Universidade Federal de São Carlos– UFSCar
Centro de Ciências Exatas e de Tecnologia– CCET
Departamento de Engenharia Elétrica– DEE
Programa de Pós-Graduação em Engenharia Elétrica– PPGEE

Controle $\mathcal{H}\infty$ Não Linear Adaptativo baseado em Processos Gaussianos para Quadrotor

Guilherme Rossi de Avelar Oliveira

Orientador: Roberto Santos Inoue

São Carlos
2023

Guilherme Rossi de Avelar Oliveira

Controle $\mathcal{H}\infty$ Não Linear Adaptativo baseado em Processos Gaussianos para Quadrotor

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.

Área de concentração: Sistemas Elétricos e Eletrônicos

Orientador: Roberto Santos Inoue

São Carlos
2023

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Engenharia Elétrica

Folha de Aprovação

Defesa de Dissertação de Mestrado do candidato Guilherme Rossi de Avelar Oliveira, realizada em 17/02/2023.

Comissão Julgadora:

Prof. Dr. Roberto Santos Inoue (UFSCar)

Prof. Dr. Adriano Almeida Goncalves Siqueira (USP)

Prof. Dr. João Vitor de Carvalho Fontes (UFSCar)

O Relatório de Defesa assinado pelos membros da Comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Engenharia Elétrica.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

A utilização de Veículos Aéreos Não Tripulados (UAVs), pode estar sujeita a perturbações externas, como rajadas de vento, forças aerodinâmicas e incertezas paramétricas, dependendo do local onde será utilizado. Portanto, o controle automático deve ser robusto o suficiente para realizar uma tarefa pré-estabelecida. Neste trabalho, propomos uma estratégia de controle robusta baseada no modelo dinâmico para mitigar distúrbios externos e incertezas paramétricas. O critério $\mathcal{H}_\infty$ foi usado para outros tipos de robôs e será aplicado ao quadrotor neste trabalho. Além desse caráter robusto do critério de controle adotado, este trabalho usa processos gaussianos para complementar o modelo matemático do quadrotor, estimando as incertezas não modeladas pelo modelo nominal. Esta combinação é apoiada garantindo a estabilidade do sistema de controle robusto resultante. Uma análise sobre o rastreamento de trajetória de todos os controladores é feita, considerando diferentes níveis de incerteza paramétrica e perturbações externas. Junto com o estudo dos controladores, também é realizado um estudo completo sobre a implementação dos controles em um quadrotor, propondo um sistema de localização baseado na odometria da aeronave e em um algoritmo SLAM visual, utilizando-se da câmera presente na mesma.

Palavras-chave: Controle. MAV. Escala. Robusto. Adaptativo. Processos Gaussianos.

Abstract

The use of Unmanned Aerial Vehicles (UAVs), may be subject to external disturbances, such as wind gusts, aerodynamic forces, and parametric uncertainties, depending on the location where it is to be used. Therefore, the automatic control must be robust enough to perform a pre-established task. In this work, we propose a robust control strategy based on the dynamic model to mitigate external disturbances and parametric uncertainties. The $\mathcal{H}_\infty$ criterion has been used for other types of robots and will be applied to quadrotor aircraft in this work. In addition to this robust character of the adopted control criterion, this work uses Gaussian processes to complement the mathematical model of the quadrotor, estimating the uncertainties not modeled by the nominal model. This combination is backed up by ensuring the stability of the resulting robust control system. An analysis of the trajectory tracking of all controllers is made, considering different levels of parametric uncertainty and external disturbances. Along with the development of the controllers, a complete study is made about the implementation of control algorithms on a real quadrotor, proposing a localization system based on the odometry of the aircraft and on a visual SLAM algorithm, using the camera present in it.

Keywords: Quadrotor. Gaussian Process. Control. H Infinity. Tracking.

Lista de ilustrações

Lista de tabelas

Lista de siglas

MAV - Veículo Micro Aéreo
SLAM - Algoritmo Simultâneo de Localização e Mapeamento
vSLAM - Algoritmo Visual de Localização e Mapeamento
ROS - Robot Operating System
PTAM - Mapeamento e Rastreamento Paralelos
FK - Filtro de Kalman
FKE - Filtro de Kalman Estendido
GPS - Sistema de Posicionamento Global

Sumário

Capítulo 1

Introdução

Nos últimos anos, o estudo de técnicas de controle aplicadas a VANTs (Veículos Aéreos Não Tripulados) tem crescido muito. Atualmente, diversos produtos utilizam tecnologias desenvolvidas por essas obras e já estão presentes no mercado. Isso se deve à crescente disponibilidade de recursos computacionais de alto desempenho, aos avanços nas tecnologias de transmissão de dados e posicionamento global e ao constante desenvolvimento de sensores inerciais para a indústria automotiva e de dispositivos, permitindo o desenvolvimento de aeronaves cada vez mais confiáveis e versáteis. e menores, e de baixo custo, (INOUE, 2012). Dentre os vários tipos de aeronaves existentes, destaca-se o quadrotor, por ser um VANT de quatro asas rotativas que possui excelente versatilidade, facilidade de manter-se estável em voo pairado e realizar manobras em qualquer direção ((MADANI; BENALLEGUE, 2007), (HEHN; D'ANDREA, 2011), (BONNA; CAMINO, 2015)).

Tendo os quadrotores como base, foram estudadas diversas aplicações com variados objetivos, como por exemplo, aplicações militares, mapeamento aéreo, fotografia e até mesmo serviços voltados para agricultura de precisão como em (HERWITZ et al., 2004). Outros trabalhos podem ser citados, por exemplo, em (BILLS; CHEN; SAXENA, 2011) para navegação visual autônoma em ambientes estruturados, em (FAIGL et al., 2010) para vigilância autônoma, em (NG; SHARLIN, 2011) para interação homem-máquina, em (HIGUCHI; SHIMADA; REKIMOTO, 2011) como assistente desportivo, em (WEISS et al., 2010) para obtenção de mapas 3D; fotos aéreas, operações de busca e salvamento, detecção de incêndios e proteção florestal (BERNARD et al., 2011; YUAN; ZHANG; LIU, 2015), entre outros. Para testar os projetos de controle desenvolvidos, muitos trabalhos utilizam quadrotors em voos *indoor* com o auxílio de equipamentos de captura de movimentos de alta precisão. Para a grande maioria das aplicações utilizando VANTs, é interessante a utilização de um controle automático de trajetória, eliminando a necessi-

dade de um acompanhamento humano, fato esse que fomentou a busca de controles cada vez mais precisos que facilitem e ampliem a gama de aplicações envolvendo tal tecnologia.

Segundo (LUUKKONEN, 2011), para desenvolver o controle automático de uma aeronave é necessário conhecer seu modelo matemático. Existem várias maneiras de realizar essa descrição matemática em que diferentes estudos levam em consideração diferentes pontos. Portanto, em (HOFFMANN et al., 2007) e (HUANG et al., 2009) o modelo dinâmico do quadrotor foi estudado com a adição de propriedades aerodinâmicas complexas. Em (SANTANA; BRANDãO; SARCINELLI-FILHO, 2015), um quadrotor comercial de baixo custo foi usado para controlar a posição da aeronave. Com base em estudos realizados sobre o controle de sistemas robóticos, três abordagens são constantemente utilizadas na literatura: na primeira, o modelo de um sistema robótico é considerado completamente conhecido e utilizável para o controlador (LEWIS; ABDALLAH; DAWSON, 1993); na segunda, os parâmetros do modelo são desconhecidos e são estimados com base na propriedade robótica de parametrização linear; e na terceira, o modelo é desconhecido e uma abordagem inteligente (baseada em redes neurais ou lógica fuzzy) é usada para estimar o modelo (ver, por exemplo, (CHANG, 2000) e (CHANG, 2005)). Várias estratégias foram e estão sendo pesquisadas em relação ao controle do quadrotor. Controladores clássicos como o PID foram estudados em (TAYEBI; MCGILVRAY, 2004), (DIKMEN et al., 2010), (ZUO, 2010) e (BOUABDALLAH et al., 2004), A técnica de controle *backstepping*, pode ser vista em (MADANI; BENALLEGUE, 2006) e (ZEMALACHE; BEJI; H.MARREF, 2005). Outras técnicas de controle como $\mathcal{H}_\infty$, (RAFFO; ORTEGA; RUBIO, 2010), controladores LQR, (BOUABDALLAH et al., 2004), e controladores não lineares com saturação, (CASTILLO; R.LOZANO; A.DZUL., 2005) e (ESCARENO; C.SALAZAR-CRUZ; LOZANO, 2006).

O controle automático de um quadrotor deve ser capaz de atender aos mais diversos usos de trabalho e sabe-se que essas aeronaves estão diretamente sujeitas a perturbações externas e incertezas paramétricas nos parâmetros de seu modelo matemático, tornando o desafio de controlar um sistema robótico em circunstâncias como o descrito. A técnica de controle $\mathcal{H}_\infty$ tem sido utilizada para atenuar distúrbios externos. E em relação às incertezas paramétricas, técnicas de controle adaptativo têm sido utilizadas. Por exemplo, em (RAFFO; ORTEGA; RUBIO, 2015), uma estratégia de controle robusta foi proposta para executar um caminho predefinido, combinando uma abordagem *backstepping*, para controlar os movimentos translacionais, com um controle $\mathcal{H}_\infty$ para estabilizar o quadrotor considerando perturbações externas como, por exemplo, forças e momentos aerodinâmicos atuando em todos os graus de liberdade, além de incertezas paramétricas. Abordagens robustas são comparadas com técnicas de controle clássicas para encontrar o melhor desempenho em algumas situações, como em (ORTIZ; MINCHALA; REINOSO, 2016) que comparou um controlador robusto $\mathcal{H}_\infty$ e um controlador PID convencional para resolver a orientação problema de regulação do sistema. Em (LIU; MA; TU, 2018) um controle

robusto adaptativo *backstepping* baseado no modelo de voo quadrotor foi proposto para que a compensação robusta e a compensação adaptativa sejam aplicadas para lidar com distúrbios externos e incertezas dos parâmetros de inércia, respectivamente. Em (DEMIRCIOGLU; BASTURK, 2017) desenvolveu um comportamento adaptativo e controle de altura para quadrotors. Em seu trabalho, considerou-se que a massa total da aeronave, bem como a inércia, o comprimento das hélices, o coeficiente de empuxo e o coeficiente de arrasto das hélices não eram conhecidos. Também foi assumido que o quadrotor está sujeito a distúrbios de vento desconhecidos para ser uma soma finita de senoides com frequências, amplitudes e fases desconhecidas. Em (PAZELLI; TERRA; SIQUEIRA, 2011) e (NOGUEIRA et al., 2013), foi formulada e investigada a aplicação de diversas técnicas de controle adaptativo $\mathcal{H}_\infty$ não lineares baseadas em parametrização linear, redes neurais e sistemas fuzzy para manipuladores sujeitos a incertezas paramétricas e perturbações externas, resultados experimentais demonstraram a eficácia dos controladores para os testes realizados.

Em busca de uma melhoria no desempenho dos controles, alguns autores usaram processos Gaussianos para estimar o modelo de sistemas robóticos. Em (BERKENKAMP; SCHOELLIG, 2014) processos Gaussianos foram aplicados para aprender um modelo dinâmico não linear desconhecido combinado com teoria de controle robusta para garantir a estabilidade enquanto melhora gradualmente o desempenho. Para ilustrar o processo, (BERKENKAMP; SCHOELLIG, 2014) usou um exemplo de carrinho de pêndulo invertido. Em (CAO; LAI; ALAM, 2016), um esquema de controle hierárquico foi aplicado para resolver o problema de rastreamento de trajetória de um quadrotor no qual o modelo da aeronave foi treinado por dados empíricos usando técnicas de processo Gaussiano. E em (WANG; THEODOROU; EGERSTEDT, 2017), uma estratégia de aprendizagem recursiva baseada em processos Gaussianos foi desenvolvida para aprender a complexa e não linear dinâmica 3D de um quadrotor, propondo um algoritmo de amostragem adaptativa para reduzir o custo computacional no processo de aprendizagem.

Para implementar tais estratégias de controle automático, é necessário um estudo extensivo de como obter a localização do quadrotor, mapear o ambiente e utilizar os sensores disponíveis na aeronave. Diversos estudos foram feitos no sentido de realizar um sensoriamento remoto como solução para um mapeamento 3D do ambiente veja por exemplo (OWENS; SYCARA; SCERRI, 2009; SUZUKI et al., 2011). Com relação aos sensores utilizados para tal tarefa, sensores ativos como *laser scanners* ou sensores RGB-D, embora utilizados com sucesso para tarefas de SLAM em robôs terrestres, representam um problema para quadricópteros com carga útil e reservas de energia muito limitada. Nesses casos, torna-se interessante utilizar uma câmera passiva, que além de fornecer rica informação sensorial, consome relativamente pouca energia e possui pouca massa (SADAT et al., 2014).

Muitos trabalhos têm utilizado câmeras em VANTs para problemas de mapeamento

3D (MAYER; BARTELSEN, 2008; SUZUKI et al., 2011; WEISS et al., 2010). Esse problema é conhecido em robótica como localização e mapeamento simultâneos visual ou SLAM Visual (*Visual Simultaneous Localization and Mapping*), ou seja, consiste na estimativa da postura da câmera e do modelo do ambiente durante a navegação a partir de uma sequência de imagens. Baseado na forma em que o SLAM Visual extrai informações da imagem (FORSTER; PIZZOLI; SCARAMUZZA, 2014; MUR-ARTAL; TARDOS, 2015; SCHÖPS; ENGEL; CREMERS, 2014), pode-se classificar como um método baseado em características (KLEIN; MURRAY, 2007) ou um método direto (ENGEL; SCHÖPS; CREMERS, 2014). O método baseado em característica utiliza somente informações que se conformam com a extração de característica para elaborar a reconstrução 3D, isso é uma limitação. Apesar disso, este possui um custo computacional baixo, o que é importante para aplicações em tempo-real, e justifica o grande número de trabalhos desse método em celulares, por exemplo: (KLEIN; MURRAY, 2009; LI; KIM; MOURIKIS, 2013; TANSKANEN et al., 2013) . Já o método direto utiliza toda a informação da imagem para elaborar a reconstrução 3D, porém possui um elevado custo computacional, sendo geralmente implementado em computadores. Em celulares, recentemente em (SCHÖPS; ENGEL; CREMERS, 2014), conseguiu se utilizar o método direto para obter mapas de profundidade semi-densos de ambientes internos em aplicações de realidade aumentada.

