Departamento de Engenharia
Mecatrônica e de Sistemas Mecânicos

Escola Politécnica da
Universidade de São Paulo

Escola Politécnica da Universidade de São Paulo

Marcus Felipe Fracaroli Pavani — Nº USP: 8038330
Thiago Resende Kruppa Villani — Nº USP: 8037983

Trabalho Final

Sistema de Pouso Autônomo de Drones Assistido por Imagens

Orientador: Prof. Dr. Jun Okamoto Jr.

São Paulo, Brasil
2018

Resumo

Em geral, drones comerciais apresentam rotinas de pouso monitorado com baixos índices de precisão. Tendo em vista o crescente interesse na aplicação de drones em atividades de alta destreza, torna-se mandatório o desenvolvimento de rotinas que viabilizem aterrissagens autônomas mais precisas. Sendo assim, o trabalho aqui relatado, desenvolvido no Laboratório de Percepção Avançada (LPA) da Escola Politécnica da Universidade de São Paulo (POLI-USP) em parceria com o Hospital Israelita Albert Einstein, destina-se a aumentar a precisão de pouso de um drone DJI Matrice 600 por meio da implementação de um algorítmo de computação visual, utilizando a metodologia IBVS (Image-based vision servoing). A aeronave possui como sistema embarcado um Raspberry Pi Zero e foi utilizada como ferramenta de obtenção de imagens uma Raspberry Pi Camera. Testes em ambientes virtuais e em campo foram realizados de modo a comprovar o aumento obtido na precisão do pouso, de modo a garantir uma aterrissagem em uma área circular média de $0{,}81\ m^2$, em um teste de hipótese com 99 % de significância estatística.

Palavras-chave: Precisão de pouso em drones – Pouso assistido por imagens – IBVS – DJI Matrice 600 – Raspberry Pi Zero – Raspberry Pi Camera

"O sucesso é ir de fracasso em fracasso sem perder o entusiasmo."

– Winston Churchill

Agradecimentos

Primeiramente, gostaríamos de agradecer ao Hospital Israelita Albert Einstein por disponibilizar o drone utilizado nesse trabalho, possibilitando assim a expansão do conhecimento em uma área inovadora, com tecnologias pouco utilizadas no meio acadêmico.

Agradecemos também o Prof. Dr. Jun Okamoto Jr. e toda a equipe do Laboratório de Percepção Avançada da Escola Politécnica da Universidade de São Paulo pela oportunidade de trabalharmos nesse projeto.

O autor Marcus Pavani agradece a seus pais Celso e Nellie Pavani e a todos os seus amigos por terem revertido esse ano de 2018, que começou de uma forma bastante ruim, em um ano de superação e experiências maravilhosas. Agradecimentos especiais a Matheus Cripa, Gabriel Gruber, Victor Adloff, Enrico Roberto, Gabriela Fernandes, Maria Clara Montes, Paulo Andrade, Caroline Souza e o co-autor desse trabalho, Thiago Villani.

O autor Thiago Villani agracede a seu pai, sua mãe e sua companheira Camila, que, durante as trajetórias árduas do projeto, forneceram força, inspiração e carinho de formas únicas e inestimáveis. Em especial, agradecimentos ao irmão, Lucas, cujo sorriso leve e mente curiosa o inspiram continuamente.

Índice de Figuras

Figura 1 - DJI Matrice 600 em voo. Fonte: [12] 12
Figura 2 - Esquemático referente à estações de pouso recarregáveis. Fonte: [19] 16
Figura 3 - Exemplos de alvos utilizados para pouso assistido por imagens. 19
Figura 4 - DJI Matrice 600. Fonte: [12] 20
Figura 5 - Representação esquemática da operação do drone 21
Figura 6 - Representação do objetivo de melhora na precisão do pouso 22
Figura 7 - Framework para definição do processador embarcado. Fonte: [29] 27
Figura 8 - Conexões entre os hardwares externos e o DJI Matrice 600. Fonte: [30] 29
Figura 9 - (Esquerda) Camera Zenmuse Z3; (Direita) Raspberry Pi Camera. 30
Figura 10 - Representação da comunicação entre desenvolvedor e drone 33
Figura 11 - Teste em ambiente virtual 33
Figura 12 - Exemplo da comunicação com a ferramenta do PuTTY 34
Figura 13 - Representação do cubo de tolerância para deslocamentos do drone 38
Figura 14 - Deslocamento de 0,15 metros a partir do algoritmo de movimentação 38
Figura 15 - Movimentação na direção x com comando de $0{,}5\,m$ 39
Figura 16 - Movimentação na direção y com comando de $0{,}75\,m$ 39
Figura 17 - Movimentação na direção z com comando de $2{,}0\,m$ 40
Figura 18 - Alvo utilizado como guia para pouso do drone 40
Figura 19 - Fluxograma Simplificado do Algoritmo de Computação Visual 43
Figura 20 - Seleção da Imagem de brilho mínimo durante a captura 43
Figura 21 - Exemplo do frame após o pré-processamento 44
Figura 22 - Exemplo do resultado após deteccção de contronos do quadrado 45
Figura 23 - Resultado após o processo de homografia 46
Figura 24 - Resultado do processo de deteccção de triângulos 46
Figura 25 - Resultado do processo de cálculo da distância do alvo para centro do frame 47
Figura 26 - Imagem base utilizada no simulador próprio. Adaptada de [41]. 49
Figura 27 - Exemplo de imagem recebida pelo algoritmo de pouso assistido. 50
Figura 28 - Exemplo de imagem recebida pelo algoritmo de pouso assistido. 50
Figura 29 - Mapa dos pontos de teste salvos 51
Figura 30 - Posição relativa entre os pontos salvos com cotas 52
Figura 31 - Exemplo de falha na detecção do alvo devido a reflexo e sombra. 57
Figura 32 - Movimentação na direção x com comando de $0{,}41\,m$ 59

Figura 33 - Movimentação na direção y com comando de $-1{,}68\,m$. 60
Figura 34 - Movimentação na direção z com comando de $-2{,}39\,m$. 60
Figura 35 - Movimentação na direção x com comando de $0{,}60\,m$. 61
Figura 36 - Movimentação na direção y com comando de $0{,}8\,m$. 61
Figura 37 - Movimentação na direção z com comando de $-1{,}65\,m$. 61
Figura 38 - Movimentação na direção x com comando de $-0{,}15\,m$. 64
Figura 39 - Movimentação na direção y com comando de $-0{,}56\,m$. 65
Figura 40 - Movimentação na direção z com comando de $-2{,}12\,m$. 65

Índice de Tabelas

Tabela 1 - Especificações do drone DJI Matrice 600. Fonte: [12] 20

Tabela 2 - Especificações do Controlador A3. Fonte: [23] 25

Tabela 3 - Especificações do Raspberry Pi Zero. Fonte: [13] 26

Tabela 4 - Processadores embarcados analisados no trabalho de Hulens [29] 28

Tabela 5 - Especificações da Raspberry Pi Camera. Fonte: [32] 31

Tabela 6 - Testes para determinação dos parâmetros do alvo 41

Tabela 7 – Resultados obtidos nos testes em campo 56

Tabela 8 - Erros finais para uma movimentação de $0{,}41;\ -1{,}68;\ -2{,}39\ m$. 59

Tabela 9 - Erros finais para uma movimentação de $0{,}60;\ 0{,}8;\ -1{,}65\ m$. 60

Tabela 10 - Precisão dos voos por comando único. 63

Tabela 11 - Erros finais para um comando único (voo número 2). 64

Tabela 12 - Resultados gerais dos voos bem sucedidos 66

Tabela 13 - Teste de confiabilidade 66

Sumário

1. INTRODUÇÃO 11

2. TÉCNICAS DE POUSO DE PRECISÃO 15

2.1 Estações de Pouso 15

2.2 Documentação de Projeto 19

3. ANÁLISE DE HARDWARE 24

3.1 Definição do Hardware 25

3.2 Definição da Câmera 30

4. ANÁLISE DE SOFTWARE 32

4.1 Configuração de Software 32

4.2 Comunicação por Tópicos 35

4.3 Controle de Posição 36

4.4 Computação Visual 40

4.4.1 Captura da Imagem 42

4.4.2 Pré-Processamento da Imagem 44

4.4.3 Detecção do Quadrado 45

4.4.4 Detecção dos Triângulos 46

4.4.5 Cálculo da Posição do Alvo 47

4.4.6 Envio dos Comandos ao Drone 48

4.4.7 Perda do Alvo 48

5. METODOLOGIA 49

5.1 Testes em Simulador 49

5.2 Set-Up do Experimento 51

5.3 Testes do Algoritmo de Pouso 52

5.4 Variações entre os testes em ambiente virtual e em campo 53

6. RESULTADOS E DISCUSSÕES 55

6.1 Coleta de Dados 55

6.2 Análise dos Pousos Falhos 57

6.2.1 Reflexo e Sombras 57

6.2.2 Distanciamento Contínuo 58

6.3 Análise de Pousos com Precisão Insatisfatória 62

6.3.1 Implantação de Pousos por Comando Único 63

6.4 Análise de Pousos com Sucesso 65

6.4.1 Validação Estatística dos Resultados 65

6.4.2 Oportunidades de Avanços nos Experimentos 66

6.5 Considerações Sobre os Experimentos 68

7. CONCLUSÃO 69

8. BIBLIOGRAFIA 73

1. Introdução

Em análise das tendências de aplicações dos drones, nota-se um crescente na busca por técnicas que garantam uma precisão maior no momento do pouso de uma aeronave não-tripulada, seja em locais de pouso fixos ou plataformas móveis [1]. Uma atividade que visa capitalizar no desenvolvimento de tais técnicas é a de transporte de cargas.

Grandes empresas como a Amazon já sinalizaram serviços de entrega operados por drones [2], outros trabalhos acadêmicos trazem discussões sobre dificuldades subsequentes do uso de drones para transporte de produtos como a criação de estações de pouso para recarga [3], ou a discussão de otimização de malhas híbridas em que drones e caminhões atuam juntos para efetuar as entregas [4], [5].

Contudo mediante o baixo tempo de autonomia dos drones atuais em situações de transporte de cargas, é imediata a necessidade de pontos de reabastecimento energético ou troca da mercadoria para um drone carregado, e neste momento é essencial que o pouso em tais estações seja feito de forma eficaz e sem comprometer a performance da operação.

Discutiremos, no trabalho aqui presente, algumas das soluções encontradas na literatura, desde as que buscam estratégias mais robustas como o uso de sonares [6] ou sistemas de câmera no local do pouso para cálculo da trajetória do drone [7], até outros trabalhos que discutem soluções com base em assistência de imagens e rotinas de computação visual para guiar o drone durante o pouso, também chamado de IBVS (Image-Based Vision Servoing) [8], [9], [10] e [11].

O trabalho aqui descrito visa atuar neste segmento de técnicas de pouso envolvendo auxílio por computço visual e coloca-se como uma contribuição original ao meio ao propor tal implementação em um DJI Matrice 600 [12], um drone de alta performance, com auxílio de um microprocessador Raspberry Pi Zero [13] embarcado.

O trabalho por nós desenvolvido trata-se de uma continuação de um projeto realizado no Laboratório de Percepção Avançada (LPA) da Escola Politécnica da Universidade de São Paulo (POLI-USP) em conjunto com o Hospital Albert Einstein no segundo semestre de 2017.

O escopo inicial do projeto consistia na criação de um sistema de transporte de cargas executado por um drone autônomo, representado na Figura 1, entre dois pontos, denominados Base A e B. Nessas localidades, operadores carregariam e descarregariam produtos no drone. Com o auxílio de um servidor implementado em Java [14], o catalogamento dos produtos e das viagens se daria em um software de banco de dados chamado MySQL [15].

Uma vez carregado o drone, por exemplo, na Base A, o mesmo percorreria uma trajetória definida por pontos manualmente pré-programados, até a Base B. Neste momento, dar-se-ia início à manobra de pouso monitorado, manobra essa que é programada de fábrica pela própria DJI. Contudo, foi observado em testes que tal manobra de pouso monitorado fornecia uma precisão abaixo do aceitável para o projeto: o local de pouso do drone se dava dentro de uma área de aproximadamente $9\ m^2$.

Figura 1 - DJI Matrice 600 em voo. Fonte: [12]

Tendo em vista a necessidade de um pouso mais preciso por motivos de eficiência operacional e segurança para os operadores humanos, o projeto atual destina-se a garantir uma precisão maior mediante a implementação de um pouso assistido por imagens. Na nova arquitetura aqui desenvolvida, o drone extrai imagens do solo por meio de uma câmera e tal imagem fornece um insumo para uma recalibração da posição do drone. Mediante um sucessivo conjunto de comandos e verificação do alinhamento

do mesmo com um certo alvo buscamos atingir um pouso com uma maior precisão, que se limite à uma área de aproximadamente de $1\ m^2$.

O projeto aqui descrito busca, portanto, comprovar a hipótese de que a implementação de uma rotina de pouso assistida por imagens garante uma melhora na precisão do pouso em até 9 vezes. De forma paralela, buscamos garantir comprovações secundárias com relação a integração de processadores embarcados de fácil programação e baixo custo com controladores complexos de drones comerciais, de modo a demonstrar a possibilidade de expansão da performance de pouso de drones a partir de hardwares externos.

As conjecturas acima descritas foram abordadas a partir dos seguintes desenvolvimentos:

- Implementação do hardware para obtenção de imagens e integração do mesmo com o processador embarcado existente, Raspberry Pi Zero.
- Criação de uma rotina para obtenção das imagens e tratamentos das mesmas por meio de um algoritmo de computação visual, utilizando bibliotecas de computação visual existentes, conhecidas como OpenCV [16], para detecção de um alvo pré-definido.
- Definição de uma correlação entre o resultado da detecção do alvo com o deslocamento necessário no plano horizontal dada uma altura arbitrária.
- Implementação de uma rotina para movimentação do drone a partir de comandos sem interface humana.
- Confronto dos resultados dos testes em ambiente de simulador, usando o software DJI Assistant 2, com testes no mundo real a fim de validar a melhora na precisão.

A continuação deste documento obedece a seguinte estrutura: O próximo capítulo trará uma contextualização do projeto aqui desenvolvido com a literatura acadêmica no que concerne as estratégias utilizadas para a melhora na precisão do pouso de aeronaves não-tripuladas, e detalharemos ainda os requerimentos do projeto

aqui desenvolvido. O capítulo 3 traz a análise do hardware utilizado e apresenta os conceitos envolvidos na tomada de decisão referente ao microprocessador embarcado. O capítulo 4 traz uma descrição dos softwares utilizados e na estrutura de comunicação do código desenvolvido com o drone, exemplificando os programas auxiliares utilizados e, por fim, trazendo um detalhamento do algoritmo de computação visual desenvolvido, o qual assegura a detecção e rastreamento do alvo pré-definido. O capítulo 5 visa elucidar a metodologia utilizada em nossos experimentos, cujos resultados encontrados são descritos no capítulo 6, onde, enfim, validamos as hipóteses definidas no início do trabalho. O capítulo 7 traz as conclusões obtidas com o desenvolvimento do projeto bem como os próximos passos para trabalhos futuros.

2. Técnicas de Pouso de Precisão

Neste capítulo serão expostas diferentes técnicas para pousos de precisão de aeronaves não tripuladas de modo a sumarizar ao leitor os principais avanços encontrados na literatura associados às estações de pouso nessas situações, bem como os algoritmos para tal utilizados. Ao final deste capítulo, anunciamos a nossa solução proposta para o problema aqui descrito.

2.1 Estações de Pouso

Definimos uma estação de pouso como o local em que uma dada aeronave deve praticar pelo menos uma das manobras de aterrisagem e decolagem, as estações podem compor os limites da trajetória total do drone ou podem compor pontos intermediários no percurso.

Apesar dos avanços no desenvolvimento de baterias de alto rendimento e baixo peso, um dos principais gargalos para a disseminação de drones como meio de transporte de cargas continua a ser a baixa durabilidade das baterias quando comparada com o rendimento de combustíveis fósseis para entregas realizadas com motos ou caminhões.

Ademais, o rendimento das baterias é colocado ainda mais em evidência quando a aeronave em questão deve carregar algo além do peso próprio, como o drone utilizado neste trabalho. Considerado um dos equipamentos mais refinados no segmento de drones com capacidade de transporte de produtos, o DJI Matrice 600 tem, com máxima carga útil, uma autonomia de apenas 16 min [12]. Um outro drone, o DJI Mavic Pro 2, é anunciado com 29 min de autonomia de voo. Contudo, isso contempla apenas o seu peso próprio de 907g [17]. Sendo assim, em casos de operações longas, estações de pouso são necessárias para garantir a recarga das baterias ou troca de drones para a continuidade da operação. É possível encontrar na literatura trabalhos que propõem tais pontos de recarga elétrica para drones.

Khonji [3] descreve em seu trabalho o design e implementação de um sistema de recarga intuitivo autônomo para drones elétricos operando em baterias. Como demonstração, ele descreve um sistema de recarga para um drone DJI Matrice 100, contendo um sistema embarcado Linux.

Tseng [18] realiza um estudo empírico para modelar o problema de performance das baterias de drones em diferentes cenários de voos e fornece um algoritmo para otimização de planos de voo e recarga das baterias.

Em 2016 a empresa Amazon publicou um esquemático para estações de recarga de drones a serem utilizados no seu sistema de entregas aéreas [2]. Tais locais, serviriam também como postes de iluminação pública, conforme mostra a Figura 2.

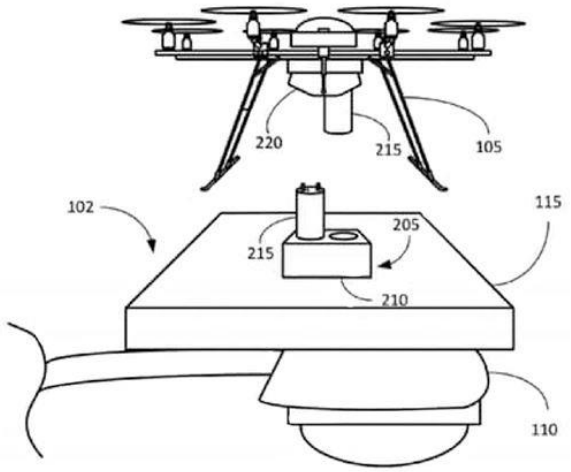

Figura 2 - Esquemático referente à estações de pouso recarregáveis. Fonte: [19]

Outro ponto importante sobre as estações de pouso está em sua detecção pelos drones e, consequentemente, na precisão do pouso nos locais desejados. Podemos encontrar na literatura propostas de solução mais complexas para o aumento da precisão do pouso de drones, as quais envisionam maior flexibilidade com relação ao ambiente do pouso.

