UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Paulo Victor Galvão Simplício

Controle Robusto e Inteligente de Quadricópteros Sujeitos a Distúrbios de Vento

São Carlos

2021

Paulo Victor Galvão Simplício

Controle Robusto e Inteligente de Quadricópteros Sujeitos a Distúrbios de Vento

Dissertação apresentada à Escola de Engenharia de São Carlos da Universidade de São Paulo, para obtenção do título de Mestre em Ciências - Programa de Pós-Graduação em Engenharia Elétrica.

Área de concentração: Sistemas Dinâmicos

Orientador: Prof. Dr. Marco Henrique Terra

São Carlos

2021

Trata-se da versão corrigida da dissertação. A versão original se encontra disponível na EESC/USP que aloja o programa de Pós-Graduação de Engenharia Elétrica.

Ficha catalográfica elaborada pela Biblioteca Prof. Dr. Sérgio Rodrigues Fontes da EESC/USP com os dados inseridos pelo(a) autor(a).

Simplício, Paulo Victor Galvão
S612c Controle Robusto e Inteligente de Quadricópteros Sujeitos a Distúrbios de Vento / Paulo Victor Galvão Simplício; orientador Marco Henrique Terra. São Carlos, 2021.

Dissertação (Mestrado) - Programa de Pós-Graduação em Engenharia Elétrica e Área de Concentração em Sistemas Dinâmicos -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2021.

1. Controle Robusto. 2. Quadricóptero. 3. Redes Neurais. 4. Rastreamento de Trajetória. 5. Estimativa de Distúrbios. I. Título.

Eduardo Graziosi Silva - CRB - 8/8907

FOLHA DE JULGAMENTO

Candidato: Bacharel **PAULO VICTOR GALVÃO SIMPLÍCIO.**

Título da dissertação: "Controle robusto e inteligente de quadricópteros sujeitos a distúrbios de vento".

Data da defesa: 16/12/2021.

Comissão Julgadora	**Resultado**
Prof. Titular **Marco Henrique Terra** **(Orientador)** (Escola de Engenharia de São Carlos – EESC/USP)	aprovado
Prof. Dr. **Guilherme Vianna Raffo** (Universidade Federal de Minas Gerais/UFMG)	aprovado
Prof. Dr. **Luciano Cunha de Araújo Pimenta** (Universidade Federal de Minas Gerais/UFMG)	aprovado

Coordenador do Programa de Pós-Graduação em Engenharia Elétrica:
Prof. Associado **João Bosco Augusto London Junior**

Presidente da Comissão de Pós-Graduação:
Prof. Titular **Murilo Araujo Romero**

Este trabalho é dedicado à minha mãe, Maria,
e à minha tia, Vônica, por todo apoio e incentivo.

AGRADECIMENTOS

Agradeço a Deus, por me proporcionar saúde e resiliência, tornando possível a realização deste sonho.

À minha família, em especial à minha mãe e minha tia, por sempre fazerem o possível e impossível para que eu possa realizar meus sonhos.

Ao meu orientador, Prof. Dr. Marco Henrique Terra, pela oportunidade, orientação, paciência, colaboração e conselhos durante a execução deste trabalho.

Ao Prof. Dr. Roberto Santos Inoue, pela disposição, auxílio e contribuições a este projeto.

À minha namorada, Beatriz, pelo companheirismo, paciência e incentivos ao longo desses anos.

Aos colegas do Laboratório de Sistemas Inteligentes (LASI) e do Centro de Robótica da Universidade de São Paulo (CRob), por todo apoio, troca de experiências e companheirismo durante todo curso de mestrado.

Aos professores e colaboradores do Departamento de Engenharia Elétrica e de Computação, pelos incentivos e pelo suporte.

Aos meus grandes amigos, que mesmos distantes estiveram ao meu lado nesta jornada.

À instituição Universidade de São Paulo (USP), pela infraestrutura concedida.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES - Proc. 88887.509571/2020-00) pela concessão de bolsa de mestrado e apoio financeiro durante projeto.

Ao Instituto Nacional de Ciência e Tecnologia para Sistemas Autônomos Cooperativos (InSAC) pelo financiamento dos materiais utilizados neste trabalho e infraestrutura concedida.

"O mais importante da vida não é a situação em que estamos, mas a direção para a qual nos movemos."
Oliver Wendell Holmes

RESUMO

SIMPLÍCIO, P. V. G. **Controle Robusto e Inteligente de Quadricópteros Sujeitos a Distúrbios de Vento**. 2021. 129p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2021.

O desenvolvimento de sistemas de controle aplicados a quadricópteros que atuam em ambientes hostis é um desafio complexo. Isto porque, para obter um bom desempenho de voo autônomo com robustez, deve-se projetar um sistema de controle que considere não só distúrbios externos como também outros fatores que possam depreciar a qualidade de voo. Recentemente, uma das técnicas mais utilizadas é o controle inteligente composto por redes neurais artificiais (RNAs). Neste caso, as redes são combinadas com controladores, formando uma arquitetura capaz de se adaptar ao ambiente de operação e, consequentemente, reduzir o erro de seguimento de trajetória. Diante disso, este trabalho tem o objetivo de desenvolver arquiteturas inteligentes para controle de posição de quadricópteros, melhorando o desempenho de voo durante o rastreamento de trajetória. As arquiteturas propostas combinam um regulador linear quadrático robusto (RLQR) com redes neurais profundas. Além disso, o desempenho das arquiteturas propostas é avaliado através de um estudo comparativo, utilizando outros três controladores presentes na literatura: regulador linear quadrático (LQR), proporcional-integral-derivativo (PID) e linearização por realimentação (LR). As arquiteturas foram desenvolvidas utilizando a plataforma *robot operating system* (ROS) e os experimentos foram realizados primeiramente em um ambiente simulado e, em seguida, utilizando um quadricóptero comercial, o Parrot™ Bebop 2.0. Para ambos os casos, foram realizados dois conjuntos de experimentos, com e sem a aplicação de distúrbio de vento na aeronave. Os resultados mostraram que a utilização de redes neurais combinadas com controladores, robustos ou não, melhoram o desempenho de voo de quadricópteros. Isto foi visualizado tanto para condições normais de voo quanto para voos em que o quadricóptero foi submetido à influência de distúrbios de vento.

Palavras-chave: Controle Robusto. Quadricóptero. Redes Neurais. Rastreamento de Trajetória. Estimativa de Distúrbios.

ABSTRACT

SIMPLÍCIO, P. V. G. **Robust and Intelligent Control of Quadrotors Subject to Wind Disturbances**. 2021. 129p. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2021.

The development of control systems applied to quadrotors operating in hostile environments is a complex challenge. This is because, to obtain good autonomous flight performance with robustness, one must design a control system that considers external disturbances and factors that can depreciate flight quality. Nowadays, one of the techniques on the rise is intelligent control composed of artificial neural networks (ANNs). In this case, the networks are combined with controllers, forming an architecture capable of adapting to the operating environment and, consequently, reduce trajectory tracking error. In this context, this work aims to develop intelligent architectures to improve the flight performance of autonomous quadrotors. The proposed architectures combine a Robust Linear Quadratic Regulator (RLQR) with deep neural networks. Furthermore, the performance of the proposed architectures is evaluated through a comparative study using three other controllers present in the literature: linear quadratic regulator (LQR), proportional-integral-derivative (PID) and feedback linearization (FL). The architectures were developed using the robot operating system (ROS) platform, and the experiments were performed first in a simulated environment and then with a commercial quadrotor, the Parrot^{TM} Bebop 2.0. Two sets of experiments were performed for both cases, with and without applying wind disturbance to the aircraft. The results showed that using neural networks combined with controllers, robust or not, improves quadrotor flight performance. The improvement was seen both for normal flight conditions and for flights where the quadrotor was subjected to wind disturbances.

Keywords: Robust control. Quadrotor. Neural Networks. Trajectory Tracking. Disturbance Estimation.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Adam	*Adaptive Moment Estimation*
AdaGrad	*Adaptive Gradient*
AdaDelta	*Adaptive Learning Rate*
ADRC	*Active Disturbance Rejection Control*
CHADC	*Composite Hierarchical Antidisturbance Control*
CRob	Centro de Robótica da Universidade de São Paulo
DAC	*Disturbance Accommodation Control*
DOB	*Disturbance Observer*
DOBC	*Disturbance Observer-Based Control*
DUEA	*Disturbance and Uncertainty Estimation and Attenuation*
DNN	*Deep Neural Network*
EAM	Erro Absoluto Médio
EHGSO	*Extended High-Gain State Observer*
ESC	*Electronic Speed Controller*
ESO	*Extended State Observer*
EQM	Erro Quadrático Médio
FK	Filtro de Kalman
GD	Gradiente Descendente
IA	Inteligência Artificial
IMU	*Inertial Measurement Unit*
LQR	Regulador Linear Quadrático
LR	Linearização por Realimentação
MBGD	*Mini-batch Gradient Descent*
MMQ	Método dos Mínimos Quadrados

MPC	*Model Predictive Control*
Nadam	*Nesterov-accelerated Adaptive Moment Estimation*
PID	Proporcional-Integral-Derivativo
PMC	Perceptron Multicamadas
POB	*Perturbation Observer*
PWM	*Pulse Width Modulation*
ReLU	*Rectified Linear Unit*
RLQR	Regulador Linear Quadrático Robusto
RNA	Rede Neural Artificial
ROS	*Robot Operating System*
SGD	*Stochastic Gradient Descent*
UIO	*Unknown Input Observer*
USP	Universidade de São Paulo
VANT	Veículo Aéreo Não Tripulado

LISTA DE SÍMBOLOS

ϕ	Ângulo de arfagem
θ	Ângulo de rolagem
ψ	Ângulo de guinada
ν	Vetor de controle para o Bebop 2.0
γ	Parâmetros identificados do modelo dinâmico
Γ_A	Matriz de parâmetros identificados relacionados à matriz A do modelo
Γ_B	Matriz de parâmetros identificados relacionados à matriz B do modelo
q_b	Vetor de estados (sistema de coordenadas locais)
q_g	Vetor de estados (sistema de coordenadas globais)
A	Matriz de estado contínua
B	Matriz de entrada contínua
R_t	Matriz de rotação
δA	Matriz contínua de incerteza do estado
δB	Matriz contínua de incerteza da entrada
d	Distúrbio externo
B_d	Matriz que mapeia o distúrbio externo
m	Massa do quadricóptero
I_z	Momento de inercia com respeito ao eixo Z
$\tilde{q}$	Erro de seguimento de trajetória
q^d	Vetor de trajetória desejada
u	Entrada de controle virtual
I_n	Matriz identidade
$\tilde{x}$	Vetor de estado aumentado do erro
y	Saída de um sistema dinâmico

X	Matriz de variáveis independentes
θ	Vetor de parâmetros desconhecidos
ϵ	Vetor de erro
$\hat{\theta}$	Vetor de parâmetros conhecidos
J	Função de custo
υ	Coeficiente de identificação para o eixo X
σ	Coeficiente de identificação para o eixo Y
τ	Coeficiente de identificação para o eixo Z
ι	Coeficiente de identificação para rotação em torno do eixo Z
$\hat{\gamma}$	Vetor de parâmetros desconhecidos
$\mathcal{X}$	Conjunto de amostras rotuladas
a	Saída do neurônio artificial
$g(a)$	Função de ativação
W	peso sináptico da rede neural
b	Limiar de ativação da rede neural
$\mathcal{E}$	Erro quadrático médio
y_p	Saída fornecida pela rede
y_d	Saída desejada da rede
η	Taxa de aprendizado
∇	Operador de gradiente
υ_t	Termo de *momentum*
β_1	Constante de ajuste do efeito do *momentum*
β_2	Termo de regulação para o *RMSProp*
s^{dw}	Termo de *momentum* de segunda ordem
$\widehat{\upsilon}$	Atualização do momento de primeira ordem
$\widehat{s}^{dw}$	Atualização do momento de segunda ordem

F	Matriz de estado discreta
G	Matriz de entrada discreta
δF	Matriz discreta de incerteza do estado
δG	Matriz discreta de incerteza da entrada
K	Ganho do controlador
L	Matriz do sistema em malha fechada com RLQR/LQR
Q	Matriz de ponderação dos estados
R	Matriz de ponderação da entrada de controle
P	Matriz de ponderação para o RLQR/LQR
$\mathcal{J}$	Função de custo para o RLQR/LQR
μ	Parâmetro de penalidade
E_{F_i}	Matriz conhecida de incerteza referente à F
E_{G_i}	Matriz conhecida de incerteza referente à G
H	Matriz auxiliar de incerteza
Δ	Matriz de contração
K_p	Ganho proporcional para PID
K_d	Ganho derivativo para PID
K_i	Ganho do integrador para PID
K_1	Ganho proporcional para LR
K_2	Ganho derivativo para LR
s	Vetor de estado aumentado discretizado
Θ	Matriz de estado aumentado discretizada
Ξ	Matriz de entrada aumentada discretizada
Ω	Matriz de observação
z	Vetor de medição
d_a	Distúrbio discretizado

d_b	Derivada do distúrbio discretizado
u_{com}	Lei de controle composta
u_{dist}	Lei de controle de compensação de distúrbio
Λ	Ganho de malha fechada de compensação
K_{com}	Ganho de compensação
G_d	Matriz que mapeia distúrbios externos discretizada
$\widehat{d}$	Estimativa do distúrbio
u_{dist}^{pd}	Sinal de compensação proporcional-derivativo
α	Ganho proporcional do sinal de compensação
λ	derivativo do sinal de compensação
W_y	Velocidade média de vento no eixo Y
ω	Velocidade angular
φ	Raio da trajetória

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, o interesse em pesquisas relacionadas a veículos aéreos não tripulados (VANTs) autônomos aumentou consideravelmente, sobretudo referentes a quadricópteros. De acordo com Valavanis e Vachtsevanos (2014), o quadricóptero pode ser definido como um VANT de asas rotativas, dotado de uma estrutura rígida com quatro motores fixados em suas extremidades. A crescente utilização deste tipo de aeronave se dá, principalmente, pelas suas características e particularidades. Sua construção, com hélices rotativas, possibilita a realização de voo pairado e a facilidade para decolar e pousar em locais com espaço limitado (MAHONY; KUMAR; CORKE, 2012).

As características físicas combinadas com técnicas de controle avançadas possibilitaram a utilização do quadricóptero nos mais variados processos, como mapeamento aéreo e fotografia (LIU *et al.*, 2021), operações em ambientes urbanos (ADKINS, 2019), agricultura de precisão (DUGGAL *et al.*, 2016), missões de vigilância e reconhecimento (YANG *et al.*, 2009) e até aplicações militares (XIAONING, 2020). Entretanto, um dos principais problemas está em garantir que o quadricóptero possa operar e realizar a atividade que lhe foi designada com segurança e confiabilidade diante das adversidades do ambiente.

Diante disso, diversas técnicas de controle foram propostas ao longo dos anos, sendo as principais classificadas em lineares, não lineares e inteligentes (ABDELMAKSOUD; MAILAH; ABDALLAH, 2020). Em se tratando de controladores lineares, destacam-se: o controlador Proporcional-Integral-Derivativo (PID) (LI; LI, 2011; BOUABDALLAH; NOTH; SIEGWART, 2004), o Regulador Linear Quadrático (LQR - do inglês, *Linear Quadratic Regulator*) (COHEN; ABDULRAHIM; FORBES, 2020; MARTINS; CARDEIRA; OLIVEIRA, 2019) e o controlador $H\infty$ (CHEN; HUZMEZAN, 2003). Já em relação às técnicas de controle não lineares, destacam-se: Linearização por Realimentação (LR) (VOOS, 2009; BONNA; CAMINO, 2015), controle por modos deslizantes (HERRERA *et al.*, 2015; LEE; KIM; SASTRY, 2009), *Backstepping* (ZEMALACHE; BEJI; MARREF, 2005) e até mesmo algumas versões do controle $H\infty$, como apresentado em Raffo, Ortega e Rubio (2010) e Raffo, Ortega e Rubio (2011). Contudo, a utilização dessas técnicas de controle isoladas pode apresentar limitações quando a aeronave está sujeita a incertezas paramétricas e distúrbios externos.

Diferentemente da maioria das técnicas de controle convencionais (lineares e não lineares), o controle inteligente possui a capacidade de se adaptar ao ambiente de operação e à falta de informações da planta. Nesse sentido, é possível implementar controladores inteligentes para operar em condições em que se tenha mudança nos parâmetros do processo e distúrbios com magnitudes desconhecidas (LIU, 2017). De acordo com Kim, Gadsden e Wilkerson (2020), as principais técnicas inteligentes são: controle *fuzzy* (LEE; KIM, 2014;

COZA; MACNAB, 2006), redes neurais artificiais (RNAs) (SARABAKHA; KAYACAN, 2019; VARSHNEY; NAGAR; SAHA, 2019) e até mesmo algumas variações do controle preditivo baseado em modelo (*MPC - Model Predictive Control*) (ASWANI *et al.*, 2011; ASWANI; BOUFFARD; TOMLIN, 2012).

A linha de pesquisa que contempla a aplicação de RNAs em quadricópteros tem recebido uma crescente atenção por pesquisadores do campo da robótica aérea. Isso porque, segundo Abdelmaksoud, Mailah e Abdallah (2020), combinar RNAs com controladores, lineares ou não, aumenta a robustez e a eficiência do sistema de controle proposto. No trabalho de Khosravian e Maghsoudi (2019), por exemplo, uma RNA foi combinada com um controlador PID não linear para melhorar o seguimento de trajetória de um quadricóptero. Neste caso, a RNA é responsável por calcular os parâmetros de controle e enviar para o PID durante a trajetória. Já no trabalho de Dierks e Jagannathan (2010), uma RNA foi utilizada para aprender a dinâmica total de um quadricóptero, incluindo incertezas não lineares do modelo. A partir do método proposto, foi possível gerar uma lei de controle inteligente capaz de controlar os seis graus de liberdade do quadricóptero.

Dentro do contexto de aplicação de RNAs em quadricóptero, é possível encontrar na literatura trabalhos que utilizam redes neurais profundas (*DNNs - Deep Neural Networks*). Isto pode ser explicado pela capacidade que esse tipo de rede possui de aproximar funções não lineares complexas com desempenho superior. No trabalho de Sarabakha e Kayacan (2019), uma DNN foi utilizada para melhorar o rastreamento de trajetória de um quadricóptero. A rede desenvolvida foi treinada, primeiramente, de forma *offline*, visando reproduzir o funcionamento do controlador PID e, posteriormente, de forma *online*, utilizando lógica *fuzzy*. No trabalho de Li *et al.* (2017), DNNs foram combinadas com o controlador PID para reduzir o erro de rastreamento de trajetória. Neste caso seis DNNs são responsáveis por enviar sinais de referências para cada um dos graus de liberdade do quadricóptero. Já no trabalho de Varshney, Nagar e Saha (2019), uma DNN foi projetada para ser capaz de reproduzir o funcionamento de um controlador MPC. Os resultados simulados demonstraram um desempenho satisfatório da rede implementada com um custo computacional menor que o do controlador.

Durante voos em ambientes externos, além das incertezas relacionadas ao modelo dinâmico do quadricóptero, este é afetado também por distúrbios de vento. Este tipo de distúrbio pode causar a instabilidade da aeronave e colocar a operação em risco. Para tratar este problema, inicialmente, diversas técnicas para estimativa e atenuação de distúrbios foram desenvolvidas, sendo as mais comuns: *Active Disturbance Rejection Control - ADRC*, *Disturbance Observer Based Control - DOBC*, *Disturbance Accommodation Control - DAC* e *Composite Hierarchical Antidisturbance Control - CHADC*.

Recentemente, arquiteturas de controle que utilizam RNAs estão sendo propostas para tratar os efeitos causados por distúrbios de vento em quadricópteros. Essas arquiteturas

combinam RNAs com controladores convencionais e, em alguns casos, com técnicas de estimativa e atenuação de distúrbios. Diante disso, Sierra e Santos (2019) propuseram uma estratégia de controle inteligente através da combinação de uma RNA com um controlador PID. A arquitetura proposta atua para atenuar distúrbio de vento e variações na massa da aeronave. Em Bellahcene *et al.* (2021), uma rede neural de base radial é combinada com um controlador $\mathcal{H}_\infty$ tanto para atenuar os erros de modelagem quanto para reduzir os efeitos do distúrbio de vento que afeta o quadricóptero.

Com relação à combinação de RNAs com observadores de distúrbios, no trabalho de Pi, Ye e Cheng (2021), por exemplo, é proposta uma arquitetura robusta. Neste caso, uma rede neural baseada em aprendizagem por reforço atua em conjunto com um observador de distúrbio, responsável por estimar a magnitude de rajadas de vento que afetam o corpo do quadricóptero. Experimentos práticos consistiram em manter o quadricóptero pairando enquanto é afetado por distúrbios de vento. Os resultados mostraram uma redução de até 75% do erro de posição. No trabalho de Lazim *et al.* (2018), um observador de distúrbio inteligente foi desenvolvido para atuar em conjunto com um controlador LR. Os autores utilizaram uma rede neural de base radial para melhorar a estimativa de distúrbio que, em seguida, foi atenuada por um compensador, formando uma arquitetura DOBC. Os resultados de simulações mostraram uma melhoria tanto para o caso em que o quadricóptero é mantido pairando no ar quanto para seguimento de trajetória.

Considerando o cenário exposto, este trabalho propõe o desenvolvimento de arquiteturas de controle inteligentes para rastreamento de trajetória de um quadricóptero afetado por distúrbios de vento. As arquiteturas propostas são responsáveis pelo controle de posição da aeronave e combinam um Regulador Linear Quadrático Robusto (RLQR) com redes neurais profundas. Além disso, outros três controladores são utilizados para comparar com o desempenho do controlador RLQR, são eles: LQR, PID e LR. O RLQR utilizado, proposto por Terra, Cerri e Ishihara (2014), baseia-se em parâmetros de penalidade e mínimos quadrados regularizados, sendo capaz de fornecer um algoritmo que não depende de parâmetros de ajuste em aplicações *online*. Este controlador necessita apenas do ajuste das matrizes de ponderação e incertezas previamente definidas, que o torna excelente para aplicações em tempo real.

1.1 Objetivos

Este trabalho tem como objetivo melhorar o desempenho de quadricópteros na tarefa de seguimento de trajetória quando o mesmo é afetado por distúrbios de vento. Para isto, duas arquiteturas de controle inteligentes serão propostas. Em geral, ambas integram o controlador RLQR e redes neurais profundas. Para efeito de comparação, os controladores LQR, PID e LR são também avaliados dentro do *framework* desenvolvido.

A primeira arquitetura de controle proposta utiliza uma DNN para melhorar o

desempenho do quadricóptero. A partir da trajetória desejada fornecida, esta rede tem o objetivo de produzir um sinal de referência factível para o controlador com base em experiências passadas. Neste caso, ao invés de minimizar o erro entre a trajetória desejada e o estado atual do quadricóptero, o controlador minimiza o erro entre a trajetória de referência fornecida pela DNN e o seu estado atual.

A segunda arquitetura de controle proposta utiliza uma DNN em conjunto com os controladores apresentados dentro da arquitetura DOBC. Neste caso, a DNN tem o objetivo de estimar o distúrbio de vento que afeta a aeronave com base no erro de posição. Em seguida, o sinal é enviado a um compensador, que gera uma lei de controle composta capaz de atenuar o distúrbio de vento.

1.2 Organização do Texto

Os próximos capítulos desta dissertação estão organizados da seguinte forma:

- **Capítulo 2:** neste capítulo será fornecida uma base teórica a respeito da modelagem de um quadricóptero comercial. Em seguida, será apresentada a formulação para o problema de rastreamento de trajetória. Por fim, será apresentada a técnica utilizada para identificação dos parâmetros desconhecidos do modelo dinâmico da aeronave.

- **Capítulo 3:** serão apresentados os conceitos e os princípios que regem o funcionamento de redes neurais artificiais e redes neurais profundas. De forma sucinta, este capítulo aborda a arquitetura da rede, funções de ativação, algoritmos de treinamento e critérios de desempenho.

- **Capítulo 4:** serão apresentadas as principais técnicas para estimativa e atenuação de distúrbios externos em sistemas de controle. Posteriormente, serão apresentados os controladores utilizados para o problema de rastreamento de trajetória.

- **Capítulo 5:** neste capítulo, serão apresentadas as arquiteturas de controle propostas formadas pela combinação de redes neurais com os controladores apresentados no Capítulo 4.

- **Capítulo 6:** será apresentada a metodologia utilizada para o desenvolvimento do trabalho. Neste capítulo, será apresentado o aparato experimental utilizado, métodos para estimativa de posição e orientação do quadricóptero, formato da trajetória desejada, treinamento das redes neurais, parâmetros dos controladores e índices de desempenho utilizados para avaliação das arquiteturas propostas.

- **Capítulo 7:** serão apresentados os resultados das arquiteturas propostas aplicadas à tarefa de seguimento de trajetória. As arquiteturas serão avaliadas, primeiramente,

em um ambiente simulado e, em seguida, em experimentos práticos. Os resultados englobam voos em condições normais e voos sob influência de distúrbios de vento.

- **Capítulo 8:** este capítulo conclui o trabalho proposto e fornece uma diretriz para trabalhos futuros.

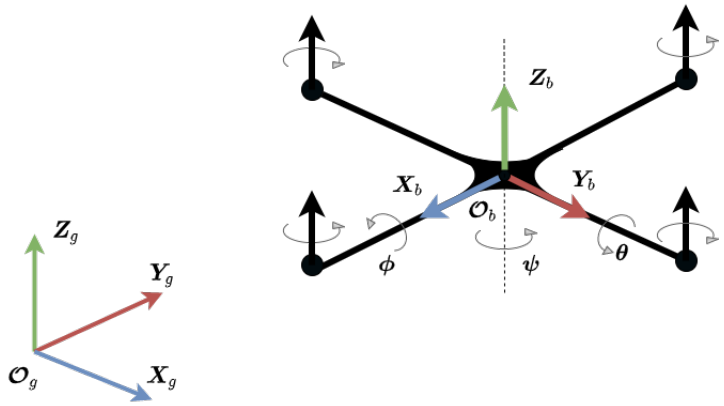
2 QUADRICÓPTERO

Este capítulo está dividido em quatro seções. A seção 2.1 apresenta, de forma sumária, as principais particularidades e técnicas utilizadas para obtenção de modelos dinâmicos de quadricópteros. Em seguida, a seção 2.2 apresenta as principais características do Parrot^{TM} Bebop 2.0 e seu respectivo modelo matemático. Já a seção 2.3 apresenta a formulação do problema de rastreamento de trajetória. Por fim, a seção 2.4 apresenta a técnica utilizada para identificação dos parâmetros desconhecidos do modelo da aeronave.

2.1 Características e Técnicas de Modelagem de um Quadricóptero

Segundo Valavanis e Vachtsevanos (2014), o quadricóptero pode ser definido como uma aeronave dotada de uma estrutura transversal rígida com quatro motores instalados em suas extremidades. Normalmente, possui configuração de voo em cruz, em que dois de seus motores rotacionam em sentido horário e os outros dois rotacionam em sentido anti-horário. A Figura 1 apresenta um diagrama de um quadricóptero genérico com configuração de voo em cruz e seu respectivo sistema de coordenadas.

Figura 1 – Sistemas de coordenadas globais e locais para um quadricóptero.



Fonte: Elaborado pelo autor.

O quadricóptero possui seis graus de liberdade, que podem ser dados pelos três ângulos de Euler $(\phi_b, \theta_b, \psi_b)$ e pelos três eixos de referência (x_b, y_b, z_b) (MAHONY; KUMAR; CORKE, 2012). Além disso, pode-se considerar quatro entradas de controle, caracterizando o sistema como subatuado.

Ao longo dos anos, diversos modelos matemáticos foram propostos para representar a dinâmica de quadricópteros. Tanto a formulação de Newton-Euler utilizada nos trabalhos de Mahony, Kumar e Corke (2012), Madani e Benallegue (2007), Heng *et al.* (2016) quanto a formulação de Euler-Lagrange utilizada em Leão (2015), Alaiwi e Mutlu (2018), Balasubramanian e Vasantharaj (2013) são abundantes na literatura. Além disso, é possível

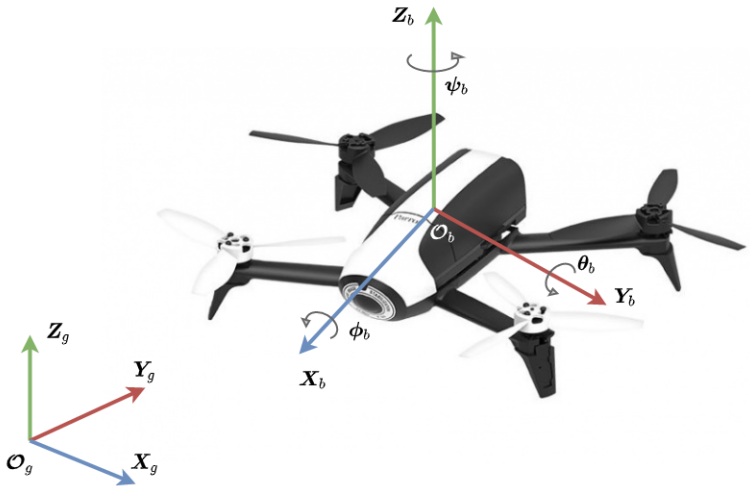
encontrar modelos que utilizam as duas formulações, como em Önder Efe (2015). Com o proposito de obter um modelo matemático simplificado que represente quadricópteros com fidelidade, em Santana *et al.* (2014) foi desenvolvido um modelo dinâmico para um Parrot^{TM} Ar.Drone. Este modelo foi adaptado para um Parrot^{TM} Bebop 2.0 no trabalho de Benevides *et al.* (2019a) e demonstrou boa aproximação da dinâmica do quadricóptero.

2.2 Modelo Matemático para o Parrot^{TM} Bebop 2.0

O quadricóptero Bebop 2.0 possui diversas características que o torna interessante para realização de pesquisas voltadas ao seguimento de trajetória de forma autônoma. Dentre elas, podem-se destacar a presença de uma Unidade de Medição Inercial (*IMU - Inertial Measurement Unit*) dotada de magnetrômetro, acelerômetro e giroscópio de três eixos; um sensor de ultrassom para voos com altitude menor que 8 metros e um sensor barométrico. Além disso, possui uma câmera de alta resolução e uma unidade capaz de realizar processamento de imagem para detecção de padrões de movimento utilizando algoritmos de fluxo óptico. Por fim, vale destacar as principais especificações técnicas da aeronave: velocidade horizontal máxima de 16 m/s, velocidade vertical máxima de 6 m/s, processador dual core com GPU quad-core, memória flash de 8 GB e Wi-Fi aéreo com antena de 2.4 GHz e 5 GHz de dipolo com alcance de 300 m.

Além das características do Parrot^{TM} Bebop 2.0, este modelo de quadricóptero se torna mais interessante, sobretudo, pela facilidade de comunicação com o *Robot Operating System* (ROS) e por ter um simulador próprio capaz de representar o ambiente real com boa fidelidade, permitindo a realização de testes preliminares. A Figura 2 apresenta o quadricóptero que será utilizado neste trabalho com os sistemas de coordenadas globais e locais adotados.

Figura 2 – Parrot^{TM} Bebop 2.0 e sistemas de coordenadas adotados.



Fonte: Elaborado pelo autor.

Para controlar o Bebop 2.0 no ar é necessário enviar comandos de velocidades, os quais são enviados por *Wi-Fi*, normalizados dentro da faixa $[-1.0, +1.0]$, e podem ser representados pelo vetor $\nu = [\nu_{\dot{x}_b}\ \nu_{\dot{y}_b}\ \nu_{\dot{z}_b}\ \nu_{\dot{\psi}}]$, em que

- $\nu_{\dot{x}_b}$ representa o comando de rotação em torno do eixo y_b, o que resulta em movimento na direção de x_b;
- $\nu_{\dot{y}_b}$ representa o comando de rotação em torno do eixo x_b, o que resulta em movimento na direção de y_b;
- $\nu_{\dot{z}_b}$ representa o comando de velocidade linear em torno do eixo z_b, o que resulta em movimento linear na direção de z_b;
- $\nu_{\dot{\psi}}$ representa o comando de velocidade angular que causa rotação em torno do eixo z_b, o que resulta em movimento angular em torno de z_b.

Considerando o modelo desenvolvido em Santana *et al.* (2014), temos que a dinâmica simplificada que descreve o comportamento do Bebop 2.0 pode ser dada por

$$\begin{aligned} \ddot{x}_b &= -\gamma_2 \dot{x}_b + \gamma_1 \nu_{\dot{x}_b} \\ \ddot{y}_b &= -\gamma_4 \dot{y}_b + \gamma_3 \nu_{\dot{y}_b} \\ \ddot{z}_b &= -\gamma_6 \dot{z}_b + \gamma_5 \nu_{\dot{z}_b} \\ \ddot{\psi}_b &= -\gamma_8 \dot{\psi}_b + \gamma_7 \nu_{\dot{\psi}_b} \end{aligned}, \tag{2.1}$$

no qual os parâmetros γ_1 a γ_8 são coeficientes a serem identificados através de experimentos em laboratório, aliados às técnicas de identificação e estimativa de parâmetros disponíveis na literatura. As variáveis $\dot{x}_b$, $\dot{y}_b$, $\dot{z}_b$, representam velocidades lineares com relação às posições x_b, y_b e z_b, respectivamente, e a velocidade angular em torno do eixo z_b é representada por $\dot{\psi}_b$.

Este modelo dinâmico pode ser representado na forma matricial, como definido pela Equação 2.2

$$\ddot{q}_b = \Gamma_A(\gamma)\dot{q}_b(t) + \Gamma_B(\gamma)\nu(t), \tag{2.2}$$

em que $q_b = [x_b\ y_b\ z_b\ \psi_b]^T$ é o vetor de coleção de estados com relação às coordenadas locais da aeronave e as matrizes do modelo são dadas por

$$\Gamma_A = -\begin{bmatrix} \gamma_2 & 0 & 0 & 0 \\ 0 & \gamma_4 & 0 & 0 \\ 0 & 0 & \gamma_6 & 0 \\ 0 & 0 & 0 & \gamma_8 \end{bmatrix}, \qquad \Gamma_B = \begin{bmatrix} \gamma_1 & 0 & 0 & 0 \\ 0 & \gamma_3 & 0 & 0 \\ 0 & 0 & \gamma_5 & 0 \\ 0 & 0 & 0 & \gamma_7 \end{bmatrix}.$$

Considerando que, em baixas velocidades, a estabilização de atitude é garantida pelo sistema de controle interno do Bebop 2.0, temos que os ângulos de rolagem (ϕ) e

arfagem (θ) são praticamente nulos. Desse modo, podemos considerar apenas a rotação em torno do eixo z, dado pelo ângulo de guinada (ψ). Por conseguinte, levando em consideração o sistema de coordenadas globais apresentado na Figura 2, o modelo descrito na Equação (2.2) pode ser reescrito como

$$\begin{aligned} \ddot{x_g} &= \gamma_1 \nu_{\dot{x}_b} \cos\psi_b - \gamma_2 \dot{x_b} \cos\psi_b - \gamma_3 \nu_{\dot{y}_b} \sin\psi_b + \gamma_4 \dot{y_b} \sin\psi_b \\ \ddot{y_g} &= \gamma_1 \nu_{\dot{x}_b} \sin\psi_b - \gamma_2 \dot{x_b} \sin\psi_b + \gamma_3 \nu_{\dot{y}_b} \cos\psi_b - \gamma_4 \dot{y_b} \cos\psi_b \\ \ddot{z_g} &= \gamma_5 \nu_{\dot{z}_b} - \gamma_6 \dot{z_b} \\ \ddot{\psi_g} &= \gamma_7 \nu_{\dot{\psi}} - \gamma_8 \dot{\psi_b} \end{aligned} \tag{2.3}$$

tendo como forma matricial a Equação (2.4), definida por

$$\ddot{q_g} = -A(\gamma, t)\dot{q_b}(t) + B(\gamma, t)\nu(t), \tag{2.4}$$

em que $q_g = [x_g \ y_g \ z_g \ \psi_g]^T$ é o vetor de coleção dos estados com relação ao quadro de coordenadas globais e $q_b = [x_b \ y_b \ z_b \ \psi_b]^T$ representa o vetor de coleção dos estados com relação às coordenadas locais da aeronave. As matrizes A e B são definidas por

$$A = \begin{bmatrix} \gamma_2\, cos(\psi_b) & -\gamma_4\, sin(\psi_b) & 0 & 0 \\ \gamma_2\, sin(\psi_b) & \gamma_4\, cos(\psi_b) & 0 & 0 \\ 0 & 0 & \gamma_6 & 0 \\ 0 & 0 & 0 & \gamma_8 \end{bmatrix}, \ B = \begin{bmatrix} \gamma_1\, cos(\psi_b) & -\gamma_3\, sin(\psi_b) & 0 & 0 \\ \gamma_1\, sin(\psi_b) & \gamma_3\, cos(\psi_b) & 0 & 0 \\ 0 & 0 & \gamma_5 & 0 \\ 0 & 0 & 0 & \gamma_7 \end{bmatrix}.$$

As transformações em relação às coordenadas locais e globais são dadas através de uma matriz de rotação R_t, considerando as seguintes relações:

$$\begin{aligned} \dot{q_g} &= R_t \dot{q_b} \\ \ddot{q_g} &= R_t \ddot{q_b} \end{aligned}, \tag{2.5}$$

em que R_t é uma matriz de transformação homogênea que determina a rotação em torno do eixo z, definida como

$$R_t = \begin{bmatrix} cos(\psi_b) & sin(\psi_b) & 0 & 0 \\ -sin(\psi_b) & cos(\psi_b) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Desse modo, o modelo simplificado do quadricóptero em termos de coordenadas globais, passa a ser

$$\ddot{q_g} = -A(\gamma, t)R_t^T \dot{q_g}(t) + B(\gamma, t)\nu(t). \tag{2.6}$$

Se considerarmos, ainda, distúrbios externos, o modelo do sistema apresentado na Equação (2.6) se torna

$$\ddot{q}_g = -A(\gamma, t)R_t^T \dot{q}_g(t) + B(\gamma, t)\nu(t) + B_d(t)d(t), \tag{2.7}$$

em que d é o vetor de distúrbios externos e B_d representa a matriz que mapeia os distúrbios externos, definida como

$$B_d = \begin{bmatrix} \frac{1}{m} & 0 & 0 & 0 \\ 0 & \frac{1}{m} & 0 & 0 \\ 0 & 0 & \frac{1}{m} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix},$$

no qual m e I_z correspondem a massa e ao momento de inércia da aeronave com relação ao eixo Z, respectivamente.

2.3 Formulação do Problema de Rastreamento de Trajetória

A partir do modelo apresentado em (2.7) é possível deduzir o espaço de estado do erro para implementação posterior dos controladores. Considerando $\ddot{q}^d$ e $\dot{q}^d$ como acelerações e velocidades de referências, respectivamente, o erro de acompanhamento de referência e suas derivadas são definidos por

$$\begin{aligned} \tilde{q}(t) &= q_g(t) - q^d(t), \\ \dot{\tilde{q}}(t) &= \dot{q}_g(t) - \dot{q}^d(t), \\ \ddot{\tilde{q}}(t) &= \ddot{q}_g(t) - \ddot{q}^d(t). \end{aligned} \tag{2.8}$$

Substituindo as equações do erro (2.8) na equação referente ao modelo do quadricóptero que considera distúrbios externos, apresentado em (2.7), temos

$$\ddot{\tilde{q}}(t) = -AR_t^T \dot{\tilde{q}}(t) + B\nu - AR_t^T \dot{q}^d(t) - \ddot{q}^d(t) + B_d d. \tag{2.9}$$

Definindo o vetor de entrada de controle virtual como

$$u(t) = B\nu + AR_t^T \dot{q}^d(t) - \ddot{q}^d(t), \tag{2.10}$$

a equação dinâmica do espaço de estado do erro se torna

$$\ddot{\tilde{q}}(t) = AR_t^T \dot{q}^d(t) + I_n u + B_d d, \tag{2.11}$$

e o vetor de controle real resulta em

$$\nu = B^{-1}(u + \ddot{\tilde{q}}^d(t) + AR_t^T \dot{q}^d(t)). \tag{2.12}$$

Para representar o modelo em espaço de estado baseado no erro, define-se um vetor de estado aumentado $x(t)$, em que

$$\tilde{x}(t) = \begin{bmatrix} \dot{\tilde{q}} \\ \tilde{q} \end{bmatrix} = \begin{bmatrix} \dot{q}_g - \dot{q}^d \\ q_g - q^d \end{bmatrix}.$$

A partir de uma entrada de controle virtual, a representação do sistema no espaço de estado do erro é dada por

$$\dot{\tilde{x}}(t) = \begin{bmatrix} -AR_t^T & 0 \\ I & 0 \end{bmatrix} \tilde{x} + \begin{bmatrix} I \\ 0 \end{bmatrix} u + \begin{bmatrix} B_d \\ 0 \end{bmatrix} d. \tag{2.13}$$

A partir do modelo apresentado em (2.13) é possível implementar controladores com o intuito de obter uma lei de controle que minimize o vetor de estado $\dot{\tilde{x}}$, fazendo com que o quadricóptero siga a trajetória previamente definida.

