UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Mecânica

Alan Ferreira Pinheiro Tavares

Método Híbrido de Detecção e Estimativa de Posição de Marcador para Aterrissagem Autônoma de Quadricóptero

CAMPINAS
2021

Alan Ferreira Pinheiro Tavares

Método Híbrido de Detecção e Estimativa de Posição de Marcador para Aterrissagem Autônoma de Quadricóptero

Tese de doutorado apresentada à Faculdade de Engenharia Mecânica da Universidade Estadual de Campinas como parte dos requisitos exigidos para obtenção do título de Doutor em Engenharia Mecânica, na Área de Mecatrônica.

Orientador: Prof. Dr. Paulo Roberto Gardel Kurka.

ESTE TRABALHO CORRESPONDE À VERSÃO FINAL DA TESE DEFENDIDA PELO ALUNO Alan Ferreira Pinheiro Tavares, E ORIENTADO PELO PROF. DR. Paulo Roberto Gardel Kurka.

CAMPINAS
2021

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

T197m

Tavares, Alan Ferreira Pinheiro, 1989-
Método híbrido de detecção e estimativa de posição de marcador para aterrissagem autônoma de quadricóptero / Alan Ferreira Pinheiro Tavares. – Campinas, SP : [s.n.], 2021.

Orientador: Paulo Roberto Gardel Kurka.
Tese (doutorado) – Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Marcadores fiduciais. 2. Visão por computador. 3. Veículos autônomos. 4. Algoritmos computacionais. 5. Redes neurais convolucionais. I. Kurka, Paulo Roberto Gardel, 1958-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Hybrid marker position detection and estimation method for autonomous quadcopter landing
Palavras-chave em inglês:
Fiducial markers
Computer vision
Autonomous vehicles
Computational algorithms
Convolutional neural networks
Área de concentração: Mecatrônica
Titulação: Doutor em Engenharia Mecânica
Banca examinadora:
Paulo Roberto Gardel Kurka [Orientador]
Hélio Pedrini
Ely Carneiro de Paiva
Marcelo Rudek
Caio Fernando Rodrigues
Data de defesa: 08-01-2021
Programa de Pós-Graduação: Engenharia Mecânica

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0003-1644-9999
- Currículo Lattes do autor: http://lattes.cnpq.br/2333777207469549

UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA MECÂNICA

TESE DE DOUTORADO ACADÊMICO

Método Híbrido de Detecção e Estimativa de Posição de Marcador para Aterrissagem Autônoma de Quadricóptero

Autor: Alan Ferreira Pinheiro Tavares
Orientador : Paulo Roberto Gardel Kurka

A Banca Examinadora composta pelos membros abaixo aprovou esta Tese:

Prof. Dr. Paulo Roberto Gardel Kurka, Presidente
DEMA/FEM/UNICAMP

Prof. Dr. Hélio Pedrini
IFSP/BRAGANÇA PAULISTA

Prof. Dr. Ely Carneiro de Paiva
DPM/FEM/UNICAMP

Prof. Dr. Marcelo Rudek
DPM/FEM/UNICAMP

Prof. Dr. Caio Fernando Rodrigues
DPM/FEM/UNICAMP

A Ata de defesa com as respectivas assinaturas dos membros encontram-se no SIGA/Sistema de Fluxo de Tese e na Secretaria do Programa da Unidade.

Campinas, 08 de Janeiro de 2021

Dedicatória

Aos meus queridos pais, Ana Tavares e Lenir Tavares, minha estimada irmã Luana Tavares. Ao meu tio Manoel José Tavares e minha querida prima Manoela Tavares, pelos quais sinto muito orgulho e gratidão por todo carinho, incentivo e apoio que sempre me ofereceram. Este trabalho foi desenvolvido com o apoio do Governo do Estado do Amazonas por meio Fundação de Amparo à Pesquisa do Estado do Amazonas - FAPEAM, com a concessão de bolsa de estudo.

Agradecimentos

Ao meu orientador, Prof. Dr. Paulo Roberto Gardel Kurka, pelo apoio, orientações estabelecidas e pela grande parceria durante estes anos.

A toda minha família e amigos de laboratório que me seguiram nessa jornada de pós-graduação.

A Universidade Estadual de Campinas, a Universidade do Porto e aos seus professores e colaboradores.

A Connect Robotics, pelo suporte dado durante meu intercâmbio na cidade do Porto em Portugal.

A Acta Robotics, pelo suporte dado durante os estudos e experimentos realizados com o drone fornecido pela empresa.

Este trabalho foi desenvolvido com o apoio do Governo do Estado do Amazonas por meio Fundação de Amparo à Pesquisa do Estado do Amazonas - FAPEAM, com a concessão de bolsa de estudo.

Eis que o ser humano é um ser singular e imperativo, então olháreis para além das nuvens. Só assim teremos novos sonhos e perspectivas, resguardados de infinitas possibilidades de um futuro concreto, que um dia foi um sonho.

Alan Tavares

Resumo

Tavares, Alan Ferreira Pinheiro. Método Híbrido para Detecção e Estimativa de Marcador para Aterrissagem Autônoma de Quadricóptero. 2021. 81p. Tese (Doutorado). Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, Campinas.

O uso de Veículos Autônomos Aéreos Não Tripulados (VAANTs) para aplicações comerciais inovadoras vem ganhando notoriedade, com isso, torna-se essencial o desenvolvimento de métodos avançados de localização e navegação robótica. Marcadores visuais como as *April Tags* ou *ArUco* destacam-se como técnicas de localização mais utilizadas na robótica por terem uma abordagem de detecção robusta utilizando somente recursos de visão computacional. Contudo, existem limitações de detecção e estimação de posição em casos de longas distâncias, oclusão parcial ou uso de câmeras com alta distorção, condições facilmente encontradas, por exemplo, em situações de pouso e decolagem autônoma de. Logo, os sistemas atuais necessitam de alta tecnologia de sensoriamento como IMU, GPS ou *Laser* que, dependendo de sua precisão, podem resultar em custos elevados. Estes sensores de navegação podem apresentar também erros da ordem de metros e de forma cumulativa, principalmente no que diz respeito a pousos ou decolagens utilizando GPS próximos a prédios, redes elétricas ou montanhas que bloqueiam ou anulam completamente o sinal. A presente tese traz como solução alternativa um método capaz de identificar a posição e fornecer informações de localização do robô para casos de pouso autônomo de VAANTs do tipo quadricóptero, em um alvo estático de identificação única. O método proposto utiliza uma detecção precisa de forma a diminuir a limitações do estado da arte. O mesmo método utiliza apenas sensoriamento de visão a bordo e algoritmo computacional sem necessitar de nenhuma informação prévia sobre a localização da plataforma de pouso ou qualquer infraestrutura externa.Dentro do estado da arte, o método se mostra único por utilizar técnicas de rede neural convolucional (CNN) de forma a complementar a técnica clássica das *tags* do tipo *ArUco*. O método híbrido realiza a fusão das duas técnicas usando filtro de Kalman, tornando a tarefa de detecção e estimativa de posição mais eficiente, com resposta de atuação mais rápida. Cria-se nessa tese um banco de dados *(dataset)* de imagens sintéticas e reais para o treinamento das redes neurais. O método proposto é verificado através de simulações numéricas nas plataformas robóticas *ROS* e *Gazebo*. Por fim, é validado através de experimentos reais qualitativos e quantitativos.

Palavras-chave: Pouso Autônomo, Marcador Fiduciário, Visão Computacional, Redes Neurais.

Abstract

Tavares, Alan Ferreira Pinheiro. Hybrid Marker Position Detection and Estimation Method for Autonomous Quadricopter Landing. 2021. 81p. Tese (Doutorado). Faculdade de Engenharia Mecânica, Universidade Estadual de Campinas, Campinas.

The use of Unmanned Aerial Vehicles (UAVs) for innovative commercial applications has been gaining notoriety, thus, it is essential to develop advanced methods of location and robotic navigation. Visual markers such as the April or ArUco tags stand out as the most widely used robotic location techniques in robotics because they have a robust detection approach using only computer vision resources. However, there are limitations of position detection and estimation in cases of long distances, partial occlusion or use of high distortion cameras, conditions easily found, for example, in situations of UAV to landing and takeoff. Therefore, current systems require high sensing technology such as IMU, GPS or Laser, which, depending on their accuracy, can result in high costs. These navigation sensors can also show errors in the order of meters and cumulatively, especially with regard to landings or takeoffs using GPS near buildings, electrical networks or mountains that block or completely cancel the signal. The present thesis proposes as a solution, a method capable of identifying and providing robot location information for cases of autonomous landing of UAVs of the quadricopter type, on a static single identification target. The thesis proposes an accurate detection in order to overcome state of the art limitations. Only on-board vision sensing and computational algorithm are used without requiring any prior information on the location of the landing platform or any external infrastructure. Within the state of the art, the method is unique in that it uses convolutional neural network (CNN) techniques in order to complement the classic technique of ArUco type tags. The hybrid method fuses the two techniques using a Kalman filter, making the task of position detection and estimation more efficient and the response faster. In this thesis a database (dataset) of synthetic and real images is created for the training of neural networks. The proposed method is verified through numerical simulations on the ROS and Gazebo robotic platforms. The proposed method is validated in this work through real qualitative and quantitative experiments.

Key words: Autonomous Landing, Fiduciary Marker, Computer Vision, Neural Networks.

Lista de Ilustrações

Lista de Tabelas

SUMÁRIO

1 Introdução

1.1 Motivação

O uso de Veículos Autônomos Aéreos Não Tripulados (VAANTs), também conhecidos como *drones*, quadricópteros, ou quadrotores autônomos, vêm ganhando notoriedade nas últimas décadas devido à sua alta versatilidade, manobrabilidade, avanço tecnológico computacional, seguido também de aplicações inovadoras em quase todos os setores, sejam elas com propósito militar, de pesquisa, comércio, vigilância ou de lazer. Justifica-se também este interesse por conta do crescente número de fabricantes, redução de preços, melhoria da qualidade dos componentes e sensores envolvidos, demandando assim técnicas mais robustas para garantir a segurança das aeronaves (SANTOS, M. F. DOS, 2014).

Segundo definições de Duan et al. (2010), VAANT é um tipo de sistema muito complexo com diferentes componentes de hardware, como o Sistema de Posicionamento Global (GPS), Unidade de Medida Inercial (IMU), controlador e diferentes componentes de software, para processamento de imagem, planejamento de trajetória, algoritmos de estabilização, navegação e decolagem e pouso. Costa (2012) destaca a importância da IMU no sistema de localização dos VAANTs que integra bússola, barômetro, sonares e sensor GPS. Tanto IMU quanto a bússola indicam a atitude; Barômetro e sonar podem ser usados para leitura de altitude, e o GPS para obter dados de posicionamento e também altitude.

Contudo, estes sensores de navegação podem apresentar erros da ordem de metros e de forma acumulativa, principalmente quando se diz respeito a pousos ou decolagens utilizando GPS próximos a prédios, redes elétricas ou montanhas que bloqueiam ou anulam completamente o sinal de GPS. Este fato deprecia a qualidade, confiabilidade e custo da missão, visto que se faz necessário a utilização de infraestrutura externa ou aquisição de sensores com altíssima precisão e com blindagem eletrostática específica, resultando em custo elevado.

Logo, a visão computacional é uma excelente solução de baixo custo e alto retorno de informações espaciais para o controle de VAANTs, principalmente no que se diz respeito a sistemas de decolagem e pouso autônomo que é praticamente usando em qualquer tipo de missão com VAANT.

As câmeras de sensor exteroceptivo ou monocular são apresentadas com uma ótima solução para navegação e pouso, além de atribuir precisão em navegação possui a vantagem de ser um sensor passivo, de baixa sensibilidade a ruídos e independente do sinal de GPS, permitindo voo em ambientes onde esse sinal não possui boa qualidade (GUO ET AL., 2014). Pode-se citar algorítimos de código aberto como de Forster et al. (2014) e Mur-Artal e Tardós (2016), que trazem alternativas para soluções de navegação visual monocular através de uma abordagem semi-direta que elimina a necessidade de técnicas custosas computacionalmente para estimativa de posição e orientação.

Destaca-se ainda o avanço das pesquisas na área de redes neurais artificiais, as quais vêm agregando benefícios computacionais de identificação de objetos com uma precisão elevada. Pode-se citar o trabalho de Kim e Chen (2015) como referência, o qual usa as redes neurais para classificação e navegação de cenas internas, utilizando uma rede neural convolucional (CNN) para aprender uma estratégia de controle baseada em dados capturados de vôos realizados por um piloto especialista humano.

Uma das aplicações dos estudos desta tecnologia está presente no setor de encomendas *(delivery)* utilizado por serviço de entregas. Onde a Agência Nacional de Aviação Civil (Anac) autorizou o início da atividade de entregas via *drone* de produtos comprados pela internet ou por telefone. Por enquanto, o sistema funciona em formato de teste, e apenas uma empresa foi liberada pela agência: a brasileira *Speedbird*. Contudo, existem limitações impostas pela Anac para a realização desses testes. Ao passo que o operador não precisará ter o veículo em seu campo de visão o tempo todo, o raio de funcionamento desse formato será de 2,5 km. Além disso, os colaboradores da *Speedbird* precisarão seguir as regras já definidas para pilotagem no Regulamento Brasileiro de Aviação Civil Especial (RBAC-E) n. 94 da Anac, segundo notícias de Agosto de 2020 (MÜLLER, 2020).

Outra consideração importante feita por Chavez (2016), destaca que em praticamente todos os tipos de aplicações de VAANTs tem a carga útil como o grande fator limitante para alta autonomia. A quantidade de baterias transportadas durante uma missão, restringe o tempo de voo. Sendo assim, os veículos voadores precisam retornar a uma estação de recarga após um curto período de operação. Neste contexto, a aterrissagem autônoma torna-se essencial para economizar energia. Além disso, ao realizar tarefas em que seja necessário um voo longo contínuo ou viagens de longa distância, o retorno a uma estação base não é prático. Nesses cenários, seria mais conveniente implantar uma estação base móvel capaz de transportar energia suficiente para vários ciclos de recarga. Os VAANTs, então, teriam que realizar pouso periódico autônomo em estações de ancoragem baseadas em terra ou voadoras para recarregar e se tornar operacional novamente.

1.2 Objetivo

O presente trabalho tem como objetivo desenvolver um procedimento robusto de orientação ao pouso autônomo, de um veículo aéreo não tripulado do tipo quadricóptero. A tese propõe uma detecção e estimação de pose eficaz que supera limitações de longa distância e oclusão parcial do alvo. O trabalho pressupõe a utilização única de sensoriamento por visão para a realização da tarefa de pouso. As etapas descritas a seguir correspondem às principais ações envolvidas para a obtenção do objetivo proposto.

- Levantamento bibliográfico sobre os principais conceitos e desafios de pesquisa referentes a pouso de precisão autônoma utilizando veículos quadrotores;
- Embarque e estudo das bibliotecas de *OpenCV*, *ArUco* e *Tensorflow* vinculados a linguagem de programação C/C++ ou *Python* integradas a platadorma *ROS - Robot Operating System*;
- Estruturação de um dicionário de marcador fiducial que apresente bom desempenho no quesito identificação precisa do marcador e confiabilidade de posição;
- Parametrização de um algorítimo de detecção fiducial que tenha como saída a posição e orientação do marcador de pouso em relação ao quadrotor, utilizando a técnica *ArUco* como base;
- Utilização dos Simuladores *Gazebo* ou *V-REP* para integrar a comunicação de controle do quadrotor com o algorítimo desenvolvido e realizar testes simulados e integrados via *ROS*;
- Construção de base de dados sintéticos de treinamento de rede neural;
- Construção de base de dados reais de treinamento de rede neural;
- Realização de treinamento dos dados obtidos para identificar o marcador desejado utilizando técnica de Rede Neural Convolucional (CNN);
- Desenvolvimento um estimador de posição e orientação para o quadricóptero através de fusão sensorial utilizando filtros matemáticos como *Outlier* e *Butterworth*;
- Determinação dos melhores parâmetros visando eficiência na identificação de marcadores a longa distância, oclusões parciais ou até mesmo distorção de câmera;

1.3 Estado da Arte

Neste tópico, será abordado uma revisão da literatura sobre as principais técnicas utilizadas para pouso de precisão para VAANT do tipo quadrotor, bem como, a revisão literária do uso de marcadores fiduciais, abordando os principais trabalhos que utilizam técnicas para detecção de marcadores visuais como solução do problema de localização. Priorizou-se técnicas que utilizaram somente visão computacional a fim de criar um estado da arte mais próximo do proposto pela tese, a qual visa contornar erros inerentes a sensores de posicionamento como IMU e GPS ou sensores ativos como lasers e sonares.

1.3.1 Revisão Literária das Principais Técnicas de Pouso

Inicialmente é realizada uma revisão das principais técnicas de decolagem e pouso, a qual é exposta de forma resumida na Tabela 1.1, onde se destaca a principal abordagem e contribuição científica de cada referência. Em seguida, realiza-se uma análise comparativa destes trabalhos em relação ao trabalho proposto e os principais desafios superados por cada técnica, sendo o principal objetivo deste tópico.

A literatura sugere que a maioria das implementações de técnicas para pouso disponíveis podem ser classificadas em 2 (duas) categorias: A primeira com sistemas que fazem uso de sensores de imagem e a segunda com sistemas cujas capacidades de navegação são baseadas em sensores ativos, como lidar, sonares em casos de ambientes internos e IMU e GPS em ambiente externo.

Destaca-se inicialmente o trabalho de Falanga et al. (2017), que traz uma abordagem prática, contando com um algorítimo de visão computacional avançado. Diferente da tese proposta, o trabalho de Falanga et al. (2017) propõe para pouso uma fusão de multi-sensores para localização do veículo aéreo além de detecção e estimativa de movimento da plataforma móvel usando uma segunda câmera. O algorítimo de detecção visual da plataforma móvel é aplicado em um filtro de *Kalman* atrelado ao modelo cinemático da plataforma, resultando em uma identificação da plataforma móvel com alta frequência.

Tabela 1.1: Resumo das principais abordagens e contribuições científicas relacionadas à decolagem e pouso de VANTs usando somente visão computacional.

Abordagem	**Contribuição**	**Referências**
Odometria Monocular Estimação da Plataforma	Sistema de Pouso Autônomo em Plataforma Móvel	Falanga et al. (2017)
Controlador Back Propagation	Rede Neural Como Manobra de Pouso	Ananthakrishnan et al. (2017)
Orientação de Navegação Intrínseca de Tau	Trajetórias de Pouso Baseado Espaço-Temporal	Vetrella et al. (2018)
Servo Visual com Ultrassônico Fusão Filtro de Kalman	Pouso Autônomo Rastreamento de Objeto	He et al. (2019)
Aprendizagem por Reforço Profundo e Rede (DDPG)	Insights dos Controles Estados da Aeronave	Tang e Lai (2020)

Ao compararmos o trabalho de Falanga et al. (2017) com a presente tese, verifica-se que a estratégia de detecção do marcador visual não utiliza redes neurais, logo, o sistema de identificação do marcador de pouso é somente eficaz quando se tem uma captura de imagem perfeita, sem oclusões. A principal contribuição do trabalho de Falanga et al. (2017) são os resultados práticos, que diferente da presente tesa, visa um pouso em uma plataforma móvel, ilustrado na Figura 1.1.

Figura 1.1: Quadrotor pousando em uma plataforma móvel, (FALANGA ET AL., 2017).

Destaca-se alguns trabalhos relevantes como de Polvara et al. (2017), Ananthakrishnan et al. (2017) e Rodriguez-Ramos et al. (2019). Assim como a presente tese, usam. *Convolutional Neural Network* (CNN) para classificação de imagem, contudo, com uma ênfase em controle de navegação através de cenas internas de voo. A rede é treinada para aprender uma estratégia de controle, logo são baseadas em voos realizados por um piloto humano especializado.

No trabalho de Vetrella et al. (2018), desenvolveu-se um algoritmos flexível de geração de trajetória, sua contribuição científica permitiu altos níveis de autonomia para as fases críticas da missão, como decolagem, cobertura de área e pouso. Diferente da tesa apresentada, o trabalho de Vetrella et al. (2018) apresenta uma abordagem de orientação que usa a teoria de orientação intrínseca da *Tau* aprimorada para criar trajetórias espaço-temporais em 4D para um tempo-para-contato desejado com uma plataforma de aterrissagem rastreada por um sensor visual.