Xu [20] propõe em seu trabalho um novo framework para execução de pousos autônomos de UAVs a bordo de navios sob quaisquer condições climáticas. Para tal, ele se baseia em objetos com emissão infravermelha na pista de pouso, associados a

receptores infravermelhos no drone. Tendo em vista que o navio é uma estação de pouso com potencial de mobilidade, os autores discorrem neste trabalho sobre estratégias de voo nas proximidades do alvo para otimizar a detecção do alvo guia.

Outros trabalhos buscam se distanciar da análise de imagens para guiar o drone até a estação desejada e se alicerçam em outras formas de orientar o drone ao meio. Papa [6] descreve em seu trabalho a criação de um mapa tridimensional do local de pouso do drone por meio de um modelo de sensor ultrassônico. Ondas de som refletidas produzidas pelo sonar são capturadas e analisadas de modo a construir um plano onde o pouso do UAV seria seguro e preciso. Nesta modelagem o mesmo sistema que assegura maior precisão no pouso também viabiliza a detecção de objetos durante o voo, além de configurar uma estratégia independente de imagens. É interessante apontar que a utilização de um sonar traz a dificuldade do controle de temperatura, uma vez que ela está diretamente associada com a velocidade do som naquele meio. Durante o trabalho, os autores apontam que a negligência na correção da temperatura trouxeram erros consideráveis na precisão do pouso.

Zhou [7] propõe em seu trabalho uma estrutura mais robusta que consiste em usar câmeras instaladas ao redor do local de pouso, as quais fotografam o drone quando ele está nas proximidades e calculam a trajetória ideal para que o UAV pouse. Resultados apresentados no trabalho mostram que esse sistema de videometria garantiu precisão de até 10 cm no momento do pouso.

Todavia, as estratégias relatadas previamente envolvem usos de hardwares de alto custo além de trazerem dificuldades de implementação devido ao número elevado de novas variáveis a serem consideradas. O trabalho aqui relatado destina-se a provar uma melhora na precisão do pouso de um hexacóptero por meio da presença de imagens-guia nas estações de pouso, também referidas nesse trabalho como "alvos", somadas a algoritmos eficientes de detecção e rastreamento das mesmas.

A utilização de computação visual para pousos de aeronaves configura uma solução barata e rápida com muitos exemplos podendo ser encontrados na literatura. Lange [8]

expõe, em seu trabalho, uma proposta de alvo para guiar o pouso da aeronave, bem como um sistema de movimentação com base nas imagens adquiridas. Segundo ele o pouso assistido por imagens gerou uma precisão de cerca de 98% no alvo.

Qiu [9] relata em seu trabalho um projeto similar dedicado à instalação de locais de pouso para drones operando na inspeção de linhas de transmissão de energia elétrica. Nesse trabalho, o sistema de processamento embarcado é um Raspberry Pi.

Saripalli [10] conduz um trabalho similar, utilizando uma Raspberry Pi Camera como hardware para captura de imagens. Battiato [11] propõe um sistema para pouso autônomo em bases móveis a partir de guias visuais de um alvo, utilizando o drone DJIS900, o processador embarcado Jetson TXI e a câmera Ocam Camera. Maiman [21] desenvolve no seu projeto uma otimização da precisão do drone ao combinar cálculos do GPS do drone DJI Matrice 100 e cruzar com as informações obtidas pela captura das imagens do alvo composto por um QR-Code, o processador embarcado neste projeto foi um Raspberry Pi.

É interessante destacar que os alvos nos trabalhos descritos acima traziam similaridades no que concerne a constituição padrões geométricos para fácil identificação pelo algoritmo de computação visual. Os autores em [8] utilizaram o alvo, exposto na região superior esquerda da Figura 3, composto por um hexágono e uma sequência de círculos concêntricos. Em [11], região superior direita da Figura 3, há uma opção por um quadrado contendo um círculo, que por sua vez contém um 'x' (xis), como forma de alvo para guiar o pouso da aeronave.

Outros trabalhos optam por regiões pré-definidas para instruir a aeronave durante o pouso, como o trabalho de Shang [22], em que os autores utilizam as passarelas de pouso de aviões, representadas na região inferior esquerda da Figura 3, como guia para pouso de drones. A utilização de estruturas pré-definidas requer elementos adicionais para ajudar na diferenciação do resto do ambiente quando a imagem está sendo analisada pelo processador embarcado. Nesse caso, os autores utilizaram lâmpadas para ajudar a aeronave a detectar o alvo.

Por fim observamos no trabalho [21] o uso de um QR Code, como mostrado na região inferior direita da Figura 3, como forma de alvo. O uso de QR Codes pode, mediante a identificação e leitura deles, trazer informações adicionais e indicar qual o local de pouso que o drone deve utilizar em seguida. Tal solução se torna mais atrativa em casos de maiores malhas logísticas de entrega com drones, em que a aeronave deve saber qual a estação de pouso a mesma se encontra.

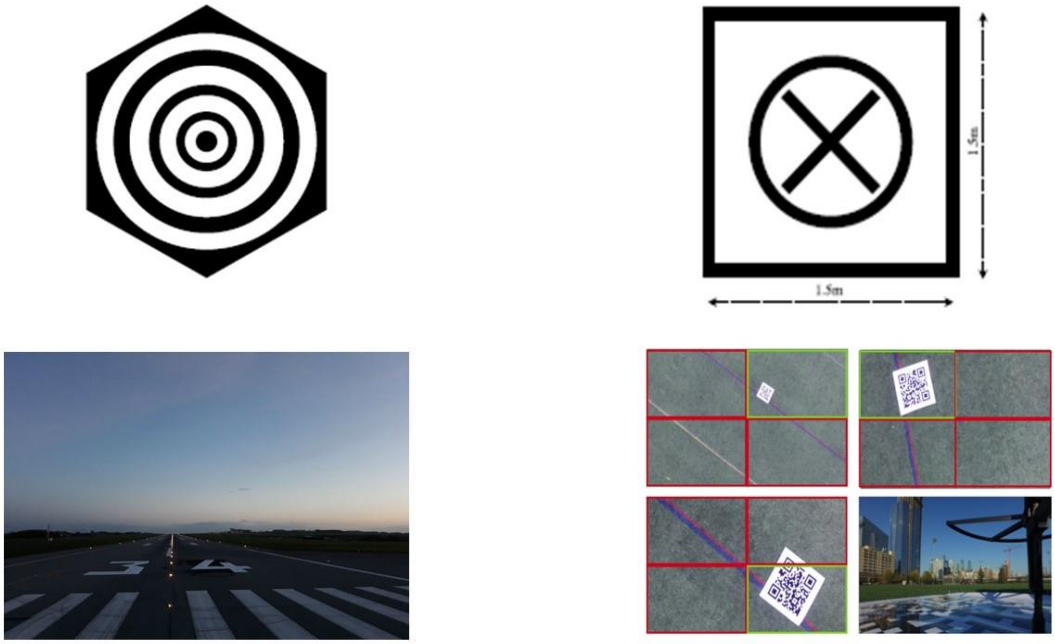

Figura 3 - Exemplos de alvos utilizados para pouso assistido por imagens.
Fontes: [8], [11], [22] e [21].

2.2 Documentação de Projeto

O projeto aqui desenvolvido trata-se de uma continuação de um projeto iniciado em 2017, em uma parceria entre o Hospital Albert Einstein e o Laboratório de Percepção Avançada de Escola Politécnica da Universidade de São Paulo. No momento de início desse trabalho, o drone a ser utilizado já havia sido determinado e o DJI Matrice 600, conforme mostrado na Figura 4, nos foi então disponibilizado para a realização dos testes.

Os pré-requisitos expostos pelo hospital faziam requerimento de uma aeronave capaz de suportar cargas elevadas, uma vez que o objetivo final da aplicação é usar o drone para transporte autônomo de carga viva como órgãos ou amostras fisiológicas.

O DJI Matrice 600 consiste em um hexacóptero com capacidade de carga de até 6 quilogramas, o que, nesse segmento, corresponde a uma carga bastante elevada. A Tabela 1 traz algumas das especificações da aeronave.

Figura 4 - DJI Matrice 600. Fonte: [12]

Tabela 1 - Especificações do drone DJI Matrice 600. Fonte: [12]

Diagonal Wheelbase	1133 mm
Aircraft Dimensions	1668 mm x 1518 mm x 759 mm (Propellers, frame arms and GPS mount unfolded) 640 mm x 582 mm x 623 mm (Frame arms and GPS mount folded)
Package Dimensions	620 mm x 320 mm x 505 mm
Weight (with six TB47S batteries)	9.1 kg
Max Takeoff Weight	15.1 kg
Motor Model	DJI 6010
Propeller Model	DJI 2170
Hovering Accuracy (P-Mode, with GPS)	Vertical: ±0.5 m, Horizontal: ±1.5 m

Max Angular Velocity	Pitch: 300°/s, Yaw: 150°/s
Max Pitch Angle	25°
Max Speed of Ascent	5 m/s
Max Speed of Descent	3 m/s
Max Wind Resistance	8 m/s
Max Flight Altitude above Sea Level	2500 m
Max Speed	18 m/s (No wind)
Hovering Time	No payload: 35 min 6 kg payload: 16 min
Flight Controller Model	A3
Remote Controller Max Transmission Distance	FCC Compliant: 3.1 miles (5 km) CE Compliant: 2.1 miles (3.5 km)

A operação do drone consiste em receber a carga em um ponto de carregamento, também chamado de Base A. O carregamento de materiais é acompanhado por uma interface em Java em que o usuário insere o código do produto sendo transportado.

Finalizada essa etapa, o operador sinaliza para o drone a viabilidade do voo. O drone viaja, então, por vários pontos pré-programados entre a estação de carregamento A e a estação de pouso B. Tais pontos são gravados manualmente utilizando o controle remoto da aeronave em um voo manualmente guiado. A Figura 5 traz uma representação esquemática desta operação.

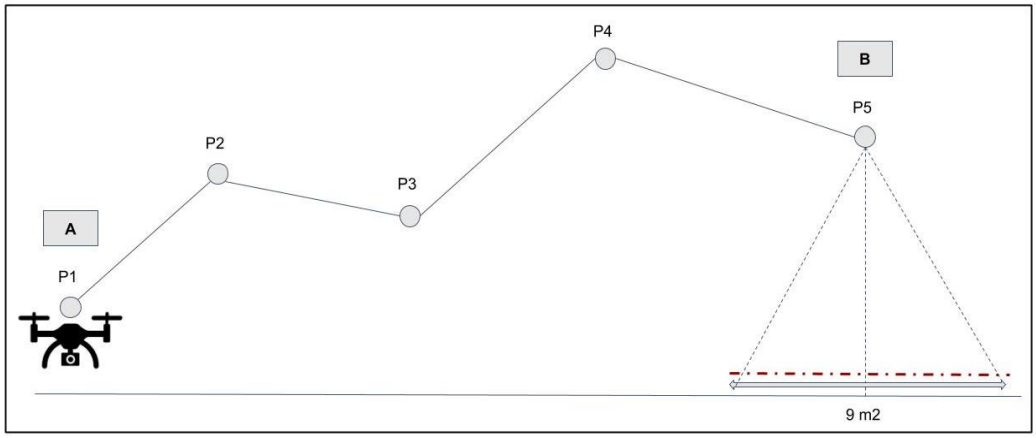

Figura 5 - Representação esquemática da operação do drone

Por fim o drone inicia sua rotina de pouso automático por GPS, o qual forneceu, nos testes préviamente realizados, uma precisão de $9\ m^2$. Após o pouso, o operador da Base B retira o produto e cadastra o recebimento na interface em Java e sinaliza o término da operação ao sistema.

O nosso trabalho destina-se a melhorar a precisão do pouso do drone por meio de computaçao visual. Ou seja, através do uso de uma câmera e de um alvo pré-definido, um sistema de detecção do alvo foi implantado para reduzir a área de pouso para $1\ m^2$. Medições da precisão do pouso em testes em ambiente virtual e real foram realizadas e a melhora na precisão do pouso foi calculada estatisticamente. A Figura 6 traz uma representação desse objetivo.

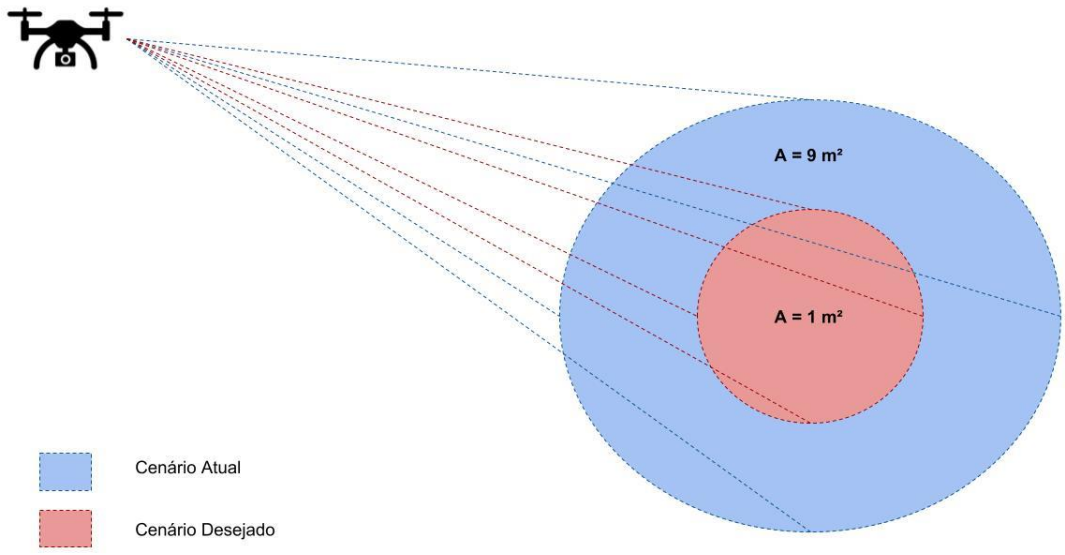

Figura 6 - Representação do objetivo de melhora na precisão do pouso

O projeto traz ainda, como ganho secundário, a demonstração da viabilidade de associar drones de última geração com processadores embarcados de fácil manuseio e baixo custo, tais como o Raspberry Pi Zero, de forma a expandir o horizonte de aplicações da aeronave em questão.

Sendo assim o projeto resume-se a comprovar duas hipóteses, enumeradas a seguir:

I. A utilização de um algoritmo de detecção e rastreamento de imagens auxilia na precisão do voo de uma aeronave autônoma. Neste caso, buscamos a precisão de até $1\ m^2$ para a área de pouso do drone.

II. É possível a integração de processadores embarcados de baixo custo para viabilizar a expansão dos horizontes de atuação do drone. Neste caso exemplificaremos essa possibilidade ao embarcar um Raspberry Pi Zero no drone DJI Matrice 600.

3. Análise de Hardware

Um crescimento do horizonte de aplicações para drones pode ser observado nos últimos anos, variando desde ferramenta de auxílio em desastres naturais até como uma solução para o problema logístico de empresas de entrega. Contudo, todas essas aplicações trazem consigo a similaridade de incentivar o desenvolvimento de algoritmos que permitam ao drone fazer tomadas de decisões mais precisas e rápidas e, portanto, atingir um estado de autonomia cada vez maior.

O desafio por traz do desenvolvimento de tais algoritmos é uma derivação da dificuldade na escolha do hardware ideal para viabilizar a autonomia no drone. Conforme a complexidade das decisões se eleva, é necessário um hardware que seja compatível com os softwares a serem utilizados, e que viabilize o processamento das informações em um tempo hábil sem comprometer a operação.

Ao mesmo tempo, UAVs tem um trade-off inato no que concerne o controle com seu peso e poder de carga. Sendo assim, drones com capacidades de carregar objetos pesados ganham uma inércia adicional, a qual compromete a sua controlabilidade e resulta em movimentos de menor velocidade e destreza, enquanto drones de menor porte atingem velocidade maiores e tem um espaço de trabalho para atuação mais flexível.

O limite entre esses dois tipos de UAVs é uma função do desenvolvimento continuo de novas formas de controlar atuadores e gerir a energia necessária para o movimento. Contudo, a escolha do hardware para viabilizar a aplicação dos drones deve acompanhar essa evolução para garantir coerência com os limites físicos de operação da aeronave.

Podemos segmentar os hardwares utilizados em um n-coptero em dois grupos principais: O hardware de controle e hardware de interação externa. O primeiro grupo corresponde ao hardware dedicado a garantir o controle do robô durante o voo, e, portanto, assegurar o acionamento dos motores da forma correta para desempenho da

manobra desejada. O segundo grupo corresponde ao hardware dedicado a garantir a interação do UAV com o meio que rodeia, incluindo sensores e câmeras.

A segmentação nesses dois grupos não significa que sejam necessariamente hardwares separados, podemos ter o mesmo processador acoplado aos elementos que permitem tanto o controle da aeronave quanto sua interação com o meio.

3.1 Definição do Hardware

No caso de drones comerciais, o hardware de controle é geralmente uma caixa-preta separada, da qual o usuário tem controle por meio de um protocolo de comunicação especifico para cada aeronave. No caso do drone utilizado nesse projeto, por exemplo, o hardware de controle do drone é denominado A3 e possui um canal de comunicação serial com o mundo externo. Na Tabela 2, temos as especificações deste controlador.

Tabela 2 - Especificações do Controlador A3. Fonte: [23]

Maximum Wing Resistance	$<$ 10m/s
Max Yaw Angular Velocity	150 deg/s
Mas Tilt Angle	35°
Ascent	5m/s
Descent	4m/s
SDK Port	API/CAN2
Recommended Battery	3S to 12S LiPo
Operating Temperature	-10°C to +45°C

Nos casos de distinta separação do hardware de controle, devido a origem comercial do drone, o hardware de interação externa é chamado de processamento embarcado e tem que ser desenvolvido separadamente de forma a obedecer a dois requisitos: assegurar uma comunicação eficaz com o hardware de controle e garantir o processamento das informações obtidas do mundo externo seja por câmeras e/ou sensores. Em geral, os dados obtidos são convertidos por meio de um algoritmo para

informações que possam ser enviadas para o hardware de controle, e assim, a aeronave possa executar a manobra.

Nos casos em que processamento embarcado é empregado, os algoritmos são, em geral, de menor complexidade. McGee [24], por exemplo, utiliza, em seu trabalho, segmentação por cores com um processador Pentium III para detecção e evasão de objetos no céu. Meier [25] utiliza um detector de marcadores para viabilizar o seguimento de trajetórias pré-definidas pelo UAV, com um processador Cortex-A9.

De forma mais recente, existem trabalhos que atingem os requerimentos de performance em tempo real com algoritmos mais complexos em plataformas embarcadas em drone. Contudo, os algoritmos foram especialmente adaptados para máxima performance nos hardwares utilizados em cada projeto [26], [27] e [28].

