos de interesse.

Figura 18 Qualidade da Imagem, (A) Foto da câmara com API Android, (B) Foto da câmera com wrapper OpenCV.



Fonte: Elaborado pelo autor.

Antes dos testes foram encontrados alguns possíveis problemas para este modelo, devido o atrativo utilizado nas armadilhas *MCPhail* apresentarem uma cor semelhante a mosca-das-frutas, podendo causar falsos positivos como pode ser visto na Figura 19. Devido ao problema da coloração do atrativo ser semelhando a cor das moscas-das-frutas, os testes e resultados se deram

coletando as moscas e já filtradas sem a proteína, facilitando assim a identificação.

Figura 19 Atrativo detectado como falso positivo.

Fonte: Elaborado pelo autor.

4.6 Modelo supervisionado para Classificação das Moscas-das-frutas

Este método constitui da análise dos dados extraídos do *dataset* de imagens para realizar a definição do limiar entre os parâmetros analisados visando à contagem e identificação das moscas-das-frutas. Os resultados obtidos deram-se através da extração dos dados utilizando algoritmos escritos em JAVA, onde a partir de imagens foram coletados dados referentes a cor, a área, ao número de regiões e resultado da comparação dos histogramas das moscas, no qual foi definido um limiar dos parâmetros através da verificação do valor mínimo, máximo e média para cada um deles. A extração dos dados foi realizada primeiramente com as imagens da mosca *A. fraterculus* e em sequência com a mosca *C. capitata*, a fim de separar os dados de maneira organizada facilitando a leitura posterior. Após a aquisição dos dados das moscas, foi gerada uma base de dados contendo as informações extraídas de cada mosca assim como a identificação da sua espécie. Os dados obtidos por meio deste modelo servirão tanto para a classificação manual quanto para treinamento do classificador bayesiano.

4.7 Definição do Modelo Bayesiano para Classificação das Moscas

O classificador semi-supervisionado bayesiano (*Naive Bayes*) foi pré-definido para realizar a identificação das espécies *C. capitata* e *A. fraterculus*. O método proposto para os testes de validação dos parâmetros de treinamento deu-se através dos dados extraídos de um *dataset* próprio. No algoritmo de treinamento foram utilizados como entrada os parâmetros referentes a área, número de regiões e o resultado da comparação dos histogramas. Para realizar a comparação dos histogramas, duas imagens foram definidas como *ground truth*, uma delas contendo uma amostra da imagem da *C. capitata* e outra contendo a mosca *A. Fraterculus* no qual seus histogramas são comparados com o histograma da mosca ou objeto a ser identificado. Os dados utilizados durante a parametrização do classificador bayesiano assim como para seu treinamento e aplicação de rótulos se deram a partir dos dados adquiridos através do método supervisionado.

Foram realizados 20 testes para a obtenção dos dados, durante esta etapa o processo, o algoritmo bayesiano utilizou os dados de 900 imagens para o treinamento, sendo o restante (100) utilizado para realizar os testes de classificação onde posteriormente foi feita uma verificação manual sobre a taxa de acerto do algoritmo.

A taxa de acerto foi calculada em forma de porcentagem, uma vez conhecida à procedência de dados das moscas testas. Ainda durante os testes, para cada execução foram selecionadas 15 moscas da espécie A. Fraterculus e cinco da mosca C. Capitata de forma aleatória.

4.8 Redefinição do *Ground Truth Baseado* na Classificação Manual

Esta metodologia foi definida a fim de maximizar os resultados da contagem e identificação das moscas-das-frutas devido grande parte dos experimentos serem realizados em ambientes com iluminação não controlada e disposição de angulação da câmera não fixada, causando assim variações no processo de aquisição de imagem. Um dos métodos já utilizados no processo de detecção das moscas-das-frutas dá-se através do comparativo de histogramas entre a mosca ou objeto a ser identificado com o *ground truth,* verificando sua similaridade. Devido às moscas definidas como *ground truth* também apresentarem variações de distância da câmera e iluminação de

maneira diferente da mosca a ser testada, propõem-se uma nova abordagem, em que consistem redefinir o *ground truth* baseado na classificação manual do usuário. Desta forma a identificação e contagem devem ser feitas duas vezes, no primeiro momento o algoritmo utilizara o *ground truth* padrão do sistema até que três moscas de cada espécie sejam classificadas, após isto, um novo *ground truth* é definido baseado na classificação manual do usuário. Após a nova definição do *ground truth* a contagem e identificação são reiniciadas, e as seis moscam já classificadas inicialmente, serão reclassificadas novamente a fim de garantir a maior taxa de acerto. Nesta metodologia utilizou-se 15 imagens em ambiente externo com sombra, 15 imagens em ambiente externo com sol e 15 imagens em ambiente interno com iluminação ambiente, onde todas as imagens continham 10 moscas da espécie *A. fraterculus* e 10 moscas da espécie *C. capitata*.

5 RESULTADOS

Neste capitulo será abordado os resultados dos testes realizados de acordo com a metodologia proposta no Capitulo 4, no qual após analise serviram para conclusão do desenvolvimento do aplicativo para o smartphone Android.

5.1 Detecção da Peneira

Como resultado dos testes para realizar a detecção da peneira, obteve-se como melhor resultado para o intervalo de raio mínimo e máximo para os valores entre 60 a 120 para função *HoughCircles*, desta forma foram definidos os valores para raio mínimo igual a 60 e para raio máximo igual a 120 a fim de realizar a detecção da peneira com maior taxa de acerto. Através destes valores para o raio mínimo e máximo, foi possível alcançar uma precisão de 97,5% de acerto na detecção da peneira e apresentando apenas 2,5% de taxa de erro. Na *Figura 20* (A, B) podemos ver peneiras detectadas com sucesso e na *Figura 20* (C, D) podemos ver exemplos de peneiras detectadas com falha. Alguns erros nesta etapa podem ocorrer em fator da iluminação e devido angulação da câmera durante a aquisição da imagem na *Figura 20* (C), um pedaço da peneira foi recortada, onde uma mosca acabou sendo cortada ao meio da imagem, já na *Figura 20* (D), pode notar que o fundo da imagem aparece juntamente a com a peneira podendo ocorrer a contagem de falsos positivos nesta região de “não peneira”. Na Tabela 3 exibe todos os intervalos testados, no qual pode-se concluir que o raio entre 60 à 120 obtiverão resultados positivos e apresentando um baixo índice de erro.

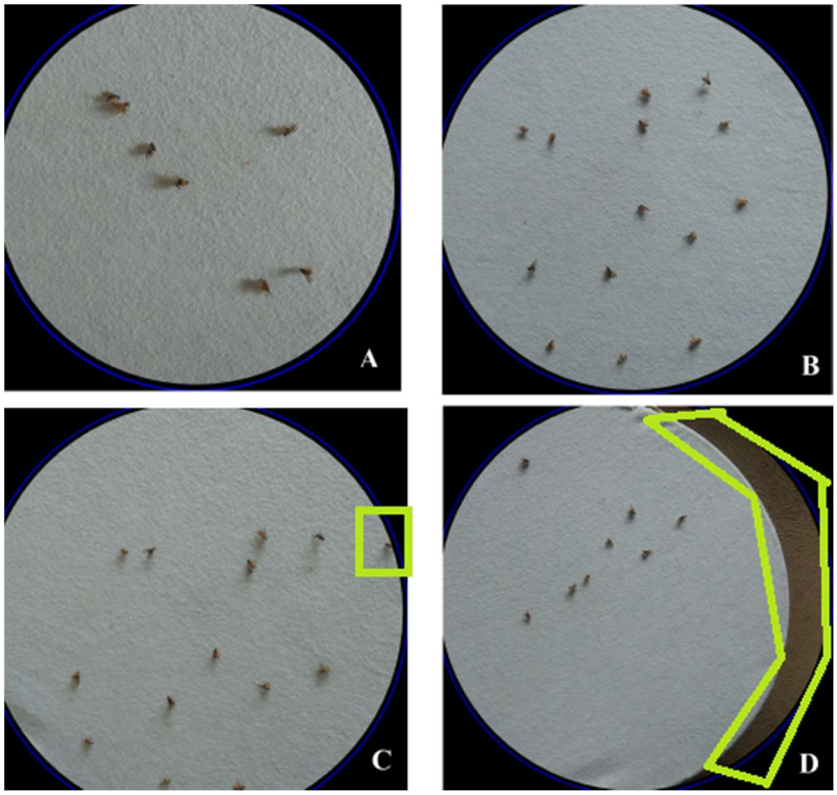
Tabela 3 Intervalos testados para detecção da peneira.

