CENTRO PAULA SOUZA

FACULDADE DE TECNOLOGIA DE SANTO ANDRÉ

Tecnologia em Mecatrônica Industrial

Rafael Freitas Batista

Tabata Aparecida de Oliveira Silva

CONTROLE SEMIAUTÔNOMO PARA DRONE

Santo André - SP
2020

Rafael Freitas Batista

Tabata Aparecida de Oliveira Silva

CONTROLE SEMIAUTÔNOMO PARA DRONE

Monografia apresentada ao Curso de Tecnologia em Mecatrônica Industrial da FATEC Santo André como requisito parcial para obtenção do título de Tecnólogo em Mecatrônica Industrial.

Orientador: Prof. Dr. Edson Caoru Kitani.

Coorientador: Prof. Me. Murilo Zanini de Carvalho.

Santo André - SP
2020

FICHA CATALOGRÁFICA

B333c

Batista, Rafael Freitas.

Controle semiautônomo para drone / Rafael Freitas Batista, Tabata Aparecida de Oliveira Silva. - Santo André, 2020. – 55f: il.

Trabalho de Conclusão de Curso – FATEC Santo André. Curso de Tecnologia em Mecatrônica Industrial, 2020.

Orientador: Prof. Edson Caoru Kitani

1. Mecatrônica. 2. Drone. 3. Controle semiautônomo. 4. Visão computacional. 5. Estabilidade. 6. Processamento de imagens. 7. Landmark. 8. Sensores. 9. Tecnologia. I. Silva, Tabata Aparecida de Oliveira. II. Controle semiautônomo para drone.

629.892

Rafael Freitas Batista

Tabata Aparecida de Oliveira Silva

CONTROLE SEMIAUTÔNOMO PARA DRONE

Trabalho de Conclusão de Curso apresentado a FATEC SANTO ANDRÉ como requisito parcial para obtenção do título de Tecnólogo em Mecatrônica Industrial.

BANCA EXAMINADORA

Local: Fatec Santo André
Horário: 20:13
Data: 17/ 07/ 2020

Prof. Dr. Edson Caoru Kitani.
Presidente da Banca
Fatec Santo André

Prof. Me. Murilo Zanini de Carvalho.
Primeiro membro da Banca
Fatec Santo André

Prof. Me. Paulo Tetsuo Hoashi.
Segundo membro da Banca
Fatec Santo André

Santo André - SP
2020

AGRADECIMENTOS

Ao nosso orientador Prof. Dr. Edson Caoru Kitani pelo incentivo, visão das dificuldades e acompanhamento das soluções propostas ao longo da realização deste trabalho.

Ao nosso coorientador Prof. Me. Murilo Zanini de Carvalho pelo auxílio com os recursos para a realização deste trabalho.

Ao Prof. Me. Fernando Garup Dalbo por todas as orientações para adequar este documento dentro das normas ABNT.

A todo o corpo docente da Faculdade de Tecnologia de Santo André, por ter proporcionado a nós todo o conhecimento e as habilidades presentes no curso de tecnologia em Mecatrônica Industrial, que agregam um valor imensurável na nossa formação profissional.

A nossa família pelo apoio, incentivo e compreensão da nossa ausência em algumas ocasiões.

Aos nossos amigos por todo apoio e companheirismo que vivenciamos ao longo do curso.

O nosso muito obrigado!

“Mesmo no lugar mais escuro, a menor luz é a que mais brilha.”
(Kingdom Hearts)

RESUMO

O projeto tem como objetivo apresentar o processo do desenvolvimento de um algoritmo para estabilização de um AR.*Drone* com parâmetros de dois *landmarks* (pontos de referência) visuais. O meio de *interface* entre o *notebook* e o *drone* é pela *API* CV *DRONE*, que fornece comandos para controle e requisição de informações de sensores do *drone*, também possui integrada a biblioteca *OpenCV*. Com as técnicas de processamento de imagem e visão computacional contidas no *OpenCV,* o algoritmo tem a função de extrair informações da vídeo transmissão feita pelo *drone*, de forma que o algoritmo realiza a separação das cores mantendo as de interesse, e faz o reconhecimento do formato geométrico de um círculo, para assim interpretar os *landmarks*. Estes *landmarks* são dois círculos, um vermelho, que quando identificado o algoritmo faz com que o *drone* suba voo, e o outro azul que é a referência para estabilizar a posição do *drone* em termos de altura. O pouso é feito por meio de um contador incluso no algoritmo, durante o período que o *drone* permanece estabilizado dentro da faixa de tolerância pré-estabelecida.

Palavras Chave: *Drone*. Processamento de imagens. Visão computacional. CV *Drone*. *OpenCV.*

ABSTRACT

The Project “Semiautonomous control for drone” aims to present the proccess of developing an algorithm to stabilize an AR.Drone with parameters of two visual landmarks (reference points). The interface between the computer and the drone is through the API CV DRONE, wich provides commands for controlling and requesting informtion from the drone's sensors, with OpenCV's library integrated as well. With the image's processing and computer vision techniques inside the OpenCV, the algorithm has the function of extracting information from the drone's streaming vídeo, so that it can perform the separation of colors, keeping those of interest and distinguishing the circle's shape, for the landmark's interpretation. These landmarks are two circles: one red, which when it is identified makes the drone take flight; and the other blue, which is the reference landmark for the drone to stabilize its position in terms of height. The landing is done by a counter included in the algorithm, if the drone stays stabilized within the pre-established tolerance range.

Key words: Drone. Image processing. Computer vision. CV Drone. OpenCV

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVEATURAS E SIGLAS

ARM	*Advanced RISC Machine.*
ANAC	Agência Nacional de Aviação Civil.
API	*Application Programming Interface.*
CCD	*Charge Coupled Device.*
CMOS	*Complementary Metal Oxide Semiconductor.*
DDR	*Double-Data-Rate.*
DECEA	Departamento de Controle do Espaço Aéreo.
DHCP	*Dynamic Host Configuration Protocol.*
ESSID	*Extended Service Set Identifier.*
FAT	*File Allocation Table.*
FPS	*Frames Per Second.*
GPS	*Global Positioning System.*
HSV	*Hue Saturation and Value.*
IDE	*Integrated Development Environment*.
OpenCV	*Open Source Computer Vision Library.*
PIC	*Programmable Interface Controller.*
QVGA	*Quarter Video Graphics Array.*
RAM	*Random Access Memory.*
RGB	*Red, Green and Blue.*
SDK	*Software Development Kit.*
TCP	*Transmission Control Protocol.*
UAV	*Unmanned Aerial Vehicle.*
UDP	*User Datagram Protocol.*
USB	*Universal Serial Bus.*
VANT	Veículos aéreos não tripulados.
Wi-Fi	*Wireless Fidelity.*

SUMÁRIO

1. INTRODUÇÃO

Atualmente, diversas pesquisas têm sido feitas no campo de desenvolvimento e controle de *drones*. Devido a sua praticidade, o *drone* têm chamado a atenção de empresas que visam maior autonomia, como por exemplo, quando o objetivo da companhia é realizar entregas. Entretanto o veículo apresenta duas condições de operação que limitam sua utilização, sendo elas a eficiência energética e a outra a geolocalização (FREITAS, 2019; SARAIVA, 2018). O Quadro 1 um apresenta cinco modelos de *drone*, sua respectiva eficiência energética e custo médio.

Quadro 1 - Comparativo de autonomia

Modelo do *drone*	**Eficiência do sistema**	**Custo médio**
DJI *Phantom 4 PRO+*	Bateria com autonomia de voo de 30 minutos.	R$14.599,90
Mavic Pro Fly More Combo	Bateria com autonomia de voo de 27 minutos.	R$8.699,90
Spark Fly More Combo	Bateria com autonomia de voo de 16 minutos.	R$3.175,12
Syma X8hw	Bateria com autonomia de voo entre 5 e 7 minutos.	R$743,90
Ml2123 Fq777	Bateria com autonomia de voo entre 8 a 10 minutos,	R$389,90

Preços e modelos referentes ao acesso feito em: 20 de outubro de 2019.
Fonte: https://www.reviewbox.com.br/drone/

Como apresentado no Quadro 1, a eficiência energética de um *drone* é muito limitada, embora o *drone* DJI *Phantom* 4 *PRO+* apresente maior eficiência quanto aos demais, o custo médio dele é alto, tornando-se fator que pode ser considerado crucial na tomada de decisão das empresas em adquirir este tipo tecnologia (FREITAS, 2019). Quanto ao critério de geolocalização, os problemas de localização do *drone* no espaço são os recursos disponíveis como exemplo os sistemas de posicionamento global, *Global Position System (GPS)*, que utilizam os satélites e um receptor *GPS*, localizando a posição em função do deslocamento.

Tanto os satélites quanto os receptores possuem internamente um relógio, que operam em escala de nano segundos. A comunicação é feita por envio de sinais de radiofrequência, que percorrem numa velocidade equivalente à 300mil quilômetros por segundo quando percorrido no vácuo, acompanhados da informação do horário que este sinal foi encaminhado ao receptor. Quando este recebe o sinal encaminhado do

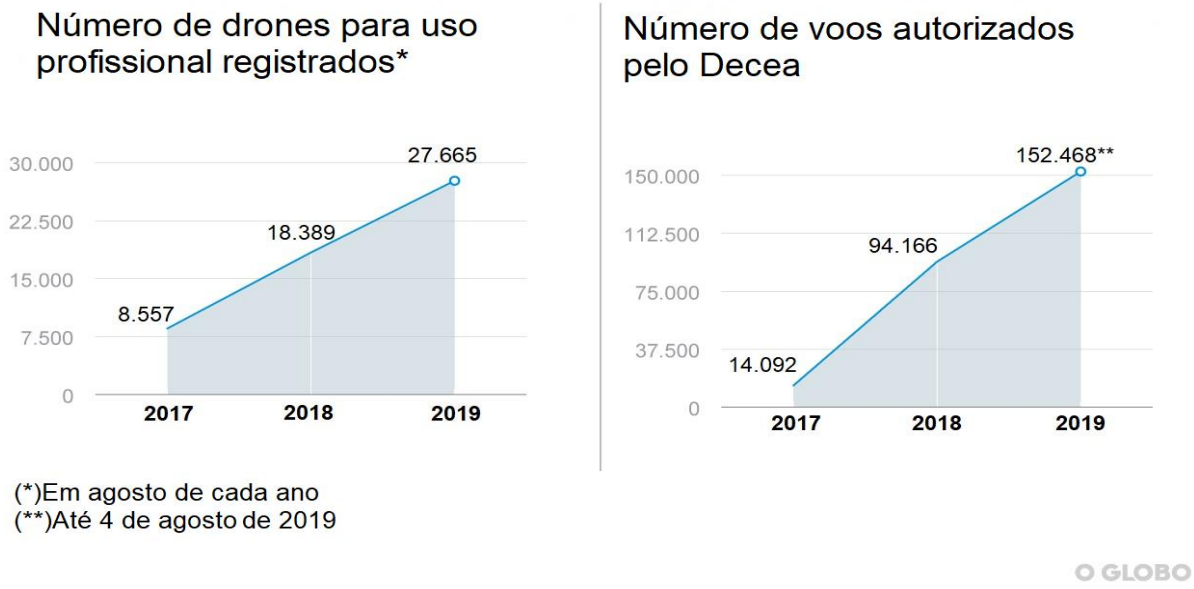
satélite, ele realiza o cálculo do *delay* (atraso) deste sinal e fornece a posição atual. Entretanto, os receptores *GPS* possuem uma margem de erro de posicionamento geográfico, podendo atingir à 5 metros.

Há também problemas com condições climáticas, que podem ocasionar maior atraso na transmissão e até mesmo a perda do sinal. É inviável para um *drone* operar em locais fechados com sistema de *GPS*, devido a precisão de um receptor com a margem de erro de 3 a 5 metros e ao *delay* que ocorre no momento da transmissão dos sinais. Este *delay* pode consumir tempo suficiente para que ocorra uma colisão, ou o *drone* estabelecer outra rota, enquanto o sistema de localização está recebendo o valor da posição estimada anterior (PHILIPE, 2016; SARAIVA, 2018).

1.1. Motivação

O mercado brasileiro de *drones* apresenta mais de 720 empresas envolvidas na cadeia produtiva desse mercado e, a atual empregabilidade dessa tecnologia é crescente, como apontados pelos números de autorizações de voos, indicados pela Agência Nacional de Aviação (ANAC) e pelo Departamento de Controle do Espaço Aéreo (DECEA) com objetivo profissional, indicado na Figura 1, sendo que foram solicitadas mais de 150 mil autorizações em 2019 (CAPETTI, 2019).

Figura 1 - Números apontados pela ANAC e DECEA



Adaptado de: https://oglobo.globo.com/economia/numero-de-drones-cresce-50-no-pais-ja-sao-usados-da-entrega-de-pizza-ao-transporte-de-sangue-23949123

Uma das aparições desta tecnologia com o âmbito profissional ocorreu no feriado do dia 1º de Maio de 2018, quando o edifício Wilton Paes de Almeida, que era

localizado no centro da capital paulista, sofreu um incêndio que ocasionou o desabamento do prédio. Uma equipe pioneira do projeto *Dronepol* em auxílio da Secretaria Municipal de Segurança Urbana, utilizou *drones* para fazer a procura de sobreviventes em meio aos escombros (RODRIGUES, 2018).

Outra aparição mais recente, é no atual cenário de pandemia da COVID-19. Pesquisadores da Universidade Federal de Pernambuco (UFPE) desenvolveram um *software* para que *drones* sejam controlados autonomamente e por meio de uma câmera térmica, eles fazem o monitoramento da temperatura das pessoas em multidões, e a distância (TARDE NACIONAL, 2020).