Uma das limitações de um sistema SLAM Visual de uma câmera monocular é que ela possui um ambiguidade de escala que dificulta seu uso em certas aplicações (CONCHA et al., 2016). Para superar esse problema, uma técnica conhecida como SLAM Visual Inercial, o qual consiste na utilização do SLAM Visual auxiliado por um sistema de navegação inercial (do inglês *Inertial Navigation System* (INS)), tem sido amplamente utilizada. Essa técnica consiste em fundir o SLAM Visual com informações de giroscópios e acelerômetros, geralmente usando um filtro de Kalman estendido (HESCH et al., 2014; LI; KIM; MOURIKIS, 2013; LI; MOURIKIS, 2012). Sensores como magnetômetros e módulos GPS geralmente não são usados em ambientes internos (TANSKANEN et al., 2013), já que as medidas do magnetômetro se deterioram na presença de materiais ferromagnéticos e sinais de GPS se tornam indisponíveis em ambientes internos ou em ambientes urbanos. No entanto, esses sensores têm sido utilizados com sucesso para a localização de veículos aéreos não tripulados em tarefas externas, uma vez que fornecem medições absolutas que eliminam o desvio na estimativa de posicionamento (FARREL, 2008). Isso motiva o uso desses sensores em abordagens SLAM externas, como em (SCHLEICHER et al., 2010; SHEPARD; HUMPHREYS, 2014).

1.1 Objetivos

Para este projeto propõe-se para um MAV comercial, um sistema de navegação robusto a distúrbios externos e a incertezas paramétricas em ambientes com ausência de sinais de GPS e sistemas externos de captura de movimento. Para a localização do MAV propõe-se a utilização de SLAM Visual Monocular e para o controle robusto a distúrbios externos e a incertezas paramétricas propõe-se um controle adaptativo $\mathcal{H}\infty$ não linear baseado em processos gaussianos. Serão testados três métodos adaptativos diferentes para estimar as incertezas não modeladas pelo controle, um deles é baseado em Redes Neurais, um baseado em Processos Gaussianos *Online* e o último em em Processos Gaussianos *Offline*.

Para isto será necessário atingir os seguintes objetivos intermediários:

- ❏ Desenvolvimento do Controlador $\mathcal{H}\infty$ Não Linear para MAV.
- ❏ Desenvolvimento e análise de algoritmo adaptativo baseado em Redes Neurais.
- ❏ Desenvolvimento e análise de algoritmo adaptativo baseado em Processos Gaussianos *Online*.
- ❏ Desenvolvimento e análise de algoritmo adaptativo baseado em Processos Gaussianos *Offline*.
- ❏ Estudo dos blocos necessários para implementação dos controles.
- ❏ Estimativa da escala métrica do SLAM Visual utilizando os dados de odometria do MAV e os dados de postura do SLAM.
- ❏ Fusão dos dados do SLAM Visual com os dados de odometria do MAV para estimativa de orientação e posição do MAV.

1.2 Justificativas

Atualmente, os MAVs são utilizados em aplicações nas mais diversas áreas fazendo com que cresça a realização de pesquisas que buscam gerar tecnologia para estes tipos de aeronaves. Em especial, MAVs do tipo quadricóptero são utilizados em aplicações de militarismo, agricultura de precisão, mapeamento de locais externos e internos, inspeção de rede elétricas, trabalhos de difícil acesso ou de risco elevado para vida, entre outros.

Esses MAVs geralmente são equipados com câmeras monoculares, sistema de odometria e GPS. Contudo, em muitas dessas aplicações não se pode contar com localização baseada em GPS ou por sistemas externos de câmera, pois são aplicações em ambiente internos ou ambientes com prédios ou arvores altos que causam oclusões nas medidas dos sinais GPS. Diversos projetos têm investido em diferentes métodos de estimação de escala para localização dos MAVs porque um ponto crucial do controle desse tipo de robô, é

justamente a qualidade dos dados obtidos pelos sensores. A precisão de medidas como velocidade, aceleração e posição possuem um enorme impacto na qualidade do controle, montagem de mapas visuais e performance desse tipos de aplicações. Isso motiva o desenvolvimento de algoritmos cada vez mais precisos na obtenção de informações essenciais para essa área.

Outro ponto crucial nesse tipo de projeto é a escolha das técnicas de controle. Em muitas situações os MAVs estão sujeitos a distúrbios externos, como rajadas de vento; forças aerodinâmicas; e incertezas paramétricas, devido ao acréscimo de cargas (como câmeras, sensores, atuadores, computadores, etc) para a execução de uma determinada aplicação. Isso motiva a utilização de controladores robustos a distúrbios externos e incertezas paramétricas, já que não é possível modelar matematicamente todos esses parâmetros com precisão. Algoritmos de controle adaptativo também possuem um papel importante nessas aplicações, o ajuste dos parâmetros em um sistema de aprendizado contribui para uma melhor performance desses sistemas. Assim sendo, é proposto um controle robusto não linear baseado em $\mathcal{H}_\infty$ para atenuar distúrbios externos e resolver o problema seguimento de trajetória, com um algoritmo de processos gaussianos adaptativo para estimar incertezas paramétricas e a comparação de de mais dois métodos adaptativos diferentes em cima desse controle, avaliando a performance de cada um. O primeiro método é baseado em Redes Neurais e já obteve bons resultados nesse contexto, porém o método de Processos Gaussianos ainda não foi implementado neste contexto e será proposto nesse projeto. Também é feito o desenvolvimento dos processos necessários para implementação de tal controle em um quadrotor comercial, montando um sistema com Filtro de Kalman para estimativa de orientação e posição do MAV, utilizando sua câmera para a obtenção das coordenadas.

Capítulo 2

Formulação do Problema

Neste trabalho, no Capítulo 3, é desenvolvido um controle robusto $\mathcal{H}_\infty$ não-linear para resolver o problema de seguimento de trajetória, atenuando distúrbios externos e um algoritmo adaptativo baseado em processos Gaussianos para estimar incertezas paramétricas. Portanto, antes da apresentação da estratégia de controle, nas seções a seguir, será exibido o modelo quadrotor e sua representação no espaço de estados para o problema de controle não linear $\mathcal{H}_\infty$.

2.1 Considerações iniciais

- **Superfície localmente plana** Considerando as limitações de distância de voo devido ao esgotamento da bateria, a rotação e curvatura da Terra são desconsideradas. As forças gravitacionais também são consideradas uniformes.

- **Corpo rígido** O quadrotor é considerado um corpo rígido com massa M e inércia $\mathcal{J}$ em seu centro de massa. Isso implica que qualquer vetor unindo dois pontos do corpo é sempre invariante. A massa e a inércia são consideradas constantes.

- **Corpo do MAV** O corpo do MAV é considerado como um corpo rígido de massa m, com tensor de inércia $\mathcal{J}$ no seu centro de massa.

- **Controles automáticos de altitude e arfagem/rolamento** Utilizamos neste projeto algumas funções já integradas no Parrot Bebop 2. A primeira delas é o *Hover*, que faz com que o MAV realize a força necessária para compensar a força da Gravidade, eliminando a necessidade da modelagem dinâmica desse termo. Também nos utilizamos do sistema automático de controle de arfagem e rolamento do Parrot.

Essas rotações configuram o movimento de translação em y e em x respectivamente, em relação ao sistema de coordenadas R_B do corpo do MAV.

2.2 Quadrotor

O quadrotor Parrot Bebop 2.0, ilustrado na Figura 1, é um veículo aéreo autônomo desenvolvido e comercializado pela Parrot. Foi projetado para ser controlado por joystick, smartphones e tablets através de protocolos específicos de comunicação WiFi. O Bebop 2.0 está equipado com dois sistemas embarcados. Uma IMU contendo: um magnetômetro de 3 eixos, um giroscópio de 3 eixos, um acelerômetro de 3 eixos e um sensor ultrassônico para análise de vôo acima de 8 metros de altitude e um sensor barométrico. Também possui um processamento de imagem que utiliza algoritmos de fluxo óptico para detecção de padrões de movimento. Esta unidade possui dois sensores ópticos, uma câmera monocular na frente e outra localizada na parte inferior da aeronave, que são utilizadas para detectar padrões de referência no solo a fim de obter estabilidade de atitude em voo pairado. O sistema principal do quadrotor é capaz de realizar automaticamente procedimentos de decolagem, pouso e estabilização de atitude da aeronave, além de responder a comandos de movimento.

Figura 1 – Quadrotor Parrot Bebop 2. (Retirado de: https://www.parrot.com)

2.3 Modelagem Dinâmica

Para implementar os controladores de rastreamento, primeiro precisamos definir seu modelo dinâmico. O quadrotor tem 6 graus de liberdade, 3 eixos de referência (x,y,z) e 3 ângulos de Euler (θ, ϕ, ψ),

A mecânica do corpo rígido para quadrotors é frequentemente formulada com base nas equações de Euler-Lagrange, como visto em (JOHANSSON, 1990) e (RAFFO; ORTEGA; RUBIO, 2015). Dadas as considerações iniciais apresentadas anteriormente, o sistema pode ser definido como:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}\right) - \frac{\partial \mathcal{L}}{\partial \mathbf{q}} = \tau + \delta, \tag{2.1}$$

onde $\mathcal{L}$ é o Lagrangiano do sistema, τ representa o torque/forças aplicadas, δ as forças externas, e q é o vetor do sistema de coordenadas transposto contendo todos os 6 graus de liberdade do quadrotor, dados a seguir:

$$q = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix}. \tag{2.2}$$

Resolvendo as derivadas de Euler-Lagrange (2.1), o sistema fica:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = N(q)\nu + \delta(q), \tag{2.3}$$

onde M é a matriz de inércia, positiva e simétrica, $G(q)$ representa as forças gravitacionais e $C(q,\dot{q})\dot{q}$ é o vetor de Coriolis e as forças centrípetas. Expandindo as matrizes, temos o sistema (2.4) abaixo:

$$\begin{bmatrix} mI & 0 \\ 0 & \mathcal{J} \end{bmatrix} \begin{bmatrix} \ddot{\xi} \\ \ddot{\eta} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & C_{\eta\eta} \end{bmatrix} \begin{bmatrix} \dot{\xi} \\ \dot{\eta} \end{bmatrix} + G(q) = N(q)\nu + \delta(q), \tag{2.4}$$

onde m é a massa do quadrotor, $\xi = [x, y, z]^T$ é o vetor translacional, $\eta = [\phi, \theta, \psi]^T$ é o vetor rotacional, $\mathcal{J}$ é a matriz de inércia relativa à parte rotacional, C_{nn} é a matriz de Coriolis, N é a matriz de força e ν é a vetor de ação de controle.

A matriz de força N é dada por:

$$N(q) = \begin{bmatrix} W'_\eta & 0 \\ 0 & R \end{bmatrix}, \tag{2.5}$$

onde W_η é a matriz de transformação das velocidades angulares e R a matriz rotacional, vista abaixo como:

$$W'_\eta = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\theta & -S_\theta \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix}, \tag{2.6}$$

$$R = \begin{bmatrix} C_\psi C_\theta & C_\psi S_\theta S_\phi - S_\psi C_\phi & C_\psi S_\theta C_\phi + S_\psi S_\phi \\ S_\psi C_\theta & S\psi S_\theta S_\phi + C_\psi C_\phi & S_\psi S_\theta C_\phi - C_\psi S_\phi \\ -S_\theta & C_\theta S_\phi & C_\theta C_\phi \end{bmatrix}, \tag{2.7}$$

onde C_x é equivalente a $cos(x)$ e S_x como $sin(x)$.

A matriz $\mathcal{J}$ pode ser escrita genericamente como segue:

$$\mathcal{J} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{xy} & I_{yy} & I_{yz} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}, \tag{2.8}$$

e a matriz de Coriolis C_{nn} pode ser definida de acordo com (CASTILLO; R.LOZANO; A.DZUL., 2005):

$$C_{nn} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}, \tag{2.9}$$

com

$c_{11} = 0,$

$c_{12} = (I_{yy} - I_{zz})(\dot{\theta}C_\phi S_\phi + \dot{\psi}S_\phi^2 C_\theta) + (I_{zz} - I_{yy})\dot{\psi}C_\phi^2 C_\theta - I_{xx}\dot{\psi}C_\theta,$

$c_{13} = (I_{zz} - I_{yy})\dot{\psi}C_\phi S_\phi C_\theta^2,$

$c_{21} = (I_{zz} - I_{yy})(\dot{\theta}C_\phi S_\phi + \dot{\psi}S_\phi^2 C_\theta) + (I_{yy} - I_{zz})\dot{\psi}C_\phi^2 C_\theta + I_{xx}\dot{\psi}C_\theta,$

$c_{22} = (I_{zz} - I_{yy})\dot{\phi}C_\phi S_\phi,$

$c_{23} = -I_{xx}\dot{\psi}S_\theta C_\theta + I_{yy}\dot{\psi}S_\phi^2 S_\theta C_\theta + I_{zz}\dot{\psi}C_\phi^2 S_\theta C_\theta,$

$c_{31} = (I_{yy} - I_{zz})\dot{\psi}C_\theta^2 S_\phi C_\phi - I_{xx}\dot{\theta}C_\theta,$

$c_{32} = (I_{zz} - I_{yy})(\dot{\theta}C_\phi S_\phi S_\theta + \dot{\phi}S_\phi^2 C_\theta) + (I_{yy} - I_{zz})\dot{\phi}C_\phi^2 C_\theta + I_{xx}\dot{\psi}S_\theta C_\theta$
$- I_{yy}\dot{\psi}S_\phi^2 S_\theta C_\theta - I_{zz}\dot{\psi}C_\phi^2 S_\theta C_\theta,$

$c_{33} = (I_{yy} - I_{zz})\dot{\phi}C_\phi S_\phi C_\theta^2 - I_{yy}\dot{\theta}S_\phi^2 C_\theta S_\theta - I_{zz}\dot{\theta}C_\phi^2 C_\theta S_\theta + I_{xx}\dot{\theta}C_\theta S_\theta.$

Conforme descrito na Seção 2.1, o quadrotor Parrot Bebop 2 já possui controle automático dos ângulos de inclinação (ϕ, θ). Tendo isso em vista, não controlaremos esses parâmetros. Também podemos eliminar o termo de força gravitacional $G(q)$ do nosso modelo, considerando o sistema *hover* embutido presente no Bebop 2.

Dessa forma, podemos definir o vetor do sistema q_0 abaixo:

$$q_0 = \begin{bmatrix} x \\ y \\ z \\ \psi \end{bmatrix}. \tag{2.10}$$

Com tais simplificações, a matriz de força N torna-se a matriz identidade. A matriz de inércia é reduzida a $J = I_{zz}$, a matriz de Coriolis torna-se nula e o sistema torna-se:

$$\begin{bmatrix} mI & 0 \\ 0 & I_{zz} \end{bmatrix} \begin{bmatrix} \ddot{\xi} \\ \ddot{\eta} \end{bmatrix} = v + \delta(q). \tag{2.11}$$

Considerando $\delta = w$ e adicionando incertezas paramétricas, o sistema final rearranjado fica:

$$\ddot{q} = Bv + B_d\omega, \tag{2.12}$$

onde $B_d = M^{-1}$ é a matriz relativa aos distúrbios externos, ω representa o vetor de distúrbios e $B = M^{-1}$.

2.4 Vetor de ação de controle

O vetor de ação de controle ν contém os comandos a serem enviados ao quadrotor. O Parrot Bebop 2 especificamente só recebe valores dentro de $[-1, 1]$ como comandos. Por exemplo, é possível enviar 1 para a direção $x+$ do Bebop para acelerar o quadrotor até sua capacidade máxima naquela direção. Com isso em mente, o vetor de controle ν é limitado com a função tangente hiperbólica conforme abaixo:

$$\nu_{sat} = tanh(\nu_{raw}), \tag{2.13}$$

sendo ν_{raw} as entradas brutas de controle e ν_{sat} é o vetor ajustado entre $-1, 1$.