Nª	Intervalo	Ocorrência	
		Peneira	Não Peneira
1	0 à 10	0	274
2	10 à 20	0	147
3	20 à 30	0	108
4	30 à 40	0	35
5	40 à 50	2	5
6	50 à 60	1	1
7	60 à 70	4	0
8	70 à 80	4	1
9	80 à 90	7	0
10	90 à 100	9	0
11	100 à 110	11	0
12	110 à 120	9	0
13	120 à 130	7	0
14	130 à 140	3	0
15	140 à 150	3	0
16	150 à 160	2	0
17	160 à 170	0	0
18	170 à 180	0	0
19	180 à 190	0	0
20	190 à 200	0	0

Fonte: Elaborado pelo autor.

Figura 20 Detecção de peneira - (A, B) peneiras encontradas sem erro, (C, D)

peneiras encontradas com erro.



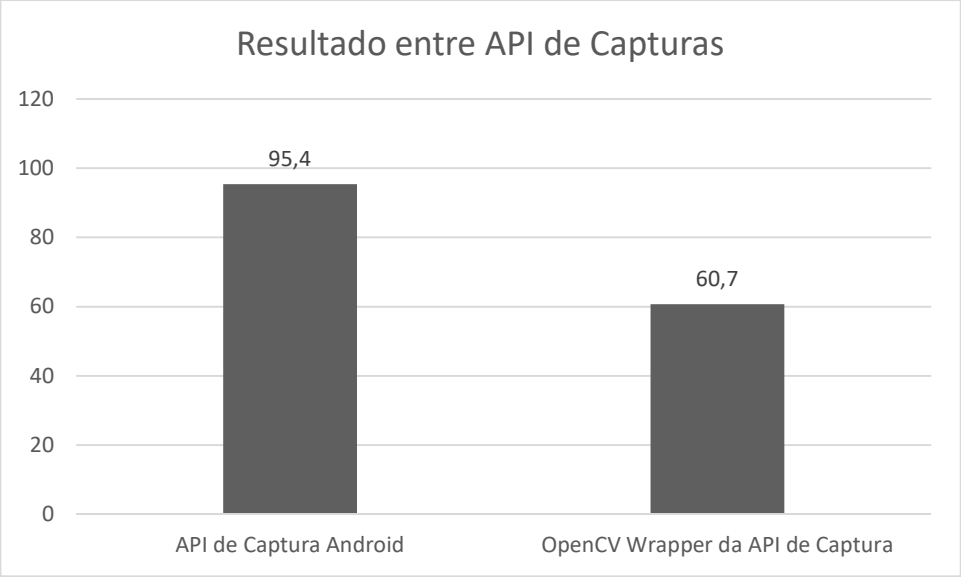
Fonte: Elaborado pelo autor.

5.2 Detecção de Moscas-das-frutas no Espaço de Cor HSV

Na fase de testes, mais precisamente durante a etapa de aquisição de imagem foi detectado que a resolução das imagens obtidas a partir da câmera utilizando a biblioteca OpenCV era limitada pela resolução da tela do smartphone e limitada na parametrização dos parâmetros de configuração provendo imagens de baixa qualidade, podendo desta forma comprometer as próximas etapas. Assim optou-se em realizar um teste prévio utilizando a câmera nativa do sistema Android ou a câmera do OpenCV um *wrapper* da câmera do sistema Android. O teste realizado consistiu e verificar contagem das moscas-das-frutas através do intervalo de cores do modelo HSV, toda via, os testes foram realizados utilizando a metodologia descrita na sessão anterior e foi analisando as variações no canal H, S e V. Como resultado pode-se verificar que a taxa de acerto quando utilizado a câmera nativa do Android apresentou uma taxa de acerto de 60.7% contra uma taxa de acerto de 95.4% quanto testado utilizando o *wrapper* de câmera da biblioteca OpenCV, na *Figura 21* é exibido um gráfico comparativo entre estes dois modos de utilização da câmera.

O baixo desempenho neste caso decorrido pelo uso do wrapper do OpenCV, ocorre devido ao OpenCV diminuir a qualidade e resolução da imagem.

Figura 21 Resultado em % na taxa de precisão com as APIs de captura de imagens do Android e Wrapper do OpenCV.



Fonte: Elaborado pelo autor.

Mesmo encontrando problemas no início da etapa de segmentação, foram realizados testes para encontrar o intervalo de cor adequado para detecção das moscas-das-frutas. Desta forma foi analisada a combinação dos intervalos de cada canal do espaço de cor HSV como descrito na metodologia. Tendo realizado os testes e gerado as tabelas com os dados, pode-se verificar que no canal H, que os intervalos de 20º a 140º obtiveram os melhores resultados de detecção das moscas-das-frutas. No qual os valores do canal a H sofreram variações em um intervalo de 20º a cada teste, na *Tabela 4* podemos ver os intervalos onde obtiveram os melhores resultados no canal H.

Tabela 4 Intervalo de cores testado no canal H

Intervalo	Frequência em %	
	Moscas	**Não Mosca**
0º <H <=20º | 0%<S<=100% | 0%<V<=100%	0%	0%
20º <H <=40º | 0%<S<=100% | 0%<V<=100%	12,5%	0%
40º <H <=60º | 0%<S<=100% | 0%<V<=100%	6,25%	0%
60º <H <=80º | 0%<S<=100% | 0%<V<=100%	75%	37,5%
80º <H <=100º | 0%<S<=100% | 0%<V<=100%	75%	37,5%
100º <H <=120º | 0%<S<=100% | 0%<V<=100%	43,75%	6,25%
120º <H <=140º | 0%<S<=100% | 0%<V<=100%	18,75%	0%
140º <H <=160º | 0%<S<=100% | 0%<V<=100%	18,75%	6,25%
160º <H <=180º | 0%<S<=100% | 0%<V<=100%	0%	6,25%
180º <H <=200º | 0%<S<=100% | 0%<V<=100%	0%	18,75%
200º <H <=220º | 0%<S<=100% | 0%<V<=100%	0%	0%
210º <H <=240º | 0%<S<=100% | 0%<V<=100%	0%	25%
240º <H <=260º | 0%<S<=100% | 0%<V<=100%	0%	0%
260º <H <=280º | 0%<S<=100% | 0%<V<=100%	0%	0%
280º <H <=300º | 0%<S<=100% | 0%<V<=100%	0%	12,5%
300º <H <=320º | 0%<S<=100% | 0%<V<=100%	0%	0%
320º <H <=340º | 0%<S<=100% | 0%<V<=100%	0%	25%
340º <H <=360º | 0%<S<=100% | 0%<V<=100%	0%	0%

Fonte: Elaborado pelo autor.

Desta forma verificou-se o intervalo de cor HSV com melhor resultado, testando cada um dos canais de cor, permitindo um melhor processo de segmentação. Os canais S e V sofreram variações de 0,2 em cada teste. Os resultados podem ser vistos na Tabela 5.

Tabela *5* Intervalo de cores testado no canal S e V

Intervalo	Total	
	Moscas	**Não Mosca**
S>0,0 e S<=0,2 & V>0,0 e V<=1,0	70	147
S>0,2 e S<=0,4 & V>0,0 e V<=1,0	70	51
S>0,4 e S<=0,6 & V>0,0 e V<=1,0	70	5
S>0,6 e S<=0,8 & V>0,0 e V<=1,0	5	0
S>0,8 e S<=1,0 & V>0,0 e V<=1,0	0	85
S>0,0 e S<=0,2 & V>0,0 e V<=0,2	0	97
S>0,2 e S<=0,4 & V>0,0 e V<=0,2	10	0
S>0,4 e S<=0,6 & V>0,0 e V<=0,2	0	0
S>0,6 e S<=0,8 & V>0,0 e V<=0,2	0	0
S>0,8 e S<=1,0 & V>0,0 e V<=0,2	0	80
S>0,0 e S<=0,2 & V>0,2 e V<=0,4	70	25
S>0,2 e S<=0,4 & V>0,2 e V<=0,4	70	17
S>0,4 e S<=0,6 & V>0,2 e V<=0,4	30	0
S>0,6 e S<=0,8 & V>0,2 e V<=0,4	0	5
S>0,8 e S<=1,0 & V>0,2 e V<=0,4	5	10
S>0,0 e S<=0,2 & V>0,4 e V<=0,6	70	375
S>0,2 e S<=0,4 & V>0,4 e V<=0,6	40	10
S>0,4 e S<=0,6 & V>0,4 e V<=0,6	45	0
S>0,6 e S<=0,8 & V>0,4 e V<=0,6	5	0
S>0,8 e S<=1,0 & V>0,4 e V<=0,6	0	0
S>0,0 e S<=0,2 & V>0,6 e V<=0,8	70	310
S>0,2 e S<=0,4 & V>0,6 e V<=0,8	5	0
S>0,4 e S<=0,6 & V>0,6 e V<=0,8	5	0
S>0,6 e S<=0,8 & V>0,6 e V<=0,8	0	0
S>0,8 e S<=1,0 & V>0,6 e V<=0,8	0	20
S>0,0 e S<=0,2 & V>0,8 e V<=1,0	0	5
S>0,2 e S<=0,4 & V>0,8 e V<=1,0	0	0
S>0,4 e S<=0,6 & V>0,8 e V<=1,0	0	5
S>0,6 e S<=0,8 & V>0,8 e V<=1,0	0	5
S>0,8 e S<=1,0 & V>0,8 e V<=1,0	0	25

Fonte: Elaborado pelo autor.