2.4 Identificação dos Parâmetros do Modelo Dinâmico

Identificação de parâmetros desconhecidos em um modelo de sistema dinâmico é uma das áreas de estudo mais importante da teoria de controle. Em geral, identificar parâmetros em um sistema dinâmico está relacionado com a resolução de um problema de otimização, no qual a função objetivo relaciona as variáveis de saída e entrada com a parte funcional do modelo que contém os parâmetros desconhecidos. Nas últimas décadas, diversas técnicas surgiram com o intuito de prover soluções ótimas para este problema, sendo o método de mínimos quadrados (MMQ) um dos mais utilizados atualmente. Este método é capaz de encontrar uma solução ótima para os parâmetros desconhecidos do sistema dinâmico através do seguinte sistema

$$y = X\theta, \tag{2.14}$$

no qual θ é o conjunto de parâmetros desconhecidos que se deseja estimar, $X \in \mathbb{R}^{n \times n}$ é a matriz de variáveis independentes conhecidas, e y é o vetor de variáveis dependentes conhecidas. A estimativa dos parâmetros que compõe o vetor θ é dada através da minimização do erro entre o modelo e os dados experimentais. Dessa forma, tem-se

$$\epsilon = y - X\hat{\theta}, \tag{2.15}$$

em que ϵ é o vetor de erro relacionado ao processo de estimativa. Através da minimização do somatório do erro da Equação (2.16), encontra-se o valor ótimo para o vetor de parâmetros desconhecidos θ.

$$J = \sum_{i=1}^{n} \epsilon^2(i) = \epsilon^T \epsilon. \tag{2.16}$$

Substituindo a Equação (2.15) na Equação (2.16), obtemos

$$\begin{aligned} J &= (y - X\hat{\theta})(y - X\hat{\theta}) \\ &= y^T y - yX\hat{\theta} - \hat{\theta}X^T y + \hat{\theta}X^T X\hat{\theta}. \end{aligned} \tag{2.17}$$

Com o intuito de minimizar a função de custo, derivamos J com relação a θ e igualamos a zero

$$\frac{\partial J}{\partial \hat{\theta}} = -2X^T y + 2X^T X\hat{\theta}, \tag{2.18}$$

$$= 0 \tag{2.19}$$

resultando em

$$X^T X\hat{\theta} = X^T y. \tag{2.20}$$

Isolando o vetor de parâmetros desconhecidos que se deseja estimar, temos

$$\hat{\theta} = (X^T X)^{-1} X^T y, \tag{2.21}$$

em que $(X^T X)^{-1} X^T$ é a pseudo inversa de X.

Para a estimativa dos parâmetros do modelo dinâmico do Bebop 2.0, buscamos resolver a Equação (2.21) a partir da aplicação das técnicas desenvolvidas em Benevides *et al.* (2019b). A ideia principal é computar a relação entrada-saída a partir do envio de comandos de velocidades ν para obter os respectivos comandos de posição e atitude. Dessa forma, é possível obter os parâmetros do modelo que relacionam a entrada e saída do sistema.

Primeiramente, para aplicação da técnica descrita, faz-se necessária a criação de uma matriz de regressão. Para isto, considere a Equação (2.1) que descreve a dinâmica do Bebop 2.0. Para simplificação, podemos rescrever em termos das seguintes variáveis e computar os valores para cada iteração i.

$$\begin{array}{llll}
\vartheta_1^i = cos\,\psi^i\,\nu_{\dot{x}_b}^i & \sigma_1^i = sen\,\psi^i\,\nu_{\dot{x}_b}^i & & \\
\vartheta_2^i = -cos\,\psi^i\,\dot{x}_b^i & \sigma_2^i = -sen\,\psi^i\,\dot{x}_b^i & \tau_1^i = \nu_{\dot{z}}^i & \iota_1^i = \nu_{\dot{\psi}}^i \\
\vartheta_3^i = -sen\,\psi^i\,\nu_{\dot{y}_b}^i & \sigma_3^i = cos\,\psi^i\,\nu_{\dot{y}_b}^i & \tau_2^i = -\dot{z}^i & \iota_2^i = -\dot{\psi}^i \\
\vartheta_4^i = sen\,\psi^i\,\dot{y^i}_b & \sigma_4^i = -cos\,\psi^i\,\dot{y^i}_b & &
\end{array}$$

Com isso, podemos formar as matrizes de regressão da seguinte forma:

$$\underbrace{\begin{bmatrix} \ddot{x}_g^1 \\ \vdots \\ \ddot{x}_g^n \\ \ddot{y}_g^1 \\ \vdots \\ \ddot{y}_g^n \end{bmatrix}}_{y} = \underbrace{\begin{bmatrix} 1 & \vartheta_1^i & \vartheta_2^i & \vartheta_3^i & \vartheta_4^i \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \vartheta_1^n & \vartheta_2^n & \vartheta_3^n & \vartheta_4^n \\ 1 & \sigma_1^i & \sigma_2^i & \sigma_3^i & \sigma_4^i \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \sigma_1^n & \sigma_2^n & \sigma_3^n & \sigma_4^n \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} 0 \\ \gamma_1 \\ \gamma_2 \\ \gamma_3 \\ \gamma_4 \end{bmatrix}}_{\gamma} + \underbrace{\begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix}}_{\epsilon}, \tag{2.22}$$

$$\underbrace{\begin{bmatrix} \ddot{z}_g^1 \\ \vdots \\ \ddot{z}_g^n \end{bmatrix}}_{y} = \underbrace{\begin{bmatrix} 1 & \tau_1^i & \tau_2^i \\ \vdots & \vdots & \vdots \\ 1 & \tau_1^n & \tau_2^n \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} 0 \\ \gamma_5 \\ \gamma_6 \end{bmatrix}}_{\gamma} + \underbrace{\begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix}}_{\epsilon}, \tag{2.23}$$

$$\underbrace{\begin{bmatrix} \ddot{\psi}_g^1 \\ \vdots \\ \ddot{\psi}_g^n \end{bmatrix}}_{y} = \underbrace{\begin{bmatrix} 1 & \iota_1^i & \iota_2^i \\ \vdots & \vdots & \vdots \\ 1 & \iota_1^n & \iota_2^n \end{bmatrix}}_{X} \underbrace{\begin{bmatrix} 0 \\ \gamma_7 \\ \gamma_8 \end{bmatrix}}_{\gamma} + \underbrace{\begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix}}_{\epsilon}. \tag{2.24}$$

Diante disso, para as Equações (2.22), (2.23) e (2.24), podemos computar $\hat{\gamma} = (X^T X)^{-1} X^T y$, em que $\hat{\gamma}$ é o vetor de parâmetros desconhecidos do modelo, X é a matriz de regressão e y é a saída do sistema. Os resultados dos cálculos forneceram o vetor de parâmetros $\gamma = [\gamma_1, ..., \gamma_8]$ para o modelo do Bebop 2.0.

Foram realizados voos com o quadricóptero através do envio de entradas de controle em x e y. Neste caso foi utilizado a Equação (2.22) para obter os parâmetros $\gamma_1, \gamma_2, \gamma_3$ e γ_4. Em seguida, foram enviados comandos de entrada relacionados com o eixo z da aeronave, em que a Equação (2.23) foi utilizada para obtenção dos parâmetros γ_5 e γ_6. Por último, entradas de controle relacionadas com o ângulo em torno de z (ψ) foram enviadas ao quadricóptero. Neste caso, a Equação (2.24) foi utilizada para obtenção dos parâmetros γ_7 e γ_8. Os parâmetros obtidos para os experimentos com o simulador *Parrot-Sphinx* e com o quadricóptero real são apresentados na Tabela 1.

Tabela 1 – Parâmetros identificados do modelo do Bebop 2.0.

	γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	γ_7	γ_8
Parrot-Sphinx	$4,8204$	$0,1210$	$5,5099$	$0,2490$	$4,6575$	$3,3021$	$6,2467$	$-0,4075$
Bebop Real	$4,5922$	$0,4349$	$5,2157$	$0,4364$	$4,6026$	$3,0992$	$5,9388$	$-0,3980$

3 REDES NEURAIS

Este capítulo apresenta os conceitos fundamentais sobre redes neurais artificiais que foram pertinentes para a execução deste trabalho. A seção 3.1 introduz o ramo da inteligência artificial do ponto de vista do aprendizado de máquina. A seção 3.2 aborda, de forma sumária, os principais detalhes a respeito das arquiteturas e funções de ativação utilizadas em redes neurais artificiais. Por fim, a seção 3.3 apresenta as particularidades de redes neurais profundas, com foco nos algoritmos de otimização, técnicas de regularização e métricas de avaliação utilizadas neste trabalho.

3.1 Aprendizado de Máquina

O aprendizado de máquina é um campo da inteligência artificial (IA) que pode ser definido como um conjunto de métodos e técnicas capaz de extrair conhecimentos de um conjunto de dados (BISHOP, 2006). Dessa forma, é possível aplicar um algoritmo de aprendizado de máquina treinado para inferir a respeito de um determinado problema, utilizando experiências adquiridas no processo de treinamento.

Diferentes tipos de aprendizados de máquinas podem ser utilizados para a solução dos mais variados problemas. Em geral, a forma de aprendizado destes algoritmos é utilizada para classificá-los, no qual se destacam o aprendizado supervisionado, não supervisionado e aprendizado por reforço (GÉRON, 2019).

- **Aprendizado Supervisionado:** durante o treinamento são fornecidos dados de entradas com suas respectivas saídas desejadas. Dessa forma, a partir de um conjunto de entradas fornecidas e rotuladas com suas respectivas saídas, a rede realiza os passos de treinamento a fim de alcançar os resultados desejados. A Equação (3.1) apresenta o formato do conjunto de treinamento fornecido para o aprendizado supervisionado.

$$\mathcal{X} = \left\{(x_i, y_i)_{i=1}^{N}\right\}, \tag{3.1}$$

  em que x_i é o dado de entrada, y_i é a saída correspondente, N é o número total de amostras fornecidas, e $i = \{1, ..., N\} \in \mathbb{R}$.

- **Aprendizado Não Supervisionado:** diferentemente do aprendizado supervisionado, a rede não possui conhecimento sobre os resultados de saída que se deseja alcançar. Desse modo, a partir dos dados de entrada, a rede tenta encontrar similaridades entre as instâncias de treinamento. Neste tipo de aprendizado não há *feedback* baseado nos resultados fornecidos pela rede, uma vez que não há saídas rotuladas para que a rede possa se basear, restando apenas as observações dos dados fornecidos

na tentativa de encontrar padrões. A Equação (3.2) apresenta o formato do conjunto de treinamento fornecido para este tipo aprendizado.

$$\mathcal{X} = \left\{(x_i)_{i=1}^{N}\right\}, \tag{3.2}$$

em que x_i é o dado de entrada, N é o número total de amostras fornecidas, e $i = \{1, ..., N\} \in \mathbb{R}$.

- **Aprendizado por Reforço:** o aprendizado por reforço utiliza um sistema de recompensa e punição para forçar o agente a tomar a melhor decisão em busca de um objetivo. Diferentemente dos métodos citados acima, no aprendizado por reforço não há dados de entradas e saídas, o agente inicia sem experiência prévia e a partir das recompensas recebidas, busca atingir a otimalidade.

3.2 Redes Neurais Artificiais

Redes Neurais Artificiais constituem uma das técnicas mais importantes de aprendizado de máquina. Seu princípio de funcionamento se baseia na obtenção de conhecimento com base no sistema nervoso central de seres humanos, semelhante aos processos de aprendizagem do neurônio biológico (AGGARWAL, 2018). O sistema nervoso humano contém neurônios interligados por dendritos e axônios. Os dendritos são responsáveis por receber impulsos elétricos de entrada e os axônios transmitem os impulsos computados pelos neurônios através de sinapses, região que conecta os dendritos aos axônios (AGGARWAL, 2018). Atualmente, os algoritmos de RNAs são aplicados para resolver problemas em diversas áreas do conhecimento, realizando, principalmente, predições, regressão numérica, classificação, processamento de dados e tarefas de reconhecimento de padrões.

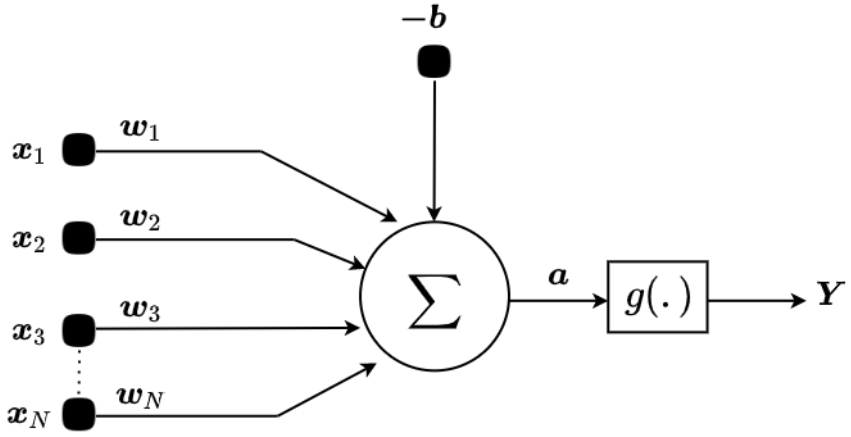
Neste trabalho, as redes neurais desenvolvidas serão aplicadas à tarefa de regressão, em que a partir de um conjunto de variáveis de entrada, a rede é capaz de predizer valores numéricos. Esse tipo de rede é utilizada nos mais variados problemas, como predição de valores de casas, predição de coordenadas e até mesmo para fornecer uma previsão a respeito do comportamento do mercado financeiro.

3.2.1 Arquitetura e Funções de Ativação

A arquitetura geral de uma RNA é composta por um neurônio artificial, que desempenha o papel semelhante a um neurônio biológico. O primeiro modelo matemático de um neurônio artificial foi desenvolvido em 1958 por Rosenblatt (1958), baseado no trabalho de McCulloch e Pitts (1943). Esse modelo ficou conhecido como perceptron e a Figura 3 apresenta a sua estrutura, em que seu funcionamento consiste em receber um conjunto de entradas $\{x_1, x_2, ..., x_N\}$, que em seguida são multiplicadas por pesos

sinápticos $\{w_1, w_2, ..., w_N\}$ e subtraídas pelo valor de limiar de ativação b_i. Por fim, a saída no neurônio passa por uma função de ativação linear, gerando a saída y do perceptron.

Figura 3 – Modelo do Perceptron.



Fonte: Elaborado pelo autor.

A Equação (3.3) apresenta as relações matemáticas que acontecem durante o processamento de um conjunto de variáveis de entrada pelo perceptron.

$$a = \sum_{i=1}^{N} x_i w_i - b_i, \tag{3.3}$$

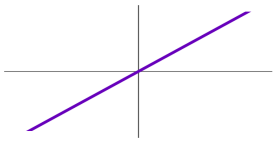
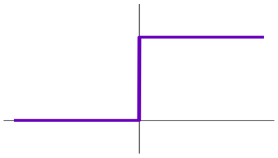
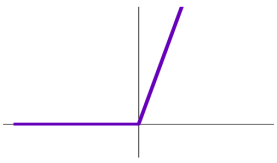
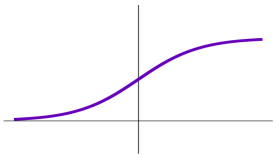
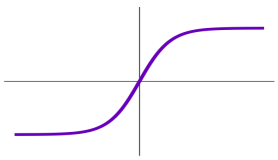
em que x_i representa o vetor de entrada, w_i é o peso sináptico, b_i corresponde ao limiar de ativação e a_i representa o resultado da ponderação das entradas, no qual $i = \{1, 2, ..., N\}$. O valor de a_i obtido é utilizado pela função de ativação g, que limita a saída do neurônio e gera uma nova saída representada pela Equação (3.4).

$$y = g(a). \tag{3.4}$$

Existem diversas funções de ativação que podem ser utilizadas para limitar a saída de um neurônio artificial, sendo que a escolha da função ideal pode variar de acordo com o problema que se quer resolver. As principais funções utilizadas atualmente são apresentadas na Tabela 2.

Existem diferentes arquiteturas de RNAs que podem ser encontradas na literatura, sendo a principal delas, o Perceptron Multicamadas (PMC). Esse tipo de rede apresenta ao menos uma camada oculta dotada de uma série de neurônios e, em alguns casos, podem ser compostas por diversas camadas ocultas (SILVA *et al.*, 2016). Em redes PMC, cada neurônio da camada de entrada é conectado a todos os neurônios da camada oculta que, por sua vez, são conectados a todos os neurônios da camada de saída. Entretanto, os neurônios situados em uma mesma camada não se conectam (SKANSI, 2018).

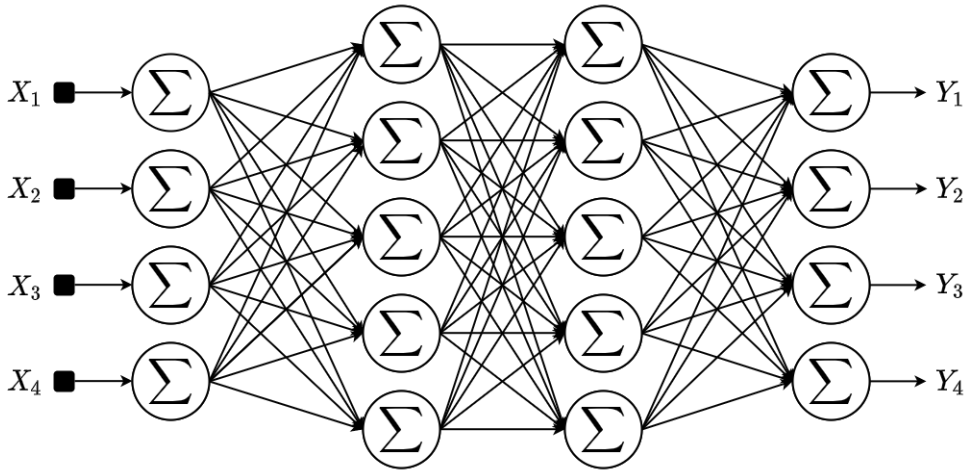
Tabela 2 – Principais Funções de ativação utilizadas para limitar a saída de neurônios em redes neurais.

Função	**Equação**	**Gráfico**
Linear	$g(a) = a_i$	
Degrau ou Limiar	$g(a) = \begin{cases} 1, & a \geq 0 \\ 0, & a < 0 \end{cases}$	
Linear Retificada ($ReLU$)	$g(a) = max(0, a)$	
Logística (*Sigmoid*)	$g(a) = \frac{1}{1+e^{-a}}$	
Tangente Hiperbólica	$g(a) = \frac{e^{a}-e^{-a}}{e^{a}+e^{-a}}$	

Segundo Aggarwal (2018), a rede do tipo PMC também é chamada de rede neural de propagação para frente (*Feedforward Neural Networks*), uma vez que o sinal fornecido na entrada da rede se propaga para frente, camada por camada, até a saída. Além disso, esse formato de rede permite utilizar o algoritmo de *Backpropagation* (RUMELHART; HINTON; WILLIAMS, 1985) a fim de corrigir o erro entre o sinal desejado e o sinal gerado pela rede. Este algoritmo de otimização, conhecido também como regra delta generalizada, é um dos mais utilizados para treinamento de RNAs do tipo PMC. A Figura 4 apresenta a arquitetura de uma RNA de múltiplas camadas, composta por uma camada de entrada, duas camadas ocultas e uma camada de saída.

Com relação às redes PMC, vale destacar, ainda, que possuem uma maior capacidade de extrair conhecimento quando comparadas ao perceptron. Maiores detalhes a respeito de

Figura 4 – Arquitetura de uma RNA de múltiplas camadas.



Fonte: Elaborado pelo autor.

redes PMC, sobretudo referentes as que possuem múltiplas camadas ocultas (profundas), serão apresentadas na seção seguinte.

3.3 Redes Neurais Profundas

Redes neurais profundas (*DNN - Deep Neural Networks*) fazem parte do campo do aprendizado de máquina dentro da área da inteligência artificial. Esse tipo de rede possui a capacidade de aprender de uma grande quantidade de dados e são compostas por múltiplas camadas de processamento (CHOLLET, 2017). Em geral, possuem duas ou mais camadas ocultas e são capazes de lidar, também, com uma grande quantidade de dados não estruturados.

3.3.1 Treinamento e Algoritmos de Otimização

O treinamento de uma DNN consiste na aplicação de técnicas específicas que visam o ajuste automático dos pesos sinápticos da rede através da minimização de uma função de custo. Após o treinamento, a rede estará apta a fornecer soluções gerais para a classe de problemas em que foi treinada.

Considerando uma DNN, cujo processo de aprendizagem será realizado de forma supervisionada, é possível aplicar o algoritmo de retropropagação (*Backpropagation*) para treinamento da rede. De forma sumária, este algoritmo consiste em duas etapas principais. A primeira, propagação para frente (*forward*), os dados de entrada são propagados do início até o final da rede, camada por camada, até gerar uma saída, dado um conjunto de entradas aleatórias fornecidas. Em seguida, na etapa de propagação para trás (*Backward*) é realizado uma comparação entre a saída desejada e a saída fornecida pela rede (AGGARWAL, 2018). A diferença é computada como um valor de erro, que é retropropagado pela rede para atualizar os pesos. Em outras palavras, a partir da função que gera o erro entre a saída

desejada e a predita pela rede, a fase de retropropagação calcula o gradiente da função de perda na camada final e utiliza a regra da cadeia para atualizar os pesos, visando garantir que as entradas convirjam para a saída desejada. A Equação (3.5) apresenta uma função de perda típica, utilizada para calcular o erro de predição de uma rede neural.

$$\mathcal{E}(i) = \frac{1}{2}\sum_{i=1}^{N}(y_p(i) - y_d(i))^2, \tag{3.5}$$

em que $y_p(i)$ é a i-ésima saída fornecida pela rede, $y_d(i)$ representa o valor desejado para a i-ésima amostra, N é o número total de neurônios em uma camada e k, corresponde a um neurônio.

Com o propósito de otimizar o processo de minimização da função de custo, é possível escolher diferentes algoritmos que refletem no momento de atualização dos pesos da rede. O mais comum, Gradiente Descendente (*GD - Gradient Descent*), atualiza os pesos da rede apenas quando todas as amostras foram computadas. O segundo, Gradiente Descendente Estocástico (*SGD - Stochastic Gradient Descent*), atualiza os pesos da rede após cada amostra ser computada, em geral, funciona melhor para conjuntos de dados pequenos (LECUN; BENGIO; HINTON, 2015). O terceiro, *Mini-Batch Gradient Descent (MBGD)* particiona o conjunto de dados em lotes, em que os pesos da rede são atualizados após cada lote ser computado. Valores comuns para os lotes do MBGD são: 32, 64, 128 e 256. Para estes casos, o ajuste dos pesos sinápticos é definido pela Equação (3.6).

$$\mathrm{W}_{t+1} = \mathrm{W}_t - \eta\nabla J(\mathrm{W}), \tag{3.6}$$

em que W_t são os pesos atuais da rede, W_{t+1} são os valores de peso calculados e $J(\mathrm{W})$ é a função de custo, no qual o gradiente é computado. Uma função de custo comum quando se utiliza o algoritmo de GD ou quando se tem uma quantidade pequena de dados, é o erro quadrático médio, definido na Equação (3.7).

$$\mathcal{E}_c = \frac{1}{N}\sum_{i=1}^{N}\mathcal{E}(i), \tag{3.7}$$

no qual $\mathcal{E}$ é dado pela Equação (3.5) e N é o número total de amostras fornecidas à rede.

Além das técnicas de descida de gradiente apresentadas, é importante mencionar a técnica de *momentum* (POLYAK, 1964), utilizada tanto com o SGD, como também com algoritmos de otimização mais eficientes, que serão apresentados na sequência. Esta técnica é utilizada para acelerar o aprendizado, fazendo com que o gradiente continue decaindo na mesma direção que a média de decaimento do passos anteriores. Para isto é adicionado um novo parâmetro β_1, tal que $0 \leq \beta_1 < 1$ é responsável por prevenir que a velocidade com que o gradiente decai aumente demasiadamente (GOODFELLOW; BENGIO; COURVILLE, 2016). Com isso, diferentemente do SGD que faz com que o gradiente decaia em pequenos

passos, a técnica de *momentum* faz com que o gradiente dê grandes passos na direção correta e reduza o tamanho dos passos na direção contrária, aumentando a velocidade de treinamento da rede. A Equação (3.8) apresenta esta técnica.

$$\upsilon_{t+1} = \beta_1 \upsilon_t - (1 - \beta_1) \nabla J(\mathrm{W}). \tag{3.8}$$

Se levarmos em consideração a Equação (3.6), utilizada para atualização dos pesos da rede com o algoritmo SGD, temos que atualização dos pesos considerando a técnica de *momentum* apresentada é dada por

$$\mathrm{W}_{t+1} = \mathrm{W}_t + \beta_1 \upsilon_t + (1 - \beta_1) \left[-\eta \nabla J(\mathrm{W})\right]. \tag{3.9}$$

Segundo Goodfellow, Bengio e Courville (2016), a maioria dos algoritmos de otimização utilizados para treinamento de redes profundas estão entre o SGD e o GD, ou seja, utilizam mais de um conjunto de treinamento, porém, menos do que todos os exemplos de treino. Esses algoritmos são chamados de *mini-batch stochastic methods* ou apenas *stochastic methods*, tendo como principal exemplo o MBGD apresentado. Com o intuito de otimizar o ajuste dos pesos da DNN, principalmente quando se tem uma quantidade grande de dados e parâmetros, diversos algoritmos de otimização surgiram baseados nos *stochastic methods*, sendo os principais: AdaGrad *(Adaptive Gradient)*(DUCHI; HAZAN; SINGER, 2011), AdaDelta (Adaptive learning rate) (ZEILER, 2012), RMSProp[1], Adam *(Adaptive moment estimation)* (KINGMA; BA, 2014) e Nadam (DOZAT, 2016) *(Nesterov-accelerated Adaptive Moment Estimation)*. A seguir, são apresentados os detalhes referentes aos algoritmos de otimização que foram utilizados para treinamento das redes desenvolvidas neste trabalho.

Diferentemente de muitos outros algoritmos de otimização, o *RMSProp* funciona bem para funções não convexas. Isso porque acumula apenas o gradiente das iterações mais recentes, utilizando a média móvel de decaimento exponencial (GOODFELLOW; BENGIO; COURVILLE, 2016). Dessa forma, converge mais rapidamente para o ponto de mínimo global. O funcionamento acontece a partir da seguinte equação

$$s_{t+1}^{\mathrm{dw}} = \beta_2 s_t^{\mathrm{dw}} + (1 - \beta_2) \nabla J(\mathrm{W}) \otimes \nabla J(\mathrm{W}), \tag{3.10}$$

em que β_2 é a taxa de decaimento, normalmente configurada com valor de $0,9$. Para este caso, a atualização dos parâmetros da rede é dada por

$$\mathrm{W}_{t+1} = \mathrm{W}_t - \eta \frac{\nabla J(\mathrm{W})}{\sqrt{s_{t+1}^{\mathrm{dw}}} + \epsilon}, \tag{3.11}$$

[1] Algoritmo desenvolvido por Geoffrey Hinton e Tijmen, apresentado no curso de redes neurais pela plataforma Coursera em 2012 (*Slide* 29, disponível em :https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf)

em que ϵ é uma constante de estabilidade numérica utilizada para evitar divisão por zero.

O algoritmo Adam *(Adaptive moment estimation)* pode ser considerado como uma combinação do *RMSProp* com a técnica de *momentum* (GÉRON, 2019). A ideia principal é calcular taxas de aprendizado adaptativas para cada parâmetro da rede em treinamento. A partir da técnica de *momentum*, é capaz de armazenar a média dos valores dos gradientes passados e, com o *RMSProp*, é capaz de armazenar a média móvel dos valores dos gradientes de segunda ordem. Para isto, são aplicadas as Equações (3.12) e (3.13), utilizando os parâmetros β_1 e β_2, já apresentados nos algoritmos acima,

$$\upsilon_{t+1} = \beta_1 \upsilon_t - (1 - \beta_1)\, \nabla J(\mathrm{W}), \tag{3.12}$$

$$s^{d\mathrm{w}}_{t+1} = \beta_2 s^{d\mathrm{w}}_t + (1 - \beta_2) \nabla J(\mathrm{W}) \otimes \nabla J(\mathrm{W}), \tag{3.13}$$

em que υ é o momento de primeira ordem e $s^{d\mathrm{w}}$ é o momento de segunda ordem, ambos inicializados com zero. Em seguida, são computadas as versões atualizadas ou corrigidas após a aplicação de um fator de decaimento, que são dadas pelas seguintes equações:

$$\widehat{\upsilon}_{t+1} = \frac{\upsilon_{t+1}}{1 - \beta_{1,t+1}}, \qquad \widehat{s}_{t+1} = \frac{s_{t+1}}{1 - \beta_{2,t+1}}, \tag{3.14}$$

em que a atualização dos pesos da rede para este caso é dada pela Equação (3.15)

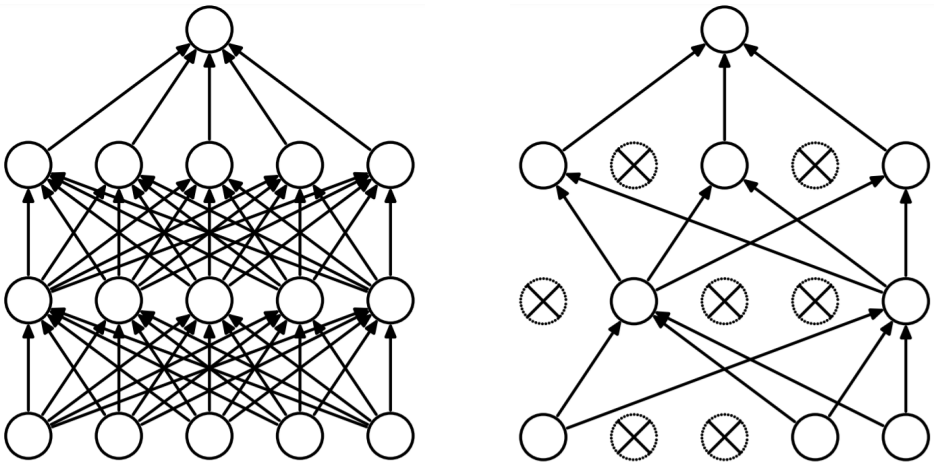
$$\mathrm{W}_{t+1} = \mathrm{W}_t - \eta \frac{\widehat{\upsilon}_{t+1}}{\sqrt{\widehat{s}_{t+1}} + \epsilon}. \tag{3.15}$$

3.3.2 Técnicas de Regularização

Além da utilização de algoritmos de otimização adequados para o treinamento da rede, é importante, também, a utilização de técnicas de regularização. Essas técnicas são essenciais para garantir que a rede funcione bem não apenas com conjunto de dados utilizado para treinamento como também com um conjunto de dados novo. Em outras palavras, as técnicas de regularização fornecem à rede a capacidade de generalização, ou seja, atuam para evitar o sobreajuste (*overfitting*) dos pesos sinápticos. Na literatura, as técnicas de regularização mais comuns são $\mathcal{L}_1$ e $\mathcal{L}_2$ *regularization* e *dropout*.

A técnica de *dropout*, proposta por Srivastava *et al.* (2014), altera, de forma dinâmica, a topologia da rede durante o treinamento. Neste caso, alguns neurônios da rede são inativados de acordo com uma probabilidade definida, ou seja, o valor de ativação desses neurônios são zerados. A ideia é que a partir do desligamento de algumas unidades da rede, esta se torne mais simples e evite o fenômeno de *overfitting*. A Figura 5 apresenta o funcionamento da técnica de *dropout* durante o treinamento da rede. Neste caso, a Figura 5 (a) apresenta uma rede neural multicamadas padrão e a Figura 5 (b) apresenta esta mesma rede com algumas unidades desligadas devido à técnica de *dropout*.

Figura 5 – Demonstração da técnica de *dropout* aplicada a uma rede neural.



(a) Rede neural padrão.

(b) Rede neural com técnica de *dropout*.

Fonte: Retirado de Srivastava *et al.* (2014).

3.3.3 Critérios de Desempenho

Uma etapa de grande importância no treinamento de DNNs é a escolha de critérios para avaliação do desempenho da rede. As métricas de avaliação constituem em uma ferramenta poderosa para análise dos resultados fornecidos pela DNN. Além disso, facilitam a tomada de decisão em relação a possíveis ajustes que podem ser realizados na topologia da rede para treinos futuros. As métricas mais comuns utilizadas para o problema de regressão são apresentadas na Tabela 3. Neste caso, $\hat{y}_i$ corresponde a i-ésima saída da rede, e y_i é a i-ésima saída desejada correspondente, e n corresponde a quantidade de amostras.

Tabela 3 – Métricas comuns para avaliação de uma rede neural aplicada à tarefa de regressão.

Métrica	**Sigla**	**Equação**
Erro Relativo Médio	*MRE*	$\frac{1}{n}\sum_{i=1}^{n}\frac{\lvert\hat{y}_i - yi\rvert}{y_i}$
Erro Quadrático Médio	*MSE*	$\frac{1}{n}\sum_{i=1}^{n}\frac{\lVert\hat{y}_i - yi\rVert^2}{y_i}$
Raiz do Erro Quadrático Médio	*RMSE*	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\parallel \hat{y}_i - y_i \parallel^2}$
Raiz do Erro Quadrático Médio Relativo	*RMSE-R*	$\sqrt{\frac{1}{n}\sum_{i=1}^{n}\parallel \frac{\hat{y}_i - yi}{y_i} \parallel^2}$
Erro Médio Absoluto	*MAE*	$\frac{1}{n}\sum_{i=1}^{n}\mid \hat{y}_i - yi \mid$

4 CONTROLE, ESTIMATIVA E ATENUAÇÃO DE DISTÚRBIOS

As incertezas que afetam uma planta em um projeto de controle, normalmente, são provenientes de duas fontes: distúrbios internos e distúrbios externos (XIE; GUO, 2000). Este capítulo trata das principais técnicas para estimativa e atenuação de distúrbios externos presentes na literatura. Além disso, aborda a arquitetura de Controle Baseado em Observação de Distúrbios (*DOBC - Disturbance Observer-Based Control*), que será utilizada neste trabalho para combinar os controladores com o estimador inteligente proposto. Por fim, serão abordados os detalhes teóricos referentes aos controladores utilizados.

4.1 Estimativa e Atenuação de Distúrbios

Distúrbios externos são responsáveis por afetar de forma negativa o desempenho e estabilidade de sistemas de controle. Na maioria dos casos, não podem ser mensurados de forma direta e, consequentemente, é difícil implementar uma técnica de controle para atenuar ou eliminar tais distúrbios sem adicionar sensores ao sistema. Devido a isso, diversas técnicas para estimativa de distúrbios foram propostas na tentativa de gerar uma ação de controle baseada no distúrbio estimado, visando reduzir ou eliminar os efeitos causados no projeto de controle de sistemas dinâmicos (CHEN *et al.*, 2016).

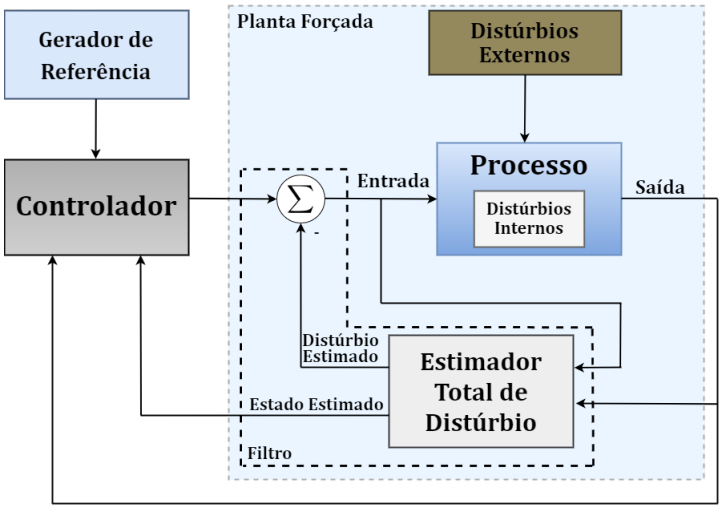
Embora tenham surgido diversas arquiteturas de controle para estimar e atenuar distúrbios externos, a ideia fundamental de funcionamento é a mesma: criar um mecanismo de observação de distúrbios e um compensador baseado na estimativa obtida (GAO, 2014). Essas técnicas podem ser encontradas na literatura pelo nome de *Disturbance and Uncertainty Estimation and Attenuation (DUEA)* e, dependendo das condições de operação do sistema, podem ser imprescindíveis para garantir não só os critérios de desempenho e estabilidade desejados como também os requisitos de robustez durante o seguimento de trajetória (CHEN *et al.*, 2016).

Segundo Yang *et al.* (2017), diversos métodos e algoritmos que surgiram para tratar distúrbios externos em sistemas de controle falharam em oferecer uma ação de controle robusta e adaptativa. Entretanto, técnicas DUEA desempenham um nível de funcionamento que está entre o controle robusto e o controle adaptativo, podendo ser utilizadas para suprir as deficiências que o projeto desses tipos de controladores apresentam. Isto porque são capazes de se adaptar as incertezas que afetam o sistema em vez de estimar diretamente os parâmetros incertos (LI *et al.*, 2014).

As técnicas DUEA combinadas com controladores robustos apresentam um enorme potencial para garantir estabilidade e desempenho robusto em sistemas afetados por distúrbios internos e externos. Dentre as técnicas mais utilizadas atualmente, podem-se

destacar: *Active Disturbance Rejection Control (ADRC)* (HAN, 2009), *Composite Hierarchical Antidisturbance Control (CHADC)* (GUO; CAO, 2014), *Disturbance Accommodation Control (DAC)* (JOHNSON, 1968), *Disturbance Observer-Based Control (DOBC)* (LI *et al.*, 2014) e *Extended High-Gain State Observer (EHGSO)* (FREIDOVICH; KHALIL, 2008). Para ilustrar o princípio de funcionamento das técnicas descritas acima, considere a Figura 6, em que é possível verificar um bloco composto por um controlador e um bloco composto pelo estimador e compensador de distúrbios.

Figura 6 – Diagrama de estimativa e rejeição de distúrbio.



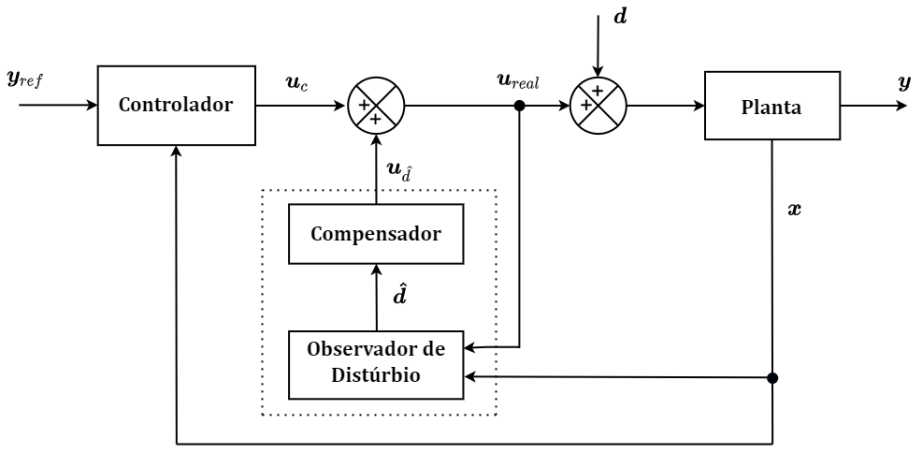
Fonte: Adaptado de Gao (2014).