O trabalho de He et al. (2019) aborda o problema de rastreamento e aterrissagem em um objeto arbitrário de um quadrotor. Desenvolveu-se um método de servo visual baseado em imagem que usa apenas uma câmera voltada para baixo para rastrear e pousar de um quadrotor. O algoritmo de detecção visual permite ao usuário especificar um objeto arbitrário para detectar seus movimentos no plano da imagem.

No plano horizontal, usa-se o controlador de velocidade lateral para manter o rastreamento com o objeto no centro da imagem da câmera. Uma vez que o quadrotor satisfaça a condição de pouso, ele segue para a fase de pouso na velocidade vertical planejada. Diferente da presente tese, He et al. (2019) usa um sensor de ultrassom executando o filtro kalman para detectar a condição de pouso e nos experimentos, mostrou-se que o sistema é capaz de rastrear um objeto arbitrário e pousar nele com sucesso.

Ainda referente a trabalhos com pouso autônomo, destaca-se o trabalho Tang e Lai (2020) que aborda sobre a problemática na fase de pouso continua, sendo uma das tarefas mais cruciais e difíceis de atingir. A pesquisa implementou o uso de DDPG *(Deep Deterministic Policy Gradient)*, uma abordagem DRL *(Deep Reinforcement Learning)* na tentativa de encontrar políticas de pousos de aeronaves dadas os requisitos projetados, ou recompensas. Verifica-se uma contribuição científica diferente da abordada na presente tese, visto que o trabalho de Tang e Lai (2020) focou em desenvolver políticas de controle para pousos, fornecendo *insights* dos controles e estados da aeronave durante o pouso, gerando orientações sobre pouso para pilotos ou projetos de controladores.

1.3.2 Revisão Literária do Uso de Marcadores Fiduciais como *Landmarks*

A seguir, uma revisão literária sobre o uso de marcadores fiduciais como marcadores de localização (*Landmarks*), abordando os principais trabalhos que utilizam técnicas para detecção de marcadores visuais como solução do problema de localização. Contudo, evidencia-se graves problemas de identificação em condições não normais de captura de imagem, como no caso de oclusões ou forte distorções e problemas de contraste de luz.

A principal motivação de usar sensores de imagem é seu peso leve e baixo consumo de energia. Para este fim, a literatura sugere várias abordagens que utilizam marcadores visuais para simplificar a problema de localização, como exemplo, têm-se o trabalho de Pestana et al. (2016), que faz uso de marcadores visuais *ArUco* de Munoz-Salinas (2012), a abordagem proposta por Jayatilleke e Zhang (2013) que requer um conhecimento prévio de todas as poses de referência, enquanto que em Faigl et al. (2013), o VAANT é necessário para manter os marcadores externos sempre visíveis. Nessas abordagens, o uso de marcadores visuais fornece uma localização precisa, porém com detecção limitada em imagens borradas e com longa distância.

Existem vários tipos de marcadores, cada um deles pertencente a um dicionário. Cada biblioteca propôs seu próprio conjunto de marcadores. Então, temos *ArToolKit+*, *Chilitags*, *AprilTags* e, claro, o dicionário *ArUco*. O *design* de um dicionário é importante, já que a ideia é que seus marcadores sejam o mais diferentes possíveis para evitar confusões. Um sistema de referência fiducial é composto por um conjunto de marcadores válidos e um algoritmo que realiza sua detecção e possivelmente correção em imagens. Vários sistemas de marcadores fiduciais têm sido propostos na literatura como mostrado na Figura 1.2.

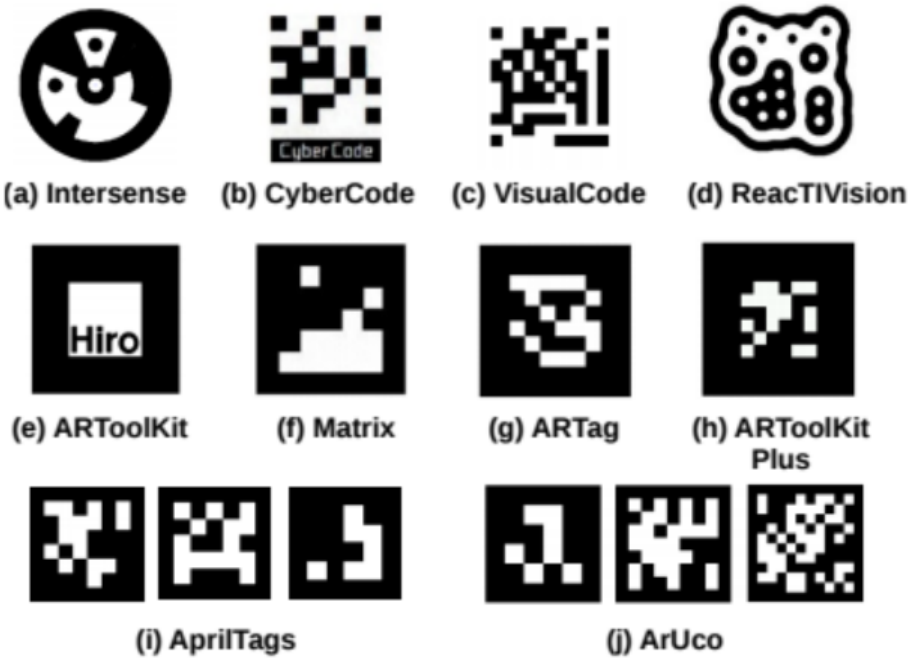

Figura 1.2: Exemplos de outros tipos de marcadores fiduciais, (GARRIDO-JURADO ET AL., 2016).

As propostas mais simples consistem em usar pontos como marcadores fiduciais, como LEDs, esferas retrorreflexivas ou planas (DORFMULLER E WIRTH; RIBO ET AL., 1998; 2001), que podem ser segmentadas usando técnicas básicas sob condições controladas. Sua identificação é geralmente obtida a partir da posição relativa dos marcadores e geralmente envolve um processo complexo. Outras abordagens usam marcadores circulares planares onde a identificação é codificada em setores circulares ou anéis concêntricos (MEASUREMENTS ET AL.; NAIMARK E FOXLIN, 1998; 2002). No entanto, os marcadores circulares geralmente fornecem apenas um ponto de correspondência (o centro), tornando necessária a detecção de vários deles para estimar a pose.

Outros tipos de marcadores fiduciais são baseados na detecção de *blob*. *Cybercode* (REKIMOTO E AYATSUKA, 2000) ou *VisualCode* (ROHS E GFELLER, 2004) são derivados da tecnologia de códigos de barras 2D como *MaxiCode* ou *QR*, mas também pode fornecer com precisão vários pontos de correspondência. Outros marcadores fiduciais populares são os marcadores de amebas *ReacTIVision* (KALTENBRUNNER E BENCINA, 2007), que também são baseados na detecção de *blob* e seu *design* foi otimizado usando algoritmos genéticos. Alguns autores propuseram o uso de classificadores treinados para melhorar a detecção em casos de má iluminação e desfoque causado pelo movimento rápido da câmera (CLAUS E FITZGIBBON, 2005).

Uma alternativa para as abordagens anteriores são os sistemas de marcadores fiduciais baseados em quadrados. Sua principal vantagem é que a presença de quatro pontos proeminentes pode ser empregada para obter a pose, enquanto a região interna é usada para identificação (usando um código binário ou um padrão arbitrário, como uma imagem). Na categoria de padrões arbitrários, um dos sistemas mais populares é o *ARToolKit* (KATO E BILLINGHURST, 1999), um projeto de código aberto que tem sido amplamente utilizado na última década, especialmente na comunidade acadêmica.

Marcadores *ARToolKit* são compostos por uma borda preta larga com uma imagem interna que é armazenada em um banco de dados de padrões válidos. Apesar de sua popularidade, tem algumas desvantagens. Primeiro, ele usa uma abordagem de correspondência de modelos para identificar marcadores, obtendo altas taxas de confusão de falsos positivos e inter-marcadores (FIALA, 2005) . Em segundo lugar, o sistema usa um limiar global fixo para detectar quadrados, tornando-o muito sensível a diferentes condições de iluminação.

Outras abordagens baseadas em sensores de imagem para realizar a navegação e localização, evitar colisões são os trabalhos de Zingg et al. (2010) e Lippiello et al. (2011), onde *Pyramidal Lukas-Kanade* é usado para a estimativa do fluxo óptico. As técnicas utilizam informações computadas por algoritmos de fluxo óptico para obter um mapa de profundidade.

Outro trabalho de relevância é o de F. Romero-Ramirez et al. (2019), o trabalho propõe uma solução para o problema dos marcadores fiduciais quadrados. O artigo trata de desafios semelhantes ao da presente tese, como detecção ineficaz sob oclusão, levando em consideração que uma pequena oclusão em qualquer parte do marcador já o torna indetectável. Outro desafio abordado no artigo é a limitação de detecção por seu tamanho.

Logo é proposto no trabalho de F. Romero-Ramirez et al. (2019) uma técnica que estima a pose da câmera em aplicações como robôs, veículos não tripulados e realidade aumentada, a partir da criação do Marcador Fractal, um novo tipo de marcador que é construído como uma agregação de marcadores quadrados, uns nos outros, de maneira recursiva. Além disso, os marcadores fractais podem ser usados sob oclusões severas.

A contribuição identificada de F. Romero-Ramirez et al. (2019) está atrelada a detecção de marcadores performando bem mesmo em situações de oclusão e longa distância, sendo extremamente útil em situações de pouso autônomo. Os trabalhos de F. Romero-Ramirez et al. (2019), bem como o de Li et al. (2020) mostram inicialmente, uma alta similaridade com o objetivo da presente tese. Contudo, ao verificar suas abordagens a fundo, pode-se identificar diferentes estratégias nas abordagens. Visto que a presente tese utiliza uma abordagem de um sistema híbrido que utiliza visão computacional e redes neurais como uma alternativa.

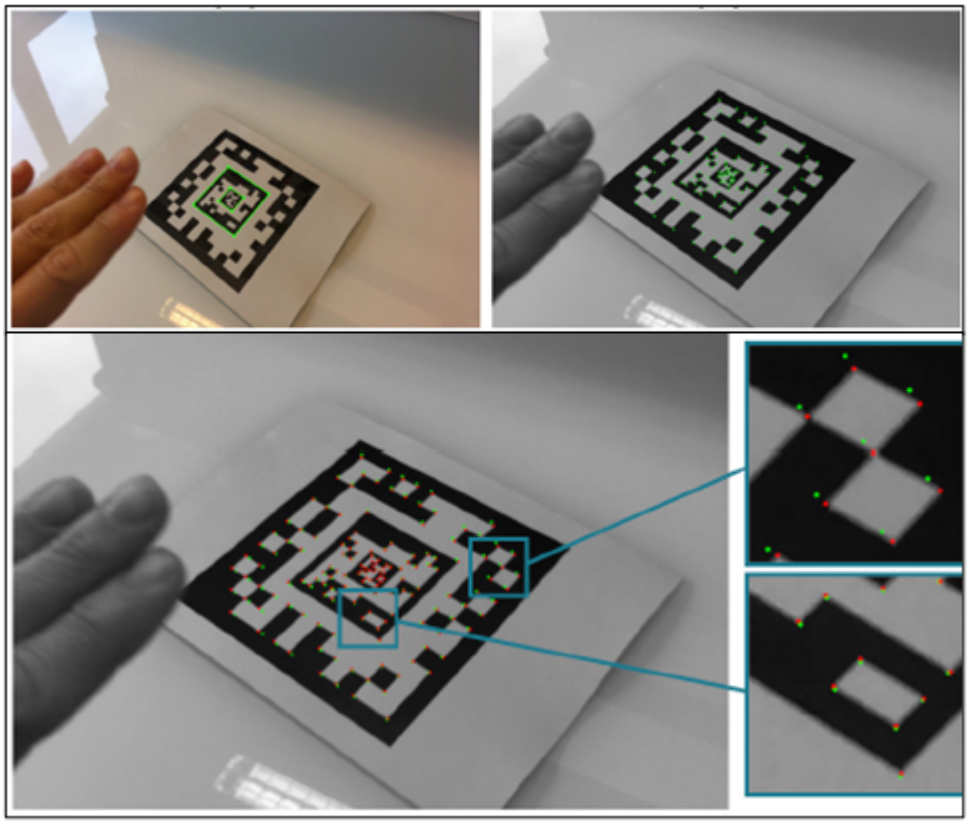

Figura 1.3: Processos de estimativa inicial de posição, refinamento da estimativa de pose com oclusão, *Fractal Markers* (F. ROMERO-RAMIREZ ET AL., 2019).

1.4 Contribuição Científica

Após uma ampla revisão literária sobre o estado da arte e das principais técnicas, foi possível propor o método com uma abordagem híbrida única. A abordagem consiste em usar apenas visão computacional e estimativas de pose de rede neural, fundidas por meio de um filtro de Kalman. A proposta é computacionalmente eficiente, supera as restrições de oclusão parcial e atinge uma ampla faixa de detecção na situação de observação do alvo a longa distância. Além disso, o método não depende de nenhuma infraestrutura externa que não seja o próprio marcador de pouso e usa apenas sensoriamento monocular a bordo para realizar a tarefa de pouso autônomo. Outra contribuição é a criação de um banco de dados de treinamento. Composto por dados sintéticos e reais em diferentes distâncias do marcador, bem como diferentes posições e imagens com oclusão parcial e semi-distorcida. O esquema simplificado do método proposto é ilustrado na Figura 1.4.

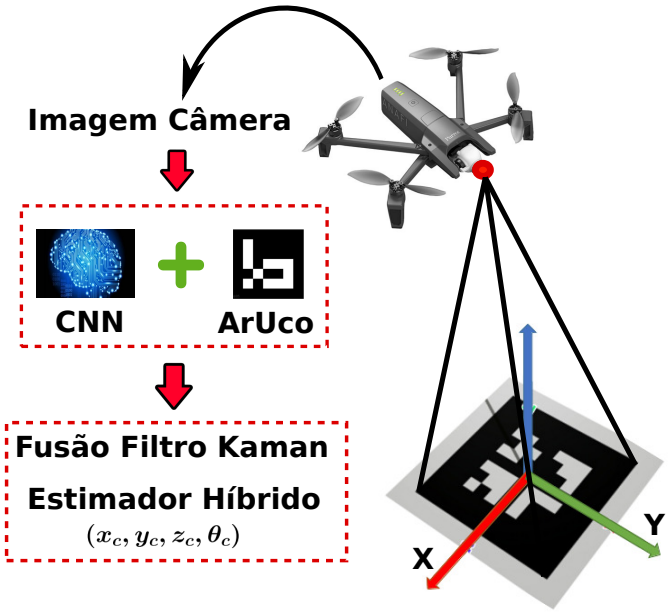

Figura 1.4: Esquema simplificado do método proposto.

O assunto da presente tese faz parte de um conjunto de pesquisas e trabalhos do grupo de automação e navegação robótica do Laboratório de Processamento de Sinais e Análise de Sistemas Dinâmicos do Departamento de Sistemas Integrados da Faculdade de Engenharia Mecânica da UNICAMP. Dentre os trabalhos realizados, destacam-se projetos de visão computacional e navegação autônoma através de plataformas móveis *Android*.

Destaca-se como inspiração, um projeto temático submetido pelo grupo de laboratório à Companhia Energética de Minas Gerais (CEMIG), cujo o objetivo era de realizar a navegação autônoma de VANTs para inspeção de redes de energia elétrica. A partir deste macro projeto, observou-se as lacunas existentes no estado da arte das técnicas de inspeção de linhas elétricas, e foram propostos temas de pesquisa afins que que contribuam para o aperfeiçoamento dessa tarefa de monitoramento. Esse contexto motivou os estudos da presente tese que deu continuidade a um intercambio submetido e aprovado pelo Programa *Santander* Mobilidade Internacional na cidade do Porto em Portugal. Logo, realizou-se uma parceira com a Faculdade de Engenharia Mecânica (FEUP), da do Universidade do Porto (UP) e a empresa *Connect Robotics* encubada na mesma Universidade.

1.5 Estruturação da Tese

Os Capítulos seguintes descrevem o trabalho da seguinte forma: O Capítulo 2 apresenta conceitos preliminares sobre os fundamentos matemáticos de detecção de objetos com marcador fiducial e rede neural utilizados na criação do método; O Capítulo 3 descreve o estimador hibrido; O Capítulo 4 descreve o sistema de aterrissagem autônoma; O Capítulo 5 descreve a implementação e apresenta dos resultados e discussões; Por fim, o Capítulo 6 aborda todas as conclusões do trabalho realizado e sugestões de trabalhos futuros.

2 Conceitos Preliminares

Nesse capítulo, aborda-se os conceitos preliminares da tese. Serão descritas as definições e fundamentos básicos das técnicas de identificação usadas. As etapas de detecção de marcadores fiduciais *ArUco* e a técnica de detecção de objetos utilizando redes neurais convolucionais (CNN) são detalhadas, mostrando sua importância para o desenvolvimento do método proposto na tese. Por fim, será apresentada uma introdução aos ambientes de simulação utilizados na tese.

2.1 Detecção de Marcador Baseado em Visão

A detecção de marcadores tem como principal funcionalidade ser um ponto de referência ou medida. O marcador é um objeto de imagem quadrada que também é chamado de marcador fiducial, são os mais populares no campo da realidade aumentada. Uma vez que um único marcador fornece os quatro pontos necessários para estimar a pose da câmera (dado que a câmera esteja devidamente calibrada). Em geral, os marcadores baseados em quadrados usam um código binário interno para identificação, detecção de erros e correção (GARRIDO-JURADO ET AL.; F. J. ROMERO-RAMIREZ ET AL., 2014; 2018). Um exemplo de marcador fiducial é ilustrado na Figura 2.1.

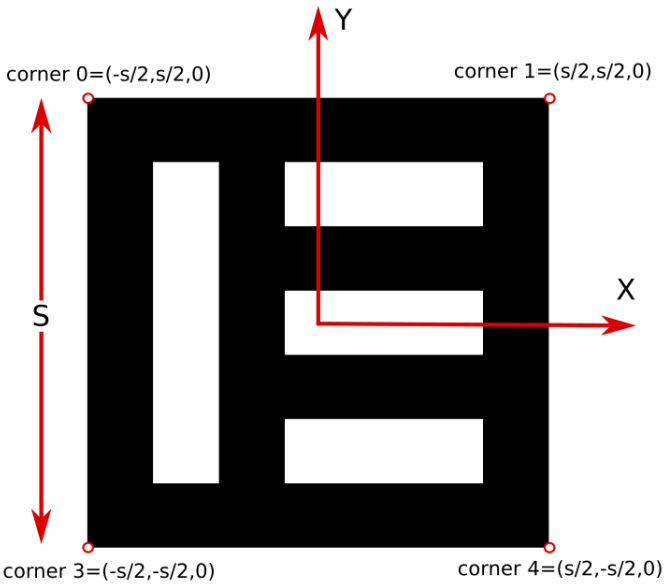

Figura 2.1: Exemplificando os quatro cantos fiduciais.

Os marcadores são compostos por uma borda preta externa e uma região interna que codifica um padrão. O padrão é binário e identifica cada marcador de maneira única. Dependendo do dicionário, existem marcadores com mais ou menos bits. Quanto mais bits, mais palavras no dicionário e menor a chance de confusão. No entanto, mais bits significa que mais resolução é necessária para a detecção correta, os detalhes extras podem ser verificados no trabalho de Garrido-Jurado et al. (2016).

O processo de detecção deste tipo de marcador pode ser dividido em duas etapas principais. O primeiro passo é a pesquisa de candidatos, que consiste em encontrar formas quadradas na imagem que se parecem com marcadores. O segundo passo é o estágio de identificação, onde a codificação interna dos candidatos é analisada para determinar se realmente são marcadores e se pertencem ao conjunto considerado válido, também conhecido como dicionário.

Segundo Munoz-Salinas (2012), o processo de detecção do alvo *ArUco* se inicia com a aplicação de um limiar adaptativo, de modo a obter as bordas. Em seguida, são identificados os contornos dos alvos. Além dos alvos, são detectadas também várias bordas indesejadas. Estas bordas com pequeno número de pontos são eliminadas. Na sequencia é realizada uma aproximação poligonal do contorno, de modo a manter os contornos côncavos com exatamente quatro cantos (retângulos).