O drone utilizado neste trabalho, o DJI Matrice 600, havia sido parte de um projeto iniciado no ano de 2017 do laboratório de percepção avançada (LPA) da Escola Politécnica da Universidade de São Paulo (POLI-USP), que visava a prova de conceito sobre a utilização do hexacóptero em um sistema de entregas entre duas bases de carga e descarga. Durante a realização de tal trabalho a equipe responsável decidiu por utilizar como plataforma embarcada o Raspberry Pi Zero. A Tabela 3 traz as especificações desse microprocessador.

Tabela 3 - Especificações do Raspberry Pi Zero. Fonte: [13]

CPU	BCM2835 1GHz ARM11
RAM Memory	512MB
OS	Raspian Jessie - Linux
Dimensions	65 mm x 30 mm x 5mm
GPIO	40

Para os propósitos do trabalho desenvolvido em 2017 o Raspberry Pi atendia com excelência os requerimentos, uma vez que seu sistema operacional suportava o SDK (Software Development Kit) do Matrice 600 e conseguia realizar a troca de dados

com o controlador A3, principalmente informando o momento de decolagem, posições de voo e aterrisagem, sem falhas ou delay. Contudo, para o desenvolvimento do trabalho aqui relatado, o processador embarcado deveria também ser capaz de garantir a execução do algoritmo de computação visual de forma a assegurar a aterrisagem do drone no intervalo de $1\ m^2$ conforme as especificações do projeto.

No trabalho de Hulens [29] é exposto um framework, representado na Figura 7, para a definição do processador embarcado em UAVs mais adequado de acordo com a complexidade do algoritmo utilizado.

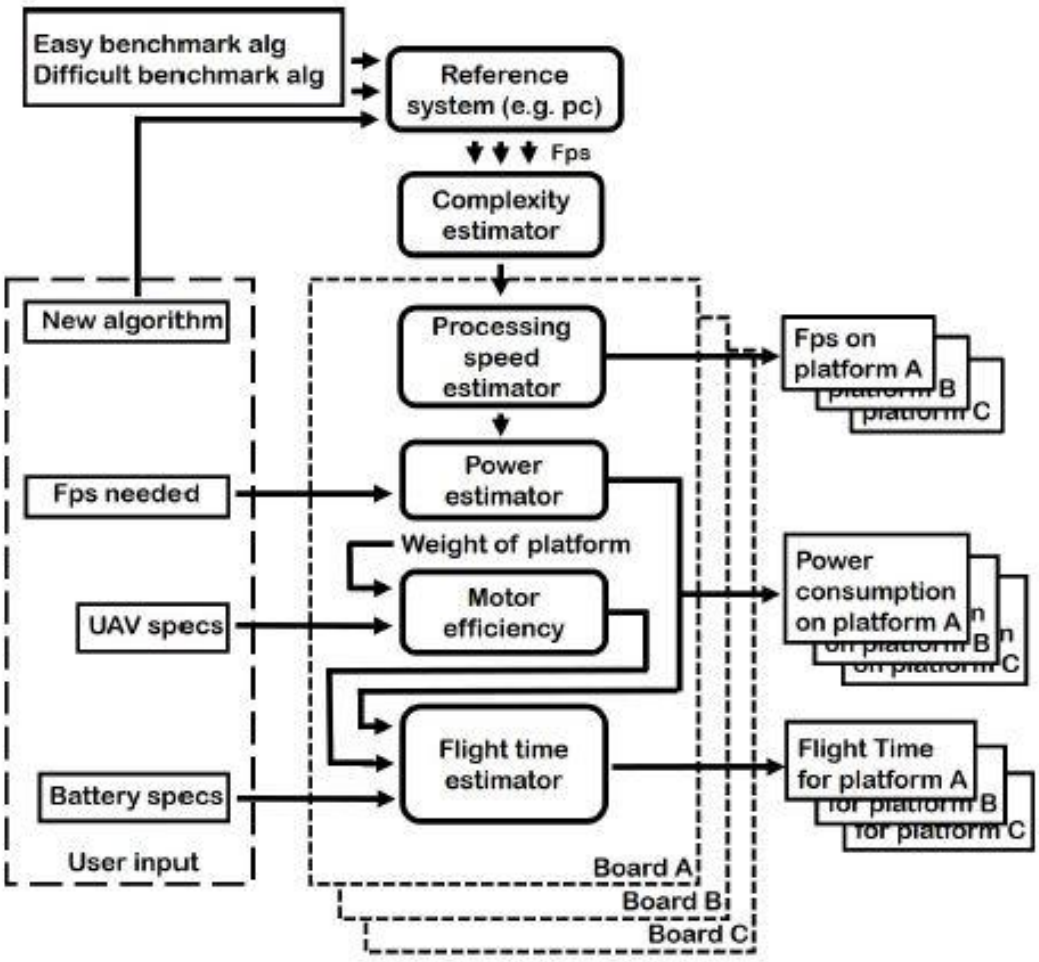

Figura 7 - Framework para definição do processador embarcado. Fonte: [29]

Nesta pesquisa foram englobadas diversos processadores, conforme mostra a Tabela 4, e cada um foi avaliado em certos critérios de performance como fps (frames por segundo), consumo de potência, e velocidade de processamento, para uma dada aeronave (Parrot A.R 2.0) rodando dois algoritmos de computação visual.

Tabela 4 - Processadores embarcados analisados no trabalho de Hulens [29]

Name	Processor	Memory	Weight (gram)	Power@100% (Watt)	Volume (cm^3)
Desktop	Intel I7-3770	20GB	740	107	4500
Raspberry PI model B	ARM1176JZF-S	512MB	69	3,6	95
Odroid U3	Samsung Exynos4412	2GB	52	6,7	79
Odroid XU3	Samsung Exynos 5422	2GB	70	11	131
Jetson	Cortex A15	2GB	185	12,5	573
mini-ITX atom	Intel Atom D2500	8GB	427	23,5	1270
mini-ITX intel I7	Intel I7-4770S	16GB	684	68	1815
Nuc	Intel I5-4250U	8GB	550	20,1	661
Brix	Intel I7-4500	8GB	172	26	261

Conforme relatado na conclusão do trabalho, o Raspberry Pi, ainda que sendo uma plataforma de baixa velocidade de processamento quando comparadas com as outras placas do estudo, foi capaz de desempenhar processamento em tempo real de ambos os algoritmos de computação visual, sendo um deles um programa de detecção de pessoas. Portanto, sendo o algoritmo desenvolvido no trabalho aqui descrito a detecção de um alvo mais simples que as características envolvidas na detecção de pessoas, é válida a conclusão de que o Raspberry Pi Zero é uma ferramenta suficiente para o desempenho esperado para o drone no projeto.

Seguindo as recomendações expostas no site da fabricante do drone, o Raspberry Pi Zero contendo a plataforma operacional em Linux é conectado ao controlador A3 por meio de uma conexão USB-TTL conforme representado na Figura 8.

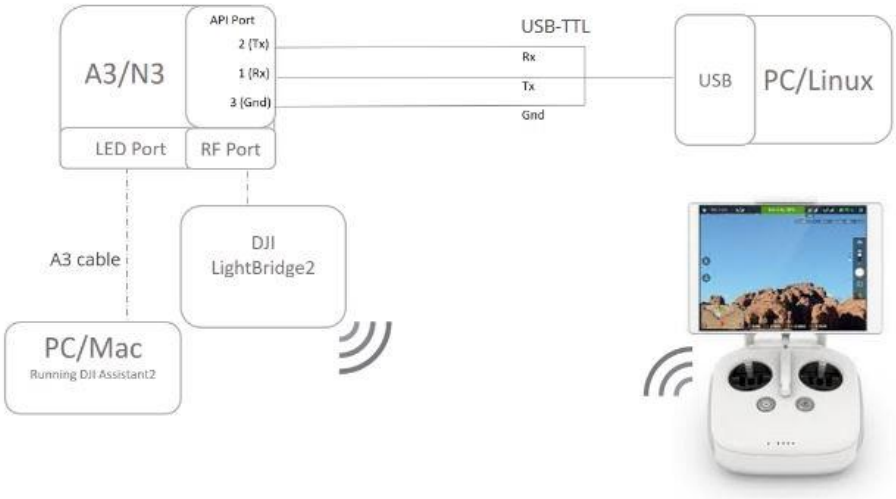

Figura 8 - Conexões entre os hardwares externos e o DJI Matrice 600. Fonte: [30]

É válido observar que os requerimentos para a performance do drone neste projeto envolvem um comportamento autônomo e, portanto, a conexão física entre um PC contendo o software DJI Assitant 2 e o drone, conforme consta na Figura 8, não existirá fora do contexto de testes em ambiente virtual. A conexão remota entre o controle e o drone se mantem como medida de segurança para os casos em que é necessário o pouso com intervenção humana.

A presença de um processador embarcado adiciona uma liberdade maior ao sistema quando comparado aos casos em que o processamento é feito remotamente e os comandos são enviados via wireless seja por radiofrequência ou sinal Wi-Fi.

Barták [31] descreve em seu trabalho uma metodologia para pouso de quad-cópteros autônomos utilizando processamento remoto de imagens com alvos pré-definidos, o drone utilizado por ele foi o Parrot A.R 2.0 que possui comunicação Wi-Fi com o computador que executa o processamento remoto. Contudo, o distanciamento entre o cálculo do movimento a ser executado e o hardware que o executa pode trazer movimentos erráticos ou falhas de comunicação por completo, conforme exemplifica Barták em seu trabalho.

Sendo assim, a conexão direta, conforme o caso do projeto aqui descrito, entre o controlador A3 e o Raspberry Pi Zero certifica uma comunicação de dados limpa e com baixa probabilidade de rompimento. Como desvantagem, o embarcamento do processador leva a um peso extra no drone, o que prejudica a sua destreza e o tempo de autonomia.

3.2 Definição da Câmera

Tendo em vista que o projeto é intimamente relacionado com o processamento de imagens, foi necessário o levantamento de hardwares que viabilizassem a captura de imagens como insumo para os algoritmos de computaçao visual, que, por sua vez, geram as manobras a serem executadas pelo drone, para, assim, garantirem um pouso autônomo eficiente.

O hexacóptero utilizado já possui uma câmera acoplada a ele, a DJI Zenmuse Camera Z3, representada a esquerda na Figura 9. Entretanto, essa câmera não possui conexão via SDK com o drone, o que implica que os frames por ela capturados são inacessíveis via código. Entramos em contanto com a equipe de desenvolvedores da DJI, em Junho de 2018, para buscar esclarecimentos sobre métodos alternativos de obter tais frames da Zenmuse Z3 mas a equipe nos respondeu que essa integração está em desenvolvimento na nova versão do SDK, contudo, sem previsão de lançamento.

Figura 9 - (Esquerda) Camera Zenmuse Z3; (Direita) Raspberry Pi Camera.

Mediante a inviabilidade de utilizar a câmera presente, fez-se necessária a aquisição de uma nova câmera. O hardware escolhido foi a Raspberry Pi Camera (ou RaspiCam), representada a direita na Figura 9, um dispositivo feito especialmente para a plataforma Raspberry Pi. Dentre as vantagens da RaspiCam podemos destacar o baixo peso e a facilidade de conexão com o processador embarcado escolhido. A Tabela 5 traz as principais especificações da câmera.

Tabela 5 - Especificações da Raspberry Pi Camera. Fonte: [32]

Weight	3 gramas
Still Resolution	8 MB
Sensor	Sony IMX219
Optical Size	¼”
Horizontal field of view	62.2°
Sensor Image Area	3.68 x 2.76 mm
Focal length	3.04 mm

É interessante observar que a escolha do conjunto entre Raspberry Pi e a RaspiCam viabilizam um conjunto de hardware para interação externa de baixo custo, o qual torna possível a atribuição de autonomia para operações do drone mediante o desenvolvimento de códigos de computação visual.

4. Análise de Software

Neste capítulo, traremos um resumo dos softwares utilizados para a implementação e testes do nosso algoritmo de computação visual de modo a descrever ao leitor a arquitetura dos códigos desenvolvidos e a forma como se dá sua comunicação com o drone.

4.1 Configuração de Software

Tal como todo projeto de robótica, a forma como o programador interage com o robô é de extrema importância. Logo, a frequência com que ele se comunica e os protocolo de dados a serem recebidos e fornecidos são essenciais para assegurar um desenvolvimento eficaz de qualquer pedaço de código. O capítulo a seguir trará a estrutura em software do nosso projeto de forma a elucidar a forma como nossos objetivos foram traduzidos em código e enviados ao drone.

Como dito anteriormente, o DJI Matrice 600 possui um controlador de voo denominado A3 [23], o qual garante o acionamento correto dos rotores do drone para a movimentação desejada, bem como armazena informações referentes ao voo como a posição do GPS, estado da bateria e altitude. A comunicação efetiva do A3 com os rotores é desconhecida e por isso é modelada como uma caixa preta.

Com a finalidade de viabilizar uma comunicação por parte de desenvolvedores externos, a equipe de DJI desenvolveu um SDK (Software Development Kit) para a linha Matrice. Um SDK, ou kit de desenvolvimento de software, é tipicamente um conjunto de ferramentas que permitem a criação de aplicações para determinados pacotes de software, software framework, plataforma hardware, sistema operacional ou desenvolvimento em plataformas similares [33] [34].

Sendo assim ao embarcarmos o SDK, instalando-o no Raspberry Pi Zero, temos acesso, por meio de linguagem C++, a uma pluralidade de comandos e informações do drone. A versão do SDK utilizada neste projeto é a 3.7 lançada em 14 de Agosto de

2018. O SDK 3.7 fornece acesso a um uma classe denominada *Vehicle* a qual nos permite a construção de um objeto drone, que contém todas as características do drone no mundo real prontos para manipulação por código.

A Figura 10 traz uma representação simplificada das etapas de comunicação com o SDK com o DJI Matrice 600. É valido apontar os canais de dupla comunicação, especialmente entre o Raspberry Pi Zero e o controlador A3. Esse tipo de feedback de informações assegura uma comunicação rápida e com mais dados fornecidos ao desenvolvedor para compreensão de erros na estabilidade do voo.

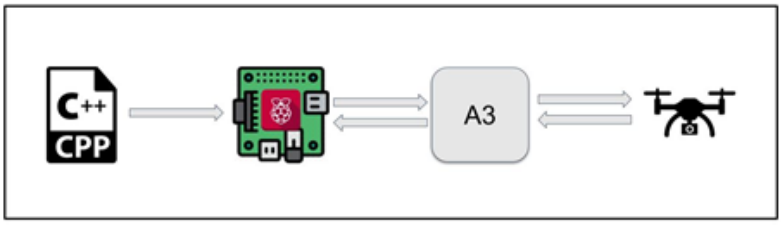

Figura 10 - Representação da comunicação entre desenvolvedor e drone

Para os testes em ambiente virtual, um outro software, chamado DJI Assistant 2, foi utilizado. Quando estabelecida uma conexão serial entre o hexacóptero e um PC, esse software permite que todos os dados que seriam enviados do A3 para os rotores reais, sejam enviados para uma representação virtual do drone em um ambiente de simulação, conforme mostra a Figura 11.

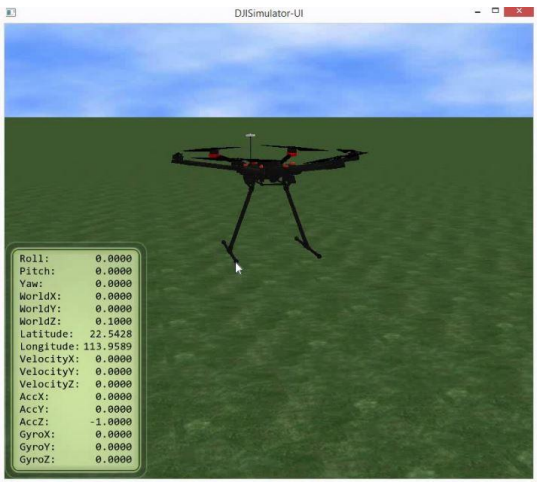

Figura 11 - Teste em ambiente virtual

O Raspberry Pi Zero possui como sistema operacional o Linux versão Raspian Jessie Stretch. A escolha pelo Linux foi um requerimento de software indicado no site da DJI.

É importante destacar que a embarcação do processador diretamente ao drone traz duas abordagens de comunicação possível. A primeira consiste no Raspberry Pi Zero agindo como um copiador de sinais oriundos do A3 e os enviando, seja por rádio frequência ou Wi-Fi, para um desktop remoto realizar o processamento e devolver as informações necessárias ao A3. A segunda arquitetura, está na incumbência do processamento diretamente no Raspberry Pi Zero e, assim, na garantia da autonomia plena da aeronave.

No projeto aqui relatado, visando prevenção de perdas de sinais, optou-se pela segunda forma de comunicação. Sendo assim, foi necessária a utilização de um software para garantir a comunicação com o Raspberry Pi Zero para transferência dos códigos a serem processados nele. Para tal utilizamos o software de comunicação SSH, PuTTY [35], e o software de cliente FTP, WinSCP [36].

O PuTTY viabiliza que o desenvolvedor acesse um cliente remoto desde que ambos estejam na mesma rede, conforme mostra a Figura 12. Após definirmos qual o IP do Raspberry Pi Zero na rede do roteador utilizado, podemos acessar o conteúdo do processador embarcado e, portanto, copiar, criar e compilar arquivos. O WinSCP, por sua vez, foi utilizado para a transferência de arquivos externos ou download dos arquivos contidos no Raspberry Pi Zero para edição externa.

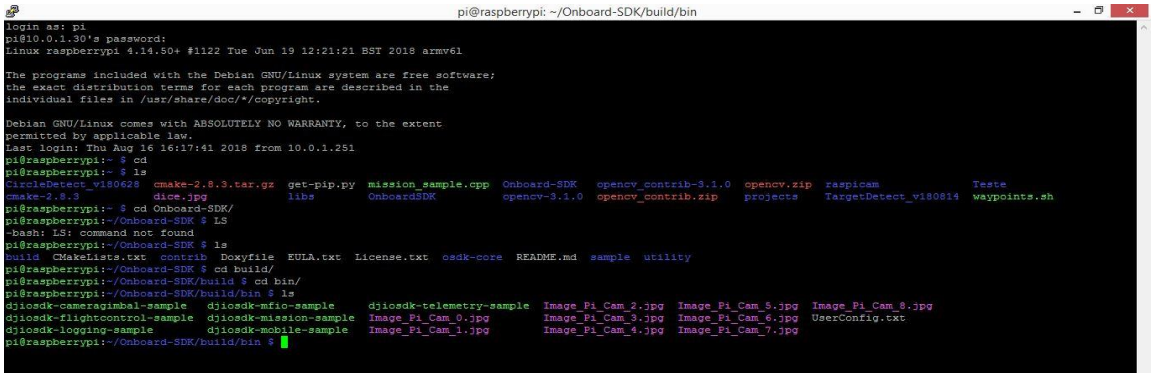

Figura 12 - Exemplo da comunicação com a ferramenta do PuTTY

4.2 Comunicação por Tópicos

A arquitetura da programação utilizando o SDK 3.7 em C++ consiste em uma programação por tópicos. Neste tipo de programa, configuram-se dois elementos: o publicador e subscritor. O primeiro, emite uma mensagem sem se preocupar com quem irá recebê-la, transmitindo-a em uma frequência definida e com um protocolo de comunicação acordado. O subscritor, por usa vez, seleciona os tópicos que deseja conhecer e as usa da forma que convir no código [37]. Como boas práticas, em geral, os elementos do código atuam sob ambos os papéis, para que, assim, o subscritor possa enviar mensagens ao publicador inicial atestando o recebimento da mensagem.