Dentre os segmentos da cadeia produtiva, o de soluções de *software* é promissor para evoluir no mercado de *drones*, pois os recursos mais usuais disponíveis para esta tecnologia possuem limitações, o que abre espaço para a realização de pesquisas e propostas de desenvolvimento, de forma que a tecnologia ganhe mais ramos de aplicações e maior acessibilidade.

1.2. Objetivo

Embora uma das limitações da tecnologia seja quanto a eficiência, o objetivo deste trabalho é o desenvolvimento de um algoritmo de controle para o *drone* operar de forma semiautônoma em ambientes fechados, sendo capaz de prover a estabilização de posição do *drone*, tendo como referencial um *landmark* (ponto de referência*)*, utilizando os recursos de visão computacional.

1.3. Metas

As principais metas deste projeto são:

- Realizar comunicação do *drone* com o *notebook*, apresentando os dados de navegação do *drone*;
- Desenvolver um algoritmo capaz de realizar a detecção de cores e identificação de um círculo, que será o *landmark;*
- Desenvolver um algoritmo com controle de comandos do *drone* à critério da cor identificada.

1.4. Organização do trabalho

Este trabalho está organizado da seguinte forma:

No capítulo 2, apresentamos os conceitos que foram necessários como fundamentação teórica para a realização deste trabalho, no capítulo 3 a metodologia de desenvolvimento do projeto, no capítulo 4 são apresentados e discutidos os resultados do desenvolvimento do trabalho e, finalmente no capítulo 5 as conclusões do projeto.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é abordado alguns temas relevantes e relacionados ao desenvolvimento deste projeto - baseados em pesquisas e estudos pertinentes ao mesmo.

2.1. Trabalhos relacionados

2.1.1. Sistemas de navegação e controle para veículos aéreos não tripulados e suas aplicações

Santana (2016) apresenta o desenvolvimento de um sistema de localização global para ambientes interiores, utilizando o método linear de fusão sensorial para rastreamento de dados de posição.

O método consiste em uma fusão dos dados obtidos pelo piloto automático do *drone* e as informações obtidas através da câmera frontal do *drone*, processada por um algoritmo de visão computacional, uma versão linear do filtro de Kalman, baseado em modelos matemáticos por meio de uma matriz de transição de estados, para realizar a estimativa do estado de voo do *drone* e então, atualizar a posição a partir dos dados obtidos pelos sensores.

Ele aponta que, uma vez que os dados tratados pelo piloto automático fornecem boas estimativas do seu movimento e possuem uma taxa de atualização relativamente rápida (no caso em seu desenvolvimento, ele aponta em torno de 200 Hz para o AR.*Drone* 2.0), torna-se possível estimar os dados necessários para obter o controle do *drone*, utilizando uma formulação cinemática.

Na utilização desse método as posições globais sofreram o efeito de deslizamento, que é onde a estimativa não se aproxima do valor real da variável, devido ao fato da velocidade ser considerada constante e modelada como um ruído branco e as posições serem estimadas através de integrações dos valores da velocidade registrada pelo piloto automático.

Entretanto em ambientes internos este problema foi solucionado utilizando as imagens da câmera frontal do *drone* e um algoritmo que reconheceria a imagem e recuperaria a posição relativa, entre a câmera e a imagem, relacionando as coordenadas da imagem com as coordenadas métricas tridimensionais do sistema global. A Figura 2 apresenta a captura da cena, com o elemento que fornece a referência de posição relativa e o resultado do tratamento de imagem que o autor

utiliza.

O método que o autor utiliza para a aplicação do filtro de Kalman é por meio da biblioteca *OpenCV,* com suporte da *interface* de programação de aplicativos, *Application Programming Interface* (*API), CV Drone.*

Figura 2 - Marcação visual sem tratamento e com tratamento

Adaptado de: (SANTANA, 2016).

2.1.2. Navegação autônoma, reconhecimento e desvio de obstáculos com *drones* multirotores

Saraiva (2018) é apresentado o desenvolvimento de um algoritmo de processamento de imagem capaz de realizar a identificação de uma linha, onde o objetivo era que o *drone* fosse capaz de seguir a trajetória da linha e realizar o reconhecimento e desvio de obstáculos ao longo do trajeto.

Com um fundo de alto contraste, ele utilizou o método de detecção de borda Canny, para o reconhecimento da linha, onde para utilizar este método é necessário realizar um pré-processamento da imagem. O pré-processamento consiste em: converter as cores da imagem e aplicar um filtro de desfoque. A conversão da imagem é feita de forma que o sistema ao captar uma imagem colorida, ou seja que possui os três canais de cores [*RGB*: *Red* (Vermelho)*, Green* (Verde) e *Blue* (Azul)], transforme a imagem em escala de cinza, dessa forma o sistema não consome recursos de forma demasiada e a resposta é mais rápida comparada quando é utilizado a imagem colorida.

Para realizar o desfoque, ele utilizou o filtro g*aussian blur,* este filtro utiliza a função gaussiana para distribuir em todas as dimensões a cor do *pixel* com a intensidade dada pela função, dessa forma o filtro realça o contorno da imagem e ajuda a eliminar ruídos. Com a detecção consolidada, ele utilizou o algoritmo *probabilistic Hough transform* para obter os pontos cartesianos que compõem o contorno capturado. Para o reconhecimento de objetos, ele adicionou a aplicação de algoritmos que fazem a dilatação e a erosão de uma imagem. O processo de dilatação

na imagem serve para eliminar pequenos pontos pretos, que são considerados ruídos dentro de uma área branca, deixando a imagem mais limpa. O processo de erosão é complementar à de dilatação, usado para a imagem fique mais delineada e com menor distorção. A Figura 3 apresenta a imagem original e o resultado da aplicação dos filtros citados.

Figura 3 - Aplicação dos filtros de imagem

Adaptado de: (SARAIVA, 2018)

Para finalizar o objetivo, Saraiva (2018) utilizou uma máscara para ocultar objetos não pertinentes ao trajeto da linha. O método foi realizado com o uso da biblioteca *OpenCV*.

2.2. *Drones* e Veículos Aéreos Não Tripulados

Devido a popularização, é muito comum ouvir falar sobre *drones* e Veículos Aéreos Não Tripulados (VANTs), ou do Inglês *Unmanned Aerial Vehicle* (*UAV*). Todavia o uso deste tipo de tecnologia não é algo recente. A primeira utilização deste tipo de veículo foi em 22 de agosto de 1849, quando a cidade de Veneza sofreu um ataque aéreo, que consistiu em balões carregados de explosivos.

As guerras na Europa foram o grande impulsionador do desenvolvimento de VANTs. Já no Brasil, o primeiro registro de desenvolvimento de um VANT ocorreu em 1982. O Centro Técnico Aeroespacial em um projeto conjunto com a Companhia Brasileira de Tratores e outras empresas, começaram a produzir *drones* para atender

as demandas militares no país. Os VANTs começaram a ganhar produção voltada para usos civis na última década, onde o principal modelo é o de asa fixa, e por isso é o que ganhou mais avanços e melhorias, sendo utilizado na construção civil, agropecuária e mineração. Existem outros exemplos para aplicações que são:

- Monitoramento de investimento;

Fornecimento em tempo real de cada fase do processo de construção, imagens detalhadas e dados para modelagem 3D.

- Logística médica;

Entrega de medicamentos em zonas rurais ou de difícil acesso.

- Monitoramento de risco;

O número de sensores capazes de serem instalados em *drones* permitem análises até mesmo para catástrofes naturais, também permitindo alertar qualquer civil na área afetada.

- Análises de solo e de campo;

Utilizando sensores óticos é possível fazer um mapeamento de solo e de lavoura, e com sensores térmicos e de umidade é possível saber quais partes da plantação que precisam ser regadas (PECHARROMÁN; VEIGA, 2016).

Além do modelo de asa fixa há os modelos de asas rotativas, também conhecidos como multirotores ou de forma popular, *drone*. A palavra *'drone'* vem do Inglês, e significa 'zangão'. Este nome foi adotado pelo barulho característico do modelo. No Brasil, o nome '*drone'* é associado a modelos multirotores que são fabricados com o foco de uso recreativo e os VANTs são associados a modelos de asa fixa e multirotores que tem o foco de produção para fins comerciais, ou seja, não recreativos (PECHARROMÁN; VEIGA, 2016; ITARC, 2018).

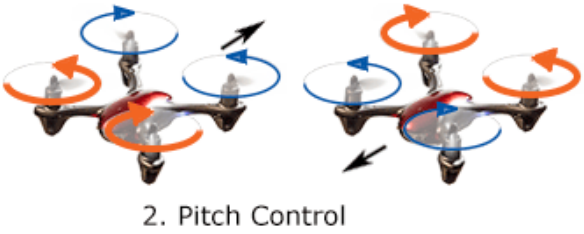
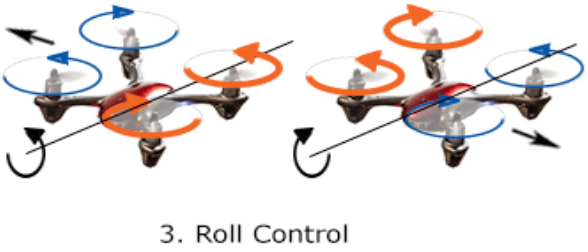
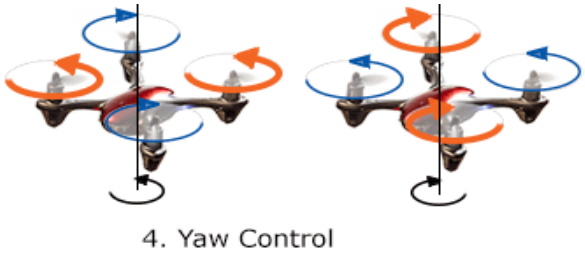
Multirotores apresentam benefícios quanto à aerodinâmica em relação aos de asa fixa, eles podem se deslocar em qualquer direção mesmo em baixas velocidades e também podem permanecer pairando no ar. Um multirotor é um modelo baseado no design de um helicóptero, cujo a força de propulsão é realizada pela quantidade de motores disponíveis no modelo.

O mais usual dos multirotores é o tipo quadrirrotor. Isto significa que ele possui quatro motores em sua estrutura. As hélices do *drone* são dispostas de forma horizontal, onde as adjacentes possuem sentidos de giro diferentes, para que não seja necessário compensar o momento angular dos propulsores. De forma dinâmica, ele apresenta seis graus de liberdade, correspondente à capacidade de se deslocar em

três dimensões. O modelo também é capaz de rotacionar em torno de três eixos imaginários. A Figura 4 é a representação dos graus de liberdade dos *drones*, ilustrado no modelo AR.*Drone*.

Os eixos imaginários são nomeados de vertical, longitudinal e transversal (ou lateral), dispostos perpendiculares entre si com o ponto de intersecção localizado no centro de gravidade do *drone*, originam os movimentos rotativos chamados *Pitch, Roll* e *Yaw* (SILVA, 2015; SALDANHA, 2019). O Quadro 2 ilustra e descreve os movimentos que um *drone* pode realizar.

Quadro 2 - Movimentos de um *drone*

Ilustração	Descrição
	O *Throttle*, ou aceleração, é o movimento utilizado para controlar a altitude do *drone*, através de um movimento linear vertical. Esse movimento se dá pelo aumento ou diminuição da rotação dos quatro motores simultaneamente.
	O *Pitch*, ou arfagem, é o movimento rotacional no eixo transversal, utilizado para controlar o ângulo de arfagem, baseado no centro do *drone*. Esse ângulo de inclinação permite que o *drone* se desloque para frente ou para trás.
	O *Roll*, ou rolagem, é o movimento rotacional no eixo longitudinal utilizado para controlar o ângulo de rolagem, baseado no centro do *drone*. Esse ângulo de inclinação permite que o *drone* desloque para a esquerda ou direita.
	O *Yaw*, ou guinada, é o movimento rotacional no eixo vertical, utilizado para o controle da orientação da guinada em sentido horário e anti-horário, baseado no centro do *drone*. Esse ângulo permite o movimento de rotação do *drone*.

Fonte do texto: (SANTANA, 2016)
Fonte das Figuras: https://www.translatorscafe.com/unit-converter/en/calculator/multicopter-lipo-battery/

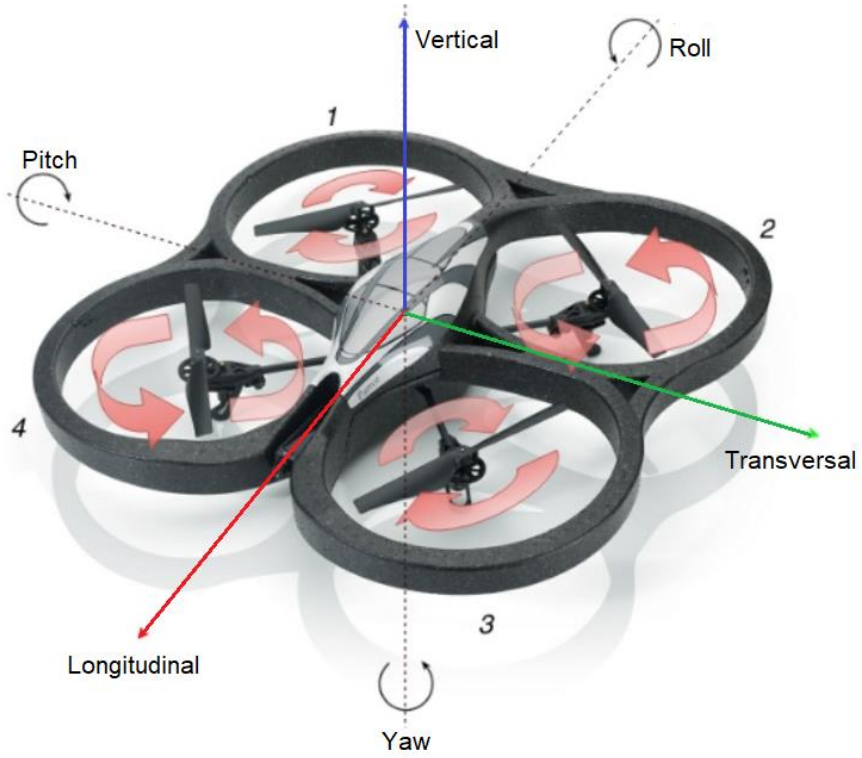
No Brasil, os órgãos regulamentadores de VANTs e *drones* são a ANAC, que é a responsável pela regulamentação e fiscalização das atividades de aviação civil

bem como a infraestrutura aeronáutica e aeroportuária do país, e o DECEA que é responsável pelo tráfego aéreo e serviços de navegação (ANAC, 2006; DECEA, 2001).