Os cantos são ordenados no sentido anti-horário e os retângulos muito próximos entre si são removidos, pois a detecção de bordas normalmente detecta a parte externa e interna da borda do marcador, preservando apenas a borda externa. Outro processo é tratar a perspectiva de projeção de modo a obter uma vista frontal da área de um retângulo, usando uma transformação projetiva. Usando o algoritmo de Otsu (OTSU, 1979) e (ARTERO E TOMASELLI, 2000), assume-se uma distribuição bimodal e encontra-se o limiar que maximiza a variância extra-classe, mantendo uma baixa variação intra-classe. O marcador então é dividido numa grade 6x6, das quais as 25 células internas contém as informações de identificação. O resto corresponde à coroa externa. Inicialmente, verifica-se a presença da coroa. Em seguida, as 25 células internas são lidas para verificar se fornecem um código válido (pode ser necessário rotacionar o alvo para se obter um código válido). Caso seja um marcador válido, refinam-se os cantos usando uma técnica de interpolação sub-pixel.

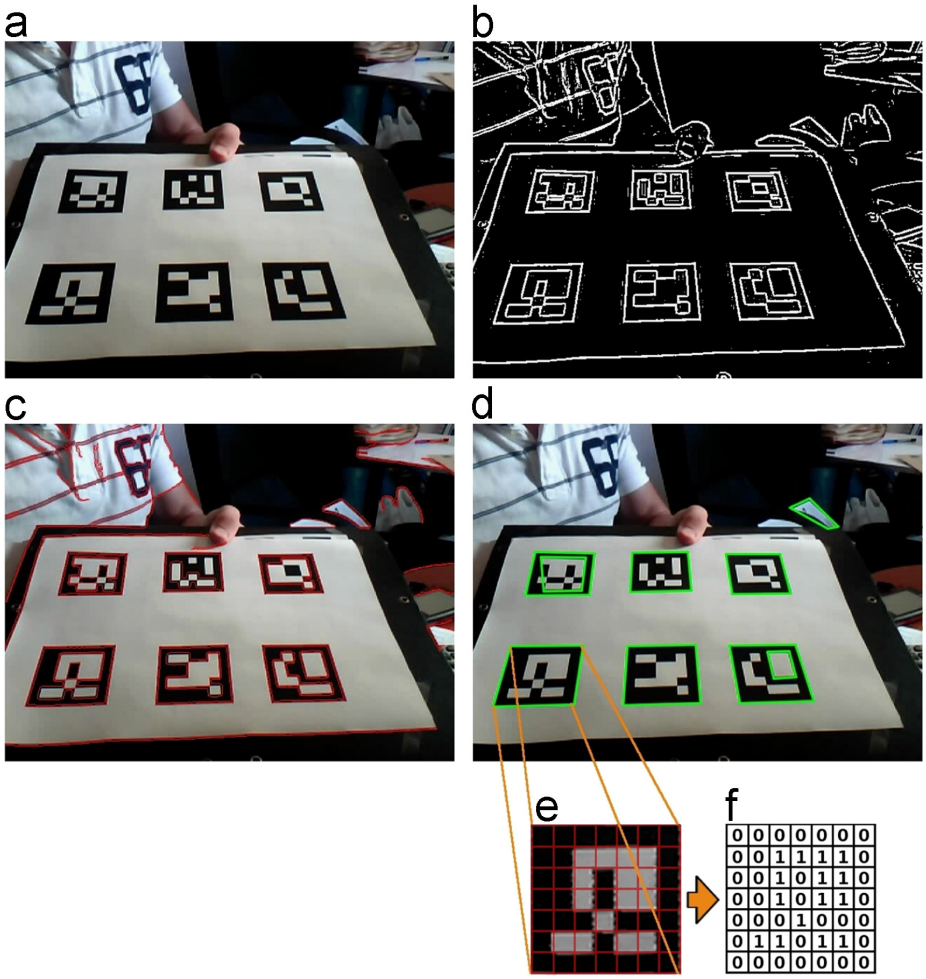

Figura 2.2: Processamento de imagem para detecção automática de marcadores. (a) Imagem original. (b) Resultado da aplicação da limiarização local. c) Detecção de contornos. (d) Aproximação poligonal e remoção de contornos irrelevantes. (e) Exemplo de transformação de marcador após perspectiva. (f) Atribuição de bits para cada célula., (GARRIDO-JURADO ET AL., 2014).

Destaca-se que a biblioteca de implementação do *ArUco 3* é de código aberto, criada para detectar marcadores fiduciais quadrados em imagens. Tendo a câmera devidamente calibrada, poderá ser estimado a pose da câmera em relação aos marcadores. A biblioteca é escrita em C++, mas há ferramentas para usar a biblioteca em outras linguagens como *Python*.

2.1.1 Utilização do Marcador Quadrado: Aplicação como Marcador de Pouso

Marcadores planares quadrados tornaram-se um método popular para estimar poses em aplicações como robôs autônomos, veículos não tripulados e treinadores virtuais. Os marcadores permitem estimar a posição de uma câmera monocular com custo mínimo, alta robustez e velocidade. Basta criar marcadores com uma impressora normal, colocá-los no ambiente desejado para cobrir a área de trabalho e registrar sua localização a partir de um conjunto de imagens (F. J. ROMERO-RAMIREZ ET AL., 2018).

O trabalho *ArUco* de Garrido-Jurado et al. (2014) que depois foi aprimorado em F. Romero-Ramirez et al. (2019) é provavelmente o sistema mais popular para detecção de marcadores atualmente porque adapta-se a iluminação não uniforme, além ter integrado em sua biblioteca os *Marcadores Fractais*, que são compostos por vários marcadores quadrados fiduciais de tamanhos diferentes em seu interior. Pode ser detectado a partir de uma grande variedade de distâncias, bem como resolver problemas de oclusão parcial ou total do marcador.

As características deste marcador, o tornam uma ferramenta poderosa para estimativa de pose de câmera em um grande número de aplicações, como robôs, veículos não tripulados e realidade aumentada. Vale ressaltar que é um estudo consolidado que está em constante atualização junto ao grupo de pesquisa "Aplicações da Visão Artificial"(AVA) da Universidade de Córdoba.

Figura 2.3: Aplicação do marcador Fractal Markers em pousos autônomos, F. Romero-Ramirez et al. (2019).

Além disso, os autores propuseram um método para obter códigos binários ótimos (em termos de distância entre inter-marcadores) usando Programação Linear Inteira Mista Garrido-Jurado et al. (2016). *Chilitags* Bonnard et al. (2013) é uma variação do *ArUco* que emprega um método mais simples para decodificar os códigos binários do marcador. Tendo como base, resultados do trabalho de F. J. Romero-Ramirez et al. (2018), verifica-se que o método tem um mau comportamento em imagens de alta resolução.

2.2 Detecção de Marcadores Baseados em Redes Neurais

De acordo com Girshick et al. (2014), o sistema de detecção de objetos consiste em três módulos. O primeiro gera uma região de interesse, categorizada de forma independente. Essas regiões de interesse definem o conjunto de detecções de candidatos disponíveis para o nosso detector. O segundo módulo é uma importante rede neural convolucional que extrai um vetor de recursos de comprimento fixo de cada região. O terceiro módulo é um conjunto de classes específicas. Para as propostas da região, esse método usa pesquisa seletiva para permitir uma comparação controlada com o trabalho de detecção anterior.

2.2.1 Redes Neurais Convolucionais (CNN)

Basicamente, a arquitetura de uma CNN consiste em três conjuntos de camadas: convolução, agrupamento (*pooling*) e camadas conectadas, como ilustrado na Figura 2.4. Cada uma delas possui uma função específica na propagação do sinal de entrada (ARAÚJO ET AL., 2017).

As camadas convolucionais são responsáveis por extrair atributos das imagens de entrada. Elas empregam filtros que acionam pequenos locais de uma imagem, e que são repetidos por toda imagem. As camadas de *pooling* são utilizadas para simplificar as informações da camada de convolução. Para isso existem alguns métodos, como por exemplo: *max pooling* (calcula o máximo da região), *mean pooling* (calcula a média da região) e o *min pooling* (calcula o mínimo da região). O *max pooling*, no qual o valor máximo é passado como saída, é o mais utilizado. Por fim as camadas totalmente conectadas são responsáveis por conectar todos os neurônios da camada de *Pooling* à cada camada de saída. Elas podem ser interpretada uma rede *Multilayer Perceptron*, mas desenvolvida de modo a demandar o mínimo pré-processamento possível (LIMA ET AL., 2020).

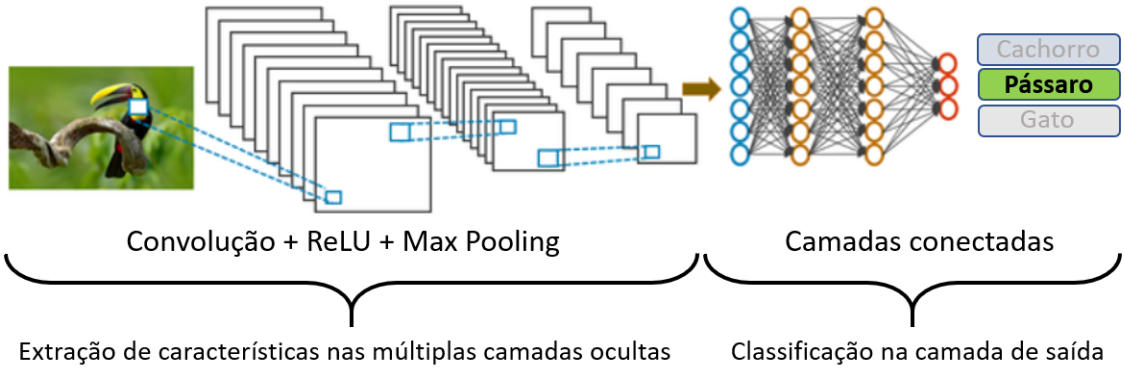

Figura 2.4: Modelo de Rede Neural Convolucional.

O *pool* é semelhante no sentido em que quebra a imagem usando janelas pequenas; no entanto, a operação executada nessa pequena janela geralmente é (média, máxima ou min) para combinar a pequena janela em um único *pixel*. Após um conjunto de convoluções e agrupamentos, a saída final é inserida em uma camada totalmente conectada, que é uma rede neural de avanço convencional para gerar um resultado. Pode-se pensar nas camadas de *pool* e convolução como uma forma de pré-processamento de imagem semelhante ao que foi feito na visão computacional tradicional, exceto que os parâmetros como a matriz em cada camada de convolução são decididos pela descida do gradiente.

2.2.2 Keras

Keras é uma API (*Application Programming Interface*) de *deep learning* escrita em *Python* que funciona sobre a plataforma de aprendizagem de máquina *TensorFlow* desenvolvida pela Google. Ela possuiu uma interface simples e altamente produtiva para resolver problemas de aprendizagem de máquina, fornecendo blocos de construção para o desenvolvimento de soluções com alta produtividade (KERAS, 2020).

Keras tem como vantagem tirar o máximo proveito dos recursos de escalabilidade, podendo ser executada em TPU (*Tensor processing unit*) ou em GPU (*Graphics Processing Unit*), além de permitir salvar o modelo treinado e ser executada por exemplo, em dispositivos móveis. Dadas as suas características, *Keras* tem ficando em primeiro lugar em termos de menções em trabalhos científicos e também tem sido adotada por pesquisadores de grandes organizações científicas, como a NASA (KERAS, 2020).

2.2.3 Regressão *Bounding Box*

Uma *Bounding Box* contém quatro vértices de coordenadas da imagem $\mathbf{b} = (b_x, b_y, b_w, b_h)$ atrelado a um valor de x. A tarefa da função de Regressão *Bounding Box* é regredir possíveis candidatas a *Bounding Box* '$\mathbf{b}$' para uma caixa delimitadora '$\mathbf{g}$', usando um regressor $f(x,\mathbf{b})$. O processo é aprendido a partir de uma amostra de treinamento $(\mathbf{g}_i,\mathbf{b}_i)$), para minimizar a função de perda L_1 da *Bounding Box*, $L_{loc}(f(x_i,\mathbf{b}_i),g_i)$. Para incentivar uma regressão invariável à escala e à localização, L_{loc} opera no vetor de distância $\Delta = (\delta_x,\delta_y,\delta_w,\delta_h)$ definido por:

$$\begin{aligned} \delta_x = (g_x - b_x)/b_w, \delta_y = (g_y - b_y)/b_h \\ \delta_w = \log(g_w - b_w), \delta_h = \log(g_h - b_h) \end{aligned} \tag{2.1}$$

Como a regressão *bounding box* geralmente realiza pequenos ajustes em '$\mathbf{b}$', os valores numéricos da Equação (2.1) podem ser muito pequenos. Portanto, a perda de regressão é geralmente muito menor que a perda de classificação. Para melhorar a eficácia do aprendizado multitarefa, Δ geralmente é normalizado por sua média e variância, ou seja, δ_x é substituído na Equação (2.2):

$$\delta_x^{'} = \frac{\delta_x - \mu_x}{\sigma_x} \tag{2.2}$$

2.2.4 Modelo *MobileNet*

Sendo o *MobileNet V1*, o modelo escolhido para ser utilizado nesta tese, se faz importante levantar alguns conceitos. Segundo o trabalho de Hollemans (2018), a grande ideia por trás do modelo é que as camadas convolucionais podem ser substituídas pelas chamadas convoluções separáveis em profundidade. Visto que as camadas são essenciais para as tarefas de visão computacional, porém com alto custo computacional. Sendo assim, o trabalho da camada de convolução é dividido em duas subtarefas: primeiro, há uma camada de convolução em profundidade 3x3 "*Depthwise convolution*"que filtra a entrada, seguida por uma camada de convolução 1x1 "*Pointwise Convolution*"que combina esses valores filtrados para criar novos recursos. O esquemático desse processo

é ilustrado na Figura 2.5:

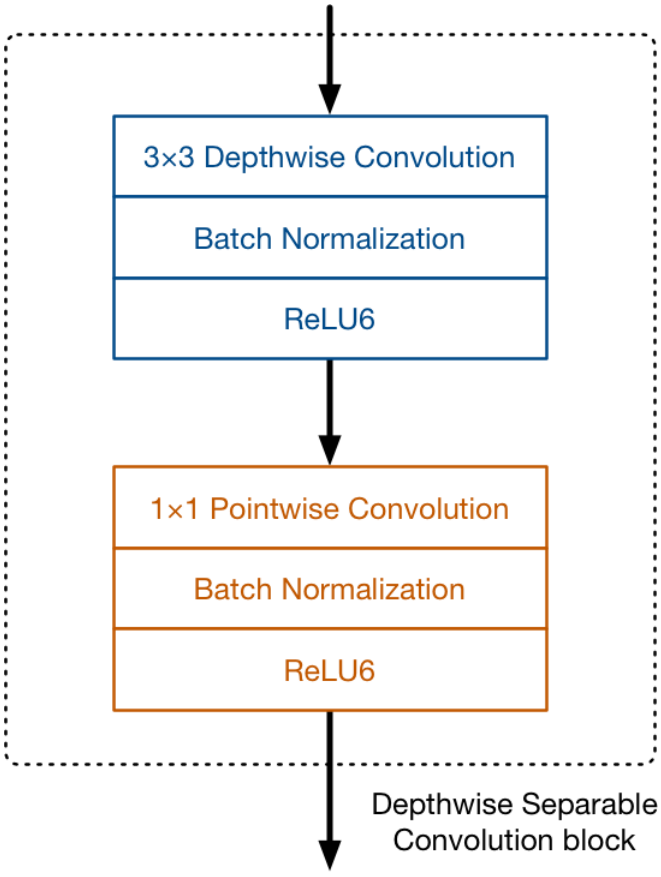

Figura 2.5: Convolução separável em profundidade (HOLLEMANS, 2018).

Juntas, as convoluções em profundidade (*depthwise*) e em ponto (*Pointwise*) formam um bloco de convolução separável em profundidade "*depthwise Separable Convolution Block*". Esse sistema é bem similar à convolução tradicional, porém com eficiência de tempo mais elevada. A arquitetura completa do *MobileNet V1* consiste em uma convolução 3×3 regular como a primeira camada, seguida por 13 vezes o bloco de construção acima.

Ainda seguindo o trabalho de Hollemans (2018), verifica-se o uso de camadas de convolução que são seguidas pela normalização em lote. Tal estratégia, segue os procedimentos comuns de arquiteturas modernas. A função de ativação usada pelo *MobileNet* é *ReLU6*. Ela impede que as ativações se tornem muito grandes, a formulação é vista na Equação (2.3):

$$y = min(max(0, x), 6) \tag{2.3}$$

Os autores do artigo científico *MobileNet* Howard et al. (2017) descobriram que a função *ReLU6* é mais robusta que a função tradicional *ReLU* ao usar computação de baixa precisão. Em um classificador baseado em *MobileNet*, normalmente existe uma camada média global de *pool* no final, seguida por uma camada de classificação totalmente conectada ou uma convolução 1x1 equivalente e um *softmax*. Na verdade, há mais de um *MobileNet*. Foi projetado para ser uma família de arquiteturas de redes neurais. Sendo assim, existem vários hiperparâmetros que permitem jogar com diferentes trocas de arquitetura. Também faz com que o formato da função pareça mais um sigmoide, como é visto na Figura 2.6:

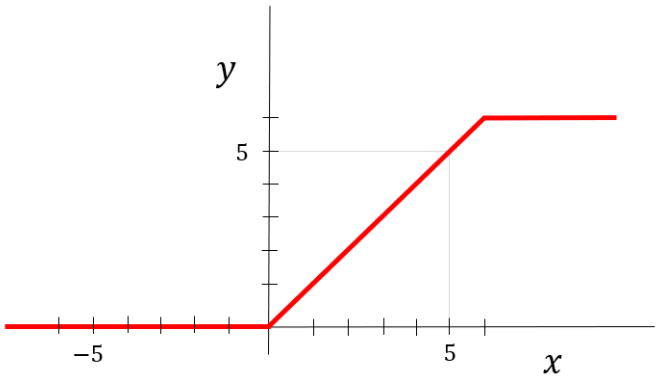

Figura 2.6: Função de ativação ReLU6.

Segundo Howard et al. (2017), o mais importante desses hiperparâmetros é o multiplicador de profundidade, também conhecido como “multiplicador de largura”. Isso muda quantos canais existem em cada camada. O uso de um multiplicador de profundidade de 0,5 reduzirá pela metade o número de canais usados em cada camada, o que reduz o número de cálculos por um fator de 4 e o número de parâmetros aprendíveis por um fator 3. Portanto, é muito mais rápido que o modelo completo, mas também menos preciso.

Graças à inovação da técnica de convoluções separáveis em profundidade "*Depthwise Separable Convolutions*", o modelo *MobileNet V1* consegue realizar um trabalho computacional cerca de 9 vezes menor quando comparado com outras redes neurais similares de mesma precisão. Esse tipo de camada funciona tão bem que conseguimos obter modelos com 200 ou mais camadas para

serem executados em tempo real. Logo, verifica-se uma boa utilidade desse modelo para aplicações em tempo real com quadricópteros autônomos que necessitam de visão computacional e redes neurais intensas, eficazes e com baixo custo computacional.

2.3 Software: Interface de Comunicação ROS

O sistema operacional do robô (ROS- *Robot Operating System*) licenciado sob uma licença BSD de código aberto é a ferramenta escolhida para realizar a interface de comunicação com o quadrotor e seu simulador viabilizado através do *Gazebo*. Segundo os desenvolvedores o sistema (FOUNDATION, 2019) "O ROS fornece bibliotecas e ferramentas para ajudar os desenvolvedores de *software* a criar aplicativos robóticos. Ele fornece abstração de hardware, *drivers* de dispositivos, bibliotecas, visualizadores, transmissão de mensagens, gerenciamento de pacotes e muito mais."

GAZEBO

O *Gazebo* é o simulador padrão usado na estrutura do *ROS*. Embora sejam projetos separados, há um pacote para *Gazebo* no repositório oficial do *ROS*. Este é um pacote mantido pelos próprios mantenedores do *Gazebo*, o Fundação de Robótica *Open Source*. Ele contém *plugin*s que fazem a integração do ROS e do Gazebo. Esses *plugins* podem ser anexados para objetos na cena do simulador, e fornecer fácil método de comunicação com *ROS*.