4.2 Controle Baseado em Observação de Distúrbio

Segundo Guo e Cao (2014), o controle baseado em observação de distúrbio surgiu por volta de 1980 com o intuito de estimar distúrbios externos em sistemas robóticos. Atualmente, é uma das técnicas de estimativa e atenuação de distúrbios mais utilizadas, principalmente por ser intuitivo e dispor de uma arquitetura modular. Além disso, em comparação com outras técnicas similares, proporciona uma análise mais rigorosa a respeito da estabilidade e de outros requisitos de desempenho do sistema (CHEN *et al.*, 2016). Desde o seu surgimento, a técnica DOBC tem sido utilizada nos mais variados sistemas, incluindo: manipuladores robóticos (CHEN *et al.*, 2000), sistemas aeroespaciais e nucleares (LI; YANG, 2013), controle de servo motor (SUN; WANG; XIN, 2018) e controle de quadricópteros (SMITH; LIU; CHEN, 2016).

O princípio de funcionamento da arquitetura DOBC segue o princípio generalizado mencionado na seção anterior, em que é necessário um controlador principal e um estimador de distúrbio. Dessa forma, é possível projetar um compensador com base na dinâmica do sistema para atenuar ou eliminar os distúrbios externos. A Figura 7 apresenta a arquitetura do controle baseado em observação de distúrbio.

Figura 7 – Arquitetura de controle baseado em observação de distúrbio.



Fonte: Adaptado de Li *et al.* (2014).

Considerando a Figura 7, temos que y_{ref} é a trajetória de referência, y é a saída do sistema, x é o vetor composto pelos estados do sistema, d é o distúrbio externo, $\hat{d}$ representa a estimativa do distúrbio proveniente do estimador, u_c é o sinal do controle do controlador, $u_{\hat{d}}$ é o sinal de controle dado pelo compensador com base na estimativa do distúrbio, e u_{real} é o sinal de controle composto a partir dos dois sinais de controle calculados anteriormente. Observe que na ausência de distúrbios externos, o bloco interno responsável pela estimativa e compensação de distúrbio não é ativado. Dessa forma, o controlador principal pode ser projetado com base nos requisitos de desempenho e estabilidade do sistema, sem considerar distúrbios e incertezas externas. Já o bloco interno é projetado apenas para estimar e rejeitar os distúrbios externos que afetam o desempenho do sistema (CHEN *et al.*, 2016).

Ao longo dos anos, diversos tipos de estimadores de distúrbio surgiram para integrar a arquitetura DOBC e outras arquiteturas que se baseiam na estimativa de distúrbios, como as mencionadas na seção 4.1. Alguns exemplos mais comuns na literatura, são: *Unknown Input Observer (UIO)* (HOSTETTER˜; MEDITCH, 1973), *Extended State Observer (ESO)* (MA *et al.*, 2016; GUO; ZHAO, 2011), *Perturbation Observer (POB)* (KWON; CHUNG, 2003) e *Disturbance Observer (DOB)* (LI *et al.*, 2014; SMITH; LIU; CHEN, 2016; YANG *et al.*, 2017). Além disso, foram propostas adaptações para o Filtro de Kalman (FK) e suas principais variantes, com o intuito de obter estimativas ótimas dos distúrbios que afetam o sistema. Para mais detalhes sobre observadores de distúrbios, veja (RADKE; GAO, 2006).

Recentemente, alguns trabalhos propuseram a possibilidade de utilizar sistemas inteligentes, como redes neurais artificiais, para rejeitar ou atenuar a influência de distúrbios externos em sistemas robóticos. Em vista disso, é possível encontrar na literatura uma gama de arquiteturas compostas por redes neurais para controle de posição e orientação de quadricópteros sujeitos a distúrbios externos, veja, por exemplo, (XU *et al.*, 2017; XU *et al.*, 2019; HUO; MENG; JIN, 2019).

4.3 Controladores

Esta seção apresenta os controladores implementados para formação das arquiteturas de controle propostas. Estes controladores foram combinados com redes neurais para rastreamento de trajetória de um quadricóptero. Os detalhes referentes as arquiteturas desenvolvidas são apresentados no capítulo seguinte.

4.3.1 Regulador Linear Quadrático Recursivo (LQR)

O Regulador Linear Quadrático (LQR) é um controlador ótimo que permite a obtenção de uma lei de controle através da minimização de um funcional de custo. Para isto, considere o sistema linear representado no espaço de estado descrito em (4.1).

$$x_{i+1} = F_i x_i + G_i u_i; \quad i = 0, ..., N. \tag{4.1}$$

A partir de um estado inicial não nulo x_0, deve-se encontrar um sinal de controle u_i que leve o estado do sistema para zero ($x = 0$) de maneira ótima. Para isso, o seguinte problema de minimização deve ser resolvido

$$J = \frac{1}{2}\sum_{i=0}^{\infty}\{x_i^T Q x_i + u_i^T R u_i\}, \tag{4.2}$$

sendo Q $\succeq 0$ e $R \succ 0$ matrizes de ponderação que definem a importância do erro e, em geral, o desempenho do controlador (OGATA, 2010). A lei de controle ótima para o LQR é dada por

$$u_i^* = -K x_i, \tag{4.3}$$

em que o ganho K é dado por

$$K = (R + B^T S B)^{-1} B^T S A, \tag{4.4}$$

e S satisfaz a equação algébrica de Riccati, definida como

$$A^T S A - A^T S B R^{-1} B^T S A + Q - S = 0. \tag{4.5}$$

Uma nova abordagem para o Regulador Linear Quadrático para sistemas lineares de tempo discreto é apresentado em Cerri (2009). Uma grande vantagem desta abordagem parte do desenvolvimento de um novo funcional de custo através da combinação de função penalidade e função custo do tipo jogos. Dessa forma, a recursividade pode ser obtida sem a necessidade de ajuste de parâmetros auxiliares, o que é de grande importância para aplicações online e, consequentemente, para o controle de rastreamento de trajetória de quadricópteros. Para este caso, volte a considerar a Equação (4.1). A função custo agora é dada por

$$\mathcal{J} = x_{N+1}^T P_{N+1} x_{N+1} + \sum_{i=0}^{N} \mathcal{L}_i(x_i, u_i), \tag{4.6}$$

com uma expressão auxiliar definida por

$$\mathcal{L}_i(x_i, u_i) = x_i^T Q x_i + u_i^T R u_i; \quad i = 0, ..., N, \tag{4.7}$$

e sendo as matrizes de ponderação conhecidas, $P_{N+1} \succ 0$, $Q \succ 0$ e $R \succ 0$. Procura-se resolver o seguinte problema de minimização

$$\begin{aligned} &\min_{x_{i+1}, u_i} \left\{ x_{i+1}^T P_{i+1} x_{i+1} + \sum_{i=0}^{N} \mathcal{L}_i(x_i, u_i) \right\} \\ &s.a \;\; x_{i+1} = F_i x_i + G_i u_i; \;\; i = 0, ..., N. \end{aligned} \tag{4.8}$$

De forma análoga, o problema de minimização pode ser representado como um problema de minimização irrestrita do funcional quadrático

$$\mathcal{J}_i(x_{i+1}, u_i) = \begin{bmatrix} x_{i+1} \\ u_i \end{bmatrix}^T \begin{bmatrix} P_{i+1} & 0 \\ 0 & R_i \end{bmatrix} \begin{bmatrix} x_{i+1} \\ u_i \end{bmatrix} + \Phi^T \begin{bmatrix} Q_i & 0 \\ 0 & \mu I \end{bmatrix} \Phi, \tag{4.9}$$

em que

$$\Phi = \left\{ \begin{bmatrix} 0 & 0 \\ I & -G_i \end{bmatrix} \begin{bmatrix} x_{i+1} \\ u_i \end{bmatrix} - \begin{bmatrix} -I \\ F_i \end{bmatrix} x_i \right\}.$$

Em relação às variáveis x_{i+1} e u_i para cada valor fixado no parâmetro de penalidade (μ). A solução ótima é alcançada quando $\mu \to +\infty$ (CERRI, 2009). De forma recursiva é dada por:

$$\begin{bmatrix} x_{i+1}^* \\ u_i^* \\ \tilde{J}_i(x_{i+1}^*, u_i^*) \end{bmatrix} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & x_i^{*T} \end{bmatrix} \begin{bmatrix} L_i \\ K_i \\ P_i \end{bmatrix} x_i^*, \tag{4.10}$$

em que

$$\begin{bmatrix} L_i \\ K_i \\ P_i \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -I \\ 0 & 0 & F_i \\ I & 0 & 0 \\ 0 & I & 0 \end{bmatrix}^T \begin{bmatrix} P_{i+1}^{-1} & 0 & 0 & 0 & I & 0 \\ 0 & R_i^{-1} & 0 & 0 & 0 & I \\ 0 & 0 & Q_i^{-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I & -G_i \\ I & 0 & 0 & I & 0 & 0 \\ 0 & I & 0 & -G_i^T & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ -I \\ F_i \\ 0 \\ 0 \end{bmatrix}. \tag{4.11}$$

O vetor de controle ótimo recursivo é dado por

$$\begin{bmatrix} x_{i+1}^* \\ u_i^* \end{bmatrix} = \begin{bmatrix} L_i \\ K_i \end{bmatrix}, \qquad i = 0, ..., N. \tag{4.12}$$

As deduções e a solução deste problema podem ser encontradas com detalhes em Cerri (2009). Quando aplicado em quadricópteros, este controlador possibilita o controle de rastreamento de trajetórias através da minimização do erro calculado entre a posição de referência com relação à posição atual da aeronave.

4.3.2 Regulador Linear Quadrático Robusto Recursivo (RLQR)

No trabalho de Terra, Cerri e Ishihara (2014), foi desenvolvido um Regulador Linear Quadrático Robusto para sistemas lineares de tempo discreto sujeitos a incertezas paramétricas. Para isso, o regulador baseia-se em parâmetros de penalidade e mínimos quadrados regularizados, fornecendo um algoritmo que não depende de parâmetros de ajuste em aplicações online. Considere o modelo linear de tempo discreto com incertezas paramétricas no espaço de estados:

$$x_{i+1} = (F_i + \delta F_i)x_i + (G_i + \delta G_i)u_i; \quad i = 0, ..., N, \tag{4.13}$$

no qual $x_i \in \mathbb{R}^n$ é o vetor de estado, $u_i \in \mathbb{R}^m$ é a entrada de controle, $F_i \in \mathbb{R}^{nxn}$ e $G_i \in \mathbb{R}^{nxm}$ são matrizes de parâmetros nominais com dimensão apropriada do sistema e N é um inteiro que define o número de interações. O estado inicial x_0 é constante e conhecido, e as matrizes $\delta F_i \in \mathbb{R}^{nxn}$ e $\delta G_i \in \mathbb{R}^{nxm}$ são matrizes de incertezas desconhecidas, modeladas como

$$[\delta F_i \quad \delta G_i] = H_i \Delta_i [E_{F_i} \quad E_{G_i}]; \quad i = 0, ..., N, \tag{4.14}$$

sendo $H_i \in \mathbb{R}^{nxk}$, $E_{F_i} \in \mathbb{R}^{nxn}$, $E_{G_i} \in \mathbb{R}^{nxn}$ matrizes conhecidas, determinadas, na maioria dos casos, de forma heurística. $\Delta_i \in \mathbb{R}^{kxl}$ é uma matriz arbitrária com $\|\Delta_i\| \leq 1$. O RLQR é obtido através da solução do seguinte problema de otimização:

$$\min_{x_{i+1}, u_i} \max_{\delta F_i, \delta G_i} \left\{ \tilde{J}_i^{\mu}(x_{i+1}, u_i, \delta F_i, \delta G_i) \right\}, \tag{4.15}$$

sendo $\tilde{J}_i^u$ o funcional de custo quadrático regularizado, definido como

$$\tilde{J}_i^{\mu}(x_{i+1}, u_i, \delta F_i, \delta G_i) =$$

$$\begin{bmatrix} x_{i+1} \\ u_i \end{bmatrix}^T \begin{bmatrix} P_{i+1} & 0 \\ 0 & R_i \end{bmatrix} \begin{bmatrix} x_{i+1} \\ u_i \end{bmatrix} + \Phi^T \begin{bmatrix} Q_i & 0 \\ 0 & \mu I \end{bmatrix} \Phi, \tag{4.16}$$

em que $P_{N+1} \succ 0$ $Q \succ 0$ e $R \succ 0$ são matrizes conhecidas, $\mu > 0$ é um parâmetro de penalidade fixo, responsável por garantir que a igualdade da Equação (4.13) se mantenha, e

$$\Phi = \left\{ \begin{bmatrix} 0 & 0 \\ I & -G_i - \delta G_i \end{bmatrix} \begin{bmatrix} x_{i+1} \\ u_i \end{bmatrix} - \begin{bmatrix} -I \\ F_i + \delta F_i \end{bmatrix} x_i \right\}.$$

De acordo com Terra, Cerri e Ishihara (2014), o RQLR baseia-se na solução $(x_{i+1}^T(\mu), u_i^*(\mu))$ do problema de otimização $min - max$, no qual o intuito é obter a menor magnitude do estado e a menor magnitude da ação de controle nos piores casos de incertezas paramétricas, para cada parâmetro de penalidade $\mu > 0$. A solução ótima pode

ser encontrada de forma recursiva e é dada pelas equações (4.17) e (4.18). Além disso, quando o parâmetro de penalidade $\mu \to \infty$, a robustez do controlador é alcançada.

$$\begin{bmatrix} x^*_{i+1}(\mu) \\ u^*_i(\mu) \\ \tilde{J}^\mu_i(\hat{x}_{i+1}(\mu), \hat{u}_i(\mu)) \end{bmatrix} = \begin{bmatrix} I_n & 0 & 0 \\ 0 & I_m & 0 \\ 0 & 0 & x_i(\mu)^T \end{bmatrix} \begin{bmatrix} L_i, \mu \\ K_i, \mu \\ P_i, \mu \end{bmatrix} x^*_i(\mu), \tag{4.17}$$

com

$$\begin{bmatrix} L_i \\ K_i \\ P_i \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -I \\ 0 & 0 & \mathcal{F}_i^T \\ I & 0 & 0 \\ 0 & I & 0 \end{bmatrix}^T \begin{bmatrix} P^{-1}_{i+1} & 0 & 0 & 0 & I & 0 \\ 0 & R_i^{-1} & 0 & 0 & 0 & I \\ 0 & 0 & Q_i^{-1} & 0 & 0 & 0 \\ 0 & 0 & 0 & \Sigma_i\left(\mu, \hat{\lambda}_i\right) & \mathcal{I} & -\mathcal{G}_i \\ I & 0 & 0 & \mathcal{I}^T & 0 & 0 \\ 0 & I & 0 & -\mathcal{G}_i^T & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ -I \\ \mathcal{F}_i \\ 0 \\ 0 \end{bmatrix}, \tag{4.18}$$

em que,

$$\Sigma_i = \begin{bmatrix} \mu^{-1} I - \hat{\lambda}_i^{-1} H_i H_i^T & 0 \\ 0 & \hat{\lambda}_i^{-1} I \end{bmatrix}, \qquad \mathcal{I} = \begin{bmatrix} I \\ 0 \end{bmatrix}, \ \mathcal{G}_i = \begin{bmatrix} G_i \\ E_{G_i} \end{bmatrix}, \ \mathcal{F}_i = \begin{bmatrix} F_i \\ E_{F_i} \end{bmatrix}.$$

Sendo assim, a lei de controle ótima é dada por

$$u^*_i = K_i x^*_i, \qquad i = 0, ..., N. \tag{4.19}$$

Os detalhes sobre a garantia de convergência e estabilidade podem ser encontrados em Terra, Cerri e Ishihara (2014).

4.3.3 Controlador Proporcional-Integral-Derivativo (PID)

Atualmente, grande parte dos controladores em uso nas indústrias são do tipo PID. Sua ampla utilização está diretamente relacionada com a facilidade de aplicação nos mais variados projetos de sistemas de controle. Além disso, apresenta bom desempenho em variadas condições de operação, embora, em situações mais críticas, pode não oferecer um controle satisfatório do sistema (OGATA, 2010). A abordagem clássica do controlador PID pode ser utilizada para controle de rastreamento de trajetória de quadricópteros de forma mais simples que técnicas de controle ótimo. Entretanto, há de se esperar um desempenho inferior deste controlador quando comparado a controladores mais elaborados.

Para formulação do controlador PID, a lei de controle deve ter como referência as informações de erro obtidas através da diferença entre a reposta esperada e a resposta atual do sistema. Para isto, considere a Equação (4.20) a seguir

$$\tilde{q}(t) = q_g(t) - q^d(t), \tag{4.20}$$

em que o q_g é a saída do sistema e q^d é a saída desejada. Dessa forma, o vetor de controle é dado por

$$u_{(t)} = K_p\, \tilde{q}(t) + K_i \int_{t_0}^{t} \tilde{q}(t) + K_d\, \dot{\tilde{q}}(t), \tag{4.21}$$

em que K_p é o ganho proporcional, K_i é o ganho integral e K_d é o ganho derivativo.

4.3.4 Linearização por Realimentação (LR)

A técnica de linearização por realimentação (do inglês, *feedback linearization*) foi inicialmente proposta por Brockett (1978). Considerada como uma técnica precursora para controle de sistemas não lineares, foi de extrema importância para a evolução dos estudos nesta linha de pesquisa. A ideia central desta técnica é transformar algebricamente sistemas não lineares em sistemas totalmente ou parcialmente lineares (SASTRY, 2013).

Atualmente, a técnica de linearização por realimentação é utilizada nas mais variadas aplicações, como quadricópteros, helicópteros autônomos, robôs industriais e veículos autônomos terrestres. A obtenção da lei de controle clássica desta técnica será apresentada a seguir. Para isto, considere o sistema não linear descrito na Equação (4.22)

$$x^{(n)} = f(x) + g(x)u, \tag{4.22}$$

em que $f(x)$ e $g(x)$ são funções não lineares do estado $x = [x,\ \dot{x},\ \ddot{x},\ ...,\ x^{(n-1)}]$ e u é a entrada de controle do sistema. Podemos representar o sistema na forma companheira, obtendo

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} x_2 \\ x_3 \\ \vdots \\ f(x) + g(x)u \end{bmatrix}. \tag{4.23}$$

Se assumirmos $g(x) \neq 0$, temos que uma entrada de controle pode ser dada por

$$u = \frac{v - f(x)}{g(x)}, \tag{4.24}$$

em que a realimentação por linearização resultante é dada por

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} x_2 \\ x_3 \\ \vdots \\ v \end{bmatrix}, \tag{4.25}$$

no qual v, no problema de regulação, é uma lei de controle que possui a seguinte forma,

$$v = -k_1 x_1 - k_2 x_2 - k_3 x_3, ..., -k_n x_n. \tag{4.26}$$

Para o problema de rastreamento de trajetória, temos que a lei de controle passa a ser dada pela Equação (4.27) a seguir

$$v = x_n^d - k_1 \tilde{x}_1 - k_2 \tilde{x}_2 - k_3 \tilde{x}_3, ..., -k_n \tilde{x}_n, \tag{4.27}$$

em que x^d é a trajetória desejada e $\tilde{x}$ é o erro de rastreamento de trajetória. Para o modelo do quadricóptero utilizado neste trabalho, temos que a lei de controle de linearização por realimentação passa a ser dada por

$$\nu = B^{-1}(u - AR_t^T \dot{q}), \tag{4.28}$$

em que $u = \ddot{q}^d - K_1 \tilde{q} - K_2 \dot{\tilde{q}}$, com q^d sendo a trajetória desejada, q o vetor de estados do quadricóptero e $\tilde{q}$ é o erro de rastreamento de trajetória.

5 DEFINIÇÃO DAS ARQUITETURAS PROPOSTAS

Considerando os conceitos apresentados nos capítulos 3 e 4, neste capítulo serão apresentadas as duas arquiteturas de controle propostas para melhorar o rastreamento de trajetória de quadricópteros. Em ambas as arquiteturas, o objetivo é combinar redes neurais profundas com os controladores apresentados na seção 4.3.

A seção 5.1 apresenta os detalhes da arquitetura 1 proposta, em que a DNN tem a função de enviar um sinal de referência factível ao controlador com base em experiências passadas. Esta arquitetura de controle foi baseada no trabalho de Li *et al.* (2017), que combina seis DNNs com um controlador PID para controlar os seis graus de liberdade de um quadricóptero.

A seção 5.2 apresenta os detalhes de funcionamento da arquitetura 2, em que a DNN atua como estimador de distúrbio em conjunto com o filtro de Kalman e com os controladores implementados, formando uma arquitetura DOBC. Neste caso, a DNN é responsável por estimar o distúrbio de vento que afeta o quadricóptero e, em seguida, um compensador tem a função de atenuar o distúrbio, melhorando o seguimento de trajetória.

5.1 Rede Neural Como Referência Para os Controladores

Esta seção aborda o funcionamento da primeira arquitetura de controle proposta, em que a DNN tem a função de enviar um sinal de referência para os controladores implementados. Em geral, para controle de posição no problema de rastreamento de trajetória, o controlador recebe a trajetória desejada e fornece uma lei de controle que visa minimizar o erro de posição do quadricóptero durante o percurso. Entretanto, com a arquitetura 1 proposta, o controlador passa a receber a trajetória de referência proveniente da DNN ao invés da trajetória desejada. A diferença é que o sinal de referência fornecido pela DNN é computado a partir do conhecimento adquirido durante o treinamento, considerando a posição atual e a posição desejada da aeronave. Em outras palavras, após o treinamento, a rede se torna capaz de prover um sinal ao controlador que facilita a minimização do erro de rastreamento de trajetória.

A ideia principal é que se a trajetória atual for igual à trajetória desejada, a rede deverá fornecer esse valor de referência guardado para que o rastreamento continue com o menor erro possível. Por outro lado, quando a trajetória do quadricóptero for diferente da desejada, o mapeamento aprendido pela rede irá gerar um sinal de referência factível para o controlador, visando corrigir o curso do quadricóptero e fazendo com que este convirja mais rapidamente para o trajeto desejado. Neste caso, o sinal fornecido pela rede possui magnitude ligeiramente diferente da trajetória desejada, podendo ter amplitude maior ou

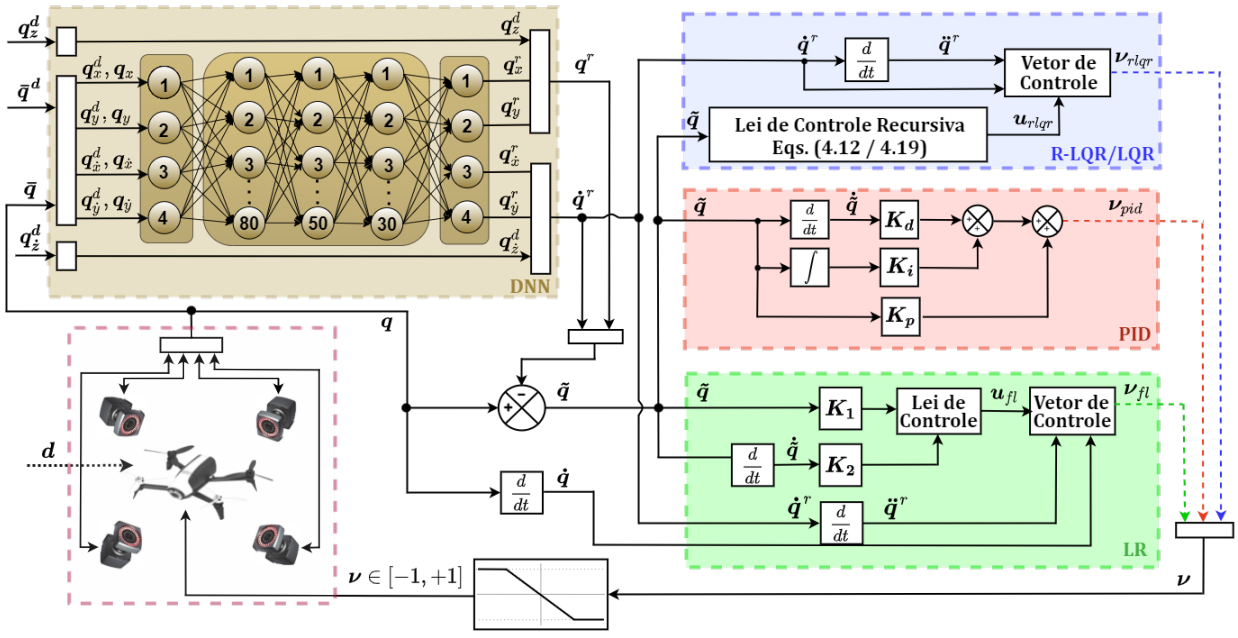
menor.

Considerando que o objetivo da DNN é mapear um sinal de referência ($\bar{q}^r$) a cada instante de tempo e enviar ao controlador, baseado nos valores da trajetória desejada ($\bar{q}^d$) e atual ($\bar{q}$) do quadricóptero, a rede é treinada de forma *offline* através do método de aprendizado supervisionado, no qual os detalhes serão discutidos na subseção 6.3.1. Após o treinamento, o conhecimento adquirido é aplicado em tempo real junto ao controlador, promovendo uma melhoria de desempenho durante o seguimento de trajetória.

O funcionamento da arquitetura proposta em conjunto com os controladores pode ser visualizado na Figura 8. Vale lembrar que, quando as incertezas paramétricas não são consideradas, o RLQR recai na forma do LQR. Pela figura, observa-se que a rede obtém informações tanto do gerador de trajetória (trajetória desejada) quanto do sistema de odometria do quadricóptero (trajetória atual). Note que para voos experimentais, neste trabalho, a rede obtém informações sobre a trajetória atual do quadricóptero pelo sistema de captura de movimentos óptico, que será discutido na subseção 6.2.2.

A trajetória desejada é formada pelo vetor $q^d = [q^d_x, q^d_y, q^d_z, q^d_\psi]$ (seção 6.4), sendo q^d_ψ definido como zero. Visando simplificar, o sinal fornecido pela rede atua nos eixos x e y, mantendo o eixo z com valor padrão fornecido pelo gerador de trajetória. Dessa forma, a entrada da rede passa a ser composta pelo vetor $\{(\bar{q}_i, \bar{q}^d_i)\}$, em que $\bar{q}_i = [q_x, q_y, q_{\dot{x}}, q_{\dot{y}}]$ e $\bar{q}^d_i = [q^d_x, q^d_y, q^d_{\dot{x}}, q^d_{\dot{y}}]$, correspondem a posições e velocidades atuais e desejadas nos eixos x e y, respectivamente. A saída gerada é composta pelo vetor $\bar{q}^r = \{q^r_x, q^r_y, q^r_{\dot{x}}, q^r_{\dot{y}}\}$, correspondendo à posições e velocidades de referência.

Figura 8 – Rede neural profunda como referência para os controladores.



Fonte: Elaborada pelo autor.

Vale ressaltar, ainda, que na Figura 8, $\nu = [\nu_{\dot{x}_b}\ \nu_{\dot{y}_b}\ \nu_{\dot{z}_b}\ \nu_{\dot{\psi}}]$ representa o vetor de velocidade enviado ao quadricóptero e d é o distúrbio de vento que afeta a aeronave. A fim de facilitar o entendimento a respeito da arquitetura proposta e complementar o diagrama apresentado na Figura 8, o algoritmo 1 apresenta os passos de funcionamento, tendo como exemplo o controlador RLQR. Note que para todos os controladores implementados, o algoritmo é o mesmo. À vista disso, modifica-se apenas a etapa de inicialização do controlador e a sua respectiva lei de controle.

Algoritmo 1 Rede neural como referência para os controladores

Entrada: Trajetória desejada q^d
Saída: Vetor de velocidade ν

1: Inicialização do RLQR: considere (4.13), (4.14), and (4.16) com F_i, G_i, E_{F_i}, E_{G_i}, $Q_i \succ 0$, e $R_i \succ 0$ conhecidas para todo instante i.
2: **Enquanto** receber a trajetória desejada **faça**
3: Guarda q^d e q
4: Calcula $\dot{q}^d$ e $\dot{q}$
5: Computa q^r e $\dot{q}^r$
6: Calcula $\tilde{q}$ e $\dot{\tilde{q}}$
7: Calcula o ganho RLQR com a Eq. (4.18)
8: $u_{rlqr,i} = K_{rlqr,i}x_i$
9: Usa u_{rlqr} em (2.12) para gerar ν
10: **Fim Enquanto**

5.2 Rede Neural Como Estimador de Distúrbios

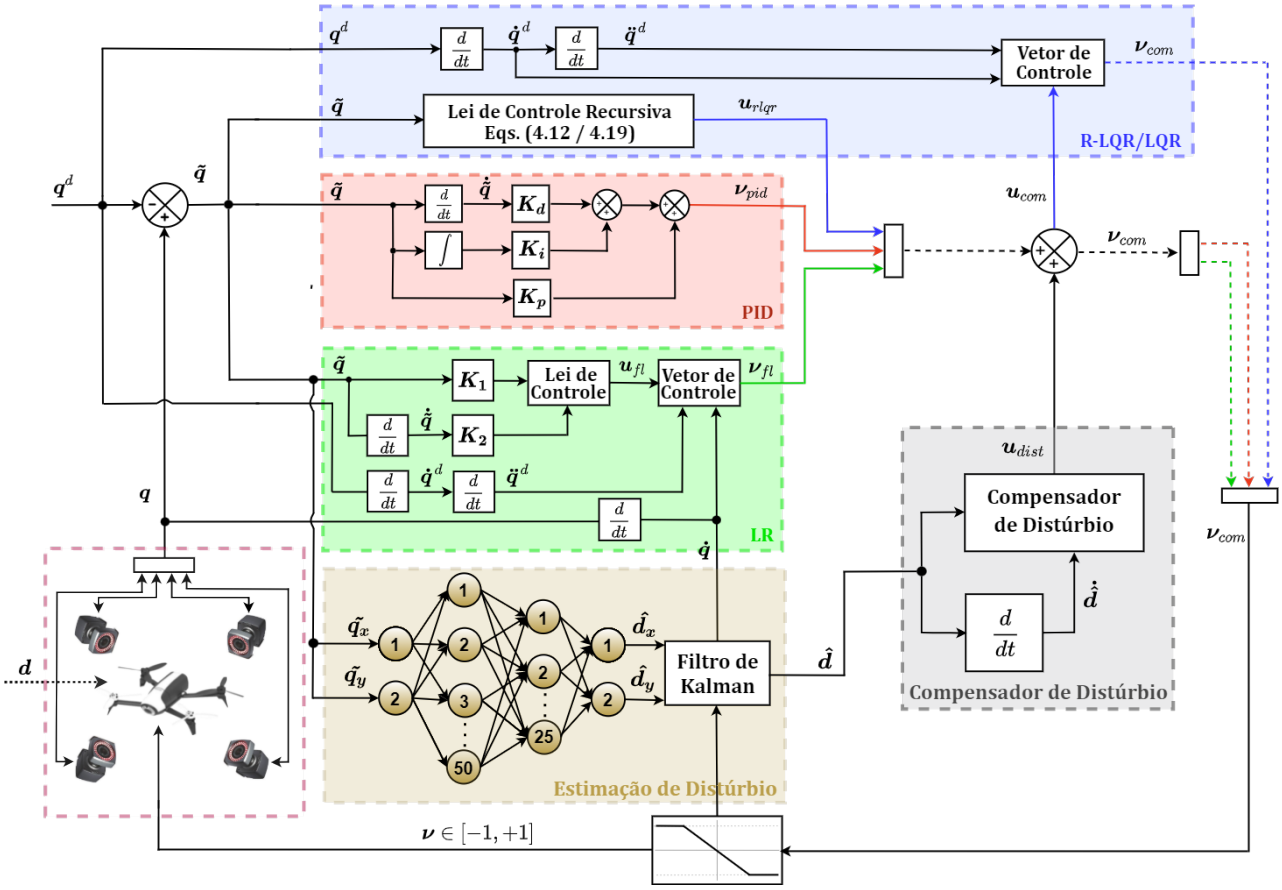
Esta seção aborda os detalhes de funcionamento da segunda arquitetura de controle proposta, em que a DNN é responsável por estimar os distúrbios externos. Para este caso, foi utilizada a arquitetura DOBC (seção 4.2) com os controladores mencionados na seção 4.3. Aqui, os controladores são responsáveis pelo seguimento de trajetória, a DNN pela estimativa de distúrbio e um compensador foi projetado para atenuação do distúrbio estimado.

O objetivo principal da DNN é fornecer uma estimativa do distúrbio nos eixos x e y a partir do erro de trajetória $\tilde{q}_x$ e $\tilde{q}_y$, respectivamente. Para isto, foi levantado um *dataset*, em que distúrbios de vento com magnitudes distintas foram aplicados ao quadricóptero e o erro de posição gerado por estes distúrbios foram guardados com o respectivo rótulo (Força em *Newtons* do distúrbio). Durante o funcionamento, o sinal de distúrbio fornecido pela DNN é filtrado e direcionado a um compensador. Posteriormente, uma lei de controle composta é gerada com capacidade de atenuar os distúrbios de vento que afetam a aeronave.

A Figura 9 apresenta o funcionamento da arquitetura proposta. De forma similar à Figura 8, cada controlador é utilizado de forma individual com a rede neural proposta. O erro de posição é calculado utilizando a posição atual e desejada a cada instante de

tempo. Para os voos no ambiente simulado, a posição atual a cada instante foi obtida do sistema de odometria do quadricóptero; para os experimentos práticos, foram utilizadas as informações do sistema de captura de movimentos Vicon® (subseção 6.2.2). Após os valores de erro passar pela rede, o sinal de distúrbio gerado é suavizado pelo filtro de Kalman. Em seguida, o compensador (bloco cinza) é responsável por atenuar o distúrbio, formando a lei de controle composta.

Figura 9 – Arquitetura de controle baseado em observação de distúrbio com rede neural profunda e filtro de Kalman.



Fonte: Elaborada pelo autor.

Nas próximas subseções serão apresentadas as etapas de filtragem e compensação do distúrbio estimado. A partir disso, será apresentada a lei de controle composta, que utiliza tanto o sinal de controle proveniente do controlador quanto da estimativa do distúrbio.

5.2.1 Filtro de Kalman

O filtro de Kalman (FK) (KALMAN, 1960), pode ser definido como um conjunto de ferramentas matemáticas importantes, capazes de fornecer uma estimativa das variáveis de processo através da minimização do erro quadrático médio (WELCH; BISHOP, 2006). Desde que foi proposto, em 1960, o FK passou a ser um dos principais algoritmos da engenharia e, atualmente, ainda é um dos mais utilizados. Este algoritmo não necessita de

dados históricos do sistema em que atua, apenas com informações dos estados no tempo anterior é capaz de fornecer uma estimativa ótima dos estados futuros de um sistema dinâmico estocástico, considerando distúrbios e incertezas de processo e medição (ULLAH; FAYAZ; KIM, 2019).

Através da utilização da técnica de estado aumentado, o filtro de Kalman é capaz de fornecer não só a forma filtrada dos estados de um sistema como também dos distúrbios externos que o afeta (KIM; PARK, 2017). O funcionamento deste filtro é apresentado na Tabela 4, onde é possível verificar os detalhes de cada etapa desempenhada pelo filtro.

Tabela 4 – Equações do filtro de Kalman.

Elemento	**Equações**
Modelo do Sistema	$s_{i+1} = \Theta_i \widehat{s}_i + \Xi_i u_i$
Modelo de Saída	$z_i = \Omega_i s_i + v_s$
Matrizes de Ruídos	Ruído de Processo: $Q : w_i \sim (0, Q_i)$ Ruído de Medição: $R : v_i \sim (0, R_i)$.
Valores Iniciais	$\hat{x}_0^+ = E[x_0], P_0^+ = E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T]$
Etapa de Predição	Predição do Estado: $\widehat{s}_{i+1\|i} = \Theta_{i+1}\widehat{s}_{i\|i} + \Xi_{i+1}u_{i+1}$. Predição da Covariância do Erro: $P_{i+1\|i} = \Theta_{i+1}P_{i\|i}\Theta_{i+1}^T + Q_{i+1}$.
Etapa de Medição	Ganho de Kalman: $K_{i+1} = P_{i+1\|i}\Omega_{i+1}^T(\Omega_{i+1}P_{i+1\|i}\Omega_{i+1}^T + R_{i+1})^{-1}$. Correção da Estimativa: $\widehat{s}_{i+1\|i+1} = \widehat{s}_{i+1\|i} + K_{i+1}(z_{i+1} - \Omega_{i+1}\widehat{s}_{i+1\|i})$. Correção da Covariância do Erro: $P_{i+1\|i+1} = (I - K_{i+1}\Omega_{i+1})\, P_{i+1\|i}$

Considerando a Tabela 4, temos que $s \in \mathbb{R}^{n\times 1}$ é o vetor de entradas, $\Theta \in \mathbb{R}^{n\times n}$ é a matriz do sistema, $\Xi \in \mathbb{R}^{n\times n}$ é a matriz de entrada, $u \in \mathbb{R}^{n\times 1}$ é o vetor de controle, $\omega \in \mathbb{R}^{n\times 1}$ é o vetor que representa o ruído de processo, com distribuição normal, média zero e covariância dada pela matriz $Q : \omega_i \sim (0, Q_i)$. Para equação de medição, considere $z \in \mathbb{R}^{m\times 1}$ é o vetor de medição, $\Omega \in \mathbb{R}^{m\times n}$ é a matriz de medição, que representa a saída do sistema, $v \in \mathbb{R}^{m\times 1}$ é o vetor que representa o ruído de medição, com distribuição normal, média zero e covariância dada pela matriz $R : v_i \sim (0, R_i)$.

Com intuito de obter a forma filtrada dos estados e dos distúrbios que afetam o sistema, é construído um vetor de estado aumentado. Neste caso, o distúrbio d e sua

respectiva derivada são dispostos em conjunto com o vetor de estados x, formando:

$$s_{i+1} = \begin{bmatrix} x_{i+1} \\ d_{a,i+1} \\ d_{b,i+1} \end{bmatrix}, \tag{5.1}$$

em que o distúrbio externo em tempo discreto e sua respectiva derivada são dados por

$$\begin{bmatrix} d_{a,i+1} \\ d_{b,i+1} \end{bmatrix} = \begin{bmatrix} I & TI \\ 0 & I \end{bmatrix} \begin{bmatrix} d_{a,i} \\ d_{b,i} \end{bmatrix} + \begin{bmatrix} \mathrm{w}_{a,i} \\ \mathrm{w}_{b,i} \end{bmatrix},$$

no qual, $W_{a,i}$ e $W_{b,i}$ são ruídos de processo associados. Dessa forma, para o sistema descrito na Tabela 4, o sistema na forma aumentada considerando o vetor de estados é dado por

$$\begin{aligned} \underbrace{\begin{bmatrix} x_{i+1} \\ d_{a,i+1} \\ d_{b,i+1} \end{bmatrix}}_{s_{i+1}} &= \underbrace{\begin{bmatrix} \Psi_i & G_{d,i} & 0 \\ 0 & I & TI \\ 0 & 0 & I \end{bmatrix}}_{\Theta} \underbrace{\begin{bmatrix} x_i \\ d_{a,i} \\ d_{b,i} \end{bmatrix}}_{s} + \underbrace{\begin{bmatrix} \Upsilon_i \\ 0 \\ 0 \end{bmatrix}}_{\Xi} u_i, \\ z_k &= \underbrace{\begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\Omega} \underbrace{\begin{bmatrix} x_i \\ d_{a,i} \\ d_{b,i} \end{bmatrix}}_{s} + v_i, \end{aligned} \tag{5.2}$$

em que $\Psi_i = I + TAR_t^T$ e $\Upsilon_i = T\,B(t)$ são obtidos a partir da discretização da Equação (2.6), utilizando o método de Euler. Desse modo, são obtidas as formas filtradas dos estados e do distúrbio estimado pela DNN. Em seguida, o distúrbio estimado passa pela etapa de compensação, em que um compensador é responsável por atenuar seus efeitos no sistema.

