José Ricardo Ferreira Cardoso

Desenvolvimento de Estrutura Robótica para Aquisição e Classificação de Imagens (ERACI) de Lavoura de Cana-de-Açúcar

Brasil

2020

José Ricardo Ferreira Cardoso

Desenvolvimento de Estrutura Robótica para Aquisição e Classificação de Imagens (ERACI) de Lavoura de Cana-de-Açúcar

Dissertação apresentada à Faculdade de Ciências Agrárias e Veterinárias – Unesp, Campus de Jaboticabal, como parte das exigências para a obtenção do título de Mestre em Agronomia

Universidade Estadual Paulista

Faculdade de Ciências Agrárias e Veterinárias de Jaboticabal

Programa de Pós-Graduação em Agronomia (Ciência do Solo)

Orientador: Dr. Carlos Eduardo Angeli Furlani

Coorientador: Dr. José Eduardo Pitelli Turco

Brasil

2020

C268d

Cardoso, José Ricardo Ferreira
Desenvolvimento de Estrutura Robótica para Aquisição e Classificação de Imagens (ERACI) de Lavoura de Cana-de-Açúcar / José Ricardo Ferreira Cardoso. -- Jaboticabal, 2020
147 p. : il., tabs., fotos, mapas

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências Agrárias e Veterinárias, Jaboticabal
Orientador: Carlos Eduardo Angeli Furlani
Coorientador: José Eduardo Pitelli Turco

1. Agricultura de Digital. 2. OpenCV. 3. Open Source. 4. Raspberry Pi. 5. Visão Computacional. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências Agrárias e Veterinárias, Jaboticabal. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

UNIVERSIDADE ESTADUAL PAULISTA

Câmpus de Jaboticabal

CERTIFICADO DE APROVAÇÃO

TÍTULO DA DISSERTAÇÃO: DESENVOLVIMENTO DE ESTRUTURA ROBÓTICA PARA AQUISIÇÃO E CLASSIFICAÇÃO DE IMAGENS (ERACI) DE LAVOURA DE CANA-DE-AÇÚCAR

AUTOR: JOSÉ RICARDO FERREIRA CARDOSO
ORIENTADOR: CARLOS EDUARDO ANGELI FURLANI

Aprovado como parte das exigências para obtenção do Título de Mestre em AGRONOMIA (CIÊNCIA DO SOLO), pela Comissão Examinadora:

Prof. Dr. CARLOS EDUARDO ANGELI FURLANI
Departamento de Engenharia Rural / FCAV / UNESP - Jaboticabal

Dr. JONES MENDONÇA DE SOUZA
Instituto Federal de São Paulo-IFSP/Campus Barretos / Barretos/SP
(VIDEOCONFERÊNCIA)

Prof. Dr. CRISTIANO ZERBATO
Departamento de Engenharia Rural / FCAV / UNESP - Jaboticabal
(VIDEOCONFERÊNCIA)

Jaboticabal, 29 de junho de 2020

Este trabalho é dedicado a todos aqueles que
acreditam que a vida é sagrada e agem para valoriza-la. Em especial, dedico ao meu irmão Luiz Henrique Ferreira Cardoso (in memorian) que demonstrou durante toda a sua vida esta maravilhosa verdade.

Agradecimentos

Agradeço primeiramente a Deus que me cumulou dos dons necessários para desenvolver este projeto. Agradeço também aos meus pais, José Cardoso e Cleide Ferreira Cardoso, que me criaram com os valores de respeito, amor e força de vontade; graças a seus ensinamentos consegui atingir os objetivos traçados até aqui. Agradeço a minha esposa Joyce Aparecida Santana e minha filha Gabriela Aparecida Santana Cardoso que me suportaram e me apoiaram durante todo o período em que me preparei e desenvolvi este trabalho. Agradeço ao meu irmão Luiz Henrique Ferreira Cardoso (*in memorian*) por sempre ter me inspirado a buscar sempre a vitória com respeito ao próximo e a si mesmo. Agradeço também aos meus amigos Jones, Tiago e João Paulo que estiveram sempre ao meu lado quando precisei de ajuda. Por fim, Agradeço também aos meus orientadores Carlos Eduardo Angeli Furlani e José Eduardo Pitelli Turco que acreditaram na minha capacidade e me ajudaram sempre que foi necessário. A todos meu muito obrigado.

"(...) A ciência e a técnica estão ordenados para o homem,
do qual provêm sua origem e seu crescimento;
portanto, encontram na pessoa e em seus valores morais a indicação
de sua finalidade e a consciência de seus limites.
(Catecismo da Igreja Católica, 2293)

Resumo

A agricultura digital tem contribuído com a melhoria da eficiência na aplicação de insumos ou no plantio em local pré-determinado, resultando no aumento da produtividade. Nesta realidade a aplicação de técnicas de Processamento de Imagens Digitais, bem como a utilização de sistemas que utilizam a Inteligência Artificial, tem ganhado cada vez mais a atenção de pesquisadores que buscam a sua aplicação nos mais diversos meios. Com o objetivo de desenvolver um sistema robótico que utiliza um sistema de visão computacional capaz analisar uma imagem e, detectar basicamente a presença de cana-de-açúcar e planta daninha, bem como a ausência de qualquer planta, o projeto desenvolvido unificou conhecimentos sobre estas duas áreas da ciência da computação com a área de robótica e agricultura que, culminou no desenvolvimento de uma estrutura robótica com ferramentas gratuitas, como é o caso dos softwares e hardwares modulares voltados para o ensino de informática em escolas. A união de tudo isso resultou em uma estrutura de software e hardware que captura e armazena imagens em um banco de dados; além de possibilitar a classificação de imagens pelos usuários habilitados por meio de aplicativo Android. Por meio da verificação da acurácia entregue pelos algoritmos de Machine Learning, com injeção cíclica e, pela análise do tempo de resposta, foi constatado que o sistema é capaz, munido destas informações, de gerar classificadores que, remotamente são carregados pelo DRR (Dispositivo Robótico Remoto) e estes, por sua vez, foram capazes de classificar imagens em lavoura de cana-de-açúcar em tempo real.

Palavras-chave: Agricultura de Digital, OpenCV, Open Source, Raspberry Pi[1], Visão Computacional.

[1] Pequeno computador utilizado para o ensino de computação e desenvolvimento de projetos (RASPBERRY PI FOUNDATION, 2020)

Abstract

Digital agriculture has contributed to improving efficiency in the application of inputs or planting in a predetermined location, resulting in increased productivity. In this reality, the application of Digital Image Processing techniques, as well as the use of systems that use Artificial Intelligence, has increasingly gained the attention of researchers who seek its application in the most diverse media. In order to develop a robotic system that uses a computer vision system capable of analyzing an image and, basically detecting the presence of sugarcane and weed, as well as the absence of any plant, the project developed unified knowledge about these two areas of computer science with the area of robotics and agriculture that culminated in the development of a robotic structure with free tools, such as software and modular hardware aimed at teaching computer science in schools. The combination of all this resulted in a software and hardware structure that captures and stores images in a database; in addition to enabling the classification of images by users enabled through the Android application. By checking the accuracy delivered by the Machine Learning algorithms, with cyclic injection and, by analyzing the response time, it was found that the system is capable, equipped with this information, to generate classifiers that are remotely loaded by the RRD (Remote Robotic Device) and these, in turn, were able to classify images in sugarcane fields in real time.

Keywords: Computer Vision, OpenCV, Open Source, Digital Agriculture, Raspberry Pi.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

AF	Área Foliar
AOT	*Ahead Of Time*
AP	Agricultura de Precisão
API	*Application Programming Interface*
APK	*Android Package*
ART	*Android Runtime*
BD	Banco de Dados
CPU	*Central Process Unit*
CSI	*Camera Serial Interface*
DAO	*Data Access Object*
DEX	*Dalvik Executable*
DGGC	Dispositivo Gerador e Gestor de Conhecimento
DM	Dispositivo Móvel
DPI	*Dots Per Inch*
DRR	Dispositivo Robótico Remoto
DT	*Decision Tree*
ERACI	Estrutura Robótica para Aquisição e Classificação de Imagens
FD	Forma como o Sistema foi Desligado
GC	*Garbage Colector*
GPIO	*General Purpose Input/Output*
GPS	*Global Positioning System*
HAL	*Hardware Abstraction Layer*
HDMI	*High-Definition Multimedia Interface*

HSV	*Hue, Saturation and Value*
HTTP	*HyperText Transfer Protocol*
IA	Inteligência Artificial
IBGE	Instituto Brasileiro de Geografía e Estatística
IDL	*Interface Definition Language*
IFSP	Instituto Federal de São Paulo
IoT	*Internet of Things*
IP	*Internet Protocol*
JIT	*Just in Time*
JPEG ou JPG	*Joint Photographic Experts Group*
JVM	*Java Virtual Machine*
KNN	*K-Nearest Neighbors*
LED	*Light Emitting Diode*
LiDAR	*Light Detection And Ranging*
MDM	*Multiscale Distance Matrix*
ML	*Machine Learn*
MLP	*Multilayer Perceptron*
NB	*Naive Bayes*
NDVI	*Normalized Difference Vegetation Index*
OpenCV	*Open Source Computer Vision*
ORM	*Object Relacionl Mapping*
PID	Processamento de Imagens Digitais
pixel	*Picture elements*
QMR	Quantidade Mínima para Regularidade
REST	*Representational State Transfer*
RF	*Random Forest*

RGB	*Red, Green and Blue*
RL	*Logistic Regression*
RPI	*Raspberry Pi*
SIG	Sistema de Informação Gerencial
SO	Sistema Operacional
SSH	*Secure Shell*
SSID	*Service Set Identifier*
SVM	*Support Vector Machine*
TCP/IP	*Transmission Control Protocol/Internet Protocol*
TMF	Tempo máximo de Funcionamento
TSF	Tempo que o Sistema Permaneceu em Funcionamento
TTL	*Time to Live*
UDP/IP	*User Datagram Protocol/Internet Protocol*
UML	*Unified Model Language*
Unesp	Universidade Estadual Paulista
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
USB	*Universal Serial Bus*
WAN	*Wide Area Network*
XML	*Extensible Markup Language*

Sumário

1 Introdução

A cultura da cana-de-açúcar se destaca no Brasil devido à importância de seus produtos finais, mesmo com as dificuldades relativas ao custo de produção que, segundo a (CONAB, 2020) concentra-se principalmente na colheita, devido às exigências socioambientais. Além disso, outros fatores podem influenciar no desperdício de insumos, como por exemplo, a dispensa de fertilizantes ou herbicidas em área de manobra de veículos. Estes fatos tornam a aplicação de insumos de forma eficiente, algo indispensável. Outro fato a ser considerado é que, a aplicação destes insumos pode ocorrer de diversas formas dependendo do momento em que se aplica. Esta aplicação ocorrer durante o pré-plantio (ocorrendo diretamente nos sulcos), durante os estádios de crescimento e durante a pós colheita. (VITTI; OTTO; FERREIRA, 2015). Dentre as técnicas utilizadas para esta finalidade está a utilização de equipamentos munidos de sistemas de informação, capazes de efetuar as ações necessárias nos locais previamente determinados (BERNARDES; BELARDO, 2015).

A aplicação, tanto de nutrientes quanto de defensivos em culturas cultivadas em fileiras, deve levar em conta a densidade populacional da planta, que é data pela combinação do espaçamento entre fileiras com o número de plantas por unidade de distância na fileira, bem como a existência de áreas não cultivadas utilizadas para manobra do maquinário agrícola (BERNARDES; BELARDO, 2015). Para melhorar a eficiência na aplicação destes insumos e assim reduzir desperdícios, é feito uso de várias tecnologias no âmbito da Agricultura de Precisão (AP), como o geoprocessamento, para permitir o mapeamento da fertilidade e o desenvolvimento da cultura com intervenção de manejo localizado (RESENDE; COELHO, set. 2017).

A utilização de equipamentos que obtém informações da sanidade das plantas como, sensor de refletância do dossel, sensores NDVI (*Normalized Difference Vegetation Index*), que apresentam respostas em tempo real, tem sido utilizada com mais frequência no campo e, tem sido uma das opções mais promissoras para suprir o gargalo tecnológico (CASTRO, 2016). Além disso, a difusão dos dispositivos interconectados pela rede IoT (*Internet of Things*) são uma tendência clara, juntamente com o desenvolvimento de equipamentos para o sensoriamento remoto (GIMENEZ; MOLIN, 2018).

Toda esta preocupação com a eficiência gerou a produção de diversos trabalhos científicos que foram absorvidos pela indústria de equipamentos agrícola e, resultou na construção de novos equipamentos (SPEKKEN; BRUIN; MOLIN, 2014), como o trator autônomo apresentado pela Case IH®, em 2017 em um evento do setor agrícola. A empresa disse durante este evento que:

> ´´este trator tem como finalidade o aumento da produtividade e precisão na atividade agrícola, permitindo a operação em 24 h".

O veículo em questão tem incorporado em si múltiplas câmeras e GPS e, faz uso de um sistema óptico de detecção remota chamado de LiDAR® (*Light Detection And Ranging*) que dá a capacidade ao veículo de medir as propriedades da luz refletida e obter a distância em relação a um objeto (SUPPA, 2020).

Também, neste cenário, a Trimble® comercializa um sistema de pulverização chamado de WeedSeeker® que pode ser acoplado na barra do pulverizador de qualquer marca e modelo de máquina. Este sistema é composto de um sensor NDVI (*Normalized Difference Vegetation Index*) ativo e, pode ser regulado para trabalhar em um intervalo de 60 cm de altura em relação ao solo com o propósito de possibilitar uma pulverização seletiva diretamente na planta de interesse (TRIMBLE, 2020) .

Segundo (Khmag; Al-Haddad; Kamarudin, 2017) o reconhecimento da planta de interesse é importante para que se possa distinguir se deve ou não ocorrer a aplicação de um determinado nutriente ou defensivo. Para isso as folhas de plantas provaram ser fontes viáveis de informação e, é amplamente utilizada na identificação das diferentes espécies. Atualmente, esta tarefa é realizada por especialistas, o que, diante das necessidades de alta produtividade no campo, é inviável de ser aplicado.

Dentro do contexto da utilização de tecnologias aplicadas na agricultura, há diversos estudos científicos que utilizam técnicas de Processamento Imagens Digitais (PID) e Inteligência Artificial (IA) para a captura, segmentação, extração de características e a classificação de uma imagem (GONZALEZ; WOODS, 2010; LUGER, 2013). Segundo (KANG; OH, 2018) a forma com que as imagens são obtidas, uma com o objeto de estudo sobre um fundo branco e, outra sem qualquer preparação prévia, merece atenção especial, podendo a escolha implicar em um alto desempenho do sistema e sacrifício da conveniência do usuário, enquanto, outra pode trazer grande conveniência para o usuário, porém, com sacrifício do desempenho do sistema.

A preocupação com o rápido desenvolvimento de novas tecnologias, eliminando tarefas baseadas em tecnologia já existentes no mercado, tem levado a difusão e utilização de *software*, *framework* e *hardware Open Source*. Neste sentido, uma única placa de computador de baixo custo como o *Raspberry Pi* ® (RPI), que permite utilizar bibliotecas de processamento de imagens, como o *Open Source Computer Vision* – OpenCV que pode ser utilizado para a construção de dispositivos robóticos que permitem o processamento e o reconhecimento de padrões em imagens (OSROOSH; KHOT; PETERS, 2018).

Baseado nos trabalhos disponibilizados em diversos meios, tanto acadêmicos quanto os mercadológicos, a agricultura de precisão é uma das áreas mais promissoras para a redução de custos e o aumento da produtividade. Isto se dá devido a suas técnicas, que

levam em conta a variabilidade nos mais diversos âmbitos da agricultura.

Considerando que, a tecnologia da informação e a eletrônica são meios pelos quais esta forma de fazer agricultura tem atuado no campo, a produção de um sistema de visão computacional é justificado, pelo que se espera em ganhos na eficiência e redução de custos para o agricultor, assegurando, principalmente, a mitigação de custos relacionados ao desperdício de insumos, como por exemplo, fertilizantes e herbicidas.

Tudo isto leva ao questionamento de sobre a possibilidade de ser criado um sistema robótico capaz de permitir a aquisição, armazenamento, e o reconhecimento de padrões em imagens, por meio de software que utiliza visão computacional para analisar uma imagem e, detectar basicamente a presença de cana-de-açúcar e planta daninha, bem como a ausência de qualquer planta utilizando *software* e *hardware open source* voltado, basicamente, para o uso escolar (RASPBERRY PI FOUNDATION, 2020) e, com característica modular.

Para responder a este questionamento, no presente trabalho é apresentado os resultados obtidos com o desenvolvimento da primeira parte do projeto que tem como objetivo geral a implementação de uma estrutura robótica capaz de detectar a presença de cana-de-açúcar e planta daninha, bem como a ausência de qualquer planta. Com este fim foram estipulados os seguintes objetivos específicos: a) adquirir e armazenar imagens em um banco de dados; b) permitir a geração de conhecimento computacional por meio de algoritmos de visão computacional e gerir este conhecimento; e c) realizar reconhecimento de padrões em imagem em tempo real.

2 Revisão Bibliográfica

2.1 Cana-de-Açúcar

A cana-de-açúcar é originária da Nova-Gjiné, de onde ela foi levada para o sul da Ásia. A primeira evidencia de seu estado sólido data de 500 a.C., na Pérsia. No Brasil ela foi trazida pelos Portugueses no fim do século XVI. Depois disso foi levada para as Américas Central e do Sul (MOZAMBANI, 2006).

Ela é uma cultura de grande importância econômica para o Brasil e, diversos Estados brasileiros apresentam condições climáticas favoráveis ao seu cultivo que se dá principalmente entre os paralelos 35º LN e 35º LS com duração do período de crescimento vegetativo variando de 9 meses a mais de 2 anos, conforme o local onde ocorreu o plantio (AUDE, 1993).

A finalidade principal desta planta é a produção de sacarose para a fabricação de açúcar e álcool como fonte alternativa de combustível; este último garantiu o crescimento desta cultura, a partir do ano de 1975, com a criação do programa Proálcool. Com isso, ela foi classificada como a segunda mais importante no ranking, em termos de valores, conforme a análise sobre a geração de valores realizada pelo IBGE no ano de 2013. Esta importância é incrementada quando se leva em consideração a movimentação de toda cadeia produtiva como os relacionados aos suprimentos de insumos, máquinas e implementos, comércio e indústria, não se resumindo apenas para o mercado, mas também pela sua contribuição ambiental, permitindo a disponibilização de energia renovável e a promoção do aumento da carga de matéria orgânica no solo por meio de seus subprodutos. Além disso, no campo energético ela tem o importante papel de substituir compostos cancerígenos adicionados a gasolina devido à característica de elevada octanagem (NASTARI, 2015).

A cana-de-açúcar (*Saccharum spp.*) tem seu desenvolvimento ocorrendo na forma de touceira com a parte aérea formada por colmos, folhas e inflorescências. A parte subterrânea é formada basicamente por raízes e rizoma (ARALDI et al., 2010). Após o plantio em condições ambientais favoráveis, são iniciadas as atividades meristemáticas nos primórdios radiculares e no poro da gema, culminando com o desenvolvimento das raízes do tolete e a emergência de um pequeno broto na superfície do solo. Depois de determinado estádio de desenvolvimento as gemas localizadas na base do colmo primário se intumescem originando o broto. A fase de perfilhamento intenso da touceira se dá quando é atingido o máximo da produção de perfilhos, chegando algumas variedades a produzir 25 ou mais colmos por touceira (MAGRO C. R.; LACA-BUENDIA, 2010).

Na inflorescência ela deixa de produzir folhas e colmos indicando o período em que

a mesma deve ser colhida. Este florescimento ocorre pelo estímulo e pela diferenciação meristemática que no hemisfério sul ocorrem naturalmente entre os meses de fevereiro a abril com o florescimento acontecendo entre os meses de abril a junho. A interação entre o fotoperíodo, umidade e a radiação solar podem manter ou prevenir a transformação do ápice da planta sendo o fotoperíodo o maior influenciador. Esta florescência está diretamente relacionada com a produtividade, sendo recomendado, quando existe a possibilidade de manejo varietal, a utilização de variedades com potencial menos ou não florífero. Quando não é possível esse manejo é indicado o uso de produtos inibidores do florescimento visto que as perdas com a interversão da sacarose para formação da panícula, durante o florescimento, são bem relevantes (ARALDI et al., 2010).

Do ponto de vista morfológico (MAGRO C. R.; LACA-BUENDIA, 2010) apresentaram um trabalho que permite a observação de características da touceira de cana-de-açúcar. Para isso eles sortearam 1,0 m em cada parcela e mediram o comprimento e a largura da porção mediana de uma folha +3 para determinar a área foliar (AF). Como resultado de seu experimento chegaram a uma variação na área foliar de 80.902,16 a 36.769,20 cm^2; a altura da touceira teve uma variação de 0,49 a 0,45 m e; o diâmetro do colmo teve variação de 16,55 a 17,03 mm. Todos estes valores foram levantados por meio da análise da cana plantada entre as profundidades de 10 a 50 cm.

Um dos fatores que influenciam negativamente no desenvolvimento da cultura

Figura 1 – Lavoura de Cana-de-açúcar na Fazenda do IFSP-campus Barretos.

Fonte: Autoria Própria

são as plantas daninhas que consequentemente interferem na sua produtividade. O grau de competição na associação mato-cultura depende da intervenção de fatores ligados à comunidade infestante, que envolvem a composição específica, distribuição, cultura, espécie, espaçamento, densidade, época e duração do período de convívio entre as culturas, além das condições edáficas e climáticas, bem como de seus tratos. A ocorrência de um ou mais desses componentes de interferência poderá reduzir a quantidade de colmos colhidos e diminuir o número de cortes economicamente viáveis. Com base nisso (FIGUEIREDO et al., 2013) concluíram com seu experimento que a mato competição das espécies *Brachiaria brizantha* e *Brachiaria decumbens*, consideradas invasoras, provocaram a redução das características morfoanatômicas e de produção da cana-de-açúcar, sendo a segunda a espécie que mais influenciou negativamente sua espessura foliar.

Mesmo com todo avanço, o setor tem momentos em que apresenta reflexos negativos na produtividade e nos custos de produção, o que exige adaptações importantes nas

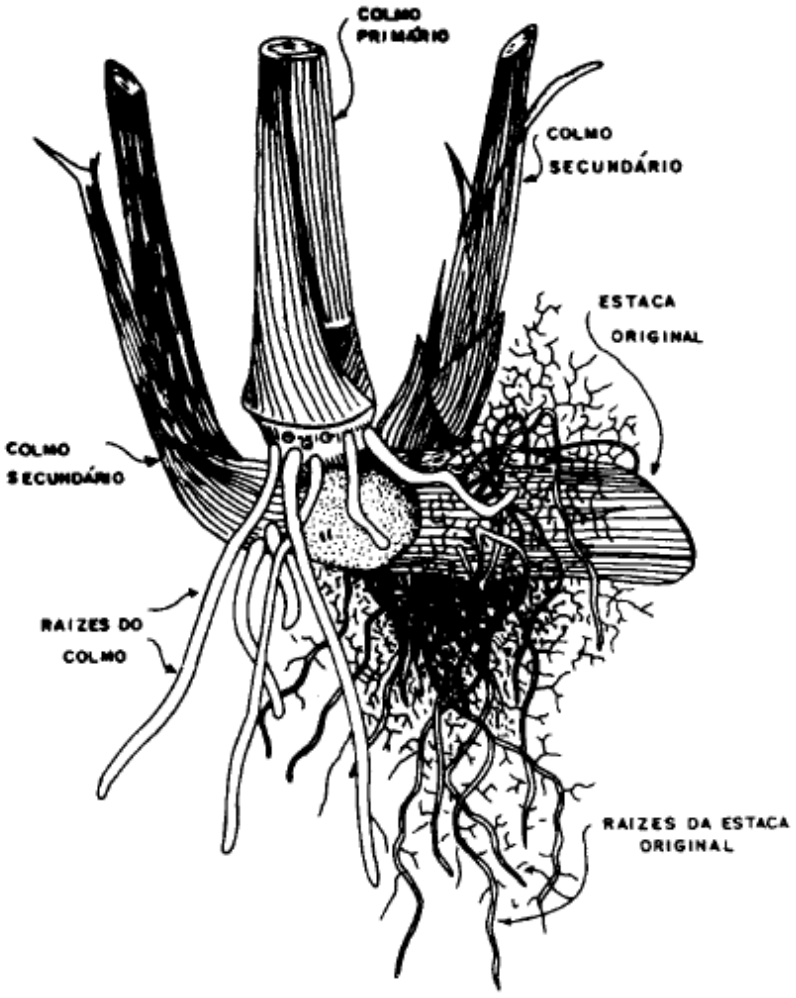


Figura 2 – Principais constituintes que se desenvolvem a partir do tolete da cana-de-açúcar durante o perfilamento.

Fonte: (MALAVOLTA; HAAG, 1964) apud (ARALDI et al., 2010)

técnicas, equipamentos e procedimentos adotados na produção. Assim, foi bem recebida a mecanização tanto do plantio quanto da colheita e, a melhoria das tecnologias para isso tem recebido grande atenção por parte da academia, indústria de máquinas e implementos, bem como dos produtores que desejam melhorar a eficiência produtiva (NASTARI, 2015).

A incorporação de todas estas novas tecnologias na cadeia produtiva, resultou também em um viés negativo, aumentando o custo histórico que era de 15 dólares por tonelada para 25 dólares por tonelada, expondo assim a necessidade de se buscar meios que otimizem esse sistema (RODRIGUES, 2015).

2.2 Robótica

Com o aumento demográfico mundial ocorre também a necessidade do aumento da produtividade agrícola e, uma das técnicas para isso, é a utilização de robôs que tem velocidade de operação superior a de humanos. No sentido de alcançar novas habilidades vários estudos são realizados no âmbito do desenvolvimento de métodos, processos, sistemas, sensores e equipamentos com vistas à integração dos sistemas de produção e a sustentabilidade dos mesmos.

Com isso é observado por meio de estudos bibliográficos, que eles estão alcançando as habilidades de trabalhar continuamente e de forma consistente com o mínimo de manutenção (HACKENHAAR; HACKENHAAR; ABREU, 2014).

A robótica tem estado nas mentes dos seres humanos desde o momento em que foi possível construir coisas. Embora, em princípio, humanoides sejam robôs, projetados e regidos pelos mesmos princípios, a aceitação dos robôs manipuladores industriais pelo mercado permitiu o seu desenvolvimento e o aperfeiçoamento nesta área, que culminou na capacidade destes em realizar diversas tarefas e operações com precisão e praticidade, além de não requererem os elementos comuns de segurança e de conforto que os seres humanos necessitam.

A palavra robô tem sua origem na obra de Karel Capek intitulada "*Rossum's Universal Robots*" que apresenta um cenário no qual um bioprocesso poderia criar máquinas com aparência humana, desprovidas de emoções e almas, porém fortes e obedientes a seus mestres. Em sua história a palavra "*rabota*" que significa trabalhador foi inventada e é utilizada até hoje (NIKU, 2013).

Falando de termos utilizados dentro deste contexto, "robótica" indica a disciplina associada ao uso e programação de robôs e, a engenharia robótica refere-se à construção destas máquinas e seus equipamentos. Assim, a *International Organization for Standardization* (ISO) 10218 (1992) define o robô como:

> "... uma máquina manipuladora com vários graus de liberdade controlada

> automaticamente, reprogramável, multifuncional, que pode ter base fixa ou móvel para utilização e aplicações de automação industrial".

Do ponto de vista histórico, após a Segunda Guerra Mundial, foram projetadas máquinas com o viés de aumentar a produtividade e melhorar a qualidade dos produtos, além de permitir a manipulação de materiais de grande periculosidade como os nucleares. A princípio essas máquinas eram controladas por cartões perfurados lidos por uma espécie de olho elétrico; cenário esse que evoluiu com a incorporação de dispositivos de memória e o advento dos computadores pessoais (NIKU, 2013). Segundo este mesmo autor, o robô é basicamente composto de: manipulador ou o explorador, atuador final, atuadores, sensores, controlador, processador e software; com cada uma destas partes sendo responsável por uma funcionalidade.

Para localizar objetos em um espaço tridimensional, bem como a sua própria posição, os robôs fazem uso de um sistema de referência que permita a descrição desses atributos e a sua transformação de um sistema para outro. Assim, a ciência que trata do movimento sem considerar as forças que o causam é chamada de cinemática e, ela se preocupa com o estudo da posição, velocidade, aceleração e todas as derivadas de ordem superior das variáveis de posição. Para que o robô interaja com outros objetos ele faz uso dos manipuladores que consistem de elos quase rígidos conectados por juntas, de forma a permitir o movimento relativo dos elos vizinhos. Essa característica permite descrever que os robôs podem conter diferentes graus de liberdade. Esses graus são os números de variáveis de posição independentes, que teriam de ser especificadas para que todas as peças do mecanismo possam ser localizadas (CRAIG, 2012).

A cinemática pode ser dividida em duas partes: uma que trata do problema de geometria estática de computar a posição e, a orientação do efetuador do manipulador chamada de cinemática direta. A outra se preocupa com os possíveis conjuntos de ângulos de junta que poderiam ser usados para se obter a posição e orientação desejada, a partir de uma dada posição e, orientação do efetuador do manipulador chamada de cinemática inversa. Dentro da robótica, a dinâmica é o campo que se dedica ao estudo de todas as forças necessárias para causar o movimento a fim de acelerar um manipulador desde o repouso, manter o efetuador em velocidade constante e, por fim, desacelerar até o repouso (CRAIG, 2012).

Esses robôs podem ser classificados como robôs industriais e robôs não industriais; robôs fixos (braços robóticos) e robôs móveis (veículo robótico). Destes, os robôs fixos possuem um espaço limitado para mover seus manipuladores. Este espaço é denominado de espaço ou volume de trabalho e, corresponde basicamente aos locais possíveis para o posicionamento e uso de sua ferramenta. Por outro lado, os robôs moveis são carregados de um grande potencial para uso na agricultura de precisão, trazendo a vantagem de poder fazer uso das diversas teorias em controle robótico já fundamentadas e consolidadas

(HACKENHAAR; HACKENHAAR; ABREU, 2014).

O sucessivo desenvolvimento de tecnologia afins, trouxeram um aumento produtivo atrelado a uma maior eficiência econômica. Isto culminou na tendência para a substituição das máquinas grandes e pesadas por outras baseadas na informação, por estas permitirem a realização de operações autônomas viáveis e confiáveis (HACKENHAAR; HACKENHAAR; ABREU, 2014). Esses mesmos autores descreveram, em seu artigo "Robótica na Agricultura", sobre os avanços tecnológicos na área de robótica voltada para a agricultura onde, um destes projetos foram o desenvolvimento da "Fabrica de Planta". Este projeto em si, consiste no desenvolvimento de um sistema em que o cultivo hidropônico ocorre em um ambiente controlado sob iluminação artificial. Neste sistema os computadores e os robôs controlam o processo de plantio de mudas, adubação, sanidade, corte da raiz, embalagem e pesagem resultando em produtos sem defeitos, doenças ou danos causados por insetos.

O primeiro robô, que se tem notícia, utilizado no campo com finalidade agrícola foi desenvolvido em 1998 por Astrand e Baerveldt. Basicamente ele auxiliava o produtor na tarefa de controle de ervas daninhas. Em 2004, Bak e Jakobsen desenvolveram um pequeno robô capaz de viajar entre as linhas das culturas para registrar a localização de plantas daninhas usando uma câmera e um sistema de posicionamento Global (GPS) receptor. Também, em 2004, Hofstee, Grift e Tian desenvolveram um algoritmo de visão computacional para ser utilizado no campo e seu deslocamento se dava por meio de orientação autônoma (HACKENHAAR; HACKENHAAR; ABREU, 2014).

Quando se pensa na construção de robôs para uso no campo e com a finalidade de auxiliar o produtor agrícola, é necessário que se leve em consideração o desenvolvimento da cultura a ser beneficiada, por isso, o conhecimento de que durante o estágio de desenvolvimento da planta existe a necessidade de se medir o nitrogênio, estresse hídrico em plantas, infestações de insetos e plantas daninhas. O mecanismo passível de ser utilizado para isso são as câmeras, que devem ser acopladas ao dispositivo. Os robôs a serem utilizado para isso não precisam ser necessariamente grandes e podem ser menores que as máquinas agrícolas convencionais. No local de trabalho do dispositivo eles poderão agir de forma cooperativa e realizar tarefas como a pulverização, mitigando os riscos ao ser humano. Outra característica de um robô, pode ser a integração com um Sistema de Informação Gerencial (SIG) permitindo assim lidar com vários tipos de dados como: informações de campo, tipo de cultura, tipo de solo, produtividade, qualidade, informações do agricultor, custo, química e de fertilizantes. De posse das informações coletadas ou alimentadas manualmente, o sistema pode se comunicar com o robô móvel que, por meio de sensores, informa os dados sobre a eficiência do trabalho, nível de combustível, fertilizante e substâncias químicas, contidas em cada tanque. Em 2006 foi apresentado um robô para pulverização agrícola que permite a ação precisa em tempo real com a coleta de informações de posição, incidência de doenças e pragas transmitindo-as a um

atomizador ou pulverizador que regula a necessidade de maior ou menor quantidade de defensivos. Isto garante que além da mitigação do desperdício uma melhor condição de trabalho ao agricultor que não terá contato com produtos altamente tóxicos. O sistema proveniente de estudos na china permitiu o desenvolvimento de robô com as características citadas apresentando um erro variável inferior a 10 % (HACKENHAAR; HACKENHAAR; ABREU, 2014).

Quando são realizadas pesquisas que apresentam essas tecnologias nos vem questionamentos sobre o porquê eles não estão em funcionamento em todos os campos, inclusive àqueles próximos a nós e, a resposta para isso foi dada por (HACKENHAAR; HACKENHAAR; ABREU, 2014) que apresenta ao final de seu artigo a seguinte conclusão:

> "...alguns dos fatores que culminam na resistência de sua implantação na agricultura são a fragilidade das máquinas, tecnologia mecânica dispendiosa, trabalho sob limite da capacidade da máquina, bem como a eficiência do trabalho ainda a ser melhorado e adaptado a diversas situações. Tudo isso se soma a dificuldade em diluir os custos monetários em diversas operações uma vez que eles são desenvolvidos para apenas uma aplicação. ... a tecnologia pode desempregar, mas também em outro projeto social, pode facilitar o trabalho e aumentar a produção. Isso mostra a necessidade de um controle hegemônico das tecnologias, para que se possa ter uma sociedade onde a terra, o trabalho, a técnica e os seus frutos possam ser socializados."

2.3 Visão Computacional

Dar ao computador a capacidade de enxergar o mundo, de forma similar ao sistema de visão humano é cada vez mais cotidiano por conta das diversas bibliotecas e linguagens de programação, que abstraem conceitos matemáticos tornando seu uso mais simples. As menções sobre a visão computacional datam da década de 1950. Em 1982, Ballard e Brown apresentaram uma obra intitulada "*Computer Vision*" definindo-a como uma ciência que estuda e desenvolve tecnologias que permite as máquinas enxergarem e extrair as características do meio, através de imagens capturadas por diferentes tipos de sensores e dispositivos. Essas informações extraídas permitem reconhecer, manipular e processar dados sobre os objetos que compõem a imagem capturada. Um grande problema na implementação destes sistemas foi a falta de informações, sobre como as imagens são interpretadas no cérebro humano. Por esta razão, são realizadas diversas pesquisas que visam entender seu funcionamento, como: estudos sobre a neurobiologia da visão, inteligência artificial aplicada a área da robótica, processamento de imagens e o reconhecimento de padrões (BARELLI, 2018).

(GONZALEZ; WOODS, 2010) definem imagem como uma função bidimensional de intensidade da luz $f(x, y)$, onde x e y denotam as coordenadas espaciais e o valor de f em qualquer ponto (x, y) proporcional ao brilho da imagem naquele ponto. A função $f(x, y)$ é

discretizada tanto em coordenadas espaciais quanto em brilho que pode ser considerada como uma matriz cujos índices de linhas e de colunas identificam um ponto na imagem, e o correspondente valor do elemento da matriz identifica o nível de cinza naquele ponto. Cada elemento da matriz é chamado de elemento da imagem ou *pixels*, abreviação de "*picture elements*".

O processamento de imagens digitais abrange uma ampla escala de *hardware*, *software* e fundamentos teóricos (GONZALEZ; WOODS, 2010). Para que um sistema de propósito geral seja capaz de desempenhar as operações de processamento de imagens, ele poderá ser organizado em subsistemas que abstraem as seguintes etapas: aquisição de imagem, armazenamento, processamento, comunicação e exibição da imagem. Respectivamente, cada um destes subsistemas podem executar as seguintes tarefas e das seguintes formas, assim:

- a aquisição da imagem pode ser realizada por qualquer dispositivo de captura que podem ser: de digitalização, como *Scanner*, Tomógrafos, Raio X, ou câmera capaz de converter sinais luminosos em sinais elétricos, dentre outros;
- o armazenamento poderá ocorrer em memória de um dispositivo computacional, podendo esta ser volátil ou não volátil;
- o processamento da imagem poderá ser utilizado para resolver possíveis problemas na imagem capturada que ocasionalmente podem atrapalhar a sua interpretação;
- em ocasiões onde é necessário a transferência da imagem para um outro dispositivo remoto o desenvolvedor pode optar por diferentes meios dentre os quais estão as redes de computadores e ondas eletromagnéticas;
- a exibição dessas imagens pode ocorrer em diversos dispositivos que tenham a capacidade de converter sinais elétricos em sinais luminosos, como monitores de TV.

(COPPIN, 2012) diz em seu livro "Inteligência Artificial" que o processo de reconhecimento de padrões em imagem pode ser desmembrado nos seguintes e principais estágios:

- captura de imagem;
- detecção de bordas;
- segmentação;
- segmentação tridimensional;
- reconhecimento e análise.

Aqui, a detecção de bordas pode ser obtida pela análise das descontinuidades na imagem e, permite a estimação de quantos objetos estão presentes em uma cena. Para isso pode ser utilizado dentre outras técnicas a convolução e o de detecção de bordas de *Canny* (GONZALEZ; WOODS, 2010).

Uma vez que as bordas são conhecidas é possível segmentar a imagem em áreas homogêneas. Dentre os métodos possíveis para realizar a segmentação de uma imagem podem ser considerados aqui a imposição de limite que envolve encontrar a cor de cada pixel em uma imagem e, depois considerar *pixels* adjacentes como sendo da mesma área desde que as cores deles sejam parecidas o suficiente. Outro método similar é chamado de separação e junção, que corresponde a tomar uma área que não seja homogênea e separá-la em duas ou mais áreas menores, cada uma das quais sendo homogênea e, a segunda, é tomar duas áreas que sejam idênticas e adjacentes e combina-las em uma área maior (GONZALEZ; WOODS, 2010).

A classificação das bordas tem a função de determinar se uma borda é convexa, côncava ou de obstrução. Com isso o sistema pode fazer mais avaliações sobre a natureza, forma e a posição relativa dos objetos na imagem. Para completar este processo pode ser feito uso do algoritmo de *Waltz* por meio da seleção de uma classe possível para uma junção, passando para uma junção adjacente, de forma a tentar aplicar um rótulo que garanta que, o rótulo de todas as junções seja válido. Dentre as informações que podemos obter sobre determinada imagem e, que pode ser de grande valor para o reconhecimento de padrões, por parte do sistema de visão computacional é a textura da imagem, que é considerado um aspecto essencial do mundo visual. Esta característica pode ser definida como o padrão que percebemos na superfície de um objeto ou em uma área permitindo, por exemplo, a distinção de se uma imagem é mato, seixos, nuvens ou telhas, por exemplo (COPPIN, 2012).

Considerando que o reconhecimento das folhas leva ao reconhecimento da planta (ULUTURK; UGUR, 2012) aplicou técnicas de pré-processamento seguida da extração de 7 características onde, dentre elas estão: área, extensão e a excentricidade.

Uma outra técnica para realizar o reconhecimento de plantas pela análise de sua folhagem é por meio de um descritor de forma, chamado de *Multiscale Distance Matrix* (MDM), que pode ser utilizado para capturar a geometria da forma invariante à translação, rotação, dimensionamento e simetria bilateral. Este descritor quando combinado com uma redução em sua dimensão pode melhorar sua discriminação e tem demonstrado bom desempenho nesta tarefa tornando-o recomendável para implementação em sistemas de tempo real (HU et al., 2018).

Além das propostas acima podemos citar:

- a aproximação automática baseada na informação visual de suas folhas considerando

a margem da folha e seus pontos salientes (Mouine; Yahiaoui; Verroust-Blondet, 2013);

- reconhecimento da folha por meio de um correlacionamento entre a imagem a ser classificada e os modelos armazenados em uma base de conhecimento (HSIAO et al., 2014).
- utilização de diferentes processadores de imagens e técnicas de aprendizado de máquinas incluindo: raios de centroides, momentos invariantes, utilização de Canny para detecção de bordas e a realização de operações morfológicas em imagens (CODIZAR; SOLANO, 2016).

(LUKIC; TUBA; TUBA, 2017) sugerem que para obter sucesso nesta área, as características como cor, forma, textura, momentos de Hu e dados do histograma devem ser consideradas em conjunto com a utilização de diferentes classificadores.

3 Material e Métodos

3.1 Descrição da Área

A captura das imagens, bem como os testes do ERACI (Estrutura Robótica para Aquisição e Classificação de Imagens), foram realizados no município de Barretos, localizada no estado de São Paulo (Figura 3). O município encontra-se a uma altitude de 544,0 m, Latitude: 20° 33' 33" Sul, Longitude: 48° 34' 8" Oeste; apresenta clima tropical e chove muito menos no inverno que no verão. De acordo com a Köppen e Geiger o clima é classificado como Aw (ALVARES et al., 2013). Segundo a entidade (CLIMATE-DATA, 2020), a pluviosidade média anual é 1.309,0 mm; temperatura média de 22,8 °C; a temperatura média mais alta ocorre no mês de fevereiro chegando a 24,5 °C e a temperatura média mais baixa em junho é de 19,4 °C. Seu solo tem características bem diversificadas passando pelo Latossolo roxo estrófico argiloso da margem do Rio Pardo e, Latossolo vermelho escuro distrófico arenoso até Argissolos na parte oeste do município (SILVA, 2014; INPE; AMBDATA, 2020).

Para adequar o sistema a diferentes tipos de algoritmos da área de visão computacional, foram capturadas imagens nas mais diversas áreas por meio do Dispositivo Robótico Remoto (DRR) (Figura 3). A captura das imagens rurais, bem como o teste do dispositivo, neste mesmo ambiente, foram realizados em dois locais distintos. Um deles foi realizado no campus agrícola do Instituto Federal de São Paulo – Campus Barretos, localizado na zona rural do município a latitude 20°30'7.65"S e longitude 48°33'56.82"O e, o outro foi na Fazenda Santa Adelaide, também localizada na cidade de Barretos sob a latitude 20°34'41.74"S e longitude 48°35'40.89"O.

Foram também coletadas imagens na área urbana do município de Barretos, com o propósito de permitir um treinamento do sistema de forma mais robusta, contemplando áreas fora do ambiente rural. Ao todo o DRR percorreu, capturou imagens e foi testado dentro de uma área de $171.728,00\ m^2$ ($17,1728\ ha$) (Tabela 1).

Tabela 1 – Localidade e área coberta pelo sistema para a aquisição de imagens e teste do sistema.

Indice	**Local**	**Área (m^2)**
1	Fazenda Santa Adelaide	45.306,00
2	Fazenda Santa Adelaide	28.741,00
3	Fazenda Santa Adelaide	19.150,00
4	Município de Barretos	43.586,00
5	IFSP - Campus Barretos	34.945,00
Total		171.728,00

Neste contexto, a:

- área 1 contribuiu com imagens de cana-de-açúcar envolta por uma densa presença de ervas-daninha;
- área 2 com imagens com densa presença de solo visível;
- área 3 com imagens com alta densidade de cana-de-açúcar;
- área 4 com imagens urbanas com a presença de pessoas, árvores, solos, edificações e veículos;
- área 5 com todas as características observadas nas áreas anteriores, porém com densidade menor.

No campus agrícola do IFSP campus Barretos, embora todo tipo de características (Tabela 7) tenham sido observadas com a captura de imagens pelo ERACI, foi dada maior ênfase na captura de imagens da lavoura de cana-de-açúcar na fazenda experimental. Neste

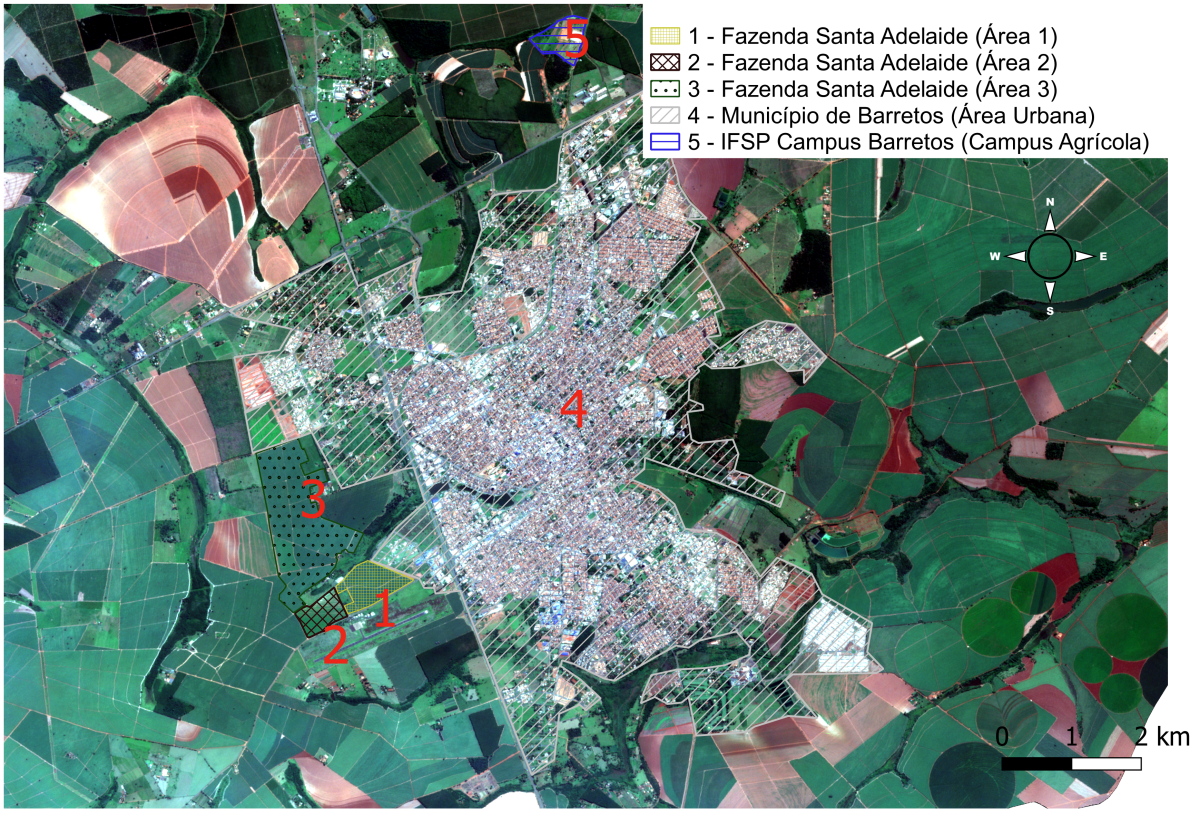


Figura 3 – Mapa mostrando toda a área coberta para a captura e realização dos testes de visão computacional para o ERACI.

