UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA MECATRÔNICA

ILDO ROSA DE SOUZA JUNIOR

AMBIENTE DE SUPORTE PARA POUSO AUTÔNOMO DE DRONES VIA RECONHECIMENTO DE IMAGENS

Joinville
2023

ILDO ROSA DE SOUZA JUNIOR

AMBIENTE DE SUPORTE PARA POUSO AUTÔNOMO DE DRONES VIA RECONHECIMENTO DE IMAGENS

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do título de bacharel em Engenharia Mecatrônica no curso de Engenharia Mecatrônica, da Universidade Federal de Santa Catarina, Centro Tecnológico de Joinville.

Orientador: Dr. Alexandro Garro Brito

Joinville
2023

Dedico este trabalho a meus pais, a minha irmã, meus cães, minha família, meus amigos, todos aqueles que de alguma forma contribuíram para o meu aprendizado ao longo da vida e à proteção de Deus, pois Ele está sempre presente em meu caminho.

AGRADECIMENTOS

Ao meu orientador, professor Alexandro, sou grato pela orientação ao longo do desenvolvimento deste trabalho.
Aos meus amigos Arthur Elzinga, Gustavo Serafim e Pedro H. Melo pela colaboração no decorrer deste projeto.
À todos os professores que contribuíram significativamente para minha jornada acadêmica, meu sincero agradecimento.
À minha família, pelo constante apoio e incentivo ao longo dessa jornada, expresso minha profunda gratidão.
À todos aqueles que, mesmo não mencionados, de alguma forma contribuíram para o meu aprendizado ao longo da vida e para o desenvolvimento deste trabalho.

Lembre-se que as pessoas podem tirar tudo de você, menos o seu conhecimento (Albert Einstein).

RESUMO

A maioria dos drones disponíveis no mercado, que são comercializados a preços acessíveis, não possuem um assistente de pouso devido ao aumento de custo ao adicionar câmeras e sistemas adicionais, os drones sem essa função podem se tornar um problema quando pessoas inexperientes tentam pousar o drone e acabar cometendo erros que possam comprometer seu funcionamento. Considerando isso, este trabalho aborda a implementação de um ambiente de suporte para pouso autônomo de drones via reconhecimento de imagens. Foi desenvolvido um método de pouso via processamento de imagem em tempo real por um notebook através de classificadores Haar Cascade em conjunto com a biblioteca OpenCV, implementado na linguagem Python, utilizando um ESP32-CAM e um roteador INTELBRAS WRN 240, que são responsáveis pela captura, transmissão e recebimento de imagens, um hardware desenvolvido para automatizar o controle do drone e, assim, aplicado ao drone modelo Syma X8HG para estabelecer comandos de pouso para o drone. Serão examinadas as características dos drones ao longo do tempo, detalhando o método e o processo de detecção de objetos, e os softwares utilizados no processo.

Palavras-chave: Pouso de Drone. Detecção de alvo. Imagem Tempo Real. Processamento de imagem.

ABSTRACT

Most drones available on the market, which are sold at affordable prices, do not have a landing assistant due to the increase in cost when adding additional cameras and systems, drones without this function can become a problem when inexperienced people try to land the drone and end up making mistakes that could compromise its functioning. Considering this, this work addresses the implementation of a support environment for autonomous drone landing via image recognition. A landing method was developed via real-time image processing by a notebook using Haar Cascade classifiers in conjunction with the OpenCV library, implemented in the Python language, using an ESP32-CAM and an INTELBRAS WRN 240 router, which are responsible for capturing , transmission and reception of images, hardware developed to automate drone control and, thus, applied to the Syma X8HG model drone to establish landing commands for the drone. The characteristics of drones over time will be examined, detailing the method and process of detecting objects, and the software used in the process.

Keywords: Drone landing. Target detection. Real Time Image. Image processing.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS E SÍMBOLOS

UAVs	Unmanned Aerial Vehicles
LIDAR	Light Detection and Ranging
FPV	First Person View
HSV	Hue, Saturation e Value
GCSs	Ground Control Stations
OpenCV	Open Source Computer Vision Library
LASC	Laboratório de Automação e Sistemas de Controle
UFSC	Universidade Federal de Santa Catarina
YOLO	You Only Look Once
R-CNN	Region-based Convolutional Neural Network

SUMÁRIO

1 INTRODUÇÃO

Os drones, também conhecidos como Unmanned Aerial Vehicles (UAVs), inicialmente criado para fins militares, atualmente auxiliam atividades comerciais de curto e longo alcance, coleta de dados em lugares inacessíveis, lazer e outras aplicações. Esses veículos podem ser equipado com câmeras, mísseis e/ou cargas e permitem coletar vários tipos de dados digitais, visuais e ambientais, podendo variar em tamanho, desde centímetros de diâmetro até envergaduras alcançando 20 metros de comprimento, como no caso do modelo *MQ-9 Reaper*, e com base no número de hélices que possuem são classificados em rotor único, multirotor ou de asa fixa.

A tecnologia de pouso de um drone pode ser por operação manual (ALEOTTI et al., 2017), operação remota, por Light Detection and Ranging (LIDAR) (BAREISS; BOURNE; LEANG, 2017), First Person View (FPV) (PFEIFFER et al., 2022; TEZZA; LAESKER; ANDUJAR, 2021) e por processamento de imagem (KHADKA et al., 2020). O pouso por processamento de imagem é o reconhecimento de uma imagem analizada por computação para identificar o local de pouso.

A implementação de um sistema de pouso para drones que ainda não tem a função de pouso por processamento de imagem pode auxiliar o pouso em locais de difícil acesso, facilitar o manuseio por usuários iniciantes e diminuir o risco de danos durante o pouso. A ampliação do acesso ao uso do drone, assim como evitar danos ao mesmo, é uma preocupação que vem crescendo ao longo do tempo na literatura da área (BAREISS; BOURNE; LEANG, 2017; BEN-MOSHE; KELLER, 2022; KHADKA et al., 2020; CAMPOY; MONDRAGÓN; OLIVARES-MENDEZ, 2013).

Para tanto, vê-se que Pereira (2017) executa um projeto de um robô terrestre que rastreia um alvo, limitado a um plano de duas dimensões (2D), utilizando inteligência artificial no processamento de imagem. Já Quiles (2018) projetou um sistema de identificação de veiculos aéreos não tripulados por processamento de imagem para drones. Os trabalhos de ambos exploram a possibilidade de implementação de sistemas de processamento de imagens para reconhecimento de um alvo.

Desse modo, nota-se que Costa et al. (2019) aborda conceitos como dinâmica do corpo rígido, cinemática dos rotores, controle de altitude e estabilização angular de um UAV equipado com rotores inclináveis e câmera orientável. Além disso, Campoy, MondragÓn e Olivares-Mendez (2013) realizou a implementação de um sistema de pouso autônomo para um helicóptero elétrico SR20 utilizando as informações visuais fornecidas por uma câmera para controlar a altitude e a posição junto a um controlador fuzzy. O trabalho de ambos direciona a possibilidade do pouso autônomo por processamento de imagens.

Nesse sentido realizou-se a implementação de um sistema de pouso autônomo de um drone por processamento de imagem, um ESP32-CAM, acoplado ao drone modelo Syma X8HG e utilizando uma rede Wireless Fidelity (WIFI), a câmera do ESP32-CAM envia as imagens para um servidor que a deixa disponivel para acesso remoto, o que possibilita o processamento da imagem por métodos de detecção de objetos utilizando a biblioteca OpenCV em Python estabelecendo, assim, um comando de pouso para o drone.

1.1 OBJETIVOS

Para resolver a problemática de pouso autônomo do drone, propõe-se os seguintes objetivos.

1.1.1 Objetivo Geral

O objetivo principal deste trabalho é desenvolver um sistema de pouso para o drone baseado em processamento de imagem em tempo real.

1.1.2 Objetivos Específicos

Os seguintes objetivos específicos foram propostos para este trabalho:

a) Desenvolver um método para recebimento da imagem;
b) Processar a imagem aplicando um método de detecção de objetos;
c) Estabelecer comandos de pouso para o drone.

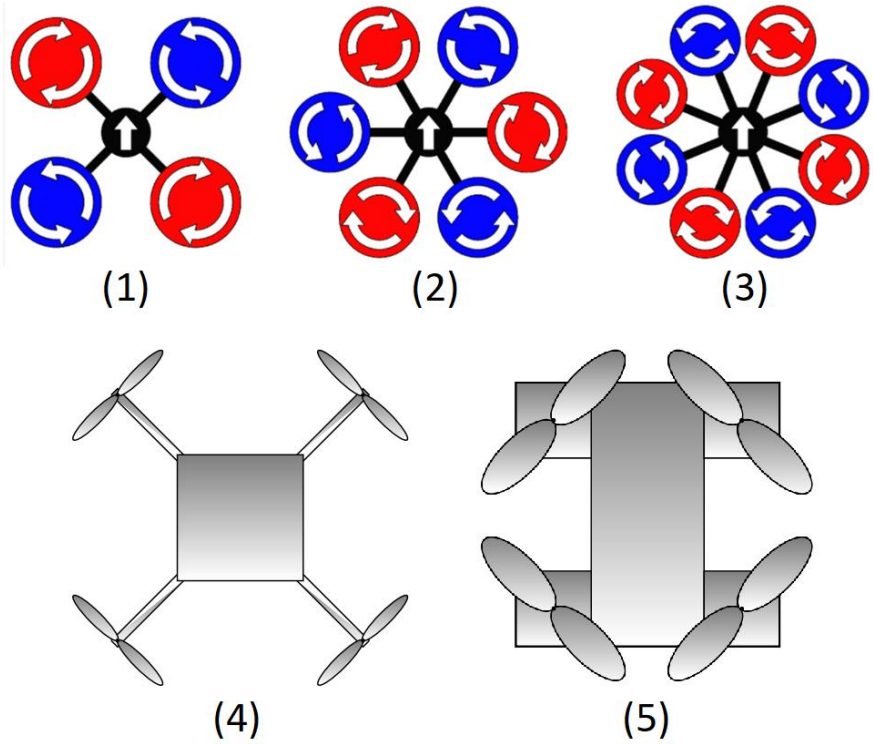
2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será feita uma revisão sobre alguns fundamentos teóricos necessários que contribuem no desenvolvimento do sistema de pouso para o drone. Analisar as características dos drones multirotor com rotores fixos nos permite conhecer o que vamos controlar, a visão computacional permitirá que o drone em conjunto com a estação de controle em solo capture e interprete as informações visuais no ambiente estabelecendo, assim, um comando automatizado de pouso do computador para o drone.

2.1 DRONES MULTIROTOR COM ROTORES FIXOS

Os drones multirotor com rotores fixos tem se consolidado no mercado, sobretudo no cenário recreativo, devido aos preços competitivos gerados pela produção mais simples e diversificada dos mesmos (PINA, 2022). Os drones mais comuns encontrados no mercado atualmente possuem quatro (quadricópteros), seis (hexacópteros) ou oito (octacópteros) rotores e seu corpo central pode ser no formato em X ou em H conforme, a Figura 1.

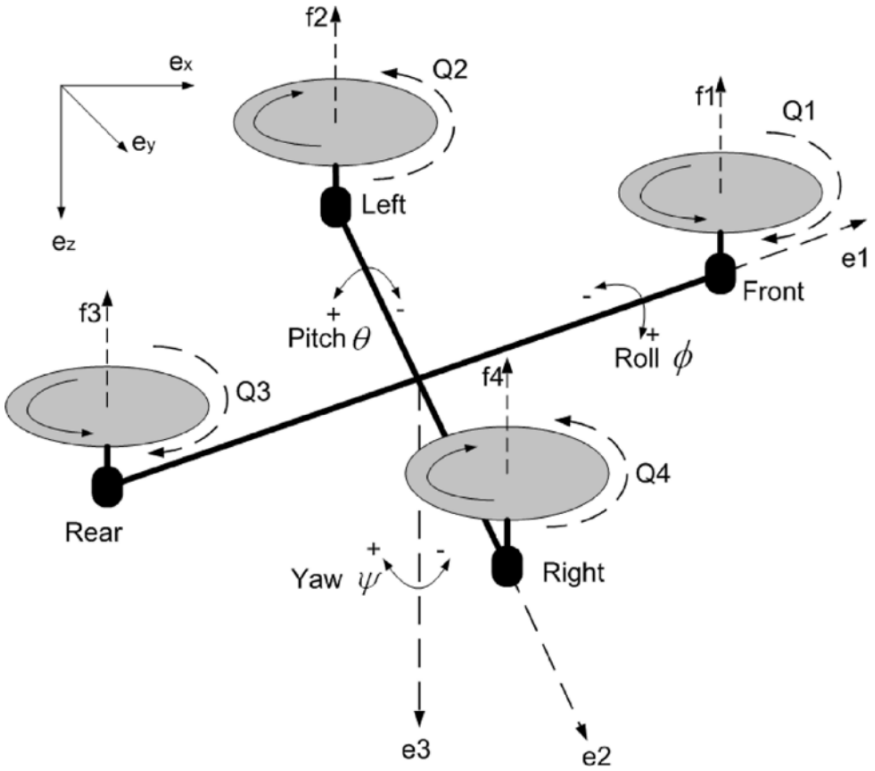
Figura 1 – Configurações dos multirotores convencionais. (1) Quadricópteros. (2) Hexacópteros. (3) Octacópteros. (4) Estrutura no formato em X. (5) Estrutura no formato em H.



Fonte: Costa et al. (2019) e Kotarski et al. (2021).

A modelagem das forças e torques reativos de um drone, com corpo central rígido no formato em X e equipado com quatro rotores, podem ser vistas na Figura 2. A variação relativa da velocidade de cada rotor permite a alteração do torque dos mesmos, possibilitando movimentos para frente, trás, esquerda, direita, subida e descida. O conjunto de rotores esquerdo e direito giram no sentido inverso do conjunto de rotores frente e trás para evitar o desvio de guinada gerado pelos torques reativos (TAYEBI; MCGILVRAY, 2006).

Figura 2 – Forças e momentos que atuam sobre um quadricóptero.



Fonte: Tayebi e McGilvray (2006).

Os drones multirotor são fáceis de pilotar comparados com drones de asa fixa por apresentar maior flexibilidade de manobra, realizam decolagens e aterrissagens verticais, pairam durante o voo, são projetados para dobrar e caber em caixas menores e tem o menor custo comparado a outros tipos de drones (PINA, 2022). No entanto, tem-se a limitação de tempo de voo em função da capacidade da bateria, são vulneráveis aos ventos fortes devido sua aerodinâmica, também não consegue-se atingir altas velocidades como os de asas fixas (KOTARSKI et al., 2021).

2.2 VISÃO COMPUTACIONAL

A visão computacional pode ser definida como um campo científico dedicado à extração de informações de imagens digitais. Essas informações podem variar desde a identificação e medições espaciais para navegação até aplicações de

realidade aumentada. Outra perspectiva de definição envolve suas aplicações práticas, destacando a construção de algoritmos capazes de compreender o conteúdo das imagens para usos diversos (KRISHNA, 2017).

Através da visão computacional, torna-se possível mensurar o espaço ao redor de um robô e construir um mapa do seu ambiente, também é viável rotular objetos na imagem, categorizar toda a cena e analisar elementos como pessoas, ações, gestos e rostos. A aplicação da visão computacional é vasta, abrangendo áreas como efeitos especiais, modelagem urbana 3D, reconhecimento de cena, detecção facial, reconhecimento óptico de caracteres, carros autônomos, checkout automático, interação baseada em visão, realidade aumentada, realidade virtual e várias outras aplicações (KRISHNA, 2017).

A compreensão da visão computacional depende de um conhecimento sólido das câmeras e do processo físico de formação de imagens para obter inferências simples a partir de valores de pixels individuais, combinar as informações disponíveis em múltiplas imagens em um todo coerente, impor alguma ordem a grupos de pixels para separá-los uns dos outros ou inferir informações de forma e reconhecer objetos usando informações geométricas ou técnicas probabilísticas (FORSYTH; PONCE, 2011).

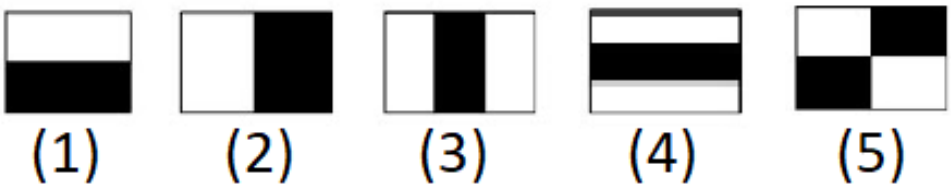
2.3 DETECÇÃO DE OBJETOS

De acordo com Soo (2014), devido a detecção de objetos a partir de imagens utilizando características do mesmo consumirem um alto processamento durante a realização dos cálculos, apenas trabalhando com os valores de pixels RGB contidos na imagem, tornou-se inviavel sua aplicação em sistemas simples. Para tanto, Viola e Jones (2001) propuseram um método eficaz de detecção de objetos usando classificadores em cascata baseados em recursos Haar.

Os recursos Haar que podem ser vistos na Figura 3, facilitam a descoberta de bordas ou linhas na imagem, ou destacam áreas onde há uma mudança repentina nas intensidades dos pixels. O cálculo do haar consiste em determinar a diferença entre a soma das intensidades dos pixels na região mais escura e na região mais clara, tornando-se o nosso núcleo convolucional. Para tornar o processo de soma das intensidade dos pixels em regiões retangulares mais rápido, introduziu-se a imagem integral. Este método utiliza apenas quatro pixels nos cálculos de soma, independente do tamanho da área da imagem a ser calculada, como mostra a Figura 4 (VIOLA; JONES, 2001).

Para selecionar os recursos calculados é utilizado o Adaboost, algoritmo idealizado em (FREUND; SCHAPIRE, 1997) onde ele encontra o melhor limite que classificará os objetos em positivos e negativos, ou seja, irá atualizar os pesos das

Figura 3 – Exemplos de recursos Haar. (1) Recurso de aresta horizontal. (2) Recurso de aresta vertical. (3) Recurso de linha vertical. (4) Recurso de linha horizontal. (5) Recurso quadrirretângulo.



Fonte: Elaborado pelo autor (2023).

Figura 4 – Imagem integral e a aplicação a recursos semelhantes a Haar.