2.3.1. AR.*Drone*

O AR.*Drone* é um *drone* do tipo quadrirrotor, projetado pela empresa francesa *Parrot* que deu início ao projeto em 2004 e foi apresentado ao público em 2010, na exposição *Consumer Eletrônics Show.* Foi lançado ao mercado em agosto do mesmo ano. O foco da fabricação deste *drone* era que o produto ganhasse popularidade por meio de entretenimento, para uso recreativo, entretanto o modelo tornou-se popular também no meio acadêmico (BRISTEAU et al., 2011).

Figura 4 - Graus de liberdade



Adaptado de: https://www.slideshare.net/danishshrestha1/thesis2-71100803

Atualmente, o modelo está presente nas versões: AR.*Drone* 1.0, AR.*Drone* 2.0, AR.*Drone* 2.0 *Elite Edition* e o AR.*Drone* 2.0 *Power Edition.* As informações técnicas a seguir pertencem ao AR.*Drone* 2.0 *Power Edition.*

A eletrônica embarcada comporta; uma placa mãe incorporada a um processador Cortex A8 1*GHz* com o núcleo *ARM* de 32*bits* rodando a 800MHz com o sistema operacional Linux 2.6.32, que gerencia de forma simultânea diversos threads, uma memória *RAM DDR*2 1*GB* em 200MHz, também integrada com um chip *Wi-Fi.* A

placa de navegação composta por um microcontrolador *PIC* de 16*bits* em 40MHz, que realiza a *interface* com os sensores.

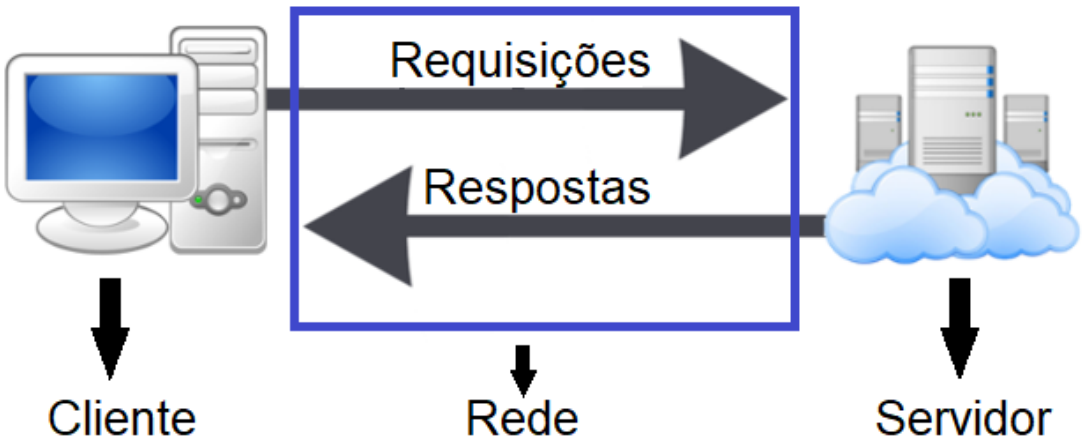
Estes são: acelerômetros de três eixos, com a precisão de aproximadamente 50mg, giroscópio de três eixos com precisão de 2,000°/s, magnetômetro de três eixos com a precisão de 6°, sensor de pressão com precisão de aproximadamente 10Pa e um sensor ultrassônico.

Estes sensores compõem uma unidade de medida inercial que opera na faixa de 200Hz. Há disponível um conector *USB* padrão, pode ser usada para salvar arquivos de gravação de vídeo em unidades de memória compatível com *USB*, em formato *FAT*32. E o módulo *GPS* da própria fabricante.

Também há integrado duas câmeras, uma orientada de forma horizontal com a resolução *HD* 720p (1280 x 720), com abertura de 92º grande-angular diagonal, à 30 *frames* por segundo (*FPS*). A outra de forma vertical, com resolução *QVGA* (320 X 240), à 60 *FPS*, para medir a velocidade em relação ao chão. Possui quatro motores sem escova do tipo '*inrunner*", com 14.5W de potência (PARROT, 2014).

A conexão com o AR.*Drone* é feita por meio de uma rede *Wi-Fi*, no qual o tipo de arquitetura da aplicação é denominado de cliente servidor. De forma sucinta, o modelo consiste em elementos fornecedores de recursos para a rede, este é denominado como servidor. Porém, o servidor só atua quando recebe uma requisição, de um outro elemento que por sua vez, é denominado como cliente. A Figura 5 ilustra de forma básica este processo (CANAL TI, 2018).

Figura 5 - Modelo Cliente Servidor



Adaptado de: https://www.gta.ufrj.br/ensino/eel878/redes1-2016-1/16_1/p2p/modelo.html

O cliente pode ser um computador, *notebook, tablet* ou *smartphone* que tenha suporte a rede *Wi-Fi*. A conexão com o servidor é estabelecida por um *ESSID*, por meio da rede “ar*drone*2”. Em questão ao AR.*Drone*, o servidor *DHCP* fornece o endereço IP e acesso as portas *TCP* e *UDP*. Dentre as portas disponíveis, estão:

- *UDP* 5556

Esta porta é designada para recebimento dos comandos AT. Estes comandos realizam o controle do *drone*.

- *UDP* 5554

Esta porta é designada para *navdata*. Ela contém as Informações de status, posição, velocidade, velocidade de rotação do motor e outras informações fornecidas pelos dos sensores.

- *TCP* 5555

Esta porta é designada para a transmissão de vídeo, em tempo real. Para esta operação, para o cliente é necessário o uso de um *codec*.

- *TCP* 5559

Esta porta é designada para recuperar dados de configuração e informações importantes, como o envio de informações de configuração (PARROT, 2012).

2.3. Ferramenta de integração computacional

Embora a fabricante forneça o kit de desenvolvimento de *software*, *Software Development Kit* (*SDK)*, que se trata de um código fonte disponível na linguagem de programação em C/C++, ele está restrito a desenvolvedores que utilizam o sistema operacional Linux. Demais desenvolvedores independentes criaram *interfaces* e plataformas de programação, tanto em C/C++ quanto em outras linguagens de programação, como Java Script e Python, que também são compatíveis com outros sistemas operacionais, como o *Windows*. Uma das *interfaces* é a *API CVDrone* (SANTANA, 2016).

Uma *API* é uma *interface* que realiza a integração de um *software* com outro, fornecendo a interação entre os sistemas de maneira rápida e segura, independente da linguagem de programação de cada um (FERNANDES, 2018).

O *CV Drone* é uma *API* que fornece um meio de estabelecer a comunicação do *notebook* com o *drone* por meio de uma rede wireless, de forma que seja feito o envio

dos comandos, recebimento das informações dos sensores e transmissão de vídeo, a ferramenta é composta por um conjunto de recursos, sendo eles:

- FFmpeg

É um *framework* multiplataforma para gravar, converter e transmitir áudio e vídeo. Este *framework* é capaz de fazer diversas tarefas com arquivos multimidia, como: codificar, decodificar, transcodificar, multiplexar, demultiplexar, fazer *stream*, filtro e reproduzir qualquer tipo de multimidia, indo dos formatos mais antigos até os mais atuais. E é uma ferramenta gratuita (BRITO, 2016).

- *OpenCV*

O *OpenCV* é uma biblioteca multiplataforma, para desenvolver aplicações na área de visão computacional. Desenvolvida pela Intel em 2000, a biblioteca possui um conjunto de funções para manipular e processar imagens digitais, também disponível de forma gratuita (BARELLI, 2018).

- *Pthreads*

O POSIX threads, ou *Pthreads,* é uma biblioteca padrão para uso, criação e gerenciamento de *threads*. Um *thread* é um fluxo de controle de execução. Diferente do conceito de processo, que de forma básica se trata de um agregador de recursos, como códigos e dados, os *threads* são criados no contexto de um processo compartilhando o mesmo espaço de endereçamento, que não são independentes.

Dentro de um sistema, eles funcionam como um subsistema dentro do programa. A sua aplicação pode ser do tipo *multithread,* que é quando se executa o processamento de múltiplas tarefas no mesmo programa, ou *single thread* onde só existe uma única linha de execução e somente uma tarefa pode ser executada de cada vez (GARCIA, 2017).

- *Sockets*

Para comunicar duas aplicações diferentes e permitir que elas troquem informações por meio de uma rede, é necessário que haja um mecanismo de transporte de dados. Um dos mecanismos disponíveis é o *socket*. O s*ocket* também é uma *API,* quando é utilizada para realizar programas é comumente utilizada na arquitetura cliente/servidor.

O cliente e o servidor trocam mensagens através de *sockets* de forma que o cliente se conecta ao servidor e inicia um laço para envio e recebimento das mensagens específicas da aplicação. Existem dois tipos de sockets, do tipo *TCP/IP* e do tipo *UDP/IP* (PANTUZA, 2017).

2.4. Visão Computacional

A visão computacional é a tecnologia que se baseia no funcionamento da visão humana. O ser humano possui um sistema visual que depende da luminosidade para que os olhos sejam capazes de captar imagens do ambiente. O processo consiste quando a retina extraí a informação da cena visualizada e encaminha ao cérebro por meio do córtex visual.

Na região do córtex visual, essas informações são processadas, de forma que o córtex analisa a imagem, classifica os objetos e as suas respectivas dimensões. O âmbito dos anos 50 era criar sistemas capazes de replicar o funcionamento do olho, do córtex visual e reproduzir a forma que o cérebro funciona, entretanto eles não possuíam poder computacional para chegar ao devido resultado. Atualmente o objetivo que obteve mais sucesso foi o de criar sistemas de dispositivos que se assemelham aos olhos (DATA SCIENCE ACADEMY, 2018).

Dentre os dispositivos existentes para aquisição de imagens, encontram-se:

- Sensor de ultrassom: utiliza ondas de som em alta frequência para medir a distância de objetos em relação a sua posição. Muito utilizado em máquinas de ultrassonografia que fazem a aquisição de imagens da estrutura interna do organismo;
- Sensor Infravermelho: dispositivo que é sensível ao calor. Muito utilizado em câmeras de vigilância noturna pois todo corpo sólido a uma temperatura acima do zero absoluto emite radiação infravermelha. Essa radiação, invisível ao olho humano, pode ser captada por esses dispositivos e convertida em imagem;
- Câmeras digitais: com o mesmo princípio de funcionamento das câmeras fotográficas, que consiste em uma câmara com um pequeno orifício contendo uma lente convergente que permite com que a luminosidade externa se propague em seu interior, as câmeras digitais possuem um conjunto com lentes que conduzem a iluminação para o seu interior chegando aos sensores, que convertem a luminosidade em elétrons, propagando em cada célula da imagem.

Dos meios citados, daremos foco as câmeras digitais. O principal componente de uma câmera digital é o sensor, que pode ser do tipo de carga acoplada, *Charge Coupled Device (CCD),* ou semicondutor de óxido metálico, *Complementary Metal Oxide Semiconductor* (*CMOS)*.

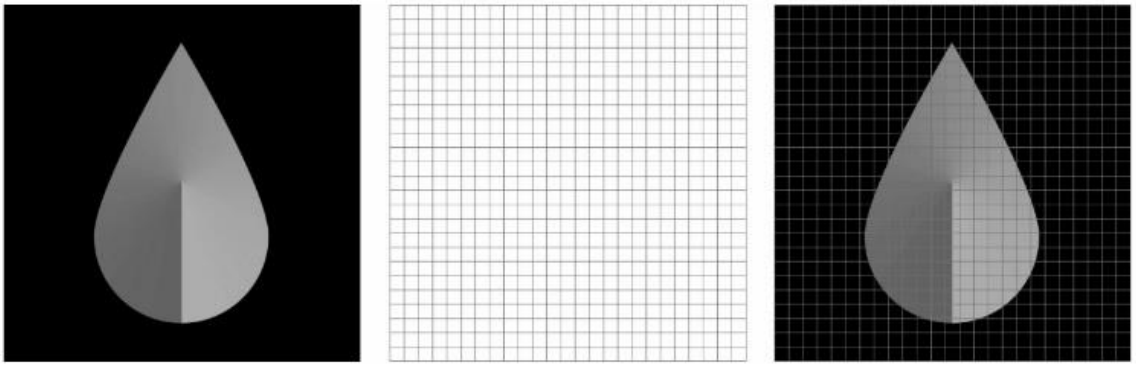
Os sensores *CCD* são compostos por uma matriz de capacitores fotossensíveis, cada célula da matriz ao receber a luz do ambiente, fornece um sinal de tensão analógico, proporcional à intensidade luminosa captada.

Este sinal é direcionado a um conversor analógico para digital (A/D), que finaliza o processo transformar iluminação ambiente em sinal digital. Os sensores *CMOS* possuem o mesmo princípio de funcionamento, entretanto sem a necessidade de um conversor A/D, pois há a presença de transistores nas células, fornecendo na saída o sinal digital. A diferença entre eles é que o *CMOS* possui menor custo de fabricação e menor sensibilidade luminosa, sendo mais propenso a produzir imagens com ruídos.