Pode-se concluir através dos dados das Tabelas 4 e Tabela 5 deste experimento que o melhor intervalo de cores no canal HSV para detecção das moscas-das-frutas corresponde à faixa de valores de:

Para o canal H: 20º até 140º

Para o canal S: 0,4 até 0,6

Para o canal V: 0,0 até 1,0

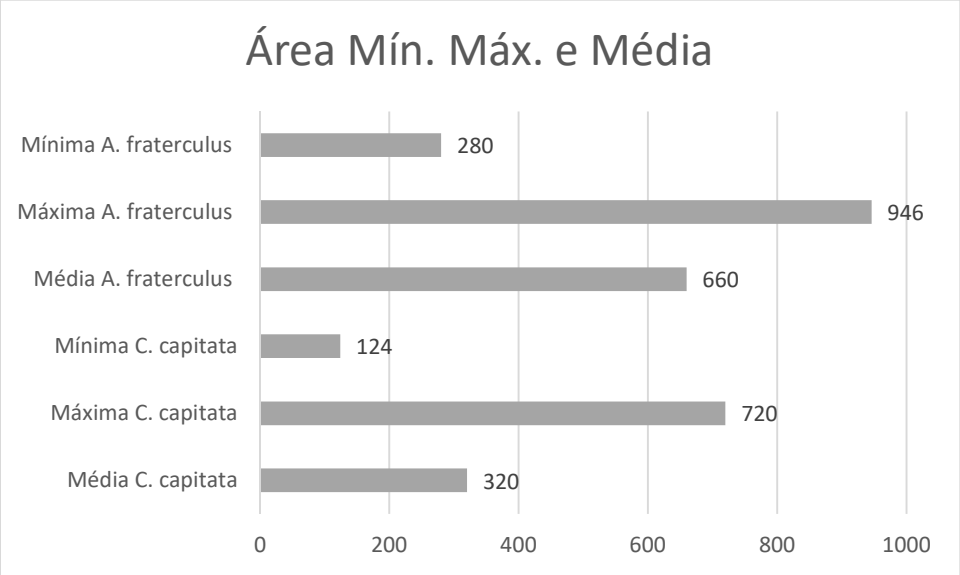
Mesmo apresentando falhas e detectando falsos negativos nesta etapa, tais ocorrências serão tratas posteriormente através de outras metodologias.

5.3 Resultados Estatísticos de Tamanho e Comparação de Histogramas.

Através do modelo proposto para realizar a classificação supervisionada para moscas-das-frutas, foi possível obter os resultados especificados nesta sessão. Devido à cor de ambas as moscas *C. capitata* ou *A. fraterculus*, serem similares e uma única espécie apresentar diversas tonalidades similares a da outra espécie, foi preciso encontrar outros critérios de realizar a distinção entre as espécies.

Por meio da segmentação e após a aplicação da erosão e dilatação foi possível calcular a área da mosca utilizando a função *contouArea,* desta forma foi testado e definido um limiar entre as espécies exibidas na Figura 22. Um fator preocupante durante a análise dos dados foi que o tamanho da área máxima da mosca *C. capitata* encontrar dentro do intervalo de tamanho da área da *A. Fraterculus* devido à detecção de clusters da mosca *C. Capitata,* nestes casos a contagem e identificação manual apresentam uma vantagem em relação à contagem e identificação através do classificado bayesiano.

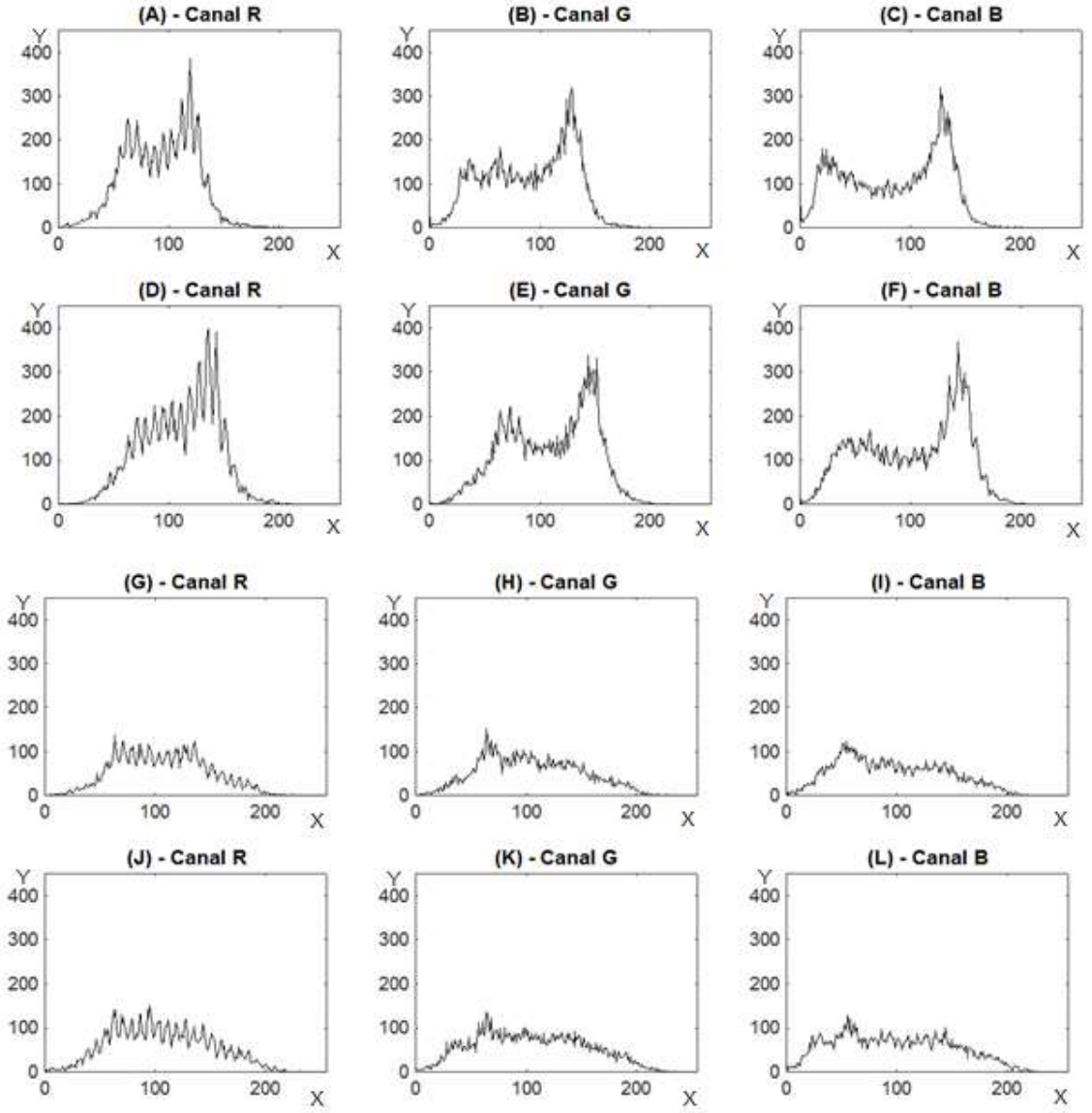
Figura 22- Gráfico com tamanho da área das moscas-das-frutas



Fonte: Elaborado pelo autor.