Com a adaptação do vetor de ação para o Bebop, agora é necessária a tradução desses valores ajustados para o modelo dinâmico. Em (PINTO et al., 2020) e (BENEVIDES et al., 2019) foram feitos voos de calibração para determinar a relação entre as entradas do Bebop e a aceleração obtida pelo quadrotor. Nesse sentido, é possível criar a relação que segue:

$$\nu = K_a\nu_{sat}, \tag{2.14}$$

sendo ν a entrada de controle usada na modelagem de controle e K_a é obtido de (BENEVIDES et al., 2019) a matriz de escala de aceleração Bebop como segue

$$K_a = \begin{bmatrix} 4.33 & 0 & 0 & 0 \\ 0 & 4.12 & 0 & 0 \\ 0 & 0 & 4.42 & 0 \\ 0 & 0 & 0 & 5.92 \end{bmatrix}. \tag{2.15}$$

2.5 Modelagem dos distúrbios externos

A simulação da perturbação do vento (distúrbios externos) foi escolhida como um termo de aceleração que cabe na Eq. 2.12 dentro de ω. A distribuição do vento é selecionada como uma função senoidal, aplicada nos eixos x, y e z separadamente, em três momentos diferentes do voo t_x, t_y e t_z respectivamente, escolhidos de acordo com o experimento. Para ajudar na análise, a aceleração máxima da perturbação do vento foi tomada como uma porcentagem da aceleração máxima do quadrotor Parrot Bebop 2, variando o valor conforme necessário em cada experimento. O modelo simulado do distúrbio de vento segue as equações:

$$\begin{aligned} d_x &= \nu_{pct}(sin(t - t_x)^2), \\ d_y &= \nu_{pct}(sin(t - t_y)^2), \\ d_z &= \nu_{pct}(sin(t - t_z)^2). \end{aligned} \tag{2.16}$$

onde v_{pct} é a porcentagem desejada em relação a entrada de controle máxima para o experimento (aproximadamente $5m/s^2$ de acordo com os parâmetros em 2.15).

2.6 Representação de espaço de estado para controle não linear $\mathcal{H}_\infty$

Adota-se $\ddot{q}^d$ e $\dot{q}^d$ respectivamente como acelerações e velocidades de referência, pode-se somar e subtrair $\ddot{q}^d$ de um dos lados da Equação 2.13 que descreve o modelo do quadricóptero aqui utilizado, obtendo

$$\ddot{q} = \ddot{q}^d - \ddot{q}^d + B\nu + B_d\omega. \tag{2.17}$$

Definindo $\tilde{q} = q - q^d$, $\dot{\tilde{q}} = \dot{q} - \dot{q}^d$ e $\ddot{\tilde{q}} = \ddot{q} - \ddot{q}^d$ como erro de acompanhamento da referência e suas derivadas, a equação anterior será

$$\ddot{\tilde{q}} = -\ddot{q}^d + B\nu + B_d\omega. \tag{2.18}$$

De acordo com as considerações feitas até aqui, define-se um vetor de estado ($\tilde{x}$) para representar o espaço de estados baseado no erro

$$\tilde{x} = \begin{bmatrix} \dot{\tilde{q}} \\ \tilde{q} \end{bmatrix} = \begin{bmatrix} \dot{q} - \dot{q}^d \\ q - q^d \end{bmatrix}. \tag{2.19}$$

A representação quase-LPV, em espaço de estado, para o controle de acompanhamento de trajetória do *drone* é dada por

$$\dot{\tilde{x}} = \begin{bmatrix} 0 & 0 \\ I & 0 \end{bmatrix} \tilde{x} + \begin{bmatrix} I \\ 0 \end{bmatrix} u + \begin{bmatrix} B_d \\ 0 \end{bmatrix} \omega, \tag{2.20}$$

com

$$u = -\ddot{q}^d + B\nu, \tag{2.21}$$

ou

$$\nu = B^{-1}(u + \ddot{q}^d). \tag{2.22}$$

Considere agora a seguinte transformação de estado

$$\tilde{z} = \begin{bmatrix} \tilde{z_1} \\ \tilde{z_2} \end{bmatrix} = T_0\tilde{x} = \begin{bmatrix} T_{11} & T_{12} \\ 0 & I \end{bmatrix} \begin{bmatrix} \dot{\tilde{q}} \\ \tilde{q} \end{bmatrix}, \tag{2.23}$$

sendo T_{11}, $T_{12} \in \mathfrak{R}^{2\times 2}$ matrizes constantes a serem determinadas. Assume-se que a matriz T_{11} é diagonal, de maneira a facilitar escolhe-se $T_{11} = t_{11}I$. Aplicando a transformação de estado (2.23) em (2.20), obtém-se

$$\dot{\tilde{x}} = T_0^{-1}T_0 \begin{bmatrix} 0 & 0 \\ I & 0 \end{bmatrix} T_0^{-1}T_0\tilde{x} + T_0^{-1}T_0 \begin{bmatrix} I \\ 0 \end{bmatrix} u + T_0^{-1}T_0 \begin{bmatrix} B_d \\ 0 \end{bmatrix} \omega,$$

$$= T_0^{-1} \begin{bmatrix} T_{11}^{-1}T_{12} & -T_{12}^2T_{11}^{-1} \\ T_{11}^{-1} & -T_{11}^{-1}T_{12} \end{bmatrix} T_0\tilde{x} + T_0^{-1}T_0 \begin{bmatrix} I \\ 0 \end{bmatrix} u + T_0^{-1}T_0 \begin{bmatrix} B_d \\ 0 \end{bmatrix} \omega,$$

$$\begin{aligned} &= T_0^{-1} \begin{bmatrix} 0 & 0 \\ T_{11}^{-1} & -T_{11}^{-1}T_{12} \end{bmatrix} T_0\tilde{x}, \\ &+ T_0^{-1} \begin{bmatrix} T_{11}^{-1}T_{12} & -T_{12}^2T_{11}^{-1} \\ 0 & 0 \end{bmatrix} T_0\tilde{x}, \\ &+ T_0^{-1} \begin{bmatrix} (T_{11})(B\nu - \ddot{q}^d) \\ 0 \end{bmatrix} + T_0^{-1} \begin{bmatrix} B_d \\ 0 \end{bmatrix} T_{11}\omega, \end{aligned}$$

$$\begin{aligned} &= T_0^{-1} \begin{bmatrix} 0 & 0 \\ T_{11}^{-1} & -T_{11}^{-1}T_{12} \end{bmatrix} T_0\tilde{x}, \\ &+ T_0^{-1} \begin{bmatrix} T_{12} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\tilde{q}} \\ \tilde{q} \end{bmatrix}, \\ &+ T_0^{-1} \begin{bmatrix} (T_{11})(B\nu - \ddot{q}^d) \\ 0 \end{bmatrix} + T_0^{-1} \begin{bmatrix} B_d \\ 0 \end{bmatrix} T_{11}\omega, \end{aligned}$$

$$= T_0^{-1} \begin{bmatrix} 0 & 0 \\ T_{11}^{-1} & -T_{11}^{-1}T_{12} \end{bmatrix} T_0\tilde{x},$$
$$+ T_0^{-1} \begin{bmatrix} T_{12}\dot{\tilde{q}} - T_{11}\ddot{q}^d + T_{11}B\nu \\ 0 \end{bmatrix} + T_0^{-1} \begin{bmatrix} B_d \\ 0 \end{bmatrix} T_{11}\omega,$$

$$= T_0^{-1} \begin{bmatrix} 0 & 0 \\ T_{11}^{-1} & -T_{12}T_{11}^{-1} \end{bmatrix} T_0\tilde{x},$$
$$+ T_0^{-1} \begin{bmatrix} B \\ 0 \end{bmatrix} T_{11}[-B^{-1}(\ddot{q}^d - T_{11}^{-1}T_{12}\dot{\tilde{q}}) + \nu],$$
$$+ T_0^{-1} \begin{bmatrix} B_d \\ 0 \end{bmatrix} T_{11}\omega,$$

Definindo

$$A_T = T_0^{-1} \begin{bmatrix} 0 & 0 \\ T_{11}^{-1} & -T_{11}^{-1}T_{12} \end{bmatrix} T_0, \tag{2.24}$$

$$M_T = T_0^{-1} \begin{bmatrix} M \\ 0 \end{bmatrix} T_{11}, \tag{2.25}$$

$$B_T = T_0^{-1} \begin{bmatrix} B_d \\ 0 \end{bmatrix} T_{11}, \tag{2.26}$$

$$F_0(x_e) = B^{-1}(\ddot{q}^d - T_{11}^{-1}T_{12}\dot{\tilde{q}}), \tag{2.27}$$

obtém-se,

$$\dot{\tilde{x}} = A_T\tilde{x} + M_T(-F_0(x_e) + \nu) + B_T\omega \tag{2.28}$$

nesse sentido pode-se definir as incertezas modeladas do sistema

$$\mathcal{M} = M + \Delta M, \tag{2.29}$$

e considerando

$$F_0 = F + \Delta F, \tag{2.30}$$

obtém-se o sistema

$$\dot{\tilde{x}} = A_T\tilde{x} + M_T(-F - \Delta F) + \nu) + B_T\omega, \tag{2.31}$$

com

$$F = M(\ddot{q}^d - T_{11}^{-1}T_{12}\dot{\tilde{q}}), \tag{2.32}$$

$$\Delta F = \Delta M(\ddot{q}^d - T_{11}^{-1}T_{12}\dot{\tilde{q}}) + \delta, \tag{2.33}$$

sendo ΔF que representa as incertezas do sistema, na qual ΔM são as incertezas modeladas relativas à M, e δ são as incertezas não modeladas.

Capítulo 3

Controle

Esta Seção propõe o controle não linear $\mathcal{H}_\infty$ baseado em processos Gaussianos. Primeiro, mostramos na Seção 3.1 o Controle Não Linear $\mathcal{H}_\infty$ adaptativo e sua formulação para obter as três equações necessárias para resolver os parâmetros do sistema; depois disso, na Seção 3.3 descrevemos o Processo Gaussiano como um método adaptativo online; em seguida, modelamos a lei de controle do controle não linear $\mathcal{H}_\infty$ usando o processo gaussiano off-line na Seção 3.4 e explicamos a modelagem do processo gaussiano para o controle off-line e seus cálculos na Seção 3.4 .

3.1 Controle $\mathcal{H}_\infty$ não linear adaptativo

Conforme visto em (PAZELLI; TERRA; SIQUEIRA, 2011) e (NOGUEIRA et al., 2013), considere o sistema robótico descrito em (2.28) com $d \in \mathcal{L}_2[0,\infty)$. Dado um nível de atenuação $\gamma > 0$ e uma matriz de ponderação $Q = Q^T > 0$, se existirem uma matriz simétrica definida positiva $K = K^T > 0$ e uma matriz T_0 satisfazendo a equação algébrica de Riccati

$$\begin{bmatrix} 0 & K \\ K & 0 \end{bmatrix} - T_0^T [I \;\; 0]^T \left(R^{-1} - \frac{1}{\gamma^2} T_{11}^2 \right) [I \;\; 0] T_0 + Q = 0, \tag{3.1}$$

sendo R a matriz de ganho tal que $R < \gamma^2 I$, então a lei de controle adaptativo $\mathcal{H}_\infty$ baseado em sistemas inteligentes é dada por

$$\nu = F + \Xi\Theta + T_{11}^{-1} u + u_s \tag{3.2}$$

com

$$\dot{\Theta} = Proj[-Z^{-1}\Xi^T T_{11}[I \;\; 0]T_0\tilde{x}], \tag{3.3}$$

$$u = -R^{-1}[I \;\; 0]T_0\tilde{x}, \tag{3.4}$$

$$u_s = -k(x_e)sgn(T_{11}[I \;\; 0]T_0\tilde{x}), \tag{3.5}$$

sendo Z o ganho adaptativo, $\Xi\Theta$ representa a incerteza ΔF, $Proj$ é a função de projeção, neste caso utilizada como a função tangente hiperbólica e sgn é a função de saturação, dada por

$$sgn(T_{11}[I \;\; 0]T_0\tilde{x}) \triangleq [sgn((T_{11}[I \;\; 0]T_0\tilde{x})_1) \dots sgn((T_{11}[I \;\; 0]T_0\tilde{x})_n)]^T. \tag{3.6}$$

Esta lei de controle garante que todas as variáveis do sistema em malha fechada (2.28, 3.2-3.5) são limitadas e que a seguinte desigualdade é satisfeita

$$\int_0^T \left(\tilde{x}^T Q\tilde{x} + u^T Ru\right) dt \leq \; \tilde{x}^T(0)P_0\tilde{x}(0) + \tilde{\Theta}^T(0)Z\tilde{\Theta}(0) + \gamma^2 \int_0^T (d^T d)dt.$$

Considerando a matriz Q fatorada da seguinte forma

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{12}^T & Q_{22} \end{bmatrix}. \tag{3.7}$$

podemos simplificar a Equação (3.1) em três equações

$$q_{11} - t_{11}^2 r_u^{-1} + \gamma^{-2} t_{11}^4 = 0, \tag{3.8}$$

$$q_{22} - t_{12}^2 r_u^{-1} + \gamma^{-2} t_{11}^2 t_{12}^2 = 0, \tag{3.9}$$

$$k + q_{12} - t_{11} t_{12} r_u^{-1} + \gamma^{-2} t_{11}^3 t_{12} = 0. \tag{3.10}$$

Desta forma, t_{11}, t_{12} e $k > 0$ podem ser encontrados para o menor fator de atenuação $\gamma > 0$. O Algoritmo 1 do Apêndice A.1, mostra como encontrar tais parâmetros.

3.2 Controle $\mathcal{H}\infty$ não linear adaptativo baseado em Redes Neurais

Considerando o modelo $\mathcal{H}\infty$ não linear baseado no modelo do descrito na Seção 3.1, têm-se que a Equação (3.2) contempla o termo $\Xi\Theta$ que representa a incerteza ΔF. Nesta sessão temos a estimação desse termo com base em redes neurais.

