Universidade Federal de Juiz de Fora

Faculdade de Engenharia

Engenharia Elétrica - Habilitação em Robótica e Automação Industrial

Vinicius Ferreira Vidal

Controle de pouso de veículo quadrotor auxiliado por Visão Computacional

Juiz de Fora

2016

Vinicius Ferreira Vidal

Controle de pouso de veículo quadrotor auxiliado por Visão Computacional

Trabalho apresentado à disciplina de Trabalho de Conclusão de Curso do curso de graduação de Engenharia Elétrica - Habilitação em Robótica e Automação Industrial da Universidade Federal de Juiz de Fora

Orientador: Leonardo de Mello Honório

Juiz de Fora

2016

Ficha catalográfica elaborada através do programa de geração
automática da Biblioteca Universitária da UFJF,
com os dados fornecidos pelo(a) autor(a)

Vidal, Vinicius Ferreira.
Controle de pouso de veículo quadrotor auxiliado por visão computacional / Vinicius Ferreira Vidal. -- 2016.
82 p.

Orientador: Leonardo de Mello Honório
Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Juiz de Fora, Faculdade de Engenharia, 2016.

1. VAANT. 2. Visão Computacional. 3. Posicionamento auxiliado por visão. 4. Pouso autônomo. I. Honório, Leonardo de Mello, orient. II. Título.

Vinicius Ferreira Vidal

Controle de pouso de veículo quadrotor auxiliado por Visão Computacional

Trabalho apresentado à disciplina de Trabalho de Conclusão de Curso do curso de graduação de Engenharia Elétrica - Habilitação em Robótica e Automação Industrial da Universidade Federal de Juiz de Fora

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Leonardo de Mello Honório - Orientador
Universidade Federal de Juiz de Fora

Professor M. Eng. Murillo Ferreira dos Santos
Centro Federal de Educação Tecnológica de Minas Gerais

Professor Dr. Leonardo Rocha Olivi
Universidade Federal de Juiz de Fora

UNIVERSIDADE FEDERAL DE JUIZ DE FORA
FACULDADE DE ENGENHARIA
COORDENAÇÃO DO CURSO DE ENGENHARIA ELÉTRICA – ROBÓTICA E AUTOMAÇÃO INDUSTRIAL

ATA DE APRESENTAÇÃO DE TRABALHO FINAL DE CURSO

DATA DA DEFESA: 08/12/2016
CANDIDATO: VINICIUS FERREIRA VIDAL
ORIENTADOR: PROF. LEONARDO DE MELLO HONÓRIO

BANCA EXAMINADORA:

PROF. LEONARDO DE MELLO HONÓRIO – UNIVERSIDADE FEDERAL DE JUIZ DE FORA
PROF. LEONARDO ROCHA OLIVI – UNIVERSIDADE FEDERAL DE JUIZ DE FORA
PROF. MURILLO FERREIRA DOS SANTOS – CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

TÍTULO DO TRABALHO:

CONTROLE DE POUSO DE VEÍCULO QUADROTOR AUXILIADO POR VISÃO COMPUTACIONAL

LOCAL: FACULDADE DE ENGENHARIA, ANFITEATRO DA PPEE

Em sessão pública, após exposição de 50 minutos, o candidato foi arguido oralmente pelos membros da banca, o candidato foi:

(x) APROVADO
( ) REPROVADO

Na forma regulamentar foi lavrada a presente Ata que é abaixo assinada pelos membros da banca na ordem determinada e pelo candidato:

PRESIDENTE DA BANCA: _____

_____

_____

CANDIDATO: Vinicius F. Vidal

Juiz de Fora, 08 de dezembro de 2016.

Vistos:

Leonardo Rocha Olivi
Coordenador da Engenharia Elétrica

AGRADECIMENTOS

Ao meu pai Luiz Fernando e minha mãe Maria da Consolação, pela educação, dedicação e base para poder desenvolver meus estudos da melhor forma possível.

A minha irmã Larissa por ajudar em todos os meus trabalhos.

A todos os meus familiares pelo incentivo e dedicação durante esse curso de Engenharia.

Ao meu orientador, professor Leonardo Honório, pela pela atenção e tempo dedicados no trabalho.

A todos os amigos do GRIn, pelos bons momentos e ajuda durante todo o processo.

Agradecimento especial aos companheiros de equipe Mathaus Ferreira e Murillo Ferreira, pelo apoio essencial para a conclusão desse trabalho.

Aos bons amigos, pelo apoio e incentivo no curso e on bons momentos durante esses anos.

A TBE - Transmissoras Brasileiras de Energia, pelo apoio no projeto.

“Ter sucesso é ir de fracasso em fracasso, mas sem perder o entusiasmo”
(Winston Churchill)

RESUMO

A pesquisa e o uso de VAANTs (Veículos Aéreos Autônomos Não Tripulados) vêm ganhando maior atenção e novas ideias nas últimas décadas. Isto se deu devido ao avanço tecnológico e o seu comportamento autônomo, o que demanda técnicas robustas para garantir a segurança da aeronave e de terceiros. O presente trabalho propõe o uso de técnicas de visão computacional a fim de auxiliar a localização segura de um VAANT do tipo quadrotor no momento do pouso ao final de missão, adicionando precisão e segurança, contornando erros inerentes a sensores de posicionamento, como GPS. Para tal foi desenvolvido um *landmark* a ser detectado pela câmera através de técnicas de obtenção de contorno e reconhecimento de formas. Foram utilizadas operações morfológicas, *thresholding* e outras técnicas matemáticas a fim de se obter processamento rápido, simples e robusto. A câmera escolhida foi o modelo esportivo *GoPro Hero 3*, com algoritmo desenvolvido para correção da distorção. Dessa forma, programa-se o veículo para realizar uma determinada missão, onde no momento do pouso passava a operar o sistema de localização por visão computacional aqui construído. O quadrotor foi construído no laboratório GRIn (Grupo de Robótica Inteligente) da UFJF (Universidade Federal de Juiz de Fora) e utiliza código aberto (C++) de controle com protocolo *Mavlink* para comunicação com o algoritmo de monitoramento de missão e pouso. O algoritmo de visão computacional desenvolvido na linguagem *Python* capta os dados e distâncias obtidos na leitura, realiza segmentação das características da imagem e envia comandos de voo para o quadrotor através do protocolo mencionado. O sistema utiliza um *laptop* conectado serialmente com o quadrotor, com sinal roteado por portas TCP internas para a GCS (rodando no próprio *laptop*) e o algoritmo de controle de alto nível do VAANT (planejamento de missão e localização por visão) e via Wi-fi com a câmera por protocolo TCP/IP, sendo então todo o processamento de imagem executado no laptop. Como conclusão, tem-se que os métodos foram bastante eficazes e de rápido processamento, garantindo segurança ao pouso da aeronave, além de as técnicas de posicionamento por visão poderem ser estendidas para outras aplicações em VAANTs, como inspeção de linhas de transmissão e outros equipamentos envolvidos em ângulos e posição pré-definidas. Trabalhos futuros buscariam embarcar a técnica em uma *companion board*, a fim de automatizar o processo e reduzir possíveis interferências de comunicação.

Palavras-chave: VAANT. Visão Computacional. Posicionamento auxiliado por visão. Pouso Autônomo.

ABSTRACT

The research and use of UAVs (Unmanned Aerial Vehicles) has grown in attention and new ideas in the last couple of decades, mainly due to technology improvement of its construction parts, sensors and controllers. When it comes to its autonomous behavior, robust techniques make themselves necessary to assure security of the aircraft and people around. This work presents the use of computer vision techniques to improve the positioning of a quadrotor UAV in the landing moment, adding security and precision, overcoming intrinsic errors of other positioning sensors, such as GPS. A landmark was built to be detected by the camera with contour detection, image thresholding and other mathematical techniques in order to obtain fast, simple and robust processing and results. The camera used was the GoPro Hero 3 model, with a developed algorithm to remove the image distortion. With this set up, the vehicle can be programmed to develop any mission, with this work?s control algorithm taking part with the vision aided positioning in the landing moment. The quadrotor was conceived and built at the GRIn (Grupo de Robótica Inteligente) laboratory at UFJF (Juiz de Fora Federal University), and uses open source code (C++) for its control with Mavlink communication protocol to exchange messages with the vision based control algorithm for mission and landing. This algorithm was developed in Python language which develops the image characteristics segmentation, extracting distances and altitude from the data. The control commands are sent to the quadrotor using the mentioned protocol. The system is structured with a remote laptop connected with a serial port to the quadrotor, and a router transmits messages back and forth among the GCS (Ground Control Station, on the laptop) and the high level control algorithm (mission planning and vision based control) through TCP ports to the quadrotor itself. The camera is connected via Wi-Fi built-in network to the laptop, so all the image processing is done inside the last one. The results obtained in conclusion show the efficacy and fast processing of the vision methods and ensure safety to the landing moment. Besides, the positioning techniques can be extended to other UAV applications, such as transmission lines and utility equipment inspection and monitoring. Future works look forward to embed the processing in a companion board attached to the UAV, so the process would become automatic and the risk of communication interference and lost heavily reduced.

Key-words: UAV, Computer Vision, Vision Aided Positioning, Autonomous Landing.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

VAANT	Veículo aéreo autônomo não tripulado
UFJF	Universidade Federal de Juiz de Fora
GCS	*Ground control station*
GPS	*Global Positioning System*
SITL	*Software in the Loop*
RGB-D	*Red-Green-Blue-Depth*
TCP	*Transmission Control Protocol*
IDE	*Integrated Development Environment*
HUD	*Heads-Up Display*
MAVLink	*Micro Air Vehicle Communication Protocol*
	LRF *Laser Rangefinder*

SUMÁRIO

1 Introdução

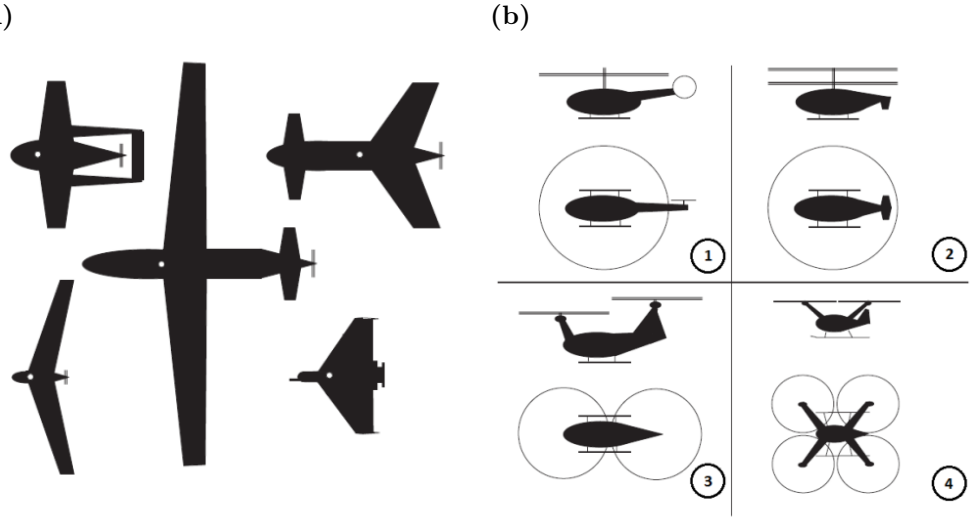
O uso de VAANTs (Veículos Autônomos Aéreos Não Tripulados), também conhecidos como *drones*, em especial do tipo *quadrotor*, vem tendo cada vez mais aceitação nos dias atuais, seja para propósitos de pesquisa, comerciais, militares ou de lazer. Essa expansão vem sendo facilitada pelo crescente número de fabricantes e melhoria da qualidade dos componentes e sensores envolvidos [38].

1.1 Construção

VAANTS podem ter sua forma construtiva em diversas modalidades, como em asa fixa (exemplo: avião), ilustradas na Figura 1 a esquerda, ou asa móvel (exemplo: helicóptero), como em 1 a direita. Também são encontradas outras formas de construção como balões e dirigíveis [12]. Modelos de asa fixa são utilizados normalmente para o varrimento de áreas extensas, por apresentarem velocidades mais elevadas e melhor autonomia [16], porém necessitam de velocidade horizontal mínima de sustentação para se manterem no ar.

Um ponto positivo na utilização de balões e dirigíveis são a excelente autonomia em voo, porém sua velocidade reduzida e alta susceptibilidade à interferência de ventos o restringe ao emprego em áreas extensas de cobertura [15].

Figura 1 – Formas construtivas de VAANTs. a) Exemplos de aeronaves de modelo asa fixa. b) Exemplos de aeronaves de modelo asa móvel.

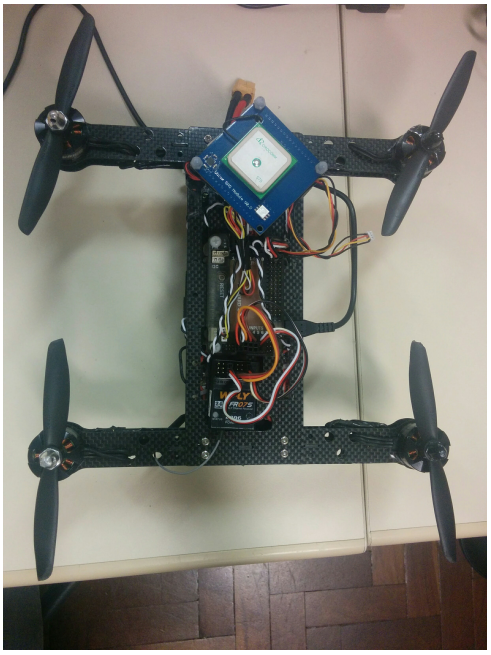


Fonte: [3]

Os veículos de asa móvel do tipo helicóptero ou multicóptero possuem excelente manobrabilidade e precisão, além da capacidade de pairar durante voo sem necessidade de

velocidade horizontal. O emprego deste modelo de aeronave, em especial do tipo quadrotor, tem tido bastante aceitação em aplicações como monitoramento e inspeção [16]. Um *drone* do tipo *quadrotor* está ilustrado na Figura 2.

Figura 2 – Quadrotor desenvolvido na Universidade Federal de Juiz de Fora.



1.2 Contexto histórico

Historicamente, o primeiro registro de VAANT a levantar voo com sucesso data de 1918, a aeronave *Sperry's Aerial Torpedo* (Figura 3). É considerado precursor dos misseis-guiados modernos, e foi utilizado em testes durante a I Guerra Mundial [33].

Figura 3 – Aeronave *Sperry's Aerial Torpedo*, construída no ano de 1916 por Lawrence e Elmer Sperry.

Fonte: [33]

O desenvolvimento de VAANTs esteve relacionado inicialmente a laboratórios de centros militares. Houve crescente investimento a partir da década de 40, no auge da II Guerra Mundial, com a produção do modelo V-1 de asa fixa pela Alemanha para uso em operações bélicas [33]. Nas décadas seguintes os avanços tecnológicos permitiram o emprego de VAANTs em missões como reconhecimento e espionagem [1].

A partir da década de 1970 surgiu o desenvolvimento de *drones* mais baratos, leves e eficientes, o que permitiu a utilização em aplicações civis. A título de curiosidade, a primeira aeronave a cruzar o oceano Atlântico na época foi o modelo *Aerosonde*, utilizada para coleta de dados meteorológicos e vigilância pelo exército americano [33].

Figura 4 – Modelo *Aerosonde*.

Fonte: [33]

No Brasil, o projeto Acauã foi o primeiro em termos de veículos aéreos autônomos, realizado pela parceria entre o CTA (Comando-Geral da Tecnologia Aeroespacial), o IPqM (Instituto de Pesquisas da Marinha) e o CTEX(Centro Tecnológico do Exército), em 1984. Seu objetivo era de coletar dados climatológicos e inspeção de gasodutos e oleodutos. Sua envergadura era 5,1 por 4,8 metros de comprimento, com peso total de 120 kg [8].

Figura 5 – Aeronave do projeto Acauã.

Nas décadas de 90 e início do século XXI destacam-se projetos de VAANTs em universidades brasileiras, como a Universidade Federal de Minas Gerais - UFMG.

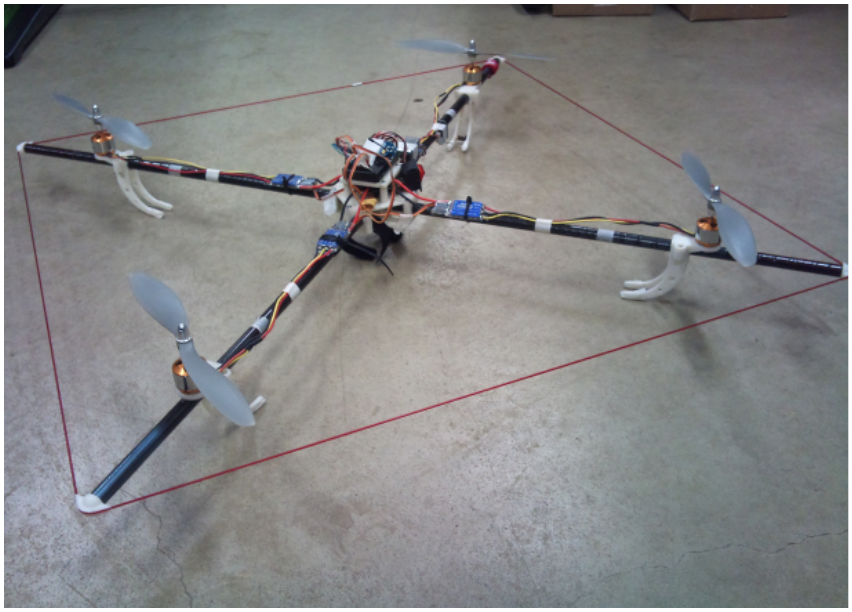
No ano de 2000 foi concluído o projeto "Controle e Implementação de um Dirigível Autônomo"(CIDA), com uma aeronave no modelo dirigível para o estudo da cooperação entre veículos autônomos terrestres e aéreos [9].