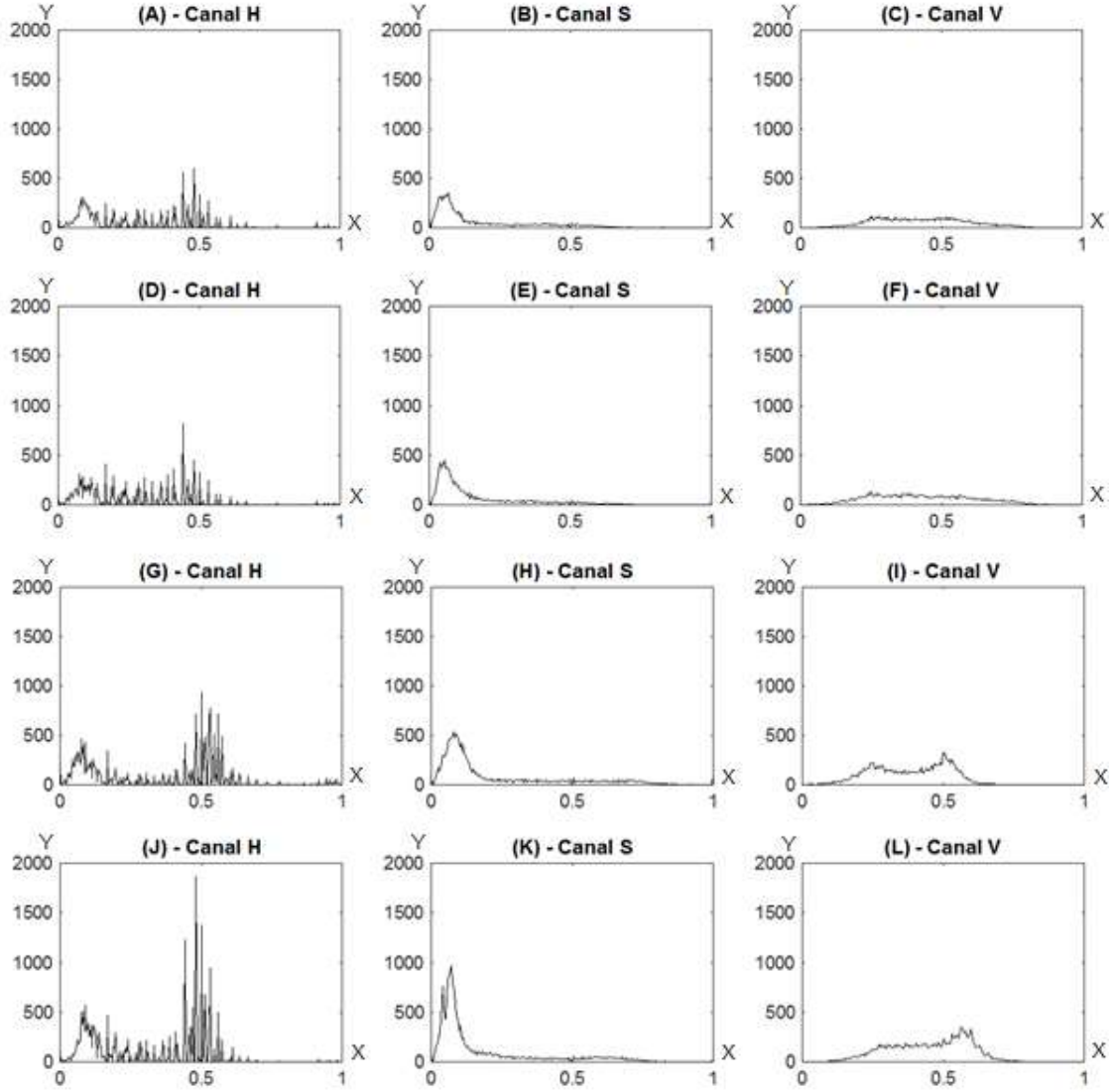
A utilização de comparação de histograma em imagens também já foi utilizada para classificação de insetos por Zhu e Zhang (2010). Durante os testes comparando os histogramas das moscas-das-frutas encontradas com o *ground truth* pré-definido no sistema foi obtido um resultado de 74% de acerto para moscas-das-frutas da espécie *A. Fraterculus podemos* ver na *Figura 23* (A, B, C) o histograma da primeira mosca *A. Fraterculus*, na *Figura 23* (D, E, F) a segunda mosca, ambos os resultados dos histogramas nos respectivos canais R G e B , na *Figura 25* (A, B, C) podemos ver o histograma do *ground truth* definido para a mosca A. *Fraterculus*, também respectivamente nos canais R G e B.

Figura 23 Histograma em RGB das moscas a serem comparadas. 1ª A. Fraterculus (A, B, C), 2ª A. Fraterculus (D, E, F), 1ª C. Capitata (G, H, I), 1ª C. Capitata (J, K, L).



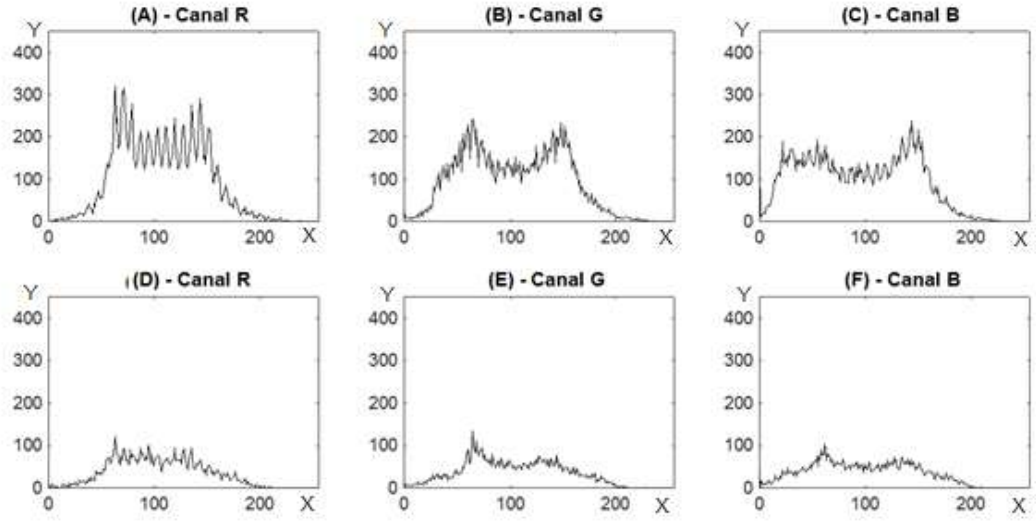
Fonte: Elaborado pelo autor.

Figura 24 Histograma em HSV das moscas a serem comparadas. 1ª *A. Fraterculus* (A, B, C), 2ª *A. Fraterculus* (D, E, F), 1ª *C. Capitata* (G, H, I), 1ª *C. Capitata* (J, K, L).



Fonte: Elaborado pelo autor.

Figura 25 Histograma das imagens definidas como *ground truth em RGB*. (A,B,C) *A. Fraterculus* e (D, E, F) *C. Capitata*.



Fonte: Elaborado pelo autor.

Para os resultados de identificação utilizando a comparação por histograma para a mosca *C. Capitata* no método proposto apresentou uma taxa de certo de 70% de acerto quando comparado com o *ground truth* especifico para sua espécie. Na *Figura 23* (G, H, I) temos os histogramas em RGB da primeira mosca *C. Capitata* e na *Figura 23* (J, K, L) pode-se ver os histogramas da segunda mosca*, C. Capitata* e na Figura (D, E, F) podemos ver o histograma do *ground truth* definido para a mosca *C. Capitata* em RGB. Para otimizar o método atual, foi proposto o método de comparação de histograma onde o *ground truth* foi definido de forma dinâmica pelo usuário e seus resultados são descritos na seção 5.5.