Fonte: Landsat-8 (INPE, 2020)

ambiente, a cana-de-açúcar foi plantada de forma mecanizada no mês de março de 2019. Já, em relação a Fazenda Santa Adelaide, que embora atue no mercado agropecuário no ramo de cana-de-açúcar, soja, pastagens e gado, apenas a área com soja e cana-de-açúcar foram utilizadas para este projeto. O canavial, neste caso, foi formado nos anos de 2008, 2016, 2017 e 2018. Em ambos os locais o plantio foi realizado utilizando rebolos, de forma mecanizada em área sequeira.

3.2 Sistemas Distribuídos

Com o desenvolvimento das redes de computadores e a sua consequente evolução, foi possível montar sistemas de computação compostos por grandes quantidades de computadores conectados por uma rede de alta velocidade. A estes sistemas deu-se o nome de sistemas distribuídos. Assim, este tipo de sistema é definido como:

> ... um conjunto de computadores independentes que se apresenta a seus usuários como um sistema único e coerente (TANENBAUM; STEEN, 2007).

Dentre os aspectos a serem considerados por esta definição, é que ele consiste em componentes autônomos. Os usuários, podem ser tanto pessoas como programas que se comportam como se estivessem lhe dando com um único sistema. Isso significa que as partes computacionais precisam colaborar entre si (TANENBAUM; STEEN, 2007).

A principal meta de um sistema distribuído é facilitar aos usuários, e às aplicações, o acesso a recursos remotos e seu compartilhamento de maneira controlada e eficiênte. A conecção destes usuários e recursos também facilita a colaboração e a troca de informações, o que é claramente ilustrado pelo sucesso da Internet com seus protocolos simples para trocar arquivos, vídeos, audios, imagens dentre outros (TANENBAUM; STEEN, 2007).

Outra meta importante deste tipo de sistema está relacionado à ocultação do fato de que seus processos e recursos estão fisicamente distribuídos por vários computadores. A Tabela 2 apresenta as diferentes formas de transparência em um sistema distribuído (TANENBAUM; STEEN, 2007).

Outra meta importante de sistema distribuídos é a abertura. Esta característica corresponde ao oferecimento de serviços de acordo com regras padronizadas que descrevem a sintaxe e a semântica desses serviços. Esses serviços, por sua vez, são especificados por meio de interfaces, que costumam ser descritas em uma linguagem de definição de interface *Interface Definition Language* (IDL). Neste tipo de sistema também é importante que haja a interoperabilidade garantindo assim que sistemas ou componentes possam coexistir e trabalhar em conjunto. Assim, não menos importante, está a capacidade de um software

Tabela 2 – Diferentes formas de transparência em um sistema distribuído.

Transparência	**Descrição**
Acesso	Oculta diferenças na representação de dados e no modo de acesso a um recurso
Localização	Oculta o lugar em que um recurso está localizado
Migração	Oculta que um recurso pode ser movido para outra localização
Relocação	Oculta que um recurso pode ser movido para uma outra localização enquanto em uso
Replicação	Oculta que um recurso é replicado
Concorrência	Oculta que um recurso pode ser compartilhado por diversos usuários concorrentes
Falha	Oculta a falha e a recuperação de um recurso

Fonte: (FAROOQUI; LOGRIPPO; MEER, 1995)

criados no sistema distribuído A poder, sem modificação, ser executado em um sistema distribuído B (TANENBAUM; STEEN, 2007).

Para que esse sistema seja flexível é crucial que o sistema seja organizado como um conjunto de componentes relativamente pequenos e de fácil substituição ou adaptação. Isso implica que devemos fornecer definições não somente para as interfaces de nível mais alto, mas também para as interfaces com partes internas do sistema e suas interações. Com relação a escalabilidade do sistema ele deve ser capaz de receber mais usuários ou recursos, permitir que usuários e recursos possam estar geograficamente longes uns dos outros e possibilitar uma fácil gestão de dados e recursos. Uma das formas que podem ser utilizadas para mitigar possíveis problemas de escalabilidade é a capacidade deste em realizar comunicação assíncrona (TANENBAUM; STEEN, 2007).

Os sistemas distribuídos podem ser classificados como: Sistemas de Computação Distribuídos, Sistemas de Informação Distribuídos e Sistemas Embutidos Distribuídos. Os sistemas de computação distribuídos são utilizados para tarefas de computação de alto desempenho e podem ser do tipo computação em *cluster*, onde o hardware subjacente consiste em um conjunto de estações de trabalho semelhantes, conectados por meio de uma rede local de alta velocidade com cada nó executando o mesmo Sistema Operacional (SO) e, computação em grade, em que as partes computacionais costumam ser montadas com federação de computadores, na qual cada sistema pode cair sob um domínio administrativo diferente com *hardware*, *software* e tecnologias de redes empregada diferentes. Os Sistemas de Informação Distribuídos são utilizados quando é necessário que as aplicações possam interoperar. Este tipo de sistema pode consistir de servidor de aplicação, como banco de dados ou programas que possam ser executados remotamente. Por fim, existem também os sistemas distribuídos embutidos em que os equipamentos costumam ser caracterizados por seu pequeno tamanho, pela alimentação por bateria, por sua mobilidade ou por terem

somente uma conexão sem fio (TANENBAUM; STEEN, 2007).

Segundo (GRIMM et al., 2004) aplicações pervasivas devem ter três requisitos fundamentais:

1. Adotar mudanças contextuais;
2. Incentivar composição *ad hoc*;
3. Reconhecer compartilhamento como padrão.

O ERACI foi projetado para funcionar sob estas características. Por este motivo ele foi desenvolvido de forma modular, com suas partes separadas geograficamente e independentes, garantindo assim: segurança, escalabilidade e transparência. Estas características permitem a troca de informações entre os dispositivos que o compõe, além de tornar possível a existência de vários especialistas conectados, por meio de Dispositivos Móveis (DM) com o Dispositivo Gerador e Gestor de Conhecimento (DGGC), para realizar a pré-classificação manual das imagens armazenadas no Banco de Dados (BD) (Figura 4).

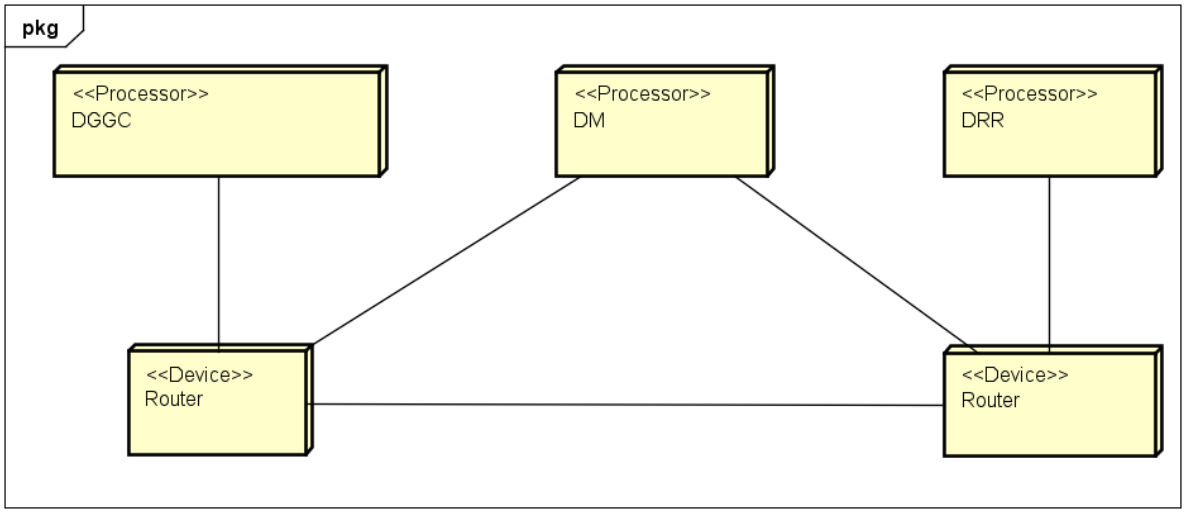


Figura 4 – Diagrama de instalação do sistema apresentando em cada um dos nós os dispositivos envolvidos em sua interação.

Fonte: Autoria Própria utilizando UML (SILVA; VIDEIRA; ATLÂNTICO, 2001)

3.3 Sistemas de Tempo Real

Segundo (KOPETZ, 2011) um sistema de computador em tempo real é um sistema de computador em que a correção do comportamento do sistema não depende apenas dos resultados lógicos dos cálculos, mas também do tempo físico em que esses resultados são

produzidos. Por comportamento do sistema, entendemos a sequência de saídas no tempo de um sistema.

Ainda, segundo o mesmo autor, a interface entre o operador humano e o sistema de tempo real é chamado de interface homem-máquina e a interface entre o objeto controlado e o sistema de tempo real é chamado de interface de instrumentação. A interface homem-máquina consiste de dispositivos de entradas e dispositivos de saída (KOPETZ, 2011) (Figura 5).

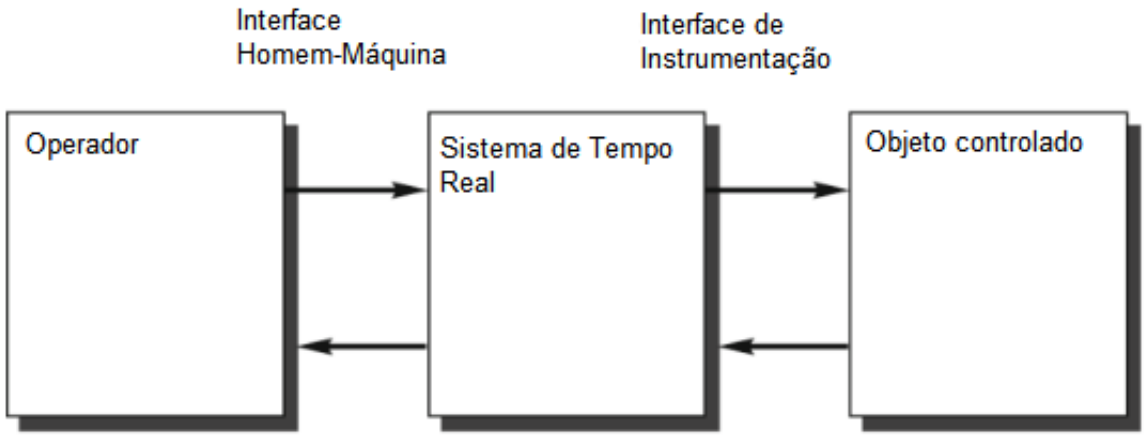


Figura 5 – Sistema de Tempo Real.

Fonte: Diagrama baseado em (KOPETZ, 2011)

Um sistema computacional de tempo real reage ao estimulo de um ambiente dentro do intervalo de tempo ditado pelo ambiente. O instante que um resultado deve ser produzido é chamado de *deadline*. Se esse *deadline* aconteceu atrasado, ele é classificado como *deadline soft*, caso contrario ele é *deadline firm*. Se consequências severas podem ocorrer quando um *deadline firm* acontece de forma errada, a *deadline* é chamada *hard* (KOPETZ, 2011).

Segundo (COSTA, 2008a), as aplicações que fazem transmissão de dados em sistemas distribuídos na Internet, devem atuar em um tempo de resposta que seja o menor possível e o mais constante possível, de forma a manter tanto a noção de tempo real quanto a qualidade da reprodução das mídias recebidas através da infraestrutura disponível. Desta forma, atrasos grandes prejudicam a natureza do tempo real da comunicação e, atrasos variáveis prejudicam a continuidade de reprodução da mídia.

3.4 Hardware

3.4.1 Introdução

Para que o sistema receba manutenção e também seja de fácil mobilidade, ele foi implementado utilizando dispositivos modulares que possam ser interconectados pela rede de computadores.

No diagrama de instalação (Figura 4), o nó Dispositivos Robóticos Remotos (DRR) corresponde a estrutura que é levada ao campo escolhido, para realizar a captura de imagens e a realização de testes. Ele está conectado a um nó chamado Router, que é um roteador TP-Link® modelo TL-MR 3420, capaz de permitir conexão com a Internet por meio de outro roteador pela porta RJ-45 *Wide Area Network* (WAN) ou por meio de um modem 3G/4G conectado à sua porta *Universal Serial Bus* (USB). Quando o DRR está conectado ao roteador, é possível fazer uso de um dispositivo móvel conectado a este mesmo roteador, para realizar seu controle e monitoramento em tempo real por meio do aplicativo ERACI instalado nele.

Na outra ponta do diagrama (Figura 4), o nó Dispositivo Gerador e Gestor do Conhecimento (DGGC) tem softwares capazes de armazenar e gerir os dados de imagens, classes, usuário e classificações realizadas pelo usuário exercendo o papel de professor do subsistema de visão computacional. Além disso, O DGGC tem a capacidade de manter serviços *HyperText Transfer Protocol* (HTTP) e, gerar arquivos baseados em algoritmos de *Machine Learning* (ML), capazes de realizar o reconhecimento de padrões em imagens. O dispositivo representado por este nó também está conectado a um roteador TP-Link, porém o modelo é o WR-840N, que permite a conexão com a Internet apenas por meio de sua porta RJ-45 WAN.

Quando os dois dispositivos extremos, DRR e DGGC, estão em funcionamento e conectados aos seus respectivos roteadores e, estes por sua vez, estão conectados entre si por meio de rede cabeada ou pela Internet, eles podem trocar informações. Neste caso o DRR pode enviar as imagens capturadas ao DGGC que as processa, extrai as características e armazena as imagens e as características no BD. O DGGC envia os arquivos classificadores ao DRR para que este seja capaz de reconhecer os padrões nas imagens que estão sendo "vistas".

O nó DM corresponde ao dispositivo móvel que tem o aplicativo ERACI instalado. Por meio deste aplicativo é possível gerir os dados armazenados no BD e realizar classificações das imagens nele armazenadas.

O sistema também foi organizado para permitir que as imagens possam ser capturadas nos locais e momentos convenientes. Assim, foi criado um sistema supervisório a ser instalado no DGGC que permite controlar e obter informações do DRR. Além disso

é possível acompanhar o que o DRR está “vendo”, bem como, estas informações visuais estão sendo classificadas pelo algoritmo de visão computacional. As trocas de informações entre estes dispositivos (DRR e DGGC) ocorrem por meio de *sockets Transmission Control Protocol/Internet Protocol* (TCP/IP) para dados de texto com informações sobre controle e monitoramento e, *User Datagram Protocol/Internet Protocol* UDP/IP para dados de imagens enviadas pelo DRR.

A forma como o sistema é organizado permite que possam existir vários DRR e DM e, a interação entre eles, ocorra por meio da rede de computadores, com a utilização dos protocolos HTTP, TCP/IP e UDP/IP. Dentro de uma arquitetura cliente/servidor, cada DRR e DM se comporta como um cliente conectado ao DGGC, que é capaz de receber inúmeras conexões de ambos os clientes. Assim, é possível que, quando em um ambiente de produção, possam existir vários DRR carregando imagens para um único servidor capaz de gerar modelos de ML cada vez melhores. Por conseguinte, é também possível que haja vários usuários classificando imagens a serem utilizadas pelos algoritmos de ML para a geração deste “conhecimento".

Cada um dos nós do diagrama (Figura 4) possui uma série de componentes de software, com as mais diversas funcionalidades incorporadas. Estas funcionalidades são divididas em subsistemas do DRR, subsistemas do DGGC e subsistemas do aplicativo para DM. Todos estes subsistemas foram implementados utilizando basicamente 3 plataformas, sendo elas: *Java*, *Python* e *Android*.

3.4.2 Dispositivo Robótico Remoto (DRR)

Para facilitar a realização de testes do sistema em ambiente controlado e também de fácil acesso, optou-se, a princípio, por construir o DRR de forma que fosse possível capturar e reconhecer imagens em área gramada que tem, em menor escala de tamanho, características semelhantes à da cana-de-açúcar. Para satisfazer esta necessidade inicial, o DRR foi implementado em um chassi de acrílico com espessura de 3 mm e dimensão de $21,0x14,7$ cm, equivalente ao apresentado na (Figura 6). Neste chassi foram afixados duas rodas motorizadas e uma roda articulada não motorizada. Cada uma das rodas motorizadas é impulsionada por dois motores DC (*direct corrent* ou corrente contínua) com tensão de funcionamento que varia de $3,0$ V a $6,0$ V, 125 RPM de velocidade de rotação, quando alimentado por uma tensão de $3,0$ V e, $5,5$ Kg/cm quando alimentado por uma tensão de $6,0$ V, acoplados a um redutor de eixo único.

Neste chassi foi acoplado um módulo ponte-H (Figura 7) que faz uso de um circuito integrado ST L298N, capaz de controlar até 2 motores. A máxima corrente de alimentação suportada por este módulo é de $4,0$ A e a máxima tensão de alimentação de $35,0$ V. A potência máxima disponibilizada por este módulo é de $25,0$ W.

Controlar o ângulo de ação da câmera permite a seleção das imagens a serem capturadas. Para isso foi utilizado um kit de suporte pan/tilt com dimensões de $30,0x11,5x30$ mm (Figura 8), micro servo motor tower pro 9G SG90 com velocidade de operação variando de $0,10s/60\circ$ a $0,12s/60\circ$ quando alimentado entre as tensões de $4,8$ a $6,0$ V. Sua capacidade de torque varia de $1,2$ kg/cm a $1,6$ kg/cm quando alimentado com uma tensão variando de $4,8$ V a $6,0$ V.

Este subsistema permite que a câmera possa ser posicionada para captura de imagem em um intervalo 180∘ na vertical e 180∘ na horizontal.

O sistema utiliza um módulo de câmera Pi NoIR V2 (No Infrared – Sem filtro

Figura 6 – Chassi utilizado para afixar os módulos eletrônicos, pan/tilt e o computador Raspberry Pi.

Fonte: Autoria Própria

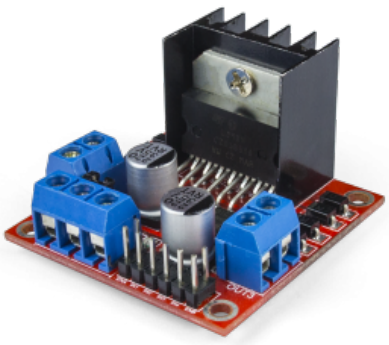


Figura 7 – Módulo ponte H utilizado para aplicar o controle de direção, sentido e velocidade do sistema.

Fonte: Autoria Própria

infravermelho) que possui um sensor IMX219 de 8 *megapixels* da Sony ® (Figura 9). Ele não tem o filtro infravermelho, o que lhe garante a capacidade de capturar imagens na ausência de luz, sendo necessário para isto, apenas a utilização de emissor de luz na frequência infravermelha.

Sua conexão ocorre por meio de um cabo de fita de 15 *cm* na porta CSI (*Camera Serial Interface* – Interface serial para câmera) do Raspberry Pi. O acesso a ela pode ser feito por meio das API – *Application Programming Interface* (Interface de Programação de Aplicativos) MMAL e V4L e, por inúmeras bibliotecas criadas por terceiros, incluindo a biblioteca Pycamera Python (RASPBERRY PI FOUNDATION, 2020).

Figura 8 – Estrutura pan/tilt que atua para posicionar a câmera para a captura das imagens.

Fonte: Autoria Própria

Figura 9 – PI Câmera NoIR responsável por capturar as imagens no ambiente de atuação do dispositivo.

Fonte: Autoria Própria

A este sistema foi acoplado um circuito composto com 20 LEDs (840 nm), capaz de fornecer iluminação infravermelho, acionadas por um sensor fotovoltaico (Figura 10).

Figura 10 – Circuito refletor de LED infravermelho acionado por sensor fotovoltaico.

Fonte: Autoria Própria

Tanto o módulo ponte-H (Figura 7) quanto o refletor infravermelho (Figura 10) são alimentados por uma bateria selada recarregável capaz de fornecer 12 V DC e corrente nominal de $1,3$ Ah.

O rastreamento dos locais onde o DRR captura e/ou reconhece as imagens é importante para o desenvolvimento de projetos ligados a AP conforme pode ser visto na seção **??**. Para isto o sistema possui um módulo GPS modelo NEO-6M GPS (Figura 11). Ele permite rastrear os locais onde as imagens foram capturadas ou classificadas pelo dispositivo.

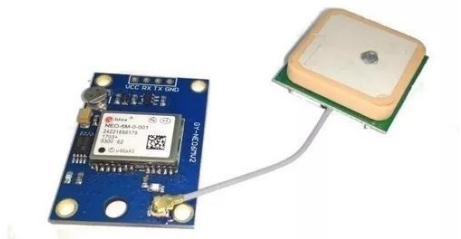


Figura 11 – Modulo GPS modelo NEO-6M GPS utilizado para obter a posição do ERACI no solo.

Fonte: Autoria Própria

Para acionar ou desligar o subsistema de dispersão de insumos agrícolas, foi acoplado ao chassi um modulo digital de controle de relês composto de dois canais e dois relês (Figura 12) que, utiliza como controle o sinal TTL (*time to live* – tempo do ciclo) e as bobinas do relê trabalham com uma tensão de 5 V DC e corrente de 75 mA. Ele permite uma carga nominal de 12 A e 125 V AC ou 7 A e 250 VAC. A carga nominal do módulo é de 10 A e seu tempo de acionamento é de 10 ms.

Figura 12 – Módulo Relê utilizado para acionar/desligar o dispersor de insumos.

Fonte: Autoria Própria

Para controlar toda a estrutura de hardware do sistema e realizar os processamentos para a classificação das imagens, o DRR e o DGGC são executados sobre um computador Raspberry Pi 3 modelo B+ (Figura 52) com a seguinte configuração (RASPBERRY PI FOUNDATION, 2020):

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz;
- 1GB LPDDR2 SDRAM;
- 2.4GHz e 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE;
- Gigabit Ethernet over USB 2.0 (tempo máximo de processamento 300 Mbps);
- Extensão de 40-Pinos GPIO header;
- Full-size HDMI;
- 4 USB 2.0 ports;
- Porta para conexão a câmera Raspberry Pi – CSI;
- DSI display port for connecting a Raspberry Pi touchscreen display;

- 4-pole stereo output and composite video port;
- Micro SD port para carregar e abrir o sistema operacional e armazenar dados;
- 5V/2.5A DC power input;
- Power-over-Ethernet (PoE) support (requires separate PoE HAT).

3.4.3 Roteador Wireless TP-Link TL-WR840N

Para que seja possível a alteração simples de localização, o *hardware* responsável pela Geração e Gestão de Conhecimento é organizado para funcionar sob uma rede fixa gerida por um roteador *Wireless* TP-Link modelo N° TL-WR840N. Este roteador é configurado para fornecer um endereço IP fixo para o DGGC e, o que facilita a localização deste na rede, independentemente de onde ambos estejam. Ele é de simples configuração e, para que ele possa utilizar uma Internet disponível em qualquer localidade, basta conectar sua entrada WAN a uma das saídas de rede de um outro dispositivo. A Figura 13 apresenta o respectivo roteador.

Figura 13 – Roteador Wireless utilizado para modularização do sistema de Geração e Gestão do Conhecimento.

Fonte: Autoria Própria

3.4.4 Roteador Wireless TP-Link TL-MR3420

Para garantir a modularidade de conexão, bem como a mobilidade do DRR, é utilizado um Roteador *Wireless* TP-Link TL-MR3420 que permite disponibilizar ao dispositivo robótico uma conexão entre ele e o dispositivo móvel e, também, com a Internet, garantindo a ele a possibilidade de troca de informações entre o DGCC. Isto é possível graças à disponibilidade de uma porta USB para conexão a um modem 3G/4G. Este roteador foi configurado para sempre fornecer endereço IP dinâmico aos dispositivos conectados a ele. Estes IPs dinâmicos foram definidos para estar dentro de um intervalo de 192.168.2.100 a 192.168.2.254. Ele é de simples configuração e, para que ele possa utilizar uma Internet disponível em qualquer localidade, basta conectar sua entrada WAN a uma das saídas de rede de um outro dispositivo ou, simplesmente conectar um modem 3G/4G a sua porta USB. A Figura 14 apresenta o respectivo roteador.

Figura 14 – Roteador Wireless utilizado para modularização no Dispositivo Robótico Remoto.

Fonte: Autoria Própria

3.4.5 Custo efetivo dos componentes de Hardware

A implementação do hardware utilizando os itens essenciais apresentados para esta fase do projeto foi de aproximadamente R$ 2.139,46. Levando em conta a cotação do dólar americano no valor de R$ 5,32 o valor total em dólar é de $ 402.15. Todos os valores dos itens essenciais são apresentados na (Tabela 3).

3.5 Software

3.5.1 Introdução

Todo o sistema foi organizado para permitir que as imagens sejam capturadas nos locais e momentos convenientes. Com este propósito, foi criado inicialmente um sistema supervisório a ser instalado no DGGC. Assim, ele permite que o DRR seja controlado remotamente, além de permitir que o usuário obtenha dele informações sobre temperatura do sistema, tempo restante da bateria e uso do processador. Além disso é possível acompanhar o que o DRR está "vendo" por meio de sua câmera, bem como, como estas informações visuais estão sendo classificadas pelo algoritmo de visão computacional.

Tabela 3 – Custo dos itens de hardware essenciais para a confecção desta fase do projeto.

Item	**Quantidade**	**Valor Unitário**	**Valor Total**	**Valor Total em Dolar ($)**
Raspberry Pi 3 B+	2	R$ 399,00	R$ 798,00	$ 150.00
Roteadores Wifi 2 Antenas	2	R$ 159,00	R$ 318,00	$ 59.77
Cartão de Memória Micro Sdxc 100mb/s 128gb	2	R$ 266,90	R$ 533,80	$ 100.34
Câmera Noir	1	R$ 185,00	R$ 185,00	$ 34.77
Placa de Expansão da Bateria de Lítio Ups com Bateria de Lít	1	R$ 149,52	R$ 149,52	$ 28.11
Kit Suporte Pan Tilt Completo com Servos Montados	1	R$ 44,90	R$ 44,90	$ 8.44
Modulo GPS Neo-6m Gy Neo6mv2	1	R$ 52,50	R$ 52,50	$ 9.87
Ponte H	1	R$ 14,75	R$ 14,75	$ 2.77
Carregador Móvel Portátil Power Bank	1	R$ 42,99	R$ 42,99	$ 8.08
			R$ 2.139,46	$ 402.15

A comunicação entre estes aplicativos ocorrem por meio de *sockets Transmission Control Protocol/Internet Protocol* (TCP/IP) para dados de texto com informações sobre controle e monitoramento e *User Datagram Protocol/Internet Protocol* UDP/IP para dados de imagens enviadas pelo DRR.

O esquema de conexão entre o DRR, DGGC e o DM foi apresentado na (Figura 4). O ERACI é desenvolvido para que seja possível existir várias instancias de DRR e DM e, a interação entre eles, possa ocorrer por meio da rede de computadores utilizando os protocolos HTTP, TCP/IP e UDP/IP. Dentro de uma arquitetura cliente/servidor, cada DRR e DM se comporta como um cliente conectado ao DGGC, que é capaz de receber inúmeras conexões de ambos os clientes. Assim é possível que, quando em um ambiente de produção, possa existir vários robôs carregando imagens para um único servidor capaz de gerar modelos de *machine learn* cada vez melhores. Por conseguinte, é também possível que haja vários usuários classificando imagens a serem utilizadas pelos algoritmos de IA para a geração deste “conhecimento"de forma automática.

Cada um dos nós dentro da notação UML (*Unified Model Language* – Linguagem de Modelagem Unificada) possui uma série de componentes de software com as mais diversas funcionalidades incorporadas. Estas funcionalidades são divididas em subsistemas do DRR, subsistemas do DGGC e subsistemas do aplicativo para DM, conforme apresentado nas figuras 15, 16 e 17, respectivamente.

Todo o sistema ERACI foi implementado utilizando basicamente 3 plataformas de desenvolvimento, sendo elas: *Java*, *Python* e *Android*. A forma como cada um dos subsistemas funciona e interage é apresentada com seus respectivos diagramas de componentes mais a frente neste texto.

3.5.2 Java

Java é uma plataforma integral, com uma biblioteca ampla que contempla uma grande quantidade de códigos reutilizáveis e, um ambiente de execução que fornece serviços como segurança, portabilidade para diferentes sistemas operacionais e coleta de lixo automática (HORSTMANN; CORNELL, 2010).

Ela foi criada no início da década de 1990 e introduzida no mercado no ano de 1995. A Sun Microsystem ®detentora da tecnologia a disponibilizou gratuitamente para a comunidade de desenvolvimento de *software* mantendo todos os direitos relativos à linguagem e as ferramentas de sua autoria. Os autores desta linguagem a classificaram como:

Simples: Sua sintaxe não exige a declaração de arquivos de cabeçalho, aritmética de ponteiro, estruturas, uniões, sobrecarga de operadores, classes básicas virtuais, dentre outras.

Orientada a objetos: Permite a implementação de software orientado a objetos, primando pela atuação do desenvolvedor na solução do problema e não das ferramentas

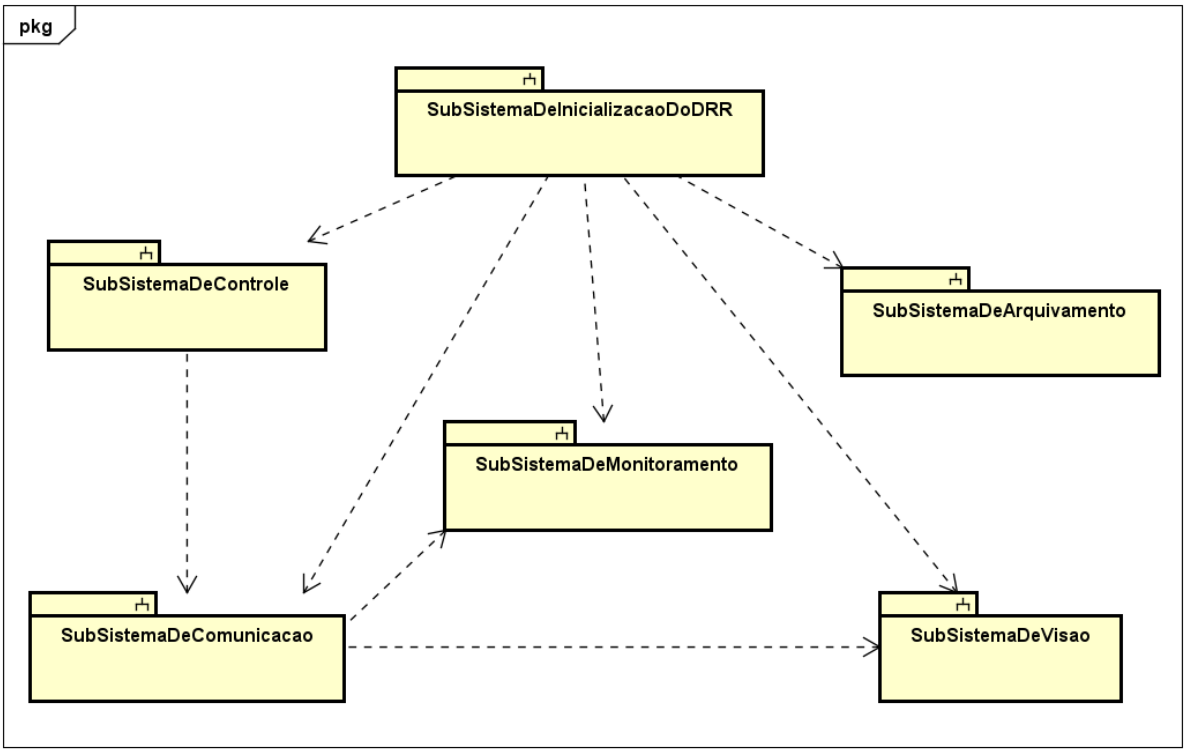


Figura 15 – Diagrama que mostra os subsistemas que compõem o Dispositivo Robótico Remoto (DRR).

Fonte: Autoria Própria

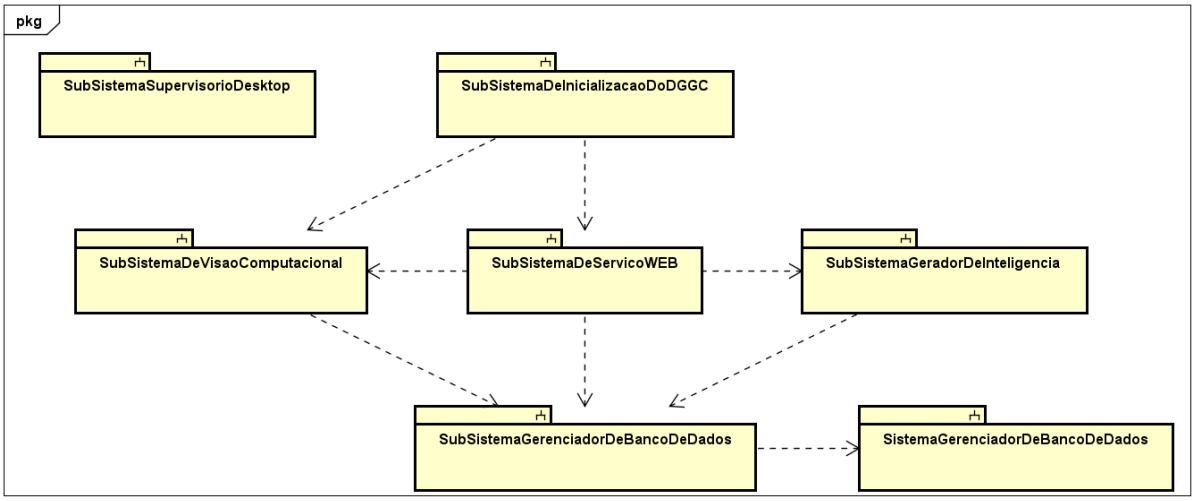


Figura 16 – Diagrama que mostra os subsistemas que compõem o Dispositivo Gerador e Gestor do Conhecimento (DGGC).

Fonte: Autoria Própria

a serem utilizadas. Essa característica permite a fácil reutilização do código e a criação de códigos mais limpos devido ao oferecimento de mecanismos de abstração, encapsulamento e hereditariedade.

Compatibilidade com redes: Java tem uma extensa biblioteca de rotinas para lidar com protocolos TCP/IP, como HTTP e FTP que permitem abrir e acessar objetos pela internet com a mesma facilidade de acesso a um sistema local.

Robusto: Java é concebido para escrever programas que precisam ser confiáveis de vários modos, dando ênfase na verificação preliminar de possíveis problemas, verificação dinâmica posterior e eliminação de situações propensas a erros.

Seguro: Java é capacitada para ser utilizada em ambientes em rede/distribuídos com forte foco na segurança, permitindo a construção de sistemas livres de vírus e adulterações.

Arquitetura neutra: A compilação do código fonte gera arquivos de objetos neutros em relação à arquitetura, pois é executado sobre um sistema em tempo real do Java. O compilador *Java* faz isso gerando instruções *bytecode* que não tem nada a ver com uma arquitetura de computador específica. Em vez disso, elas são projetadas para ser fácil de interpretar em qualquer máquina que tenha uma *Java Virtual Machine* (JVM) instalada e, ali, instantaneamente convertidas em código de máquina nativo (Figura 18).

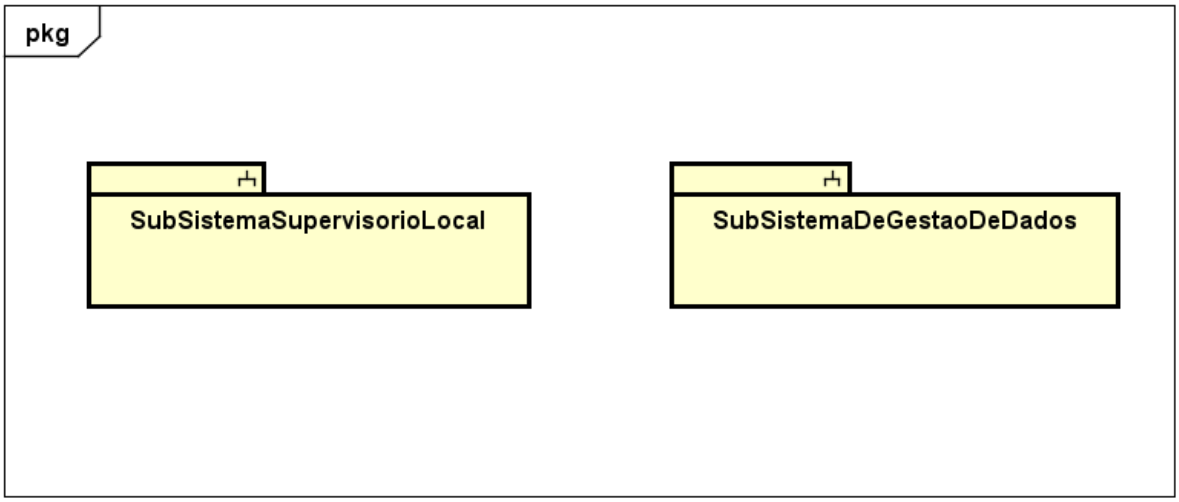


Figura 17 – Diagrama que mostra os subsistemas que compõem o Aplicativo para dispositivo móvel (DM).

Fonte: Autoria Própria

3.5.3 Python

Python é uma linguagem de programação interpretada, orientada a objetos, de alto nível e semântica dinâmica. Ela é implementada para permitir a estruturação de alto nível que, combinadas com a digitação e a ligação dinâmica a tornam atraente para o *Rapid Application Development*, além de ser utilizada como linguagem de script para conectar componentes existentes (PYTHON SOFTWARE FOUNDATION, 2020).

Por possuir uma sintaxe simples e fácil de aprender, ela enfatiza a legibilidade e consequentemente reduz o custo de manutenção do programa.

O *Python* também suporta módulos e pacotes, o que incentiva a modularidade do programa e a reutilização de código. Seu interpretador e a extensa biblioteca padrão estão disponíveis na forma de código-fonte ou binário sem custo para todas as principais plataformas e, podem ser distribuídos livremente.

Por não existir uma etapa de compilação, o ciclo de edição, teste e depuração é incrivelmente rápido. Assim, um *bug* ou uma entrada incorreta nunca causará uma falha de segmentação, pois quando o interpretador descobre um erro, ele gera uma exceção. Quando a exceção não é capturada o interpretador realiza um rastreamento de pilha (PYTHON SOFTWARE FOUNDATION, 2020).

Um depurador no nível da fonte, permite a inspeção de variáveis locais e globais, avaliação de expressões arbitrárias, definição de pontos de interrupção, revisão de código de linha a linha. Este depurador é escrito também em *Python*, o que atesta o poder introspectivo desta tecnologia (PYTHON SOFTWARE FOUNDATION, 2020).

3.5.4 Android

Conforme é apresentado pela própria documentação do *Android*, ele é um sistema operacional baseado no núcleo Linux implementado para ser utilizado em *smartphones*, *tablets* e outros dispositivos destinados a permitir mobilidade. A *Google Inc.* ® detém

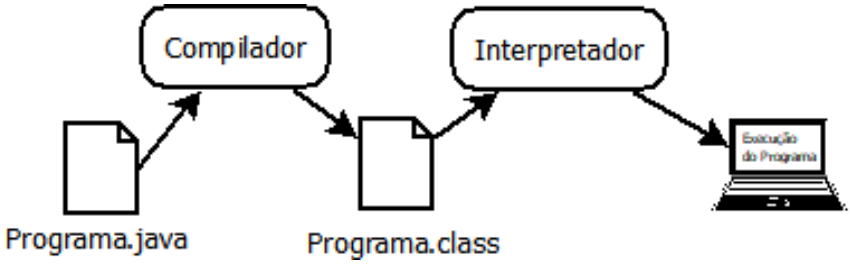


Figura 18 – Diagrama que mostra o processo de compilação e interpretação do código Java.

Fonte: Autoria Própria

todos os direitos desta tecnologia (DEVELOPERS, 2020).

Ela também é uma plataforma que permite a criação de aplicativos para uma grande variedade de dispositivos móveis. Para desenvolver aplicativos *Android* são necessários alguns programas e ferramentas disponibilizadas gratuitamente para os sistemas operacionais *OS X*, *Windows* e *Linux*. Dentre as ferramentas necessárias para isso encontram-se o *Java Development Kit*, *Android SDK* e o *Android Studio*.

Os aplicativos criados por esta tecnologia são empacotados em arquivos APK que contém o código compilado e os demais recursos, dentre eles: XMLs e imagens utilizadas pelo aplicativo.

A (Figura 19) apresenta a arquitetura de componentes desta tecnologia. Na parte inferior da Figura é possível notar que toda tecnologia é baseada num *Kernel Linux* que é responsável pelo encadeamento e gerenciamento de memória de baixo nível. Isto permite que sejam aproveitados os principais recursos de segurança e, que os fabricantes dos dispositivos desenvolvam *drivers* de *hardware* para um *kernel* conhecido.

A camada de abstração de *hardware* (HAL) fornece interfaces que expõem as capacidades de *hardware* do dispositivo para a estrutura da API do *Java*. Esta camada consiste de módulos de biblioteca, que implementam uma interface para um tipo específico de componente de *hardware*.

A partir da versão 5.0 (API nível 21), cada aplicativo *Android* executa o próprio processo com uma instância própria do *Android Runtime* (ART).

O ART é projetado para executar várias máquinas virtuais em dispositivos com pouca memória que estejam executando arquivos *Dalvik Executable* (DEX) que é um formato de *bytecode* projetado especialmente para *Android*, otimizado para oferecer consumo mínimo de memória.

O ART disponibiliza os recursos de compilação *ahead-of-time* (AOT) e *just-in-time* (JIT) e, Coleta de lixo (GC) otimizada.

A partir do *Android* 9 (nível de API 28) a conversão dos arquivos de formato *Dalvik Executable* (DEX) de um pacote de aplicativos usa um código de máquina mais compacto.

O *Android* também contém um conjunto das principais bibliotecas de tempo de execução que fornecem a maioria da funcionalidade da linguagem de programação *Java*, inclusive alguns recursos de linguagem *Java 8* que a biblioteca da API *Java* usa.

Uma grande variedade de componentes *Android*, bem como seus principais serviços são implementados em código nativo que exige bibliotecas nativas programadas em C e C++. A plataforma em si, fornece as *Java Framework APIs* para expor a funcionalidade de algumas dessas bibliotecas nativas aos aplicativos.

As APIs formam blocos de programação que permitem a criação de aplicativos de

forma simplificada, possibilitando a reutilização de componentes e serviços de sistema modulares, como: sistema de visualização rico e extensivo, gerenciador de recursos, gerenciador de notificação, gerenciador de atividade e provedores de conteúdo.

O *Android* vem com um conjunto de aplicativos que permitem o envio e recebimento de *e-mail* e SMSs, além de, calendário, navegador de internet, gerenciador de contatos, dentre outros. Esses aplicativos não são mais especiais que outros aplicativos, podendo o usuário instalar outros mais adequados a suas necessidades.

Os aplicativos do sistema tem funcionamento equivalente àqueles voltados para os usuários e, fornecem capacidades principais que os desenvolvedores podem acessar pelos próprios aplicativos.

3.5.5 Processamento de Imagens Digitais

3.5.5.1 Introdução

No servidor de serviços é implementado a Classe PDI para a extração de características da imagem e, assim ser possível a classificação em uma etapa posterior. Esta classe é composta dos mais diversos métodos necessários para a obtenção da imagem armazenada no BD e, sua conversão em RGB (*Red, Green and Blue* – Vermelho, Verde e Azul), conversão em HSV (*Hue, Saturation and Value* – Matiz, Saturação e Valor), tons de cinza e, diversos métodos para realizar o pré-processamento e segmentação das imagens. Cada um dos métodos mais relevantes é descrito a seguir.

3.5.5.2 Método para a obtenção da imagem do banco de dados

Este método, opcionalmente, recebe um id da imagem que se queira obter do banco de dados. Caso não seja passado nenhum parâmetro, são levantados os ID ("*Identity*") extremos do BD e, por meio de um gerador randômico um deles é escolhido.

Assim, este método seleciona o registro no BD correspondente ao número ID desejado, realiza o mapeamento objeto relacional por meio do *SQLAlchemy* (SQLALCHEMY, 2020) e o retorna ao requisitante.

3.5.5.3 Método para a extração da imagem a partir dos bytes armazenados no Banco de Dados

Este método é assinado com o parâmetro opcional, que se refere ao objeto de imagem armazenado. Caso o parâmetro não seja passado pelo requisitante este método chama o método para a obtenção do objeto de imagem no banco de dados (Subseção 3.5.5.2).