Sendo assim, definem-se n redes neurais $\xi_k^T\Theta_k$, $k = 1, \dots, n$, compostas de neurônios não lineares em todas camadas escondidas e neurônios lineares nas camadas de entrada

e saída, com parâmetros ajustáveis Θ_k na camada de saída((CHANG, 2000), (CHEN; CHANG; LEE, 1997)). As saídas das redes neurais são da forma

$$\xi_k^T \Theta_k = \sum_{i=1}^{p_k} H\left(\sum_{j=1}^{5n} w_{ij}^k q_{ej} + b_i^k \right) \Theta_{ki}, \tag{3.11}$$

com

$$\xi_k = \begin{bmatrix} H\left(\sum_{j=1}^{5n} w_{1j}^k q_{ej} + b_1^k\right) \\ \vdots \\ H\left(\sum_{j=1}^{5n} w_{p_k j}^k q_{ej} + b_{p_k}^k\right) \end{bmatrix}, \Theta_k = \begin{bmatrix} \Theta_{k1} \\ \vdots \\ \Theta_{kp_k} \end{bmatrix},$$

com p_k sendo o número de neurônios da camada escondida e $\mathbf{q_{e_i}^d}$ como o vetor de erro das posições, velocidades e acelerações. Os pesos w_{ij}^k e o bias b_i^k para $1 \leq i \leq p_k$, $1 \leq j \leq 5n$ e $1 \leq k \leq n$ são assumidos constantes e $H(\cdot)$ é escolhido como a função tangente hiperbólica

$$H(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}.$$

Desse modo, a rede neural completa pode ser denotada por

$$\Delta \hat{F}(q_e, \Theta) = \begin{bmatrix} \Delta \hat{F}_1(q_e, \Theta_1) \\ \vdots \\ \Delta \hat{F}_n(q_e, \Theta_n) \end{bmatrix} = \begin{bmatrix} \xi_1 & 0 & \dots & 0 \\ 0 & \xi_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \xi_n \end{bmatrix} \begin{bmatrix} \Theta_1 \\ \vdots \\ \Theta_n \end{bmatrix} = \Xi\Theta. \tag{3.12}$$

e Θ calculado a partir da Equação (3.3).

3.3 Processo Gaussiano Adaptativo Online

Esta seção também aborda a estimação do termo ΔF_0, porém com a utilização Processos Gaussianos. Para estimar a incerteza ΔF com Processos Gaussianos, define-se n processos Gaussianos $\xi_k \Theta_k$, $k = 1, \dots, n$

$$\xi_k \Theta_k = \sigma^2 exp\left(- \frac{||q_{e_i}^k - q_{e_i}^{d\,k}||^2}{2l^2} \right) \Theta_k, \tag{3.13}$$

sendo ξ_k o *kernel* dado por

$$\xi_k = \sigma^2 exp\left(- \frac{||\mathbf{q_{e_i}^k} - \mathbf{q_{e_i}^{d\,k}}||^2}{2l^2} \right), \tag{3.14}$$

em que σ^2 é um fator de escala que determina a variação da função a partir de sua média, l é a escala de comprimento, $\mathbf{q_{e_i}^d}$ é o vetor das posições, velocidades e acelerações desejadas

calculadas a cada iteração i e $\mathbf{q_{e_i}}$ é o vetor das posições, velocidades e acelerações reais calculadas a cada iteração i.

Desse modo, o processo Gaussiano completo pode ser escrito como

$$\Delta\hat{F}(q_e) = \begin{bmatrix} \Delta\hat{F}_1(q_e,\Theta_1) \\ \vdots \\ \Delta\hat{F}_n(q_e,\Theta_n) \end{bmatrix} = \begin{bmatrix} \xi_1 & 0 & \dots & 0 \\ 0 & \xi_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \xi_n \end{bmatrix} \begin{bmatrix} \Theta_1 \\ \vdots \\ \Theta_n \end{bmatrix} = \Xi\Theta. \tag{3.15}$$

O fator adaptativo Θ é encontrado por (3.3).

3.4 Processo Gaussiano Offline

Chamamos esse processo de Offline porque estimamos as incertezas antes da execução do controlador. Para tanto, utilizamos as variáveis q_e, q_e^d e F_0, previamente salvas de outro controlador implementado. No Controle Não Linear $\mathcal{H}_\infty$ baseado em Processo Gaussiano Offline, não temos a lei de controle adaptativo. Então a lei de controle (3.2), torna-se

$$\nu = F + \Delta\widehat{F} + T_{11}^{-1}u. \tag{3.16}$$

Definimos um processo Gaussiano dado por

$$f(x) \backsim \mathcal{GP}(m(x), k(x,x')) = f(x) \backsim \mathcal{N}(m(x), k(x,x')), \tag{3.17}$$

onde $m(x)$ é o valor médio e a covariância é dada por *kernel*

$$k(x,x') = \sigma^2 exp\bigg(-\frac{||x-x'||^2}{2l^2}\bigg), \tag{3.18}$$

onde σ^2 é um fator de escala que determina a variância da função com base em sua média, l é o fator de comprimento, x os dados de entrada e x' os dados de teste.

Podemos então determinar $\Delta\widehat{F}$ sendo

$$\Delta\widehat{F} = k(q_e,q_e)[k(q_e,q_e^d) + \sigma_n^2 I]^{-1}F, \tag{3.19}$$

onde σ_n^2 é o parâmetro que estima a perturbação esperada no conjunto de dados, q_e^d é a soma das posições, velocidades e acelerações desejadas calculadas e salvas previamente, e q_e é a soma das posições, velocidades e acelerações reais calculadas e salvas previamente.

Usamos o algoritmo definido em (RASMUSSEN; WILLIAMS, 2006) para calcular a incerteza do modelo dinâmico:

$$L = cholesky(k + \sigma_n^2 I); \tag{3.20}$$

$$\alpha = L^T \backslash (L \backslash F); \tag{3.21}$$

$$\Delta\widehat{F} = k(q_e, q_e^d)^T\alpha, \tag{3.22}$$

usando este algoritmo repetidamente para cada entrada de dados q_e, sendo *cholesky* a decomposição matricial de Cholesky.

Capítulo 4

Resultados

Os resultados a seguir foram obtidos simulando o modelo dinâmico e as equações do controlador em um ambiente MATLAB, introduzindo as incertezas paramétricas do sistema e a perturbação do vento conforme desejado. A trajetória desejada usada nos experimentos é uma elipse. A velocidade escolhida é de 0,3 m/s e 2000 iterações em cada voo, que dura 20 segundos.

Os controladores utilizados nas simulações são os descritos no capítulo 3, com a adição de um controle *Feedback Linearization*, demonstrado na seção 4.1.1, sendo este o controle mais simples implementado, apenas para mérito de comparação com os outros controles implementados.

4.1 Parâmetros dos controles

Antes de demonstrar os resultados, iremos primeiramente mostrar os parâmetros de ajuste usados para cada um dos controles no estudo comparativo.

4.1.1 Feedback Linearization

Usamos um controlador de linearização de feedback proporcional-derivativo como o controlador mais básico nos experimentos. Sua lei de controle pode ser definida como:

$$\nu = M(u + \ddot{q}^d), \tag{4.1}$$

onde M é a massa do sistema, $\ddot{q}^d$ é a aceleração da trajetória desejada, e com:

$$u = -K_p\tilde{q} + -K_d\dot{\tilde{q}}, \tag{4.2}$$

onde K_p é o ganho proporcional e K_d é o ganho derivado. Os ganhos usados nos experimentos para este controlador são

$$K_p \ = \ 50 \times I_{4\times4}, \ K_d \ = \ 15 \times I_{4\times4}$$

4.1.2 Controle adaptativo não-linear $\mathcal{H}_\infty$ baseado em Redes Neurais

Os parâmetros para o controle $\mathcal{H}_\infty$ não-linear usados foram obtidos resolvendo as equações (3.8)-(3.10), e são $\gamma = 0,028$, $q_{11} = 0,5$, $q_{22} = 23$, $q_{12} = 0$ e $r_u = 0,02$, dandonos os parâmetros $t_{11} = 0,14$, $t_{12} = 0,94$ e $k = 3,39$. A lei adaptativa para ajustar a saída da rede neural é baseada na Eq. 3.3 com $Z = 0,2 \times I_{28\times28}$ e $k(x_e) = \frac{1}{0,2}\sqrt{\tilde{x}^T\tilde{x}}$. Definimos $\Delta\hat{F}_0(x_e,\Theta) = \left[\Delta\hat{F}_1(x_e,\Theta_1)\ \Delta\hat{F}_2(x_e,\Theta_2)\ \Delta\hat{F}_3(x_e,\Theta_3)\ \Delta\hat{F}_4(x_e,\Theta_4)\right]$ com $p_k = 7$ neurônios na camada oculta, o vetor de viés $b_k = [\ -3\ -2\ -1\ 0\ 1\ 2\ 3]$ e a ponderação matriz para a camada oculta $\Omega_i^k = \left[\omega_{ij}^k\right] = \big[-1\ -1\ -1\ -1\ -1\ -1\ -1\ -1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\big]$.

4.1.3 Controle não-linear $\mathcal{H}_\infty$ baseado em Processos Gaussianos *Offline*

Os parâmetros para o processo gaussiano offline foram definidos empiricamente, obtendo-se os seguintes valores: $\sigma_n^2 = 1 \times 10^{-5}$, $\sigma^2 = 1 \times 10^{-5}$ e $l = 10$. Os parâmetros relativos ao controle $\mathcal{H}_\infty$ usados foram mantidos os mesmos da Sec. 4.1.2.

4.1.4 Controle não-linear $\mathcal{H}_\infty$ baseado em Processos Gaussianos *Online*

Os parâmetros de controle $\mathcal{H}_\infty$ foram os mesmos da Seção 4.1.2. A lei adaptativa para ajustar a covariância, calculada pela Eq. 3.18, é implementada com base na Eq. 3.3 com $Z = 30 \times I_{4\times4}$ e $k(q_e) = \frac{1}{0,2}\sqrt{\tilde{x}^T\tilde{x}}$. Definimos $\Delta\hat{F}_0(q_e,\Theta) = \Xi\Theta$ para ser calculado e atualizado a cada iteração i do experimento, definido pelos parâmetros $\sigma^2 = 1 \times 10^{-5}$ e $l = 10$.

Como em 4.1.2, determinamos inicialmente os valores para Θ empiricamente, começando com valores aleatórios e atualizando-os ao final de cada experimento com os últimos valores encontrados. Os melhores valores encontrados para este parâmetro foram

$$\Theta^T = \left[0.6128\ 0.1821\ 0.2602\ 0.4578\right].$$

4.2 Seguimento de trajetória

O primeiro experimento foi a comparação de trajetória entre todos os controladores. As condições escolhidas foram incertezas paramétricas de 50% em cima da massa e inércia do quadrotor e uma perturbação do vento de $5m/s^2$ (100% do input máximo) seguindo a Sec. 2.5 modelo.

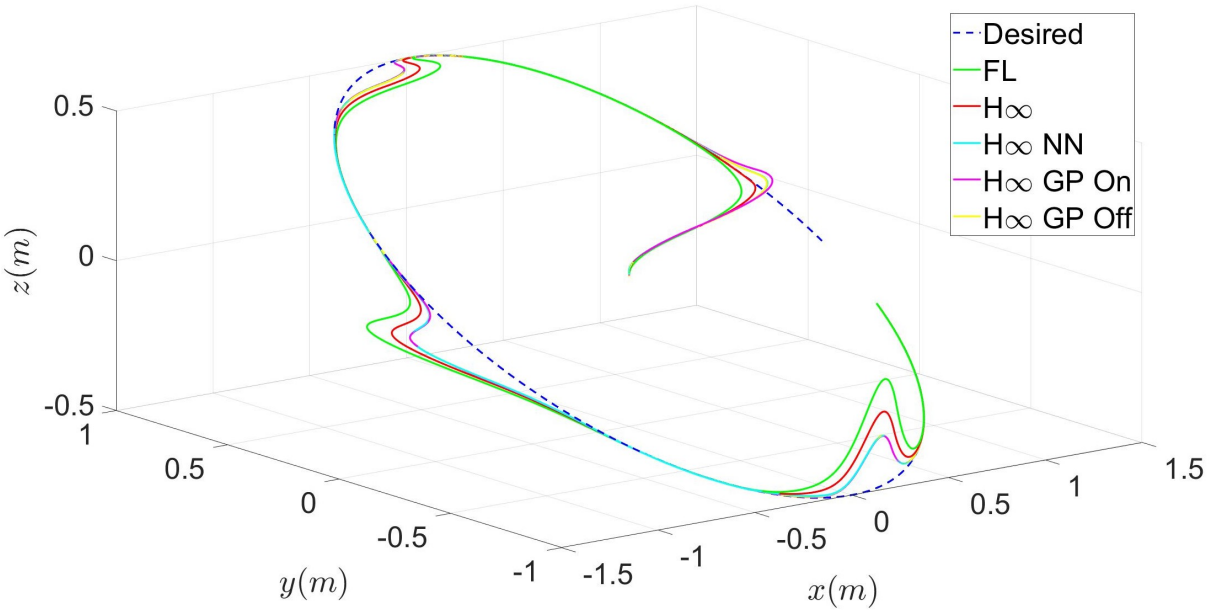


Figura 2 – Comparação da trajetória de todos os controladores em um voo com distúrbios de vento. (Autoria própria)

É possível ver a influência do ruído durante os três períodos de tempo diferentes e a capacidade de cada controlador recuperar a trajetória para a trajetória desejada. Visualmente é possível analisar que o controle *Feedback Linearization* obteve o pior desempenho, logo em seguida temos o controle $\mathcal{H}_\infty$ não-linear sem métodos adaptativos para estimação das incertezas. Com a adição dos métodos adaptativos (redes neurais, processos gaussianos *online* e *offline*) a melhora na performance é facilmente percebida. Uma análise mais detalhada sobre os controladores é feita nas próximas seções.

4.3 Análise de performance

Para analisar o desempenho entre todos os controladores, examinamos a norma euclidiana nas simulações de rastreamento de trajetória para medir o erro de posição e as entradas de controle em cada iteração. Temos as seguintes equações:

$$E_{q_p}(i) = \frac{1}{N}\sum_{j=1}^{N} ||q_{p_j} - q_{p_j}^d|| \tag{4.3}$$

para erros de posição (x, y, z) e

$$V(i) = \frac{1}{N} \sum_{j=1}^{N} ||\nu_j|| \tag{4.4}$$

para entradas de controle, onde q_p é o vetor contendo as posições dos eixos x, y e z; ν é o vetor de controle de entrada; j a iteração atual e N o número total de iterações em cada voo.

A tabela 1 mostra os resultados obtidos ao comparar todos os controladores para o controle de Linearização com realimentação. Para a comparação de desempenho, usamos o tracking error médio obtido pela Eq. 4.3, e para consumo de energia, usamos a Eq. 4.4.

Tabela 1 – Estudo comparativo.

-	$\mathcal{H}_\infty$	$\mathcal{H}_\infty$ NN	$\mathcal{H}_\infty$ offline	$\mathcal{H}_\infty$ online
Desempenho (%)	24,82	38,04	38,10	38,75
Consumo de energia (%)	-76,63	-795,8	-668,1	-793,2

A rede neural e ambos os métodos de processos gaussianos fornecem uma melhoria de desempenho equivalente em comparação com o controle de linearização de *feedback*. O melhor desempenho alcançado foi pelo algoritmo adaptativo do processo Gaussiano online, dando também uma das melhores melhorias de consumo de energia, ao lado da Rede Neural que foi o controlador de menor consumo de energia.

4.4 Análise da influência das incertezas paramétricas

Para entender a influência das incertezas paramétricas em cada controlador, fizemos um experimento aumentando as incertezas em cima da massa e inércia do sistema, matriz B_d.