Segundo fontes do site do fabricante, o *Gazebo* começou como um projeto na Universidade do Sul da Califórnia. Mais tarde, foi integrado no *framework ROS* por John Hsu, que era um pesquisador sênior da *Willow Garage*, mantenedor original da *ROS*. Desde então, o *Gazebo* foi mantido pela *Open Source Robotics Foundation*, que é uma *spin-off* da *Willow Garage*, e o mesmo mantenedor que cuida de *ROS*. Gazebo é um projeto de código aberto, disponível para qualquer pessoa, fornecido através de uma licença do tipo *Apache License 2.0*. O *Gazebo* é executado apenas no *Linux*, porém terá suporte para a plataforma *Windows* também. Seu lançamento está previsto para acontecer nas próximas versões. Na Figura 2.7, ilustra-se o ambiente de simulação criado para realização dos testes simulados de aterrissagem autônoma.

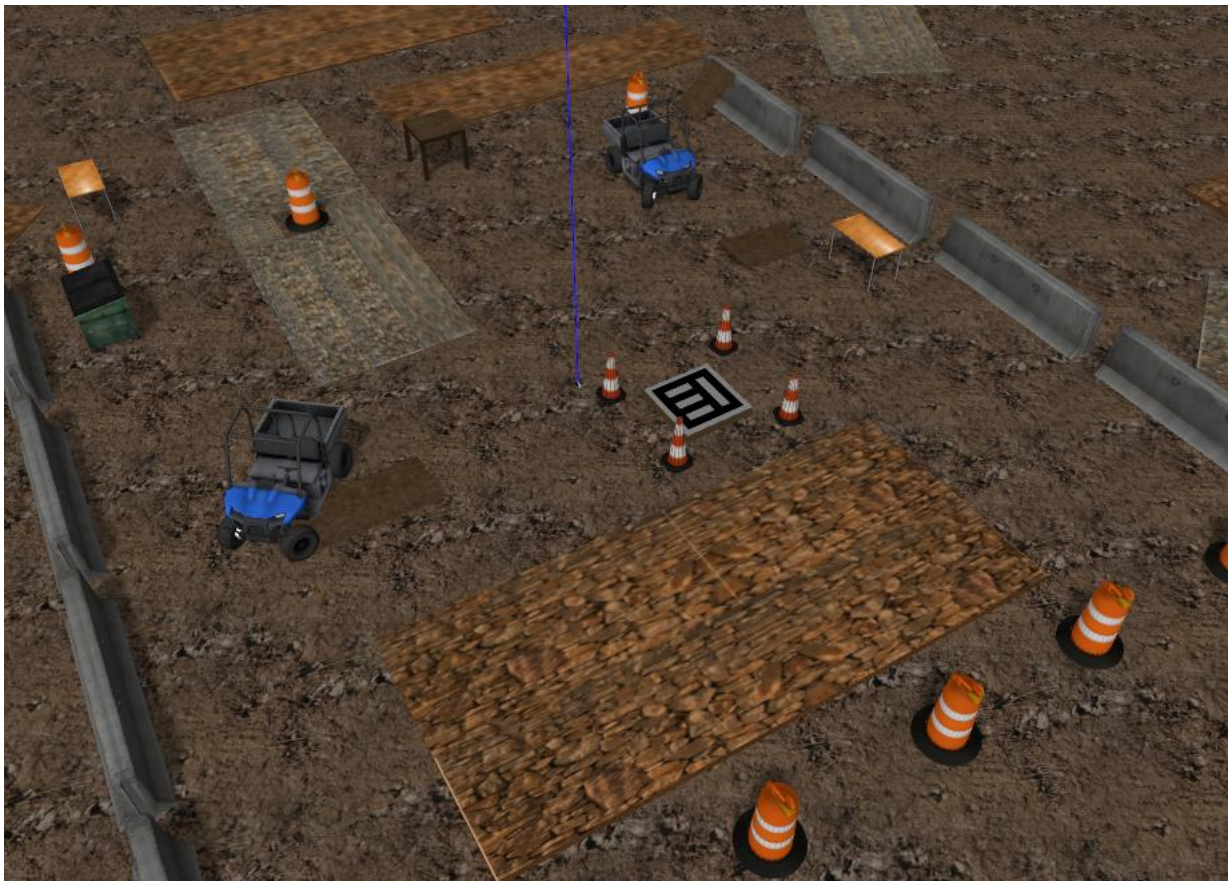

Figura 2.7: Ambiente de Simulação Criado para o Pouso Autônomo.

V-REP

V-REP é um simulador robótico desenvolvido pela *Coppelia Robótica*, com sede em Zurique, na Suíça. É um *software* comercial, que pode ser obtido gratuitamente em sua versão educacional. *V-REP* tem suporte para *Windows*, *Linux* e sistemas operacionais Mac. É possível usar 7 diferentes linguagens de programação com *V-REP*, o idioma padrão sendo *LUA*. O cenário virtual simulado no *V-REP* possui uma plataforma de *software* robótico que permite o uso de modelos dinâmicos e interfaces com uma variedade de linguagens de programação (ROHMER ET AL., 2013). A Figura 2.8 ilustra uma típica simulação de pouso de quadricóptero sobre um marcador do tipo *ArUco*.

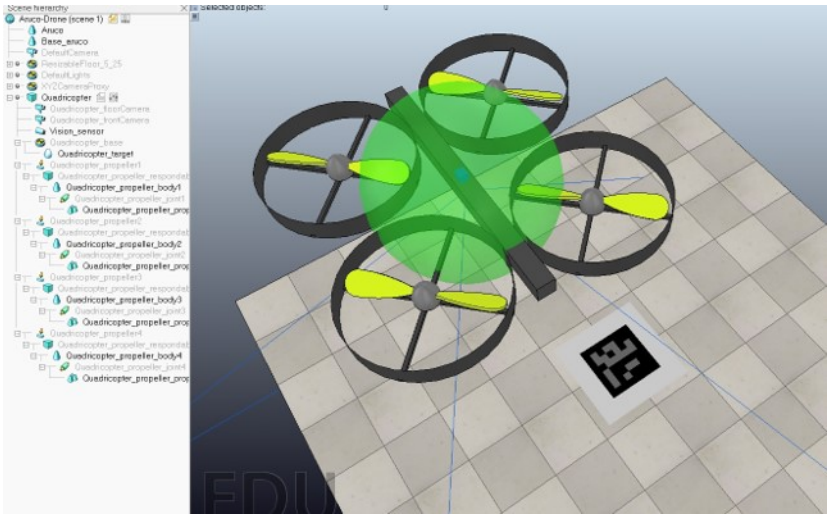

Figura 2.8: Simulação de pouso autônomo no ambiente *V-REP*.

2.4 Simulador do Quadricóptero: *Sphinx*

Sphinx é uma ferramenta de simulação inicialmente pensada para cobrir as necessidades dos engenheiros da empresa *Parrot* que desenvolvem *software* de quadricóptero. O conceito principal é executar um *firmware* de *drone Parrot* em um *Notebook*, em um ambiente isolado e bem separado do sistema *host*. *Gazebo* é amplamente utilizado para simular o ambiente físico e visual do quadricóptero. Ele vem com vários recursos amigáveis:

- Visualizar dados de voo em tempo de execução através do navegador da *web*;
- Modificação do comportamento do *drone* em tempo de execução;
- Totalmente *scriptable* (usando linhas de comando);
- Permite construa de cenas próprias;
- Sensor de vídeo e *streaming*;
- Conecta a qualquer tipo de controlador, como o *Freeflight* executado em um *smartphone*;
- Pode ser executado remotamente (por exemplo, em um servidor distante);

3 Método do Estimador de Posição Híbrida

Neste capítulo, descreve-se a fundamentação teórica e matemática para o desenvolvimento do estimador híbrido de posição e orientação da câmera embarcada no quadricóptero. Inicialmente, será descrito uma visão geral do método, a Figura 1.4 ilustra bem o esquema simplificado do método proposto. Em seguida, um detalhamento das estimativas de posição da câmera usando *ArUco* e *CNN*. Por fim, descreve-se o estimador híbrido usando o filtro de Kalman, detecção de *outlier*, fusão e filtragem de dados de posição.

3.1 Visão Geral do Método

As etapas fundamentais para a implementação do método estimador de pose híbrido são ilustradas na Figura 3.1. As primeiras etapas referentes a detecção do marcador foram vistas no Capítulo 2. Na segunda etapa, verifica-se o núcleo do método, através de uma abordagem híbrida única. A abordagem consiste primeiramente em estimar a pose do marcador em relação à câmera usando apenas visão computacional. Em seguida, estimar os mesmo marcador, porém agora estimando somente a posição, especificamente sem orientação, usando rede neural.

Na terceira etapa, apresenta-se três hipóteses de processamento. Na primeira hipótese, verifica-se que nenhuma estimativa foi recebida com sucesso, sendo assim, refeita a leitura das estimativas através de um laço de repetição. O gatilho das demais hipóteses são verificadas caso haja recebimento correto de dados de pelo menos uma das técnicas, fazendo com que o laço de repetição da primeira hipótese seja interrompido.

Rotulando o gatilho da segunda hipótese no momento em que pelo menos um dos estimadores retorna um dado de forma correta, verifica-se a filtragem dos dados e a publicação da pose estimada através dessa única técnica. Por fim, a terceira hipótese é verificada quando duas técnicas computam os dados com sucesso, assim o método híbrido aplicará uma fusão sensorial, exercendo seu princípio de funcionamento perfeito. Destaca-se que a estimação de posição foi desenvolvida junta ao método visto que a rede neural só retorna a detecção de um objeto treinado.

A primeira técnica baseia-se apenas em visão computacional (*ArUco*), retornando posição e orientação angular, representada através das coordenadas (x, y, z, θ).

A segunda usa redes neurais (*CNN*), retornando apenas as coordenadas de posição (x, y, z) sem a orientação angular, devido a limitações do modelo usado. Um filtro de *Kalman* recebe as informações das duas técnicas para ser realizada a estimativa híbrida de pose(posição linear + orientação angular). Destaca-se o uso dos filtros de média móvel com passagem de banda regulados com base na altura estimada do quadricóptero.

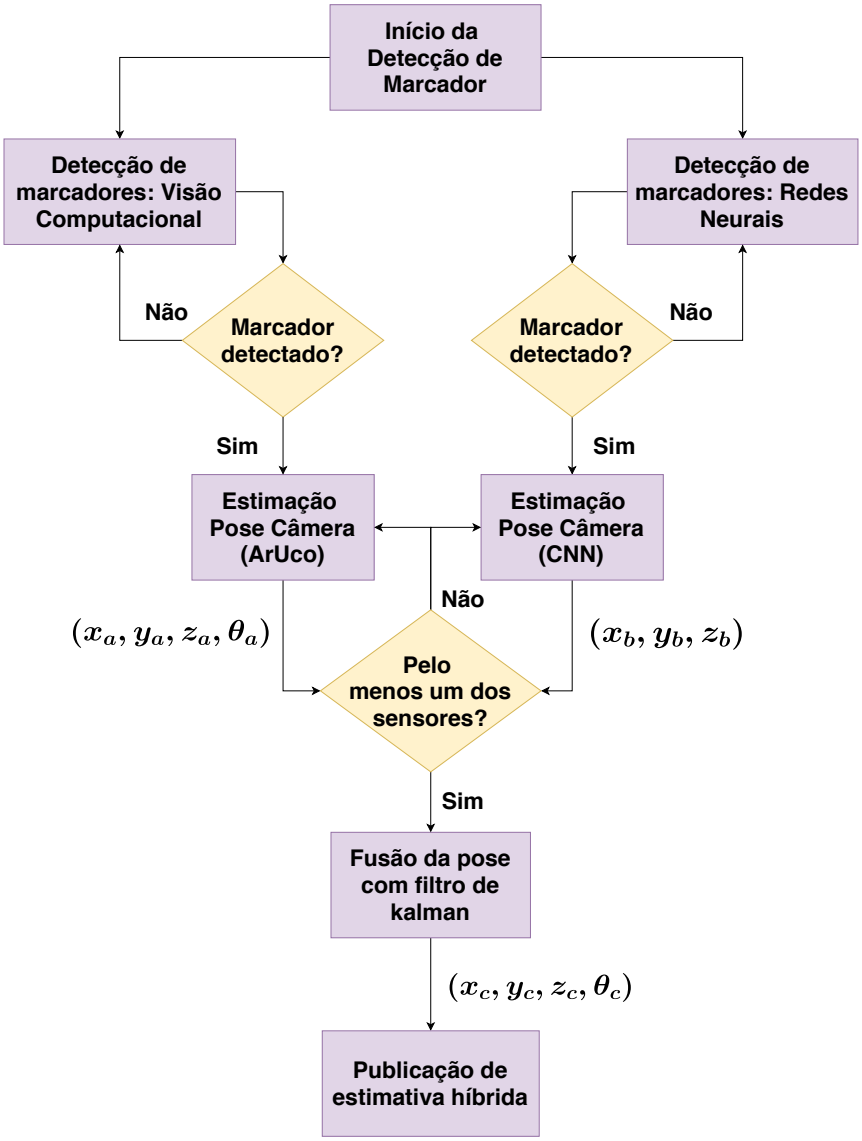

Figura 3.1: Etapas do método híbrido proposto.

3.2 Estimando posição da câmera com o ArUco

De acordo com (MUNOZ-SALINAS, 2012), uma imagem *ArUco* de uma câmera calibrada pode ser usada para estimar sua pose (ou seja, sua posição 3D no espaço em relação ao marcador). A detecção dos quatro cantos de um marcador permite a aplicação de estimadores de pose planares. A posição da câmera é estimada a partir de um ponto de referência no centro do marcador.

É extremamente importante observar que as estimativas de pose usando apenas quatro pontos coplanares estão sujeitas a ambiguidade. Um marcador pode projetar um mesmo *pixel* em dois locais diferentes na câmera. Em geral, a ambiguidade pode ser resolvida se a câmera estiver próxima ao marcador. No entanto, à medida que o marcador se torna pequeno (quando visto à distância), podem ocorrer erros de estimativa de canto. Uma pequena oclusão em qualquer parte do marcador pode torná-lo indetectável.

ArUco estima a pose da câmera com base nos parâmetros intrínsecos da câmera e nas especificações exatas do tamanho do marcador. A biblioteca informa a posição relativa dos marcadores e da câmera. Os valores são dados pelos vetores $Rvec$ e $Tvec$, representando a transformação do marcador para a perspectiva da câmera.

A base matemática para estimativa de câmera com o *ArUco* usa o modelo de câmera pinhole. Neste modelo, uma vista de cena é formada pela projeção de pontos 3D no plano da imagem usando uma transformação de perspectiva, iniciando na Equação (3.1).

$$s.m' = A[R|t]M' \tag{3.1}$$

Dada uma câmera calibrada através de seus dados extrínsecos com base em uma referência global, é possível projetar um ponto 3D (X, Y, Z) no espaço de coordenadas do mundo da seguinte maneira.

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.2}$$

- (X,Y,Z) são as coordenadas de um ponto 3D no espaço de coordenadas do mundo
- (u,v) são as coordenadas do ponto de projeção em *pixels*
- A é uma matriz de câmera ou uma matriz de parâmetros intrínsecos
- (cx,cy) é um ponto principal que geralmente está no centro da imagem
- fx,fy são as distâncias focais expressas em unidades de *pixel*

Portanto, se uma imagem da câmera é dimensionada por um fator, todos esses parâmetros devem ser dimensionados (multiplicados / divididos, respectivamente) pelo mesmo fator. A matriz de parâmetros intrínsecos não depende da cena visualizada. Assim, uma vez estimado, pode ser reutilizado enquanto a distância focal for fixa (no caso de lentes de zoom).

A matriz de rotação-translação conjunta $[R|t]$ é chamada de matriz de parâmetros extrínsecos. É usado para descrever o movimento da câmera em torno de uma cena estática, ou vice-versa, o movimento rígido de um objeto na frente de uma câmera estática. Ou seja, $[R|t]$ converte as coordenadas de um ponto (X, Y, Z) em um sistema de coordenadas fixo em relação à câmera. A transformação acima é equivalente ao seguinte (quando $z \neq 0$):

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t \tag{3.3}$$

$$x' = x/z \tag{3.4}$$

$$y' = y/z \tag{3.5}$$

$$u = f_x.x' + c_x \tag{3.6}$$

$$v = f_y.y' + c_y \tag{3.7}$$

A Figura 3.2 ilustra o modelo da câmera pinhole.

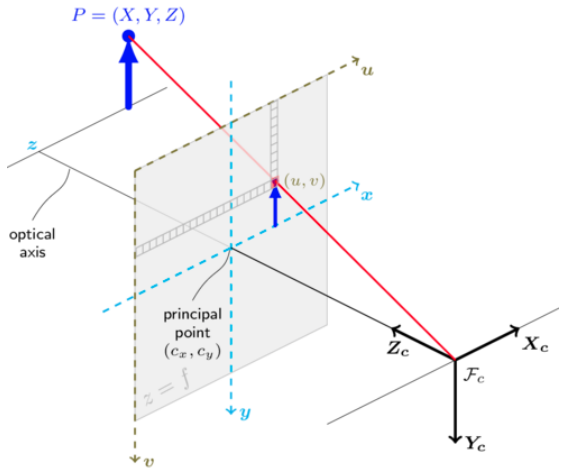

Figura 3.2: Modelo de câmera pinhole (DOXYGEN-OPENCVS, 2018).

Lentes reais geralmente apresentam alguma distorção, especialmente distorção radial e leve distorção tangencial. Para mais informações, é indicada a leitura de (DOXYGEN-OPENCVS, 2018)

3.3 Estimando posição da câmera com a CNN

As redes neurais convolucionais não são aplicadas em processamento de imagens, necessariamente, para obter a estimativa da pose da câmera. Sua principal função é detectar objetos com uma boa margem de confiabilidade, desde que seja criado um banco dados de elevada pluralidade com base nas situações procuradas. No entanto, é possível também obter informações como largura e altura do objeto, uma vez que a técnica fornece a detecção do objeto através de um rótulo quadrado.

De acordo com (GIRSHICK ET AL., 2014), o sistema de detecção de objetos consiste em

três módulos, os quais foram integrados no método proposto. O primeiro gera propostas de uma região independente de categoria. Essas propostas definem o conjunto de detecções de candidatos disponíveis para o nosso detector. O segundo módulo é uma importante rede neural convolucional que extrai um vetor de recursos de comprimento fixo de cada região. O terceiro módulo é um conjunto de classes específicas. A presente técnica usa pesquisa seletiva para permitir uma comparação controlada com o trabalho de detecção anterior, ilustrado na figura 3.3.

Figura 3.3: Modelo de RCNN.

Utilizou-se de cálculos geométricos para gerar informações sobre o centro do marcador referenciada pelos eixos centrais da imagem Cx e Cy. Escolheu-se um marcador quadrado e com constraste para facilitar a identificação com precisão. A detecção ocorre com sucesso nos casos de oclusão parcial, desde que se tenha a visibilidade completa de pelo menos uma aresta do quadrado. Nesses casos, o maior valor absoluto de aresta é usado para atribuir o valor da largura do marcador.

Informações como aresta no eixo X e Y podem ser usadas aplicando o modelo de câmera pinhole. Assim é possível determinar a distância do marcado em relação à câmera e as posições horizontal e vertical relativas do marcador em relação ao mundo real. Descobrir a taxa de transformação da câmera usada no processo de captura de imagens também é uma etapa essencial.

A imagem é formada no plano focal da câmera de acordo com suas lentes. A taxa de transformação da câmera pode ser obtida através da Equação (3.8), através da semelhança do triângulo observado no diagrama montado em Figura (3.4).

$$f = x.Z/X \tag{3.8}$$

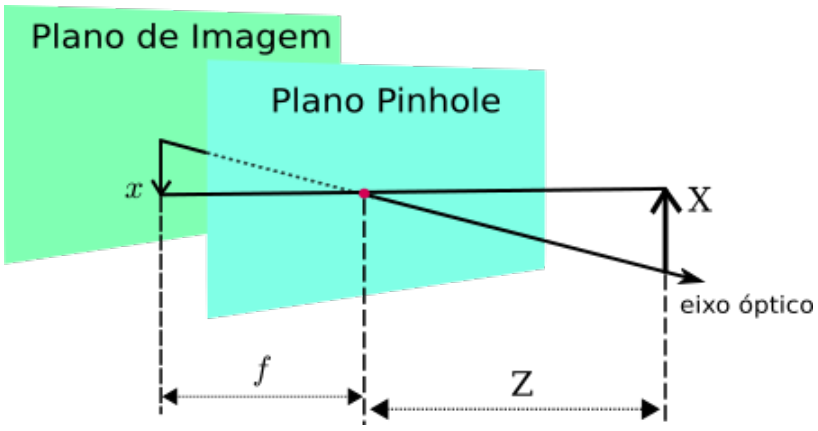

Figura 3.4: Diagrama esquemático da geometria pinhole.

f é a taxa de transformação, x é a maior medida entre largura e altura identificada na imagem em *pixels*. Z é a distância em metros do marcador para a câmera em metros e X é a largura ou altura do marcador, também em metros. Converteu-se as distâncias em *pixels* em metros no mundo real, a partir do conhecimento da distância focal f.