24	46	37	24	33	27
8	11	20	31	6	35
8	44	33	9	25	38
14	36	19	12	24	20
18	37	33	15	9	46
48	7	30	30	43	44

(a) Imagem original

24	70	107	131	164	191
32	89	146	201	240	302
40	141	231	295	359	459
54	191	300	376	464	584
72	246	388	479	576	742
120	301	473	594	734	944

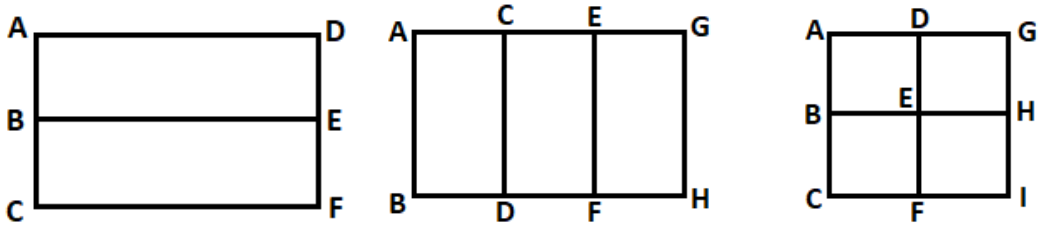
(b) Imagem integral

24	46	37	24	33	27
8	11	20	31	6	35
8	44	33	9	25	38
14	36	19	12	24	20
18	37	33	15	9	46
48	7	30	30	43	44

(c) A soma da área laranja na imagem original se dá pela soma dos 12 números, resultando em 283

24	70	107	131	164	191
32	89 A	146	201	240	302 B
40	141	231	295	359	459
54	191	300	376	464	584
72	246 C	388	479	576	742 D
120	301	473	594	734	944

(d) A soma da área laranja na imagem integral se dá pela equação A+D-B-C= 89+742-302-246= 283



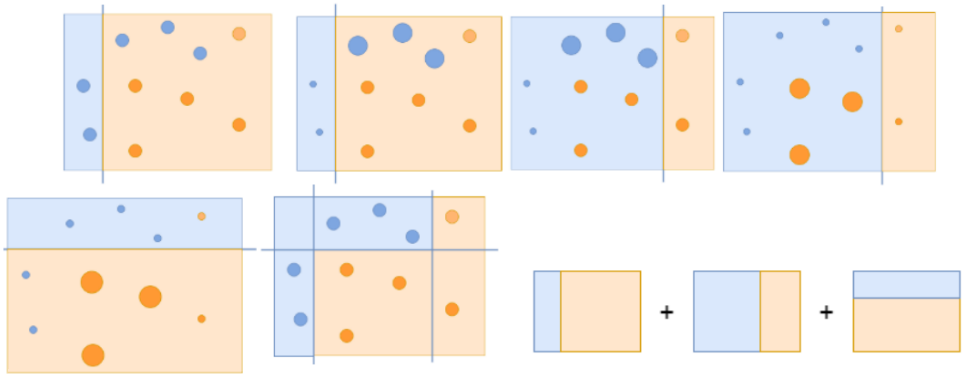
(e) Recurso de 2 retângulos (f) Recurso de 3 retângulos (g) Recurso de 4 retângulos

Fonte: Elaborado pelo autor (2023).

imagens até obter o menor erro em seu conjunto de características conforme a Figura 5. Em virtude dos classificadores fracos sozinhos conseguirem apenas classificar corretamente um número mínimo de imagens, é aplicado o classificador forte, que

se trata da combinação linear entre os melhores classificadores fracos de recursos (FORSYTH; PONCE, 2011).

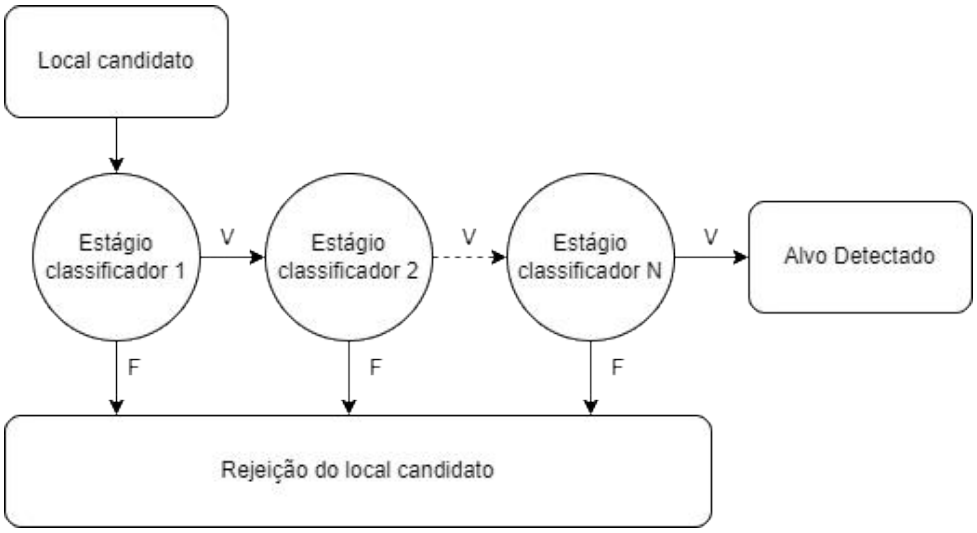
Figura 5 – Adaptive Boosting.



Fonte: Rahmad et al. (2020).

O classificador forte capaz de detectar o alvo é baseado no método de classificação em cascata, conforme ilustrado na Figura 6. Esse classificador em cascata é composto por uma sequência de estágios de classificadores, onde cada estágio consiste em um conjunto de classificadores fracos. Durante o processo, ao passar a janela de detecção sobre uma região da imagem rotulamos-a como negativa se caso não passe em algum dos estágios classificadores, ou como positiva se passar por todos os estágios classificadores assim localizando o alvo (SONKA; HLAVAC; BOYLE, 2014).

Figura 6 – Detecção cascata aplicada a cada local da subjanela da imagem analisada.



Fonte: Elaborado pelo autor (2023).

Os classificadores simples de estágio inicial rejeitam os locais candidatos menos prováveis de deteção, mantendo a avaliação de um falso negativo muito baixo. Classificadores complexos de estágio posteriores eliminam mais falsos positivos e a probabilidade de invocar esses classificadores diminui de acordo com a quantidade

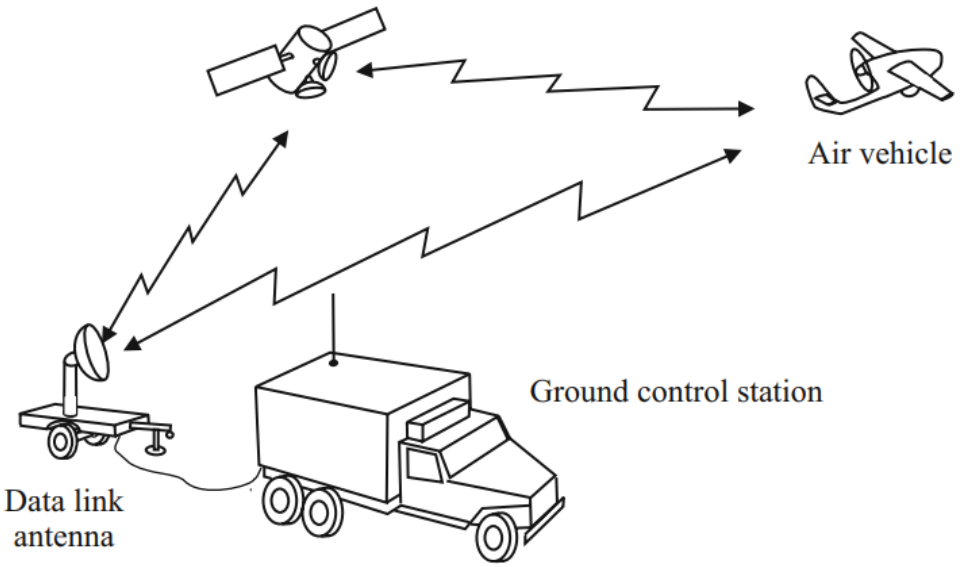
de estágios classificadores presentes na cascata, enquanto ainda definido para não rejeitar verdadeiros positivos. As detecções múltiplas devem ser pós-processadas para produzir uma única detecção final por local identificado na imagem (SONKA; HLAVAC; BOYLE, 2014).

2.4 ESTAÇÃO DE CONTROLE EM SOLO

As Estações de Controle de Solo (GCSs) são dispositivos de hardware e software que permitem monitorar, pilotar e comandar a aeronave não tripulada. Embora o nome sugira que essas estações sejam estacionárias, elas podem ser usadas em qualquer lugar, seja no solo, no mar ou no ar. A importância do GCS é inegável, pois permite aos operadores monitorar a rota, detectar erros, visualizar informações e respostas do UAV em tempo real e agir comforme seja necessário (COLOMINA; MOLINA, 2014).

Um sistema UAV genérico é composto por veículos aéreos, uma ou mais GCS, os dados a serem transmitidos e o enlace de dados conforme a Figura 7. Em alguns sistemas, a GCS pode estar contida em uma maleta, composta por um controle remoto e algum display, aprimorada por microprocessadores ou hospedados em um laptop robusto. No outro extremo temos a GCS localizadas em estruturas fixas a milhares de quilômetros de onde o veículo aéreo está voando e utilizando retransmissores de satélite para manter as comunicações com o veículo aéreo (FAHLSTROM; GLEASON, 2012).

Figura 7 – Sistema genérico UAV.



Fonte: Fahlstrom e Gleason (2012).

A GCS envia comandos para o UAV a fim de realizar ações como alterar sua direção de vôo ou direcionar uma câmera para um alvo e recebe dados que contém informações sobre o estado da aeronave, posição atual e transmissão de

vídeo em tempo real (JOVANOVIC; STARCEVIC, 2008). Em geral, as informações da imagem obtida precisam de processamento para compensar as condições externas e identificar a localização e dimensão de alvos de interesse, assim, analizando e extraindo informações e transmitindo aos operadores (NATARAJAN, 2001).

2.5 OPENCV

OpenCV (Open Source Computer Vision Library) é uma biblioteca de visão computacional de código aberto que é utilizada para uma variedade de operações de processamento de imagens e vídeos. Essa ferramenta é projetada para resolver problemas de visão computacional, abrangendo desde o tratamento e processamento de imagens até o treinamento e aplicação de métodos de reconhecimento facial e rastreamento de objetos (MINICHINO; HOWSE, 2015).

A biblioteca é escrita em C e C++ e foi projetado para ser multiplataforma, podendo ser executada em computadores (Linux, Windows, Mac OS, etc) e em dispositivos móveis (Android, Maemo, iOS). Há desenvolvimento ativo em interfaces para Python, Ruby, Matlab e outras linguagens. Um dos objetivos do OpenCV é fornecer uma infraestrutura de visão computacional simples de usar que ajude as pessoas a criar rapidamente aplicativos de visão bastante sofisticados. A biblioteca OpenCV contém mais de 500 funções que abrangem muitas áreas da visão, incluindo inspeção de produtos de fábrica, imagens médicas, segurança, interface de usuário, calibração de câmera, visão estéreo e robótica (BRADSKI; KAEHLER, 2008).

2.6 CASCADE TRAINER GUI

Cascade trainer GUI é uma ferramenta projetada por Amin Ahmadi utilizada para treinar, testar e melhorar os modelos de classificadores em cascata. Diferencia-se por compor uma interface gráfica para definir os parâmetros e facilitar o uso da ferramenta OpenCV para treinamento e teste de classificadores. Possui apenas versão para Windows e para treinar o classificador deve-se fornecer uma pasta com duas subpastas chamadas ”p” e ”n” que refere-se as imagens positivas e negativas (PHASE; PATIL, 2019).

3 METODOLOGIA

Para desenvolver o pouso autônomo do drone por reconhecimento de imagem, tem-se a divisão de cada objetivo específicos em partes menores para poder ter-se um controle melhor sobre a situação. O primeiro objetivo de desenvolver um método para recebimento da imagem se dividirá nas seguintes etapas: Capturar a imagem e, em seguida, transmitir a imagem para um servidor/computador.

Para o segundo objetivo de processar a imagem aplicando um método de detecção de objetos propõe-se as seguintes etapas: Aplicar um tratamento adequado nas imagens coletadas, realizar o processo de treinamento dos classificadores de detecção e executar testes para analisar seu desempenho. O terceiro objetivo de estabelecer o comando de pouso para o drone propõe-se as seguintes etapas: Conseguir mandar comandos simples para o drone e em seguida desenvolver uma lógica que integre todos os objetivos específicos do trabalho.

O drone que serviu como base para o desenvolvimento e implementação do sistema de pouso autônomo por reconhecimento de imagem foi disponibilizado pelo Laboratório de Automação e Sistemas de Controle (LASC) da Universidade Federal de Santa Catarina (UFSC). O modelo do drone é Syma X8HG, que devido à sua capacidade de voo estável e à facilidade de integração de componentes adicionais, permite a realização de experimentos e testes de sistemas autônomos. A Figura 8 mostra o drone em questão.

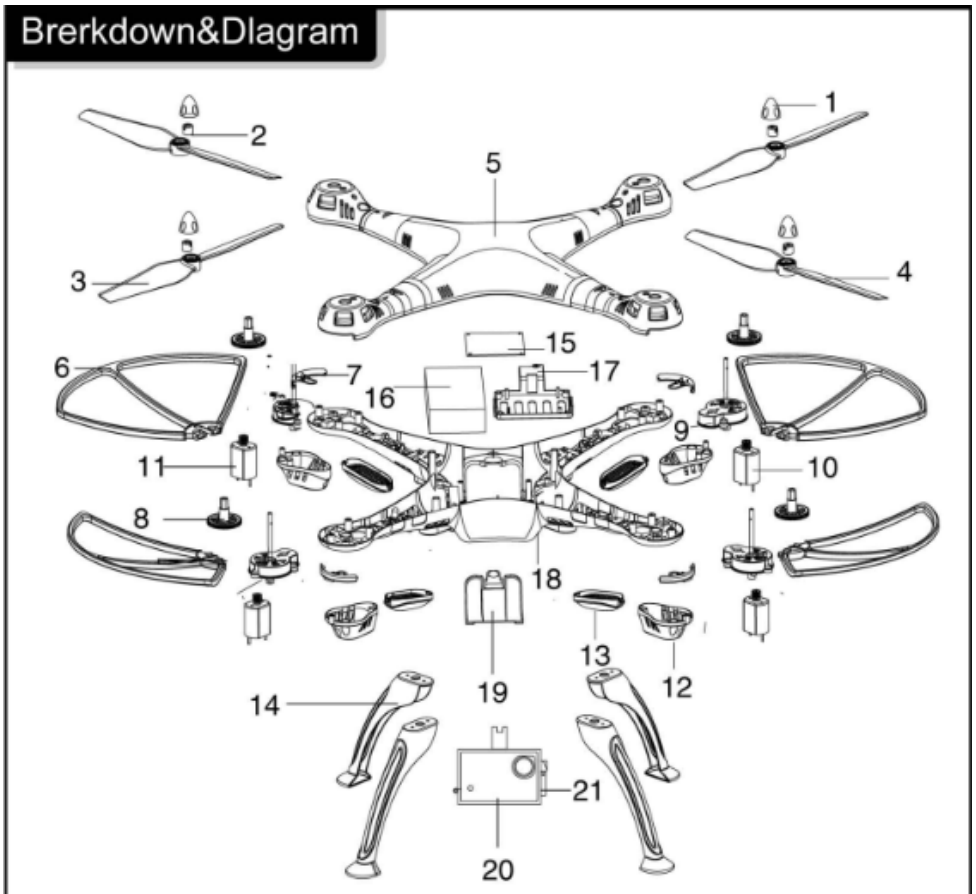
Figura 8 – Drone SYMA X8HG.

Fonte: Elaborado pelo autor (2023).

3.1 SYMA X8HG

O drone Syma X8HG é um quadricóptero caracterizado pela sua estrutura em formato de "X", essa configuração proporciona à aeronave flexibilidade, agilidade e capacidade de resistir a ventos mais fortes, contém um estabilizador de direção por um giroscópio de 6 eixos e é possivel realizar voos tanto em ambientes internos quanto externos. A estrutura utiliza designs modulares conforme a Figura 9, tornando simples a instalação, o reparo e manutenção.

Figura 9 – Diagrama e detalhamento do drone.



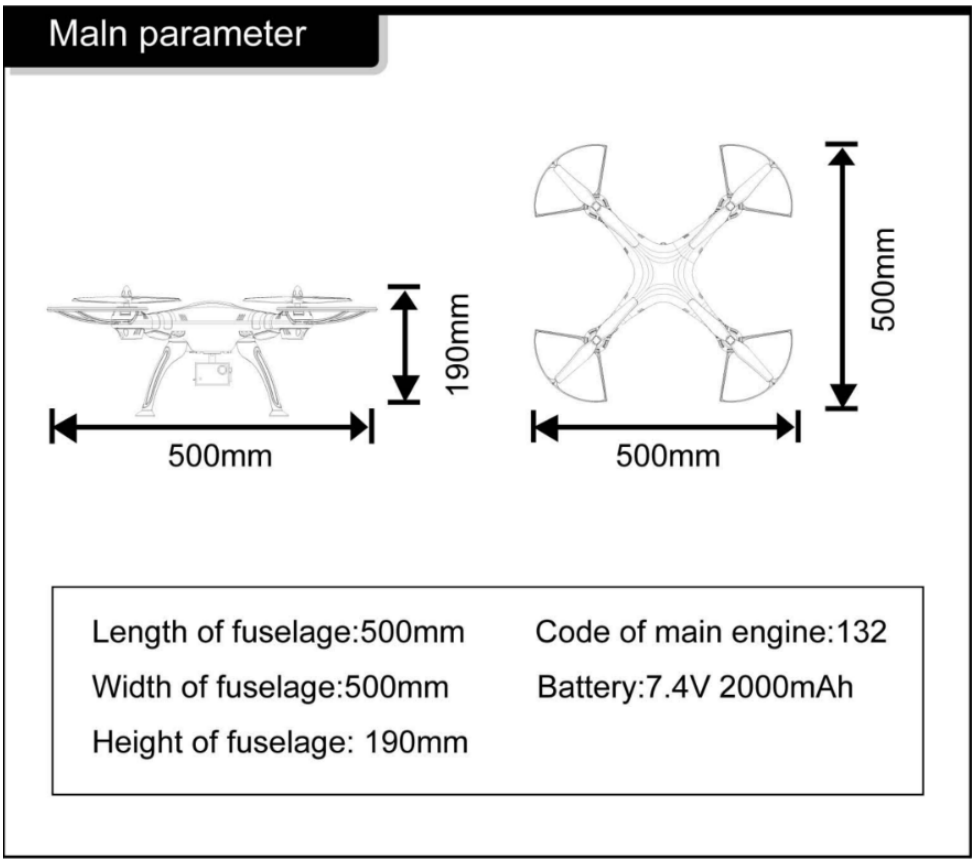
Code	Description	Quantity	Code	Description	Quantity	Code	Description	Quantity
1	Blade cover	4	8	Cear	4	15	Receiver board	1
2	Blade lockstitch	4	9	Main stand	4	16	Battery	1
3	Reversing blades	2	10	Reversing motor	2	17	Base of Dash Receiver	1
4	Rotating blades	2	11	Rotating motor	2	18	Lower body	1
5	Upper body	1	12	Motor cover	4	19	Battery cover	1
6	Protecting frames	4	13	Light boards	4	20	Camera	1
7	Ornament part	4	14	Landing skids	4	21	Camera hanger	1

Fonte: SYMATOYS (2023).

O drone é pilotado através de um controle remoto de 2.4GHz que proporciona um alcance de 70 metros, é equipado com uma bateria de lítio 7.4V/2000mAh que oferece um tempo de voo razoável, geralmente em torno de 7-10 minutos, dependendo

das condições de voo e do uso da câmera, enquanto o tempo de recarga da bateria é inferior a 200 minutos. Possui uma câmera de alta definição (HD) que permite a captura de fotos e vídeos aéreo e também conta com protetores de hélices que ajudam a evitar danos em caso de colisões. A Figura 10 mostra as dimensões do drone.

Figura 10 – Dimensões do drone.



Fonte: SYMATOYS (2023).