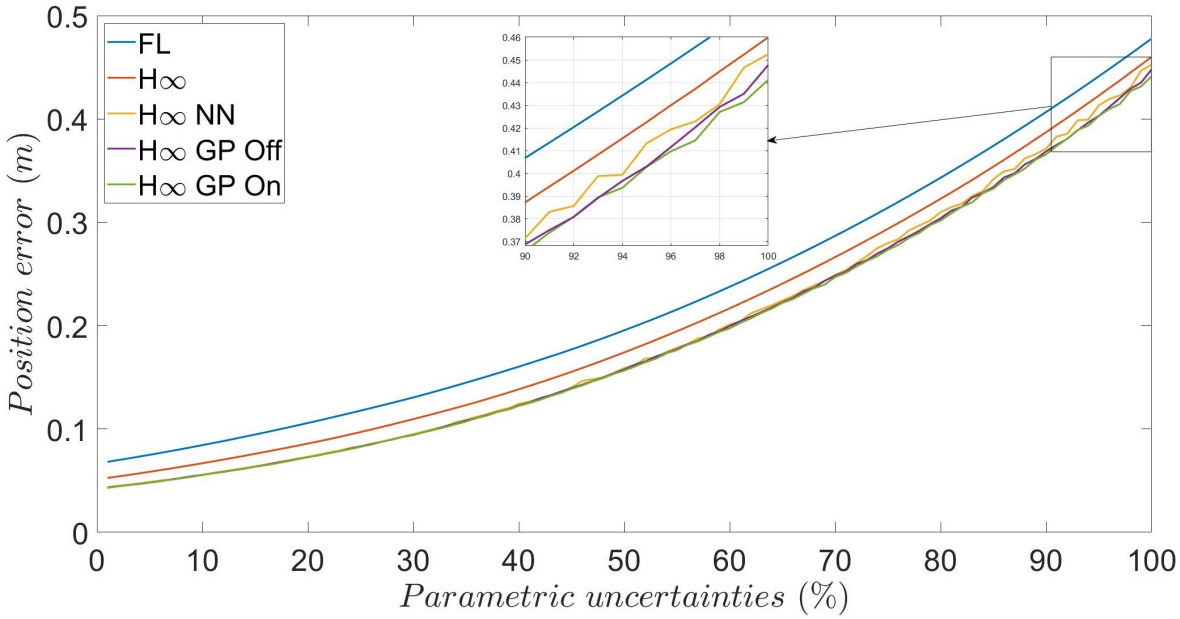


Figura 3 – Erro médio por voo com níveis crescentes de incerteza paramétrica. (Autoria própria)

Podemos ver que ambos os controles baseados em Processos Gaussianos lidam melhor com incertezas do que todos os outros controles, alcançando quase 10 centímetros de precisão de rastreamento em relação ao controle Feedback Linearization.

4.5 Análise da influência do distúrbio de vento

O experimento envolvendo distúrbios externos tem o mesmo formato que 4.4. As perturbações simulam rajadas de vento durante o voo do quadrotor, em 4 segundos no eixo x, 9 segundos no eixo y e 14 segundos no eixo z. Realizamos uma sequência de vôos de simulação de distúrbios externos crescentes, com base na entrada de controle máxima tomada de 2.15, de 0% a 120%.

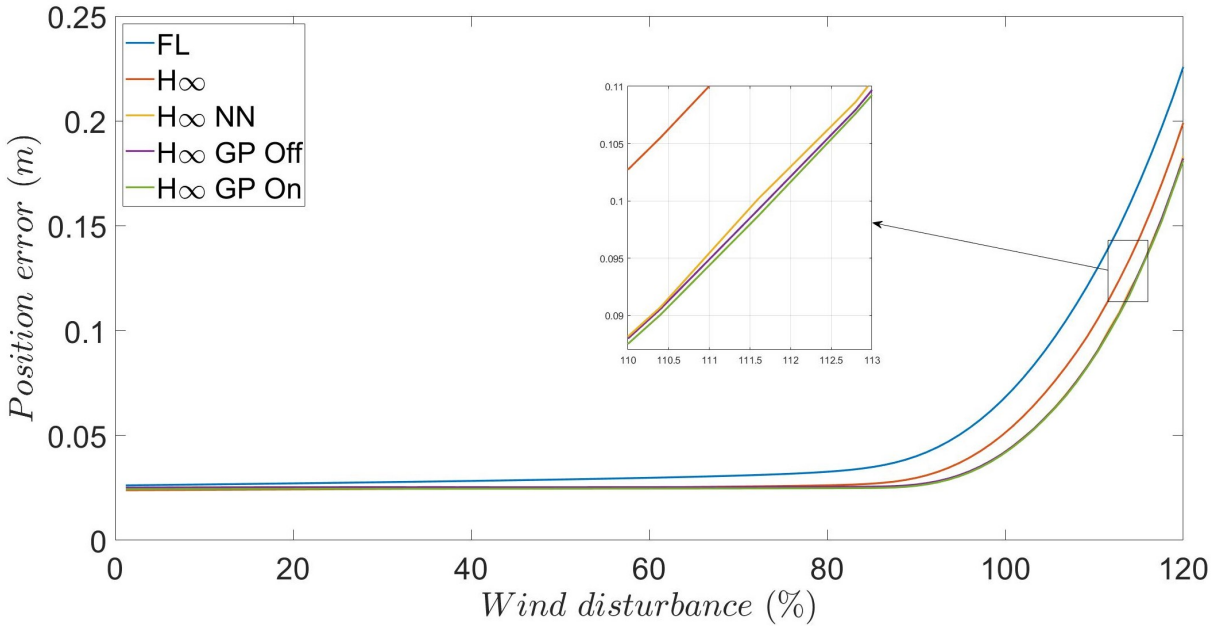


Figura 4 – Erro médio por voo com o aumento da perturbação do vento com base na entrada de controle máxima. (Autoria própria)

Com relação a distúrbios externos, o controle de Processos Gaussianos também tem um desempenho melhor do que os outros controles. Após 100% de entrada máxima de perturbação, o erro escala de maneira logarítmica, pois o controle não pode lidar com a aceleração perturbada. Até esse ponto, os controles lidam com perturbações periódicas do vento com um erro médio de menos de 10 centímetros.

Capítulo 5

Implementação

Durante o período do Mestrado não foi possível implementar o controle desenvolvido no Parrot Bebop 2. Nesse capítulo é apresentado estudo realizado de como seria possível aplicar o controle em um *drone* Parrot Bebop 2, analisando todo as etapas necessárias, como por exemplo: transmissão do vetor de controle ao *drone*, obtenção da posição real do quadrotor, criação de um sistema de coordenadas para referência de trajetória e análise de escala do sistema criado. A Seção 5.1 explica sobre o *framework* ROS utilizado para comunicação entre o computador e o Parrot Bebop 2. Na Seção 5.2 é demonstrado o sistema de captura de movimento Vicon-T40S para realizar obter a posição em tempo real do MAV. O simulador *Sphinx* é demonstrado na Seção 5.3. Em 5.4 é apresentado o pacote Bebop Autonomy, feito para integração entre o *framework* ROS e o Parrot Bebop, assim como o pacote *drone_dev*, mostrado na Seção 5.5. A parte de criação do sistema de coordenadas e inicialização é apresentada com o algoritmo SLAM na Seção 5.6 e por fim os algoritmos de obtenção de escala para o sistema de coordenadas é mostrado na Seção 5.6.4.

5.1 *Robot Operating System*

O ROS (FOUNDATION, 2007) é uma *framework* flexível para desenvolvimento de programas para robôs. Ele está sendo utilizado nesse projeto para realizar a implementação e comunicação entre os diferentes subsistemas do sistema de navegação. Os dados desse sistema são separados em mensagens, divididas em tópicos definidos como *publishers* (publicação de mensagens) e *subscribers* (recebimento de mensagens). Essa característica facilita muito o desenvolvimento de projetos envolvendo plataformas robóticas. O ROS possui algumas mensagens padronizadas que são muito importantes, principalmente em

aplicações que trabalham com sistema de coordenadas e controle de posicionamento. As principais mensagens para este projeto estão definidas abaixo:

- geometry_msgs/pose: essa mensagem contém duas informações, o *point* e o *quaternion*, que são a posição x, y e z de um ponto no espaço e a sua orientação na forma de um quatérnio, respectivamente.

- geometry_msgs/twist: essa mensagem é definida para representar a velocidade de um ponto no espaço, separada em velocidade linear e velocidade angular, ambos são vetores x, y e z.

- nav_msgs/odometry: esse tipo de mensagem é reservado para sistemas de odometria, fornecendo uma estimativa da posição e velocidade obtidas pelos sensores. A posição é dada como uma mensagem geometry_msgs/pose e a velocidade é dada no formato geometry_msgs/twist.

- sensor_msgs/image: essa mensagem contém a imagem obtida por alguma câmera presente no sistema, no nosso caso, a imagem câmera frontal do Parrot Bebop 2.

5.2 Sistema de captura de movimento

O sistema de captura de movimento Vicon-T40S será utilizado como *ground-truth* para validação da localização métrica obtida pelos três diferentes métodos abordados nesse projeto. Esse sistema de alta precisão de captura de movimento é comercializado pela Vicon®.

A Figura 5 mostra o sistema de captura Vicon-T40S, composto de um conjunto de quatro câmeras e um *software* de captura de movimento. A infraestrutura apresentada está disponível no Laboratório de Sistemas Inteligentes da Universidade de São Paulo, Campus São Carlos [1]

O sistema baseia-se na emissão de luz infravermelha através de uma matriz de LEDs infravermelhos (IR) localizados nas câmeras. Marcadores reflexivos, localizados em pontos estáticos no objeto a ser rastreado, refletem a luz IR que é então detectada pelas câmeras através de um filtro IR óptico. Dessa maneira, apenas os marcadores são reconhecidos, Figura 5.2. A Figura 5.2 ilustra uma câmera Vicon® modelo T40S. Estas comunicam-se com um servidor dedicado via fibra óptica através do protocolo Giganet. As câmeras fornecem informações ao servidor que então aplica os algoritmos para determinação da localização dos marcadores. O processamento fornece informações de posição, orientação em ângulos de Euler e rotação em quatérnios dos objetos rastreados.

O *software* de captura de movimento Tracker da Vicon® faz parte do sistema de captura (Figura 7). Nele são definidas as configurações de operação, comunicação e o sistema de

[1] http://www1.sel.eesc.usp.br/lasi/

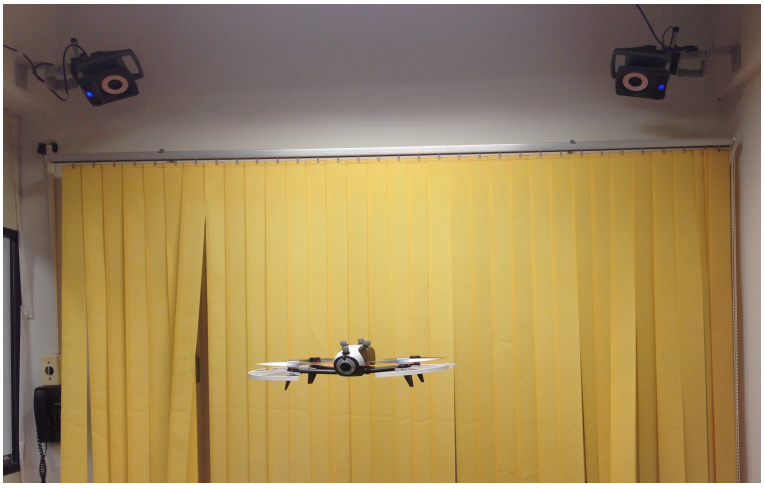


Figura 5 – O sistema Vicon-T40S em uma captura de movimento no laboratório.

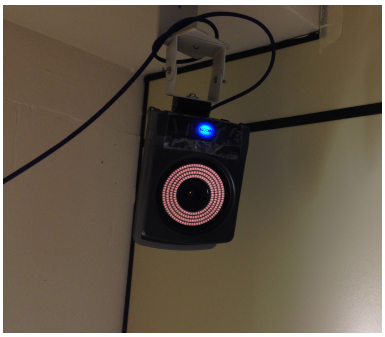
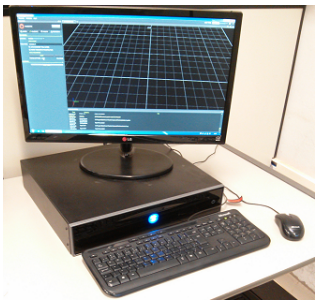


Figura 6 – (a) Quatro marcadores reflexivos são fixados no dorso superior do Parrot Bebop 2; (b) Uma das quatro câmeras Vicon modelo T40S.

coordenadas locais. Além disso, é por meio do Tracker que as informações do servidor dedicado são difundidas na rede Ethernet. O pacote *vicon_bridge* é o pacote que realiza a comunicação do sistema Vicon® com o plataforma ROS.



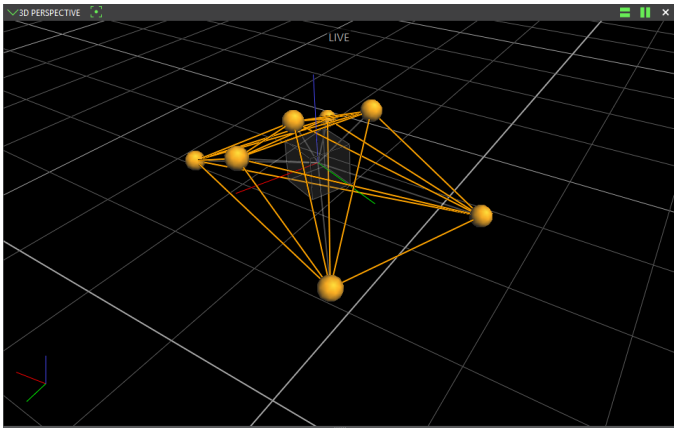


Figura 7 – (a) Computador responsável pela aquisição de dados das câmeras; (b) Através dos marcadores reflexivos, o Parrot Bebop 2 é representado no software Tracker. (Autoria própria)

5.3 Parrot-Sphinx

Parrot-Sphinx é uma ferramenta de simulação de aeronaves quadricóptero da empresa *Parrot*, cujo objetivo é simular e validar estratégias de projetos desenvolvidos antes de aplicar com a aeronave real. Para estas simulações, utiliza-se o simulador *Gazebo*, amplamente empregado em sistemas robóticos, para visualização e simulação da física do quadricóptero. Outra vantagem é a possibilidade de integrar o sistema ROS para controlar a simulação. A Figura 8 apresenta a simulação do quadricóptero utilizando o Parrot-Sphinx juntamente com o Gazebo e a comunicação em ROS.