Estimou-se a distância do objeto através da Equação (3.9), obtendo uma borda quadrada ao redor do marcador e tendo valores conhecidos para X, x e f. A orientação do marcador, no entanto, não pode ser determinada por essa abordagem.

$$Z = X.f/x \tag{3.9}$$

Os eixos X e Y do $body\_frame$ são definidos como os eixos de referência da câmera em relação ao marcador com os eixos X_c e Y_c ilustrados na Figura 3.5. As distâncias p a serem cobertas estão relacionadas ao centro da imagem, onde o centro de gravidade do quadricóptero pode ser localizado e são calculados em termos de *pixels*, de acordo com a Equação (3.10) e a Equação (3.11).

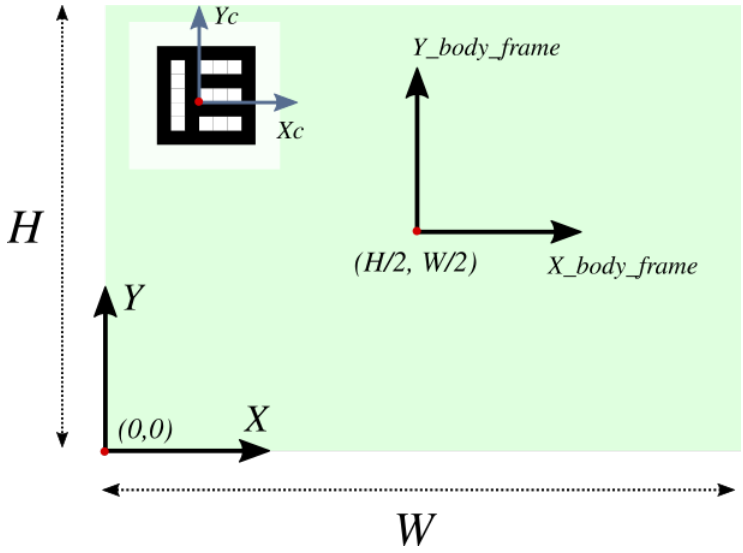

Figura 3.5: Eixos orientados para obter coordenadas da imagem.

$$p_x = X_c - W/2 \tag{3.10}$$

$$p_y = Y_c - H/2 \tag{3.11}$$

W e H são a largura e a altura da imagem, respectivamente, como mostra a Figura 3.5, e os pontos X_c e Y_c são o centro do marcador. A técnica usada para calcular as distâncias em X e Y consiste em obter uma constante K que relaciona esse diâmetro ao número correspondente de *pixels* na imagem. A constante é então encontrada para cada imagem, conforme declarado na Equação (3.12).

$$K = X/x \tag{3.12}$$

X é a maior dimensão entre a largura e a altura do marcador e x seu valor de *pixel* correspondente na imagem. Com essa constante, é possível determinar qualquer distância entre dois pontos da imagem e o número de *pixels* entre eles. M é a distância métrica no mundo real e p em *pixels*, conforme a Equação (3.13).

$$M = K.p \tag{3.13}$$

3.4 Estimativa de pose híbrida usando o Filtro Kalman

3.4.1 Detecção de Outlier

Na estatística, *outlier* é um valor aberrante ou valor atípico, logo é uma observação que apresenta um grande afastamento das demais da série. Dentro do conceito prático da proposta é um passo indispensável, e deve ser feito antes da aplicação do filtro Kalman, visto que tem a finalidade de eliminar os valores discrepantes da técnica *CNN*. A tarefa é realizada através de uma função via *software* que acumula um número K de amostras de posição. Dentro desta amostra é feito o cálculo de erro médio absoluto das poses medidas. Se o valor aferido estiver fora de um valor de *threshold* já pré-determinado, o último valor de amostra é considera um *outlier*, sendo retirado da amostragem. O algoritmo descrito é apresentado na Tabela 1.

Algorithm 1: Detecção de Outlier.

Input: Ponto x a ser detectado.
Output: x é discrepante, denotado por $flag\_outlier$.
Inicialização : K = 10, i = 0, mean = 0, vector = []

```
if i < k then
    vector[i] = vector[i] + x
else
    mean = sum(vector) / K
    if |mean − x| > threshold then
        Considera x como um outlier, flag_outlier = true
    else
        x não é um outlier, flag_outlier = false
        Retira o primeiro valor
        Adicione os dados mais recentes recebidos no vetor
i = i + 1
return flag_outlier
```

3.4.2 Fusão de Dados

O termo Fusão de Dados (*Data Fusion*) tem obtido cada vez mais destaque na literatura internacional de sensoriamento remoto. Entretanto, no Brasil o conceito é raramente empregado e quando utilizado é confundido com a chamada fusão de resoluções espaciais tratada na literatura internacional.

Geralmente, realizar a fusão de dados tem várias vantagens. Essas vantagens envolvem principalmente melhorias na autenticidade ou disponibilidade dos dados. Exemplos do primeiro são detecção aprimorada, confiança e confiabilidade, bem como redução na ambiguidade de dados, enquanto a extensão da cobertura espacial e temporal pertencem à última categoria de benefícios. A fusão de dados também pode fornecer benefícios específicos para alguns contextos de aplicação (KHALEGHI ET AL., 2013).

Existem vários problemas que tornam a fusão de dados uma tarefa desafiadora. A maioria desses problemas surgem dos dados a serem fundidos, imperfeição e diversidade das tecnologias de sensor e da natureza do ambiente de aplicação. Vale destacar os principais desafios de fusão encontrados no presente trabalho:

Imperfeição dos dados: Os dados fornecidos pelos sensores são sempre afetados por algum nível de imprecisão e também pela incerteza nas medições. Os algoritmos de fusão de dados devem ser capazes de expressar tais imperfeições com eficácia e de explorar a redundância de dados para reduzir seus efeitos.

Dados conflitantes: Fusão de tais dados pode ser problemática, especialmente quando o sistema de fusão é baseado no raciocínio de crença evidencial e na regra de combinação. Para evitar a produção de resultados contra-intuitivos, qualquer algoritmo de fusão de dados deve tratar dados altamente conflitantes com cuidado especial.

Tempo operacional: No caso de sensores homogêneos, a frequência de operação dos sensores pode ser diferente. Um método de fusão de dados bem projetado deve incorporar várias escalas de tempo para lidar com essas variações de tempo nos dados. Em configurações de fusão distribuída, diferentes partes dos dados podem percorrer diferentes rotas. Esse problema precisa ser tratado, especialmente em aplicativos em tempo real.

A fusão de dados utilizada nesse trabalho é composta pelos filtro de *Kalman* e *Butterworth*. O filtro Kalman fornece uma solução recursiva para o problema de filtragem linear discreta. É um conjunto de equações matemáticas que podem estimar o estado de um processo, com erro quadrado médio mínimo.

O filtro é muito poderoso, pois não apenas pode estimar estados passados, presentes e até futuros, mas também funciona mesmo em modelos de sistema desconhecidos. É uma ferramenta de engenharia fundamental que é amplamente utilizada em controle e robótica e para várias tarefas de estimativa em sistemas autônomos. Devido à sua eficiência, o filtro Kalman também é um dos métodos mais populares para fusão de dados com vários sensores (KHALEGHI ET AL., 2013).

O filtro Kalman foi projetado para sistemas com formato linear de espaço de estado e o estado do filtro é representado por duas variáveis:

- $\hat{x}_k$, a estimativa do estado a posteriori no tempo k, dada observações até o momento e incluindo no tempo k;
- P_k, matriz de covariância da estimativa a posteriori (uma medida da precisão mensurada da estimativa do estado).

O filtro Kalman pode ser escrito como uma única equação, no entanto, geralmente é conceituado como duas fases distintas: "*Predict*"e "*Update*". A fase de predição usa a estimativa de estado do passo anterior para produzir uma estimativa do estado no passo atual. Portanto, também devemos considerar as seguintes variáveis:

- F_k, o modelo de transição de estado;
- H_k, o modelo de observação;
- Q_k, a covariância do ruído do processo;
- R_k, a covariância do ruído de observação;
- B_k, o modelo de entrada de controle que é aplicado ao vetor de controle u_k;

Então, temos na etapa *"Predict"* na Equação (3.14) e Equação (3.15):

$$\hat{x}_k = F_{k-1}\hat{x}_{k-1} + B_k u_k \tag{3.14}$$

$$P_k = F_k P_{k-1} F_k^{\mathsf{T}} + Q_k \tag{3.15}$$

$\hat{x}_k = [xp, yp, zp]^T$ é o vetor de estado estimado, que rastreia a posição da câmera em relação a um quadro inercial e P_k é o matriz de previsão de covariância. Nesta formulação de Kalman, não se usa a entrada de controle u_k (ou seja, $B_k = 0$), pois os dados de observação do marcador visual *(AruCo)* e da rede neural *(CNN)* são usados diretamente para propagar o estado na etapa de previsão. Consequentemente, o modelo de transição de estado F_k é uma matriz de identidade 3×3 conforme mostrado na Equação (3.16)

$$F_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

Um conjunto de 6 leituras de posição sobre os eixos x, y e z é usado como dados de medição como na Equação (3.17).

$$z_k = \begin{bmatrix} x_{cnn} & y_{cnn} & z_{cnn} & x_{aru} & y_{aru} & z_{aru} \end{bmatrix}^T \tag{3.17}$$

Portanto, o modelo de observação H_k pode ser definido como uma matriz 6×3 incluindo dois blocos de matrizes de identidade 3×3, conforme expresso na equação Equação (3.18).

$$H_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.18}$$

Portanto, temos a etapa *Update* na Equação (3.19), (3.20), (3.21) em que $\tilde{y}_k$ é a inovação residual, S_k é a covariância residual e K_k é o ganho ideal de Kalman.

$$\tilde{y}_k = z_k - H_k \hat{x}_k \tag{3.19}$$

$$S_k = H_k P_k H_k^{\mathsf{T}} + R_k \tag{3.20}$$

$$K_k = P_k H_k^{\mathsf{T}} S_k^{-1} \tag{3.21}$$

Aqui, podemos calcular o valor atualizado de $\hat{x}_k$ e P_k. O vetor de estados atualizado $\hat{x}_k$ e a matriz de covariância P_k são então estimados nas Equação (3.22) e Equação (3.23), onde I_3 é uma matriz identidade 3×3.

$$\hat{x}_k = \hat{x}_k + K_k \tilde{y}_k \tag{3.22}$$

$$P_k = (I_3 - K_k H_k)\, P_k \tag{3.23}$$

Uma preocupação com a fusão de medições é o problema de dados ausentes ou incorretos. Muitos fatores, como oclusão parcial, distância longas, distorção ou excesso de luz, podem atra-

palhar os dados fornecidos pela câmera. Nos casos em que o intervalo de tempo de falha é muito breve, o filtro Kalman é capaz de prever um valor de posição combinando os dados das duas técnicas e gerando uma estimativa precisa. Por outro lado, quando o intervalo de tempo de falha de uma técnica é alto, a outra técnica domina a estimativa da posição.

Esse controle é feito alternando os valores de covariância dos sensores com base no tempo de detecção de cada sensor. Por exemplo, quando ocorre uma oclusão parcial, a detecção de *ArUco* provavelmente falhará; portanto, apenas a pose da *CNN* será destacada pelo filtro Kalman neste momento em conjunto com os dados passados. Em resumo, o filtro Kalman pode lidar com situações com dados ausentes e gerar uma estimativa robusta.

3.4.3 Pós-Filtragem: Filtro *Butterworth*

O filtro *Butterworth* é um tipo de projeto de filtros eletrônicos. Contudo, ele também pode ser implementado de forma digital, logo foi escolhido na concepção do método proposto, por conta de sua atuação eficaz ao utilizar médias móveis e ser auto-regressivo. Ele é desenvolvido de modo a ter uma resposta em frequência o mais plana o quanto for matematicamente possível na banda passante. Além disso, possuí fácil aplicação com bibliotecas prontas e implementadas em *Matlab* e linguagem *Python*.

4 Sistema de Aterrissagem Autônoma

Neste capítulo, descreve-se a plataforma de pouso escolhida, bem como a estrutura modular do sistema. Também são estabelecidos parâmetros heurísticos, estruturação de dicionários e parametrização do algoritmo para a implementação do método no sistema de aterrissagem autônoma.

4.1 Estrutura Modular

O sistema foi desenvolvido a partir de uma estrutura modular. O mesmo permite fácil modificação ou substituição dos algoritmos dentro de cada módulo, sem exigir alterações nos outros. Requer relativamente poucas alterações para serem adaptadas a diferentes plataformas (por exemplo, asas fixas), algoritmos ou cenários.

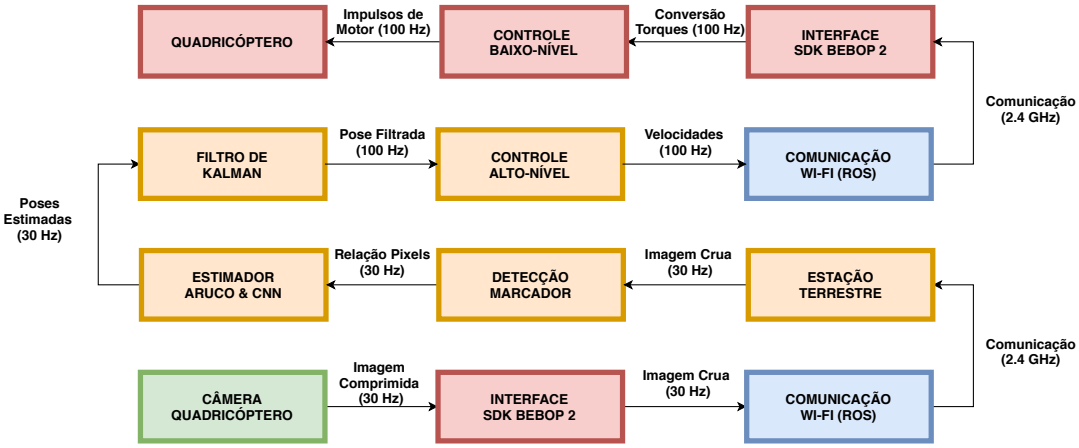

Figura 4.1: O esquema representa os quadros de processo do sistema de aterrissagem autônoma. Caixas azuis representam o módulo de comunicação, caixas amarelas representam o sistema de *software* da estação terrestre. As caixas vermelhas representam a interface de *hardware* e *software* integrada do quadricóptero.

A Figura 4.1 fornece uma visão geral dos módulos estruturados. O monitoramento em tempo real é realizado, com a possibilidade de troca de comandos e mensagens, a partir de uma estação de controle de solo (GCS). A comunicação é realizada usando a plataforma *ROS* (QUIGLEY ET AL., 2009) com o sistema *WI-FI* de um quadricóptero *SDK Bebop2*.

4.2 Plataforma de Pouso Autônoma: Estruturação Marcador Fiducial

A plataforma de pouso autônoma foi escolhida de tal forma a favorecer uma detecção única por ambas as técnicas utilizadas no método. A proposta foi desenvolvida para ser uma solução escalável, atribuindo novos marcadores de pouso caso seja necessário. Visto que a detecção foi desenvolvida com base nos dicionários de biblioteca do *ArUco*. O marcador testado e escolhido para a plataforma de pouso é ilustrado na Figura 4.2. Sua forma é um marcador quadrado *ArUco* com o *Id = 273*.

No caso de marcadores de pouso, é importante analisar o melhor tamanho e forma, com o objetivo de favorecer as condições de pouso. Através de testes, verificamos que, quanto mais alto a aeronave inicia o procedimento de pouso, maior deve ser o alvo a ser detectado, consequentemente, exigirá uma maior resolução de de captura de imagem.

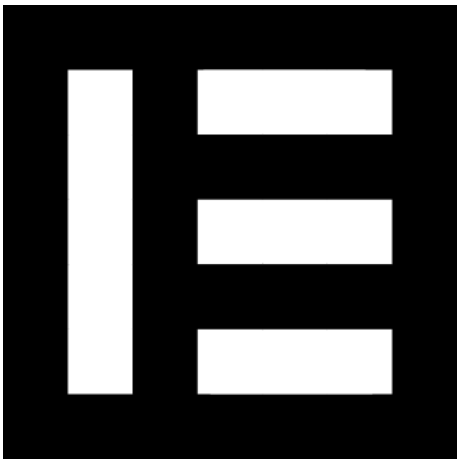

Figura 4.2: Marcador usada na plataforma de aterrissagem onde o método híbrido proposto é aplicado, *Id = 273*.

Alvos maiores são mais difíceis de produzir, transportar e armazenar. Eles se tornam ilegíveis em altitudes mais baixas à medida que excedem o campo de visão da câmera. Se um pouso de precisão for aceitável a partir de uma altitude mais baixa, um alvo menor poderá ser usado. Então, nesta tese, usamos um tamanho de marcador de 0,5 metro, pois é um dos tamanhos mais comuns para missões de pouso e decolagem. O valor é adequado para o quadricóptero iniciar uma detecção com uma altura média de 15m a 20m.

Primeiramente é estruturado um marcador fiducial que retorne identificação única ou próximo disso para gerar confiabilidade. Existem vários tipos de codificações, chamadas de *dicionários*. A biblioteca de marcadores usada neste trabalho é *ArUco* com o dicionário *Orginal-Aruco* que possui número de identificação 273, ilustrado na Figura 4.2.

O *ArUco* também suporta outros tipos de dicionário, como *AprilTag*, *ArToolKit+*, *ARTAG* e *CHILITAGS*. Além disso, pode-se realizar a criação de um dicionário customizado, como ilustrado na Figura 4.3.

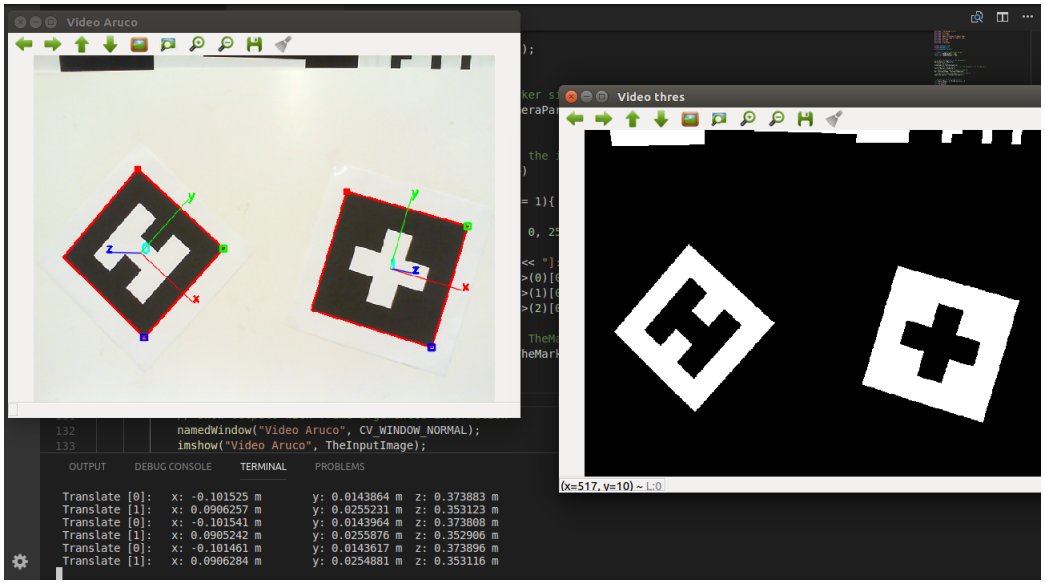

Figura 4.3: Criação de Marcador Fiducial Customizado.