Esse modelo de drone conta com proteção contra baixa tensão que entra em ação quando a bateria está fraca assim limitando o drone a uma altura de segurança. Possui proteção contra sobrecorrente ativada em caso de impacto ou obstrução das pás e também possui uma função de calibração de nível que permite ajustar o drone a uma superfície nivelada garantindo um voo estável. Todas essas características emitem códigos luminosos pelas quatro luzes indicadoras localizadas na parte inferior do drone com intuito de informar o status do sistema.

Este drone também é equipado com ajuste de altura automático possibilitando um voo pairado, ou seja, que mantém a aeronave na altura desejada após soltar o joystick de aceleração tornando mais fácil para iniciantes controlar o drone. Possui também o modo *headless* que simplifica o controle ao não precisar se preocupar com a orientação frontal do drone, ou seja, independentemente da direção para a qual o drone está apontando os comandos de controle se baseiam na sua perspectiva, o que torna a pilotagem mais intuitiva e menos complexa.

3.2 RECEBIMENTO DE IMAGEM

A transmissão e recepção de imagens são áreas importantes da transmissão de dados, especialmente no contexto dos sistemas de comunicação sem fio. Com o avanço da tecnologia, a transmissão de imagens em tempo real está se tornando cada vez mais popular em diversas aplicações, como vigilância, monitoramento de tráfego e telemedicina. Nesse sentido, é importante garantir a qualidade e a confiabilidade da recepção dessas imagens, a fim de garantir a veracidade das informações transmitidas e possibilitar uma tomada de decisão adequada.

Para garantir a qualidade e a confiabilidade da captura, transmissão e recepção dessas imagens foram impostas restrições no ambiente de trabalho a fim de conseguir escolher algo que resolva o problema e seja viável. A primeira restrição foi que o máximo de distância que o drone pode alcançar é 10 metros, a segunda restrição é que o ambiente onde o drone é controlado seja fechado e a terceira restrição é que o processamento de imagem será feito pela estação de controle em solo. Diante das condições, foi feito uma pesquisa de dispositivos conforme mostrado na Tabela 1.

Tabela 1 – Tabela comparativa de dispositivos.

Características	ESP32-CAM	RS282	Rasp Pi/módulo de câmera
Transmissao	Wifi	RF(5.8GHz)	Wifi
Distância	50m	1km	50m
Processamento	Baixo	Médio	Alto
Qualidade	2MP	800TVL	8MP
Receptor	Não	Sim	Não
Peso	7g	20g	13g
Preço	Baixo	Alto	Alto

Fonte: Elaborado pelo autor (2023).

Levando em conta que o drone Syma X8HG tem um controlador de 2,4GHz com largura de banda entre 27-72MHz, o peso da câmera do drone junto com o suporte é de 48.8g e ele possui um alcance máximo de controle de 70 metros, analisando a tabela, optou-se por escolher o ESP32-CAM pois é a opção mais acessível em termos de custo-benefício e é ideal para aplicações em que a captura e transmissão de imagens são sua principal função.

3.2.1 Captação e transmissão de imagem

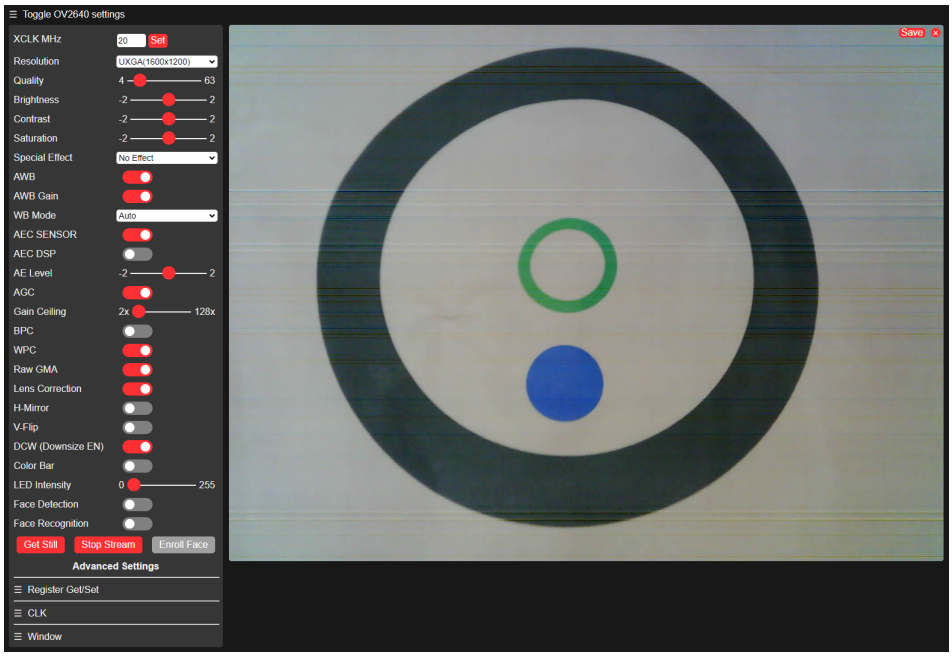
O ESP32-CAM é equipado com uma câmera OV2640 de 2 megapixels, capaz de capturar imagens coloridas em até resolução UXGA (1600x1200 pixels) e suporta gravação de vídeo em resolução reduzida devido a latência gerada. É programável com a linguagem de programação Arduino e os principais passos a serem seguidos na programação são a inicialização da câmera, configuração do ponto de acesso,

configuração da câmera em geral e também outras funções da câmera como captura da imagem armazenando ou transmitindo-as dependendo dos requisitos do projeto.

O ESP32-CAM pode ser configurado para transmitir a imagem via Wi-Fi para um servidor, aplicativo móvel ou plataforma de nuvem, portanto, aproveitaremos desta funcionalidade para hospedar um servidor Web de streaming de vídeo que pode ser acessado localmente. Para isso, utilizou-se como base um exemplo de Camera Web server disponibilizado na plataforma do Arduino IDE e foi feito algumas modificações para que conseguíssemos executá-lo. Os arquivos utilizados para a gravação do ESP32-CAM e o passo a passo seguidos estão no disponíveis no Apendice github.

Ao inserir o endereço IP gerado pelo ESP32-CAM no navegador a partir de qualquer dispositivo, é possivel acessar o servidor web que nos oferece a capacidade de alterar as configurações da câmera, como resolução, taxa de quadros, balanço de branco e muito mais conforme mostrado na Figura 11, criando uma flexibilidade para adaptar a câmera às condições específicas do ambiente. O acesso via navegador web facilita a integração com outros sistemas que necessitam das informações provenientes do WebServer gerado pelo ESP32-CAM como streaming de vídeo em tempo real.

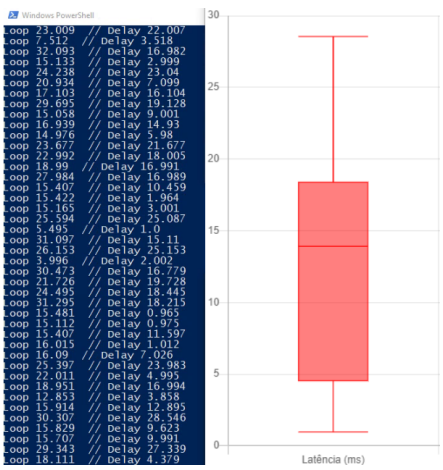
Figura 11 – Tela do servidor Web programado na ESP32-CAM.



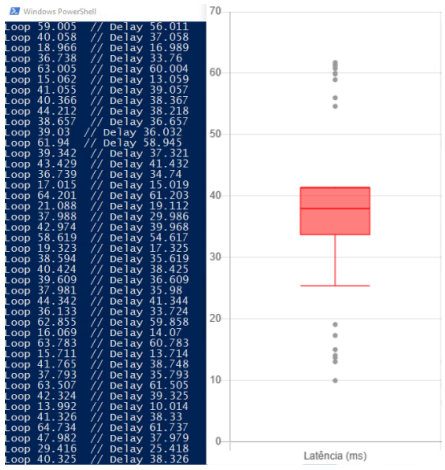
Fonte: Elaborado pelo autor (2023).

A transmissão de vídeo em tempo real pode enfrentar latência devido à compressão, codificação, decodificação, o tamanho da resolução ou a alta taxa de quadros do video. Além disso, quanto mais distante o ESP32-CAM estiver do ponto de acesso, maior pode ser a latência devido à perda de pacotes ou sinal fraco, podendo se tornar um problema em aplicações que exigem respostas instantâneas no caso de pousar um drone remotamente. A Figura 12 nos mostra a latência introduzida pelas configuração padrão do WebServer em diferentes resoluções.

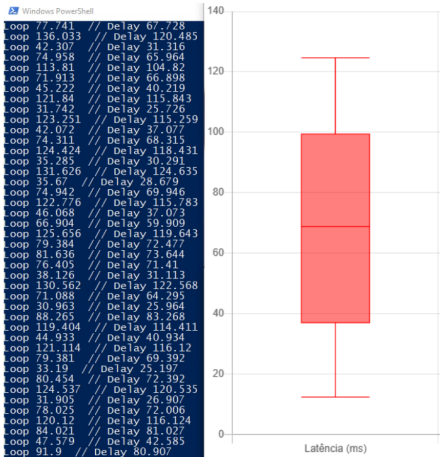
Figura 12 – Latência de captura, transmissão e recepção de imagens em várias resoluções.



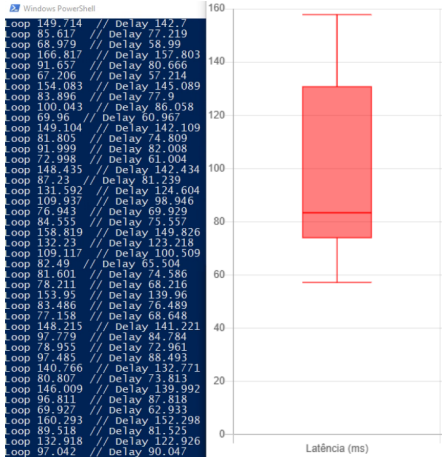
(a) Latência mediana de 13.91ms na resolução 320x240



(b) Latência mediana de 37.98ms na resolução 640x320



(c) Latência mediana de 68.85ms na resolução 1280x720



(d) Latência mediana de 83.4ms na resolução 1600x1200

Fonte: Elaborado pelo autor (2023).

As informações das imagens foram coletadas a uma distância de 50cm entre o ESP32-CAM e o ponto de acesso, a captura dos dados de latência em milisegundos foram adquiridos através da impressão dos mesmos no terminal, sendo possível perceber a presença da latência. As resoluções menores como 320x240 e 640x480 mostraram ter uma latência mediana de 13.91ms e 37.98ms, respectivamente, enquanto as resoluções maiores como 1280x720 e 1600x1200 apresentaram uma maior latência mediana sendo 68.85ms e 83.4ms.

No contexto do projeto, a resolução de 320x240 parece ser a mais adequada devido as resoluções menores gerarem menos dados a serem transmitidos, o que requer menos poder de processamento para a captura e transmissão das imagens. Essa escolha permite que o sistema opere de maneira ágil e responsiva, reduzindo a latência e garantindo uma transmissão de dados eficiente e estável, contribuindo para a execução da aplicação em tempo real.

3.3 PROCESSAMENTO DE IMAGEM

O processamento de imagem é uma disciplina fundamental no campo da visão computacional que se concentra na manipulação, análise e interpretação de imagens digitais. Essa área possibilita que sistemas de computador compreendam características visuais, como cores, formas, texturas e padrões, e tomem decisões com base nessas informações. Desempenhando um papel crucial em diversas aplicações, desde o aprimoramento da qualidade visual de fotografias até a detecção de objetos em vídeos em tempo real.

Na área de drones e UAVs, através de câmeras e sensores, o processamento de imagem permite o mapeamento em tempo real do ambiente auxiliando nas decisões de navegação, a identificação de problemas de saúde em plantas, como doenças e deficiências nutricionais, a detecção de danos e desgaste em infraestruturas críticas, bem como a localização de locais seguros para pousar, entre outras aplicações. Portanto entende-se que o processamento de imagem é uma tecnologia habilitadora que torna os drones e UAVs versáteis e valiosos em uma ampla gama de aplicações.

Nesse trabalho iremos utilizar o processamento de imagem para identificar o local de pouso do drone, portanto, nota-se que Carvalho et al. (2022) executa uma análise comparativa entre os principais algoritmos de detecção facial Haar Cascade, HOG, CNN, YOLO e DeepFace em um ambiente controlado com 1000 imagens sendo 600 positivas e 400 negativas. Dado o enfoque deste trabalho ser na detecção de objetos, não podemos presumir que os resultados apresentados por Carvalho et al. (2022) serão replicados aqui. Diante dessa consideração, elaboramos o Quadro 1, selecionando três dos cinco métodos de processamento de imagem mencionados, a fim de comparar e escolher qual método será utilizado no trabalho.

De acordo com Carvalho et al. (2022), seus resultados mostraram que a maioria dos algoritmos obtiveram uma taxa de exatidão acima de 80%, vale destacar que o CNN atingiu 98,9% sendo a maior exatidão, em contrapartida apresentou o pior desempenho por imagem sendo de 16 segundos, enquanto os métodos Haar Cascade e YOLO mostraram uma taxa de acurária de 82% e 91% com latências de 0,05 e 0,45 segundos respectivamente. Por fim, YOLO se destaca considerando os critérios de menor latência e maior exatidão.

Todos os métodos apresentados nos permitem a identificação e localização objetos específicos em imagens ou vídeos. No entanto, para a nossa aplicação em específico, o Haar Cascade nos oferece uma solução eficaz e eficiente em termos de recursos computacionais, isso é fundamental para garantir a eficiência do sistema e minimizar qualquer latência que possa afetar a resposta em tempo real. Além disso, os possíveis pontos negativos do método não têm impacto significativo para interferir na conclusão do objetivo proposto.

Quadro 1 – Comparativo entre 3 algoritmos de detecção facial

Método de Processamento de Imagem	Haar Cascade	You Only Look Once (YOLO)	Region-based Convolutional Neural Network (R-CNN)
Descrição	Utiliza características Haar e classificadores treinados para detecção de objetos por meio de janelas deslizantes.	Utiliza uma única rede neural para prever bounding boxes e classes de objetos em uma única passagem.	Propõe regiões de interesse e aplica redes neurais convolucionais em cada região.
Aplicações	Reconhecimento facial, detecção de veículos, detecção de gestos.	Detecção de objetos em tempo real, veículos autônomos.	Detecção de objetos, segmentação de objetos.
Vantagens	Rápido, eficaz para objetos bem definidos.	Alta precisão e velocidade.	Preciso e flexível.
Desvantagens	Sensível a variações de iluminação e orientação.	Requer poder computacional significativo.	Mais lento devido ao processamento em várias etapas.

Fonte: Elaborado pelo autor (2023).

3.3.1 Processo de treinamento de classificadores Haar Cascade

Como já comentado, o processo de análise de um objeto utilizando o algoritmo de reconhecimento de padrões de imagens proposto por Viola e Jones (2001) começa com a imagem de entrada, em seguida é calculado sua imagem integral que é uma etapa de pré-processamento com o propósito de evitar cálculos repetitivos e excessivos na fase de soma de valores, então são aplicados os recursos Haar em uma ordem predefinida que com base na diferença de luminosidade indicarão se aquele fragmento da imagem possui chances de integrar o objeto de interesse.

Logo depois é utilizado o algoritmo de aprendizagem baseado no AdaBoost capaz de extrair pequenos conjuntos de recursos críticos, classificadores fracos, de uma grande quantidade de recursos. Por fim é utilizado o método que combina os classificadores fracos em uma estrutura em cascata se tornando um classificador forte concentrando-se apenas nas regiões promissoras da imagem assim aumentando

a velocidade do detector e nos fornecendo garantias estatísticas de que as regiões descartadas não contêm o objeto de interesse.

O processo de treinamento de classificadores Haar Cascade é complexo e requer um conjunto de dados adequados e ajustes cuidadosos de parâmetros para obter resultados satisfatórios. Nesse contexto, o OpenCV se destaca como uma das bibliotecas de visão computacional mais populares e amplamente adotadas. Essa popularidade se deve ao fornecimento de uma ampla gama de ferramentas e recursos voltados para o processamento de imagens e visão computacional, assim simplificando muitos aspectos do treinamento de classificadores Haar Cascade.

Além disso, a biblioteca possui uma documentação completa e tutoriais detalhados que servem como guias ao longo de todo o processo de treinamento de classificadores Haar Cascade. O OpenCV é notável por sua otimização de desempenho, uma característica fundamental para o treinamento de classificadores Haar Cascade, que pode envolver conjuntos de dados volumosos e cálculos computacionalmente intensivos. Essa otimização assegura que o processo de treinamento seja executado de maneira eficiente e eficaz.

Para treinar um classificador Haar Cascade com o intuito de detectar um alvo precisamos criar um conjunto de imagens positivas que contenham o alvo em várias posições e variações de iluminação, criar um conjunto de imagens negativas que não contenham o alvo, criar uma lista que descreva a localização das imagens negativas e para as imagens positivas utilizamos o comando opencv_createsamples que nos permite gerar vários aspectos positivos a partir de uma única imagem de objeto positiva ou podemos fornecer um coleção de imagens previamente marcadas para gerar o arquivo .vec com as amostras positivas.

Após executar o comando apresentado na Figura 13, o OpenCV iniciará o processo de treinamento e salvará os arquivos do classificador resultante no diretório de saída especificado.

Figura 13 – Comando para treinar um classificador Haar Cascade com o OpenCV.

```
opencv_traincascade
  -data <diretorio_de_saida> -vec <arquivo_vec> -bg <arquivo_negativo>
  -numPos <numero_de_imagens_positivas> -numNeg <numero_de_imagens_negativas>
  -w <largura_do_quadro> -h <altura_do_quadro>
  -numStages <numero_de_estagios> -featureType <tipo_de_recurso>
  -precalcValBufSize <tamanho_do_buffer_val> -precalcIdxBufSize <tamanho_do_buffer_idx>
```

Fonte: Elaborado pelo autor (2023).

Os argumentos mais comuns mostrados na Figura 13 podem ser interpretados como:

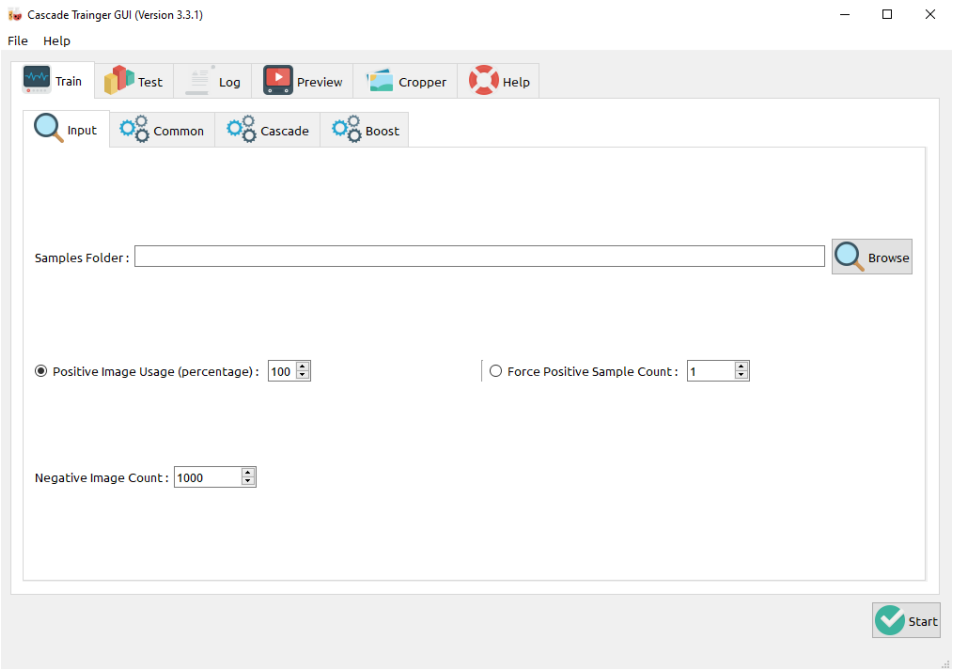
- <diretório_de_saída>: O diretório onde os arquivos de saída do classificador treinado serão armazenados.
- <arquivo_vec>: O arquivo .vec que contém as informações sobre as imagens

positivas. Este arquivo é criado a partir das imagens positivas usando a ferramenta opencv_createsamples.

