UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE UM ALGORITMO DE PLANEJAMENTO DE TRAJETÓRIA EM AMBIENTES DESCONHECIDOS E NÃO ESTRUTURADOS PARA UAVS

LIDIA GIANNE SOUZA DA ROCHA

ORIENTADORA: PROFA. DRA. KELEN CRISTIANE TEIXEIRA VIVALDINI

São Carlos – SP
28 de outubro de 2021

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE UM ALGORITMO DE PLANEJAMENTO DE TRAJETÓRIA EM AMBIENTES DESCONHECIDOS E NÃO ESTRUTURADOS PARA UAVS

LIDIA GIANNE SOUZA DA ROCHA

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Inteligência Artificial

Orientadora: Profa. Dra. Kelen Cristiane Teixeira Vivaldini

São Carlos – SP

28 de outubro de 2021

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Defesa de Dissertação de Mestrado da candidata Lidia Gianne Souza da Rocha, realizada em 30/08/2021.

Comissão Julgadora:

Profa. Dra. Kelen Cristiane Teixeira Vivaldini (UFSCar)

Prof. Dr. Roberto Santos Inoue (UFSCar)

Prof. Dr. Marcelo Becker (USP)

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

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

AGRADECIMENTOS

Gostaria de agradecer a minha família pelo amor incondicional e por sempre apoiar minhas decisões.

A minha orientadora, Kelen Vivaldini, por todo suporte, pela paciência, pelos conselhos, pelas conversas, e por ter acreditado em min.

Aos professores, Roberto Inoue e Marcelo Becker, pelas orientações ao decorrer do trabalho.

Aos meus amigos, Lorena Chacon, Sidnir Carlos, Kaíque Pinto, Victória Barros, João Soares, por estarem comigo diariamente dando suporte emocional e fazendo as coisas ficarem mais leves.

Aos amigos de laboratório, João Benevides, Kenny Caldas, Diego Soler, Igor Araújo, por me guiarem e ajudarem a chegar nesse momento.

Gostaria de agradecer à CAPES pelo apoio financeiro. Agradecer também a parceria com a equipe do *Flying* U2 e os laboratórios de pesquisa LASI - ESSC (USP) e LARIS - UFSCar.

RESUMO

ROCHA, L. G. S. (2021). Desenvolvimento de um Algoritmo de Planejamento de Trajetórias em Ambientes Desconhecidos e Não Estruturados para UAVs. 110p Dissertação de Mestrado – Departamento de Computação, Universidade Federal de São Carlos, São Carlos, 2020.

Para tornar um UAV (*Unmanned Aerial Vehicle*) autônomo é necessário que o mesmo realize ações sem a interferência de um ser humano em suas missões. Diversas missões autônomas de UAVs, independente da área de operação, necessitam de um planejamento de trajetória, podendo atuar em ambientes desconhecidos, 3D, não estruturados, e com obstáculos dinâmicos dependendo da missão designada. Por este motivo, é essencial que o algoritmo seja capaz de desviar de obstáculos, já que nem sempre o ambiente será conhecido. Bem como, levar em consideração no planejamento as restrições de movimentos para realizar curvas suavizadas. Neste contexto, algoritmos de planejamento de trajetória são adotados podendo ser utilizado técnicas clássicas, meta-heurísticas ou de aprendizado de máquina. Um aspecto a ser analisado é que dentre as técnicas existentes e mais utilizadas, qual seria a melhor técnica para atuar nestes ambientes. Desta forma, este trabalho visa analisar as métricas de tempo e distância percorrida, além da variância, média e desvio padrão de cada métrica das técnicas A*, APF, PRM, RRT, RRT-C, PSO, GWO, GSO e RL considerando ambientes desconhecidos, 3D, não estruturados e com obstáculos dinâmicos. Os algoritmos que obtiverem os melhores resultados são testados em 3D no simulador para UAV, sendo realizada uma análise mais profunda, considerando *completeness*, distância, tempo, uso da cpu, uso da memória, prevenção de colisão, e robustez. O teste final para validar os algoritmos de planejamento de trajetória é feito em ambiente real.

Palavras-chave: UAV, planejamento de trajetória, obstáculo dinâmico, ambiente não estruturado, ambiente desconhecido

ABSTRACT

ROCHA, L. G. S. (2021). Development of a Path Planning Algorithm in Unknown and Unstructured Environments for UAVs. 110 MSc Dissertation (Master) – Departamento de Computação, Universidade Federal de São Carlos, São Carlos, 2020.

To make a UAV (Unmanned Aerial Vehicle) autonomous, it must perform actions without human interference in its missions. Regardless of the area of operation, several autonomous UAV missions require path planning, being able to operate in unknown, 3D, unstructured environments, and with dynamic obstacles depending on the assigned mission. Thus, it is essential that the algorithm can avoid obstacles since the environment will not always be known. As well as take into account the movement restrictions to achieve smooth curves in the planning. Several path planning algorithms can be adopted among classic, meta-heuristic, or machine learning techniques. One observable aspect among the existing and most used techniques is, "which would be the best technique to work in these environments". The algorithms that obtain the best results are tested in 3D in the UAV simulator. A deeper analysis is performed, considering completeness, distance, time, CPU usage, memory usage, collision prevention, and robustness. The final test to validate the trajectory planning algorithms is done in a real environment.

Keywords: UAV, path planning, dynamic obstacle, unstructured environment, unknown environment

LISTA DE FIGURAS

LISTA DE TABELAS

GLOSSÁRIO

ACO – *Ant Colony Optimization*

AEP – *Autonomous Exploration Planner*

AI – *Artificial Intelligence*

ALAN – *Adaptive Learning for Multi-Agent Navigation*

ANAC – *Agência Nacional de Aviação Civil*

APF – *Artificial Potential Field*

BA – *Bat Algorithm*

BIM – *Building Information Model*

BOA – *Biogeography Optimization Algorithm*

CADRL – *Collision Avoidance with Deep Reinforcement Learning*

CS – *Cuco Search*

DC – *Departamento de Computação*

DDQN – *Double Deep Q Network*

DRL – *Deep Reiforcement Learning*

ESDF – *Euclidean Signed Distance Function*

EUA – *Estados Unidos da América*

FAA – *Federal Aviation Administration*

FA – *Firefly Algorithm*

GA – *Genetic Algorithm*

GNSS – *Global Navigation Satellite System*

GPS – *Global Position System*

GSO – *Glowworm Swarm Optimization*

GWO – *Grey Wolf Optimization*

HPA* – *Hierarchical Path-Finding Star*

IBA – *Integrated Bat Algorithm*

IMU – *Inertial Measurement Unit*

IoT – *Internet of Things*

LQR – *Linear Quadratic Regulator*

LiDAR – *Light detection and Ranging*

MDP – *Markov Decision Process*

MEA* – *Means-Ends Analysis Star*

MPC – *Model Predictive Control*

MPNet – *Motion Planning Networks*

MRS – *Muti Robot Systems*

MSOS – *Modified Symbiotic Organisms Search*

ORCA – *Optimal Reciprocal Collision Avoidance*

PRM – *Probabilistic Roadmap*

PSO – *Particle Swarm Optimization*

R-LQR – *Robust Linear Quadratic Regulator*

RH-NBV – *Receding Horizon Next-Best-View*

RL – *Reiforcement Learning*

ROS – *Robot Operating System*

RRT* – *Rapid Exploring Random Tree Star*

RRT-C – *Rapid Exploring Random Tree Connect*

RRT – *Rapid Exploring Random Tree*

RST – *Recursive and Smoothed Trajectory*

SCA – *Sine Cosine Algorithm*

SIL – *Software-in-the-Loop*

SSA – *Salp Swarm Algorithm*

UAV – *Unmanned Aerial Vehicle*

UAV – *Unmanned Aerial Vehicle*

UFSCar – *Universidade Federal de São Carlos*

VIN – *Value Iteration Networks*

WOA – *Whale Optimization Algorithm*

WPN – *Waypoint Planning Networks*

SUMÁRIO

Capítulo 1

INTRODUÇÃO

Neste capítulo apresenta-se a contextualização, a motivação, os objetivos para o desenvolvimento deste trabalho, assim como as contribuições. Por fim, a estrutura em que a dissertação está dividida.

1.1 Contextualização

Nos últimos anos, UAVs (*Unmanned Aerial Vehicles* - Veículos Aéreos Autônomos) veem se tornando cada vez mais comum e usuais (TSIATSIS et al., 2018). A Agência Nacional de Aviação Civil (ANAC) relatou que a utilização de UAVs para fins profissionais cresceu 36% entre o ano de 2019 à julho de 2020 no Brasil. De acordo com a FAA (*Federal Aviation Administration*), há 407 mil UAVs profissionais nos Estados Unidos (EUA), com previsão de triplicar até 2023.

Para fazer os primeiros UAVs se locomoverem, era necessário pré-planejar ou pilotá-los em tempo real (STEINMETZ, 1927) (GUTIERREZ; D'HARO; BANCHS, 2016) (FREIMUTH; KÖNIG, 2018). Com o passar dos anos, a tecnologia foi evoluindo e os UAVs foram se tornando mais autônomos sendo capazes de determinar suas próprias trajetórias (PATLE et al., 2019) (AGGARWAL; KUMAR, 2020), e assim passaram a ser utilizados em missões tais como: operações de resgate em áreas acidentadas (KULKARNI et al., 2020); monitoramento de condições de estradas (BREEN et al., 2020); que exigem alta precisão, como pintura ou fotogrametria (VEMPATI et al., 2019) (PULITI et al., 2020), vigilância (VANDENBERG; SWEARS, 2020), agricultura (MONDAL et al., 2020), construção (DUTTA et al., 2020), busca e salvamento (ALOTAIBI; ALQEFARI; KOUBAA, 2019), monitoramento de ambientes (VIVALDINI et al., 2018), detecção de incêndios em florestas (NGUYEN et al., 2021), serviços de entrega (SHE; OUYANG, 2021), entre outros.

Podemos observar que o planejamento de trajetória é essencial para estas missões pois visa encontrar um caminho contínuo para o robô se locomover entre o ponto inicial e o objetivo (GALVEZ; DADIOS; BANDALA, 2014) (LI; YANG; XU, 2019), e tem a finalidade de determinar a melhor trajetória entre esses pontos para o UAV seguir de forma autônoma, sem colidir e sendo o mais suavizada possível, obedecendo as restrições do UAV.

Outra característica é que estas missões podem envolver cenários completamente diferentes, atuando em ambientes 3D, não estruturados, desconhecidos e com obstáculos dinâmicos. Algumas características de cada tipo de ambiente que o algoritmo de planejamento de trajetória a ser desenvolvido nessa dissertação se propõe a resolver é mostrada a seguir:

- **Ambiente 3D:** Nem sempre é possível completar uma missão utilizando apenas um mapa 2D. Há situações em que o UAV precisará passar por baixo ou por cima do obstáculo, como por exemplo nas missões em cavernas (SAM et al., 2020). É comum encontrar pesquisas de UAVs realizando o planejamento de trajetória 2D, porém não há muitos que o implementam em 3D, sendo que em várias missões é necessário que o UAV consiga se locomover em 3D, como por exemplo: reconhecimento de ambiente (CHENG; KELLER; KUMAR, 2008), missões de busca e resgate (HEIDARI; ABBASPOUR, 2014) (FAESSLER et al., 2016), reconstrução de imagem 3D (KOCH; KÖRNER; FRAUNDORFER, 2019), fotogrametria (COMBA et al., 2018), exploração (WEISS et al., 2011), entre outros.

- **Ambiente desconhecido:** Há missões em que o UAV não terá conhecimento do ambiente em que irá trabalhar, por isso é necessário que ele consiga se movimentar em ambientes desconhecidos (SAM et al., 2020) (SUDHAKAR et al., 2020). Diversas aplicações para resolver esse tipo de problema foram desenvolvidas, como o reconhecimento de terrenos (SCHMID et al., 2020), operações de resgate (THRUN et al., 2003), limpeza (LEE; BAEK; OH, 2011), e vigilância (VALLEJO et al., 2009). Tendo como objetivo principal a exploração de um ambiente com informações limitadas, buscando a localização a partir de pontos específicos ou com uma exploração completa.

- **Ambiente não estruturado:** Pode ser necessário que o UAV entre em situações de risco em ambientes mais complexos, como grutas (SAM et al., 2020), florestas (SUDHAKAR et al., 2020) ou resgate em uma área de risco (LLASAG et al., 2019), ou seja, ambientes não estruturados são aqueles que possuem obstáculos posicionados aleatoriamente, impedindo qualquer formação lógica do ambiente, carecendo de uma orientação implícita (BLÖSCH et al., 2010) (FRANCIS et al., 2017). Sendo assim, é importante que o UAV

esteja preparado para adentrar em ambientes complexos para concluir sua missão, podendo dessa maneira, diminuir os riscos de vida para o ser humano, que normalmente realiza a tarefa.

- **Obstáculos dinâmicos:** Os ambientes tendem a se modificar com o passar do tempo. Como por exemplo passagens de pessoas e/ou animais, objetos caindo (galhos de árvore, pedras em áreas de risco), entre outros. Esses obstáculos são descobertos apenas quando já estão perto do UAV, precisando de uma técnica de replanejamento em tempo real para desviar do obstáculo em tempo hábil e continuar a execução da rota. Essas situações podem ser vistas em florestas (SUDHAKAR et al., 2020) e cavernas (SAM et al., 2020), ou ambientes urbanos como missões de *delivery* (LI et al., 2017). Foram desenvolvidos diversas pesquisas para desviar de obstáculos dinâmicos (YERSHOVA et al., 2005) (EVERETT; CHEN; HOW, 2018) (KAUFMANN et al., 2018) (YAN; XIANG; WANG, 2019) (LI et al., 2020) (SONG et al., 2020).

1.2 Motivação

Segundo a ANAC, o uso de UAVs cresceu mais de 165% nos últimos 3 anos, sendo que 40% são para finalidades profissionais como agricultura, monitoramento, *delivery*, entre outros. Além disso, o mercado global de UAVs na área de inspeção deve atingir um valor de $32 bilhões de dólares até 2027 (MARKET, 2021). Com isso, é possível perceber que o mundo está passando por uma mudança de paradigmas. Há alguns anos atrás era algo fora de qualquer escopo que um robô pudesse fazer a tarefa de um humano. Porém, atualmente estão sendo desenvolvidos robôs aéreos com essa capacidade.