Um dos modelos de programação fundamentados em publicador-subscritor mais disseminados é o ROS (Robotic Programming System) [38]. Contudo, o SDK fornecido pela DJI já possui uma estrutura pronta para tal arquitetura e, portanto, não optamos por seguir uma implementação com ROS. A fim de ilustrar a forma de utilização desse formato de programação com drone, trazemos a seguir um pseudo-código referente a decolagem do drone.

Algoritmo de Decolagem

```
// Rotina de Inicialização do Drone
    do:
        Acessar dados do drone
    if erro
    |   Fim do programa
    else
    |   continua

// Rotina de Decolagem
    do:
        Publica no tópico de TakeOff
        Aguarda confirmação do A3
    if confirmação
    |   Subscreve no tópico de rotação dos motores (MotorsSpeed)
    |   Subscreve no tópico de posição do GPS (GPSData)
    |   altitude = GPSData.z
```

```
        else
        |       Fim do programa

    // Processo de Decolagem

        While: (Altitude menor que 1,2 m) E (Rotação dos motores ≠ 0)
                Subscreve no tópico de STATUS do drone
        Loop


    // Finalização da comunicação

                Imprime valor atual da altitude
                Cancela as subscrições
                Limpa as mensagens trocadas

        End
```

É interessante destacar a flexibilidade desse dessa arquitetura de programação, uma vez que poderíamos acessar outras informações sobre o drone durante a decolagem, como a velocidade angular nos seus eixos ou até a temperatura no A3.

4.3 Controle de Posição

Um ponto que diferencia o DJI Matrice 600 da maioria dos drones comerciais é o fácil acesso à sua posição devido ao GPS e, portanto, os comandos de deslocamento são diretamente para a posição desejada. Trabalhos como [39] e [40] tratam de drones cujo controle está na velocidade e por isso um PID é necessário para que haja uma etapa de integração e, com algum grau de precisão, consigamos controlar a posição do drone.

Portanto, nos códigos aqui realizados, para execução de deslocamentos, basta a subscrição nos tópicos do GPS para que possamos dar instruções diretas do ponto de destino que o drone deve atingir através das seguintes equações:

$$X_{GPS_{desejado}} = X_{GPS_{atual}} + \partial x \qquad (1)$$

$$Y_{GPS_{desejado}} = Y_{GPS_{atual}} + \partial y \qquad (2)$$

$$Z_{GPS_{desejado}} = Z_{GPSatual} + \partial z \qquad (3)$$

Uma vez conhecido os intervalos de deslocamento $(\partial x, \partial y, \partial z)$ a partir do algoritmo de computação visual, subscrevemos no tópico do GPS e, após incrementos, conferimos se o drone já chegou ou não na posição (x,y,z) desejada de acordo com o GPS. É válido apontar que esse método de deslocamento é incremental e não absoluto, uma vez que, para o deslocamento absoluto, necessitaríamos realizar operações nas latitudes e longitudes, e pequenos erros nas casas decimais de tais operações poderiam acarretar em manobras de risco no mundo real.

Contudo, as equações expostas previamente jamais poderiam configurar um movimento real, pois o drone utilizado possui uma inércia associada, o que significa que, conforme as distâncias entre a posição desejada e a posição atual ficam menores, a velocidade vai diminuindo igualmente: como um carro que inicia acelerando e depois realiza uma frenagem para parar na posição desejada. Sendo assim, tal como no exemplo do carro, cria-se uma tolerância da posição desejada e admite-se que, uma vez dentro do cubo de tolerância por um certo intervalo de tempo, o drone pode parar de incrementar ou reduzir sua velocidade. A Figura 13 mostra uma representação deste cubo, e portanto as condições de contorno das equações (1),(2) e (3) são:

$$||X_{GPS_{desejado}} - X_{GPS_{atual}}|| \leq \varepsilon x/2 \qquad (4)$$

$$||Y_{GPS_{desejado}} - Y_{GPS_{atual}}|| \leq \varepsilon y/2 \qquad (5)$$

$$||Z_{GPS_{desejado}} - Z_{GPS_{atual}}|| \leq \varepsilon z/2 \qquad (6)$$

Para os testes executados, foram utilizados $\varepsilon x = \varepsilon y = \varepsilon z = 20\ cm$ tais recomendações foram seguidas de acordo com o site da DJI sobre a precisão do GPS.

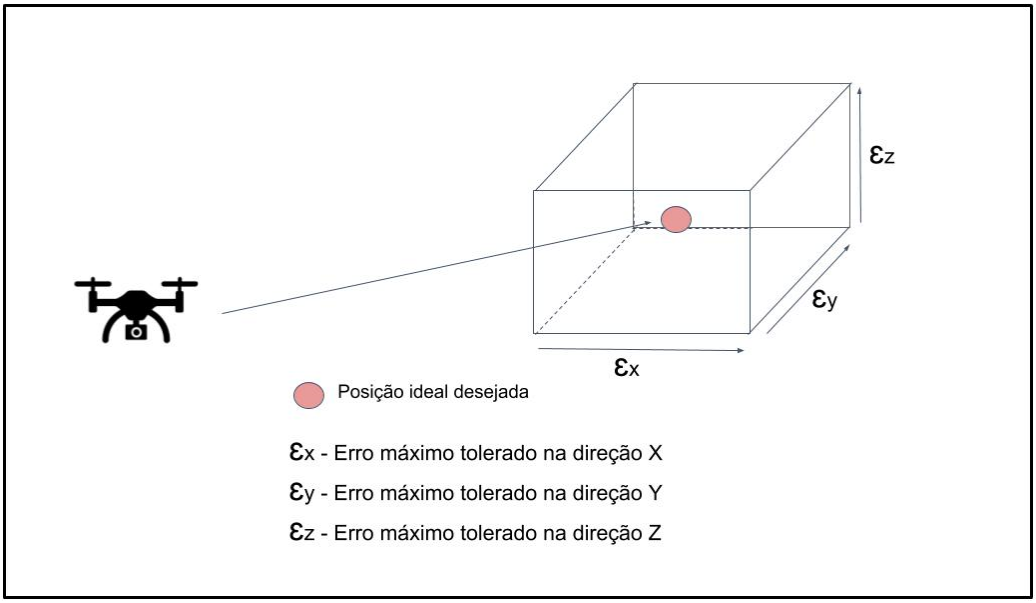

Figura 13 - Representação do cubo de tolerância para deslocamentos do drone

O principal problema com o mecanismo de movimentaçao do reside em deslocamentos de baixa amplitude. Sendo assim, executamos testes em ambiente virtual para deslocamentos de até 15 cm em Y, conforme mostra a Figura 14, para o qual tivemos um deslocamento efetivo segundo o GPS de $5{,}07\ cm$ e, portanto, um erro de 66,7%.

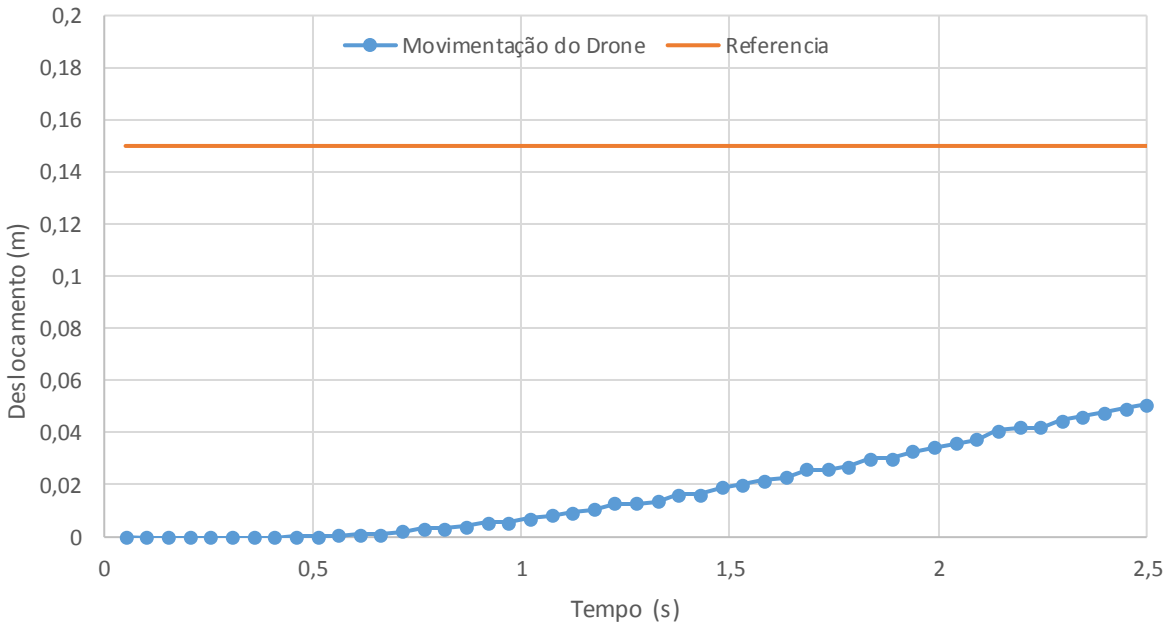

Figura 14 - Deslocamento de 0,15 metros a partir do algoritmo de movimentação

Novamente observa-se que após romper a barreira do cubo de tolerância em $\varepsilon x/_2 = 10\ cm$ a aeronave já inicia frenagem, mas uma vez que a aceleração positiva foi breve a inercia acaba por ser o fator predominante e leva a uma frenagem mais rápida. Ainda assim os resultados se mostraram satisfatórios para o objetivo de garantir o pouso com $1m^2$ de precisão.

Para casos de movimentos com amplitudes maiores, como o movimento $[0{,}5;\ 0{,}75;\ 2{,}0]\ m$ exemplificado na Figura 15, Figura 16 e Figura 17, observamos erros percentuais bastante menores e, consequentemente, movimentos muito mais precisos.

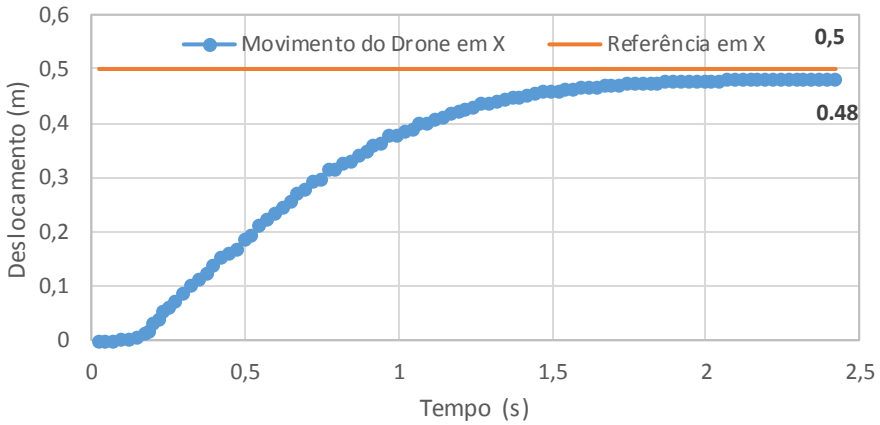

Figura 15 - Movimentação na direção x com comando de $0{,}5\ m$.

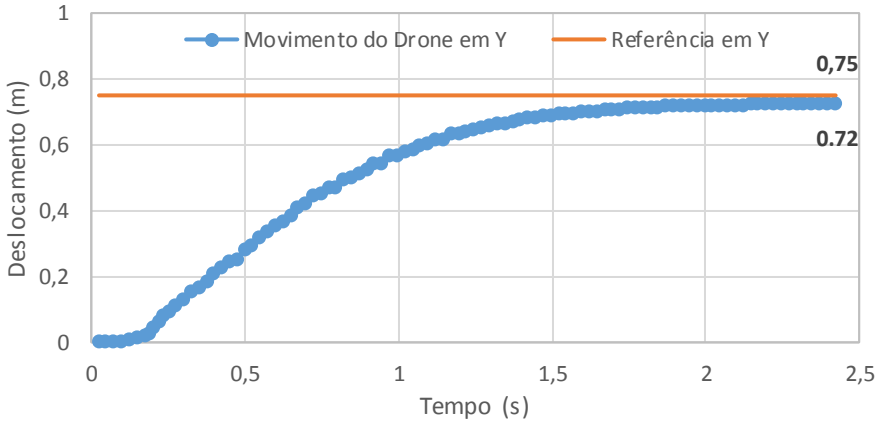

Figura 16 - Movimentação na direção y com comando de $0{,}75\ m$.

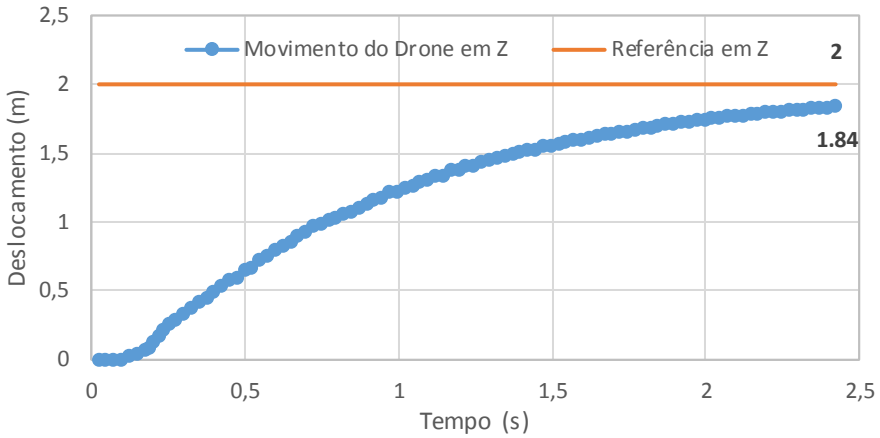

Figura 17 - Movimentação na direção z com comando de $2{,}0\ m$.

4.4 Computação Visual

A movimentação do drone vem a partir do resultado do tratamento da imagem capturada pela câmera realizado pelo algoritmo de computação visual, treinado para detectar e extrair posicionamentos de um alvo pré-definido. O alvo escolhido consiste em um quadrado com quatro triângulos internos formados a partir das suas diagonais como mostra a Figura 18. A opção por esse alvo reside na facilidade de definir geometricamente as referências e nos distanciarmos da dependência de detecção de cores, uma vez que essa estratégia de análise cromática falha com frequência em variações de luminosidade.

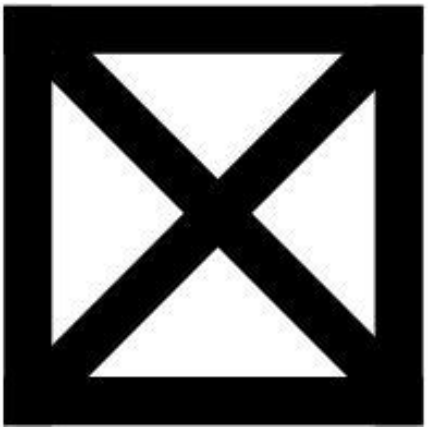

Figura 18 - Alvo utilizado como guia para pouso do drone

O alvo foi construído sobre uma placa de poli-acetileno a partir de adesivos de fita isolante. Desta forma foi possível a manipulação de certos parâmetros da figura afim de consolidarmos um alvo final que não comprometesse o requerimento do pouso de $1\ m^2$, mas que, ao mesmo tempo, pudesse ser detectado de alturas razoáveis.

Os parâmetros analisados foram: o tamanho do lado do quadrado, a espessura do lado do quadrado e a espessura das diagonais. No laboratório, executamos experimentos para verificar a detectabilidade do alvo nos casos extremos de máxima proximidade e máximo distanciamento para diferentes casos dos parâmetros descritos acima. A Tabela 6 traz os resultados desses estudos.

Conforme podemos observar, o estudo de número 4 forneceu a maior amplitude de atuação indicando que o drone é capaz de detectar o alvo a partir de 7,5 metros de altura e até 1,3 metros do chão. O limite inferior se justifica pois conforme o drone chega perto do local de pouso o alvo excede os limites do campo de visão da câmera e. Dado esse limite inferior e uma margem de segurança de 70 cm, configuramos o código para inicializar o processo de pouso monitorado a partir de 2 metros de altura.

Tabela 6 - Testes para determinação dos parâmetros do alvo

Teste	Espessura Interna	Espessura Externa	Lado	Distância Mínima	Distância Máxima	Diferença
1	5	5	70	130	600	470
2	6.5	6.5	70	150	630	480
3	8	8	70	150	750	600
4	8	6.5	70	130	750	620
5	8	5	70	140	700	560
6	3.5	3.5	50	100	350	250
7	5	5	50	110	500	390
8	6.5	6.5	50	120	550	430

O código de computação visual deve, então, detectar o alvo no frame capturado pela câmera, calcular a distância do centro do alvo para o centro do frame e converter essa distância em pixels para metros, e, assim, o algoritmo de movimentação descrito préviamente pode movimentar o drone.

A seguir, detalharemos o funcionamento das rotinas envolvidas no algoritmo de computação visual. Para facilitar a compreensão, disponibilizamos um fluxograma simplificado do código de visão computacional na Figura 19. Posteriormente, discutiremos cada uma de suas partes.

4.4.1 Captura da Imagem

Primeiramente, a imagem da RaspiCam deve ser capturada. Contudo, a câmera demora um certo número de frames para se adaptar à luminosidade do local. Testes realizados nos mostraram que após 20 frames capturados a imagem possui uma claridade adequada para a detecção do alvo através do algoritmo de computação visual. A Figura 20 exemplifica uma imagem com brilho suficiente para detecção do alvo.