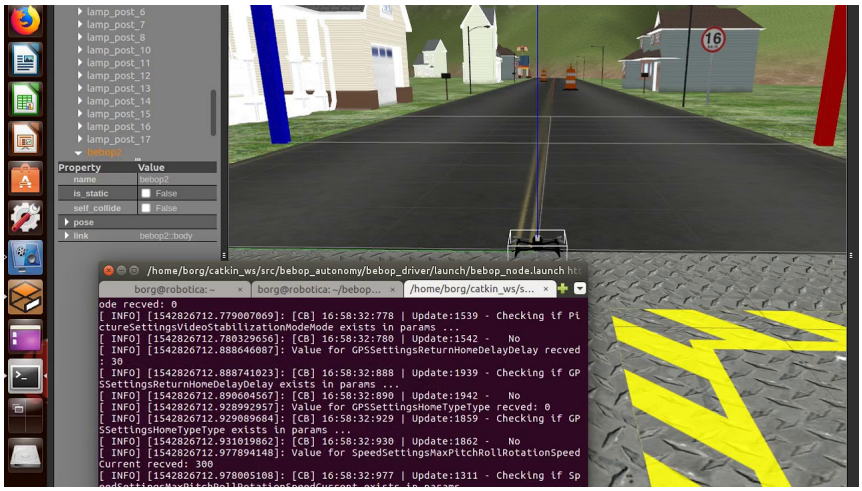


Figura 8 – Simulação do Parrot-Sphinx com o ambiente Gazebo utilizando o quadricóptero Parrot Bebop 2.0 (Retirado de <https://www.youtube.com/watch?v=Uh9lW-SCGQ8&ab_channel=FC3A1biodeMiranda>)

.

5.4 Bebop Autonomy

Bebop Autonomy é um driver do sistema ROS, projetado para o MAV Parrot Bebop, baseado em um pacote oficial da Parrot, o ARDroneSDK3. Este driver facilita a conexão, controle, recebimento de dados, salvamento download de mídias e enviar e rodar rotinas de voo pré-programadas. A parte mais importante na utilização deste pacote é a configuração direta da troca de mensagens entre o Parrot Bebop e o ROS, permitindo o acesso à muitos tópicos importantes relacionados ao estado da aeronave. Com isso podemos destacar alguns tópicos que podem ser enviados e/ou recebidos pelo quadrotor que são essenciais ao projeto:

- Tópicos para leitura
  - Câmera: A imagem da câmera frontal do Bebop é publicada no tópico *image_raw* sendo uma mensagem do tipo sensor_msgs/image.

  - Odometria: Os dados dos sensores de odometria são publicados no tópico *odom*, são mensagens do tipo geometry_msgs/twist, com taxa de update de 5Hz.

❑ Tópicos para envio/controle.

  - Decolagem: Publicar no tópico *takeoff* uma mensagem do tipo std_msgs/Empty.
  - Aterrissagem: Publicar no tópico *land* uma mensagem do tipo std_msgs/Empty.
  - Movimentação: Publicar no tópico *cmd_vel* mensagens do tipo geometry_msgs/twist.

5.5 Drone dev

Drone dev é uma plataforma desenvolvida em (BENEVIDES et al., 2019), baseada no ROS, voltada para o controle robusto e recursivo de MAVs. Ela utiliza os pacotes *vicon_bridge* e *bebop_autonomy* como base. Essa plataforma tem um papel extremamente importante no desenvolvimento do controle deste projeto. Algumas de suas funções como por exemplo a estimação de parâmetros de controle e planejamento de trajetória são essenciais, além de possuir sistemas de segurança para a inicialização e execução testes do controlador. O controle de posição é habilitado contanto que uma mensagem específica esteja sendo enviada ao tópico */bebop/joy*. O tópico *traj_planner* é responsável pela publicação dos *waypoints* da trajetória a ser seguida pelo MAV e pelos pontos de referência para que sejam visualizados em um gráfico, montado pelo tópico *plot_generator*. Essa plataforma ajuda na implementação do controle de posição e envia os comandos de aceleração para o MAV pelo tópico *bebop_driver*.

5.6 SLAM

Algoritmos SLAM (*Simultaneous Localization And Mapping*) consistem em métodos que permitem a robôs móveis autônomos estimar sua posição e simultaneamente obter informações relevantes sobre o ambiente em que se encontram (HULETSKI; KARTASHOV; KRINKIN, 2015). Em 2003, desenvolveu-se o MonoSLAM em (DAVISON, 2003), este foi o primeiro SLAM Visual Monocular. Os SLAM visuais monoculares utilizam apenas uma câmera e consequentemente, apresentam ambiguidade de escala. Uma vez que as imagens obtidas por apenas uma câmera não inferem dados de profundidade, enquanto que imagens estéreas utilizam múltiplas câmeras, de tal forma que é possível obter a estimativa de escala por meio de técnicas de triangulação (DEBEI, 2016). No entanto, os SLAMs monoculares necessitam de hardwares muito mais simples para serem implementados, assim

resultam em sistemas mais leves, compactos e econômicos, logo são muito interessantes para aplicações em MAVs. Em contraposição, os softwares necessários para MonoSLAMs são muito mais complexos apesar de se evitar os problemas de cálculo de *baseline* necessários para o *Stereo* SLAM, descritos em (LEMAIRE et al., 2007). Por consequência, o SLAM Monocular é de grande interesse para diversas áreas de estudo.

5.6.1 ORB-SLAM

Um dos algoritmos SLAM mais utilizados é o ORB-SLAM, desenvolvido em (MUR-ARTAL; MONTIEL; TARDOS, 2015) . O ORB-SLAM é um dos SLAMs mais clássicos e como indica o nome faz uso do detector ORB (RUBLEE et al., 2011). Em implementações como (GRANA et al., 2013), também usa-se a técnica de *Bag of Words* (BoW). A BoW corresponde ao uso de palavras chaves sem ordenação que descrevem os objetos ou características da imagem obtida. Essas palavras são representada em um vetor de ocorrência que permite deduzir diferentes aspectos de uma imagem.

Foram feitos aperfeiçoamentos no ORB-SLAM, nesse caso, o ORB-SLAM 2 (MUR-ARTAL; TARDOS, 2017). O ORB-SLAM 2 é o primeiro SLAM *open source* para câmeras stereo, monocular e RGB-D. Esse SLAM mantém as estruturas principais do ORB-SLAM, no entanto, no ORB-SLAM 2 utiliza-se o DBoW2 (GALVEZ-LÓPEZ; TARDOS, 2012) para relocalização e um gráfico de co-visibilidade (STRASDAT et al., 2011) para reconhecer pontos comuns entre quadros. Esse SLAM Visual também classifica pontos como próximos ou distantes. Tal classificação permite utilizar os pontos próximos para obter um fator de escala e pontos distantes para obter informações precisas de rotação. Para este projeto utilizamos uma versão modificada do ORB-SLAM 2, que possui algumas funcionalidades extras úteis ao desenvolvimento. O ORB-SLAM 2 publica uma série de tópicos importantes ao desenvolvimento deste projeto, alguns deles são:

- orb_slam_mono/pose: esse tópico nos fornece os dados de posição obtidos pelo ORB-SLAM 2 em uma mensagem do tipo geometry_msgs/twist.

- orb_slam_mono/

5.6.2 Ajuste de escala

Como citado na Seção 2.2, o MAV utilizado possui uma câmera monocular. As câmeras monoculares, em contraposição às câmeras estéreo, possuem apenas um "olho". Por causa disso, não existe diferença de perspectiva na imagem obtida. A existência de mais de uma perspectiva é justamente o que fornece noção de profundidade na imagem. Como isso não existe na câmera monocular, necessitamos recorrer à métodos externos para conferir uma escala métrica à imagem, fundamental para realização do controle de posição do MAV.

Métodos de SLAM com câmeras monoculares são muito utilizados devido a seu valor acessível e seu sistema compacto e leve, especialmente para Micro Aéreos Veículos (MAVs). Este tipo de SLAM Visual criam mapas adimensionais, ou seja, não há uma determinação métrica para unidade base desse mapeamento. Consequentemente, não é possível determinar, somente utilizando o SLAM, o fator entre as distâncias no mapa e as distâncias no mundo real.

Para superar esse problema de escala, faz-se uso de diferentes abordagens. Inicialmente, fez-se uso do sistema de posicionamento global, em inglês, *Global Positioning System* (GPS), mas além da necessidade de um outro sensor, em ambientes urbanos a medidas obtidas pelo GPS não são confiáveis (LOTHE et al., 2010). Então, tem-se o uso de sensores de profundidade, que oferecem no decorrer do processo de localização e mapeamento, medidas relativamente confiáveis de distâncias que podem ser utilizadas para estimar o fator de escala como em (URZUA; MUNGUÍA; GRAU, 2017; ESRAFILIAN; TAGHIRAD, 2016). Já em (SUDDERTH et al., 2006), para estimar o fator de escala empregou-se o reconhecimento de objetos, cuja a escala é conhecida. E por fim, existem métodos que combinam medidas de inércia com as informações obtidas pelos algoritmos de SLAM, tal como (GEHRIG et al., 2017; NTZI et al., 2011; TANSKANEN et al., 2013). Em (TANSKANEN et al., 2013), por exemplo, a obtenção do módulo de escala métrica visual da cena capturada consiste em calcular um parâmetro de escala através de mínimos quadrados comparando o deslocamento estimado pela visão através de um conjunto de imagens com o deslocamento estimado de movimento obtido por uma unidade de medida inercial. Em (ENGEL; STURM; CREMERS, 2012; GEHRIG et al., 2017) foram utilizados o método da máxima verossimilhança para determinar o fator de escala do SLAM monocular para o mapa gerado e para a posição do robô. E em (NTZI et al., 2011), resultados bons, precisos e com bons tempo de convergência abaixo da estimativa de escala foram obtidos através do filtro de Kalman estendido que realiza a fusão de dados da unidade de medida inercial com os dados de uma câmera monocular.

Para este problema do vSLAM, simplificamos o modelo matemático para girar em torno da escala métrica do algoritmo visual. A Equação (5.1) define o modelo do sistema:

$$\lambda_{k+1} = \lambda_k + \mathrm{w}_k^{\lambda}, \tag{5.1}$$

Na qual λ_k é o fator de escala baseado an distância viajada em d^V e d^O na iteração k, e w_k^{λ} é o ruido branco Gaussiano. Definindo o vetor de espaço de estados como $\mathbf{x}_{k+1} = [\lambda_{k+1}]$, os vetores de ruído como $\mathbf{w}_k = [\mathrm{w}_k^{\lambda}]$ e $\mathbf{v}_k = [\mathrm{v}_k^{\lambda}]$, as matrizes do sistema como $F_k = 1$, $H_k = d_k^V$, nós podemos definir a representação em espaço de estados como sendo:

$$\mathbf{x}_{k+1} = \lambda_k + \mathbf{w}_k, \tag{5.2}$$

$$z_k = \lambda_k d_k^V + \mathbf{v}_k, \tag{5.3}$$

na qual o vetor de medidas pode ser definido como $\mathbf{y}_k = d_k^O$. As distâncias viajadas d_k^V e d_k^O são atualizadas a cada instante k após o MAV parar, para capturar pequenos movimentos devido à distúrbios externos ou incertezas nas medidas.

5.6.3 Inicialização do SLAM

Antes de que qualquer estimativa seja feita, é necessário inicializar o SLAM, ou seja, fornecer uma imagem com pontos suficientes para que um sistema de coordenadas/nuvem de pontos, sejam criados. Para tal, é realizada uma rotina de voo programada previamente. Este voo inicial é definido com uma trajetória quadrada com lados $d = 30$cm, situada no plano xy, como pode ser visto na Figura 9. Essa rotina é realizada logo após a decolagem do MAV e é baseada somente nos dados de posição fornecidos pela odometria.

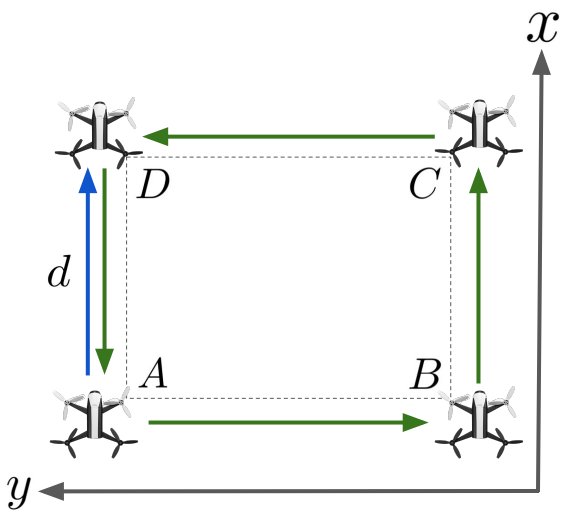


Figura 9 – Rotina de inicialização do SLAM A -> B -> C -> D. Após a inicialização é realizado o voo A -> D para cálculo de escala. (Retirado de (CALDAS et al., 2022)).

5.6.4 Estimativa de Escala

Após a inicialização do SLAM, o MAV realiza um voo em linha reta, parando logo após. Essa parte está representada pela seta azul, também na Figura 9. Esta rotina é feita para se obter a distância d viajada entre os pontos A e D, tanto pela odometria quanto pelo SLAM. O trajeto realizado nos fornece um par de pontos $\{(d^V, d^O)\}$, utilizados para o cálculo de escala durante este intervalo.

5.6.4.1 Método do Filtro de Kalman

Como apresentado na Seção 5.6.2, o modelo em tempo discreto do MAV para cálculo de escala é definido pela Equação (5.1). Com base nesse modelo, um filtro de Kalman

pode ser implementado para estimar o fator de escala para a informação obtida pelo vSLAM. As equações para a fase de predição são:

$$\hat{\lambda}_{k|k-1} = \hat{\lambda}_{k-1|k-1} + \mathbf{w}_k, \tag{5.4}$$

$$\mathbf{P}_{k|k-1} = P_{k-1|k-1} + \mathbf{Q}_k, \tag{5.5}$$

e para a fase de atualização:

$$\tilde{\mathbf{y}}_k = d_k^O - \hat{\lambda}_{k|k-1} d_k^V, \tag{5.6}$$

$$\mathbf{S}_k = d_k^V \mathbf{P}_{k|k-1} d_k^{V^T} + R_k, \tag{5.7}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} d_k^{V^T} \mathbf{S}_k^{-1}, \tag{5.8}$$

$$\hat{\lambda}_{k|k} = \hat{\lambda}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k, \tag{5.9}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k d_k^V)\mathbf{P}_{k|k-1}. \tag{5.10}$$

O critério de parada para as iterações do Filtro de Kalman foi definido como ζ: quando a diferença entra a distância viajada obtida pela odometria e distância viajada obtida pelo vSLAM com a escala calculada, for menor que ζ, as iterações terminam. Após isso, a posição do vSLAM com escala (p_{sORB}^V), começa a ser publicada como um tópico no ROS, dada pela seguinte equação:

$$p_{sORB_k}^V = \hat{\lambda}(p_k^V - p_{D_N}^V) + p_{D_N}^O. \tag{5.11}$$

na qual $p_{D_N}^V$ e $p_{D_N}^O$ são as últimas posições do vSLAM e da odometria, na iteração N, que é o passo final do Filtro de Kalman. Assim sendo, a origem das coordenadas do vSLAM com escala é a ultima posição obtida pela odometria do MAV.

Um resumo das etapas do método proposto baseado em Filtro de Kalman está definido no Algoritmo 2, na seção de Anexos A.1.

Esse estudo teve continuidade no trabalho (CALDAS et al., 2022; CALDAS; INOUE; TERRA, 2022) realizado em parceria com laboratório LASI da USP São Carlos [2] nele são apresentados resultados simulados e experimentais da estimativa de escala. Esse resultados são resumidos nas seções 5.6.5 e 5.6.6.