O sinal digital é composto por valor binário, que representa o valor do *pixel*. O *pixel* é um ponto luminoso, quando armazenado e ordenado junto a outros *pixels* em linhas e colunas, compõem uma imagem. Quanto maior o número de *pixels* ordenados, maior será a quantidade de detalhes que a imagem poderá exibir.

Esta quantidade de *pixels* determina a resolução espacial da imagem a ser formada. A Figura 6 é um exemplo do processo da formação da imagem, onde à esquerda é a representação da cena capturada, no centro é a matriz de células presentes no sensor e à direita é o resultado do sinal coletado em forma da união dos pontos luminosos, resultando a imagem da referência capturada (BARELLI, 2018).

Figura 6 - Demonstração dos *pixels*

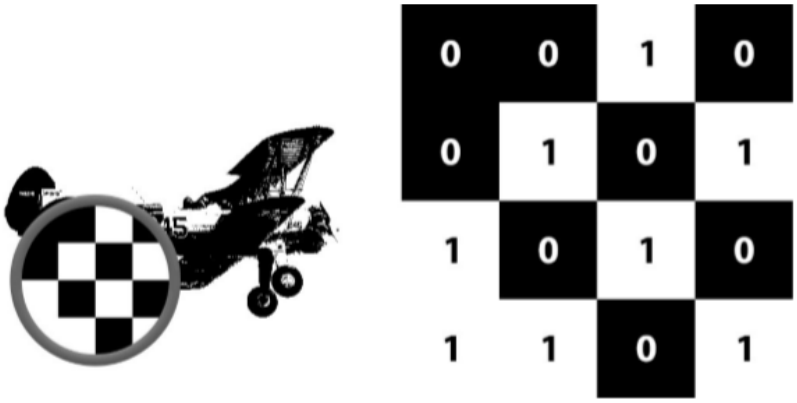


Fonte: (BARELLI, 2018).

Além da resolução espacial, há a resolução de intensidade. Esta por sua vez representa a variação perceptível de níveis de intensidade em uma imagem digital, como na quantidade de cor que cada *pixel* apresenta. Exemplificando, existem as imagens em preto e branco que são compostas apenas por dois *bits,* onde o *bit* zero representa o preto e o *bit* um representa o branco.

Ou seja, é uma imagem com a resolução de intensidade de dois *bits.* As imagens em preto em branco também são conhecidas como imagens binárias, e são a forma mais simples de se representar uma figura, pois os *pixels* variam em apenas dois tons de cor. A Figura 7 apresenta um exemplo de imagem digital em preto e branco e sua respectiva intensidade de bit.

Figura 7 - Intensidade de *bit* em preto e branco



Fonte: (BARELLI, 2018).

A imagem em escala de cinza é composta pela resolução de intensidade de 8 *bits*, onde o gradiente de variação do preto ao branco é representado pelos valores de 0 a 255, apresentando até 256 tons de cor diferentes, partindo do preto ao branco. A Figura 8 apresenta um exemplo de uma imagem em escala de cinza e sua intensidade de bit.

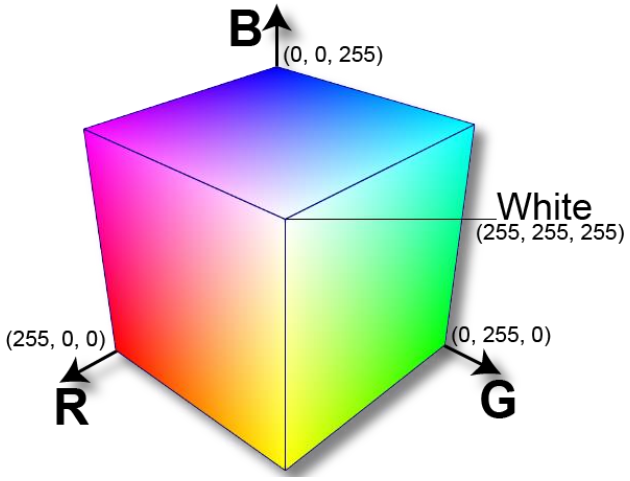
Figura 8 - Intensidade de *bit* em escala de cinza

Fonte: (BARELLI, 2018).

A imagem colorida é disposta em um espaço de cor composto por 24 *bits,* distribuídos em três canais distintos: R, G e B. Com a resolução de intensidade luminosa de 8 *bits* cada, onde os valores variam de 0 a 255, este espaço de cor é composto pelas cores; vermelho, verde e azul. Quando misturadas, em diferentes

intensidades representam as demais cores. De forma conceitual, uma imagem colorida pode ser representada por três matrizes bidimensionais, sendo cada matriz a representação de cada canal do espaço *RGB*. A Figura 9 representa o espaço de cor RGB por meio de um cubo (BARELLI, 2018; GONZALEZ; WOODS, 2010).

Figura 9 - Espaço *RGB*



Fonte: https://de.cleanpng.com/png-vkb15q/

Gonzalez e Woods (2010) apontam que em muitas aplicações, não somente em sistemas de visão computacional, não se faz o uso necessário de imagens com todas as intensidades presentes dentro do espaço *RGB*, devido a limitação de alguns dispositivos em exibir uma determinada variação de cores e também devido ao tempo de processamento dos dispositivos para toda gama de informação disponível dentro do espaço *RGB*. Citam também que para isto, são empregadas técnicas de processamento de imagens digitais para fazer o refinamento e separação dos aspectos interessados para a aplicação desenvolvida.

2.4.1. Processamento de imagens digitais

O impulso para o uso de técnicas de processamento de imagens digitais é fornecer a melhora das informações de forma visual, processamento dos dados para transmissão, armazenamento e representação de uma imagem. As técnicas são divididas em tipos de categoria de processamento, sendo processos de; baixo, médio e alto nível.

Um processo de baixo nível é caracterizado como o tipo de processo que recebe como entrada uma imagem e devolve em seu término, uma imagem. Das técnicas empregadas encontra-se métodos de pré-processamento, são filtros que são utilizados para exemplo: reduzir ruídos, realçar detalhes da imagem e a mudança no

contraste. Os processos de médio nível recebem uma imagem e retornam características extraídas da imagem. Um dos métodos presentes é o processo de segmentação, que consiste em separar regiões ou objetos de uma imagem. Já os de alto nível são caracterizados por realizar análises do que compõe a imagem e desta forma retornar todo um conjunto rico de detalhes. Este nível apresenta uma linha tênue entre o processamento de imagem e a visão computacional (GONZALEZ; WOODS, 2010).

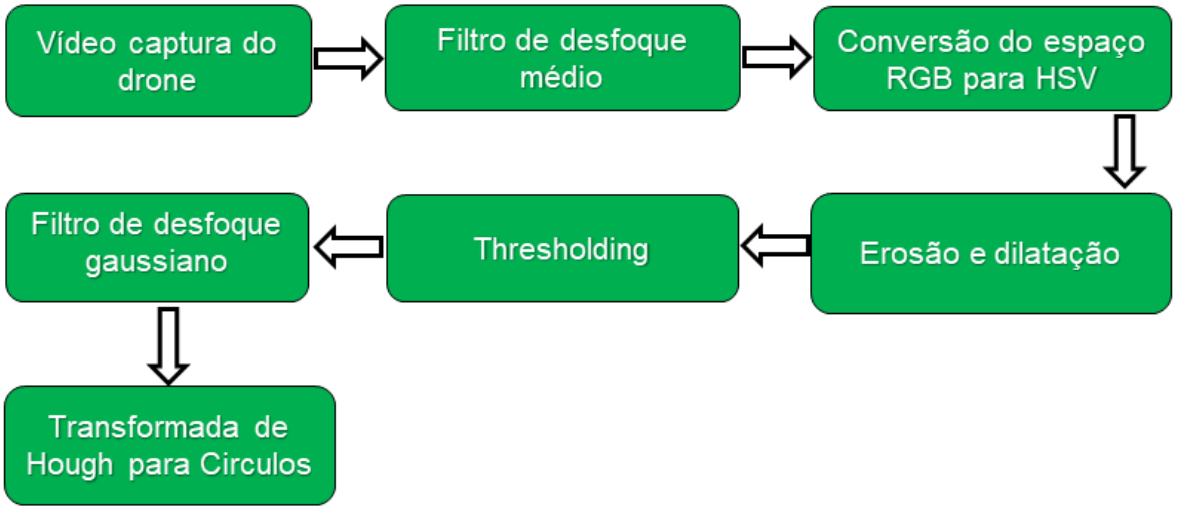
3. METODOLOGIA

Utilizamos a *API CVDrone* por meio do ambiente de desenvolvimento integrado, *Integrated Development Environment (IDE), Visual Studio da* Microsoft na versão 2017, devido a compatibilidade com a *API.* Separamos o desenvolvimento do projeto em duas etapas, a parte de visão computacional, que engloba todo o processo de tratamento de imagem e a parte de controle do *drone* por meio dos comandos que a *API* fornece.

3.1. Aquisição e tratamento de imagem

Este tópico apresenta o conceito dos recursos que utilizamos para o desenvolvimento do algoritmo deste projeto. A ordem do processo está apresentada na Figura 10.

Figura 10 - Fluxo do processamento da imagem



Fonte: Autores, 2020.

O resultado de cada etapa do fluxo de processamento encontra-se no capítulo quatro.

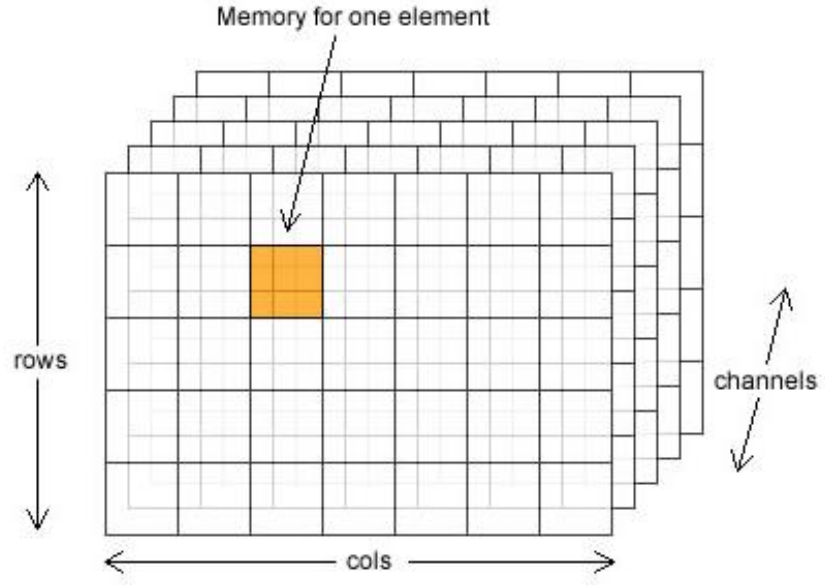
3.1.1. Vídeo captura

Por meio da função ‘ar*drone.getImage*()’ obtemos os *frames* transmitidos da câmera do *drone*. Associamos essa função em dois elementos de classe cv::Mat(), comando do *OpenCV,* de forma que um dos elementos tem por objetivo manter os dados originais e exibir em tela a transmissão, e o outro para todo o tratamento de imagem.

Conforme especificado em *Opencv* (2019b) a classe cv::Mat() de forma simplificada, é composta por duas partes de dados. Uma delas é denominado como

cabeçalho de matriz, que comporta informações como: tipo da matriz, método usado para armazená-la e o endereço de memória. E a outra é um ponteiro para armazenar os valores dos *pixels* compostos na matriz. A Figura 11 apresenta um exemplo de modelo de estrutura de um elemento Mat().

Figura 11 - Estrutura de um elemento Mat



Fonte: https://aishack.in/tutorials/2d-matrices-cvmat-opencv/

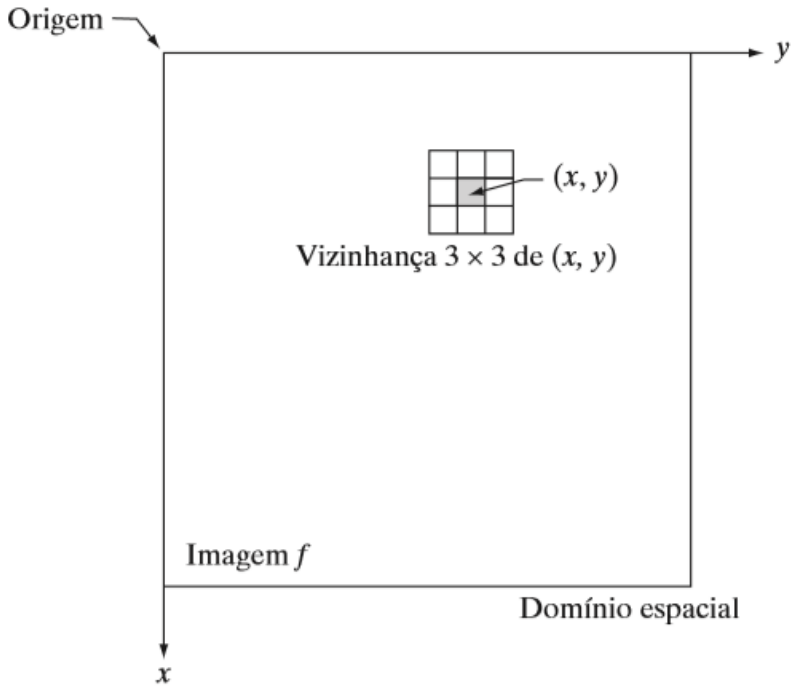
3.1.2. Filtro de desfoque médio e gaussiano

O filtro de desfoque médio, também conhecido como filtro de média, é denominado como um filtro espacial de suavização do tipo linear. Isto significa que, ele realiza a varredura da imagem e modifica os valores dos níveis de intensidade dos *pixels* com o valor da média dos vizinhos do elemento de âncora, que é o elemento central da área de um *kernel*, ou máscara de filtragem. Exemplificando, a Figura 12 apresenta o domínio espacial, que é a seção do plano expandido pelas coordenadas de uma imagem, o *kernel* é representado pelo conjunto (x, y) e os quadrados à sua volta representam os chamados vizinhos.