Profissionalmente, os UAVs veem sendo utilizados na agricultura (POPOVIĆ et al., 2017a), em serviços de *delivery* (SHI; NG, 2018), busca e resgate (JUAN; SANTOS; ANDÚJAR, 2018), vigilância (CURIAC; VOLOSENCU, 2015), pintura (VEMPATI et al., 2018), inspeção e monitoramento (CUI et al., 2017) (POPOVIĆ et al., 2017b) (LI et al., 2020), entre outros. Ao utilizar técnicas de Planejamento de Trajetória para otimizar esses serviços é possível completá-los com mais eficiência e em menor tempo.

Em todas as missões citadas anteriormente, o UAV precisa se locomover entre 2 pontos, ou mais. Em um cenário pequeno, ou onde o UAV precise se locomover linearmente entre dois pontos, pode parecer uma missão simples de se solucionar. Porém, nenhuma das missões citadas anteriormente envolve um cenário como esse. Como por exemplo: em missões de *delivery*, o UAV precisará se locomover em ambientes não estruturados, e pode precisar navegar por áreas

desconhecidas; para realizar busca e resgate de pessoas, ele voará por ambientes desconhecidos. Além disso, nessas missões é possível que passe um animal ou uma pessoa no caminho do UAV, então o mesmo precisará replanejar a rota visando desviar do obstáculos e minimizar os custos para chegar no seu objetivo. Todas essas ações podem ser feitas através de um algoritmo de planejamento de trajetória, o qual proporciona uma maior precisão da execução das missões.

Há diversas pesquisas sobre estudos de técnicas de planejamento de trajetória, como será apresentado no Capítulo 2. Todas as técnicas são capazes de se movimentar nos mais diversos ambientes, porém cada algoritmo apresenta melhor desempenho em diferentes aplicações, pois terão objetivos específicos. Os objetivos podem variar entre chegar mais rápido no objetivo, explorar mais o ambiente, manter a maior distância possível até os obstáculos, entre outros. Atualmente, é possível separar as principais missões em que os UAVs estão sendo utilizados e definir qual será o melhor algoritmo para auxiliar em uma missão com aquelas características e otimizá-lo ao máximo, sem perda de performance.

1.3 Objetivos

O objetivo geral deste trabalho é desenvolver um algoritmo de planejamento de trajetória que permita ao UAV completar missões em ambiente 3D, desconhecido, não estruturado, e com obstáculos dinâmicos, tendo uma velocidade estimada de 1 m/s. O UAV fará o desvio de obstáculos dinâmicos que se movimentam linearmente e o mapeamento de ambientes desconhecidos. Desta forma, este trabalho visa:

- Desenvolver uma arquitetura modular;
- Realizar trajetórias suavizadas e confiáveis;
- Replanejamento seguro e rápido.

O objetivo específico consiste em avaliar e comparar diversas técnicas de planejamento de trajetória (clássicas, meta-heurísticas e aprendizado de máquina) visando conhecer suas vantagens e desvantagens. Com isso, é possível otimizar as melhores delas, ou até mesmo agrupá-las, para fazer um algoritmo otimizado para diversos ambientes, considerando as vantagens de cada algoritmo estudado.

1.4 Contribuições

Uma das contribuições deste trabalho é a avaliação de diversas técnicas de planejamento de trajetória, observando suas principais vantagens e desvantagens, e avaliar cada técnica que obteve melhor performance em determinado ambiente.

Esta avaliação foi realizada através de testes executados em plataformas de simulação 2D e 3D, e em ambiente real. Em ambiente 2D foi realizado uma análise, considerando cada ambiente, sobre como é o comportamento do algoritmo e quais são suas principais dificuldades, podendo ser em: tamanho do ambiente, quantidade de obstáculos e complexidade do ambiente. No ambiente 3D, foi realizada uma análise estatística dos algoritmos considerando o tempo para retornar a trajetória, o comprimento da trajetória, CPU, memória e *completeness*. Além de terem sido realizados voos no simulador com os algoritmos em 3D, os quais foram avaliados em relação ao tempo de voo, tempo para gerar o algoritmo, comprimento da trajetória, CPU, memória, e uso da bateria. As informações obtidas das simulações em 2D e 3D são capazes de concluir qual algoritmo possui melhor desempenho em cada ambiente e em que momento do voo é melhor usar cada um. Por fim, no ambiente real é avaliado a robustez do planejador e verificado se o um UAV real consegue completar a missão, de forma satisfatória, utilizando o planejador.

Outra contribuição é a arquitetura do planejador proposto, a qual foi desenvolvida baseada em módulos, ou seja, as funcionalidades do UAV podem ser facilmente alteradas modificando ou adicionado novos módulos. Na arquitetura proposta, está presente os módulos de controle, localização, mapeamento, planejador local estático, planejador local dinâmico, e o planejador de trajetórias. Cada um deles possui diversos algoritmos implementados, desse modo, pode ser feito a avaliação entre diversas técnicas e utilizar a melhor para cada situação.

1.5 Estrutura do trabalho

Esta dissertação está dividida em 7 capítulos. No Capítulo 2, apresenta-se a revisão bibliográfica, mostrando as principais técnicas para realizar o planejamento de trajetória, sendo clássicas, baseadas em amostragem, meta-heurísticas e de aprendizado de máquina. Além de mostrar como é tratado a suavização da trajetória e a evasão dos riscos presentes durante a mesma. Por fim, realizou-se um levantamento de trabalhos com objetos semelhantes ao dessa dissertação abrangendo o conhecimentos dos métodos adotados.

No Capítulo 3, apresenta-se as definições de planejamento de trajetória, sua importância e

como validá-lo. No Capítulo 4, explica-se as técnicas utilizadas e as otimizações feitas. No Capítulo 5, mostra-se as estratégias propostas para solucionar os principais problemas para o planejador, ou seja, ambientes desconhecidos, obstáculos dinâmicos, suavização de curvas e minimização de riscos.

No Capítulo 6, apresenta-se como resultados as simulações e análises realizadas até o momento. Por fim, no Capítulo 7 mostra-se as conclusões do trabalho, assim como os artigos desenvolvidos e em desenvolvimento.

Capítulo 2

REVISÃO BIBLIOGRÁFICA

Neste capítulo apresenta-se os trabalhos relacionados ao planejamento de trajetória visando sua melhor performance em ambientes desconhecidos, 3D, e não estruturados com obstáculos dinâmicos. Para se ter um planejamento de caminho ideal para estes ambientes, não basta apenas gerar um caminho que leve o UAV do ponto inicial até o final, é preciso que a trajetória seja suavizada, devido as restrições do mesmo. Também é necessário minimizar os riscos que podem afetar seu funcionamento ou colocar a vida de um terceiro em risco, principalmente, em ambientes dinâmicos, que a movimentação dos obstáculos pode ser imprevisível. Desta forma, para realizar o planejamento de trajetória diversos métodos são abordados com solução, tais como as técnicas clássicas, técnicas meta-heurísticas e técnicas de aprendizado de máquina, apresentando-se na Seção 2.1, 2.2 e 2.3, respectivamente. Na Seção 2.4, apresenta-se técnicas de suavização de trajetória e a minimização de riscos do UAV, na Seção 2.5. Por fim, na Seção 2.6, apresenta-se os problemas similares aos que o presente trabalho aborda, ou seja, problemas em ambientes desconhecidos, dinâmicos, não estruturados, e 3D.

2.1 Técnicas Clássicas

As técnicas clássicas utilizam o conhecimento do ambiente para prever qual será o melhor local para se movimentar visando chegar em seu destino com o menor caminho possível, sendo as primeiras técnicas de planejamento de trajetória a ser desenvolvida. Estas técnicas podem ser dividas entre técnicas clássicas exatas e aproximadas (SOUSA et al., 2017).

As técnicas clássicas exatas são baseadas em modelos matemáticos, sempre retornando o mesmo caminho, podendo sempre confirmar se a rota existe ou não, devido sua alta precisão (PATLE et al., 2019). Porém, o custo computacional é diretamente proporcional a complexidade do ambiente e à incerteza do ambiente. Exemplos dessas técnicas são os algoritmos: *A Star* (A*)

e *Artificial Potential Field* (APF).

Já as técnicas clássicas aproximadas realizam buscas no espaço livre utilizando técnicas de amostragem no espaço livre (CHOSET et al., 2005) (LAVALLE, 2006), sendo necessário menos custo computacional em relação a técnicas clássicas. As técnicas aproximadas são métodos probabilisticamente completos, sendo assim, quanto maior o número de amostras a probabilidade de se encontrar a melhor solução irá aumentar (LAVALLE, 2006). Exemplos dessa abordagem são os algoritmos *Rapid Exploring Random Tree* (RRT), *Rapid Exploring Random Tree Connect* (RRT-C) e *Probabilistic Roadmap* (PRM).

Apresentamos abaixo alguns trabalhos que abordaram as técnicas clássicas para resolver o problema de planejamento de trajetória.

Tan et al. (2016) propõem uma junção entre o algoritmo A* e o APF para gerar a trajetória em um ambiente 3D. O A* foi usado para realizar o planejamento da trajetória no cenário e o APF foi usado para suavizar as curvas. A simulação demonstrou confiabilidade na trajetória gerada e alto desempenho na suavidade das curvas.

Perez-Grau et al. (2018) apresentam uma arquitetura de UAV visando a navegação em áreas em que o GPS (*Global Position System*) não tem sinal. Para o planejamento de trajetória foi feita uma comparação entre o algoritmo RRT e o *Lazy Theta**-W. O algoritmo do *Lazy Theta**-W demonstrou um tempo de planejamento e tamanho da trajetória demasiadamente menor. Essa diferença deve-se às vantagens computacionais que os grafos apresentam em relação às árvores, ou seja, a possibilidade de reutilizar a computação anterior para replanejar mais facilmente do que os algoritmos baseados em amostragem. Desse modo, o processamento computacional do algoritmo foi otimizado.

Chen et al. (2017) desenvolvem uma melhoria para o APF capaz de evitar o mínimo local e de se movimentar em ambientes desconhecidos com objetivos dinâmicos. Os campos de repulsão e atração foram modificados para levar em consideração a direção a ser tomada entre os pontos e a distância relativa entre o UAV e o alvo, assim como no trabalho desenvolvido por Lei et al. (2017). Desse modo, a distância percorrida foi diminuída e melhorada a prevenção de obstáculos em comparação ao método tradicional. As simulações realizadas demonstraram a eficácia do método para planejamentos de trajetória on-line.

Li (2019) usam a teoria do casos no APF para modificar os coeficientes de atração e repulsão, sendo que os coeficientes são controlados dentro de um intervalo e, em seguida, o grupo de coeficientes ideal é selecionado, melhorando o método tradicional. As simulações demonstraram que o APF melhorado aumenta a precisão e a estabilidade da rota, reduz a duração e o

tempo da rota e melhora a capacidade do algoritmo de se adaptar a mapas não estruturados.

No trabalho de Yao (2021) propõem uma técnica clássica exata baseada no princípio de que a menor trajetória consiste nas tangentes, denominada de *RimJump**. No algoritmo, o limite dos obstáculos é aumentado para garantir a segurança do planejamento e então é construído um grafo de tangentes. Para encontrar o menor caminho entre as tangentes é utilizado o *Breadth First Search* (BFS). O algoritmo foi comparado com o A*, RRT, Dijkstra, e Theta*. O *RimJump** retornou uma trajetória menor que o A* e em tempo similar ao do RRT, que é o algoritmo com um dos menores tempo de resposta.

Chen et al. (2019) apresentam uma melhoria do algoritmo PRM, que é baseado na estratégia de amostragem aleatória, com o APF. A força potencial do campo é usada para selecionar os melhores pontos, visando evitar colisões nos pontos de amostragem. A função custo é definida para evitar que o algoritmo caia no mínimo local quando o ambiente for estreito, mostrando eficiência, em relação ao PRM tradicional. Para validar o algoritmo foi realizado simulações em ambientes estáticos 2D e 3D.

Wang et al. (2019b) propõem uma melhoria do algoritmo RRT* (*Rapid Random Tree Star*) para que funcione em ambiente 3D e dinâmicos, e que seja capaz de desviar de obstáculos. Foi utilizado a superposição do campo gravitacional do APF para fazer o RRT* ser capaz de fazer buscas em um ambiente 3D. Após a trajetória estática ser gerada, os obstáculos dinâmicos são identificados, e um replanejamento é feito levando em consideração a trajetória inicial e os obstáculos dinâmicos. Por último, a suavização das curvas é realizada levando em consideração o baricentro dos triângulos formados. O RRT* mostrou resultados melhores do que o RRT em relação ao tempo de simulação tamanho da trajetória.

Zhang, Xu e Yao (2018) implementam uma melhoria do algoritmo RRT-Connect com o APF. A função do campo de atração é adaptada para auxiliar a encontrar os pontos randômicos. Desse modo o RRT-Connect se mostrou melhor que o RRT, RRT-Connect e APF tanto em distância como em tempo.

Noreen et al. (2019) relizam uma comparação entre os algoritmos RRT*-AB e *Means-Ends Analysis Star* (MEA*). Sendo o RRT*-AB uma variação ao RRT* que converge rapidamente para um valor ótimo ou quase ideal. E o MEA* é uma variação do A* e *Hierarchical Path-Finding Star* (HPA*) que consome menos memória e tempo computacional. O RRT*-AB se mostrou mais eficiente ao gerar trajetórias em ambientes não estruturados. Já o MEA* gera trajetórias menores com melhores tempos de execução e requer menos memória, demonstrando-se eficiente para uso em dispositivos com memória restrita, tendo como tempo de complexidade $O(V) + O(E)$.

Xu, Deng e Shimada (2021) propõem um planejador de exploração dinâmica baseado no algoritmo PRM. Além de ser otimizado com o algoritmo *Euclidean Signed Distance Function* (ESDF) para maximizar a exploração. O algoritmo foi validado com simulações, comparando-o ao algoritmo de fronteira, *Autonomous Exploration Planner* (AEP) e *Receding Horizon Next-Best-View* (RH-NBV). O algoritmo se mostrou melhores resultados em exploração, tamanho da trajetória, e tempo, além de ter mostrado capacidade para explorar ambientes dinâmicos.