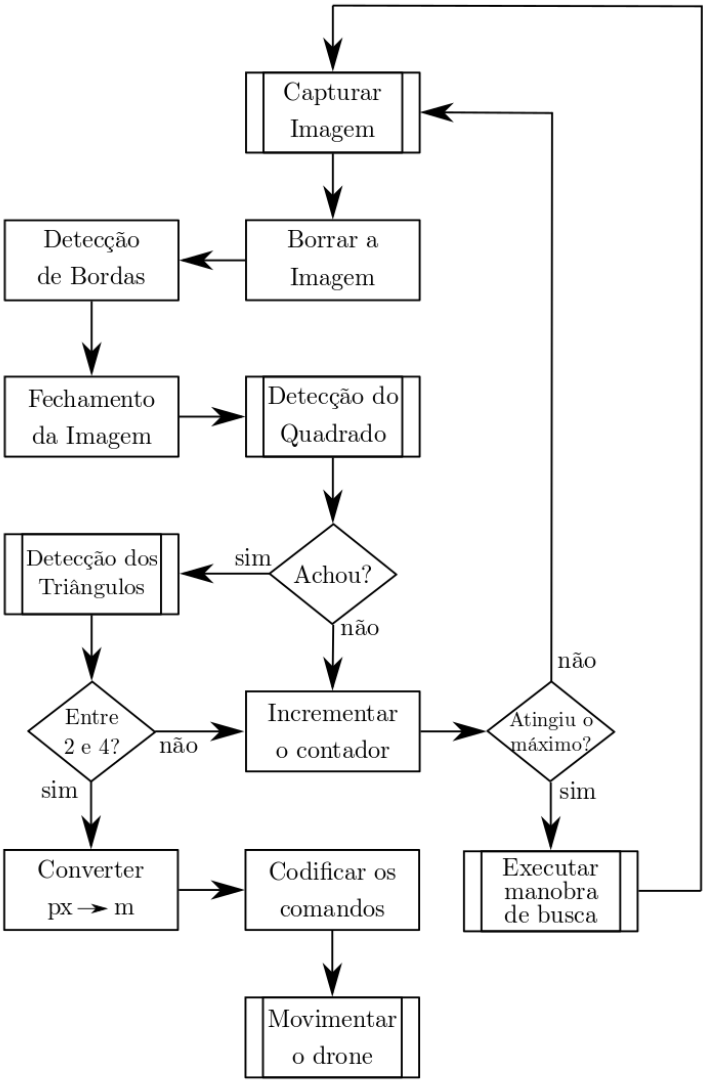

Figura 19 - Fluxograma Simplificado do Algoritmo de Computação Visual

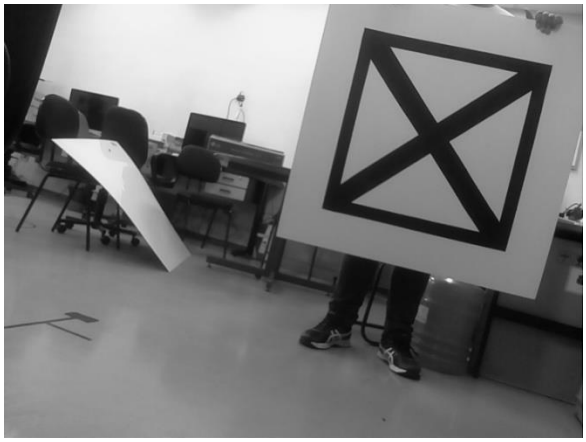

Figura 20 - Seleção da Imagem de brilho mínimo durante a captura

4.4.2 Pré-Processamento da Imagem

O primeiro passo do pré-processamento de imagens para visão computacional normalmente consiste na conversão do espaço de cor BGR para Escala de Cinza. Como a imagem capturada pela RaspiCam já é devolvida em escala de cinza, esse passo pode ser ignorado.

O próximo passo é borrar a imagem levemente. Esse tratamento é útil para a remoção de ruídos, que podem interferir com o processo de detecção de bordas, o qual é feito logo em seguida, onde todas as bordas detectadas são demarcadas em branco enquanto o resto da imagem permanece preto.

Finalmente, fazemos a abertura e o fechamento da imagem. Esses processos consistem na erosão e subsequente dilatação (e vice-versa) das bordas. O primeiro, ajuda a remover pontos brancos do lado de fora do alvo, enquanto o último remove pontos pretos do lado de dentro.

Através de alguns testes, descobrimos que o processo de abertura não é vantajoso nessa aplicação em específico. Além disso, o processo de fechamento ajuda na detecção nos casos em que o alvo está próximo da câmera, mas pode dificultar a mesma nos casos em que a distância é maior. Por isso, a variável `iterationsClose` pode assumir quaisquer valores no intervalo $\{0, 3\} \in \mathbb{Z}$. A Figura 21 traz um exemplo do resultado da imagem após esse processo.

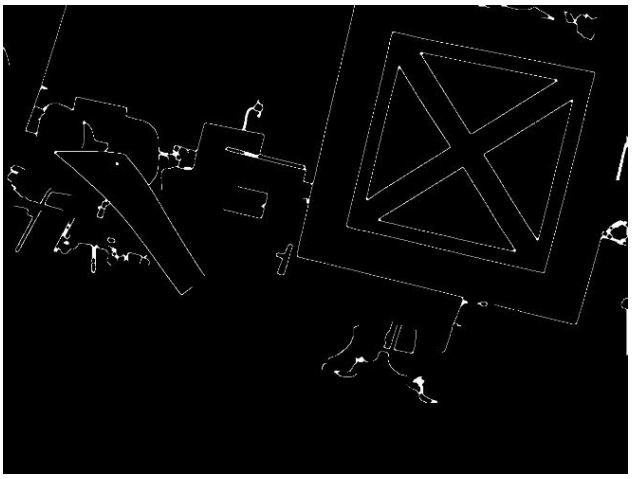

Figura 21 - Exemplo do frame após o pré-processamento

4.4.3 Detecção do Quadrado

Depois de finalizado o pré-processamento, dá-se início à detecção do alvo de fato. A primeira coisa a se fazer é a detecção do quadrado externo. Usando a função `findContours` da biblioteca do OpenCV, obtemos um vetor com todos os contornos presentes na imagem. Para a detecção do quadrado, um laço é passado através desse vetor.

Para cada contorno, uma aproximação com linhas retas é feita, dado uma certa precisão perimetral, reduzindo assim o número total de pontos do contorno. Se a aproximação resultante possuir exatamente 4 pontos, isso é um forte indicador de que o contorno em questão é um quadrado.

O próximo passo é a análise das proporções, i.e. a razão entre a largura e a altura da caixa delimitadora do contorno. Se a mesma se encontrar entre 0.8 e 1.2, temos mais uma indicação de que esse contorno é um quadrado. Contudo, esse quadrado pode estar torto em relação à camera, i.e. não estar voltado diretamente para a câmera. Por este motivo, executamos uma homografia para obter a vista frontal do possível quadrado. Os resultados das operações de detecção do quadrado e da posterior homografia estão representados na Figura 22 e na Figura 23.

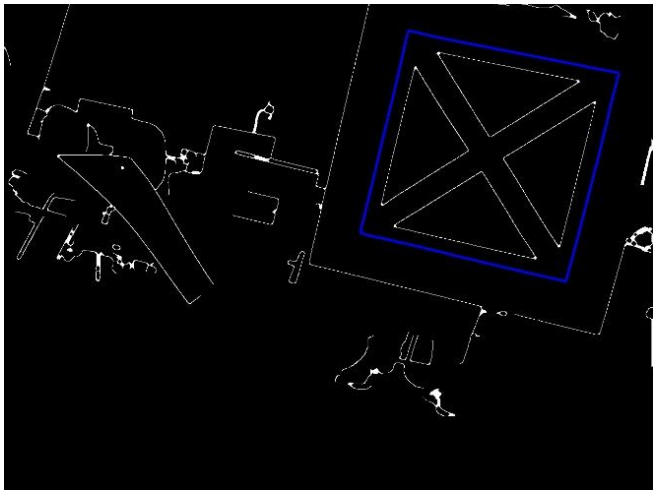

Figura 22 - Exemplo do resultado após deteccção de contronos do quadrado

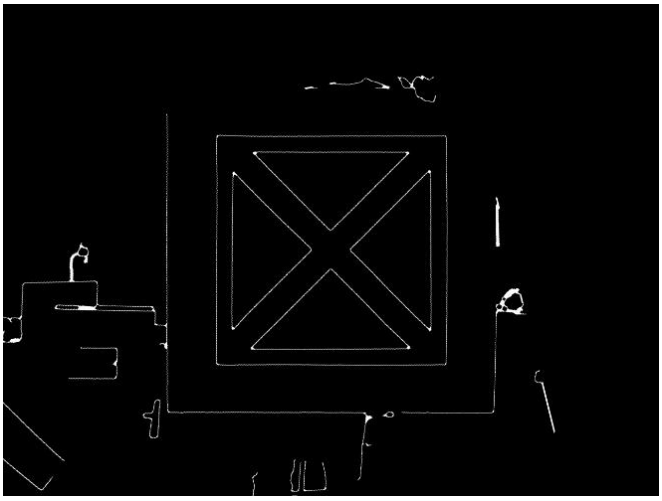

Figura 23 - Resultado após o processo de homografia

4.4.4 Detecção dos Triângulos

Para termos certeza de que o quadrado detectado é de fato parte do alvo, passamos agora para a fase de detecção de triângulos. Para isso, apagamos o contorno do quadrado detectado (pintamos de preto) e recortamos o frame ao redor do mesmo.

Mais uma vez, utilizando a função `findContours`, passamos um laço pelo vetor obtido à procura de contornos que possam ser aproximados por três pontos. Se ao menos dois triângulos forem encontrados, assumimos que a detecção do alvo foi bem-sucedida. O resultado da detecção de triângulos está representado na Figura 24.

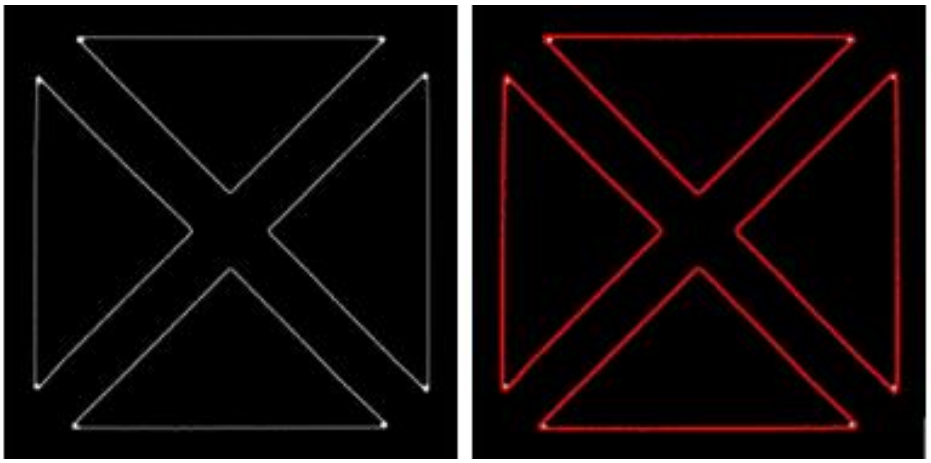

Figura 24 - Resultado do processo de deteccção de triângulos

4.4.5 Cálculo da Posição do Alvo

Uma vez encontrado o alvo, é necessária a realização do cálculo da posição do mesmo em relação ao drone. Para isso, realizamos o cálculo do centro de massa do quadrado externo com a ajuda da função `moments` do OpenCV.

Uma vez sabido o centro de massa, a distância em pixels entre o centro do quadrado e o centro do frame é facilmente calculada. Para convertermos essa distância em pixels para metros, utilizamos a razão entre o tamanho real do alvo e o tamanho percebido na imagem como mostrado na equação a seguir:

$$pixelScale = \frac{4 * realTargetSize}{Perimeter} \qquad (7)$$

Esse método elimina a necessidade de sabermos a altura do drone a cada instante. A Figura 25 traz uma representação do resultado após o cálculo da distância do centro do objeto para o centro do frame da câmera.

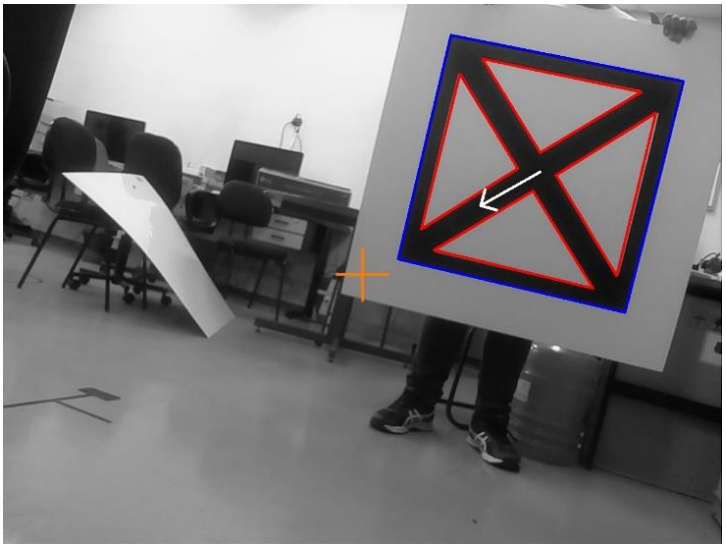

Figura 25 - Resultado do processo de cálculo da distância do alvo para centro do frame

4.4.6 Envio dos Comandos ao Drone

A distância a ser percorrida é então enviada ao drone. Junto com esse movimento, também comandamos a descida de metade da altura atual do drone.

4.4.7 Perda do Alvo

Uma falha na detecção do alvo pode ocorrer por vários motivos, entre eles: vibrações na hora da captura que podem acarretar em uma imagem não nítida; erros de posição do drone, que podem resultar na saída do alvo do campo de visão da câmera; dependendo da distância entre o drone e o alvo, o processo de fechamento da imagem (ou a ausência do mesmo) pode acarretar em uma falha de detecção. Nesses casos, é importante que a rotina de pouso não falhe e tente reencontrar o alvo autonomamente.

A primeira coisa a se fazer é iniciar um contador `missedDetections` que indica a quantidade de vezes que o drone falhou em em detectar o alvo. A cada iteração, mudamos o valor da variável `iterationsClose` e `comandamos o drone a subir 1 metro,` a fim de aumentarmos as chances de detecção.

No caso de isso não funcionar e o contador atingir um certo valor máximo, iniciamos uma manobra onde o drone retorna para a última posiçao onde o alvo foi encontrado e depois percorre um quadrado ao redor do ponto onde se encontra. Essa manobra consiste em vários passos, e, a cada etapa, uma nova captura é feita. Se, em algum momento durante essa manobra, o alvo for detectado com sucesso, a manobra é encerrada e o processo continua normalmente. Porém, caso a manobra chegue ao fim e o alvo não tenha sido detectado, o pouso autônomo não poderá ser efetuado com segurança. O drone estabiliza então no ponto onde está e uma mensagem de erro é exibida no terminal PuTTY, para que um humano tome o controle e realize o pouso.

5. Metodologia

Neste capítulo, elucidaremos o leitor sobre a metodologia utilizada nos testes em ambiente virtual e campo. Ao final, discutiremos as principais adaptações necessárias para a implementação do projeto no mundo real.

5.1 Testes em Simulador

Primeiramente, testamos o voo autônomo do drone utilizando o simulador contido em um software da própria fabricante, chamado DJI Assistant 2.0. Esse simulador nos permitiu realizar testes para a decolagem e movimento autônomo para alguns pontos salvos. Esse software, porém, é bastante limitado e não havia a possibilidade de realizarmos testes para o pouso assistido por imagens com as ferramentas já existentes. Por esta razão, desenvolvemos um simulador próprio para a realização de testes do algoritmo de pouso assistido. O simulador desenvolvido por nós, tem como base de funcionamento uma imagem quadrada de 18900 pixels de lado, adaptada por nós de [41], como mostra a Figura 26.

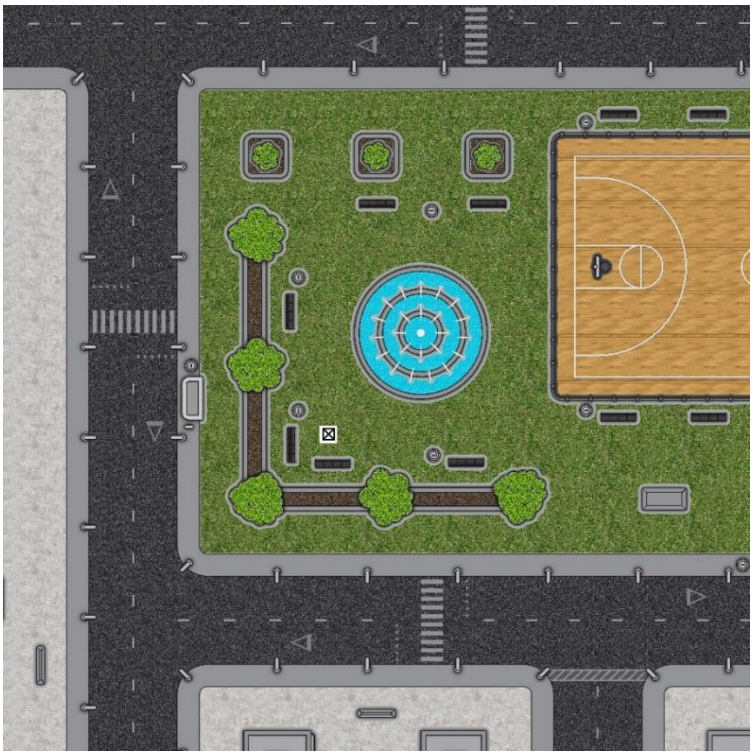

Figura 26 - Imagem base utilizada no simulador próprio. Adaptada de [41].

Esse tamanho grande nos permite recortar a imagem em frames muito menores simulando o que a RaspiCam estaria capturando durante o voo real, sem perdermos qualidade e detalhes.

Próximo do centro da imagem, se encontra o alvo onde queremos pousar. Através da posição atual do drone, calculamos o campo de visão da câmera virtual e recortamos o frame nessa área. Convertemos, então, para o espaço de cores em escala de cinza e alimentamos esse frame ao nosso algoritmo de pouso assistido. A Figura 27 e a Figura 28 exemplificam os frames recebidos pelo algoritmo.

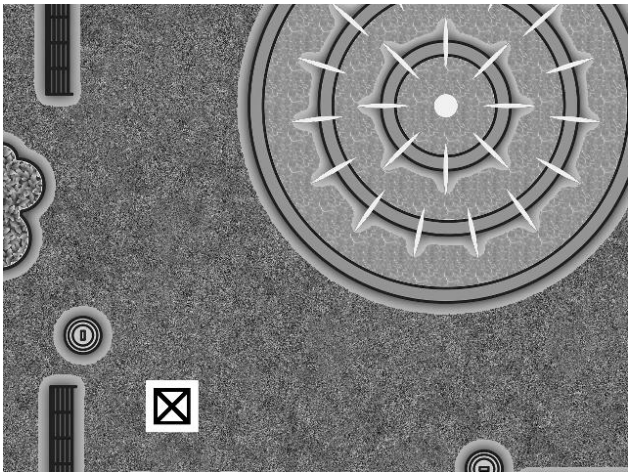

Figura 27 - Exemplo de imagem recebida pelo algoritmo de pouso assistido. Posição do drone em relação o centro da imagem original: x = 0, y = 0, z = 15.