Figura 6 – Dirigível do projeto CIDA, desenvolvido na UFMG.

Fonte: [9]

Mais recentemente o emprego e estudo em multicópteros vem sendo notado na comunidade cientifica, envolvendo construção, sensoriamento e controle do veículo. A dissertação de mestrado de [15] compara diversas técnicas de controle com foco na estabilidade da aeronave do tipo quadrotor (Figura 7), através de modelo matemático aplicado a bancada de teste.

Figura 7 – Quadrotor desenvolvido na UFJF como bancada de testes para modelos de controladores.



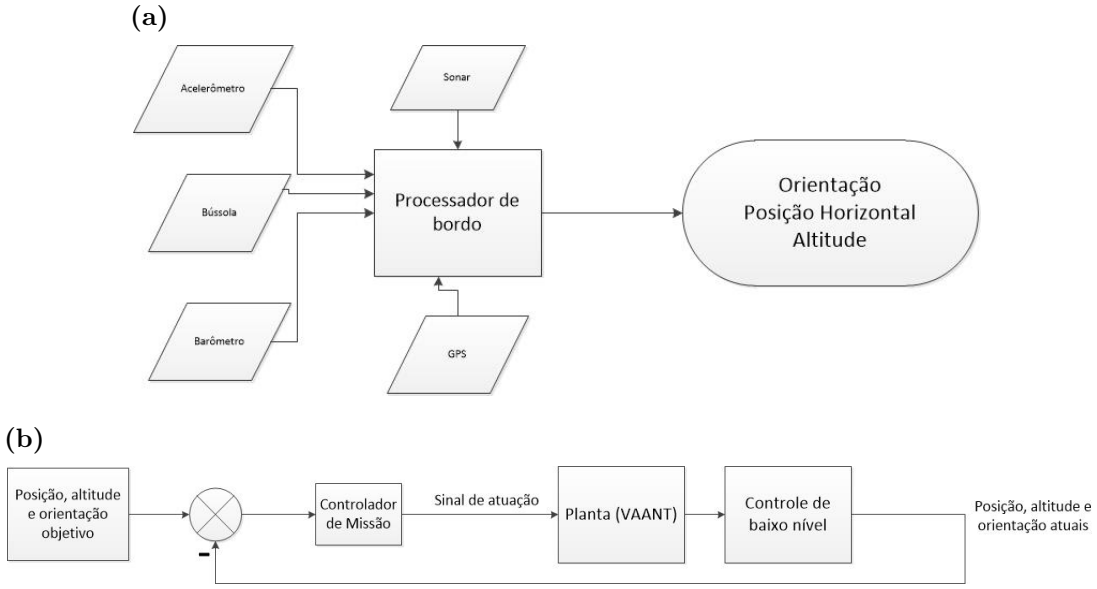
Fonte: [15]

1.3 Sensores, controle e navegação

Para o voo autônomo dessas aeronaves geralmente se utiliza de uma placa controladora com IMU (*Inertial Measurement Unit*) integrada, bússola, barômetro, sonares e sensor GPS [15], [38].

O controle de baixo nível busca obter dados para cálculo de orientação (ou atitude), posição horizontal e altitude da aeronave. Para o controle de alto nível são englobados esses dados no planejamento e supervisão de trajetórias e missões [6]. O diagrama de blocos da Figura 8 ilustra como se relaciona o uso dos sensores com o controle de baixo e alto nível do veículo. Tanto a IMU quanto a bússola indicam a atitude do *drone*; barômetro e sonar podem ser usados para leitura de altitude, e GPS para obter dados de posicionamento e também altitude.

Figura 8 – a) Diagrama com leitura de sensores e processamento de baixo nível. b) Esquema simplificado de controle de alto nível (missão).



O sensor GPS por de fato pode apresentar erros da ordem de metros, o que deprecia a qualidade e confiabilidade da missão, principalmente no momento do pouso. Para essa situação, uma solução adotada é o uso de uma câmera como sensor exteroceptivo, o que além de atribuir precisão possui as vantagens de ser um sensor passivo, sem contato e menos sensível ao ruído, que não depende do sinal de GPS, o que permite o voo em ambientes onde esse sinal não possui boa qualidade, como ambientes fechados [23].

A câmera também é usada como sensor anti colisão em diversas aplicações tanto acadêmicas quanto comerciais (*obstacle avoidance*). Os trabalhos de [18], [43] e [37] propõe estratégias que se valem do uso de câmeras RBG-D (*red-blue-green-deepth*) para obtenção dos dados do ambiente, em fusão com dados provenientes da IMU, filtrados para calcular o controle da aeronave a partir do conceito de estabilidade de Lyapunov. Resultados incluem paradas de ação na presença de perigo de colisão, principalmente em ambientes fechados, escapadas laterais e frontais, ou busca de pontos seguros quando vislumbrado um problema. Vale ressaltar o uso de sonares como outra fonte de dados de distância a serem fundidos em um filtro no *software* de controle para melhor informação do ambiente.

O modelo comercial *Mavic* da fabricante *DJI* disponibiliza um modo ao usuário onde utiliza sua câmera frontal para detectar objetos que se aproximam na imagem e assim parar o movimento do aeronave se algo acusar-se muito próximo, independente da entrada de rádio ou outro comando do usuário.

1.4 Aplicações de VAANTs

1.4.1 Fotografia e filmagem

Vários modelos comerciais voltados para o lazer e até mesmo para filmagem profissional possuem uma câmera acoplada. A resolução e o seu objetivo vem a variar com a faixa de aplicação à qual são destinados. Como exemplo, a Figura 9 traz um *drone* modelo *Inspire* da fabricante *DJI*, com câmera profissional acoplada de filmagem em resolução 4K e fotos de 12 *Mega pixels*.

Figura 9 – *Drone* comercial utilizado para fotografia e filmagem.

1.4.2 Agricultura

O monitoramento de plantações também pode ser realizado por VAANTs, e é uma aplicação vasta entre os fabricantes, desde a construção da aeronave até o *software* de análise dos dados coletados. A facilidade obtida de a qualquer momento acionar a aeronave para executar uma missão planejada de forma simples, colhendo resultados satisfatórios e a um preço razoável quando comparado ao investimentos em outros equipamentos utilizados para o mesmo propósito no plantio, são as maiores vantagens do emprego de VAANTs nessa área [45].

Para o estudo da lavoura é empregada uma câmera de raios infravermelhos. Com ela pode-se basicamente observar a temperatura das folhas, e a partir daí prever questões como a desidratação e presença de doenças nas plantas, após comparação com um banco de dados sobre imagens de lavouras saudáveis [34]. A Figura 10 ilustra o *drone* da fabricante *DJI* modelo *Phantom 3*, modificado com uma câmera de raios infravermelhos permanente utilizado pela empresa *Agribotix* para monitoramento de plantações, e o modelo *Hornet*, uma aeronave asa fixa também desenvolvida para esse propósito.

Figura 10 – Modelos comerciais de *drones* empregados em lavouras. a) Modelo *Phantom 3* customizado pela empresa *Agribotix*. b) Modelo comercial *Hornet* da empresa *Agribotix*.

(a) **(b)**

Fonte: [45]

1.4.3 Entregas de encomendas

Drones também são estudados para serem utilizados pelo serviço de entregas em algumas regiões como nos Estados Unidos, Austrália e Suíça. No primeiro, o serviço *Prime Air* da companhia *Amazon* promete atender a encomendas de até 5 libras em meia hora, com veículos cobertos de redundâncias *"sense and avoid"* para garantir a segurança durante o tráfego em rotas especiais. Um dos protótipos está ilustrado na Figura 11.

Figura 11 – Modelo conceito de *drone* fabricado pela empresa *Amazon* para realizar entregas.

Na Suíça estão sendo testados *drones* para entregas de encomendas pesando até um quilograma por distâncias que se estendem a dez quilômetros, segundo notícias de Agosto deste ano. São utilizados os veículos produzidos pela fabricante *Matternet*, e o design e planejamento devem se manter em teste pelos próximos cinco anos.

1.4.4 Mapeamento

Drones voltados para mapeamento fotogramétrico de superfície geralmente possuem design que primam pelo maior tempo de voo possível para aumentar a área mapeada em um único voo, por isso é comum ser construído em formato asa-fixa. São utilizadas técnicas de processamento de imagem em 3D para produção de modelos de elevação digital (*Digital Elevation Models - DEM*) [36]. Câmeras especiais empregadas como câmera de infravermelho, RGB profissional ou multiespectral, assim como sensor GPS e de distância, como sonares e *lidars*, coletam dados para reconstrução do ambiente.

Um exemplo de *drone* comercial com esse objetivo é o *eBee* da fabricante *SenseFly*. O mesmo possui câmera customizada pelo fabricante, e o processamento de dados obtidos pode ser realizado em tempo real.

Outra solução possível foi desenvolvida pela fabricante *3DR* com o modelo *Solo*, voltada para mapeamento também em 2D e com dados processados remotamente por serviços de terceiros. Aplicada em construções, mapeamento de terrenos, linhas de transmissão, subestações, segurança pública, entre outros. Algoritmos possibilitam inserção de projeções 3D sobre o mapeamento, simples reconstrução do ambiente ou monitoramento.

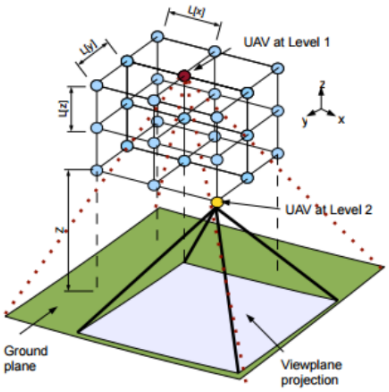
1.4.5 Salvamento

O termo SAR (*Search and Rescue*) vem sendo empregado na literatura para tratar de *drones* de salvamento. *Drones* podem alcançar com mais facilidade ambientes hostis ou de difícil acesso ao ser humano, contribuindo com o resgate em tempo reduzido. Aplicações encontradas seriam contra afogamento no mar ou lago, para ação mais rápida pelo salva-vidas ao entregar boias ou coletes, ou para encontrar pessoas desaparecidas ou vasculhamento em desastres.

Também são encontradas pesquisas nessa área na literatura. Em [41] são utilizados *drones* de prateleira e estudados algoritmos para otimizar o tempo gasto para identificação de vítimas em busca autônoma sobre uma região que sofrera algum desastre (Figura 1.4.5).

Figura 12 – Estudo de emprego de *drones* para busca em terrenos onde ocorreram desastres de forma otimizada. a) Exemplo de *drone* utilizado na pesquisa. b) Ilustração dos pontos no espaço aéreo a serem vasculhados e otimizados para navegação.

(a) **(b)**



Fonte: [41]

1.5 Objetivos e estruturação do trabalho

A fim de reforçar a segurança e autonomia de uma aeronave *drone* do tipo quadrotor, dados obtidos com a câmera vão ser utilizados para localização no ambiente, com enfoque principal no momento do pouso. Para isso o quadrotor será construído e a ele será acoplado um sistema de visão. Algoritmos de controle serão integrados através de uma rede de comunicação para garantir o processamento de dados em tempo hábil, assim como capacidade de monitoramento da missão desenvolvida pela aeronave, para que tudo corra em segurança e a precisão seja averiguada.

Os capítulos seguintes descrevem o trabalho da seguinte forma: o capítulo 2 traz tecnologias e trabalhos atuais envolvendo o pouso autônomo de *drones*; o capítulo 3 descreve técnicas matemáticas para localização da aeronave no ambiente; o capítulo 4 cita as linguagens e bibliotecas empregadas no desenvolvimento dos algoritmos no trabalho; o capítulo 5 desenvolve toda técnica de análise por visão computacional utilizada no algoritmo de controle; o capítulo 6 trata da rede para comunicação entre as diversas partes do projeto; o capítulo 7 mostra os resultados da construção do quadrotor; o capítulo 8 mostra como foi simulado o processo antes da execução do pouso de forma prática; o capítulo 9 contém toda a parte prática, com resultados de pouso e sistema de visão em campo; por fim, o capítulo 10 traz todas as conclusões do trabalho realizado.

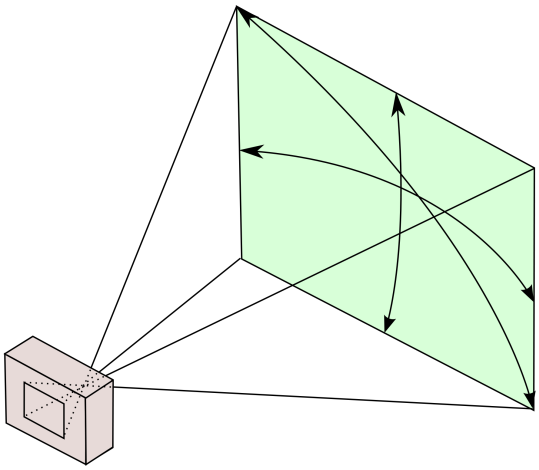
2 Estado da arte em *Autonomous Landing*

2.1 Câmera

Vários equipamentos são utilizados na literatura para localização do quadrotor com foco no pouso, como sensor de GPS, sonares e LRF, porém o emprego da câmera garante informações variadas do ambiente a partir de um sensor passivo e de custo relativamente baixo [26]. Câmeras também são geralmente leves, o que é favorável para não exigir muito esforço do quadrotor devido à carga.

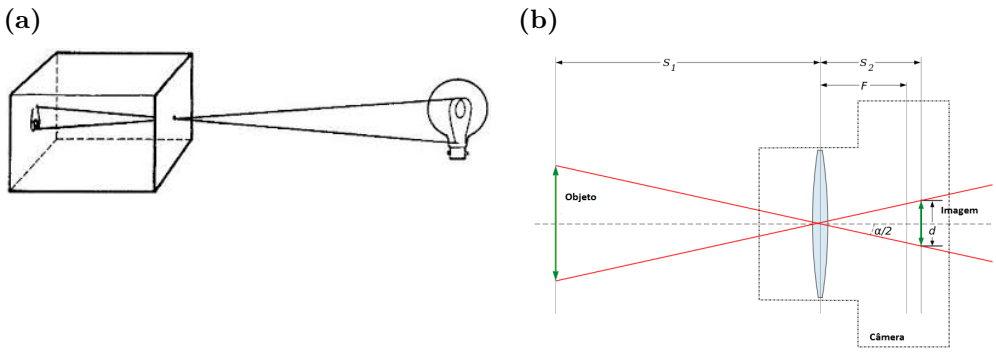
Detalhes importantes que caracterizam câmeras e lentes são a sua abertura angular, campo de visão (*field of view - fov*), definição e distância focal. Considerando o mundo visto pela câmera como uma esfera, a abertura angular diz respeito ao ângulo dessa esfera captado pela lente, como ilustrado na Figura 13. A partir dessa abertura o sensor óptico capta a imagem e tudo que nela pode estar contido é considerado o campo de visão.

Figura 13 – Ilustração do conceito de ângulo de visão e o campo de visão.



A definição também está ligada ao sensor óptico, e representa o quão detalhadamente a imagem captada será no processo de digitalização, e é expressa em *pixels*. A distância focal pode ser melhor entendida a partir do modelo *pinhole* de câmera, ilustrado na Figura 2.1. Neste simples modelo, as lentes são consideradas pontuais, e os raios atravessam esse ponto, formando a imagem no plano do sensor óptico, daí a distância focal estaria entre esse plano à lente.

Figura 14 – Conceitos de estudo de câmeras fotográficas. a) Modelo ilustrativo do conceito de *pinhole camera*. b) Ilustração da distância focal *focal length* F física em uma câmera.



A precisão na localização através da imagem é considerada satisfatória para aplicações como o pouso, alcançando a ordem de centímetros [10]. Tipos de câmeras que podem ser utilizados são a câmera com imagem estéreo e câmeras monoculares, porém a primeira tem performance reduzida quando a plataforma e o quadrotor se portam a uma distância muito maior que a distância de referência de medidas, portanto a segunda é preferida como sensor de visão para pouso. Há também na literatura o emprego de câmera omnidirecionais [29], à qual é acoplada uma lente "olho de peixe"de forma a ter maior área de visão, porém dessa forma ocorre a distorção da imagem e a distância real deve ser aproximada por um polinômio de quarta ordem. O caso da distorção da imagem também ocorre em modelos de câmeras de ação, como a linha *GoPro*, que possui uma lente com abertura angular de 120 graus.

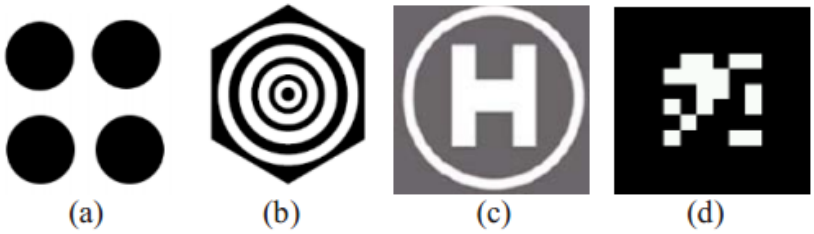
Os dados obtidos da câmera podem ser fundidos através de filtros no controlador de voo com dados provenientes de sensores inerciais, como acelerômetros, magnetômetros, barômetros ou sonares para a estimação mais exata de dados como altitude e velocidade do veículo, além da pose atual.

2.2 Marcadores Visuais

Para a determinação da pose do quadrotor em relação ao local de pouso são desenvolvidos marcadores visuais, ou *landmarks*, que são postos no local de pouso e possuem um padrão reconhecível através de algoritmo. Os dois tipos de marcadores mais comuns encontrados na literatura são pontuais e baseados em curvas, nos quais os padrões podem ser reconhecidos através de formas geométricas, como círculos e retângulos (*blobs*) e arestas de canto [26]. Podem ser coloridos para serem detectados por filtragem do algoritmo através de uma cor que destoe do ambiente ao redor, porém o marcador em preto e branco fornece maior contraste para a imagem em escala de cinza, o que promove processamento de dados mais simples e resultados satisfatórios na identificação dos padrões.