Dentre os trabalhos citados nesta seção, as técnicas clássicas exatas e aproximadas utilizam o MatLab para realizar as simulações, sem realizar testes em ambiente real ou simuladores de UAVs. Esses trabalhos tendem a implementar otimizações nas técnicas para melhorar seus resultados, tanto para minimizar a trajetória quanto para suavizar as curvas. Tan et al. (2016), Perez-Grau et al. (2018), Wang et al. (2019b) e Chen et al. (2019) propuseram uma solução para o planejamento de trajetória em ambiente 3D, também simulados no MatLab.

2.2 Técnicas Meta-Heurísticas

As técnicas meta-heurísticas possuem grande capacidade para lidar com a incerteza, pois são técnicas de resolução heurísticas que buscam uma boa solução para o problema em tempo computacional e processamento hábil. Geralmente, não há necessidade de indicar explicitamente os passos até o resultado, que seriam específicos para cada caso. Sendo assim, as técnicas meta-heurísticas devem ser entendidas como um conjunto de métodos e procedimentos genéricos e adaptáveis, para os quais outras técnicas conhecidas seriam ineficazes (DOKEROGLU et al., 2019). Exemplos de técnicas meta-heurísticas são: *Salp Swarm Algorithm* (SSA), *Bat Algorithm* (BA), *Cuco Search* (CS), *Glowworm Swarm Optimization* (GSO), *Bee Colony Algorithm* (ABC), *Biogeography Optimization Algorithm* (BOA), *Grey Wolf Optimization* (GWO), *Firefly Algorithm* (FA), *Whale Optimization Algorithm* (WOA), *Genetic Algorithm* (GA), *Ant Colony Optimization* (ACO), e *Particle Swarm Optimization* (PSO), dentre outras.

Saxena et al. (2019) propõem um SSA para o planejamento de trajetória de multi-UAVs em um ambiente 3D. O SSA atua em tempo real e com a possibilidade de replanejamento. A simplicidade, flexibilidade e mecanismo sem gradiente do SSA o impedem de cair em mínimo local, melhorando a velocidade da convergência. O algoritmo foi comparado com diversas abordagens clássicas e reativas, e mostrou melhor custo e menor tempo.

Lin et al. (2019) propõem um algoritmo de planejamento de trajetória baseado no BA para ambientes 2D e 3D. Primeiro, o APF aprimorado é adotado para acelerar o processo de convergência da atualização da posição do morcego. Segundo, a estratégia ótima de taxa de sucesso

é proposta para melhorar o peso da inércia adaptativa do BA. Por fim, a teoria do caos é proposta para evitar cair em mínimo local. Além de utilizar o método B-Spline para a suavização de curvas. Comparado com a estratégia padrão de APF e a teoria do caos, o algoritmo aprimorado aumenta significativamente a taxa de sucesso de encontrar uma trajetória adequada e diminui o tempo de convergência.

Hu et al. (2019) estudam o planejamento de trajetória para realizar entregas em áreas urbanas, evitando o tráfego, baseado na meta heurística CS. Sendo desenvolvido um modelo conceitual contendo todos os elementos-chave da tarefa de entrega. O modelo desenvolvido levou em consideração características urbanas, como a quantidade de edifícios, e da missão de entregador que lhe foi dada, como a inclinação das mercadorias. Foi realizado experimentos comparando-o com o PSO, e o CS se demostrou superior.

Pandey, Shukla e Tiwari (2018) apresentam uma nova abordagem para planejamento de trajetória, mantendo uma distância segura dos obstáculos na trajetória. Nesse trabalho, foi feita uma melhoria no GSO adicionando duas etapas do algoritmo genético: operadores genéticos e cruzamento (*cross-over*), visando melhorar suas características de exploração e intensificação, além de ajudar o algoritmo a obter melhores resultados e convergência mais rápida. O GSO foi validado comparando-o com o PSO, BBO, e IBA, tanto em ambiente 2D como 3D.

Song et al. (2019) apresentam um planejamento de trajetória 3D baseado em BOA para UAVs de alta velocidade. O APF é usado para evitar saturação, por sua peculiaridade de suavidade e limitação. BOA é utilizado para os parâmetros do APF visando obter o melhor desempenho em estabilidade e velocidade. A partir das simulações realizadas, o algoritmo foi validado provando boa estabilidade e alta velocidade de operação.

Dewangan, Shukla e Godfrey (2019) utilizam a técnica GWO para solucionar o planejamento de trajetórias em ambientes 3D, evitando colisão entre obstáculos e outros UAVs. O desempenho do algoritmo foi comparado com diversos algoritmos que utilizam técnicas clássicas, como o Dijkstra, A* e D*, e meta-heurísticas, como BA, BOA, PSO, GSO, WOA e *Sine Cosine Algorithm* (SCA). O algoritmo fornece convergência mais rápida e evita mínimos locais, o que resulta em baixo custo de cálculo de trajetória e melhores resultados quando comparado com outros algoritmos meta-heurísticos e clássicos. A solução proposta utiliza menos número de parâmetros, melhorando assim a velocidade de convergência.

Kumar et al. (2020) utilizam uma das meta-heurísticas mais básicas, a Busca Tabu. O algoritmo informa ao robô móvel para não visitar as posições visitadas anteriormente e auxilia o robô a obter o melhor ângulo para evitar os obstáculos. A modificação proposta pelo autor permite que robô se locomova entre a origem e o destino em qualquer ambiente. Foram realizados

diversos testes, tanto em ambiente simulado como em ambiente real.

Patle et al. (2018) apresentam a implementação do FA para o planejamento de trajetória de robôs móveis em ambientes desconhecidos com obstáculos estáticos e dinâmicos. O algoritmo proposto explora com eficiência o ambiente e melhora a pesquisa global em menor número de iterações e, portanto, pode ser facilmente implementado para evitar obstáculos em tempo real, especialmente em ambientes dinâmicos. Além disso, minimiza os cálculos computacionais e evita a movimentação aleatória dos vaga-lumes. O desempenho do FA proposto é melhor em termos de otimização de trajetória quando comparado a outras abordagens de planejamento de trajetória.

Tuba et al. (2019) utilizam o *Harmony Search Algorithm* (HSA) aprimorado para o planejamento de trajetória em zonas de perigos com obstáculos estáticos. O HSA foi melhorado utilizando o parâmetro de aceitação de memória para controlar o equilíbrio entre exploração e intensificação, além de que para encontrar soluções na vizinhança é ajustado o tom da ramificação da largura de banda. Por fim, há uma randomização para aumentar a diversidade das soluções da população. Os resultados da simulação mostram que o algoritmo proposto produz melhores resultados em relação ao comprimento da trajetória gerado.

Pan, Liu e Liu (2018) propõem uma melhoria do WOA para o planejamento de trajetória de *Unmanned Combat Aerial Vehicle* (UCAV). Foi proposto um novo critério de julgamento para selecionar o processo de cercar presas ou procurar presas na WOA, ou seja, um parâmetro de autoajuste baseado na qualidade da adequação do agente em vez de um valor aleatório usado no WOA original. O agente com maior aptidão, ou seja, agente superior, atualiza sua posição em relação ao melhor agente encontrado até o momento. Pelo contrário, o agente com menor aptidão, isto é, agente inferior, atualiza sua posição em relação a um agente de referência que é selecionado aleatoriamente na população. O algoritmo aprimorado foi comparado com o original e obteve um melhor custo e menor trajetória.

Xin et al. (2019) utilizam GA para o planejamento de trajetória para *Unmanned Surface Vehicle* (USV). Geralmente, o GA possui população prematura e a lenta velocidade de convergência, por isso o trabalho propôs o aumento do cromossomo usando a inversão de vários domínios. Assim como uma segunda avaliação de aptidão foi realizada para eliminar filhos indesejáveis e reservar os indivíduos mais vantajosos. O algoritmo modificado foi implementado no sistema de navegação de um USV, em que foi realizado um estudo comparativo revelando melhoras no comprimento da trajetória e o custo do tempo, além de uma velocidade de convergência mais rápida.

Song et al. (2020) propõem uma estratégia de planejamento de trajetória com base em

Fuzzy Logic (FL) e na otimização do algoritmo ACO. Foi feita uma otimização baseada em ranks no ACO e introduzido o sistema de elitismo, as quais não se adaptam bem em ambientes dinâmicos. Por isso, o ACO foi integrado ao FL, FLACO, para encontrar a trajetória ideal. O FLACO foi comparado ao ACO e a um ACO melhorado, e gerou a trajetória mais curta.

Chen, Luo e Zhai (2019) propõem uma nova estratégia de prevenção de obstáculos para garantir a segurança de voo dos UAVs. O ambiente é modelado com o auxílio do sensor laser TFmini. Em seguida, o método Dijkstra foi empregado para pesquisar as trajetórias viáveis para o UAV. Por fim, a trajetória global ideal foi obtida com base no algoritmo PSO aprimorado, o qual vetoriza as funções objetivo e avaliam todos os pontos em um padrão de pesquisa de uma só vez.

Zhou et al. (2021) descrevem um algoritmo de planejamento de trajetória baseado em BA para ambientes estáticos em 3D. O BA é otimizado junto com o ABC, nomeado de *Integrated Bat Algorithm* (IBA). O ABC é utilizado para melhorar a baixa capacidade de busca local do BA. As simulações são feitas em MATLAB e mostram uma melhora no tempo do algorimto de aproximadamente 50%, e o tamanho da trajetória é cerca de 14% menor. Além disso, comparando com outros algoritmos de planejamento de trajetória de enxame tradicional, como o PSO, o IBA pode planejar um caminho de voo mais rápido, mais curto, mais seguro e sem colisão para UAVs.

Dong (2021) propõem uma estratégia para o planejador de trajetória completar a missão, mesmo em ambientes desconhecidos. São considerados três principais aspectos: planejamento de trajetória, otimização de trajetória, e controle de estado. Primeiramente, através de um algoritmo heurístico baseado na otimização dinâmica, uma trajetória ideal é encontrada no ambiente, obedecendo as restrições dinâmicas. Além disso, de acordo com a trajetória encontrada, uma função polinomial é usada para expressá-la, e a trajetória é rapidamente gerada pelo método da menor derivada de deslocamento para otimização, que fornece entrada para o módulo de controle do UAV. Por fim, a máquina de estado é usada para gerenciar o estado atual do UAV.

Todos os trabalhos apresentados utilizaram o MatLab para fazer as simulações iniciais. Os trabalhos de Dewangan, Shukla e Godfrey (2019), Song et al. (2019), Pandey, Shukla e Tiwari (2018), Hu et al. (2019), Lin et al. (2019) e Saxena et al. (2019) aprimoraram as técnicas para solucionar o planejamento de trajetória em ambientes 3D com rotas suavizadas. No entanto, essas rotas tendem a estar muito próximas dos obstáculos e não houve nenhum trabalho que operasse com ambientes não estruturados, apenas poucos que utilizavam ambientes desconhecidos.

2.3 Técnicas de Aprendizado de Máquina

Também é possível utilizar técnicas de aprendizado de máquina, baseadas em aprendizado supervisionado, aprendizado não supervisionado e/ou aprendizado por reforço nos problemas de planejamento de trajetória. As técnicas de aprendizado supervisionado precisam ser treinadas e então seguem o padrão dos seus dados de treinamento. As técnicas de aprendizado não supervisionado aprende de acordo com os padrões dos dados, geralmente, os clusterizando, não sendo necessário o treinamento prévio. O aprendizado por reforço é treinado para obter uma sequência de decisões. O agente aprende a atingir uma meta para completar tarefas incertas e complexas. O seu treinamento é um sistema de tentativa e erro para encontrar a solução do problema baseado recompensas e penalidades pelas ações que executa (LISON, 2015).

Bency, Qureshi e Yip (2019) afirmam que os métodos de busca de caminhos tendem a ser exponencialmente custosos computacionalmente de acordo com que a complexidade e dimensão do ambiente aumentam. Por isso, os autores propõem um planejador de trajetória rápido para ambientes estáticos utilizando a rede neural recorrente *Long Short Term Memory* (LSTM), usada para determinar trajetórias de ponta a ponta de maneira iterativa, em tempo real, que gera implicitamente planos de trajetória ideais com perda mínima de desempenho de forma compacta. O algoritmo demonstrou uma execução em tempo fixo, independentemente da complexidade do espaço de configuração, superando os algoritmos populares de busca de caminhos em ambientes não estruturados e dimensões maiores. A técnica foi testada em simulação usando as bibliotecas Keras e TensosFlow no Python.

Toma et al. (2021b) propõem um algoritmo híbrido entre o LSTM, usado como kernel local, e o A*, usado como kernel global, denominado de *Waypoint Planning Networks* (WPN). O WPN produz uma solução computacionalmente mais eficiente e robusta comparando com o A*, *Motion Planning Networks* (MPNet) e *Value Iteration Networks* (VIN). Foram realizados simulações em ambientes 2D, no PathBench (TOMA et al., 2021a), para validar o algoritmo, os quais mostraram benefícios da técnica tanto em eficiência quanto em generalização. Além disso, o espaço de busca do WPN é consideravelmente menor do que do A*, e pode operar em mapas parciais.

Saha e Dasgupta (2017) desenvolvem um planejador de trajetória para um ambiente, inicialmente, desconhecido. O robô não possui um mapa a priori de seu ambiente, mas tem acesso a informações prévias acumuladas por ele mesmo pela navegação em ambientes semelhantes, mas não idênticos. Foi implementado um algoritmo baseado em aprendizado de máquina supervisionada chamado Processo de Decisão Semi-Markoviana com Inconsciência e Trans-

ferência (*Semi-Markov Decision Process with Unawareness and Transfer* - SMDPU-T), em que um robô registra uma sequência de suas ações em torno de obstáculos como sequências de ação que são então reutilizadas para aprender manobras adequadas e sem colisões em torno de obstáculos mais complexos no futuro. Os resultados mostraram que o SMDPU-T leva 24% do tempo de planejamento e 39% do tempo total para resolver as mesmas tarefas de navegação em comparação com um planejador de caminho recente baseado em amostragem, vistos na simulação pelo robô Corobot do simulador WeBots.