5.6.5 Simulação do algoritmo de escala

Para avaliação do algoritmo de escala escolhido e comparação com o sistema de Odometria, foi utilizada a plataforma *Sphinx*. Durante a simulação foram realizados dois voos manuais, um com trajetória hexagonal e um com trajetória quadrada. As trajetórias estimadas pela Odometria, pelo vSLAM com Filtro de Kalman e obtida pela *Vicon* foram comparadas durante estes dois cenários e podem ser vistas na Figura 10.

[2] http://www1.sel.eesc.usp.br/lasi/

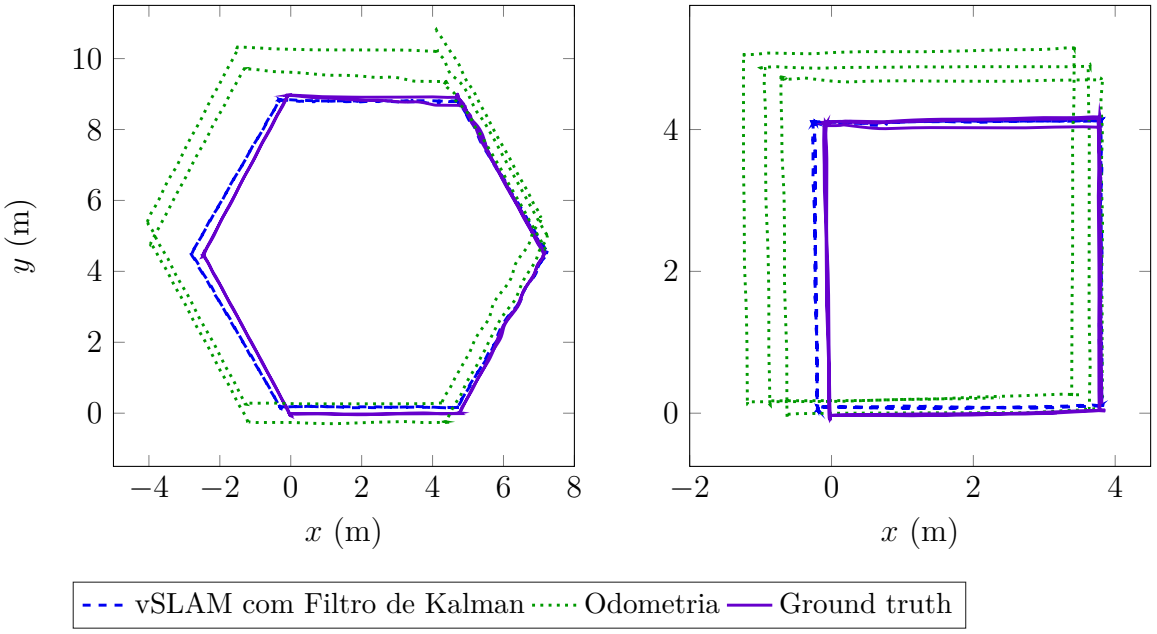


Figura 10 – Comparação da trajetória estimada com o vSLAM e a Odometria interna do Bebop 2. (Retirado de (CALDAS et al., 2022))

É possível observar o acúmulo de erro com o passar do tempo na trajetória estimada pela Odometria, motivo principal de utilizarmos o vSLAM como base para o controle.

5.6.6 Resultados experimentais

O algoritmo de estimação de escala foi testado em um ambiente fechado, utilizando o Bebop 2. Foi realizado um voo manual com trajetória retangular de 1×2 m, comparando os dados obtidos pela Odometria e pelo vSLAM, utilizando o sistema de câmeras Vicon MTS como *ground truth*.

Podemos ver a comparação entre as trajetórias estimadas na Figura 11. Para melhor visualização, apenas a perspectiva xy está sendo mostrada. Uma comparação entre o erro presente nos eixos x, y e z, em relação à distância percorrida no voo, pode ser vista na Figura 12.

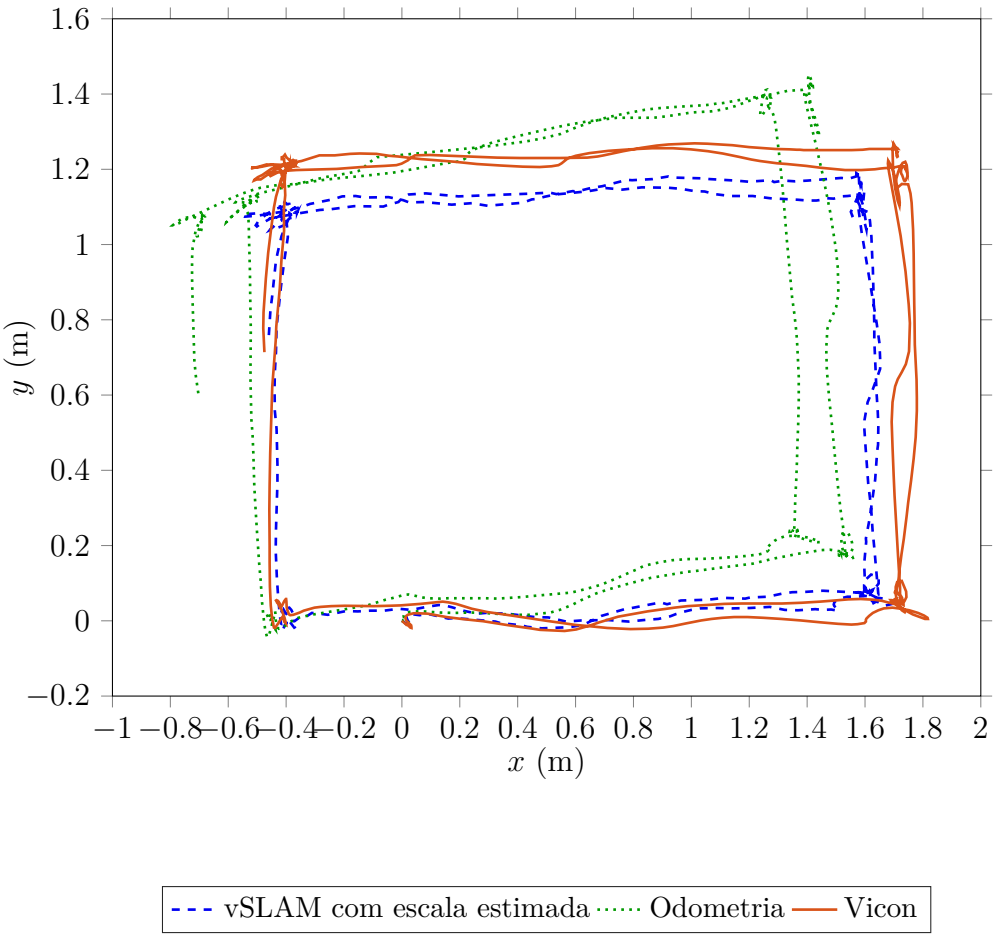


Figura 11 – Comparação entre as trajetórias estimadas pelo vSLAM com Filtro de Kalman e pela Odometria. (Retirado de (CALDAS et al., 2022))

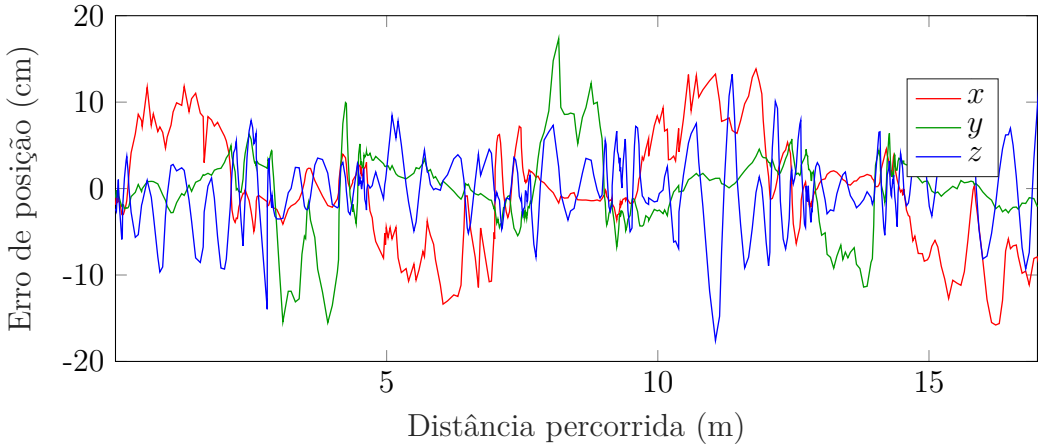


Figura 12 – Erro apresentado no algoritmo proposto, nos eixos x, y, e z, em relação à distância percorrida. (Retirado de (CALDAS et al., 2022)).

Conclusão

Neste trabalho, propomos um controle $\mathcal{H}_\infty$ não linear baseado em processos Gaussianos. Duas abordagens foram definidas, na primeira, o processo Gaussiano é sintonizado por uma lei de controle adaptativo, e na segunda, usamos um processo Gaussiano treinado offline. Também escrevemos o modelo dinâmico de um quadrotor comercial como uma representação em espaço de estado para o controle não linear $\mathcal{H}_\infty$ e implementamos o controle não linear $\mathcal{H}_\infty$ baseado no processo gaussiano para análise comparativa. Com base nos resultados da simulação do modelo Parrot Bebop, analisamos o desempenho e o consumo de energia, verificando também o comportamento dos controladores devido a incertezas paramétricas de massa e inércia e a perturbações externas como rajadas de vento. As abordagens inteligentes (Processo Gaussiano e Rede Neural) mostraram uma melhoria significativa de desempenho ao reduzir a energia consumida sob variações e perturbações paramétricas. O modelo inteligente baseado no processo gaussiano e na rede neural apresentou desempenho semelhante, com o método online do processo gaussiano obtendo o melhor desempenho. Desta forma, o processo Gaussiano poderia ser uma alternativa viável à Rede Neural, como mostram os resultados. Além do desenvolvimento e análise dos algoritmos de controle em simulação, também foi desenvolvido um sistema de obtenção de coordenadas com escala métrica para ambientes internos, integrado à um computador com o *framework* ROS, o que torna possível a implementação dos controles desenvolvidos em um *drone* Parrot Bebop 2.

Referências

BENEVIDES, J. R. S. et al. Parameter estimation based on linear regression for commercial quadrotors. In: **Simpósio Brasileiro de Automação Inteligente (SBAI)**. Ouro Preto, MG: [s.n.], 2019.

_____. Ros-based robust and recursive optimal control of commercial quadrotors. **International Conference on Automation Science and Engineering (CASE)**, 2019.

BERKENKAMP, F.; SCHOELLIG, A. P. Learning-based robust control: Guaranteeing stability while improving performance. In: **International Conference on Intelligent Robots and Systems (IROS)**. [S.l.: s.n.], 2014.

BERNARD, M. et al. Autonomous transportation and deployment with aerial robots for search and rescue missions. **Journal of Field Robotics**, v. 8, p. 914–931, 2011.

BILLS, C.; CHEN, J.; SAXENA, A. Autonomous MAV flight in indoor environments using single image perspective cues. In: **IEEE International Conference on Robotics and Automation (ICRA)**. Shanghai, China: [s.n.], 2011. ISSN 1050-4729.

BONNA, R.; CAMINO, J. Trajectory tracking control of a quadrotor using feedback linearization. In: **XVII International Symposium on Dynamic Problems of Mechanics**. São Sebastião, SP, Brasil: [s.n.], 2015.

BOUABDALLAH, S. et al. PID vs LQ control techniques applied to an indoor micro quadrotor. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems.** [S.l.: s.n.], 2004.

CALDAS, K. A.; INOUE, R. S.; TERRA, M. H. Commercial mav velocity estimation using gaussian process regression for drift reduction. In: IEEE. **IEEE Sensors**. [S.l.], 2022. p. 1–4.

CALDAS, K. A. Q. et al. Autonomous robust navigation system for mav based on monocular cameras. In: **International Conference on Unmanned Aircraft Systems (ICUAS)**. [S.l.: s.n.], 2022. p. 1343–1349.

CAO, G.; LAI, E. M.-K.; ALAM, F. Gaussian process model predictive control of unmanned quadrotors. In: **International Conference on Control, Automation and Robotics (ICCAR)**. HICO, Gyeongju, Korea: [s.n.], 2016.

CASTILLO, P.; R.LOZANO; A.DZUL. Stabilization of a mini rotrocraft with four rotors. In: **IEEE Control Systems Magazine.** [S.l.: s.n.], 2005.

CHANG, Y. C. Neural network-based $\mathcal{H}_\infty$ tracking control for robotic systems. **IEE Proceedings of Control Theory Applications**, v. 147, n. 3, p. 303–311, 2000.

_____. Intelligent robust control for uncertain nonlinear time-varying systems and its application to robotic systems. **IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics**, v. 35, n. 6, p. 1108–1119, 2005.

CHEN, B. S.; CHANG, Y. C.; LEE, T. C. Adaptive control in robotic systems with $\mathcal{H}_\infty$ tracking performance. **Automatica**, v. 33, n. 2, p. 227–234, 1997.

CONCHA, A. et al. Visual-inertial direct SLAM. In: **IEEE International Conference on Robotics and Automation**. Zaragoza, Spain: [s.n.], 2016.

DAVISON, A. J. Real-time simultaneous localisation and mapping with a single camera. In: **Ninth IEEE International Conference**. [S.l.: s.n.], 2003. p. 1403–1410.

DEBEI, S. **An Experimental Comparison of Monocular and Stereo Visual Fastslam**. Dissertação (Mestrado) — Univesita' Degli Studi Di Padova, 2016.

DEMIRCIOGLU, H.; BASTURK, H. I. Adaptive attitude and altitude control of a quadrotor despite unknown wind disturbances. In: **IEEE Annual Conference on Decision and Control (CDC)**. Melbourne, Australia: [s.n.], 2017.

DIKMEN, I. C. et al. Attitude control of a quadcopter. In: **International Conference on Recent Advances in Space Technologies.** Chennai, India: [s.n.], 2010.

ENGEL, J.; SCHÖPS, T.; CREMERS, D. LSD-SLAM: Large-scale direct monocular SLAM. In: **European Conference on Computer Vision (ECCV)**. Zurich, Switzerland: [s.n.], 2014.

ENGEL, J.; STURM, J.; CREMERS, D. Camera-based navigation of a low-cost quadrocopter. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Vilamoura, Portugal: [s.n.], 2012.

ESCARENO, J.; C.SALAZAR-CRUZ; LOZANO, R. Embedded control of a four-rotor UAV. In: **American Control Conference.** [S.l.: s.n.], 2006.

ESRAFILIAN, O.; TAGHIRAD, H. D. Autonomous flight and obstacle avoidance of a quadrotor by monocular SLAM. **International Conference on Robotics and Mechatronics (ICRoM)**, 2016.

FAIGL, J. et al. Surveillance planning with localization uncertainty for UAVs. In: **Israeli Conference on Robotics**. Herzlia, Israel: [s.n.], 2010. Disponível em: <http://www.icr2010.org.il/>.