5.4 Resultados Através do Modelo Bayesiano para Classificação das Moscas

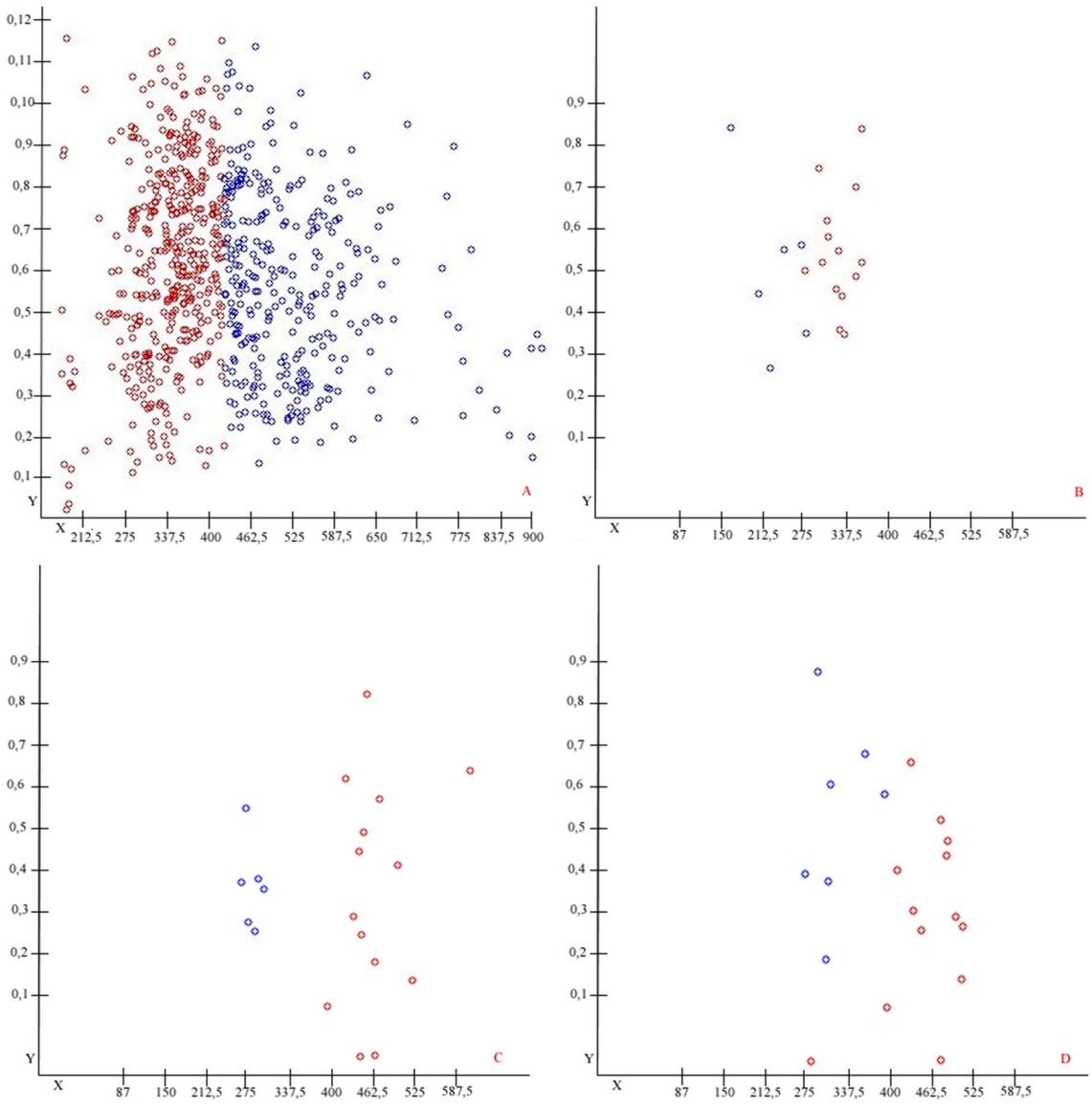
Os dados aqui explorados partem do resultado da probabilidade, a *priori* dos conjuntos de treinamento no qual representa a melhor identificação das moscas separadas em duas classes, neste método sabe-se que as diferentes intensidades de iluminação e ângulo da câmera podem causar variações nos parâmetros utilizados durante as atribuições de rótulos nestes objetos.

Os resultados desta abordagem são referentes à metodologia proposta, toda via, acredita-se que esses resultados possam ser otimizados se houver um número maior de dados para treinamento.

A partir do Classificador *Naive Bayes*, foram realizados os testes e pode-se alcançar uma taxa de acerto 76% de acerto durante a identificação das moscas.

Na *Figura 26* podemos ver o gráfico dos testes onde os marcadores vermelhos são utilizados para representar a mosca *C. Capitata* e o marcador azul para representar a mosca *A. Fraterculus*, ambas as moscas foram estão representadas em relação a coordenadas *X* e *Y,* onde Y é o valor de comparação dos histogramas e X representado pelo tamanho da região dos objetos. O principal critério para a identificação das moscas neste teste foi o tamanho da mosca após a segmentação. Na *Figura 26* A, é possível ver o gráfico de treinamento e pode-se notar que o número de regiões é um bom parâmetro para ser utilizado para a identificação das moscas e na *Figura 26* B, C e D podemos ver o resultado de alguns testes realizados utilizando o classificador bayesiano, mostrando visivelmente a diferença de tamanho das espécies das moscas.

Figura 26 Gráfico da classificação bayesiano das moscas das frutas.



Fonte: Elaborado pelo autor.

5.5 Taxa de Acertos Utilizando o *Ground Truth Baseado* na Classificação Manual

Neste método analisado pode-se aferir a precisão de acertos levando à consideração a escolha do usuário durante a classificação manual das três primeiras moscas de cada espécie realizando assim uma nova definição de *ground truth*. Durante os primeiros testes realizados, em ambiente fechado com iluminação não controlada não apresentou grande ganho em relação aos outros métodos, apresentando uma taxa de acerto semelhante, para certificar dos dados, foram feitos novos testes, desta vez em ambientes externos e interno

com variação de luz em ambos os lugares, forçando que a iluminação utilizada no *ground truth* original fosse diferente da utilizada nas novas imagens obtidas. Na *Tabela* 6 podemos uma comparação onde temos a taxa de acerto do GT (*ground truth*) fixo, e taxa de acerto com o GT dinâmico, no qual é definido de acordo com a classificação do usuário. Nestes testes verificou-se que o uso do *ground truth* dinâmico apresentou melhores resultados quando utilizando, principalmente quando utilizado em ambientes externo com sol, uma vez que o *ground truth* original foi produzido em ambiente Interno.

Tabela 6 - Quadro de comparação da taxa de acerto com *ground truth* dinâmico

Ambiente	**Iluminação**	**Acerto % *GT* Fixo**	**Acerto % *GT* Dinâmico**
Externo	Sombra	72%	84%
Externo	Sol	70%	87%
Interno	Iluminação Ambiente	74%	84%

Fonte: Elaborado pelo autor.

6 CONCLUSÕES

Este trabalho apresentou uma revisão bibliográfica e uma metodologia descrevendo o processo de contagem das moscas-das-frutas das espécies *C. capitata* e *A. fraterculus* assim como uma proposta de ferramenta de visão computacional baseada em smartphone.

Após a fase testes e obtenção dos resultados, verificou-se que o aplicativo proposto e atual trabalho estava de acordo a proposta inicial.

Neste trabalho também foram apresentados intervalo ideal de cor para detecção das moscas-das-frutas investigadas utilizando o espaço de cor HSV. Devido este modelo não ser capaz de distinguir as espécies das moscas-das-frutas foi analisado um classificador semi-supervisionado *Naive Bayes* assim como uma classificação utilizando a pré-classificação realizada pelo usuário e redefinindo o *ground truth* para identificação das moscas-das-frutas por meio da comparação de histogramas.

De acordo com os resultados apresentados foi possível demostrar a eficiência dos algoritmos e técnicas utilizadas para contagem das moscas-das-frutas.

6.1 Trabalhos Futuros

Este trabalho pode verificar algumas técnicas e verificar entre as elas suas respectivas taxas de acerto, toda via ficam alguns trabalhos futuros relativos ao aplicativo e algoritmos os seguintes pontos:

- Otimizar os algoritmos utilizados
- Expandir o *dataset* de imagens através das imagens obtidas pelo smartphone
- Explorar o uso dos recursos do Android como acelerômetro para verificar a inclinação do aparelho.
- Realizar um teste de campo com o aplicativo
- Realizar testes com outras espécies de moscas-das-frutas

Na área de pesquisa referente aprendizagem de máquina, o problema de identificação de moscas ou outros insetos pode ser também investigado através de técnicas de DL, utilizando redes neurais profundas ou redes neurais profundas convolucionais.

7 REFERÊNCIAS BIBLIOGRÁFICAS

ANDROID, Título: A. **Android Developer Guide.** Disponível em: < https://developer.android.com> Acesso em 10 de jan 2017.