Yue e Zhang (2018) propõem um planejador de trajetórias para UAVs baseado no algoritmo K-means, uma técnica de aprendizado de máquina não supervisionado, e no algoritmo de *Simulated Annealing* (SA), uma técnica meta-heurística, para resolver os problemas de multi-UAVs com multi-missões sob restrições. Primeiramente, são definidas zonas em que não se pode voar. Então, a técnica de decomposição decompõe a área válida em diversos sub-pontos. Por fim, o algoritmo K-means é usado para agrupar os pontos e o SA realiza o planejamento entre as sub-rotas. O algoritmo foi testado no ambiente de simulação MatLab.

Lee, Park e Joe (2020) propõem o uso de aprendizado por reforço para fazer o melhor planejamento de trajetória, visando maximizar a cobertura em um certo ambiente. A potência do sinal em cada posição é usada como recompensas e penalidades no algoritmo. Desse modo, o planejador não irá precisar das informações de toda a rede, já que a técnica é baseada no processo Markoviano. Com isso, foi obtido uma trajetória que consegue maiores taxas de transferência em relação aos caminhos padrões.

Também é possível utilizar a técnica de aprendizado por reforço para otimizar outros algoritmos, como mostrado em Qu et al. (2020b). Os autores propõem um algoritmo de aprendizagem por reforço baseado em GWO para realizar o planejamento de trajetória para UAVs em um ambiente 3D. O planejamento de trajetória foi considerado como um problema de otimização para encontrar a trajetória ideal minimizando a função de custo. Então, foi desenvolvido as operações de exploração, intensificação, ajuste geométrico e ajuste ideal. O desempenho em cada operação foi controlado pelo aprendizado por reforço. Por fim, curva cúbica do B-spline foi inserida para suavizar a rota gerada. O desempenho foi testado com outras variações do mesmo algoritmo, como o *Inspired Grey Wolf Optimizer* (LONG et al., 2018b), *Exploration Enhanced Grey Wolf Optimizer* (LONG et al., 2018a), *Numerical Grey Wolf Optimizer* (KUMAR; KUMAR, 2017) e o próprio GWO, que mostrou-se gerar menores rotas para serem usadas em UAVs. Os algoritmos citados foram testados no ambiente de simulação MatLab.

Bayerlein et al. (2021) propõem uma abordagem de planejamento de trajetória baseada em aprendizado de reforço profundo por multi-agente para colher dados de dispositivos IoT

(*Internet of Things*). A técnica utilizada é a *Double Deep Q Network* (DDQN) processando convulsivamente as informações do mapa local e global focando na posição dos agentes. As simulações mostram que a abordagem permite que os agentes cooperem de forma eficaz dividindo as tarefas de coleta de dados entre si, adaptam-se a grandes ambientes complexos com eficiência de tempo.

As técnicas de aprendizado de máquina tem ganhado bastante reconhecimento nos últimos anos, inclusive na área do planejamento de trajetória. Há diversos artigos explicando a teoria de como funciona e resultados matemáticos, mas sem nenhuma simulação ou voo em ambiente real. Alguns trabalhos mais recentes mostram essas técnicas sendo utilizadas em ambientes não estruturados, desconhecidos e com obstáculos dinâmicos (WANG et al., 2017; YAN; XIANG; WANG, 2019; WU et al., 2019).

Também há trabalhos que utilizam técnicas de aprendizado de máquina para fazer o desvio de obstáculo, com os algoritmos como: ORCA (*Optimal Reciprocal Collision Avoidance*) (ALEJO et al., 2014), CADRL (*Collision Avoidance with Deep Reinforcement Learning*) (LI et al., 2019), ALAN (*Adaptive Learning for Multi-Agent Navigation*) (GODOY et al., 2015).

2.4 Suavização de Trajetória

A suavização de curvas é uma das principais características que deve ser implementada no planejamento de trajetória. Pois dessa maneira a trajetória se torna mais fácil para ser seguida pelo UAV, não tendo a necessidade que a velocidade seja reduzida durante o trajeto.

Existem diversas técnicas para realizar a suavização na trajetória, entre elas temos o *Moving Average Method* que foi utilizado por Li (2019). Nesta técnica a curva é dada pela regressão linear dos mínimos quadrados ponderados do caminho, que tende a manter as maiores distâncias e ir suavizando até voltar a trajetória original.

Krell et al. (2019) utilizam o algoritmo *Pure Pursuit Path* para suavizar as curvas. O algoritmo foi criado como um método de encontrar geometricamente a curvatura que direcionará o robô para um ponto de caminho escolhido P_d. O algoritmo também é capaz de calcular a velocidade angular necessária para mover o robô de sua posição atual até ser possível localizar um ponto de observação à frente do robô.

Dewangan, Shukla e Godfrey (2019) e Saxena et al. (2019) utilizaram a estratégia de *Fifth Order Polynominal Based Trajectory Fitting* para suavizar as curvas. Esse algoritmo é baseado no movimento e velocidade do UAV que pode ser representado por um polinômio de quinto

grau, envolvendo o tempo. Ao gerar a trajetória, os subpontos são obtidos na fase inicial. Os quais podem ajudar o UAV a evitar qualquer obstáculo ou curva acentuada. Para produzir uma rota suavizada, a velocidade e o movimento do UAV devem ser utilizados nessa estratégia.

Há métodos mais conhecidos para realizar a suavização das curvas, como a Curva de Bézier, utilizada por Xin et al. (2019). Sendo uma curva polinomial expressa como a interpolação linear entre alguns pontos representativos, chamados de pontos de controle. Tharwat et al. (2019) propõem que a técnica de *Chaotic Particle Swarm Optimization* fosse utilizada para otimizar o ponto inicial e final da curva de Bézier em uma trajetória. A otimização dessa curva conseguiu gerar trajetórias menores.

Outra técnica bastante conhecida é a *B-Spline* usada por Pan, Liu e Liu (2018), Pandey, Shukla e Tiwari (2018), Qu et al. (2020b) e Qu et al. (2020a). *B-spline*, ou *spline* de base, é uma função *spline*, uma curva definida matematicamente por dois ou mais pontos de controle, que possui um determinado grau, suavidade, e partição do domínio. É possível computar os pontos da curva utilizando somente interpolações lineares sucessivamente a partir dos pontos de controle, utilizando o algoritmo de Boor. A curva de *B-Spline* é uma generalização da curva de Bézier.

Também há o caminho de Dubins, o qual normalmente se refere à curva mais curta que conecta dois pontos no plano euclidiano 2D. Possui uma restrição na curvatura do caminho com tangentes iniciais e terminais prescritas para o mesmo. O veículo que a percorre só pode avançar. Tenezaca et al. (2020) e Primatesta, Guglieri e Rizzo (2019) implementaram as curvas nas trajetórias após já terem sido formadas pelo RRT* e A*.

Outra maneira de realizar a suavização da trajetória é a partir de técnicas clássicas, como as mostradas nesse capítulo, na Seção 2.1. Song et al. (2019) e Tan et al. (2016) utilizaram o algoritmo de APF. O nó central é definido pelo BOA e A*, respectivamente, e o caminho suavizado é construído de acordo com a força gravitacional do ambiente, funcionando para fazer curvas em ambientes 3D.

Li et al. (2020) propõem que o algoritmo *Cubic Spline* seja usado como planejamento de trajetória. Como o algoritmo é baseado em *splines* a trajetória é bem suavizada. No entanto, o autor propõe o uso do PSO para otimizar a trajetória e tornar possível utilizar a técnica proposta em ambientes incertos e dinâmicos. Os autores consideram um problema de planejamento de caminho de robô originado de um cenário de inspeção de fábrica de robôs. No problema, o robô está em um ambiente dinâmico incerto, ou seja, um objeto alvo em movimento e vários obstáculos estáticos e dinâmicos. Uma estratégia de posicionamento inercial é proposta para permitir que o robô preveja a posição do alvo com antecedência. A partir dessa posição prevista,

o caminho do robô é gerado pela interpolação cúbica de splines e, em seguida, um algoritmo aprimorado de otimização de enxame de partículas com um fator de feedback positivo aleatório na atualização de velocidade otimiza o caminho. Os resultados experimentais mostram que o método proposto pode evitar com sucesso os obstáculos e alcançar o objeto alvo. Além disso, a estratégia de posicionamento inercial e a melhoria da otimização do enxame de partículas podem reduzir efetivamente o caminho do robô.

Mas também é possível definir o próprio algoritmo de suavização de trajetória, como feito por Deray et al. (2019), em que é proposto uma curva suave por partes para o planejamento e controle de um robô de base móvel. A trajetória tem que se situar em um spline com n-ésimas derivadas em todos os pontos. Formulado como um problema de otimização não linear de múltiplos objetivos, permite impor restrições suaves, como evitar colisão, velocidade, aceleração, entre outros. O processo é em tempo real e permite que o robô navegue em cenários não estruturados e dinâmicos. Foi feito uma comparação com o estado da arte e obteve-se que a trajetória proposta tem menor aceleração média e são mais eficientes em termos de curvatura e energia pseudo cinética.

Zhang, Zhang e Zou (2021) utilizam a teoria do tempo-ideal com base no modelo dinâmico do robô para planejar a trajetória, construir o modelo de otimização de trajetória sob as restrições do caminho geométrico e do torque. O algoritmo utiliza o *Input Shaping Algorithm* invés das restrições do *Jerk* no modelo de otimização para suavizar a trajetória. Os resultados mostraram que utilizando esse algoritmo é possível gerar uma trajetória suavizada em tempo ótimo.

Letizia, Salamat e Tonello (2021) propõem um algoritmo de planejamento de trajetória suavizado e recursivo (RST - *Recursive and Smoothed Trajectory*) para UAVs. O RST constrói a trajetória recursiva como um caminho polinômico liso satisfazendo qualquer limitação dinâmica. As incertezas das restrições também são modeladas e incluídas no cálculo dos coeficientes da trajetória polinomial. Além disso, devido à sua formulação recursiva, não é necessário uma etapa de otimização, a trajetória suavizada é gerada automaticamente. A eficácia do algoritmo é demonstrada numericamente através de dois cenários comparados com *Minimum-Snap Piecewise Polynomial Trajectory Algorithm*, além de ser mostrado a vantagem de um caminho com curvas suavizadas.

Também é possível criar a curva de uma trajetória a partir da geometria analítica como mostrado por Noreen et al. (2019) em seu processo para finalizar a trajetória do algoritmo A*, e por Wang et al. (2019b), o qual detecta uma área que forma um triângulo, desse modo é definido o ponto central do mesmo para realizar a suavização baseado nele.

2.5 Minimização de Riscos

Para se obter um planejamento de trajetória ideal, na maioria das missões, é preciso que o UAV se locomova do ponto inicial até o ponto final sem colidir, em segurança. No entanto, em alguns casos isso não significa apenas evitar obstáculos, estáticos e/ou dinâmicos, pode ser necessário garantir uma margem de segurança para evitar colisão em algum problema físico do UAV. Como por exemplo em ambientes urbanos, em que essa distância irá proporcionar uma maior segurança às pessoas (HU et al., 2019).

Primatesta, Guglieri e Rizzo (2019) apresentam um planejamento de trajetória para UAV com o objetivo de minimizar o risco da população em ambientes urbanos. Neste trabalho é proposto o algoritmo riskA*, baseado no A*, para fazer o planejamento de trajetória e algoritmo *Borderland*, que é uma extensão do trabalho de Lumelsky e Stepanov (1987), para fazer o replanejamento. Primeiramente, é calculado um mapa off-line com fatores de risco estáticos, com o algoritmo riskA* que visa minimizar riscos. E em seguida, é feito o planejamento de caminho on-line com um mapa de risco dinâmico, a partir do algoritmo *Borderland* proposto, ajustar rapidamente apenas a parte do caminho envolvida pelo início de alterações dinâmicas relevantes no fator de risco. Após o planejamento do caminho, é utilizado as curvas de Dubins para realizar a suavização do caminho.

Também podem haver informações externas que sejam interessantes ao UAV para que não entre em uma área de risco. Sankararaman e Goebel (2018) consideram em seu *framework* o que será necessário identificar para denominar se uma área é segura ou não, tais como: informações sobre o vento, dos obstáculos do ambiente, da movimentação do próprio UAV, de falha, e de energia.

Sankararaman e Goebel (2018) propõem um *framework* baseado na identificação de riscos para prever áreas seguras e perigosas para o UAV se locomover. O clima, obstáculos dinâmicos (com trajetórias e velocidade não definidas), falha no sinal de GPS e degradação dos sensores como incertezas são definidos como incertezas. E o gerenciamento de bateria, falhas no sistema, eficiência do controle, estabilidade e dinâmica como restrições do desempenho do veículo. As áreas de risco devem ser levadas em consideração ao construir a trajetória ideal para evitar requisitar do UAV uma operação impossível ou inviável para o momento.

Sharma e Tokekar (2021) propõem um método de planejamento de trajetória baseado nos riscos de imagens aéreas contendo oclusões. Foi utilizado técnicas de aprendizado profundo de pintura e segmentação semântica para identificar áreas obstruídas, e substituí-las pelo o que pode estar por trás, para então planejar uma trajetória, com o A*, considerando a incerteza das

previsões.

Mokhtari e Wagner (2021a) desenvolvem um tomador de decisões baseado em risco que utiliza um modelo de aprendizado por reforço pré-treinado. O tomador de decisões utiliza um algoritmo de planejamento de trajetória baseado em risco de alto nível (MOKHTARI; WAGNER, 2020) em conjunto com o controle de baixo nível baseado em aprendizado por reforço (MOKHTARI; WAGNER, 2021b). Os métodos são avaliados no simulador CARLA (DOSOVITSKIY et al., 2017) e demonstraram uma melhora na segurança dos veículos permitindo que os mesmos desviem de situações de riscos.

Por fim, o gerenciamento de bateria também deve ser levado em consideração impossibilitando que eventuais problemas ocorram, tais como: o UAV pouse em locais não planejados, a missão seja interrompida antes do término, ou que a comunicação seja interrompida. Além disso, através do gerenciamento da bateria é possível fazer com que o UAV retorne a base, se recarregue e volte para realizar suas missões. Uma arquitetura com esse objetivo foi implementada por Malyuta et al. (2020), que proporcionou que o UAV tenha uma longa duração de funcionamento autonomamente. O UAV realizou experimentos em ambiente *indoor* durante 11 horas, e durante 4 horas em ambiente *outdoor*. Para ficar tanto tempo nestas missões, o UAV foi capaz de identificar e pousar em bases para recarregar sua bateria.