O resultado do filtro médio de desfoque é um efeito de borra na imagem que reduz os ruídos, como pequenos detalhes e descontinuidades, apresentando como resultado uma suavização de forma que os contornos ganhem um formato mais realçado, entretanto os objetos da imagem passam a ter menor definição. É necessário definir o valor do tamanho da área de *kernel*. O filtro gaussiano também é um filtro espacial de desfoque, porém funciona de uma maneira diferente do filtro

médio. Ele utiliza a função de desvio padrão nos *pixels* vizinhos do elemento de âncora do *kernel*. A vantagem de usar este filtro é que a imagem perde menos definição nos detalhes dos contornos. Além de definir o valor do tamanho da área de *kernel*, é necessário definir o valor de desvio padrão em X e Y. Outros valores necessários para a função são calculados automaticamente baseados no tamanho da área de *kernel* definida (GONZALEZ; WOODS, 2010).

Figura 12 - Representação do filtro espacial



Fonte: (GONZALEZ; WOODS, 2010).

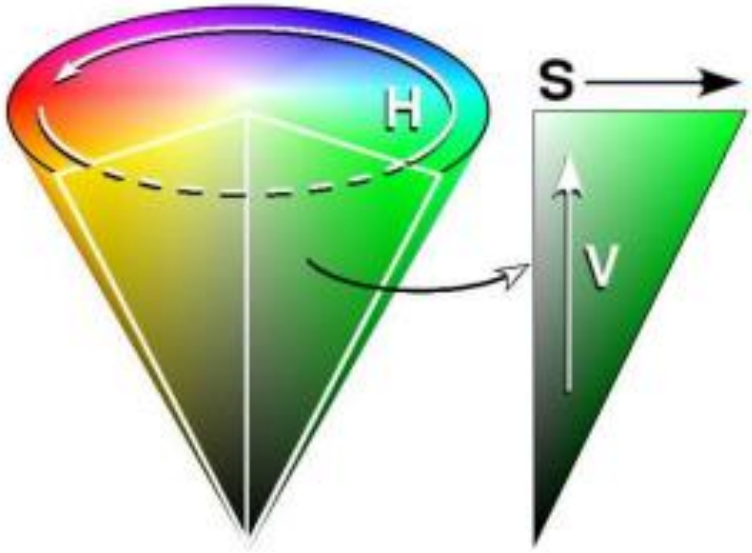
3.1.3. Conversão do espaço *RGB* para *HSV*

O espaço *HSV* é um modelo de representação de cor diferente do espaço *RGB*. Todavia também comporta três canais, que representam os elementos: H (*Hue* – Matriz), S (*Saturation* – Saturação) e V (*Value* – Valor). A principal diferença deste modelo é que ele fornece um melhor recurso para processos com a separação de cores, sendo um dos principais modelos para processos de segmentação, que é o procedimento de separar objetos de interesse, com foco de representação de cores em sistemas de visão computacional. Este modelo de cor é caracterizado por definir a cor pelo seu tom e brilho e não pela soma de vermelho, verde e azul presente na imagem.

A Matriz (H) representa a tonalidade da cor, sendo o parâmetro que informa ao sistema se a cor de interesse é o azul ou vermelho, de forma que a cor não tenha

influência de tons de preto ou branco. A Saturação (S) representa um aspecto do modelo que nos passa a sensação de uma cor “viva” ou não. Quanto maior for o valor da saturação, mais a cor será representada de forma brilhante e quanto menor mais fosca. E o Valor (V) representa a escala de claridade da cor, quanto maior for o valor, maior será a claridade contida na imagem. A Figura 13 apresenta a forma geométrica do espaço *HSV*, tornando de forma visual a influência de cada parâmetro (BARELLI, 2018).

Figura 13 - Modelo do espaço *HSV*



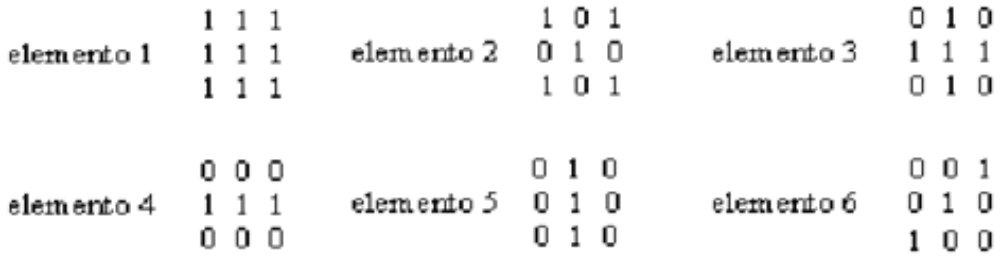
Fonte: http://www.idroneexperience.com/docs/Manual_processamento_imagem.pdf

O procedimento de segmentação como consiste em realizar a separação de um objeto de interesse é utilizado nesta aplicação pois o objetivo é separar as cores vermelho e azul, que são as cores dos *landmarks,* por meio do espaço *HSV*.

3.1.4. Erosão e dilatação

Ambos são operadores de processamento morfológico, muito utilizados no processamento de imagem, não apenas para eliminar ruídos, mas também para separar estruturas segmentadas isoladas ou uni-las. Ambos procedimentos necessitam do uso de um elemento estruturante. O elemento estruturante é uma matriz que quando declarada deve receber os valores de dimensão e formato, com o objetivo de realizar uma varredura nos dados da imagem e modificar o contorno da área de interesse. Os tipos mais comuns são mostrados na Figura 14. A função deste contorno é definida pelo tipo de operação morfológica.

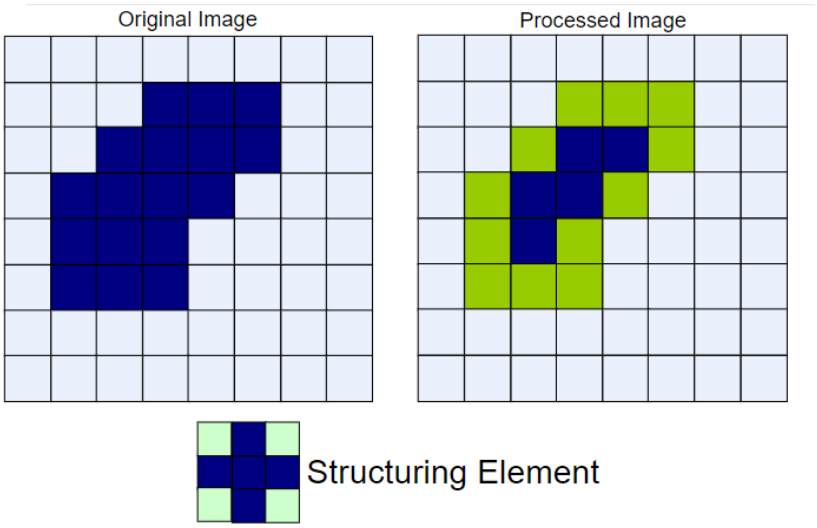
Figura 14 - Modelos mais comuns de elemento estruturante



Adaptado de: http://www.dpi.inpe.br/spring/portugues/tutorial/filtragem.html#:~:text=Para%20filtros%20morfol%C3%B3gicos%2C%20as%20m%C3%A1scaras,da%20mediana%2C%20eros%C3%A3o%20e%20dilata%C3%A7%C3%A3o.

O processo de erosão da imagem consiste na varredura do elemento estruturante por todos os pontos pertencentes a imagem, de forma que o encontro do *pixel* do elemento estruturante com o *pixel* do ponto de interesse da imagem, remova o *pixel* de contorno do objeto da imagem. Na Figura 15 temos o exemplo deste processo, onde os *pixels* azuis representam a região de interesse, os *pixels* verdes são os que foram removidos pelo processo de erosão e o elemento estruturante apresentado no formato de cruz (BARELLI, 2018; GONZALEZ; WOODS, 2010).

Figura 15 - Processo de erosão

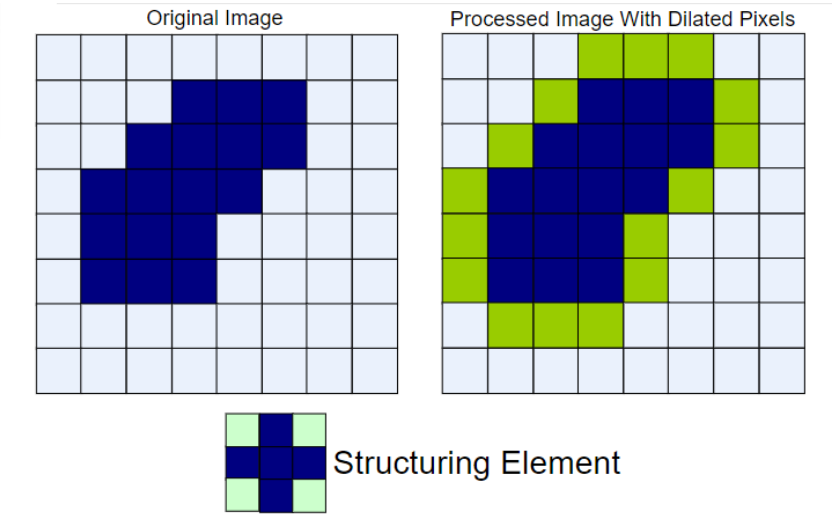


Fonte: https://slideplayer.com/slide/5097426/

O processo de dilatação é o oposto da erosão, consiste na varredura do elemento estruturante por todos os pontos pertencentes a imagem, porém ao encontro do *pixel* do elemento estruturante com o *pixel* do ponto de interesse da imagem, ele adiciona *pixels* na superfície do contorno do objeto na imagem. Na Figura 16 temos o exemplo deste processo, onde os *pixels* azuis representam a região de interesse, os

pixels verdes são os que foram adicionados pelo processo de dilatação e o elemento estruturante apresentado no formato de cruz (BARELLI, 2018; GONZALEZ; WOODS, 2010).

Figura 16 - Processo de dilatação

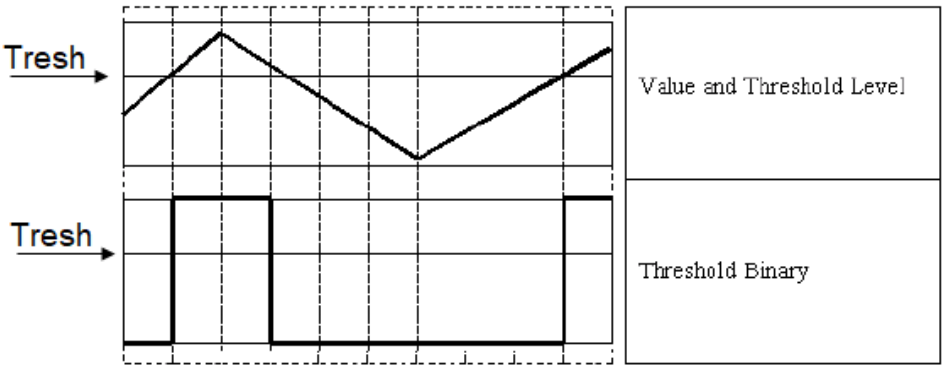


Fonte: https://slideplayer.com/slide/5097426/

3.1.5. *Thresholding*

Também é uma técnica utilizada para segmentação, todavia é por meio de um limiar (*tresh*). O procedimento requer que o valor de *tresh* e o valor máximo, recebam um valor de zero a 255, assim o sistema realiza a comparação de níveis de intensidade, de *pixel* para *pixel* contidos da imagem, e separa a região de interesse. Dentre os métodos de aplicação de *threshold,* o mais simples de utilização é chamado de *Binary Thresholding.* A Figura 17 demonstra de forma gráfica o resultado do método *binary threshold* (MALLICK, 2015).

Figura 17 - *Binary Thresholding*

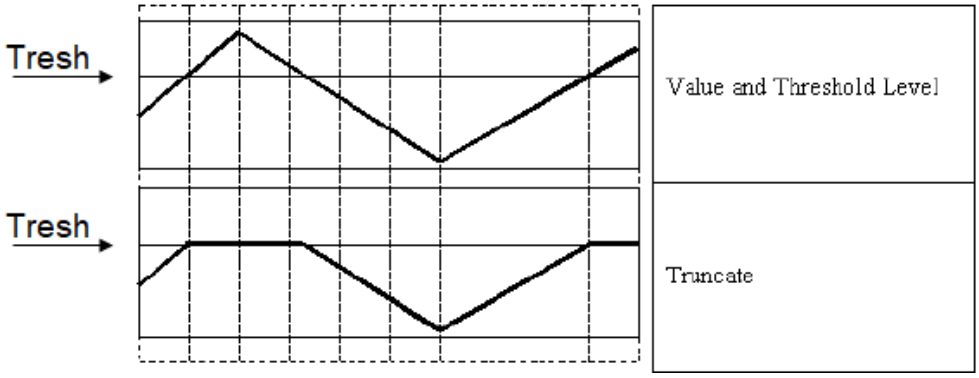


Adaptado de:
https://docs.opencv.org/trunk/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576ac7e89a5e95490116e7d2082b3096b2b8

Quando aplicado a uma imagem, os valores de *pixel* da imagem maiores que o *thresh* definido, receberam valor máximo, e os valores de *pixel* menores que o *thresh* definido, recebam valor mínimo, ou seja, este método resulta a segmentação em uma imagem binária.

O método que utilizamos é o *Thresh Trunc*, A Figura 18 demonstra de forma gráfica o resultado deste método.