O trabalho de [26] apresenta exemplos de marcadores comumente utilizados, ilustrados na Figura 15. Marcadores pontuais, como os circulares ilustrados na letra (b), retornam a posição com considerável fidelidade, enquanto o em formato de H e baseado em retângulos fornecem de forma satisfatória a pose da aeronave a cada instante de amostragem, de forma a auxiliar quando a orientação do pouso deve ser levada em consideração.

Figura 15 – Exemplos de marcadores: (a) Pontual; (b) Circular; (c) Formato de H; (d) Fiducial baseado em quadrados.



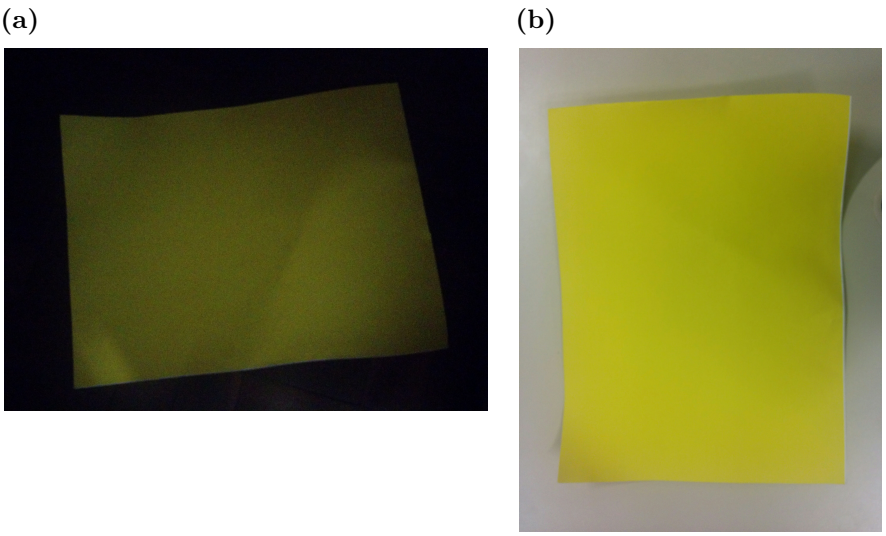
Fonte: [26]

Em [13] um marcador que combinava fatores pontuais com curvas foi desenvolvido para guiar o pouso de um quadrotor sobre uma base em uma determinada orientação, a fim de carregar o veículo enquanto parado. A precisão da posição final deveria estar em menos de 5 centímetros em relação ao centro da base, e 10 graus de rotação, o que provou-se satisfeito nos resultados. A base possuía *design* redundante para garantir o funcionamento em altitudes elevadas e menores.

Outra técnica pode ser vista em [10], na qual o marcador possui um padrão composto por arestas de forma a determinar a pose do quadrotor. A plataforma trabalha em conjunto com o veículo, acendendo LEDs sobre as arestas no momento do pouso, e a falha no reconhecimento do padrão remete um sinal ao quadrotor para que paire e aguarde a retomada do reconhecimento. As luzes se apagam quando o pouso é concluído, e novamente precisão de centímetros é obtida nos resultados.

Um grande problema enfrentado com os marcadores e a utilização da câmera como sensor no geral tem a ver com a luminosidade do ambiente de estudo. Pode ser observado na Figura 16 como a percepção da cor pela câmera é alterada pela luminosidade do ambiente, o que prejudicaria o filtro da imagem em busca do objeto colorido. Uma saída adotada seria o uso da técnica de *adaptative threshold* como realizado em [25], onde os parâmetros de filtro são adaptados para a luminosidade do local automaticamente.

Figura 16 – Visualização com *webcam* de um tom de amarelo em condições diversas de iluminação. a) Amarelo capturado em pouca luz. b) Amarelo capturado em maior iluminação.



Outra interferência da luminosidade tem a ver com a reflexão da luz pelo marcador, o que pode desfigurar a característica do mesmo. O material de que é feito o mesmo deve ser observado para minimizar este problema, e algoritmos de reconhecimentos mais robustos devem ser desenvolvidos.

2.3 Transmissão de dados

Para controle da aeronave a distância com monitoramento em tempo real, com possibilidade de troca de comandos e mensagens em geral, é empregada uma GCS (*Ground Control Station*, ou estação de controle terrestre) em um microcomputador. Existem GCSs de código aberto na internet bastante disseminadas, como *Mission Planner* e *Qground Control*, com as mais gerais aplicações relacionadas a veículos aéreos, mas também há na literatura casos de criação de uma GCS própria, como visto em [28], desenvolvida em C++ utilizando *OpenGL* para aplicações específicas.

A troca de mensagens com o quadrotor pode ser feita via sinal de rádio ou Wifi em frequências como 2.4 GHz, assim como empregado em [31]. Para trafego de dados de imagem e processamento o mesmo trabalho utiliza um transmissor para as imagens colhidas pela câmera com a frequência de 1.2 GHz e um receptor conectado ao laptop, onde ocorre o processamento e os cálculos para controle da aeronave. Essa técnica se faz necessária pois não há memória e capacidade de processamento na placa controladora de voo para tratar as imagens obtidas pela câmera.

Outra forma de processamento das imagens obtidas seria através do uso de um processador embarcado, ou *companion board*. Há varias placas que realizam essa função, como a *Odroid, Beaglebone Black* ou a *Raspberry Pi*, empregada em [2] para controle

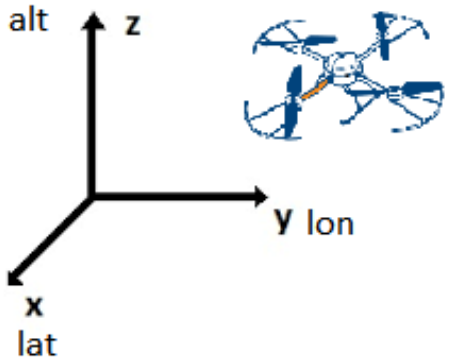
do pouso de um quadrotor assistido por visão computacional. No caso dessa forma de processamento a troca de dados ocorre via cabo com ligação serial ou por protocolo I2C, a depender do modelo, com vasta documentação na internet para estabelecer comunicação.

As *companion boards* promovem uma comunicação com menos riscos de ruídos e perdas de comunicação por ditância ou mau tempo, porém geralmente adicionam mais peso à aeronave.

3 Localização da aeronave no ambiente

No intuito de monitorar a localização e dinâmica do *quadrotor* no espaço é utilizado ferramental matemático e teoria de eixos coordenados, ou *frames* [14]. O mesmo possui uma origem e três direções espaciais principais, as quais atribuem posição e orientação à aeronave a partir da pose das mesmas em relação à origem.

Figura 17 – Eixos coordenados ilustrando o sistema global de georreferenciamento do quadrotor.



A base para a localização do quadrotor no planeta se dá pelo *frame* global, que leva em consideração o sistema de coordenadas globais, o que compreende-se pela latitude no eixo x, longitude no eixo y e altitude em relação ao nível do mar no eixo z da Figura 17. Por meio desse *frame* pode-se localizar e delimitar qualquer parte do globo através de coordenadas do sensor de GPS (*Global Positioning System*) presente na aeronave.

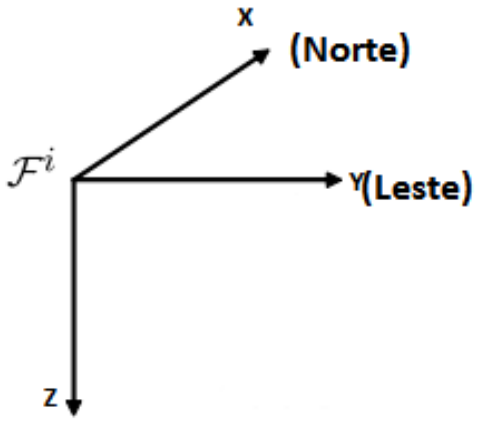
No que diz respeito à região de voo do quadrotor, tratando as coordenadas com unidades de distância de forma mais prática, *quadrotor* existem três *frames* principais para identificação de posição, orientação e dinâmica e também para facilitar o monitoramento durante missão, os quais seriam os *frames* inercial, ou *inertial frame*, relativo ao veículo (*vehicle frame*) e fixo ao corpo do veículo (*body frame*) [1].

A necessidade de tais *frames* se dá pelo fato de as leituras de sensores como acelerômetro e giroscópio retornarem valores relativos ao corpo do *quadrotor*, assim como cálculos do controle da dinâmica do *quadrotor*, enquanto leituras de GPS remetem valores relativos aos *frames* inercial e global [6]. Além disso, pontos de missão e a navegação são relacionados primeiramente aos últimos.

3.1 O frame inercial *Inertial frame*

O *inertial frame* tem sua origem no local de decolagem da aeronave. O eixo x aponta para o norte, eixo y para leste e eixo z para o centro da Terra. O *frame* está presente na Figura 18, e segue a convenção NED (*North-East-Down*), de comum emprego em veículo aéreos.

Figura 18 – *Inertial frame*, seguindo a convenção NED.

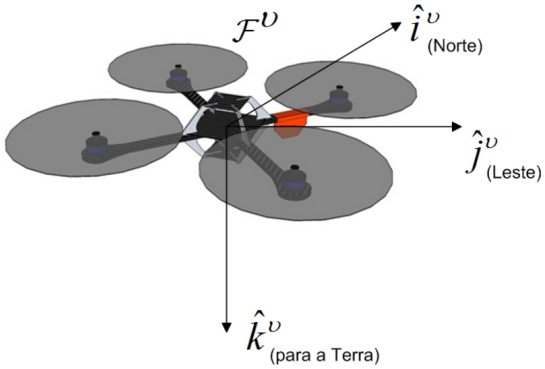


Fonte: Adaptado de [1]

3.2 O frame do veículo *Vehicle frame*

O *frame* possui origem no centro de gravidade do veículo, com seus eixos x e y sempre apontando para o norte e leste da Terra, respectivamente. O eixo z continua a apontar para o centro da Terra, seguindo assim a convenção NED.

Figura 19 – *Vehicle frame*, segundo convenção NED.

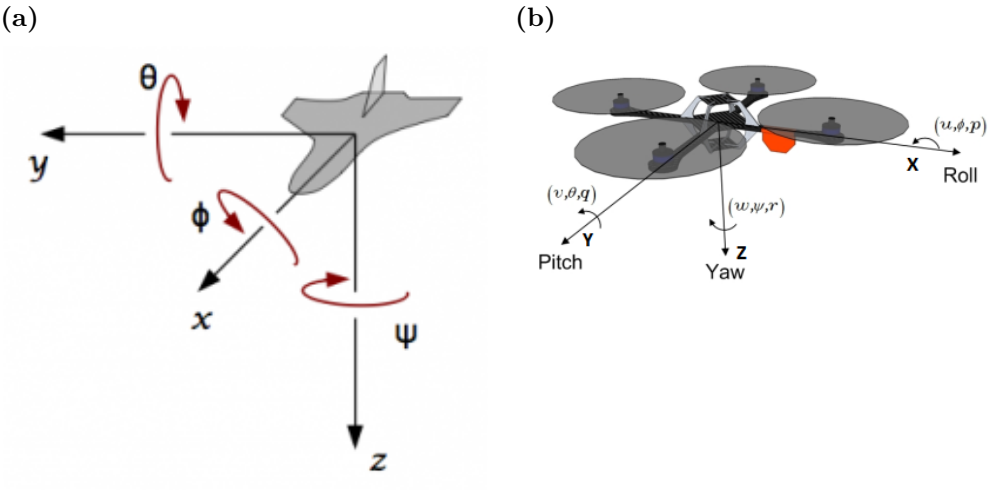


Fonte: [1]

3.3 O frame do corpo do veículo *Body frame*

O *Body frame* está acoplado ao corpo do quadrotor, seguindo todos os seus movimentos, e com origem também em seu centro de gravidade. O eixo x aponta para o nariz da aeronave, eixo y para a asa direita e k para o centro da Terra [1], mantendo a convenção NED. A Figura 20 ilustram esse *frame* anexado ao corpo de aeronaves diversas.

Figura 20 – *Body frames* em aeronaves diversas. a) *Body frame* em aeronave modelo asa fixa. b) *Body frame* em aeronave modelo quadrotor.



Fonte: [1]

3.4 Conclusões parciais

Tendo conhecimento de como o veículo se localiza no que diz respeito aos eixos coordenados, é possível saber como passar comandos de controle para as distâncias a serem percorridas, ou coordenadas a serem atingidas, no processo de planejamento de missão e, mais especificamente para este trabalho, no processo de busca do posicionamento correto para o pouso seguro da aeronave.

A forma mais direta escolhida para se enviar comandos a partir de dados da imagem seria referindo o movimento ao *body frame* do quadrotor. O *inertial frame* é de excelente ajuda para a análise de registros do deslocamento da aeronave após a missão.

4 Código desenvolvido para monitoramento

Para o presente trabalho foi desenvolvido um algoritmo de controle na linguagem *Python* [44], uma linguagem com estruturas de alto nível, interpretada, de fácil interação com elementos de *hardware* e outras linguagens de programação e sintaxe simples, e além de tudo uma plataforma *Open Source*. Ela encoraja o emprego de módulos e bibliotecas, de forma a favorecer a portabilidade e leitura do código. Outra vantagem de se empregar essa linguagem de programação foi a existência de bibliotecas de alta eficácia e que permitiam o emprego das técnicas de visão computacional e comunicação com o quadrotor em alto nível.

Existem variadas distribuições da linguagem *Python* disponíveis na internet. A escolhida para o trabalho foi a *Anaconda* [46], por apresentar criação fácil de diversos ambientes e pacotes para análise científica de dados. A versão do *Python* foi a 2.7, pela compatibilidade com as bibliotecas utilizadas.

Para a visão computacional foi escolhida a biblioteca *Open Source Computer Vision library (OpenCV)* [51], a qual oferece suporte para *Python* e contém milhares de algoritmos desenvolvidos com técnicas clássicas e de estado da arte em visão computacional, incluindo detecção de faces e objetos e segmentação de imagens das mais diversas formas, e promove interação fácil com a câmera para captura e processamento de vídeo. Essa biblioteca é utilizada para detecção do *landmark* no trabalho.

Para a comunicação com o quadrotor foi empregada a biblioteca *dronekit*, a qual é importada como módulo pelo código. A mesma é uma *Application Program Interface (API)* desenvolvida para fazer a ponte entre o quadrotor e o código de controle através do protocolo *Mavlink*, explicado em mais detalhes na Seção 6.3. O código de controle é executado em outro computador que não seja a placa controladora de voo, com maior capacidade de processamento, como um computador embarcado *Companion Board* ou mesmo o próprio computador onde se encontra a GCS. Com o emprego dessa biblioteca é possível inserir mais funcionalidades ao quadrotor que são de processamento mais custoso ou crítico, como visão computacional, modelagem 3D e planejamento de caminhos durante o voo.

Após a conexão com o quadrotor, feita normalmente através de uma porta serial, os dados do mesmo podem ser encaminhados à GCS através de uma porta TCP do próprio computador. A aeronave é interpretada como um objeto instanciado no código, no qual as várias funções e propriedades agem sobre e são características do próprio quadrotor. Funcionalidades listadas na documentação da biblioteca apontam para as seguintes funções principais da API.

- Receber e atuar sobre parâmetros e estado do veículo.

- Receber notificações assíncronas sobre mudança de estado.
- Guiar o quadrotor a uma posição determinada.
- Enviar mensagens customizadas para controlar o movimento do quadrotor e equipamentos a ele conectados.
- Criar e gerenciar *waypoints*.
- Sobrescrever comandos do rádio controle.

Por fim, para o auxílio da programação foi utilizada a IDE *Visual Studio* [50], por apresentar interface amigável agrupar todos os módulos criados em um único projeto com um único ambiente de *Python*.

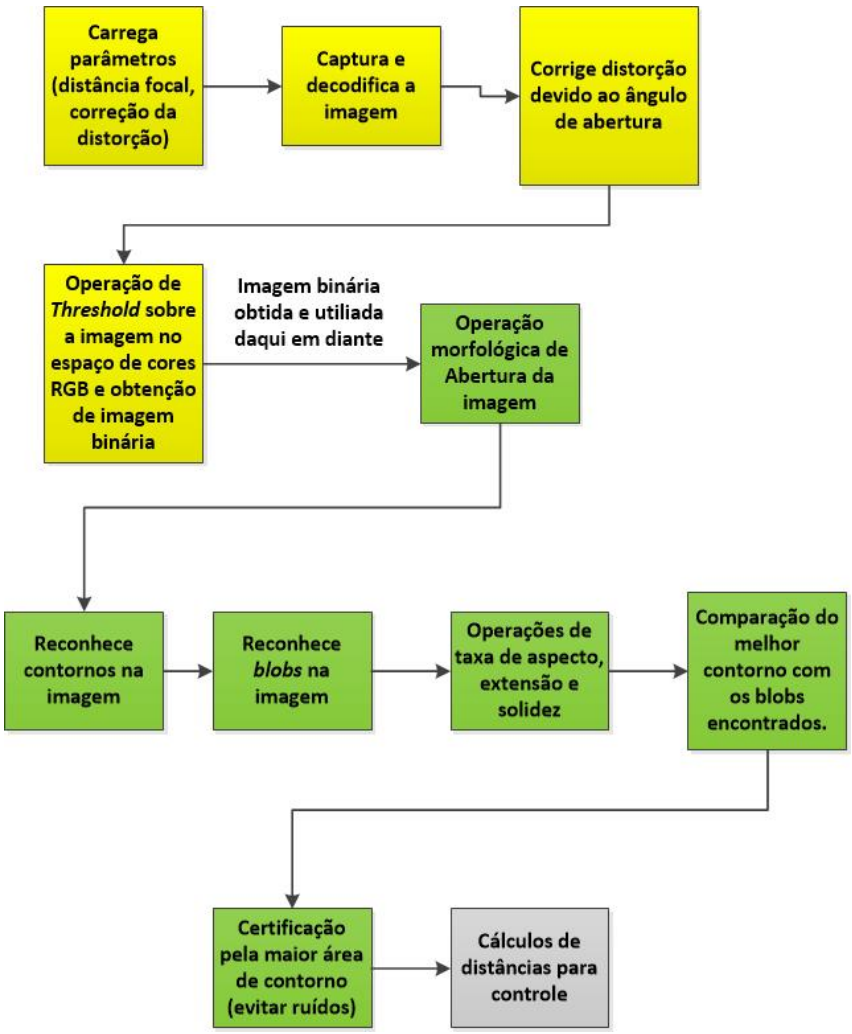
5 Sistema de Visão

O presente trabalho utiliza uma câmera *GoPro Hero 3* para reconhecimento de características na imagem para assim controlar o quadrotor. A característica que se busca reconhecer é o *landmark* presente na Figura 21. O círculo central branco contornado por um envoltório preto produz um alto contraste e identificação no caso do processamento da imagem em escala de cinza, portanto foi escolhido para o projeto.