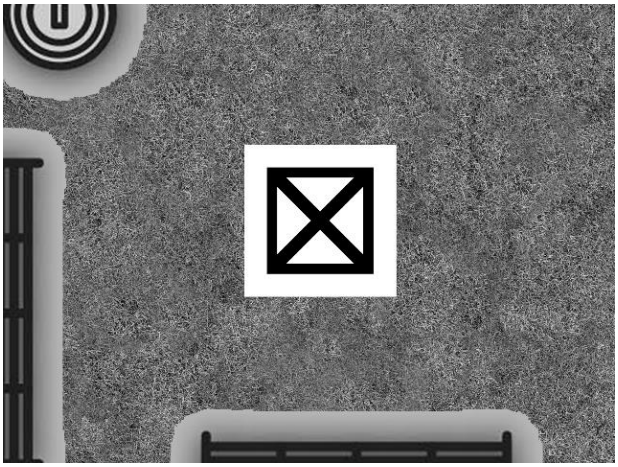

Figura 28 - Exemplo de imagem recebida pelo algoritmo de pouso assistido. Posição do drone em relação o centro da imagem original: x = -2.8, y = -3.3, z = 5.

5.2 Set-Up do Experimento

Os experimentos foram realizados na área de eventos localizada ao lado do prédio da Administração da Escola Politécnica da USP. As coordenadas de GPS (latitude, longitude e altura) de alguns pontos foram salvas e esses pontos foram numerados de 0 a 6, como mostrado na Figura 29 e na Figura 30. Tais pontos foram escolhidos estrategicamente para que pudéssemos testar o voo autônomo do drone com vários percursos distintos.

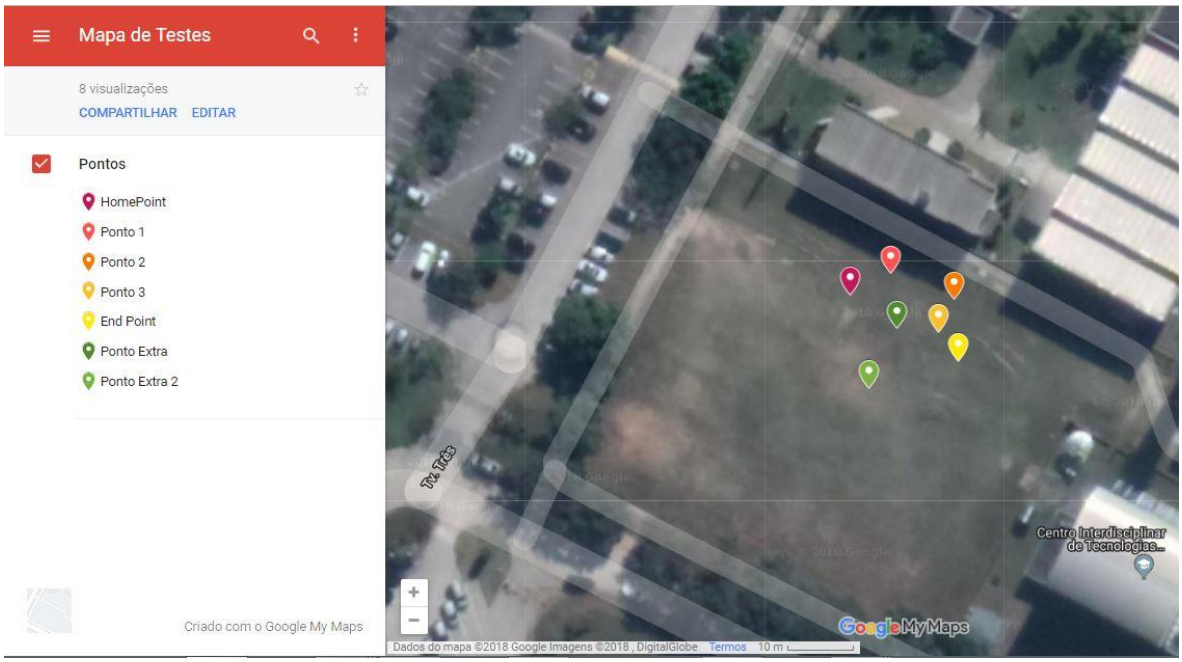

Figura 29 - Mapa dos pontos de teste salvos

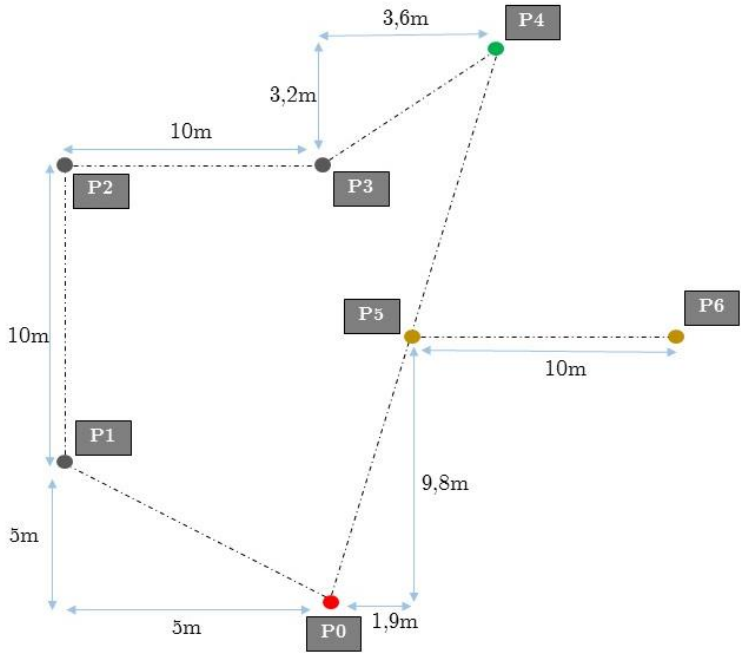

Figura 30 - Posição relativa entre os pontos salvos com cotas

Após os testes de movimentação autônoma, um último ponto GPS (P7) foi salvo, com as seguintes coordenadas: -0.411128024 de latitude, -0.815596781 de longitude e 4 metros de altura.

5.3 Testes do Algoritmo de Pouso

A fim de garantir a melhor configuração para os pousos, realizamos sucessivos voos autônomos para a posição P7 utilizando uma rotina que permite a movimentação do drone, dadas as coordenadas absolutas de um ponto. Após múltiplos envios do drone até o ponto P7, posicionamos gravetos imediatamente embaixo do centro do drone em cada viagem a fim de determinar um ponto central que compensasse os erros individuais de cada viagem autônoma. Medições apontaram para uma região de cerca 40 cm de diâmetro. Por fim, posicionamos o alvo no centro dessa região.

Em seguida ligamos um roteador configurado na mesma rede em que o Raspberry Pi havia sido configurado para se conectar. Um computador remoto também é

conectado nessa rede e inicia uma comunicação SSH com o Raspberry Pi via PuTTY, com a finalidade de enviar os comandos para o drone.

Em seguida os seguintes passos são executados:

1. Decolagem monitorada do drone;
2. Deslocamento arbitrário em direçao arbitrária;
3. Movimentação autônoma até P7;
4. Início do algorítmo de pouso assistido por imagens até 2 metros de altura;
5. Pouso monitorado do drone.

O item 2 se mostrou bastante importante devido a dois fatores: primeiramente, a distância entre o ponto de partida e P7 determina a velocidade com que o drone realiza a movimentação. Essa velocidade, atrelada ao método de frenagem do drone, influencia na precisão atingida; em segundo lugar, a direçao arbitrária determina a rotaçao do drone no momento de início do algorítmo de pouso. Na primeira manobra executada durante o pouso, o drone se alinha com o norte geográfico. Essa rotaçao gera uma imprecisão, que pode variar bastante de acordo com o alinhamento inicial do drone.

Desta forma, pudemos obter resultados expressivos a partir de 26 voos, realizados em diversas condições de iluminação, vento e posição inicial. Cada voo fornecia um relatório no qual as informações mais relevantes eram salvas em um arquivo .txt no Raspberry Pi, bem como as fotos utilizadas pelo drone para guiar o pouso. De forma externa, executamos a gravação dos voos com uma câmera remota e usando a câmera Zenmuse Z3 acoplada no Matrice 600. Por fim, com o auxilio de uma trena, medimos a distância do centro do drone pousado para o centro do alvo no chão.

5.4 Variações entre os testes em ambiente virtual e em campo

O simulador por nós desenvolvido permitiu a elucidação de partes vitais do código e foi o alicerce para verificação da funcionalidade simultânea da movimentação do drone com a detecção do alvo. Contudo, nos testes iniciais em campo verificamos

peculiaridades dinâmicas do drone e notamos que certas limitações são adicionadas quando utilizamos o Raspberry Pi como único processador da lógica.

Os erros encontrados para a cinemática do drone no ambiente de simulação se mostram bem diferentes dos observados na vida real, o que levou à introdução de movimentos de contrapeso para balancear translações involuntárias do drone.

Além disso, o algoritmo inicial de visão computacional trazia uma lentidão extrema para os pousos e foi necessária uma otimização do código para acomodar as limitações do Raspberry Pi.

Um terceiro ponto que se mostrou necessário na condução dos experimentos reais foi a redução do número de comandos enviados pelo computador para o Raspberry Pi tendo em vista a fragilidade da comunicação Wi-Fi entre o microprocessador e um computador remoto.

6. Resultados e Discussões

Neste capítulo, apresentamos os resultados obtidos a partir dos dados coletados nos testes em campo, de forma a verificar se a hipótese inicial do projeto foi comprovada com significância estatística. Apresentamos, ainda, uma discussão referente às falhas observadas durante os experimentos de forma a colaborar com outras verificações encontradas na literatura.

6.1 Coleta de Dados

Para cada um dos 26 voos analisados foram coletadas as seguintes informações

- Quantidade de imagens obtidas para execução do voo
- Quantidade de movimentos performados pelo drone
- Tempo total até o pouso
- Distância entre o centro do drone no fim da manobra até o centro do alvo
- Classificação em sucesso ou falha
- Relatório de voo contendo detalhes da cinemática de cada movimento

A quantidade de imagens nos permite compreender o número de fotografias necessárias pelo drone para obter um direcionamento preciso que garanta um pouso dentro dos requerimentos.

A quantidade de movimentos indica o número de vezes que a função destinada a movimentar o drone foi solicitada. É importante destacar que, embora haja uma proporcionalidade entre a quantidade de imagens e a quantidade de movimentos, essa correlação não é direta, uma vez que o drone pode perder o alvo e neste caso realizar um número maior de movimentos até que ele efetivamente capture imagens do alvo.

O tempo total da manobra foi extraído a partir das gravações dos voos, feitas com uma câmera externa, e a distância entre o centro do drone o fim da manobra até o centro do alvo foi medido com uma trena.

Por fim, a classificação em sucesso ou falha seguia o seguinte critério: caso o drone fosse capaz de pousar sem qualquer interferência humana com uma distância

menor que $1\,m$ do centro do alvo, este voo era considerado um sucesso, do contrário era um pouso falho. A Tabela 7 traz a consolidação desses resultados.

Tabela 7 – Resultados obtidos nos testes em campo

Voo	Imagens Capturadas	Movimentos	Tempo (s)	Precisão (m)	Status
1	1	3	22	0,37	Sucesso
2	3	5	43	0,15	Sucesso
3	1	3	23	1,58	Falha
4	3	6	42	0,62	Sucesso
5	2	3	42	0,22	Sucesso
6	11	18	138	0,47	Sucesso
7	1	3	30	0,76	Sucesso
8	2	5	40	0,33	Sucesso
9	3	6	50	0,02	Sucesso
10	1	3	30	0,9	Sucesso
11	-	-	-	-	Falha
12	4	7	68	0,33	Sucesso
13	-	-	-	-	Falha
14	1	3	27	1,3	Falha
15	6	5	30	1,22	Falha
16	9	14	137	0,6	Sucesso
17	2	4	48	0,37	Sucesso
18	-	-	-	-	Falha
19	2	5	46	0,65	Sucesso
20	2	3	38	0,83	Sucesso
21	1	2	33	0,84	Sucesso
22	-	-	-	-	Falha
23	5	4	70	0,65	Sucesso
24	1	3	27	0,62	Sucesso
25	5	6	69	0,74	Sucesso
26	5	6	60	0,14	Sucesso

6.2 Análise dos Pousos Falhos

6.2.1 Reflexo e Sombras

Os eventos em que os pousos não foram bem-sucedidos foram marcados por fatores externos que evidenciam as limitações do algoritmo de pouso aqui gerado. Os voos 11 e 18, por exemplo, foram impossibilidades devido à alta incidência de luz solar no alvo, gerando reflexos e impossibilitando, assim, a detecção efetiva do alvo pelo drone.

A Figura 31 traz um exemplo deste caso de reflexão e também aponta outro fator externo: a sombra do próprio drone. Novamente, em casos de sol intenso, a sombra do drone fica mais acentuada e pode colaborar para impossibilidade do drone em detectar os contornos do alvo.

Tal constatação traz a conclusão de que o sistema atual deve ser aplicado em condições amenas, de preferência com luminosidade controlada. Uma outra solução seria confeccionar o alvo em material difuso.

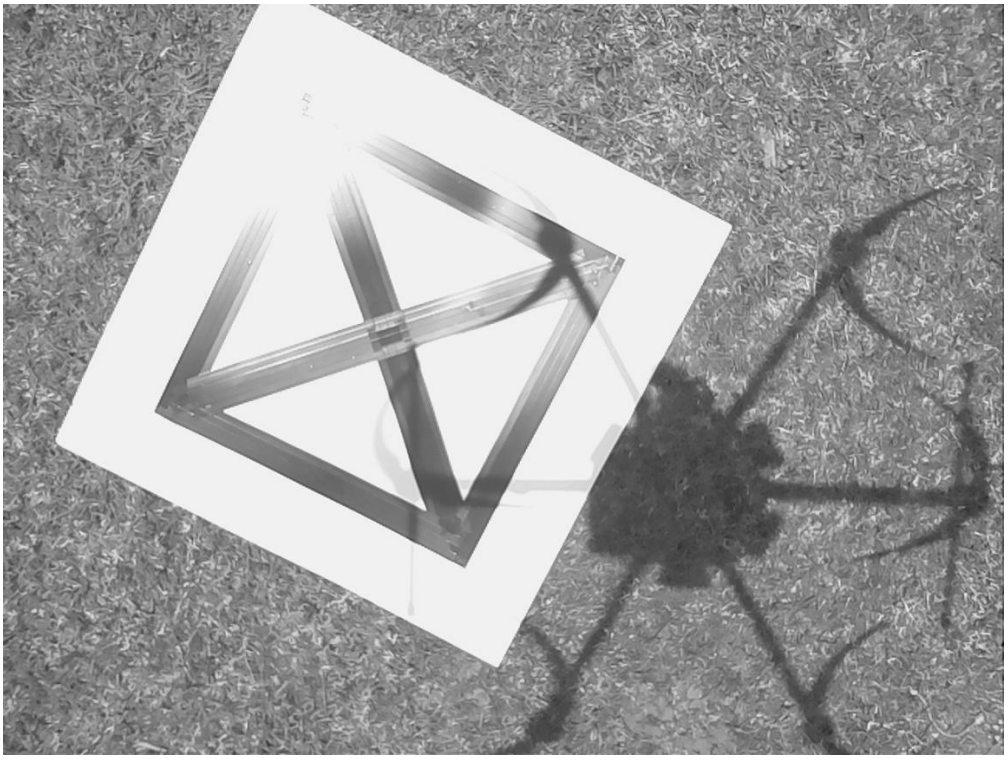

Figura 31 - Exemplo de falha na detecção do alvo devido a reflexo e sombra.

6.2.2 Distanciamento Contínuo

Outra questão observada nos eventos de falha do pouso são os casos em que há um distanciamento contínuo do drone com relação ao alvo até que o pouso seja inviável. Isso ocorre devido a uma imprecisão elevada do drone na direção x na medida em que executa movimentos.

Sempre que a aeronave executa um movimento há uma tendência de translação do drone no sentido positivo de x. Como forma de balancear essa verificação empírica de imprecisão, incluimos, em certas movimentações, um comando extra para percorrer uma distância no sentido negativo de x. Contudo, em situações que o erro acumulado era demasiado, esse contorno não se tornava efetivo, e resultava nos eventos de pousos falhos conforme os voos 13 e 22. Tal imprecisão é ainda mais acentuada quando o voo é executado sob fortes velocidades de vento.

Após uma série de voos prévios a coleta desses dados, constatamos que um comando de $-0{,}75\ m$ em x satisfazia os requerimentos necessários e forneciam uma precisão satisfatória para os propósitos do projeto. Contudo, é valido o corolário de que a adição de novos pesos ao drone ou configurações climáticas diferentes, como maiores ventos, demandam uma revisão deste parâmetro ou estudo mais detalhado das imprecisões do sistema de movimentação por GPS do drone.

Na literatura podemos encontrar estudos que colaboram para constatações de erros em offset para o DJI Matrice 600 Pro, especialmente mais acentuados nas direções horizontais (xy) do que na vertical (z). Ekaso [34] exemplifica medições que comprovam erros encontrados no sistema de cinemática em tempo real para georreferenciamento direto. Em especial, destacamos o resultado do autor sobre o offset de $0{,}7\ m$ na direção horizontal para velocidades acima de $4\ m/s$. O trabalho de Ekaso ainda traz a conclusão de que movimentos verticais tem precisão até duas vezes maior do que movimentos no plano xy da aeronave.

No contexto do trabalho aqui desenvolvido nota-se que a velocidade no momento da centralização do alvo excede $4\,m/s$ e, portanto, passamos a observar os erros de posicionamento referenciados.

As Figura 32 a Figura 34 evidenciam os gráficos de movimentação do drone nos 3 eixos em um movimento de centralização para o alvo. No movimento em questão, ocorrido no voo 2, o drone calculou que, para centralização do alvo, ele deveria realizar a movimentação $[0{,}41;\ -1{,}68;\ -2{,}39]$ em metros. Os erros finais, em percentual, podem ser verificados na Tabela 8.

Tabela 8 - Erros finais para uma movimentação de $[0{,}41;\ -1{,}68;\ -2{,}39]\ m$.