5.2.2 Compensação do Distúrbio Estimado

Com a estimativa filtrada do distúrbio fornecido pela DNN, um compensador deverá ser projetado para atenuar ou rejeitar a influência do distúrbio real no sistema de controle. Segundo Chen *et al.* (2016) e Li *et al.* (2014), para o caso em que uma planta é afetada por distúrbios de vento ou até mesmo por torques causados por dinâmicas não modeladas, um compensador pode ser obtido seguindo os passos a seguir.

Considere a lei de controle gerada pelo RLQR, $u_{rlqr,i} = -K_{rlqr,i}x_i$, em que K_{rlqr} é o ganho ótimo recursivo e x é o vetor de estados. Para este controlador, a lei de controle composta é formada por:

$$u_{com,i} = u_{rlqr,i} + u_{dist,i}, \tag{5.3}$$

em que $u_{dist,i} = K_{com}\widehat{d}_i$ é a lei de controle de compensação de distúrbio. Definindo $\Lambda := F_i - G_iK_{rlqr,i}$ como sendo um ganho de malha fechada e utilizando a lei de controle composta u_{com} com o sistema (2.6) discretizado, temos que

$$x_{i+1} = \Lambda x_i + G_i K_{com,i} \widehat{d}_i + G_{d,i} d_i. \tag{5.4}$$

Podemos isolar o vetor de estados x para obter

$$x_i = \Lambda^{-1}(x_{i+1} - G_i K_{com,i} \widehat{d}_i - G_{d,i} d_i). \tag{5.5}$$

Considerando que a saída do sistema é dada por $y_i = C x_i$, temos que

$$y_i = C\Lambda^{-1}\Big[x_{i+1} - G_i K_{com,i} \widehat{d}_i - G_{d,i} d_i\Big]. \tag{5.6}$$

Com isso, considerando que a estimativa $\hat{d}$ irá convergir para o valor do distúrbio real, é possível encontrar um ganho K_{com} que elimine o distúrbio na saída do sistema y.

$$K_{com,i} = -\left[C\Lambda^{-1} G_i\right]^{-1} C\Lambda^{-1} G_{d,i}. \tag{5.7}$$

De acordo com Benevides *et al.* (2021) e Paiva (2019), é possível que sistemas de tempo real controlados remotamente sejam afetados por atrasos de comunicação, acarretando a perda de desempenho. Neste caso, pode-se antecipar a atuação do compensador de distúrbio, utilizando sua derivada. Dessa forma, é gerada uma lei de compensação de controle proporcional-derivativa, como descrita pela Equação (5.8).

$$u^{pd}_{dist,i} = \alpha K_{com,i} \widehat{d}_i + \lambda\, \widehat{d}_{i+1}, \tag{5.8}$$

em que $|\alpha| < 1$ and λ são escalares, definidos como parâmetros de projeto. Com isso, a lei de controle composta (Equação (5.3)) é reescrita da seguinte maneira:

$$u_{com,i} = u_{rlqr,i} + u^{pd}_{dist,i}. \tag{5.9}$$

Vale ressaltar que esta metodologia descrita para encontrar o ganho de compensação para o RLQR foi utilizada para todos os outros controladores implementados. O funcionamento geral da arquitetura 2 proposta pode ser resumido pelo algoritmo 2, em que utiliza o controlador RLQR como exemplo.

Algoritmo 2 Rede neural como estimador de distúrbios

Entrada: Trajetória desejada q^d

Saída: Vetor de velocidade ν

1: Inicialização do RLQR: considere (4.13), (4.14), and (4.16) com F_i, G_i, E_{F_i}, E_{G_i}, $Q_i \succ 0$, e $R_i \succ 0$ conhecidas para todo instante i.
2: **Enquanto** receber a trajetória desejada **faça**
3: Guarda q^d e q
4: Calcula $\dot{q}^d$ e $\dot{q}$
5: Computa $\tilde{q}$ e $\dot{\tilde{q}}$
6: Calcula o ganho RLQR com a Eq. 4.17
7: $u_{rlqr,i} = K_{rlqr,i} x_i$
8: Estima d com a DNN
9: Computa forma filtrada de d
10: Calcula $\dot{d}$
11: $u^{pd}_{dist,i} = \alpha K_{com,i} \widehat{d}_i + \lambda \, \widehat{d}_{i+1}$
12: $u_{com,i} = u_{rlqr,i} + u^{pd}_{dist,i}$
13: Usa u_{com} em (2.12) para gerar ν
14: **Fim Enquanto**

6 METODOLOGIA

Este capítulo apresenta a metodologia e as ferramentas utilizadas para desenvolvimento deste trabalho. Primeiramente, a seção 6.1 apresenta a plataforma experimental e a forma de comunicação com o quadricóptero. Em seguida, a seção 6.2 apresenta as formas de estimativa de posição e orientação utilizadas para realimentar o sistema de controle. A seção 6.3 detalha o treinamento das redes neurais implementadas. Já na seção 6.4, é apresentada a trajetória desejada enviada ao Bebop 2.0 durante os experimentos. Na seção 6.5 são descritos os parâmetros dos controladores utilizados tanto nas simulações quanto nos experimentos práticos. Por último, a seção 6.6 apresenta os índices utilizados para avaliar, de forma quantitativa, o desempenho das arquiteturas propostas e dos controladores implementados.

Vale ressaltar que para desenvolvimento de todo o *framework* proposto, foi utilizado um *notebook* Dell Inspiron 14 7460 com sistema operacional Ubuntu na versão 16.04 LTS. O *hardware* utilizado possui processador Intel Core i7-7500U de 2 núcleos com velocidade de até 3,5GHz, memória RAM de 16 GB DDR4 e placa de vídeo Nvidia Geforce 940MX.

6.1 Plataforma Experimental

O quadricóptero escolhido para experimentos práticos foi o Parrot^{TM} Bebop 2.0, cujas principais características foram apresentadas no Capítulo 2. Além de ser um quadricóptero de baixo custo, o Bebop 2.0 foi escolhido, sobretudo, pela facilidade de comunicação com a plataforma *Robot Operating System* (ROS). Essa facilidade surgiu devido à criação de um pacote ROS, desenvolvido pelos cientistas do *Autonomy Lab* da Universidade Simon Fraser, baseado no *kit* de desenvolvimento de software da Parrot^{TM}. Este pacote, disponibilizado em código aberto, é intitulado de `bebop_autonomy`[1] e é responsável pela comunicação com a aeronave. Ademais, é possível receber e enviar informações para o Bebop 2.0 utilizando diversas fontes, como *joystick*, *smartphone* e até mesmo por um teclado de computador.

O ambiente experimental montado é apresentado na Figura 10, em que é possível observar o quadricóptero Bebop 2.0 e o sistema gerador de distúrbio de vento no espaço de testes disponível no Centro de Robótica da Universidade de São Paulo (CRob). Os detalhes a respeito dos materiais e métodos utilizados serão apresentados nas seções seguintes deste capítulo.

[1] http://wiki.ros.org/bebopautonomy - 2015.

Figura 10 – Bebop 2.0 pairando no ar em ambiente experimental montado.

Fonte: Elaborada pelo autor.

6.1.1 Simulador *Parrot-Sphinx*

Com o intuito, inicialmente, de atender as necessidades que surgiam durante o desenvolvimento de software, os engenheiros da Parrot™ desenvolveram um simulador realista para o Bebop 2.0, o *Parrot-Sphinx*. Atualmente, o simulador é disponibilizado em código aberto, baseado no simulador Gazebo do ROS. Este simulador permite que o usuário controle o Bebop 2.0 em diversos cenários com uma boa fidelidade da realidade, sendo capaz de simular não só a cinemática do veículo como também a sua dinâmica. Além disso, permite a visualização dos dados de voos e a configuração de características físicas do ambiente, como velocidade linear do vento e parâmetros da atmosfera em geral.

Figura 11 – Bebop 2.0 pairando no ar em cenário do *Parrot-Sphinx*.

Fonte: Elaborada pelo autor.

O simulador *Parrot-Sphinx* foi utilizado neste trabalho para realização dos experimentos e validação dos algoritmos propostos. Dessa forma, foi possível avaliar e ajustar

as arquiteturas de controle desenvolvidas de modo a obter resultados satisfatórios. A Figura 11 apresenta o quadricóptero Bebop 2.0 pairando no ar em um cenário simulado do *Parrot-Sphinx*.

6.1.2 Plataforma ROS

O ROS (QUIGLEY *et al.*, 2009) é uma plataforma de código aberto dotada de uma série de ferramentas que possibilita a programação de diferentes tipos de robôs. A partir do seu surgimento, a necessidade que os programadores tinham de criar *softwares* específicos para programar sistemas robóticos foi extinta. Com o ROS, passou-se a ter um sistema completo, com capacidade para desenvolver aplicações para os mais variados dispositivos robóticos. Suas capabilidades possibilitaram o surgimento de um grande ecossistema formado por uma vasta comunidade de programadores, o que viabilizou o surgimento de diversos tutoriais, pacotes e algoritmos, facilitando ainda mais o acesso inicial ao ROS.

O funcionamento do ROS baseia-se na integração entre diferentes nós que carregam informações sobre o sistema robótico. Esses nós podem representar, por exemplo, sensores ou atuadores, cujas mensagens geradas por cada um deles são publicadas em tópicos acessíveis a outros nós. Dessa maneira, pode-se ler as informações recebidas por sensores e enviar comandos para os atuadores com facilidade. Uma das principais vantagens do ROS é a possibilidade de utilização de diversas linguagens de programação, como *Python*, C++, Java e LISP. Neste trabalho, esta vantagem possibilitou a implementação dos controladores em C++ e o desenvolvimento das redes neurais em *Python*, utilizando bibliotecas específicas.

6.1.3 Geração de Distúrbio de Vento

A seguir são apresentadas as fontes de distúrbio de vento utilizadas nos experimentos. Primeiramente, é apresentado o modelo do distúrbio utilizado no ambiente simulado. Em seguida, é apresentada a fonte de distúrbio de vento utilizada nos experimentos práticos.

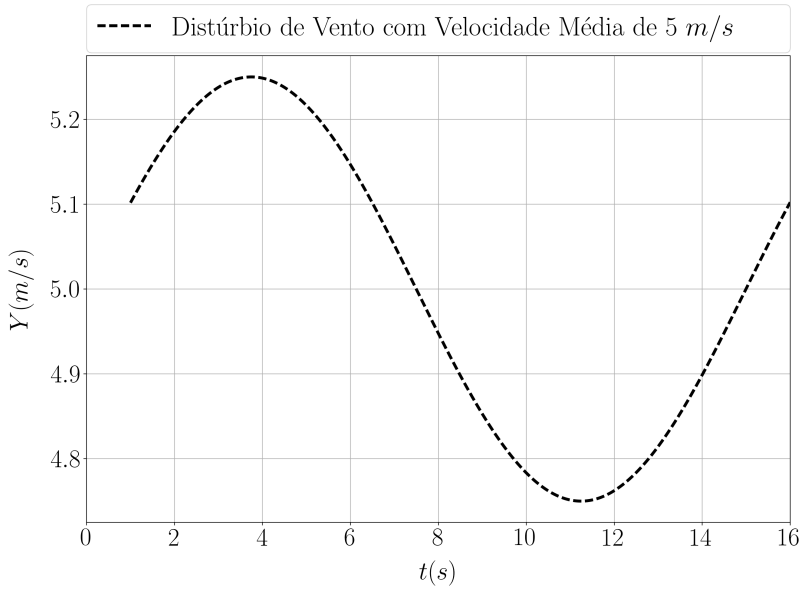
6.1.3.1 Distúrbio de Vento para o *Parrot-Sphinx*

Para os casos em que o quadricóptero é afetado por distúrbios de vento, foi utilizado o recurso de simulação de vento disponível no próprio simulador. A Equação (6.1) apresenta o distúrbio utilizado e a Figura 12 ilustra o formato da onda para um distúrbio configurado com velocidade média de 5 m/s.

$$Wind_y = W_y\,(1 + 0.05 \cdot sen(2 \cdot \pi \cdot \frac{t}{15})), \tag{6.1}$$

em que W_y representa a velocidade média do vento e t é o tempo em segundos.

Figura 12 – Distúrbio de vento utilizado no ambiente simulado.



Fonte: Elaborada pelo autor.

6.1.3.2 Distúrbio de Vento para Experimentos Práticos

Para os experimentos práticos, foi montado um ambiente experimental a partir da criação de um gerador de distúrbio de vento. Para isto, foi utilizado um motor sem escovas (*brushless*) acionado por um ESC (*Electronic Speed Controller*) através de sinal de PWM (*Pulse Width Modulation*) e alimentado por uma bateria LiPo. Os detalhes a respeito dos componentes utilizados são apresentados na Tabela 5.

Tabela 5 – Materiais utilizados para geração de distúrbio de vento.

Material	Descrição
Motor *brushless*	E-flite Delta-V 15 do tipo *fan*, 69mm, 3600Kv
Bateria	LiPo de 3 células (3s - 11,1 V), 25C e 3300 mAh
ESC	Hobbywing Skywalker 60 A UBEC, 2-6s LiPo
Gerador de PWM	Placa ESP8266, Sinal do PWM de 50 Hz, resolução de 10 bits
Tripé	Universal com altura regulável

A montagem do sistema é descrita na Figura 13, em que o sistema completo é apresentado na Figura 13(a) e o motor utilizado para gerar o distúrbio de vento é apresentado na Figura 13(b). A altura do sistema é de aproximadamente $1,0$ m, causando distúrbio no quadricóptero ao longo de toda a trajetória.

Com o intuito de verificar a velocidade de vento que afeta o quadricóptero durante o seguimento de trajetória, foi utilizado um anemômetro portátil da marca Incoterm®, modelo TAN100. Desse modo, foi possível obter a velocidade máxima e mínima de vento gerada pelo motor. Além disso, foi sintetizado um mapa de vento com as respectivas velocidades

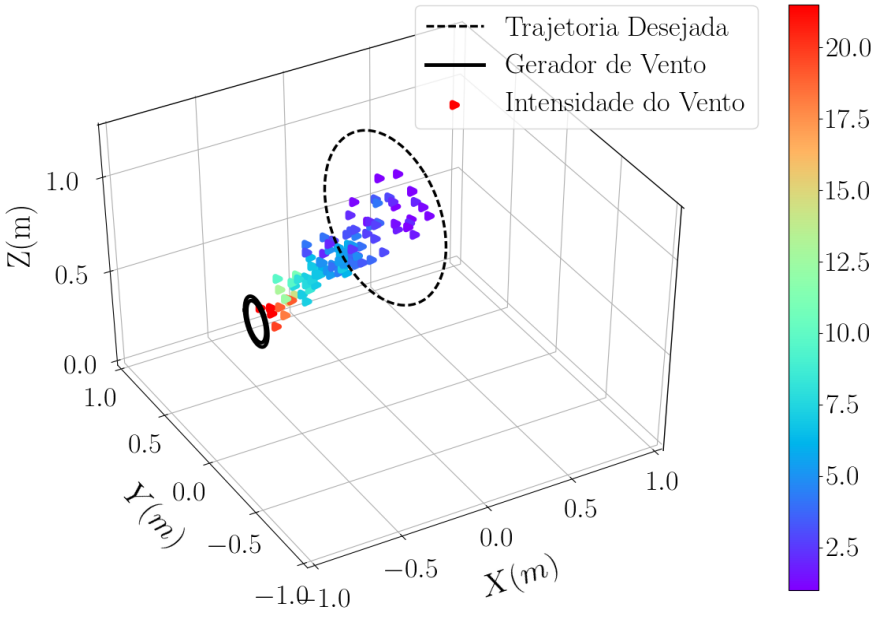
Figura 13 – Montagem do gerador de distúrbio de vento para experimentos práticos.

(a) Montagem completa do sistema. (b) Motor gerador de vento.

Fonte: Elaborado pelo autor.

geradas pelo sistema de distúrbio montado. Esse mapa é apresentado na Figura 14, em que é possível observar as diferentes velocidades de vento que afeta o quadricóptero em diferentes pontos da trajetória. Através dos pontos de intensidade de vento coletados, foi possível visualizar uma velocidade de vento máxima de aproximadamente $20,5$ m/s e uma mínima de $1,0$ m/s, a depender da proximidade do anemômetro com o gerador de distúrbio de vento.

Figura 14 – Mapa do distúrbio de vento que afeta o quadricóptero durante o seguimento de trajetória.



Fonte: Elaborada pelo autor.

Vale ressaltar que para criação do mapa que ilustra o padrão de distúrbio de vento, foi utilizado o sistema de captura de movimentos Vicon®. Este sistema, também utilizado para estimativa de posição e orientação do quadricóptero, será detalhado na seção seguinte.

6.2 Estimativa de Posição e Orientação

Com o objetivo de realimentar o sistema de controle durante o rastreamento de trajetória, faz-se necessária a obtenção da posição e orientação da aeronave a cada instante de tempo. A seguir, serão discutidas as duas formas utilizadas neste trabalho para obtenção da posição e orientação do quadricóptero.

6.2.1 Sistema de Odometria

As informações provenientes do sistema de odometria do Bebop 2.0 foram utilizadas para os experimentos com o simulador *Parrot-Sphinx*. Neste caso, os dados de posição e orientação do quadricóptero são fornecidos por um tópico do ROS intitulado de `/bebop/odom`, que simula as informações fornecidas pela Unidade de Medição Inercial (*IMU - Inertial Measurement Unit*). A IMU é uma tecnologia essencial para navegação autônoma de veículos aéreos. Sendo composta por giroscópios e acelerômetros, possui a capacidade de fornecer posição, velocidade e orientação da aeronave em tempo real. Esses dados são usados como realimentação para o controlador projetado, que gera uma lei de controle capaz de minimizar o erro de posição entre a trajetória desejada e atual do quadricóptero.

6.2.2 Sistema de Captura de Movimentos *Vicon®*

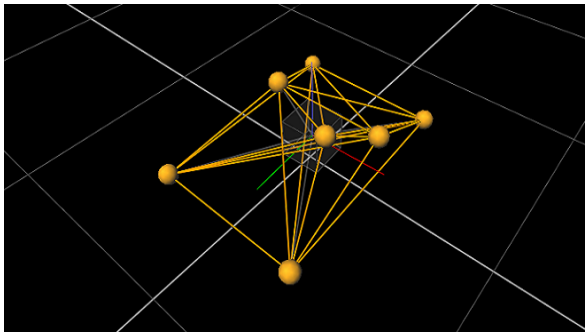
Além da utilização do sistema de odometria do Bebop 2.0, foi utilizado um sistema de captura de movimento de alta precisão Vicon® (VICON, 2012). Esse tipo de sistema é utilizado para estimar a posição e orientação de pessoas ou objetos utilizando a tecnologia de emissão de luzes infravermelhas. Atualmente, podem ser utilizados em diversas aplicações, entre elas, destacam-se: desenvolvimento de sistemas robóticos, aplicações médicas, desenvolvimento de jogos, entretenimento, entre outras (AL HABSI *et al.*, 2016).

Em geral, o sistema de captura de movimentos Vicon® é composto por quatro câmeras específicas e seu funcionamento se dá através da instalação de marcadores reflexivos no corpo do objeto que se quer rastrear. Diante disso, o sistema emite luz infravermelha que é refletida pelos marcadores presentes no objeto e, em seguida, é detectada pelas câmeras. Neste trabalho, as câmeras Vicon® são responsáveis por gerar um sistema de referência para o Bebop 2.0. Os dados de posição e orientação são enviados para um software específico que, logo depois, envia ao *notebook* que contém o sistema de controle. A informação é transferida a uma taxa de 100 Hz, utilizando protocolo de comunicação Wi-Fi.

O Centro de Robótica da Universidade de São Paulo (CRob), localizado no campus de São Carlos, dispõe de um ambiente destinado à realização de experimentos com veículos aéreos de pequeno porte. O ambiente contém quatro câmeras Vicon® instaladas nos cantos superiores e foi utilizado para realização dos experimentos práticos com o quadricóptero. A Figura 15 apresenta os principais componentes utilizados para estimativa de posição e orientação com o sistema de captura de movimentos Vicon®.

Figura 15 – Aparato experimental para estimativa de posição e orientação com sistema Vicon®. (a) Marcadores reflexivos instalados no corpo do Bebop 2.0. (b) Visualização do Bebop 2.0 pelo *software* da Vicon®. (c) Uma das quatro câmeras Vicon® utilizadas nos experimentos. (d) Bebop pairando no ar com duas das câmeras em funcionamento no canto superior.

(a)



(b)

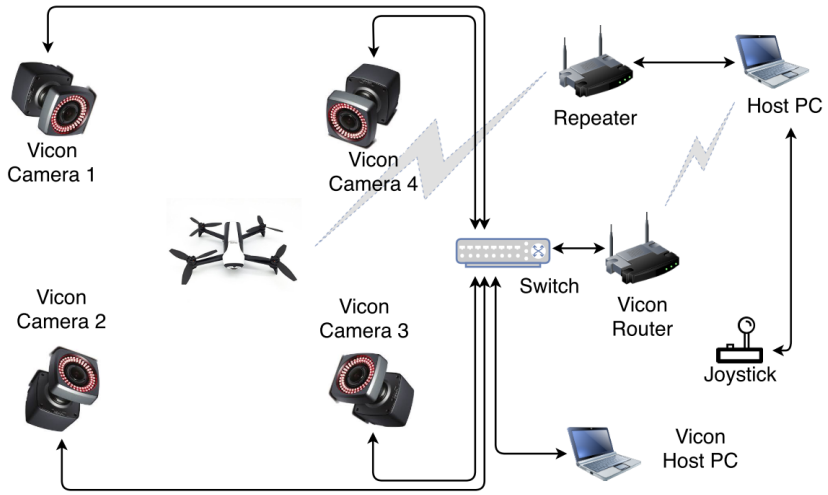
(c)

(d)

Fonte: Elaborada pelo autor.

O esquemático completo de funcionamento, utilizando o sistema de captura de movimentos Vicon® para realização dos experimentos práticos, é apresentado na Figura 16. Esta figura apresenta a trajetória que o sinal proveniente das câmeras realiza até alcançar o *notebook* que contém os controladores implementados. O *joystick*, presente na figura, é responsável pela decolagem e aterrissagem, além de funcionar como uma ferramenta de segurança, possibilitando assumir o controle da aeronave quando necessário.

Figura 16 – Esquemático de funcionamento para experimentos práticos.



Fonte: Retirado de Benevides *et al.* (2019a).

6.3 Treinamento e Avaliação das Redes Neurais

Nesta seção são abordados os detalhes referentes ao treinamento e avaliação das redes neurais desenvolvidas. Vale ressaltar que, para desenvolver as DNNs apresentadas, foi utilizada a linguagem *Python* através do *framework Keras* com *backend* em *Tensorflow*.

6.3.1 Rede Neural Como Gerador de Referência

A arquitetura da DNN que atua como referência para os controladores pode ser definida, de acordo com a literatura, como uma rede de múltiplas camadas. Neste caso, a arquitetura da rede é composta por uma camada de entrada; três camadas ocultas, contendo 80, 50 e 30 neurônios, respectivamente; e uma camada de saída. O número de camadas e de neurônios presentes em cada uma delas foram escolhidos de forma empírica. Para tanto, uma série de treinamentos foram realizados em que buscou-se ajustar e otimizar a rede com base no desempenho apresentado para a tarefa designada. Dessa maneira, foi possível alcançar a arquitetura apresentada na Figura 8.

Para treinamento da rede, foi utilizada a técnica de aprendizado supervisionado, em que um conjunto de amostras rotuladas foi fornecido. Neste caso, para criação do *dataset*, foram realizados voos com o quadricóptero, utilizando cada um dos quatro controladores implementados. Os voos consistiram em realizar o seguimento de trajetória circular no plano XYZ com o raio variando de $0,2$ m à $1,0$ m. Dessa forma, foram treinadas quatro DNNs, em que, após o treinamento, cada uma atuou de forma independente em conjunto com o respectivo controlador utilizado no treinamento (RLQR, LQR, PID e LR). A partir dos voos realizados, foi possível obter as amostras de dados com informações tanto da trajetória atual quanto da desejada do quadricóptero a cada instante de tempo. No total foram obtidas 10.000 amostras para compor o *dataset*, sendo que, deste montante, 90%

foram utilizadas para a etapa de treinamento e 10% para a etapa de testes.

Após a obtenção das amostras, deu-se seguimento para criação do conjunto de treinamento da rede, que foi formado por $\{(\bar{q}_i, \bar{q}_{i+1}),\ \bar{q}_i^d\}$, em que $\bar{q}_i$ e $\bar{q}_{i+1}$ representam a posição do quadricóptero no instante i e $(i+1)$, respectivamente; e $\bar{q}^d$ representa seu correspondente dado de saída (rótulo). Observe que $\{(\bar{q}_i, \bar{q}_{i+1})\} \rightarrow \bar{q}_i^d$, utilizado no treinamento, é um mapeamento aproximado de $\{(\bar{q}_i, \bar{q}_i^d)\} \rightarrow \bar{q}_i^r$ (Figura 8). Desse modo, quando a trajetória atual do quadricóptero for igual à desejada, a rede entende que o rastreamento está sendo perfeito e $\bar{q}_i^r$ será igual a $\bar{q}_i^d$. Contudo, quando a trajetória atual do quadricóptero for diferente da trajetória desejada, a rede fornece um sinal de referência com base em experiências passadas, com o intuito de fazer com que $\bar{q}_i$ convirja para $\bar{q}_i^d$. Os parâmetros utilizados para treinamento são apresentados na Tabela 6.

Tabela 6 – Parâmetros de treinamento da rede neural (Arquitetura 1).

Funções de ativação	ReLU
Função de perda	Erro quadrático médio (EQM)
Algoritmo de otimização	*RMSprop*
Taxa de aprendizagem	0,008
Desligamento (*dropout*)	25%
Inicialização dos pesos	*Xavier* (GLOROT; BENGIO, 2010))
Inicialização dos *biases*	0
Número máximo de épocas	60
Tamanho do *batch*	128

A função de ativação ReLU foi escolhida, sobretudo, por ser uma das mais eficientes no treinamento de redes neurais profundas. Uma de suas grandes vantagens é a capacidade de evitar o desaparecimento do gradiente. Em outras palavras, evita que o gradiente assuma valores muito próximos de zero, situação que dificultaria o ajuste dos pesos sinápticos da rede. Além disso, após a comparação com outras funções de ativação, verificou-se um desempenho melhor da rede com a função ReLU.

Com relação ao algoritmo de otimização escolhido, além do *RMSprop* utilizado, foram testados o gradiente descendente estocástico e o *adaptive moment estimation* (Adam). Após uma análise comparativa entre os três, para a tarefa desempenhada pela rede neste trabalho, notou-se um melhor desempenho após o treinamento com o algoritmo *RMSprop*. Já com relação a técnica de regularização, foi observado um melhor desempenho da rede com a aplicação de *dropout* de 25%. Ademais, os parâmetros restantes utilizados no treinamento da rede foram ajustados de forma empírica, em que buscou-se encontrar o melhor desempenho da rede sem que ocorresse sobreajuste dos pesos sinápticos.

Com os parâmetros de treinamento definidos (Tabela 6), a rede levou cerca de 4.000 iterações para convergir. Para avaliação, a métrica utilizada foi o erro absoluto

médio (EAM), em que se buscou calcular o valor absoluto da diferença entre a predição da rede e o conjunto de amostras rotuladas. Em seguida, após o treinamento, a DNN foi avaliada com o *dataset* de teste e a partir de voos experimentais utilizando o quadricóptero, apresentando desempenho satisfatório para a tarefa tratada neste trabalho.

6.3.2 Rede Neural Como Estimador de Distúrbios

Assim como a DNN utilizada como referência para os controladores, a DNN para estimativa de distúrbio possui a arquitetura de uma rede de múltiplas camadas. Neste caso, além das camadas de entrada e saída, a rede é formada por duas camadas ocultas com 50 e 25 neurônios, respectivamente. A topologia da rede, apresentada na Figura 9, foi obtida de forma empírica, considerando a tarefa de estimativa de distúrbio.

O método de aprendizado supervisionado também foi utilizado para treinamento da rede. Com intuito de fazer com que a DNN seja capaz de fornecer o distúrbio de vento em *Newtons* que afeta o quadricóptero, o erro de posição foi utilizado como entrada da rede. No total foram obtidas 2000 amostras para compor o *dataset*, sendo que 90% foram utilizadas para a etapa de treinamento e 10% para a etapa de testes.

Para criação do *dataset*, foi utilizado como auxílio o simulador *Parrot-Sphinx* com suporte do pacote *BebopS*, que fornece a possibilidade de aplicar o distúrbio de vento no quadricóptero no formato de uma força em *Newtons*. Com isso, o quadricóptero foi mantido pairando no ar e um distúrbio de vento com magnitudes distintas foi aplicado à aeronave para computar o erro de posição. Primeiramente, foi aplicado o distúrbio com magnitudes que variam entre $0,1$ N a $0,5$ N ao longo do eixo X do quadricóptero. Em seguida, o mesmo procedimento foi realizado para o eixo Y da aeronave. Dessa forma, foi possível obter o erro de posição do quadricóptero nos eixos X e Y a partir da intensidade do distúrbio de vento que o afeta. Para este caso, a Tabela 7 apresenta os parâmetros de treinamento da rede utilizados.

Tabela 7 – Parâmetros de treinamento da rede neural (Arquitetura 2).

Funções de ativação	ReLU
Função de perda	Erro quadrático médio (EQM)
Algoritmo de otimização	*Adam*
Taxa de aprendizagem	0,001
Inicialização dos pesos	*Xavier*(GLOROT; BENGIO, 2010))
Inicialização dos *biases*	0
Número máximo de épocas	30
Tamanho do *batch*	32

Com relação ao algoritmo de otimização escolhido, para este caso, o *adaptive moment estimation* (Adam) demonstrou um melhor desempenho, fornecendo respostas

melhores que o gradiente descendente estocástico e o *RMSProp* testados. Já em relação à função de ativação, a função ReLU demonstrou, também, melhores respostas para a tarefa desempenhada pela rede. Assim como para a primeira rede, os demais parâmetros utilizados no treinamento foram ajustados de forma empírica, com objetivo de encontrar a melhor topologia da rede.

Durante o treinamento, verificou-se que a rede levou cerca de 1.600 iterações para convergir. O erro absoluto médio (EAM) foi utilizado para calcular a diferença entre a predição da rede e o conjunto de amostras rotuladas. Após o treinamento, a DNN foi avaliada com o *dataset* de teste e com uma série de voos experimentais, utilizando o quadricóptero tanto no ambiente simulado quanto no ambiente real.

6.4 Trajetória Desejada

A trajetória desejada que foi utilizada nos experimentos possui um formato circular no plano XYZ, com raio de $0,3$ m. O envio da trajetória para o quadricóptero é dado através de um nó do ROS, que é responsável por publicar no tópico `/bebop/waypoint` tanto a posição quanto a velocidade desejada a uma frequência de 50 Hz. Esse tópico, que disponibiliza $q^d = [q_x^d, q_y^d, q_z^d, q_\psi^d]^T$ e $\dot{q}^d$, é acessível tanto à DNN quanto aos controladores, sendo:

$$q^d = \begin{bmatrix} \varphi\cos\omega t & \varphi\sin\omega t & \frac{\varphi}{2}\sin\omega t & 0 \end{bmatrix}^T, \tag{6.2}$$

$$\dot{q}^d = \begin{bmatrix} -\omega\varphi\sin\omega t & \omega\varphi\cos\omega t & \frac{\omega\varphi}{2}\cos\omega t & 0 \end{bmatrix}^T, \tag{6.3}$$

em que $\omega = 0,5$ rad/s é a velocidade angular, dada pela razão entre a velocidade média $v = 0,15$ m/s e o raio $\varphi = 0,3$ m. Para cada experimento, o quadricóptero parte da origem $(x = 0, y = 0, z = 0)$ e realiza o trajeto proposto.

6.5 Parâmetros dos Controladores

Esta seção apresenta os parâmetros utilizados para os controladores tanto na simulação com o *Parrot-Sphinx* quanto nos experimentos práticos.

6.5.1 Parâmetros para o Regulador Linear Quadrático (LQR)

Para simulação e para os experimentos práticos utilizando o LQR, as matrizes de ponderação escolhidas após a análise de diversos experimentos foram:

$$Q = \begin{bmatrix} 1.2 \cdot I_{2\times2} & 0 & 0 \\ 0 & I_{4\times4} & 0 \\ 0 & 0 & 5 \cdot I_{2\times2} \end{bmatrix}, \qquad R = 0,15 \cdot I_{4\times4} \qquad P = I_{8\times8}.$$

Para o compensador de distúrbios com o LQR no simulador *Parrot-Sphinx* foram utilizados os seguintes parâmetros:

$$\alpha = 0,5 \text{ e } \lambda = 0,002.$$

Já para os experimentos práticos utilizando o LQR, temos que os parâmetros do compensador de distúrbios foram os seguintes:

$$\alpha = 0,4 \text{ e } \lambda = 0,001.$$

6.5.2 Parâmetros para o Regulador Linear Quadrático Robusto (RLQR)

No caso do controlador RLQR, as matrizes de ponderação e de incertezas utilizadas tanto no simulador *Parrot-Sphinx*, quanto nos experimentos reais foram as seguintes:

$$Q = \begin{bmatrix} 1.2 \cdot I_{2\times2} & 0 & 0 \\ 0 & I_{4\times4} & 0 \\ 0 & 0 & 5 \cdot I_{2\times2} \end{bmatrix}, \qquad R = 0,15 \cdot I_{4\times4}, \qquad P = I_{8\times8},$$

$$H = \begin{bmatrix} I_{4\times4} & 0 \\ 0 & 0_{4\times4} \end{bmatrix}, \qquad E_G = 0,01875 \cdot I_{4\times4}, \qquad \mu = 1 \cdot 10^{15},$$

$$E_F = \begin{bmatrix} 0,0354 & 0 & 0 & 0 & 0,0285 & 0 & 0 & 0 \\ 0 & 0,0354 & 0 & 0 & 0 & 0,0285 & 0 & 0 \\ 0 & 0 & 0,0295 & 0 & 0 & 0 & 0,1425 & 0 \\ 0 & 0 & 0 & 0,0295 & 0 & 0 & 0 & 0,1425 \end{bmatrix}.$$

Para o compensador de distúrbios com o RLQR no ambiente simulado, foram utilizados os seguintes parâmetros:

$$\alpha = 0,12 \text{ e } \lambda = 0,001.$$

Em relação aos parâmetros do compensador de distúrbios utilizados nos experimentos práticos, temos:

$$\alpha = 0,25 \text{ e } \lambda = 0,01.$$

6.5.3 Parâmetros para o Controlador Proporcional Integral Derivativo (PID)

Para a simulação com o controlador PID, as matrizes de ganho foram escolhidas heuristicamente, após uma série de experimentos As matrizes escolhidas foram as seguintes:

$$K_p = \begin{bmatrix} 0,35 & 0 & 0 & 0 \\ 0 & 0,35 & 0 & 0 \\ 0 & 0 & 3,0 & 0 \\ 0 & 0 & 0 & 1,0 \end{bmatrix}, \qquad K_d = \begin{bmatrix} 0,4 & 0 & 0 & 0 \\ 0 & 0,4 & 0 & 0 \\ 0 & 0 & 0,4 & 0 \\ 0 & 0 & 0 & 1,0 \end{bmatrix},$$

$$K_i = \begin{bmatrix} 0,001 & 0 & 0 & 0 \\ 0 & 0,001 & 0 & 0 \\ 0 & 0 & 0,01 & 0 \\ 0 & 0 & 0 & 0,01 \end{bmatrix}.$$

Para o compensador de distúrbios com o PID no ambiente simulado, foram utilizados os seguintes parâmetros:

$$\alpha = 0,06 \text{ e } \lambda = 0,00025.$$

Para os experimentos reais, os parâmetros do compensador de distúrbios foram:

$$\alpha = 0,06 \text{ e } \lambda = 0,00025.$$

6.5.4 Parâmetros para o Controlador Linearização por Realimentação (LR)

$$K_1 = \begin{bmatrix} 1,25 & 0 & 0 & 0 \\ 0 & 1,25 & 0 & 0 \\ 0 & 0 & 3,5 & 0 \\ 0 & 0 & 0 & 1,0 \end{bmatrix}, \qquad K_2 = \begin{bmatrix} 1,1 & 0 & 0 & 0 \\ 0 & 1,1 & 0 & 0 \\ 0 & 0 & 2,0 & 0 \\ 0 & 0 & 0 & 2,0 \end{bmatrix}.$$

Para o compensador de distúrbios com o LR no ambiente simulado, foram utilizados os seguintes parâmetros:

$$\alpha = 0,04 \text{ e } \lambda = 0,0015.$$

Já para o compensador de distúrbios utilizados nos experimentos práticos, foram utilizados os seguintes parâmetros:

$$\alpha = 0,05 \text{ e } \lambda = 0,001.$$

6.5.5 Parâmetros para o Filtro de Kalman

O filtro de Kalman, responsável por fornecer as formas filtradas dos estados e dos distúrbios do sistema, foi configurado com as seguintes matrizes de ponderação:

$$Q = \begin{bmatrix} I_{4\times4} & 0 & 0 \\ 0 & 10 \cdot I_{4\times4} & 0 \\ 0 & 0 & 10 \cdot I_{4\times4} \end{bmatrix}, \qquad R = 1 \cdot I_{8\times8}.$$

6.6 Índices de desempenho

Com o intuito de analisar estatisticamente o desempenho das arquiteturas propostas para cada um dos controladores implementados, foi utilizada a norma ℓ_1 e o erro médio absoluto para as variáveis de posição controladas. Vale ressaltar que, para uma maior precisão dos dados, os cálculos foram realizados com a média de 5 experimentos para cada um dos controladores implementados em cada uma das arquiteturas. Dessa forma, as métricas abaixo correspondem a média dos experimentos realizados, em que:

$$E(i) = \frac{1}{5}\sum_{j=1}^{5} \|q_i - q_i^d\|, \tag{6.4}$$

representa a norma ℓ_1, e

$$\overline{E}(i) = \frac{1}{N}\sum_{i=1}^{N} E(i), \tag{6.5}$$

representa o erro médio absoluto, em que q_i e q_i^d são os vetores de posição e posição desejada, respectivamente; i corresponde ao valor de uma iteração com N sendo a quantidade máxima de iterações; j representa o número atual do experimento para um total de 5.

Para calcular o percentual de melhoria das arquiteturas propostas em relação aos controladores isolados, foi utilizado o seguinte índice percentual:

$$I^{\%} = \left(1 - \frac{\overline{E}}{\overline{E}_{ref}}\right) \times 100\,, \tag{6.6}$$

em que $\overline{E}$ é o erro médio absoluto para cada controlador em cada uma das arquiteturas propostas e $\overline{E}_{ref}$ é o erro médio absoluto para os controladores isolados.

7 RESULTADOS E DISCUSSÕES

O presente capítulo está dividido em duas seções: a primeira aborda os resultados referentes à implementação das arquiteturas de controle propostas utilizando o Bebop 2.0 no simulador *Parrot-Sphinx*; a segunda seção apresenta os resultados práticos com o Bebop 2.0 no ambiente experimental montado. Em cada uma das seções, são apresentados os resultados com e sem a aplicação de distúrbio de vento no quadricóptero. Com intuito de obter uma melhor confiabilidade acerca de cada uma das arquiteturas apresentadas, os resultados abaixo representam a média de 5 experimentos para cada um dos controladores utilizados.