Por meio deste objeto é chamado o método que realiza a extração dos *bytes* do objeto e cria a imagem RGB na memória. Por fim, o método retorna um objeto referente a imagem já no formato RGB. A (Figura 20) apresenta 3 exemplos de imagens capturadas pelo DRR.

3.5.5.4 Método para a criação da imagem RGB na memória

Este método obrigatoriamente deve receber um parâmetro referente aos *bytes* da imagem armazenadas no BD. Por meio do método *fromBuffer* da biblioteca *NumPy* (NUMPY, 2020) são realizados testes para verificar qual o melhor tipo para a composição da imagem RGB que podem ser: *np.float64*, *np.float32*, *np.float16*, *np.float_*, *np.uint64*, *np.uint32*, *np.uint16*, *np.uint8*, *np.int64*, *np.int32*, *np.int16*, *np.int8*.

Este método retorna para o requisitante um objeto de imagem RGB. O resultado da extração pode ser visto na (Figura 20).

3.5.5.5 Métodos para extração de canais de imagens RGB

Estes métodos recebem uma imagem RGB como parâmetro e retornam ao requisitante o canal desejado, podendo este ser Vermelho, Verde ou Azul. Para isso são disponibilizados 3 funcionalidades distintas com a utilização do método *split* da biblioteca OpenCV (OPENCV, 2020). A (Figura 21) mostra o resultado da aplicação destes métodos às imagens apresentadas na (Figura 20).

3.5.5.6 Métodos para a conversão de formato de imagem RGB

Estes métodos recebem uma imagem RGB e realizam a conversão destas para Tons de Cinza ou HSV, conforme o método requisitado (Figura 22). Para conseguir estes resultados é utilizada a função *cvtColor* da biblioteca *OpenCV* (OPENCV, 2020). Esta função recebe 2 parâmetros que são a imagem original e o tipo de ação que se deseja realizar. Estas ações são definidas em uma constante da própria biblioteca e, para cada uma das duas tarefas podem ser: *COLOR_RGB2HSV* ou *COLOR_RGB2GRAY*.

3.5.5.7 Extração dos canais de imagem HSV

As imagens HSV são formadas pela superposição de 3 canais: Hue(Matiz), Saturation (Saturação) e Value(Valor). Cada um destes canais é responsável por uma característica da imagem e, a Classe PDI disponibiliza 3 funcionalidades, onde, cada uma delas extrai cada uma das características almejadas pelo requisitante. Ambos os métodos fazem uso da função *split* da biblioteca *OpenCV* (OPENCV, 2020). O resultado da execução de cada um destes métodos é apresentado na (Figura 23).

3.5.5.8 Equalização do Histograma de Imagem

Existem situações em que é necessário se ajustar a distribuição de intensidade de cores em uma imagem. Esta necessidade vem muitas vezes da variabilidade de iluminação em ambientes e, diferentes momentos de captura da imagem.

Para mitigar este problema este método faz uso da função *equalizeHist* da biblioteca *OpenCV ®* (OPENCV, 2020). A (Figura 24) apresenta as imagens em tons de cinza, após a aplicação da função de histograma da imagem.

3.5.5.9 Métodos para a segmentação da imagem

A Classe PDI também disponibiliza alguns métodos para segmentar imagens por cor, limiar, movimento e bordas.

Os métodos para segmentação por cor recebem uma imagem HSV e realizam sua segmentação levando em conta um intervalo predefinido para as cores Amarelo, Azul, Vermelho e Verde. Para isto é feito uso da função *inRange* da biblioteca OpenCV (OPENCV, 2020). Esta função recebe 3 parâmetros: a imagem a ser segmentada, um vetor correspondente a tonalidade mais clara da cor que se deseja segmentar e, por fim um vetor correspondente a tonalidade mais escura.

Para a segmentação de cada uma destas cores foram utilizados os seguintes valores para as tonalidades claras e escuras (Tabela 3).

Tabela 4 – Relação de tonalidades claras e escuras utilizadas para a segmentação de imagem por cor.

	Tonalidade mais clara	**Tonalidade mais escura**
Amarelo	[10, 100, 100]	[40, 255, 255]
Azul	[90, 100, 100]	[140, 255, 255]
Verde	[40, 100, 100]	[90, 255, 255]
Vermelho	[160, 100, 100]	[200, 255, 255]

A (Tabela 4) apresentou os vetores referentes as intensidades de cores para a delimitação dos limites inferiores e superiores necessários para a segmentação. A primeira posição refere-se a matiz, a segunda referente a saturação e a última corresponde à intensidade de brilho. A (Figura 25) apresenta alguns exemplos da aplicação do processo de segmentação por cor.

Para a segmentação por binarização, a Classe PDI disponibiliza 4 métodos que levam em conta aspectos específicos como: binarização direta e inversa por *Otsu* e, binarização direta e inversa adaptativa.

Os métodos adaptativos exigem que seja escolhida uma máscara matricial que será aplicada na imagem de forma a realizar a adaptação da estimativa de *threshould* ou limiar e, a constante de subtração da média ou média ponderada.

Para este projeto foram utilizados uma máscara de tamanho 5 e a constante de subtração também igual a 5. Os resultados da aplicação destes métodos de segmentação por binarização são apresentados na (Figura 26).

A Classe PDI também disponibiliza um método que permite segmentar objetos, levando-se em conta seu deslocamento no espaço. Este método recebe duas imagens RGB como entrada. Primeiramente ele realiza a operação de subtração de pixel entre duas imagens por meio da função subtract (OPENCV, 2020) e, em seguida, converte a imagem subtraída para o formato HSV. Depois disto realiza a operação de segmentação por cor HSV. O resultado destes processos é uma nova imagem que mostram apenas a última posição correspondente imagem do objeto que foi deslocado. A (Figura 27) apresenta um exemplo da aplicação do método para a segmentação por movimento.

Para realizar a segmentação (Figuras 28 a 30) por bordas a Classe PDI disponibiliza uma funcionalidade que recebe uma imagem como parâmetro. Depois disso ele realiza procedimentos para preparar a imagem e, eliminar possíveis ruídos por meio de filtro para uniformizar a imagem (Subseção 3.5.5.13) e, por fim, aplica a detecção de bordas por *Canny* (Subseção 3.5.5.12).

3.5.5.10 Ajuste de Perspectiva

Podem ocorrer situações em que a imagem foi capturada de uma perspectiva diferente da ideal. Para solucionar este problema pode-se utilizar a funcionalidade de ajuste de perspectiva da Classe PDI. Este método usa a função *warpPerspective* que ajusta a perspectiva de uma imagem tendo como referência uma matriz predefinida de pontos gerada pela função *getPerspectiveTransform*, ambas da biblioteca do *OpenCV* (OPENCV, 2020). A função *getPerspectiveTransform* recebe dois parâmetros que correspondem aos pontos iniciais e aos pontos finais do espaço a ser considerado. Depois disto, é aplicado o método *warpPerspective* que recebe como parâmetro a imagem original, matriz correspondente a perspectiva obtida pela função *getPerspectiveTransform* e, por último, o tamanho em *pixel* da nova imagem a ser gerada. Este método retorna para o requisitante a imagem com a perspectiva ajustada.

Para este projeto a função *getPerspectiveTransform* foi parametrizada para ajustar a perspectiva de forma a capturar parte da imagem abaixo de seu centro vertical. A (Figura 31) exibe o resultado da aplicação deste método.

3.5.5.11 Filtro para a suavização e eliminação de ruídos

Durante a etapa de pré-processamento de uma imagem, algumas vezes se faz necessário suaviza-las e/ou eliminar ruídos. Para isso a Classe PDI disponibiliza diversos funcionalidades que implementam as funções do OpenCV voltadas para a aplicação de filtros (OPENCV, 2020).

O método $cv2.blur(img, (5, 5))$ recebe como parâmetros a imagem a ser filtrada e a dimensão da máscara a ser aplicada. Para este projeto foi aplicado uma máscara dimensionada em $5x5$. A Figura 32 apresenta um exemplo da aplicação deste filtro.

O filtro $cv2.GaussianBlur(img, (5, 5), 0)$ recebe como parâmetro a imagem a ser filtrada, a dimensão da máscara a ser aplicada e por último o grau de suavização. Para este projeto os dois últimos parâmetros foram $5x5$ e 0 (zero). Um exemplo do resultado da aplicação deste filtro é apresentado na (Figura 33).

O filtro $cv2.medianBlur(img, 5)$ recebe apenas dois parâmetros que são a imagem e a intensidade da suavização. Para este projeto foi escolhido o valor 5 para o último parâmetro. A (Figura 34) mostra o resultado da aplicação deste filtro.

A função $cv2.bilateralFilter(img, 5, 75, 75)$ exige que sejam informados quatro parâmetros. Destes, o primeiro é a imagem que receberá o tratamento e, o segundo indica o tamanho do filtro. Cabe ressaltar que quanto maior o valor do segundo parâmetro, mais lenta será a execução do processo. O terceiro parâmetro é o *sigma* color e o quarto o *sigma space*. Esses dois últimos parâmetros estão diretamente relacionados ao tratamento de ruído e suavização e, é recomendado que seus valores não sejam menores que 10 para se obter resultados significativos (BARELLI, 2018). Para este projeto foram definidos os seguintes valores para do segundo ao quarto parâmetros: 5, 75 e 75. Na (Figura 35) é apresentado o resultado da aplicação deste filtro em algumas das imagens capturadas pelo DRR.

3.5.5.12 Realce de bordas

Algumas vezes é necessário realizar o realce das bordas durante a fase de pré-processamento e, para isso, a Classe PDI disponibiliza funcionalidades extraídas da biblioteca do OpenCV (OPENCV, 2020).

Por meio do Método Sobel desta biblioteca é possível aplicar filtros nas direções X (horizontal), Y (vertical) ou ambos, X e Y. Esta função recebe 3 parâmetros sendo estes: referente a imagem da qual se deseja realçar as bordas, o tipo de variável que armazenará o valor que representa cada pixel (para este sistema foi utilizado o parâmetro *cv2.CV_8U* que corresponde a valores inteiros que variam de 0 a 255 e são armazenados em 8 *bytes*). O terceiro e quarto parâmetro definem como o realce será aplicado. Neste sentido no primeiro método o terceiro parâmetro é igual a 1 e o quarto é igual a zero para realçar apenas as bordas horizontais. Já para o segundo método foi utilizado o valor 0 no terceiro parâmetro e o valor 1 para o quarto. Você pode ver o resultado da aplicação destes filtros nas (Figura 36).

O método *Laplacian* da biblioteca *OpenCV* (OPENCV, 2020) permite a aplicação do filtro laplaciano. Este método requer apenas dois parâmetros, sendo o primeiro a

imagem que receberá o tratamento e, o segundo, assim como a função *Sobel* corresponde ao tipo de variável que armazenará o valor que representa o pixel. Para este projeto, o valor deste parâmetro é *CV_8U*. O resultado da aplicação deste filtro pode ser observado na (Figura 37).

A Classe PDI também permite realizar o aguçamento de bordas. Para isso é feito uso da função *Laplacian* seguida da função *subtract* da mesma biblioteca (OPENCV, 2020). A função *subtract* recebe dois parâmetros, sendo o primeiro a imagem original e, o segundo a imagem filtrada pela função *Laplacian*. A (Figura 38) mostra como a imagem original ficará após a aplicação deste filtro.

A funcionalidade para a realização do filtro de desaguçamento de bordas também utiliza uma sequência de funções da *OpenCV* sobrepostas para se atingir o resultado desejado (OPENCV, 2020). Primeiro a imagem em tons de cinza é suavizada por um filtro gaussiano; depois é realizada a subtração da imagem que em seguida é multiplicada por um número escalar. Por último é realizado a adição por meio do método *add* da mesma biblioteca. Este método recebe 2 parâmetros: o primeiro é a imagem original e, o segundo, é uma imagem contendo os detalhes que se deseja suprimir. Você pode verificar o resultado da aplicação deste filtro em uma imagem observado a (Figura 39).

O filtro de bordas de Canny é bastante utilizado pelo ERACI e faz uso da função *Canny* da biblioteca *OpenCV* e recebe 3 parâmetros obrigatórios (OPENCV, 2020). O primeiro é a imagem em tons de cinza que receberá o tratamento, o segundo e o terceiro correspondem a intensidade de detecção, mínimo e máximo, respectivamente que, para este projeto foram: 3 e 3. A (Figura 40) apresenta o resultado da aplicação deste filtro com os valores de parâmetros mencionados.

A funcionalidade que permite a aplicação do filtro de bordas de Hough utiliza a sobreposição das funções *Canny* e da função *HoughLinesP* (OPENCV, 2020). A primeira função foi parametrizada com os valores: imagem em tons de cinza, 70, 255. A segunda função recebe 5 parâmetros. O primeiro corresponde a imagem a ter as linhas detectadas, o segundo corresponde a distância perpendicular da origem até a linha, o terceiro corresponde ao ângulo perpendicular da linha ao eixo, o quarto e quinto parâmetros correspondem aos valores de comprimento mínimo e máximo da linha. Para este projeto foram utilizados como valores para estes parâmetros: A imagem filtrada por *Canny*, 1, $\pi/180$, 10, 200. A imagem resultante da aplicação deste filtro é mostrada na (Figura 41).

3.5.5.13 Filtro para operações morfológicas

As operações morfológicas desempenham a modificação do formato ou da estrutura dos objetos em uma imagem. Para suprir necessidades relacionadas a isto, a Classe PDI disponibiliza os métodos para a aplicação de filtros de erosão, dilatação, abertura, fechamento, uniformizar e morfológico. Ambos os métodos fazem uso de um elemento

estruturante que podem ser retangular, elíptico ou em forma de cruz. Para a criação destes elementos estruturantes pode ser utilizada a função *getStructuringElement* da biblioteca *OpenCV* (OPENCV, 2020).

A operação de erosão ocorre pela execução da função *erode* da mesma biblioteca. Esta função recebe os parâmetros referentes a imagem original, elemento estruturante e o número de iterações.

Já a operação de dilatação acontece por meio da utilização da função *dilate* que também recebe os mesmos parâmetros da função *erode*. Na prática a diferença entre os dois métodos acima é que o primeiro realiza a tarefa de desgastar a imagem e, a outra realiza o inverso. A (Figura 42) mostra a transformação da imagem após a aplicação destes filtros.

A operação de abertura é caracterizada pela operação de erosão seguida da operação de dilatação e para isto. Já a operação de fechamento é utilizada para preencher a imagem corrigindo pontos na imagem de interesse que foram danificados durante o processo de binarização. Ela basicamente consiste na operação de dilatação seguida pela operação de erosão, ou seja, é o inverso da operação de abertura. Ambas os métodos utilizam a função *morphologyEx* que recebe 3 parâmetros. Estes são a imagem a ser modificada, o tipo de filtro a ser aplicado e o elemento estruturante a ser utilizado. A única diferença entre as duas operações é a escolha do tipo de filtro a ser aplicado pois, para se realizar a operação de abertura, utiliza-se a constante *MORPH_OPEN*, caso contrário é utilizado a constante *MORPH_CLOSE*. Você pode visualizar um exemplo destes processos na (Figura 43).

Quando é conveniente aplicar filtros que visam uniformizar o brilho em uma imagem, pode-se fazer uso da funcionalidade para a uniformizar-la que, basicamente, aplica um filtro de abertura em uma imagem, depois realiza a subtração da imagem filtrada pela imagem original e, por fim realiza a adição da imagem filtrada pelo filtro de subtração por ela mesma. A (Figura 44) mostra um exemplo da aplicação deste filtro.

O filtro morfológico também é utilizado para realizar modificações na imagem e, assim como o método anterior, faz uso da função *morphologyEx* (OPENCV, 2020). A diferença neste caso é a possibilidade de se escolher 3 tipos diferentes de parâmetros para o filtro; podendo este ser: *MORPH_GRADIENT*, *MORPH_TOPHAT* e *MORPH_BLACKHAT*. Cada um tem um propósito diferente, impactando diretamente na imagem. O primeiro é caracterizado pela diferença entre a dilatação e a erosão, o segundo é a diferença entre a imagem de entrada e a abertura da imagem e, por fim, o último é a diferença entre o fechamento da imagem de entrada e a imagem de entrada. Pela (Figura 45) você pode observar exemplos da aplicação deste filtro utilizando cada um destes parâmetros.

3.5.5.14 Detecção de cantos na imagem

A detecção de cantos é uma forma de extrair características de uma imagem. Para isso a Classe PDI disponibiliza os métodos para a detecção de cantos de Harris e detecção de cantos de ShiThomasi. O primeiro método faz uso da função *cornerHarris* da biblioteca *OpenCV* que recebe 4 parâmetros sendo estes a imagem de entrada em tons de cinza, *blockSize* que é o tamanho dos *pixel* vizinhos considerados como cantos para detecção, *ksize* que é o parâmetro de abertura derivado do filtro *Sobel* utilizado e, por último k que é um parâmetro livre do detector de *Harris* (OPENCV, 2020).

O segundo método utiliza a função *goodFeaturesToTrack* que também recebe 4 parâmetros. Estes parâmetros correspondem: a imagem de entrada em tons de cinza, o número de cantos que se deseja encontrar, o nível de qualidade que é um valor entre 0 e 1 e por último a distância euclidiana mínima entre os cantos detectados.

A (Figura 46) mostra um exemplo da aplicação destes filtros para a detecção de cantos em imagens.

3.5.5.15 Extração de características

São extraídas três características consideradas fundamentais para o reconhecimento de padrões (BARELLI, 2018). Para isso foi implementado três métodos responsáveis para a extração das características relativas ao aspecto e dimensão da imagem bem como suas características inerciais. O primeiro método extrai a moda, média e desvio padrão referente a composição das cores de uma imagem. O segundo extrai características referente à dimensão da imagem segmentada por cor. O último extrai os 24 momentos da imagem bem como os 7 momentos invariantes de HU (HU et al., 2012).

Estas características são extraídas da imagem e armazenada no banco de dados, em registro vinculado por relacionamento a imagem para agilizar os processos de treinamento e, geração dos classificadores. Além disso, estes métodos estão presentes nos DRR e, neste caso, eles são extraídos em tempo real e aplicados aos algoritmos de classificação.

Esta metodologia para extração das características da imagem foi escolhida para a realização de teste do ERACI, devido à sua simplicidade e abrangência.

3.5.5.16 Extração de Imagem Assinalada

Este método assinala uma imagem com base em suas características. Para isso o método, primeiramente, segmenta por cor, utilizando o espaço de cores HSV baseando-se nos valores da moda dentro de um limiar dimensionado pelo desvio padrão da imagem, convertida em tons de cinza. Depois disso, ele realiza a detecção de bordas por meio de *Canny* e, assim por meio do centro de massa e da média dimensional obtida desta mesma imagem segmentada, é desenhado um círculo e inserido uma cruz no ponto que corresponde

ao centro de gravidade da imagem. O tamanho do círculo e da cruz são proporcionais a área segmentada.

Por fim, a imagem RGB, segmentada e circulada é retornada para o requisitante. A (Figura 47) mostra a imagem resultante da execução deste método.

3.5.6 Geração de Inteligência Artificial

3.5.6.1 Introdução

A geração dos classificadores acontece por meio do script python *inteligencia.py* que será apresentado na (Seção 4.2) desta dissertação. Este módulo quando iniciado realiza o carregamento dos dados cadastrados e pré-classificados por usuários, por meio do aplicativo ERACI instalado no DM.

Com os dados carregados em memória é realizado o treinamento de diversos classificadores pelos algoritmos: *K-Nearest Neighbors (KNN)*, *Support Vector Machine (SVM)*, *Logistic Regression (RL)*, *Naive Bayes (NB)*, *Decision Tree (DT)*, *Random Forest (RF)* e *Multilayer Perceptron (MLP)*.

Todos estes algoritmos são abstraídos pelo módulo *scikitlearn* (SCIKIT-LEARN, 2020) disponível para ser utilizado na construção de programas em *python*. Esta abstração permite que o desenvolvedor se concentre mais na solução do problema, garantindo a este uma melhor produtividade na construção de sistemas baseados nas técnicas de aprendizagem de máquina.

Estes classificadores são gerados continuamente, sempre que ocorre um novo treinamento programado para acontecer a cada 50 novas classificações adicionadas pelo(s) usuário(s) especialistas.

Após o treinamento dos classificadores é verificada a acurácia de cada um deles, por meio do método *kfold* (SCIKIT-LEARN, 2020). O classificador é armazenado na forma de arquivo no DGGC para posterior utilização pelo próprio sistema de visão computacional quanto para o envio, por meio de serviço específico, ao DRR.

A função de previsão do módulo de inteligência artificial, decide dentre os diversos algoritmos, qual aquele que apresentou a melhor acurácia durante os testes realizados durante as etapas de treinamento. Primeiramente verifica qual o melhor classificador para prever a superclasse relativa aos valores característicos da imagem e estima a superclasse que melhor representa a imagem. Depois, realiza o mesmo procedimento para verificar a subclasse desta imagem. Como resultado da função de previsão, é retornado uma tupla contendo a, superclasse, classe e um valor para acurácia do processo. Este valor de acurácia do processo é calculado por meio do valor médio resultante da soma dos valores de acurácia do algoritmo previsor da superclasse, subclasse e do resultado da função *predict_proba*

do classificador para a subclasse estimada. Esta técnica permite a incorporação de uma abordagem conexionista no sistema, conforme é apresentado por (LUGER, 2013).

3.5.6.2 K-Nearest Neighbors (KNN)

O KNN é uma metodologia de aprendizagem baseada em instancias. Esse tipo de algoritmo não constrói um modelo após o treinamento e sim, armazena os exemplos de treinamento. Assim, a generalização ou a previsão é feita somente quando uma nova instancia precisa ser classificada. Ele basicamente identifica um grupo de objetos K no conjunto de treinamento que estão mais próximos do objeto de teste e, atribui um rótulo baseado na classe mais dominante desta vizinhança.

A verificação é feita por meio da Equação (3.1) que fornece a distância euclidiana entre as características do novo objeto e os objetos presentes na base de dados. Depois disso ele verifica os K objetos mais próximos do novo objeto e o classifica com base na maioria como é mostrado na Equação (3.2).

$$DE(x, y) = \sqrt{\sum_{i}^{p}(xi - yi)^2} \tag{3.1}$$

Onde, DE é a Distância Euclidiana e, x e y são vetores de características do objeto.

Com base na distância euclidiana, é verificado quais as K classes estão mais próximas do novo objeto a ser classificado. Desta forma, o novo objeto é classificado na classe predominante dentre as K classes (Equação 3.2).

$$\hat{y} = arg, max \sum_{(xi,yi)\in Iz} F(v = yi) \tag{3.2}$$

Onde, $F(.) = 1$ se o argumento $(.)$ for verdadeiro e 0 caso contrário; v é o rótulo da classe.

Assim, ele calcula a distância de semelhança entre um conjunto de treinamento, $(x, y) \in I$, e o novo objeto a ser classificado, $z = (\hat{x}, \hat{y})$ para determinar sua lista de vizinhos mais próximos. Na Equação (3.2) x representa o objeto de treinamento e y a classe de treinamento correspondente; $\hat{x}$ e $\hat{y}$ representam os mesmos dados, porém, relativo ao novo objeto a ser classificado (AWAD; KHANNA, 2015).

O valor de k deve ser cuidadosamente escolhido pois, um valor muito pequeno pode resultar na captura de valores "ruidosos" e a escolha de um valor muito grande pode resultar na captura de outras classes; ambos os casos promovem uma classificação errônea do novo objeto.

Para verificar o melhor valor para o atributo K, a função responsável pela geração do modelo analisa qual a melhor vizinhança que resulte na melhor acurácia do classificador.

Este valor foi estipulado para ser um dos números inteiros impares dentro de um range de 1 a quantidade de classes possíveis para um novo objeto.

Após realizar os devidos testes o módulo grava o classificador no dispositivo e assim, ele poderá ser utilizado posteriormente por ele próprio ou ser transferido por meio da web para um DRR.

3.5.6.3 Suport Vector Machines (SVM)

Esse algoritmo corresponde a métodos de aprendizagem supervisionados que analisam e reconhecem padrões. Seu funcionamento consiste em uma tarefa de aprendizagem de duas classes. Assim, ele constrói um modelo ou uma função de classificação que atribui novas observações a uma das duas classes de um hiperplano, o que faz dele um classificador linear binário não probabilístico. Um modelo SVM mapeia as observações como pontos no espaço, de modo que elas são classificadas em uma partição separada que é dividida pela maior distância até o ponto de dados de observação mais próximo de qualquer classe. Desta forma ele prevê que uma nova observação pertence a uma classe baseada em qual lado da partição eles caem. Por sua vez, os vetores de suporte são os pontos de dados mais próximos do hiperplano que divide as classes (AWAD; KHANNA, 2015).

Na (Figura 48) você pode perceber que o algoritmo estabelece durante o treinamento um *optimal hyperplane* (hiperplano ótimo) com uma *maximum margin* (margem máxima) utilizada para melhorar a classificação por meio da generalização. O cálculo do hiperplano utiliza equação similar a apresentada na Equação (3.3).

$$\vec{W} \cdot \vec{X} + b = 0 \tag{3.3}$$

Onde, w e x são os vetores de dados utilizados para o treinamento e, b é uma constante a ser calculada. Para calcular a margem máxima o algoritmo utiliza a Equação (3.4) de forma a obter o menor erro possível.

$$erro = \frac{1}{2}|W|^2 + c\sum_i ai \tag{3.4}$$

Onde, c é a punição por uma classificação incorreta, ai é a distância entre o hiperplano e as classes existentes.

3.5.7 Regressão Logística (RL)

A regressão logística é um modelo probabilístico de classificação estatística que prevê a probabilidade da ocorrência de um evento. Ela modela a relação entre uma variável dependente categórica X e um desfecho categórico dicotômico ou característica Y. A função logística (Equação 3.5) pode ser expressa como:

$$P(Y|X) = \frac{e^{\beta 0+\beta 1X}}{1 + e^{\beta 0+\beta 1X}} \tag{3.5}$$

A função logística pode ser reescrita e transformada como o inverso da função logística chamada de *logit* ou *log-odds*, que é a chave para generalizar os coeficientes da regressão logística como mostra a Equação (3.6).

$$logit(P(Y|X)) = \ln\left(\frac{P(Y|X)}{1 - P(Y|X)}\right) = \beta 0 + \beta 1X \tag{3.6}$$

Como pode ser visto na figura 49, a função logística pode receber um range de valores de entrada ($\beta 0 + \beta 1X$) variando de $-\infty$ a $+\infty$, proporcionando a saída ($P(Y|X)$) restrita a valores entre 0 e 1.

A transformação de *logit* de $P(Y|X)$ fornece um alcance dinâmico para regressão linear e pode ser convertido de volta em probabilidades. O método de regressão logística se encaixa em uma curva de regressão, utilizando os coeficientes de regressão $\beta 0$ e $\beta 1$, como mostrado na equação (3.6), onde a resposta de saída é uma variável binária e X é numérica. Como a curva de função logística não é linear, a transformação *logit*, conforme é mostrado na figura 49, é usada para realizar a regressão linear, na qual $P(Y|X)$ é a probabilidade de sucesso Y por um determinado valor de X. Usando o modelo linear generalizado, uma equação estimada de regressão logística (Equação 3.7) pode ser formulada como,

$$logit(P(Y = 1)|X1, X2, X3, ..., Xn) = \beta 0 + \sum_{k=1}^{n} \beta kXk \tag{3.7}$$

Estima-se que os coeficientes $\beta 0$ e βk $(k = 1, 2, ..., n)$ utilizem a *maximum likelihood estimation* (MLE) para modelar a probabilidade de que a variável dependente Y assuma o valor de 1 para os valores dados de $Xk(k = 1, 2, ..., n)$.

A regressão logística é amplamente usada em áreas em que o resultado é apresentado em formato binário. No caso deste projeto, a sua aplicação realiza a verificação por meio da comparação conhecida como um contra todos, ou seja, ela verifica se o novo objeto é de uma classe, caso não seja, ele pode ser de qualquer outra classe e, assim, em forma de cadeia, todas as possibilidades são verificadas até que se encontre a classe mais provável para este novo objeto (AWAD; KHANNA, 2015).

3.5.8 Naive Bayes (NB)

O Naive Bayes é um classificador probabilístico que aplica a teoria de Bayes com uma forte suposição de independência, de tal forma que a presença de uma característica individual não esteja relacionada a presença de outra. Na prática este algoritmo analisa uma base de dados histórica e gera uma tabela de probabilidade. Esta tabela de probabilidade

será consultada pelo algoritmo sempre que for necessário a classificação de um novo objeto (AWAD; KHANNA, 2015).

Assim, suponha que os recursos de entrada $x1, x2, ..., xn$ são condicionalmente independentes uns dos outros, dado os rótulos de classe Y tais como (Equação 3.8):

$$P(x1, x2, ..., xn|Y) = \prod_{i=1}^{n} P(xi|Y) \tag{3.8}$$

Um exemplo da aplicação deste classificador pode ser a classificação de duas classes $(0, 1)$. Neste caso definimos $P(i|x)$ como a probabilidade de que o vetor de medição $x = \{x_1, x_2, ..., x_n\}$ pertence à classe i. Além disso, definimos uma pontuação de classificação (Equação 3.9 e Equação 3.10):

$$\frac{P(1|x)}{P(0|x)} = \frac{\prod_{j=1}^{n} f(xj|1)P(1)}{\prod_{j=1}^{n} f(xj|0)P(0)} = \frac{P(1)}{P(0)} = \prod_{j=1}^{n} \frac{f(xj|1)}{f(xj|0)} \tag{3.9}$$

$$\ln \frac{P(1|x)}{P(0|x)} = \ln \frac{P(1)}{P(0)} + \sum_{j=1}^{n} \ln \frac{f(xj|1)}{f(xj|0)} \tag{3.10}$$

Onde, $P(i|x)$ é proporcional a $f(x|i)P(i)$ e $f(x|i)$ é a distribuição condicional de x para objetos da classe i.

Este é um modelo eficaz e imensamente atraente devido à sua simplicidade e robustez. Como este algoritmo não requer aplicação de esquemas complexos para a estimativa de parâmetros iterativos para grande conjunto de dados.

3.5.8.1 Decision Tree (DT)

Uma *Decision Tree* ou Árvore de Decisão é simplesmente um processo passo a passo para decidir a categoria que algo pertence. Para resolver um problema deste tipo poderia ser utilizado um fluxograma de perguntas em que a resposta fosse dada de forma encadeada até que se chegue a uma determinada categoria, a qual o objeto provavelmente pertença (HARTSHORN, 2016).

Quando treinamos um classificador por meio deste algoritmo, é criado como resultado uma árvore de regras que, quando percorrida, leva à uma determinada classe de objetos. Um exemplo de árvore de decisão pode ser visto na Figura 50.

No exemplo apresentado na (Figura 50), a classe à qual a fruta pertence estará relacionada com as respostas resultantes de cada nó do grafo. Este exemplo foi apresentado por (HARTSHORN, 2016).

Para uma melhor eficiência deste algoritmo podem ser utilizados dois cálculos (Equações 3.11 e 3.12) que fornecerão os melhores atributos candidatos para cada um dos nós (HARTSHORN, 2016).

$$Entropy(S) = \sum_{i=1}^{c} -p_i \dot{log} 2 p_i \tag{3.11}$$

$$Entropy(S) = Entropy(S) - \sum_{v} \varepsilon Values(A) \frac{S_{v)}}{S} Entropy(S_{v)}) \tag{3.12}$$

Onde, p_i é a quantidade de vezes que uma determinada característica aparece na lista de características dividido pelo número de registros.

O $Gain(S, A)$ tem como resultado um valor referente ao ganho de informação de um determinado atributo ou característica. Com base neste ganho, é estabelecido qual atributo será analisado primeiro, pertencendo a raiz da árvore e, quais os posteriores, até que se chegue no atributo folha que dará o resultado final da classificação.

Dentro do contexto de árvores de decisão, as vezes é necessário que se realize a poda. Para isto, pode se utilizar os conceitos de *bias* e/ou variância. Assim, *bias* (ou Viés) consiste no levantamento de erros por classificação errada e, variância que corresponde a erros por sensibilidades pequenas provenientes a mudanças na base de treinamento. Este último geralmente ocorre quando o algoritmo se adapta de mais a uma base de treinamento, conhecido como *overfitting*.

No ERACI, o classificador foi parametrizado para utilizar a entropia para estabelecer os nós de cada uma das árvores.

3.5.8.2 Random Forest (RF)

A *Random Forest* ou floresta randômica é uma abordagem de aprendizagem em conjunto para realizar classificações. Basicamente ela faz uso de uma coleção de árvores de decisão (DT) correlacionadas para isto. Assim, em vez de trabalhar apenas com uma única solução baseada na saída de uma árvore profunda, a floresta randômica agrega a produção de um número de árvores rasas, formando uma camada adicional para agrupamento (HARTSHORN, 2016).

Neste algoritmo os n preditores são agrupados para solucionar um problema de classificação ou de regressão por meio da média. Enquanto as árvores de decisão possuem uma alta variância e um alto bias, a floresta randômica utiliza uma média de múltiplas árvores de decisão para melhorar o desempenho da estimativa (HARTSHORN, 2016).

Uma árvore de decisão, em termos conjuntos como visto anteriormente, representa um classificador fraco. O termo floresta denota o uso de uma série de árvores de decisão para tomar uma decisão de classificação. Ao fazer uma média através do conjunto de árvores, ela promove a redução da variância da estimativa final (HARTSHORN, 2016).

O ERACI realiza a verificação de acurácia de diferentes quantidades de árvores, podendo estas variar entre 1 e 40, além de utilizar a função de entropia para estabelecer os nós

de cada uma das árvores. O classificador que prove a melhor relação, acurácia/quantidade, é utilizada para o armazenamento em arquivo e o envio para os DRR, para realizar a classificação das imagens em tempo de execução.

3.5.8.3 Multilayer Perceptron (MLP)

Este é um algoritmo baseado em uma rede de neurônios simples que mapeia conjuntos de dados de entrada em um conjunto de saídas. Um MLP compreende várias camadas de nós totalmente conectados por um grafo direcionado, no qual cada nó, com exceção do nó de entrada, é um neurônio com uma função de ativação não linear. O componente fundamental de um MLP é o neurônio. Assim, em um MLP, um par de neurônios é conectado em duas camadas adjacentes, usando bordas ponderadas (LUGER, 2013).

A (Figura 51) mostra um exemplo de MLP formado por três camadas de neurônios, incluindo uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Ela é alimentada pela camada de entrada e apresenta um resultado na camada de saída. Estes resultados são calculados em uma abordagem de *feedforward* da camada de entrada para a camada de saída (AWAD; KHANNA, 2015).

O número de neurônios de entrada depende das dimensões das características de entrada; o número de neurônios de saída é determinado pelo número de classes. O número de camadas ocultas e o número de neurônios em cada camada oculta dependem do tipo de problema que está sendo resolvido. Menos neurônios resultam em aprendizado ineficiente; um número maior de neurônios resulta em generalização ineficiente. Um MLP usa uma técnica de aprendizagem supervisionada chamada *backpropagation* para treinamento da rede. Em suas simples instanciações, o perceptron calcula uma saída y (Equação 3.13) processando uma combinação linear de entradas ponderadas de valor real através de uma função de ativação linear (LUGER, 2013).

$$y = \phi(\sum_{i=1}^{n})W_i X_i + b \tag{3.13}$$

Onde, w representa o vetor de presos, x é o vetor de entrada, b é o *bias*, e ϕ é a função de ativação. Geralmente, os sistemas MLP escolhem a função sigmoide logística (Equação 3.14) ou a função tangente hiperbólica (Equação 3.15) como função de ativação (AWAD; KHANNA, 2015).

$$f(x) = \frac{1}{(1 + e^{-x})} \tag{3.14}$$

$$f(x) = \tanh(x) \tag{3.15}$$

Essas funções oferecem conveniência estatística, pois são lineares perto da origem e saturam rapidamente quando afastadas da origem (AWAD; KHANNA, 2015).

O processo de aprendizagem MLP ajusta os pesos da camada oculta, de modo que o erro de saída seja reduzido. Começando com os pesos aleatórios, o MLP alimenta os sinais de padrão de entrada através da rede e reproduz o sinal de erro, começando pela saída. O sinal de erro de propagação é composto pela diferença entre valores reais ($On(t)$) e desejados (Tn). A função de erro pode ser resumida como apresentada na (Equação 3.16) (AWAD; KHANNA, 2015).

$$E(O_n(t)) = T_n - O_n(t) \tag{3.16}$$

O objetivo do processo de aprendizagem é minimizar a função de erro. Para encontrar o valor mínimo da função de erro, diferencie-a, em relação à matriz de peso. Para influenciar a taxa de convergência e, assim, reduzir os tamanhos das etapas em que os pesos passam por uma alteração adaptativa, é utilizado um parâmetro de aprendizagem $\eta(< 1)$. O peso do $i - esimo$ conectado à saída $j - esimo$ pode ser atualizado pela regra mostrada na (Equação 3.17). Esta equação mostra a adaptação do peso de forma iterativa na qual uma fração de erro de saída na iteração $(t + 1)$ é adicionada ao peso existente da iteração t (AWAD; KHANNA, 2015).

$$w_{ij}(t + 1) - w_{ij}(t) = \eta E(O_j(t)) \tag{3.17}$$

A melhor configuração para a MLP a ser utilizada pelo ERACI foi conseguida por meio da utilização da metodologia de tentativa e erro. Ela mostrou que para se chegar ao melhor resultado classificatório era conveniente utilizar 1000 iterações, com taxa de aprendizagem de $0,000010$ e 4 camadas ocultas com 400 neurônios cada uma. Os outros parâmetros da rede foram mantidos com seus valores padrão (SCIKIT-LEARN, 2020).

3.5.9 Configuração e Instalação dos Componentes e Módulos

Todos os sistemas do Robô foram estruturados no SO (Sistema Operacional) Raspbian Lite. O SO foi instalado em cartão de memória microSD SanDisk Ultra® de 64 GB. Este cartão foi previamente formatado utilizando o software SD Formater® versão 5.0.1. Para a gravação do SO no cartão foi utilizado o software Win32 Disk Imager 1.0.0. Após a preparação, o cartão de memória foi inserido no RPI que foi iniciado e configurado. A primeira configuração ocorreu com o auxílio de monitor e teclado acoplados no Raspberry Pi por meio das portas HDMI e USB e seguiu a seguinte sequência (RASPBERRY PI FOUNDATION, 2020):

1. Foi acessado a aplicação raspi-config por meio do comando sudo raspi-config.

2. Alteração do password.
3. Configuração de rede.
4. Configuração das opções localização.
5. Habilitação da Picamera e conexão SSH.

Em seguida foi instalado o SAMBA que é um servidor de arquivos remotos compatíveis como Windows. Com ele é possível compartilhar arquivos com o Windows e até criar um controlador de domínios. Para isso foram realizados os seguintes passos (ARTHUR, 2019):

1. sudo apt-get install samba samba-common-bin
2. sudo mkdir /home/eyerobot/
3. sudo chmod 777 eyerobot
4. sudo nano /etc/samba/smb.conf

Ao final do arquivo foi adicionado os seguintes comandos:

[eyerobot]
comment = Compartilhamento EyeRobot
path = /home/eyerobot
create mask = 0777
directory mask = 0777
writable = yes
security = share
browseable = yes
public = yes

Em seguida foi realizada a configuração da conexão serial e, para isso foi editado o arquivo "*sudo nano /boot/config.txt*" onde foi acrescentado ao final a linha "*enable_uart=1*". Após alterar este arquivo foram executados os seguintes comandos (HENDRIX, 2016):

1. *sudo systemctl stop serial-getty@ttyS0.service.*
2. *sudo systemctl disable serial-getty@ttyS0.service.*
3. *sudo nano /boot/cmdline.txt.*
4. Foi removido a linha "*console=serial0,115200*".

Em seguida foi instalado o OpenCV por meio do comando (PAJANKAR, 2015):

sudo apt-get install python3-opencv

Para que o sistema seja iniciado automaticamente após o carregamento dos recursos do Sistema Operacional foi editado o arquivo *rc.local* utilizando o comando (HENDRIX, 2016):

sudo nano /etc/rc.local

Ao final dele, antes do comando *exit 0*, foi adicionado o comando:

sudo python3 /home/eyerobot/init.py &

3.5.10 Acionamento do Robô

Para acionar o DRR é necessário seguir os passos a seguir:

1. Ligar o roteador;
2. Ligar o DRR;
3. Ligar o robô;
4. Conectar dispositivo móvel ao roteador;
5. Desativar troca de dados pelo chip do celular;
6. Abrir aplicativo móvel ERACI;
7. Clicar sobre o ícone do supervisório;
8. Clicar sobre o endereço válido listado.

Esta forma foi estabelecida para que se consiga o resultado esperado com a sua utilização. O DRR está configurado para se conectar ao *Service Set Identifier* (SSID) de nome "*ERACI_network*". Desta forma, quando o DRR é ligado ele procura este ponto de acesso *wifi* e, ao encontrar, conecta-se a ele. Caso este ponto de acesso esteja conectado à WAN ou de forma cabeada ou por meio de modem 3G conectado a ele, o DRR se conectará também com o DGGC.

Independente desta conexão com a rede WAN ele está apto a ser monitorado e controlado pelo DM por meio do subsistema supervisório. Para que isso seja possível, é necessário que o dispositivo móvel também esteja conectado ao mesmo ponto de acesso que o DRR está e que a troca de dados pelo chip no DM esteja desativado.

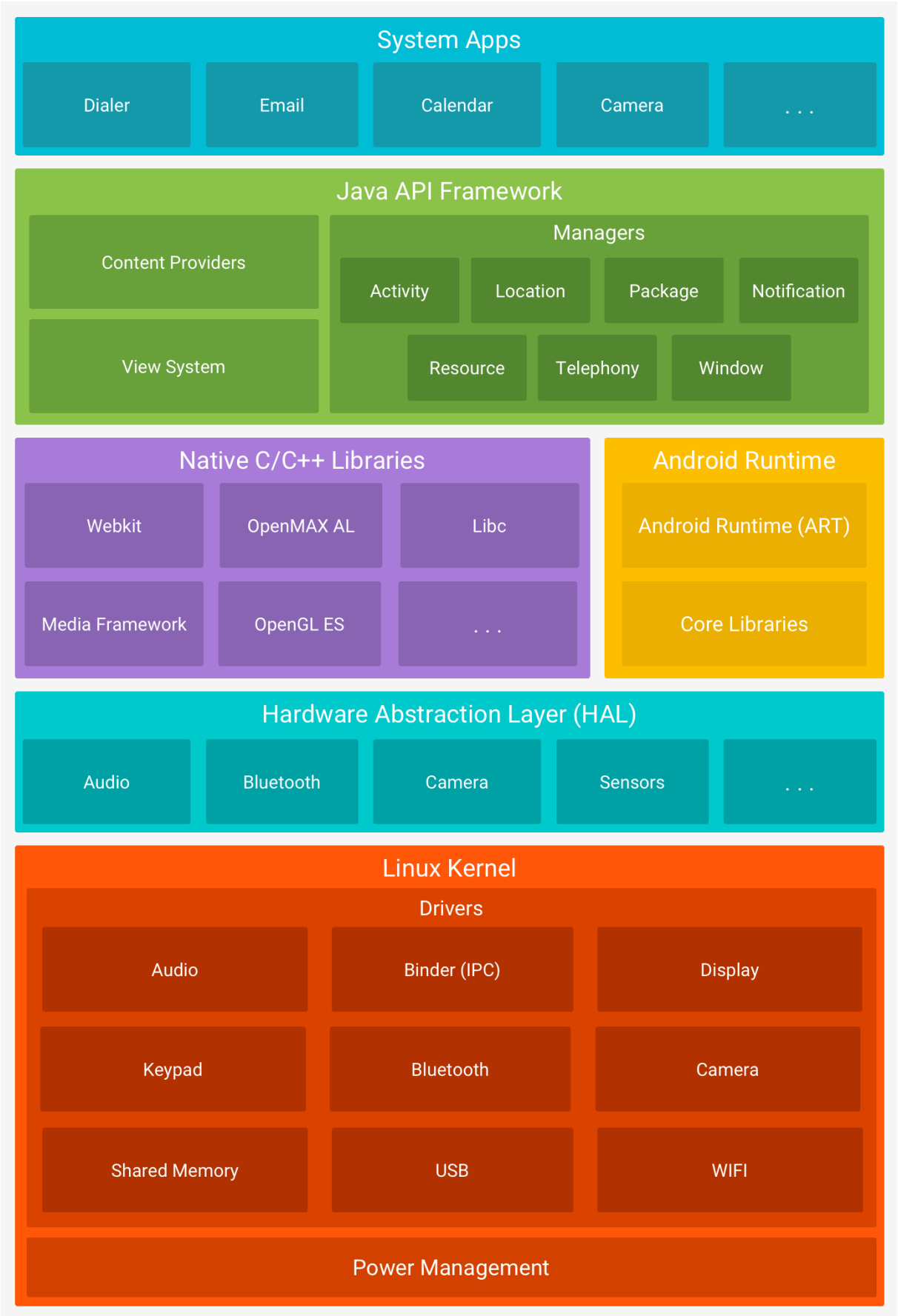


Figura 19 – Pilha de software do Android.

Fonte: (DEVELOPERS, 2020)

(a) (b) (c)

Figura 20 – Amostra de imagens capturadas pelo ERACI e extraídas do BD para serem processadas e analisadas. O número presente acima de cada uma delas correspondem ao número identificador (id) no BD. (a) Imagem de solo sem cobertura vergetal; (b) Imagem capturada com a câmera do DRR perpendicular às fileiras; (c) Imagem capturada com a câmera do DRR à 60° com e o disposivo perpendicular às fileiras.