Figura 18 - *Truncate Thresholding*



Adaptado de:
https://docs.opencv.org/trunk/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576ac7e89a5e95490116e7d2082b3096b2b8

Quando aplicado a uma imagem, os valores de *pixel* da imagem maiores que o *thresh* definido, receberam o valor de *Tresh*, e os valores de *pixel* menores que o *thresh* definido permanecem inalterados, ou seja, este método resulta a parte segmentada em escala de cinza (MALLICK, 2015).

3.1.6. Transformada de Hough para identificação de círculos

De forma geral, a transformada de Hough é um método utilizado para identificar formas geométricas, que possam ser representadas de forma paramétricas, em imagens digitais. Dessa forma, é possível definir o mapeamento entre o domínio de espaço da imagem e o espaço de parâmetros. Hough fez o uso da equação denominada *declive-intercepte*, definida pela função: y = ax +b, para representação paramétrica de uma linha. Este método requer o uso de um acumulador matricial, pois para cada seguimento de linha encontrado dentro do espaço de imagem requer encontrar também dois parâmetros para os seguimentos: a e b. O procedimento faz com que cada pixel seja examinado e tenha os parâmetros específicos dos valores calculados extraídos pela equação.

Após calculados, esses parâmetros de um dado *pixel* atribuem um valor para os parâmetros de a e b, e o acumulador sofre um incremento. Após toda a varredura

na imagem, o sistema busca no acumulador os maiores valores, pois eles são parâmetros que indicam os prováveis segmentos de linha da imagem.

Para a identificação de formas circulares, é utilizado um espaço parametrizado para ordenar os parâmetros de referência que definem um a forma geométrica de um círculo, que são: as coordenadas X e Y do centro do círculo, que dentro do espaço de parâmetros são adotadas como a e b, e seu raio (JAMUNDÁ, 2000).

No *OpenCV*, a função encontra círculos em imagens em escala de cinza, devido ao método *"gradient method".* Como parâmetros, ele fornece o tamanho do raio do círculo identificado e os valores posição do centro do círculo, em X e em Y (OPENCV,2019a).

3.2. Comandos da *API* para controle do *drone*

A *API* dispõe comandos para fazer a requisição de informações e envio dos comandos AT. Para maior compreensão, listamos os utilizados no projeto, apresentados no Quadro 3 e no Quadro 4, seguido da descrição de cada comando utilizado no algoritmo.

Quadro 3 - Comandos de requisição

Função	Descrição
open()	Inicializa a comunicação de envio dos comandos para o *drone*.
close()	Finaliza a comunicação de envio dos comandos
getImage()	Requisita o envio dos frames das câmeras do *drone*.
getVersion()	Requisita a atual versão de firmware.
getPitch()	Requisita valores do ângulo rotativo no eixo transversal, em radianos.
getRoll()	Requisita valores do ângulo rotativo no eixo longitudinal, radianos.
getYaw()	Requisita valores do ângulo rotativo no eixo vertical, radianos.
getBatteryPercentage()	Requisita a porcentagem de carga da bateria.
getAltitude()	Requisita os valores de altitude, em metros.

Fonte: (PUKU0X, 2016).

Para os comandos de movimentação, a Figura 19 apresenta o sistema de coordenadas referenciais da *API* para controle do *drone*.

Figura 19 - Sistema de coordenadas

Fonte: https://github.com/puku0x/cvdrone/wiki/FAQ

Fazendo uma associação com o sistema de coordenadas, o ângulo rotativo *Pitch* é correspondente ao eixo Y, *Roll* ao eixo X e *Yaw* ao eixo Z.

Quadro 4 - Comandos de controle

Função	Descrição
takeoff()	Comando de decolagem do *drone*.
landing()	Comando de pouso do *drone*.
emergency()	Comando de emergência, para colisões ou baixa porcentagem da bateria, o comando força o pouso do *drone*.
move()	Comando de movimentação do *drone* no plano 2D, mantém o *drone* pairando em Z.
move3D()	Comando de movimentação no plano 3D.
onGround()	Comando de verificação de estado de repouso do *drone*
setFlatTrim()	Comando de calibração no plano horizontal, referenciados nos controladores internos de inclinação e rotação.
update()	Atualiza a versão do *drone*.
setCamera()	Faz a mudança dos canais da câmera.

Fonte: (PUKU0X, 2016).

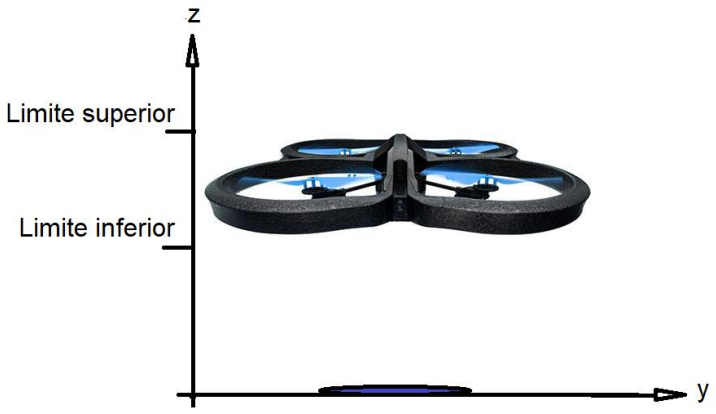
3.3. Convenção de operação do projeto.

Define-se Manual como “feito com a mão” (MANUAL, 2020). E define-se autônomo como “que determina suas próprias normas; que não depende de outro” (AUTÔNOMO, 2020). Ou seja, manual é aquilo que é executado com as mãos e autônomo é aquilo que é provido de mecanismo capaz de executar ou regular uma ação requerida, dispensando operadores.

Para o projeto, podemos dizer que manual seria a situação onde o *drone* é controlado por uma pessoa, enquanto autônomo seria um *drone* capaz de funcionar sem estar sendo controlado por alguém. O motivo de desenvolver um algoritmo para o controle semiautônomo foi adquirir e desenvolver maior conhecimento sobre *drones* e visão computacional, deixando de entrar em um critério mais complexo, que é o controle totalmente autônomo.

A ideia do projeto é, quando o círculo vermelho for identificado, o sistema enviará comando para o *drone* subir e trocar o canal da câmera. Com o círculo azul colado no chão, a estabilização da altura do *drone* será feita por meio dos valores de do raio do *landmark*, fornecidos pela transformada de Hough em tempo real. O *drone* deve permanecer dentro de uma faixa de estabilidade que será definida entre os valores de dois limites, superior e inferior, como exemplificado na Figura 20.

Figura 20 - Modelo do projeto



Adaptado de: https://lojab2b.comercialfoto.pt/lojas-marca/outlet/drones/art-PF721003BJ/parrot-ar.drone-2-power-edition.aspx

Enquanto o *drone* permanece dentro da faixa de estabilidade, o sistema irá iniciar um contador, que contará por até meio minuto. Após a contagem, o sistema envia o comando de pouso. Limitamos o tipo de controle de estabilidade em relação a altura por questões de indisponibilidade de espaço. A navegação do *drone* é feita de forma manual, e não está inclusa no projeto desenvolvido.

4. RESULTADOS

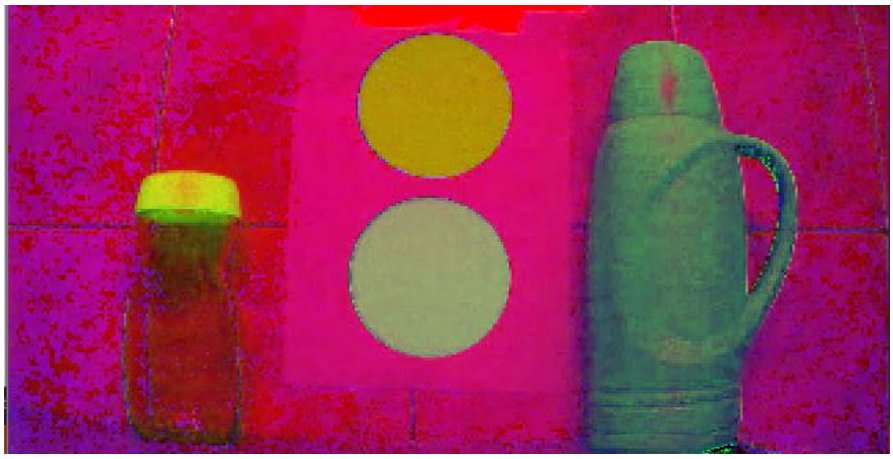
Com o *drone* imóvel, fizemos a aquisição das imagens para maior detalhamento do resultado de cada processo do tratamento de imagens. A imagem original, representada na Figura 21, contém os *landmarks* que utilizamos para controle do *drone* (Sendo o vermelho para subir voo e o azul para se estabilizar), uma garrafa verde e um vidro com a tampa amarela. A seleção dos objetos para a imagem foi feita de forma a exemplificar o processo de segmentação da cor.

Figura 21 - Imagem referência em *RGB*

Fonte: Autores, 2020.

O espectro de cor dentro do espaço *HSV* é diferente do *RGB*. A Figura 22 representa o mesmo cenário da Figura 21, todavia dentro do espaço *HSV*.

Figura 22 - Conversão em *HSV*



Fonte: Autores, 2020.

A Figura 23 apresenta o resultado da aplicação do filtro de desfoque médio dentro do espaço *HSV*. Percebe-se uma suavização na imagem, comparado a Figura 22, sobre efeito quadriculado e o ressalto das bordas de contorno de cada objeto.

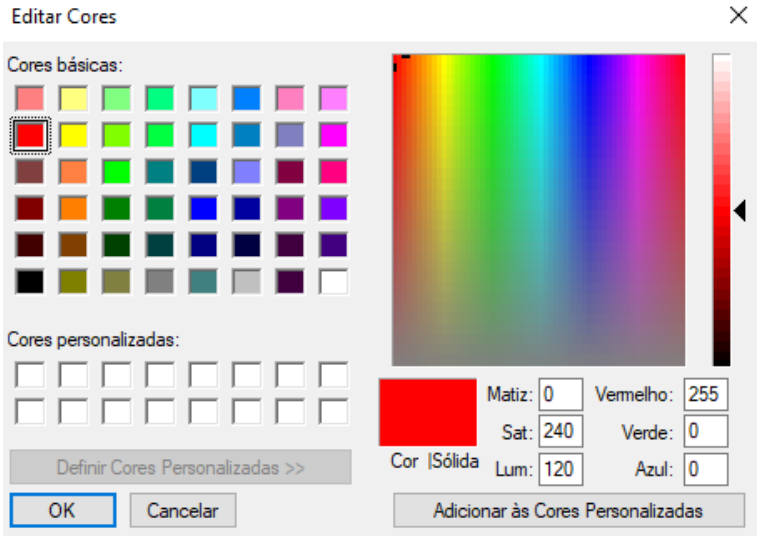
Figura 23 - Filtro de desfoque médio

Fonte: Autores, 2020.

O processo de segmentação é feito pelo comando *inRange*, onde dentro do espaço *HSV* ele faz a separação da imagem com os parâmetros de matriz, saturação e valor definidos por um escalar (OPENCV, 201-).

Existem ferramentas de conversão das cores dentro do espaço *RGB*, que mostram o valor da cor convertido em outros espaços. O *Paint*, como demonstrado na Figura 24, apresenta os valores das cores tanto no espaço *RGB* quanto no *HSV*. Outras ferramentas também fazem a conversão da cor em escala hexadecimal.

Figura 24 - Tabela de cores do *Paint*

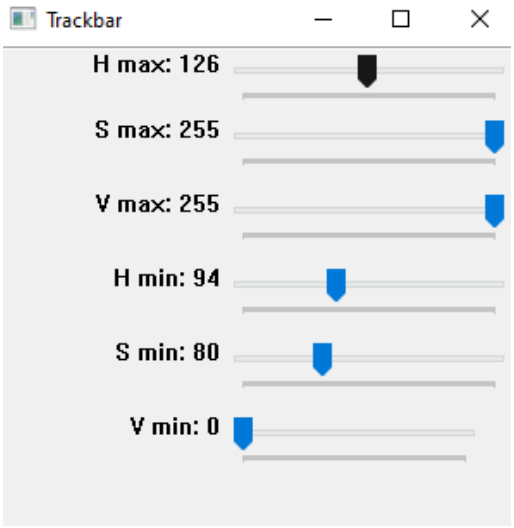


Fonte: Paint

Entretanto, o *OpenCV* requisita os valores de dois parâmetros de cada canal para fazer a segmentação, denominados *Upper* e *Lower* (OPENCV, 201-). Conforme demonstrado em Gupta (2017) uma forma de encontrar os valores desejados é utilizar um *TrackBar*, com os parâmetros de 0 a 255. Como especificado em Microsoft (2017)

um *TrackBar* é um recurso de controle que permite ao usuário configurar os intervalos desejados em uma faixa de rolagem, definindo os valores de máximo e mínimo. A Figura 25 apresenta o modelo de um *TrackBar*.

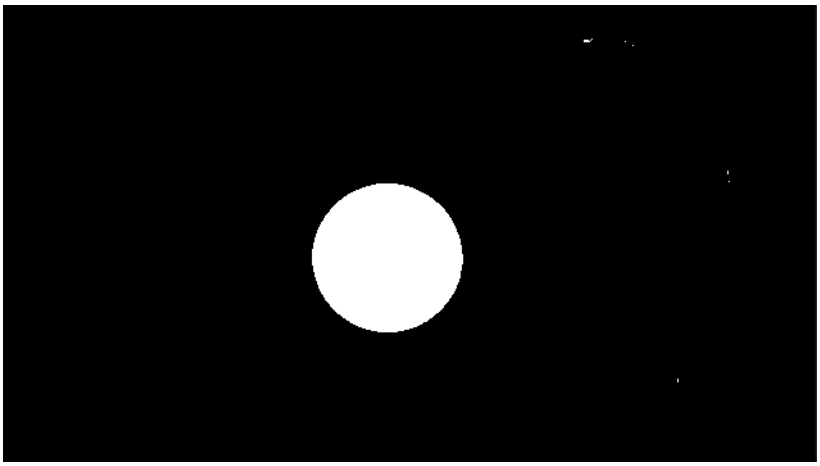
Figura 25 - *Trackbar*



Fonte: Autores, 2020.