BARBOSA, F. R.; GONC¸ ALVES, M. E. d. C.; MOREIRA, W. A.; ALENCAR, J. A. d.; SOUZA, E. A. d.; SILVA, C. S. da; SOUZA, A. d. M.; MIRANDA, I. d. G. Arthropodspest and predators associated with mango trees at the Vale do São Francisco, Northestern Brazil. **Neotropical Entomology**, [S.l.], v.34, n.3, p.471–474, 2005.

BAUCH, C.; RATH, T. Prototype of a vision based system for measurements of white fly infestation. In: **International Conference on Sustainable Greenhouse Systems-Greensys2004 691**. 2004. p. 773-780.

BENGIO, Y. Learning Deep Architectures for AI. **Foundations and Trends in Machine Learning**, Montreal, vol. 2, pp 2-127, 2009.

BOISSARD, P.; MARTIN, V.; MOISAN, S. A cognitive vision approach to early pest detection in greenhouse crops. Comput. Electron. Agric. v. 62, n. 2, p. 81-93, Julho, 2008.

Bradski, G., **Opencv Library**, Dr. Dobb's Journal of Software Tools, 2000.

BRAGA SOBRINHO, R.; MALAVASI, A.; OMETO, A. Manual operacional para levantamento, detecção, monitoramento e controle de moscas-das-frutas**. Embrapa Agroindústria Tropical. Circular técnica,** [S.l.], 2001.

BRAGA SOBRINHO, R.; SILVA, V.E. da; ARAUJO, E.L. de; FILGUEIRAS, A. L.; MESQUITA, A.L.M.; OMETO, A.C.F. Survey, identification and monitoring of fruit fly species in fruit production areas of the states of Rio Grande do Norte and Ceará - Brazil. In: CONGRESSO BRASILEIRO DE ENTOMOLOGIA, 16., 2000, Fortaleza. **Anais..**. Fortaleza: CBE, 2000. p. 638.

OLIVEIRA, M. R. V. de; MORAES, S. V. P. de; Moscas-das-Frutas Quarentenárias Potenciais para o Brasil. Brasilia: Embrapa Cerrados, 2006.

CANDEIAS, A. L. B.; SILVA, E. A. Extração de Estradas de uma imagem ETM+ Landsat usando Morfologia Matemática. In: Congresso Brasileiro de Cadastro Técnico Multifinalitário (COBRAC), 2004, Florianópolis. **Anais..**. Florianópolis, UFSC, 2004. Artigos, 8p.

CARVALHO, R.S. **Metodologia para monitoramento populacional de mosca-das-frutas em pomares comerciais.** Cruz das Almas: EMBRAPA/CNPMF, 2005. (EMBRAPA/ CNPMF. Comunicado Técnico, n. 75)

CHACON, G. T. **Aplicação de técnicas de processamento digital de imagens para a Detecção de MARFEs no JET**. 2012. Tese (Doutorado)-Centro Brasileiro De Pesquisas Físicas.

CHAO, R., MACÍA-VÁZQUEZ, G., ZALAMA, E., GÓMEZ-GARCÍA-BERMEJO, J., & Perán, J. R.. Automated Tracking of Drosophila Specimens. **Sensors**, v. 15, n. 8, p. 19369-19392, 2015.

CHAVES-GONZÁLEZ,CORKE, P. **Robotics, vision and control:** fundamental algorithms in MATLAB. [S.l.]: Springer, 2011. v.73.

COSTA, L. d. F. D.; CESAR JR, R. M. **Shape analysis and classification:** theory and practice. [S.l.]: CRC Press, Inc., 2000.

CUNHA, M. M. da; S. FILHO, H. P.; NASCIMENTO, A. S.; **Manga: fitossanidade: frutas do Brasil**. Brasília: Embrapa – SPI, 2000. 104 p.

DARWIN, Ian F. **Android Cookbook: Problemas e soluções para desenvolvedores Android**. Novatec Editora, 2012.

DE MEYER, M. Phylogeny of the genus *Ceratitis* (Dacinae: Ceratitidini). **Fruit ßies (Tephritidae): phylogeny and evolution of behavior. CRC, Boca Raton, FL,** [S.l.], p.409–428, 2000.

DING, Weiguang; TAYLOR, Graham. Automatic moth detection from trap images for pest management. **Computers and Electronics in Agriculture**, v. 123, p. 17-28, 2016.

DUDA, R. O., HART, P. E., STORK, D. G. **Pattern Classification**. Wiley-Interscience Publication, 2000.

FACHINELLO, J. C.; NACHTIGAL, R. C.; KERSTEN, E. Fruticultura: fundamentos e prática. Pelotas: Embrapa, 2010.

FACON, J. Morfologia Matemática: Teoria e Exemplos. **Curitiba: Editora Universitária Champagnat da Pontifícia Universidade Católica do Paraná**, 1996.

FAO. Food and Agriculture Organization of the United Nations - for a world without hunger. Disponível em: <http://faostat.fao.org> Acesso em: 25 nov. 2016.

FAUSETT, L. Fundamentals of neural networks: architectures, algorithms, and applications. **Neotropical Entomology**, [S.l.], 1994.

FIGUEIREDO, C. M.; NAKAMURA, E. Computação móvel: Novas oportunidades e novos desafios. **Belo Horizonte: Universidade Federal de Minas Gerais,** [S.l.], 2003.

FORSYTH, D. A. e PONCE, J. (2002). **Computer Vision: A Modern Approach.** Prentice Hall - US ed., 693 p. - ISBN: 0130851981.

GARTNER. **Technology Research — Gartner Inc.**

GHODS, SARA, AND VAHHAB SHOJAEDDINI. "A novel automated image analysis method for counting the population of whiteflies on leaves of crops." Journal of Crop Protection 5.1 (2015): 59-73.

GONZALEZ, R. C.; WOODS, R. E. Digital imaging processing. **Massachusetts: Addison-Wesley,** [S.l.], 1992.

GONZALEZ, R. C.; WOODS, R. E. **Processamento de imagens digitais**. [S.l.]: Edgard Blucher, 2000.

HINTON, G. E.; DENG, L.; YU, D.; DAHL, G. E.; MOHAMED, A.; JAITLY, N.; SENIOR, A.; VANHOUCKE, V.; NGUYEN, P.; SAINATH, T. N.; KINGSBURY, B. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. **IEEE Signal Process**, p. 82–97, 2012.

HUTCHINSON, S.; HAGER, G. D.; CORKE, P. I. A tutorial on visual servo control. **Robotics and Automation, IEEE Transactions on,** [S.l.], v.12, n.5, p.651–670, 1996.

WEIZHENG, S., YACHUN, W., ZHANLIANG, C., & HONGDA, W. (2008, December). Grading method of leaf spot disease based on image processing. In **Computer Science and Software Engineering, 2008 International Conference** on (Vol. 6, pp. 491-494). IEEE.

JAIN, K. A. **Fundamentals of Digital Image Processing**. 1 st ed. Englewood Cliffs: Prentice-Hall Information and System Science Series, 1989.

KAUR, Parmjit & SHARMA, Sumit. Google Android a Mobile Platform: A Review. Disponível em <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6799598>. Acesso em: 03/07/2014.

KIM, S.; CHOI, Y.; LEE, M. Deep learning with support vector data description. **Neurocomputing** 165. 2015. p. 111 – 117. Available at: <http://www.sciencedirect.com/science/article/pii/S092523121500380X>. Acessado em: Fevereiro 2017.

KUMARA, G. H. A. J. J., Kimitoshi Hayano, and Keita Ogiwara. "Image analysis techniques on evaluation of particle size distribution of gravel." Intern. J. of Geomate 3.1–5 (2012): 290-297.

LAGANI` ERE, R**. OpenCV 2 Computer Vision Application Programming Cookbook:** Over 50 recipes to master this library of programming functions for real-time computer vision. [S.l.]: Packt Publishing Ltd, 2011.

LAGANIÈRE ,. **OpenCV 2 Computer Vision Application Programming Cookbook**. 1. ed. [S.l.]: PACKT, 2011.

LECHETA, R. Aprenda a criar aplicações para dispositivos móveis com o Android SDK. **São Paulo, NOVATEC,** [S.l.], 2009.

LECHETA, R. R. **Google Android-3ª Edição:** Aprenda a criar aplicações para dispositivos móveis com o Android SDK. [S.l.]: Novatec Editora, 2013

LIU, Y., ZHANG, J., RICHARDS, M., PHAM, B., ROE, P. e CLARKE, A. (2009). Towards continuous surveillance of fruit flies using sensor networks and machine vision. Proceedings - 5th **International Conference on Wireless Communications, Networking and Mobile Computing**, WiCOM 2009.