7.1 Resultados com o Simulador Parrot-Sphinx

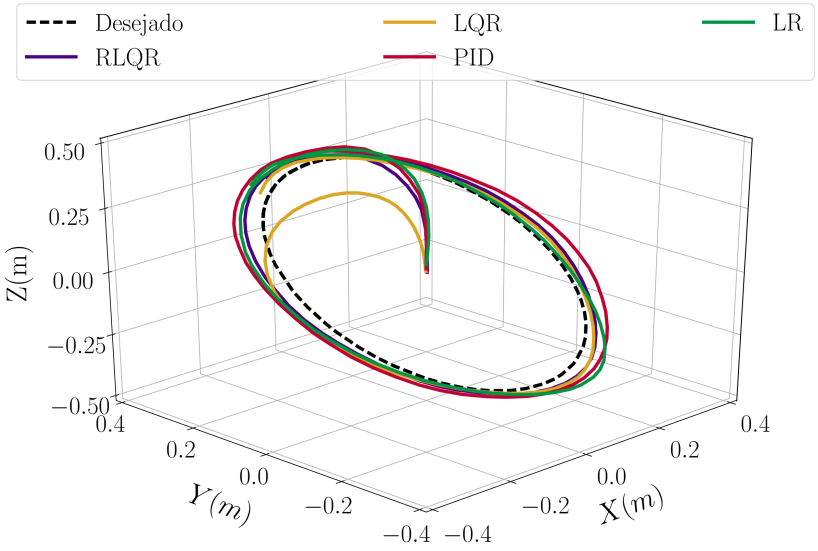
Nesta seção, são apresentados os resultados de voos utilizando o simulador *Parrot-Sphinx* (subseção 6.1.1). Os experimentos correspondem à realização do rastreamento de trajetória (seção 6.4) com o Bebop 2.0, utilizando a metodologia descrita no Capítulo 6. Inicialmente, são apresentados os resultados utilizando os controladores isolados, ou seja, sem auxílio da rede neural. Em seguida, são expostos os resultados utilizando a primeira arquitetura de controle proposta. Por fim, são apresentados os resultados utilizando a arquitetura DOBC composta pela rede neural profunda e filtro de Kalman. Inicialmente foram avaliados os desempenhos para cada caso sem a aplicação de distúrbio de vento e, em seguida, são apresentados os resultados quando o quadricóptero está sujeito a distúrbios de vento laterais constantes com velocidade média de 5 m/s.

7.1.1 Experimentos sem Aplicação de Distúrbios de Vento

Os resultados apresentados nesta seção compreendem o conjunto de experimentos realizados para condições normais de voo, ou seja, quando o quadricóptero não é afetado por uma fonte de distúrbio de vento externa. Neste caso, são apresentadas as figuras e os índices de desempenho para os controladores isolados e para as arquiteturas de controle propostas. Primeiramente, a fim de avaliar o desempenho dos controladores, os resultados são discutidos para cada uma das arquiteturas propostas. Em seguida, é feita uma comparação entre as arquiteturas desenvolvidas.

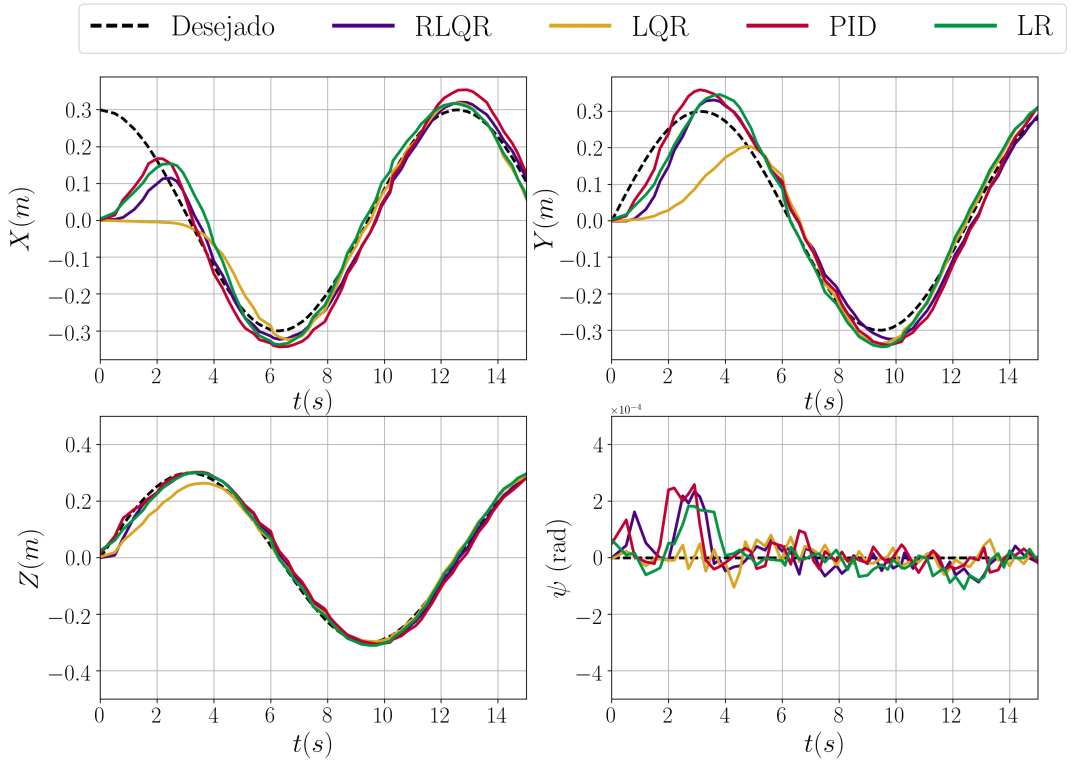
As Figuras 17, 18, 19 apresentam os resultados de voos utilizando os controladores isolados, em que a vista 3D do rastreamento de trajetória é apresentada na Figura 17 e as variáveis controladas (x, y, z, ψ) são apresentadas na Figura 18. A evolução do erro para este caso é exposta na Figura 19, sendo complementada pelos índices de desempenho apresentados na Tabela 8.

Figura 17 – Rastreamento de trajetória circular em 3D para cada controlador implementado.



Fonte: Elaborada pelo autor.

Figura 18 – Variáveis de posição e orientação controladas.

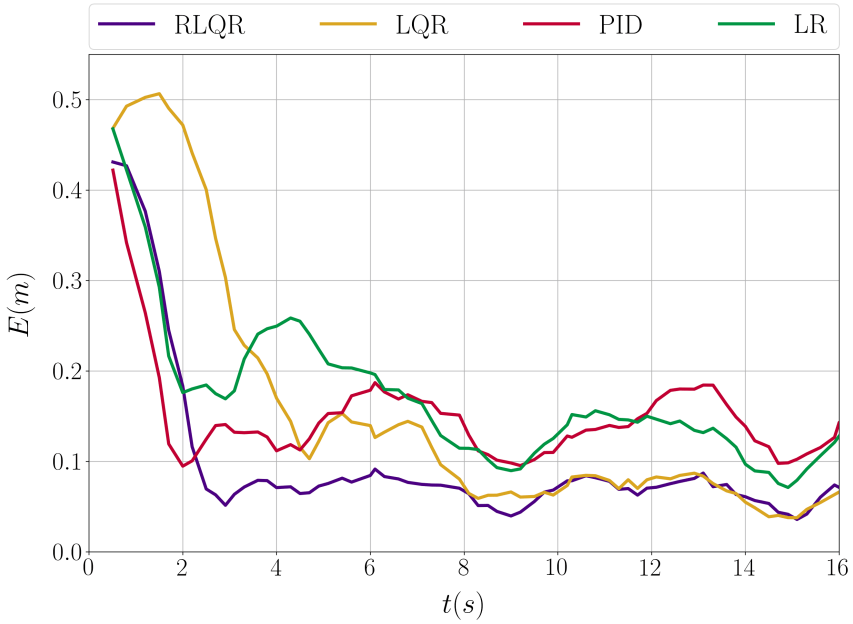


Fonte: Elaborada pelo autor.

Para este caso, é possível perceber que, entre os controladores implementados, o RLQR obteve o menor erro de posição ($\overline{E}$) na tarefa de rastreamento de trajetória. Além disso, apresenta o menor erro máximo após decorridos $3s$ de voo, tempo médio que o quadricóptero leva para convergir para a trajetória desejada. Por outro lado, o controlador PID apresentou o menor desempenho na tarefa de rastreamento de trajetória e o maior erro máximo entre os controladores implementados. Em relação ao LQR, percebe-se que se mostrou com desempenho similar ao RLQR, entretanto, é visível que necessita de um tempo maior para convergir para a trajetória desejada. Em vista disso, o erro máximo do controlador LQR ficou muito próximo ao do controlador PID.

Com relação ao tempo de resposta inicial, é possível observar que o controlador PID convergiu mais rapidamente para a trajetória desejada. Este fato pode ser melhor visualizado na Figura 19, onde o erro de posição para o PID decresce mais rapidamente que os dos outros controladores, entretanto, durante a trajetória, se mantém mais elevado que os controladores RLQR e LQR.

Figura 19 – Evolução do erro de posição ao longo do tempo para cada controlador implementado.



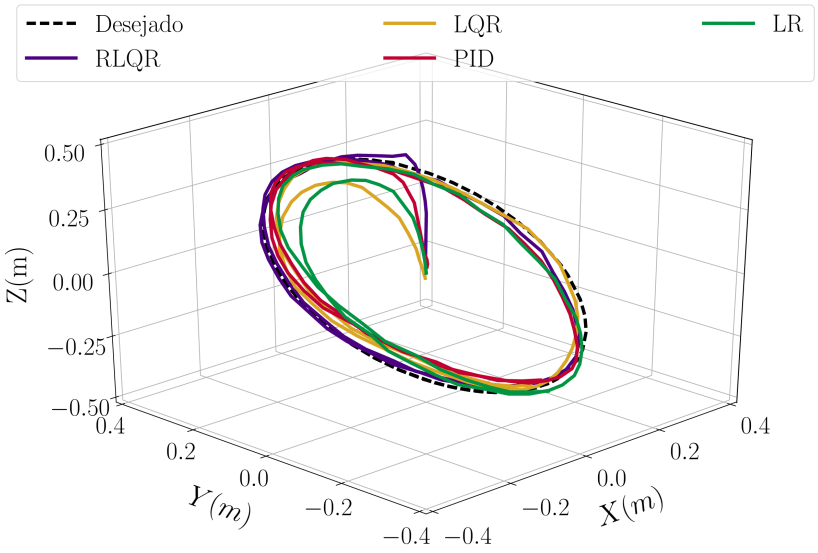
Fonte: Elaborada pelo autor.

Tabela 8 – Índices de desempenho para voos sem aplicação de distúrbio de vento.

Métricas	**Controladores Isolados**			
	RLQR	**LQR**	**PID**	**LR**
Erro de Posição [m]	0,0677	0,0821	0,1410	0,1336
Erro Máximo após 3s [m]	0,0987	0,1700	0,1872	0,2587

As Figuras 20, 21, 22 apresentam os resultados de voos utilizando a arquitetura 1, em que a DNN é responsável por auxiliar o controlador durante o seguimento de trajetória. Diante disso, a vista 3D é apresentada na Figura 20 e as variáveis controladas (x, y, z, ψ) são apresentadas na Figura 21. Para este caso, a evolução do erro é exposta na Figura 22 e os índices de desempenho avaliados são apresentados na Tabela 9.

Figura 20 – Rastreamento de trajetória circular em 3D, utilizando a rede neural como referência para os controladores.

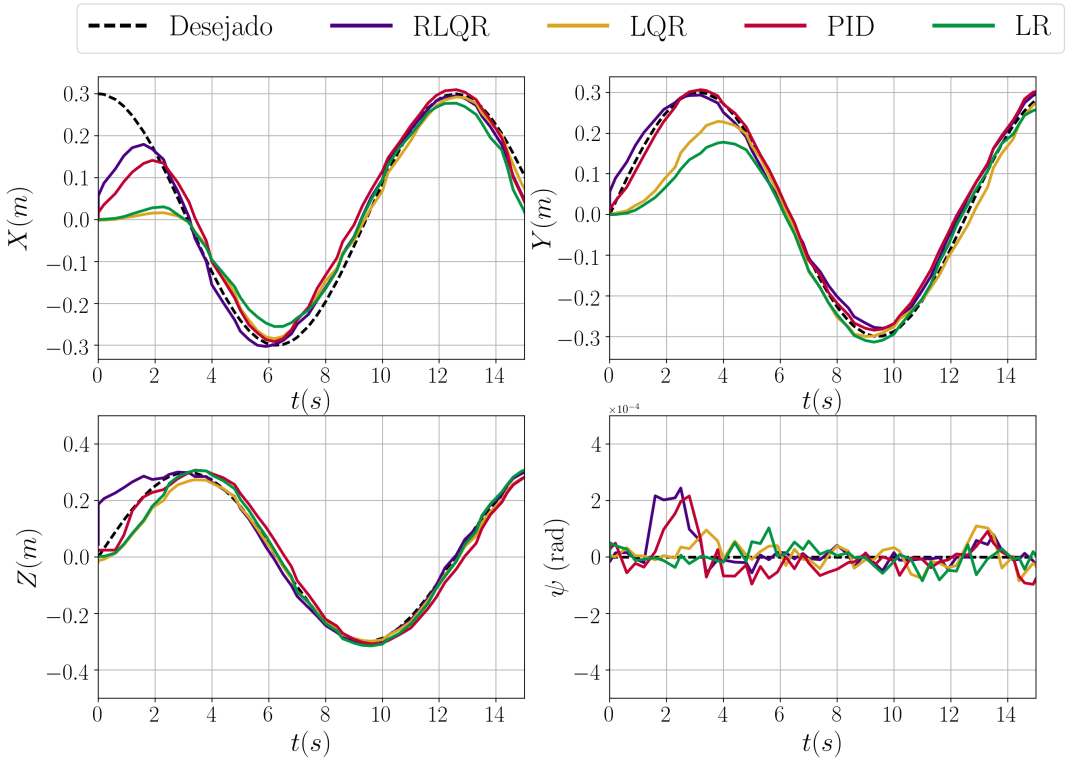


Fonte: Elaborada pelo autor.

Para a arquitetura de controle 1 proposta, é possível perceber que assim como para o primeiro caso analisado (controladores isolados), o RLQR obteve o menor erro de rastreamento de trajetória e o menor erro máximo após decorridos 3s de voo. Como esperado, o desempenho dos controladores PID e LR foram semelhantes, com o primeiro levando uma ligeira vantagem, deixando o LR com o pior desempenho entre os controladores analisados. Perceba que de acordo com as Figuras 20, 21, 22, o controlador LR possui um tempo de convergência similar ao do LQR, sendo maior que o dos controladores RLQR e PID. Novamente o controlador LQR foi o que mais se aproximou do desempenho do RLQR, porém, com um tempo de convergência maior, o que contribuiu para um aumento do erro ao longo da trajetória.

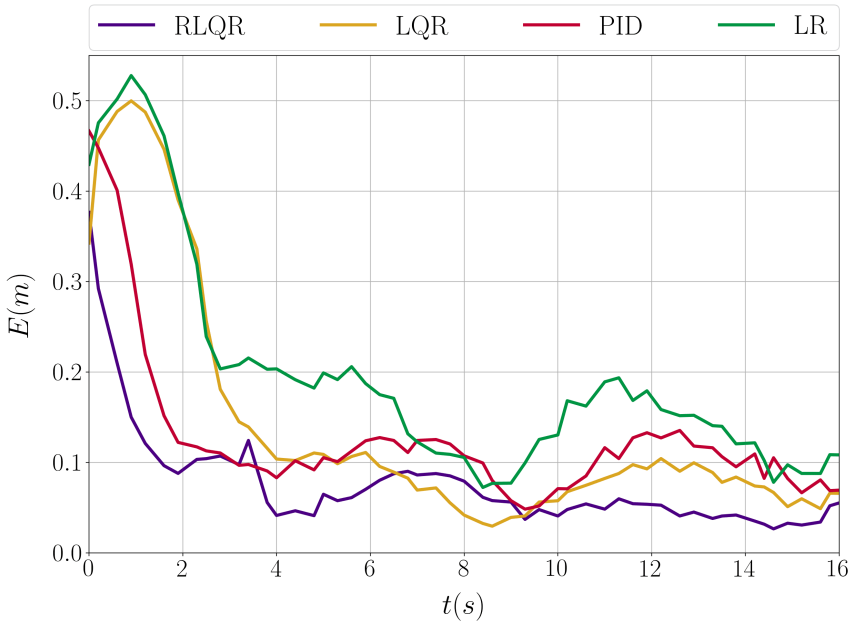
Com o objetivo de analisar a arquitetura proposta, a Tabela 9 apresenta os percentuais de melhoria de cada controlador em conjunto com a DNN em relação aos controladores isolados. Diante disso, é possível verificar que todos os controladores melhoraram seus desempenhos na tarefa de rastreamento de trajetória. Até mesmo os controladores RLQR e LQR que já haviam demonstrado bons desempenhos, tiveram melhoras significativas.

Figura 21 – Variáveis de posição controladas, utilizando a rede neural como referência para os controladores.



Fonte: Elaborada pelo autor.

Figura 22 – Evolução do erro de posição ao longo do tempo, utilizando a rede neural como referência para os controladores.



Fonte: Elaborada pelo autor.

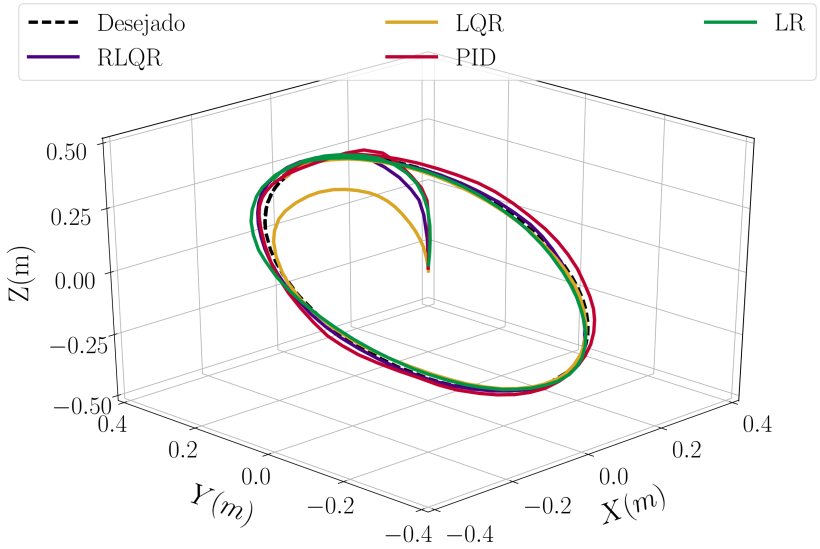
Ademais, nota-se que o PID teve o maior percentual de melhoria (27, 30%), ou seja, foi o controlador que teve o maior aumento de desempenho quando comparado à sua versão isolada.

Tabela 9 – Índices de desempenho para voos sem aplicação de distúrbio de vento, utilizando a rede neural como referência para os controladores.

Métricas	**Rede neural como gerador de referência**			
	RLQR	**LQR**	**PID**	**LR**
Erro de Posição [m]	0,0570	0,0712	0,1025	0,1242
Erro Máximo após 3s [m]	0,0903	0,1111	0,1362	0,2060
Melhoria [%]	15, 81%	13, 28%	27, 30%	7, 04%

As Figuras 23, 24, 25 apresentam os resultados de voos utilizando a arquitetura de controle 2 proposta, em que a rede neural atua como estimador de distúrbio. Para este caso, a vista 3D do rastreamento de trajetória é apresentada na Figura 23 e as variáveis controladas (x, y, z, ψ) são apresentadas na Figura 24. A evolução do erro é exibida na Figura 25, sendo complementada pelos índices de desempenho apresentados na Tabela 10.

Figura 23 – Rastreamento de trajetória circular em 3D, utilizando controle baseado em observação de distúrbio com rede neural atuando como estimador de distúrbio.



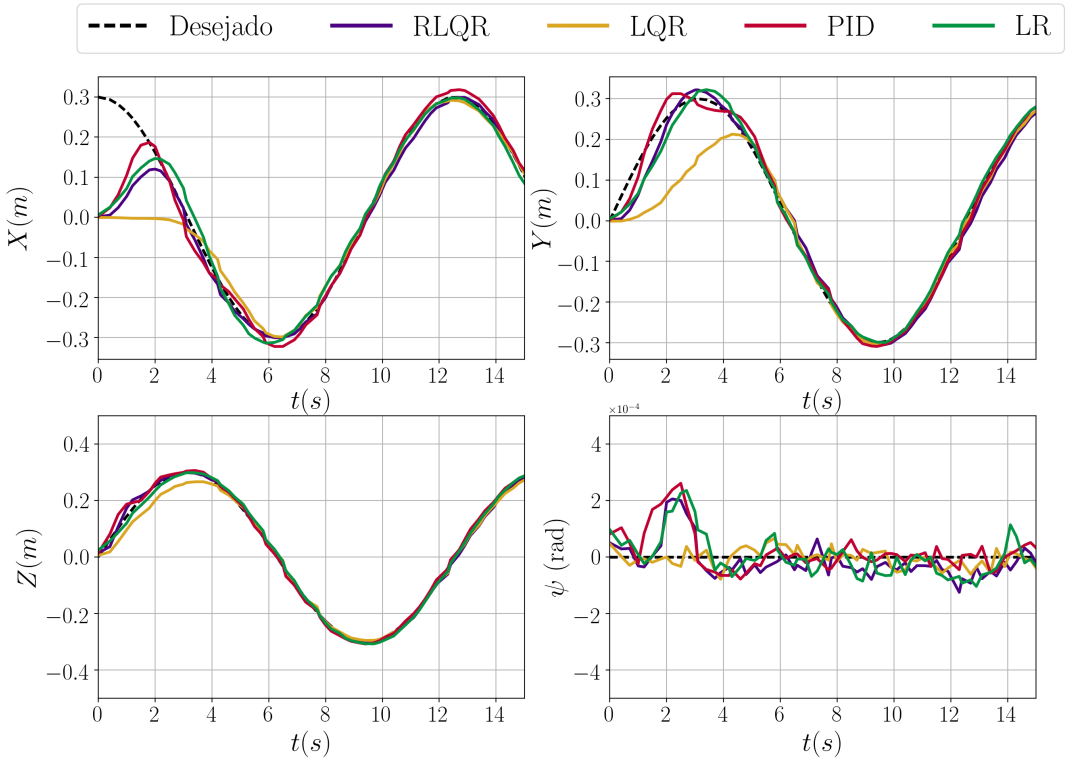
Fonte: Elaborada pelo autor.

Analisando as curvas de seguimento de trajetória apresentadas na Figura 23 e os índices de desempenho apresentados na Tabela 10, percebe-se que os controladores PID e LR tiveram resultados similares, sendo o segundo com o menor desempenho. Os

controladores RLQR e LQR também demonstraram resultados similares, com o melhor desempenho ficando para a versão robusta, principalmente pela resposta mais rápida no início da trajetória. Além disso, é possível verificar o menor erro máximo por parte do RLQR após decorridos $3s$ de voo.

A arquitetura DOBC com a rede neural atuando como estimador de distúrbio foi capaz de melhorar o desempenho de todos os controladores implementados. Analisando a Tabela 10, pode-se observar os percentuais de melhoria para cada um deles. Neste caso, o desempenho dos controladores RLQR e LQR ficaram ainda mais próximos.

Figura 24 – Variáveis de posição controladas, utilizando controle baseado em observação de distúrbio com rede neural atuando como estimador de distúrbio.

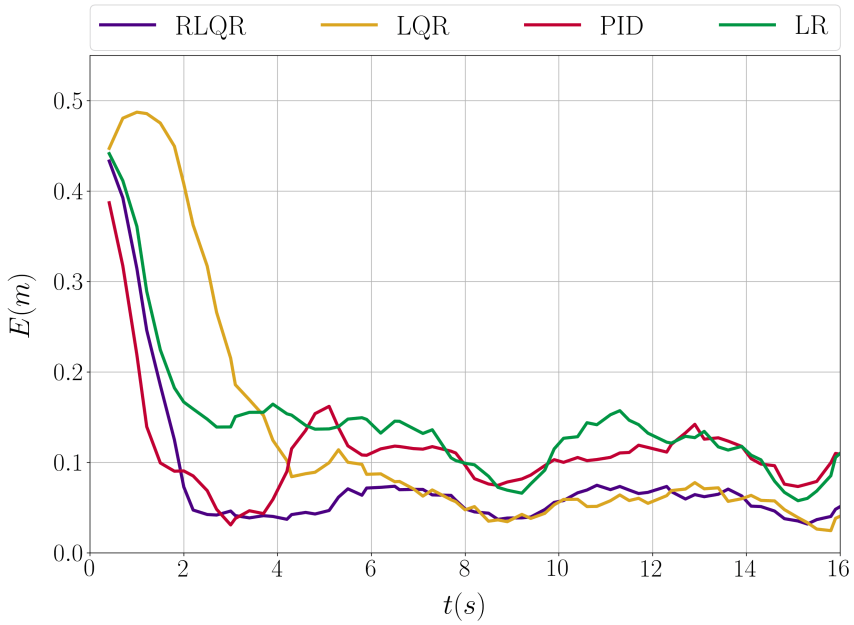


Fonte: Elaborada pelo autor.

Tabela 10 – Índices de desempenho para voos sem aplicação de distúrbio de vento, utilizando controle baseado em observação de distúrbio com rede neural atuando como estimador de distúrbio.

Métricas	**Rede neural como estimador de distúrbio**			
	RLQR	**LQR**	**PID**	**LR**
Erro de Posição [m]	0,0578	0,0603	0,1067	0,1150
Erro Máximo após 3s[m]	0,0748	0,1246	0,1621	0,1690
Melhoria [%]	14, 62%	26, 55%	24, 33%	13, 92%

Figura 25 – Evolução do erro de posição ao longo do tempo, utilizando controle baseado em observação de distúrbio com rede neural atuando como estimador de distúrbio.



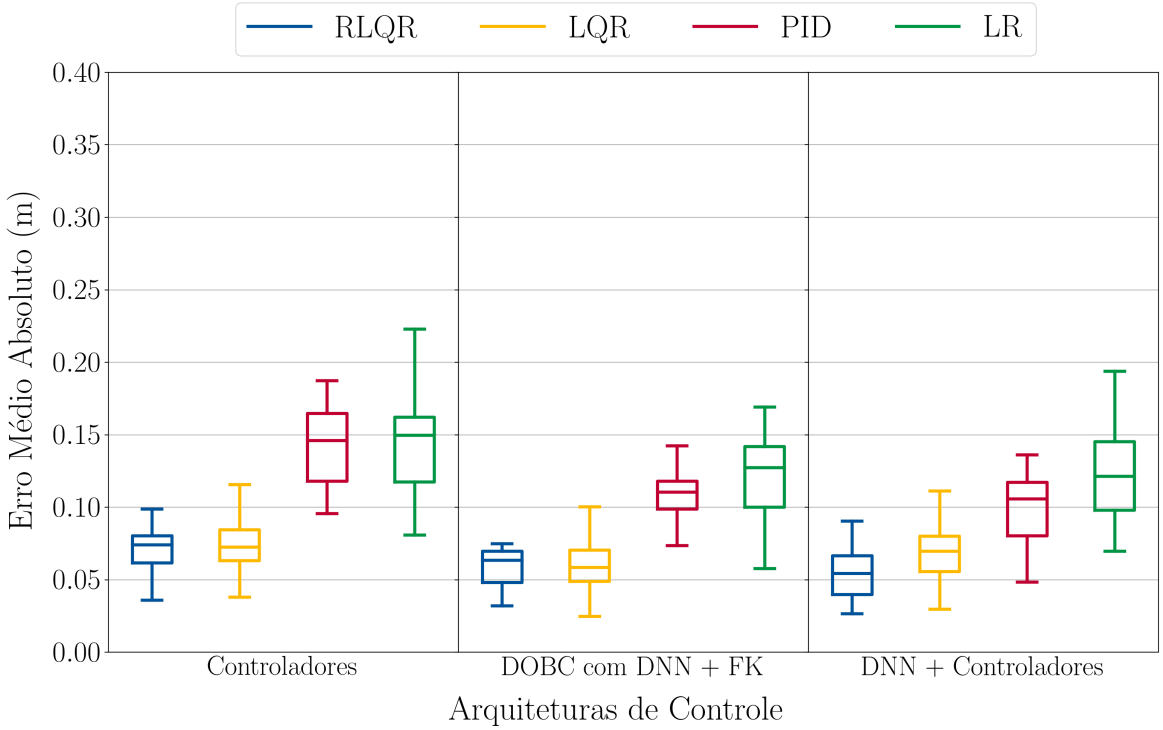
Fonte: Elaborada pelo autor.

Com o intuito de analisar o desempenho dos controladores paras os três casos apresentados acima, a Figura 26 apresenta o desempenho de cada uma das arquiteturas de controle propostas. Com isso, é possível perceber que ambas as arquiteturas tiveram desempenho superior quando comparadas à utilização dos controladores isolados. Além disso, é possível verificar desempenhos muito próximos para as duas arquiteturas propostas neste trabalho. Sendo que, de acordo com a Tabela 11, a arquitetura 1 apresentou melhor desempenho para os controladores RLQR e PID. Já a arquitetura 2 demonstrou um melhor desempenho paras os controladores LQR e LR. Vale lembrar que a arquitetura de controle 1 corresponde à rede neural que atua como gerador de referência para os controladores. Já a arquitetura de controle 2 refere-se à rede neural que atua como estimador de distúrbio, formando a arquitetura DOBC.

Tabela 11 – Percentual de melhoria das arquiteturas de controle propostas em relação aos controladores isolados.

Arquiteturas de Controle	**Percentual de Melhoria relativa ao uso isolado ($\overline{E}$)**			
	RLQR	**LQR**	**PID**	**LR**
Arquitetura de controle 1	$15,81\%$	$13,28\%$	$27,30\%$	$7,04\%$
Arquitetura de controle 2	$14,62\%$	$26,55\%$	$24,33\%$	$13,92\%$

Figura 26 – Boxplot do desempenho de cada controlador para os casos em que não há auxílio da rede neural profunda, à esquerda; que há auxílio da rede, atuando como estimador de distúrbio, no centro; e que há auxílio da rede, atuando como gerador de referência para os controladores, à direita.



Fonte: Elaborada pelo autor.

7.1.2 Experimentos com Aplicação de Distúrbios de Vento

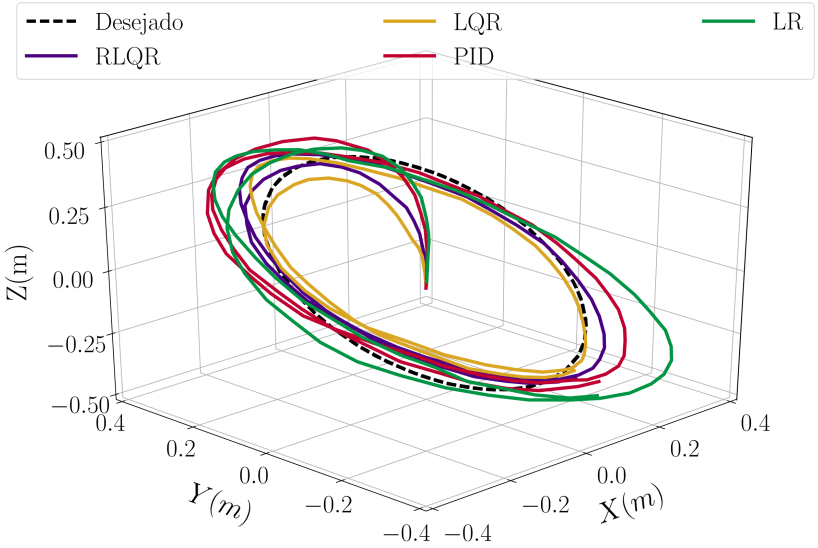
Nesta seção, são apresentados os resultados de rastreamento de trajetória quando o quadricóptero é afetado por distúrbio de vento (subseção 6.1.3.1). Assim como foi apresentado para condições normais de voo, os resultados são discutidos para cada uma das arquiteturas propostas e, em seguida, é realizada uma comparação entre as arquiteturas desenvolvidas.

Vale lembrar que o distúrbio de vento utilizado no ambiente simulado foi apresentado na subseção 6.1.3.1, a partir da Figura 12 e Equação (6.1), que representa um distúrbio de vento lateral constante com velocidade média de 5 m/s.

As Figuras 27, 28, 29 apresentam os resultados de voos utilizando os controladores isolados quando a aeronave está sob influência de distúrbio de vento. Diante disso, a vista 3D do rastreamento de trajetória é apresentada na Figura 27 e as variáveis controladas (x, y, z, ψ) são apresentadas na Figura 28. A evolução do erro para este caso é exposta na Figura 29, sendo complementada pelos índices de desempenho apresentados na Tabela 12.

Para este caso, é possível perceber que, entre os controladores implementados, o LR obteve o maior erro de rastreamento de trajetória, assim como o maior erro máximo após

Figura 27 – Rastreamento de trajetória circular em 3D quando o quadricóptero é afetado por distúrbio de vento.



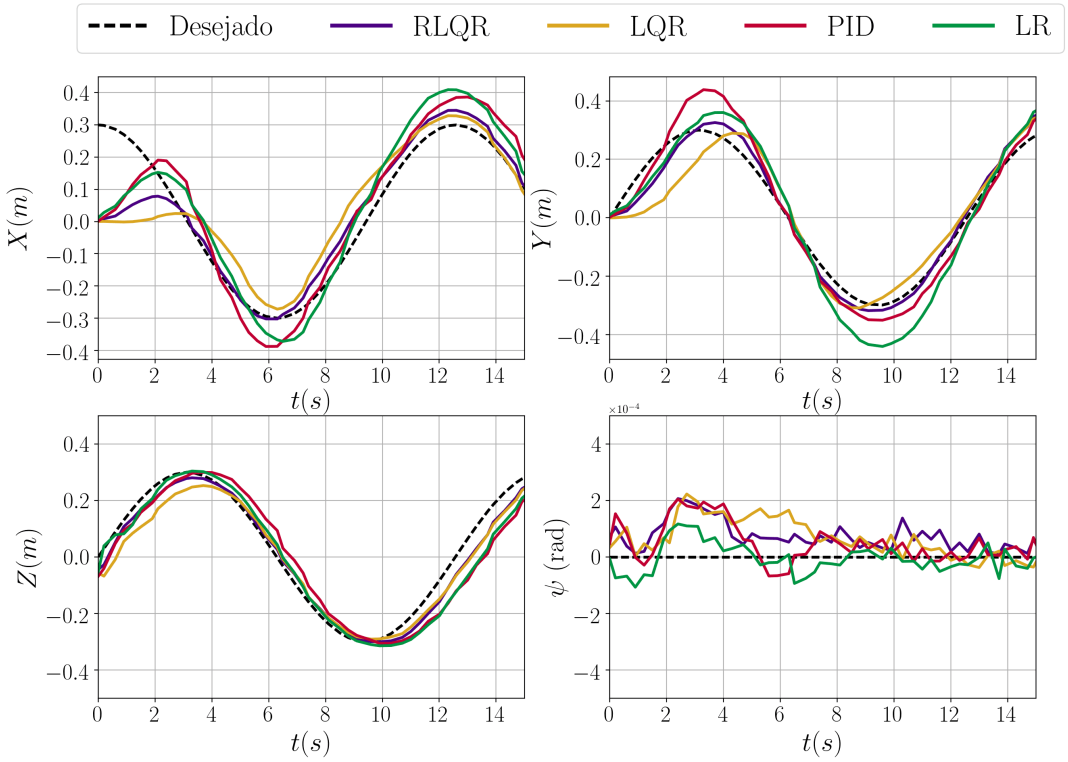
Fonte: Elaborada pelo autor.

decorridos 3s de voo. Por outro lado, o controlador RLQR apresentou o melhor desempenho na tarefa de rastreamento de trajetória e o menor erro máximo entre os controladores implementados. De acordo com a Figura 27, pode-se perceber que os controladores LQR e RLQR tiveram os melhores desempenhos quando comparados aos demais controladores. A partir dos índices numéricos apresentados na Tabela 12, é possível verificar que o LQR teve um menor desempenho global em relação ao RLQR. Já o controlador PID, teve um desempenho avaliado entre o LQR e o LR, demonstrando uma resposta rápida no início da trajetória.

Tabela 12 – Índices de desempenho para voos com aplicação de distúrbio de vento.

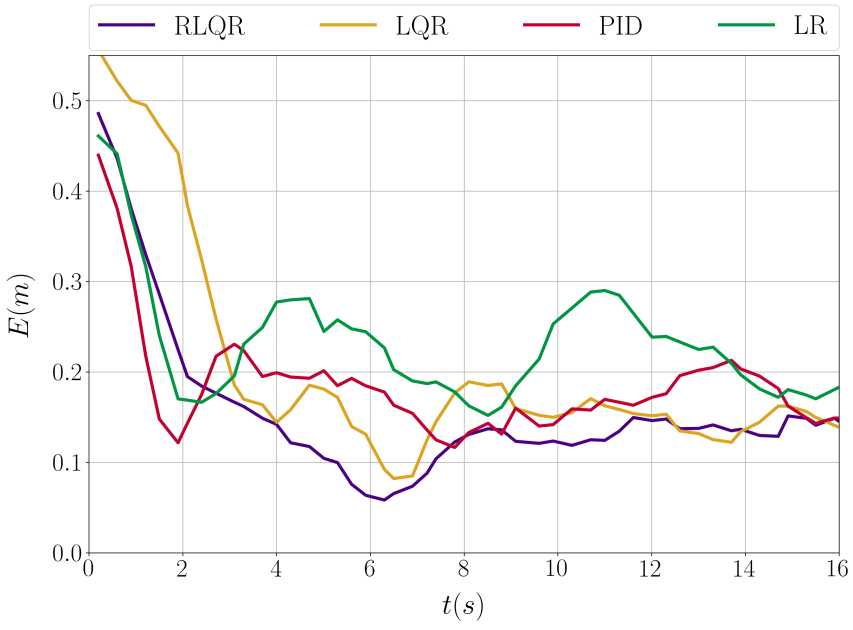
Métricas	**Controladores Isolados**			
	RLQR	**LQR**	**PID**	**LR**
Erro de Posição [m]	0,1175	0,1288	0,1648	0,1982
Erro Máximo após 3s[m]	0,1515	0,1892	0,2130	0,2902

Figura 28 – Variáveis de posição controladas quando o quadricóptero é afetado por distúrbio de vento.



Fonte: Elaborada pelo autor.

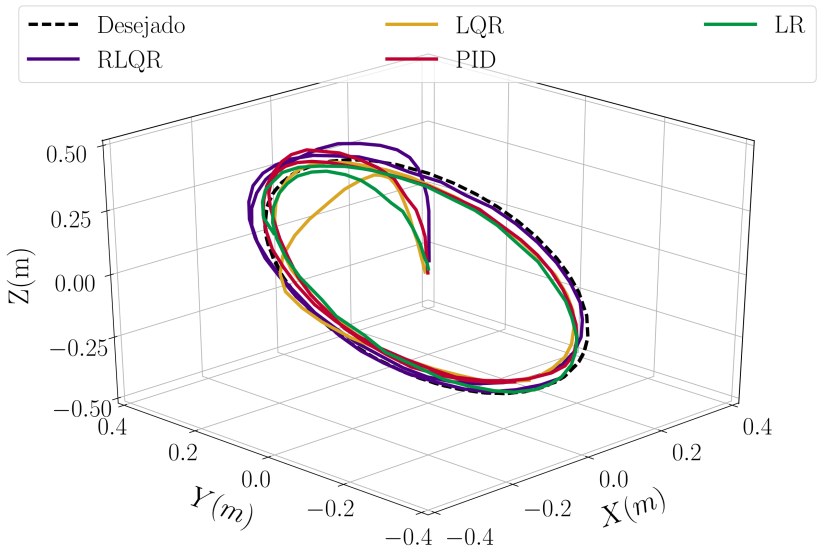
Figura 29 – Evolução do erro de posição ao longo do tempo para cada controlador implementado quando o quadricóptero é afetado por distúrbio de vento.



Fonte: Elaborada pelo autor.

As curvas apresentadas nas Figuras 30, 31, 32 apresentam os resultados de voos utilizando a arquitetura 1, em que a DNN é responsável por auxiliar o controlador durante o seguimento de trajetória. Diante disso, a vista 3D é apresentada na Figura 30 e as variáveis controladas (x, y, z, ψ) são apresentadas na Figura 31. Para este caso, a evolução do erro é evidenciada na Figura 32 e os índices numéricos de desempenho são apresentados na Tabela 13.

Figura 30 – Rastreamento de trajetória circular em 3D, utilizando a rede neural como referência para os controladores para o caso em que o quadricóptero é afetado por distúrbio de vento.