Fonte: Autoria Própria

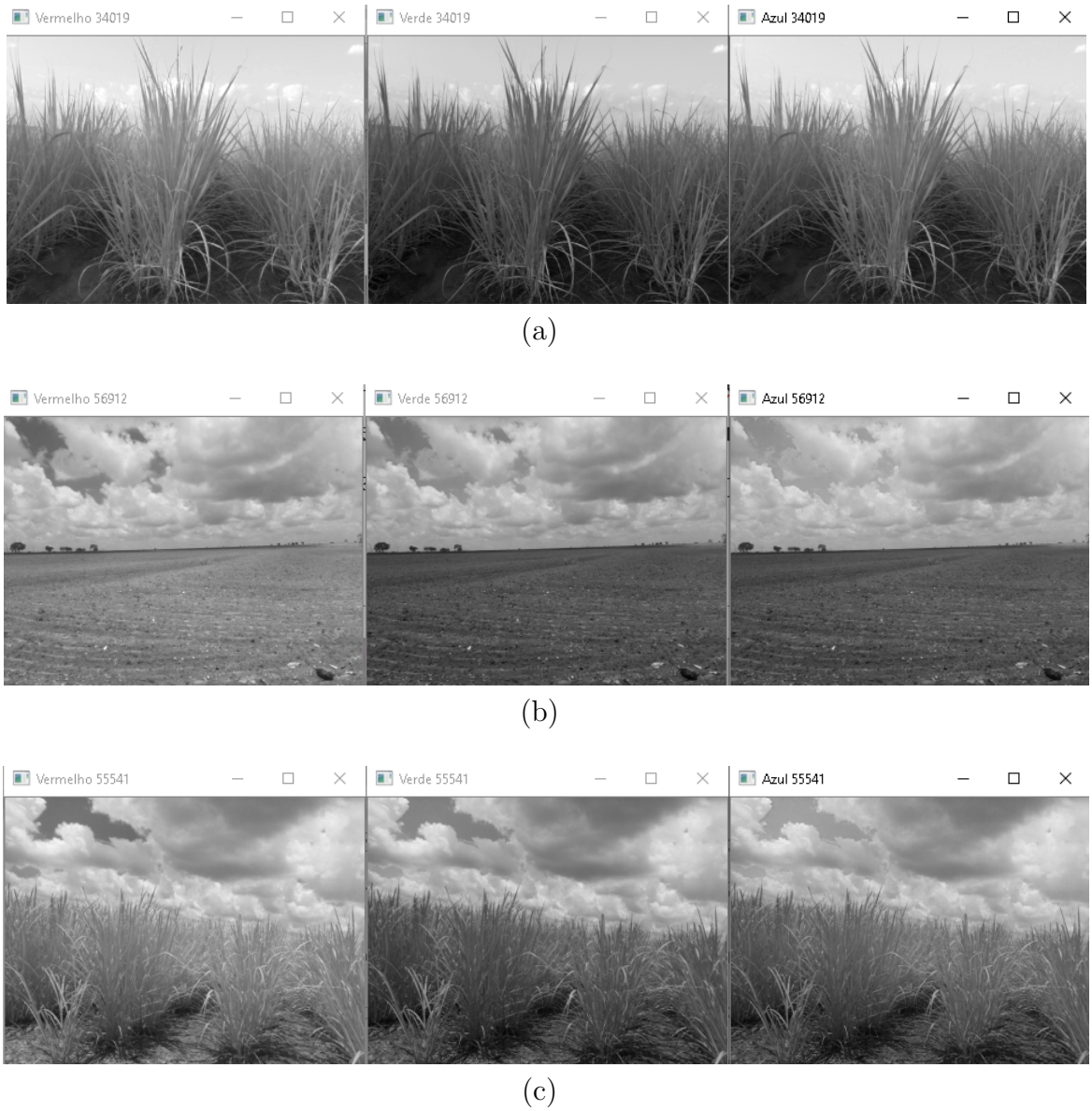
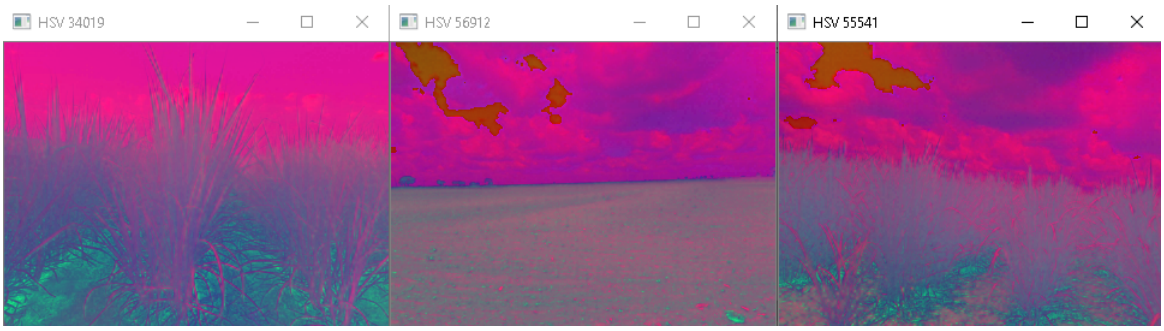


Figura 21 – Canais extraídos de Imagens RGB. (a) Fileiras de Cana-de-açúcar em estagio de crescimento mais avançado que as imagem apresentadas em (c); (b) Solo sem cobertura vegetal

Fonte: Autoria Própria



(a)

(b)

Figura 22 – Imagens convertidas. (a) em HSV e (b) em tons de cinza.

Fonte: Autoria Própria

(a) (b) (c)

Figura 23 – Resultado da aplicação dos métodos para a extração de cada um dos canais de uma imagem HSV para 3 imagens diferentes. (a) Matiz, (b) Saturação e (c) Valor.

Fonte: Autoria Própria

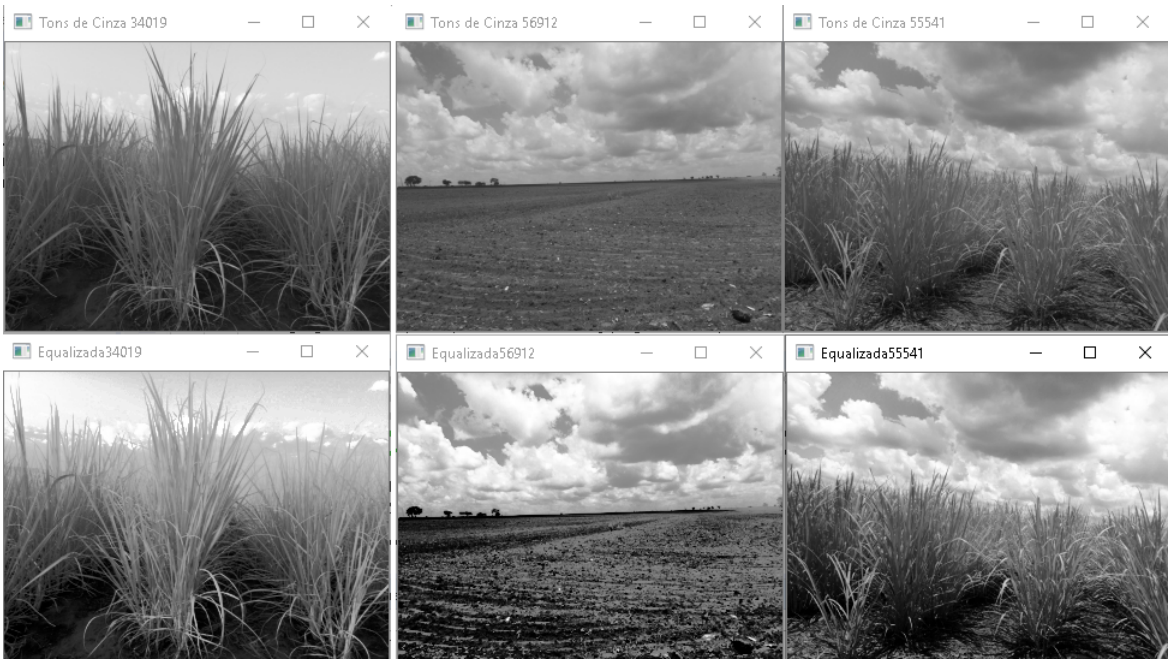


Figura 24 – Resultado da aplicação do método para equalização da imagem.

Fonte: Autoria Própria

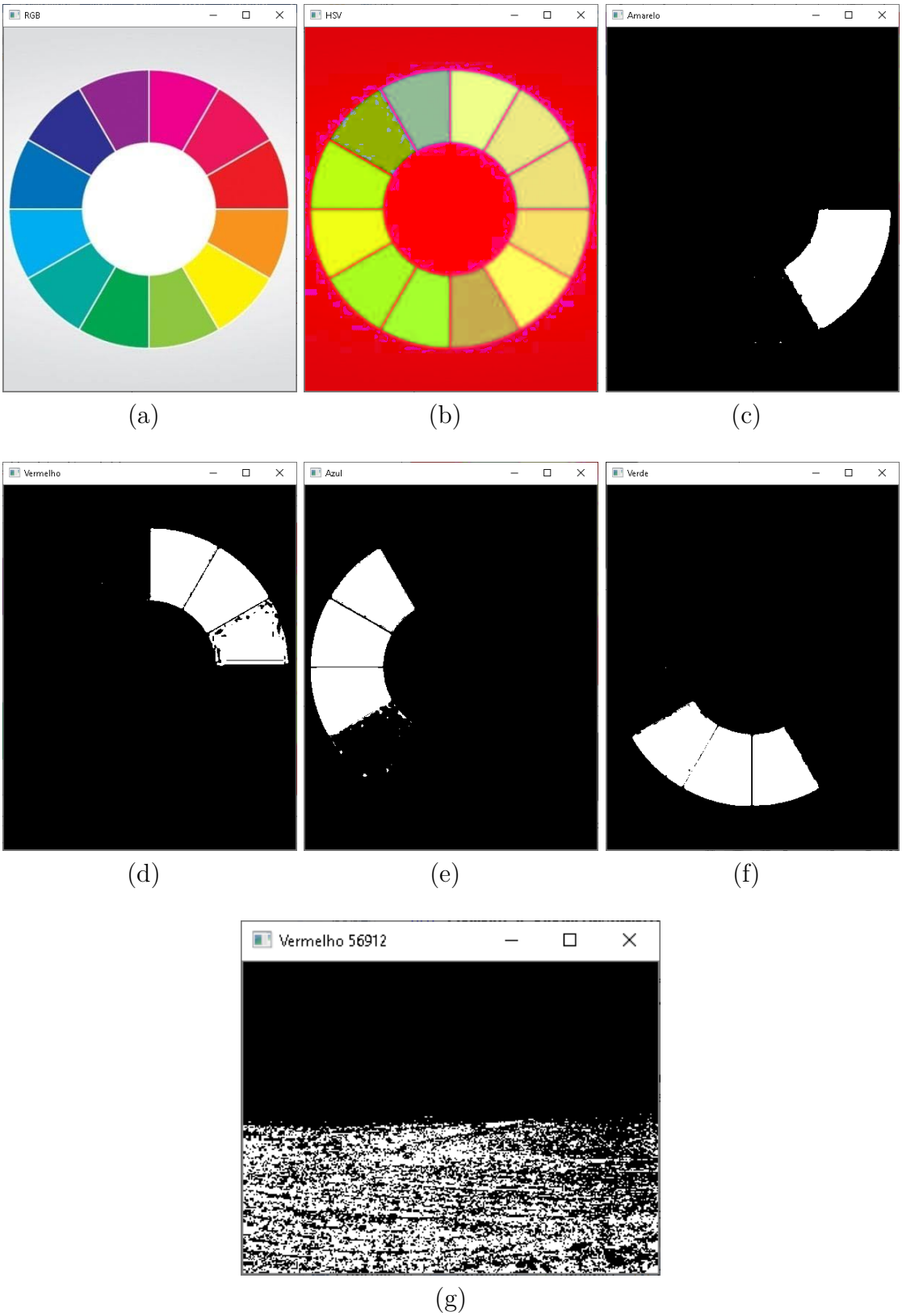
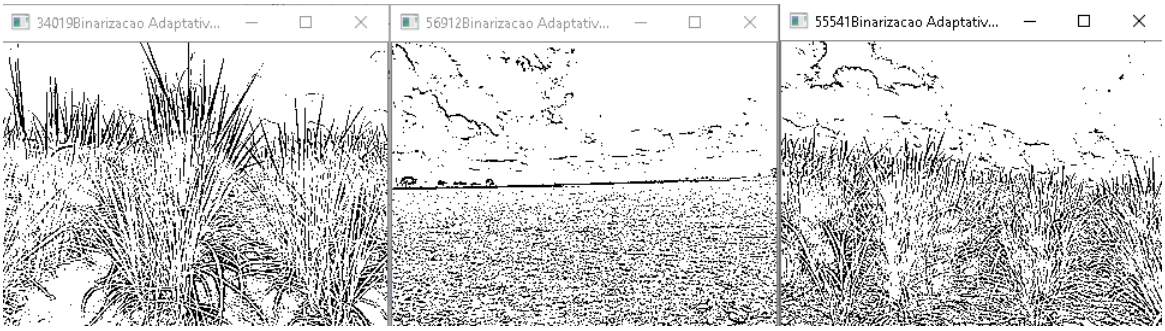


Figura 25 – Resultado da aplicação dos métodos para a segmentação de imagens por cor de uma paleta de cores. (a) Imagem Original em RGB; (b) Imagem convertida em HSV; (c) Segmentação da cor amarela; (d) Segmentação da cor Vermelho; (e) Segmentação da cor Azul; (f) Segmentação da cor Verde; (g) Segmentação da cor vermelho aplicado a imagem de ID 56912 extraída do banco de dados.

Fonte: Autoria Própria

(a)



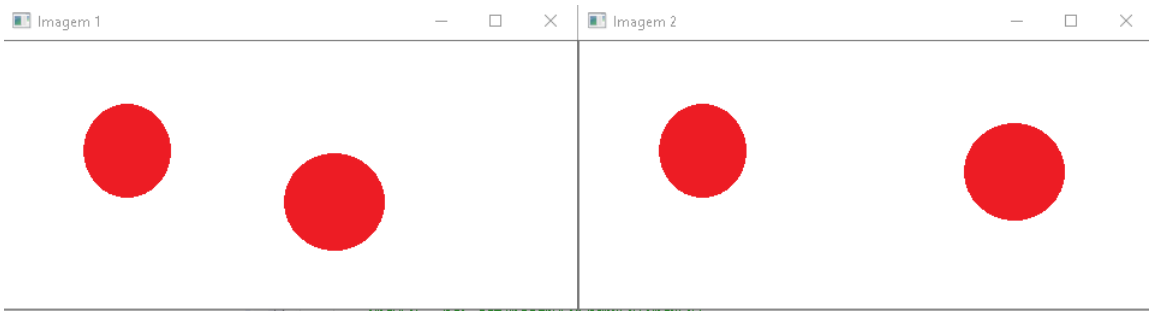
(b)

(c)

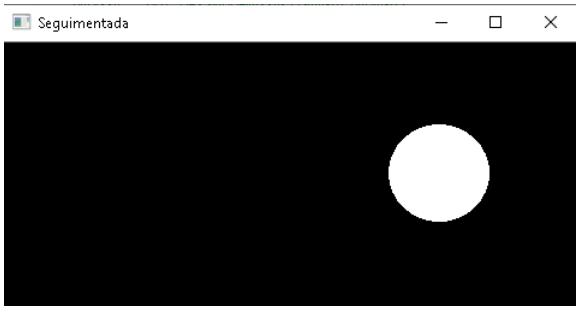
(d)

Figura 26 – Imagens segmentadas utilizando binarização (a) binarização adaptativa direta; (b) binarização adaptativa inversa; (c) binarização direta por Otsu; (d) binarização inversa por Otsu.

Fonte: Autoria Própria



(a)



(b)

Figura 27 – Resultado da aplicação do método para segmentação por deslocamento de um círculo vermelho em duas imagens. (a) duas imagens com dois círculos vermelhos, onde o segundo círculo foi deslocado para a direita na segunda imagem; (b) imagem que mostra a segmentação do objeto deslocado.

Fonte: Autoria Própria

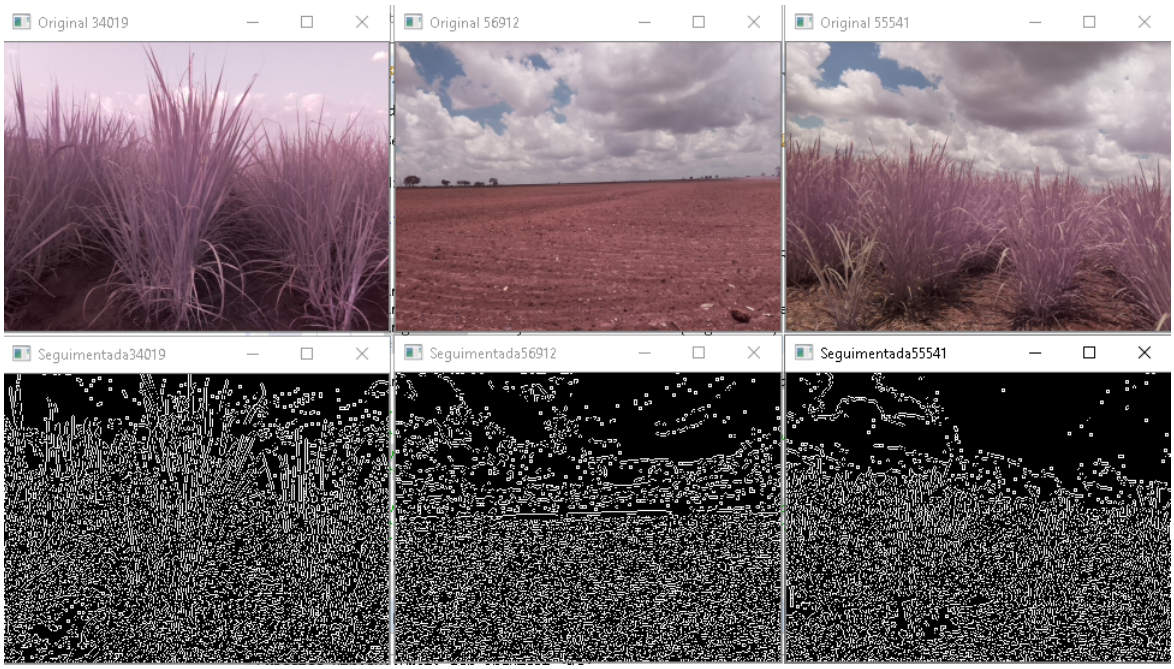


Figura 28 – Resultado da aplicação do método para a segmentação por bordas com sobreposição dos métodos para: equalização do histograma, binarização adaptativa inversa, uniformização e filtrar bordas com *Canny*.

Fonte: Autoria Própria

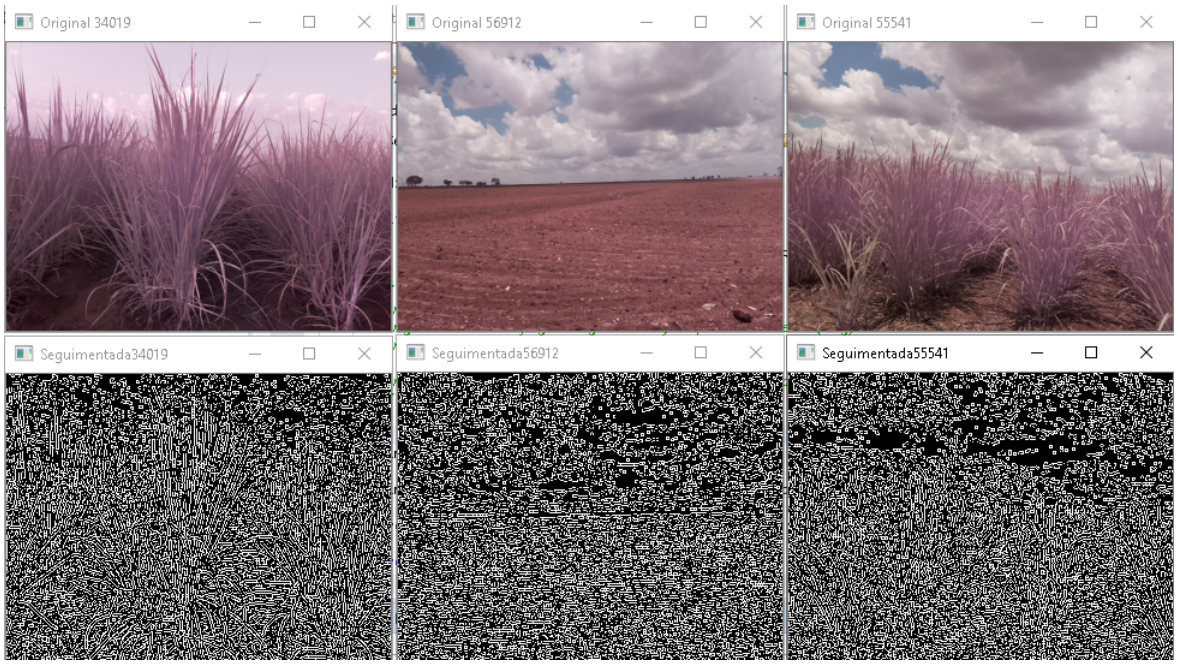


Figura 29 – Resultado da aplicação do método para a segmentação por bordas com sobreposição dos métodos para: uniformizar e filtrar bordas com *Canny*.

Fonte: Autoria Própria

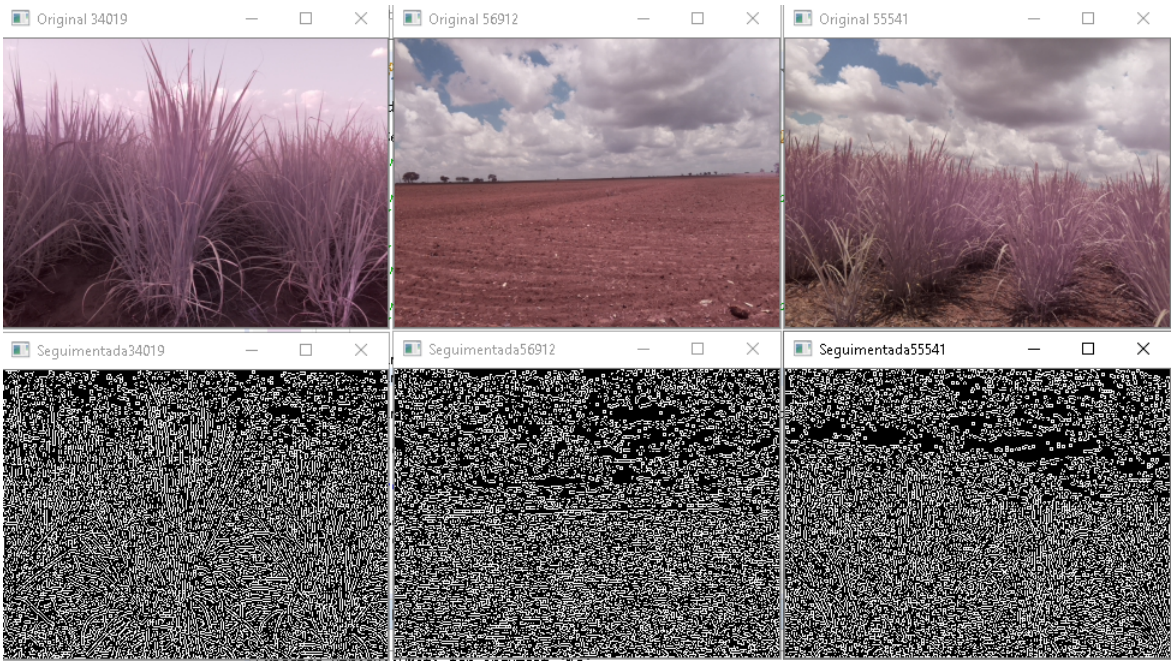
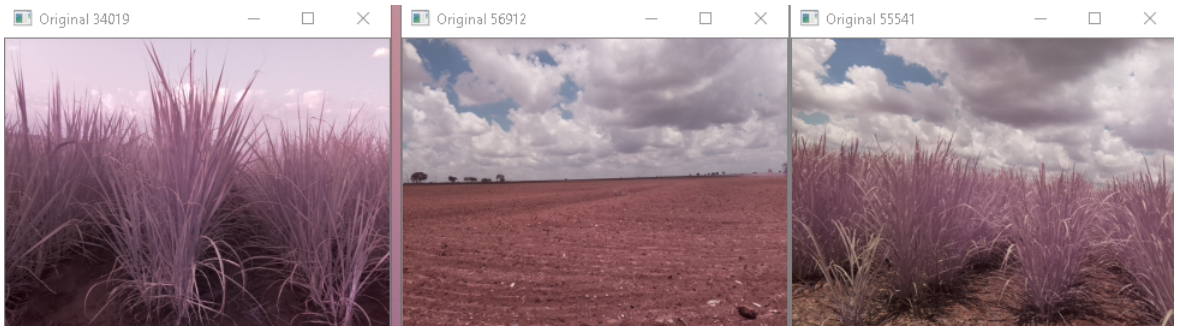
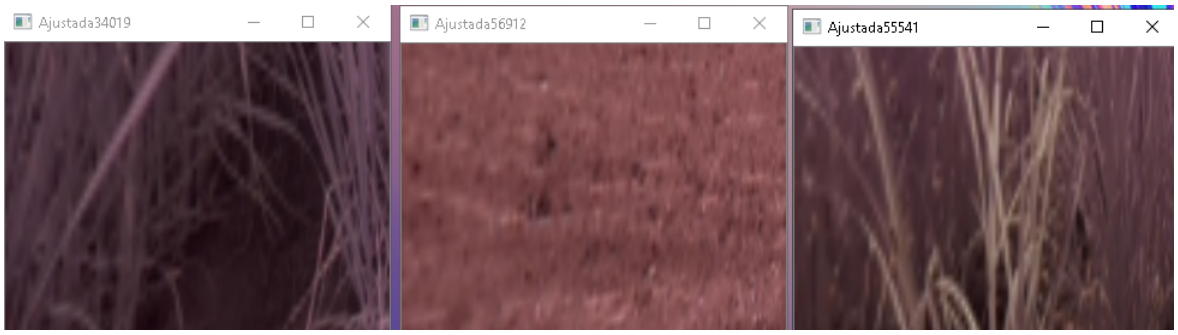


Figura 30 – Resultado da aplicação do método para a segmentação por bordas com sobreposição dos métodos para: equalização do histograma, uniformizar e filtrar com *Canny*.

Fonte: Autoria Própria



(a)



(b)

Figura 31 – Exemplo da aplicação do método de ajuste de perspectiva. (a) imagem original; (b) imagem gerada após o ajuste de perspectiva.

Fonte: Autoria Própria

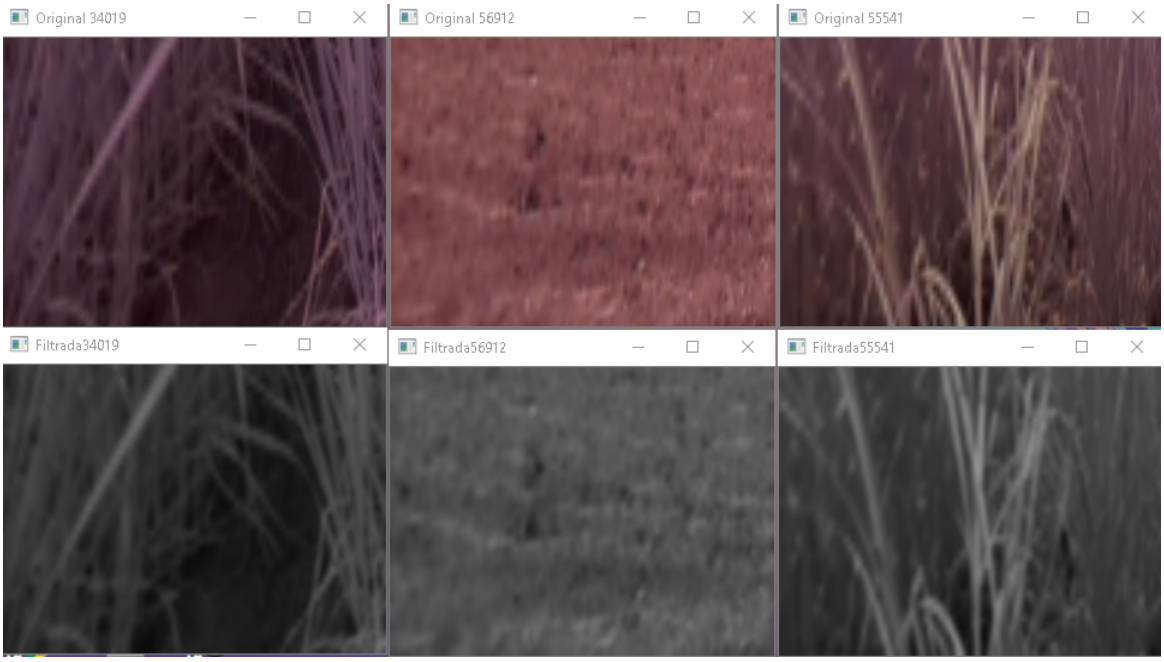


Figura 32 – Exemplo da aplicação do filtro Blur nas imagens com perspectiva ajustada.

Fonte: Autoria Própria

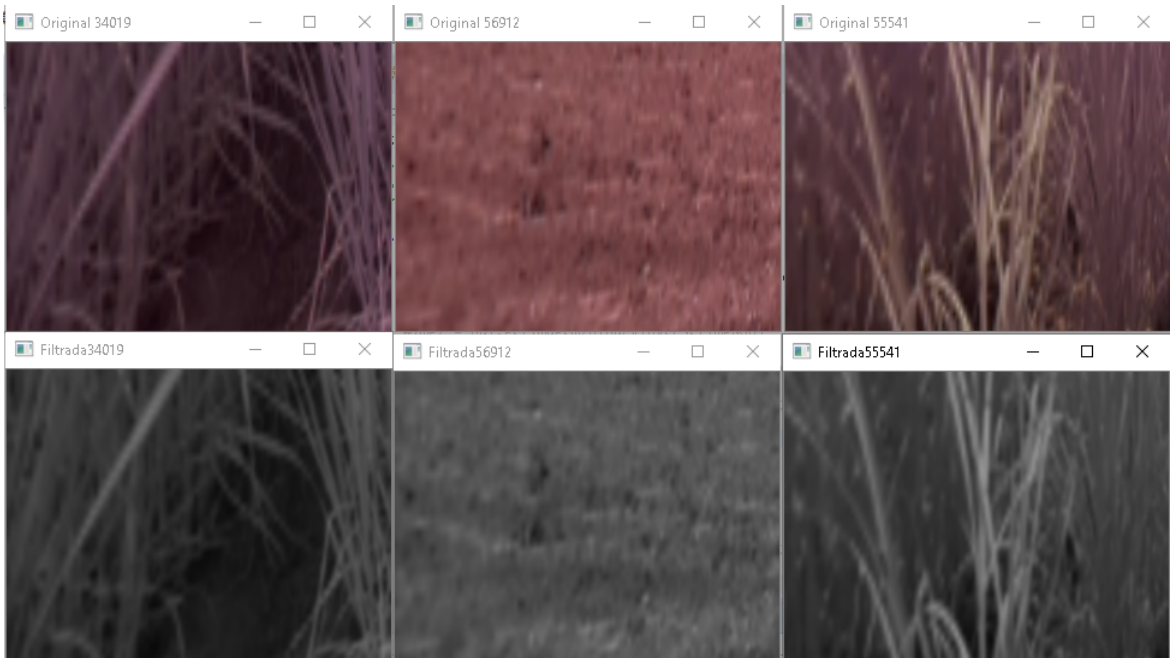


Figura 33 – Exemplo da aplicação do filtro gaussiano nas imagens com perspectiva ajustada.

Fonte: Autoria Própria

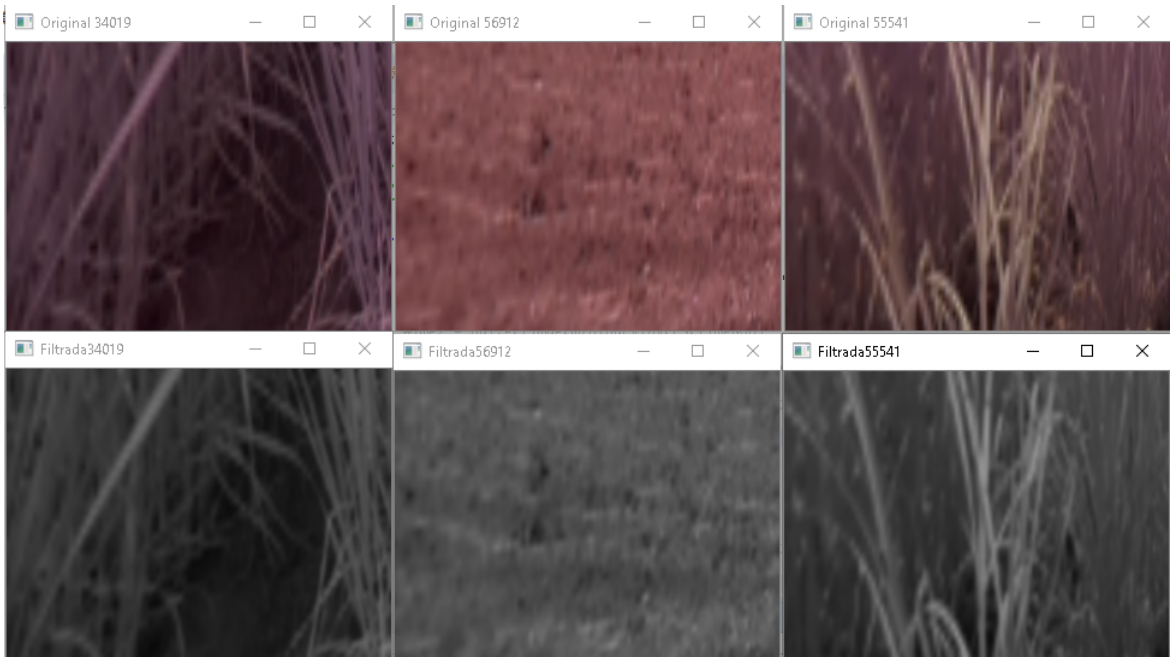


Figura 34 – Exemplo da aplicação do filtro filtro mediano nas imagens com perspectiva ajustada.

Fonte: Autoria Própria

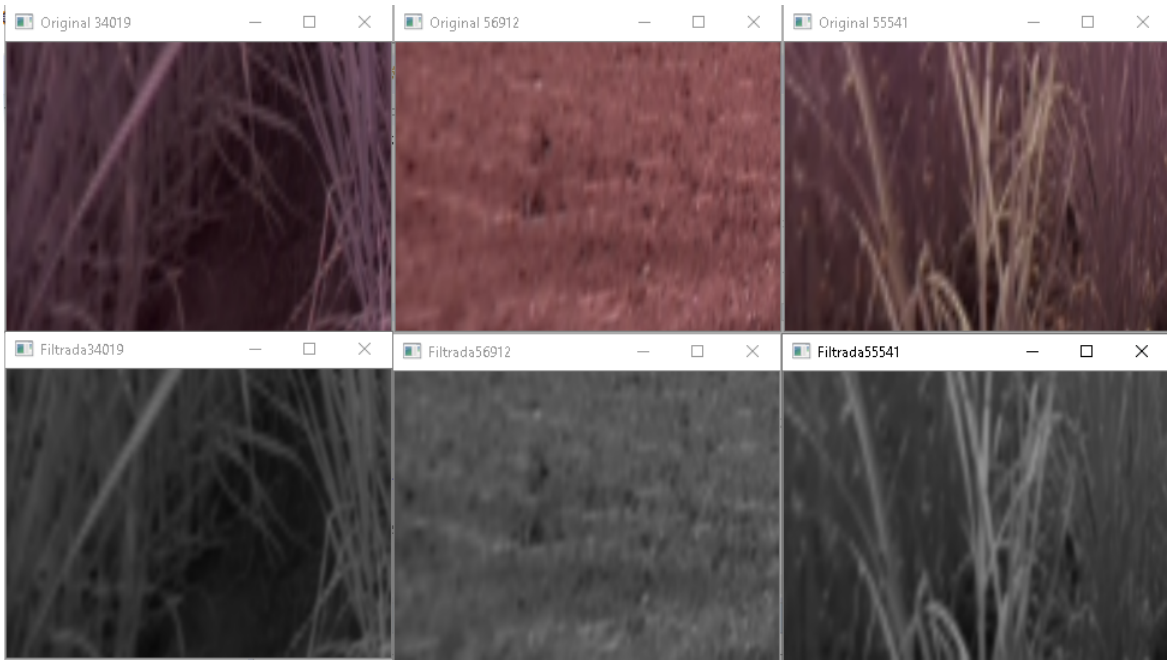
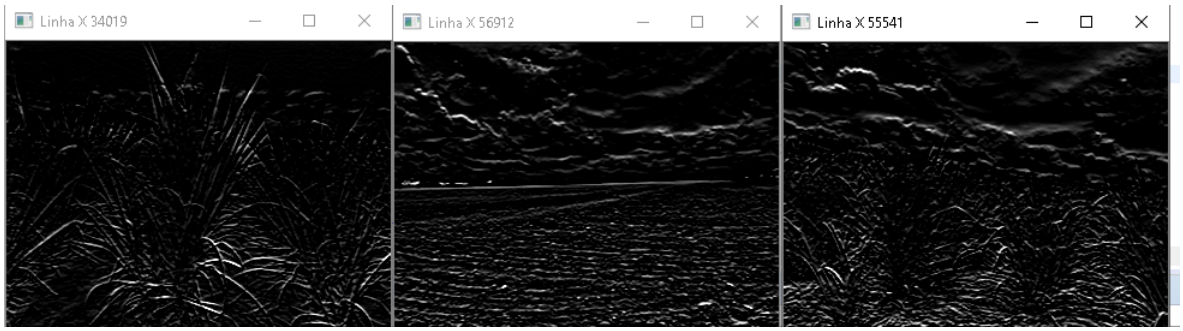


Figura 35 – Exemplo da aplicação do filtro filtro bilateral nas imagens com perspectiva ajustada.

Fonte: Autoria Própria



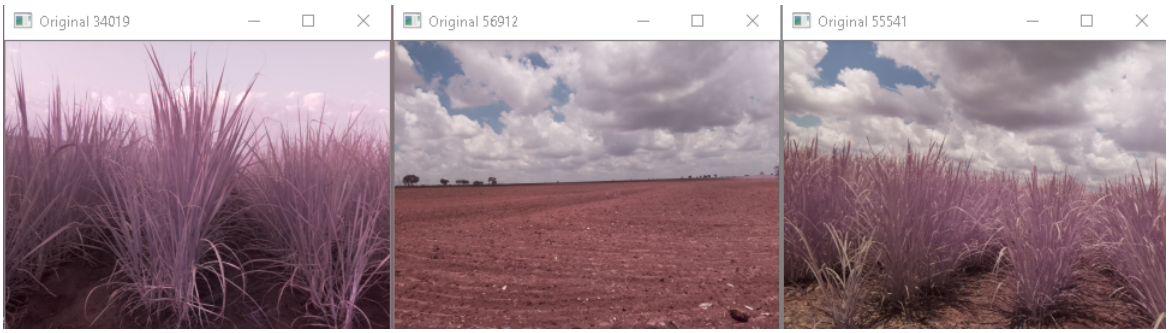
(a)

(b)

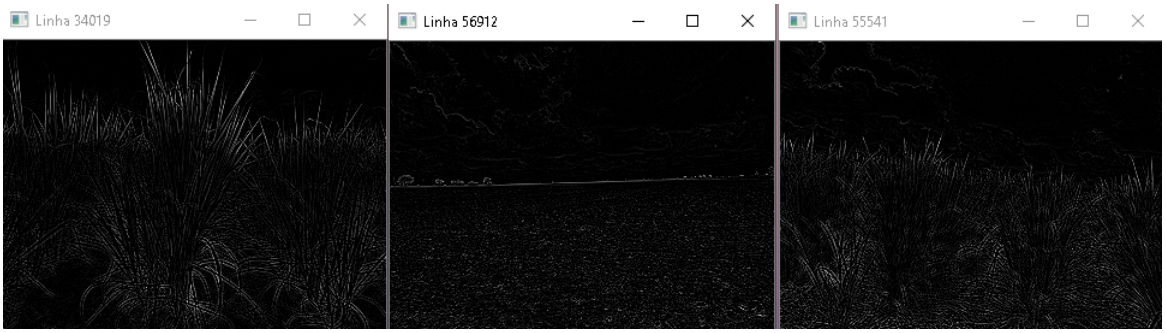
(c)

Figura 36 – Exemplo da aplicação dos filtros de Sobel. (a) aplicação do filtro na horizontal; (b) aplicação do filtro na vertical; (c) aplicação dos filtros na vertical e horizontal e, a soma de ambos.

Fonte: Autoria Própria



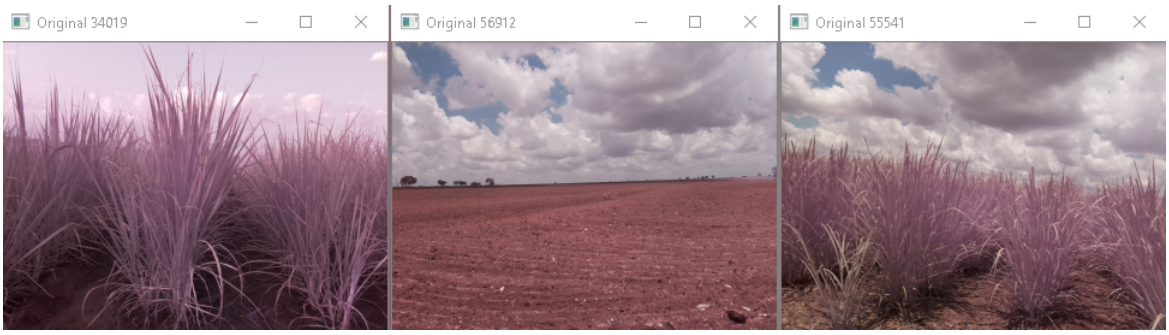
(a)



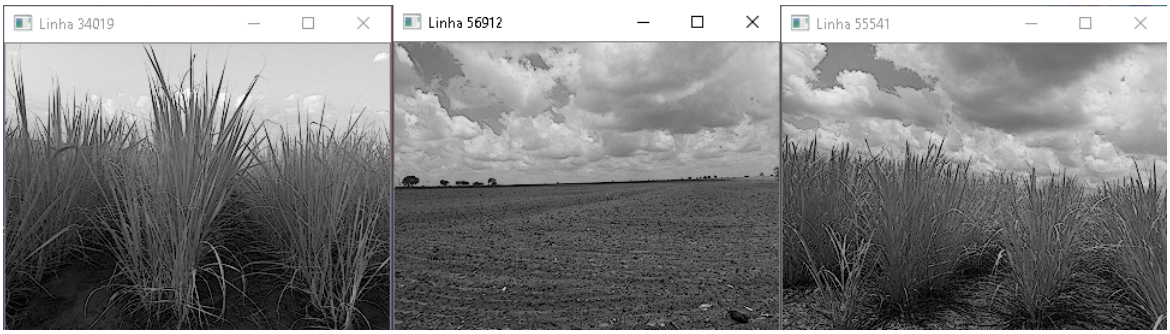
(b)

Figura 37 – Exemplo da aplicação do do filtro de bordas laplaciano. (a) Imagem original; (b) Resultado da aplicação do filtro laplaciano.

Fonte: Autoria Própria



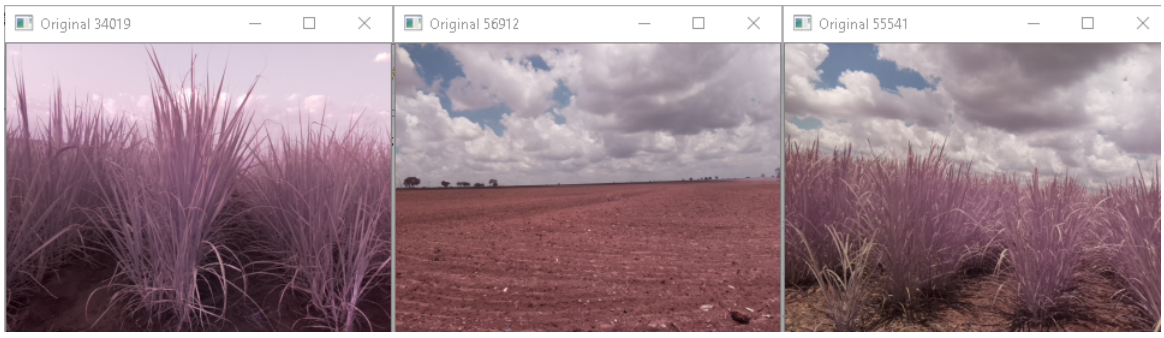
(a)



(b)

Figura 38 – Exemplo da aplicação do filtro de agucamento de bordas. (a) imagem original; (b) imagem após a aplicação do filtro para aguçamento das bordas.

Fonte: Autoria Própria

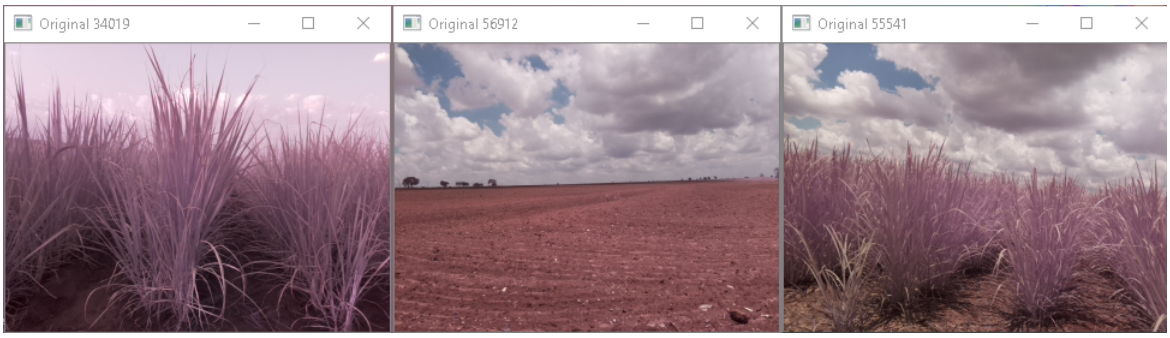


(a)

(b)

Figura 39 – Exemplo da aplicação do método filtro de desagucamento de bordas. (a) Imagem Original; (b) Imagem resultante da aplicação do filtro.