2.6 Problemas Similares

Diversos pesquisadores tentam resolver problemas similares ao dessa dissertação, ou seja, determinar o melhor caminho para UAV em ambientes não estruturados, 3D, desconhecidos e/ou com obstáculos dinâmicos, esses trabalhos são apresentados através de simuladores diversificados.

Wang et al. (2019a) utilizaram o *framework* Python RVO2 3D para realizar suas simulações 3D. Os autores propõem uma DRL em Q-*learning*, sendo o voo ideal determinado por o algoritmo guloso, o *e-greedy*, para ambientes desconhecidos e dinâmicos. O algoritmo se mostrou eficiente para calcular a trajetória ótima, tendo o ponto de convergência da rede variando entre 11 e 12 segundos e o número de obstáculos entre 5 e 10. No entanto, os resultados só foram comparados com a mesma rede antes do treinamento.

Já Krell et al. (2019) utilizaram o MatLab para mostrar os resultados matemáticos e o simulador Gazebo 3D para fazer a simulação do UAV em um ambiente 3D. Os autores propõem um planejamento de trajetórias com auxílio do PSO, uma técnica de otimização meta-heurística, para ambientes desconhecidos, onde foi utilizado o LiDAR (*Light Detection And Ranging*) para

mapear os ambientes, o PSO para realizar o *Route Planning* no cenário e o algoritmo *Pure Pursuit Path Tracking* para definir a trajetória suavizada entre cada ponto definido. O algoritmo e a navegação robusta evitaram obstáculos em diferentes ambientes desconhecidos.

Qu et al. (2020a) também apresentam uma otimização para o algoritmo GWO com *Modified Symbiotic Organisms Search* (MSOS), em ambiente 3D no MatLab, tendo o objetivo de realizar o planejamento de trajetória em áreas complexas e perigosas. As habilidade de exploração e intensificação são combinadas de ambos algoritmos. O GWO foi simplificado para acelerar a velocidade de convergência e reter a capacidade de exploração da população. A fase de comensalismo do *Symbiotic Organisms Search* (SOS) foi modificada para melhorar a capacidade de intensificação. Por fim, a curva B-*spline* foi utilizada para suavizar a rota gerada e torná-la adequada para o UAV.

Li et al. (2020) consideram o problema de planejamento de trajetória para inspeção em fábricas, considerando um ambiente desconhecido, com alvo em movimento e obstáculos estáticos e dinâmicos. Uma estratégia de posicionamento inercial foi proposta para permitir que o robô preveja a posição do alvo com antecedência. A partir dessa posição prevista, a trajetória do robô foi gerado pela Interpolação Cúbica de Splines e, em seguida, um PSO otimizado foi usado, o qual considera um *feedback* positivo e aleatório na atualização de velocidade. Comparado com os métodos clássicos, o algoritmo se mostrou mais eficiente em um ambiente dinâmico e incerto. Comparado com outros algoritmos meta heurísticos mostrou melhor desempenho ao encontrar a trajetória mais curta, de forma estável.

Biswas et al. (2019) levam em consideração ambientes desconhecidos e dinâmicos apresentando um planejamento de trajetória baseado no PSO para sistemas autônomos em ambientes desconhecidos. Também é apresentado um novo método para identificar terrenos, atribuindo diferentes pesos aos tipos de terreno, o quais medem as características de capacidade de travessia nesse terreno. Os autores combinaram a estratégia de reconhecimento de terreno e de prevenção de obstáculos com o problema de planejamento de trajetos para planejar efetivamente uma trajetória ideal. Além disso, seguindo essa metodologia, o agente autônomo é capaz de atingir a localização da meta sem colisões. Os resultados da simulação em ambientes desconhecidos, e com obstáculos dinâmicos, validam a viabilidade da metodologia proposta e a eficiência o problema de planejamento em terreno desconhecido.

Zhang et al. (2020) apresentam um algoritmo de planejamento de caminho em tempo real para veículos aéreos não tripulados furtivos em um ambiente dinâmico complexo 3D. Primeiramente, foi utilizado o A* aprimorado com o modelo preditivo de controle enquanto a informação do ângulo é adicionada ao planejador. Combinando a análise cinemática do UAV e

a análise de desempenho de detecção do radar, as rotas originais e as rotas de replanejamento podem satisfazer os requisitos de voos reais. Os resultados foram simulados utilizando o MatLab 2017Ra.

Lopez et al. (2019) apresentam um planejador de trajetórias 3D em tempo real para UAVs visando obter um caminho viável, ideal e livre de colisões em ambientes dinâmicos e complexos. O mapa do ambiente é gerado baseado na probabilidade de ir para cada espaço. Sempre que é feito o planejamento esse mapa é consultado e então explorado com o A*, utilizando a função de custo do APF. Foram realizados os testes no simulador V-REP.

Li (2021) propõem uma estratégia para utilizar o RRT em tempo real, em ambiente dinâmico desconhecido. A estratégia é baseada em inserir novos nós no ambiente de acordo com o conceito de desvio de obstáculos e nas restrições não holonômicas. Desse modo, o custo computacional para replanejar as rotas e desviar dos obstáculos é reduzida. Além de que a diferença de tamanho entre a nova rota e a original é pequena. As simulações foram realizadas em MATLAB.

Xie et al. (2021) propõem uma abordagem baseado em aprendizado por reforço profundo (DRL - *Deep Reinforcement Learning*) utilizando apenas informações locais, dos sensores. O planejamento de trajetória é formulado como um processo de decisão de Markov parcialmente observável. A rede neural recorrente com memória temporal é construída para abordar o problema de observação parcial, extraindo informações cruciais das sequências estado-ação. Foi desenvolvido uma estratégia de seleção de ação que combina o valor atual da recompensa e o valor de estado-ação para reduzir a exploração. Além disso, é construído dois *pools* de memória e é proposto um mecanismo de *replay* adaptativo baseado na frequência de falha. Os resultados do experimento de simulação mostram que o método tem melhorias significativas sobre a *Deep Q-Network* em termos de estabilidade e eficiência de aprendizado. As simulações foram realizadas no V-REP e a abordagem se mostrou capaz de planejar trajetórias em ambientes complexos e desconhecidos em 3D.

Nesta seção foram apresentados os trabalhos que se assemelham ao proposto, levando em consideração os problemas que os autores propõem solucionar, como ambientes desconhecidos, não estruturado, 3D, com obstáculo dinâmico, e com curvas suavizadas. A síntese dos trabalhos pode ser vista na Tabela 2.1.

Nos trabalhos apresentados na Tabela 2.1, os autores utilizaram para realização de testes variados simuladores, e nenhum dos autores testaram o planejador em ambiente real. Desta forma, como visto na Tabela 2.1, nenhum dos trabalhos citados apresentam todas as características do planejador proposto neste trabalho. A maioria dos trabalhos citados apresentam curvas suavizadas, respeitando as restrições dos UAVs. Em nenhum dos trabalhos, a minimização de riscos

Tabela 2.1: Trabalhos similares.

	Wang et al. (2019a)	Krell et al. (2019)	Qu et al. (2020a)	Li et al. (2020)	Biswas et al. (2019)	Zhang et al. (2020)	Lopez et al. (2019)	Li (2021)	Xie et al. (2021)	Esta Dissertação
Ambiente 3D	X	X	X			X	X		X	X
Ambiente desconhecido	X	X		X	X			X	X	X
Ambiente não estruturado						X		X	X	X
Obstáculos dinâmicos	X			X	X	X	X X	X		X
Curvas suavizadas	X	X	X	X	X		X		X	X

foi considerada. Apesar de Qu et al. (2020a) ser o que apresenta menos características similares a esse trabalho, é o que melhor descreve os resultados, realizando análises estatísticas, sendo mais fácil realizar comparações.

2.7 Considerações Finais

Neste capítulo apresentou-se uma revisão bibliográfica sobre planejamento de trajetória com foco em UAVs. Podemos observar neste levantamento que no planejamento de trajetória pode-se utilizar diversas técnicas, como técnicas clássicas exatas, técnicas clássicas aproximadas, técnicas meta-heurísticas e técnicas de aprendizado de máquina. Bem como, entender referente à cada técnica suas vantagens e desvantagens em sua aplicação em dado cenário. Desse modo, foi possível analisar os cenários que obtiveram os melhores resultados, e à partir destas informações aprimorar cada técnica para suprir suas deficiências.

Em seguida, foi realizado um estudo sobre as técnicas de suavização de trajetória, pois os UAVs possuem diversas restrições, não podendo seguir qualquer trajetória. Assim, pode-se entender o funcionamento destas técnicas e a melhor opção para ser usada em cada ambiente, com baixo tempo de processamento e que minimize os riscos de colisão ou falhas devido suas restrições.

Os UAVs autônomos precisam se manter seguros em missões, evitando colidir com obstáculos, estáticos e dinâmicos, principalmente quando atuar em ambientes desconhecidos. Por isso foi feito um estudo das técnicas que visam minimizar os riscos de colisão. Por fim, foi mostrado problemas similares ao que essa dissertação propõe, ou seja, planejar a trajetória em ambientes desconhecidos, não estruturado, 3D, e dinâmicos.

Nos trabalhos pesquisados notou-se poucos pesquisas realizadas considerando ambientes 3D. Para os que consideram obstáculos, tendem-se a utilizar poucos e em sua maioria estáticos. As simulações realizadas, geralmente, foram em um ambiente para desenvolvimento e análise de dados, como o MatLab e Python, poucos trabalhos mostraram um ambiente de simulação para UAV ou realizou testes em ambiente real, assim como para ambientes desconhecidos.

Como fechamento da análise realizada à partir desta revisão, foram escolhidas técnicas de cada categoria, que demonstraram os melhores resultados, com intuito de avaliá-las através de comparações visando definir as melhores técnica para cada ambiente. Entre as técnicas clássicas foram escolhidas o A* e o APF. Entre as técnicas clássicas aproximadas foram escolhidas o PRM, RRT e o RRT-C. Entre as técnicas meta-heurísticas foram escolhidas o PSO, GWO e GWO. Entre as técnicas de aprendizado de máquina foi escolhida o RL.

Importante ressaltar que, cada uma delas possui suas especificidades que torna cada técnica única, podendo ser melhor usada em ambientes diferentes. Como por exemplo, o A* tende a retornar o menor caminho mas sua complexidade aumenta exponencialmente com o tamanho do ambiente, então, dependendo do ambiente em que será aplicado o RRT pode ser uma escolha melhor, já que sua complexidade não aumenta muito em relação ao ambiente.

No próximo capítulo será mostrado as técnicas escolhidas e como foram implementadas para solucionar os problemas que este trabalho visa solucionar. Além de descrever as otimizações realizadas para alcançar esses resultados.

Capítulo 3

PLANEJAMENTO DE TRAJETÓRIA

Neste capítulo apresenta-se uma visão geral da navegação para robótica móvel. Na Seção 3.1 e na Seção 3.2 explica-se a importância do Planejamento de Trajetória. Na literatura, existe algumas confusões entre as definições dos conceitos de Planejamento de Rota, Planejamento do Movimento, e Planejamento de Trajetória, por isso, na Seção 3.3 é explicado a diferença entre cada um deles. O Planejamento de Trajetória pode ser dividido entre planejador local estático e dinâmico, assim explica-se a diferença entre ambos na Seção 3.4. E por fim, mostra-se na Seção 3.5 como validar um planejador de trajetória.

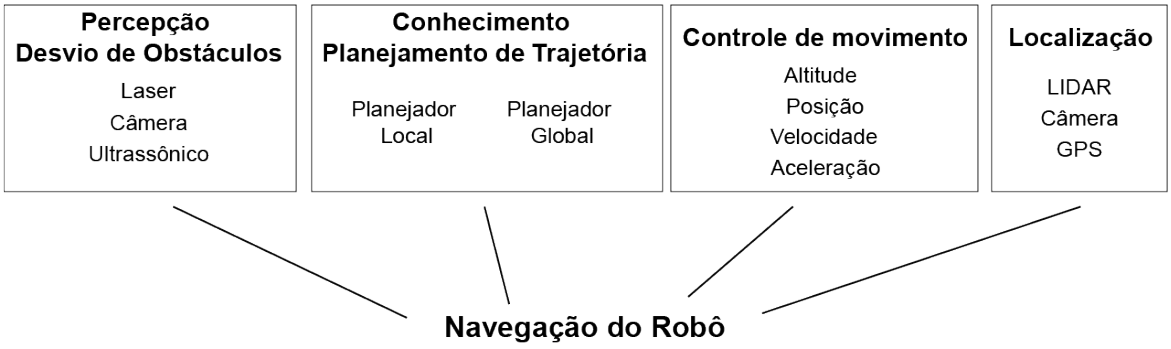
3.1 Navegação na Robótica Móvel

A navegação é uma das competências mais desafiadoras exigidas um robô móvel autônomo (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). Quando vamos utilizar um robô móvel, independente da tarefa que receba, é certo que irá precisar navegar, no mínimo, entre dois pontos. Por isso, a navegação também é uma competência essencial na robótica móvel. Inicialmente, todo robô precisa saber das respostas para as seguintes perguntas:

- Onde estou?
- Para onde vou?
- Como faço para chegar lá?

Para responder a essas perguntas o robô segue a estrutura mostrada na Figura 3.1. Os sensores enviam informações para os módulos de localização e desvio de obstáculo (Onde estou?). Desse modo, o planejador local estático pode determinar qual será a melhor trajetória a tomar

FIGURA 3.1 – Estrutura Básica de Navegação para Robôs Móveis.



Fonte: Própria Autora

e o planejador local dinâmico poderá tomar a decisão de para onde desviar caso haja obstáculo (Para onde vou?). Além disso, o módulo de controle poderá definir a melhor velocidade e aceleração a ser tomada para realizar a tarefa (Como faço para chegar lá?).

Segundo Siegwart, Nourbakhsh e Scaramuzza (2011), o sucesso da navegação é dependente de seus quatro elementos básicos:

- **Percepção**: o robô deve interpretar seus sensores para extrair dados significativos;
- **Localização**: o robô deve determinar sua posição no ambiente;
- **Cognição**: o robô deve decidir como agir para atingir seus objetivos;
- **Controle de Movimento**: o robô deve controlar a modulação do motor para alcançar a trajetória desejada.