LONG, F.; ZHANG, H.; FENG, D. D. Fundamentals of content-based image retrieval. In: **Multimedia Information Retrieval and Management.** [S.l.]: Springer, 2003.

LUGER, G. F. Artificial intelligence: structures and strategies for complex problem solving.[S.l.]: Pearson education, 2005.

MALAVASI, A.; R. A. ZUCCHI & R. L. SUGAYAMA. 2000. Biogeografia, p. 93-98. *In*: A. MALAVASI & R. A. ZUCCHI (eds.). **Moscas-das-frutas de importância econômica no Brasil: conhecimento básico e aplicado**. Ribeirão Preto, Holos Editora, 327 p.

METCALF, C. L.; FLINT, W. P.; METCALF, R. L. **Destructive and useful insects, their habits and control.** New York: McGraw-Hill Book Co., Inc., 1962.

MONTOYA, P.; TOLEDO, J.; BRECEDA, S. F. Trampas y atrayentes para la detección y monitoreo de moscas de la fruta (Diptera: Tephritidae). **Simposium sobre trampas y atrayentes en detección, monitoreo y control de plagas de importancia económica. JF Gaytan y PJ Montoya (Eds.), Sociedad Mexicana de Entomología y El Colegio de la Frontera Sur, Manzanillo, Colima, México,** [S.l.], p.17–25, 2006.

MUNSHI, A.; GASTER, B.; MATTSON, T. G.; GINSBURG, D. **OpenCL programming guide.** [S.l.]: Pearson Education, 2011.

NAVA, D.; BOTTON, M. Bioecologia e controle de *Anastrepha fraterculus* e *Ceratitis capitata* em pessegueiro. **Embrapa Clima Temperado. Documentos,** [S.l.], 2010.

NUNES, A. M.; MÜLLER, F. A.; GONC¸ALVES, R. d. S.; GARCIA, M. S.; COSTA, V. A.; NAVA, D. E. Frugivorous flies and their parasitoids in the cities of Pelotas

and Capão do Leão, Rio Grande do Sul, Brazil. **Ciência Rural,** v.42, n.1, p.6–12, 2012.

NUNES, A. N,; MÜLLER, F. A.; GONÇALVES, R. da S.; GARCIA, M. S.; COSTA, V. A.; NAVA, D. E. Moscas frugívoras e seus parasitoides nos municípios de Pelotas e Capão do Leão, Rio Grande do Sul, Brasil. Santa Maria: **Ciência Rural**, v.42, n.1, p.6-12, 2012.

OpenCV, **Use OpenCL in Android camera preview based CV application**, 2016. Disponível em:< http://docs.opencv.org/trunk/d7/dbd/tutorial_android_ocl_intro.html />. Acesso em: 16 de setembro. 2016.

PAL, N. R.; PAL, S. K. (1993) A Review on Image Segmentation Techniques. **Pattern Recognition**, v. 26, n. 9, p.1277-1294. DOI: 10.1016/0031-3203(93)90135-J.

PARANHOS, B. A. J. (2008) Moscas-das-frutas que oferecem riscos à fruticultura brasileira. Disponível em:< https://www.alice.cnptia.embrapa.br/alice/bitstream/doc/158610/1/OPB2070.pdf > p. 2-4

RUIZ, P.; MARCEDES, D.; ARDILES, C.; ISIDORA, A. Análisis y diseñõ de un circuito para lograr la automatización de las trampas usadas por SENASA en el monitoreo de la mosca de la fruta. **Ciência Rural,** [S.l.], 2011.

ROCHA, T. da. **Uma proposta para a classificação de ações humanas baseada nas características do movimento e em redes neurais artificiais.** 83f. 2012. Dissertação (Engenharia de Sistemas Eletrônicos e de Automação), Faculdade de Tecnologia, Departamento de Engenharia Elétrica - Universidade de Brasília.

ROGERS, R.; LOMBARDO, J.; MEDNIEKS, Z.; MEIKE, B. **Desenvolvimento de Aplicações Android**. [S.l.: s.n.], 2009.

SALLES, L.A.B. Moscas-das-frutas Anastrepha fraterculus (Wied., 1830): Bioecologia e controle. Pelotas: **EMBRAPA/CNPFT**, 1991.

SALLES, L.A.B.; KOVALESKI, A. Moscas-das-frutas em macieiras e pessegueiros no Rio Grande do Sul. Pelotas: **Horti Sul**, v.1, n.3, p.5-9, 1990.

SALOMON, D. **Data Compression the Complete Reference**. Third Edition. Springer, 2004.

SANTOS, J.P.; CORRENT, A.R.; BERTON, O.; SCHWARZ, L.L.; DENARDI, F. Incidência de podridão-branca em frutos de macieira com e sem ferimentos. **Revista Brasileira de Fruticultura**, v.30, n.1, p.118-121, 2008.

SHIVAKUMAR, P.; KISTLER, M.; KECKLER, S. W.; BURGER, D.; ALVISI, L. Modeling the effect of technology trends on the soft error rate of combinational

logic. In: DEPENDABLE SYSTEMS AND NETWORKS, 2002. DSN 2002. PROCEEDINGS. INTERNATIONAL CONFERENCE ON, 2002. **Anais**… [S.l.: s.n.], p.389–398, 2002.

SOLOMON, C.; BRECKON, T. **Fundamentals of Digital Image Processing**: A practical approach with examples in Matlab. [S.l.]: John Wiley & Sons, 2011.

SOUZA FILHO; M.F.; RAGA. A.; ZUCCHI, R.A. São Paulo. In: MALAVASI, A.; ZUCCHI, R.A. (Ed.). **Moscas-ds-frutas de importância econômica no Brasil: conhecimento básico e aplicado.** Ribeirão Preto: Holos Editora, 2000. p. 277-283.

SUGAYAMA, R.; MALAVASI, A. Ecologia comportamental. In: A. Malavasi & R.A. Zucchi (Eds.), **Moscas-das-frutas de importância econômica no Brasil: conhecimento básico e aplicado.** Ribeirão Preto: Holos. 2000. p103-108.

SZELISKI, R. (2011) Computer Vision: Algorithms and Applications. Texts in Computer Science. Londres: Editora Springer. 812 p.

THEODORIDIS, S., KOUTROUMBAS, K. Pattern Recognition. 4 ed. San Diego, California, USA, Academic Press, 2009.

TOYAMA, K.; KRUMM, J.; BRUMITT, B.; MEYERS, B. (1999) Wallflower: Principles and Pratice of Background Maintenance**. IEEE International Conference on Computer Vision**, v.1, p. 255-261. DOI: 10.1109/ICCV.1999.791228.

TUPINAMBÁ, A. L. R. DistributedCL: middleware de processamento distribuído em GPU com interface da API OpenCL., [S.l.], 2013.

VERKASALO, Hannu. Analysis of smartphone user behavior. In: **Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR), 2010 Ninth International Conference on**. IEEE, 2010. p. 258-263.

WEIZHENG, S.; YACHUN, W.; ZHANLIANG, C.; HONGDA, W. Grading method of leaf spot disease based on image processing. In: COMPUTER SCIENCE AND SOFTWARE ENGINEERING, 2008 INTERNATIONAL CONFERENCE ON, 2008. **Anais**... [S.l.: s.n.],. v.6, p.491–494, 2008.

ZHU, Le-Qing; ZHANG, Zhen. Auto-classification of insect images based on color histogram and GLCM. In: **Fuzzy Systems and Knowledge Discovery (FSKD), 2010 Seventh International Conference on**. IEEE, 2010. p. 2589-2593.

ZUCCHI, R.A. 2008. Fruit flies in Brazil - Anastrepha species their host plants and parasitoids. Disponível em: <www.lea.esalq.usp.br/anastrepha/>. Acesso em: 25 nov. 2016.