Figura 21 – *Landmark* desenvolvido para o projeto, com círculo central medindo 1,40 metros.

O esquema da Figura 22 demonstra como é identificada a característica circular na imagem pelo algoritmo desenvolvido em *Python* com o auxílio da biblioteca *OpenCV*, obtendo um resultado robusto de forma simples.

Figura 22 – Esquema sobre o processamento da imagem, onde o contorno e *blob* que se desejam encontrar correspondem ao círculo central do *landmark*.



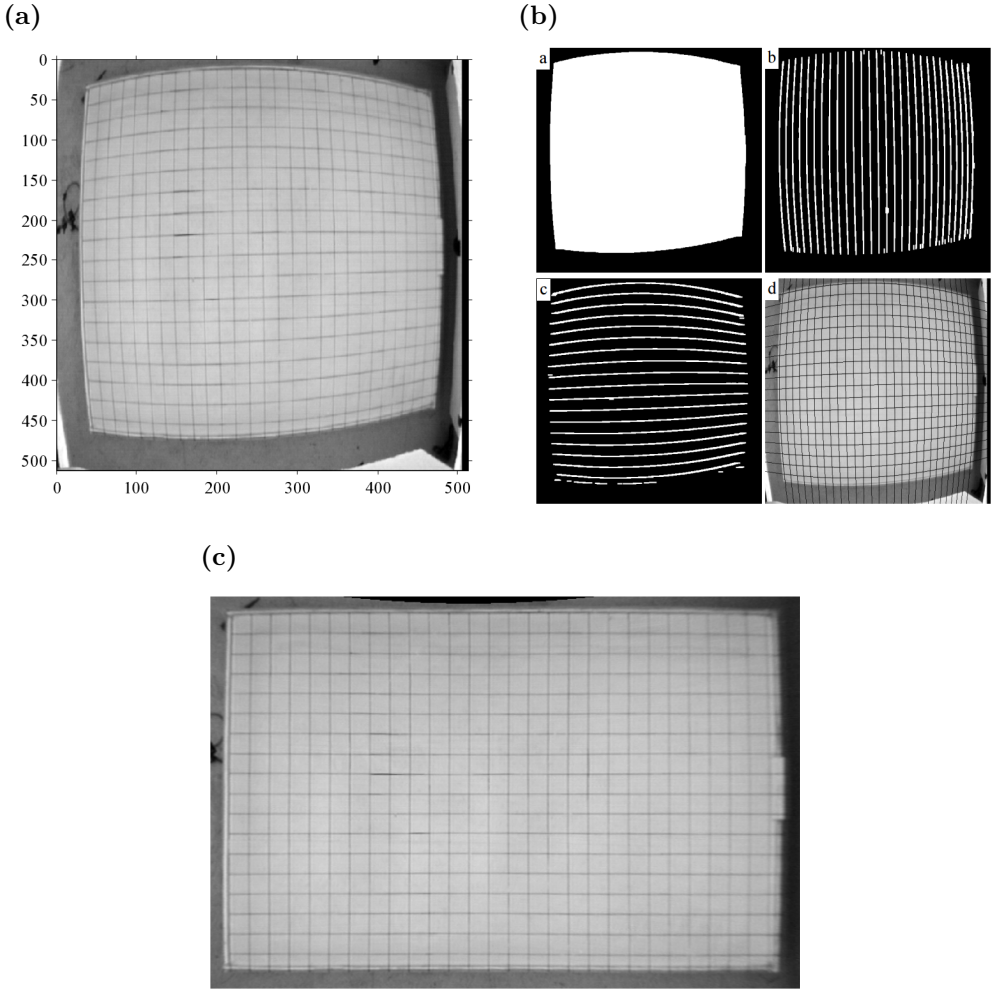
5.1 Correção da distorção

Vários tipos de lente, em particular as que buscam aumentar o campo de visão capturado pela câmera, atribuem distorção à imagem capturada, o que dificulta o processo de mapeamento de características na mesma. Essa distorção geralmente ocorre de forma radial, tangencial ou em formato de barril, e existem soluções tanto na academia quanto em *softwares* comerciais para correção deste fenômeno, várias delas envolvendo o processo de calibração da câmera, onde se obtêm dados parâmetros para essa [7].

O estudo realizado por [4] traz um algoritmo que busca corrigir a distorção em barril, a qual comprime os *pixels* centrais e dilata os da periferia, criando um efeito parabólico sobre retas presentes na imagem. É introduzido um processo para detecção de retas em um padrão através de aproximação parabólica, e o remapeamento da imagem é feito por relações trigonométricas após encontrados parâmetros da câmera. Vale notar que a escala da imagem é alterada no processo de reconstrução da imagem final, portanto

os valores de alguns *pixels* devem ser interpolados segundo suas vizinhanças. O resultado está exemplificado na Figura 23.

Figura 23 – Processo de correção de distorção. a) Folha original com retas encurvadas. b) Reconhecimento das retas e aproximação por parábolas. c) Resultado após correção da distorção.



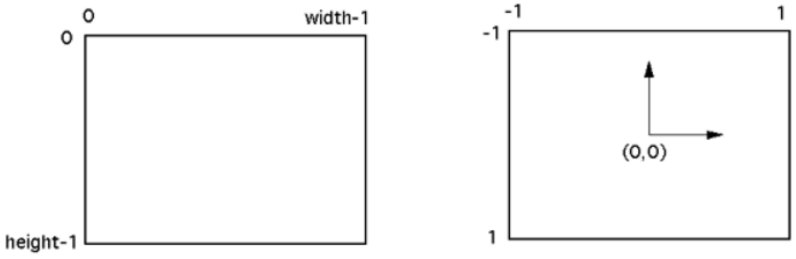
Fonte: [4]

No trabalho realizado por [19] é atacada a forma radial de distorção, mais comum entre as lentes com elevada abertura angular, em particular as de olho de peixe, onde a abertura varia de 120 a 180 graus. O mesmo se mostra rápido por utilizar técnicas geométricas ao invés de polinomiais para correção da distorção, conseguindo resultados em tempo real, e evitando aproximações para valores de *pixels*, uma vez que todos os *pixels* podem ser mapeados. Uma matriz é obtida de forma a corrigir todas as imagens provenientes após a primeira iteração do algoritmo.

Existem técnicas mais simples e disseminadas para aproximação utilizando equações que visam corrigir a distorção radial [7]. Sejam P_x e P_y as coordenadas de um *pixel* na imagem original e P'_x e P'_y as mesmas após corrigidas na imagem de saída, e a origem

das coordenadas dos *pixels* situada no centro da imagem normalizada (Figura 24). Sendo assim, a Equação 5.1 relaciona o posicionamento do mesmo nas duas imagens.

Figura 24 – Sistema de coordenadas sobre a imagem para cálculos sobre os *pixels*.

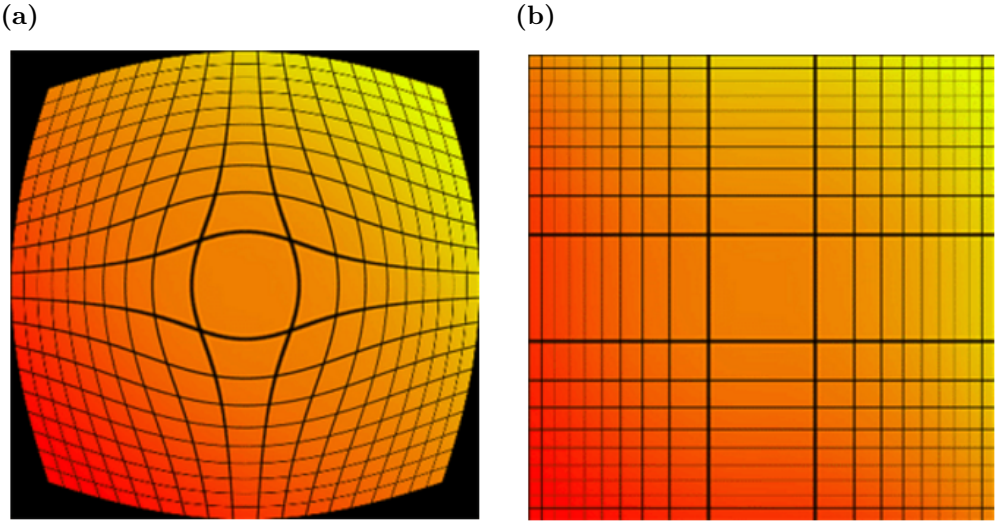


$$P'_x = P_x(1 - a_x|P|^2) \tag{5.1a}$$

$$P'_y = P_y(1 - a_y|P|^2) \tag{5.1b}$$

Sendo os parâmetros a_x e a_y particulares a cada lente. Outras equações aproximam as duas imagens por relação senoidal, quadrática, entre outras. A Figura 25 exemplifica o efeito do mapeamento supondo aproximação quadrática entre as imagens de entrada e saída.

Figura 25 – Efeito quadrático sendo corrigido na imagem distorcida. a) Linhas de imagem distorcida teoricamente de forma quadrática. b) Correção após operação sobre posicionamento dos *pixels*.

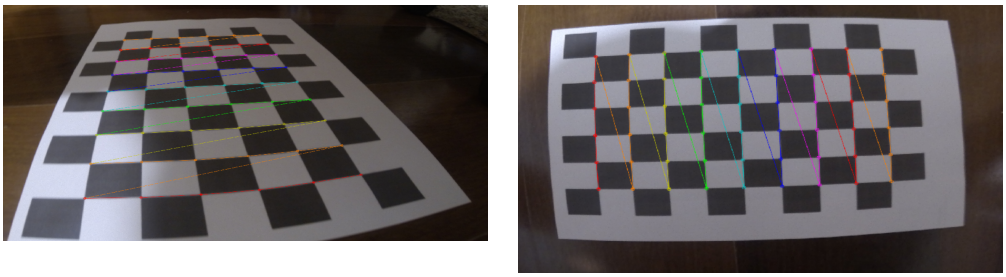


A câmera escolhida captura imagens de forma distorcida devido ao formato de sua lente. Sendo assim, deve ser desenvolvido um algoritmo para calibração da câmera e correção dessa distorção, de forma a mapear as distâncias em *pixels* de forma correta.

A biblioteca *OpenCV* contém funções desenvolvidas com intenção de auxiliar nessa correção, obtendo parâmetros da câmera. Para esse procedimento deve-se estudar um objeto com padrão bem definido a ser capturado pela câmera de vários ângulos, a fim de tentar corrigir da melhor forma com interpolação as distorções observadas no padrão. No caso geral, é utilizado um tabuleiro de xadrez, identificado como na Figura 26. Vale salientar que para uma correção mais satisfatória é necessário reduzir as dimensões e definição da imagem (*zoom out*) a fim de reduzir o alongamento de características presentes nas bordas da imagem resultante.

Figura 26 – Tabuleiro de xadrez utilizado para calibração da câmera. a) Tabuleiro em plano bastante inclinado em relação ao da câmera, distorção considerável. b) Tabuleiro em um plano mais paralelo ao da câmera, menor distorção.

(a) **(b)**



A aproximação realizada pela biblioteca é polinomial para a correção radial, e basicamente segue as regras da Equação 5.2.

$$x_{corrigido} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{5.2a}$$

$$y_{corrigido} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \tag{5.2b}$$

Para a correção tangencial, que ocorre devido à posição da câmera não ser paralela ao plano da imagem e assim alguns objetos aparentarem mais próximos uns dos outros que a realidade, é utilizada a Equação 5.3:

$$x_{corrigido} = x + [2p_1 xy + p_2(r^2 + 2x^2)] \tag{5.3a}$$

$$y_{corrigido} = y + [p1(r^2 + 2y^2) + 2p_2 xy] \tag{5.3b}$$

Logo, os parâmetros a serem calculados são como segue em 5.4.

$$coeficientes = [k_1, k_2, p_1, p_2, k_3] \tag{5.4}$$

Após a análise das imagens e obtenção dos parâmetros da câmera, os cálculos são realizados novamente sobre o posicionamento dos *pixels* a fim de corrigir a distorção, como ilustrado na Figura 27. Dessa forma, as características do *landmark* são preservadas e é possível identificá-lo de forma mais coerente no que diz respeito às suas dimensões reais.

Figura 27 – Correção da imagem. Destaque para o formato do *landmark* e as distâncias entre o mesmo e os outros objetos da cena. a) Imagem original, com campo de visão grande e *landmark* ovalado graças à distorção. b) Região de interesse da imagem, com *landmark* apresentando característica bem mais plana, circular e corrigida.

(a) **(b)**

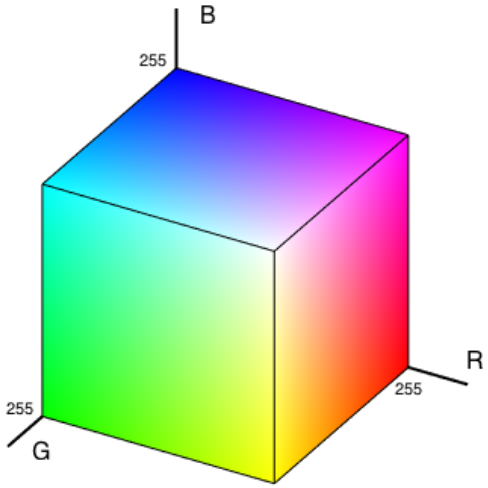
5.2 Tratamento da imagem

A imagem capturada pela câmera em formato digital é interpretada pelo computador como uma matriz de *pixels*, cada um dos elementos contendo um valor para o que foi captado do ambiente naquela posição da imagem, que pode se referir tanto a cor quanto à intensidade luminosa, por exemplo.

O espaço de cores da imagem capturada é denominado RGB. Também chamado de sistema cor-luz [35], ele basicamente combina o nível de vermelho, verde e azul encontrado em cada ponto no ambiente para resultar em outras cores diversas. A matriz resultante possui então três dimensões, sendo sua largura e altura de acordo com a resolução da câmera para cada uma das cores.

O nível é definido entre valores de 0 a 255, pois cada tom recebe um *byte* de representação. Como exemplo, um pixel com grande intensidade de vermelho possui valor (255, 0, 0), enquanto um com alto índice de azul possuiria (0, 255, 0). Os *pixels* brancos possuem grande intensidade de todas as três cores, obtendo valor próximo de (255, 255, 255), e em contrapartida os pretos apresentam valores próximos a (0, 0, 0). A Figura 28 ilustra o espaço RGB.

Figura 28 – Espaço de cores RGB, com os valores entre 0 e 255 para cada uma das dimensões e a cor resultante do *pixel*.



O primeiro passo do trabalho da imagem é referente ao seu escalonamento. A imagem original possui resolução de *pixels* geralmente elevada, o que traz detalhes considerados desnecessários para a aplicação e acarreta em maior tempo de processamento. Portanto, através da função da biblioteca *OpenCV resize()* é possível reduzir o número de pixels de uma imagem mantendo a sua característica, através de interpolação dos valores de cada *pixel* a serem transmitidos para os *pixels* da imagem resultante.

A segmentação da imagem consiste na extração de características inerentes a um objeto ao qual deseja-se buscar em uma dada região da imagem [5]. Para que seja buscado o *landmark* proposto na imagem é preciso obter a maior discrepância entre os valores dos pixels do ambiente possível. Para isso a imagem capturada pela câmera precisa ser convertida em escala de cinza.

Ao se converter uma imagem em escala de cinza calcula-se a média do valor de cada *pixel*, portanto não sendo mais o mesmo representado por três valores, mas somente um. Dessa forma os *pixels* brancos possuem um valor próximo a 255, enquanto os pretos, próximos a 0. O processamento da imagem se torna bem mais rápido, por ter reduzindo em uma dimensão a matriz que a representa, e as cores branca e preta se mantém o máximo possível distintas em termos de valores.

Por fim, realiza-se o *thresholding* da imagem. Nessa operação é estabelecido um valor de limite T e a imagem em escala de cinza é avaliada *pixel* a *pixel* sobre esse limite

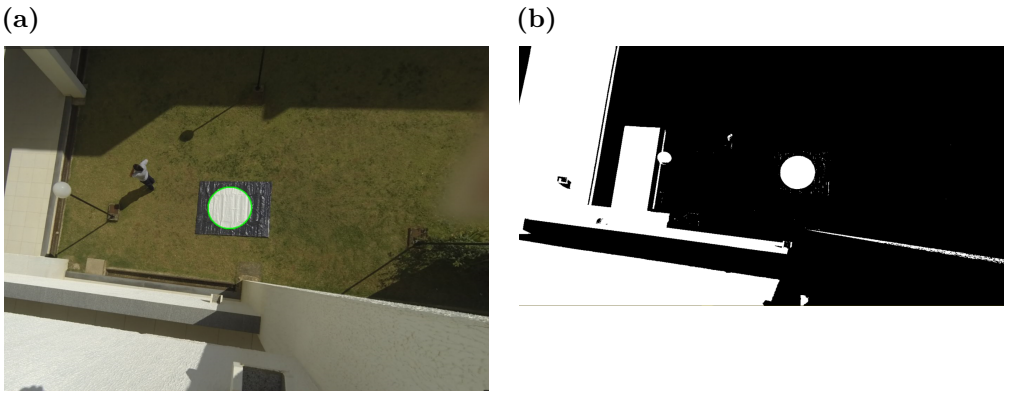
como estabelecido na Equação 5.5.

$$pixel(i,j) = \begin{cases} 255 & \text{se } pixel(i,j) > T \\ 0 & \text{caso contrário} \end{cases} \tag{5.5}$$

Onde $pixel(i,j)$ contém o valor do *pixel* em escala de cinza na posição (i,j) da imagem, e T é o valor do limite.