FARREL, J. A. **Aided Navigation GPS with High Rate Sensors**. [S.l.]: The McGRaw-Hill Companies, 2008.

FORSTER, C.; PIZZOLI, M.; SCARAMUZZA, D. SVO: Fast semi-direct monocular visual odometry. In: **IEEE International Conference on Robotics and Automation (ICRA)**. Hong Kong, China: [s.n.], 2014.

FOUNDATION, O. S. R. **ROS: Robot Operating System**. 2007. <http://www.ros.org/>. Disponível em: <http://www.ros.org/>.

GALVEZ-LÓPEZ, D.; TARDOS, J. D. Bags of binary words for fast place recognition in image sequences. **IEEE Transactions on Robotics**, Institute of Electrical and Electronics Engineers (IEEE), v. 28, n. 5, p. 1188–1197, 2012. ISSN 15523098.

GEHRIG, D. et al. Scale-corrected monocular-slam for the ar.drone 2.0. ETH Library. 2017.

GRANA, C. et al. A fast approach for integrating ORB descriptors in the bag of words model. In: SNOEK, C. G. M. et al. (Ed.). **Multimedia Content and Mobile Devices**. [S.l.]: SPIE, 2013. p. 866709. ISSN 0277786X.

HEHN, M.; D'ANDREA, R. Quadrocopter trajectory generation and control. In: **The International Federation of Automatic Control**. [S.l.: s.n.], 2011.

HERWITZ, S. et al. Imaging from an unmanned aerial vehicle: agricultural surveillance and decision support. **Computers and Electronics in Agriculture**, v. 44, p. 49–61, 07 2004.

HESCH, J. A. et al. Camera-IMU-based localization: Observability analysis and consistency improvement. **The International Journal of Robotics Reseach**, v. 33, n. 1, p. 182–201, 2014.

HIGUCHI, K.; SHIMADA, T.; REKIMOTO, J. Flying sports assistant: External visual imagery representation for sports training. In: **Augmented Human International Conference**. New York, USA: [s.n.], 2011. ISBN 978-1-4503-0426-9. Disponível em: <http://doi.acm.org/10.1145/1959826.1959833>.

HOFFMANN, G. M. et al. Quadrotor helicopter flight dynamics and control: Theory and experiment. In: . [S.l.: s.n.], 2007.

HUANG, H. et al. Aerodynamics and control of autonomous quadrotor helicopter in aggressive maneuvering. In: **IEEE International Conference on Robotics and Automation**. Kobe, Japan: [s.n.], 2009.

HULETSKI, A.; KARTASHOV, D.; KRINKIN, K. Evaluation of the modern visual SLAM methods. In: **2015 Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT)**. [S.l.]: IEEE, 2015. p. 19–25.

INOUE, R. S. **Controle robusto descentralizado de movimentos coordenados de robôs heterogêneos**. Tese (Doutorado) — Escola de Engenharia de São Carlos da Universidade de São Paulo, São Carlos, SP, Brasil, 2012.

JOHANSSON, R. Quadratic optimization of motion coordination and control. **IEEE Transactions on Automatic Control**, v. 35, n. 11, p. 1197–1208, 1990.

KLEIN, G.; MURRAY, D. Parallel tracking and mapping for small AR workspaces. In: **IEEE and ACM International Symposium on Mixed and Augmented Reality**. Nara, Japan: [s.n.], 2007.

_____. Parallel tracking and mapping on a camera phone. In: **IEEE International Symposium on Mixed and Augmented Reality (ISMAR)**. Orlando, FL, USA: [s.n.], 2009.

LEMAIRE, T. et al. Vision-based SLAM: Stereo and monocular approaches. **International Journal of Computer Vision**, v. 74, n. 3, p. 343–364, 2007. ISSN 09205691.

LEWIS, F. L.; ABDALLAH, C. T.; DAWSON, D. M. **Control of robot manipulators**. New York: Macmillan., 1993.

LI, M.; KIM, B. H.; MOURIKIS, A. Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera. In: **IEEE International Conference on Robotics and Automation (ICRA)**. Riverside, CA, USA: [s.n.], 2013. ISSN 1050-4729.

LI, M.; MOURIKIS, A. Improving the accuracy of EKF-based visual-inertial odometry. In: **IEEE International Conference on Robotics and Automation (ICRA)**. Saint Paul, MN, USA: [s.n.], 2012. ISSN 1050-4729.

LIU, Y.; MA, J.; TU, H. Robust command filtered adaptive backstepping control for a quadrotor aircraft. **Journal of Control Science and Engineering.**, v. 2018, p. 1854648:1–1854648:9, 2018.

LOTHE, P. et al. Real-time vehicle global localisation with a single camera in dense urban areas: Exploitation of coarse 3D city models. In: **2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition**. [S.l.]: IEEE, 2010. p. 863–870. ISSN 10636919.

LUUKKONEN, T. **Modeling and control of quadcopter**. [S.l.], 2011.

MADANI, T.; BENALLEGUE, A. Backstepping control for a quadrotor helicopter. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems.** [S.l.: s.n.], 2006.

_____. Sliding mode observer and backstepping control for a quadrotor unmanned aerial vehicles. In: **American Control Conference**. New York, NY, USA: [s.n.], 2007.

MAYER, H.; BARTELSEN, J. Automated 3D reconstruction of urban areas from networks of wide-baseline image sequences. In: **The International Archives of the Photogrametry, Remote Sensing and Spatial Information Sciences**. Beijing, China: [s.n.], 2008.

MUR-ARTAL, R.; MONTIEL, J. M. M.; TARDOS, J. D. ORB-SLAM: A versatile and accurate monocular SLAM system. **IEEE Transactions on Robotics**, Institute of Electrical and Electronics Engineers (IEEE), v. 31, n. 5, p. 1147–1163, 2015. ISSN 15523098.

MUR-ARTAL, R.; TARDOS, J. D. Probabilistic semi-dense mapping from highly accurate feature-based monocular slam. In: **Robotics: Science and Systems Conference**. Rome, Italy: [s.n.], 2015.

_____. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-d cameras. **IEEE Transactions on Robotics**, Institute of Electrical and Electronics Engineers (IEEE), v. 33, n. 5, p. 1255–1262, 2017. ISSN 15523098.

NG, W. S.; SHARLIN, E. Collocated interaction with flying robots. In: **IEEE International Symposium on Robot and Human Interactive Communication**. Atlanta, GA, USA: [s.n.], 2011.

NOGUEIRA, S. L. et al. Experimental investigation on adaptive robust controller designs applied to constrained manipulators. **Sensors**, v. 13, p. 5181–5204, 2013.

NTZI, G. et al. Fusion of imu and vision for absolute scale estimation in monocular slam. **Journal of Intelligent & Robotic Systems**, v. 61, p. 287–299, 2011.

ORTIZ, J. P.; MINCHALA, L. I.; REINOSO, M. J. Nonlinear robust h-infinity pid controller for the multivariable system quadrotor. In: **IEEE Latin America Transactions (LATAM T)**. [S.l.: s.n.], 2016.

OWENS, S.; SYCARA, K.; SCERRI, P. Using immersive 3D terrain models for fusion of uav surveillance imagery. In: **AIAA Infotech Aerospace Conference**. Washington, USA: [s.n.], 2009.

PAZELLI, T. F.; TERRA, M. H.; SIQUEIRA, A. A. Experimental investigation on adaptive robust controller designs applied to a free-floating space manipulator. **Control Engineering Practice**, v. 19, p. 395–408, 2011.

PINTO, A. O. et al. High-level modeling and control of the bebop 2 micro aerial vehicle. In: **International Conference on Unmanned Aircraft Systems (ICUAS)**. Athens, Greece: [s.n.], 2020.

RAFFO, G. V.; ORTEGA, M. G.; RUBIO, F. R. An integral predictive/nonlinear infinite h control structure for a quadrotor helicopter. **Automatica**, p. 10, 2010.

_____. Robust nonlinear control for path tracking of a quad-rotor helicopter. **Asian Journal of Control**, v. 17, n. 1, p. 142–156, 2015.

RASMUSSEN, C. E.; WILLIAMS, C. K. I. **Gaussian Processes for Machine Learning**. [S.l.: s.n.], 2006.

REIS, G. A. dos. Controle $\mathcal{H}\infty$ não linear de robôs móveis com rodas. **Dissertação apresentada à Escolha de Engenharia de São Carlos da Universidade de São Paulo**, 2005.

RUBLEE, E. et al. ORB: An efficient alternative to SIFT or SURF. In: **2011 International Conference on Computer Vision**. [S.l.]: IEEE, 2011. p. 2564–2571. ISSN 1550-5499.

SADAT, S. A. et al. Feature-rich path planning for robust navigation of MAVs with Mono-SLAM. In: **IEEE International Conference on Robotics and Automation (ICRA)**. Hong Kong, China: [s.n.], 2014.

SANTANA, L. V.; BRANDãO, A. S.; SARCINELLI-FILHO, M. An automatic flight control system for the ar.drone quadrotor in outdoor environments. In: **Workshop on Research, Education and Development of Unmanned Aerial Systems**. Cancun, Mexico: [s.n.], 2015.

SCHLEICHER, D. et al. Real-time hierarchical GPS aided visual SLAM on urban enviroments. **Journal of Field Robotics**, v. 27, n. 5, p. 609–631, 2010.

SCHÖPS, T.; ENGEL, J.; CREMERS, D. Semi-dense visual odometry for AR on a smartphone. In: **IEEE International Symposium on Mixed and Augmented Reality**. Munich, Germany: [s.n.], 2014.

SHEPARD, D. P.; HUMPHREYS, T. E. High-precision globally-referenced position and attitude via a fusion of visual SLAM, carrier-phase-based GPS, and inertial measurements. In: **IEEE/ION Position, Location and Navigation Symposium (PLANS)**. [S.l.: s.n.], 2014.

STRASDAT, H. et al. Double window optimisation for constant time visual SLAM. In: **2011 International Conference on Computer Vision**. [S.l.]: IEEE, 2011. p. 2352–2359. ISSN 1550-5499.

SUDDERTH, E. et al. Depth from familiar objects: A hierarchical model for 3D scenes. In: **2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)**. [S.l.]: IEEE, 2006. v. 2, p. 2410–2417. ISSN 10636919.

SUZUKI, T. et al. 3D terrain reconstruction by small unmanned aerial vehicle using SIFT-based monocular SLAM. **Journal of Robotics and Mechatronics**, v. 23, n. 2, p. 292–301, 2011.

TANSKANEN, P. et al. Live metric 3D reconstruction on mobile phones. In: **IEEE International Conference on Computer Vision**. Washington, DC, USA: [s.n.], 2013. ISBN 978-1-4799-2840-8. Disponível em: <http://dx.doi.org/10.1109/ICCV.2013.15>.

TAYEBI, A.; MCGILVRAY, S. Attitude stabilization of a four-rotor aerial robot. In: **IEEE Conference on Decision and Control**. [S.l.: s.n.], 2004.

URZUA, S.; MUNGUÍA, R.; GRAU, A. Vision-based SLAM system for MAVs in GPS-denied environments. **International Journal of Micro Air Vehicles**, SAGE Publications, v. 9, n. 4, p. 283–296, 2017. ISSN 1756-8293.

WANG, L.; THEODOROU, E. A.; EGERSTEDT, M. Safe learning of quadrotor dynamics using barrier certificates. In: **International Conference on Robotics and Automation (ICRA)**. Marina Bay Sands, Singapore: [s.n.], 2017.

WEISS, S. et al. Intuitive 3D maps for mav terrain exploration and obstacle avoidance. In: **International Conference on Unmanned Aerial Vehicles**. [S.l.: s.n.], 2010.

YUAN, C.; ZHANG, Y.; LIU, Z. A survey on technologies for automatic fores fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques. **Canadian Journal of Forest Research**, v. 45, p. 783–792, 2015.

ZEMALACHE, K. M.; BEJI, L. C.; H.MARREF. Control of an under-actuated system: Application to a four rotors rotorcraft. In: **IEEE International Conference on Robotic and Biomimetics.** Shatin, N.T. China: [s.n.], 2005.

ZUO, Z. Trajectory tracking control design with command-filtered compensation for a quadrotor. In: **IET Control Theory & Applications**. [S.l.: s.n.], 2010.

Anexos

ANEXO A

Material Adicional

A.1 Algoritmos

Algoritmo 1 Sequência para encontrar os parâmetros de 3.1, sendo que i sobrescrito corresponde à iteração atual. Obtida de (REIS, 2005)

1: Escolha um $\gamma > 0$ de valor elevado.
2: Escolha também $t_{11}^i >> \sqrt{q_{11} r_u}$.
3: Utilizando o valor de t_{11}^i calcule, através de 3.8, o nível de atenuação γ^i.
4: Calcule t_{12}^i em 3.9 utilizando t_{11}^i e γ^i.
5: Calcule k^i em 3.10 usando t_{11}^i, t_{12}^i e γ^i.
6: Se $k^i > 0$ e $\gamma^i < \gamma$ então
    $\gamma \leftarrow \gamma^i$
    $t_{11} \leftarrow t_{11}^i$
    $t_{12} \leftarrow t_{12}^i$
    $k \leftarrow k^i$
7: Se $\gamma^i > \gamma^{i-1}$ então pare. Caso contrário continue.
8: Diminua t_{11}^i.
9: Se $t_{11}^i > \sqrt{q_{11} r_u}$ então retorne ao passo 3. Caso contrário pare.

Algoritmo 2 Método de cálculo de escala baseado em Filtro de Kalman

Inicialização:

1: Decolagem do MAV.
2: Rotina de inicialização do vSLAM mostrada na Figura 9 pelas setas verdes.
3: Voo em linha reta mostrado na Figura 9 pela seta azul.

Cálculo da escala:

1: Definir $\hat{\lambda}_0 = 1$.
2: **while** $|d_k^O - \hat{\lambda}_{k,k} d_k^V| > \zeta$ **do**
3: Atualização d_k^V e d_k^O medidas.
4: $\hat{\lambda}_{k|k-1} = \hat{\lambda}_{k-1|k-1} + \mathbf{w}_k$
5: $\mathbf{P}_{k|k-1} = \mathbf{P}_{k-1|k-1} + \mathbf{Q}_k$
6: $\tilde{\mathbf{y}}_k = d_k^O - \hat{\lambda}_{k|k-1} d_k^V$
7: $\mathbf{S}_k = d_k^V \mathbf{P}_{k|k-1} d_k^{V^T} + R_k$
8: $\mathbf{K}_k = \mathbf{P}_{k|k-1} d_k^{V^T} \mathbf{S}_k^{-1}$
9: $\hat{\lambda}_{k|k} = \hat{\lambda}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k$
10: $\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k d_k^V) \mathbf{P}_{k|k-1}$
11: $k = k + 1$
12: **end while**

Atualização do vSLAM com escala estimada:

1: Armazenamento de p_N^V e p_N^O.
2: **while** Rastreamento pelo ORB-SLAM2 ativo **do**
3: $p_{sORB_k}^V = \hat{\lambda}(p_k^V - p_N^V) + p_N^O$.
4: **end while**