Apêndice A – Código Android Para Detecção da Peneira Criação da Máscara.

```
protected void processImage() {
    if (null == cameraDevice) {
        return;
    }

    continueProcess = false;

    CameraManager manager = (CameraManager)
getSystemService(Context.CAMERA_SERVICE);

    try {
        CameraCharacteristics characteristics =
manager.getCameraCharacteristics(cameraDevice.getId());
        Size[] jpegSizes = null;
        if (characteristics != null) {
            jpegSizes =
characteristics.get(CameraCharacteristics.SCALER_STREAM_CONFIGU
RATION_MAP).getOutputSizes(ImageFormat.JPEG);
        }

        int width = 4096;
        int height = 2160;

        if (jpegSizes != null && 0 < jpegSizes.length) {
            width = jpegSizes[0].getWidth();
            height = jpegSizes[0].getHeight();
        }

        ImageReader reader = ImageReader.newInstance(width,
height, ImageFormat.JPEG, 1);
        List<Surface> outputSurfaces = new
ArrayList<Surface>(2);
        outputSurfaces.add(reader.getSurface());
        outputSurfaces.add(new
Surface(textureView.getSurfaceTexture()));
        final CaptureRequest.Builder captureBuilder =
cameraDevice.createCaptureRequest(CameraDevice.TEMPLATE_STILL_C
APTURE);
        captureBuilder.addTarget(reader.getSurface());
        captureBuilder.set(CaptureRequest.CONTROL_MODE,
CameraMetadata.CONTROL_MODE_AUTO);

        captureBuilder.set(CaptureRequest.JPEG_QUALITY, (byte)
100);

        captureBuilder.set(CaptureRequest.CONTROL_SCENE_MODE,
CaptureRequest.CONTROL_SCENE_MODE_HDR);


        context = this;

        final ImageReader.OnImageAvailableListener
readerListener = new ImageReader.OnImageAvailableListener() {
            @Override
            public void onImageAvailable(ImageReader reader) {
                Image image;
```

```
                image = reader.acquireLatestImage();
                ByteBuffer buffer =
image.getPlanes()[0].getBuffer();
                byte[] bytes = new byte[buffer.capacity()];
                buffer.get(bytes);

                image.close();


                final Bitmap bmp =
BitmapFactory.decodeByteArray(bytes, 0, bytes.length);
                Bitmap myBitmap32 =
bmp.copy(Bitmap.Config.RGB_565, true);

                Mat mImage = new Mat(bmp.getHeight(),
bmp.getWidth(), CvType.CV_8UC4);

                Utils.bitmapToMat(myBitmap32, mImage);


                //Start OpenCv
                MatOfRect circles = new MatOfRect();
                //Convert to gray
                Mat mGray = new Mat();
                Mat mImageTemp = new Mat();
                Imgproc.resize(mImage, mImageTemp, new
org.opencv.core.Size(mImage.width() / 16, mImage.height() /
16));
                Imgproc.cvtColor(mImageTemp, mGray,
Imgproc.COLOR_BGR2GRAY);

                Imgproc.HoughCircles(mGray, circles,
Imgproc.CV_HOUGH_GRADIENT, 2, 300, 30, 60, 60, 300);


                float circle[] = new float[3];
                if (circles.cols() > 0) {

                    new ProcessImageAsyncTask(context, bytes) {
                        @Override
                        protected void onPostExecute(Void
aVoid) {
                            super.onPostExecute(aVoid);

                            Intent myIntent = new
Intent(AndroidCameraApi.this, PreviewActivity.class);

AndroidCameraApi.this.startActivity(myIntent);
                        }
                    }.execute();


                }

            }

        };

        reader.setOnImageAvailableListener(readerListener,
mBackgroundHandler);
```

```
        final CameraCaptureSession.CaptureCallback
captureListener = new CameraCaptureSession.CaptureCallback() {
            @Override
            public void onCaptureCompleted(CameraCaptureSession
session, CaptureRequest request, TotalCaptureResult result) {
                super.onCaptureCompleted(session, request,
result);
                createCameraPreview();
            }
        };
        cameraDevice.createCaptureSession(outputSurfaces, new
CameraCaptureSession.StateCallback() {
            @Override
            public void onConfigured(CameraCaptureSession
session) {
                try {
                    session.capture(captureBuilder.build(),
captureListener, mBackgroundHandler);
                } catch (CameraAccessException e) {
                    e.printStackTrace();
                }
            }

            @Override
            public void onConfigureFailed(CameraCaptureSession
session) {
            }
        }, mBackgroundHandler);
    } catch (CameraAccessException e) {
        e.printStackTrace();
    }


}
```

Apêndice B – Código Android Para Remover Background da peneira

```
Thread.sleep(0);


//Convert to Mat
final Bitmap bmp = BitmapFactory.decodeByteArray(bytes, 0,
bytes.length);
Bitmap myBitmap32 = bmp.copy(Bitmap.Config.RGB_565, true);

Mat mImage = new Mat(bmp.getHeight(), bmp.getWidth(), CvType.CV_8UC4);

Utils.bitmapToMat(myBitmap32, mImage);


//Start OpenCv
MatOfRect circles = new MatOfRect();
//Convert to gray
Mat mGray = new Mat();
Mat mImageTemp = new Mat();
Imgproc.resize(mImage, mImageTemp, new
org.opencv.core.Size(mImage.width() / 16, mImage.height() / 16));
Imgproc.cvtColor(mImageTemp, mGray, Imgproc.COLOR_BGR2GRAY);
Log.d("Debug", mGray.width() + " X " + mGray.height());
//Detecta circulos
Imgproc.HoughCircles(mGray, circles, Imgproc.CV_HOUGH_GRADIENT, 2,
300, 30, 60, 60, 300);


float circle[] = new float[3];
if (circles.cols() > 0) {
    circles.get(0, 0, circle);

    org.opencv.core.Point center = new org.opencv.core.Point();
    center.x = circle[0];
    center.y = circle[1];

    OpenCVBusiness openCVBusiness = new OpenCVBusiness(context);
    Mat mask = openCVBusiness.createMaskHardCode(mImage, circle[0] *
32, circle[1] * 32, (circle[2] * 32) + 70);


    Mat imagemMascarada = new Mat(mask.rows(), mask.cols(),
CvType.CV_8UC3);
    imagemMascarada.setTo(new Scalar(0, 0, 0, 0));
    Imgproc.cvtColor(mImage, mImage, Imgproc.COLOR_RGB2BGRA);

    mImage.copyTo(imagemMascarada, mask);
    Imgproc.cvtColor(imagemMascarada, imagemMascarada,
Imgproc.COLOR_RGBA2BGR);

    center.x = circle[0] * 16;
    center.y = circle[1] * 16;
    Imgproc.circle(imagemMascarada, center, (int) ((circle[2]) * 16) +
70, new Scalar(0, 0, 255, 150), 8);


    Bitmap bmpImageCrop = Bitmap.createBitmap(imagemMascarada.width(),
imagemMascarada.height(), Bitmap.Config.ARGB_8888);
```

```
    //Desenenha um retangulo, INRECT
    Imgproc.rectangle(imagemMascarada, new Point(1400,1100), new
Point(2100,1400), new Scalar(255, 0, 0, 150), 4);

    Utils.matToBitmap(imagemMascarada, bmpImageCrop);



    saveBitmap.save(bmpImageCrop, lastInsertId + "_clean", false, "",
true);

    openCVBusiness.processImage(imagemMascarada);
```

Apêndice C – Código de obtenção da comparação do histograma da anastrepha fraterculus

```
public double calcHistogramA(Bitmap imageToMatch) throws IOException {

    double res = 0;
    Bitmap bmp;
    try {
        if (this.calcHistogramSingleton) {
            AssetManager assetManager = this.context.getAssets();
            InputStream is =
assetManager.open("histogram/histo_bq_a.jpg");
            Bitmap histoBq = BitmapFactory.decodeStream(is);
            this.imgHistoBq = new Mat(histoBq.getHeight(),
histoBq.getWidth(), 0, new Scalar(4));
            bmp = histoBq.copy(Bitmap.Config.RGB_565, true);
            Utils.bitmapToMat(bmp, this.imgHistoBq);
            this.calcHistogramSingleton = false;
        }

        Mat imgToCalc = new Mat(imageToMatch.getHeight(),
imageToMatch.getWidth(), 0, new Scalar(4));
        bmp = imageToMatch.copy(Bitmap.Config.RGB_565, true);
        Utils.bitmapToMat(bmp, imgToCalc);
        Mat hist0 = new Mat();
        Mat hist1 = new Mat();
        MatOfFloat ranges = new MatOfFloat(0f, 256f);
        MatOfInt histSize = new MatOfInt(25);

        Imgproc.calcHist(Arrays.asList(this.imgHistoBq), new
MatOfInt(0), new Mat(), hist0, histSize, ranges);
        Imgproc.calcHist(Arrays.asList(imgToCalc), new MatOfInt(0),
new Mat(), hist1, histSize, ranges);

        res = Imgproc.compareHist(hist0, hist1,
Imgproc.CV_COMP_CORREL);

    } catch (IOException e) {
        e.printStackTrace();
    }

    return res;
}
```

Apêndice D – Exemplos de resultados da detecção semi-supervisionada.