Essa operação é realizada pela função *threshold*, e o resultado é uma imagem dita binária. A Figura 29 ilustra o resultado de todas as operações descritas sobre a imagem original.

Figura 29 – Resultado da operação de *threshold* na imagem. a) Imagem no domínio RGB, com círculo central do *landmark* reconhecido. b) Imagem binária após operação de *threshold.*



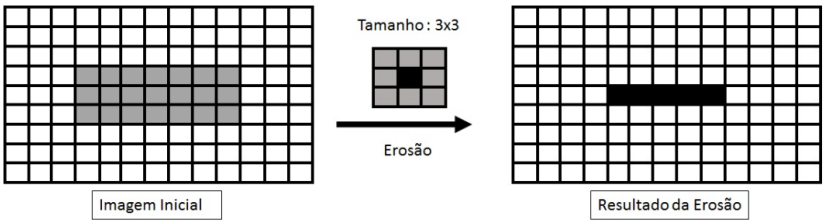
5.3 Operações morfológicas

A imagem da Figura 30b obtida na Seção 5.2 apresenta ruídos que podem vir a atrapalhar o resultado da Segmentação proposta no trabalho. Para melhorar a qualidade da imagem nesse sentido duas operações morfológicas básicas são combinadas, sendo essa a erosão e dilatação.

Para fazer as operações morfológicas, é comum o uso de *kernels*. O *kernel* é uma matriz de dimensões e valores pré determinados, que vêm a ser utilizado operando sobre a matriz da imagem por convolução, sombreando determinada região em cada instante, realizando operações sobre o *pixel* central da área sombreada [14].

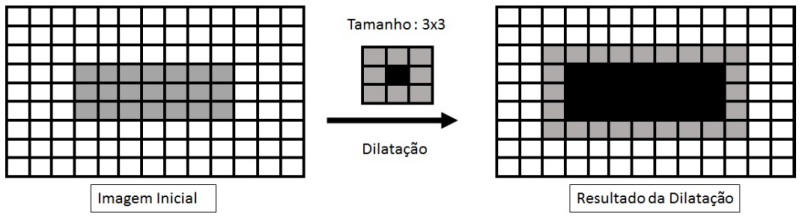
Na erosão ocorre a remoção dos contornos abruptos das formas, do seguinte modo: se todos os *pixels* da região sombreada possuírem valor 1, o *pixel* central naquele instante permanece em valor 1, caso contrário, o mesmo recebe valor 0. A Figura 30 ilustra o processo de erosão.

Figura 30 – Ilustração do processo de erosão da imagem.



No processo de dilatação ocorre o inverso: ao fazer passar o *kernel* sobre a matriz de *pixels* da imagem, se algum *pixel* da área sombreada possuir valor 1, então o *pixel* central também receberá esse valor, caso contrário receberá valor 0. A Figura 31 ilustra o processo de dilatação.

Figura 31 – Ilustração do processo de dilatação da imagem.



Ao combinar essas duas técnicas no trabalho utilizou-se a operação chamada Abertura, na qual a imagem passa por uma sequência de erosões e depois dilatações. O resultado elimina pequenos pontos considerados ruídos na imagem de acordo com as dimensões apropriadas de *kernel*, ao mesmo tempo que suaviza os contornos das formas, o que é interessante para ressaltar o círculo na imagem segmentada (Figura 32). O resultado final após essas duas operações sobre a imagem binária inicial está ilustrado na Figura 33.

Figura 32 – Ilustração da operação de abertura da imagem.

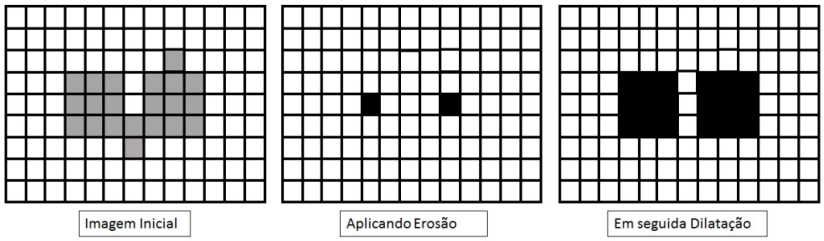
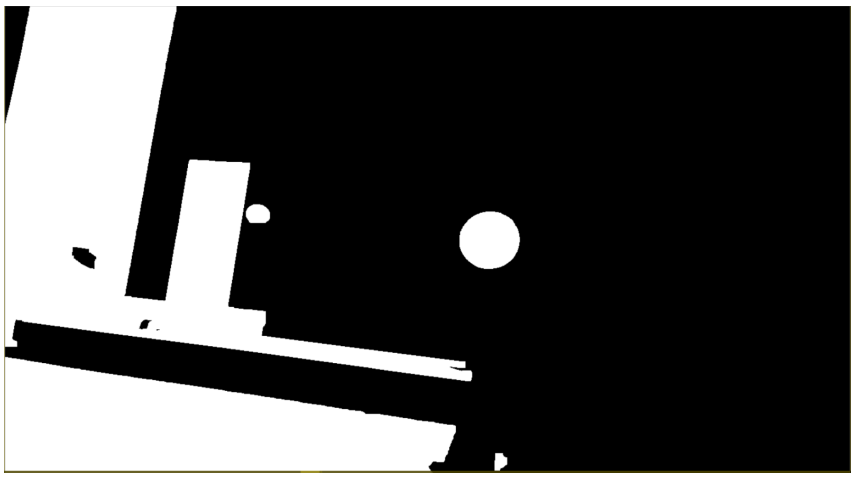


Figura 33 – Resultado da operação de abertura sobre a imagem binária.



5.4 Segmentação da imagem - extração final da característica

Para a identificação e certificação dos *blobs* utiliza-se funções da biblioteca *OpenCV* na obtenção de dados básicos como área, perímetro, e dimensões dos contornos identificados na imagem em escala de cinza, e a partir daí calcular a solidez, extensão e taxa de aspecto dos mesmos. Existe uma função implementada na biblioteca *OpenCV* chamada *SimpleBlobDetector* que busca por pontos chaves na imagem de acordo com parâmetros de interesse, como circularidade, valor do *pixel* em escala de cinza, área, entre outros. A função é utilizada para atribuir mais robustez ao algoritmo e eliminar ruídos.

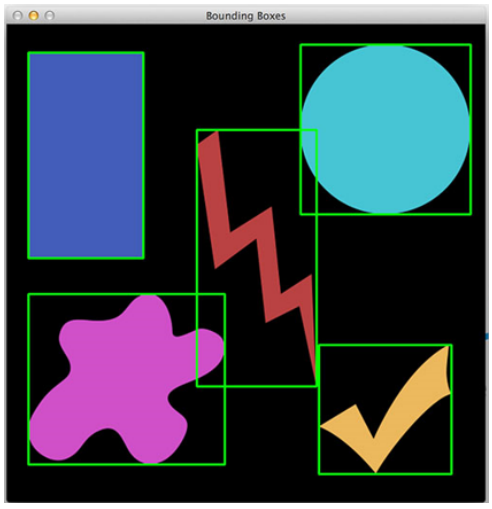
O conceito de taxa de aspecto (ar) é simples e está definido na Equação 5.6:

$$ar = \frac{W_{objeto}}{H_{objeto}} \tag{5.6}$$

Onde W_{objeto} é a largura do objeto e H_{objeto} a altura do mesmo, em *pixels*, na imagem.

Para a extensão é preciso inserir o conceito de *bounding box*. A *bounding box* obtida sobre um contorno com a biblioteca *OpenCV* é o envoltório retangular (caixa) mínimo necessário para delimitar o contorno. A Figura 34 ilustra um exemplo com *bounding boxes* sobre contornos da imagem.

Figura 34 – *Bounding boxes* em contornos na imagem.



Fonte: [53]

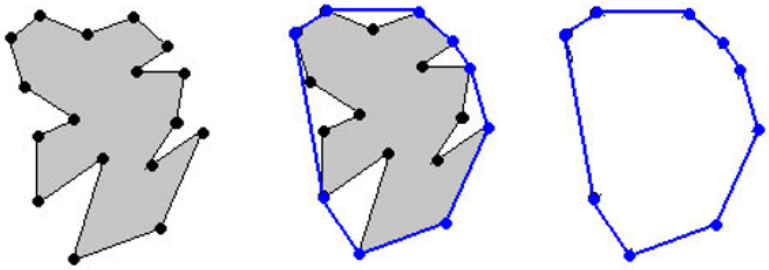
Dessa forma, a equação que define a extensão ex de um objeto é como na Equação 5.7:

$$ex = \frac{A_{objeto}}{A_{boundingbox}} \tag{5.7}$$

Onde A_{objeto} é a área do objeto e $A_{boundingbox}$ a área da *bounding box* respectiva na imagem, em *pixels*.

Outro aspecto a ser notado sobre os contornos de uma imagem é o envoltório convexo. Como o nome diz, é uma técnica que liga os pontos limites do contorno de forma a construir o menor envoltório possível que tenha forma convexa. A Figura 35 ilustra um exemplo de construção de um envoltório convexo.

Figura 35 – Exemplo de envoltório convexo.



Fonte: [53]

Sendo o envoltório construído sobre os contornos, a Equação 5.8 define a solidez *sl*

de um contorno.

$$sl = \frac{A_{objeto}}{A_{contornoconvexo}} \tag{5.8}$$

Onde A_{objeto} é a área do objeto e $A_{contornoconvexo}$ a área do contorno convexo respectivo na imagem, em *pixels*.

Com esses três dados é possível identificar um círculo

Fonte: [53]

por aproximar a taxa de aspecto para 1 (mesma altura e largura), a extensão para $\frac{\pi}{4}$ (área de um círculo sobre a do quadrado que o envolve) e a solidez para 1 (o contorno convexo deveria envolver o círculo seguindo o seu formato).

As Figuras 36, 37 e 38 ilustram o reconhecimento do círculo central do *landmark* em ambientes variados, incluindo fechados e abertos, com condições diferentes de iluminação. Para facilitar os testes em simulação um modelo em miniatura de *landmark* foi impresso e também usado na avaliação do algoritmo.

Figura 36 – Reconhecimento do *landmark* em sala iluminada. Reparar o efeito do brilho refletido pela madeira na imagem binária.

Figura 37 – Reconhecimento do *landmark* em local aberto em região ensolarada.

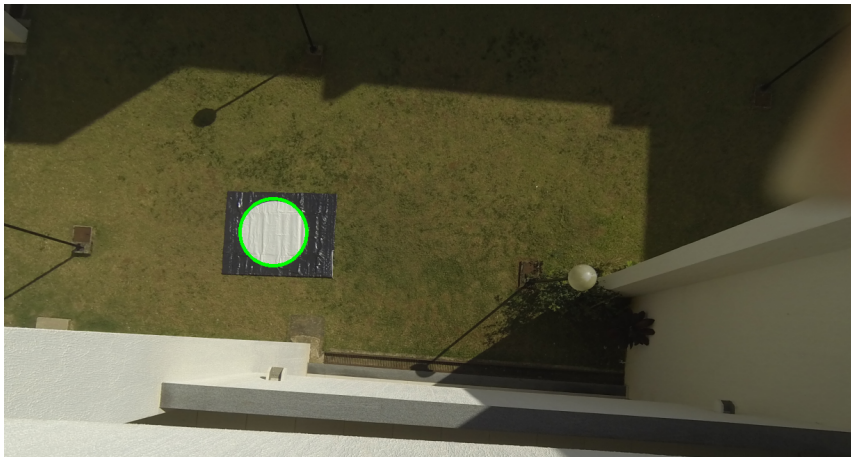
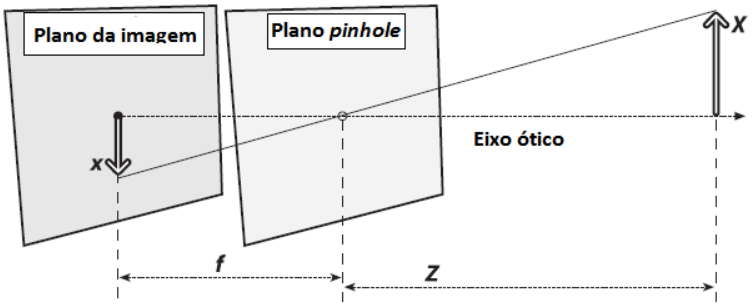


Figura 38 – Reconhecimento do *landmark* em dia nublado, em um plano inclinado em relação ao plano da câmera.

Uma vez identificado o círculo central pode-se utilizar técnicas para determinar a distância do objeto à câmera e a posição relativa horizontal do quadrotor no mundo real ao *landmark* segundo a localização do último no plano da imagem. Para tal, é preciso descobrir a taxa de transformação da câmera utilizada no processo de captura das imagens.

A imagem é formada no plano focal da câmera de acordo com a lente que a mesma possui, e através de semelhança de triângulo observada no esquema montado na Figura 39 pode-se obter a taxa de transformação da câmera através da Equação 5.9:

Figura 39 – Distância do plano focal de uma câmera no modelo *pinhole*.



$$f = \frac{x \cdot Z}{X} \tag{5.9}$$

Onde f é a taxa de transformação, x é o diâmetro do *landmark* identificado na imagem em *pixels*, Z a distância em metros do *landmark* à câmera em metros e X o diâmetro do *landmark*, também em metros. Basicamente, de posse desse valor de f é possível converter as distâncias em *pixels* para metros no mundo real.

Ao se obter uma imagem com o contorno identificado e sendo sabidos X, x e f é possível aproximar qualquer distância através da mesma semelhança de triângulo da

Figura 39 com a Equação 5.10:

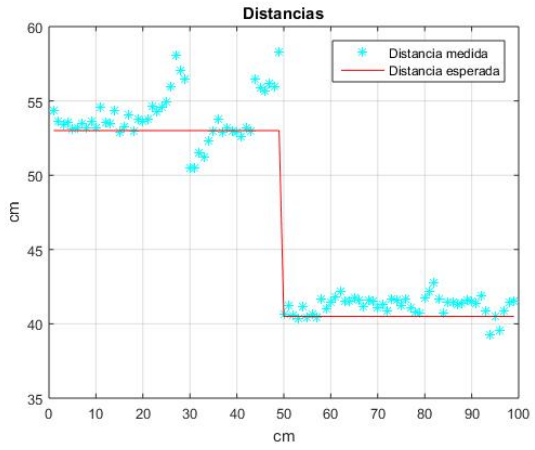
$$Z = \frac{X \cdot f}{x} \tag{5.10}$$

A câmera está posicionada aproximadamente sobre o centro de gravidade do quadrotor. Uma vez determinada a distância da câmera ao *landmark* tem-se diretamente a altitude naquele instante. A Figura 40 ilustra a altitude obtida a partir de uma distância medida no mundo real de 45 centímetros, a fim de observar a precisão dos valores calculados a partir da câmera em relação aos reais, e um gráfico de medições repetidas para confirmar a eficácia da câmera.

Figura 40 – a) *Landmark* observado a uma distância medida de 45 centímetros, com resultado obtido de 45,38 centímetros pela câmera. b) Gráfico após 100 medições em distâncias variadas, com erro médio de 1,08 centímetros.

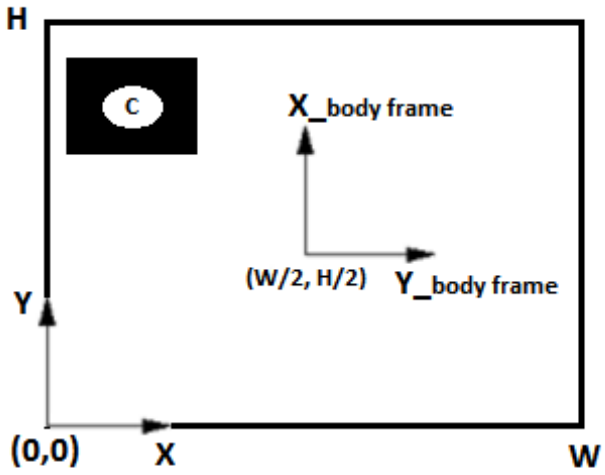
(a)

(b)



A câmera é colocada fisicamente no quadrotor de forma que a parte superior da imagem esteja orientada para o nariz da aeronave. Os eixos do X e Y do*body frame*, a partir daqui tratados por somente X e Y, têm a direção então dos eixos Y e X da imagem, respectivamente (como ilustrados na Figura 41). As distâncias p a serem percorridas estão relacionadas ao centro da imagem, local onde se situa o centro de gravidade do quadrotor, e são calculadas em termos de *pixels* segundo Equação 5.11.

Figura 41 – Eixos orientados para obtenção de coordenadas na imagem.



$$p_x = X_c - \frac{W}{2} \tag{5.11a}$$

$$p_y = Y_c - \frac{H}{2} \tag{5.11b}$$

Onde W e H são a largura e altura da imagem, respectivamente, conforme Figura 41, e o ponto C o centro do *landmark* visualizado e identificado.

É sabido o diâmetro real do círculo central do *landmark*. A técnica usada para cálculo das distâncias em X e Y consiste na obtenção de uma constante K que relaciona esse diâmetro com a quantidade de *pixels* correspondente na imagem. A constante é encontrada para cada imagem então como na Equação 5.12.

$$K = \frac{X}{x} \tag{5.12}$$

Onde X é o diâmetro do círculo em medidas reais e x o diâmetro em *pixels* obtido na imagem.