- <arquivo_negativo>: O arquivo que lista onde estão as imagens negativas.
- <número_de_imagens_positivas>: O número de amostras positivas.
- <número_de_imagens_negativas>: O número de amostras negativas.
- <número_de_estágios>: O número de estágios de treinamento. O treinamento ocorre em várias etapas, onde o classificador é refinado a cada estágio.
- <largura_do_quadro> e <altura_do_quadro>: As dimensões (em pixels) das amostras de saída.
- <tipo_de_recurso>: O tipo de recurso a ser usado. Pode ser ”HAAR”, ”LBP” ou ”HOG”.
- <tamanho_do_buffer_val> e <tamanho_do_buffer_idx>: Tamanhos dos buffers de pré-cálculo para acelerar o treinamento. Os 2 valores combinados não devem exceder a memória do sistema disponível.

A biblioteca OpenCV no próprio site disponibiliza guias, programas e scripts detalhados que cobrem todas as etapas do processo de treinamento de classificadores Haar Cascade, isso inclui desde a coleta de dados de treinamento, preparação dos dados de treinamento e execução do treinamento do modelo real. Vale mencionar que a etapa de preparação dos dados de treinamento pode ser especialmente complexa, devido à necessidade de formatar os dados de maneira adequada para o treinamento. Para simplificar muitos dos passos envolvidos nessa fase, foi utilizado o software Cascade Trainer GUI, a Figura 14 ilustra a interface gráfica interativa dessa ferramenta.

Figura 14 – Interface do Cascade Trainer Gui



Fonte: Elaborado pelo autor (2023).

Para iniciar o processo de treinamento, é necessário preparar uma estrutura de pastas específica. Primeiramente, você cria uma pasta principal para o classificador. Dentro dessa pasta principal, são criadas duas subpastas: uma chamada ”p” para armazenar as imagens positivas e outra chamada ”n” para armazenar as imagens negativas. Além disso, as guias *Common*, *Cascade* e *Boost* são usadas para configurar diversos parâmetros que personalizam o processo de treinamento do classificador. Por padrão, já é definido configurações otimizadas e recomendadas para esses parâmetros.

3.3.2 Escolha do alvo

Na área de drones e UAVs os alvos servem como referências visuais essenciais, permitindo-os a realizar com precisão tarefas como aterrissagem e decolagem em uma plataforma de lançamento, além de facilitar a coleta de dados em locais específicos. Esses alvos são projetado para serem facilmente detectados e reconhecidos por câmeras ou sensores a bordo do veículo aéreo, servindo como pontos de referências para auxiliar a aeronave na determinação de sua posição, orientação e distância em relação ao ponto de pouso desejado.

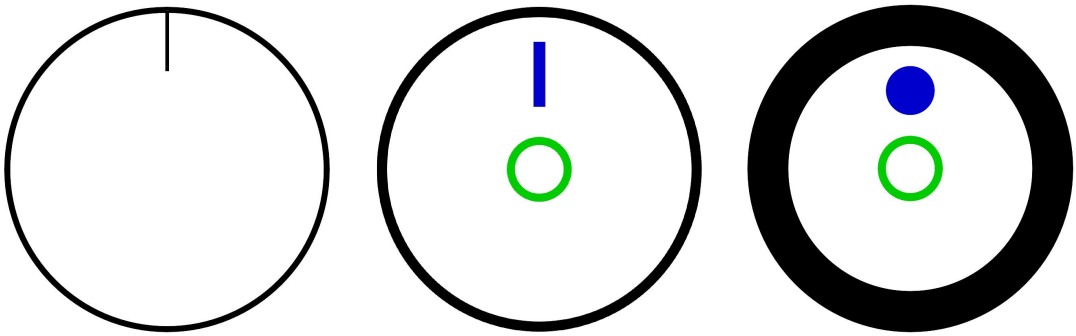
O propósito do alvo para o trabalho é fornecer uma referência visual durante a aterrissagem do drone. Para garantir que ele seja facilmente identificável e reconhecível independente da situação, optamos por utilizar um círculo como forma central do alvo, pois é facilmente distinguível em diferentes ângulos de visão. Quanto às cores do alvo escolhemos criar um forte contraste entre o alvo (preto) e o fundo (branco) e iniciamos com a ideia de dimensionar o alvo para ter aproximadamente o tamanho de uma folha A4, o que proporciona uma dimensão prática para a detecção.

Determinado o desing inicial do alvo, foi feita a implementação e a impressão do mesmo. Em seguida, foram realizados testes utilizando a câmera do drone para avaliar como o alvo realmente aparece nas imagens capturadas, e com essas imagens foi gerado e aplicado um classificador Haar Cascade para avaliar sua visibilidade e eficácia perante a máxima e mínima distancia possível entre a câmera e o alvo que permitisse a detecção correta do alvo. A Figura 15 ilustra a evolução do desenvolvimento do alvo até chegar no design final que demonstrou um desempenho satisfatório.

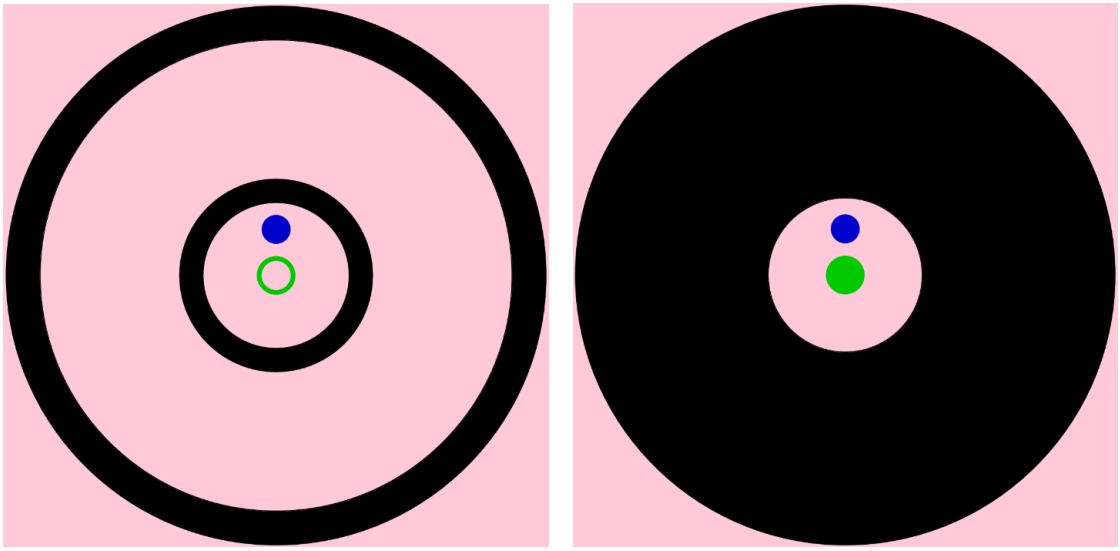
Ao aproximar a câmera do protótipo I viu-se a necessidade de inserir um alvo menor dentro dele, devido ao fato de que quando o drone se aproximava do solo acabava perdendo-o de vista antes do mesmo encostar no solo, o que poderia resultar em aterrissagens imprecisas ou perigosas. Portanto, a adição de um alvo menor dentro dos próximos protótipos mostrou se importante pois garantiria que o drone pudesse manter o alvo em vista até o momento de sua aterrissagem, aumentando a precisão e a segurança das operações.

No protótipo II, foram implementados algumas melhorias, incluindo a adição de um retângulo para fornecer uma direção visual ao drone. Durante os testes com o

Figura 15 – Evolução dos alvos.



(a) Protótipo I (b) Protótipo II (c) Protótipo III



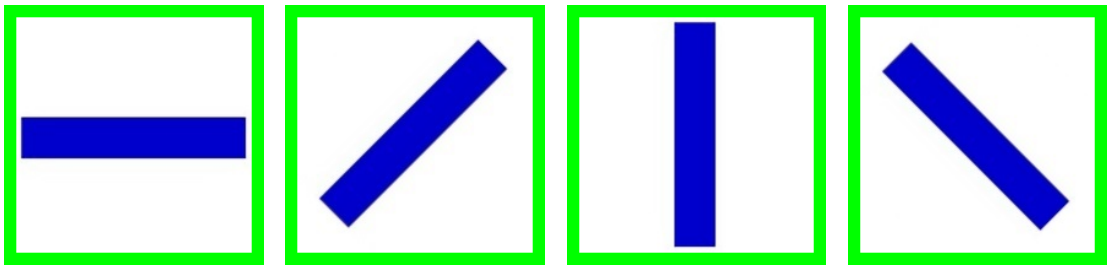
(d) Protótipo IV (e) Protótipo V

Fonte: Elaborado pelo autor (2023).

protótipo II, enfrentamos um desafio ao tentar treinar um classificador Haar Cascade para reconhecer formas geométricas com lados, devido a ter inúmeras variações ao rotacionar a imagem como mostra a Figura 16. Além disso, percebemos que as bordas do círculo externo no alvo eram muito finas, o que dificultava o reconhecimento da imagem quando a distância entre a câmera e o alvo era superior a 30 centímetros.

O protótipo III foi impresso em uma folha A4 de cor rosa, proporcionando um bom contraste com o fundo da sala e resultando em detecções mais eficazes. Embora tenha oferecido soluções eficazes para as questões identificadas anteriormente, percebeu-se a necessidade de aprimorar ainda mais o sistema, especialmente no que diz respeito à detecção do alvo em distâncias maiores. Para atender a essa demanda, adicionamos um circulo mais externo, projetado para ser facilmente detectado mesmo a uma distância de até 2,2 metros entre a câmera e o alvo, sendo necessário a impressão ser em uma folha de dimensões maiores, A1.

Figura 16 – Quadros de detecção em retângulos e circulos.

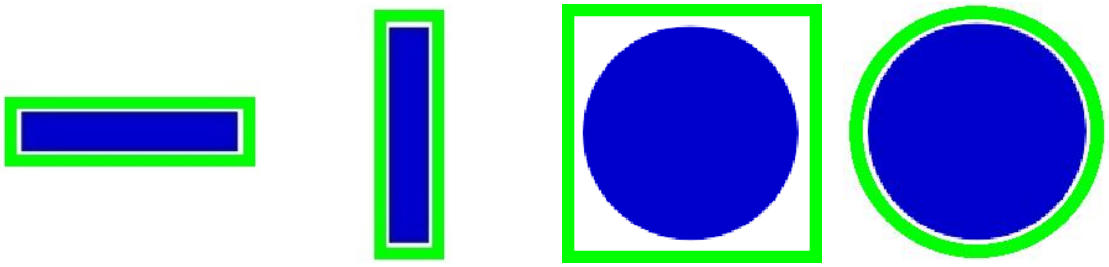


(a) Quadrado de detecção aplicado em um retângulo horizontal.

(b) Quadrado de detecção aplicado em um retângulo horizontal rotacionado 45º.

(c) Quadrado de detecção aplicado em um retângulo vertical.

(d) Quadrado de detecção aplicado em um retângulo vertical rotacionado 45º.



(e) Retângulo de detecção aplicado em um retângulo horizontal.

(f) Retângulo de detecção aplicado em um retângulo vertical.

(g) Quadrado de detecção aplicado em um círculo.

(h) Circunferência de detecção aplicado em um círculo através da Hough Circle Transform.

Fonte: Elaborado pelo autor (2023).

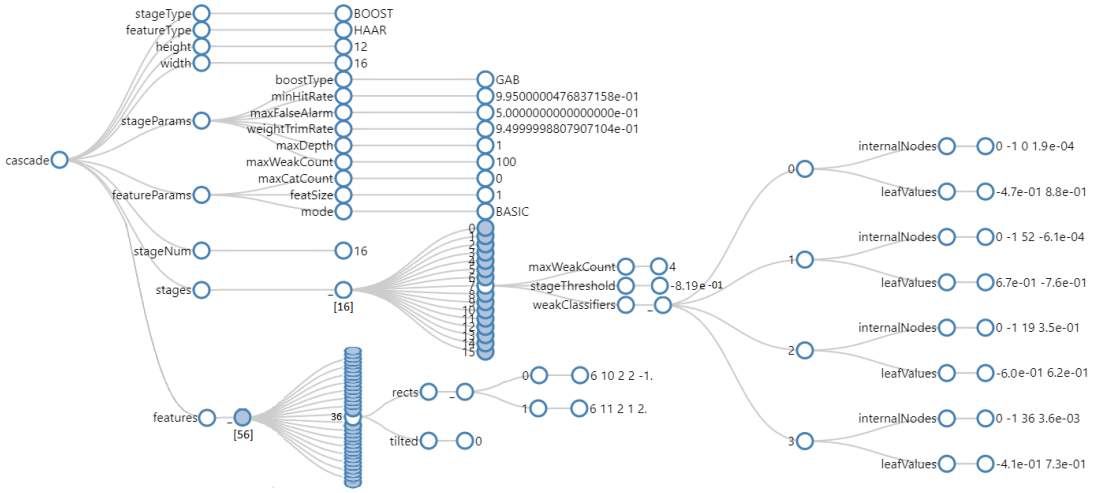
O protótipo IV foi projetado para ser impresso em uma folha A1, apresentando um fundo rosa devido as folhas de impressão serem brancas. Este protótipo apresentou resultados satisfatórios utilizando tratamento em escala de cinza, abordando todas as questões dos modelos anteriores. No entanto, durante o desenvolvimento do trabalho, observou-se que utilizar o tratamento Hue, Saturation e Value (HSV) nas imagens proporciona melhores resultados em termos de distância e precisão de detecção. Diante disso, surgiu o protótipo V, adaptado para se adequar ao tratamento de imagem em HSV, demonstrando-se satisfatório em todos os requisitos estabelecidos.

3.3.3 Criação do arquivo XML utilizado para detectar o alvo

Os treinamentos de classificadores Haar Cascade geram um arquivo XML que contém informações essenciais sobre o classificador de detecção treinado incluindo informações sobre os estágios, os recursos usados para a detecção, os limiares de decisão e outros parâmetros do modelo conforme mostrado na Figura 17. Essa estrutura em formato XML viabiliza a aplicação do mesmo em diferentes linguagens de

programação e frameworks de visão computacional.

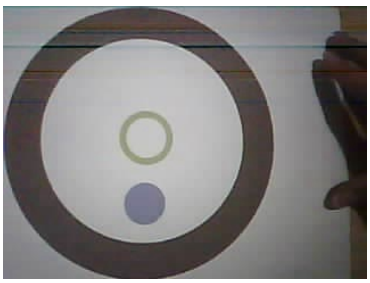
Figura 17 – Visualização em árvore do arquivo xml criado durante os treinamentos.



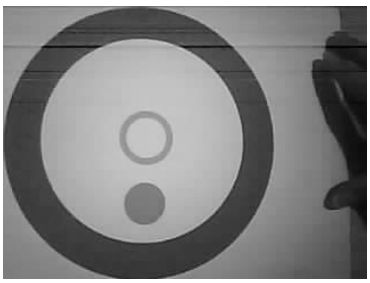
Fonte: Elaborado pelo autor (2023).

Para iniciar o processo de criação do arquivo XML do classificador Haar Cascade precisamos seguir algumas etapas. Inicialmente precisamos coletar imagens tanto positivas quanto negativas do alvo, então foi desenvolvido um código em python que gravasse o vídeo gerado no servidor WEB pela ESP32-CAM no computador e em seguida outro código foi aplicado para transformar o video em imagens e convertê-las para escala de cinza aplicando a função ”cvtColor()” e passando ”COLOR_BGR2GRAY” como argumento, caso necessário foram feitos cortes nas imagens positivas para melhorar a detecção do alvo removendo partes não úteis, como ilustrado na Figura 18.

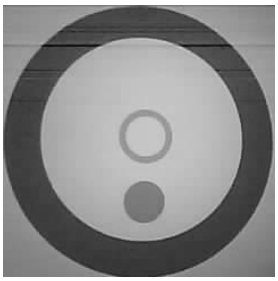
Figura 18 – Aquisição e tratamento nas imagens.



(a) Imagem original.



(b) Imagem na escala de cinza.



(c) Imagem cortada mas mantendo a proporção dos parâmetros de treinamento.

$$RGB\ to\ Gray:$$
$$0,299 \times R + 0,587 \times G + 0,114 \times B \rightarrow Y$$

(d) Conversão dos valores RGB em tons de cinza produzido pelo argumento COLOR_BGR2GRAY.

Fonte: Elaborado pelo autor (2023).

Durante o treinamento, todas as imagens, tanto positivas quanto negativas, são geralmente redimensionadas para um tamanho específico. Isso é feito pelos parâmetros ”largura_do_quadro” e ”altura_do_quadro” no comando do ”opencv_traincascade” do OpenCV. Portanto, mesmo que as imagens positivas tenham resoluções iniciais diferentes, elas serão redimensionadas para a mesma resolução durante o processo de treinamento e é importante que estes parâmetros estejam na proporção correta em relação à resolução das imagens positivas, o recomendado é que as dimensões sejam cerca de 20 vezes menores do que a resolução original das imagens.