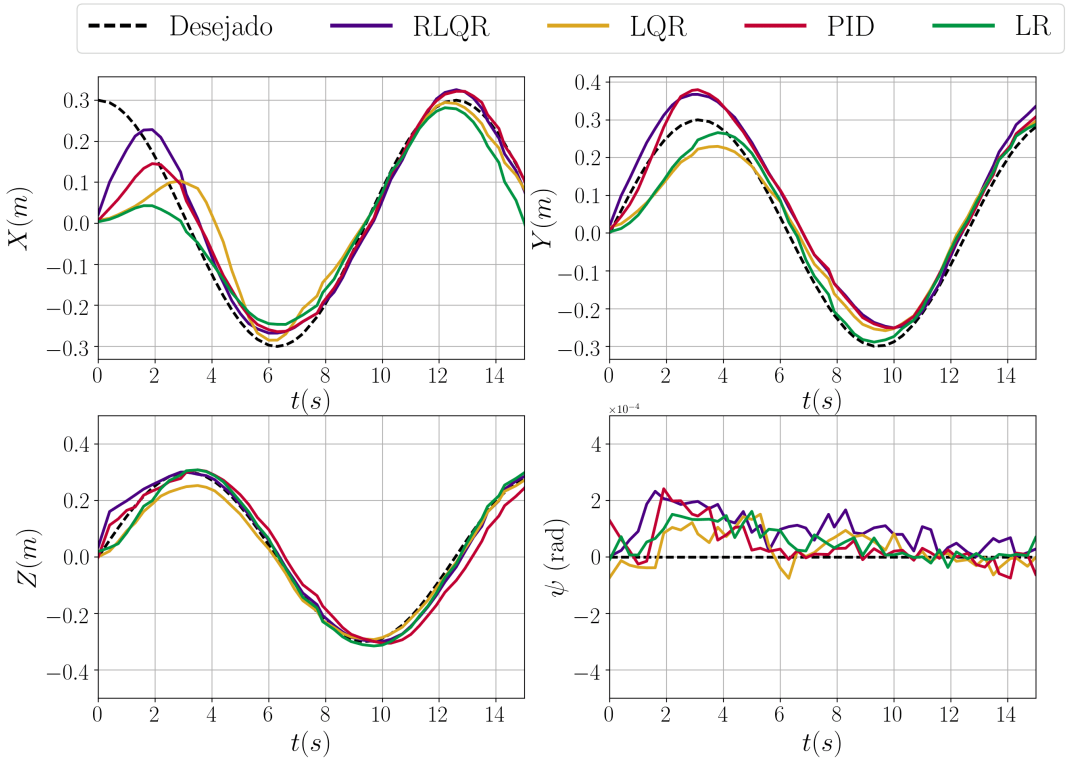


Fonte: Elaborada pelo autor.

Analisando os resultados apresentados para a arquitetura 1 proposta, pode-se observar que, assim como para o primeiro caso analisado (controladores isolados), o RLQR obteve tanto o menor erro de rastreamento de trajetória como o menor erro máximo após decorridos 3s de voo. Além disso, é possível visualizar novamente um desempenho global similar entre o RLQR e LQR (Tabela 13). Como esperado, o desempenho dos controladores PID e LR também foram semelhantes, com destaque para o PID, que apresentou o menor erro de rastreamento de trajetória quando comparado ao LR. Note que de acordo com as Figuras 30, 31, 32, assim como apresentado para os casos em que não se tem influência do distúrbio de vento, o controlador LR apresentou um tempo de resposta no início da trajetória similar ao do LQR. Novamente o controlador LQR foi que mais se aproximou do desempenho do RLQR, porém, com um tempo de convergência maior, o que contribuiu para um erro maior no início da trajetória.

A fim de analisar a arquitetura proposta, a Tabela 13 apresenta os percentuais de melhoria de cada controlador em conjunto com a DNN em relação aos controladores isolados.

Figura 31 – Variáveis de posição controladas, utilizando a rede neural como referência para os controladores para o caso em que o quadricóptero é afetado por distúrbio de vento.



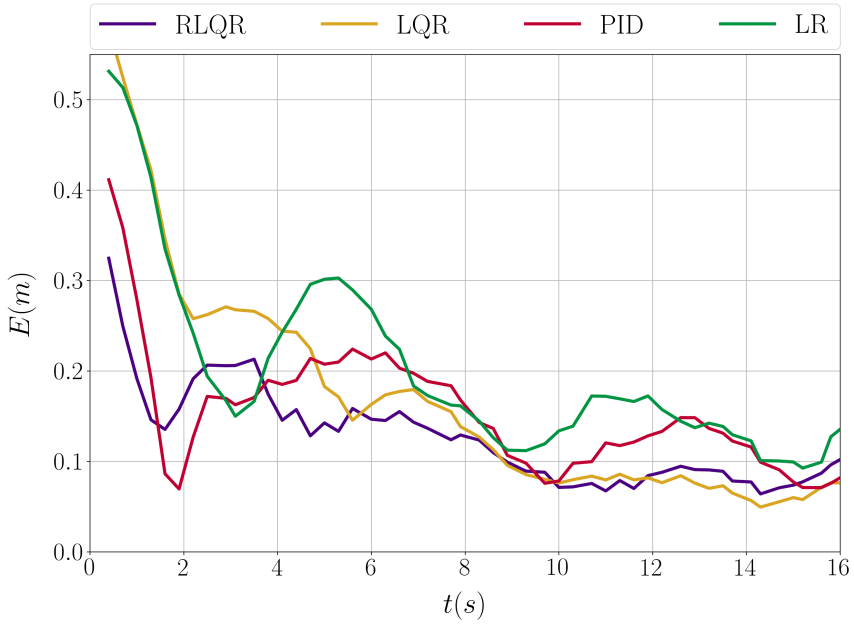
Fonte: Elaborada pelo autor.

Em vista disso, é possível verificar uma melhora por parte de todos os controladores na tarefa de rastreamento de trajetória. Com a arquitetura 1, o desempenho do controlador LQR se aproximou ainda mais do desempenho do RLQR, ambos tiveram melhoras significativas. Além disso, nota-se que o controlador LR teve o maior percentual de melhoria com $25,58\%$ entre os controladores, sendo capaz de realizar o seguimento de trajetória com um erro bem abaixo do que o analisado para o caso 1 (controladores isolados).

A seguir, as Figuras 33, 34, 35 apresentam os resultados de voos utilizando a arquitetura DOBC com rede neural atuando como estimador de distúrbio. A Figura 33 apresenta a vista 3D do rastreamento de trajetória, seguida da Figura 34, que apresenta as variáveis controladas (x, y, z, ψ). A evolução do erro é exibida na Figura 35 e, para uma análise quantitativa, a Tabela 14 apresenta os índices de desempenho avaliados.

Analisando as curvas de rastreamento de trajetória apresentadas na Figura 33 e os índices de desempenho apresentados na Tabela 14, percebe-se que o controlador LR teve o menor desempenho entre os controladores avaliados, principalmente pelo erro elevado no início da trajetória. Por outro lado, o controlador RLQR apresentou o melhor

Figura 32 – Evolução do erro de posição ao longo do tempo para cada controlador implementado, utilizando a rede neural como referência para os controladores para o caso em que o quadricóptero é afetado por distúrbio de vento.



Fonte: Elaborada pelo autor.

Tabela 13 – Índices de desempenho, utilizando a rede neural como referência para os controladores para o caso em que o quadricóptero é afetado por distúrbio de vento.

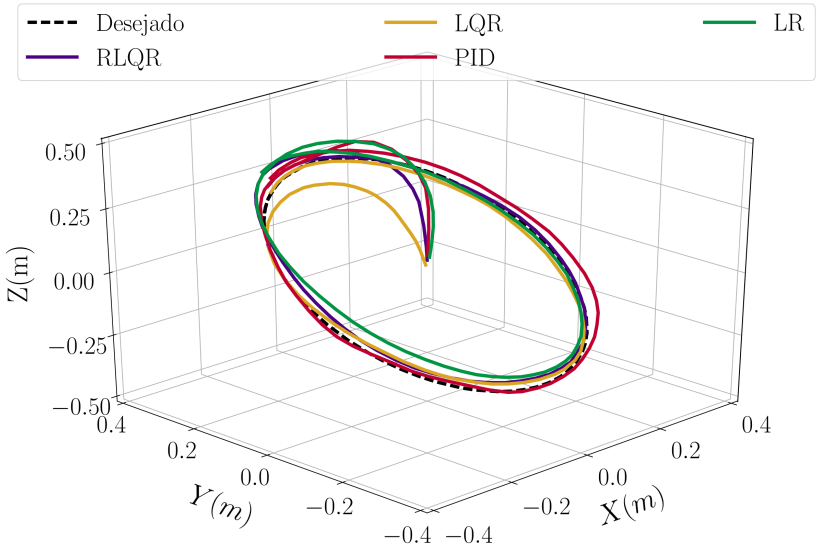
Métricas	**Rede neural como gerador de referência**			
	RLQR	**LQR**	**PID**	**LR**
Erro de Posição [m]	0,0942	0,0966	0,1330	0,1475
Erro Máximo após 3s[m]	0,1586	0,2246	0,2243	0,3029
Melhoria [%]	19, 83%	25, 00%	19, 30%	25, 58%

desempenho, com tempo de convergência similar ao do controlador PID. Além disso, é possível verificar, novamente, o menor erro máximo por parte do RLQR após decorridos $3s$ de voo, contribuindo para o bom desempenho do controlador.

Com relação à arquitetura DOBC, em que a rede atua como estimador de distúrbio (arquitetura 2), os resultados mostraram que foi capaz de melhorar o desempenho e, consequentemente, reduziu o erro de posição para todos os controladores implementados. A partir da Tabela 14, pode-se observar os percentuais de melhoria para cada um deles. Com isso, nota-se melhorias significativas por parte dos controladores PID e LR, com destaque, principalmente, para os controladores RLQR e LQR.

Sabendo que a arquitetura 2 fornece uma lei de controle composta (u_{com}), que é

Figura 33 – Rastreamento de trajetória circular em 3D para o caso em que o quadricóptero é afetado por distúrbio de vento, utilizando controle baseado em observação de distúrbio com rede neural atuando como estimador.



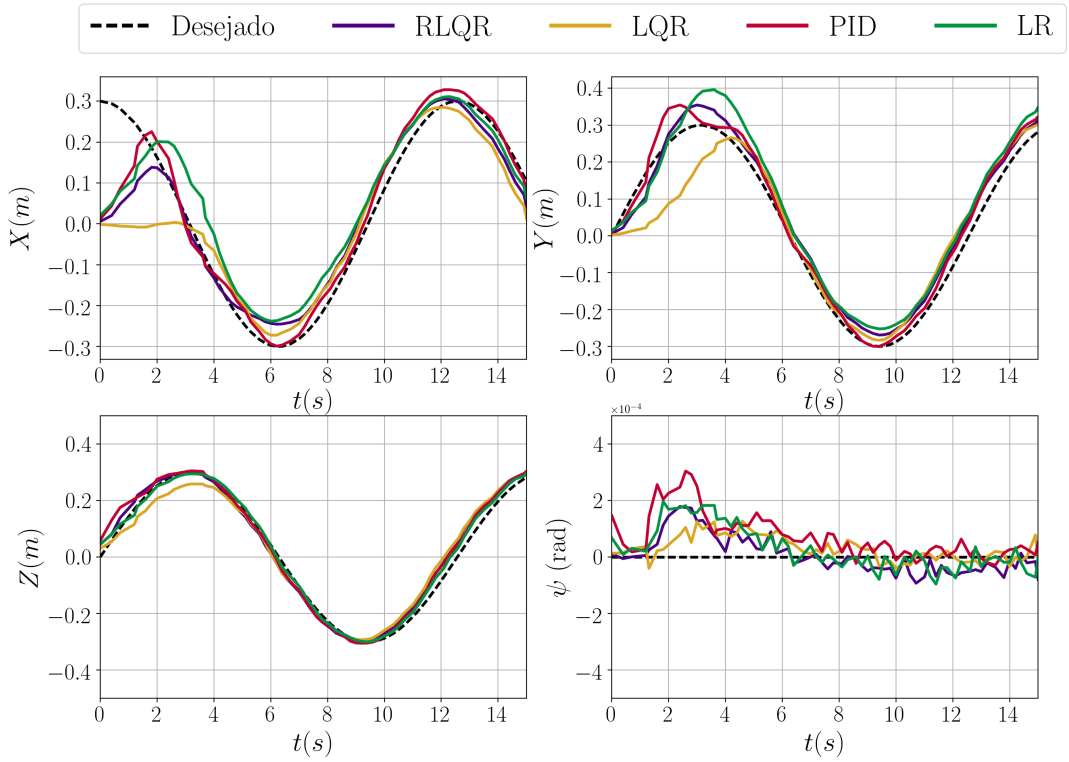
Fonte: Elaborada pelo autor.

Tabela 14 – Índices de desempenho para o caso em que o quadricóptero é afetado por distúrbio de vento, utilizando controle baseado em observação de distúrbio com rede neural atuando como estimador.

Métricas	**Rede neural como estimador de distúrbio**			
	RLQR	**LQR**	**PID**	**LR**
Erro de Posição [m]	0,0759	0,0801	0,1081	0,1452
Composição de u_{com} [%]	u_{rlqr} : 83, 62 u_{dist} : 16, 38	u_{lqr} : 71, 63 u_{dist} : 28, 37	u_{pid} : 71, 79 u_{dist} : 28, 21	u_{fl} : 80, 72 u_{dist} : 19, 28
Erro Máximo após 3s[m]	0,1725	0,1905	0,2173	0,3390
Melhoria [%]	35, 04%	37, 81%	34, 40%	26, 74%

dada pela composição entre controlador (u_{ctrl}) e distúrbio (u_{dist}), a Figura 36 em conjunto com a Tabela 14 apresentam a composição do sinal de controle. Dessa forma, pode-se observar os índices numéricos referentes a lei de controle composta para cada um dos controladores. Perceba que pra cada controlador, a composição da lei de controle é diferente. Sendo o LQR com maior influência da lei de controle proveniente do distúrbio, seguido pelo PID e LR. Já o RLQR foi o controlador que menos sofreu influência da lei de controle do distúrbio, ainda que, como apresentado acima, demonstrou o melhor desempenho no rastreamento de trajetória.

Figura 34 – Variáveis de posição controladas para o caso em que o quadricóptero é afetado por distúrbio de vento, utilizando controle baseado em observação de distúrbio com rede neural atuando como estimador.

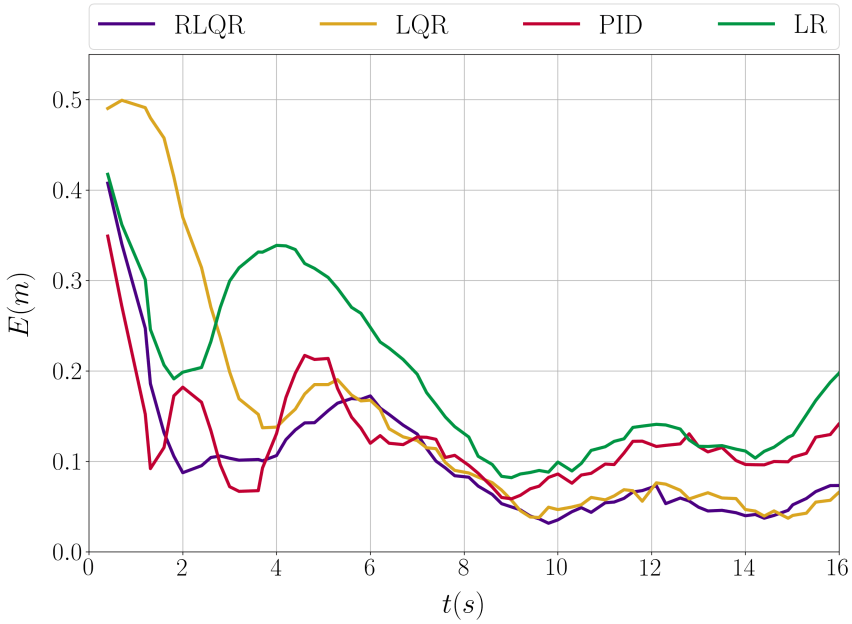


Fonte: Elaborada pelo autor.

A Figura 37 apresenta as estimativas do distúrbio em *Newtons* fornecida pela DNN e o sinal filtrado pelo filtro de Kalman para cada um dos controladores implementados. O distúrbio estimado afeta o eixo Y do quadricóptero durante todo o tempo de trajetória. Vale lembrar que essas estimativas do distúrbio de vento são utilizadas para criação da lei de controle (u_{dist}) que é somada à lei de controle proveniente do controlador (u_{ctrl}) para formação da lei de controle composta (u_{com}). Note que o filtro de Kalman tem a função de suavizar a estimativa de distúrbio fornecida pela DNN, deixando o sinal do distúrbio mais comportado.

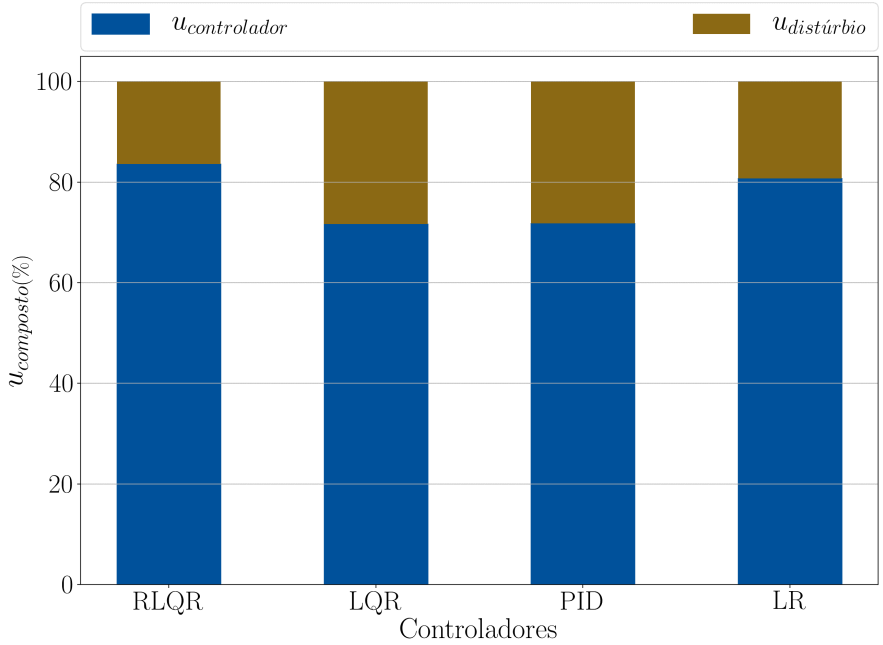
Com o intuito de analisar o desempenho das arquiteturas propostas para cada controlador implementado, a Figura 38 apresenta o boxplot para comparação do desempenho de rastreamento de trajetória. Com isso, é possível perceber que ambas as arquiteturas tiveram desempenho superior quando comparadas com à utilização dos controladores isolados. Além disso, é possível analisar um desempenho superior por parte da arquitetura 2, apresentando a menor média de erro para todos os controladores. Esse resultado pode ser explicado pelo fato desta arquitetura ser voltada à estimativa e atenuação de distúrbios externos. Entretanto, para o controlador LR, as duas arquiteturas de controle propostas demonstraram desempenhos muito próximos. Para uma análise quantitativa, a Tabela 15

Figura 35 – Evolução do erro de posição ao longo do tempo para o caso em que o quadricóptero é afetado por distúrbio de vento, utilizando controle baseado em observação de distúrbio com rede neural atuando como estimador.



Fonte: Elaborada pelo autor.

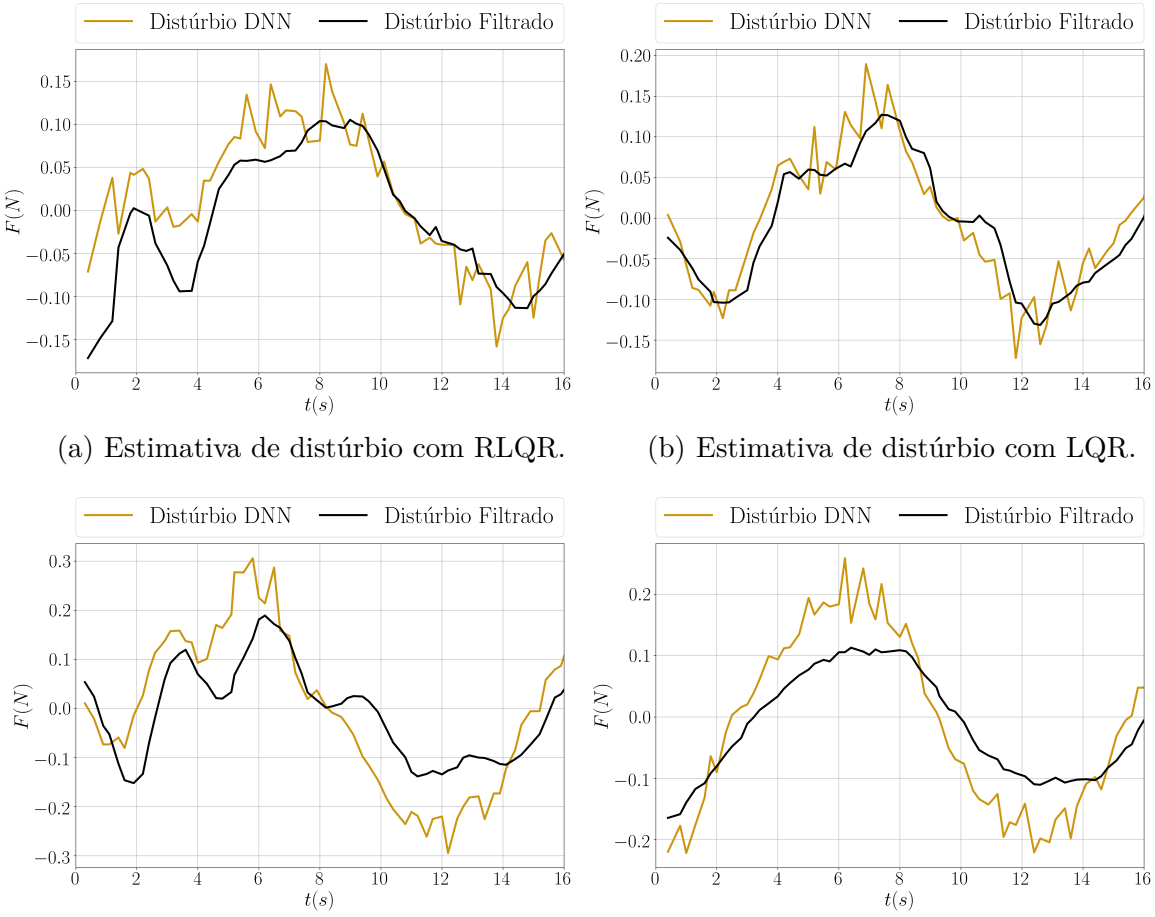
Figura 36 – Composição da lei de controle baseado em observação de distúrbio para cada controlador implementado.



Fonte: Elaborada pelo autor.

apresenta o percentual de melhoria das duas arquiteturas de controle propostas em relação aos controladores isolados. Dessa forma, é possível confirmar o melhor desempenho da arquitetura 2 para o caso que o quadricóptero é afetado por distúrbio de vento.

Figura 37 – Estimativas de distúrbio da Rede Neural e sinal filtrado pelo Filtro de Kalman para cada um dos controladores implementados.



(a) Estimativa de distúrbio com RLQR.

(b) Estimativa de distúrbio com LQR.

(c) Estimativa de distúrbio com PID.

(d) Estimativa de distúrbio com LR.

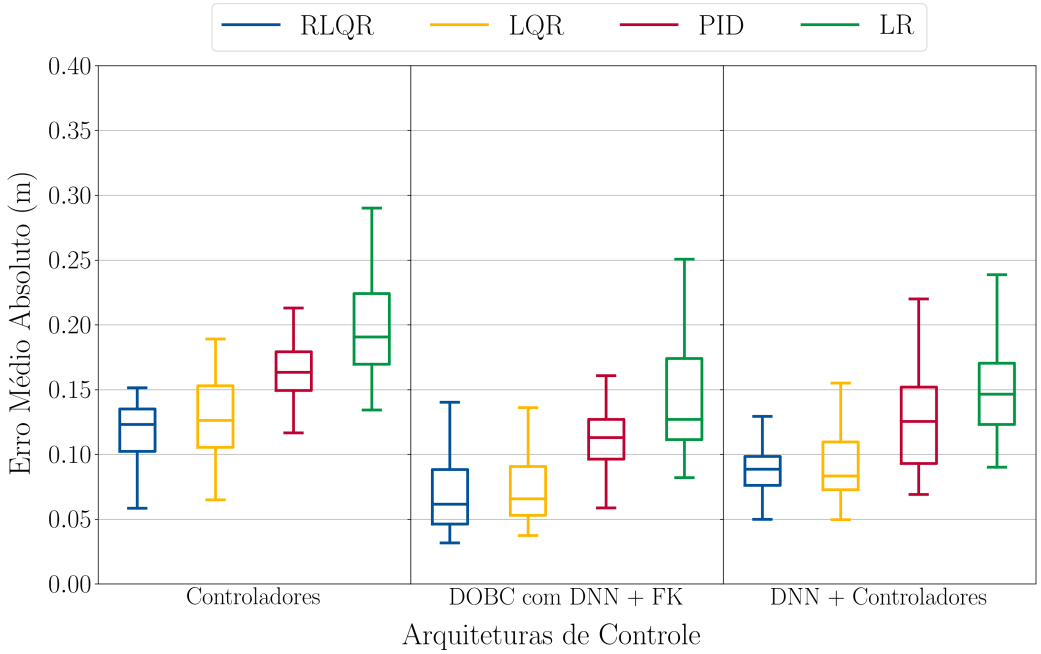
Fonte: Elaborada pelo autor.

Tabela 15 – Percentual de melhoria para as duas arquiteturas de controle propostas em relação aos controladores isolados.

Arquiteturas de Controle	**Percentual de Melhoria relativa ao uso isolado ($\overline{E}$)**			
	RLQR	**LQR**	**PID**	**LR**
Arquitetura de Controle 1	$19,83\%$	$25,00\%$	$19,30\%$	$25,58\%$
Arquitetura de Controle 2	$35,04\%$	$37,81\%$	$34,40\%$	$26,74\%$

A fim de validar a arquitetura 2 proposta, foram realizados, também, experimentos que simulam situações em que o quadricóptero é afetado por rajadas de ventos laterais. Para isto, o quadricóptero foi programado para seguir uma trajetória em linha reta no eixo X com cada um dos controladores implementados. Inicialmente foram testados os controladores isolados e, em seguida, utilizando a arquitetura 2. Uma rajada de vento com magnitude de 15 m/s foi aplicada no eixo Y da aeronave por 2 s no ponto $X = 3$ m.

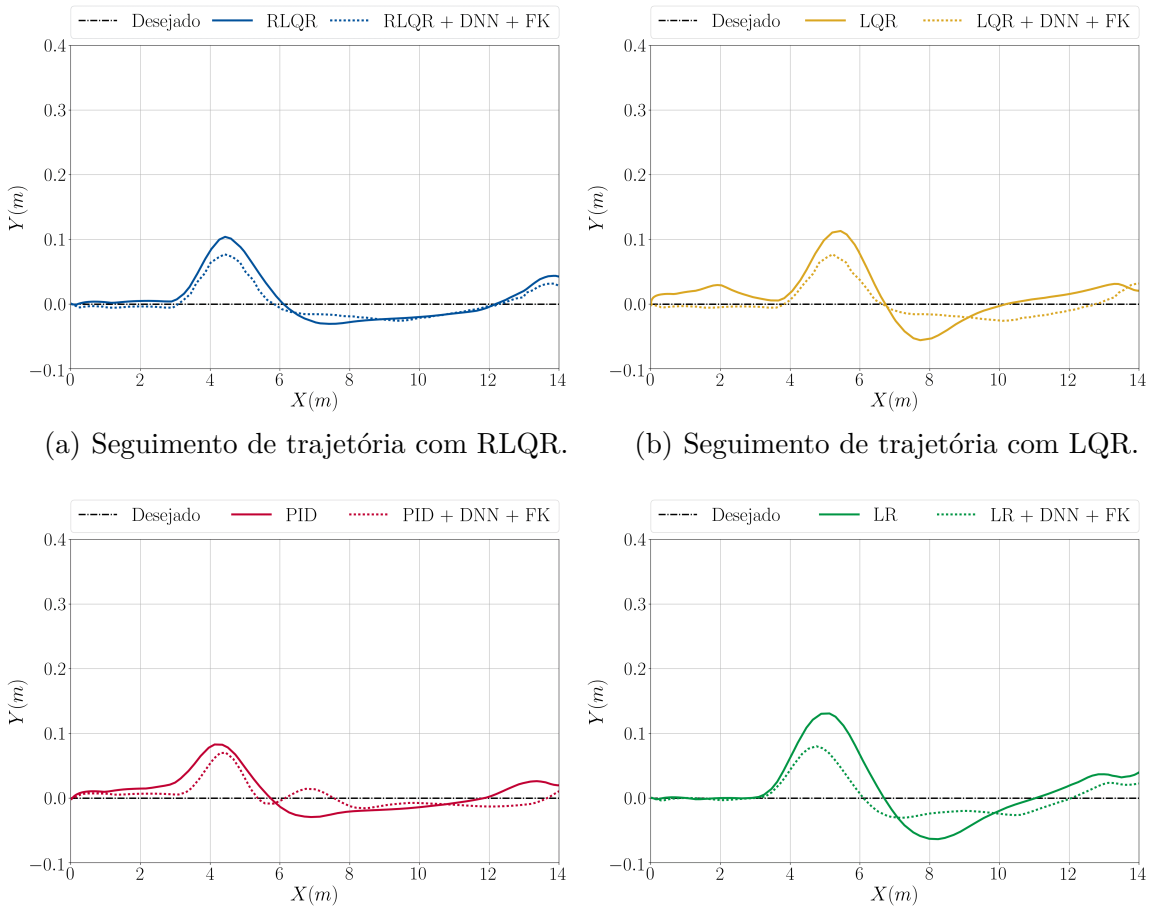
Figura 38 – Boxplot do desempenho de cada controlador com aplicação de distúrbio de vento para os casos em que não há auxílio da rede neural profunda, à esquerda; que há auxílio da rede, atuando como estimador de distúrbio, no centro; e que há auxílio da rede, atuando como gerador de referência para os controladores, à direita.



Fonte: Elaborada pelo autor.

A Figura 39 apresenta o experimento descrito, em que é possível observar, qualitativamente, o desempenho de cada um dos controladores com e sem a rede neural que estima distúrbio de vento. Neste caso, pode-se perceber que sem a utilização da DNN, os controladores RLQR e PID tiveram desempenhos semelhantes, enquanto o LR foi o controlador que mais divergiu da trajetória, seguido do LQR. Para os casos em que se utilizou a arquitetura 2, é possível perceber que todos os controladores tiveram uma melhora no desempenho, em geral, ficaram mais próximos da trajetória desejada sob a influência da rajada de vento. Além disso, apresentaram respostas mais rápidas após o distúrbio, com um tempo de retorno menor para as proximidades da trajetória desejada.

Figura 39 – Seguimento de trajetória em linha reta com aplicação de distúrbio de vento lateral com velocidade de 15 m/s no ponto $t = 3$ s.



(a) Seguimento de trajetória com RLQR.

(b) Seguimento de trajetória com LQR.

(c) Seguimento de trajetória com PID.

(d) Seguimento de trajetória com LR.

Fonte: Elaborada pelo autor.

7.2 Resultados Experimentais

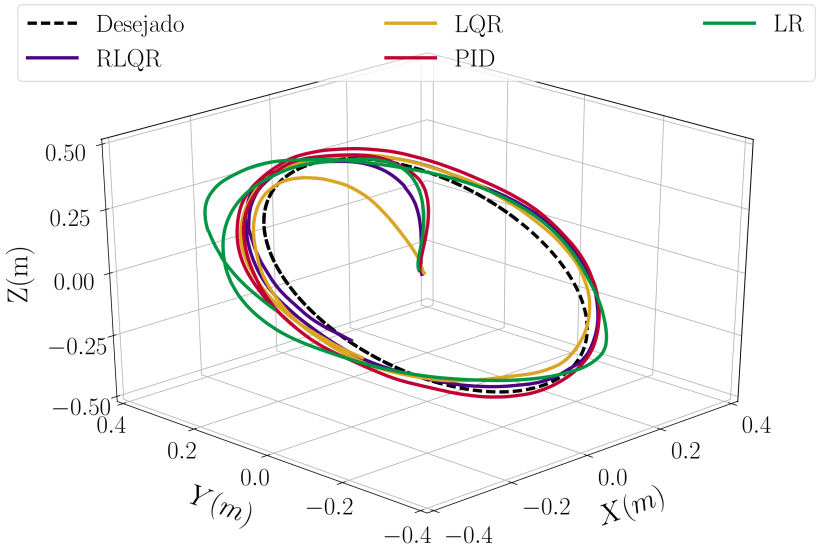
Nesta seção, são descritos os resultados de voos utilizando o quadricóptero Parrot™ Bebop 2.0. Assim como apresentado para o caso simulado, os experimentos consistem na realização do rastreamento de trajetória com o quadricóptero, utilizando a metodologia descrita no Capítulo 6. Primeiramente, são apresentados os resultados utilizando os controladores isolados, ou seja, sem auxílio da rede neural. Em seguida, são expostos os resultados utilizando as duas arquiteturas de controle propostas, em que a rede neural atua como gerador de referência e no caso em que a rede neural atua como estimador de distúrbio de vento. Os desempenhos de voos para cada caso foram avaliados sem a aplicação de distúrbio de vento e, posteriormente, com a aplicação de distúrbio, como apresentado na subseção 6.1.3.2.

7.2.1 Experimentos sem Aplicação de Distúrbios de Vento

Os resultados apresentados nesta seção compreendem o conjunto de experimentos realizados para condições normais de voo, ou seja, quando o quadricóptero não é afetado por uma fonte de distúrbio de vento externa. À vista disso, são apresentadas as figuras e os índices de desempenho tanto para os controladores isolados quanto para as arquiteturas de controle propostas. Primeiramente, os resultados são discutidos para cada caso a fim de avaliar o desempenho dos controladores dentro da arquitetura proposta. Posteriormente, os resultados das arquiteturas desenvolvidas são comparados.

As Figuras 40, 41 e 42 apresentam os resultados de voos experimentais utilizando os controladores isolados, em que a vista 3D do rastreamento de trajetória é apresentada na Figura 40 e as variáveis controladas (x, y, z, ψ) são apresentadas na Figura 41. A evolução do erro para este caso é exposta na Figura 42, sendo complementada pelos índices de desempenho apresentados na Tabela 16.

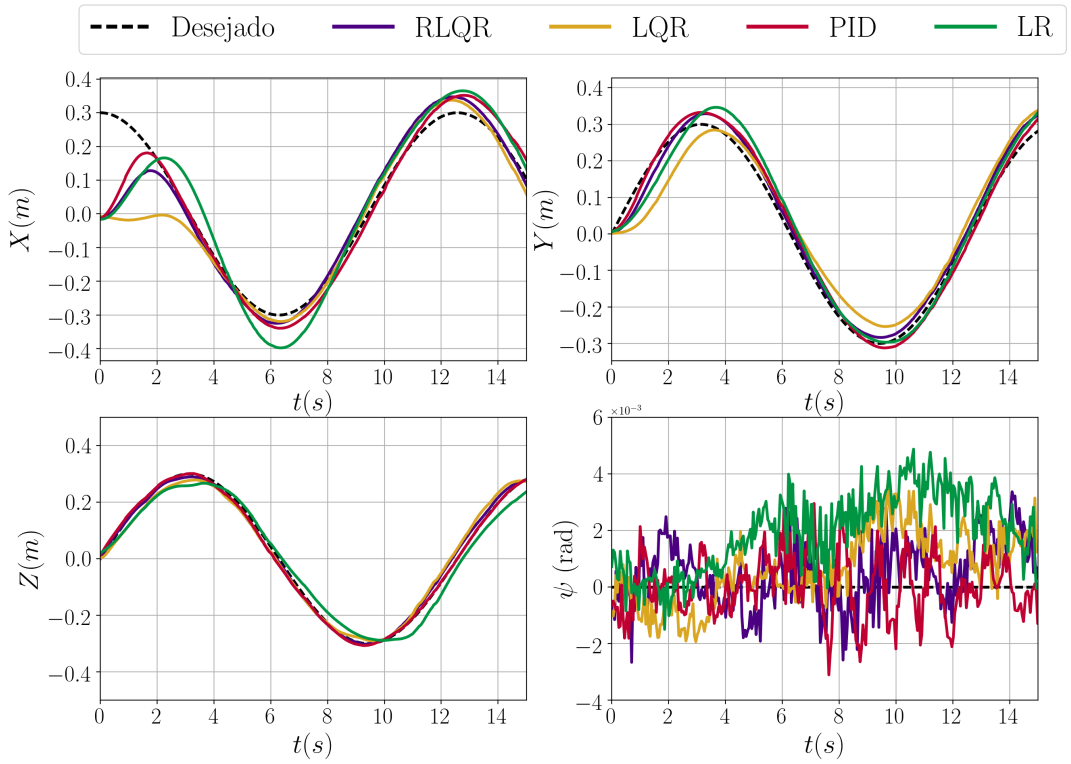
Figura 40 – Rastreamento de trajetória circular em 3D para cada controlador implementado.



Fonte: Elaborada pelo autor.

Note que, como esperado, os resultados experimentais alcançados são similares aos obtidos no ambiente simulado *Parrot-Sphinx*. Dessa forma, entre os controladores implementados, o RLQR obteve o menor erro de posição ($\overline{E}$) para a trajetória circular no plano XYZ. Além disso, apresenta o menor erro máximo após decorridos $3s$ de voo, tempo médio que o quadricóptero leva para convergir para a trajetória desejada. Por outro lado, o controlador LR apresentou o menor desempenho na tarefa de rastreamento de trajetória e o maior erro máximo entre os controladores implementados. Em relação

Figura 41 – Variáveis de posição e orientação controladas.



Fonte: Elaborada pelo autor.

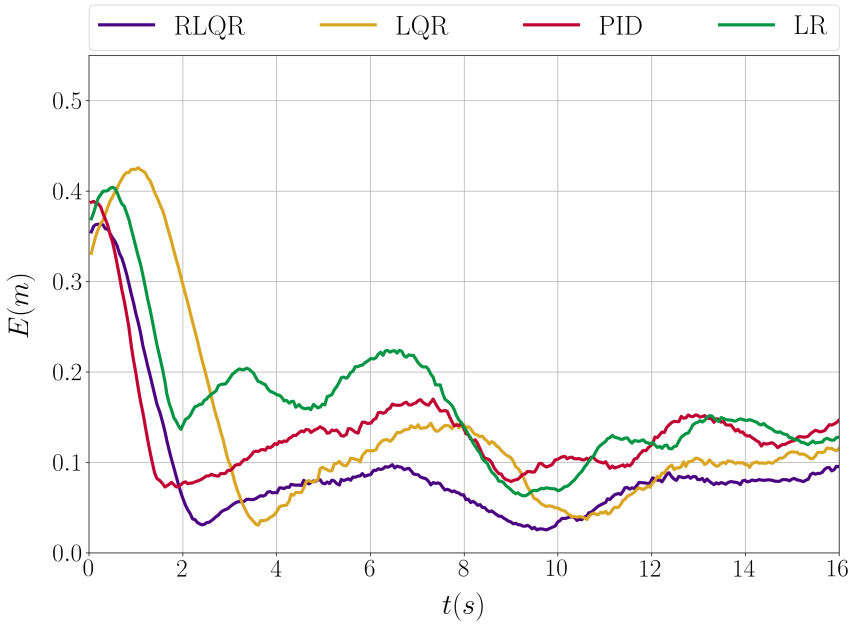
ao LQR, percebe-se que se mostrou com desempenho similar ao RLQR, entretanto, é visível que necessita de um tempo maior para convergir para a trajetória desejada, o que contribuiu para o aumento do erro de posição ao longo da trajetória. Já o controlador PID demonstrou um resultado satisfatório dado a sua simplicidade. Neste caso, tanto o erro de posição ($\overline{E}$) quanto o erro máximo se manteve entre os apresentados pelos controladores LQR e LR.

Com relação ao tempo de resposta inicial, é possível observar que o RLQR e o controlador PID tiveram tempos de convergência semelhantes. Este fato pode ser melhor visualizado na Figura 42, onde o erro de posição para esses dois controladores decresce mais rapidamente quando comparados ao LQR e LR.

Tabela 16 – Índices de desempenho para voos sem aplicação de distúrbio de vento.

Métricas	**Controladores Isolados**			
	RLQR	**LQR**	**PID**	**LR**
Erro de Posição [m]	0,0734	0,0991	0,1270	0,1485
Erro Máximo após 3s [m]	0,0976	0,1433	0,1701	0,2240

Figura 42 – Evolução do erro de posição ao longo do tempo para cada controlador implementado.