De posse desse constante é possível determinar qualquer distância entre dois pontos na imagem a partir da quantidade de pixels entre os mesmos, de acordo com a Equação 5.13.

$$M = K \cdot p \tag{5.13}$$

Onde M são as distâncias no mundo real, em metros, e p em *pixels* calculadas na Equação 5.11.

O controle busca posicionar o *drone* sobre o *landmark* para assim executar o pouso, e para isso o último deve estar sendo visualizado no centro da imagem obtida pela câmera. De posse da Equação 5.13 pode-se obter então as distâncias no mundo real a serem percorridas em X e Y a partir das distâncias em pixels na imagem do centro do *landmark* ao centro da imagem. Esse resultado é calculado e passado a frente na rotina de controle.

6 Arquitetura do sistema de controle

6.1 Controlador de voo (*autopilot*)

A placa acoplada ao quadrotor para controle de baixo e alto nível é a *Pixhawk*[52], produzida pela *PX4 autopilot*. Possui acelerômetro, giroscópio, magnetômetro e barômetro acoplados, processador 168 MHz Cortex M4F (256 KB RAM, 2 MB Flash) com *slot* para cartão micro SD e várias portas UART, CAN, I2C, SPI e ADC para conexões externas.

Figura 42 – Placa controladora *Pixhawk*.

A placa roda um sistema operacional de tempo real (RTOS) NuttX para escalonamento de tarefas e suporta o *software Ardupilot*, código livre implementado em C++ e disseminado na internet, com suporte ativo de comunidades [54].

O *software* realiza a leitura tanto dos sensores presentes da própria placa quanto dos conectados externamente, como GPS, sonar e magnetômetro externo, de forma eficiente e temporizada, de forma a fornecer informações em tempo hábil para as malhas de controle de posição, altitude e atitude, calculando assim a potência a ser enviada para cada motor em cada instante. Também é realizado o controle de missão do quadrotor, a partir da navegação sobre os *waypoints* gravados na memória.

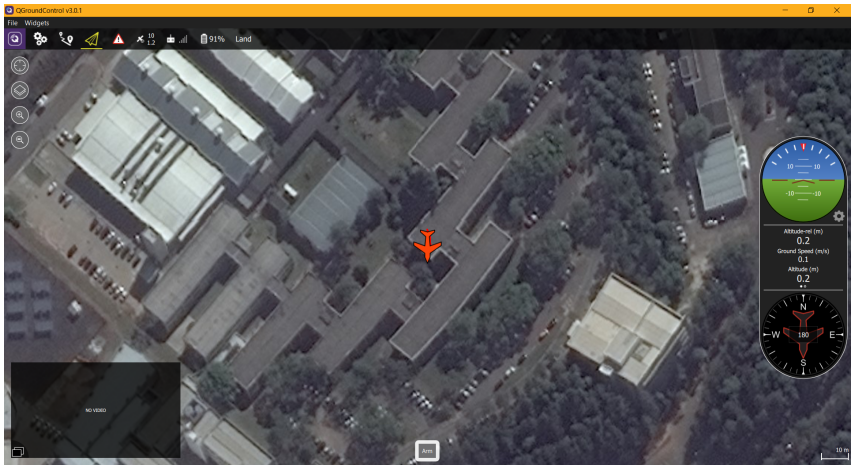
Por fim, a leitura do rádio transmissor, gravação de LOGs e checagens sobre possíveis falhas também são encontrados no código. Sua robustez é garantida devido à disseminação e à grande comunidade desenvolvedora no repositório online *Github*.

6.2 *QGround Control*

A GCS adotada para o trabalho foi o *software QGround Control* [49]. A plataforma possui código aberto que pode ser encontrado na internet, e é executada em um *laptop* com sistema operacional *Windows*.

A interface possui basicamente: um HUD onde podem ser vistos dados rápidos como altitude e principalmente orientação e nível da aeronave em tempo real; Um quadro de dados, como qualidade dos sensores, atitude da aeronave, posição e velocidade, entre outros, segundo a necessidade de monitoramento; e uma janela principal com a vista do veículo sobre um mapa no local de voo, com sua orientação e pontos de missão ilustrados.

Figura 43 – Tela inicial da GCS *QGroundControl*.



O *software* também realiza LOGs dos dados do veículo durante a missão, e os mesmos podem ser revisados em vídeos, traçados em gráficos, e exportados para outros formatos, como dados em linguagem *Matlab*.

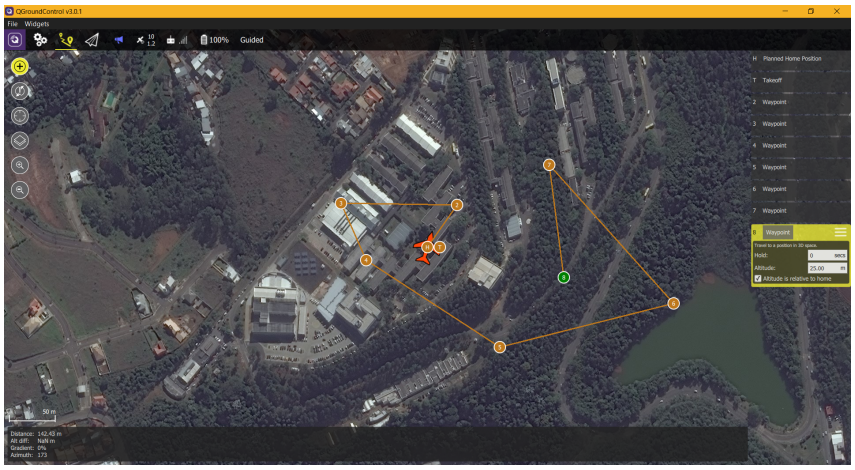
A integração com o software *Ardupilot* permite a alteração rápida de parâmetros da aeronave, como os ganhos dos controladores de posição e atitude, os valores de paradas de emergência, entre outros. O *firmware* pode ser baixado rapidamente na versão desejada do repositório *Github* e gravado no controlador de voo, de forma a garantir sempre uma versão estável.

Finalmente, a parametrização de sensores acoplados à placa controladora, como o sonar, monitor de bateria, e a calibração dos sensores já presentes na placa, barômetro, acelerômetro, giroscópio e magnetômetro, também podem ser auxiliados pelo *software*.

A conexão com o quadrotor pode ser realizada através de uma porta COM à qual o mesmo estaria ligado através de um cabo USB (com a taxa de transmissão 115200 bits por segundo), ou por um módulo xBee (normalmente 57600 bits por segundo) ou através de uma porta TCP ligada à rede, o que normalmente foi utilizado durante a simulação dos comandos nesse trabalho, e será discutida em Seções seguintes.

Após a conexão do veículo o mesmo pode ser comandado através da GCS. Todo o planejamento da missão no que diz respeito aos *waypoints* e o que fazer em cada um deles, os modos de voo, altitudes, velocidades e outros dados podem ser enviado ao veículo no momento que precede a decolagem, e após a mesma existe também possibilidade de reatividade de missão, com mudanças de trajetos, modos de voo e saídas de segurança, por exemplo, para caso de perda de comunicação.

Figura 44 – Planejamento da missão autônoma na GCS.



6.3 Protocolo *Mavlink*

O protocolo *Mavlink* (*Micro Air Vehicle Communication Protocol*) é utilizado vastamente na área da robótica, principalmente no que diz respeito a veículos aéreos. Suas mensagens são leves e contêm somente cabeçalhos [48].

O protocolo envia as estruturas de forma serial e possui mensagens de propósitos variados, funcionando em harmonia com o *software Ardupilot* e a GCS. Aplicações em separado, como o presente trabalho, também possuem suporte para integração, sendo a biblioteca contendo os cabeçalhos das mensagens desenvolvida em várias linguagens, como C++, C# e Python. O trabalho de [2] utiliza mensagens com protocolo *Mavlink* para realizar a comunicação entre o controlador de voo APM e uma placa *Raspberry Pi* embarcada para processamento de imagens no momento do pouso, acionando a funcionalidade através de um canal do rádio transmissor.

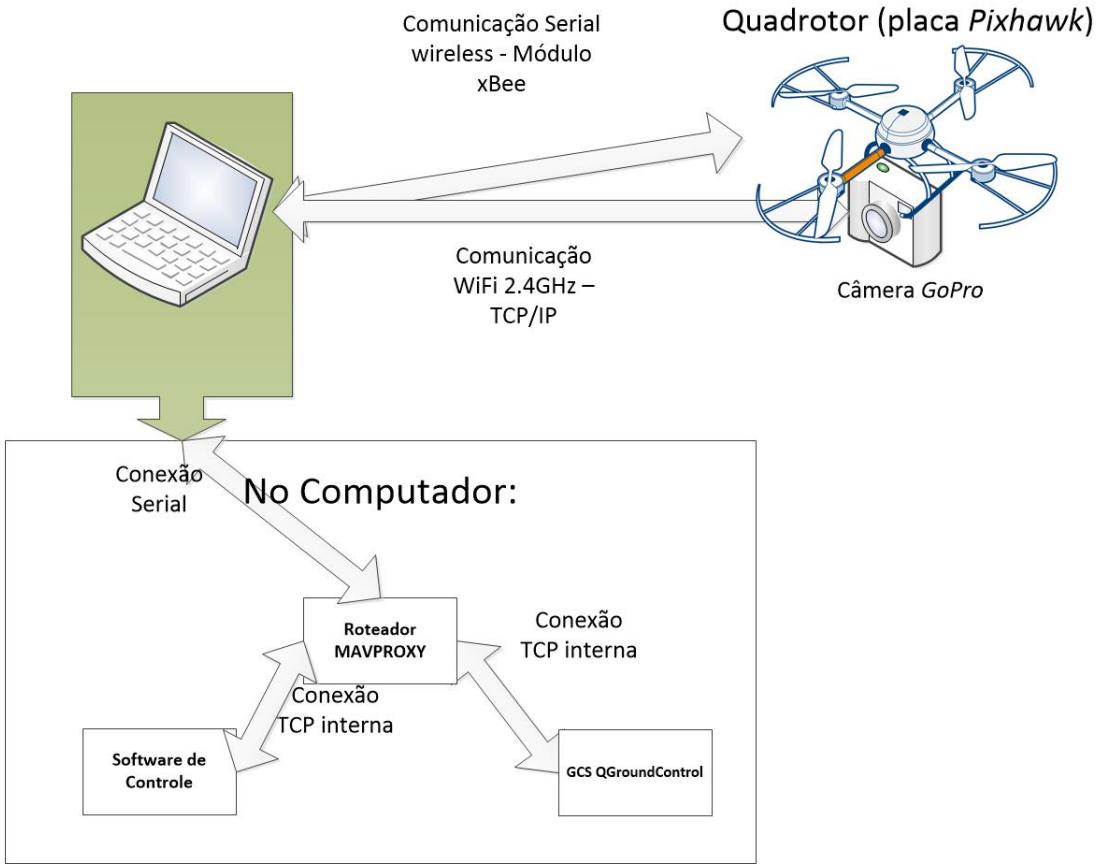
No presente trabalho, além da comunicação usual do veículo com a GCS, o protocolo é utilizado para enviar novos comandos de *waypoint* segundo resultados do controle de visão.

6.4 Sistema de comunicação e supervisão

Unindo os componentes de *hardware* e *software* de forma a garantir bom funcionamento e monitoramento dos dados obtidos durante execução do algoritmo foi projetada a

rede de comunicação, como ilustrado na Figura 45. Assim sendo, o monitoramento remoto se deu de forma segura em um raio de aproximadamente 60 metros, e poupou carga ao quadrotor por não necessitar de uma *companion board*.

Figura 45 – Sistema de comunicação entre os componentes desenvolvido para o trabalho.



7 Quadrotor desenvolvido

7.1 Aspectos construtivos

O quadrotor utilizado no projeto foi desenvolvido pela equipe de pesquisa e desenvolvimento em VAANTs no GRIn - Grupo de Robótica Inteligente da Universidade Federal de Juiz de Fora (Figura 46). É constituído em sua maioria por placas, tubos e folhas de fibra de carbono.

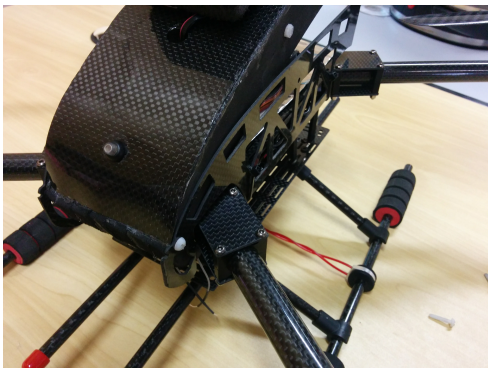
Figura 46 – Quadrotor desenvolvido utilizado para o trabalho.

Alguns dados considerados relevantes de construção estão detalhados na Tabela 7.1. O *drone* possui como detalhe especial um envoltório protetor feito em EVA, coberto por uma folha de fibra de carbono, que promove menor interferência externa no que diz respeito principalmente ao vento, proteção aos circuitos eletrônicos e um *design* mais elegante (Figura 47).

Diâmetro do braço (milímetros)	14
Altura (centímetros)	15,8
Comprimento do braço (centímetros)	17,5
Comprimento direção da asa (centímetros)	37
Comprimento direção do nariz (centímetros)	35
Motores	*RCTimer* 1000 KV
Peso (com bateria e câmera) (gramas)	1950
Dimensões de hélices (polegadas)	11 x 5
Tempo de voo (minutos)	15 a 20

Figura 47 – a) Capacete de EVA e fibra de carbono. b) Circuitos eletrônicos no interior do veículo. Destaque para folha de mumetal (dourada) alocada abaixo dos mesmos.

(a)

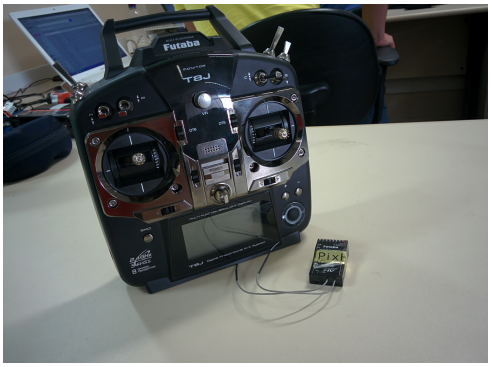


(b)

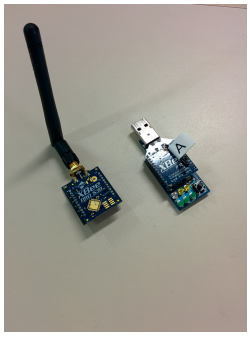
Além da placa controladora *Pixhawk*, já mencionada nesse trabalho, o quadrotor conta com um receptor de rádio e controle *FUTABA T8J*, módulo *xBee Pro* para comunicação com a GCS a longas distâncias e módulo de sensor GPS e magnetômetro externos *3DR* (Figura 48), para garantir menor interferência.

Figura 48 – a) Conjunto de transmissor e receptor de rádio. b) Módulos *Xbee* para comunicação a distância. c) Módulo de GPS e bússola externa.

(a)



(b)



(c)

Como fonte de potência do quadrotor é empregada uma bateria Lipo de 8400 mAh e 25 *Coulombs* capacidade de descarga. O conjunto de ESCs (*Electronic Speed Controller*) *QBrain* 4x1 escolhido suporta correntes máximas de 25 Amperes.

Os motores *RCTimer* de 1000 KV (Figura 49) do *drone* foram testados em laboratório e desenvolvem empuxo máximo de 950 grama-Força a uma corrente de pico de aproximadamente 18 Amperes, quando acoplados às hélices de fibra de carbono de 11x5 polegadas.

Figura 49 – Motor *RCTimer* 1000 KV.

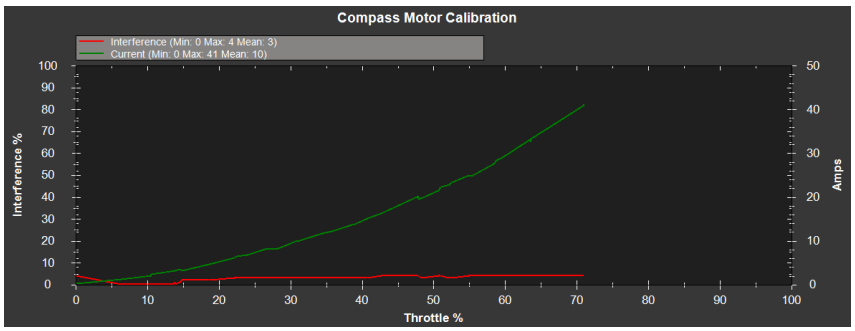
A boa prática encontrada por desenvolvedores em comunidades *online* para relacionar o peso da aeronave ao empuxo desenvolvido pelos motores se faz de que a carga a ser levantada deve ser no máximo metade do empuxo total fornecido pela propulsão. Para a aeronave, metade da propulsão total resulta em 1850 gramas-força. Dessa forma, levando em consideração o peso na Tabela 7.1, o quadrotor paira no ar com 52% da propulsão total, o que é considerado satisfatório.

7.2 Redução de interferências

Um dos maiores problemas encontrados para obtenção de resultado robusto no que diz respeito à autonomia do quadrotor foi a interferência eletromagnética, principalmente sobre a bússola da placa controladora. Além da adição do módulo externo, já citado anteriormente, foi acoplada uma folha de *mu-metal* (material muito utilizado para realizar a blindagem contra campos magnéticos externos [40]) entre a região de potência (ESC e fios dos motores) e os circuitos eletrônicos.

O campo magnético criado ao redor dos motores no momento em que giram também é de grande influência sobre a bússola. Para compensar essa interferência, que geralmente apresenta característica de crescimento linear em relação à corrente (segundo comunidade *online* em [55]), foi traçada a porcentagem de variação do valor do campo magnético sobre o valor com motores parados em função da corrente requerida da bateria para seu funcionamento. A curva está ilustrada na Figura 50, e possui um valor máximo de 4%.