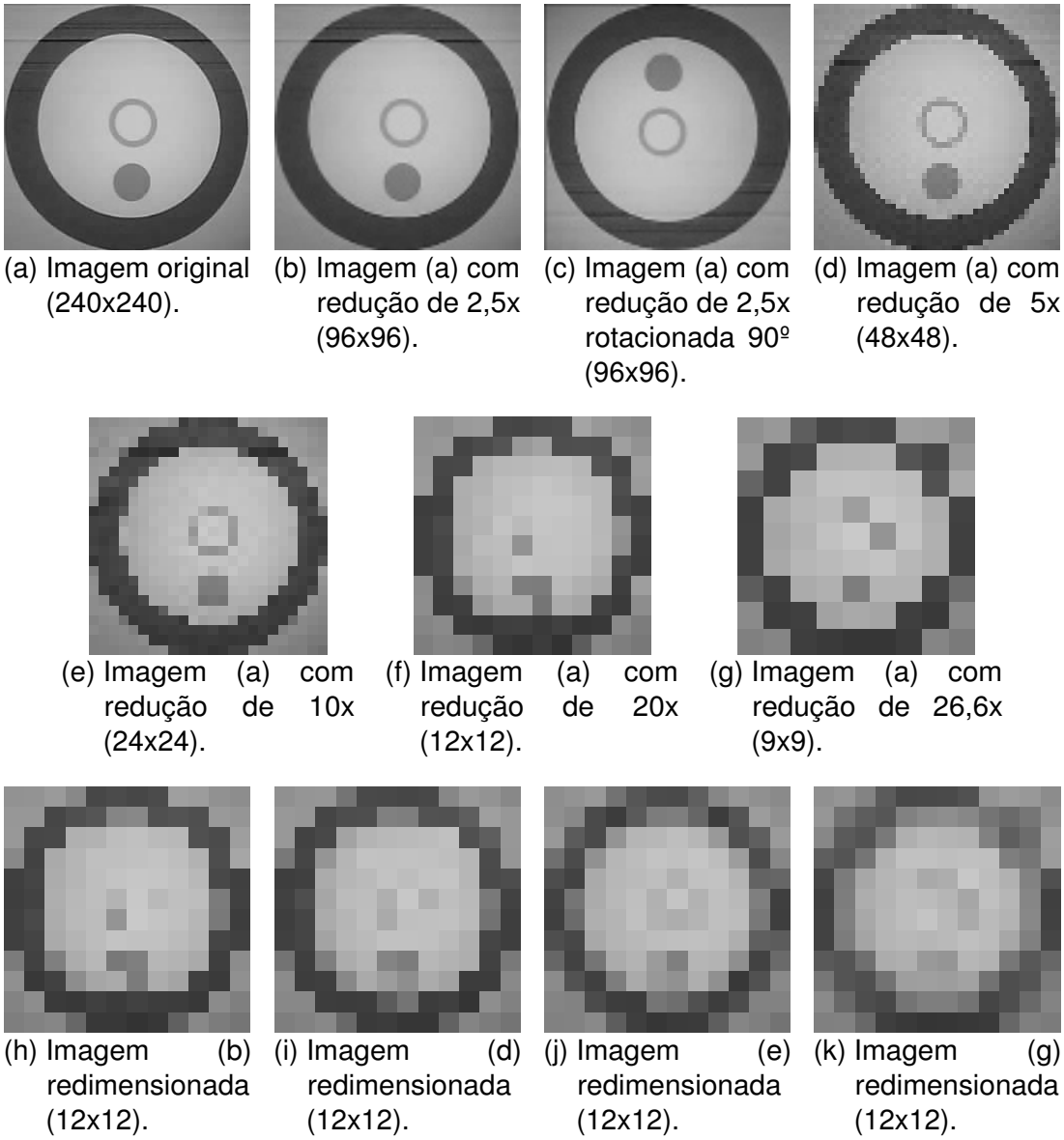
A resolução da câmera ESP32 escolhida foi de 320x240 pixels, portanto os valores de ”largura_do_quadro” e ”altura_do_quadro” utilizados para treinar a detecção dos alvos por um quadro de detecção retângular foram de 16 e 12 enquanto para um quadro de detecção quadrado foi de 12 e 12 respectivamente. Resumindo, enquanto as imagens positivas precisam ter a mesma proporção de resolução e preferencialmente serem iguais ou maiores que a ”largura_do_quadro” e ”altura_do_quadro”, as imagens negativas podem ter resoluções uma variedade de resoluções e é benéfico isto pois contribui para um treinamento mais robusto do classificador Haar Cascade.

Por fim, as imagens foram tratadas manualmente onde as imagens negativas foram redimensionadas para um tamanho menor. Já as imagens positivas, além de serem redimensionadas para um tamanho menor, foram rotacionadas em 180º, contribuindo para o aumento da variedade nas amostras positivas, conforme ilustrado na Figura 19. A rotação das imagens nos permite criar variações na luminosidade enquanto o redimensionamento para valores menores, embora resultasse em perda de detalhes, possibilitou a detecção do alvo em distâncias maiores, portanto, é necessário cuidado para não perder informações relevantes durante o processo de redimensionamento.

Para criar um arquivo XML de classificador Haar Cascade capaz de detectar o círculo central foram usadas 6536 imagens positivas e 1000 de 3784 imagens negativas disponíveis, todas previamente processadas, já para identificar o círculo intermediário utilizou-se um conjunto de 33408 imagens positivas e 1500 de 15136 imagens negativas disponíveis, e por fim, para detectar o círculo mais externo foi utilizado um conjunto de 46.720 imagens positivas e 4000 de 15136 imagens negativas disponíveis. Esses números representam a quantidade de dados de treinamento utilizados para treinar os classificadores para detectar 3 diferentes tipos de alvos.

Os classificadores foram treinados em um computador com sistema Windows (64Bit), 8 Gb de RAM e CPU Intel Core i5-4690K e as configurações de treinamento no Cascade Trainer GUI foram mantidas padrões, com exceção das configurações relacionadas à porcentagem de imagens positivas (90-95%), quantidade de imagens negativas, resolução e buffers. O algoritmo de treinamento utilizado foi o Gentle AdaBoost (GAB) que, de acordo com Friedman, Hastie e Tibshirani (2000), é uma versão modificada do Real AdaBoost onde introduz uma abordagem mais suave para

Figura 19 – Tratamento aplicado nas amostras positivas e verificação dos redimensionamentos.



(a) Imagem original (240x240). (b) Imagem (a) com redução de 2,5x (96x96). (c) Imagem (a) com redução de 2,5x rotacionada 90º (96x96). (d) Imagem (a) com redução de 5x (48x48).

(e) Imagem (a) com redução de 10x (24x24). (f) Imagem (a) com redução de 20x (12x12). (g) Imagem (a) com redução de 26,6x (9x9).

(h) Imagem (b) redimensionada (12x12). (i) Imagem (d) redimensionada (12x12). (j) Imagem (e) redimensionada (12x12). (k) Imagem (g) redimensionada (12x12).

Fonte: Elaborado pelo autor (2023).

ajustar pesos, minimizando a função de perda exponencial do Adaboost utilizando o método de Newton, tornando-o estável e robusto para dados ruidosos e com outliers.

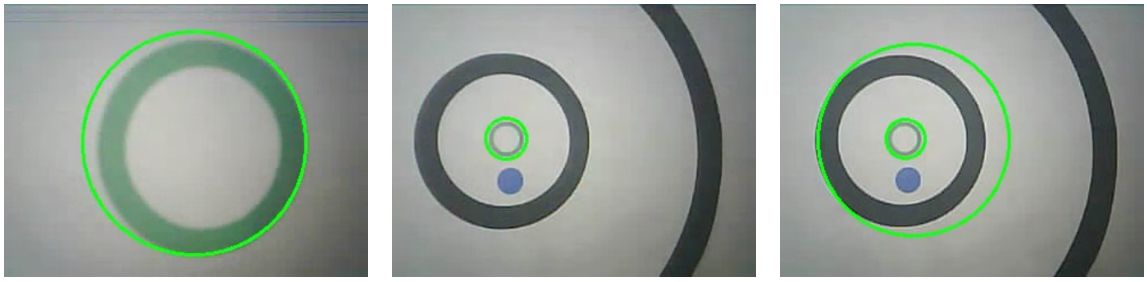
Durante o treinamento, é gerado um arquivo de log que fornece informações como a taxa de acerto (hit rate) e falsos alarmes (false alarm) obtidas no decorrer dos estágios treinados conforme ilustrado na Figura 20. Desenvolveu-se um código que nos permite testar o arquivo XML criado, permitindo verificar visualmente se é possivel detectar o alvo corretamente, se os resultados do treinamento atendem às expectativas de distância de detecção e também se gera alguma interferência com outras partes dos alvos como demonstrado na Figura 21.

Figura 20 – Log gerado durante os treinamentos.

```
END>

Training until now has taken 0 days 0 hours 0 minutes 36 seconds.

===== TRAINING 7-stage =====
<BEGIN
POS count : consumed   6209 : 6315
NEG count : acceptanceRatio    1000 : 0.000856504
Precalculation time: 3.838
+----+---------+---------+
|  N |    HR   |    FA   |
+----+---------+---------+

|   1|        1|        1|
+----+---------+---------+

|   2| 0.999034|    0.562|
+----+---------+---------+

|   3| 0.999034|    0.562|
+----+---------+---------+

|   4|  0.99549|     0.46|
+----+---------+---------+
END>
Training until now has taken 0 days 0 hours 0 minutes 43 seconds.

===== TRAINING 8-stage =====
<BEGIN
```

Fonte: Elaborado pelo autor (2023).

Figura 21 – Resultado de detecção para o circulo interno.



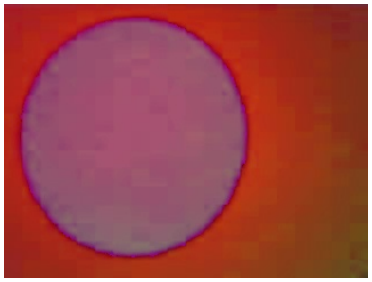
(a) Detecção de perto. (b) Detecção de longe. (c) Interferência gerada a partir de uma distância específica.

Fonte: Elaborado pelo autor (2023).

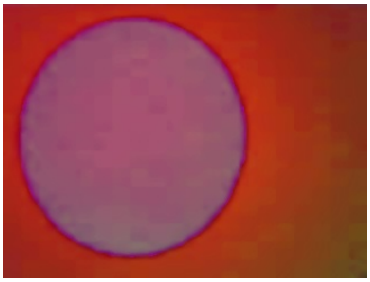
Para detectar a bola sólida azul foi utilizado outra abordagem de tratamento nas imagens pois não obtivemos um resultado que considerássemos ser suficiente. Dito isso, as imagens foram convertidas para HSV aplicando a função ”cvtColor()” e passando ”COLOR_BGR2HSV” como argumento, então foi aplicado um filtro de desfoque mediano através da função ”medianBlur()” e por fim foi aplicado limites inferiores e superiores de cores por meio da função ”inRange()” passando como argumento ”[50, 50, 50]” e ”[100, 200,200]”, a Figura 22 ilustra o processo de tratamento aplicado nas imagens obtidas.

Figura 22 – Aquisição e tratamento nas imagens.

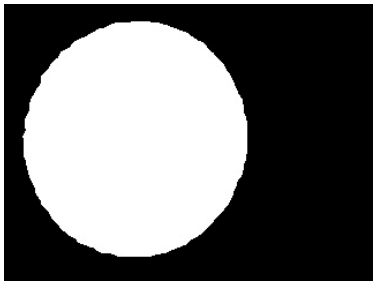
(a) Imagem original.



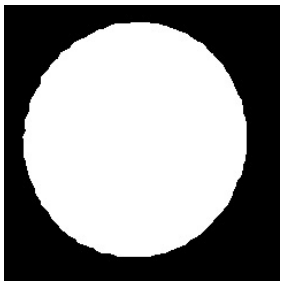
(b) Imagem na escala HSV (hue, saturation e value).



(c) Imagem desfocada por filtro mediano.



(d) Imagem filtrada pelos limites inferiores e superiores de tons de azul.



(e) Imagem cortada mas mantendo a proporção dos parâmetros de treinamento.

$$H \leftarrow \begin{cases} 60(G-B)/(V-min(R,G,B)) & \text{if } V = R \\ 120 + 60(B-R)/(V-min(R,G,B)) & \text{if } V = G \\ 240 + 60(R-G)/(V-min(R,G,B)) & \text{if } V = B \\ 0 & \text{if } R = G = B \end{cases}$$

$$S \leftarrow \begin{cases} \frac{V-min(R,G,B)}{V} & \text{if } V \neq 0 \\ 0 & \text{otherwise} \end{cases} \qquad V \leftarrow max(R,G,B)$$

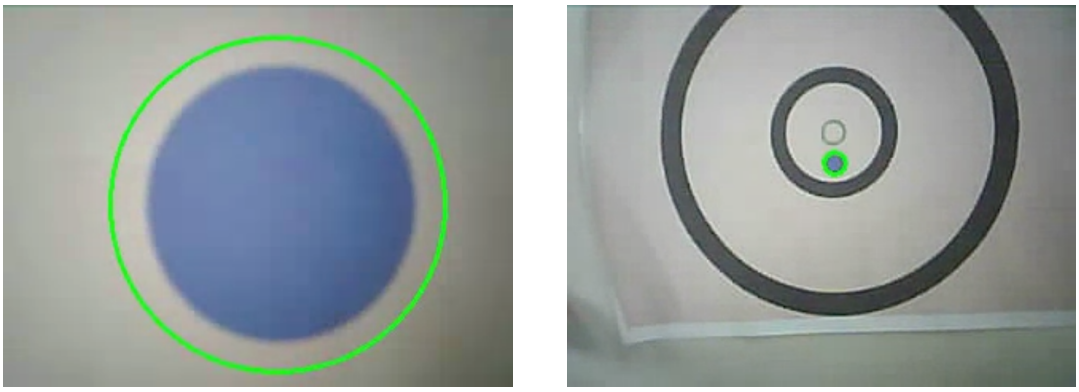
8-bit images: $V \leftarrow 255V, S \leftarrow 255S, H \leftarrow H/2(\text{to fit to 0 to 255})$

(f) Conversão dos valores RGB para a escala HSV produzido pelo argumento COLOR_BGR2HSV.

Fonte: Elaborado pelo autor (2023).

Então para treinar o classificador Haar Cascade capaz de detectar a bola sólida azul foram utilizadas 14650 imagens positivas e 2000 de 7100 imagens negativas disponíveis, todas previamente processadas. O resultado de detecção podem ser observados na Figura 23 e podemos concluir que este método de tratamento de imagem é altamente eficaz na detecção de círculos de cores sólidas, uma vez que elimina cores irrelevantes na imagem. Podemos reaproveitar o XML treinado para detectar bolas sólidas de outras cores pois é só trocar o filtro de limites inferiores e superiores de cores para a cor desejada.

Figura 23 – Resultado de detecção para a bola sólida azul.



(a) Detecção de perto. (b) Detecção de longe.

Fonte: Elaborado pelo autor (2023).

Com base no resultado satisfatório obtido, criou-se o protótipo V para atender ao tratamento de imagens por HSV. Para detectar a bola sólida verde, utilizou-se o mesmo classificador capaz de detectar a bola sólida azul. Já para treinar o classificador Haar Cascade capaz de detectar o alvo mais externo foram utilizadas 29240 imagens positivas e 2000 de 4112 imagens negativas disponíveis, todas já processadas. Foi desenvolvido um código que oferece a capacidade de realizar a calibração em tempo real dos valores do filtro de limite de cores de acordo com o ambiente.

O resultados dessas detecções podem ser observados na Figura 24, enquanto a Figura 25 apresenta a imagem utilizada para o treinamento do alvo mais externo, juntamente com uma imagem que possui ruído devido à má calibração do filtro de cores, normalmente gerado quando aplicado diferentes angulações na câmera. Esse ruído pode causar dificuldades de detecção pelo classificador caso não seja treinado para lidar com ele. Vale ressaltar que, como é uma área maior, é mais propenso a apresentar esses ruídos.

Figura 24 – Resultado de detecção para a bola sólida verde e o alvo mais externo.

(a) Distância máxima de detecção testada para a bola sólida verde.

(b) Distância máxima de detecção testada de 2,5m para o alvo externo.

Fonte: Elaborado pelo autor (2023).

Figura 25 – Imagem utilizada no treinamento e ruídos gerados.

(a) Imagem final utilizada nos treinamentos.

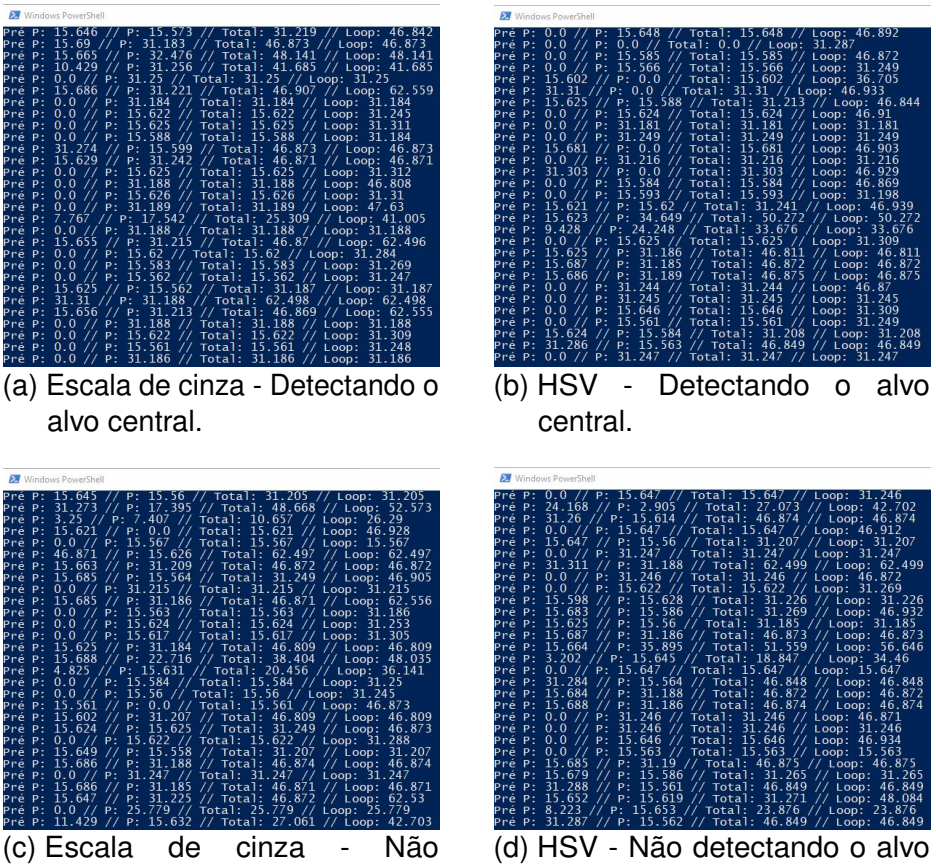
(b) Imagem original.

(c) Imagem com ruído gerado pela má calibração do filtro de limite de cores.

Fonte: Elaborado pelo autor (2023).

A latência gerada ao empregar os classificadores treinados para detectar o alvo central, tanto no tratamento utilizando escala de cinza quanto HSV, é ilustrada na Figura 26. Utilizando um notebook DELL Inspiron 14R 5437-A40 no sistema Windows, com o mínimo de aplicativos em execução e sem alterar a prioridade do processo.

Figura 26 – Dados de latência de processamento utilizando os classificadores treinados.



(a) Escala de cinza - Detectando o alvo central.

(b) HSV - Detectando o alvo central.