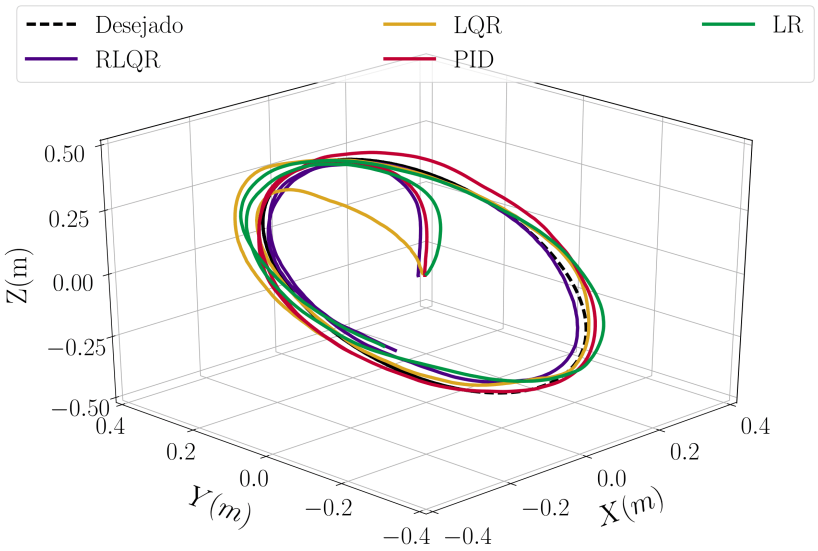
Fonte: Elaborada pelo autor.

As Figuras 43, 44 e 45 apresentam os resultados de voos utilizando a arquitetura 1, em que a DNN é responsável por auxiliar o controlador durante a tarefa de rastreamento de trajetória. Diante disso, a vista 3D é apresentada na Figura 43 e as variáveis controladas (x, y, z, ψ) são apresentadas na Figura 44. Para este caso, a evolução do erro é exposta na Figura 45 e os índices de desempenho avaliados são apresentados na Tabela 17.

Para os resultados experimentais com a arquitetura de controle 1 proposta, é possível perceber que assim como para o primeiro caso analisado (controladores isolados), o RLQR obteve o menor erro de rastreamento de trajetória e o menor erro máximo após decorridos $3s$ de voo. Além disso, o desempenho do controlador PID, neste caso, foi superior que o do controlador LR. Entre todos os controladores analisados, o LR, novamente, obteve o pior desempenho. Note que, diferentemente do caso simulado, o controlador LR apresentou um tempo de convergência similar ao do RLQR e, portanto, consideravelmente menor que o do LQR. Novamente o controlador LQR foi que mais se aproximou do desempenho do RLQR ao longo da trajetória, porém, com um tempo de convergência maior no início, o que contribuiu para um aumento do erro ao longo do percurso.

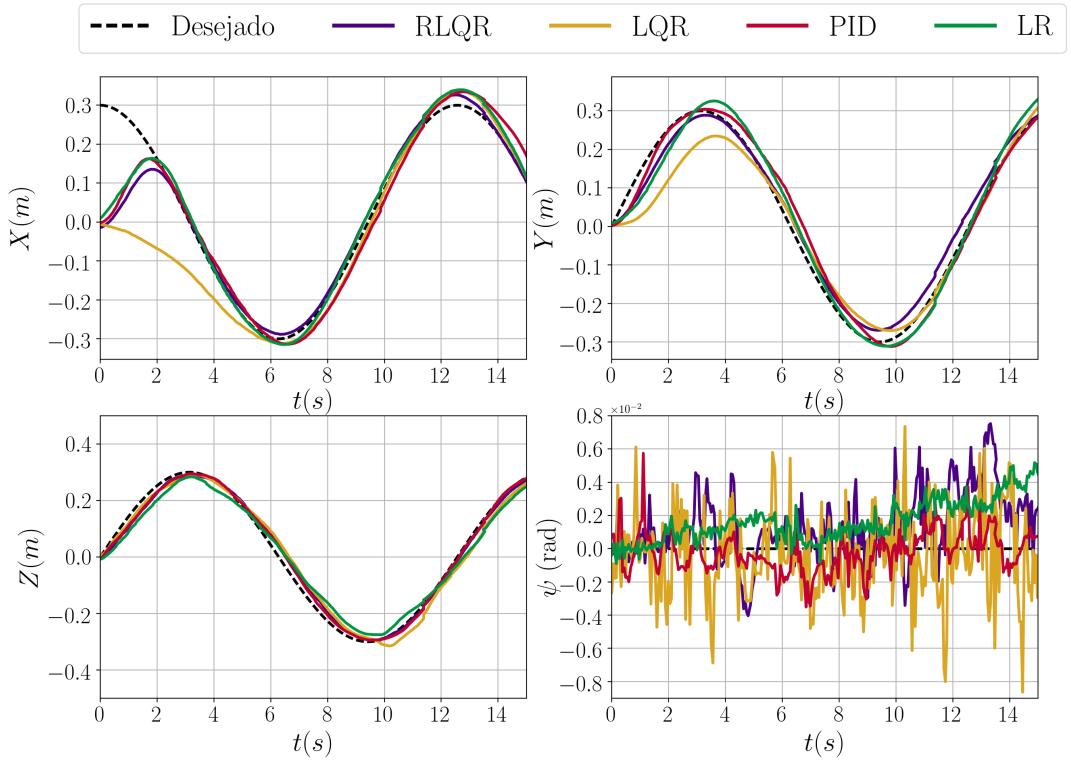
Com o objetivo de analisar quantitativamente a arquitetura proposta, a Tabela 17 apresenta os percentuais de melhoria de cada controlador em conjunto com a DNN em relação aos controladores isolados. Diante disso, é possível verificar que todos os controladores melhoraram seus desempenhos na tarefa de rastreamento de trajetória. Até mesmo os controladores RLQR e LQR que já haviam demonstrado bons desempenhos, tiveram

Figura 43 – Rastreamento de trajetória circular em 3D, utilizando a rede neural como referência para os controladores.



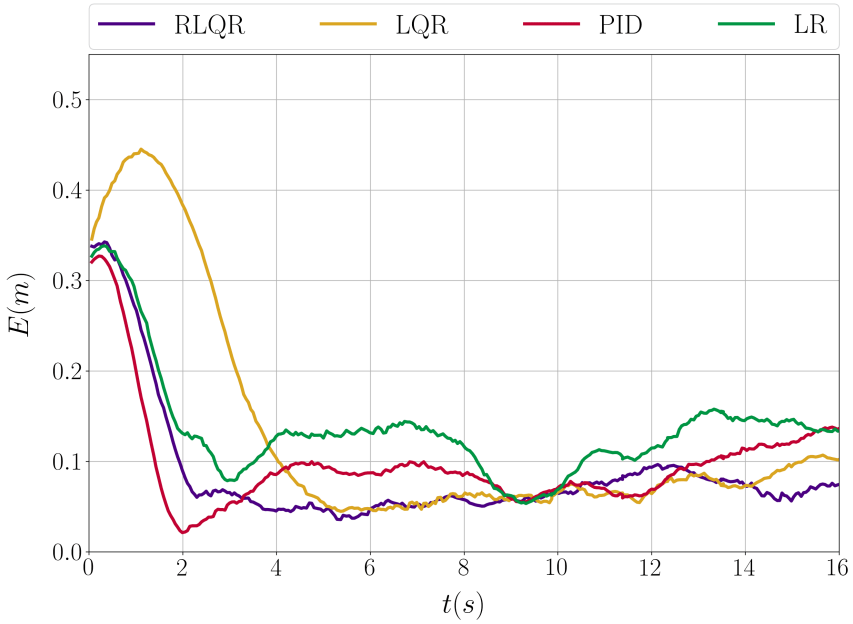
Fonte: Elaborada pelo autor.

Figura 44 – Variáveis de posição controladas, utilizando a rede neural como referência para os controladores.



Fonte: Elaborada pelo autor.

Figura 45 – Evolução do erro de posição ao longo do tempo, utilizando a rede neural como referência para os controladores.



Fonte: Elaborada pelo autor.

melhoras significativas. Ademais, diferentemente dos experimentos no ambiente simulado, em que o controlador PID obteve o maior percentual de melhoria, nos experimentos práticos foi o controlador LR que se destacou nesse requisito. Neste caso, obteve o maior percentual de melhoria com $17,98\%$, ou seja, utilizando a arquitetura 1, foi o controlador que teve o maior aumento de desempenho quando comparado à sua versão isolada.

Tabela 17 – Índices de desempenho para voos sem aplicação de distúrbio de vento, utilizando a rede neural como referência para os controladores.

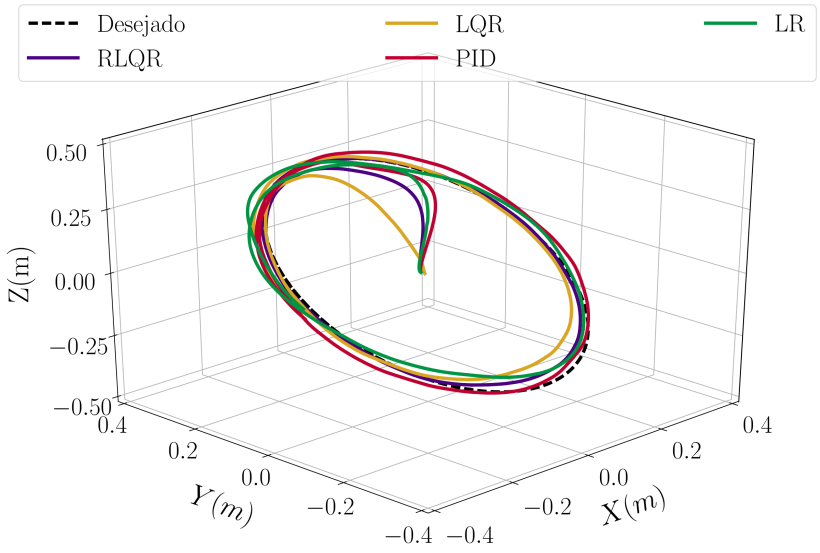
Métricas	**Rede neural como gerador de referência**			
	RLQR	**LQR**	**PID**	**LR**
Erro de Posição [m]	0,0646	0,0859	0,1182	0,1218
Erro Máximo após 3s [m]	0,0967	0,1071	0,1472	0,1579
Melhoria [%]	$11,99\%$	$13,32\%$	$6,93\%$	$17,98\%$

As Figuras 46, 47 e 48 apresentam os resultados de voos utilizando a arquitetura de controle 2 proposta, em que a rede neural atua como estimador de distúrbio. Para este caso, a vista 3D do rastreamento de trajetória é apresentada na Figura 46 e as variáveis controladas (x, y, z, ψ) são apresentadas na Figura 47. A evolução do erro é exibida na Figura 48, sendo complementada pelos índices de desempenho apresentados na Tabela 18.

Analisando as curvas de seguimento de trajetória apresentadas na Figura 46 e os índices de desempenho apresentados na Tabela 18, percebe-se que os controladores PID e

LR tiveram resultados similares, sendo o primeiro com o menor desempenho. Além disso, ambos tiveram melhoras significativas quando comparados às suas versões isoladas. Com relação ao controlador RLQR, é possível perceber que manteve o melhor desempenho entre os controladores, principalmente pelo rápido decaimento do erro de posição. Além disso, é possível verificar o menor erro máximo por parte do RLQR após decorridos $3s$ de voo. Já o controlador LQR teve o maior tempo de convergência, entretanto, ao longo da trajetória, demonstrou desempenho próximo ao do controlador RLQR.

Figura 46 – Rastreamento de trajetória circular em 3D, utilizando controle baseado em observação de distúrbio com rede neural atuando como estimador de distúrbio.



Fonte: Elaborada pelo autor.

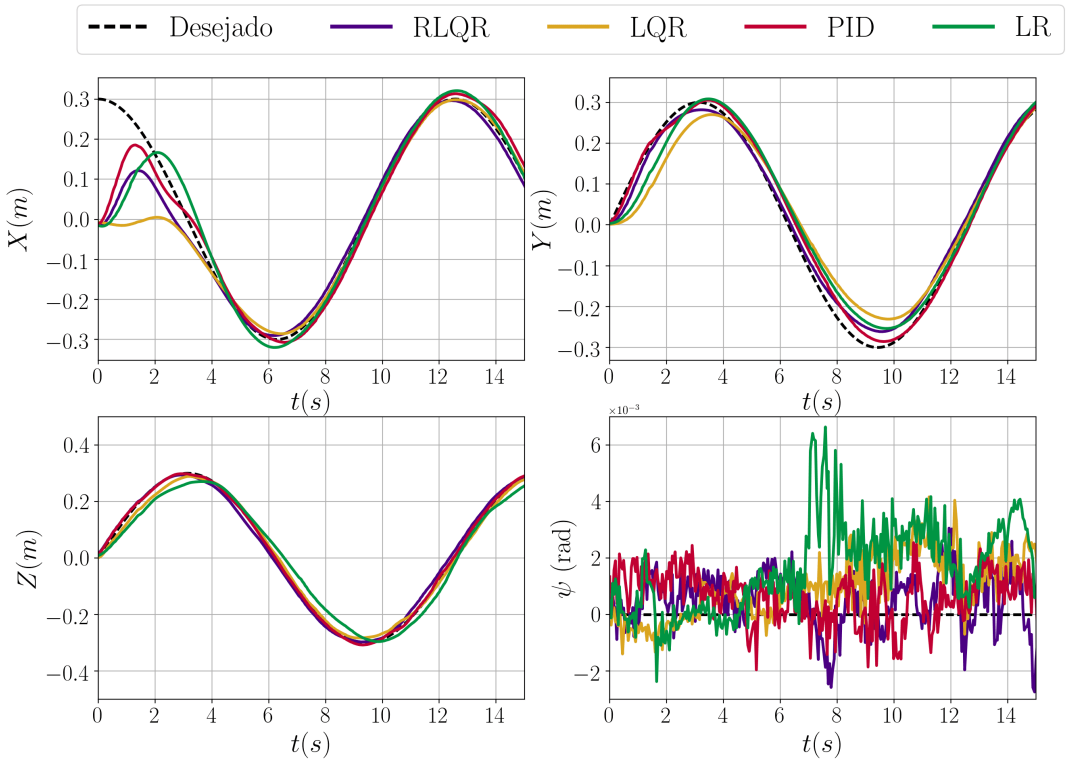
Tabela 18 – Índices de desempenho para voos sem aplicação de distúrbio de vento, utilizando controle baseado em observação de distúrbio com rede neural atuando como estimador de distúrbio.

Métricas	**Rede neural como estimador de distúrbio**			
	RLQR	**LQR**	**PID**	**LR**
Erro de Posição [m]	0,0563	0,0797	0,1071	0,1021
Erro Máximo após 3s[m]	0,0849	0,1226	0,1415	0,1571
Melhoria [%]	23, 29%	19, 58%	15, 67%	31, 24%

Com os resultados apresentados, nota-se que a rede neural atuando como estimador de distúrbio foi capaz de melhorar o desempenho de todos os controladores implementados.

Analisando a Tabela 18, pode-se observar os percentuais de melhoria para cada um deles, sendo o controlador LR, o que conseguiu o maior aumento de desempenho, com $31,24\%$. Já o controlador RLQR, que já havia demonstrado um ótimo desempenho quando utilizado de forma isolada, obteve uma melhoria de $23,29\%$.

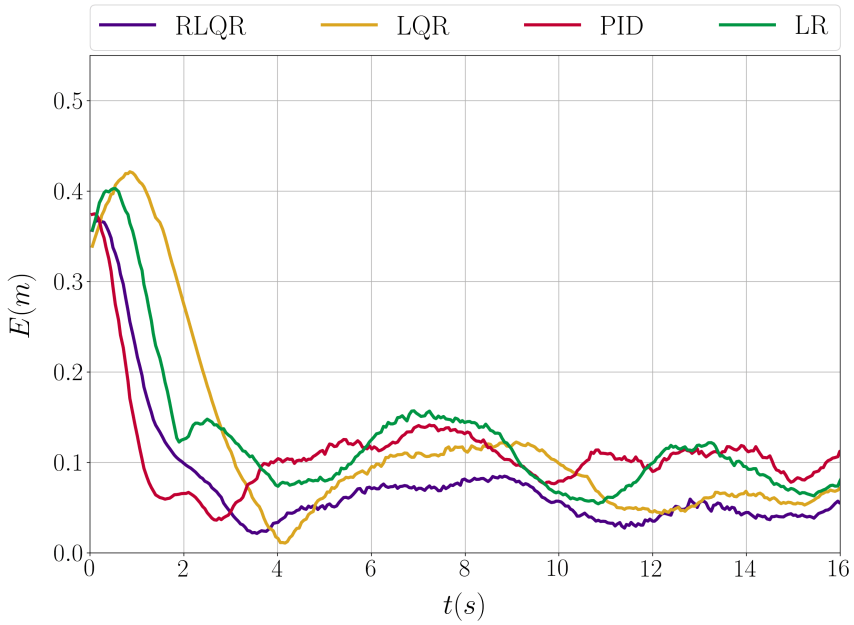
Figura 47 – Variáveis de posição controladas, utilizando controle baseado em observação de distúrbio com rede neural atuando como estimador de distúrbio.



Fonte: Elaborada pelo autor.

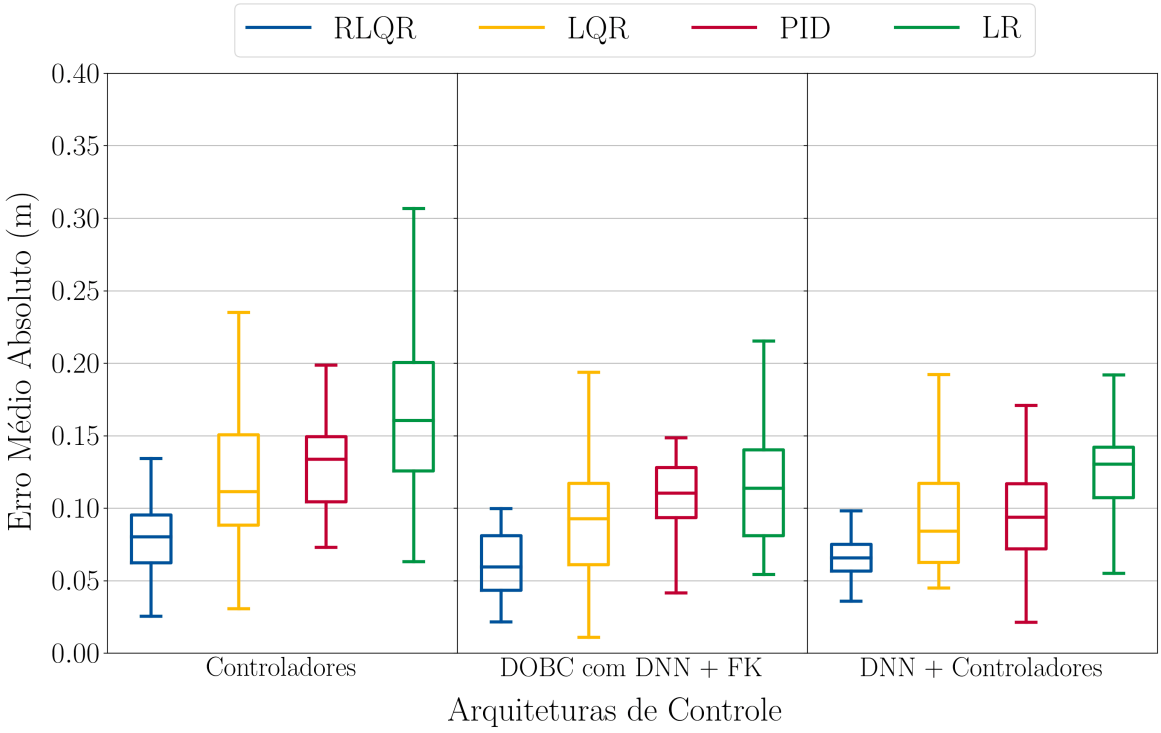
Com o intuito de analisar o desempenho dos controladores para os três casos apresentados acima, a Figura 49 apresenta o desempenho de cada uma das arquiteturas de controle propostas. Com isso, é possível perceber que ambas as arquiteturas tiveram desempenho superior quando comparadas à utilização dos controladores isolados. Além disso, é possível verificar desempenhos próximos com o controlador LQR para as duas arquiteturas propostas neste trabalho. Sendo que, de acordo com a Tabela 19, a arquitetura 2 demonstrou um melhor desempenho para todos os controladores implementados. Vale lembrar que a arquitetura de controle 1 corresponde à rede neural que atua como gerador de referência para os controladores. Já a arquitetura de controle 2 refere-se à rede neural que atua como estimador de distúrbio, formando uma arquitetura DOBC.

Figura 48 – Evolução do erro de posição ao longo do tempo, utilizando controle baseado em observação de distúrbio com rede neural atuando como estimador de distúrbio.



Fonte: Elaborada pelo autor.

Figura 49 – Boxplot do desempenho de cada controlador para os casos em que não há auxílio da rede neural profunda, à esquerda; que há auxílio da rede, atuando como estimador de distúrbio, no centro; e que há auxílio da rede, atuando como gerador de referência para os controladores, à direita.



Fonte: Elaborada pelo autor.

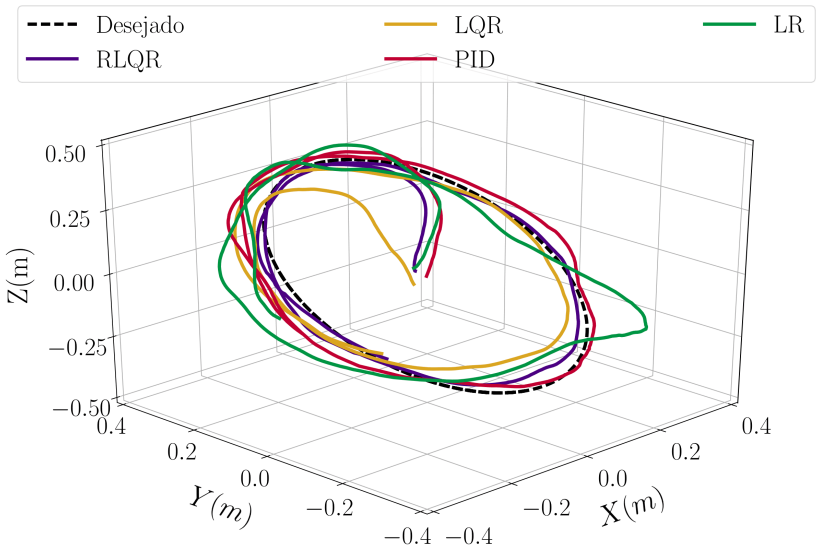
Tabela 19 – Percentual de melhoria das arquiteturas de controle propostas em relação aos controladores isolados.

Arquiteturas de Controle	**Percentual de Melhoria relativa ao uso isolado ($\overline{E}$)**			
	RLQR	**LQR**	**PID**	**LR**
Arquitetura de controle 1	$11,99\%$	$13,32\%$	$6,93\%$	$17,98\%$
Arquitetura de controle 2	$23,29\%$	$19,58\%$	$15,67\%$	$31,24\%$

7.2.2 Experimentos com Aplicação de Distúrbios de Vento

Nesta seção, são apresentados os resultados de rastreamento de trajetória quando o quadricóptero é afetado por distúrbio de vento. Assim como foi apresentado para condições normais de voo, inicialmente, os resultados de voos experimentais são discutidos para cada uma das arquiteturas propostas e, em seguida, é realizada uma comparação entre o desempenho das arquiteturas desenvolvidas. Vale lembrar que o distúrbio de vento utilizado no ambiente experimental montado foi apresentado na subseção 6.1.3.2, a partir da Figura 13.

Figura 50 – Rastreamento de trajetória circular em 3D para o caso em que quadricóptero é afetado por distúrbio de vento.



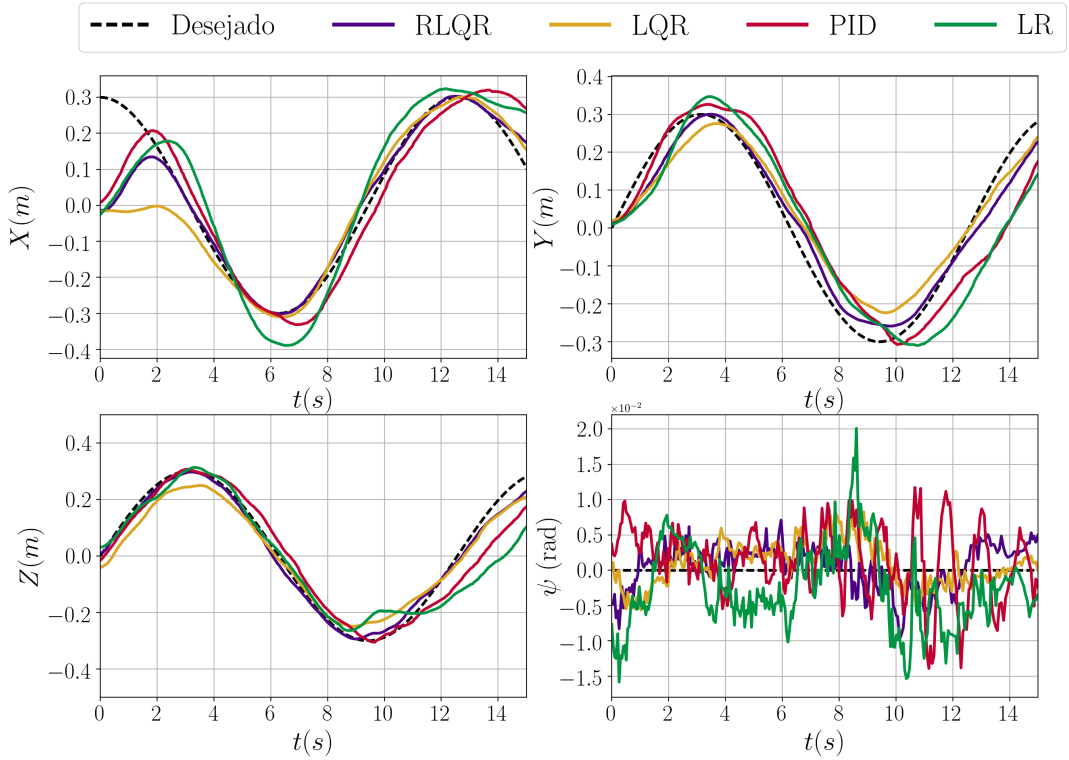
Fonte: Elaborada pelo autor.

As Figuras 50, 51 e 52 apresentam os resultados de voos reais utilizando os controladores isolados para o caso em que a aeronave está sob influência de distúrbio de vento. Diante disso, a vista 3D do rastreamento de trajetória é apresentada na Figura 50 e as variáveis controladas (x, y, z, ψ) são apresentadas na Figura 51. A evolução do erro

para este caso é exposta na Figura 52, sendo complementada pelos índices de desempenho apresentados na Tabela 20.

Para este caso, é possível perceber que, entre os controladores implementados, o LR obteve o maior erro de rastreamento de trajetória, ficando com desempenho bem inferior em relação aos demais controladores. Além disso, foi responsável pelo maior erro máximo após decorridos $3s$ de voo. Por outro lado, o controlador RLQR apresentou o melhor desempenho na tarefa de rastreamento de trajetória, com um erro de posição consideravelmente inferior quando comparado aos demais controladores. Além disso, apresentou o menor erro máximo entre os controladores implementados.

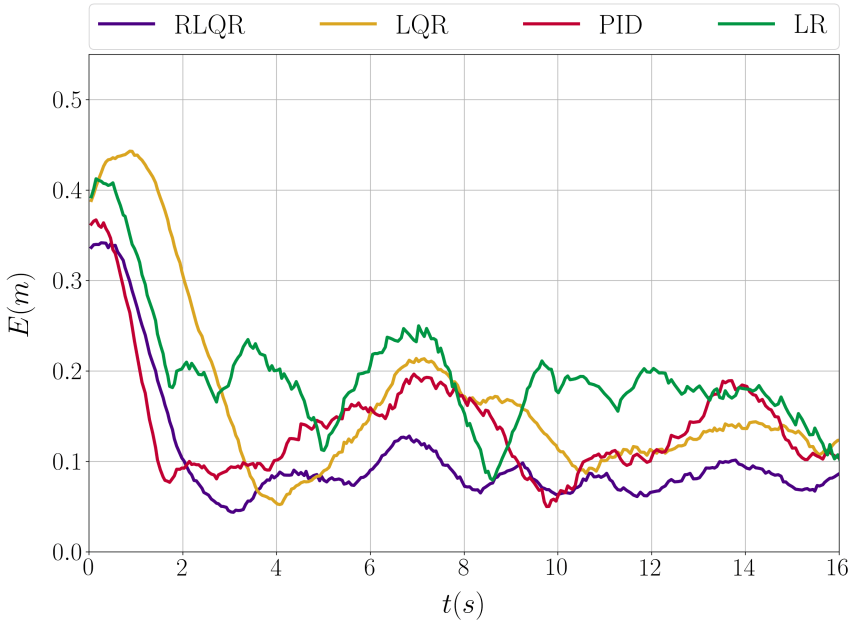
Figura 51 – Variáveis de posição controladas para o caso em que quadricóptero é afetado por distúrbio de vento.



Fonte: Elaborada pelo autor.

De acordo com a Figura 52, pode-se perceber, ainda, que os controladores LQR e PID tiveram desempenhos bem similares. Sendo o segundo com desempenho superior, sobretudo pela rápida convergência para a trajetória desejada. A partir dos índices numéricos apresentados na Tabela 12, é possível consolidar o desempenho global similar entre o LQR e o controlador PID. Além disso, verifica-se também o desempenho superior do controlador RLQR em relação aos demais controladores. Neste caso, o desempenho superior foi obtido tanto com relação ao erro de posição quanto ao erro máximo após a convergência para a trajetória desejada.

Figura 52 – Evolução do erro de posição ao longo do tempo para cada controlador implementado para o caso em que quadricóptero é afetado por distúrbio de vento.



Fonte: Elaborada pelo autor.

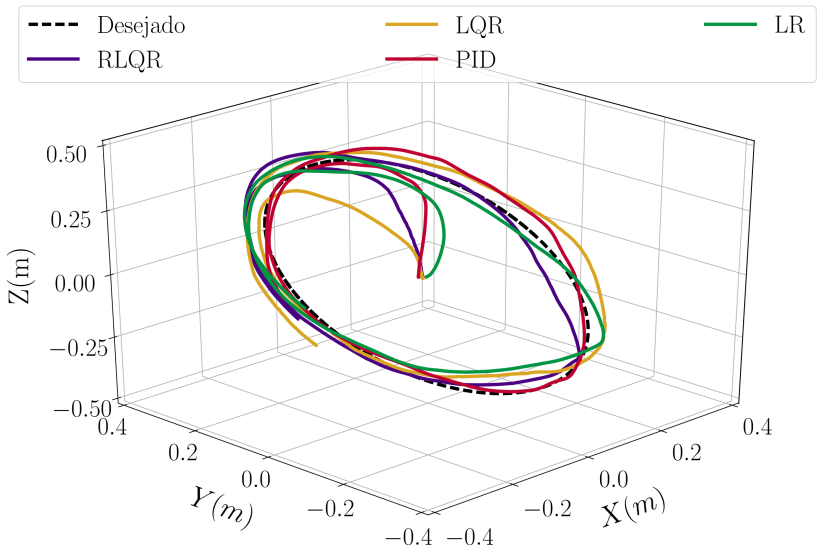
Tabela 20 – Índices de desempenho para voos com aplicação de distúrbio de vento.

Métricas	**Controladores Isolados**			
	RLQR	**LQR**	**PID**	**LR**
Erro de Posição [m]	0,0842	0,1344	0,1337	0,1712
Erro Máximo após 3s[m]	0,1282	0,2139	0,1967	0,2500

As curvas apresentadas nas Figuras 53, 54 e 55 representam os resultados de voos reais utilizando a arquitetura 1, em que a DNN é responsável por auxiliar o controlador durante o seguimento de trajetória. Diante disso, a vista 3D é apresentada na Figura 53 e as variáveis controladas (x, y, z, ψ) são apresentadas na Figura 54. Para este caso, a evolução do erro é evidenciada na Figura 55 e os índices numéricos de desempenho são apresentados na Tabela 21.

Analisando os resultados apresentados para a arquitetura 1 proposta, pode-se observar que, assim como para o primeiro caso analisado (controladores isolados), o RLQR obteve tanto o menor erro de rastreamento de trajetória como o menor erro máximo após decorridos $3s$ de voo. Além disso, é possível visualizar novamente um desempenho similar entre o LQR e o PID, sendo que, neste caso, o controlador LQR foi ligeiramente melhor que o controlador PID. Como esperado, o desempenho do controlador LR foi consideravelmente menor que os demais controladores, principalmente com relação ao controlador RLQR. Note que de acordo com Figura 55, os controladores RLQR, PID e LR

Figura 53 – Rastreamento de trajetória circular em 3D, utilizando a rede neural como referência para os controladores para o caso em que o quadricóptero é afetado por distúrbio de vento.



Fonte: Elaborada pelo autor.

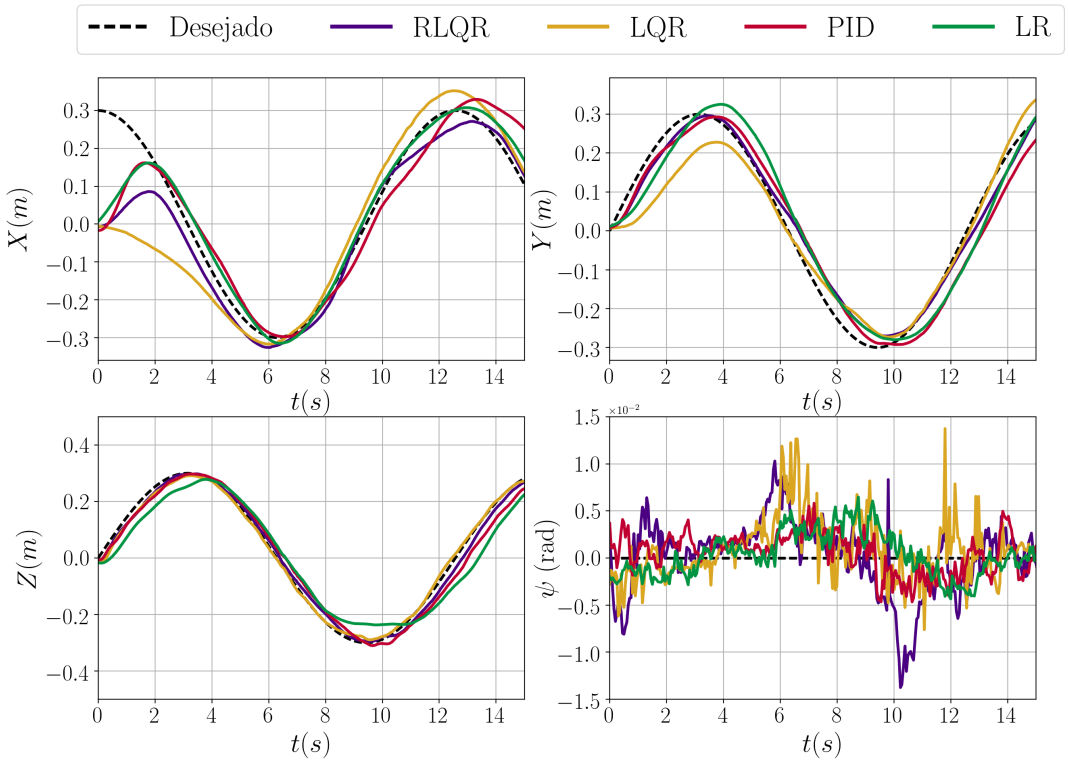
apresentaram tempos de convergência similares para a trajetória desejada, enquanto que o LQR, novamente, apresentou um tempo de convergência maior.

Tabela 21 – Índices de desempenho, utilizando a rede neural como referência para os controladores quando o quadricóptero é afetado por distúrbio de vento.

Métricas	**Rede neural como gerador de referência**			
	RLQR	**LQR**	**PID**	**LR**
Erro de Posição [m]	0,0728	0,1115	0,1121	0,1458
Erro Máximo após 3s[m]	0,1203	0,1493	0,1820	0,2109
Melhoria [%]	13, 54%	17, 04%	16, 15%	14, 84%

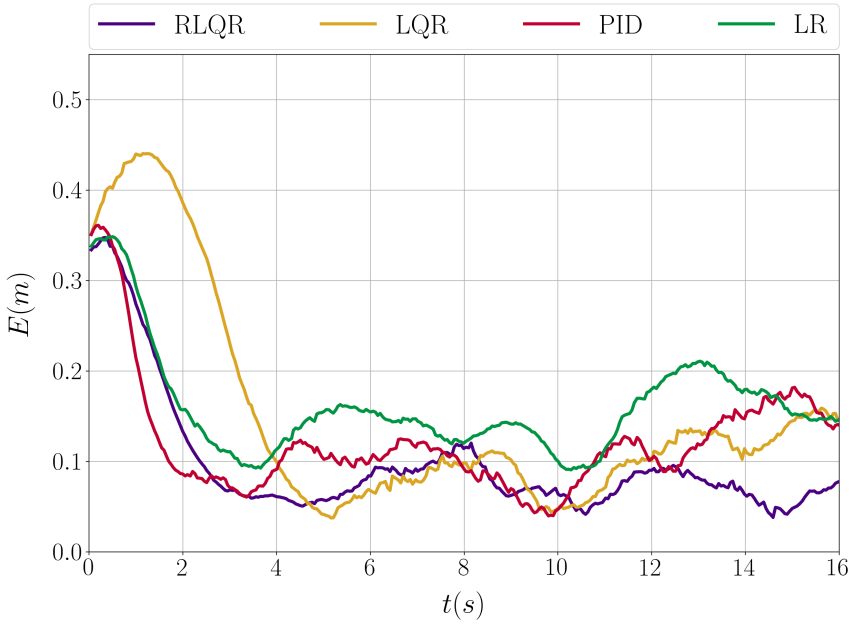
A fim de analisar a arquitetura proposta, a Tabela 21 apresenta os percentuais de melhoria de cada controlador combinado com a DNN em relação aos controladores isolados. Em vista disso, verifica-se uma melhora por parte de todos os controladores na tarefa de rastreamento de trajetória. Com a arquitetura 1, foi possível melhorar ainda mais o desempenho do controlador RLQR, que já havia demonstrado um excelente desempenho de forma isolada. Em relação aos percentuais de melhorias, nota-se que o controlador LQR obteve o maior percentual entre os controladores avaliados, com 17, 04%. Sendo capaz de realizar o seguimento de trajetória com um erro bem abaixo de sua versão isolada.

Figura 54 – Variáveis de posição controladas, utilizando a rede neural como referência para os controladores para o caso em que o quadricóptero é afetado por distúrbio de vento.



Fonte: Elaborada pelo autor.

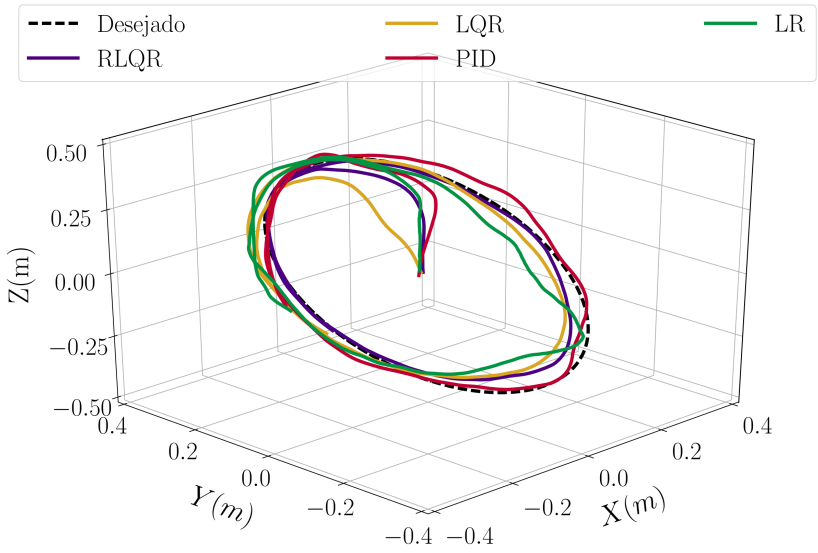
Figura 55 – Evolução do erro de posição ao longo do tempo para cada controlador implementado, utilizando a rede neural como referência para os controladores para o caso em que o quadricóptero é afetado por distúrbio de vento.