Os valores exibidos na Figura 25 são correspondentes aos de segmentação da cor azul, onde os parâmetros com ‘max’ são correspondentes aos de *Upper*, e os ‘min’ de *Lower*. A Figura 26 apresenta o resultado da segmentação, onde área de interesse é a região branca.

Figura 26 - Azul segmentado



Fonte: Autores, 2020.

Também em Gupta (2017) aborda que um fator crucial para determinação dos parâmetros da cor desejada no espaço *HSV* é a luminosidade ambiente. Ele enfatiza que quanto mais luminosidade o ambiente possuir, maior será a quantidade de detalhes em resolução de intensidade na imagem, isto implica em uma maior

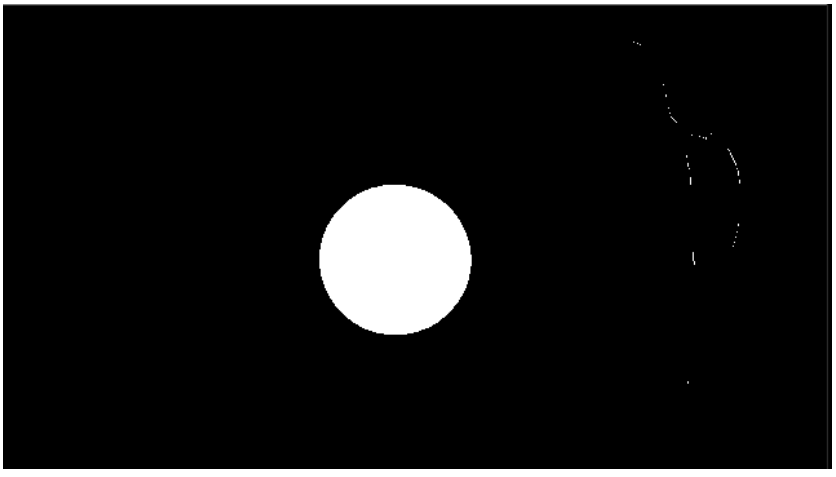
quantidade de ruídos, considere ruído qualquer contorno indesejado na imagem segmentada. A Figura 27 foi feita com maior luminosidade em relação a Figura 21, como resultado, apresenta um maior contorno da garrafa verde, ou seja, apresenta um maior ruído comparado a figura anterior.

Figura 27 - Azul segmentado com maior intensidade de luz

Fonte: Autores, 2020.

Uma das formas de diminuir os ruídos é fazer pequenas alterações dos parâmetros em *HSV* para a região segmentada. Os valores foram alterados para: Upper (Hmax:130, Smax:255, Vmax:255). E Lower (Hmin:100, Smin: 100, Vmin: 0). O resultado apresentado na Figura 28 corresponde aos valores citados.

Figura 28 - Redução dos ruídos.

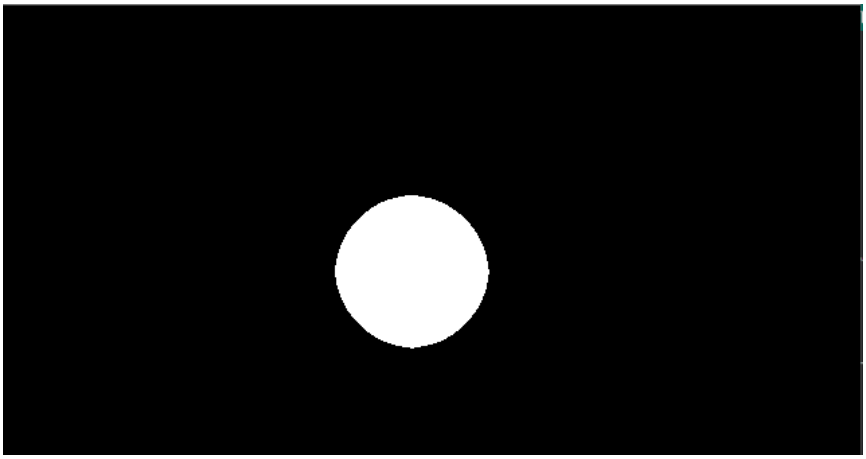


Fonte: Autores, 2020.

Embora com os parâmetros adaptados, ainda é visível um traçado singelo da garrafa verde. Para eliminar este traçado foi aplicado o método de erosão. Ponto importante, o elemento estruturante não pode obter um tamanho muito grande para situações que envolvem o formato geométrico como objetivo interessado, pois ele pode alterar muito o formato original do objeto. De forma que o círculo não sofra muita

mudança na proporção do original, aplicou-se o método de dilatação. A Figura 29 mostra o resultado da aplicação dos dois métodos.

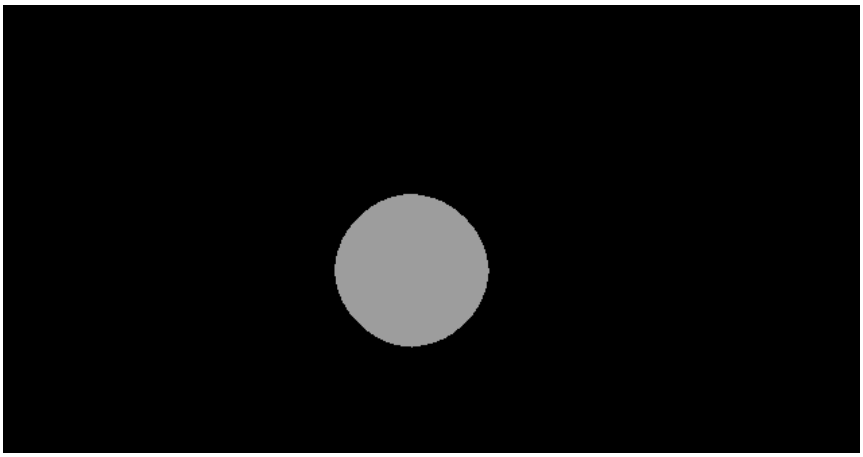
Figura 29 - Erosão e dilatação aplicados



Fonte: Autores, 2020.

Para a aplicação da transformada de Hough é necessário utilizar imagens em escala de cinza. Não há como converter a imagem segmentada em escala de cinza. O meio eficaz foi utilizar o método *truncate* de *threshold*, limitando o valor de *thresh* em um correspondente a escala de cinza. O método retorna à referência uma imagem em escala de cinza, como demonstrado na Figura 30.

Figura 30 - *Thresh Trunc* aplicado



Fonte: Autores, 2020.

Após a aplicação do filtro de desfoque gaussiano, para suavizar o delineado quadriculado referente ao resultado do processo de erosão e dilatação na imagem, é feito a aplicação da transformada de Hough para identificar círculos. Quando identificado o círculo, como demonstrado na Figura 31, o programa traça um contorno verde e um ponto no centro.

Figura 31 - Resultado dos círculos identificados

Fonte: Autores, 2020.

Ainda com o *drone* imóvel, definimos os valores de limite superior e limite inferior para a região de estabilidade. Colocamos o *landmark* azul que é para estabilidade, colado a uma mesa localizada no chão, e com uma fita métrica posicionamos o *drone* a 1,20m do *landmark,* para coletar o valor de raio do círculo azul, que é fornecido pela transformada de Hough em valores de *pixel*, e o valor medido pelo sensor de altitude do *drone* correspondentes a esta altura.

Decidimos adotar 1,20m para limite superior pois no início do projeto ocorreu que o *notebook* travou e parou de enviar comandos ao *drone*, por sorte o *drone* estava a uma altura acessível e nós conseguimos desligá-lo de forma forçada, através da desconexão da bateria.

E para limite inferior, definimos a 0,80m. Como estamos trabalhando com a referência visual, quanto mais próximo o *drone* estiver do *landmark,* maior será o valor de raio. E quanto mais distante, menor o valor de raio. Os valores correspondentes de limite superior em raio foi 30 em *pixel*, e o valor correspondente medido pelo sensor foi de 1,04m e limite inferior em raio foi 60 em *pixel*, e o valor correspondente medido pelo sensor foi de 0,62m. Também definimos uma distância mínima para o comando de *takeoff*(), quando o *landmark* está aproximadamente a 0,50m ou mais próximo do *drone*.

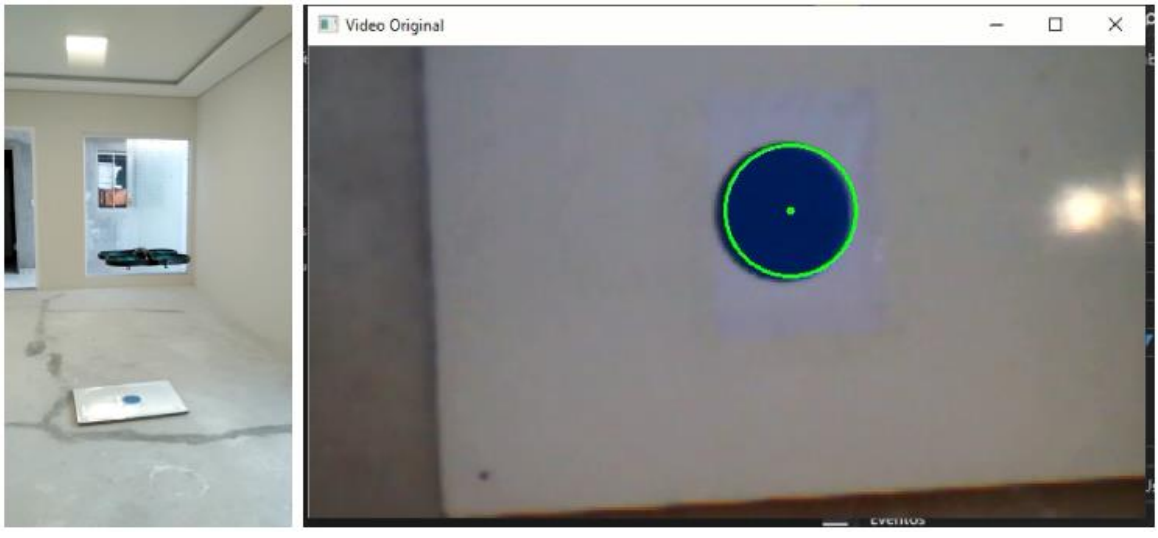
Para o teste, parametrizamos o contador para enviar o comando do *drone* pousar enquanto ele permanece dentro da região de estabilidade, por aproximadamente 25 segundos. A Figura 32 apresenta o momento do início do teste, quando ocorre o reconhecimento do círculo vermelho, que faz com que o *drone* suba voo.

Figura 32 - Comando de levantar voo

Fonte: Autores, 2020.

Como o AR.*Drone* não envia a transmissão das duas câmeras de forma simultânea, programamos que, no momento que o sistema identificasse o círculo vermelho, enviaria os comandos de subir voo e trocar o canal da câmera para estabilizar sua posição verticalmente. Por meio do console, acompanhamos e coletamos os valores de raio e altitude, fornecido pela função *getAltitude()*, em função do distanciamento do *drone* em relação ao *landmark*. O *drone* subiu e atingiu aproximadamente a 1,02m de altitude, valor medido pelo sensor, correspondente a 31 em *pixel* de raio, desacelerou permanecendo entre 0,80m e 0,87m de altitude. A Figura 33 demonstra o *drone* entrando na região de estabilidade.

Figura 33 – Estabilizado a altura



Fonte: Autores, 2020.

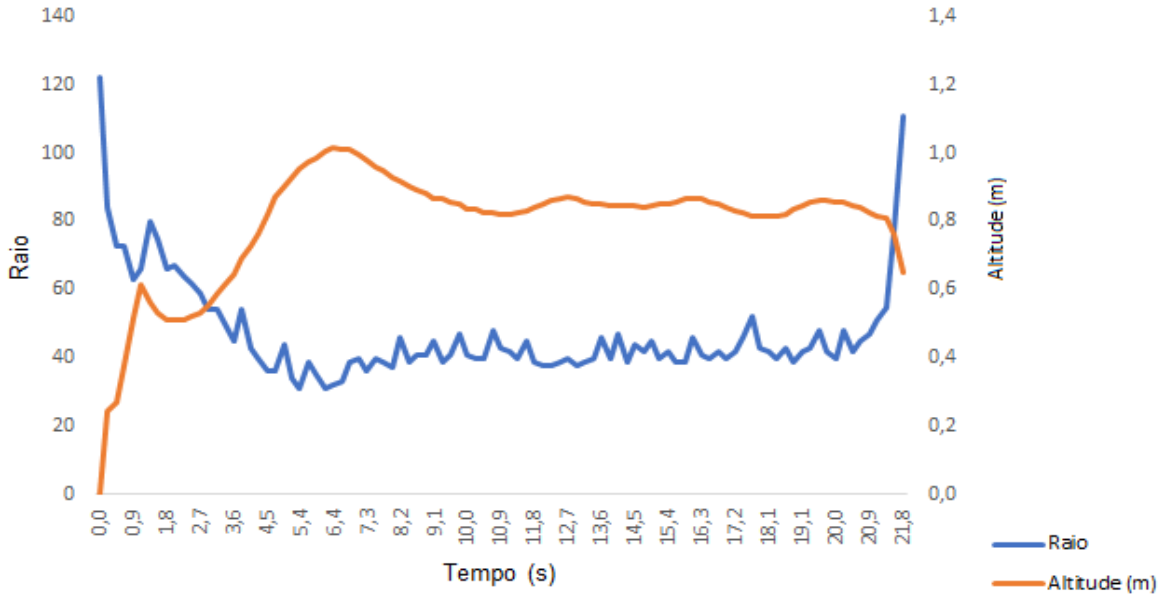
Após o tempo de contagem, o sistema enviou o comando de pouso, resultado na Figura 34.