Fonte: Autoria Própria

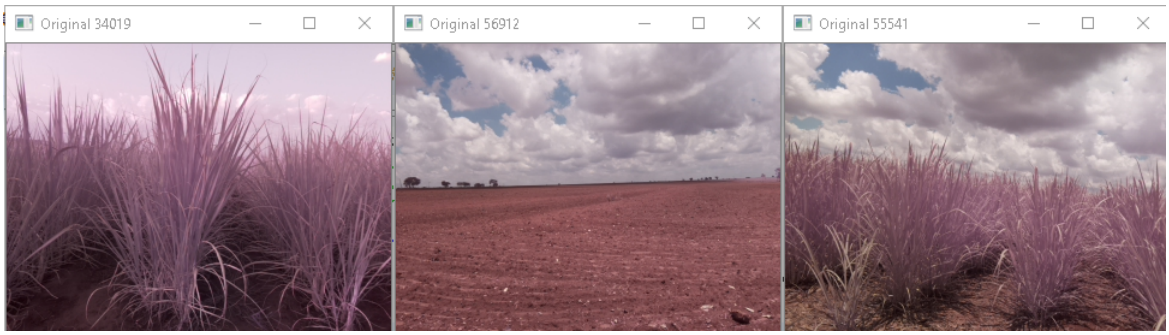


(a)

(b)

Figura 40 – Exemplo da aplicação do filtro de bordas de Canny. (a) Imagem original; (b) Imagem resultante da aplicação do filtro de Canny.

Fonte: Autoria Própria

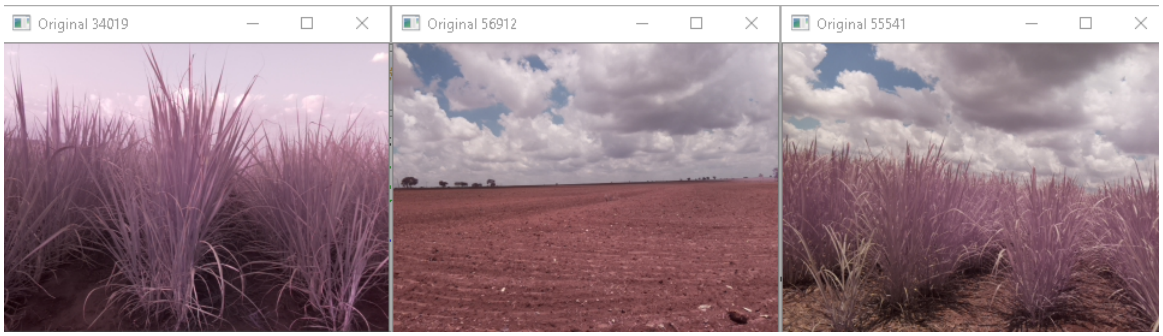


(a)

(b)

Figura 41 – Exemplo da aplicação do filtro de Hough. (a) Imagem Original; (b) Imagem após a aplicação do filtro.

Fonte: Autoria Própria



(a)

(b)

(c)

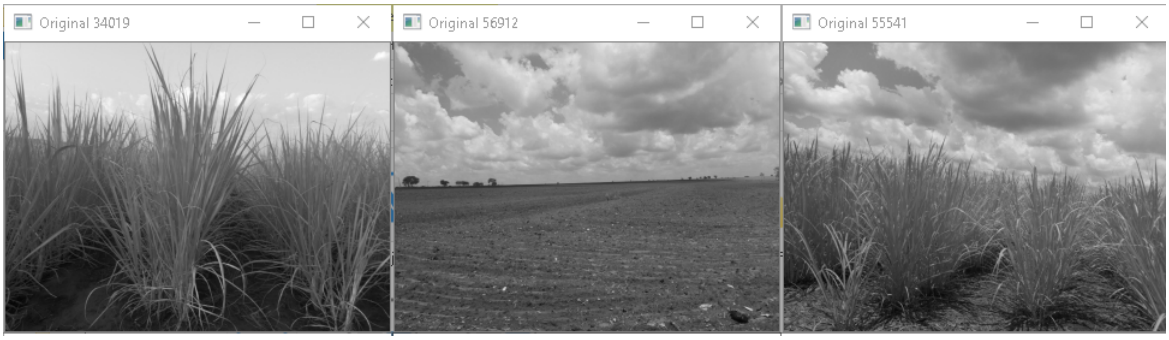
Figura 42 – Exemplo da aplicação do filtro de erosão. (a) imagem original; (b) imagem resultante da aplicação da erosão; (c) imagem resultante da aplicação da dilatação.

Fonte: Autoria Própria

(a) (b) (c)

Figura 43 – Exemplo da aplicação dos filtros de abertura e de fechamento. (a) imagem original; (b) imagem após a aplicação de abertura; (c) imagem após a aplicação de fechamento.

Fonte: Autoria Própria

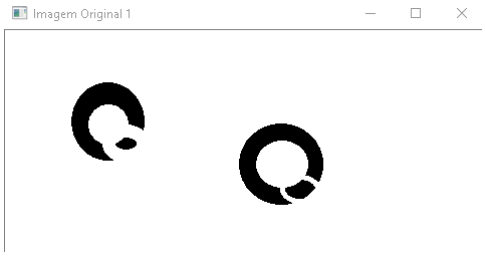


(a)

(b)

Figura 44 – Exemplo da aplicação do filtro para uniformizar. (a) Imagem original; (b) resultado da aplicação do filtro.

Fonte: Autoria Própria

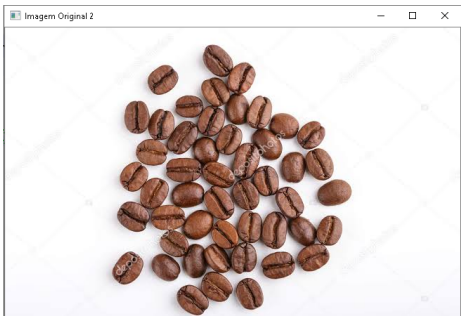


(a)

Fonte: Autoria Própria

(b)

Fonte: Autoria Própria



(c)

Fonte: (DEPOSITOPHOTOS, 2020)

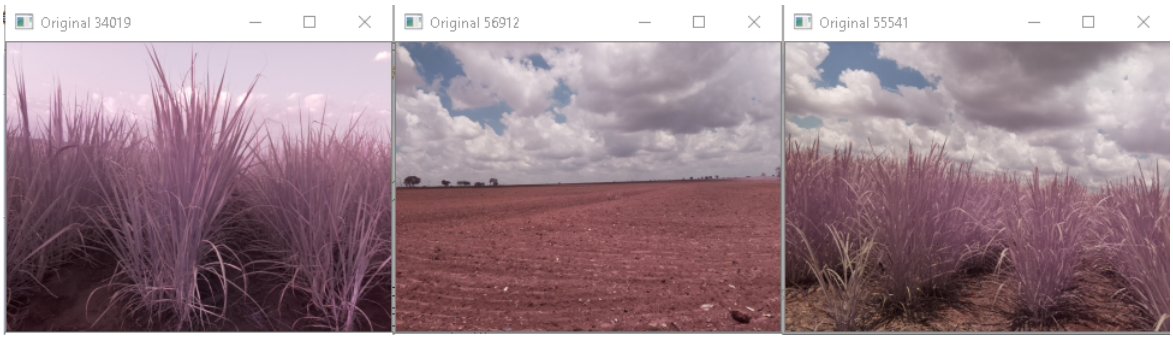
(d)

Fonte: Autoria Própria baseada na imagem obtida de (DEPOSITOPHOTOS, 2020)

(e)

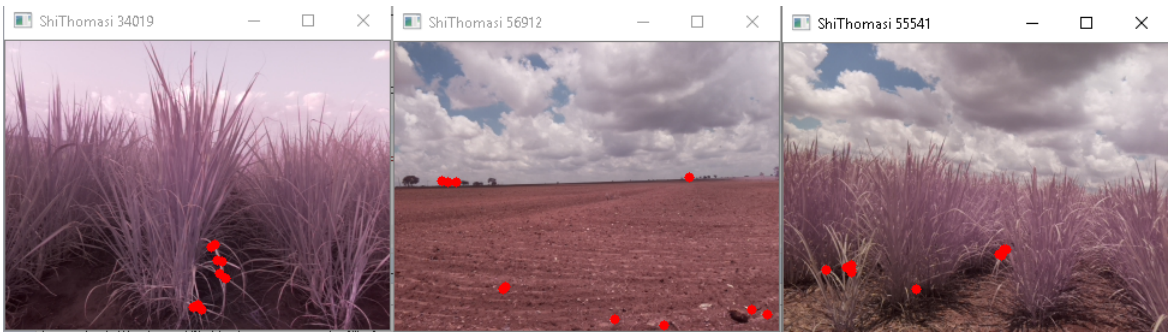
Fonte: Autoria Própria

Figura 45 – Exemplo da aplicação do filtro morfológico. (a) Imagem original (círculos); (b) MORPH_GRADIENT; (c) Imagem original (feijões); (d) MORPH_TOPHAT; (e) MORPH_BLACKHAT.



(a)

(b)



(c)

Figura 46 – Exemplo da aplicação dos filtros para detecção de cantos. (a) imagem original; (b) resultado da aplicação do filtro de detecção de cantos de Harris; (c) resultado da aplicação do filtro de detecção de cantos goodFeaturesToTrack.

Fonte: Autoria Própria

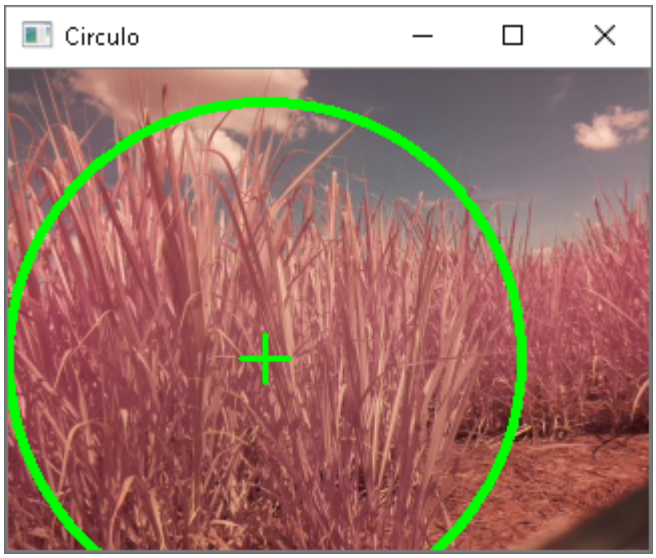


Figura 47 – Imagem assinalada levando em conta o centro de massa e a média em uma imagem pré-segmentada por cor.

Fonte: Autoria Própria

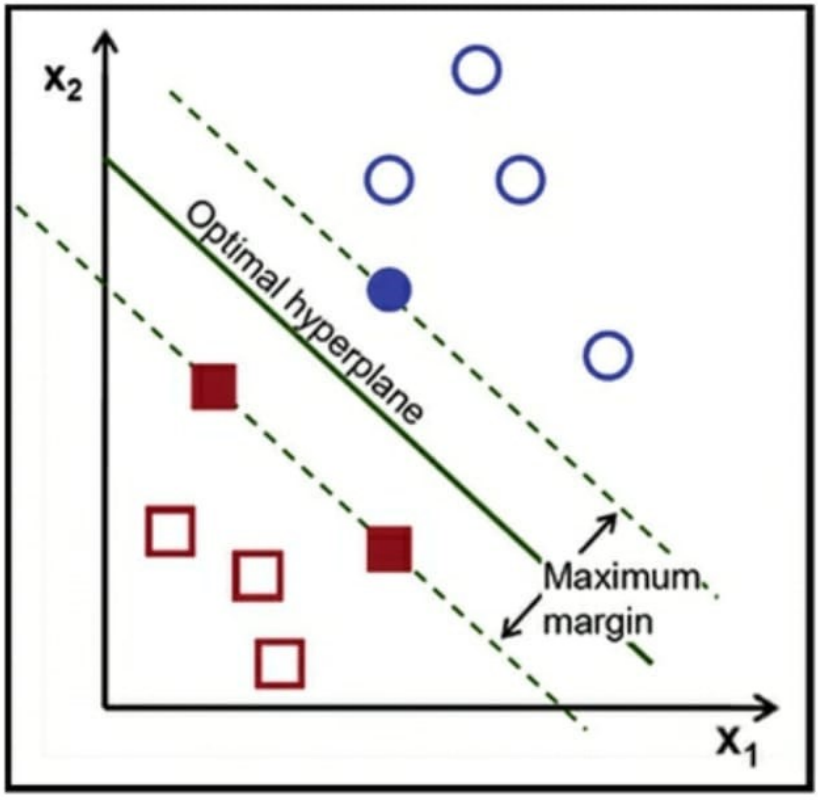


Figura 48 – O algoritmo SVM encontra o hiperplano que maximiza a menor distância entre os vetores de suporte.

Fonte: (AWAD; KHANNA, 2015)

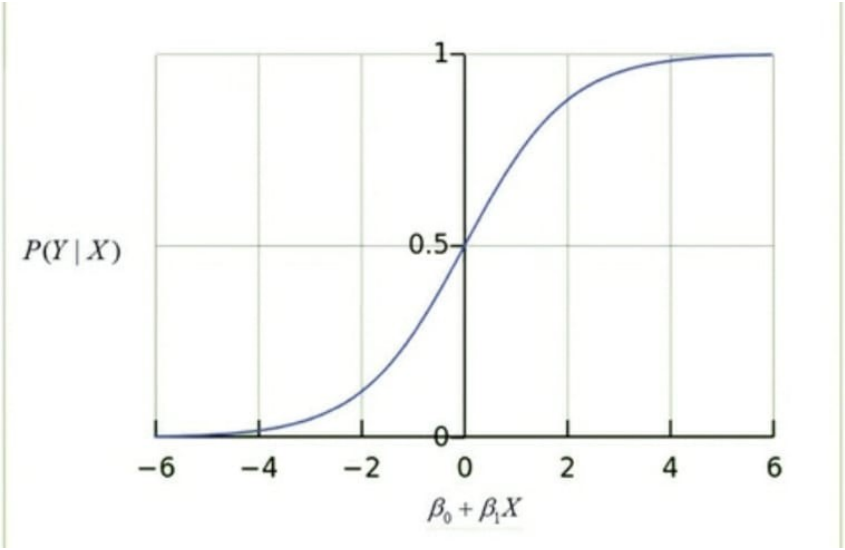


Figura 49 – Função Logística.

Fonte: (AWAD; KHANNA, 2015)

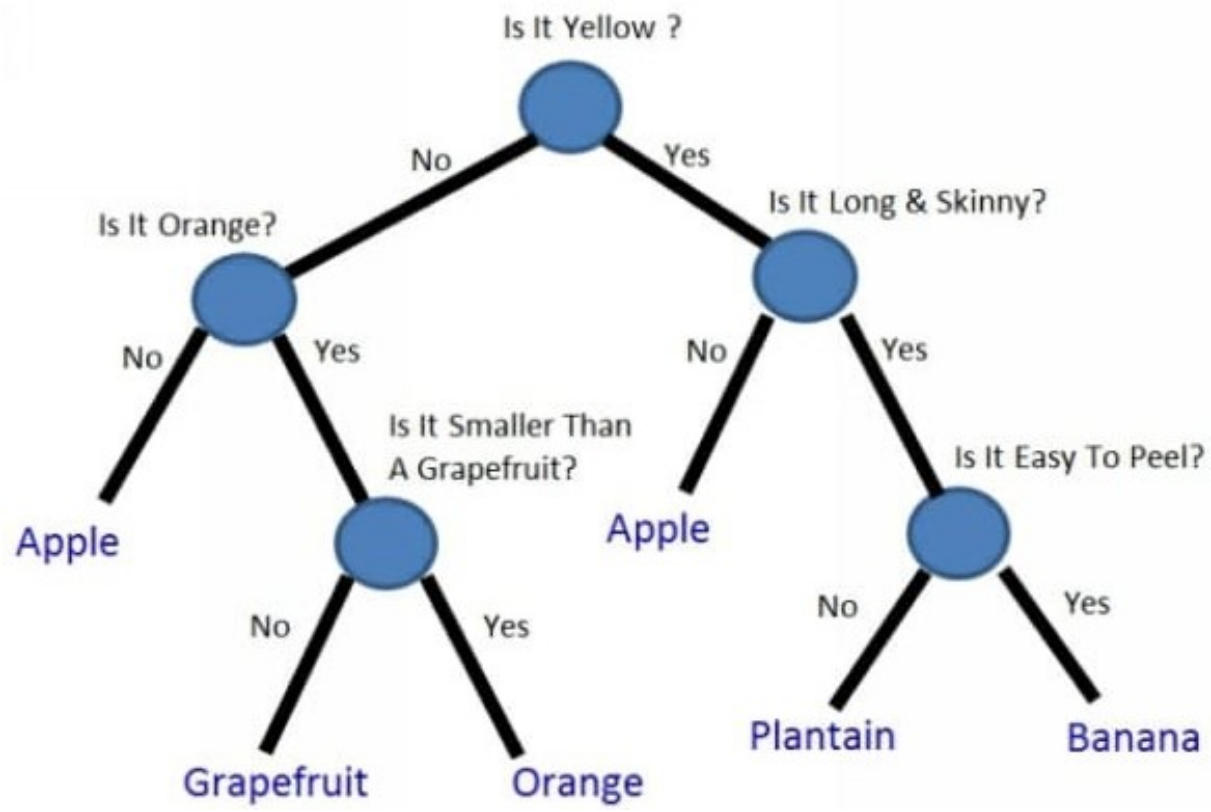


Figura 50 – Exemplo de aplicação de uma árvore de decisão para a classificação de uma fruta.

Fonte: (HARTSHORN, 2016)

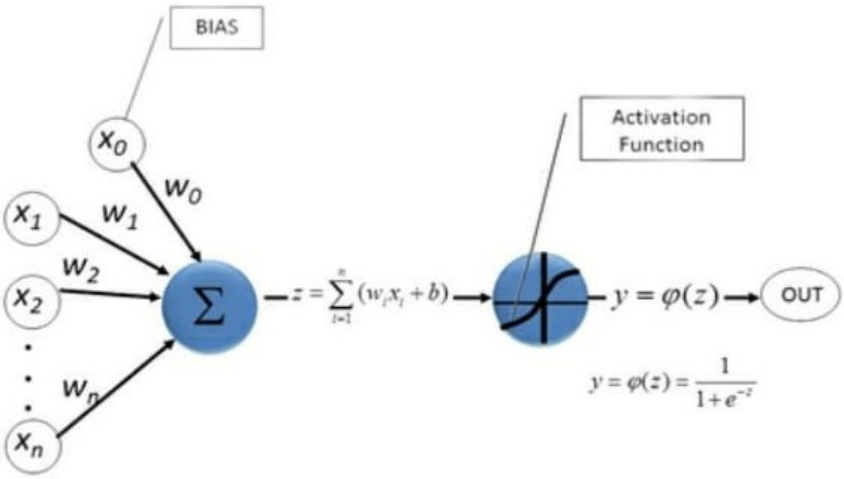


Figura 51 – Exemplo de uma MLP.

Fonte: (AWAD; KHANNA, 2015)

4 Resultados e Discussão

4.1 Hardware

4.1.1 Dispositivo Robótico Remoto (DRR)

O primeiro item desenvolvido foi o Dispositivo Robótico Remoto por meio da configuração do Raspberry Pi. A conexão entre o Raspberry PI e cada um dos módulos se dá pelos pinos GPIO conforme o diagrama apresentado na (Figura 53).

Figura 52 – Computador Raspberry Pi utilizado para controlar todo o sistema ERACI.

Fonte: Autoria Própria

Para que o *hardware* não fique exposto e vulnerável a qualquer avaria no que tange, principalmente, à desconexão acidental do cabos foi confeccionado uma estrutura plástica para envolve-la e deixar acessível (Figura 55): os botões, a estrutura pan-tilt, câmera, refletor infravermelho, *power Bank*, terminal HDMI e os terminais para alimentação do RPI e para carregamento da bateria selada de 12 V. Além disso, foram criados orifícios para permitir o acesso ao cartão de memória e terminais RJ45 e USBs. Estes orifícios são fechados com placas plásticas parafusadas na carenagem do robô.

4.1.2 Dispositivo Gerador e Gestor de Conhecimento - DGGC

O DGGC foi implementado em um computador Raspberry Pi modelo 3 B+. Este computador é idêntico ao apresentado na (Figura 52) e, é configurado de forma que ele

possa armazenar os dados necessários à geração do conhecimento e aqueles necessários à gestão desse conhecimento.

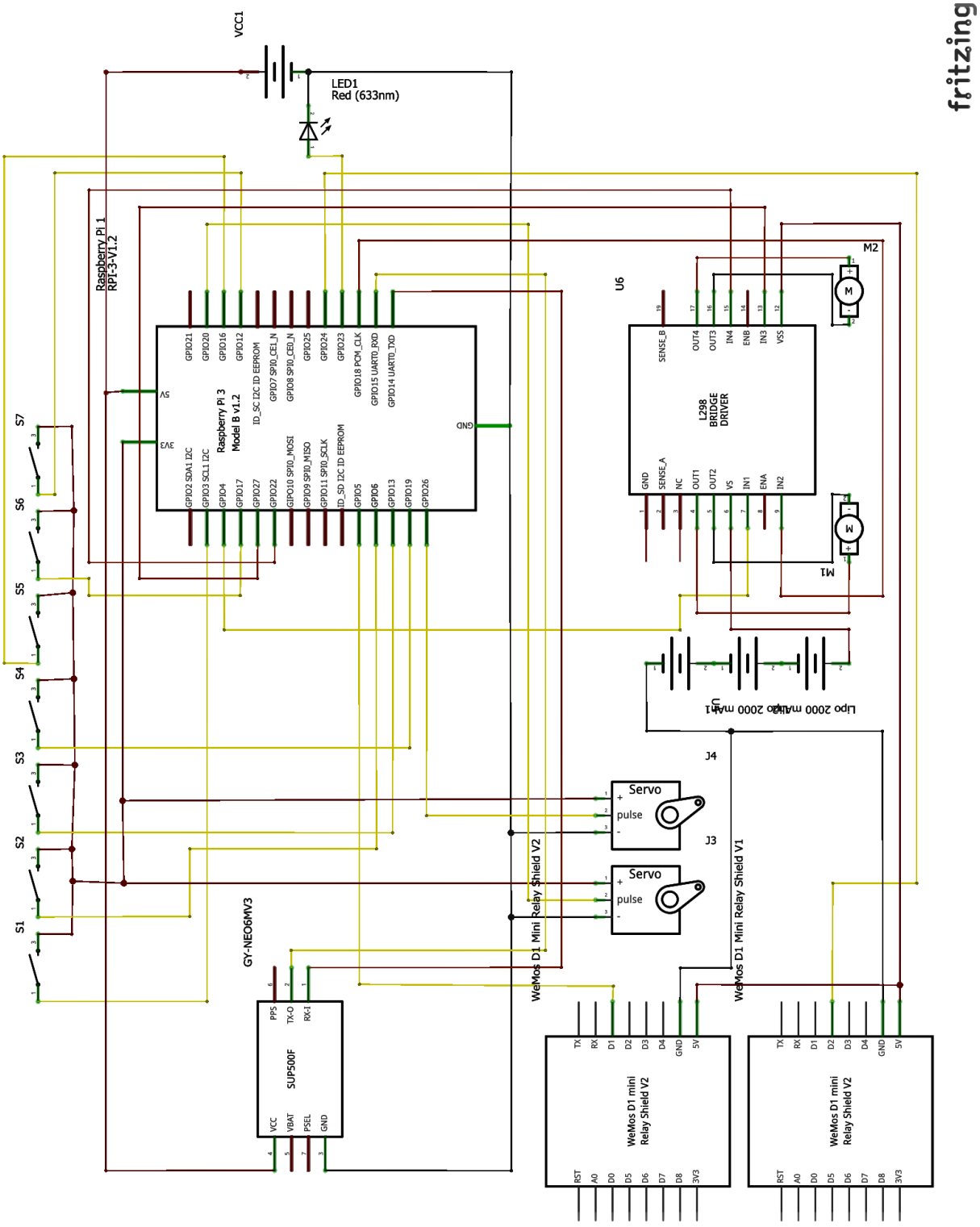


Figura 53 – Diagrama esquemático de ligações entre os módulos e o computador Raspberry Pi.

Fonte: Autoria Própria

Para que seja assegurada a execução do sistema sem interrupção por falta de energia ele utiliza um módulo HAT, responsável por gerir o fornecimento energético permitindo que ele tenha uma autonomia de funcionamento de até 3 h, mesmo que este não esteja

Figura 54 – Robô sem a proteção plastica com seus circuitos visíveis.

Autoria Própria

Figura 55 – Robô com a estrutura plástica para proteção.

Autoria Própria

conectado a uma tomada de energia elétrica. Esta estratégia foi utilizada para mitigar a possibilidade de corrupção dos dados armazenados no cartão de memória do RPI.

Com o objetivo de auxiliar na redução da temperatura em ambos os dispositivos (RPI e HAT) foi conectado um cooler aos pinos 5V e Ground do RPI. Devido a características físico-química dos componentes eletrônicos, esta atitude influencia diretamente no desempenho do dispositivo. A (Figura 56) apresenta o dispositivo e a (Figura 57) o seu diagrama esquemático.

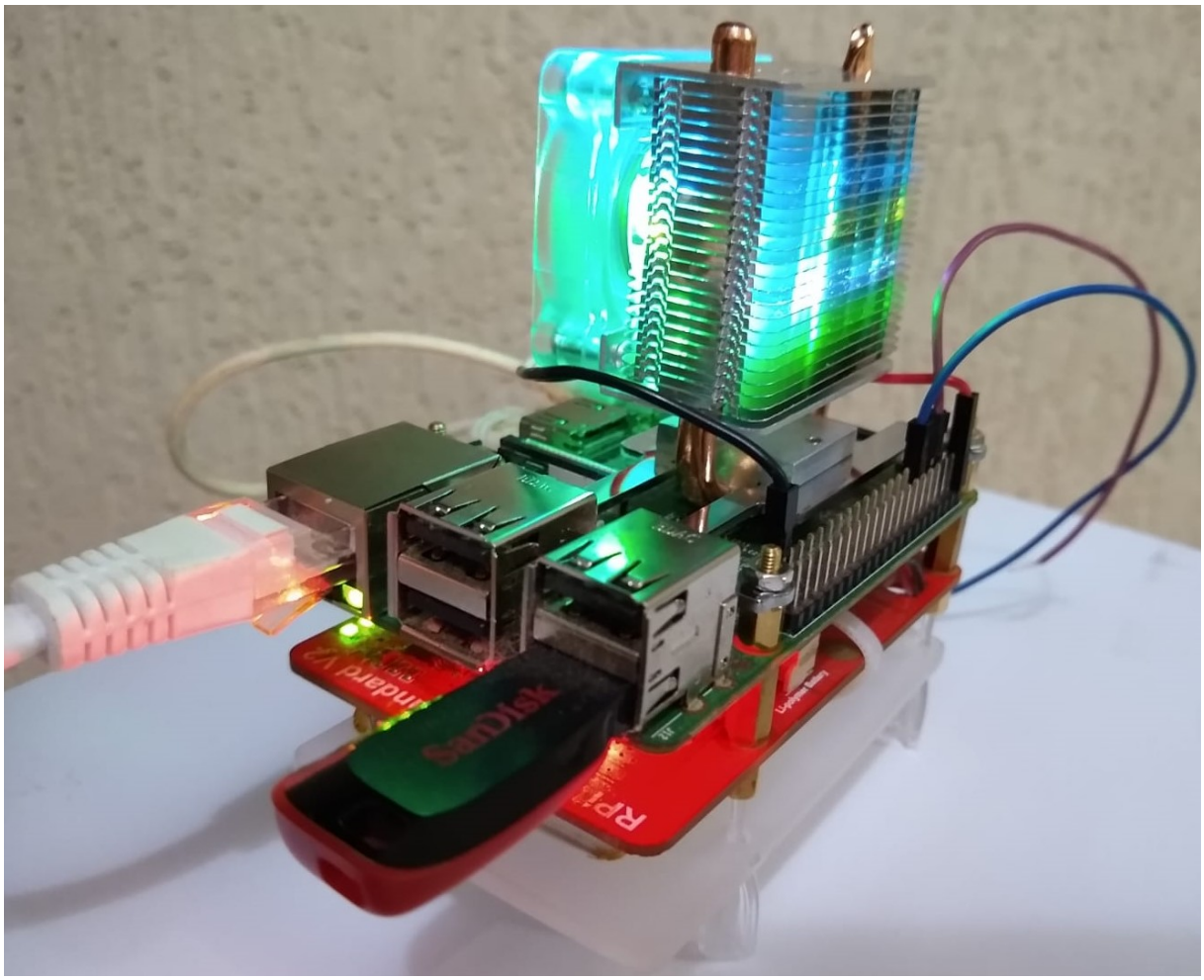


Figura 56 – Hardware do Dispositivo de Geração e Gestão de conhecimento implementado em computador Raspberry Pi modelo 3 B+ acoplado a um módulo hat de gestão de energia e a um cooler.

4.2 Software

4.2.1 Subsistema de Inicialização do Robô

Este subsistema é inicializado automaticamente como um serviço do Sistema Operacional (SO) *Raspbian* quando o DRR é ligado e, são carregados os módulos necessários

para que todo sistema funcione em harmonia. Basicamente ele é formado por apenas um *script Python* estruturado para esta tarefa. A forma como este arquivo é estruturado e, as dependências de cada componente são apresentados na (Figura 58) mostrada a seguir. Nela é possível observar que as setas indicam que o SO depende do arquivo *ini.py* e este por sua vez depende dos outros arquivos. Desta forma os subsistemas são basicamente formados pelos *scripts*:

- subsistema de controle: *controle_rpi.py*;
- subsistema de monitoramento: *controle_monitor_local.py*, *monitor.py* e *gps.py*;
- subsistema de arquivamento: *arquivista.py* e *robo_servico_arquivista.py*;
- subsistema de visão: *imagem.py*, *auto_snap.py*, *pdi.py* e *inteligencia.py*.
- Subsistema de comunicação: *client_socket.py*, *server_tcp.py* e *server_udp.py*.

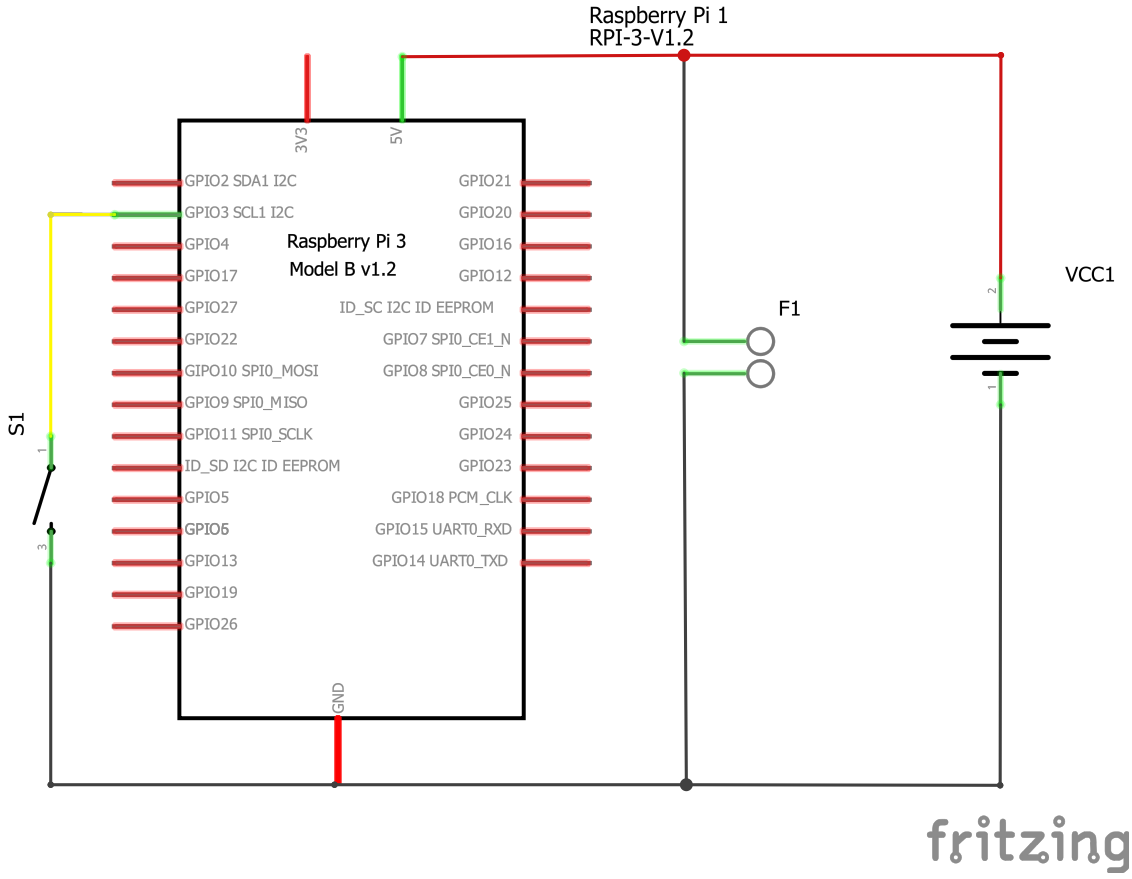


Figura 57 – Diagrama esquemático do hardware do dispositivo de geração e gestão de conhecimento.

Fonte: Autoria Própria

O subsistema de inicialização faz uso do arquivo *config.py* que tem seu respectivo objeto passado como parâmetro para os subsistemas de arquivamento, monitoramento e de comunicação. Este arquivo é instanciado por meio do método construtor que inicialmente verifica se o arquivo *config.txt* existe e, em caso negativo, solicita que o usuário preencha os dados necessários para a conexão com outros subsistemas remotos (IP ou URL do servidor, portas UDP e TCP que o DGGC está "ouvindo", portas UDP e TCP que o DRR "escuta", senha de segurança para validação de conexão em rede com clientes e servidores, intervalo de tempo em que o dispositivo capturará e armazenara as imagens visualizadas e o *threshould* utilizado para considerar que ocorreu mudança em uma imagem visualizada pelo robô. Após o preenchimento dos dados pelo usuário, estes são criptografados e armazenados no arquivo de configuração. Caso o arquivo exista, seu conteúdo é lido e os dados são carregados para a memória e descriptografados e, em seguida ficam acessíveis por métodos públicos.

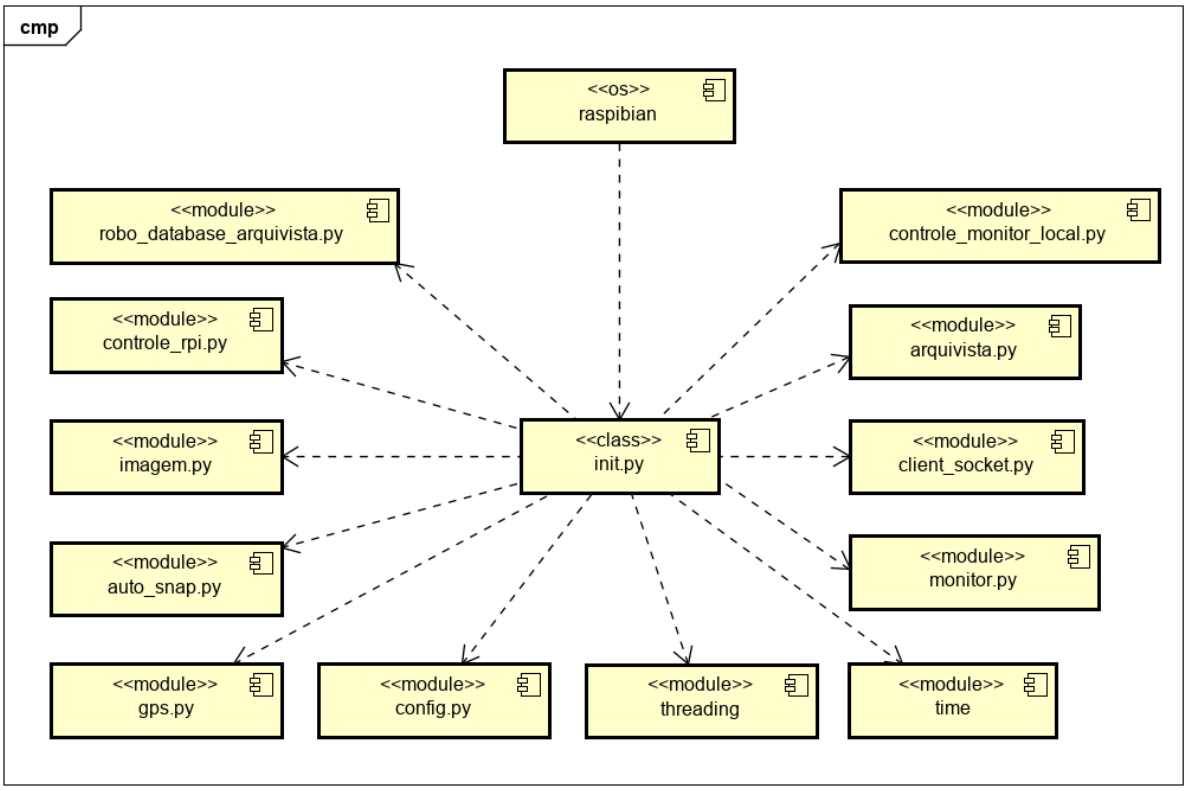


Figura 58 – Diagrama de componentes que mostra a interação entre os componentes da inicialização do subsistema robótico.

Fonte: Autoria Própria

4.2.2 Subsistema de Controle

Este subsistema tem seu carregamento no subsistema de inicialização e possui métodos para desempacotamento de instruções e a execução de ações diretamente nos

módulos *pan-tilt* (Figura 8) e ponte-H (Figura 7). Ele também é responsável por executar as tarefas de desligamento e reinicialização do sistema e, em conjunto com o subsistema de visão computacional, acionar a captura de imagens e vídeos.

Quando o sistema está conectado ao servidor ele pode receber comandos empacotados no formato:

<tipo de comando><comando>COMANDO</comando></tipo de comando>
: são desempacotados e seu conteúdo é verificado segundo as seguintes regras de escrita:

<tipo de comando> = <câmera> : comando recebido faz referência ao controle de posicionamento da câmera.

<tipo de comando> = <movimento> : movimentação ou parada do robô.

<tipo de comando> = <imagem> : comandos de captura de imagens no formato de fotografia (.png) ou de vídeo (.h264).

Quando é detectado que o comando recebido se refere à movimentação do robô, ele aciona o respectivo objeto que envia sinais lógicos para os terminais GPIO do Raspberry Pi e, assim, seja realizado o movimento ou a parada requisitada pelo subsistema supervisório. Quando é detectado que se trata de um comando para a captura de foto ou de vídeo, ele aciona os respectivos métodos no objeto Imagem. Já no caso de ser detectado que o comando recebido é para o reposicionamento da câmera Pi (Figura 9), ele aciona o objeto Olho de forma que sejam realizados os movimentos solicitados na estrutura pan/tilt (Figura 8). Todos os componentes envolvidos no funcionamento deste subsistema são apresentados na (Figura 59).

4.2.3 Subsistema de Monitoramento

Este subsistema é inicializado pelo subsistema de inicialização. No início de sua execução ele faz a leitura do arquivo de texto *monitoramentobateria.txt* que contém os dados referentes ao período anterior em que o dispositivo esteve em funcionamento como: tempo que o sistema esteve em funcionamento, o tempo de duração em que esteve funcionando sob alerta de baixa tensão, a forma que se deu o desligamento (abrupto ou não), o tempo máximo que ele funcionou sem interrupção abrupta e a data, hora e minuto que se deu o último armazenamento de dados. Essas informações são utilizadas para a inicialização das variáveis necessárias para a proteção do sistema no que se refere ao tempo de disponibilidade da bateria.

TMFSA; UTE; FD; TDA; DD/MM/AAAA HH:mm

TMF : Tempo máximo de funcionamento do sistema sem desligamento abrupto.

TSF : Tempo que o sistema permaneceu em funcionamento antes do último desligamento ou reinicialização.

FD : Forma que o sistema foi desligado ou reinicializado. Caso tenha sido de forma abrupta seu valor é True; caso tenha sido intencional recebe o valor false.

DD/MM/AAAA HH:mm : Dia, mês, ano, hora e minuto que ocorreu o último armazenamento de informação no arquivo.

Este *script python* disponibiliza em sua classe principal os métodos responsáveis por ligar e desligar o LED conectado no pino 16 do Raspberry Pi. Estes métodos são utilizados pelo subsistema de visão computacional para sinalizar de forma visual que o dispositivo está "vendo".

Após sua inicialização é inicializada uma *Thread* que a cada 60 s executa os métodos destinados a detecção de alerta de baixa tensão e verificação do tempo de disponibilidade de carga da bateria. Ambos os métodos atualizam o arquivo texto de monitoramento da bateria e verificam se a quantidade de tempo, com e sem alertas de baixa tensão, são indícios de que o sistema deve ser desligado ou reinicializado para sua autoproteção. Dentre

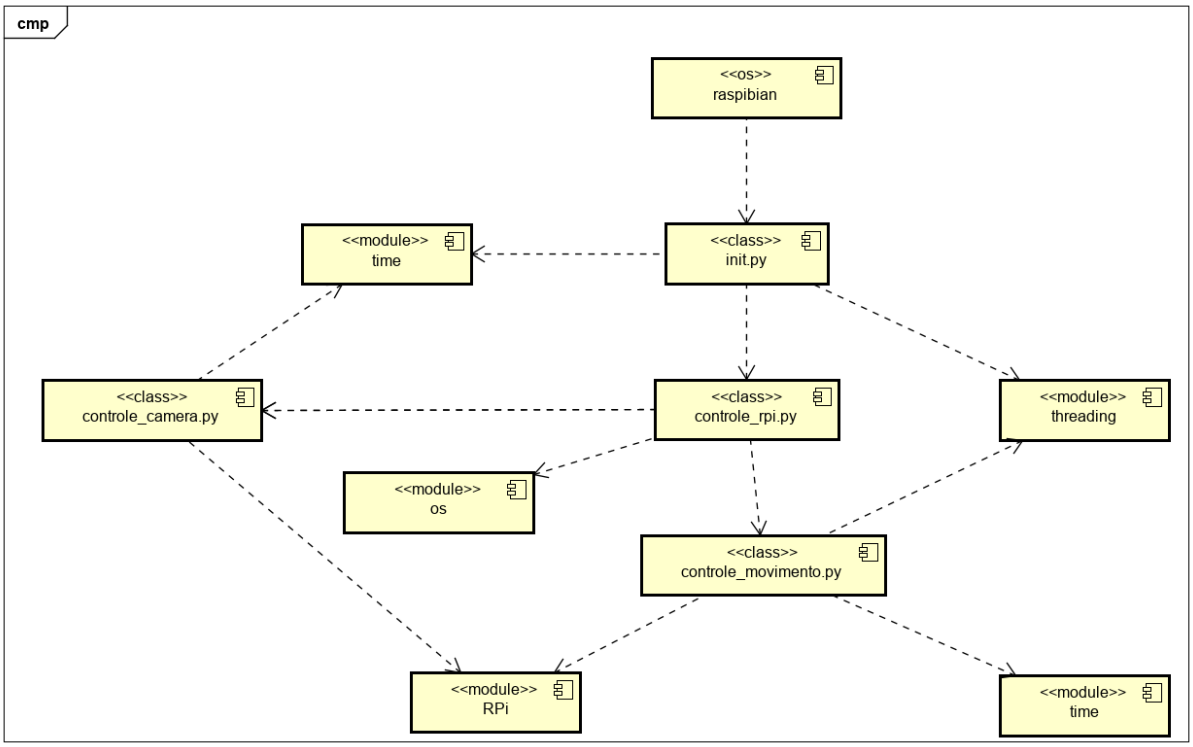


Figura 59 – Diagrama de componentes para o subsistema de controle.

Fonte: Autoria Própria

as verificações está o que se refere a estimação sobre o nível de carga da bateria. Caso seja verificado que seu nível é igual a zero e o tempo de alerta de baixa tensão é maior que zero, ele atualiza o arquivo de texto com os dados de tempo em que o dispositivo esteve em funcionamento e que o mesmo não foi desligado de forma abrupta (abrupto = *False*) e, em seguida realiza o desligamento do sistema. Caso este método identifique que o nível estimado de carga da bateria é igual a zero, porém, o tempo de alerta de baixa tensão também é igual a zero, o tempo total de disponibilidade de carga é acrescido em mais um $(+1)$ e, esta informação também é atualizada no arquivo de texto correspondente. Outra situação ocorre quanto a verificação de baixa tensão que, recebe do sistema operacional informações referentes a qualidade da energia consumida pelo dispositivo. Quando ele detecta uma qualidade ruim de energia (tensão abaixo de $4,5\ V$) e o momento temporal em que ocorreu a primeira detecção de qualidade ruim no fornecimento de energia é igual a zero, este momento é atualizado como momento inicial de ocorrência de falha. Caso seja detectado que a qualidade no fornecimento de energia está normalizada (tensão acima de $4,5\ V$) a variável que guarda o primeiro momento de detecção de baixa qualidade recebe o valor zero. Isto é feito desta forma pois existe a probabilidade de que a qualidade da energia tenha sido restabelecida ou a falha ocorrida anteriormente foi devido a algum pico momentâneo de consumo do dispositivo. Quando este método é chamado e a variável de tempo inicial de detecção de anomalia é maior que zero, o tempo de ocorrência de anomalia energética é atualizado. Caso seja constatado que o tempo de duração da anomalia é igual ou superior ao tempo estimado ele desliga o dispositivo para proteger o sistema de arquivos.