Na Figura 3.1 podemos ver todos os elementos básicos na navegação. A percepção sendo o módulo de sensores enviando dados importantes para o módulo de evitar colisão e localização. A localização sendo representada pelo próprio módulo de localização. A cognição sendo representada pelo módulo de planejamento de trajetória, que engloba os planejadores local estático e dinâmico. E o controle de movimento sendo representado pelo módulo de controle.

3.2 A importância do Planejamento de Trajetória

Os robôs podem ser utilizados para realizar diversas missões, como as citadas no Capítulo 1. Todas essas são missões diferentes que irão demandar capacidades diferentes. No entanto, a habilidade de se locomover entre dois pontos, ou mais, é comum em todas as missões.

A tarefa de se locomover entre dois pontos é responsabilidade do Planejamento de Trajetória. No entanto, essa trajetória nem sempre é apenas navegar em linha reta entre os pontos, podendo ser necessário evitar obstáculos no ambiente. Além de manter uma margem de erro segura para caso haja alguma falha de comunicação, de controle, entre outros. O planejador precisa estar preparado para gerar essa trajetória em diversos ambientes sendo simples ou não estruturados.

Uma consideração em relação aos UAVs é que para manter uma navegação constante as curvas precisam ser suaves, alterando minimamente o ângulo, evitando rotações abruptas, para obedecer as restrições dos mesmos. Em outras palavras, o robô precisa rotacionar de pouco em pouco, visando uma trajetória suavizada.

Dependendo da missão é necessário levar em consideração o ambiente 3D para realizar o planejamento de trajetória dos UAVs, para descobrir se a trajetória em 3D realmente será melhor do que a trajetória em 2D.

Também é importante que o UAV consiga se locomover em ambientes desconhecidos, criando um mapa ao decorrer da trajetória, visando atualizar a trajetória, considerando todos os obstáculos, caso haja colisão aparente. E caso haja obstáculos dinâmicos é necessário que o planejador seja capaz de desviar da maneira mais rápida possível.

Desta forma, um planejador de trajetória deve ser capaz de gerar o caminho que o UAV irá locomover em qualquer ambiente. Sendo um módulo essencial para qualquer realizar qualquer missão.

3.3 Tipos de Planejamentos

Dentro da navegação móvel existem diversas áreas de pesquisas, entre elas: Planejamento de Rota, Planejamento do Movimento e Planejamento de Trajetória. O foco dessa dissertação é no Planejamento de Trajetória. Para um melhor entendimento, segue as definições destas áreas.

O Planejamento de Rotas é responsável por apresentar a melhor rota a ser tomada entre o ponto inicial e final. Desse modo, é gerado vários pontos intermediários que possam formar um caminho sem colisão durante toda a trajetória (SOTNIK et al., 2020).

O Planejamento do Movimento (Figura 3.1) é responsável pela dinâmica e aerodinâmica do robô, como velocidade, aceleração, ângulo de rotação, entre outros. Além disso, esse planejador também é responsável por estabilizar o robô utilizando técnicas de controle, principalmente veículos aéreos em ambientes externos, devido uma corrente de vento, ou qualquer outra

variável externa (LAUMOND et al., 1998).

O Planejamento de Trajetória é responsável pela caminho que o robô irá seguir entre cada um dos pontos gerados pelo Planejamento de Rota, ou entre o ponto inicial e final. Ou seja, precisa gerar um caminho entre dois pontos em que não haja colisão. Como UAVs possuem restrições dinâmicas e aerodinâmicas a trajetória gerada precisa ser suavizada, permitindo assim que o UAV realize-a facilmente (BORTOFF, 2000).

Para a navegação de um robô móvel o Planejamento de Rota pode ser omitido, principalmente em ambientes simples. No entanto, se ele for usado o custo computacional do Planejamento de Trajetória é reduzido. Já o Planejamento do Movimento da Trajetória são planejamento essenciais para a navegação.

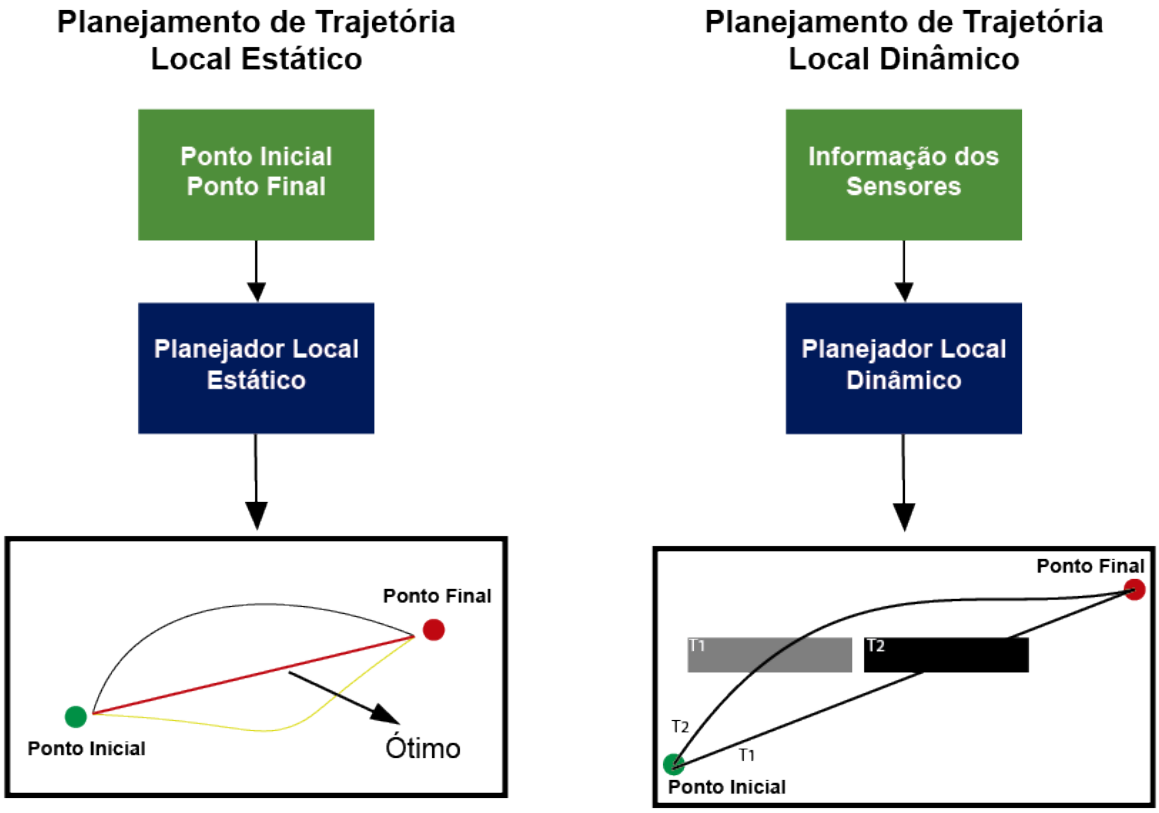
3.4 Diferença entre Planejador Local Estático e Dinâmico

Como citado anteriormente, o Planejamento de Trajetória é responsável por gerar a trajetória entre dois pontos, evitando colisão. Para completar essa missão o Planejamento de Trajetória é dividido em planejador local estático e dinâmico (VIVALDINI et al., 2010), como visto na Figura 3.2.

O planejador local estático é responsável por gerar a trajetória ótima entre o ponto inicial e o ponto final. Essa trajetória precisa evitar todos os obstáculos, e manter uma margem erro se necessário. Também precisa ser suavizada obedecendo as restrições dinâmicas e aerodinâmicas dos UAVs. Além de retornar o caminho com menor comprimento entre os pontos (HWANG; AHUJA et al., 1992).

O planejador local dinâmico é responsável por definir a melhor trajetória entre dois pontos, geralmente próximos, baseados nas informações de sensores. Sendo utilizado para evitar obstáculos, principalmente dinâmicos, durante o percurso. Por trabalhar em tempo real e considerar apenas informações dos sensores é ideal para desviar de obstáculos dinâmicos fazendo um replanejamento da rota apenas no ponto do obstáculo (HWANG; AHUJA et al., 1992). No Planejamento de Trajetória é possível usar apenas um dos planejadores, ou ambos. Ao usar os dois o planejador local estático fica responsável por gerar a trajetória suavizada entre o ponto inicial e final, evitando obstáculos. E o planejador local dinâmico fica responsável pelo replanejamento da trajetória, caso haja obstáculos dinâmicos.

FIGURA 3.2 – Planejador Local Estático e Dinâmico.



Fonte: Adaptado de Lu et al. (2018)

3.5 Como analisar um Planejador de Trajetória

Segundo Aggarwal e Kumar (2019), o maior desafio para o Planejamento de Trajetória é otimizar as seguintes métricas:

- **Comprimento do caminho**: distância total percorrida pelos UAVs desde o ponto inicial até o ponto final;
- ***Completeness***: o planejador deve sempre encontrar um caminho para se locomover entre os pontos, caso exista;
- **Eficiência de tempo**: gerar o trajetória entre o ponto inicial e final em tempo mínimo;
- **Eficiência de custo**: depende do custo computacional total, como espaço na memória e processamento necessário;
- **Eficiência de energia**: gasto de energia em termos de combustível ou bateria;

- **Robustez**: modo que os UAVs tenham a capacidade de tolerar erros de dispositivos sensíveis à posição, erros de direção de rotação, erros lineares de direção durante o planejamento do caminho; e
- **Prevenção de colisão**: mecanismo pelo qual os UAVs têm o poder de detectar as colisões, para que não ocorram danos físicos aos UAVs.

Se um Planejamento de Trajetória para UAVs obter um bom desempenho em cada uma dessas características pode ser considerado que possui uma boa performance. O comprimento do caminho, *completeness*, eficiência de tempo, eficiência de custo, e prevenção de colisão, pode ser avaliado durante a simulação. Para validar a eficiência de energia e a robustez é necessário testes em ambiente real.

Já a robustez é validada comparando os valores do comprimento de caminho entre os dados simulados e os testes em ambiente real e variação em cada eixo das trajetórias. Quanto menor a diferença entre o comprimento das duas trajetórias, e variação entre os eixos, melhor será a robustez do planejador.

3.6 Considerações Finais

Planejamento de Trajetória é uma área de estudo bem difundida na robótica, sendo essencial para a navegação. Este capítulo apresentou uma breve revisão sobre a navegação móvel ressaltando a importância do Planejamento de Trajetória. Geralmente, há uma grande confusão entre a definição de Planejamento de Trajetória e o Planejamento do Movimento. Por isso, este capítulo também explicou a diferença entre os tipos de planejamentos e como são aplicados na navegação móvel. Também abordou-se a trajetória gerada pelo planejador, bem como o papel dos planejadores local estático e dinâmico, pois cada um possui funções específicas.

Por fim, mostrou-se quais são as características de um bom planejador de trajetória e como avaliá-los. No Capítulo 4, apresenta-se as técnicas para realizar o Planejamento de Trajetória utilizadas neste trabalho e as melhorias implementadas.

Capítulo 4

TÉCNICAS DE PLANEJAMENTO DE TRAJETÓRIA

Este capítulo explica o funcionamento e os algoritmos utilizados de cada uma das técnicas adotadas para esse trabalho, ou seja: técnicas clássicas, meta-heurísticas e de aprendizado de máquina. Além de demonstrar as melhorias usadas em cada algoritmo e as que foram propostas.

As técnicas e melhorias apresentadas nesta Seção foram desenvolvidas em Python 3.6 pela autora.

4.1 Técnicas Clássicas

As técnicas clássicas foram as primeiras técnicas a serem usadas na solução para planejamento de trajetória (TAN et al., 2016) (WANG et al., 2019b) (CHEN et al., 2019). Com o uso dessas técnicas é possível confirmar se existe um caminho entre o nó origem e objetivo (AGGARWAL; KUMAR, 2020). Vale ressaltar que as técnicas clássicas são baseadas no espaço disponível no ambiente para o UAV calcular a melhor rota (LI; LU, 2019).

As técnicas clássicas podem ser dividas entre métodos exatos e aproximados, ou baseados em amostragem. Os métodos e seus algoritmos estão sendo explicados a seguir nessa seção.

4.1.1 Exatas

As técnicas clássicas exatas sempre retornam o mesmo caminho. Isso deve-se ao fato de escolher o caminho baseado unicamente em equações matemáticas. A principal desvantagem dessa abordagem é o alto custo computacional e a falha em responder à incerteza presente no ambiente. No entanto, se houver um caminho entre o nó origem e o objetivo, essas técnicas são as que tem mais probabilidade de encontrar. Além de que são altamente precisas, sendo ótimas

técnicas para um voo em tempo real (CHOSET et al., 2005).

As técnicas clássicas exatas utilizadas nessa dissertação foram os algoritmos *A-Star* (A*) e o *Artificial Potential Field* (APF).

4.1.1.1 A-Star

O algoritmo A* foi desenvolvido por Nosrati, Karimi e Hasanvand (2012) com o objetivo de melhorar o custo computacional e tempo de processamento apresentados pelos grafos em problemas de busca. O algoritmo baseia-se na utilização de uma heurística, definida para guiar o processo de busca (DUCHOĖ et al., 2014). Essa heurística estima o custo de deslocamento da origem até o destino, geralmente determinado pela distância entre os nós, o que tende a acelerar o processo exploratório. O algoritmo também acompanha o custo necessário para alcançar cada posição. A formulação matemática para tal pode ser vista na Equação 4.1, sendo g_n o custo e h_n a heurística.

$$f_n = g_n + h_n \tag{4.1}$$

Além disso, o algoritmo utiliza o conceito de conjuntos abertos e fechados para controlar os nós que já foram visitados ou não (NOSRATI; KARIMI; HASANVAND, 2012). Ou seja, os registros de todas as posições que foram alcançadas são salvos nos conjuntos abertos. Neste conjunto é adicionado os nós que já foram visitados. Devido isso, o algoritmo tende a encontrar o menor caminho, caso exista, mesmo com uma função heurística ineficiente. O pseudo código do algoritmo está descrito no Algoritmo 1.