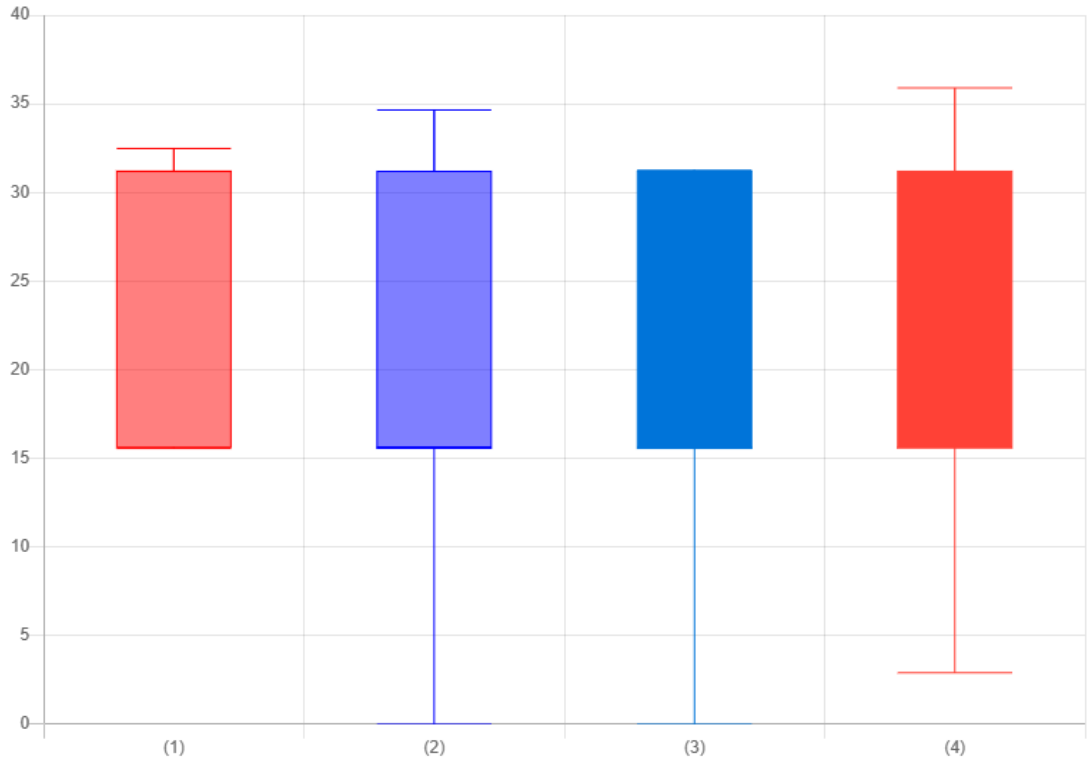
(c) Escala de cinza - Não detectando o alvo central.

(d) HSV - Não detectando o alvo central.

Fonte: Elaborado pelo autor (2023).

Notou-se que os processos não passaram de 25% de uso de CPU e através da Figura 27 podemos concluir que não há uma diferença notável de tempo de processamento entre as aplicações vistos que as medianas de todos ficaram próximas de 15.6ms.

Figura 27 – Comparação de latência de processamento dos dados coletados. (1) Escala de cinza - Detectando o alvo. (2) HSV - Detectando o alvo. (3) Escala de cinza - Não detectando o alvo. (4) HSV - Não detectando o alvo.



Fonte: Elaborado pelo autor (2023).

No decorrer do processo, observou-se que treinar alvos na escala de cinza com a mesma proporção da tela, como no caso de uma resolução de 320x240 pixels e alvos de 16x12 pixels, resultou em resultados satisfatórios apesar de precisar de uma maior área para detectar, também tentamos treinar os classificadores com resoluções maiores como 20x20 porém foi notado um maior delay de processamento. Não é recomendável inicialmente treinar com mais de 1000 imagens negativas e com mais de 20 estágios, pois o tempo de treinamento de cada estágio vai dobrando e pode não resultar em melhorias significativas, variando de acordo com cada teste.

3.4 ENVIAR COMANDO AO DRONE

Controlar um drone por meio de um computador nos permite programar e automatizar o comportamento do mesmo, seja seguindo um conjunto de comandos pré-programados ou sendo controlado por algoritmos de controle autônomos em tempo real. A implementação de um controle autônomo em um drone nos oferece diversas

vantagens e é fundamental para uma variedade de aplicações pois eles podem executar tarefas de forma mais eficiente do que operadores humanos devido a necessitar de tempo de treinamento e pilotagem e executar tarefas sem fadiga.

Além de permitir que os drones evitem obstáculos automaticamente e tomem decisões de segurança, é muito útil em realizar tarefas que envolvem voos repetitivos ou monitoramento constante, a automação economiza tempo e recursos, além de garantir uma execução consistente. Em muitos casos, a combinação de controle manual e controle por computador é usada, permitindo que o operador humano intervenha quando necessário, enquanto o drone executa tarefas autônomas. A escolha entre esses métodos depende das necessidades específicas da aplicação e das capacidades do drone em questão.

Foi proposto durante a iniciação científica a automatização do drone SYMA X8HG que só aceita comandos manuais. Para alcançar esse objetivo inicialmente replicamos todas as funções que o controle do drone permitia manualmente no computador, como botões de calibração e sequências de comandos analógicos para ligar, calibrar e desligar o drone. Em seguida, foi desenvolvida uma interface visual que possibilita controlar do drone, permitindo ajustar os parâmetros de voo de forma intuitiva. Por fim, foram realizados testes de voo, essas aplicações foram projetadas ao drone em colaboração com a equipe do drone principalmente o Pedro.

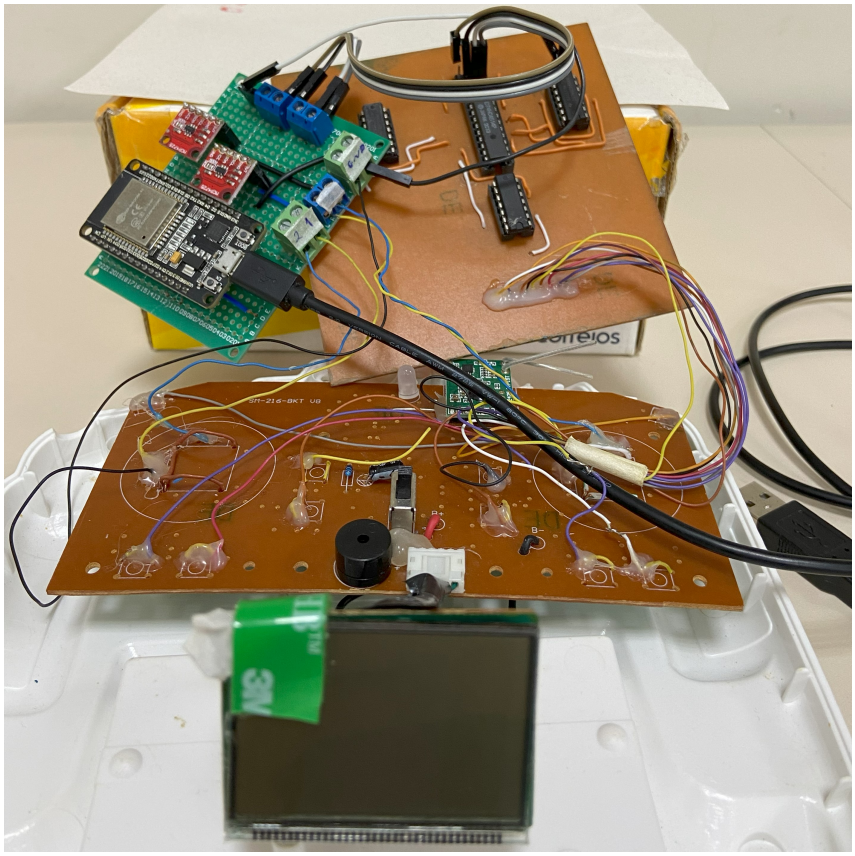
3.4.1 Hardware desenvolvido para automatizar o controle do drone

Para automatizar o controle do drone realizou-se a substituição dos botões de controle por um circuito integrado composto por um decodificador 4 para 16 modelo HD74HC4514 e por 3 comutadores quadruplo bilateral de chaves modelo SN74HC4066, já os potenciometros analógicos foram substituídos por quatro Conversores Digital-Analógico (DACs), sendo dois módulos DACs externos com resolução de 12-bits modelo MCP4725 e dois DACs integrados ao ESP32 com uma resolução de 8 bits, o que já é suficiente para realizar um controle preciso do drone. Por fim, tudo é controlado por um ESP32 e a Figura 28 mostra o hardware desenvolvido.

A comunicação entre o computador e o ESP32 foi estabelecida via USB pois nos permite ter respostas rápidas. Resumindo, O ESP32 é responsável por capturar as mensagens enviadas via USB, processando-as e, em seguida, transmitindo os comandos de ajuste e movimento para o drone. Também foi desenvolvida uma estrutura de mensagem para ser transmitida via serial que simplifica a implementação de comandos para o drone em diferentes plataformas e situações como é ilustrado na Figura 29.

As siglas ”SD” representam ”Sobe Desce”, ”HA” refere-se a ”Horário Anti-horário”, ”FT” indica ”Frente Trás” e ”ED” corresponde a ”Esquerda Direita”. Por exemplo, valores acima de 120 enviados junto ao parâmetro ”SD” fazem o drone subir, enquanto

Figura 28 – Hardware desenvolvido para automatizar o controle do drone.



Fonte: Elaborado pelo autor (2023).

Figura 29 – Estrutura de mensagem utilizada.

"S,SD120,HA70,FT70,ED50,"

Fonte: Elaborado pelo autor (2023).

valores abaixo o fazem descer. Os valores destacados em vermelho, que variam de 0 a 255, podem ser ajustados, permitindo a variação de um ou de todos eles, proporcionando um controle flexível sobre os movimentos enviados ao drone, sem a necessidade de se preocupar com as conversões enviadas para os DACs.

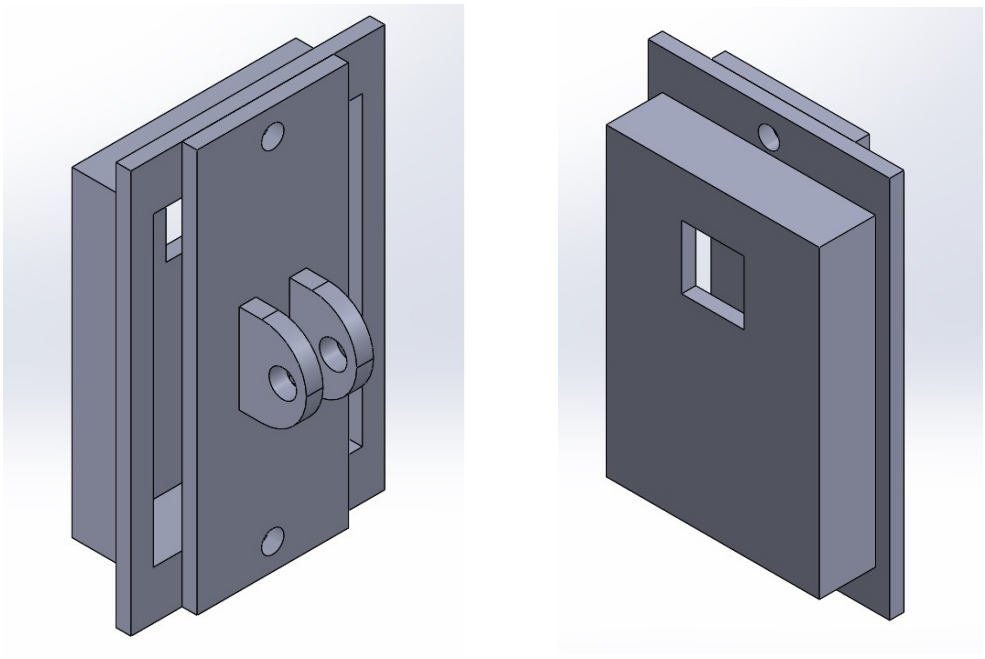
3.5 MODELAGEM DO SUPORTE E DIAGRAMA ELÉTRICO

A modelagem de estruturas personalizadas para um drone permitem que ele execute mais funções do que o padrão, por exemplo podemos acomodar cargas úteis específicas, como câmeras de alta resolução, sensores especiais, equipamentos de coleta de dados de forma organizada e protegida, e visando um suporte leve podemos manter uma autonomia de voo legal.

Para realizar o pouso autonomo no drone em questão, precisamos fixar a ESP32-CAM no mesmo, para isso foi realizado a modelagem de um suporte que

comporte a ESP32-CAM baseado no encaixe do suporte da câmera que vem no mesmo como mostra a Figura 30. A modelagem da estrutura foi feita no software SolidWorks e foi impressa na impressora 3D nos laboratórios da UFSC.

Figura 30 – Modelagem da estrutura para fixar a ESP32-CAM ao drone.

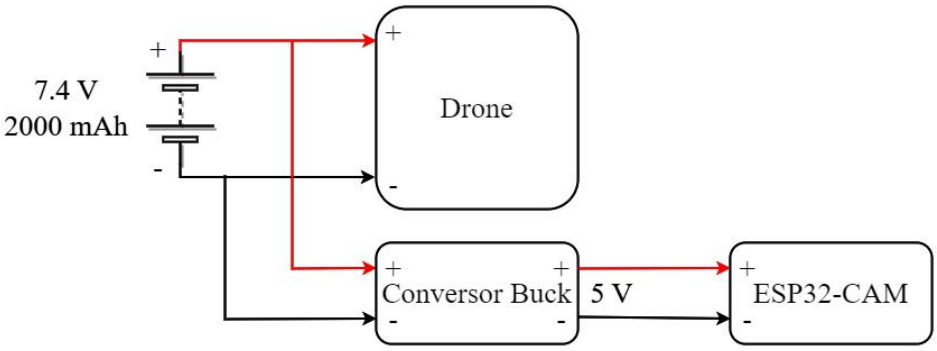


(a) Parte superior. (b) Parte inferior.

Fonte: Elaborado pelo autor (2023).

O suporte criado mostrou se eficiente pois além de ser leve, conseguiu segurar a ESP32-CAM permitindo que o sistema de pouso autônomo fosse implementado. O sistema elétrico entre o drone e a ESP32-CAM é mostrado na Figura 31 onde foi adicionado um regulador de tensão ligado em paralelo na saída da bateria para conseguir alimentar a ESP32-CAM. A Figura 32 ilustra o peso da câmera com o suporte que acompanham o drone, do suporte personalizado junto com a ESP32-CAM e do regulador de tensão.

Figura 31 – Diagrama elétrico entre o drone e a ESP32-CAM.



Fonte: Elaborado pelo autor (2023).

Figura 32 – Peso da câmera original e dos itens adicionados ao drone.

(a) Peso da câmera (30.3 g) mais o peso do suporte (18.6 g)

(b) Peso do ESP32-CAM (7.1 g) mais o peso do suporte (11 g)

(c) Peso do regulador de tensão (10.5 g)

Fonte: Elaborado pelo autor (2023).

3.6 ESTAÇÃO DE CONTROLE EM SOLO

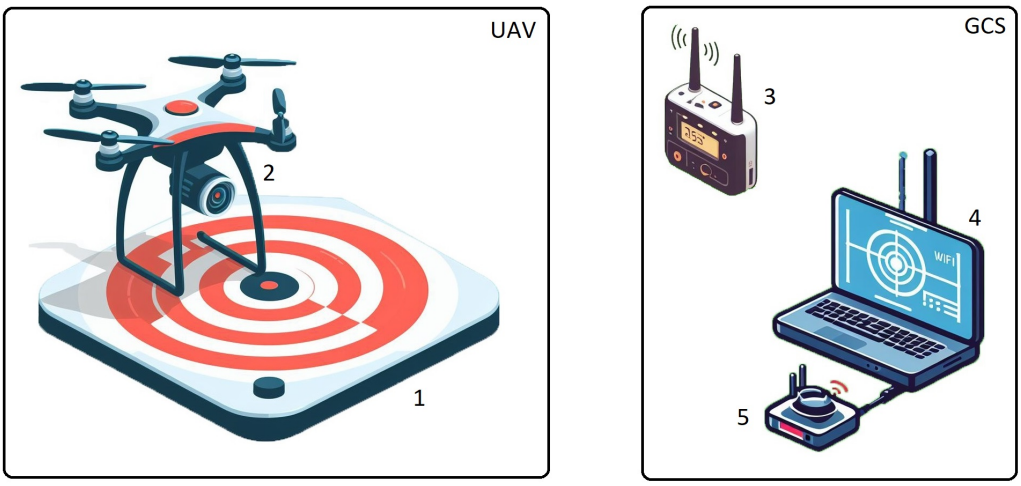
Conforme apresentado por Fahlstrom e Gleason (2012), a GCS é um local onde um operador humano monitora, controla e gerencia o UAV durante a missão, ou seja, inclui controles de voo, telas para visualização de vídeos ou imagens, sistemas de comunicação (Data links - Enlace de dados) para enviar comandos e receber informações do UAV.

Os Data Links são os meios de comunicação que permitem a transmissão de dados entre o UAV e a GCS, podem ser divididos em enlace de controle onde é utilizado para enviar comandos ao UAV, como direções de voo ou mudanças de altitude e em enlace de sensores onde são utilizados para transmitir dados coletados pelo UAV, como imagens, vídeos ou dados de sensores.

Partindo do sistema da Figura 33, temos um sistema composto por uma GCS e um UAV, o sistema UAV é composto pelo (1) alvo final desenvolvido durante o trabalho e pelo (2) drone SYMA X8HG equipado com o ESP32-CAM, enquanto no sistema GCS temos (3) um roteador INTELBRAS WRN 240, (4) um notebook DELL Inspiron 14R 5437-A40 e pelo (5) hardware desenvolvido apresentado anteriormente.

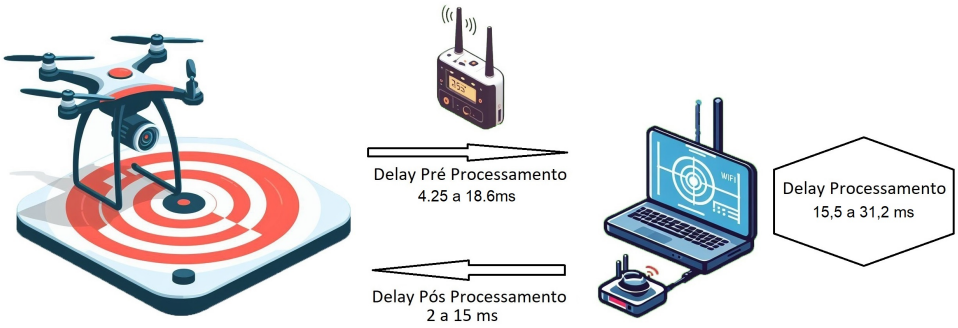
O delay crítico do sistema abordado em seções anteriores é apresentado na Figura 34, onde podemos perceber que o maior delay está na área de processamento das imagens. Durante a coleta dos dados de delay, não foram editadas as prioridades dos processos. No entanto, a implementação do processamento por threads possibilita a detecção simultânea de mais de um alvo, evitando a soma dos delays de processamento.

Figura 33 – Estação de controle em solo. (1) Alvo. (2) Drone. (3) Enlace de sensores. (4) Central de processamento. (5) Enlace de controle.



Fonte: Elaborado pelo autor (2023).

Figura 34 – Delay crítico do sistema.



Fonte: Elaborado pelo autor (2023).

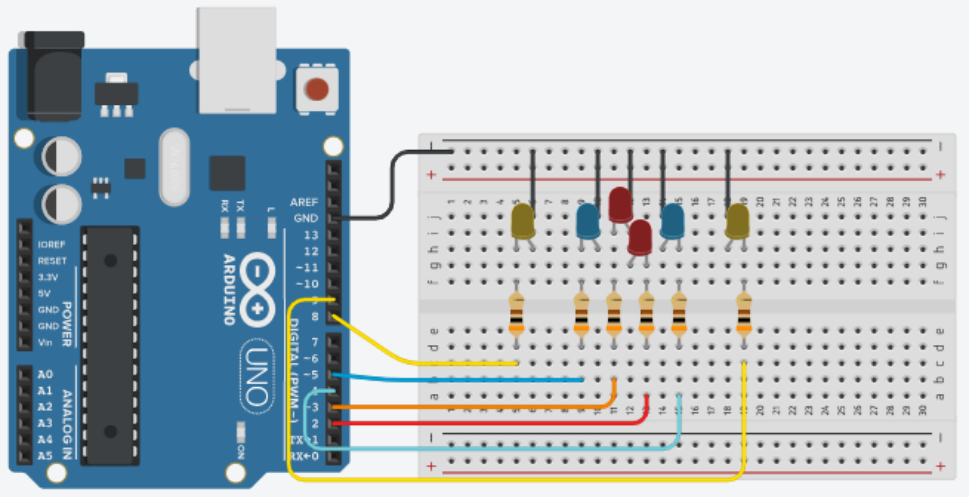
3.7 ESTRUTURAS DE TESTES

A implementação de uma rotina de testes em qualquer desenvolvimento pois nos ajudam a identificar problemas, falhas no sistema e situações de risco, a validar o algoritmo em questão e permitem o ajuste de uma série de parâmetros e configurações, permitindo que possamos fazer correções e melhorias para tornar o sistema robusto, confiável e seguro.