Destaca-se que nem todo dicionário criado terá uma performance aceitável para uso. Visto que alguns podem gerar ambiguidade a respeito de sua orientação. Os exemplos acima, reflete um pouco sobre esse conceito. Verificamos que os marcadores criados acima, o algoritmo não consegue diferenciar os quadrantes superiores em relação aos inferiores.

Uma alternativa para resolver o problema de marcadores grandes que excedem o campo de visão *Field of View* (FoV) da câmera em altitudes mais baixas é a inserção de vários marcadores de diferentes tamanhos formando uma placa de marcadores que pode ser usada para maior robustez e aumento do campo de visão *Field of View* (FoV).

4.3 Parametrização do Algorítimo de Detecção

A parametrização do algorítimo de detecção é importante porque dependendo dos valores configurados, o algorítimo tende a fornecer informações mais confiáveis de orientação, localização e distância. Com informações mais precisas teremos uma base de dados confiável para realizar comparações com outros métodos, favorecendo assim a convergência do método.

Como comentado na sessão 4.2, utilizou-se a técnica de visão computacional *ArUco* desenvolvida por Garrido-Jurado et al. (2014), a fim de servir como base de identificação confiável do marcador de pouso. Justifica-se o uso pelo fato de ser uma das ferramenta de visão computacional mais poderosas dentre o estado da arte atual.

Os melhores parâmetros foram determinados com base em testes experimentais e através da literatura (F. J. ROMERO-RAMIREZ ET AL.; GARRIDO-JURADO ET AL.; GARRIDO-JURADO ET AL., 2018; 2014; 2016). É importante notar a necessidade de mensurar com precisão o tamanho exato do marcador (borda preta a borda preta), sendo esse um dos parâmetros mais importantes da parametrização. Para evitar que os marcadores não se deformem, dobrem, batam ou se movam, é aconselhável montá-los em algo sólido, como uma placa em madeira ou acrílico.

Outro parâmetro particularmente importante de se certificar é o método de calibração, bem como a rigidez da página de calibração A4, para mais detalhes de como foi feita a calibração indica-se a leitura de Munoz-Salinas (2012). Na Figura 4.4 é ilustrado um teste de parametrização, onde é aferido a distância métrica real do quadricóptero em relação ao marcador através de um sensor de altura embarcado no quadricóptero.

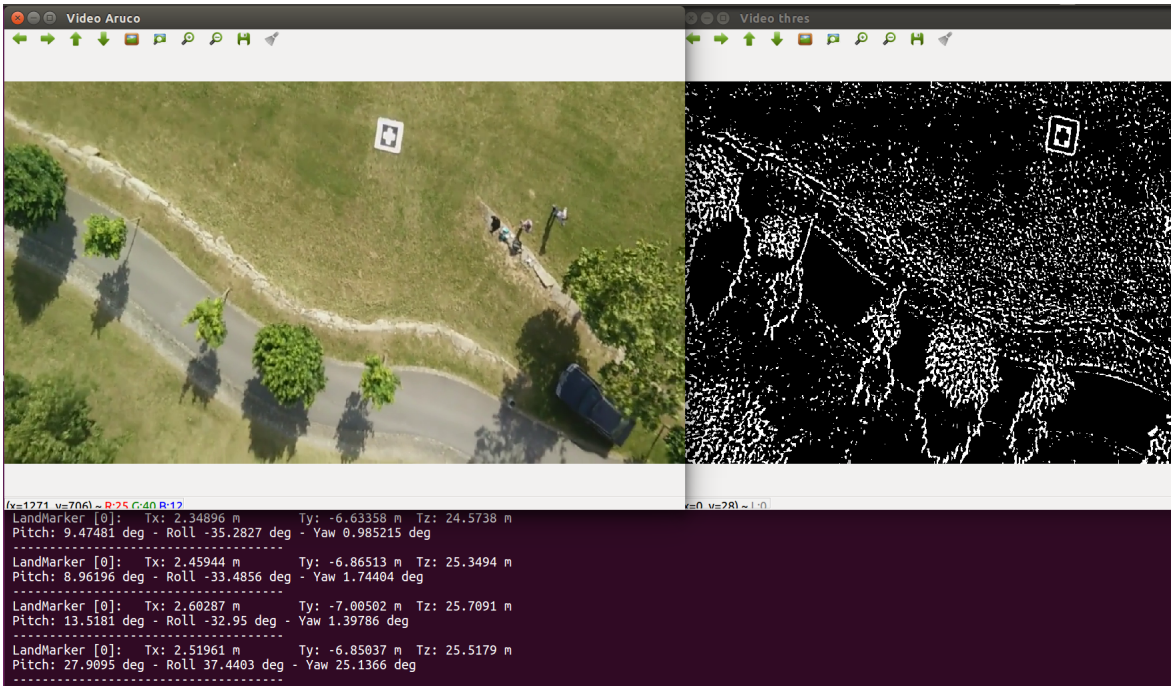

Figura 4.4: Teste de parâmetros, análise de confiabilidade dos dados de pose da câmera em relação ao marcador fiducial.

É recomendável imprimir o marcador, haja uma borda branca bem visível em torno do próprio marcador preto. A borda branca é importante para uma detecção confiável. Segue abaixo a Tabela 4.1 com os principais parâmetros utilizados.

Tabela 4.1: Parâmetros sugeridos para identificação confiável.

Parâmetro	**Valor**
Dictionary	"ARUCO"
DetectMode	DM_FAST
CornerRefinementM	CORNER_SUBPIX
ThresMethod	THRES_AUTO_FIXED
MaxThreads	4
BorderDistThres	$1.4999999664723873e-02$
LowResMarkerSize	20
MinSize	0
EnclosedMarker	0
AdaptiveThresWindowSize	-1
ThresHold	177

Após a calibração da câmera e obtenção dos parâmetros intrínsecos (matriz de calibração e coeficientes de distorção), em conjunto com o tamanho exato do marcador. A biblioteca pode informar a posição relativa dos marcadores e da câmera. Utiliza-se os vetores $Rvec$ e $Tvec$ que, representam a transformação do marcador para a câmera, também conhecidos como parâmetros extrínsecos. São basicamente as rotações 3D, onde $(Rvec = Rx, Ry, Rz)$ e translações 3D $(Tvec = Tx, Ty, Tz)$ necessárias para traduzir o sistema de referência da câmera para o sistema arbitrário.

Os elementos de rotação são expressos na fórmula de Rodrigues, assim você pode obter a matriz de rotação 3x3 equivalente usando a função do *Opencv* $cv :: Rodrigues()$. A equação de Rodrigues basicamente converta uma Matriz de rotação em ângulos de Euler através da biblioteca *OpenCV*. Senso assim, verifica-se a confiabilidade dos dados de pose com simples análise estatística.

4.4 Algorítimo de Pouso Autônomo

Um algoritmo de controle foi desenvolvido na linguagem *Python*, que é uma linguagem de alto nível, interpretada, de fácil interação com elementos de *hardware* e outras linguagens de programação e sintaxe simples, e além de tudo uma plataforma *Open Source* de fácil integração com *ROS*. Outra vantagem de se empregar essa linguagem de programação foi a existência de bibliotecas de rede neurais, técnicas de visão computacional avançada e comunicação com o quadrotor em alto nível. A Figura 4.5 ilustra a lógica de atuação para o pouso autônomo.

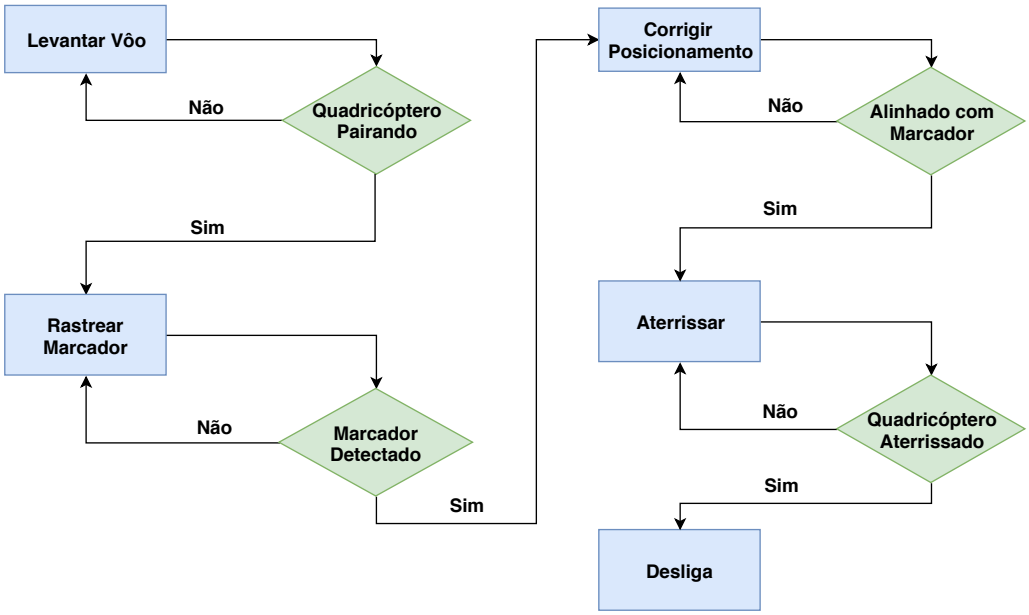

Figura 4.5: Fluxograma do estado de operação do Quadricóptero.

Verifica-se as etapas, partindo desde o pressuposto que o quadricóptero já levantou voo e está pairando. Em seguida, é iniciado o rastreamento, detecção e estimativa de posição do marcador. Estas informações são utilizadas para alinhar a posição do quadricóptero com o marcador em terra. Por fim, é acionado o seu sistema de aterrissagem autônoma até o seu desligamento após atingir seu objetivo final de pouso.

Abaixo é mostrado a lógica de programação do algorítimo de aterrissagem autônoma via plataforma *ROS*. Inicialmente é realizado a subscrição do tópico de dados da odometria, publicado pelo algorítimo anterior. Em seguida é realizado a lógica de controle através de um tópico de velocidade do quadricóptero.

Algorithm 2: Detecção da Plataforma de Pouso.

Input: Captura de Imagem Embarcada.
Output: Posição da Plataforma de Pouso.
Inicialização : $id\_find = 273$, $marker\_size = 0.5$

```
arcuo_pose = aruco_detection (camera image)
cnn_pose = cnn_detection (camera image)
hybrid_pose = kalman_filter(aruco_pose, cnn_pose)
if landing_tag found then
    distance_module = sqrt(posex^2 + posey^2)
    if |yaw_error|> yaw_threshold and |distance_module|< threshold then
        return yaw.pid_control()
    else
        return 0
    if |hybrid_pose_xy|> pose_xy_threshold then
        return pose.pid_control_xy()
    else
        if |hybrid_pose_z|> pose_z_threshold then
            return pose.pid_control_z()
        else
            return auto_landing()
else
    return finding_tag()
```

5 Implementação e Resultados Experimentais

Neste capítulo serão apresentados os experimentos realizados para validar a presente tese. Inicialmente, discute-se sobre a elaboração de um *dataset* juntamente com as condições de treinamento e teste da rede neural. Posteriormente, experimentos reais e simulados são realizados e discutidos. Foram realizadas nove testes para comparar o desempenho do Método Híbrido proposto com a detecção de marcadores baseados em visão computacional e a detecção de marcadores baseados em redes neurais. Nossos experimentos visam avaliar a capacidade de detecção à distância, robustez à oclusão parcial, precisão na estimativa da pose e velocidade do método proposto. A biblioteca utilizada para a detecção de marcadores foi a configuração DM_NORMAL.

5.1 Local dos Experimentos *Outdoor*

Os experimentos reais do quadricóptero foram realizados parcialmente nos Laboratórios da Faculdade de Engenharia Mecânica da UNICAMP e na Faculdade de Engenharia da Universidade do Porto. A parte de aquisição de parâmetros com imagem foi principalmente coletado através de testes externos (*outdoor*) em Porto, Portugal. A implementação de pouso completo foi implementada em Campinas, São Paulo. A Figura 5.1 ilustra um dos procedimentos de teste.

Figura 5.1: Local de experimento externo com Bebop 2.

5.2 Equipamentos Utilizados

Quadricóptero: *Parrot BeBop 2*

Os testes foram realizados com o quadrotor de (500gr) de peso, dimensões $200x180x110$ mm, captura imagens de alta qualidade com alcance de voo estendido de até 2 quilômetros com *Parrot Skycontroller 2*. Realiza vídeos *full HD* 1080p e fotos de 14 *megapixels* com a câmera grande angular incorporada. As imagens podem ser capturadas nos formatos RAW, JPE e DNG. Bateria de 2700mAh de alta capacidade com até 25 min de duração da bateria. Sistema de estabilização digital de 3 eixos garante vídeos suaves e estáveis. Dimensões de $200x180x110mm$. Conectividade de Wi-Fi $802.11a/b/g/n/ac$. Faixa de sinal: 300 m. Com *GPS* e *IMU* embutido.

Estimação de posição do quadricóptero é realizado através do método proposto. Sendo necessário o entendimento mínimo da relação dinâmica do quadricóptero no espaço. O mesmo possui uma origem e três direções espaciais principais, as quais atribuem posição e orientação à aeronave a partir da pose das mesmas em relação à origem. A Figura 5.2 ilustra os eixo de coordenadas no Quadrotor.

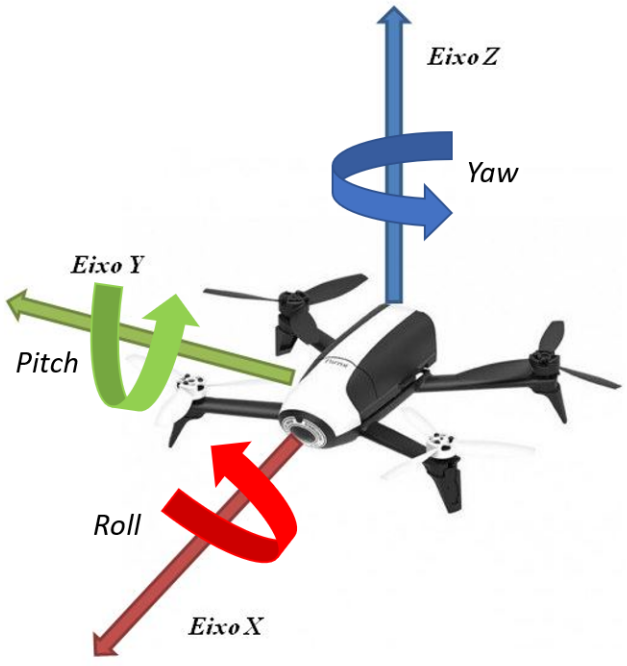

Figura 5.2: Eixo de referências do Quadrotor.

Ground Station

Utilizou-se como estação de processamento um notebook Intel®Core™ i5-9300H com 8MB de cache (2,4 GHz a 4,1 GHz com processador Intel®Turbo Boost), 32GB [2x 16GB - Dual Channel] de memória DDR4 (2666 MHZ). GPU NVIDIA® GeForce® GTX 1050 (3GB GDDR5) operando no Ubuntu 16 LTSOS.

5.3 Tipo de Controlador

O controle do quadricóptero é feito a partir de um controle de alto nível através da velocidade, tanto translacional quanto angular. Baseado em um controlador empírico de *PID* (proporcional, integral e derivativo), sua formulação tem como entrada de erro a posição estimada pelo método e como saída uma velocidade de atuação. Sua ativação segue o estado de operação do Quadricóptero ilustrado na Figura 4.5. Além disso, o controle tem um gatilho vinculado a angulação máxima da câmera que corresponde a -84 graus e vinculado ao recebimento da mensagem de pose. Os ganhos do controlador *PID* foram estimados empiricamente através de experimentos simulados. Além disso, o controle detém de heurísticas baseadas na proporcionalidade da altura do quadricóptero, tolerância erro e do módulo da distância x e y em relação ao marcador de pouso. Controlador atua de forma menos agressiva possível afim de evitar *overshooting* de controle.

5.4 Elaboração Dataset

Três fatores influenciaram a criação do banco de dados de treinamento. O primeiro fator é a diversidade de ambientes, no caso dos experimentos da tese, foram escolhidos os ambientes de asfalto, grama e cimento, os quais são os mais comuns para aterrissagem por quadricóptero. Outro fator importante foi o registro de cenários com oclusão parcial variando de 0% a cerca de 50% da imagem em diferentes ambientes e amplitudes. Vale ressaltar que a intensidade luminosa também foi pensada ao realizar o banco de dados. Outro fator essencial foi a variedade de fotos com diferentes translações e rotações da câmera em relação ao marcador de aterrissagem, especialmente imagens de longa distância. Finalmente, optamos por usar cerca de 20% das imagens simuladas (sintéticas) e 80% das imagens reais, criando assim um banco de dados híbrido. Links no *GitHub*: https://github.com/alanprodam/hybrid_detection.git

com a documentação completa do trabalho e no *google drive*: https://drive.google.com/drive/folders/1ZTMB1in1V2yIJzMeVwB7e7TVaQCq14yT?usp=sharing são disponibilizados para acesso rápido.

5.4.1 Configurações de Treinamento e Teste da CNN

O modelo usado foi $ssd\_mobilenet\_v1$ (HOLLEMANS, 2018) e implementado em conjunto com a biblioteca *TensorFlow*. Nesta parte, o modelo de detecção de objeto é treinado para detectar um objeto personalizado (marcador de aterrissagem). São usadas imagens RGB com resolução de *pixel* $856x480$, combinando *TFRecords* para os dados de treinamento e teste. O treinamento é realizado com um total de 751 imagens, 535 para treinamento, 146 para testes e 70 para validação. Os resultados mostram uma precisão de 99 % no treinamento e 95 % no teste usando interações 5041, taxa de aprendizado de 0,004 e duas camadas ocultas. A interação 5041 é usada como um modo congelado porque se mostrou mais eficiente nos testes de validação.

5.5 Resultados em Ambiente Simulado

Um modelo numérico do quadricóptero *Parrot Bebop 2* é usado no ambiente de simulação do Gazebo para validar o método de detecção de marcador híbrido. O Bebop 2 oferece um simulador pronto para uso chamado *Sphinx*. A posição exata do robô (*ground truth*) não está disponível neste simulador, pois sua odometria inercial é misturada ao ruído gaussiano branco em um código de desenvolvedor inacessível dentro do pacote. Portanto, a simulação não foi capaz de comparar efetivamente as estimativas de posição do método proposto com as outras técnicas. A simulação, no entanto, foi muito importante para verificar a funcionalidade do algoritmo, bem como a implementação do filtro Kalman. O desempenho do método pode ser observado em situações críticas de detecção, permitindo também o ajuste dos valores de covariância para estimativas mais suaves.

Foram simulados diversas procedimento de teste para decolagem e pouso, em que o quadricóptero segue um caminho circular e quadrado em torno do marcador de pouso. O ambiente simulado é testado em diferentes cenários, com o marcador de aterrissagem em diferentes posições e larguras, além de situações variadas de luminosidade e oclusão parcial do alvo. A distancia máxima testada é de cerca de 20 metros e a velocidade de avanço do quadricóptero é de cerca de 0,3

m / s. A Figura 5.3 mostra a simulação de pouso autônomo.