Figura 50 – Curva de interferência sobre o magnetômetro da placa controladora em função da corrente drenada da bateria (campo gerado pelos motores).



Da curva são retirados parâmetros que são inseridos no controlador para compensar a interferência sobre o campo magnético. É boa prática tentar manter a interferência em valores menores que 30% [55], logo a construção do *drone* foi considerada satisfatória nesse quesito.

7.3 Desempenho em voo autônomo

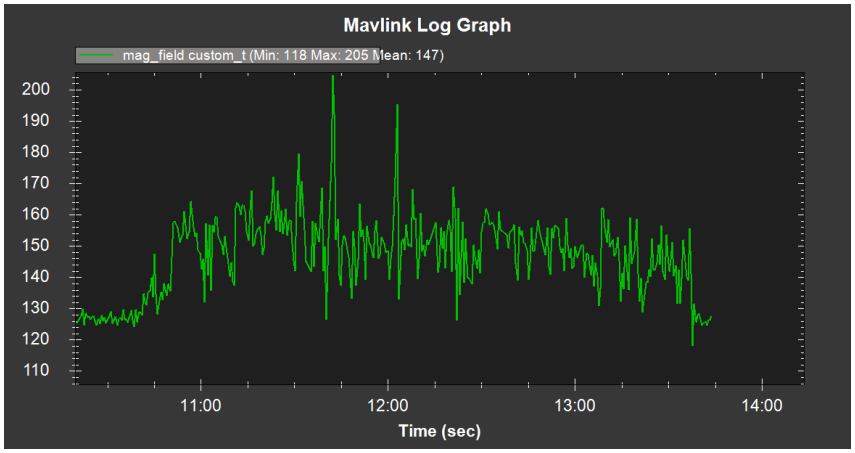
O teste em campo para a aeronave consistiu em voos autônomos que tinham como conclusão realizar a seguinte missão: alçar voo a partir do solo (*take-off*) até 10 metros de altitude, percorrer cerca de 40 metros a frente do local de partida, a 15 metros de altitude, e de lá retornar ao local de lançamento, a 10 metros de altitude. Dali, deve-se pousar (*land*). A Tabela 1 resume os comandos enviados.

Tabela 1 – Comandos para análise sobre missão autônoma.

	Comando	Altitude (metros)	X (*body frame*) (metros)	Y (*body frame*) (metros)
1	*take-off*	10	0	0
2	*waypoint*	15	40	0
3	*return to launch*	10	-40	0
4	*land*	10	0	0

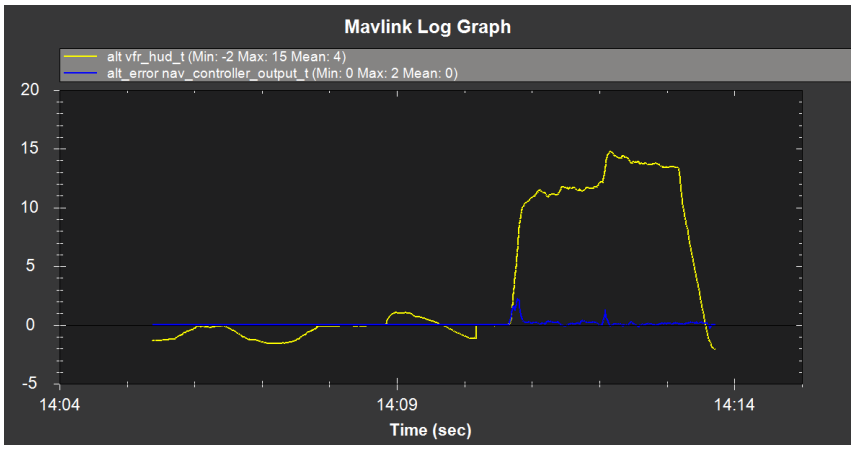
Foram colhidos LOGs de resultados da missão. O gráfico da Figura 51 ilustra como a medida do magnetômetro variou ao longo da missão. Salvo momentos atípicos de picos, pode-se notar que a variação em torno da média foi de no máximo 22,5%, o que é considerado aceitável para o bom funcionamento da aeronave.

Figura 51 – Curva de valores do campo magnético durante a missão autônoma. Variação considerada dentro dos padrões por desenvolvedores.



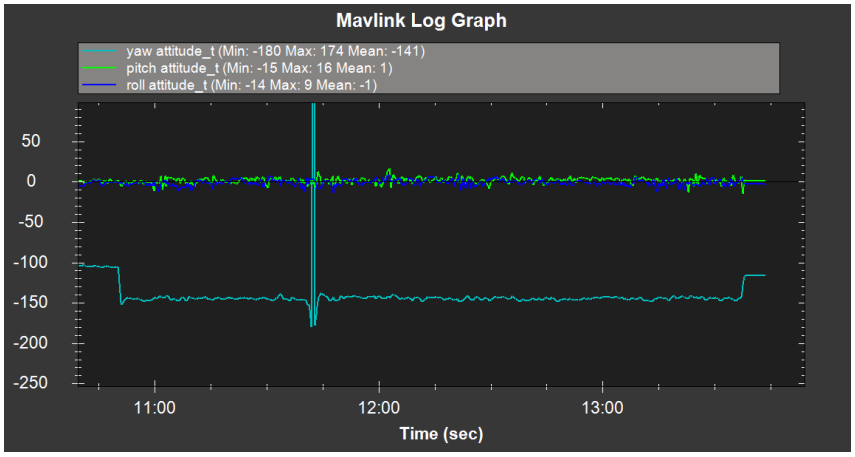
Na Figura 52 é possível ver a altitude em conjunto com o erro da mesma ao longo da missão, e notou-se também uma curva suave e aceitável.

Figura 52 – Gráfico de altitude e erro de altitude ao longo da missão. Resposta considerada satisfatória para desenvolvimento da missão.



Por fim, a Figura 53 demonstra os ângulos de *roll, pitch* e *yaw*. O movimento se deve à variação dos dois primeiros, e vale notar a constância do valor de *yaw* durante a missão, salvo pico no momento de chegada ao *waypoint*.

Figura 53 – Valores de atitude do *drone* durante missão.



8 Simulação com SITL - *Software in the Loop*

O conceito de *Software in the Loop (SITL)* consiste em transportar a parte de *hardware* do sistema, no caso a placa de controle de voo *Pixhawk*, para dentro do sistema de *software* de controle de forma simulada, a fim de testar comportamentos da mesma como se estivesse no mundo real [47]. A troca de mensagens com protocolo *Mavlink* ocorre exatamente da mesma forma que entre o *software* e a placa, e inclusive as respostas dinâmicas da aeronave tendem a ser representadas da forma mais fiel possível, mesmo não levando em consideração todas as variáveis aleatórias do ambiente, como ruídos eletromagnéticos e vento, ou falhas dos sensores.

No intuito de experimentar o código antes da implementação no quadrotor real, a biblioteca *dronekit* dispõe de sua variável *dronekit_sitl*, a qual faz o *download* do *firmware* mais atual desenvolvido pela *Ardupilot* e simula o comportamento do mesmo de dentro da própria máquina.

O algoritmo 1 descreve como o quadrotor foi controlado enquanto em missão autônoma e após o fim da mesma, no momento do pouso.

Algorithm 1: Pseudocódigo para algoritmo de controle de missão e pouso.

```
1  veiculo = inicia_conexao(porta serial);
2  aguarda_planejamento_de_missao(veiculo);
3  if veiculo.armado && veiculo.modo_automatico then
4      aguarda_fim_missao(veiculo);
5      if fim_missao then
6          camera = ligar_e_conectar(gopro);
7          modo_de_voo(veiculo, GUIDED);
8          buscar_landmark_e_corrigir_posicionamento(veiculo, camera);
9          autorizar_pouso(veiculo);
10 desarmar(veiculo);
11 desconectar(veiculo);
```

8.1 Conexão e monitoramento da missão

O primeiro passo para a simulação foi estabelecer comunicação com o quadrotor virtual, armá-lo e realizar o comando de *takeoff* através da função simple_takeoff() da biblioteca *dronekit*. Esse movimento simula exatamente o que seria feito no início do voo da aeronave real de formas tanto manual quanto autônoma.

No caso real a missão seria planejada antes do voo na GCS, porém na simulação a mesma é planejada a qualquer momento, mesmo com o quadrotor já na altura final após o *takeoff*. Portanto, o próximo passo é aguardar que alguma missão seja enviada à aeronave

por protocolo *Mavlink*, e o mesmo é verificado através do total de comandos presentes na mesma. O algoritmo permanece em aguardo enquanto não é confirmada a missão.

Após a notificação de missão existente o algoritmo avança para a etapa de monitoramento de comandos e posição. A leitura dos comandos presentes na aeronave é feita a cada iteração, de forma que se uma nova missão é planejada no momento do voo e enviada através da GCS o monitoramento se adapta automaticamente.

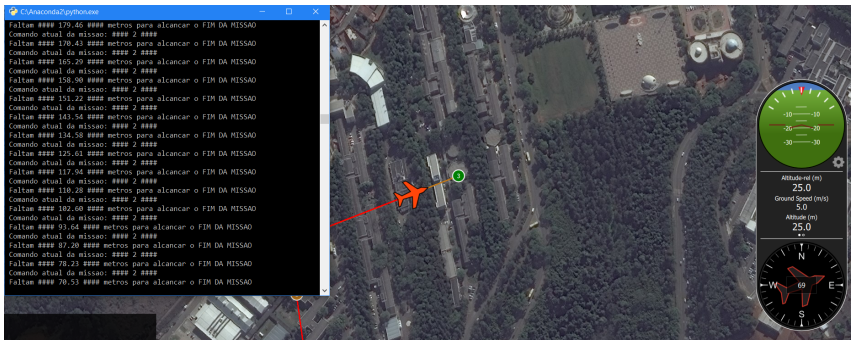
Se a missão apresenta n comandos, dentre eles geralmente *waypoints* nos quais podem ser realizados movimentos como variação de altitude e atitude do quadrotor, por exemplo, até o comando $n-1$ o monitoramento é feito somente pelo número do mesmo na fila de comandos. Ao detectar que se está navegando ao último comando da missão, o monitoramento começa a calcular a distância ao último *waypoint* através dos dados de GPS lidos por protocolo *Mavlink* através da bibliteca *dronekit*.

O cálculo da distância entre dois pontos se dá de forma simplificada, considerando o planeta como uma esfera perfeita de raio $R = 6.378.100$ metros e aproximando assim pela fórmula da Equação 8.1:

$$M = \frac{2\pi R}{360}\sqrt{d_{lat}^2 + d_{lon}^2} \tag{8.1}$$

Onde M é a distância real em metros, R o raio da Terra em metros, e d_{lat} e d_{lon} as diferenças de latitude e longitude entre os pontos, respectivamente. O resultado é considerado aceitável, como mostra a Figura 54, onde o último ponto está sendo alcançado.

Figura 54 – Fim da missão sendo monitorado pelas coordenadas de latitude e longitude do *waypoint* e veículo, com distâncias conforme equação 8.1.



8.2 Controle por visão do momento do pouso

A técnica de reconhecimento por visão foi descrita no Capítulo 5. Nessa Seção estará descrito como foi interpretada a resposta obtida por esse processo e encaminhada a ação ao quadrotor simulado.

Recapitulando, o reconhecimento da posição do *landmark* em relação à câmera (logo, ao quadrotor) retorna como dados a altitude e distâncias em relação ao *body frame*

do quadrotor.

O primeiro passo é ajustar o modo de voo do quadrotor para *GUIDED*, o qual como descrito na documentação da *Ardupilot* mantém o veículo em modo automático, porém no aguardo de comandos a serem enviados durante o voo. Isso é realizado pela função *VehicleMode()* da biblioteca *dronekit*.

Como descrito no algoritmo 2 em mais detalhes, o quadrotor se move em um *loop* enquanto sua posição relativa ao *landmark* no *body frame* não estiverem no raio de segurança mínimo definido, que representa estar aceitavelmente sobre o *landmark*. Esses são detalhes contidos na linha 8 do algoritmo 1.

Algorithm 2: Pseudocódigo de controle por visão

```
1  R_min = definir_raio_seguro();
2  while distancias_metros < R_min do
3      frame = camera.capturar();
4      imagem = segmentar_imagem(frame);
5      distancias_pixels = localizar_circulo(imagem);
6      distancias_metros = converter_metros(distancias_pixels, imagem);
7      manter_angulo_yaw(veiculo);
8      enviar_waypoint_veiculo(distancias_metros, veiculo);
9      aguardar_chegada(veiculo);
10 realizar_pouso(veiculo)
```

Logo, as distâncias nos eixos x e y do *body frame* são primeiro retornadas em *pixels* da imagem analisada, e convertidas em metros como descrito na Seção 5. Após isso são compactadas e enviadas ao veículo através da mensagem *set_position_target_local_ned*.

Para evitar deslizamentos do veículo na sua trajetória e tornar a busca mais eficiente, o quadrotor é forçado a se manter no mesmo ângulo de *yaw* com o qual chega ao final da missão durante toda a busca pelo posicionamento sobre o *landmark*. Sendo assim, conta com movimentos de *roll* e *pitch* para o controle de posição.

É importante notar que uma mensagem sobrescreve a anterior, e para isso considere os dois comandos na Tabela 2.

Tabela 2 – Comandos em série a enviar via *Mavlink*

	Eixo x	Eixo y
Comando 1	20 metros	10 metros
Comando 2	30 metros	0 metros

Se o Comando 2 for enviado imediatamente após o Comando 1, ou em algum momento durante a execução do último, o mesmo será abandonado e o Comando 2 será executado. Além disso, a taxa com que se captura as imagens pela câmera é muito mais

veloz que a dinâmica do quadrotor, o que faz necessário inserir um tempo livre de espera para o quadrotor executar o comando enviado, obtendo assim um voo mais suave. Isso é feito através da função *sleep()* por um dado período de tempo.

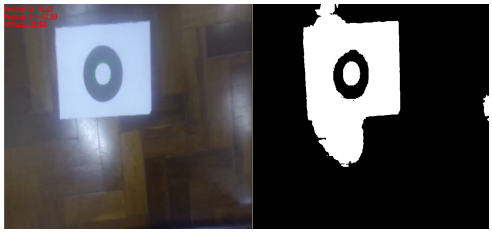
Na simulação foi utilizada uma miniatura do *landmark* como na Figura 55. Sendo assim as medidas de distâncias em X e Y do *body frame* do quadrotor são obtidas em centímetros, extraídas da imagem, mas são usadas como metros na simulação para se ter melhor visualização do deslocamento da aeronave.

Figura 55 – *Landmark* reconhecido e valor de altitude em centímetros.

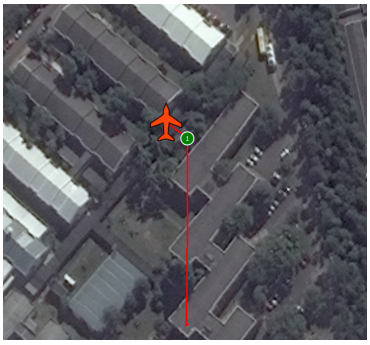
A Figura 56 mostra um resultado de uma simulação utilizando a GCS *QGroundControl*. Foi dada a missão de somente um *waypoint* localizado estritamente ao norte do local de decolagem. Após o fim da missão, o *landmark* foi detectado na região superior esquerda da imagem, ou seja, espera-se um movimento próximo ao sentido noroeste do quadrotor, que no caso estaria alinhado com o noroeste do planeta.

Figura 56 – Comando de movimentação enviado após o reconhecimento de *landmark* em simulação. a) *Landmark* reconhecido com distâncias ao centro da imagem já convertidas de *pixels* para centímetros. b) Movimento resultante no veículo simulado, conforme esperado.

(a)



(b)



O mesmo teste pode ser executado para testar todos os sentidos de movimentação do quadrotor.

Finalmente, ao avistar o *landmark* na região central da imagem, o algoritmo interpreta como aceitável o posicionamento e aciona o pouso por meio do modo de voo *LAND*, o qual reduz a altitude da aeronave em uma velocidade vertical ajustável e é enviado à aeronave pela mesma função *VehicleMode()*. Tendo a câmera capturado essa condição, a Figura 57 demonstra o pouso monitorado do quadrotor naquele ponto.

Figura 57 – Monitoramento da altitude no momento do pouso, após o *landmark* ser localizado dentro do raio mínimo aceito para pousar, conforme algoritmo 2

9 Estudo de caso

Uma vez aprovado o desempenho do quadrotor em voo autônomo e tendo robustez da troca de mensagens utilizando protocolo Mavlink, é possível utilizar o sistema de visão em campo para controlar o pouso ao fim de uma missão.

A altitude para pouso foi determinada entre 10 e 30 metros, considerada segura em locais abertos e seguindo a prática vista tanto de fabricantes quanto de comunidades da internet. Ao mesmo tempo, o piloto não deixa de enxergar o quadrotor em caso de alguma falha que necessite retomada do controle manual.

9.1 Análise do sistema de visão

O teste constituiu da captura de imagens de altitudes variando entre 15 e 25 metros, na região central da Faculdade de Engenharia da UFJF. Como pode ser visto nas Figuras 58, 59 e 60, o círculo foi detectado mesmo em local não tão aberto e isolado.

Figura 58 – Detecção de altitude e posições a voar com o quadrotor a aproximadamente vinte metros de altitude.

Figura 59 – Detecção de altitude e posições a voar com o quadrotor a também aproximadamente dezessete metros de altitude e com visada diferente do *landmark*.