Então propõe-se 3 testes, o objetivo do primeiro teste é validar a parte de transmissão, recebimento e processamento da imagem, onde é verificado visualmente através de LEDs se a resposta do sistema está adequada, para realizá-lo utilizarei um arduino e resistores de 300 Ω para acender os LEDs. A idéia deste esquemático para o teste pode ser visto na Figura 35, podendo variar as conexões de acordo com a necessidade.

O objetivo do segundo teste é validar o envio de comandos do computador para o drone, para isso foi utilizado o hardware desenvolvido e nele é possivel verificar

Figura 35 – Esquemático do teste.



Fonte: Elaborado pelo autor (2023).

no próprio display a intensidade do comando que o drone vai ser enviado para o drone. Para o último teste temos que o objetivo é validar todo o sistema de pouso autonomo do drone, envolvendo a transmissão, recebimento e processamento da imagem e o envio de comandos ao drone, assim capturando o tempo necessário para pousar e a distância que ficou do centro do alvo.

Todos os testes descritos foram realizados em um notebook DELL Inspiron 14R 5437-A40, utilizando o sistema Ubuntu 22.04, com o mínimo de aplicações abertas e com as prioridades dos processos alteradas para a maior possível a fim de amenizar os delays vistos ao longo do trabalho.

4 RESULTADOS

Este capítulo apresenta os resultados obtidos a partir das etapas de testes propostas no final do capitulo 3. O projeto concentrou-se na criação e treinamento de classificadores Haar Cascade para realizar a detecção do alvo de pouso, bem como na implementação de um sistema para a realização do pouso do drone por reconhecimento de imagem. A seguir, serão discutidos os resultados alcançados em cada uma dessas áreas, destacando as estratégias, desafios e conclusões relevantes ao longo do projeto.

4.1 TRANSMISSÃO, RECEBIMENTO E PROCESSAMENTO DA IMAGEM

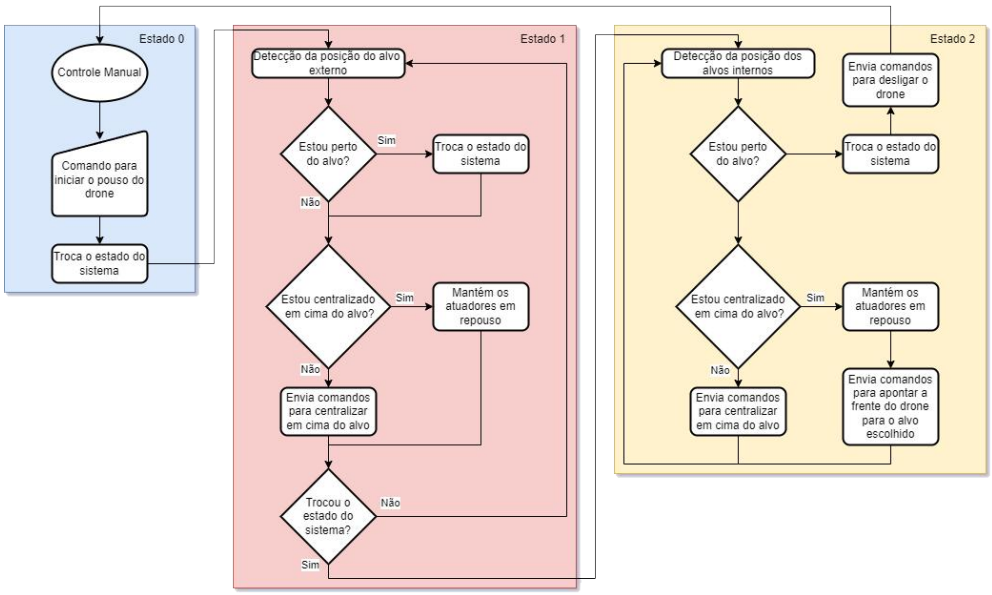
Nesta seção, é apresentado os resultados do teste proposto anteriormente para validar a etapa de captura, transmissão, recebimento e processamento da imagem. Durante o projeto foi escolhido o dispositivo de transmissão, assim como a determinação da resolução da imagem através da avaliação da latência produzida. e na eficiência do processamento.

Portanto foi implementada uma lógica pós-processamento da imagem visando a transição de detecção entre os alvos, rotação e centralização do drone, simulando um processo de pouso. Essa lógica pode ser explicada da seguinte forma: inicialmente, o drone está sob controle do operador, que, ao pressionar a tecla de início do processo de pouso, altera-se o estado para 1, indicando que está pousando. O computador, então, processa as imagens obtidas em busca de centralizar e rotacionar o drone em cima do alvo quando necessário, enviando comandos para o drone através do hardware desenvolvido.

Quando for detectada uma distância que indique a aproximação do drone ao alvo, o estado é alterado para 2, indicando que está pousando e se aproximando do solo. O computador continua atuando da mesma maneira descrita anteriormente, e, por fim, quando for detectado que o drone está a uma altura suficiente do solo para desligar, o comando é efetuado para o drone a fim de realizar um pouso suave e o estado é alterado para 0, reiniciando o loop.

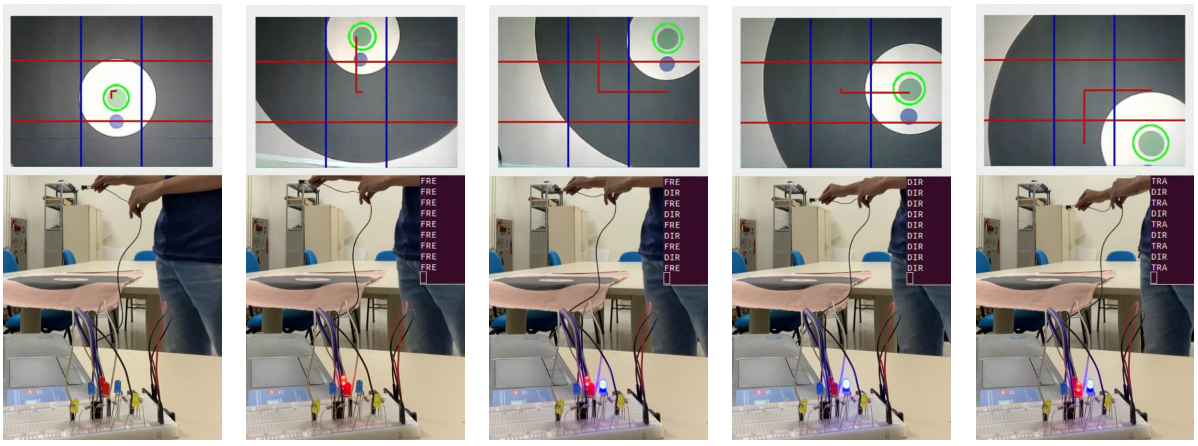
A lógica pode ser vista através da Figura 36. Portanto, espera-se que o circuito montado com LEDs, forneça um feedback em tempo real do comando a ser enviado ao drone, o mesmo pode ser verificado pela serial do programa, com o objetivo de validar o processo de transição de detecção entre os alvos, centralização do drone sobre o alvo e rotação apontando a frente do drone para a bola azul quando necessário. O resultado desses testes são apresentados nas Figuras 37 e 38.

Figura 36 – Diagrama de blocos.

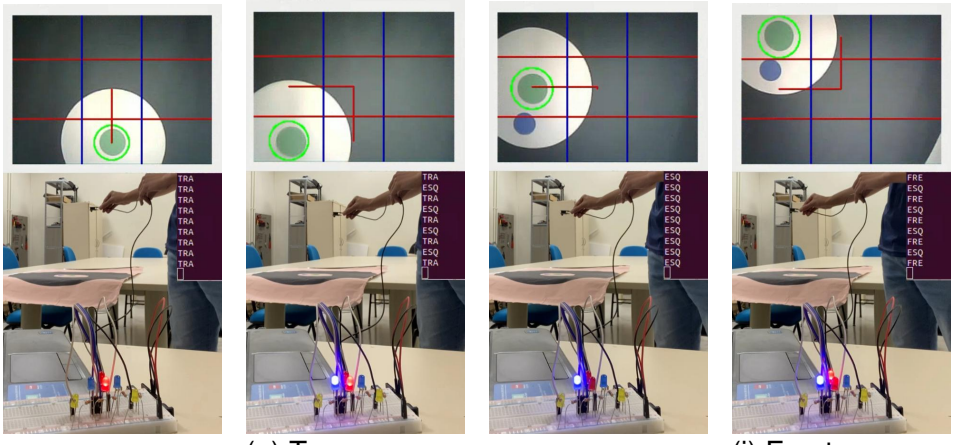


Fonte: Elaborado pelo autor (2023).

Figura 37 – Comandos de deslocamento a serem enviados ao drone pelo computador.



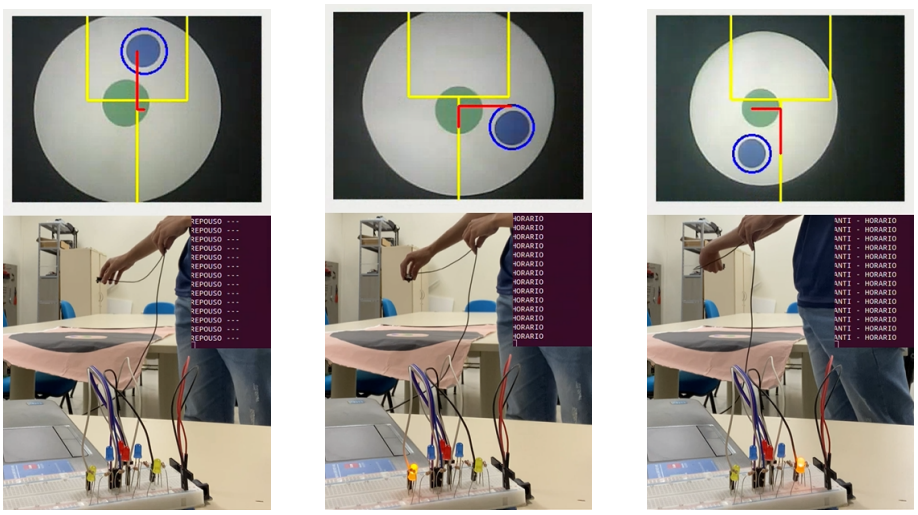
(a) Repouso. (b) Frente. (c) Frente e direita. (d) Direita. (e) Traz-direita.



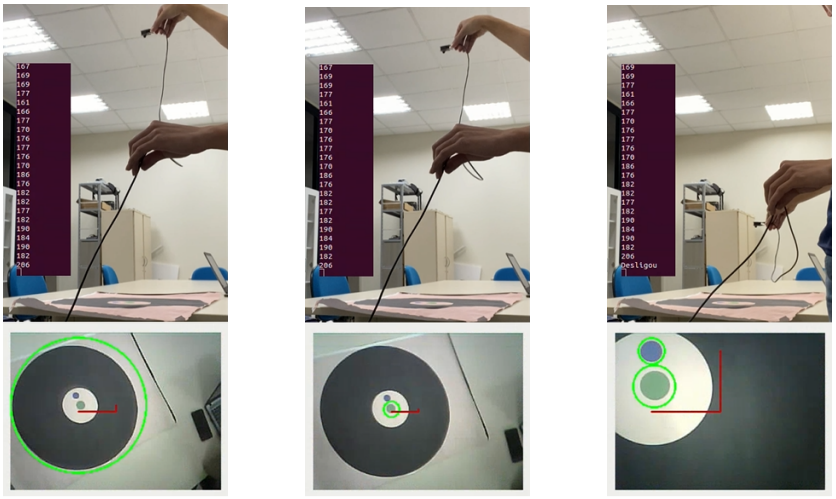
(f) Traz. (g) Traz e esquerda. (h) Esqueda. (i) Frente e esquerda.

Fonte: Elaborado pelo autor (2023).

Figura 38 – Comandos de rotação a serem enviados ao drone pelo computador e verificação da transição de estados.



(a) Repouso. (b) Horário. (c) Anti-horário.



(d) Fim do estado 1. (e) Inicio do estado 2. (f) Fim do estado 2.

Fonte: Elaborado pelo autor (2023).

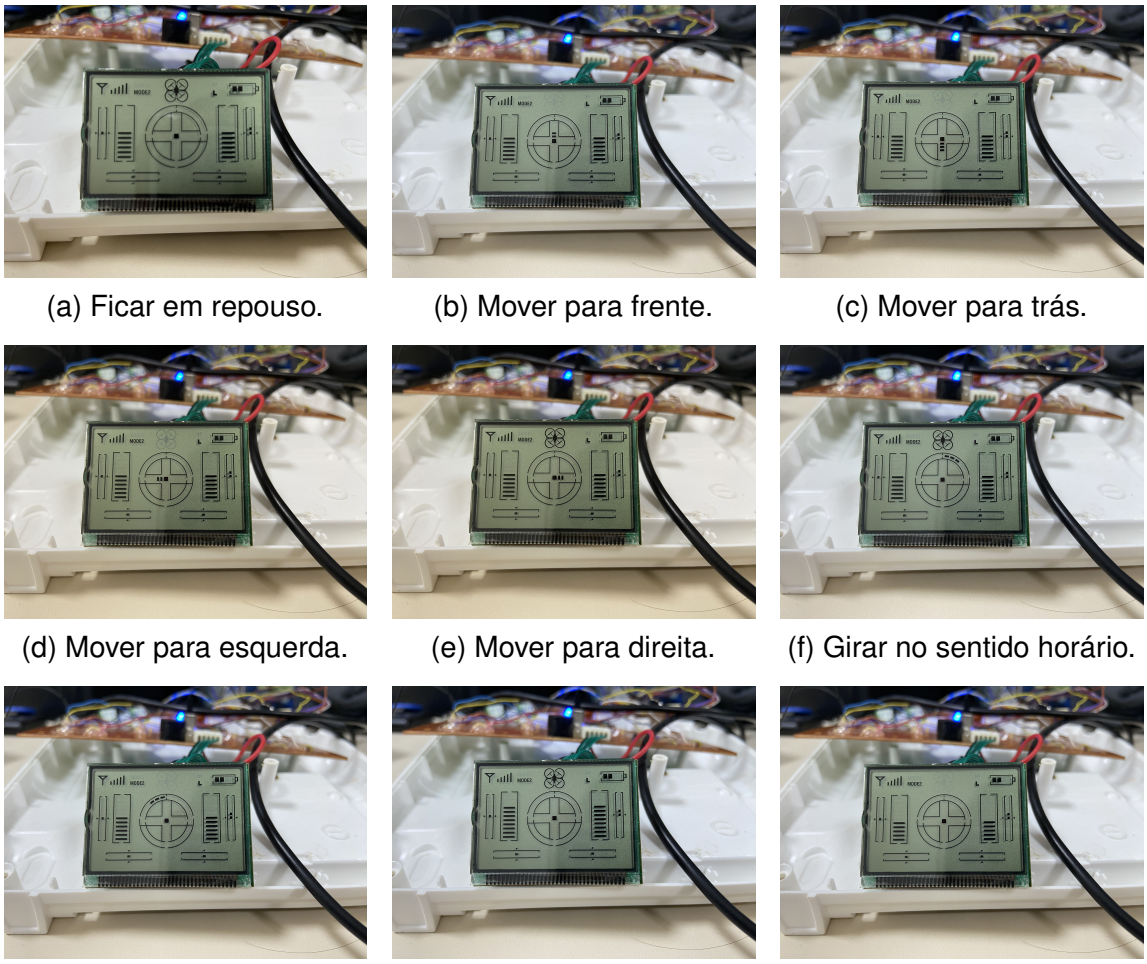
Com base nos resultados obtidos, podemos afirmar que o processo de captura, transmissão, recepção, processamento da imagem e a lógica que enviará comandos aos drone durante o pouso está de acordo com o esperado.

4.2 CONTROLAR O DRONE PELO COMPUTADOR

Nesta seção, é apresentado os resultados do teste proposto anteriormente para validar a etapa de controlar o drone pelo computador a fim de garantir que os comandos emitidos a partir do computador sejam transmitidos ao drone. Para esse teste, espera-se que seja possível calibrar inicialmente o drone e em seguida conseguir ajustar e controlar o drone pelo computador.

Para alcançar esse objetivo, foi utilizado o hardware desenvolvido durante a IC descrito na subseção 3.4.1 que nos oferece uma interface direta entre o computador e o drone. O hardware nos permite verificar a intensidade dos comandos que serão enviados ao drone através de um display incorporado no mesmo, proporcionando um feedback visual em tempo real e não necessariamente o drone precisa estar ligado. O resultado desses testes são apresentados nas Figuras 39 e 40.

Figura 39 – Calibração dos comandos enviados ao drone pelo display.



(a) Ficar em repouso. (b) Mover para frente. (c) Mover para trás.

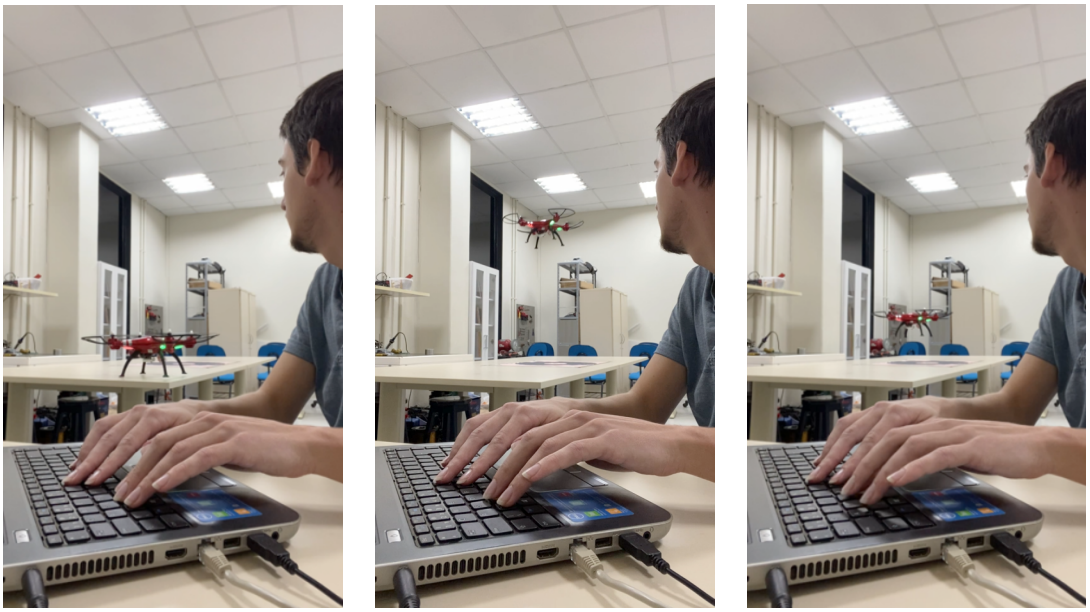
(d) Mover para esquerda. (e) Mover para direita. (f) Girar no sentido horário.