Fonte: Elaborada pelo autor.

A seguir, as Figuras 56, 57, 58 apresentam os resultados de voos reais utilizando a arquitetura DOBC com rede neural atuando como estimador de distúrbio. A Figura 56 apresenta a vista 3D do rastreamento de trajetória, seguida da Figura 57, que apresenta as variáveis controladas (x, y, z, ψ). A evolução do erro é exibida na Figura 58 e, para uma análise quantitativa, a Tabela 22 apresenta os índices de desempenho avaliados.

Figura 56 – Rastreamento de trajetória circular em 3D para o caso em que o quadricóptero é afetado por distúrbio de vento, utilizando controle baseado em observação de distúrbio com rede neural atuando como estimador.



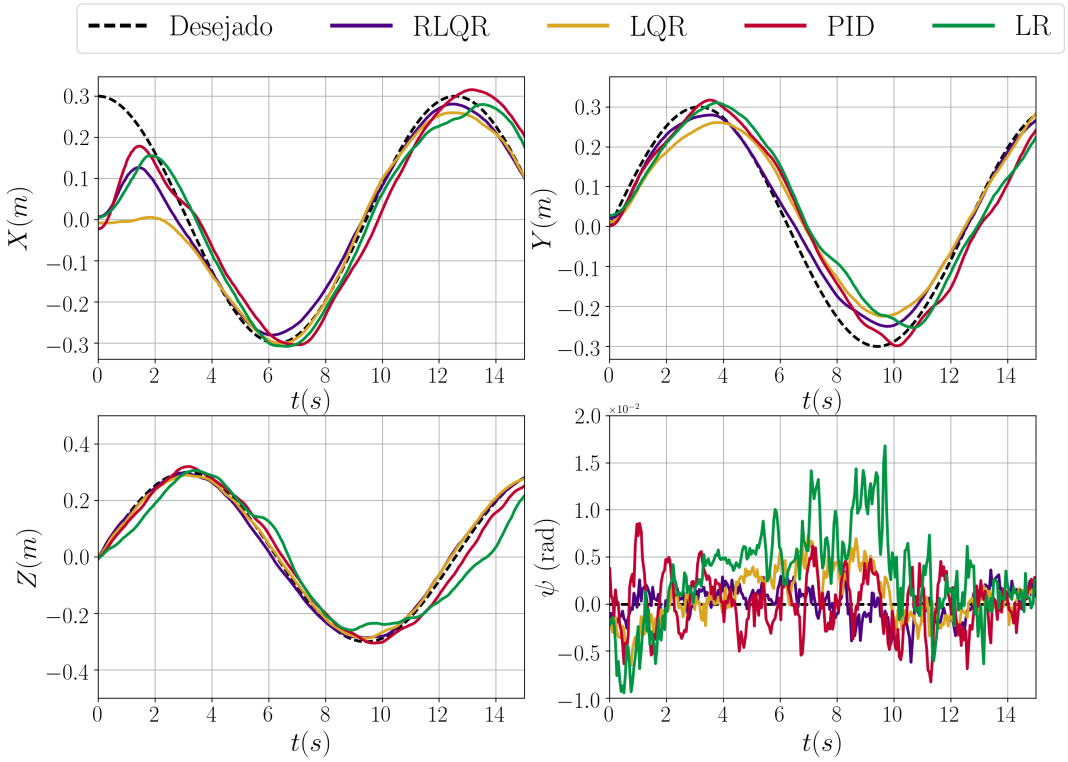
Fonte: Elaborada pelo autor.

Analisando as curvas de rastreamento de trajetória apresentadas na Figura 56 e os índices de desempenho apresentados na Tabela 22, percebe-se que, novamente, o controlador LR demonstrou o menor desempenho entre os controladores avaliados. Como esperado, o controlador RLQR apresentou o melhor desempenho, mantendo o quadricóptero próximo à trajetória durante todo o tempo do experimento. Além disso, apresentou tempo de convergência similar ao do controlador PID e o menor erro máximo após decorridos $3s$ de voo, contribuindo para o bom desempenho do controlador.

Mais uma vez os resultados globais demonstraram uma proximidade entre os controladores LQR e PID para os experimentos práticos com distúrbio de vento. Entretanto, com a arquitetura 2 proposta, o controlador LQR convergiu mais rapidamente para a trajetória desejada quando comparado a sua versão isolada e à arquitetura 1. Já o controlador PID, com a configuração de ganho adotada, continua sendo o controlador que converge mais rapidamente para a trajetória desejada quando comparado aos demais controladores.

Com relação à arquitetura DOBC, em que a rede atua como estimador de distúrbio (arquitetura 2), assim como para o caso simulado, os resultados mostraram que foi capaz de melhorar o desempenho de todos os controladores. Dessa forma, reduziu o erro de posição e o erro máximo dos controladores avaliados. A partir da Tabela 22, pode-se observar percentuais de melhoria em torno de 18% a 20% para cada um dos controladores.

Figura 57 – Variáveis de posição controladas para o caso em que o quadricóptero é afetado por distúrbio de vento, utilizando controle baseado em observação de distúrbio com rede neural atuando como estimador.

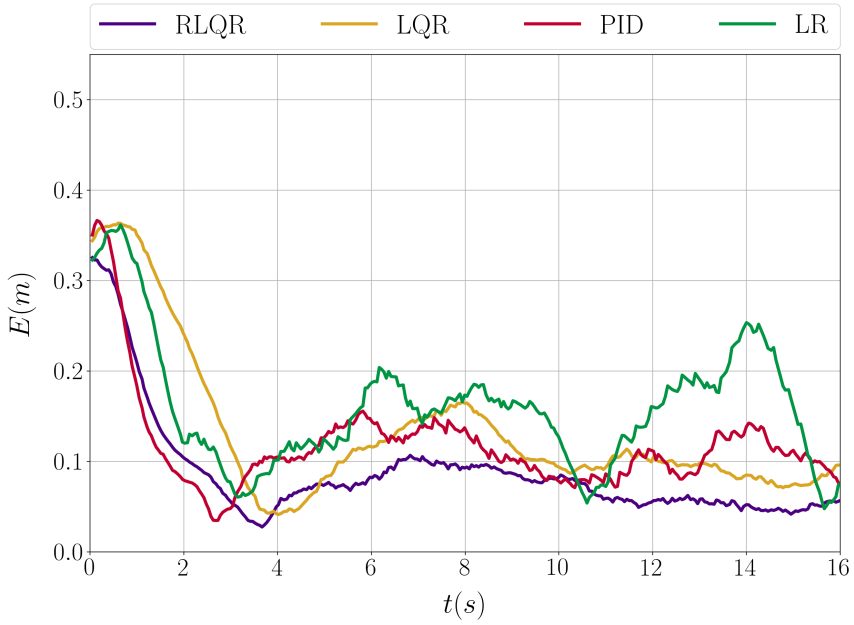


Fonte: Elaborada pelo autor.

Tabela 22 – Índices de desempenho para o caso em que o quadricóptero é afetado por distúrbio de vento, utilizando controle baseado em observação de distúrbio com rede neural atuando como estimador.

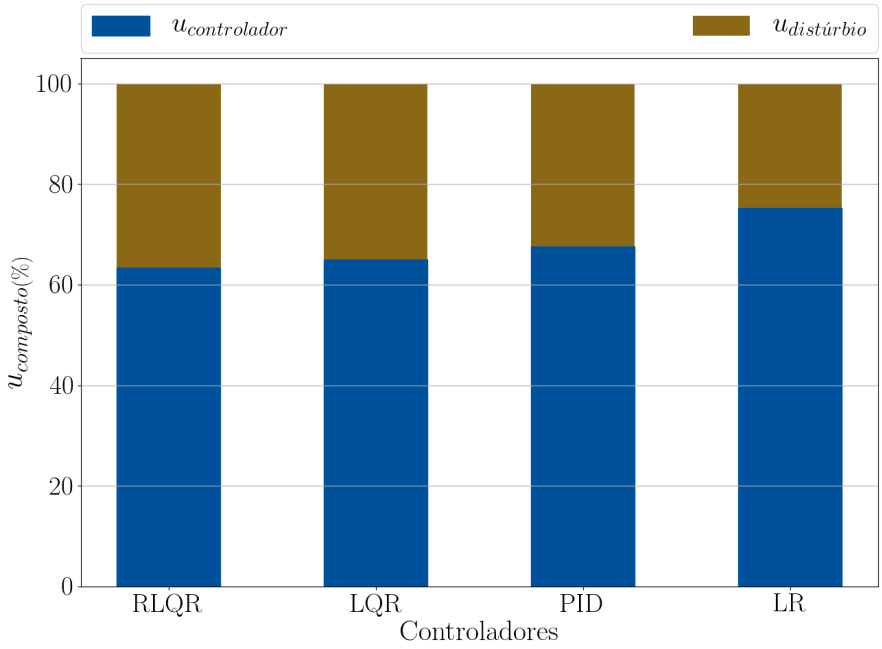
Métricas	**Rede neural como estimador de distúrbio**			
	RLQR	**LQR**	**PID**	**LR**
Erro de Posição [m]	0,0679	0,1062	0,1089	0,1401
Composição de u_{com} [%]	u_{rlqr} : 63, 81 u_{dist} : 36, 19	u_{lqr} : 65, 01 u_{dist} : 34, 99	u_{pid} : 67, 65 u_{dist} : 32, 35	u_{fl} : 75, 31 u_{dist} : 22, 06
Erro Máximo após 3s[m]	0,1069	0,1659	0,1557	0,2535
Melhoria [%]	19, 36%	20, 98%	18, 55%	18, 17%

Figura 58 – Evolução do erro de posição ao longo do tempo para o caso em que o quadricóptero é afetado por distúrbio de vento, utilizando controle baseado em observação de distúrbio com rede neural atuando como estimador.



Fonte: Elaborada pelo autor.

Figura 59 – Composição da lei de controle baseado em observação de distúrbio para cada controlador implementado.



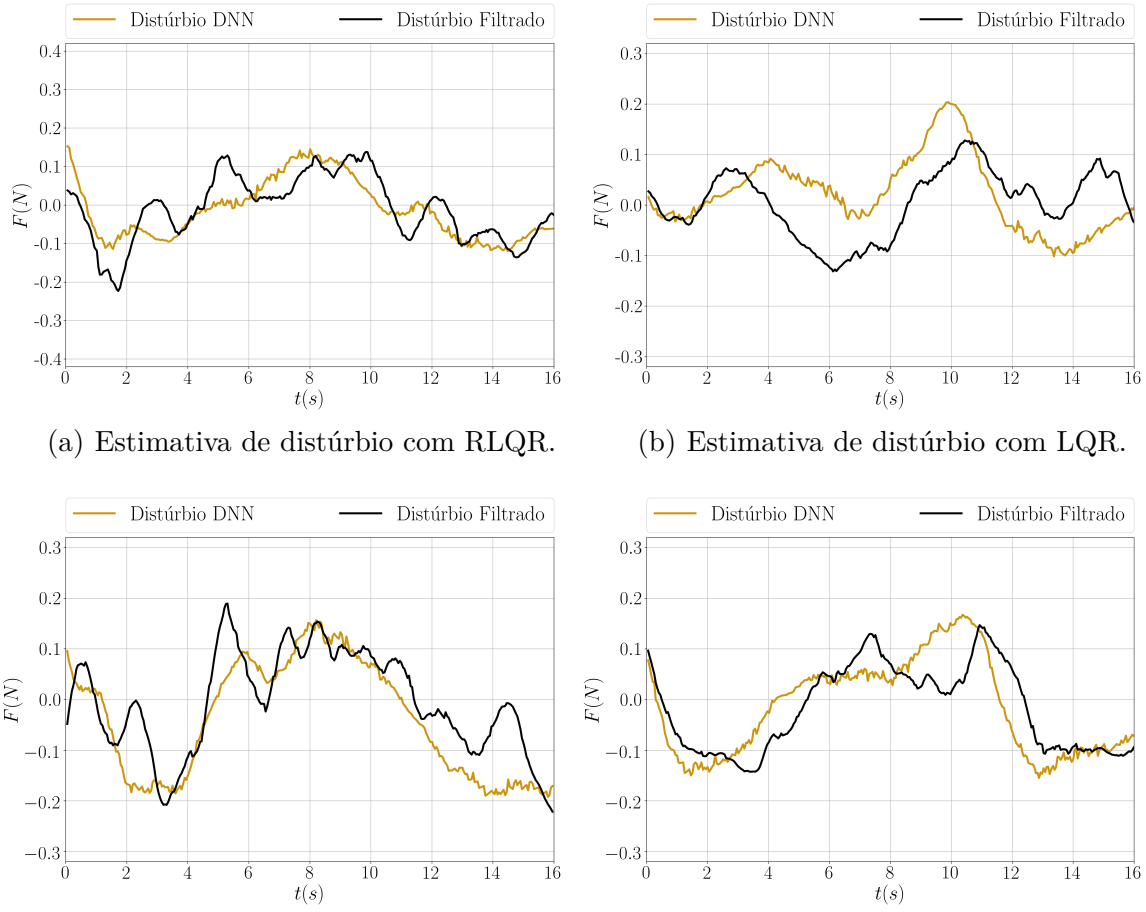
Fonte: Elaborada pelo autor.

De forma similar ao apresentado para os resultados simulados, Figura 59 em conjunto com a Tabela 22 apresentam a composição do sinal de controle. Dessa forma, pode-se observar os índices numéricos referentes a lei de controle composta para cada

um dos controladores. Note que pra cada controlador, a composição da lei de controle é diferente. Sendo, neste caso, o RLQR com maior influência da lei de controle proveniente do distúrbio, seguido pelo LQR e o PID. Já o controlador LR foi foi o que menos sofreu influência da lei de controle do distúrbio, com $22,06\%$. Ademais, vale lembrar que, para os resultados utilizando o simulador *Parrot-Sphinx*, o controlador RLQR foi que menos sofreu influência da lei de controle do distúrbio. Este fato pode ser explicado pelo uso de diferentes ganhos para o compensador utilizado.

A Figura 60 apresenta as estimativas do distúrbio em *Newtons* fornecida pela DNN e o sinal filtrado pelo filtro de Kalman para cada um dos controladores implementados. O distúrbio foi estimado ao longo do eixo Y do quadricóptero durante todo o tempo de trajetória. Vale lembrar que essas estimativas do distúrbio de vento são utilizadas para criação da lei de controle (u_{dist}) que é somada à lei de controle proveniente do controlador (u_{ctrl}) para formação da lei de controle composta (u_{com}).

Figura 60 – Estimativas de distúrbio da Rede Neural e sinal filtrado pelo Filtro de Kalman para cada um dos controladores implementados.



(a) Estimativa de distúrbio com RLQR.

(b) Estimativa de distúrbio com LQR.

(c) Estimativa de distúrbio com PID.

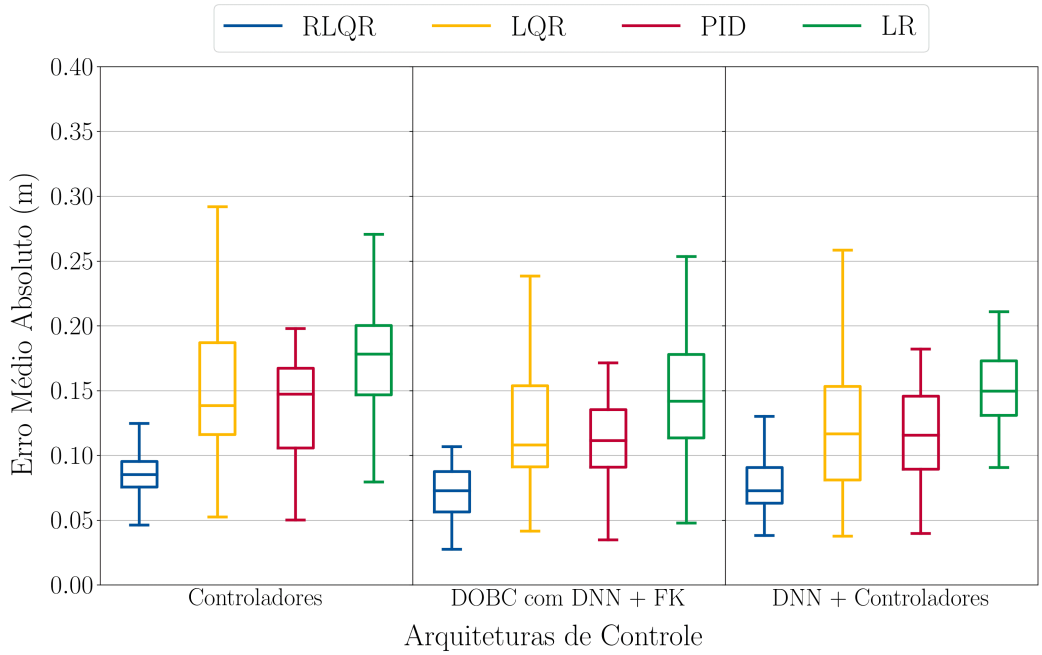
(d) Estimativa de distúrbio com LR.

Fonte: Elaborada pelo autor.

Com o objetivo de analisar o desempenho das arquiteturas propostas para cada con-

trolador implementado, a Figura 61 apresenta o boxplot para comparação do desempenho de rastreamento. Com isso, de forma similar aos resultados utilizando simulador, é possível perceber que ambas as arquiteturas tiveram desempenho superior quando comparadas com à utilização dos controladores isolados.

Figura 61 – Boxplot do desempenho de cada controlador com aplicação de distúrbio de vento para os casos em que não há auxílio da rede neural profunda, à esquerda; que há auxílio da rede, atuando como estimador de distúrbio, no centro; e que há auxílio da rede, atuando como gerador de referência para os controladores, à direita.



Fonte: Elaborada pelo autor.

Tabela 23 – Percentual de melhoria para as duas arquiteturas de controle propostas em relação aos controladores isolados (experimentos práticos).

Arquiteturas de Controle	**Percentual de Melhoria relativa ao uso isolado ($\overline{E}$)**			
	RLQR	**LQR**	**PID**	**LR**
Arquitetura de Controle 1	$13,54\%$	$17,04\%$	$16,15\%$	$14,84\%$
Arquitetura de Controle 2	$19,36\%$	$20,98\%$	$18,55\%$	$18,17\%$

Ademais, como a arquitetura 2 é voltada à estimativa e atenuação de distúrbios externos, é possível visualizar um desempenho superior para todos os controladores implementados com essa arquitetura. Para uma análise quantitativa, a Tabela 23 apresenta o percentual de melhoria das duas arquiteturas de controle propostas em relação aos controladores isolados. Dessa forma, é possível confirmar o melhor desempenho da arquitetura 2 para o caso que o quadricóptero é afetado por distúrbio de vento.

8 CONCLUSÃO

Esta dissertação de mestrado propôs duas arquiteturas robustas e inteligentes para controle de posição de um quadricóptero na tarefa de rastreamento de trajetória. As duas arquiteturas propostas são formadas por um controlador RLQR e por DNNs que atuam tanto para fornecer um sinal de referência ao controlador (arquitetura 1) quanto para estimar distúrbios de vento que afetam o quadricóptero (arquitetura 2). Além disso, outros três controladores foram utilizados dentro da estrutura proposta para efeito de comparação, são eles: LQR, PID e LR. Dessa forma, foi possível avaliar não somente o desempenho entre os controladores implementados como também o desempenho das arquiteturas com os diferentes controladores.

Todo o *framework* foi desenvolvido e integrado através do sistema ROS. Com isso, foi possível utilizar as competências e bibliotecas de diferentes linguagens de programação, como *Python* e C++, para implementação dos controladores e das redes neurais. Ademais, o sistema ROS possibilitou realizar experimentos utilizando tanto o sistema de odometria do quadricóptero quanto um sistema de captura de movimentos amplamente utilizado em robótica. Para validação das arquiteturas propostas, foram realizados experimentos utilizando um simulador 3D realista baseado no Gazebo do ROS. Posteriormente, os experimentos foram realizados com um quadricóptero comercial, o Parrot^{TM} Bebop 2.0. Em ambos os casos foram realizados dois conjuntos de testes, para condições normais de voo e para voos em que o quadricóptero foi submetido a distúrbios de vento.

Os resultados apresentaram o desempenho das arquiteturas propostas tanto de forma qualitativa quanto quantitativa. Diante disso, foi possível observar que a utilização dos controladores com as DNNs propostas forneceram uma melhoria de desempenho para o quadricóptero na tarefa de rastreamento de trajetória. Isto pôde ser confirmado através dos experimentos no ambiente simulado e com os experimentos práticos. Além disso, é possível notar que a melhoria de desempenho foi alcançada para os dois conjuntos de experimentos, com e sem a aplicação de distúrbios de vento. Entretanto, como esperado, para os casos em que o quadricóptero ficou sob influência do distúrbio, a arquitetura de controle DOBC composta pela rede neural como estimador de distúrbio alcançou um melhor resultado. Isto pode ser explicado pelo fato de que esta arquitetura é voltada exclusivamente para atenuar e compensar distúrbios externos.

Trabalhos futuros consistirão em validar as arquiteturas de controle propostas através de voos em ambientes externos. Além disso, devido à arquitetura modular desenvolvida, será possível testar outros controladores com facilidade, como o H_∞ e o MPC. Outra possibilidade é a utilização de algoritmos genéticos para otimização das matrizes de incertezas do RLQR e a utilização conjunta das duas arquiteturas inteligentes propostas.

REFERÊNCIAS

ABDELMAKSOUD, S. I.; MAILAH, M.; ABDALLAH, A. M. Control Strategies and Novel Techniques for Autonomous Rotorcraft Unmanned Aerial Vehicles: A Review. **IEEE Access**, v. 8, p. 195142–195169, 2020. ISSN 2169-3536.

ADKINS, K. A. Urban flow and small unmanned aerial system operations in the built environment. **International Journal of Aviation, Aeronautics, and Aerospace**, v. 6, n. 1, 2019. ISSN 23746793.

AGGARWAL, C. C. **Neural Networks and Deep Learning**. : Springer, 2018.

AL HABSI, S. *et al.* Integration of a Vicon camera system for indoor flight of a Parrot AR Drone. **ISMA 2015 - 10th International Symposium on Mechatronics and its Applications**, IEEE, p. 1–6, 2016.

ALAIWI, Y.; MUTLU, A. Modelling, Simulation and Implementation of Autonomous Unmanned Quadrotor. **International Journal for Science, Technics and Innovations for the Industry "Machines. Technologies. Materials"**, v. 12, n. 8, p. 320–325, 2018.

ASWANI, A.; BOUFFARD, P.; TOMLIN, C. Extensions of learning-based model predictive control for real-time application to a quadrotor helicopter. *In*: **2012 American Control Conference (ACC)**. 2012. p. 4661–4666.

ASWANI, A. *et al.* Provably safe and robust learning-based model predictive control. **Automatica**, v. 49, 07 2011.

BALASUBRAMANIAN, E.; VASANTHARAJ, R. Dynamic Modeling and Control of Quad Rotor. **International Journal of**, v. 5, n. 1, p. 63–69, 2013.

BELLAHCENE, Z. *et al.* Adaptive neural network-based robust $\mathcal{H}_\infty$ tracking control of a quadrotor UAV under wind disturbances. **International Journal of Automation and Control (IJAAC)**, v. 15, p. 28–57, 2021. ISSN 10990526.

BENEVIDES, J. R. *et al.* ROS-based robust and recursive optimal control of commercial quadrotors. **IEEE International Conference on Automation Science and Engineering**, v. 2019-Augus, p. 998–1003, 2019.

BENEVIDES, J. R. S. *et al.* Parameter Estimation Based on Linear Regression for Commercial Quadrotors. **Anais do 14º Simpósio Brasileiro de Automação Inteligente**, 2019.

BENEVIDES, J. R. S. *et al.* Disturbance Observer-based Robust Control of a Quadrotor Subject to Parametric Uncertainties and Wind Disturbance. **IEEE Access (Manuscript submitted for publication.)**, 2021.

BISHOP, C. **Pattern Recognition and Machine Learning**. : Springer, 2006. (Information Science and Statistics). ISBN 9780387310732.

BONNA, R.; CAMINO, J. Trajectory Tracking Control of a Quadrotor Using Feedback Linearization. **Proceedings of the XVII International Symposium on Dynamic Problems of Mechanics**, 2015.

BOUABDALLAH, S.; NOTH, A.; SIEGWART, R. PID vs LQ control techniques applied to an indoor micro Quadrotor. **2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**, v. 3, p. 2451–2456, 2004.

BROCKETT, R. Feedback Invariants for Nonlinear Systems. **IFAC Proceedings Volumes**, Elsevier, v. 11, n. 1, p. 1115–1120, 1978. ISSN 14746670.

CERRI, J. P. **Regulador Robusto Recursivo para Sistemas Lineares de Tempo Discreto no Espaço de Estado**. 2009. Tese (Dissertation) — Universidade de São Paulo, 2009.

CHEN, M.; HUZMEZAN, M. A combined mbpc/2 dof H_∞ controller for a quad rotor uav. *In*: . 2003.

CHEN, W. H. *et al.* A nonlinear disturbance observer for robotic manipulators. **IEEE Transactions on Industrial Electronics**, v. 47, n. 4, p. 932–938, 2000.

CHEN, W.-H. *et al.* Disturbance-Observer-Based Control and Related Methods - An Overview. **IEEE Transactions on Industrial Electronics**, IEEE, v. 63, n. 2, p. 1083–1095, 2016.

CHOLLET, F. **Deep Learning with Python**. : Manning Publications Company, 2017. ISBN 9781617294433.

COHEN, M. R.; ABDULRAHIM, K.; FORBES, J. R. Finite-horizon lqr control of quadrotors on $SE_2(3)$. **IEEE Robotics and Automation Letters**, v. 5, n. 4, p. 5748–5755, 2020.

COZA, C.; MACNAB, C. A new robust adaptive-fuzzy control method applied to quadrotor helicopter stabilization. *In*: **NAFIPS 2006 - 2006 Annual Meeting of the North American Fuzzy Information Processing Society**. 2006. p. 454–458.

DIERKS, T.; JAGANNATHAN, S. Output feedback control of a quadrotor uav using neural networks. **IEEE Transactions on Neural Networks**, v. 21, n. 1, p. 50–66, 2010.

DOZAT, T. Incorporating Nesterov Momentum into Adam. **ICLR Workshop**, n. 1, p. 2013–2016, 2016.

DUCHI, J. C.; HAZAN, E.; SINGER, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. **Journal of Machine Learning Research**, v. 12, p. 2121–2159, 2011. ISSN 01912216.

DUGGAL, V. *et al.* Plantation monitoring and yield estimation using autonomous quadcopter for precision agriculture. *In*: IEEE. **International Conference on Robotics and Automation (ICRA)**. 2016. p. 5121–5127.

FREIDOVICH, L. B.; KHALIL, H. K. Performance recovery of feedback-linearization-based designs. **IEEE Transactions on Automatic Control**, v. 53, n. 10, p. 2324–2334, 2008.

GAO, Z. On the centrality of disturbance rejection in automatic control. **ISA Transactions**, Elsevier, v. 53, n. 4, p. 850–857, 2014.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. : O'Reilly Media, 2019. ISBN 9781492032618.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. *In*: **Proceedings of the thirteenth international conference on artificial intelligence and statistics**. Sardinia, Italy: , 2010. p. 249–256.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. : The Mit Press, 2016. ISBN 0262035618.

GUO, B. Z.; ZHAO, Z. L. **Extended state observer for nonlinear systems with uncertainty**. : IFAC, 2011. v. 44. 1855–1860 p.

GUO, L.; CAO, S. Anti-disturbance control theory for systems with multiple disturbances: A survey. **ISA Transactions**, Elsevier, v. 53, n. 4, p. 846–849, 2014.

HAN, J. From PID to active disturbance rejection control. **IEEE Transactions on Industrial Electronics**, IEEE, v. 56, n. 3, p. 900–906, 2009.

HENG, X. *et al.* A trajectory tracking LQR controller for a quadrotor: Design and experimental evaluation. **IEEE Region 10 Annual International Conference, Proceedings/TENCON**, v. 2016-Janua, n. November, 2016.

HERRERA, M. *et al.* Sliding mode control: An approach to control a quadrotor. *In*: **2015 Asia-Pacific Conference on Computer Aided System Engineering**. 2015. p. 314–319.

HOSTETTER˜, G. H.; MEDITCH, J. S. On the Generalization of Observers to Systems with Unmeasurable Unknown Inputs. **Automatica**, v. 9, p. 721–724, 1973.

HUO, J.; MENG, T.; JIN, Z. Adaptive attitude control using neural network observer disturbance compensation technique. **Proceedings of 9th International Conference on Recent Advances in Space Technologies, RAST 2019**, IEEE, n. 1, p. 697–701, 2019.

JOHNSON, C. D. Optimal Control of the Linear Regulator with Constant Disturbances. **IEEE Transactions on Automatic Control**, v. 1, n. August, 1968.

KALMAN, R. E. A new approach to linear filtering and prediction problems. **Journal of Fluids Engineering, Transactions of the ASME**, v. 82, n. 1, p. 35–45, 1960.

KHOSRAVIAN, E.; MAGHSOUDI, H. Design of an intelligent controller for station keeping, attitude control, and path tracking of a quadrotor using recursive neural networks. **International Journal of Engineering**, Materials and Energy Research Center, v. 32, n. 5, p. 747–758, 2019. ISSN 1025-2495.

KIM, D. W.; PARK, C. S. Application of Kalman filter for estimating a process disturbance in a building space. **Sustainability (Switzerland)**, v. 9, n. 10, 2017.

KIM, J.; GADSDEN, S. A.; WILKERSON, S. A. A Comprehensive Survey of Control Strategies for Autonomous Quadrotors. **Canadian Journal of Electrical and Computer Engineering**, IEEE Canada, v. 43, n. 1, p. 3–16, 2020. ISSN 23318422.

KINGMA, D. P.; BA, J. L. Adam: A method for stochastic optimization. **3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings**, p. 1–15, 2014.

KWON, S. J.; CHUNG, W. K. A discrete-time design and analysis of perturbation observer for motion control applications. **IEEE Transactions on Control Systems Technology**, IEEE, v. 11, n. 3, p. 399–407, 2003.

LAZIM, I. M. *et al.* Feedback linearization with intelligent disturbance observer for autonomous quadrotor with time-varying disturbance. **International Journal of Mechanical & Mechatronics Engineering**, v. 18, p. 47–55, 10 2018.

LEÃO, W. M. **Análise Comparativa de Controladores Robustos Aplicados em Robôs Móvel e Aéreo**. 2015. Tese (Dissertação) — Universidade de São Paulo, 2015.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, 2015.

LEE, D.; KIM, H. J.; SASTRY, S. Feedback linearization vs. adaptive sliding mode control for a quadrotor helicopter. **International Journal of Control, Automation and Systems**, v. 7, n. 3, p. 419–428, 2009. ISSN 15986446.

LEE, H.; KIM, H. J. Robust control of a quadrotor using takagi-sugeno fuzzy model and an lmi approach. *In*: **2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)**. 2014. p. 370–374.

LI, J.; LI, Y. Dynamic analysis and pid control for a quadrotor. *In*: **2011 IEEE International Conference on Mechatronics and Automation**. 2011. p. 573–578.

LI, Q. *et al.* Deep neural networks for improved, impromptu trajectory tracking of quadrotors. **IEEE International Conference on Robotics and Automation (ICRA)**, IEEE, p. 5183–5189, 2017. ISSN 10504729.

LI, S.; YANG, J. Robust autopilot design for bank-to-turn missiles using disturbance observers. **IEEE Transactions on Aerospace and Electronic Systems**, IEEE, v. 49, n. 1, p. 558–579, 2013.

LI, S. *et al.* **Disturbance observer-based control: methods and applications**. : CRC press, 2014.

LIU, J. **Intelligent Control Design and MATLAB Simulation**. : Springer Singapore, 2017. ISBN 9789811052637.

LIU, J. *et al.* Accurate mapping method for uav photogrammetry without ground control points in the map projection frame. **IEEE Transactions on Geoscience and Remote Sensing**, p. 1–9, 2021.

MA, D. *et al.* Active disturbance rejection and predictive control strategy for a quadrotor helicopter. **IET Control Theory and Applications**, v. 10, n. 17, p. 2213–2222, 2016.

MADANI, T.; BENALLEGUE, A. Sliding mode observer and backstepping control for a quadrotor unmanned aerial vehicles. **Proceedings of the American Control Conference**, p. 5887–5892, 2007.

MAHONY, R.; KUMAR, V.; CORKE, P. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. **IEEE Robotics and Automation Magazine**, IEEE, v. 19, n. 3, p. 20–32, 2012.

MARTINS, L.; CARDEIRA, C.; OLIVEIRA, P. Linear quadratic regulator for trajectory tracking of a quadrotor. **IFAC-PapersOnLine**, v. 52, p. 176–181, 01 2019.

MCCULLOCH, W. S.; PITTS, W. Learning based industrial bin-picking trained with approximate physics simulator. **The bulletin of mathematical biophysics**, v. 5, p. 115–133, 1943.

OGATA, K. **Engenharia de Controle Moderno**. 5°. ed. : Pearson, 2010. 912 p.

Önder Efe, M. Sliding mode control for unmanned aerial vehicles research. **Studies in Systems, Decision and Control**, v. 24, p. 239–255, 2015.

PAIVA, M. A. D. schocio. **Controle Baseado em Observação de Distúrbio de um Quadricóptero Sujeito a Incertezas Paramétricas e Distúrbio de Vento**. 2019. Tese (Dissertação) — Universidade de São Paulo, 2019.

PI, C. H.; YE, W. Y.; CHENG, S. Robust quadrotor control through reinforcement learning with disturbance compensation. **Applied Sciences (Switzerland)**, v. 11, n. 7, 2021. ISSN 20763417.

POLYAK, B. T. Some methods of speeding up the convergence of iteration methods. **USSR Computational Mathematics and Mathematical Physics**, v. 4, n. 5, p. 1–17, 1964. ISSN 00415553.

QUIGLEY, M. *et al.* Ros: an open-source robot operating system. *In*: **ICRA Workshop on Open Source Software**. 2009.

RADKE, A.; GAO, Z. A survey of state and disturbance observers for practitioners. **Proceedings of the American Control Conference**, IEEE, v. 2006, n. 3, p. 5183–5188, 2006.

RAFFO, G. V.; ORTEGA, M. G.; RUBIO, F. R. An integral predictive/nonlinear H_∞ control structure for a quadrotor helicopter. **Automatica**, Elsevier Ltd, v. 46, n. 1, p. 29–39, 2010. ISSN 00051098. Disponível em: http://dx.doi.org/10.1016/j.automatica.2009.10.018.

RAFFO, G. V.; ORTEGA, M. G.; RUBIO, F. R. Nonlinear H_∞ controller for the quad-rotor helicopter with input coupling. **IFAC Proceedings Volumes**, v. 44, p. 13834–13839, 2011.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, v. 65, n. 6, p. 386–408, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning Internal Representations by Error Propagation. **Readings in Cognitive Science: A Perspective from Psychology and Artificial Intelligence**, n. V, p. 399–421, 1985.

SANTANA, L. V. *et al.* A trajectory tracking and 3D positioning controller for the AR.Drone quadrotor. **2014 International Conference on Unmanned Aircraft Systems, ICUAS 2014 - Conference Proceedings**, IEEE, p. 756–767, 2014.

SARABAKHA, A.; KAYACAN, E. Online Deep Learning for Improved Trajectory Tracking of Unmanned Aerial Vehicles Using Expert Knowledge. **IEEE International Conference on Robotics and Automation (ICRA)**, p. 7727–7733, 2019. ISSN 23318422.

SASTRY, S. **Nonlinear Systems: Analysis, Stability, and Control**. Springer New York, 2013. (Interdisciplinary Applied Mathematics). ISBN 9781475731088. Disponível em: https://books.google.com.br/books?id=j_PiBwAAQBAJ.

SIERRA, J. E.; SANTOS, M. Wind and Payload Disturbance Rejection Control Based on Adaptive Neural Estimators: Application on Quadrotors. **Complexity**, v. 2019, p. 1–20, 2019. ISSN 10990526.

SILVA, I. da *et al.* **Artificial Neural Networks: A Practical Course**. : Springer International Publishing, 2016. ISBN 9783319431628.

SKANSI, S. **Introduction to deep learning: From Logical Calculus to Artificial Intelligence**. 2018. v. 114. 196 p. ISBN 978-3-319-73003-5.

SMITH, J.; LIU, C.; CHEN, W. H. Disturbance observer based control for gust alleviation of a small fixed-wing UAS. **2016 International Conference on Unmanned Aircraft Systems, ICUAS 2016**, IEEE, p. 97–106, 2016.

SRIVASTAVA, N. *et al.* Dropout: A simple way to prevent neural networks from overfitting. **Journal of Machine Learning Research**, v. 15, n. 56, p. 1929–1958, 2014. Disponível em: http://jmlr.org/papers/v15/srivastava14a.html.

SUN, J.; WANG, C.; XIN, R. On Disturbance Rejection Control of Servo System Based on the Improved Disturbance Observer. **Chinese Control Conference, CCC**, Technical Committee on Control Theory, Chinese Association of Automation, v. 2018-July, p. 2554–2559, 2018.

TERRA, M. H.; CERRI, J. P.; ISHIHARA, J. Y. Optimal robust linear quadratic regulator for systems subject to uncertainties. **IEEE Transactions on Automatic Control**, v. 59, n. 9, p. 2586–2591, 2014.

ULLAH, I.; FAYAZ, M.; KIM, D. H. Improving accuracy of the Kalman filter algorithm in dynamic conditions using ANN-based learning module. **Symmetry**, v. 11, n. 1, 2019.

VALAVANIS, K.; VACHTSEVANOS, G. **Handbook of Unmanned Aerial Vehicles**. : Springer-Verlag GmbH, 2014. ISBN 9789048197088.

VARSHNEY, P.; NAGAR, G.; SAHA, I. DeepControl: Energy-Efficient Control of a Quadrotor using a Deep Neural Network. **IEEE International Conference on Intelligent Robots and Systems (IROS)**, p. 43–50, 2019. ISSN 21530866.

VICON. **Motion Capture System**. 2012. Disponível em: https://www.vicon.com/. Acessado: 2020-01-18.

VOOS, H. Nonlinear control of a quadrotor micro-uav using feedback-linearization. *In*: **2009 IEEE International Conference on Mechatronics**. 2009. p. 1–6.

WELCH, G.; BISHOP, G. An Introduction to the Kalman Filter. **In Practice**, v. 7, n. 1, p. 1–16, 2006.

XIAONING, Z. Analysis of military application of UAV swarm technology. *In*: **2020 3rd International Conference on Unmanned Systems (ICUS)**. 2020. p. 1200–1204.

XIE, L. L.; GUO, L. How much uncertainty can be dealt with by feedback? **IEEE Transactions on Automatic Control**, v. 45, n. 12, p. 2203–2217, 2000.

XU, B. *et al.* Neural Learning Control of Strict-Feedback Systems Using Disturbance Observer. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, v. 30, n. 5, p. 1296–1307, 2019. ISSN 21622388.

XU, B. *et al.* DOB-Based Neural Control of Flexible Hypersonic Flight Vehicle Considering Wind Effects. **IEEE Transactions on Industrial Electronics**, v. 64, n. 11, p. 8676–8685, 2017. ISSN 02780046.

YANG, H. C. *et al.* Implementation of an autonomous surveillance quadrotor system. **AIAA Infotech at Aerospace Conference and Exhibit and AIAA Unmanned...Unlimited Conference**, n. April, 2009.

YANG, J. *et al.* Disturbance/Uncertainty Estimation and Attenuation Techniques in PMSM Drives - A Survey. **IEEE Transactions on Industrial Electronics**, IEEE, v. 64, n. 4, p. 3273–3285, 2017.

ZEILER, M. D. ADADELTA: An Adaptive Learning Rate Method. 2012. Disponível em: http://arxiv.org/abs/1212.5701.

ZEMALACHE, K.; BEJI, L.; MARREF, H. Control of an under-actuated system: application a four rotors rotorcraft. *In*: **2005 IEEE International Conference on Robotics and Biomimetics - ROBIO**. 2005. p. 404–409.