Erro em x	66%
Erro em y	4%
Erro em z	5%

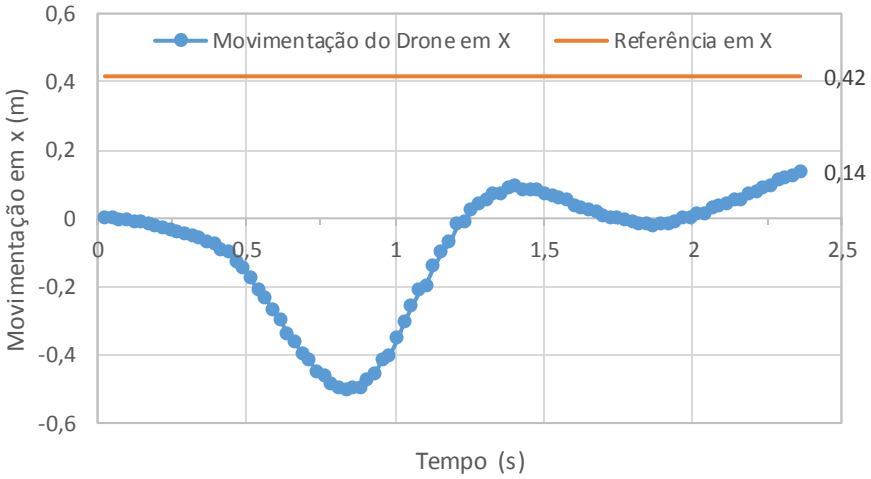

Figura 32 - Movimentação na direção x com comando de $0{,}41\ m$.

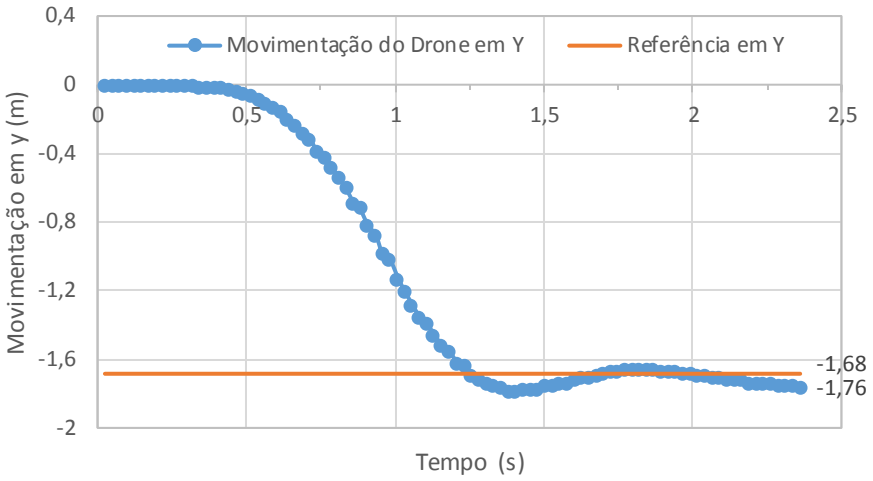

Figura 33 - Movimentação na direção y com comando de $-1{,}68\ m$.

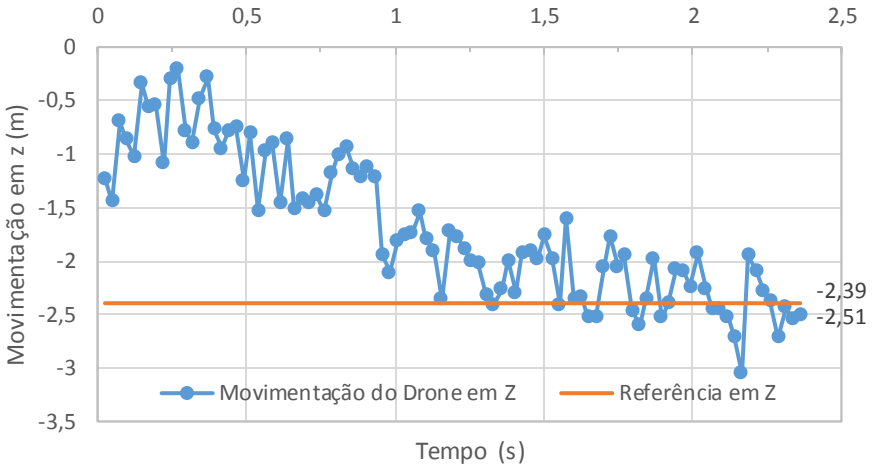

Figura 34 - Movimentação na direção z com comando de $-2{,}39\ m$.

Erros acentuados na direção y também foram observados em algumas movimentações, como mostram a Tabela 9, a Figura 35, a Figura 36 e a Figura 37, de um movimento ocorrido no voo 17, em que o drone centraliza o alvo a partir do comando $[0{,}60;\ \ 0{,}8;\ -1{,}65]\ m$.

Tabela 9 - Erros finais para uma movimentação de $[0{,}60;\ \ 0{,}8;\ -1{,}65]\ m$.

Erro em X	45%
Erro em Y	32%
Erro em Z	5%

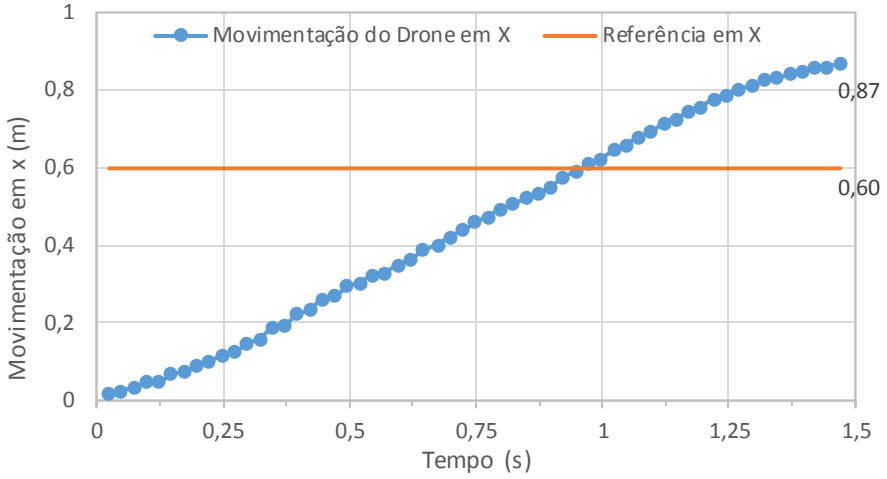

Figura 35 - Movimentação na direção x com comando de $0{,}60\ m$.

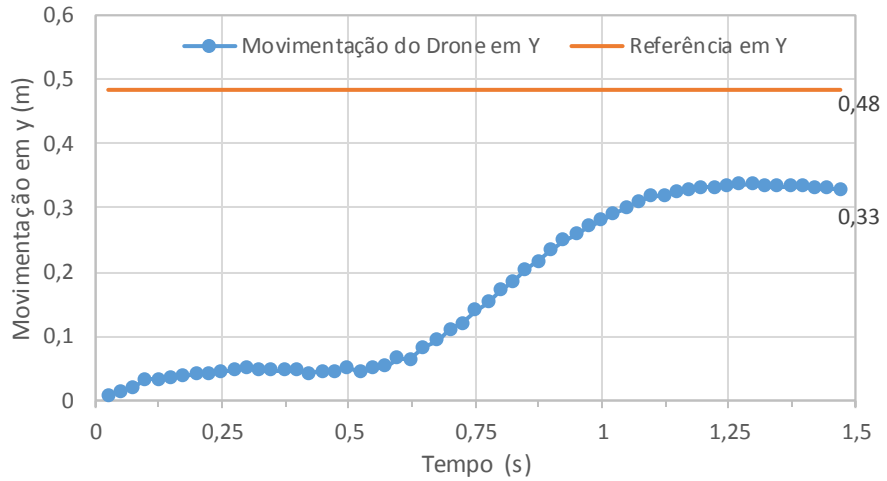

Figura 36 - Movimentação na direção y com comando de $0{,}8\ m$.

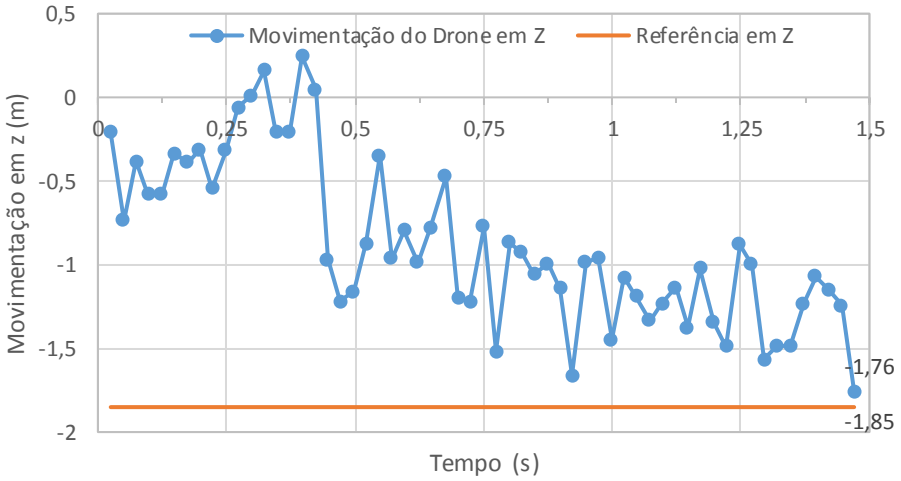

Figura 37 - Movimentação na direção z com comando de $-1{,}65\ m$.

Novamente, observamos que a movimentação em x ainda traz o maior erro porcentual, desta vez implicando em um avanço maior do que o necessário, o que colabora para justificar os comandos extras no sentido negativo de x para garantir maior precisão.

Como formas de prevenir e reduzir tais interferências destacam-se:

- Calibração prévia da aeronave: tanto de seu IMU (Unidade de Medição Inercial) quanto de sua bússola interna. A calibração demora em torno de 5 minutos e é realizada com auxílio do aplicativo DJI GO;
- Realizar um controle da velocidade, impondo limites para o fator de velocidade da aeronave durante a missão. Tal solução não foi aprofundada neste trabalho uma vez que o tempo de pouso também era um requerimento que gostaríamos de otimizar e a redução da velocidade age diretamente no aumento de tempo do pouso;
- Impor regras no algoritmo de modo a detectar esse distanciamento contínuo e mudar as regras de movimentação de modo a não gerar comandos que acumulem tantos erros de posicionamento. No caso deste trabalho, como mencionado na sessão 4.4.7, realizamos a criação de um ponto de retorno que coincidia com o último ponto onde a aeronave foi capaz de detectar o alvo.

6.3 Análise de Pousos com Precisão Insatisfatória

Tendo em vista o objetivo principal deste trabalho em garantir o aumento na precisão do voo da aeronave para uma área de cerca de $1\,m^2$, os pousos em que a aeronave realizou um pouso com mais de $1\,m$ de distância do centro do alvo foram considerados falhos.

Tais erros devem-se principalmente a manobras bruscas em que o drone acumulava movimentos de rotação com translação e, portanto, o erro estacionário da posição era muito elevado. Por vezes o drone acabava por romper o limite de altura mínima e

iniciava a fase final de pouso monitorado. Esses acúmulos foram aqui atribuídos a uma demora no processamento do comando devido ao Raspberry Pi Zero.

O limite de altura foi definido como 2,25 metros. A definição deste parâmetro foi dada através de medições empíricas onde observamos que o drone era capaz de manter uma precisão aceitável tendo em vista que o alvo já estava centralizado. É importante apontar que o aumento deste parâmetro pode resultar diretamente no aumento da imprecisão, pois, durante a fase de pouso monitorado, o drone é totalmente suscetível a movimentações devido ao vento e, assim, pode pousar distante do local onde o pouso foi iniciado.

6.3.1 Implantação de Pousos por Comando Único

Tendo em vista a busca por pousos velozes, foi considerada a possibilidade de fazermos pousos a partir de um comando único, o que significa que, uma vez que detecção do alvo fosse bem sucedida, poder-se-ia considerar um comando único que, além de centralizar a aeronave com o alvo, também a levaria até 1,5 m de altura, podendo assim dar-se início à fase de pouso monitorado.

Sendo assim, 3 experimentos foram executados utilizando esse método e os resultados são mostrados na Tabela 10.

Tabela 10 - Precisão dos voos por comando único.

Voo	Precisão (m)
1	1,46
2	1,36
3	1,02

Tais resultados evidenciam a necessidade de uma multiplicidade de comandos. Primeiramente, pois movimentos únicos tendem a potencializar erros do GPS e geram pousos imprecisos. Em segundo lugar, movimentos únicos envolvem movimentação em

um vetor com componentes em x, y e z não nulos, e, devido à possibilidade de obstáculos, tal método de controle pode levar a colisões.

A Tabela 11, a Figura 38, a Figura 39 e a Figura 40 evidenciam a potencialização dos erros de GPS, no caso do voo 2 mencionado acima, apesar da redução considerável no tempo dos comandos.

Tabela 11 - Erros finais para um comando único (voo número 2).

Erro em x	97%
Erro em y	51%
Erro em z	1%

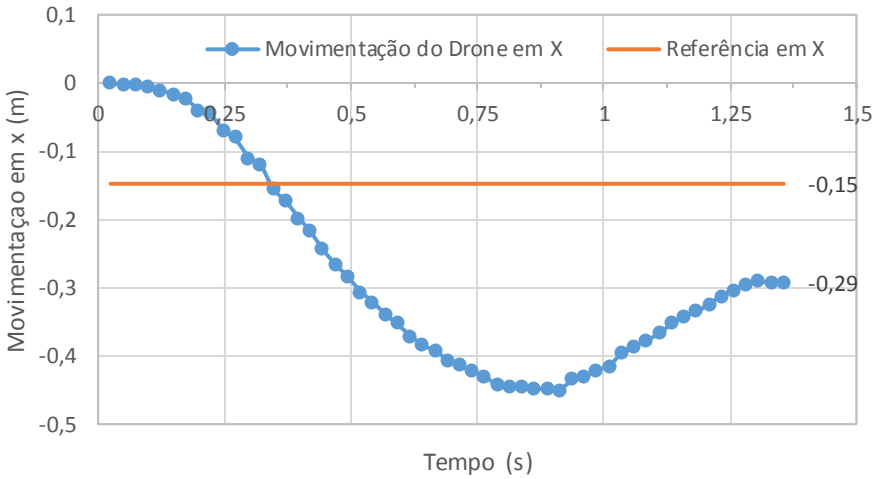

Figura 38 - Movimentação na direção x com comando de $-0{,}15\ m$.

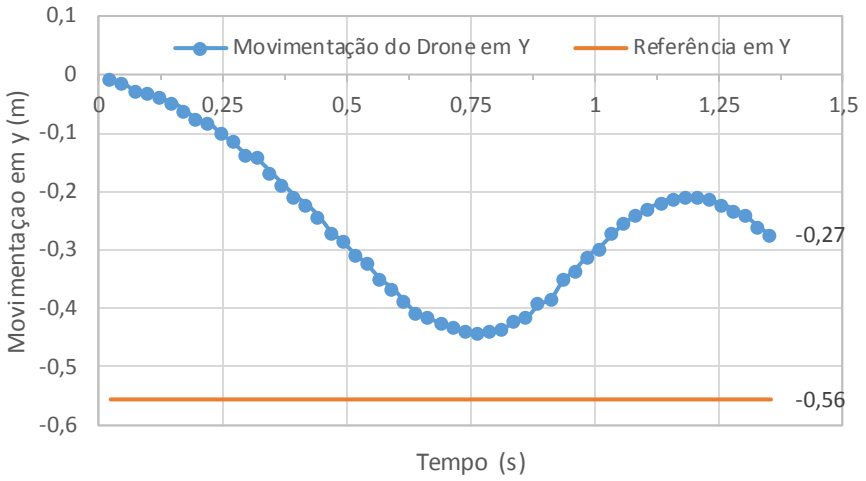

Figura 39 - Movimentação na direção y com comando de $-0{,}56\ m$.

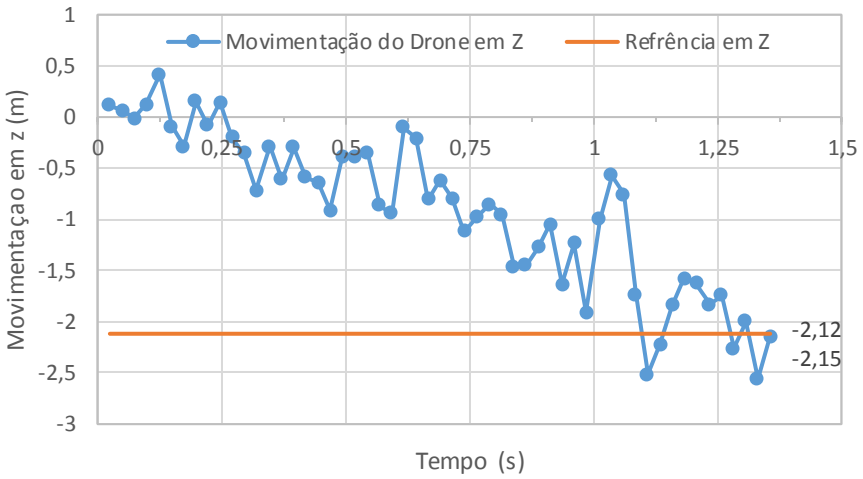

Figura 40 - Movimentação na direção z com comando de $-2{,}12\ m$.

6.4 Análise de Pousos com Sucesso

6.4.1 Validação Estatística dos Resultados

A Tabela 12 mostra os resultados gerais, calculados a partir dos dados dos voos bem sucedidos. A distância média encontrada foi de $0{,}508\ m$, o que corresponde a uma área circular média de $0{,}81\ m^2$.

Tabela 12 - Resultados gerais dos voos bem sucedidos

Quantidade média de imagens	3,37	
Quantidade média de movimentos	5,57	
Distância média de precisão	0,51	m
Desvio padrão de precisão	0,26	m
Área média circular	0,81	m^2
Tempo médio de pouso	51	s
Taxa de sucesso	73,1	%

Um teste de hipótese foi realizado para verificarmos a confiabilidade da hipótese de que teríamos um pouso com distância média de $0{,}5\ m$, o que implica em uma área circular máxima de $0.79\ m^2$. O teste de hipótese forneceu com $92{,}7$ % de confiabilidade a hipótese de uma distância média de $0{,}5\ m$, tendo sido aprovado com relevância estatística com confiabilidade de até 99 %. A Tabela 13 fornece os intervalos de confiabilidade com as respectivas áreas circulares máximas e mínimas para cada porcentagem do teste realizado.