Figura 34 - Pouso

Fonte: Autores, 2020.

Com os valores que coletamos referente ao período que o *drone* subiu e pousou, fizemos o gráfico da Figura 35 para demonstrar o resultado do comportamento do *drone* durante este período. Ressaltando que, a proporção de raio por altitude é inversa pois, quanto mais próximo do *landmark* o *drone* está, maior é o valor de raio e menor o valor de altitude.

Figura 35 - Resposta gráfica



Fonte: Autores, 2020.

As oscilações presentes no índice de raio são em função à margem de erro da precisão dimensional do círculo identificado. O tempo não está exato porque utilizamos um contador, todavia o projeto atendeu ao critério de permanecer dentro da região de estabilidade. No início do projeto utilizamos um temporizador construído com recursos da biblioteca “ctime” disponível para programação em linguagem C++.

Entretanto, quando contava o correspondente a 30 segundos, o *notebook* travava e não encaminhava mais comandos ao *drone*. A gravação do resultado está disponível em <https://www.youtube.com/watch?v=kxNg8NOMWeE>. As informações técnicas do *notebook* utilizado para a realização do projeto estão disponíveis em: < https://www.lenovo.com/br/pt/laptops/ideapad/serie-100/IdeaPad-110-15IBR-/p/88IP1000708>.

5. CONCLUSÃO

O desenvolvimento do projeto consistiu em duas partes, a escolha da técnica e desenvolvimento do algoritmo de visão computacional e o meio de *interface* com o *drone*.

A escolha de utilizar o recurso de segmentação de cores foi feita devido a capacidade de processamento computacional que tínhamos disponível, que embora conceitualmente simples apresentou resultados eficientes mediante ao nosso objetivo, estabilizar o *drone*.

O uso da *interface*, que foi feita por meio da *API,* beneficiou a construção do projeto, fornecendo os comandos de todos os recursos disponíveis para controle do *drone*, uma vez que o *SDK* da fabricante limita o acesso desses recursos para usuários *Linux*, trazendo só a *interface* básica para demais usuários, como *Windows*. Outro benefício de utilizar a *API* é que ela também traz integrado o *OpenCV,* biblioteca de código aberto para visão computacional.

A geolocalização de *drones* em ambientes fechados com o uso de sistemas de visão computacional tornam-se eficientes quando comparados a sistemas de *GPS* devido a variação da taxa de erro de proximidade de objetos ser menor. No caso do *GPS* o erro pode atingir uma faixa de até 5 metros. Entretanto, assim como o *GPS,* o meio de comunicação com o *drone* também está sujeito a perdas de sinal por se tratar de uma rede *wireless*. Este meio de comunicação é limitado quanto a distância que o dispositivo servidor se encontra do cliente. Uma das possíveis soluções para resolver este problema seria fazer o uso de repetidor de sinal *Wi-Fi*.

5.1. Propostas futuras

Como sugestão para trabalhos futuros, conforme o problema citado no capítulo um, fazer um estudo de caso com a modelagem matemática para o desenvolvimento de um *drone* que apresente maior eficiência energética à baixo custo.

Adaptar e implementar o algoritmo em outros modelos de *drone* ou em outra linguagem de programação que tenha compatibilidade com o *OpenCV*, por exemplo, Python.

Utilizar técnicas de *SLAM* para mapear a região com a câmera de um *drone*, e desenvolver um algoritmo de forma que o controle de navegação do *drone* passe a

ser autônomo.

Aplicar o método de segmentação pelo espaço *HSV* para identificação de cores, como apresentado nesta monografia, em veículos terrestres para identificação da sinalização de trânsito.

6. REFERÊNCIAS

ANAC. **Institucional**. Agência Nacional de Aviação Civil, 2006. Disponível em: <https://www.anac.gov.br/acesso-a-informacao/institucional>. Acesso em: 22 Jul. 2020.

AUTÔNOMO. Dicionário online Priberam, 2020. Disponível em: <https://dicionario.priberam.org/Aut%C3%B4nomo>. Acesso em: 22 Jul. 2020.

BARELLI, Felipe. **Introdução à visão computacional uma abordagem prática com Python e *OpenCV***. 1. ed. São Paulo: Casa do código, 2018. 256 p.

BRISTEAU, Pierre-Jean; CALLOU, François; VISSIÈRE, David; PETIT, Nicolas. *The navigation technology inside the AR.Drone micro UAV.* In*: PROCEEDINGS OF THE 18TH IFCA WORLD CONGRESS,* [s.n.]., 2011, *Milano (Italy).* **Anais eletrônicos** *... Milano:* IFAC, 2011. p. 1477- 1484. Disponível em: <http://www.asprom.com/drone/PJB.pdf>. Acesso em: 20 Abr. 2019.

BRITO, Edivaldo. FFmpeg é um verdadeiro canivete suíço para gravar, converter e transmitir áudio e vídeo. Techtudo, 2016. Disponível em: <https://www.techtudo.com.br/tudo-sobre/ffmpegl.html>. Acesso em: 20 Nov. 2019.

CAPETTI, Pedro. Número de *drones* cresce 50% no país e já são usados da entrega de pizza ao transporte de sangue. **O Globo,** Rio de Janeiro, 15 Set. 2019. Disponível em: <https://oglobo.globo.com/economia/numero-de-drones-cresce-50-no-pais-ja-sao-usados-da-entrega-de-pizza-ao-transporte-de-sangue-23949123>. Acesso em: 28 Out. 2019**.**

CANAL TI. Arquitetura cliente servidor. Canal TI, 2018. Disponível em: <https://www.canalti.com.br/arquitetura-de-computadores/arquitetura-cliente-servidor/>. Acesso em 18 Nov. 2019.

DATA SCIENCE ACADEMY. O que é visão computacional? 2018. Disponível EM: <http://datascienceacademy.com.br/blog/o-que-e-visao-computacional/> Acesso em: 25 Nov. 2019.

DECEA. **O DECEA**. Departamento de Controle do Espaço Aéreo, 2001. Disponível em: < https://www.decea.gov.br/?i=quem-somos&p=o-decea>. Acesso em: 22 Jul. 2020.

FERNANDES, André. O que é uma API? Entenda de maneira simples. Vertigo Tecnologia, 2018. Disponível em: <https://vertigo.com.br/o-que-e-api-entenda-de-uma-maneira-simples/>. Acesso em: 20 Nov. 2019.

FREITAS, Camila. *Drone*: Quais são os melhores modelos de 2019. Reviewbox, 2019. Disponível em: <https://www.reviewbox.com.br/drone/>. Acesso em: 20 Out.2019.

GARCIA, Fernando Deluno. *Threads Posix*. Embarcados, 2017. Disponível em: <https://www.embarcados.com.br/threads-posix/>. Acesso em: 20 Nov. 2019.

GONZALEZ, Rafael C.; WOODS, Richard E.. **Processamento digital de imagens.** 3. ed. São Paulo: Pearson Prentice Hall, 2010. 644 p.

GUPTA, Vikas. *Color spaces in OpenCV* (C++/Python). *Learn OpenCV*, 2017. Disponível em: <https://www.learnopencv.com/color-spaces-in-opencv-cpp-python/>. Acesso em: 18 Fev. 2020.

ITARC. História dos *drones*: como surgiram? Para que servem? Itarc, 2018. Disponível em: <https://itarc.org/historia-dos-drones/>. Acesso em: 18 de Jul. 2020.

JAMUNDÁ, Teobaldo. Reconhecimento de formas: a transformada de Hough. Seminário Visão Computacional - CPGCC/UFSC, 2000. Disponível em: <https://www.inf.ufsc.br/~aldo.vw/visao/2000/Hough/index.html#:~:text=A%20transformada%20de%20Hough%20(HT,comumente%20a%20detec%C3%A7%C3%A3o%20de%20bordas.>. Acesso em: 15 de Fev. 2020.

MALLICK, Satya. *OpenCV Threshold (Phyton, C++)*. *Learn OpenCV,* 2015. Disponível em: < https://www.learnopencv.com/opencv-threshold-python-cpp/#:~:text=Truncate%20Thresholding%20(%20type%20%3D%20THRESH_TRUNC%20),to%20the%20source%20pixel%20value.>. Acesso em: 15 Fev. 2020.

MANUAL. Dicionário online Priberam, 2020. Disponível em: <https://dicionario.priberam.org/manual>. Acesso em: 22 Jul. 2020.

MICROSOFT. *TrackBar Classe*. 2017. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/api/microsoft.office.tools.excel.controls.trackbar?view=vsto-2017>. Acesso em: 1 Maio. 2020.

OPENCV. *Hough circle transform*. *OpenCV* 2.4.13.7 *documentation*, [entre 2011 – 2019] a. Disponível em: <https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_circle/hough_circle.html>. Acesso em: 10 Fev. 2020.

OPENCV. Mat – *The basic image container*. *OpenCV* 2.4.13.7 *documentation*, [entre 2011 – 2019] b. Disponível em: <https://docs.opencv.org/2.4/doc/tutorials/core/mat_the_basic_image_container/mat_the_basic_image_container.html>. Acesso em: 02 Fev. 2020.

OPENCV. *Changing Colorspaces*. *Open Source Computer Vision*. [201-]. Disponível em: < https://docs.opencv.org/3.4/df/d9d/tutorial_py_colorspaces.html>. Acesso em: 15 Maio 2020.

PANTUZA, Gustavo. O que são e como funcionam os *sockets*. 2017. Disponível em: <https://blog.pantuza.com/artigos/o-que-sao-e-como-funcionam-os-sockets>. Acesso em: 20 Nov. 2019.

PARROT. AR.*DRONE developer guide*. 2012. Disponível em: <https://jpchanson.github.io/ARdrone/ParrotDevGuide.pdf>. Acesso em: 17 Nov. 2019.

PARROT. AR.*DRONE* 2.0 *power edition fly even longer.* 2014. Disponível em: <https://www.parrot.com/global/drones/parrot-ardrone-20-power-edition>. Acesso em: 17 Nov. 2019.

PECHARROMÁN, José María Peral; VEIGA, Ricardo. **Estudo sobre a indústria brasileira e europeia de veículos aéreos não tripulados**. Diálogos Setoriais União Europeia – Brasil, 2016. Disponível em: <http://www.mdic.gov.br/images/publicacao_DRONES-20161130-20012017-web.pdf> Acesso em: 23 Abr. 2019.

PHILIPE, Gabriel. Como funciona o *GPS*?. Oficina da net, 2016. Disponível em: <https://www.oficinadanet.com.br/post/12406-como-funciona-o-gps>. Acesso em: 28 Out. 2019.

PUKU0X. CV *DRONE*. Github 2016. Disponível em: <https://github.com/puku0x/cvdrone>. Acesso em: Jul. 2019.

RODRIGUES, Lino. Industria de *drones* movimenta R$ 300 milhões no Brasil. **Estado de Minas Economia**, São Paulo, 17 Maio 2018. Disponível em: <https://www.em.com.br/app/noticia/economia/2018/05/17/internas_economia,959372/industria-de-drones-movimenta-r-300-milhoes-no-brasil.shtml>. Acesso em: 29 Out. 2019.

SALDANHA, Ranieri da Silva Monteiro. **Controle de atitude e altura de um quadricóptero utilizando técinas de controle PID**. 2019. 58 f. Trabalho de conclusão de curso (Graduação em Engenharia Elétrica) – Universidade Federal da Paraíba, João Pessoa – PB, 2019. Disponível em: <http://www.cear.ufpb.br/arquivos/cgee/TCC/TCC_-__Ranieri_da_Silva_Monteiro_Saldanha_-_Vers%C3%A3o_Final.pdf>. Acesso em: 15 Dez. 2019.

SANTANA, Lucas Vago. **Sistemas de navegação e controle para veículos aéreos não tripulados e suas aplicações**. 2016. 153 f. Tese (Doutorado em Engenharia Elétrica) – Universidade Federal do Espírito Santo, Vitória – ES, 2016. Disponível em: <http://portais4.ufes.br/posgrad/teses/tese_10333_Tese%20de%20Doutorado%20-%20Lucas%20Vago%20Santana.pdf>. Acesso em: 21 Abr. 2019.

SARAIVA, Henrique Pinheiro. **Navegação autônoma, reconhecimento e desvio de obstáculos com *drones* multirotores**. 2018. 52 f. Trabalho de conclusão de curso (Graduação em Engenharia Elétrica) – Pontifica Universidade Católica do Rio de Janeiro, Rio de Janeiro – RJ, 2018. Disponível em: <https://www.maxwell.vrac.puc-rio.br/34465/34465.PDF>. Acesso em: 26 Fev. 2019.

SILVA, Lívio Tonini Gouveia e. **Modelagem, simulação e controle de um VANT quadrirrotor**, 2015. 56 f. Trabalho de conclusão de curso (Graduação em Engenharia de Controle e Automação) – Universidade Federal da Bahia, Salvador – BA, Brasil, 2015. Disponível em: <http://www.cceca.eng.ufba.br/cceca/wp-content/uploads/2017/02/TCC_LivioTonini_revisao6.pdf>. Acesso em: 15 Nov. 2019.

TARDE NACIONAL. *Drones* são utilizados no combate à Covid-19. 2020. EBC. Disponível em: < https://radios.ebc.com.br/tarde-nacional/2020/05/drones-estao-sendo-utilizadas-para-combater-o-covid-19>. Acesso em: 6 Jun. 2020.

APÊNDICE – Código Fonte

O código fonte está disponível em: <https://github.com/tabataoliveira/TCC_Controle_semiautonomo_para_drone/blob/master/Ap%C3%AAndice>.