Algoritmo 1: *A-Star*

```
1  início
2      Inicializar conjunto fechado
3      Inicializar conjunto aberto com nó origem
4      while Conjunto aberto != null do
5          if nó atual == nó objetivo then
6              Retornar o conjunto fechado
7          end
8          for n em nós disponíveis no ambiente do
9              if n não está no conjunto aberto then
10                 Adicionar n ao conjunto aberto
11                 Calcular a heurística de acordo com a Eq. 4.1
12                 Definir n como nó atual
13             end
14             else
15                 if Verificar se o caminho entre o nó origem e atual é menor then
16                     Transferir n do conjunto fechado para o aberto
17                     Calcular a heurística de acordo com a Eq. 4.1
18                     Definir n como nó atual
19                 end
20             end
21         end
22         Deletar n do conjunto aberto Adicionar n ao conjunto fechado
23     end
24 fim
```

4.1.1.2 APF

O algoritmo *Artificial Potential Field* cria um mapa onde o UAV é inserido baseado no campo elétrico. O método considera o UAV como uma carga pontual sendo influenciada pelo campo local. O mapa é um campo potencial (Eq. 4.2) formado pela força de atração (Eq. 4.3) do objetivo e as força de repulsão (Eq. 4.4) de cada obstáculo. A ideia da abordagem é fazer com que o veículo seja atraído pelo objetivo e repelido pelos obstáculos. Tal técnica é normalmente usada para realizar o planejamento de trajetória local (KOREN; BORENSTEIN et al., 1991). O pseudo código do algoritmo está descrito no Algoritmo 2.

$$U(W) = U_{att}(W) + U_{rep}(W) \tag{4.2}$$

$$U_{att}(W) = KP\sqrt{(x - x_{goal})^2 + (y - y_{goal})^2} \tag{4.3}$$

$$U_{rep}(W) = ETA(\frac{1}{\sqrt{(x - x_o)^2 + (y - y_o)^2}} - \frac{1}{rr})^2 \tag{4.4}$$

Na Equações 4.2, $U(W)$ é o campo potencial total, $U_{att}(W)$ é a força de atração e $U_{rep}(W)$ é o campo de repulsão, presente no ambiente. Na Equação 4.3, em que é definido o valor de $U_{att}(W)$, KP é o ganho do campo campo de atração, e (x_{goal}, y_{goal}) são as coordenadas do nó objetivo. Na Equação 4.4, em que é definido o valor de $U_{rep}(W)$, ETA é o ganho do campo repulsivo, rr é o raio do robô, e (x_o, y_o) são as coordenadas de cada nó dos obstáculos. Nas Equações 4.3 e 4.4, (x, y) são as coordenadas do nó atual.

Algoritmo 2: *Artificial Potential Field*

```
1  início
2      Calcular o campo de atração de acordo com a Eq. 4.3
3      Calcular o campo de repulsão de acordo com a Eq. 4.4
4      Calcular o campo potencial de acordo com a Eq. 4.2
5      while nó atual != nó objetivo do
6          Mover o nó atual para o nó mais próximo com menor campo potencial
7          Adicionar o nó atual ao caminho
8      end
9      Retornar o caminho
10 fim
```

4.1.2 Aproximadas

As técnicas clássicas aproximadas vem ganhado reconhecimento para resolver problemas de planejamento de trajetória. A principal desvantagem dessas técnicas é que geralmente o caminho retornado tem o formato de um zig-zag, sendo caminhos inviáveis de serem executados pelos UAVs (WANG et al., 2019b). No entanto, essas técnicas possuem baixo custo computacional, inclusive em ambiente grandes e não estruturados, pois inserem nós aleatórios em todo cenário. Sendo assim, conseguem analisar menos pontos e possuir um passo maior (CHOSET et al., 2005).

As técnicas clássicas exatas utilizadas nessa dissertação são *Probabilistic Roadmap* (PRM), *Rapid Random Tree* (RRT) e *Rapid Random Tree - Connect* (RRT-C).

4.1.2.1 PRM

O mapa probabilístico é utilizado para resolver os problemas de planejamento de trajetória em ambientes estáticos (KAVRAKI et al., 1996a). Embora existam soluções baseadas no PRM para ambientes dinâmicos. O algoritmo consiste em duas fases. A primeira é de aprendizado e a segunda de questionamento. Na primeira fase, o UAV cria um conhecimento sobre a região que deve ser explorada e a armazena em um grafo. Após a geração do mapa de rotas, é realizado o pós-processamento visando aumentar a conectividade do mapa de rotas nas regiões mais difíceis (KAVRAKI et al., 1996b). A partir de uma abordagem probabilística, as rotas são definidas de forma incremental.

Na segunda fase, o nó origem e objetivo são adicionados ao grafo. O caminho é obtido através do menor caminho do algoritmo de Dijkstra. Para melhorar o resultado do algoritmo seria necessário aumentar o número de amostra, de modo que não gere grande perda computacional (KAVRAKI et al., 1996b). O pseudo código do algoritmo está descrito no Algoritmo 3.

Algoritmo 3: *Probabilistic Road Map*

```
1  início
2      Inicializar o caminho com o nó inicial
3      Inserir diversos nós de amostragem no ambiente
4      Checar se não há colisão nos nós
5      for Nó atual != nó objetivo do
6          Escolher o nó vizinho com menor distância até o nó objetivo
7          if Último nó do caminho não colide com o nó atual then
8              Adicionar o nó atual ao caminho Atualizar o nó atual
9          end
10     end
11 fim
```

A árvore aleatória rápida gera nós aleatoriamente e é conectada ao nó disponível mais próximo (LAVALLE; KUFFNER, 2000). O algoritmo se expande no espaço livre em direção a nós posicionados aleatoriamente. Dessa forma, uma árvore é construída, e cada interação é verificada se houver uma colisão com um obstáculo. Se a distância do nó aleatório e do vértice mais próximo for maior que o fator de crescimento, ela será limitada ao valor desse fator.

4.1.2.2 RRT

O algoritmo *Rapid Random Tree* desenvolvido por (LAVALLE, 1998a), gera nós aleatoriamente que são conectados ao nó mais próximo disponível. Os nós se expandem no espaço livre em direção a outros nós posicionados aleatoriamente. Dessa forma, uma árvore é construída, e cada interação é verificada se há colisão com obstáculo. Se a distância entre o nó aleatório e o nó atual for maior que o fator de crescimento, ela será limitada ao valor desse fator.

Por ser um algoritmo baseado em amostragem, possui um custo computacional relativamente baixo (LAVALLE, 1998b). Esse algoritmo evita que rotas não sejam encontradas, caso existam, como acontece em algoritmos cinemáticos (KOSLOSKY et al., 2018). (LAVALLE, 1998b) mostrou as principais vantagens do RRT:

- a expansão do RRT é tende a visitar mais os espaços inexplorados;
- a distribuição dos vértices do RRT se aproxima da distribuição de amostras, levando a um comportamento consistente;
- o RRT é relativamente simples, o que facilita a análise de performance;
- o caminho gerado pelo RRT sempre está conectado, mesmo que o número de arestas seja mínima;
- o RRT pode ser adotado e implementado em uma grande variedade de sistemas de planejamento.

O pseudo código do algoritmo está descrito no Algoritmo 4.

Algoritmo 4: *Rapid Random Tree*

```
1  início
2      for i em interações do
3          Gerar nós aleatórios (x, y)
4          Encontrar os nós mais próximos (xn, yn)
5          if Não tiver colisão entre (x, y) e (xn, yn) then
6              Adicionar x e y ao caminho, sendo ligados a xn e yn
7          end
8          if Distância entre o último nó do caminho e o nó objetivo < tamanho de
             expansão then
9              if Não colidir entre o nó objetivo e o último nó do caminho then
10                 Adicionar o nó objetivo ao caminho Retornar o caminho
11             end
12         end
13     end
14 fim
```

4.1.2.3 RRT-C

Devido a popularidade do RRT foram implementadas diversas variações do algoritmo, visando melhorar ainda mais a performance (NOREEN; KHAN; HABIB, 2016). Sendo umas delas o *RRT-Connect* que utiliza duas árvores RRT (KLEMM et al., 2015). Uma das árvores nasce do nó de origem, T_a, e a outra do nó de destino, T_b, por essa razão também é chamado de RRT-bidirecional (LAVALLE, 2006; KUFFNER; LAVALLE, 2000). Enquanto a árvore do RRT vai crescendo, o algoritmo tenta conectar T_b com T_a, enquanto são expandidas. Caso consiga, será considerado a última expansão da árvore. Caso contrário, as duas árvores trocam de função, a árvore que estava expandindo agora vai tentar se conectar, e a que estava tentando se conectar tentará expandir. Se mesmo assim o caminho não for encontrado T_a expande até o ponto mais próximo possível (KOSLOSKY et al., 2018).

O *RRT-Connect* divide a custo computacional entre a exploração do ambiente e o crescimento das árvores, uma em direção a outra, até serem conectadas e a solução encontrada (RAJA; PUGAZHENTHI, 2012). O pseudo código do algoritmo está descrito no Algoritmo 5.

Algoritmo 5: *Rapid Random Tree - Connect*

```
1  início
2      Inicializar o caminho com o nó inicial
3      if Distância entre o nó origem até o objetivo < tamanho de expansão & não
         colidir then
4          Retornar caminho com nó origem e objetivo
5      end
6      else
7          while Árvore 2 não colide do
8              Expandir a árvore 2 como no Algoritmo 4
9          end
10         while Distância entre a árvore 1 e 2 < tamanho de expansão & não colidir do
11             Expandir a árvore 1 como no Algoritmo 4
12         end
13     end
14     Conectar os caminhos Retornar o caminho
15 fim
```

4.2 Técnicas Meta-Heurísticas

As técnicas meta-heurísticas são algoritmos genéricos para resolver problemas de otimização, sem ser necessário descrever cada passo até o resultado. A cada interação o algoritmo aprende mais sobre o problema e melhora a trajetória que será retornada no final. O algoritmo busca encontrar a melhor resposta possível em um tempo computacional aceitável (DOKEROGLU et al., 2019). Além de possuir grande capacidade para lidar com a incerteza presente no ambiente (PATLE et al., 2019).

As técnicas meta-heurísticas adotadas nessa dissertação foram as *Particle Swarm Optimization* (PSO), *Gray Wolf Optimization* (GWO), e o *Glowworm Swarm Optimization* (GSO).

4.2.1 PSO

PSO é baseado no movimento dos pássaros. O algoritmo foi criado devido o interesse em descobrir as regras subjacentes que permitiam que um grande número de pássaros se reunissem de maneira síncrona, mudando repentinamente de direção, dispersando e reagrupando, várias

vezes. A partir das simulações obteve-se que o movimento é baseado na velocidade do vizinho mais próximo e em valores aleatórios (EBERHART; KENNEDY, 1995).

Ao inicializar o algoritmo, o caminho do primeiro pássaro é gerado aleatoriamente, a velocidade é definida como zero e o caminho é definido como o melhor. A cada interação o caminho é atualizado. A velocidade é atualizada baseada em valores passados da velocidade, no melhor caminho caminho, no caminho gerado anteriormente e em valores aleatórios, como mostrado na Equação 4.5. Em seguida os valores da velocidade são limitados, já que há um valor máximo e mínimo ao qual o pássaro pode se mexer. Então, o caminho é atualizado, somando os valores do caminho anterior com a velocidade calculada. Por fim, o valores do caminho a ser gerado é limitado de acordo com os limites do ambiente.

$$p_v = p_v w + c_1 r_1 (b_p - p) + c_2 r_2 (b_p - p) \tag{4.5}$$

Na Equação 4.5, a variável p_v é a velocidade da partícula anterior e w é um valor que diminuindo linearmente de 2 até 0. Esse valor representa a quantidade de mudanças que pode ter a cada geração, ou seja, nas primeiras gerações o algoritmo tende a fazer otimizações mais bruscas e ao decorrer das interações as interações vão sendo mais suaves. O valor de c_1 e c_2 são constantes que controlam a exploração e o *exploitation* do algoritmo e as variáveis r_1 e r_2 são dois valores aleatórios diferentes, entre 0 e 1. O valor de b_p são as coordenadas do melhor caminho e p são as coordenadas do último caminho. O pseudo código pode ser visto no Algoritmo 6.

Algoritmo 6: *Particle Swarm Optimization*

```
1  início
2      Inicializar partículas
3      Gerar um caminho aleatório
4      Definir o melhor caminho
5      for i em interações do
6          Definir a velocidade da partícula de acordo com a Eq. 4.5
7          Limitar a velocidade da partícula de acordo com o máximo e mínimo permitido
8          Atualizar o caminho com a velocidade
9          Limitar o caminho com o tamanho do ambiente
10         Definir o melhor caminho de acordo com o menor custo
11     end
12     Retornar o melhor caminho
13 fim
```

4.2.2 GWO

O algoritmo GWO é inspirado no processo de caça encontrado em lobos cinzentos. os lobos cinzentos preferem viver em pequenos grupos, com média de 14 membros, seguindo uma hierarquia social. Os líderes são os lobos α (alfa), tendo autoridade para realizar as decisões do grupo. Os lobos β (beta) são os próximos, os quais auxiliam o líder a tomar uma decisão. Em seguida vem os lobos δ (delta), que dominam os lobos ômega, os quais são o mais baixo no ranking. No algoritmo do GSO, os lobos α representam a melhor solução, seguidas do beta e delta (PANDA; DAS, 2019).

Ao inicializar o algoritmo, o primeiro conjunto de lobos é formado aleatoriamente e é definido como melhor resultado para os três grupos. A cada interação para atualizar o caminho, cada um dos lobos gera um caminho de acordo com a Equação 4.6. Em seguida, é tirado a média dos valores dos três grupos para atualizar o caminho atual. O melhor caminho sempre irá para o α. Caso o caminho não seja o melhor irá ser verificado se é melhor do que o caminho β e depois de δ.

$$w_p = |r_v b_p(t_w) - p| \tag{4.6}$$

Como no PSO, o w é uma variável que diminui linearmente de 2 até 0. Esse valor representa a quantidade de mudanças que pode ter a cada geração, ou seja, nas primeiras gerações o

algoritmo tende a fazer otimizações mais bruscas e ao decorrer das interações as interações vão sendo mais suaves.