Para possibilitar a realização das ações de desligar, reinicializar, posicionar a pan/tilt e, reinicializar os valores para os dados de tempo para desligamento no próprio dispositivo, foi utilizado os botões *push button* e, para detectar seu pressionamento este subsistema faz uso do método *add_event_detect* do módulo GPIO que foi configurado para detectar a borda de subida com tempo de espera de 200 ms (RASPBERRY PI FOUNDATION, 2020).

Após a inicialização deste subsistema, é realizado, também, de forma periódica, a verificação do uso da CPU (*Central Process Unit* – Unidade Central de Processamento) e, utilizando os dados do módulo GPS (Figura 11)obtém os valores de latitude e longitude em que o dispositivo se encontra. Para que seja possível realizar todas estas tarefas, estes objetos utilizam a biblioteca *psutil* (Figura 60). A biblioteca *psutil* é utilizada para recuperar as informações sobre os processos em execução e utilização do sistema em *Python*. Dentre as informações fornecidas por esta biblioteca encontram-se dados de: uso da CPU, uso de memória, uso dos discos, rede e sensores. Já o módulo GPS, é um circuito eletrônico que recebe por meio de uma antena os dados relativos ao posicionamento geográfico que o dispositivo se encontra e, os envia ciclicamente a uma taxa de 5 *Hz* com a precisão horizontal de até $2,5\ m$, por meio serial, ao *Raspberry Pi*.

4.2.4 Subsistema de Arquivamento

Este subsistema (Figura 61), é iniciado pelo subsistema de inicialização. A princípio ele gera uma tupla contendo a data, hora, latitude, longitude e o nome do arquivo correspondente a imagem que foi capturada e armazenada no dispositivo robótico. Depois disso ele inicia uma *Thread* que executa seu conteúdo em intervalos de 60 s. A cada execução é verificado se há novos arquivos de imagem e vídeos armazenados. Quando existe uma resposta positiva sobre a existência de novos arquivos, seus nomes são armazenados em uma lista que servirá de banco de dados utilizado para envia-los ao servidor. Depois disso ele tenta realizar uma conexão com o subsistema de serviço web no DGGC (Subseção 4.2.8), e assim que a conexão é efetivada, envia estes arquivos para ele. Por fim, é realizada a remoção dor arquivos enviados do DRR..

Após finalizar o envio dos dados, no formato *json*, por meio do protocolo HTTP para o serviço que é disponibilizado no DGGC a *thread* entra no modo de espera, até que novos arquivos sejam detectados por este subsistema.

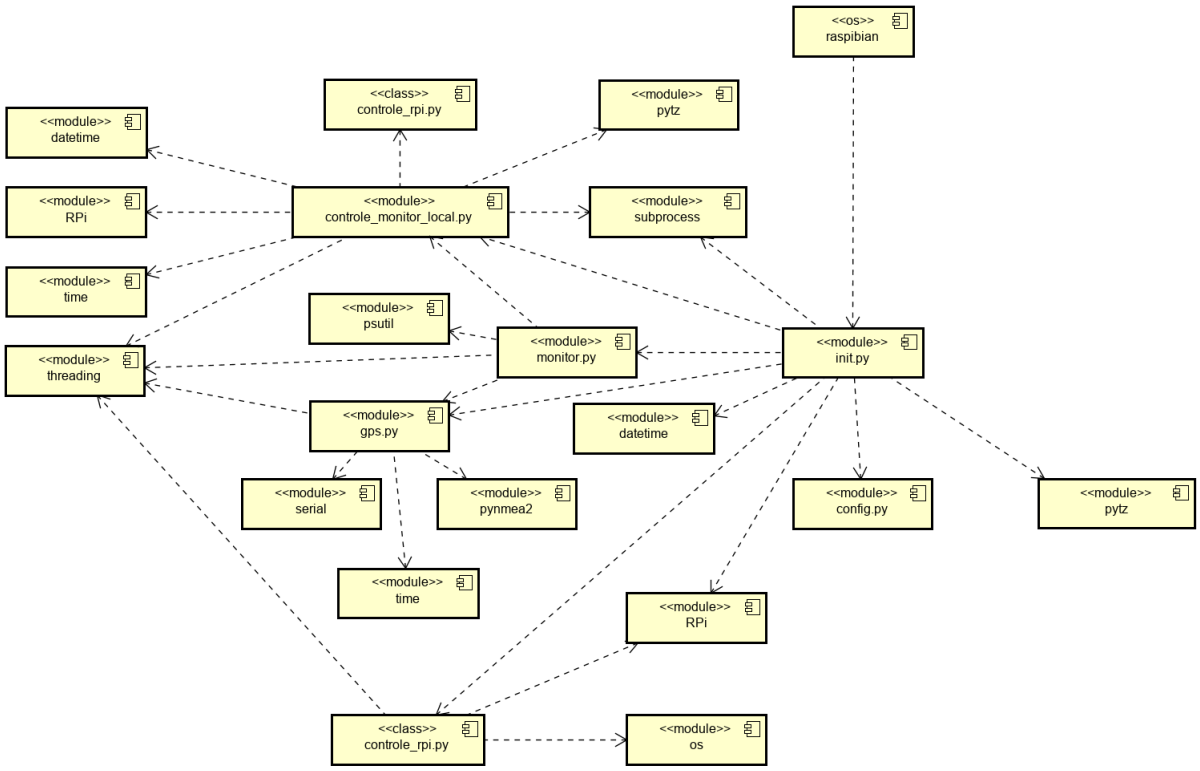


Figura 60 – Diagrama de componentes do Subsistema de Autocontrole e monitoramento.

Fonte: Autoria Própria

4.2.5 Subsistema de Comunicação

Quando este subsistema é iniciado, ele entra em um laço de repetição que verifica se existe uma conexão disponível com o servidor, utilizando dados carregados pelo módulo *config.py*. Quando a conexão com o servidor ocorre, este subsistema envia uma senha local para o servidor e aguarda o recebimento da senha do servidor para validação. Quando a senha proveniente do servidor é validada, são criados objetos responsáveis pelo seu monitoramento, multiplexação dos comandos recebidos do subsistema supervisório e, envio de imagens em tempo real, utilizando recursos de computação concorrente por meio de *threads*. Para isto o arquivo *cliente_socket.py* faz uso dos arquivos *robo_socket_monitor.py*, *robo_socket_controller.py* e *cliente_socket_imagem_udp.py*, respectivamente. Quando a senha recebida pelo robô não é validada, a conexão é desfeita e são realizadas novas tentativas de conexão, até que os requisitos para validação da conexão sejam alcançados.

Este subsistema também tem o comportamento de servidor TCP/IP e UDP/IP, que esperam por uma conexão de clientes por meio de *sockets*, operando pelas portas carregadas pelo modulo *config.py*. Sua operação conta de um *thread* que aguarda por uma conexão proveniente de um cliente e, ao receber esta solicitação, recebe também um valor de chave a ser validado pela paridade com a chave carregada pelo módulo *config.py*. Após esta validação ocorre a transmissão das imagens e dos dados de monitoramento para os dispositivos solicitantes, bem como, a habilitação do controle do DRR por parte destes, bastando para isso que eles enviem os comandos codificados no formato exigido pelo sistema de controle.

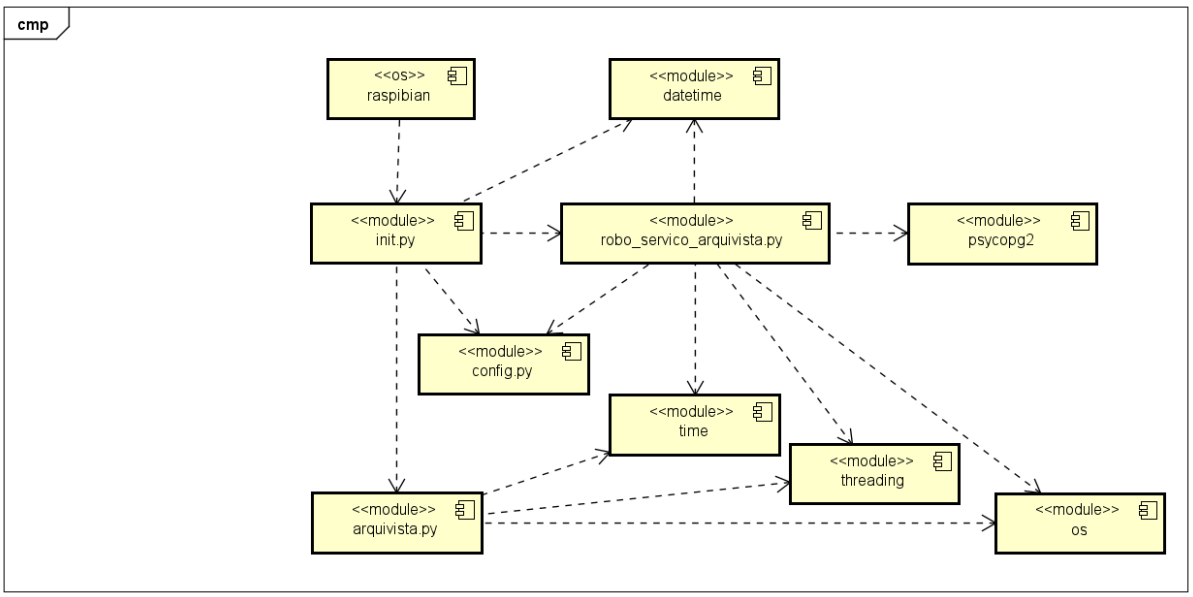


Figura 61 – Diagrama de componentes para o Subsistema de Arquivamento.

Fonte: Autoria Própria

O arquivo *config.py* é responsável por gerir as informações utilizadas pelo ERACI. Dentre estas informações estão o *threshould*, que é utilizado para detecção de movimento, intervalo de captura e *Uniform Resource Locator* (URL) do servidor, conforme as necessidades vigentes. Toda esta estrutura de software é mostrada na (Figura 62).

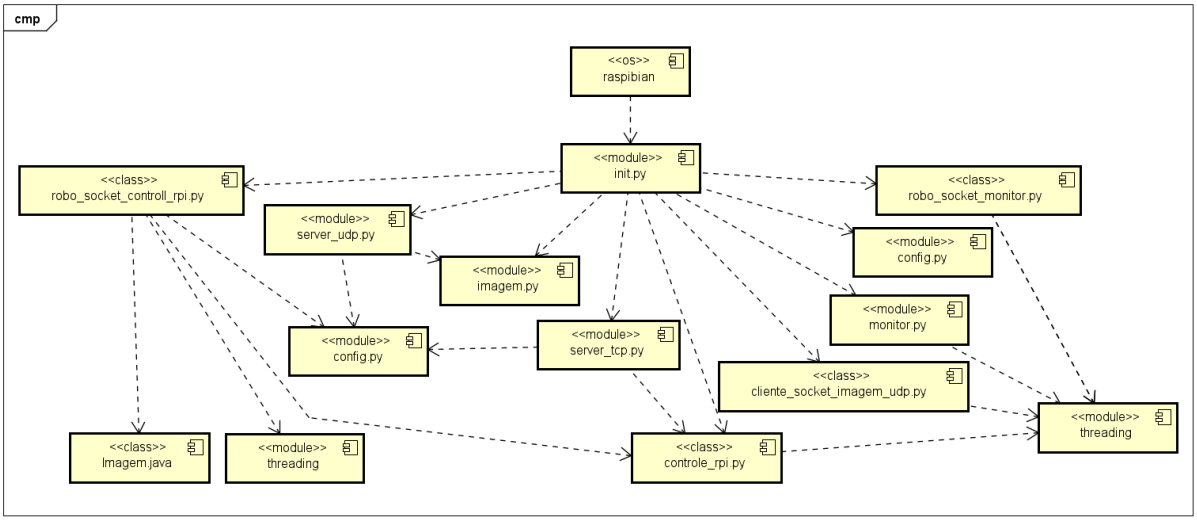


Figura 62 – Diagrama de componentes que mostra os componentes envolvidos no subsistema de conectividade com o servidor.

Fonte: Autoria Própria

4.2.6 Subsistema de Inicialização do DGGC

Este subsistema é composto basicamente pelo *script python* ___*init*___*.py* e é iniciado pelo SO Raspbian (HENDRIX, 2016). Ele é estruturado dentro de um computador RPI dedicado para o armazenamento dos dados de imagens, características, usuários, classes e classificações por meio do subsistema Gerenciador de Banco de Dados, bem como, para a execução dos subsistemas de visão computacional, gerador de inteligência artificial e de serviços web. O *script python* consiste apenas na inicialização do Subsistema de Visão Computacional e de Serviço web. A (Figura 63) apresenta esta estrutura.

4.2.7 Subsistema Supervisório *Desktop*

Este subsistema foi criado para permitir que o usuário tenha acesso ao DRR e, assim, acompanhe as imagens provenientes de sua câmera (Figura 8), os dados de uso da CPU e as coordenadas de posicionamento global (Figura 11), além de possibilitar o movimento do DRR (Figuras 6 e 7) e sua pan/tilt (Figura 8). Ele é construído e codificado para que possa ser executado em qualquer sistema operacional que possua uma *Java Virtual Machine* (JVM) instalada (Subseção 3.5.2), dando a ele a característica multiplataforma.

Este subsistema é inicializado automaticamente pelo sistema operacional. Quando isto acontece ele "ouve" pela porta 5000. Em seguida é criado um objeto que recebe um objeto *ServerSocket* como parâmetro (COSTA, 2008a; COSTA, 2008b).

O objeto responsável por registrar os *Socket Servers* é confeccionado para possibilitar uma execução de forma concorrente por meio de *thread*, que aguarda de forma cíclica por uma nova requisição de conexão de um cliente que, neste caso, é um DRR (GOETZ et al., 2008). Ao receber a solicitação de conexão, o subsistema envia uma senha para o cliente e aguarda o envio da senha do cliente como resposta. Após isso é verificado se esta conexão se trada de um DRR já adicionado na lista de robôs conectados e, em caso afirmativo, o DRR já existente, tem sua instancia destruída. Após estas verificações um novo objeto que representa o DRR é criado, executado como um *thread* e adicionado à lista de robôs. Esta nova *thread* recebe de forma cíclica as mensagens enviadas pelo robô, as desempacota e as mostra na tela do computador por meio da interface gráfica (Figura 65).

Este subsistema também faz uso do objeto *DatagramSocket* que "escuta" pela porta 5001 (COSTA, 2008a; COSTA, 2008b). Estes dois objetos são passados como parâmetro para o objeto que faz a representação da imagem, criado logo em seguida por meio de recursos de concorrência (GOETZ et al., 2008). Este *thread* "ouve" ciclicamente a porta 5001 recebendo os datagramas enviados pelo robô. Estes datagramas são desempacotados e convertidos em um objeto *ImageIcon* que é mostrado na interface gráfica do usuário por meio do objeto *JLabel* (LOY et al., 2002).

O Subsistema Supervisório *Desktop* apresenta uma interface gráfica que permite ao usuário, em tempo real:

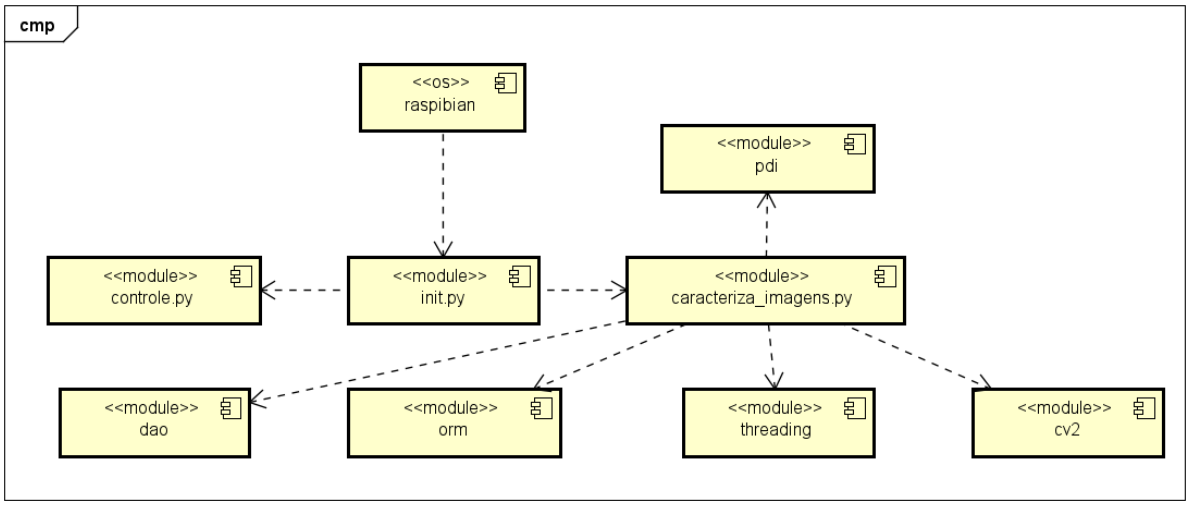


Figura 63 – Diagrama de componentes que mostra a estrutura do subsistema de inicialização do DGGC.

Fonte: Autoria Própria

- acompanhar as imagens transmitidas pelo robô;
- visualizar as suas coordenadas geográficas;
- saber o nível de utilização da CPU;
- movimentar o robô;
- posicionar a câmera;
- capturar e armazenar as imagens nas estações de trabalho onde o supervisório está instalado ou no próprio robô.

Todas as tarefas relacionadas ao controle do robô, também podem ser realizadas por intermédio de um *Joystick* conectado pela USB ou por *Bluetooth*.

A (Figura 64) mostra os componentes envolvidos no funcionamento do subsistema supervisório e a (Figura 65) apresenta a tela do sistema supervisório.

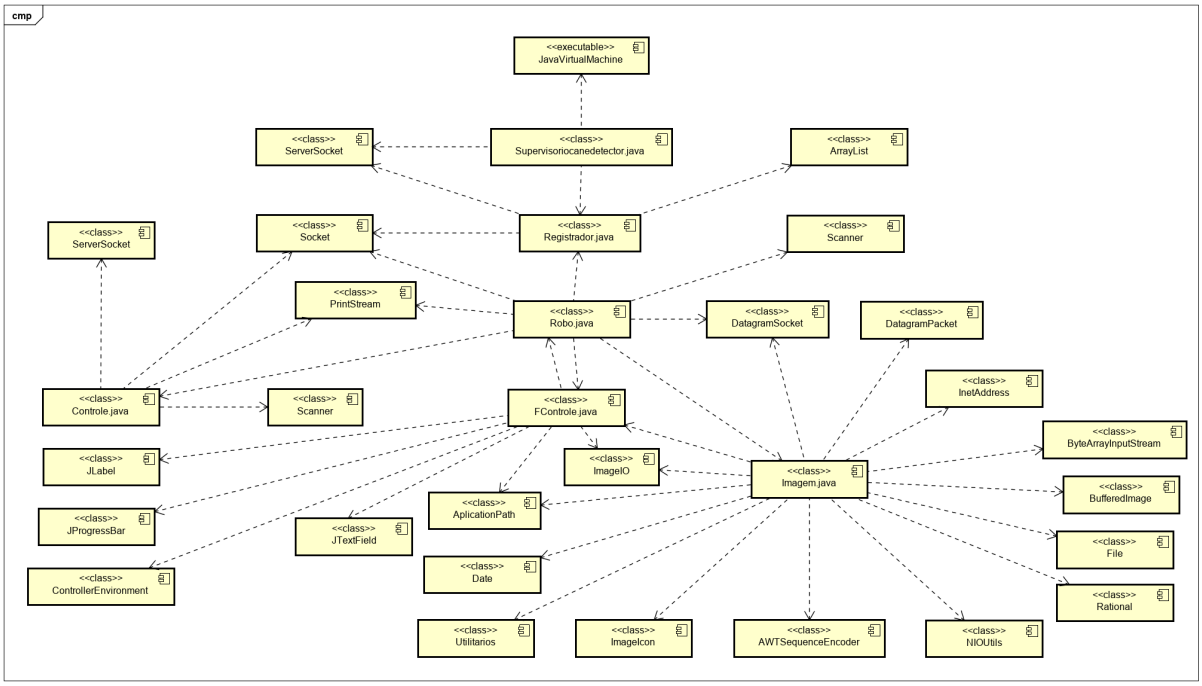


Figura 64 – Diagrama de componentes do subsistema supervisório.

Fonte: Autoria Própria

4.2.8 Subsistema de Serviço web

Este subsistema é iniciado pelo subsistema de inicialização com a instanciação do serviço *app*. Ele disponibiliza serviços pelos métodos HTTP: GET, POST, PUT e DELETE.

Nele foi utilizado a arquitetura *Representational State Transfer* (REST) por esta possuir características compatíveis com as demandas do ERACI, como: permitir a separação entre o servidor que oferece o serviço e o cliente que o consome, não armazenar informações de um cliente em uma solicitação para usa-la em outra, permitir armazenamento em cache, ser padronizada em camadas, ter interface uniforme e permitir execução sobre demanda, caso necessário (SAUDATE, 2014; FLASK, 2020).

Nesta arquitetura os recursos são representados por *Uniform Resource Identifier* (URI) que são utilizados pelo cliente para enviar solicitações ao servidor, utilizando os métodos definidos pelo protocolo HTTP (SAUDATE, 2014; FLASK, 2020).

Os serviços acessíveis por estes métodos são mostrados na Tabela 5, juntamente com suas respectivas URI de chamada e sua ação correspondente.

O recurso acessado pelo método GET do http com a URI */robot/api/v1.0/imagem/* primeiramente verifica se o usuário requisitante já possui uma imagem a ser trabalhada e, em caso positivo essa imagem é carregada do BD, sofre as alterações realizadas pelo módulo PID e de IA e, em seguida ela a envia para o requisitante. Caso não exista nenhuma imagem de trabalho, o método pega uma imagem aleatória que esteja dentro do range de permissões do usuário requisitante, adiciona seu ID em um dicionário *python* e em seguida realiza as transformações necessárias e, a envia para ele.

Um aspecto relevante a ser mencionado aqui é que todas as requisições exigem a autenticação com senha e, ao final da execução dos métodos é retornado para o requisitante uma lista no formato *json* com os registros do Banco de Dados requisitados.

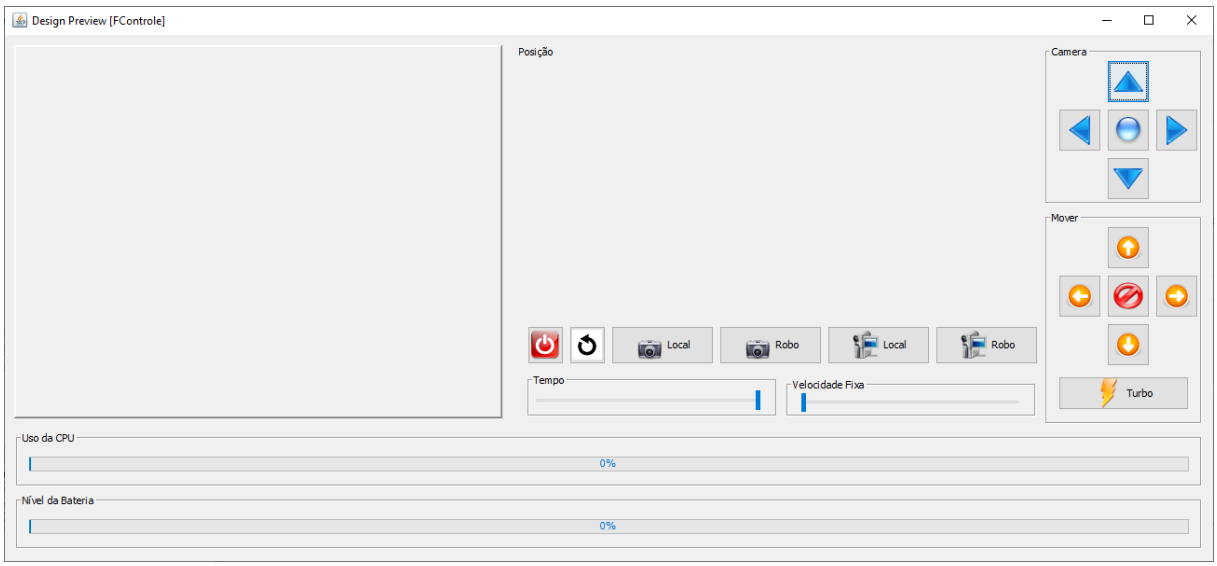


Figura 65 – Interface gráfica do sistema supervisório.

Fonte: Autoria Própria

A estrutura dos arquivos que compõe este subsistema é apresentada na Figura 66.

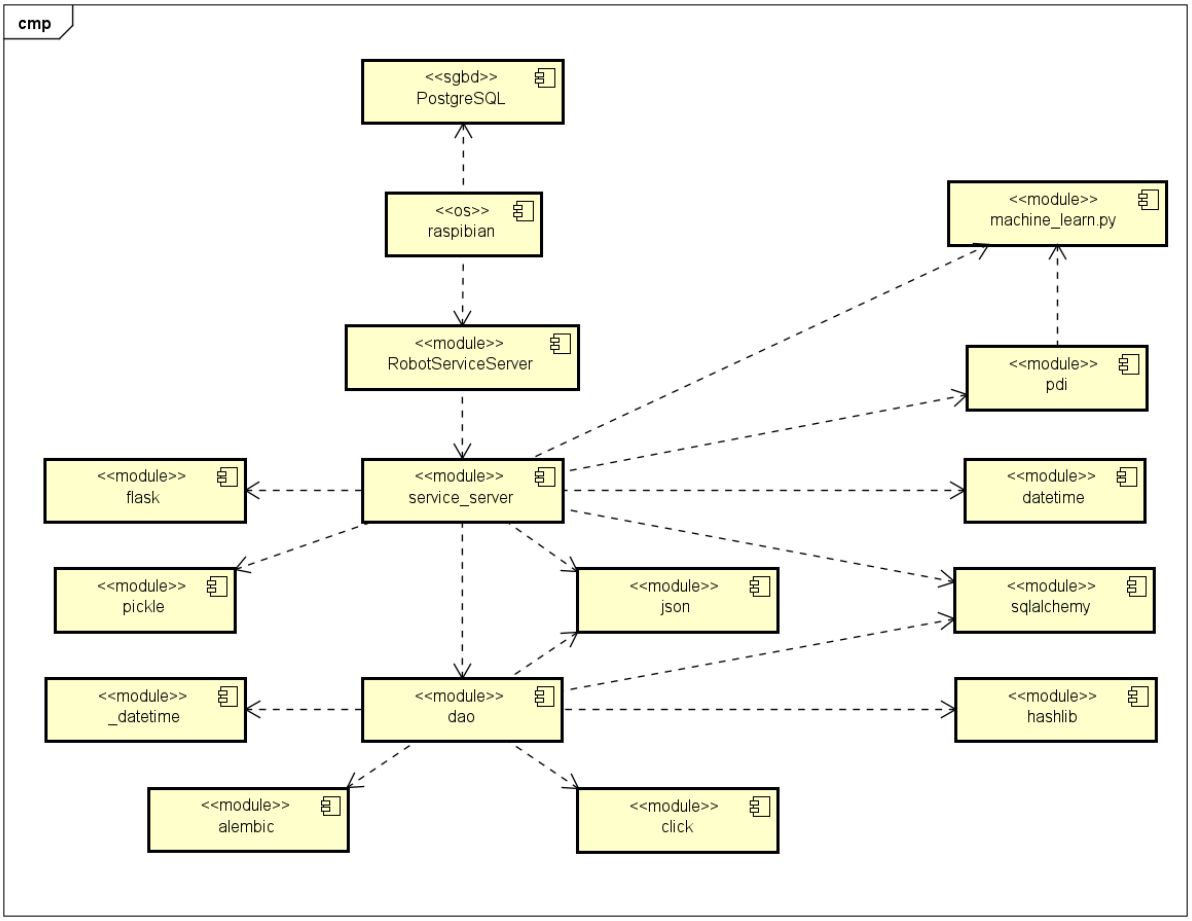


Figura 66 – Diagrama que apresenta os componentes de software do subsistema de serviço web e como eles interagem entre si.

Fonte: Autoria Própria

4.2.9 Subsistema de Gerenciamento do Banco de Dados

Quando este RPI é ligado, ele inicia o SGBD PostgreSQL ®que está configurado para escutar conexões externas pela porta 5432. Também é iniciado o módulo *RobotServiceServer* que disponibiliza os serviços necessários para a gestão de usuários, classes, classificações e imagens pelo subsistema mobile por meio da porta 5005. Para isso ele utiliza um módulo *Data Access Object* – DAO. Este módulo DAO utiliza o *sqlalquemy* para realizar o *Object Relational Mapping* – (ORM) que é a basicamente a conversão de registro de banco de dados em objeto de um sistema orientado a objetos e vice e versa (ELMASRI; NAVATE, 2018; SIAHAAN; SIANIPAR, 2019; SQLALCHEMY, 2020).

Isto foi necessário para a realização do arquivamento das imagens capturadas pelo robô, para posterior classificação e geração dos modelos de IA. Assim foi configurado no DGGC um SGBD – Sistema Gerenciador de Banco de Dados Postgree, por este ser

gratuito, *Open Source* e ter se destacado bem no mercado de bancos de dados gratuitos (OBE; HSU, 2012); características estas necessárias para um sistema que deverá funcionar sobre um hardware limitado como é o caso do RPI (HENDRIX, 2016).

O BD foi organizado de forma a conter as tabelas para a persistência de: usuários, imagens, classes, classificações, aspectos, dimensionais e inerciais, sendo os três últimos, características extraídas das imagens. Estas tabelas estão relacionadas de forma restritiva para garantir a segurança e a integridade dos dados. As tabelas utilizadas, bem como seus relacionamentos, são apresentados na Figura 67 que mostra seu respectivo diagrama de classes.

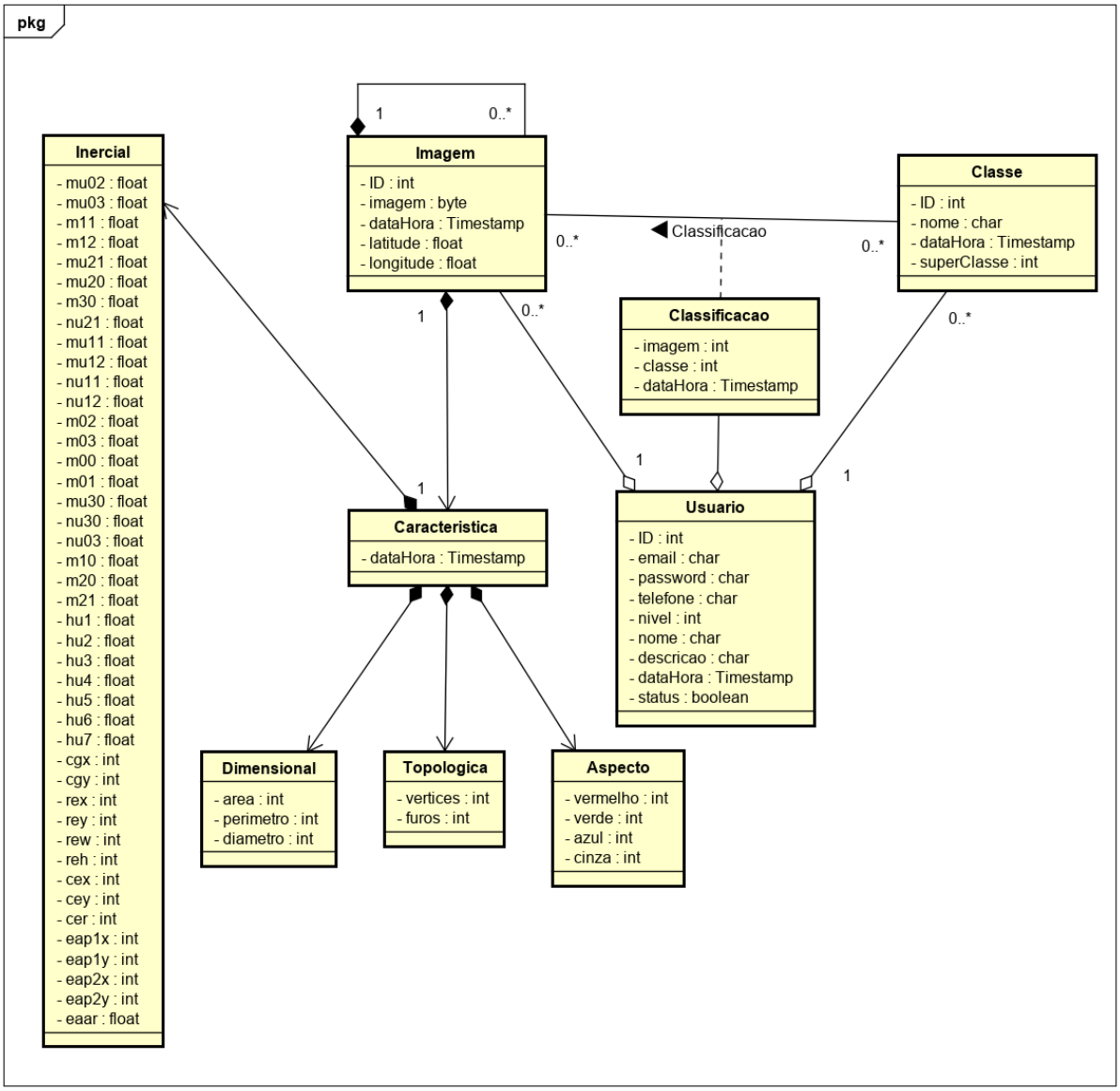


Figura 67 – Diagrama de Classes para as tabelas envolvidas no armazenamento das imagens capturadas e suas respectivas possibilidades de classificação.

Fonte: Autoria Própria

Para garantir que o banco de dados possa ser restaurado e assim garantir a segurança e integridade dos dados, foi também criado um *script shell* para que seja realizado o *backup* dos dados armazenados no esquema *robot* periodicamente. Da mesma forma foi criado um *script shell* para que seja realizado o expurgo destes *backups* para mitigar a probabilidade de sobrecarga da capacidade do disco destinado a este fim. As partes que compõe este subsistema são apresentadas na Figura 68.

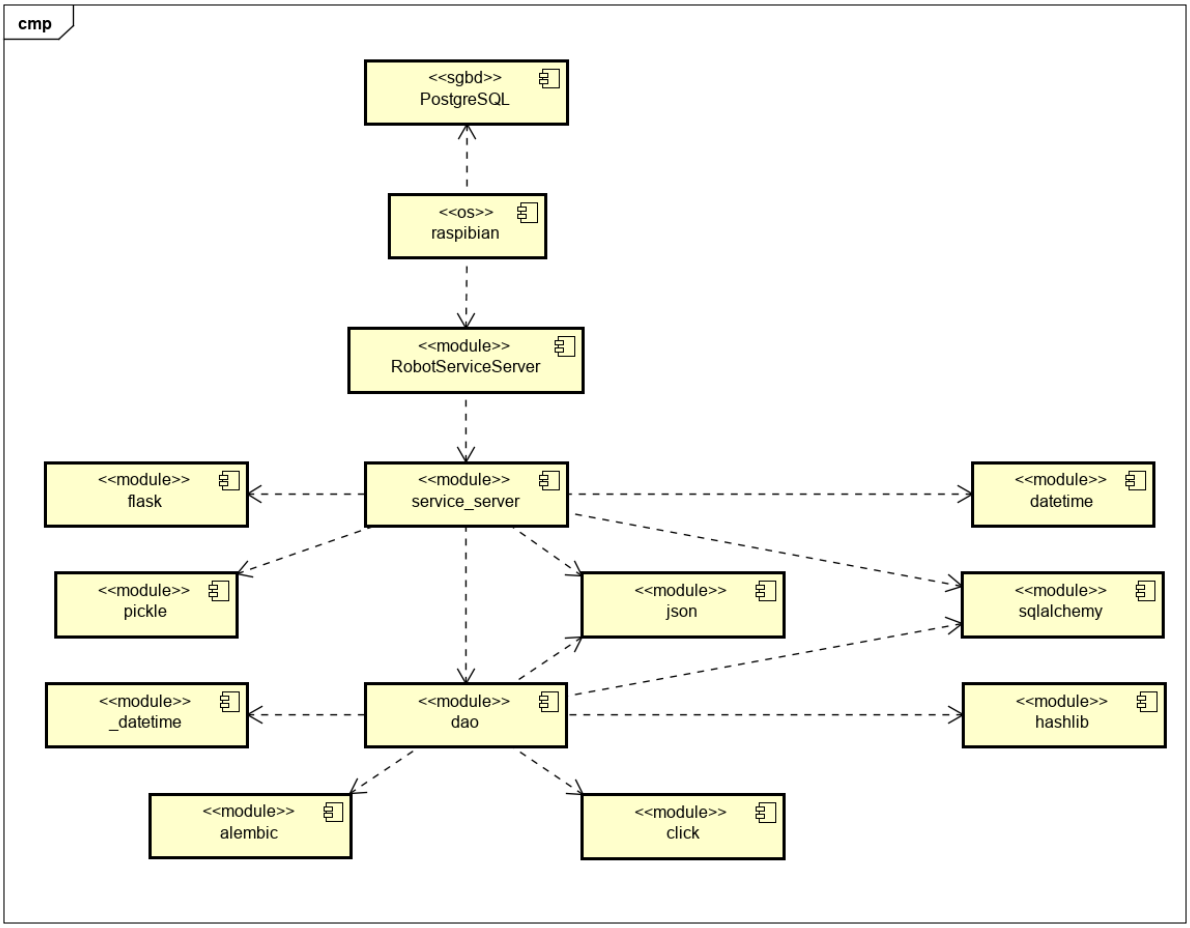


Figura 68 – Diagrama de componentes do subsistema gerenciador de banco de dados.

Fonte: Autoria Própria

4.2.10 Subsistema de Gestão de Dados

Para este projeto, também foi desenvolvido um aplicativo para ser executado nativamente em dispositivos móveis. Este tem como função principal a pré-classificação de imagens que serão posteriormente utilizadas para o treinamento do sistema inteligente.

Para minimizar riscos referente a utilização indevida dos dados de imagens, foram criados módulo para criação e gestão de usuários e classes.

Este aplicativo recebe uma imagem assinalada do subsistema de serviços web e, a partir daí, permite que o usuário exclua a imagem ou realize a classificação da mesma. As figuras 69 e 70 apresentam as telas do aplicativo.

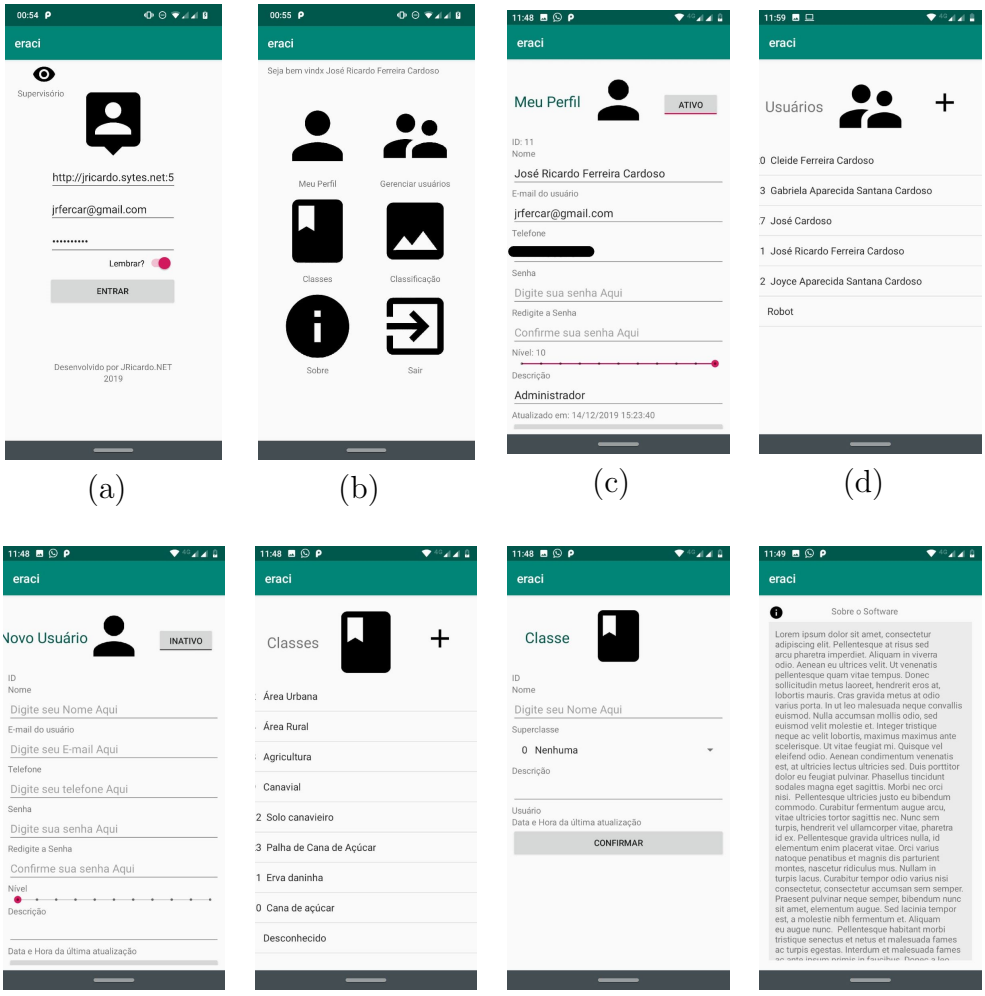


Figura 69 – Telas do aplicativo para gestão de dados do ERACI pelo Dispositivo Móvel. (a) Tela para autenticação; (b) Tela de Menu; (c)Tela para gestão do perfil; (d) Tela para criação e gestão de perfis; (e) Tela para a criação de novos usuários; (f)Tela para criação e gestão de classes; (g) Tela para criação ou atualização de classes; (h) Tela informativa sobre o aplicativo móvel.

Fonte: Autoria Própria

Para acessar o aplicativo denominado “eraci"o usuário precisa primeiramente preencher os campos de URL necessário para a conexão com o servidor de serviços, o número do telefone ou o *e-mail* cadastrado para o usuário e a sua respectiva senha. Se desejar, o usuário poderá chavear o botão lembrar para que esses dados permaneçam gravado no celular e, assim não ser mais necessário preenche-los no dispositivo conforme a preferência do usuário.

Quando o usuário aciona o botão entrar, uma tela de menu é mostrada para ele; se

o seu nível de permissão for igual a 10 ele poderá gerir usuários e classes pelos respectivos botões na tela. Isto foi criado para que seja possível a adição, atualização e exclusão de usuários e classes.

Para que um determinado usuário logado realize alterações em seu perfil, foi criado uma seção "meu perfil" para que ele faça as alterações que julgar necessário em seus dados, porém, não é permitido que ele altere para cima o nível de seu perfil que pode estar dentro dos valores inteiros de 0 até 10.

Referente às classes, apenas usuários com permissão igual a 10, como dito anteriormente, podem cria-las, alterá-las ou excluí-las. Cada classe é estruturada de forma que ela seja uma superclasse ou subclasse. Isto foi feito para que o sistema possa classificar uma imagem, primeiramente levando em conta seu todo como por exemplo: área urbana, área rural e canavial (Tabela 7). Desta forma é possível que com base na classificação da imagem, considerando seu todo, seja possível melhorar a acurácia na classificação da imagem apontada pelo algoritmo de processamento de imagens.

A tela para a classificação das imagens foi confeccionada conforme a ideia citada no parágrafo anterior. Pois, ela exige que primeiramente o usuário informe a qual superclasse a imagem pertence e, posteriormente, o usuário escolha qual a classe que o trecho apontado pelo algoritmo pertence. Após ele classificar esses dois primeiros itens nomeados como "Super classe" e "Assinalada", é disponibilizado para o usuário uma lista com os possíveis itens, que também compõe a imagem apresentada. Isto foi feito para compor o banco de dados com informações relevantes sobre a imagem e, assim, o desenvolvedor possa realizar testes ou criar novos algoritmos que possibilitam a classificação de imagens.

Outra informação importante sobre a tela de classificação de imagens, é a possibilidade de que o usuário visualize a data, hora e o local, com precisão de até 5 m, onde a imagem foi capturada pelo DRR.

No momento em que a imagem é carregada, também é realizado uma pré-classificação da mesma, permitindo assim que o usuário acompanhe a evolução do algoritmo de aprendizagem naquele momento.

4.2.11 Subsistema Superviório Local

Este subsistema funciona de forma independente do subsistema de gestão de dados (Subseção 4.2.10), embora possa ser acessado pelo mesmo aplicativo "eraci".

Ele foi confeccionado para permitir que o usuário possa visualizar o que o DRR está "vendo" e classificando a superclasse da imagem e a sua respectiva parte apontada pelo algoritmo do PID, levando em conta suas características.

Para acessá-lo, o usuário deve clicar sobre o botão supervisor na tela inicial do

respectivo aplicativo. Após isto será mostrada uma tela com os dispositivos conectados. Assim, para acessar a tela de monitoramento e controle do DRR basta que seja pressionado um dos itens apresentados na lista. Na primeira vez que o usuário requerer o acesso ao DRR é aberta uma tela pedindo que ele preencha a chave. Caso o usuário deseje remover a chave de acesso cadastrada, basta que ele realize o pressionamento longo sobre o item da lista. Depois disso, ele poderá atualizar a chave com um novo pressionamento curto. A Figura 71 apresenta as telas deste subsistema.

A tela de supervisão além de mostrar o que o DRR está "vendo", disponibiliza botões na tela, para que o usuário possa posicionar a câmera conforme a necessidade do momento.