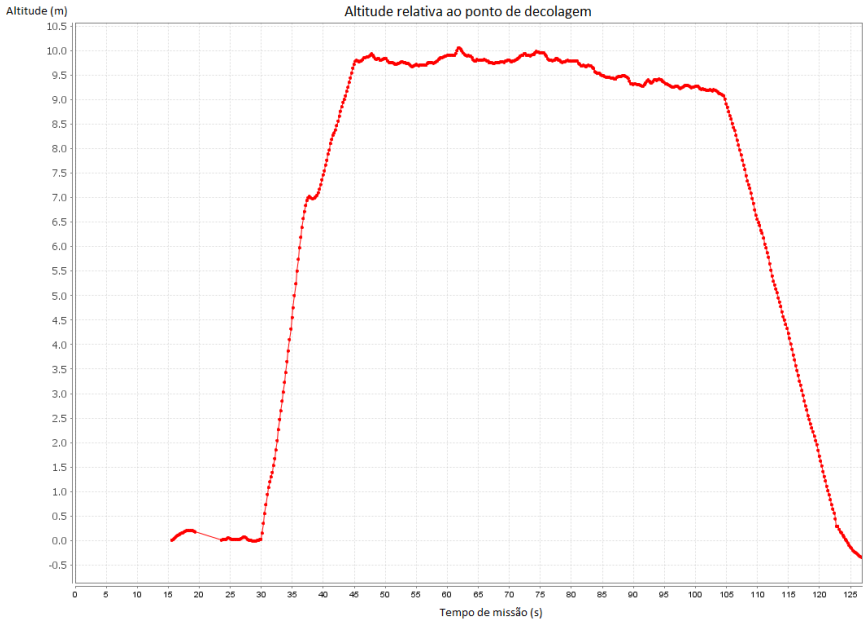
Figura 60 – Detecção de altitude e posições a voar com o quadrotor a aproximadamente quinze metros de altitude, já em um momento de pouso sobre o *landmark*.

Os valores obtidos para altitude através da câmera foram considerados satisfatórios em relação aos valores medidos pela placa controladora, adicionando informação e precisão ao pouso. Para as distâncias a serem percorridas em X e Y do *body frame* do quadrotor a mesma conclusão é válida, e a imprecisão será compensada pelo conjunto de comandos sequenciais até o momento do pouso.

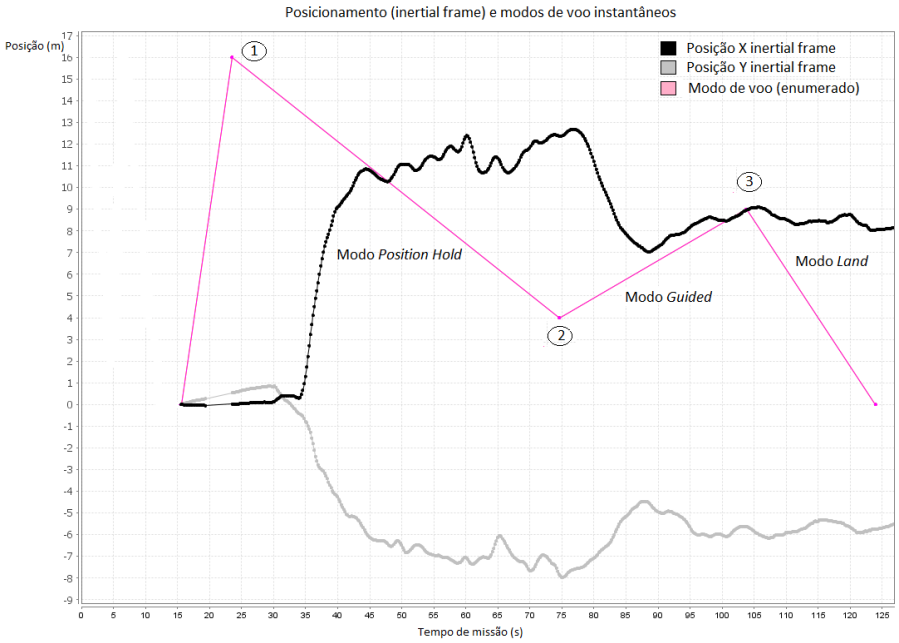
9.2 Resultados obtidos em campo

Para a comprovação do funcionamento das técnicas propostas em conjunto vários testes foram realizados. Em termos gerais, o algoritmo provou funcionar e garantir o pouso sobre o *landmark*.

Foi definido um raio de segurança de 30 centímetros em torno do centro do *ladmark*. Ao chegar no ponto final de missão, próximo do marcador, o quadrotor se movimentou como foi indicado pelo algoritmo de controle por visão. A Figura 9.2 contêm a altitude mantida pela aeronave enquanto buscava o posicionamento correto, que se manteve satisfatoriamente em torno de 10 metros.



As posições horizontais segundo o *inertial frame* estão representadas na Figura 9.2, juntamente com o modo de voo. O último é enumerado no código de controle do *drone*, de forma que o valor 16 indica modo *Position Hold*, onde a aeronave é levada com controle de altitude e posição por GPS e barômetro até um dado ponto (entre pontos 1 e 2), quando o modo assume valor 4 (*Guided*). Conforme visto no algoritmo de controle 1, a partir daí a aeronave segue comandos vindos dos dados de visão (entre pontos 2 e 3). Por fim, quando o modo de voo assume valor 9 (modo de voo *Land*), o quadrotor está seguramente sobre o *landmark* e executa a descida (ponto 3 em diante).



Por fim, o ângulo de *yaw* está representado na Figura 9.2. O mesmo deveria se manter constante durante a busca pelo ponto, e seu valor se encontra variando em uma escala de 10 graus, completamente aceitável pelo erro inerente do magnetômetro da placa controladora.

10 Conclusão

O trabalho foi concluído com êxito no que diz respeito à simulação sobre o algoritmo de comando, sendo o monitoramento da missão autônoma realizado seguramente assim como o momento do pouso.

A aeronave construída mostrou-se bastante segura, eficiente, e de fácil controlabilidade após diversos voos manuais e autônomos. No momento do pouso foram respeitados de forma satisfatória os comandos vindos do algoritmo de controle por visão computacional.

A variação do raio aceitável para pouso mostrou a distância de 30 centímetros do centro do *landmark* ao centro da imagem captada poder ser considerada como mínima, pois pequenos deslizamentos durante a busca enquanto o quadrotor paira no ar podem prejudicar o sistema de visão e fazê-lo não permitir o pouso, mesmo que o mesmo seja factível na realidade.

Trabalhos futuros podem envolver tanto novas arquiteturas de sistema quanto técnicas envolvendo a visão computacional. Mesmo que eficiente e satisfatória, a técnica de visão pode ser substituída por um algoritmo de reconhecimento de objetos através de treinamento por banco de dados, o que traria ainda mais segurança, porém ao custo de processamento mais carregado.

A inserção de uma *companion board* minimizaria problemas de comunicação entre os componentes do sistema, por substituir a necessidade do *laptop* a distância. Isso aumentaria o peso da aeronave e exigência dos motores, porém automatizaria o processo sem a necessidade do computador ligado com operador.

Por fim, a técnica confia na precisão do movimento de descida da aeronave, que mostrou-se satisfatório, porém não persegue ao longo desse movimento a posição em relação ao *landmark*, que pode vir a mudar de posição. Isso pode vir a ser implementado no algoritmo de controle por visão computacional.

Tendo posse do *hardware* e *software* utilizados nesse trabalho, o controle por visão pode ser estendido para outras aplicações, como o monitoramento de linhas de transmissão, transformadores e isoladores de potência, assim como outras que envolvam reconhecimento de objetos e padrões.

REFERÊNCIAS

[1] ALVES, Ana Sophia Cavalcanti. Estudo e Aplicação de Técnicas de Controle Embarcadas para Estabilização de Voo de Quadricopteros. Tese (Doutorado), Universidade Federal de Juiz de Fora, UFJF, 2012.

[2] AMORIM, Lúcio A. et al. Estimação de posição e atitude de um veículo aéreo não tripulado baseada em GPS, IMU e dados visuais.

[3] AUSTIN, Reg. Unmanned aircraft systems UAVs design, development and deployment. Chichester: Wiley, 2010. Disponível em: <http://public.eblib.com/choice/publicfullrecord.aspx?p=514439>. Acesso em: 14 novembro 2016. .9780470664803 0470664800 9781119964261 1119964261.

[4] BAILEY, Donald G. A new approach to lens distortion correction. Proceedings Image and Vision Computing New Zealand 2002, p. 59-64, 2002.

[5] D. H. Ballard and C. M. Brown, Computer vision. Prentice-Hall Englewood Cliffs (NJ), 1982, vol. 2.

[6] BEARD, Randal. Quadrotor Dynamics and Control Rev 0.1. 2008.

[7] BOURKE, Paul. Lens Correction and Distortion. April, 2002.

[8] BRANDAO, M. P. et al. UAV activities in Brazil. In: First Latin-American Unmanned Aerial Vehicle (UAV) Conference. 2007.

[9] CAMPOS, MF Montenegro; DE SOUZA COELHO, Lúcio. Autonomous dirigible navigation using visual tracking and pose estimation. In: Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on. IEEE, 1999. p. 2584-2589.

[10] CHAE, Heeseo et al. The IoT based automate landing system of a drone for the round-the-clock surveillance solution. In: 2015 IEEE International Conference on Advanced Intelligent Mechatronics (AIM). IEEE, 2015. p. 1575-1580.

[11] CHAUMETTE, François; MALIS, Ezio. 2 1/2 D visual servoing: a possible solution to improve image-based and position-based visual servoings. In: Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on. IEEE, 2000. p. 630-635.

[12] CHAVES, A. N. Proposta de modelo de Veículos Aéreos Não Tripulados (VANTs) cooperativos aplicados a operações de busca. Dissertação (Mestrado). Universidade de São Paulo, USP, 2013.

[13] COCCHIONI, Francesco; MANCINI, Adriano; LONGHI, Sauro. Autonomous navigation, landing and recharge of a quadrotor using artificial vision. In: Unmanned Aircraft Systems (ICUAS), 2014 International Conference on. IEEE, 2014. p. 418-429.

[14] CORKE, Peter. Robotics, vision and control: fundamental algorithms in MATLAB. Springer, 2011.

[15] COSTA, E. B. Algoritmos de Controle Aplicados à Estabilização do Voo de um Quadrotor. Dissertação (Mestrado). Universidade Federal de Juiz de Fora, UFJF, 2012.

[16] COSTA, Exuperry Barros et al. Controlador baseado em estabilidade de Lyapunov para estalibização do voo de um quadrotor.

[17] CUCCHIARA, Rita et al. A Hough transform-based method for radial lens distortion correction. In: Image Analysis and Processing, 2003. Proceedings. 12th International Conference on. IEEE, 2003. p. 182-187.

[18] DENTLER, Jan et al. A real-time model predictive position control with collision avoidance for commercial low-cost quadrotors. In: Control Applications (CCA), 2016 IEEE Conference on. IEEE, 2016. p. 519-525.

[19] DHANE, Pranali; KUTTY, Krishnan; BANGADKAR, Sachin. A generic non-linear method for fisheye correction. International Journal of Computer Applications, v. 51, n. 10, 2012.

[20] FAHIMI, Farbod; THAKUR, Karansingh. An alternative closed-loop vision-based control approach for Unmanned Aircraft Systems with application to a quadrotor. In: Unmanned Aircraft Systems (ICUAS), 2013 International Conference on. IEEE, 2013. p. 353-358.

[21] FLORES, Gerardo et al. A vision and GPS-based real-time trajectory planning for a MAV in unknown and low-sunlight environments. Journal of Intelligent & Robotic Systems, v. 74, n. 1-2, p. 59-67, 2014.

[22] GAUTAM, Alvika; SUJIT, P. B.; SARIPALLI, Srikanth. A survey of autonomous landing techniques for UAVs. In: Unmanned Aircraft Systems (ICUAS), 2014 International Conference on. IEEE, 2014. p. 1210-1218.

[23] GUO, Pengyu et al. Airborne vision-aided landing navigation system for fixed-wing UAV. In: 2014 12th International Conference on Signal Processing (ICSP). IEEE, 2014. p. 1215-1220.

[24] GUPTE, Shweta; MOHANDAS, Paul Infant Teenu; CONRAD, James M. A survey of quadrotor unmanned aerial vehicles. In: Southeastcon, 2012 Proceedings of IEEE. IEEE, 2012. p. 1-6.

[25] HUI, Cheng et al. Autonomous takeoff, tracking and landing of a UAV on a moving UGV using onboard monocular vision. In: Control Conference (CCC), 2013 32nd Chinese. IEEE, 2013. p. 5895-5901.

[26] JIN, Shaogang et al. On-board vision autonomous landing techniques for quadrotor: A survey. In: Control Conference (CCC), 2016 35th Chinese. TCCT, 2016. p. 10284-10289.

[27] KARRAY, Hassen et al. Design and implementation of a degraded vision landing aid application on a multicore processor architecture for safety-critical application. In: 16th IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC 2013). IEEE, 2013. p. 1-8.

[28] KENDOUL, Farid; ZHENYU, Yu; NONAMI, Kenzo. Embedded autopilot for accurate waypoint navigation and trajectory tracking: Application to miniature rotorcraft uavs. In: Robotics and Automation, 2009. ICRA'09. IEEE International Conference on. IEEE, 2009. p. 2884-2890.

[29] KIM, JeongWoon et al. Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera. In: Unmanned Aircraft Systems (ICUAS), 2014 International Conference on. IEEE, 2014. p. 1243-1252.

[30] LAIACKER, Maximilian et al. Vision aided automatic landing system for fixed wing UAV. In: 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2013. p. 2971-2976.

[31] LEE, Daewon; RYAN, Tyler; KIM, H. Jin. Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing. In: Robotics and Automation (ICRA), 2012 IEEE International Conference on. IEEE, 2012. p. 971-976.

[32] MOHANRAJ RAJA, Vinyojita. Vision based landing for unmanned aerial vehicle. In: Aerospace Conference, 2011 IEEE. IEEE, 2011. p. 1-8.

[33] NETO, A. A. GERAÇÃO DE TRAJETÓRIAS PARA VEÍCULOS AÉREOS AUTÔNOMOS NÃO-TRIPULADOS. Dissertação (Mestrado) ? Universidade Federal de Minas Gerais ? UFMG, 2008.

[34] PATEL, Parth N. et al. Quadcopter for agricultural surveillance. Advance in Electronic and Electric Engineering, v. 3, n. 4, p. 427-432, 2013.

[35] ROCHA, João Carlos. Cor luz, cor pigmento e os sistemas RGB e CMY. Revista Belas Artes, 2011.

[36] SANFOURCHE, Martial et al. Environment mapping & interpretation by drone. In: 2015 Joint Urban Remote Sensing Event (JURSE). IEEE, 2015. p. 1-4.

[37] SANTOS, Milton CP et al. UAV obstacle avoidance using RGB-D system. In: Unmanned Aircraft Systems (ICUAS), 2015 International Conference on. IEEE, 2015. p. 312-319.

[38] dos SANTOS, M. F. Controle tolerante a falhas de um sistema de propulsão de Hexacópteros. Dissertação (Mestrado), Universidade Federal de Juiz de Fora, UFJF, 2014.

[39] SCHAUWECKER, Konstantin; ZELL, Andreas. On-board dual-stereo-vision for autonomous quadrotor navigation. In: Unmanned Aircraft Systems (ICUAS), 2013 International Conference on. IEEE, 2013. p. 333-342.

[40] SUMNER, T. J.; PENDLEBURY, J. M.; SMITH, KoF. Convectional magnetic shielding. Journal of Physics D: Applied Physics, v. 20, n. 9, p. 1095, 1987.

[41] WAHARTE, Sonia; TRIGONI, Niki. Supporting search and rescue operations with UAVs. In: Emerging Security Technologies (EST), 2010 International Conference on. IEEE, 2010. p. 142-147.

[42] WANG, Fei et al. Guidance, navigation and control of an unmanned helicopter for automatic cargo transportation. In: Control Conference (CCC), 2014 33rd Chinese. IEEE, 2014. p. 1013-1020.

[43] ZANOBINI, Andrea. A drone anti-collision system: Maintaining a fixed distance from a target during the flight. In: Electrotechnical Conference (MELECON), 2016 18th Mediterranean. IEEE, 2016. p. 1-6.

[44] *PYTHON COMMUNITY*. Disponível em: <https://www.python.org/about/>. Acesso em: 2 de nov. 2016.

[45] *AGROBOTIX Company*. Disponível em <http://agribotix.com/>. Acesso em: 2 de nov. 2016

[46] *CONTINUUM ANALYTICS*. *Anaconda overview*. Disponível em <https://www.continuum.io/>. Acesso em: 2 de nov. 2016.

[47] *3DR Python DRONEKIT*. Disponível em: <http://python.dronekit.io/>. Acesso em: 2 de nov. 2016.

[48] *QGROUND CONTROL. MAVLINK - Micro Air Vehicle Communication Protocol*. Disponível em: <http://qgroundcontrol.org/mavlink/start>. Acesso em: 2 de nov. 2016.

[49] *QGRUND CONTROL*. Disponível em: <http://qgroundcontrol.com/>. Acesso em: 2 de nov. 2016.

[50] *MICROSOFT VISUAL STUDIO*. Disponível em: <https://www.visualstudio.com/>. Acesso em: 2 de nov. 2016.

[51] *OPENCV*. Disponível: em <http://opencv.org/>. Acesso em: 2 de nov. 2016.

[52] *PX4 AUTOPILOT*. Disponível: em <https://pixhawk.org/>. Acesso em: 2 de nov. 2016.

[53] ROSEBROCK, Andrew. *Practical Python and Opencv + Case Studies*. Disponível em: <https://www.pyimagesearch.com/practical-python-opencv/>. Acesso em: 2 de nov. 2016.

[54] *DIYDRONES - The Leading Community for Personal UAVs*. Disponível em: <http://diydrones.com/>. Acesso em: 2 de nov. 2016.

[55] *COPTER Documantation. Advanced Compass Setup*. Disponível em: <http://ardupilot.org/copter/docs/common-compass-setup-advanced.html>. Acesso em: 2 de nov. 2016.

ARDUCOPTER COMMUNITY. Advanced Compass Setup. Disponível em: <http://ardupilot.org/copter/docs/common-compass-setup-advanced.html>. Acesso em: 2 de nov. 2016.