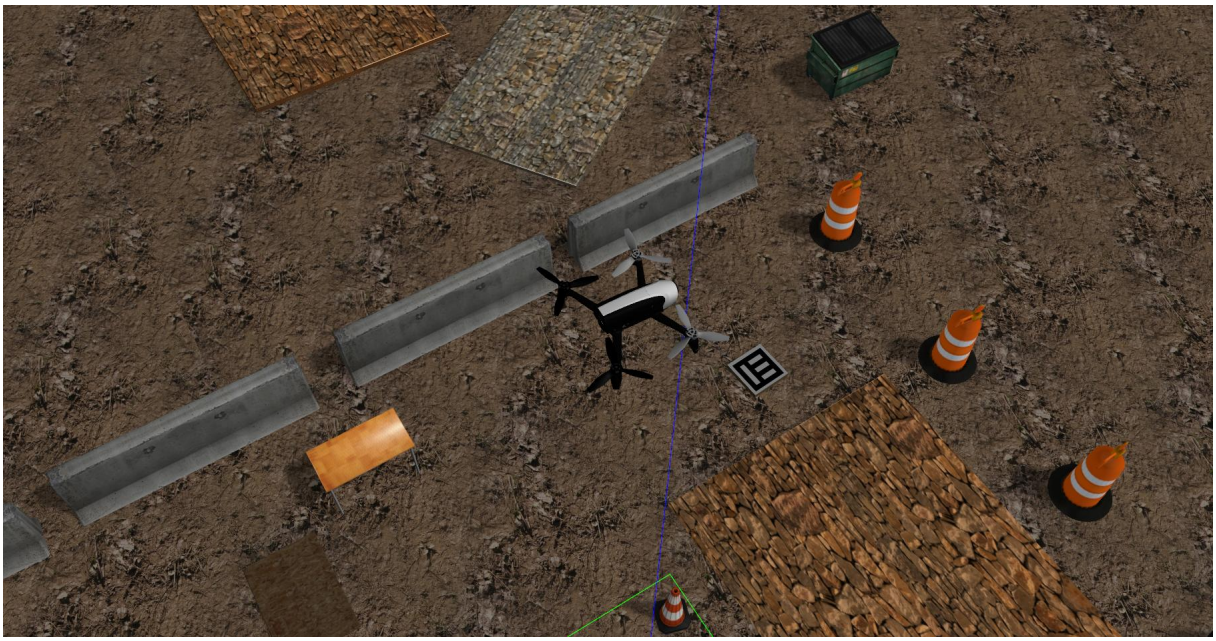

Figura 5.3: Experiência de aterrissagem autônoma em ambiente simulado.

Utilizou-se a ferramenta *RVIZ* do *ROS* para ilustrar a trajetória 3D do quadricóptero durante o pouso. Outra ferramenta útil foi o *RQT_MULTPLOT*, o qual ilustrou o gráfico 2D da posição e orientação. Atenta-se para o fato de utilizarmos neste teste simulado um marcador com *ID* = 273 e dicionário, tamanho de $0.5m$ $'ARUCO\_ORIGINAL'$ com o quadricóptero iniciando o pousando a cerca de 20 metros de altura.

5.6 Análise do Alcance de Detecção

Este experimento compara os intervalos de detecção do método proposto com uma técnica de detecção tradicional e a detecção por redes neurais convolucionais. Um marcador quadrado *ArUco* é impresso a partir de um dicionário tradicional *(ArUco original)* com id = 273 e tamanho de 50 cm, conforme ilustrado na Figura 4.2. Três vídeos com (total de 10445 quadros) são gravadas, começando em um local longe do marcador, onde o foco automático da câmera não detenha de uma imagem nítida. Em seguida a câmera é guiada pelo quadricóptero, se aproximando do marcador até realizar o pouso. A Figura 5.4 ilustra as taxas de detecção de positivos verdadeiros TPR com o objetivo de analisar o alcance da detecção do método proposto.

Dessa maneira, é possível comparar os resultados de *ArUco*, *CNN* e o método proposto na mesma sequência de vídeo. A Figura 5.4 ilustra a Taxa de Positivo Verdadeiro (TPR) de ambos os métodos em função da distância do marcador. Enquanto as linhas coloridas mostram o TPR para cada marcador individual. A área total do gráfico abrange à nossa abordagem híbrida, o qual é parametrizada a partir da linha azul turquesa. O método híbrido proposto pode detectar o alvo dentro de uma ampla faixa de distâncias, ou seja, 0,1 a 20 m, enquanto outras técnicas têm faixas de detecção mais limitadas ou irregulares.

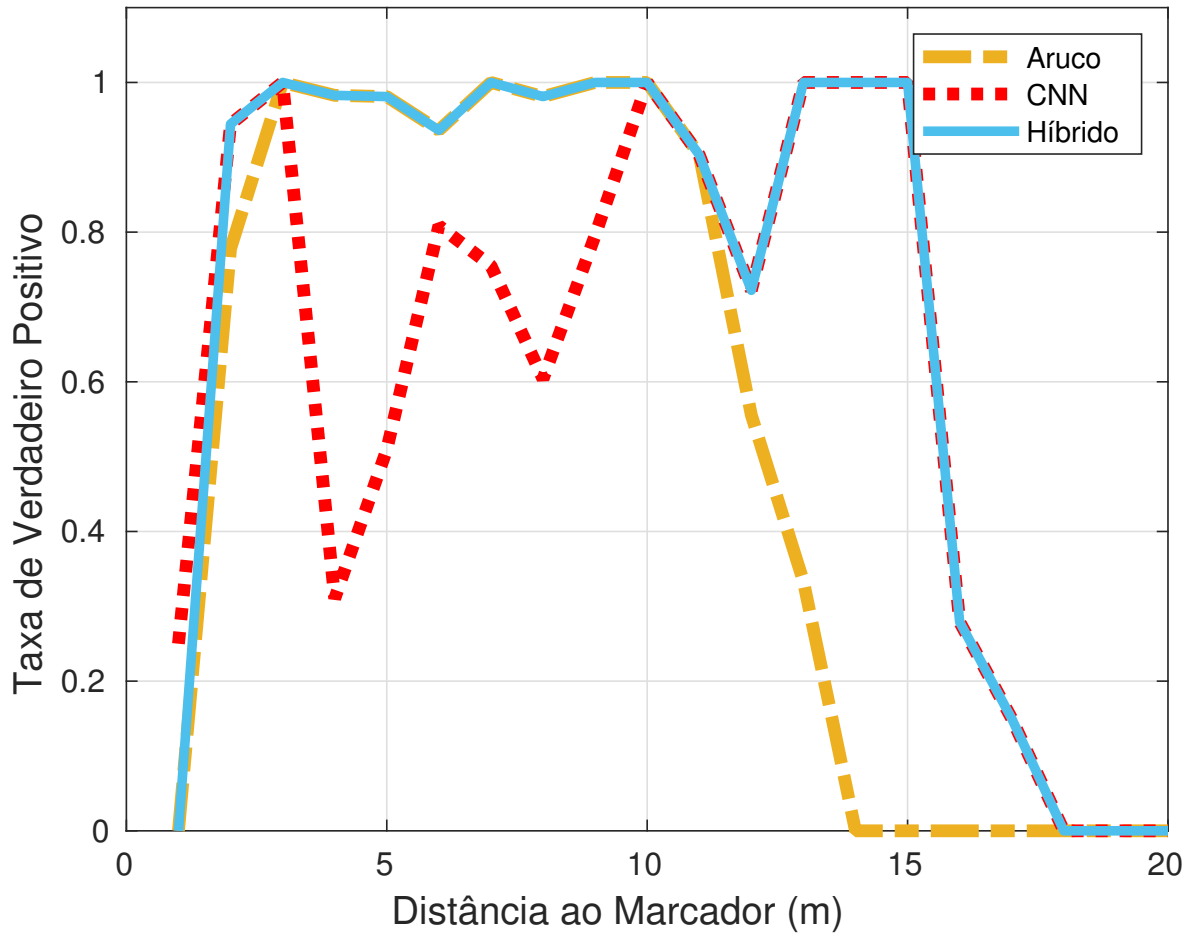

Figura 5.4: Taxas de detecção de positivos verdadeiros TPR, dependendo da distância do marcador de pouso. Cada linha colorida corresponde a um dos métodos utilizados. A linha de cor azul turquesa corresponde à faixa de detecção do método híbrido proposto, demonstrando uma ampla faixa de detecção.

5.7 Avaliação da Precisão da Localização

Outra preocupação é com a precisão da localização do método de estimativa de pose híbrido proposto. Obter um valor de referência *(Ground Truth)* de localização é indispensáveis para uma avaliação quantitativa do método. Para realizar testes confiáveis e de baixo custo, optamos por realizar testes sob iluminação interna controlada e usando 2 câmeras já disponíveis no laboratório de engenharia da UNICAMP.

Uma é a *Webcam Logitech C922*, na qual o método será aplicado, a outra é a câmera de rastreamento *Intel® RealSense™ T265*, para verificação de uma referência de posição confiável (*Ground Truth*). O *Realsense* possui resolução de $848x800$ e 30 FPS e inclui dois sensores de lente olho de peixe, uma IMU e uma *VPU Intel® Movidius™ Myriad™ 2.O T265* foi extensivamente testado e validado para desempenho, fornecendo menos de 1% de desvio de circuito fechado nas condições de uso pretendidas.

Para avaliar a precisão da localização, um caminho circular aleatório ao redor do marcador foi reconstruído. Um marcador fiducial quadrado de $Id = 273$ e tamanho $13{,}7cm$ foi usado para esses testes. Optamos por usar um marcador de tamanho pequeno para testar situações difíceis de detectar. Foram realizadas três experimentos com um total de 1042 imagens. Ao estimar a pose do marcador em tempo real, somos capazes de reconstruir o movimento relativo entre o marcador alvo e a câmera e assim comparar as técnicas em relação a referência de confiança.

Neste experimento, mantemos a face do marcador paralela à lente da câmera de estimação e movemos a câmera de acordo com o caminho projetado aleatoriamente. Todos os três testes de *Tracking Error* mostraram resultados similares, apresentando baixo desvio padrão e baixa variância quando comparado com os demais. Ilustrado na Figura 5.5 o resultado mais relevante, além disso é computado na Tabela 5.1 os resultados deste mesmo teste.

Tabela 5.1: O valor de Tracking Error em relação ao *Ground Truth*

	Tracking Error X	Tracking Error Y	Variância	Desvio Padrão
Híbrido	0.1083	0.1161	0.0035	0.0594
ArUco	0.0592	0.3690	0.0036	0.0602
CNN	0.6473	0.1742	0.0110	0.1049

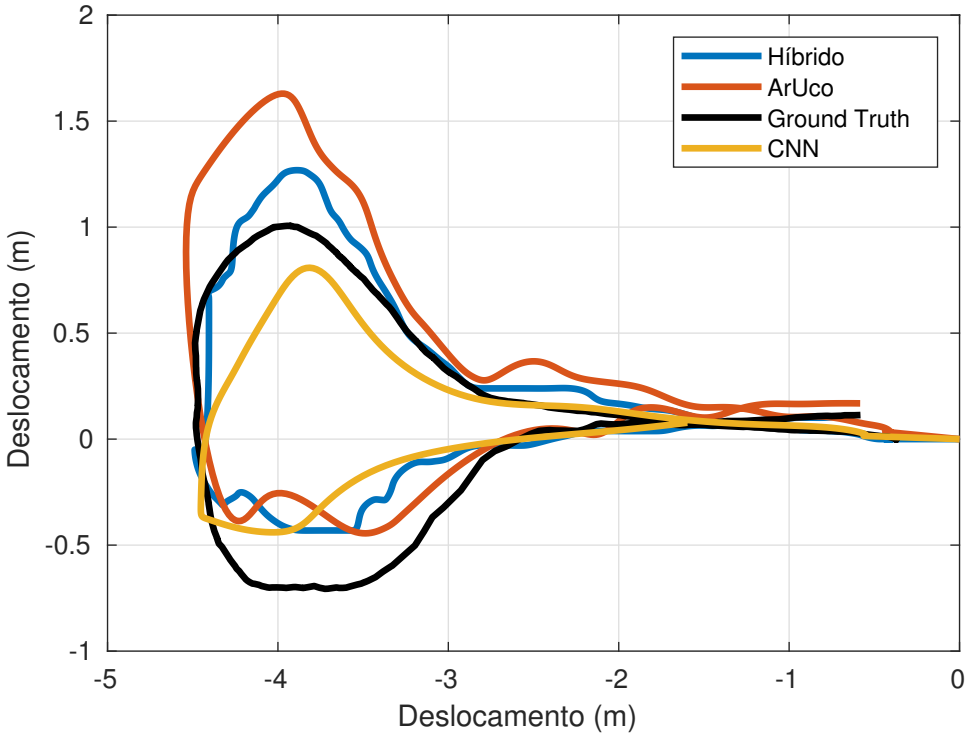

Figura 5.5: Experimentos de posição e precisão. Distância estimada nas direções x e y.

Os resultados do experimento mostram que o método de estimativa de pose híbrida é mais robusto do que as técnicas baseadas apenas em *ArUco* ou *CNN*. Limitado pela resolução da câmera, o detector visual *ArUco* pode falhar por longas distâncias, mas possui boa precisão.

Embora o detector *CNN* não consiga estimar com precisão em todos os casos, ele possui detecção relativamente estável em situações de difíceis detecção. Portanto, ao mesclar as duas técnicas, somos capazes de reforçar um ao outro e melhorar nossa estimativa em termos de alcance e precisão.

5.8 Detecção com Oclusão Parcial

O objetivo do experimento a seguir é analisar a robustez e taxa de detecção do método proposto sob diferentes porcentagens de oclusão. Um total de 6000 imagens foram tiradas de diferentes pontos de vista e distâncias (variando de 10 cm a 100 cm) sob iluminação interna controlada. Um marcador fiducial quadrado de $Id = 273$ e tamanho $9{,}5cm$ foi usado para esses experimentos, a fim de dificultar a detecção.

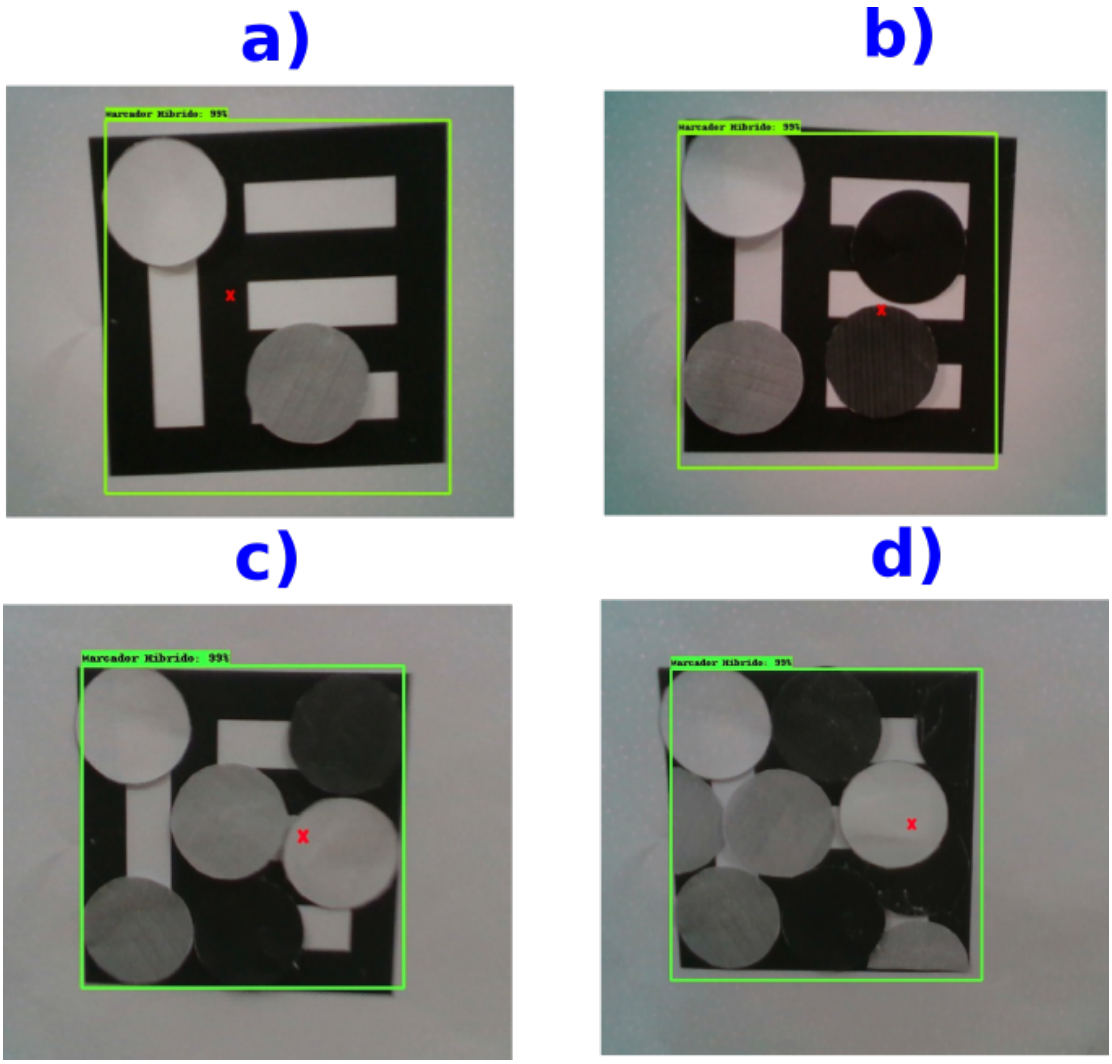

Figura 5.6: Algumas das imagens usadas para testar a detecção sob oclusão. Diferentes níveis de oclusão são adicionados à imagem: (a) 20%, (b) 40%, (c) 60%, (d) 80%

Para produzir uma oclusão sistemática e mais realista, papéis circulares de área padronizada são sobrepostos em locais aleatórios no marcador, como mostra a Figura 5.6. Os círculos foram padronizados com um valor aproximado de 10% da área total do marcador. Com base nesse padrão, é possível conhecer de forma aproximada a porcentagem da área com oclusão. A cada aumento da oclusão, os papéis circulares são realocados aleatoriamente. A cor de um círculo é selecionada aleatoriamente com diferentes tons de cinza.

Para cada porcentagem, geramos 6 testes com 300 imagens reais (1800 para cada porcentagem). Foram testadas quatro porcentagens: 20%, 40%, 60% e 80%, para que os níveis de oclusão resultantes sejam distribuídos uniformemente no intervalo [1,100] %. Acima de 80 %, a detecção se torna quase impossível porque desfigura muito a imagem. Os resultados dos experimentos são ilustrados na Figura 5.7. Logo, verifica-se uma boa detecção até 40-60% de oclusão seguindo os critérios de erro normalizado.

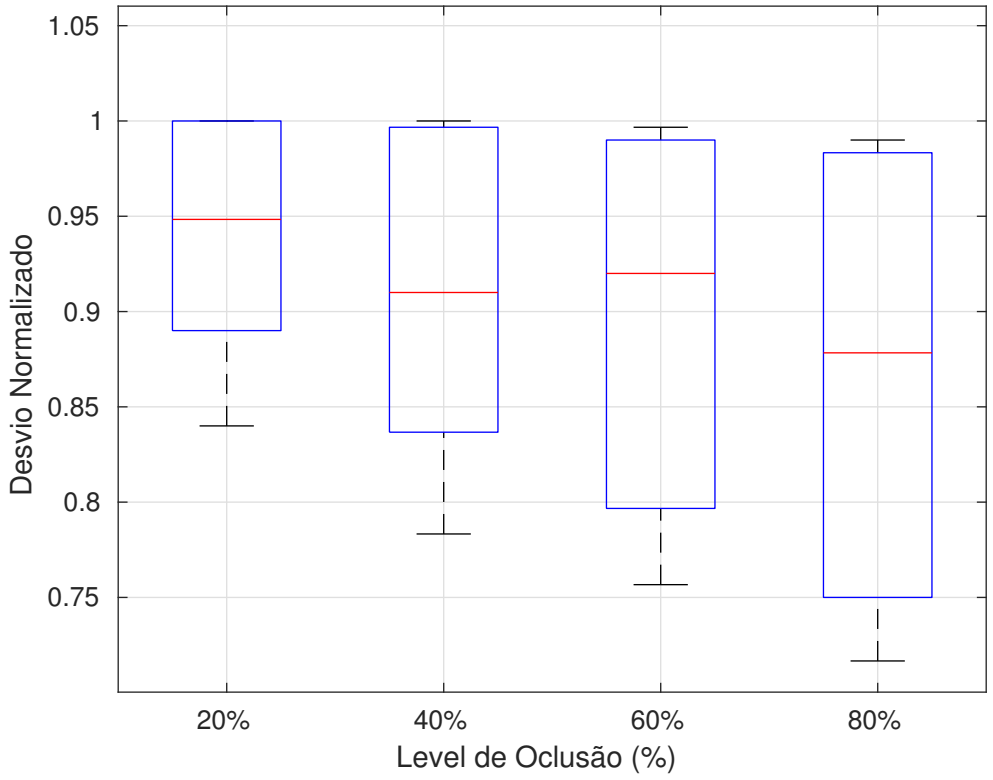

Figura 5.7: Média (vermelho) e desvio padrão (azul) do erro normalizado para diferentes níveis de oclusão.