4.3 Aquisição das Imagens

As imagens são as fontes para a geração de modelos de *Machine Learning* utilizados para realizar o reconhecimento de padrões e classificação das imagens. Assim, a qualidade deste modelo está diretamente relacionada a quantidade balanceada (CHAWLA; JAPKOWICZ; KOTCZ, 2004) e variabilidade destas imagens, que devem conter o objeto a ser classificado, chamados aqui de imagem positiva e também imagens que não contém o objeto a ser classificado, chamado de imagem negativa (SILVEIRA; BULLOCK, 2017; PAJANKAR, 2015; BARELLI, 2018). Por esta razão, foi primeiramente definido os locais onde serão capturadas as imagens correspondentes a todos os estágios de desenvolvimento da cultura, onde, primeiramente, foi buscada e conseguida uma autorização para a captura destas imagens na fazenda do Instituto Federal de São Paulo, campus Barretos e, também, foi firmada uma parceria para este fim com a administração da fazenda Santa Adelaide, também localizada no município de Barretos.

Para realizar a captura das imagens nas localidades apresentadas na (Figura 3) o DRR foi acoplado a um veículo automotor Volkswagen Fox 2008, de forma que ele fique prensado na porta de seu lado esquerdo (lado do motorista) em sua janela lateral. Assim, as partes que correspondem ao refletor infravermelho e a câmera ficam do lado de fora do veículo e o restante, como botões e o LED (Light-Emitting Diode) indicador de que ele está vendo, permaneçam no lado interno do veículo (Figura 72). A distância, entre o solo, em local plano, e a câmera do dispositivo permaneceu a aproximadamente 1,35 m, com a variação de até 0,15 m podendo ocorrer devido às irregularidades existente no terreno no ato da captura da imagem.

Foi estipulado que o robô seria levado até estas duas localidades em intervalos de aproximadamente 15 *dias* para a capturas das imagens rurais. Dentro deste intervalo, o robô é levado para diversas outras localidades para a coleta das mais diversas imagens, a fim de capturar imagens fora do contexto canavieiro, para compor toda base de imagens

necessárias para o aprimoramento e melhoria da qualidade do aprendizado da máquina (SILVEIRA; BULLOCK, 2017; HARTSHORN, 2016; AWAD; KHANNA, 2015; LUGER, 2013).

A imagem capturada foi, primeiramente, armazenada no próprio DRR no formato RGB. O método de compressão utilizado foi o *Joint Photographic Experts Group* (JPEG ou JPG). As dimensões de cada imagem foram mantidas em 320 *pixels* de largura e 240 *pixels* de altura, com resolução horizontal e vertical de 72 dpi (*Dots Per Inch*) e, 24 bits de intensidade. Depois de armazenada no DRR as imagens foram transferidas para o DGGC e removidas do dispositivo, conforme a metodologia expressa em 4.2.4.

Entre o período de 4 de setembro a 24 de janeiro de 2020 foram adquiridos um total de 36.979 imagens positivas e negativas dos mais diversos locais e ambientes, incluindo, principalmente as fazendas citadas anteriormente. Os dados apresentados na (Figura 73) foram adquiridos por meio de consulta ao banco de dados PostgreSQL.

A princípio as imagens foram armazenadas no Banco de Dados *MySQL* (MYSQL, 2020), porém, um tempo depois, por razões de desempenho e robustez, estes foram transferidos para o Banco de dados *PostgreSQL* (POSTGRESQL, 2020). Durante o processo de transferência ocorreu uma falha e o processo de transferência foi reiniciado resultando assim em algumas duplicidades de imagens no BD. Para solucionar este problema foi criado um *script Python* que percorre todos os registros da tabela "imagens" a procura destas duplicidades e eliminá-las. Para isso, as imagens foram carregadas para um buffer e disponibilizada para o OpenCV (OPENCV, 2020; SIAHAAN; SIANIPAR, 2019). Depois disso este procedimento foi incorporado na Classe PDI e passou a permitir sua aplicação a todos os métodos que necessitem desses dados de imagem.

4.4 Processo de Extração de Características da Imagem

Para este projeto foram testados vários métodos e sobreposição de métodos para realizar a extração de características da imagem, contudo, concluiu-se que para a realização dos testes com o ERACI deveria ser utilizado uma forma que abrangesse pelo menos o histograma da imagem, por meio da média, moda e desvio padrão de cada uma das camadas RGB e, da imagem em tons de cinza. Estes dados fornecem informações da imagem como um todo e não apenas dos objetos de interesse na imagem. Por esta razão também foi pensado em uma forma que simule digitalmente o processo em que o ser humano foca em um objeto específico da imagem. Para isso foi utilizado a segmentação por cor utilizando o esquema HSV. Baseado na hipótese de que quando se está focado em um objeto da imagem, as cores que compõe esta parte da imagem tendem a prevalecer em relação às outras. Para o ERACI este foco pode ser obtido por meio da distância entre o objeto e o observador e, por meio do ângulo de visão deste observador em relação ao objeto

observado. Um exemplo desta ideia é que quanto mais próximo a câmera do DRR está da cana-de-açúcar mais representativa serão as cores dela na imagem e, assim, a segmentação da imagem poderá ser realizada levando em consideração a cana-de-açúcar e não o céu, solo, edificação, veículos, ervas daninhas, dentre outros que possam estar ao redor desta, que é o objeto de interesse, num determinado momento.

Após a segmentação da imagem por cor, foi possível obter os 7 momentos invariantes de HU (HU et al., 2012). Para se chegar a esta forma de obtenção de características da imagem foram feitos alguns testes que levaram em conta a relevância das características para a realização dos testes de classificação sobre o ERACI e, o tempo gasto entre a extração das características e a classificação. Dentre as observações levou-se em conta que o ganho na geração das texturas resultantes da segmentação e o ganho em qualidade dos itens (Figura 28) e (Figura 30) foram ínfimos se comparados ao apresentado na (Figura 29). Considerando que a sobreposição de vários métodos para pré-processamento exige um maior tempo de processamento, optou-se por utilizar a sobreposição de apenas dois métodos para a segmentação das imagens, ficando assim a sobreposição dos métodos para uniformização e de detecção de bordas por *Canny* (Subseções 3.5.5.13 e 3.5.5.15).

A Tabela 6 mostra os atributos, tipo de dado e a respectiva variação dos atributos envolvidos na caracterização da imagem. Os campos de número 1 até 3 estão implícitos na segmentação da imagem por cor. Já os campos de 4 até 18 são utilizados para o treinamento e o posterior reconhecimento dos padrões e classificação da imagem.

A classificação das imagens foi realizada com base nas características extraídas conforme essa metodologia (Subseção 3.5.5.15). Para isso, o usuário faz uso do aplicativo eraci devidamente instalado no DM. Este aplicativo, quando conectado ao DGGC realiza uma chamada ao subsistema de serviço web que envia ao requisitante uma imagem assinalada, obtida de forma aleatória (Subseção 4.2.8).

Baseado nisso, foram classificadas 1.579 imagens no período de 20 de dezembro de 2019 até 25 de fevereiro de 2020 conforme bode ser observado na Figura 74.

4.5 Geração de Inteligência Artificial

Para realizar o treinamento e teste dos algoritmos foram utilizados os campos de 4 a 18 apresentados na Tabela 6. Além disso foram estabelecidas por meio do aplicativo “eraci"as possíveis classes em que uma imagem possa pertencer. Desta forma, as classes foram organizadas em classe e subclasse, possibilitando assim que uma determinada classe possa pertencer a uma super classe que melhor representa um grupo de imagens. As classes e a sua respectiva super classe são apresentadas na (Tabela 7).

Como pode ser observado na (Tabela 7), as classes foram agrupadas em 3 grupos

denominados super classes: Área Rural e Canavial possuem uma gama de 8 classes possíveis e, área urbana comporta 7 classes.

Se utilizássemos para realizar a classificação das imagens uma metodologia baseada na escolha ao acaso, pode-se dizer que teríamos de $(100/7 \sim 14\%)$ a $(100/8 \sim 12\%)$ de chances de realizar uma classificação correta. Com base nestes resultados concluiu-se que para certificar que o algoritmo está reconhecendo padrões e tomando decisão baseado nisto ele deverá ter, no mínimo uma acurácia maior que 14% para área urbana e 12% para área rural e canavial.

Com o propósito de realizar estes testes, foi criado um algoritmo que faz uso da validação cruzada utilizando *StratifiedKFold* do *python* com seu parâmetro *n_splits* definido como 10, sobre as imagens previamente classificadas no BD (SCIKIT-LEARN, 2020).

Para verificar se os algoritmos estão realmente aprendendo à medida que os dados pré-classificados são inseridos, foi criado um processo no módulo *inteligencia.py* (Seção 3.5) que realiza o acréscimo de mais amostras no classificador, de forma cíclica e incremental 50 dados por iteração e, extrai valores referentes a acurácia sob esta determinada quantidade de dados. O valor incremental de 50 registros é o mesmo utilizado para a geração automática de conhecimento com o sistema em produção, recebendo classificações realizadas pelos usuários por meio do aplicativo “eraci"instalado no dispositivo móvel. Esta quantidade foi estipulada ao acaso, e visa permitir a geração de dados que descrevem a evolução, ou não, da acurácia dos algoritmos. Com os dados gerados por este processo foi possível gerar uma série de gráficos que mostram a relação existente entre a quantidade de dados pré-classificados e a acurácia dos algoritmos classificadores.

Após a análise dos dados resultantes da estimação da acurácia, em termos de quantidade de dados utilizados para o treinamento dos classificadores, verificou-se que no início, quando existiam poucos dados para treinamento, a acurácia dos classificadores pareciam melhores. Porém, com um olhar mais criterioso sobre esta informação, foi observado que este fato ocorre devido à pouca variabilidade de itens categóricos. Conforme esta quantidade vai aumentando, maior é a dificuldade em se classificar, resultando assim em uma acurácia menor entregue pelos algoritmos. Esta análise sobre os gráficos, também mostrou que todos eles apresentam um crescimento constante no valor da acurácia, a partir de determinada quantidade de dados, o que nos leva a acreditar que, ambos os algoritmos, estão sujeitos a um limiar, onde, a partir daí, o algoritmo se estabiliza e se mantem aprendendo e melhorando o seu desempenho. Por esta razão, foi dada maior importância em se descobrir com que quantidade de dados pré-classificados os algoritmos começam a apresentar boa acurácia de forma crescente e estável.

Com este propósito foram reunidos em 4 tabelas, uma para cada agrupamento categórico, as informações sobre o ponto do gráfico onde a quantidade de dados influencia

positivamente e regularmente na acurácia dos algoritmos. Este ponto é equivalente ao mínimo global da função da acurácia projetado sobre a função de quantidade. A ele foi dado o nome de Quantidade Mínima para Regularidade (QMR).

Como pôde ser notado nos gráficos (Figuras 75 a 81) o QMR se dá conjuntamente com a menor acurácia entregue pelo respectivo algoritmo ou mínimo global e, se mantem em crescimento constante, podendo ocorrer oscilações que não ultrapassam o QMR.

As (tabelas de 8 a 11) apresentam: os classificadores, o QMR, a acurácia e o desvio padrão para respectiva acurácia.

Quando analisamos o QMR em termos da classificação de superclasses notamos que o classificador MLP atinge este ponto com 1.250 dados. Desta forma o algoritmo entrega uma acurácia aproximada de 74% com um desvio padrão de aproximadamente $0,084$.

Já, quando o QMR é analisado em termos da classificação da superclasse canavial este ponto é atingido pelo algoritmo RF, com aproximadamente 36% de acurácia e aproximadamente $0,063$ de desvio padrão. Nesta tabela, é possível notar também que, embora o MLP não seja o classificador que atinge este ponto primeiro, ele o atinge com 185 dados, porém, com acurácia melhor que a entregue pelo RF. Um ponto negativo referente ao MLP sobre estes resultados é que ele apresenta um desvio padrão com valor superior, demostrando assim um grau maior de incerteza quanto às previsões.

Quando a tabela resultante da análise do QMR (Tabela 10) é observada em relação a classificação da superclasse área rural, tem-se que a classificação começa a acontecer logo no início com um QMR igual a 10. Nesta quantidade os classificadores KNN, MLP e NB iniciam a classificação de forma regular. Nesta situação, foi considerado que a acurácia entregue pelo classificador NB que é incapaz de realizar a classificação neste ponto, porém, que apresenta, juntamente com os outros classificadores, sempre uma evolução no valor de sua acurácia (Figura 78c). Levando em consideração a acurácia entregue por estes 3 algoritmos, tem-se que o KNN e o MLP (Figura 77c e Figura 81c) tem o início de sua evolução positiva similar com uma acurácia de $12,5\%$, com um desvio padrão de aproximadamente $0,21$.

Embora esta acurácia não seja a desejada, pois ela é equivalente a acurácia obtida quando se utiliza qualquer meio de classificação aleatória, temos que, seu gráfico de acurácia equivalente mostra uma melhora significativa da sua acurácia proporcionalmente, com algumas oscilações nunca inferior aos $12,5\%$, ao aumento da quantidade de dados para treinamento.

O classificador NB foi o primeiro a atingir o ponto QMR para a classificação da superclasse área urbana, apresentando ali o valor de 249. A acurácia neste ponto foi de aproximadamente 38% e o desvio padrão de aproximadamente $0,064$.

Uma das razões a serem levantadas aqui sobre a oscilação da acurácia, durante todo o

processo de acréscimo de dados para o treinamento é o fato de ocorrerem desbalanceamentos em relação a quantidade de categorias acrescidas em cada processo iterativo (CHAWLA; JAPKOWICZ; KOTCZ, 2004). Este fato propicia que os algoritmos se tornem muito bons em reconhecer padrões referentes a uma determinada categoria e ruim em outras que tenham menos amostras (Figuras 82 a 85).

Por meio da análise de todos estes dados, é possível dizer que os métodos utilizados para realizar a classificação das imagens tem uma tendência a ter uma melhor acurácia, a partir do ponto QMR. Assim, a acurácia entregue pelos algoritmos tendem a aumentar proporcionalmente ao aumentando da quantidade de registros pré-classificados, utilizados como fontes de informação para o treinamento destes classificadores.

Devido a variação da acurácia entregue pelos algoritmos, pode-se dizer que a estratégia de verificar em tempo de execução, qual é o algoritmo que entrega a melhor acurácia para a classificação é válida, pois, desta forma, é utilizado sempre o classificador que melhor se adequa a quantidade de dados pré-classificados utilizados para treinamento. A imagem a seguir (Figura 86) apresenta um exemplo da classificação de uma imagem pelo ERACI. Nela é possível ver a superclasse, subclasse e a porcentagem relativa a confiança entregue pelo algoritmo utilizado para classificar esta imagem.

4.6 Tempo de Processamento

O tempo gasto pelo DRR para realizar todo os processos até o reconhecimento do padrão na imagem em tempo real foi capturada por uma função, com este propósito, embutida no *script imagem.py* que faz parte do subsistema de visão computacional. O resultado apresentado por esta função é apresentado no gráfico a seguir (Figura 87)).

Uma análise sobre os dados que serviram de base para a geração do gráfico (Figura 87) mostram que o tempo gasto para o processamento dos dados, desde a captura da imagem até o reconhecimento dos padrões na imagem variam entre aproximadamente $0,075651407$ e $1,69562912$ s, o que dá ao dispositivo um tempo médio para processamento de $0,196309677$ s. O desvio padrão resultante destes dados é da conta de $0,099244419$ s.

A linha de tendência do gráfico, apresentado pela linha tracejada em vermelho, mostra uma tendência em manter este tempo perto do valor mínimo, na grande maioria das vezes, o que assegura que o sistema pode ser utilizado para realizar classificações em tempo real. Porém, um melhor desempenho do sistema é desejado. Para isso, espera-se que o ERACI seja acoplado a uma estrutura que haja em sincronia com o subsistema de visão computacional, para permitir que a classificação de uma imagem seja somente realizada após a finalização de uma classificação anterior, e assim sucessivamente. Isso pode ser conseguindo, dando ao ERACI a capacidade de locomoção e posicionamento de atuadores de forma autônoma. Apenas desta forma, seu uso em produção poderá ser aconselhado.

Tabela 5 – Métodos disponibilizados pelo subsistema de serviço web.

Método HTTP	**URI**	**Ação**
GET	/robot/api/v1.0/usuario/ int:page_id>/ int:page_size>	Recuperar lista de usuários com paginação
GET	/robot/api/v1.0/usuario/ int:usuario_id>	Recuperar um usuário em específico pelo seu número ID
GET	/robot/api/v1.0/usuario/ string:identificador_usuario>	Recuperar um usuário em específico pelo seu e-mail ou número de telefone
POST	/robot/api/v1.0/usuario/	Criar um novo usuário
PUT	/robot/api/v1.0/usuario/ int:usuario_id>	Atualizar dados de um usuário
DELETE	/robot/api/v1.0/usuario/ int:usuario_id>	Exclui um usuário
GET	/robot/api/v1.0/classe/ int:page_id>/ int:page_size>	Recupera classes com paginação
GET	/robot/api/v1.0/classe/ int:classe_id>	Recupera uma classe em específico
POST	/robot/api/v1.0/classe/	Cria uma nova classe
PUT	/robot/api/v1.0/classe/ int:classe_id>	Atualiza os dados de uma classe
DELETE	/robot/api/v1.0/classe/ int:classe_id>	Excluí uma classe
GET	/robot/api/v1.0/imagem/ int:page_id>/ int:page_size>	Recupera uma lista de imagens com paginação
GET	/robot/api/v1.0/imagem/ int:imagem_id>	Recupera uma imagem em específico
GET	/robot/api/v1.0/imagem/	Recupera uma imagem aleatória com segmentação e pré-classificada
POST	/robot/api/v1.0/imagem/	Cria uma imagem
PUT	/robot/api/v1.0/imagem/ int:imagem_id>	Atualiza uma imagem
DELETE	/robot/api/v1.0/imagem/ int:imagem_id>	Excluí uma imagem
GET	/robot/api/v1.0/classificacao/ int:page_id>/ int:page_size>	Recupera uma lista de classificações
GET	/robot/api/v1.0/classificacao/ int:imagem_id>/ int:classe_id>	Recupera uma classificação em específico
POST	/robot/api/v1.0/classificacao/	Adiciona uma nova classificação
PUT	/robot/api/v1.0/classificacao/ int:imagem_id>/ int:classe_id>	Atualiza uma classificação
DELETE	/robot/api/v1.0/classificacao/ int:imagem_id>/ int:classe_id>	Excluí uma classificação
GET	/robot/api/v1.0/previsao/ int:imagem_id>	Recupera a classificação possível de uma determinada imagem
GET	/robot/api/v1.0/inteligencia/	Recupera uma lista de classificadores

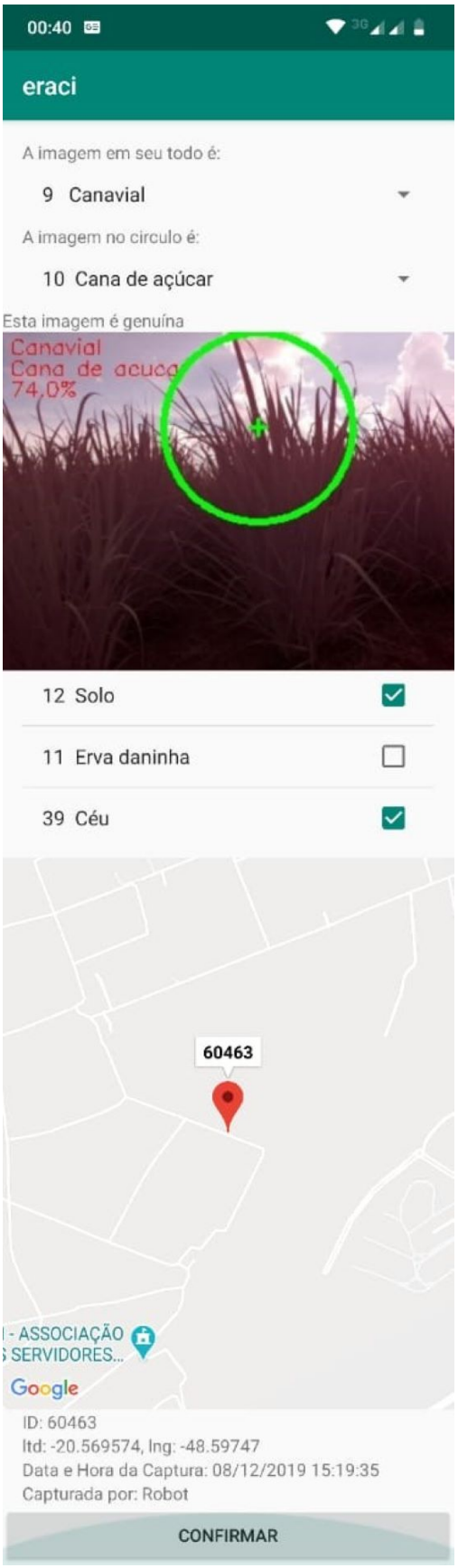


Figura 70 – Tela do aplicativo para dispositivo móvel destinada a classificação de imagens.

Fonte: Autoria Própria

(a) (b)

Figura 71 – Telas do subsistema supervisório local. (a) Tela de carregamento de dispositivos; (b) Tela de supervisão.

Fonte: Autoria Própria

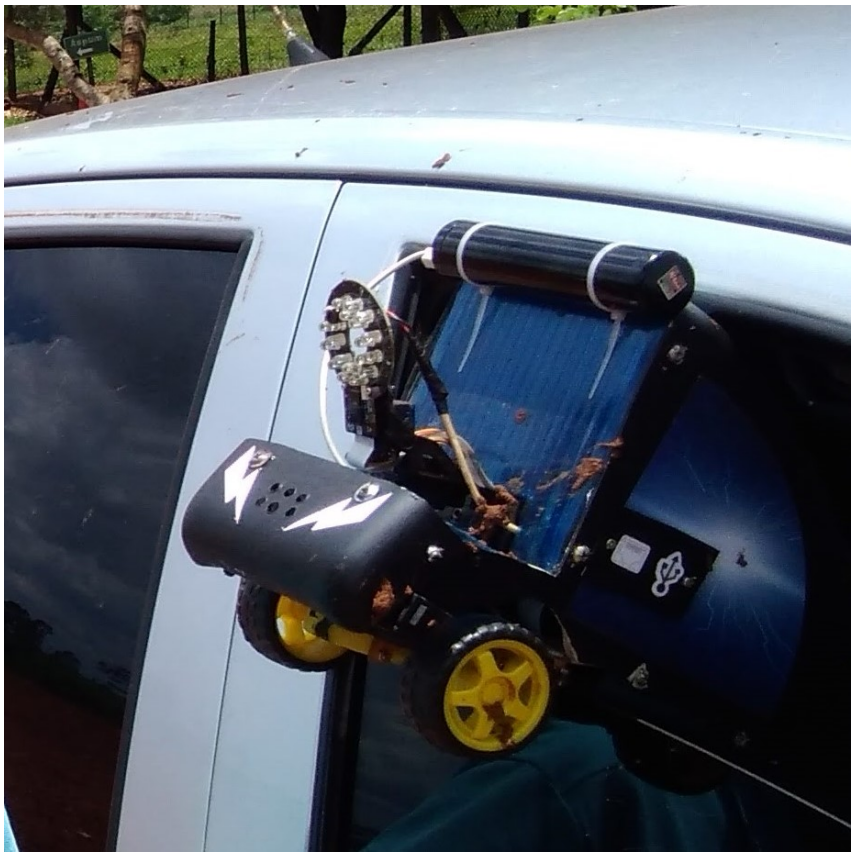


Figura 72 – ERACI acoplado ao veículo.

Fonte: Autoria Própria

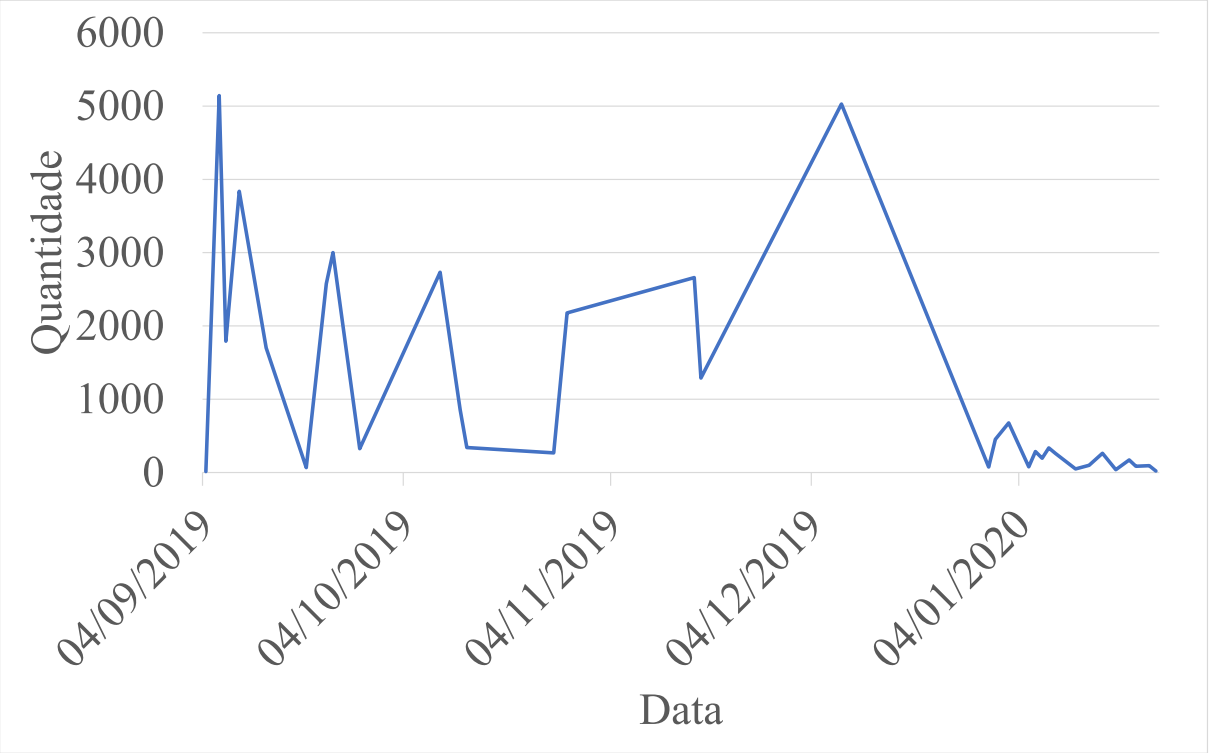


Figura 73 – Quantidade de imagens capturadas pelo dispositivo robótico móvel (DRR).

Fonte: Autoria Própria

Tabela 6 – Campos e respectivos tipos de dados utilizados para o treinamento e teste dos algoritmos.

Número	**Campo**	**Tipo**	**Variação**
1	Modavermelho	Inteiro	0 – 255
2	Modaverde	Inteiro	0 – 255
3	Modaazul	Inteiro	0 – 255
4	Mediavermelho	Float	0 – 255
5	Mediaverde	Float	0 – 255
6	Mediaazul	Float	0 – 255
7	Mediacinza	Float	0 – 255
8	Desviopadraovermelho	Float	0 – 255
9	Desviopadraoverde	Float	0 – 255
10	Desviopadraoazul	Float	0 – 255
11	Desviopadraocinza	Float	0 – 255
12	hu1	Float	∞
13	hu2	Float	∞
14	hu3	Float	∞
15	hu4	Float	∞
16	hu5	Float	∞
17	hu6	Float	∞
18	hu7	Float	∞

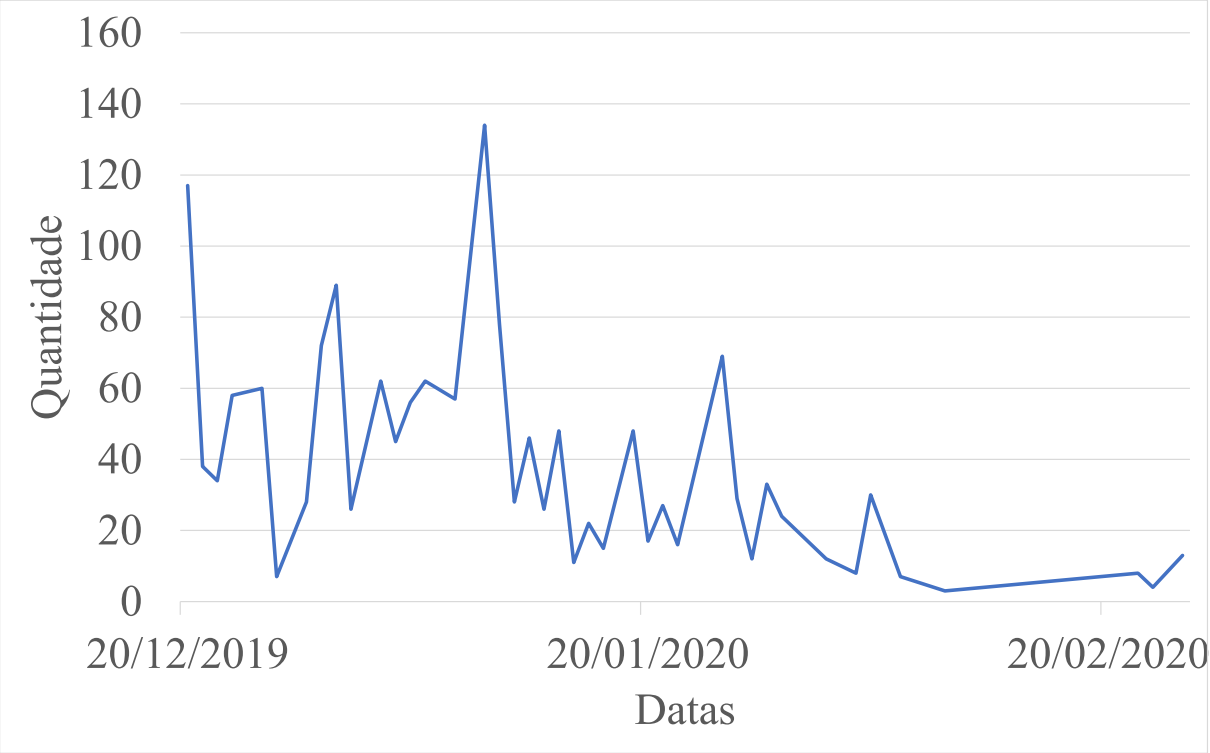


Figura 74 – Classificações manuais realizadas por meio do aplicativo eraci instalado no dispositivo móvel agrupados por data.

Fonte: Autoria Própria

Tabela 7 – Classes e super classes utilizadas para a classificação de uma determinada imagem.

Código da Classe	**Classe**	**Código da Super Classe**	**Super Classe**
54	Erva daninha	4	Área Rural
56	Árvore		
59	Solo		
61	Céu		
63	Pessoa		
69	Veículo		
86	Edificação		
92	Agricultura		
10	Cana-de-açúcar	9	Canavial
11	Erva daninha		
12	Solo		
39	Céu		
81	Árvore		
87	Edificação		
89	Veículo		
90	Pessoa		
41	Árvore	37	Área Urbana
42	Veículo		
49	Edificação		
60	Céu		
62	Pessoa		
64	Erva daninha		
65	Solo		

Tabela 8 – QMR relativo aos classificadores para a classificação de superclasses.

Classificador	**QMR**	**Acurácia**	**Desvio Padrão**
DT	1800	68,72222222	0,007737993
KNN	2000	66,90078384	0,011677682
MLP	1250	74,0803511	0,084252265
NB	2150	55,53425846	0,011706561
RF	2050	59,90313346	0,011707259
RL	1650	73,87878788	0,013388316
SVM	1600	62,99735556	0,036370276

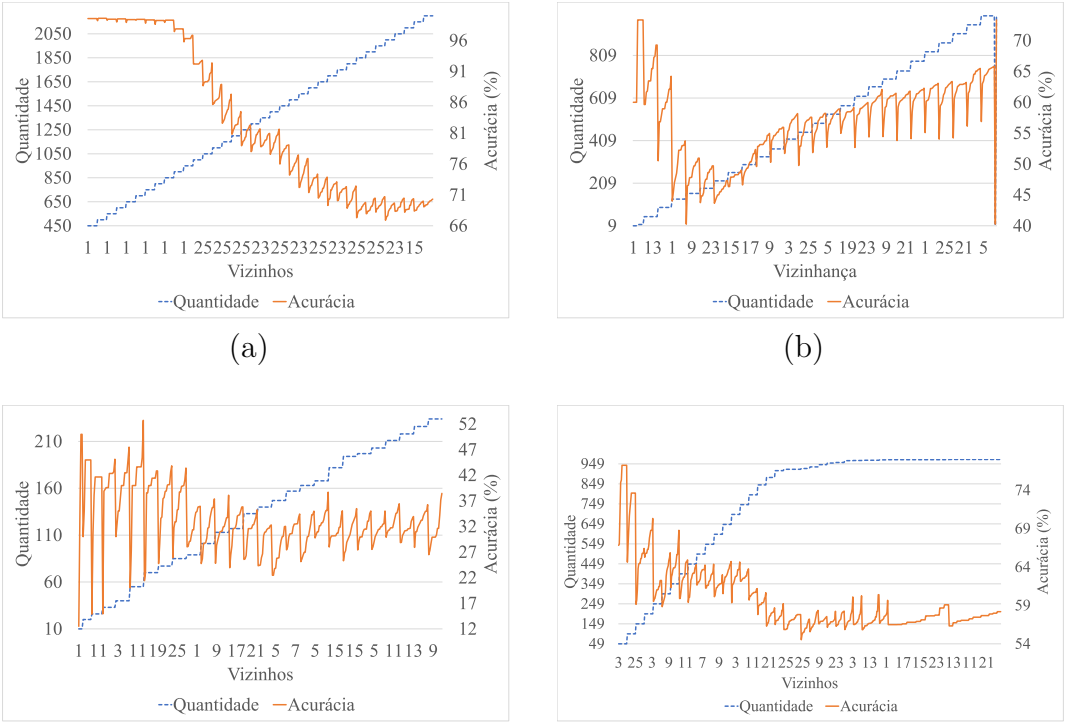


Figura 75 – Relação existente entre quantidade de dados e acurácia do algoritmo KNN. (a) Super Classe; (b) Classe Canavial; (c) Área Rural; (d) Área Urbana.

Fonte: Autoria Própria

Tabela 9 – QMR relativo aos classificadores para a classificação da subclasse canavial.

Classificador	**QMR**	**Acurácia**	**Desvio Padrão**
DT	161	38,53276353	0,085595626
KNN	161	40,26590693	0,134091588
MLP	185	42,6344086	0,108922396
NB	220	39,96496496	0,0708332
RF	134	36,52832675	0,063101137
RL	134	46,31093544	0,108465841
SVM	185	42,16845878	0,067352286

Tabela 10 – QMR relativo aos classificadores para a classificação da subclasse área rural.

Classificador	**QMR**	**Acurácia**	**Desvio Padrão**
DT	26	22,61904762	0,132030197
KNN	10	12,5	0,216506351
MLP	10	12,5	0,216506351
NB	10	0	0
RF	117	18,64035088	0,087667748
RL	26	11,30952381	0,120673422
SVM	147	26,58333333	0,109196349

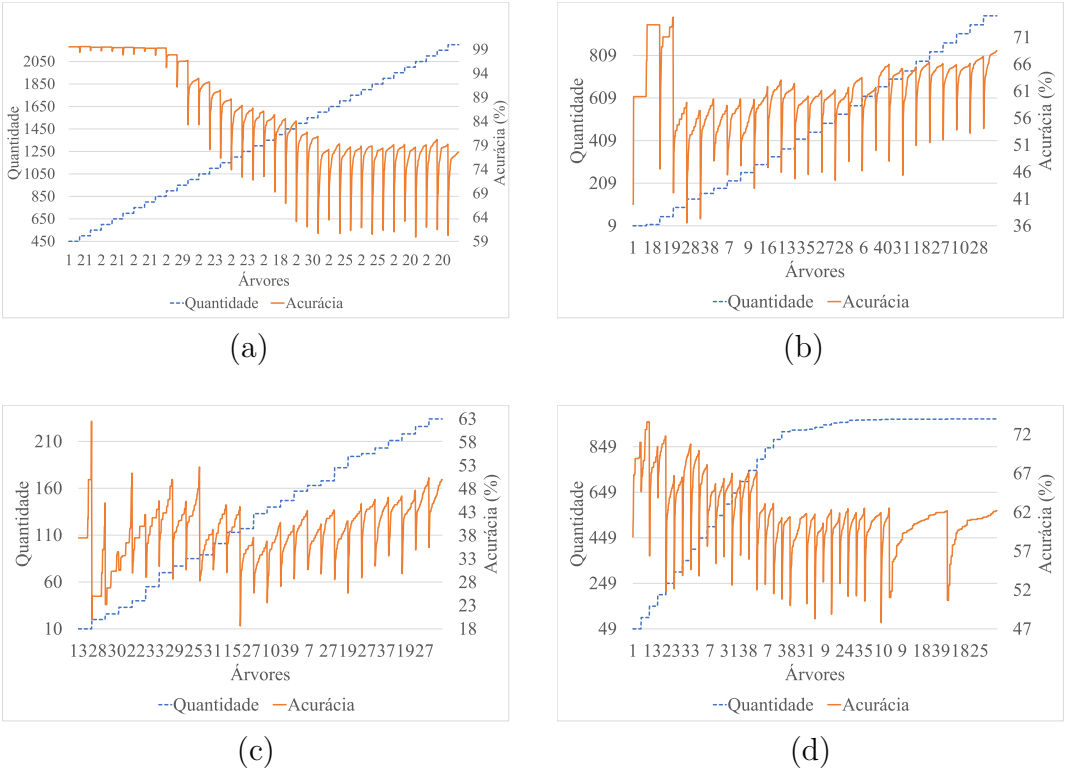


Figura 76 – Relação existente entre quantidade de dados e acurácia do algoritmo RF. (a) Super Classe; (b) Classe Canavial; (c) Área Rural; (d) Área Urbana.

Fonte: Autoria Própria

Tabela 11 – QMR relativo aos classificadores para a classificação da subclasse área urbana.

Classificador	**QMR**	**Acurácia**	**Desvio Padrão**
DT	967	51,49975718	0,039464116
KNN	926	54,52869711	0,034832129
MLP	966	52,27743271	0,058860741
NB	249	38,90824623	0,064327257
RF	969	47,79605347	0,050603489
RL	915	53,67145396	0,045285085
SVM	915	51,48922142	0,060668246

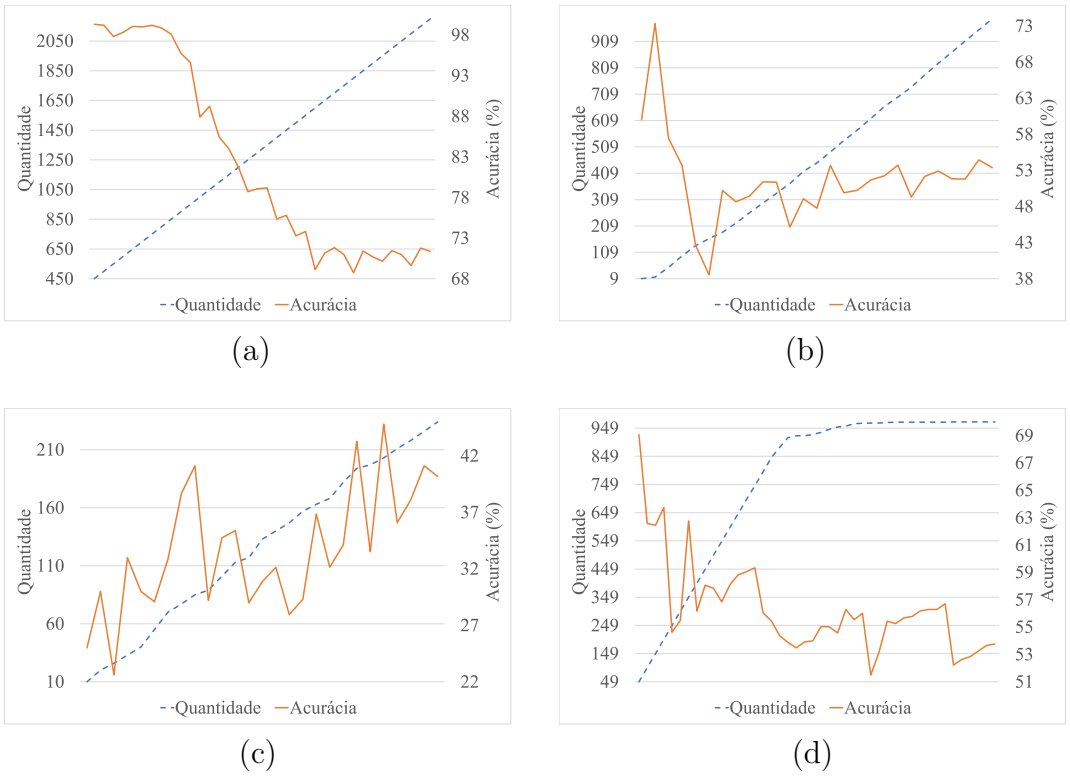


Figura 77 – Relação existente entre quantidade de dados e acurácia do algoritmo DT. (a) Super Classe; (b) Classe Canavial; (c) Área Rural; (d) Área Urbana.

Fonte: Autoria Própria

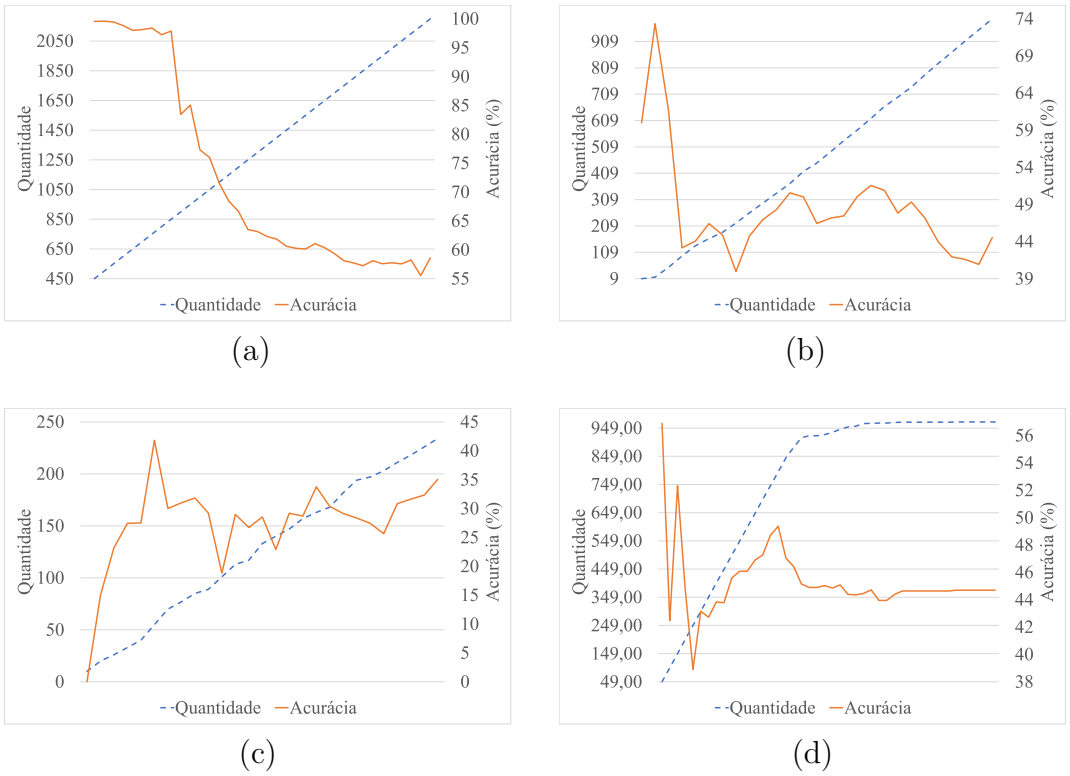


Figura 78 – Relação existente entre quantidade de dados e acurácia do algoritmo NB. (a) Super Classe; (b) Classe Canavial; (c) Área Rural; (d) Área Urbana.

Fonte: Autoria Própria

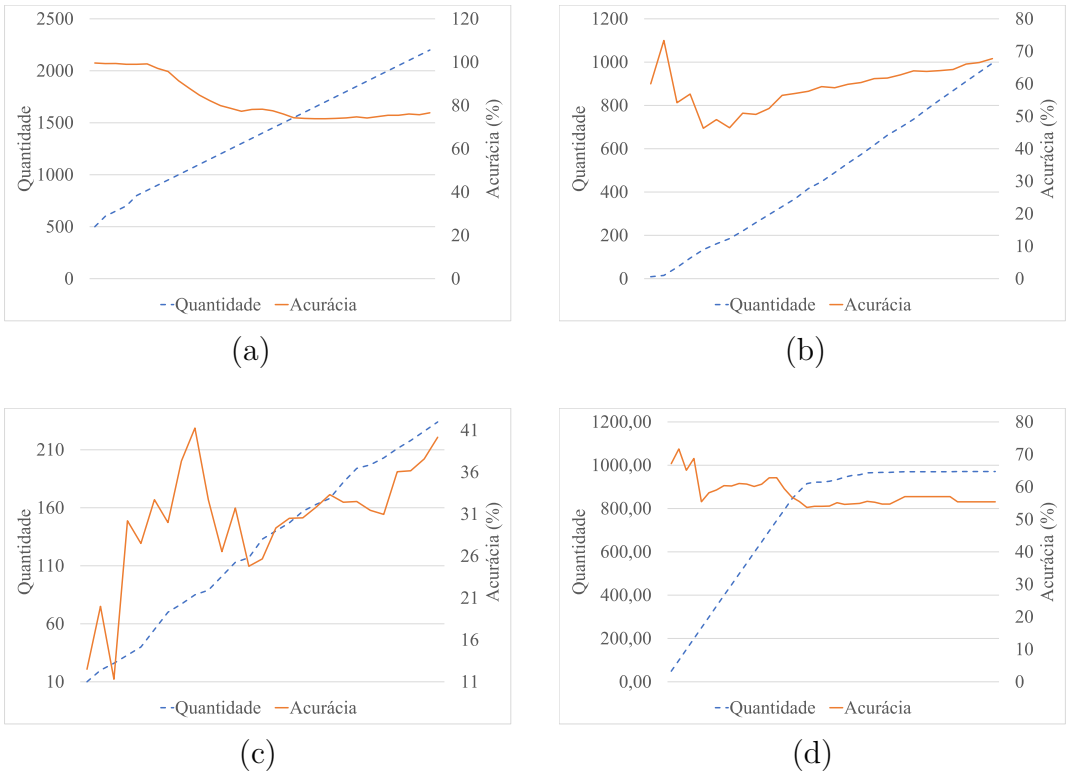


Figura 79 – Relação existente entre quantidade de dados e acurácia do algoritmo RL. (a) Super Classe; (b) Classe Canavial; (c) Área Rural; (d) Área Urbana.