Tabela 13 - Teste de confiabilidade

Confiabilidade	Intervalo de Pouso (m)		Área Mínima	Área Máxima
	Mínimo	Máximo	(m^2)	(m^2)
10%	0,40	0,61	0,51	1,17
5%	0,38	0,63	0,45	1,26
1%	0,33	0,68	0,35	1,45

6.4.2 Oportunidades de Avanços nos Experimentos

A taxa de sucesso obtida foi de $73{,}1$ %, o que correspondeu a 19 dos 26 voos executados. Alguns fatores colaborariam para o aumento desse índice, dentre os quais destacamos para experimentos futuros:

- Voo em ambiente com luminosidade controlada.

Tendo em vista que o pilar central do pouso assistido por imagens reside no reconhecimento veloz da imagem-guia é essencial que essa imagem fique exposta ao menor numero de agentes de interferência externa como luz e sombra excessiva.

- Estudo dinâmico detalhado de movimentos específicos do drone a fim de complementar a execução do tópico de movimentação do drone.

Atualmente, o movimento do drone vem da publicação de 4 variáveis – deslocamento em x, y e z e rotação em graus em torno do seu eixo próprio - em um tópico da classe de controle do drone. Essa movimentação é feita primariamente com cálculos referentes a dados oriundos do GPS, o que, para movimentos de baixa amplitude, podem levar aos erros mencionados anteriormente. Portanto, o estudo de peculiaridades inerciais do drone pode ajudar a desenvolver algoritmos paralelos que complementem os erros gerados pela movimentação exclusivamente pelo GPS.

- Implementação de um algoritmo por controle de velocidade.

Nos experimentos aqui conduzidos o drone se locomovia com controle de posição, contudo existe dentro da classe de controle do drone a possibilidade de enviar comandos de velocidade. Tais tipos de controle fornecem respostas mais rápidas e movimento mais fluido da aeronave. Contudo, demandam um estudo detalhado da inércia e tempo de resposta da aeronave, e, assim, demandam um fluxo contínuo de imagens pela câmera para rápida alteração nos sinais de velocidade. Tendo em vista que a câmera utilizada aqui atuava de forma discreta e, portanto, tinha um tempo de amostragem não desprezível, optamos por estudar os ganhos com o controle por posição.

- Voo em ambiente livre de rádio interferência.

O local no qual os testes foram realizados ficava nas vizinhanças de uma antena de telecomunicação, o que implicava em uma perturbação intensa na comunicação entre o Raspberry Pi Zero e os computadores remotos. Nos eventos de ruptura da

comunicação, o drone tem um sistema de segurança para romper comunicação permanentemente até que seja pousado manualmente.

- Utilização de um processador embarcado mais potente.

A velocidade do feedback por imagens em um pouso assistido por imagens garante melhor embasamento na geração de comandos para movimentação. Contudo, essa velocidade está diretamente associada a capacidade de processamento do processador embarcado. Nos testes executados, o Raspberry Pi Zero foi suficiente para execução da lógica proposta. Porém, por muitas vezes, notamos lentidão no processamento de dados, bem como rompimento da comunicação via sinal Wi-Fi. A presença de um computador embarcado mais potente pode aumentar a flexibilidade de implantação da mesma rotina, de modo que proporcione pousos ainda mais precisos nas mesmas condições dos voos atuais.

6.5 Considerações Sobre os Experimentos

Em análise global dos dados levantados, temos que as hipóteses propostas na concepção do projeto aqui desenvolvido foram cumpridas. Uma vez que concebemos e implementamos com sucesso uma rotina de pouso assistido por imagens que garante uma precisão em torno $0{,}9\ m^2$ para área de aterrisagem, e tal desenvolvimento foi feito por meio da embarcação de um Raspberry Pi Zero e uma RaspiCam, ambos hardwares de baixo custo.

7. Conclusão

O projeto aqui desenvolvido teve como objetivo o aumento da precisão de pouso de um drone comercial, o DJI Matrice 600, por meio da implementação de uma rotina de computação visual que permitisse o redirecionamento do drone por imagens-guia no momento da aterrisagem.

O projeto foi concebido com a hipótese de que a área precisão do pouso após a implantação do algoritmo de computação visual seria igual ou inferior a $1\,m^2$. Projetamos ainda que tal implantação poderia ser feita através da embarcação de um microprocessador de baixo custo.

A análise da literatura nos permitiu entrar em contato com diferentes técnicas de pouso para drones, desde trabalhos que consistiam no uso de sonares para detecção da região de pouso até a metodologia utilizada neste projeto, chamada de IBVS (Image-Based Visual Servoing), na qual o drone atua como um servo a partir de informações extraídas de imagens capturadas por uma câmera embarcada.

A fim de assegurar a implementação efetiva dessa estratégia para o drone, desenvolvemos, em paralelo, rotinas de movimentação do drone a partir de um comando de posição pré-definido e as rotinas de detecção de um alvo.

A revisão de hardware nos permitiu coletar revisões sobre o uso de plataformas embarcadas em drones, de modo a confirmar a viabilidade da utilização da linha de processadores da Raspberry Pi para processamento de imagens. O trabalho aqui descrito coloca-se ainda como uma contribuição original para o meio, a medida em que o processador utilizado é o Raspberry Pi Zero, caracterizado por uma capacidade de processamento mais reduzida. Na análise literária realizada, não encontramos nenhuma referência ao uso deste processador em conjunto com o Matrice 600 Pro.

Adquirimos ainda uma câmera, a Raspberry Pi Camera ou RaspiCam, a fim de coletarmos as imagens, uma vez que a câmera acoplada ao drone, a Zenmuse Z3 não possuia, até o momento da publicação deste trabalho, fluxo de imagens para o controlador A3, o qual temos acesso.

O software de simulação utilizado, o DJI Assistant 2.0, foi essencial para compreensão da lógica a ser implementada para a movimentação da aeronave, possibilitando a confecção das rotinas de movimento do drone tanto por meio de comandos incrementais quanto por comandos absolutos, i.e. latitude, longitude e altura.

A biblioteca do OpenCV foi o alicerce para o desenvolvimento da rotina de computação visual na linguagem C++. Com a finalidade de verificar a sinergia das rotinas de movimento e computação visual, um simulador próprio em C++ foi desenvolvido de modo a exemplificar a eficácia do método em um ambiente virtual único.

Os testes em campo elucidaram perturbações atenuadas nos testes virtuais, como variações no sistema de movimentação devido a erros pelo GPS, onde observou-se a tendência do drone em transladar no sentido positivo de x a cada movimento, propagando um erro de *offset* na direção horizontal. Tal erro já havia sido observado em outros trabalhos na literatura para este mesmo drone em questão. Levantamentos próprios revelaram que média de erros associados a movimentações na horizontal eram acima de 50 % e, para deslocamentos verticais, eram abaixo de 10 %. Tais erros eram exacerbados na medida em que a amplitude dos movimentos era menor. Como forma de contornar essa imprecisão foram adicionados movimentos extras no código, o que indica que a utilização do algoritmo em outro ambiente ou outra aeronave demandaria uma nova análise do comportamento dinâmico e possível alteração desses parâmetros.

Os testes em campo revelaram, ainda, falhas de comunicação entre o Raspberry Pi e computadores remotos, devido a perdas dos sinais. Tais perdas foram atribuídas à capacidade de processamento reduzida do computador embarcado e a presença de uma antena de rádio nas proximidades do campo de testes.

Os resultados foram coletados a partir da execução de uma série de voos documentados que consistiam em uma movimentação autônoma por comandos absolutos até uma posição pré-estabelecida, P7, seguido da execução do algoritmo de IBVS desenvolvido. De forma global, os resultados mostraram um média de 3,31

imagens coletadas e 5,5 movimentos por pouso. Tais dados atuam como linha de referência nos trabalhos futuros de otimização do algoritmo aqui desenvolvido de forma a reduzir ainda mais o tempo médio de pouso, hoje avaliado em $51\ s$.

Os testes demonstraram ainda a necessidade da implementação de manobras de busca para os casos em que o drone perde o alvo de vista. Tais manobras, apesar de aumentarem o tempo médio de voo, contribuem para a robustez da aplicação do algoritmo e fornecem mais segurança para a aeronave durante o pouso.

A extração da distância do centro do drone após o pouso até o centro do alvo possibilitou a composição dos dados totais dos voos que comprovaram que, com uma taxa de 73,1 % de sucesso, o drone pousou a uma distância média de $0{,}508\ m$, o qual corresponde a uma área de precisão circular média de $0{,}81\ m^2$.

Os resultados foram ainda submetidos a um teste de hipótese para verificação da relevância estatística, e foram aprovados com grau de confiabilidade de até 99 % com a probabilidade de 92,7 % de configurarem uma distribuição normal de média $0{,}5\ m$.

Sendo assim o trabalho aqui relatado cumpre de forma satisfatória os requisitos apresentados na hipótese inicial e corresponde a uma melhora de até nove vezes na precisão do pouso para o drone DJI Matrice 600 quando comparada com a precisão obtida com a rotina de pouso monitorado da aeronave.

A pesquisa aqui desenvolvida orienta-se de forma a prospectar as seguintes linhas futuras de estudo:

- União do pouso assistido por imagens com o sistema de deslocamento autônomo previamente desenvolvido.

Este projeto trata-se de uma continuação de um projeto iniciado em 2017 como uma parceria do Laboratório de Percepção Avançada da Escola Politécnica da Universidade

de São Paulo com o Hospital Israelita Albert Eisntein, que consiste em avaliar a viabilidade do uso de drones autônomos para o transporte de carga viva.

A primeira parte do projeto consolidou o sistema de deslocamento autônomo entre os pontos de partida e destino bem como gerou um sistema de comunicação entre as bases por meio de uma aplicação em Java. Sendo assim, o projeto atual deve ser incorporado pelo primeiro algoritmo de deslocamento de modo a substituir a rotina de pouso monitorado pela rotina de pouso assistido por imagens desenvolvida nesse trabalho.

- Realização de testes do sistema consolidado em novos locais.

A fim de avaliar a importância de perturbações externas na malha de controle, o sistema deve ser testado em outras localidades com diferentes níveis de luminosidade e elementos de interferência de modo a verificar a robustez do código e a necessidade de alteração na lógica desenvolvida.

- Verificação da possibilidade de obtenção de imagens pela câmera própria.

A DJI, empresa desenvolvedora do drone Matrice 600, lançou em 2018, em nota oficial, o comunicado do desenvolvimento de uma nova versão do SDK, no qual as aplicações de *Advanced-Sensing*, ou seja, acesso as imagens da câmera, seriam expandidas para a linha Matrice 600. Tal avanço permitiria a remoção da câmera externa e, provavelmente, tornaria possível uma captura de imagens mais rápida.

- Realização de estudo para novos computadores embarcados.

Apesar do Raspberry Pi Zero ter desempenhado um processamento suficiente para que as hipóteses do projeto fossem alcançadas, tem-se como projeção que a implantação de computadores embarcados mais potentes garantiria um pouso mais veloz sem comprometer a precisão alcançada.

8. Bibliografia

[1] M. Pavani, "Implementation and Analysis of a Model Predictive Controller for Landing Fixed-Wing Aircraft on Mobile Platforms," *DLR-Interner Bericht,* 2018.

[2] C. Rose, "Amazon's Jeff Bezos looks to the future," 2013. [Online]. Available: http://www.cbsnews.com/news/amazons-jeff-bezos-looks-to-the-future/.

[3] M. Khonji, M. Alshehhi, C. Tseng and C. Chau, "Autonomous Inductive Charging System for Battery-operated Electric Drones," *e-Energy,* 2017.

[4] M. Ferrandez, T. Harbison, T. Weber, R. Sturges and R. Rich, "Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm.," *Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm." Journal of Industrial Engineering and Management,* no. 9.2, pp. 374-388, 2016.

[5] S. Poikonen, X. Wang and B. Golden, "The vehicle routing problem with drones: Extended models and connections.," *Networks,* vol. 70, no. 1, pp. 34-43, 2017.

[6] U. Papa and G. del Core, "Design of sonar sensor model for safe landing of an UAV," *IEEE Metrology for Aerospace (MetroAeroSpace),* pp. 346-350, 2015.

[7] X. Zhou, Z. Lei, Q. Yu, H. Zhang, Y. Shang, J. Du, Y. Gui and P. Guo, "Videometric terminal guidance method and system for UAV accurate landing," *Proceedings SPIE ,* vol. 8387, no. Unmanned Systems Technology XIV, 2012.

[8] S. Lange, N. Sünderhauf and P. Protzel, "Protzel, P. (2008). Autonomous Landing for a Multirotor UAV Using Vision," *Workshop Proceedings of SIMPAR,* Vols. Intl. Conf. on SIMULATION, MODELING and PROGRAMMING for AUTONOMOUS ROBOTS, pp. 482-491, 2008.

[9] R. Qiu, X. Miao, S. Zhuang, H. Jiang and J. Chen, "Design and implementation of an autonomous landing control system of unmanned aerial vehicle for power line inspection," *Chinese Automation Congress ,* pp. 7428-7431, 2017.

[10] S. Saripalli, J. F. Montgomery and G. S. Sukhatme, "Vision-based Autonomous Landing of an Unmanned Aerial Vehicle," *Proceedings - IEEE International Conference on Robotics and Automation,* vol. 3, 2001.

[11] S. Battiato , G. Gallo, R. Schettini and F. Stanco , "A System for Autonomous Landing of a UAV on a Moving Vehicle," *Image Analysis and Processing,* 2017.

[12] DJI, "Matrice 600 Specs," [Online]. Available: https://www.dji.com/matrice600/info#specs. [Accessed 26 06 2018].

[13] Raspberry Pi Foundation, "Raspberry Pi Zero: The $5 Computer," 2015. [Online]. Available: https://www.raspberrypi.org/blog/raspberry-pi-zero/. [Accessed 06 11 2018].

[14] Oracle Corporation, "Java Programming Language," 2018. [Online]. Available: https://docs.oracle.com/javase/7/docs/technotes/guides/language/. [Accessed 06 11 2018].

[15] Oracle Corporation, "MySQL," 2018. [Online]. Available: https://www.mysql.com/. [Accessed 06 11 2018].

[16] OpenCV team, "OpenCV," 2018. [Online]. Available: https://opencv.org/. [Accessed 06 11 2018].

[17] DJI, "Mavic 2 Specs," [Online]. Available: https://www.dji.com/mavic-2/info. [Accessed 06 11 2018].

[18] C. Tseng, C. Chau, K. M. Elbassioni and M. Khonji, "Autonomous Recharging and Flight Mission Planning for Battery-operated Autonomous Drones," 2017.

[19] Digi-Key's European Editors, "Drone Docking with “Park Assist” for Recharging in The Field," 2016. [Online]. Available: https://www.digikey.com/en/articles/techzone/2016/nov/drone-docking-park-assist-recharging-in-the-field. [Accessed 06 11 2018].

[20] G. Xu, X. Chen, B. Wang, K. Li, J. Wang and X. Wei, "A search strategy of UAV's automatic landing on ship in all weather," *International Conference on Electrical and Control Engineering,* pp. 2857-2860, 2011.

[21] S. Maiman, M. Schulman and J. Pearlstein, "Matt; PEARLSTEIN, Josh. GOLD: GPS and Optic Landing of Drones: A Hybrid Approach.".

[22] J. Shang and Z. Shi, "Vision-based Runway Recognition for UAV Autonomous Landing," 2007.

[23] DJI, "A3 Specs," [Online]. Available: https://www.dji.com/a3/info. [Accessed 21 09 2018].

[24] T. G. McGee, R. Sengupta and K. Hedrick, "Obstacle detection for small autonomous aircraft using sky segmentation," *IEEE International Conference on Robotics and Automation,* p. 4679–4684, 2005.

[25] L. Meier, P. Tanskanen, F. Fraundorfer and M. Pollefeys, "Pixhawk: A system for autonomous flight using onboard computer vision," *IEEE international conference on Robotics and automation,* p. 2992–2997, 2011.

[26] S. Shen, Y. Mulgaonkar, N. Michael and V. Kumar, "Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor," *Robotics: Science and Systems,* 2013.

[27] C. Forster, M. Pizzoli and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," *Proc. IEEE Intl. Conf. on Robotics and Automation,* 2014.

[28] C. De Wagter, S. Tijmons, B. D. Remes and G. C. Croon, "Autonomous flight of a 20-gram flapping wing MAV with a 4-gram onboard stereo vision system," *IEEE International Conference on Robotics and Automation,* p. 4982–4987, 2014.

[29] D. Hulens , J. Verbeke and T. Goedemé , "Choosing the Best Embedded Processing Platform for On-Board UAV Image Processing," *Communications in Computer and Information Science,* vol. 598, 2016.

[30] DJI, "Onboard SDK: Hardware Setup," 2016. [Online]. Available: https://developer.dji.com/onboard-sdk/documentation/hardware-setup/index.html. [Accessed 06 11 2018].

[31] R. Barták, A. Hrasko and D. Obdrzalek, "A controller for autonomous landing of AR.Drone," *Chinese Control and Decision Conference,* pp. 329-334, 2014.

[32] Raspberry Pi Foundation, "Camera Module," 2016. [Online]. Available: https://www.raspberrypi.org/documentation/hardware/camera/. [Accessed 06 11 2018].

[33] N. Shamsee, D. Klebenov and H. Fayed, "CCNA Data Center," *Officieal Cert Guide,* p. 934, 2015.

[34] Gartner, "SDK (software development kit)," 2018. [Online]. Available: https://www.gartner.com/it-glossary/sdk-software-development-kit. [Accessed 05 07 2018].

[35] PuTTY, "Download PuTTY," [Online]. Available: https://www.putty.org/. [Accessed 21 09 2018].

[36] WinSCP, "WinSCP: Free SFTP, SCP and FTP client for Windows," [Online]. Available: https://winscp.net/eng/index.php. [Accessed 21 09 2018].

[37] Gartner, "publish/subscribe architecture," 2018. [Online]. Available: https://www.gartner.com/it-glossary/publishsubscribe-architecture. [Accessed 21 09 2018].

[38] Open Source Robotics Foundation, "ROS," [Online]. Available: http://www.ros.org/. [Accessed 21 09 2018].

[39] T. Villani and F. Lamberti, "Design and Implementation of Robotic Games based on Human-Drone interaction," *Grains Laboratory - Politecnico di Torino,* 2017.

[40] A. Bonarini, D. Martinoia and D. Calandriello, "Physically Interactive Robogames: Definition and Design Guidelines," *AI an dRobotics Lab- Department of Electronics, Information, and Bioengineering - Politecnico di Milano,* 2013.

[41] woekan, "Modern Tabletop/Virtual tabletop street map1," [Online]. Available: https://www.deviantart.com/woekan/art/Modern-Tabletop-Virtual-tabletop-street-map1-548381021. [Accessed 08 11 2018].

[42] D. D. Ekaso, "Accuracy Assessment of Real-Time Kinematics (RTK) Measurement on Unmanned Aerial Vehicles (UAV) for Direct Geo-Referencing," *Faculty of Geo-Information Science and Earth Observation of the University of Twente,* p. 52, 2018.