(g) Girar no sentido anti-horário. (h) Subida. (i) Descida.

Fonte: Elaborado pelo autor (2023).

Durante os testes, observamos alguns comportamentos inesperados e raros, nos quais não era possível enviar comandos ao drone, geralmente causados por algum tipo de pertubação nas conexões de alimentação do ESP32-CAM, muitas vezes devido a impactos fortes com o solo. Por segurança, já havíamos implementado no hardware desenvolvido um sistema que, caso não receba comandos enviados pelo computador por até 0.3 segundos, aciona automaticamente comandos de repouso a serem enviados ao drone, mantendo o drone pairando no ar.

Figura 40 – Decolagem e pouso do drone utilizando o teclado do notebook.



(a) Decolagem. (b) Iniciando processo de pouso. (c) Pousando o drone.

Fonte: Elaborado pelo autor (2023).

Esses comportamentos incomuns foram identificados principalmente quando não era possível ler as imagens capturadas e enviadas ao servidor pelo ESP32-CAM, travando o programa na parte de obtenção dos frames e, consequentemente, impedindo a leitura das teclas de controle que enviam comandos ao drone, sendo necessário reinicar o programa para voltar a funcionar corretamente.

O processo de decolagem e pouso do drone foi realizado várias vezes utilizando o teclado do notebook, portanto, com base nos resultados obtidos, podemos concluir que foi possível calibrar, ajustar e controlar o drone pelo computador de maneira eficaz.

4.3 SISTEMA DE POUSO AUTÔNOMO DO DRONE

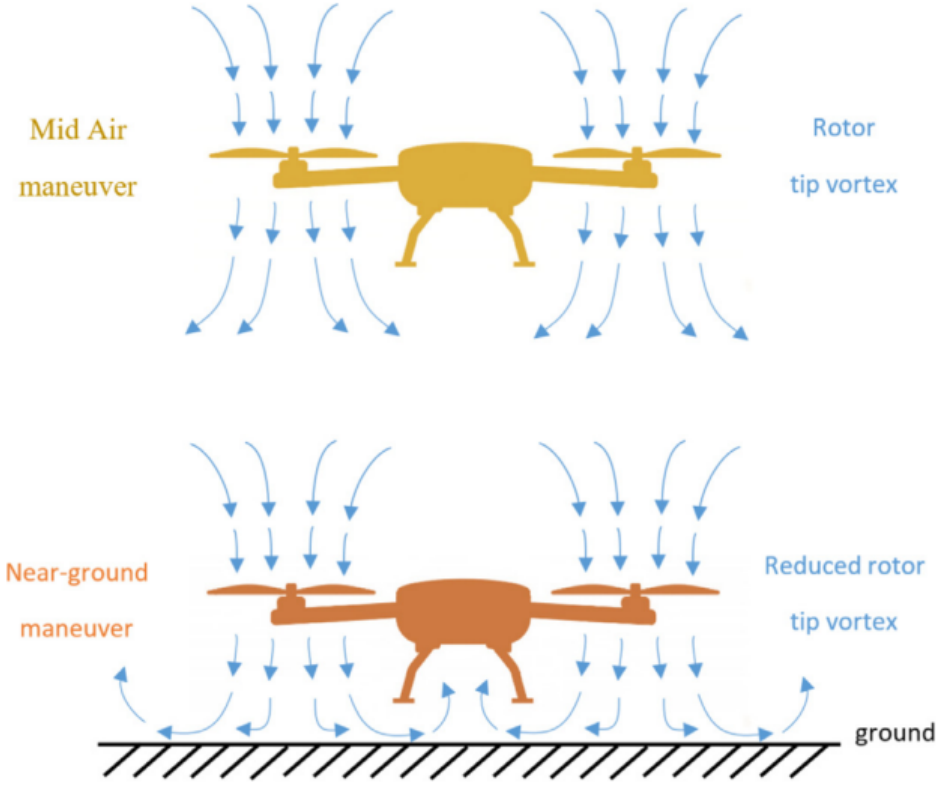
Nesta seção, é apresentado os resultados do teste proposto anteriormente para validar a transmissão, recebimento e processamento de imagens, juntamente com o envio de comandos ao drone. Ao final desse teste espera-se que seja possível validar o sistema de pouso autônomo do drone.

Ao longo da implementação foram encontrados desafios significativos ao tentar integrar esses dois sistemas. A principal dificuldade encontrada estava na tarefa de centralizar o drone sobre o alvo, a uma altura do chão de aproximadamente 1,25 metros. Isso ocorreu porque comandos fracos não eram capazes de conter a inércia gerada pelo drone, enquanto comandos mais fortes, devido à câmera estar fixa no drone, resultavam em uma perda de visão do alvo, uma vez que a câmera acompanhava o movimento do drone.

Tentamos algumas abordagens para conciliar esses sistemas, como ajustar a área de repouso na etapa de processamento de imagem e modificar a intensidade dos comandos enviados ao drone, porém não foi obtido sucesso. Dada a impossibilidade de centralizar o drone diretamente sobre o alvo e a proximidade da bola azul em relação ao centro do alvo, tornou-se inviável a tentativa de rotacionar o drone por meio de reconhecimento de imagem, que tinha como objetivo apontar a frente do drone para a bola azul.

Diante dos problemas encontrados, a solução pensada para pousar o drone via reconhecimento de imagem foi utilizar uma velocidade constante de descida, que ao se aproximar do solo o efeito aerodinâmico produzido pelo mesmo acaba aumentando sua sustentação causando uma perturbação no seu escoamento aerodinâmico como mostra a Figura 41, ou seja, o drone não consegue se manter fixo no chão.

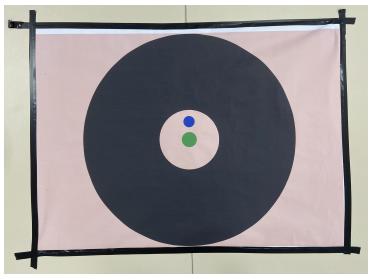
Figura 41 – Efeito solo.



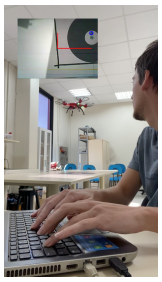
Fonte: Emran e Najjaran (2018).

Então, quando o drone reconhecer o alvo e determinar que está a uma altura adequada para pousar, o computador envia um comando de descida mais acentuado por um curto período de 0,12 segundos, seguido pelo comando de desligamento do drone, assim efetuando o pouso sobre o alvo. Com base nessa estratégia, realizamos no total 11 tentativas de pouso com o drone sendo apenas 5 delas bem-sucedidas, então coletamos informações sobre o tempo de descida entre o início do pouso até o envio do comando de desligamento e a distância do centro do alvo ao centro do drone nas amostras bem-sucedidas, conforme apresentado na Figura 42 e na Tabela 2.

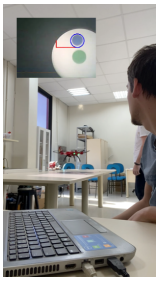
Figura 42 – Processo de aquisição dos dados.



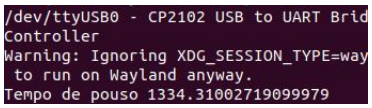
(a) Posicionamento do alvo.



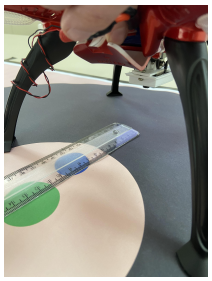
(b) Inicio do processo de pouso.



(c) Detecção da distância segura para realizar o desligamento do drone.



(d) Exemplo de aquisição do tempo pelo terminal.



(e) Medindo a distância entre o centro do drone e do alvo.

Fonte: Elaborado pelo autor (2023).

Tabela 2 – Resultados obitdos.

Medição	Distância (cm)	Tempo de pouso (s)
1	31,5	1,46
2	24,5	1,95
3	3,5	1,94
4	13	1,24
5	40,5	2,18

Fonte: Elaborado pelo autor (2023).

Analisando os resultados obtidos podemos afirmar que a distância entre o centro do drone e o alvo variou consideravelmente nas diferentes tentativas de pousar o drone autonomamente, variando de 3,5 cm a 40,5 cm, com uma mediana de 24,5 cm. Essa variação na distância estava prevista, uma vez que não foi possível centralizar o drone sobre o alvo. Da mesma forma, o tempo de pouso variou de 1,12 segundos a 2,06 segundos, com uma mediana de 1,94 segundos. Essa variação tem relação com a altura inicial do processo de pouso do drone.

A calibração dos limites de cores foi realizada apenas pela manhã para detectar separadamente os alvos verde e azul. No entanto, foi observado que as variações de luminosidade produzidas ao longo do dia afetavam os intervalos de cores percebidos pela câmera, especialmente considerando que a cor verde contém azul em sua composição. Como não foi feito a recalibração do filtro de limite de cores, notou-se

algumas situações em que o classificador podia detectar ambos os círculos, indicando a necessidade de recalibrar o filtro de limite de cores inferiores e superiores para garantir consistência nas detecções.

Os resultados obtidos na fase de pouso autônomo do drone indicam que foi bem-sucedida a implementação de todo o processo, desde a captura, transmissão, recepção, tratamento e processamento das imagens, e o envio de comandos ao drone, uma vez que foi possível reconhecer o alvo durante o pouso, identificar uma distância segura para pousar e enviar os comandos necessários para realizar o procedimento. Entretanto, os resultados também refletem as limitações do sistema, mostrando coerência com as adaptações propostas para o mesmo ser em parte realizável uma vez que não foi possível centralizar o drone sobre o alvo.

5 CONCLUSÕES

O desenvolvimento desse projeto proporcionou uma compreensão prática dos desafios envolvidos na implementação de sistemas de pouso autônomos em drones. Apesar dos desafios encontrados, os resultados obtidos fornecem uma base sólida para futuras melhorias e refinamentos.

5.1 CONSIDERAÇÕES FINAIS

Este trabalho abrange o processo de captação, transmissão e recepção de imagens utilizando o ESP32-CAM, a criação e implementação de classificadores Haar Cascade para a detecção do alvo proposto, juntamente com a automatização do controle do drone por meio de um computador, utilizando o hardware desenvolvido durante a IC, fundamentando as escolhas feitas, destacando os desafios enfrentados e apresentando os resultados obtidos.

Para o alvo proposto, os classificadores treinados com o tratamento de imagens por HSV mostraram ter uma distância máxima de detecção superior quando comparados aos classificadores treinados com o tratamento de imagens em escala de cinza, entretanto, necessita de uma calibração inicial do filtro de limite de cores superiores e inferiores para garantir uma consistência nas detecções. Entretanto, é válido destacar que ambos os classificadores, com seus respectivos tratamentos de imagem, conseguiram detectar o alvo, indicando que a escolha entre as abordagens pode depender da complexidade do alvo, das condições do ambiente e dos requisitos do projeto.

A abordagem inicial de dividir os testes entre a etapa de captação, transmissão, recepção e processamento de imagens, e a fase de controlar do drone pelo computador nos permitiu verificar e validar que os 2 sistemas funcionam corretamente. No entanto, para efetuar o pouso autônomo do drone por reconhecimento de imagem, é necessário integrar esses dois sistemas, então foi observado uma dificuldade em centralizar o drone sobre o alvo por meio do processamento de imagens, necessitando de novas abordagens para juntar esses sistemas e obter resultados bem-sucedidos.

O projeto contribui para a comunidade acadêmica com os relatos das experiências, sucessos e desafios enfrentados durante o desenvolvimento de pouso autônomo de drone via reconehcimento de imagem, servindo como um recurso educacional para estudantes e pesquisadores que desejam se envolver em projetos semelhantes.

5.2 TRABALHOS FUTUROS

Propõe-se investigar abordagens para conseguir centralizar o drone sobre o alvo durante o processo de pouso autônomo, como modificar a fixação do suporte do ESP32-CAM para garantir um movimento mais suave da câmera durante o deslocamento do drone, considerar a possibilidade de gerar pulsos para movimentar o drone e, após um intervalo de tempo, iniciar a busca pela localização do alvo. Além disso, testar estratégias para controlar o drone considerando fatores como momento de inércia do drone e tempo de resposta do sistema em si, podem ser uma alternativa a ser explorada.

Na parte de treinamento dos classificadores é possível realizar ajustes em diversos parâmetros visando melhorar a eficiência dos classificadores, sempre verificando o impacto dessas modificações no atraso do sistema. Na etapa de validação dos classificadores, a análise do desempenho pode ser estendida para diferentes resoluções de câmera, ambientes diversificados, aplicação de movimentos e angulações na camera durante a captação das imagens, proporcionando uma avaliação mais abrangente.

Por fim, é interessante considerar modificações no código do hardware desenvolvido para controlar o drone pelo computador, enfocando aspectos como a calibração e a implementação de testes em busca de otimizar a gestão do sistema de forma mais eficiente. Além disso, é essencial aprimorar a implementação do código para prevenir comportamentos inesperados, como as situações em que não era possível enviar comandos ao drone.

REFERÊNCIAS

ALEOTTI, J. et al. Detection of nuclear sources by uav teleoperation using a visuo-haptic augmented reality interface. **Sensors**, v. 17, n. 10, 2017. ISSN 1424-8220.

BAREISS, D.; BOURNE, J. R.; LEANG, K. K. On-board model-based automatic collision avoidance: application in remotely-piloted unmanned aerial vehicles. **Autonomous Robots**, v. 41, n. 7, p. 1539–1554, Oct 2017. ISSN 1573-7527.

BEN-MOSHE, B.; KELLER, A. A robust and accurate landing methodology for drones on moving targets. **Drones**, v. 6, n. 4, 2022. ISSN 2504-446X.

BRADSKI, G.; KAEHLER, A. **Learning OpenCV: Computer vision with the OpenCV library**. [S.l.]: "O'Reilly Media, Inc.", 2008.

CAMPOY, P.; MONDRAGÓN, I. F.; OLIVARES-MENDEZ, M. A. Autonomous landing of an unmanned aerial vehicle using image-based fuzzy control. **IFAC Proceedings Volumes**, v. 46, n. 30, p. 79–86, 2013. ISSN 1474-6670. 2nd IFAC Workshop on Research, Education and Development of Unmanned Aerial Systems.

CARVALHO, A. S. A. de et al. Análise comparativa entre os principais algoritmos de detecção facial: Haar cascade, hog, cnn, yolo e deepface. **OPEN SCIENCE RESEARCH V**, Editora Científica Digital, v. 5, n. 1, p. 439–454, 2022.

COLOMINA, I.; MOLINA, P. Unmanned aerial systems for photogrammetry and remote sensing: A review. **ISPRS Journal of Photogrammetry and Remote Sensing**, v. 92, p. 79–97, 2014. ISSN 0924-2716.

COSTA, T. L. et al. Modelagem e controle de um veículo aéreo não tripulado com rotores inclináveis e câmera orientável. 2019.

EMRAN, B. J.; NAJJARAN, H. A review of quadrotor: An underactuated mechanical system. **Annual Reviews in Control**, v. 46, p. 165–180, 2018. ISSN 1367-5788.

FAHLSTROM, P.; GLEASON, T. **Introduction to UAV Systems**. [S.l.]: Wiley, 2012. (Aerospace Series). ISBN 9781118396810.

FORSYTH, D.; PONCE, J. **Computer vision: A modern approach.** [S.l.]: Prentice hall, 2011.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. **Journal of Computer and System Sciences**, v. 55, n. 1, p. 119–139, 1997. ISSN 0022-0000.

FRIEDMAN, J.; HASTIE, T.; TIBSHIRANI, R. Additive logistic regression: A statistical view of boosting. **The Annals of Statistics**, v. 28, p. 337–407, 04 2000.

JOVANOVIC, M.; STARCEVIC, D. Software architecture for ground control station for unmanned aerial vehicle. In: **Tenth International Conference on Computer Modeling and Simulation (uksim 2008)**. [S.l.: s.n.], 2008. p. 284–288.

KHADKA, A. et al. Non-contact vibration monitoring of rotating wind turbines using a semi-autonomous uav. **Mechanical Systems and Signal Processing**, v. 138, p. 106446, 2020. ISSN 0888-3270.

KOTARSKI, D. et al. A modular multirotor unmanned aerial vehicle design approach for development of an engineering education platform. **Sensors**, v. 21, n. 8, 2021. ISSN 1424-8220.

KRISHNA, R. Computer vision: Foundations and applications. **Reference Book**, v. 213, 2017.

MINICHINO, J.; HOWSE, J. **Learning OpenCV 3 computer vision with python**. [S.l.]: Packt Publishing Ltd, 2015.

NATARAJAN, G. Ground control stations for unmanned air vehicles. Citeseer, 2001.

PEREIRA, R. C. **Técnica de rastreamento e perseguição de alvo utilizando o algoritmo Haar Cascade aplicada a robôs terrestres com restrições de movimento**. Dissertação (Mestrado) — Brasil, 2017.

PFEIFFER, C. et al. Visual attention prediction improves performance of autonomous drone racing agents. **PLoS ONE**, v. 17, n. 3, p. 1 – 16, 2022. ISSN 19326203.

PHASE, T. R.; PATIL, S. S. Building custom haar-cascade classifier for face detection. **Int. J. Eng. Res. Technol.(IJERT)**, v. 8, n. 12, p. 2278–0181, 2019.

PINA, R. G. Desenvolvimento de um veículo aéreo não-tripulado de segurança e monitoramento. Universidade Estadual Paulista (Unesp), 2022.

PASTOR QUILES, Y. **Object detection and tracking using an UAV**. 2018.

RAHMAD, C. et al. Comparison of viola-jones haar cascade classifier and histogram of oriented gradients (hog) for face detection. **IOP Conference Series: Materials Science and Engineering**, IOP Publishing, v. 732, n. 1, p. 012038, jan 2020.

SONKA, M.; HLAVAC, V.; BOYLE, R. **Image processing, analysis, and machine vision**. [S.l.]: Cengage Learning, 2014.

SOO, S. Object detection using haar-cascade classifier. **Institute of Computer Science, University of Tartu**, v. 2, n. 3, p. 1–12, 2014.

SYMATOYS. **MANUALS**. 2023. Disponível em: https://www.symatoys.com/downshow/x8hg-manual.html. Acesso em: 5 nov. 2023.

TAYEBI, A.; MCGILVRAY, S. Attitude stabilization of a vtol quadrotor aircraft. **IEEE Transactions on control systems technology**, IEEE, v. 14, n. 3, p. 562–571, 2006.

TEZZA, D.; LAESKER, D.; ANDUJAR, M. The learning experience of becoming a fpv drone pilot. **ACM/IEEE International Conference on Human-Robot Interaction**, p. 239 – 241, 2021.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: IEEE. **Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001**. [S.l.], 2001. v. 1, p. I–I.

APÊNDICE A - CÓDIGOS DESENVOLVIDOS DURANTE O TRABALHO

O código-fonte deste projeto está disponível no [GitHub](https://github.com/IldoJuninho/TCC-Drone).

Os códigos estão separados por pastas que indicam a seção onde os mesmos foram utilizados. Necessitam de modificações para funcionar pois foram movidos dos locais de onde foram implementados.