Fonte: Autoria Própria

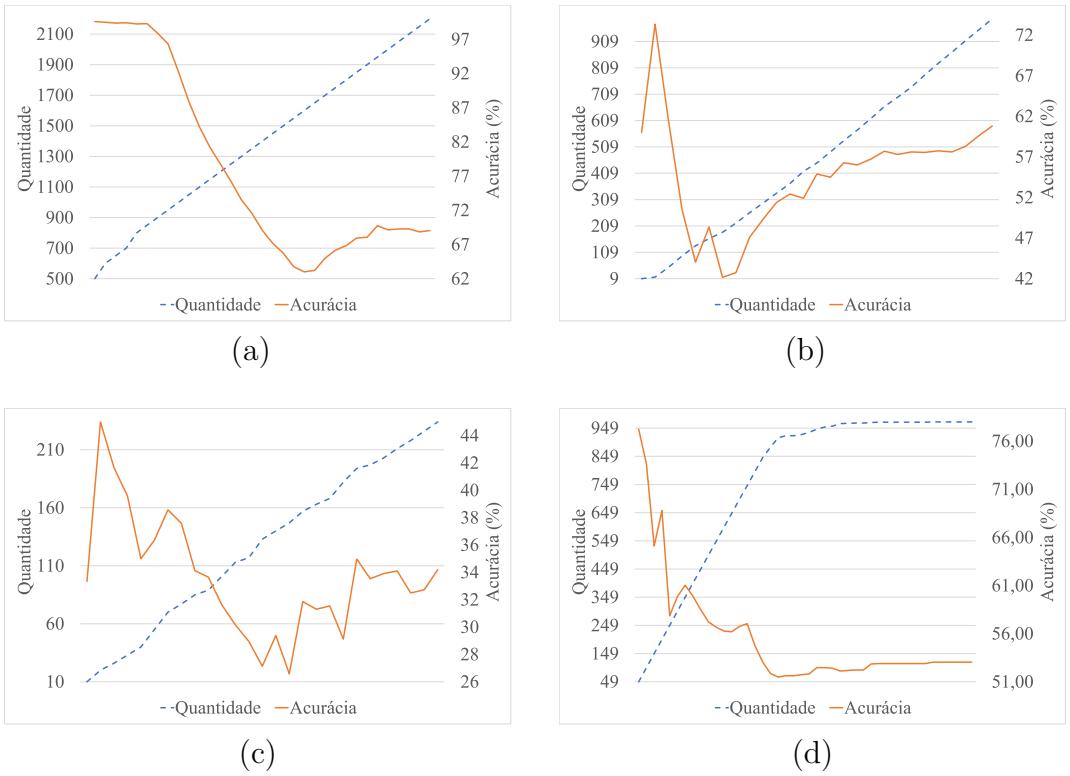


Figura 80 – Relação existente entre quantidade de dados e acurácia do algoritmo SVM. (a) Super Classe; (b) Classe Canavial; (c) Área Rural; (d) Área Urbana.

Fonte: Autoria Própria

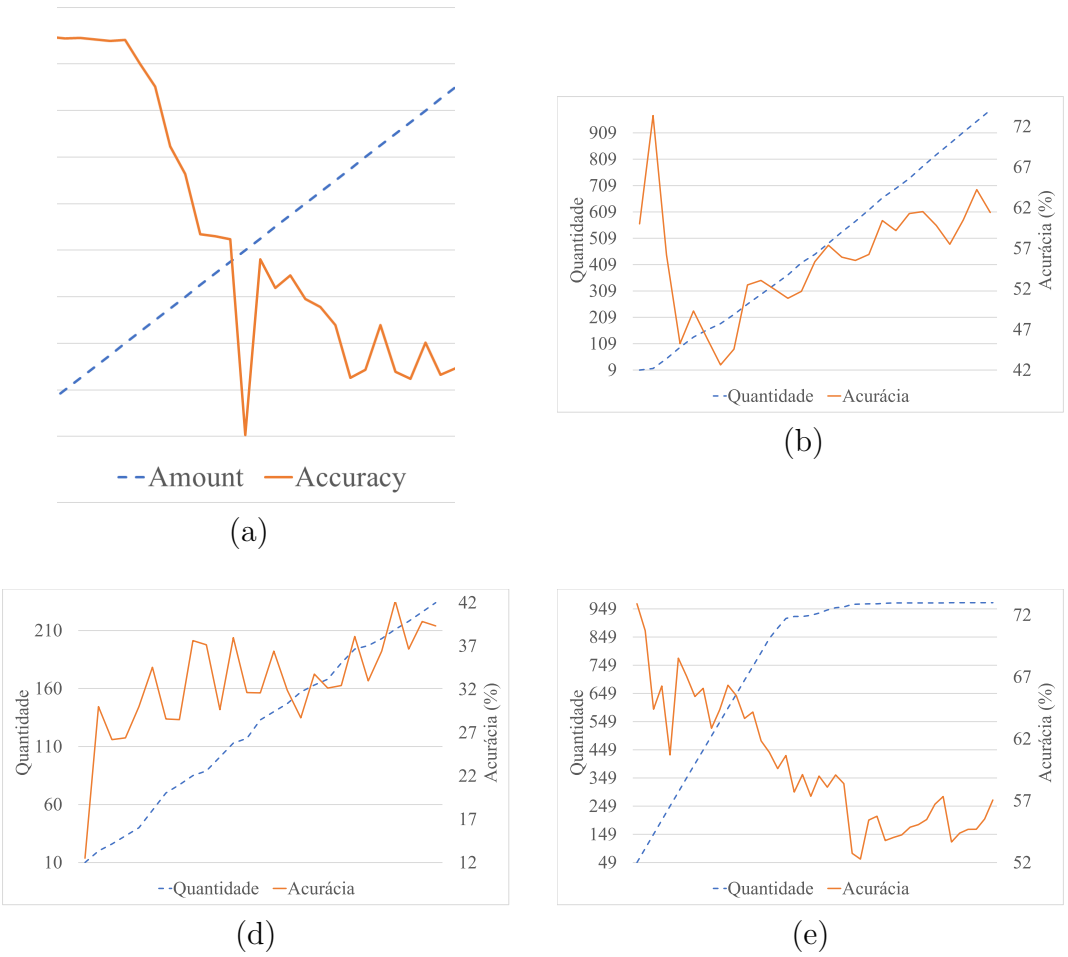


Figura 81 – Relação existente entre quantidade de dados e acurácia do algoritmo MLP. (a) Super Classe; (b) Classe Canavial; (c) Área Rural; (d) Área Urbana.

Fonte: Autoria Própria

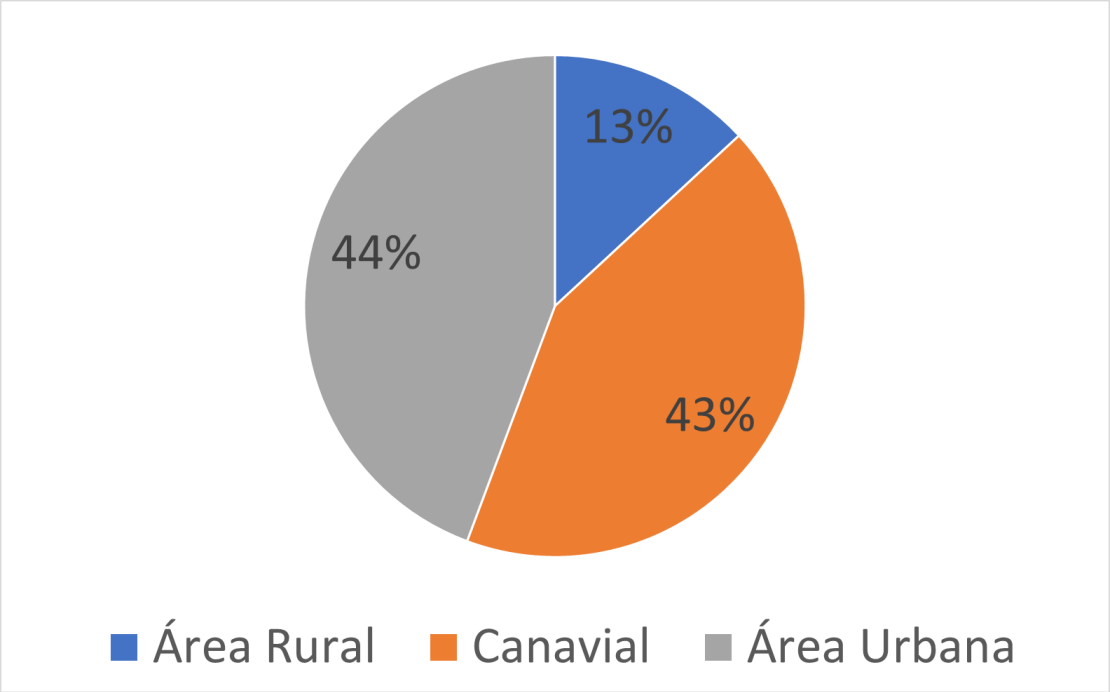


Figura 82 – Variabilidade da Quantificação de Amostras das Super Classes.

Fonte: Autoria Própria

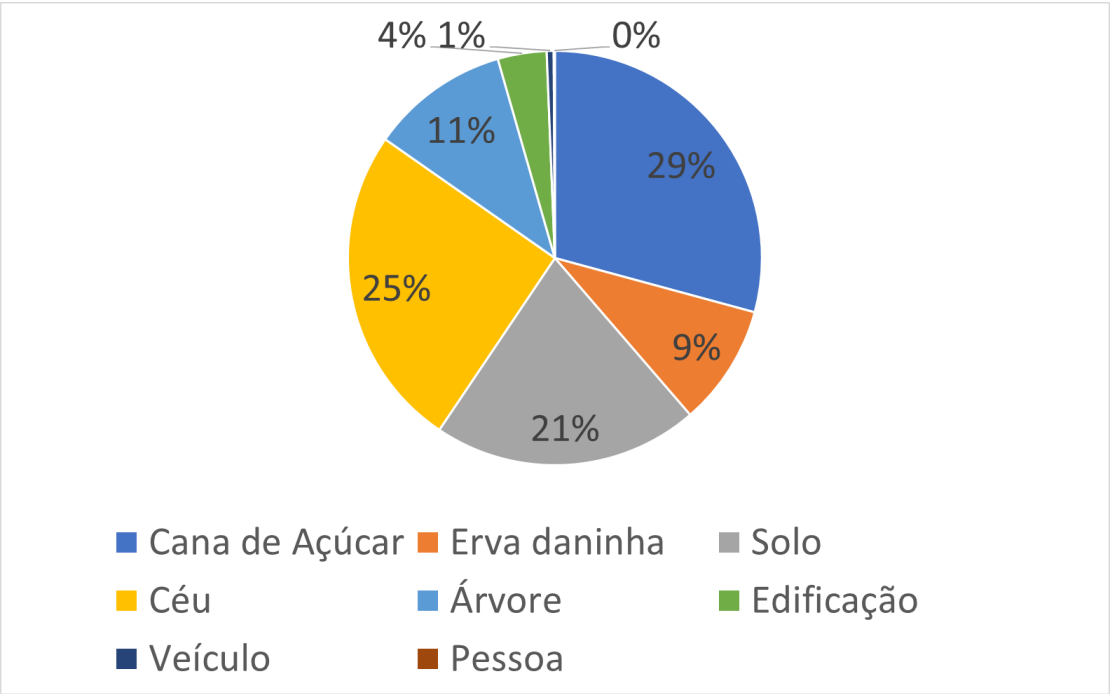


Figura 83 – Variabilidade da Quantificação de Amostras da Super Classe Canavial.

Fonte: Autoria Própria

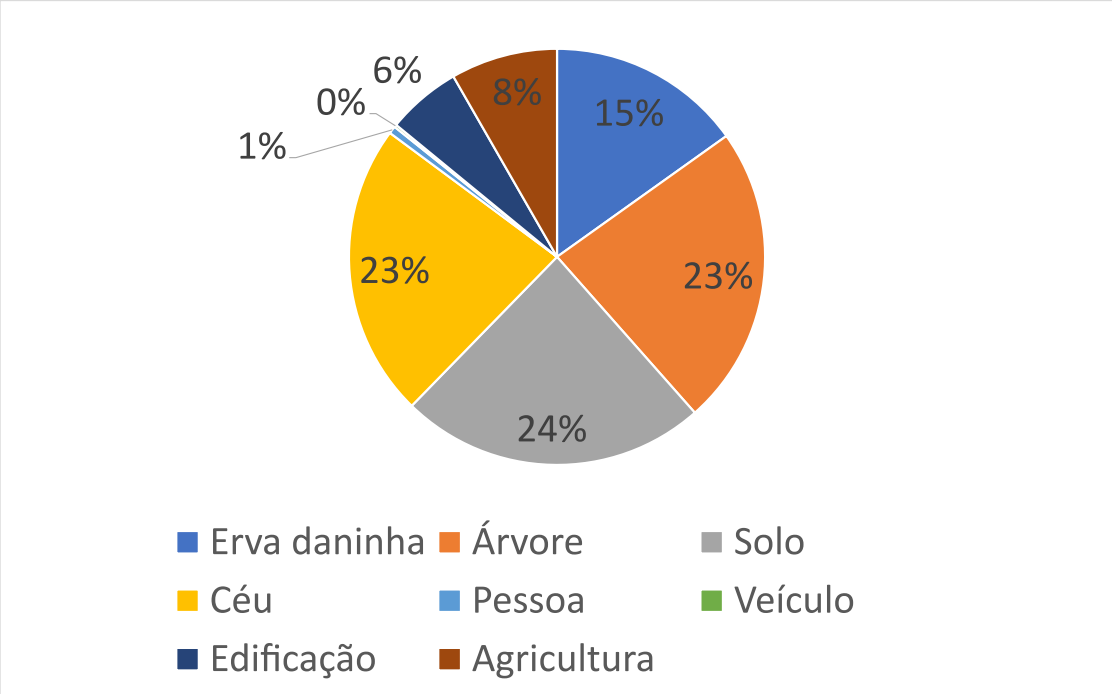


Figura 84 – Variabilidade da Quantificação de Amostras da Super Classe Área Rural.

Fonte: Autoria Própria

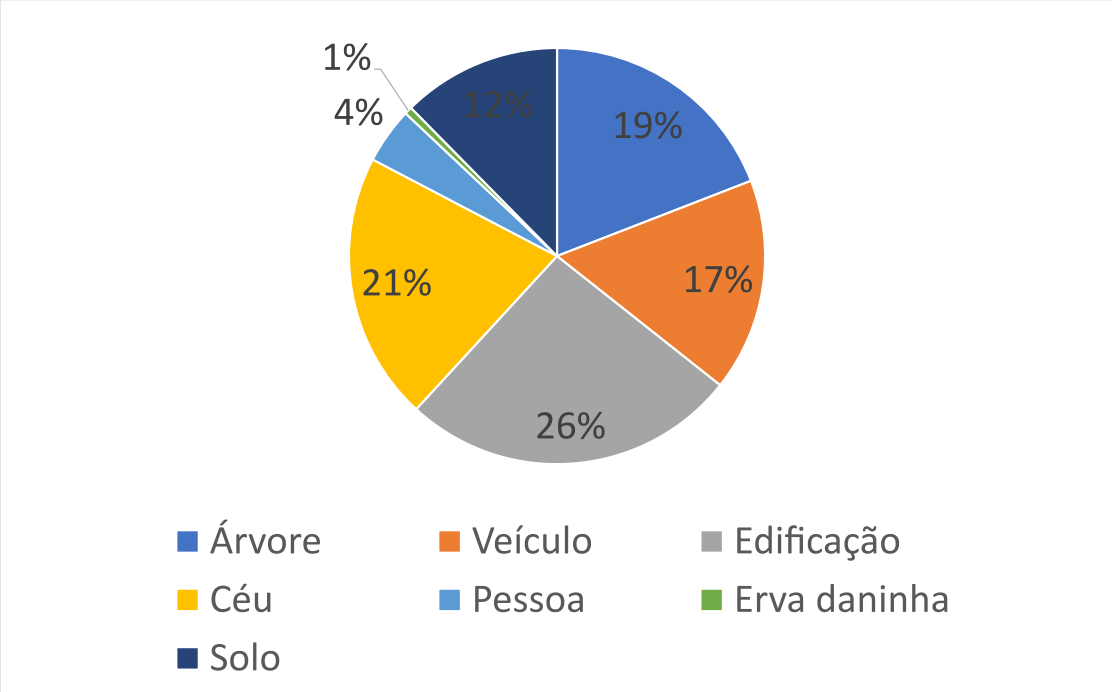


Figura 85 – Variabilidade da Quantificação de Amostras da Super Classe Área Urbana.

Fonte: Autoria Própria

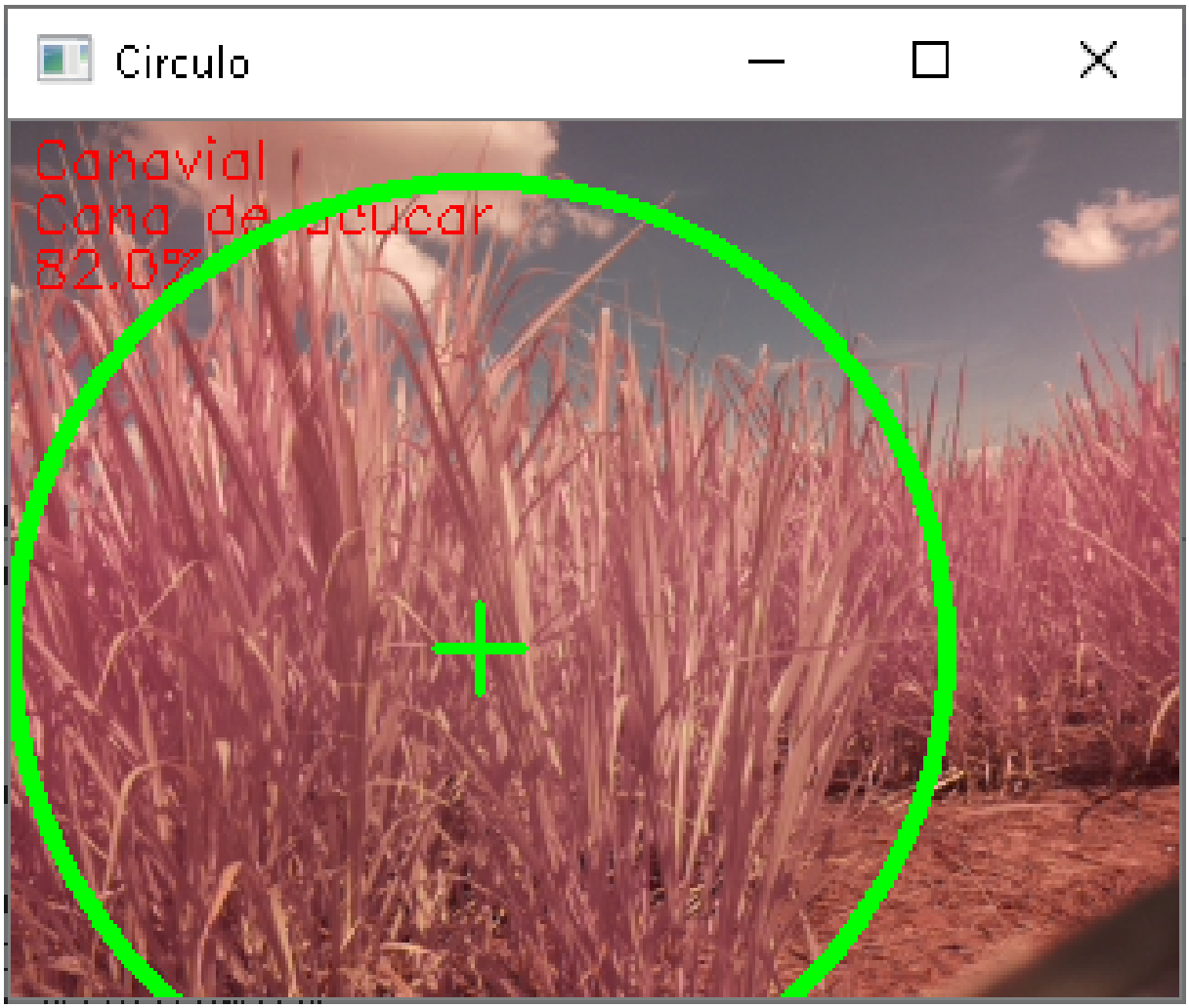


Figura 86 – Exemplo da classificação resultante da aplicação das técnicas de extração de características e classificação da imagem por meio do reconhecimento dos padrões identificados pelos algoritmos de Machine Learn.

Fonte: Autoria Própria

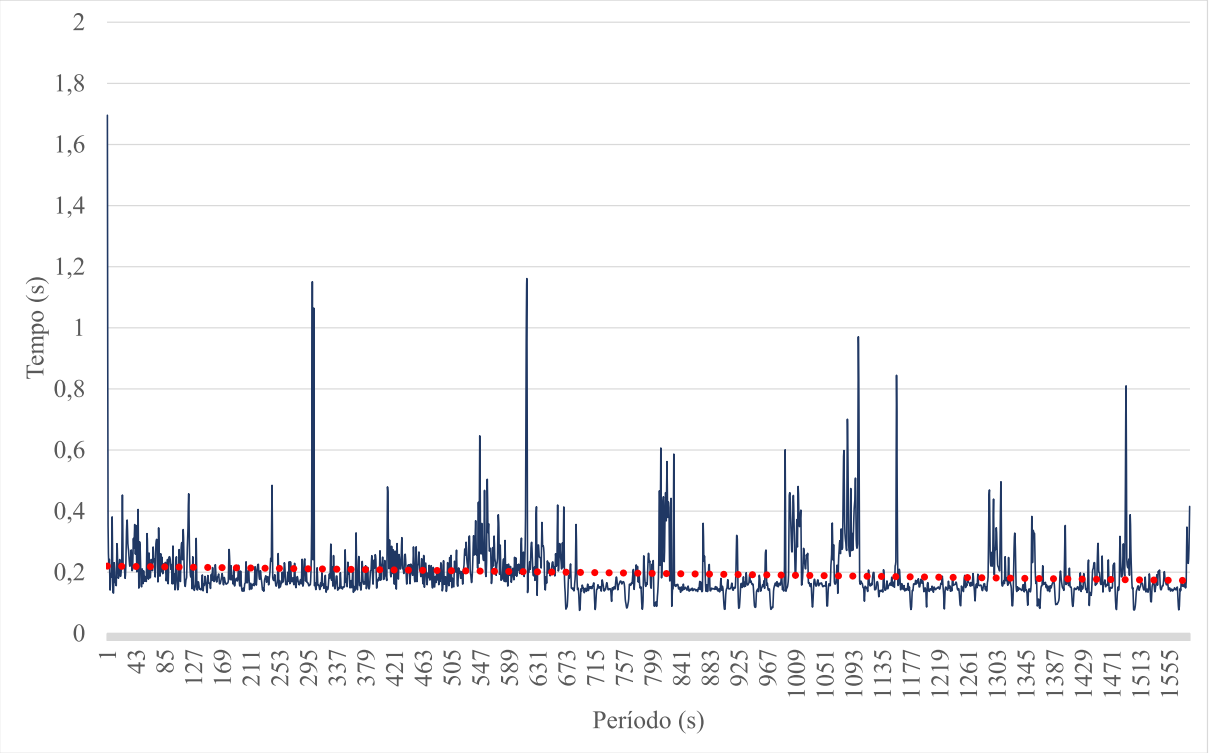


Figura 87 – Tempo de processamento necessário para realizar todos os procedimentos para extração de características e reconhecimento de padrões na imagem capturada em tempo real pelo DRR em um período de 1,585 s.

Fonte: Autoria Própria

5 Conclusão

O ERACI foi confeccionado com recursos *Open Source*, com *softwares* disponibilizados gratuitamente e *hardwares* baseado em computador de baixo custo voltado primordialmente para o ensino de informática, algo financeiramente adequado quando se trata em reproduzi-lo por empresas recém formadas.

Seu desenvolvimento mostrou que o computador *Rapberry Pi* pode ter seu uso expandido para o desenvolvimento de projetos mais robustos, servindo, não apenas como uma placa de prototipação mas também de unidade de processamento de sistemas robóticos voltados para setores produtivos, sendo assim, uma alternativa computacional de baixo custo para este fim.

Sendo assim, o projeto em si conseguiu realizar a aquisição e armazenamento das imagens capturadas em um banco de dados de forma automatizada e inteligente, pois, é capaz de detectar quando existe uma possível conexão entre o DRR que captura a imagem e o DGGC que a armazena em Banco de Dados.

Outro aspecto desejado para o ERACI foi que ele tivesse a capacidade de gerar conhecimento por meio de algoritmos de PID e IA, além de ser capaz de gerir este conhecimento. Estes objetivos foram atingidos por meio do módulo DGGC, que realiza de forma cíclica e automática a extração das características e a geração de conhecimento. Além disso, ele disponibiliza serviços web para a gestão e compartilhamento do conhecimento gerado.

O objetivo de reconhecer padrões em imagens por meio do DRR foi atingido, pois ele demostrou a capacidade de capturar imagens, enviar as imagens para o DGGC, obter os classificadores por meio de acesso a serviço publicado pelo DGGC e em seguida, realizar o reconhecimento dos padrões nas imagens em tempo real, com uma variação aceitável, porém, ainda não ideal. Além disso, a capacidade de realizar a escolha de qual classificador utilizar em tempo de execução se mostrou bem vinda, pois ela leva em conta que, um classificador pode apresentar melhor acurácia que outro diante da quantidade de amostras e sua variabilidade, existentes para treinamento em determinado momento.

Por fim, é possível concluir que o ERACI atingiu os objetivos propostos inicialmente e, que este sistema pode ser melhorado por meio da aplicação de algoritmos mais robustos para o reconhecimento de padrões e, também por meio de implementação de uma estrutura robótica capaz de possibilitar a locomoção, controlada por ele próprio, da estrutura no campo. Estas características desejadas, podem ser implementadas em projetos futuros.

Referências

ALVARES, C. A. et al. Köppen's climate classification map for brazil. *Meteorologische Zeitschrift*, Schweizerbart Science Publishers, Stuttgart, Germany, v. 22, n. 6, p. 711–728, 12 2013. Disponível em: <http://dx.doi.org/10.1127/0941-2948/2013/0507>. Citado na página 33.

ARALDI, R. et al. Florescimento em cana-de-açúcar. *Ciência Rural*, scielo, v. 40, p. 694 – 702, 03 2010. ISSN 0103-8478. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-84782010000300035&nrm=iso>. Citado 3 vezes nas páginas 23, 24 e 25.

ARTHUR, J. *Raspberry Pi: The Complete Guide to Raspberry Pi for Beginners, Including Projects, Tips, Tricks, and Programming*. Ingram Publishing, 2019. ISBN 9781925989694. Disponível em: <https://books.google.com.br/books?id=xOQlxwEACAAJ>. Citado na página 69.

AUDE, M. I. d. S. Estádios de desenvolvimento da cana-de-açúcar e suas relações com a produtividade. *Cienc. Rural*, scielo, v. 23, p. 241 – 248, 08 1993. ISSN 0103-8478. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-84781993000200022&nrm=iso>. Citado na página 23.

AWAD, M.; KHANNA, R. *Efficient Learning Machines*: Theories, concepts, and applications for engineers and system designers. Berkeley, CA, USA: Open Access, 2015. Citado 10 vezes nas páginas 62, 63, 64, 65, 67, 68, 94, 95, 96 e 118.

BARELLI, F. *Introdução à Visão Computacional: Uma abordagem prática com Python e OpenCV*. Casa do Código, 2018. ISBN 9788594188588. Disponível em: <https://books.google.com.br/books?id=CA5ZDwAAQBAJ>. Citado 4 vezes nas páginas 29, 57, 60 e 117.

BERNARDES, M. S.; BELARDO, G. d. C. Espaçamentos de plantio: espaçamentos para a cultura de cana-de-açúcar: Manejo nutricional da cultura da cana-de-açúcar. In: BELARDO MARCELO TUFAILE CASSIA, R. P. d. S. Guilherme de C. (Ed.). *Processos Agrícolas e Mecanização da Cana-de-açúcar*. [S.l.: s.n.], 2015. v. 1, p. 243–258. Citado na página 20.

CASTRO, S. G. Q. d. *Manejo da adubação nitrogenada em cana-de-açúcar e diagnose por meio de sensores de dossel*. Tese (Doutorado) — Universidade Estadual de Campinas, Faculdade de Engenharia Agrícola, Campinas, SP, Campinas, Brasil, 2016. Disponível em: <http://repositorio.unicamp.br/jspui/handle/REPOSIP/319248>. Acesso em: 09 abr. 2020. Citado na página 20.

CHAWLA, N. V.; JAPKOWICZ, N.; KOTCZ, A. Editorial: Special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, Association for Computing Machinery, New York, NY, USA, v. 6, n. 1, p. 1–6, jun. 2004. ISSN 1931-0145. Disponível em: <https://doi.org/10.1145/1007730.1007733>. Citado 2 vezes nas páginas 117 e 122.

CLIMATE-DATA. *Clima Barretos*. 2020. Acessado em: 2020-04-09. Disponível em: <https://pt.climate-data.org/america-do-sul/brasil/sao-paulo/barretos-4232/>. Citado na página 33.

CODIZAR, A. L.; SOLANO, G. Plant leaf recognition by venation and shape using artificial neural networks. In: *2016 7th International Conference on Information, Intelligence, Systems Applications (IISA)*. [S.l.: s.n.], 2016. p. 1–4. Citado na página 32.

CONAB. *Acompanhamento da safra brasileira de cana-de-açúcar*: Safra 2019/20. 2020. Disponível em: <http://www.conab.gov.br>. Citado na página 20.

COPPIN, B. *Inteligência Artificial*. Rio de Janeiro, RJ, Brasil: LTC, 2012. Citado 2 vezes nas páginas 30 e 31.

COSTA, D. *JAVA EM REDE: Programação Distribuída na Internet*. BRASPORT, 2008. ISBN 9788574523361. Disponível em: <https://books.google.com.br/books?id=_Fr0ooiSIYUC>. Citado 2 vezes nas páginas 38 e 109.

COSTA, D. *Java em Rede: Recursos Avançados de Programação*. BRASPORT, 2008. ISBN 9788574523699. Disponível em: <https://books.google.com.br/books?id=W4oBsv7lifMC>. Citado na página 109.

CRAIG, J. J. *Robótica*. São Paulo, SP, Brasil: Pearson Education do Brasil, 2012. Citado na página 27.

DEPOSITOPHOTOS. *Grãos de café*. 2020. Acessado em: 2020-04-09. Disponível em: <https://br.depositphotos.com/253931620/stock-photo-coffee-beans-some-hard-roasted.html>. Citado na página 92.

DEVELOPERS, G. *Arquitetura da Plataforma*. 2020. Acessado em: 2020-04-09. Disponível em: <https://developer.android.com/guide/platform?hl=pt>. Citado 2 vezes nas páginas 52 e 71.

ELMASRI, R.; NAVATE, S. B. *Sistemas de Banco de Dados*. 7. ed. São Paulo, SP, Brasil: Pearson Education do Brasil, 2018. Citado na página 112.

FAROOQUI, K.; LOGRIPPO, L.; MEER, J. de. The iso reference model for open distributed processing: an introduction. v. 27, n. 8, p. 1215 – 1229, 1995. ISSN 0169-7552. ISO Reference Model for Open Distributed Processing. Disponível em: <http://www.sciencedirect.com/science/article/pii/016975529500087N>. Citado na página 36.

FIGUEIREDO, P. et al. Alterações morfoanatômicas foliares da cana-de-açúcar na fase de estabelecimento em condições de matocompetição. *Planta Daninha*, scielo, v. 31, p. 777 – 784, 12 2013. ISSN 0100-8358. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0100-83582013000400003&nrm=iso>. Citado na página 25.

FLASK. *Flask web development, one drop at a time*. 2020. Acessado em: 2020-04-09. Disponível em: <https://flask.palletsprojects.com/en/1.1.x/>. Citado na página 111.

GIMENEZ, L. M.; MOLIN, J. P. Agricultura de precisão sob a perspectiva de seus diversos atores. *Informações Agronômicas*, International Plant Nutrition Institute - IPNI Brasil, 2018. Citado na página 20.

GOETZ, B. et al. *JAVA CONCORRENTE NA PRATICA*. Alta Books, 2008. ISBN 9788576082071. Disponível em: <https://books.google.com.br/books?id=zQQZvgAACAAJ>. Citado na página 109.

GONZALEZ, R. C.; WOODS, R. E. *Processamento Digital de Imagens*. 3. ed. São Paulo, SP, Brasil: Pearson Education do Brasil, 2010. Citado 4 vezes nas páginas 21, 29, 30 e 31.

GRIMM, R. et al. System support for pervasive applications. *ACM Trans. Comput. Syst.*, Association for Computing Machinery, New York, NY, USA, v. 22, n. 4, p. 421–486, nov. 2004. ISSN 0734-2071. Disponível em: <https://doi.org/10.1145/1035582.1035584>. Citado na página 37.

HACKENHAAR, N. M.; HACKENHAAR, C.; ABREU, Y. V. d. Robótica na agricultura. *Interações (Campo Grande)*, scielo, v. 16, p. 119 – 129, 06 2014. ISSN 1518-7012. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1518-70122015000100011&nrm=iso>. Citado 3 vezes nas páginas 26, 28 e 29.

HARTSHORN, S. *Machine Learning with Random Forests and Decision Trees*: A visual guide for beginners. [S.l.]: Fairly Nerdy, 2016. Citado 4 vezes nas páginas 65, 66, 95 e 118.

HENDRIX, J. *Raspberry Pi: The Essential Guide on Starting Your Own Raspberry Pi 3 Projects with Ingenious Tips and Tricks!* CreateSpace Independent Publishing Platform, 2016. ISBN 9781539972242. Disponível em: <https://books.google.com.br/books?id=IrIRMQAACAAJ>. Citado 4 vezes nas páginas 69, 70, 108 e 113.

HORSTMANN, C. S.; CORNELL, G. *Core Java*: Fundamentos. São Paulo, SP, Brasil: Pearson Education do Brasil, 2010. Citado na página 48.

HSIAO, J. et al. Learning sparse representation for leaf image recognition. In: *2014 IEEE International Conference on Consumer Electronics - Taiwan*. [S.l.: s.n.], 2014. p. 209–210. Citado na página 32.

HU, J. et al. A multiscale fusion convolutional neural network for plant leaf recognition. *IEEE Signal Processing Letters*, v. 25, n. 6, p. 853–857, 2018. Citado na página 31.

HU, R. et al. Multiscale distance matrix for fast plant leaf recognition. *IEEE Transactions on Image Processing*, v. 21, n. 11, p. 4667–4672, 2012. Citado 2 vezes nas páginas 60 e 119.

INPE. *Catálogo de Imagens*. 2020. Acessado em: 2020-04-09. Disponível em: <http://www.dgi.inpe.br/CDSR/>. Citado na página 34.

INPE; AMBDATA. *Mapa de Solos*. 2020. Acessado em: 2020-04-09. Disponível em: <http://www.dpi.inpe.br/Ambdata/mapa_solos.php>. Citado na página 33.

KANG, E.; OH, I. Weak constraint leaf image recognition based on convolutional neural network. In: *2018 International Conference on Electronics, Information, and Communication (ICEIC)*. [S.l.: s.n.], 2018. p. 1–4. Citado na página 21.

Khmag, A.; Al-Haddad, S. A. R.; Kamarudin, N. Recognition system for leaf images based on its leaf contour and centroid. In: *2017 IEEE 15th Student Conference on Research and Development (SCOReD)*. [S.l.: s.n.], 2017. p. 467–472. Citado na página 21.

KOPETZ, H. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Springer US, 2011. (Real-Time Systems Series). ISBN 9781441982377. Disponível em: <https://books.google.com.br/books?id=oJZsvEawlAMC>. Citado 2 vezes nas páginas 37 e 38.

LOY, M. et al. *Java Swing*. O'Reilly Media, Incorporated, 2002. (Java Series). ISBN 9780596004088. Disponível em: <https://books.google.com.br/books?id=W3HjIAduQfkC>. Citado na página 109.

LUGER, G. *Inteligência Artificial*. 6. ed. São Paulo, SP, Brasil: Pearson Education do Brasil, 2013. Citado 4 vezes nas páginas 21, 62, 67 e 118.

LUKIC, M.; TUBA, E.; TUBA, M. Leaf recognition algorithm using support vector machine with hu moments and local binary patterns. In: *2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*. [S.l.: s.n.], 2017. p. 000485–000490. Citado na página 32.

MAGRO C. R.; LACA-BUENDIA, J. P. Efeito da profundidade de plantio no perfilhamento da cana-de-açúcar. *FAZU em Revista*, FAZU em Revista, Uberaba, 2010. Citado 2 vezes nas páginas 23 e 24.

MALAVOLTA, E.; HAAG, H. P. Fisiologia. In: MALAVOLTA, E.; SEGALA, A. L.; GOMES, F. P. e. a. (Ed.). *Cultivo e Adubação da Cana-de-açúcar*. [S.l.: s.n.], 1964. v. 1. Citado na página 25.

Mouine, S.; Yahiaoui, I.; Verroust-Blondet, A. Plant species recognition using spatial correlation between the leaf margin and the leaf salient points. In: *2013 IEEE International Conference on Image Processing*. [S.l.: s.n.], 2013. p. 1466–1470. Citado na página 32.

MOZAMBANI, A. E. e. a. História e morfologia da cana-de-açúcar. In: SEGATO, S. e. a. (Ed.). *Atualização em produção de cana-de-açúcar*. [S.l.: s.n.], 2006. v. 1, p. 11–18. Citado na página 23.

MYSQL. *MySQL Documentation*. 2020. Acessado em: 2020-04-09. Disponível em: <https://dev.mysql.com/doc/>. Citado na página 118.

NASTARI, P. M. Perspectivas para o setor canavieiro: Manejo nutricional da cultura da cana-de-açúcar. In: BELARDO MARCELO TUFAILE CASSIA, R. P. d. S. Guilherme de C. (Ed.). *Processos Agrícolas e Mecanização da Cana-de-açúcar*. [S.l.: s.n.], 2015. v. 1, p. 31–34. Citado 2 vezes nas páginas 23 e 26.

NIKU, S. B. *Introdução a Robótica*: análise, controle, aplicações. Rio de Janeiro, RJ, Brasil: LTC, 2013. Citado 2 vezes nas páginas 26 e 27.

NUMPY. *NumPy: The fundamental package for scientific computing with Python*. 2020. Acessado em: 2020-04-09. Disponível em: <https://numpy.org/>. Citado na página 54.

OBE, R.; HSU, L. *PostgreSQL: Up and Running*. O'Reilly, 2012. (A practical guide to the advanced open source database). ISBN 9781449326333. Disponível em: <https://books.google.com.br/books?id=Q8jkIZkMTPcC>. Citado na página 113.

OPENCV. *OpenCV*. 2020. Acessado em: 2020-04-09. Disponível em: <https://opencv.org/>. Citado 8 vezes nas páginas 54, 55, 56, 57, 58, 59, 60 e 118.

OSROOSH, Y.; KHOT, L. R.; PETERS, R. T. Economical thermal-rgb imaging system for monitoring agricultural crops. *Computers and Electronics in Agriculture*, v. 147, p. 34 – 43, 2018. ISSN 0168-1699. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0168169917310360>. Citado na página 21.

PAJANKAR, A. *Raspberry Pi Computer Vision Programming*. Packt Publishing, 2015. (Community experience distilled). ISBN 9781784398286. Disponível em: <https://books.google.com.br/books?id=TSnCsgEACAAJ>. Citado 2 vezes nas páginas 70 e 117.

POSTGRESQL. *PostgreSQL Documentation*. 2020. Acessado em: 2020-04-09. Disponível em: <https://www.postgresql.org/docs/>. Citado na página 118.

PYTHON SOFTWARE FOUNDATION. *Python 3.8.2 documentation*. 2020. Acessado em: 2020-04-09. Disponível em: <https://docs.python.org/3/>. Citado na página 51.

RASPBERRY PI FOUNDATION. *Raspberry Pi Documentation*. 2020. Acessado em: 2020-04-09. Disponível em: <https://www.raspberrypi.org/documentation/>. Citado 6 vezes nas páginas 7, 22, 42, 44, 68 e 105.

RESENDE, A. V. d.; COELHO, A. M. Amostragem para mapeamento e manejo da fertilidade do solo na abordagem de agricultura de precisão. *Informações Agronômicas, Piracicaba*, v. 1, n. 159, p. 1–8, set. 2017. Citado na página 20.

RODRIGUES, A. de P. Prefácio: Reaprendendo a produzir cana-de-açúcar. In: BELARDO MARCELO TUFAILE CASSIA, R. P. d. S. Guilherme de C. (Ed.). *Processos Agrícolas e Mecanização da Cana-de-açúcar*. [S.l.: s.n.], 2015. v. 1. Citado na página 26.

SAUDATE, A. *REST: Construa API's inteligentes de maneira simples*. Casa do Código, 2014. ISBN 9788566250985. Disponível em: <https://books.google.com.br/books?id=uo-CCwAAQBAJ>. Citado na página 111.

SCIKIT-LEARN. *Machine Learning in Python*. 2020. Acessado em: 2020-04-09. Disponível em: <https://scikit-learn.org/stable/>. Citado 3 vezes nas páginas 61, 68 e 120.

SIAHAAN, V.; SIANIPAR, R. *OpenCV-Python with PostgreSQL for Absolute Beginners: A Hands-On, Practical Database-Driven Applications*. SPARTA PUBLISHING, 2019. Disponível em: <https://books.google.com.br/books?id=6o6vDwAAQBAJ>. Citado 2 vezes nas páginas 112 e 118.

SILVA, A. da; VIDEIRA, C.; ATLÂNTICO, C. *UML, metodologias e ferramentas CASE: linguagem de modelação UML, metodologias e ferramentas CASE na concepção e desenvolvimento de sistemas de informação*. [S.l.: s.n.], 2001. ISBN 9728426364. Citado na página 37.

SILVA, P. C. d. *As mudanças climáticas do município de Barretos*. Dissertação (Trabalho de Conclusão de Curso (graduação)) — Universidade de Brasília, 2014. Citado na página 33.

SILVEIRA, G.; BULLOCK, B. *Machine Learning: Introdução à classificação*. Casa do Código, 2017. ISBN 9788594188199. Disponível em: <https://books.google.com.br/books?id=XL46DwAAQBAJ>. Citado 2 vezes nas páginas 117 e 118.

SPEKKEN, M.; BRUIN, S. de; MOLIN, J. P. Ferramentas para sistematização de percursos de máquinas agrícolas sobre relevo declivoso. In: *Congresso Brasileiro de Agricultura de Precisão – ConBAP*. São Pedro, Brasil: [s.n.], 2014. Citado na página 20.

SQLALCHEMY. *The Python SQL Toolkit and Object Relational Mapper*. 2020. Acessado em: 2020-04-09. Disponível em: <https://www.sqlalchemy.org/>. Citado 2 vezes nas páginas 53 e 112.

SUPPA, C. *Agronegócio impulsiona mercado de máquinas e equipamentos*. 2020. Acessado em: 2020-04-09. Disponível em: <https://www.em.com.br/app/noticia/economia/2017/05/22/internas_economia,870728/agronegocio-impulsiona-mercado-de-maquinas-e-equipamentos.shtml>. Citado na página 21.

TANENBAUM, A. S.; STEEN, M. van. *Sistemas distribuídos*: princípios e paradigmas. São Paulo, SP, Brasil: Pearson Prentice Hall, 2007. Citado 3 vezes nas páginas 35, 36 e 37.

TRIMBLE. *Weedseeer System*: datasheet. 2020. Acessado em: 2020-04-09. Disponível em: <http://trl.trimble.com/docushare/dsweb/Get/Document-475151>. Citado na página 21.

ULUTURK, C.; UGUR, A. Recognition of leaves based on morphological features derived from two half-regions. In: *2012 International Symposium on Innovations in Intelligent Systems and Applications*. [S.l.: s.n.], 2012. p. 1–4. Citado na página 31.

VITTI, G. C.; OTTO, R.; FERREIRA, L. R. d. P. Nutrição e adubação da cana-de-açúcar: Manejo nutricional da cultura da cana-de-açúcar. In: BELARDO MARCELO TUFAILE CASSIA, R. P. d. S. Guilherme de C. (Ed.). *Processos Agrícolas e Mecanização da Cana-de-açúcar*. [S.l.: s.n.], 2015. v. 1, p. 177–206. Citado na página 20.

Considerações Finais

Este trabalho contribuiu para a validação de ferramentas *Open Source*, tanto dentro da área de hardware quanto de softwares.

Em projetos futuros, por meio das ferramentas utilizadas para a construção do ERACI, poderá ser possível a implementação de sistemas que tenham a capacidade de aplicar insumos de forma localizada, verificar o estresse hídrico da planta bem como o desenvolvimento da cultura, não apenas dentro do contexto de lavoura de cana-de-açúcar, mas em qualquer outra, em que a automatização traga benefícios quantitativos e qualitativos.

Outra situação inerente a este trabalho, é a possibilidade de se realizar a análise da planta ou do solo apenas por meio da visão computacional, eliminando a necessidade de aquisição de outros sensores. Este fato pode resultar na redução do custo de desenvolvimento de sistemas com finalidade equivalente, dentro do contexto da agricultura digital.