5.9 Tempo de Computação

Verifica-se que as etapas do método proposto nesta tese adicionam uma sobrecarga relativamente pequena ao tempo total de computação. A etapa inicial "Detecção de marcador (ArUco)", que é igual à detecção de marcador tradicional, é o processo mais demorado. Deve-se observar que "Detecção de marcador (CNN)"também se apresenta como um dos processos mais demorados, entretanto é processado em um nó paralelo para não prejudicar o tempo de computação. Além disso, observa-se que só é necessário utilizar o processo de "Fusão Pose com Filtro de Kalman"mais robusto quando ambas as técnicas de detecção e estimação são realizadas com sucesso. Portanto, o método proposto adicionará apenas uma quantidade insignificante de tempo ao cálculo total.

Tabela 5.2: Tempos médios de computação (em milissegundos) das diferentes etapas envolvidas na detecção e rastreamento do método proposto.

Processo	Tempo Médio (ms)
Detecção de Marcadores (ArUco)	29.8
Estimação Pose da Câmera (ArUco)	0.4
Detecção de Marcadores (CNN) / Paralelo	25.2
Estimação Pose da Câmera (CNN) / Paralelo	0.2
Fusão Pose com Filtro de *Kalman*	0.9
Pós-Filtragem *Butterworth*	1.1

5.10 Discursão de Resultados

Os testes simulados foram essenciais para verificar a funcionalidade do algoritmo, bem como a implementação do filtro *Kalman*. O desempenho do método foi validado de forma efetiva em situações críticas de detecção e estimação, permitindo também o ajuste dos valores de covariância para estimativas mais suaves. Logo, o método proposto se mostrou competente em realizar missões de aterrissagem autônoma em ambiente simulado.

Foram realizados três tipos de testes reais que visaram avaliar: capacidade de detecção à distância; robustez à oclusão parcial; precisão na estimativa da pose e velocidade do método proposto. Totalizando ao final, nove experimentos que qualificaram e quantificaram o desempenho do método proposto com relação a técnica baseada em visão computacional (*ArUco*) e redes neurais (CNN).

No experimento de alcance de detecção, foi testado a taxa de positivos verdadeiros (TPR) em relação a distância. O método proposto atendeu a uma ampla faixa de distâncias entre 0,1 - 20 m, enquanto as outras se mostraram limitadas ou irregulares. Em relação a precisão da localização, o método híbrido apresentou mais robustez seguindo os critérios de *Tracking Erro*, variância e desvio padrão baixos. Os resultados são atribuídos a boa precisão do detector visual *ArUco* e a estabilidade e detecção de longo alcance da *CNN* em situações difíceis como vibração. Os experimentos de detecção com oclusão apresentaram bom desempenho em situações de oclusão severa, obtendo um erro normalizado aceitável até cerca de 40-60% de oclusão. Acima de 80% a detecção se torna quase impossível, desfigurando muito a imagem. Porém, apresentou resultado bem superior quando se compara com a técnica clássica *ArUco*, cuja detecção é impossível em situações de baixíssima oclusão.

Verifica-se que os resultados dos experimentos foram satisfatórios a ponto de qualificar o quadricóptero a realizar pouso autônomo, criando condições estáveis para missões diversas como por exemplo, carregamento autônomo. Ao final dos experimentos, realizou-se mais dezenas de testes práticos voltados à aplicação de pouso autônomo, exemplificando uma aplicação comercial real. Ao logo dos testes, observou-se que o método proposto é de fácil implementação, com tempo de reposta rápida para atuação de um controle de alinhamento de alto nível. Por utilizar um quadricóptero comercial, o controle de baixo nível dos motores e o *gimbal* da câmera já vem embarcado em seu *hardware*, contribuindo para estabilizar o quadricóptero em situações vento elevado ou vibração excessiva. Contudo, ainda é visível os desafios de pertubação com vento e vibração, desafios esses superados ao aplicarmos o método proposto. O problema da iluminação *outdoor* ainda se mostra um grande desafio por dificultar a detecção dos marcadores de pouso quando submetida a uma variação abrupta.

Destaca-se a aplicação do método proposto em missões de navegação autônoma, voltada para redes de energia elétrica. A aplicação foi voltada para uma pesquisa de doutorado, realizado em ambiente controlado e com estrutura não energizada. A aplicação com quadricóptero efetuou as seguintes atividades: levantou voo em um marcador posicionado em um ponto x; navegou em um circuito de rede de transmissão feito em maquete; por fim, pousou em um segundo marcador posicionado em um ponto y.

6 Conclusão

O presente estudo é complementar aos trabalhos do grupo de automação e navegação robótica do Laboratório de Processamento de Sinais e Análise de Sistemas Dinâmicos do Departamento de Sistemas Integrados da Faculdade de Engenharia Mecânica da UNICAMP.

Foram realizados estudos sobre os desafios das técnicas de localização e navegação robótica utilizadas para pouso autônomo de quadricópteros. Foi realizado também uma revisão bibliográfica sobre as principais técnicas de decolagem e pouso, bem como a utilização dos principais marcadores fiduciais. Estes estudos evidenciaram as limitações na detecção e estimação de posição em casos de longas distâncias, oclusão parcial ou situações com alta vibração com câmeras.

As informações técnicas coletadas permitiram estabelecer parâmetros, documentações e banco de imagens (*datasets*) para realização de novos estudos sobre o tema. Com isso, foi possível desenvolver, simular e validar um método capaz de identificar e aterrissar de forma autônoma um quadricóptero em um alvo estático de identificação única. O método híbrido proposto fornece informações de localização mais precisas de forma a superar limitações do estado da arte.

O método se mostrou único por utilizar técnicas de redes neurais convolucionais (CNN) de forma complementar a técnica clássica das *tags* do tipo *ArUco*. Criando assim, um estimador híbrido a partir da fusão e filtragem das duas técnicas utilizando o filtro de *Kalman* e *Butterworth*. Resultando em uma tarefa de detecção e estimação de posição mais flexível e com resposta de atuação mais rápida e confiável quando comparada com o atual estado da arte.

O método utiliza apenas recursos de sensor visual monocular e algoritmos computacionais sem necessitar de nenhuma informação prévia sobre a localização da plataforma de pouso. Poucas técnicas seguem este mesmo princípio, onde em sua maioria se utilizam de alta tecnologia de sensoriamento com IMU, GPS ou *Laser*. Dependendo de sua precisão, podem resultar em custos elevados ou bloqueio de dados quando submetidos a navegação próximo de prédios ou redes elétricas.

A programação é composta pelas linguagens *python*, *C++* e *XML* integradas pelos sistema operacional *ROS*, que permite a simulação 3D (Gazebo) em ambiente virtual, bem como em ambientes reais. Os experimentos realizados em ambientes reais utilizaram o quadricóptero comercial

Bebop 2 e foram executados pela equipe de pesquisa em robótica do Laboratório de Processamento de Sinais, da Escola de Engenharia Mecânica da Universidade de Campinas (UNICAMP), Brasil.

Finalmente, são demonstradas as vantagens e desvantagens do uso do método. Sua única restrição está relacionada à identificação da orientação do marcador em situações de oclusão parcial e longas distâncias. Mesmo assim, a detecção é superior quando comparada a técnicas similares devido à sua alta taxa de detecção com *CNN* em situações de oclusão parcial e longas distâncias. É válido ressaltar a criação de um banco de dados (*dataset*) de imagens em ambientes virtuais e reais para a criação de novos modelos de redes neurais e testes de validação de novas técnicas. A documentação, vídeos *datasets* estão disponíveis na plataforma *GitHub* e link via *Google Drive*.

6.1 Trabalhos Futuros

- Aprofundar estudos sobre holografia para melhorar técnica de detecção usando redes neurais. Aprimorando a identificação das bordas do marcador, pode-se extrair informações como orientação do marcador em relação a câmera.
- Implementar o método desenvolvido em um novo *frame* de quadricóptero construído de forma personalizada. Desta forma, será possível implementar uma nova disposição de câmera e *hardware* para ser testado junto ao método.

Referências

Ananthakrishnan, U. S., Akshay, N., Manikutty, G., e Bhavani, R. R. (2017). Control of quadrotors using neural networks for precise landing maneuvers. In S. S. Dash, K. Vijayakumar, B. K. Panigrahi, e S. Das (Eds.), *Artificial intelligence and evolutionary computations in engineering systems* (pp. 103–113). Singapore: Springer Singapore.

Araújo, F. H., Carneiro, A., Silva, R. R., Medeiros, F. N., e Ushizima, D. M. (2017). Redes neurais convolucionais com tensorflow: Teoria e prática. *SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. III Escola Regional de Informática do Piauí. Livro Anais-Artigos e Minicursos*, *1*, 382–406.

Artero, A. O., e Tomaselli, A. M. G. (2000). Limiarização automática de imagens digitais. In *Boletim de ciências geodésicas* (p. 38-48).

Bonnard, Q., Lemaignan, S., Zufferey, G., Mazzei, A., Cuendet, S., Li, N., ... Dillenbourg, P. (2013). *Chilitags 2: Robust fiducial markers for augmented reality and robotics.* CHILI, EPFL, Switzerland. Retrieved from `http://chili.epfl.ch/software`

Chavez, G. M. (2016). *Autonomous landing on moving platforms* (Unpublished doctoral dissertation). King Abdullah University of Science and Technology, Thuwal, Kingdom of Saudi Arabia.

Claus, D., e Fitzgibbon, A. W. (2005, Jan). Reliable automatic calibration of a marker-based position tracking system. In *2005 seventh ieee workshops on applications of computer vision (wacv/motion'05) - volume 1* (Vol. 1, p. 300-305). doi: 10.1109/ACVMOT.2005.101

Costa, E. B. (2012). *Algorítmos de Controle Aplicados à Estabilização do Voo de um Quadrotor* (Unpublished master's thesis). Universidade Federalde Juiz de Fora, UFJF.

Dorfmuller, K., e Wirth, H. (1998). Real-time hand and head tracking for virtual environments using infrared beacons. In *in proceedings captech'98. 1998* (pp. 113–127). Springer.

Doxygen-OpenCVs. (2018). *Camera calibration and 3d reconstruction: Detailed description.* Retrieved from `https://docs.opencv.org/3.4.1/d9/d0c/group__calib3d.html` (Pinhole Description – Jan/2019, Link: <https://docs.opencv.org/3.4.1/d9/d0c/group$_{calib3d.html}$ >)

Duan, H., Shao, S., Su, B., e Zhang, L. (2010). New development thoughts on the bioinspired intelligence based control for unmanned combat aerial vehicle. *Sci. China Technol. Sci. 53*, *8*, 20252031.

Faigl, J., Krajnik, T., Chudoba, J., Preucil, L., e Saska, M. (2013, May). Low-cost embedded system for relative localization in robotic swarms. In *2013 ieee international conference on robotics and automation* (p. 993-998). doi: 10.1109/ICRA.2013.6630694

Falanga, D., Zanchettin, A., Simovic, A., Delmerico, J., e Scaramuzza, D. (2017, Oct). Vision-based autonomous quadrotor landing on a moving platform. In *2017 ieee international symposium on safety, security and rescue robotics (ssrr)* (p. 200-207). doi: 10.1109/SSRR.2017.8088164

Fiala, M. (2005, Oct). Comparing artag and artoolkit plus fiducial marker systems. In *Ieee international workshop on haptic audio visual environments and their applications* (p. 6 pp.-). doi: 10.1109/HAVE.2005.1545669

Forster, C., Pizzoli, M., e Scaramuzza, D. (2014, May). Svo: Fast semi-direct monocular visual odometry. In *2014 ieee international conference on robotics and automation (icra)* (p. 15-22). doi: 10.1109/ICRA.2014.6906584

Foundation, O. S. R. (2019). *Open source robotics foundation. ros.* Retrieved from `http://wiki.ros.org/` (http://wiki.ros.org/ (Visited on January 10, 2019))

Garrido-Jurado, S., noz Salinas, R. M., Madrid-Cuevas, F., e Marín-Jiménez, M. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recogni-*

tion, *47*(6), 2280 - 2292. Retrieved from http://www.sciencedirect.com/science/article/pii/S0031320314000235 doi: http://dx.doi.org/10.1016/j.patcog.2014.01.005

Garrido-Jurado, S., noz Salinas, R. M., Madrid-Cuevas, F., e Medina-Carnicer, R. (2016). Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, *51*, 481 - 491. Retrieved from http://www.sciencedirect.com/science/article/pii/S0031320315003544 doi: http://dx.doi.org/10.1016/j.patcog.2015.09.023

Girshick, R., Donahue, J., Darrell, T., e Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 580–587).

Guo, P., Li, X., e Gui, Y. (2014). Airborne vision-aided landing navigation system for fixed-wing UAV. *In: 2014 12th International Conference on Signal Processing (ICSP). IEEE,*, *1*, 1215 - 1220.

He, S., Wang, H., Zhang, S., e Tian, B. (2019). Vision based autonomous tracking and landing on an arbitrary object of quadrotor. In *2019 ieee 9th annual international conference on cyber technology in automation, control, and intelligent systems (cyber)* (pp. 474–479).

Hollemans, M. (2018). *Quick recap of version 1.* Retrieved from https://machinethink.net/blog/mobilenet-v2/ (https://machinethink.net/blog/mobilenet-v2/ (Visited on January 10, 2020))

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Jayatilleke, L., e Zhang, N. (2013, April). Landmark-based localization for unmanned aerial vehicles. In *2013 ieee international systems conference (syscon)* (p. 448-451).

Kaltenbrunner, M., e Bencina, R. (2007). reactivision: A computer-vision framework for table-based tangible interaction. In *Proceedings of the 1st international conference on tangible and*

embedded interaction (pp. 69–74). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/1226969.1226983 doi: 10.1145/1226969.1226983

Kato, H., e Billinghurst, M. (1999, Oct). Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings 2nd ieee and acm international workshop on augmented reality (iwar'99)* (p. 85-94). doi: 10.1109/IWAR.1999.803809

Keras. (2020). *Keras api reference.* Retrieved from https://keras.io/api/ (https://keras.io/api/ (Visited on July 01, 2020))

Khaleghi, B., Khamis, A., Karray, F. O., e Razavi, S. N. (2013). Multisensor data fusion: A review of the state-of-the-art. *Information fusion*, *14*(1), 28–44.

Kim, D. K., e Chen, T. (2015). Deep neural network for real-time autonomous indoor navigation. *CoRR*, *abs/1511.04668*. Retrieved from http://arxiv.org/abs/1511.04668

Li, B., Wu, J., Tan, X., e Wang, B. (2020). Aruco marker detection under occlusion using convolutional neural network. In *2020 5th international conference on automation, control and robotics engineering (cacre)* (pp. 706–711).

Lima, G. A., Bravo, D. T., e de Araújo, S. A. (2020). Utilização de redes neurais convolucionais para a detecção de objetos em imagens aéreas adquiridas por drones. *Brazilian Journal of Development*, *6*(7), 50702–50713.

Lippiello, V., Loianno, G., e Siciliano, B. (2011, Dec). Mav indoor navigation based on a closed-form solution for absolute scale velocity estimation using optical flow and inertial data. In *2011 50th ieee conference on decision and control and european control conference* (p. 3566-3571). doi: 10.1109/CDC.2011.6160577

Measurements, C. S., Knyaz, V. A., Group, H. O., e Sibiryakov, R. V. (1998). The development of new coded targets for automated point identification and non -. In *3d surface measurements, international archives of photogrammetry and remote sensing, vol. xxxii, part 5* (pp. 80–85).

Munoz-Salinas, R. (2012). *Aruco: a minimal library for augmented reality applications based on opencv.* Universidad de C´ordoba.

Mur-Artal, R., e Tardós, J. D. (2016). ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *CoRR*, *abs/*, 1610.06475. Retrieved from http://arxiv.org/abs/1610.06475

Müller, L. (2020). *Drones já podem fazer entregas no brasil.* Retrieved from https://www.tecmundo.com.br/produto/156124-drones-fazer-entregas-brasil.htm (https://www.tecmundo.com.br/produto/156124-drones-fazer-entregas-brasil.htm (Visited on Dezember 10, 2020))

Naimark, L., e Foxlin, E. (2002, Oct). Circular data matrix fiducial system and robust image processing for a wearable vision-inertial self-tracker. In *Proceedings. international symposium on mixed and augmented reality* (p. 27-36). doi: 10.1109/ISMAR.2002.1115065

Otsu, N. (1979). A threshold selection method from grey-level histograms. In *Ieee transactions on systems, man, and cybernetics* (Vol. 9, p. 377-393).

Pestana, J., Sanchez-Lopez, J. L., de la Puente, P., Carrio, A., e Campoy, P. (2016, Dec 01). A vision-based quadrotor multi-robot solution for the indoor autonomy challenge of the 2013 international micro air vehicle competition. *Journal of Intelligent & Robotic Systems*, *84*(1), 601–620. Retrieved from https://doi.org/10.1007/s10846-015-0304-1 doi: 10.1007/s10846-015-0304-1

Polvara, R., Patacchiola, M., Sharma, S., Wan, J., Manning, A., Sutton, R., e Cangelosi, A. (2017). Autonomous quadrotor landing using deep reinforcement learning. *arXiv preprint arXiv:1709.03339*.

Quigley, M., Conley, K., Gerkey, B. P., Faust, J., Foote, T., Leibs, J., ... Ng, A. Y. (2009). Ros: an open-source robot operating system. In *Icra workshop on open source software.*

Rekimoto, J., e Ayatsuka, Y. (2000). Cybercode: Designing augmented reality environments with visual tags. In *Proceedings of dare 2000 on designing augmented reality environments* (pp. 1–10). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/354666.354667 doi: 10.1145/354666.354667

Ribo, M., Pinz, A., e Fuhrmann, A. L. (2001, May). A new optical tracking system for virtual and augmented reality applications. In *Imtc 2001. proceedings of the 18th ieee instrumentation and measurement technology conference. rediscovering measurement in the age of informatics (cat. no.01ch 37188)* (Vol. 3, p. 1932-1936 vol.3). doi: 10.1109/IMTC.2001.929537

Rodriguez-Ramos, A., Sampedro, C., Bavle, H., de la Puente, P., e Campoy, P. (2019, Feb 01). A deep reinforcement learning strategy for uav autonomous landing on a moving platform. *Journal of Intelligent & Robotic Systems*, *93*(1), 351–366. Retrieved from https://doi.org/10.1007/s10846-018-0891-8 doi: 10.1007/s10846-018-0891-8

Rohmer, E., Singh, S. P., e Freese, M. (2013). V-rep: A versatile and scalable robot simulation framework. In *Intelligent robots and systems (iros), 2013 ieee/rsj international conference on* (pp. 1321–1326).

Rohs, M., e Gfeller, B. (2004). Using camera-equipped mobile phones for interacting with real-world objects. In *Advances in pervasive computing* (pp. 265–271).

Romero-Ramirez, F., Muñoz-Salinas, R., e Medina-Carnicer, R. (2019, 11). Fractal markers: a new approach for long-range marker pose estimation under occlusion. *IEEE Access*, *PP*, 1-1.

Romero-Ramirez, F. J., MuÃ±oz-Salinas, R., e Medina-Carnicer, R. (2018). Speeded up detection of squared fiducial markers. *Image and Vision Computing*, *76*, 38 - 47. Retrieved from http://www.sciencedirect.com/science/article/pii/S0262885618300799 doi: https://doi.org/10.1016/j.imavis.2018.05.004

Santos, M. F. dos. (2014). *Controle Tolerante a Falhas de um Sistema de Propulsão de Hexacópteros* (Unpublished master's thesis). Universidade Federalde Juiz de Fora, UFJF.

Tang, C., e Lai, Y.-C. (2020). Deep reinforcement learning automatic landing control of fixed-wing aircraft using deep deterministic policy gradient. In *2020 international conference on unmanned aircraft systems (icuas)* (pp. 1–9).

Vetrella, A. R., Sa, I., Popović, M., Khanna, R., Nieto, J., Fasano, G., ... Siegwart, R. (2018). Improved tau-guidance and vision-aided navigation for robust autonomous landing of uavs. In M. Hutter e R. Siegwart (Eds.), *Field and service robotics* (pp. 115–128). Cham: Springer International Publishing.

Zingg, S., Scaramuzza, D., Weiss, S., e Siegwart, R. (2010, May). Mav navigation through indoor corridors using optical flow. In *2010 ieee international conference on robotics and automation* (p. 3361-3368). doi: 10.1109/ROBOT.2010.5509777