Na Equação 4.7, w_p é a posição do lobo em cada iteração, $c1$ é uma constante do algoritmo, a qual foi utilizado o valor de 1.5. As variáveis $r1$ e $r2$ são valores aleatórios de 0 a 1. O valor de p é referente ao caminho atual, e b_p é o melhor caminho referente a categoria de lobo (t_w) que está sendo feito o cálculo. O pseudo código pode ser visto no Algoritmo 7.

$$r_v = r2(c1wr1) \tag{4.7}$$

Algoritmo 7: *Grey Wolf Optimization*

início
  Inicializar lobos
  Gerar um caminho aleatório
  Definir o melhor caminho para cada lobo **for** *i em interações* **do**
    Definir o lobo α de acordo com a Eq. 4.6 com a melhor posição de α Definir o lobo β de acordo com a Eq. 4.6 com a melhor posição de β Definir o lobo δ de acordo com a Eq. 4.6 com a melhor posição de δ Definir o melhor caminho para cada lobo Atualizar o caminho de acordo com a média entre os lobos
  end
  Retornar o melhor caminho de α
fim

4.2.3 GSO

GSO foi desenvolvido baseado no comportamento dos vaga-lumes. Os vaga-lumes possuem a habilidade de alterar a intensidade da sua luminescência. Cada vaga-lume possui um pigmento luminoso, denominado luciferina. Este pigmento é modificado de acordo com sua posição, permitindo que o vaga-lume brilhe em uma intensidade similar a da função que está sendo otimizada. Cada vaga-lume seleciona o vizinho com maior luciferina, a partir de um mecanismo probabilístico, e se move em direção a ele. No entanto, o vaga-lume só pode seguir um certo número de vermes próximos e até uma distancia limitada. Naturalmente, os vaga-lumes utilizam sua luminescência para sinalizar sua direção para outros vermes (KAIPA; GHOSE, 2017).

Ao inicializar o algoritmo, o caminho do primeiro vaga-lume é gerado aleatoriamente, sua luciferina é definida como zero e o caminho é definido sendo o melhor. Em seguida o valor de

luciferina é atualizado de acordo com o valor anterior de luciferina e a quantidade proporcional de luciferina em relação ao caminho anterior e o melhor caminho, como visto em Equação 4.8. Em seguida os melhores vizinhos são selecionados de acordo com a Equação 4.9. A partir disso, o melhor vizinho é escolhido para ser seguido, a partir de um mecanismo probabilístico. O vaga-lume que possuir o maior valor de luciferina, ou seja, os vaga-lumes são atraídos pelos vizinhos que brilham mais. Por fim, o caminho é atualizado, de acordo com a Equação 4.10.

$$l = (1 - l_d) \cdot l + l_e \cdot g_c \tag{4.8}$$

$$N = all((dist < range)\&(l < l_v)) \tag{4.9}$$

$$n_p = p + s \cdot (p_t - p)/norm(p_t - p) \tag{4.10}$$

Na Equação 4.8, a variável l_d é o decaimento da luminosidade do vaga-lume, l é a luminosidade do vaga-lume anterior, l_e é o realce da luminosidade do vaga-lume e g_c é o custo do caminho do vaga-lume anterior. Na Equação 4.9, *dist* é a distância entre cada ponto do caminho gerado pelo vaga-lume atual e o vizinho que está sendo verificado, *range* é o alcance para visualizar os vizinhos e l_v é a luminosidade do vaga-lume vizinho. Na Equação 4.11, a variável n_v é quantidade de vizinhos utilizados na interação atual.

$$p_g = \frac{l_v - l}{|\sum l - (n_v \cdot l)|} \tag{4.11}$$

$$m_v = \lambda \cdot (k_n - n_v) \tag{4.12}$$

Na Equação 4.10, a variável p é o caminho atual, s é o valor máximo em que a nova posição pode variar da atual e p_t é o caminho em que o melhor vizinho está fazendo. Na Equação 4.12, a variável *beta* é o valor para ir diminuindo aos poucos a quantidade de vizinhos viáveis para cada iteração, k_n é o valor máximo de vizinhos e n_v é quantidade de vizinhos utilizados na iteração atual. Na Equação 4.13, a variável r_b é o alcance máximo dos vaga-lumes. O pseudo código pode ser visto no Algoritmo 8.

$$range = min(r_b, max(0.1, range + m_v)) \tag{4.13}$$

Algoritmo 8: *Glowworm Swarm Optimization*

```
início
    Inicializar os vaga-lumes
    Gerar um caminho aleatório
    Definir o melhor caminho
    for i em interações do
        Atualizar a luciferina (Eq. 4.8)
        Definir os vizinhos de acordo com a Eq. 4.9
        Definir a probabilidade de movimento até cada vizinho (Eq. 4.11)
        Escolher o vizinho com maior probabilidade de ser seguido
        Atualizar o caminho (Eq. 4.10)
        Atualizar o alcance dos vizinhos (Eq. 4.13)
        Definir o melhor caminho de acordo com o menor custo
    end
    Retornar o melhor caminho
fim
```

4.3 Técnicas de Aprendizado de Máquina

A abordagem mais recente para resolver problemas de planejamento de trajetória é a partir de técnicas de aprendizado de máquina. É possível usar técnicas de aprendizado supervisionado, aprendizado não supervisionado, e de aprendizado por reforço.

Dentre essas técnicas, o aprendizado por reforço que é o mais utilizado para resolver os problemas de planejamento de trajetória dos UAVs (CARRIO et al., 2017; WANG et al., 2017; SICHKAR, 2019; QU et al., 2020a). O aprendizado por reforço aprende a se movimentar em um ambiente específico, retornando a melhor trajetória. A trajetória ótima pode ser encontrada a partir de agentes que devem realizar ações em um ambiente para maximizar a recompensa cumulativa. Essas recompensas são obtidas por movimentos que podem ser úteis (uma recompensa positiva) ou ruins, como uma parede (uma punição, recompensa negativa) (LISON, 2015).

A técnica de aprendizado de máquina utilizada nessa dissertação é o *Q-Learning*. A vantagem desse algoritmo é devido a sua simplicidade e velocidade. O *Q-Learning* tem garantia de convergência, tende a aprender a política em menos tempo e, em relação a outros algoritmos de aprendizado por reforço, a representação do estado é mais simples, facilitando o processo de

geração de trajetória para o UAV (WASKOW, 2010).

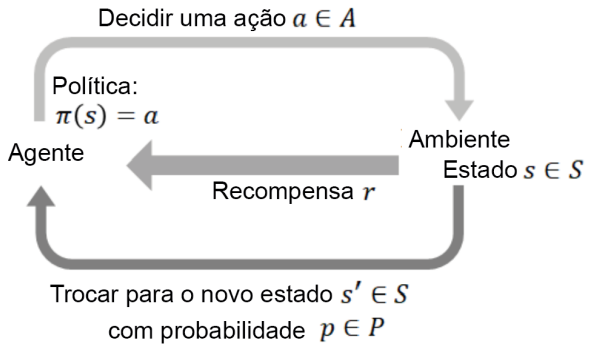
4.3.1 *Q-Learning*

O *Q-Learning* é uma técnica de aprendizado de máquina que funciona baseado em recompensa e punição, sem precisar especificar como a tarefa deve ser alcançada. O agente aprende a se comportar no ambiente através de interações baseadas em tentativa e erro (KAELBLING; LITTMAN; MOORE, 1996).

No aprendizado por reforço o agente atua no ambiente. O agente pode estar em apenas um estado do conjunto de estados ($s \in S$) do ambiente, e escolhe uma ação entre as várias possíveis ($a \in A$) para trocar entre os estados. O próximo estado em que o agente irá é decidido pela probabilidade de transição entre os estados (P). Uma vez que a ação é realizada, é recebido uma recompensa ($r \in R$) que pode ser positiva ou negativa.

A política π do agente fornece a informação sobre qual é a ação ótima a ser executada em um determinado estado s com o objetivo de maximizar as recompensas totais. Cada estado está associado a uma função de valor $v\pi$ que prevê a quantidade esperada de recompensas futuras que irá receber nesse estado, de acordo com a política π. O funcionamento do aprendizado por reforço pode ser visto na Fig. 4.1.

FIGURA 4.1 – Descrição Formal do Aprendizado por Reforço.



Fonte: Adaptado do Laboratório de *Deep Learning* da UCA

No Planejamento de Trajetória o conjunto S é o de posições possíveis em que o UAV pode se locomover. As ações possíveis A é como o UAV pode chegar até elas, ou seja, movimentando para frente, trás, esquerda ou direita. A probabilidade de transição P entre cada estado é definido pela equação de Bellman (Eq. 4.14). A recompensa r é negativa caso seja identificado colisão, caso contrário será positiva. A política π é a trajetória que o algoritmo irá retornar, informando

as melhores decisões a se tomar durante o caminho.

$$Q[pos,a] += \alpha \cdot (r + \gamma max(Q[nPos,mx]) - Q[(pos,mx),a]) \tag{4.14}$$

O aprendizado por reforço pode ser dividido em dois grupos: métodos baseados em políticas e métodos baseados em valores (QU et al., 2020a). Nos métodos baseados em política, é construído uma representação da política, a qual é mantida durante o aprendizado (NACHUM et al., 2017). Já nos métodos baseados em valores, nenhuma política é armazenada explicitamente, apenas a função de valor. A política está implícita e pode ser encontrada a partir da função de valor (NACHUM et al., 2017). Nós utilizaremos nessa dissertação os métodos baseados em valores, o *Q-Learning* (SUTTON et al., 2000; NACHUM; NOROUZI; SCHUURMANS, 2016).

O algoritmo *Q-Learning* é usado para encontrar a política ideal de seleção de ações no MDP (*Markov Decision Process*) (KIM et al., 2017). Para isso, o agente executa as ações com os maiores valores Q esperados para estimar a política ideal. A tabela Q é atualizada baseada na recompensa (QU et al., 2020a). O pseudo código da técnica de aprendizado por reforço para Planejamento de Trajetória pode ser visto no Algoritmo 9.

Algoritmo 9: *Q-Learning*

```
1  início
2      Gerar matriz de probabilidade de movimento do ambiente while nó atual != nó
         objetivo do
3          if Valor aleatório < ε then
4              Definir ação aleatoriamente
5          end
6          else
7              Definir ação de acordo com o tabela Q treinada até o momento
8          end
9          Atualizar posição Atualizar tabela Q de acordo com a Eq. 4.14
10     end
11     while nó atual != nó objetivo do
12         Definir próximo movimento de acordo com a tabela Q Atualizar posição
13     end
14     Retornar o melhor caminho
15 fim
```

4.4 Melhorias

O ideal para as trajetórias a serem seguidas pelo UAV é que façam o mínimo de curvas possível, minimizando o *Jerk* [1] e mantendo o torque constante (GOEL et al., 2018). As curvas precisam ser o mais suave possível, caso sejam necessárias, obedecendo as restrições do UAV (PANDEY; SHUKLA; TIWARI, 2018). Além de que, para um bom planejamento de trajetória é essencial ter um *completeness* de 100% ou próximo, tendo eficiência de custo e tempo. Por isso foram implementados algumas melhorias nos algoritmos citados anteriormente.

4.4.1 *Artificial Potential Field*

Lifen et al. (2016) propuseram uma melhoria no campo de repulsão do APF para evitar o problema de mínimo local. Os resultados mostraram que a melhoria pode ajudar o UAV a evitar colisões e a encontrar o caminho ideal.

O novo campo de repulsão é formado pela soma de dois campos mostrados nas Equações 4.15 e 4.16, ou seja, a fórmula para encontrar o campo potencial com essa melhoria é dado pela Equação 4.17 e não mais pela Equação 4.2.

$$U_{rep1}(W) = ETA(\frac{1}{q(W)} - \frac{1}{rr})(\frac{1}{Q(W)})Q(W) \tag{4.15}$$

$$U_{rep2}(W) = ETA(\frac{1}{q(W)} - \frac{1}{raio})dir(W)q(W) \tag{4.16}$$

$$U(W) = U_{att}(W) + U_{rep1}(W) + U_{rep2}(W) \tag{4.17}$$

O campo potencial utilizado no ambiente é definido por $U(W)$, sendo $U_{att}(W)$ o valor demostrado na Equação 4.3. $U_{rep1}(W)$ é o primeiro campo de repulsão calculado de acordo com a Equação 4.15 e $U_{rep2}(W)$ é o segundo campo de repulsão calculado de acordo com a Equação 4.16. Nas equações acima, dir é o vetor direção entre o nó atual e o nó objetivo, rr é o raio do UAV, ETA é o ganho do campo repulsivo, $Q(W)$ pode ser visto na Equação 4.18, e $q(W)$ pode ser visto na Equação 4.19. Em que $(x(W),y(W))$ são as coordenadas do nó atual e (x_{gr},y_{gr}) coordenadas dos obstáculos.

[1] Taxa de variação da aceleração

$$Q(W) = (x(W) - x_{gr})^2 + (y(W) - y_{gr})^2 \tag{4.18}$$

$$q(W) = \sqrt{Q(W)} \tag{4.19}$$

Com a melhoria do campo de repulsão do APF as trajetórias ficaram mais suaves e o algoritmo ficou mais rápido diminuindo o número de ambientes em que o UAV ficava preso em mínimo local. No entanto, ainda havia ambientes em que o Planejamento de Trajetória não conseguia alcançar o nó objetivo. Por isso, os autores desta dissertação implementaram o APF bidirecional, baseado na ideia de McIntyre, Naeem e Xu (2016).

Diferentemente de McIntyre, Naeem e Xu (2016) que variam o ponto de verificação da escolha do próximo nó entre o nó inicial e objetivo a cada iteração. Neste trabalho, os autores implementaram a troca entre o nó origem e objetivo para ocorrer apenas quando alguma das buscas se encontrar em mínimo local, sendo que o primeiro cálculo do campo potencial é feito pelo nó objetivo. Da forma proposta por McIntyre, Naeem e Xu (2016) ainda haveria casos em que ocorreria mínimos locais, especialmente em ambientes não estruturados e desconhecidos. Além de que o caminho pode ficar muito sinuoso devido tantas alterações ao campo. O pseudo código do APF bidirecional melhorado ser visto no Algoritmo 10.

Algoritmo 10: APF Bidirecional

```
1 início
2     while mínimo local ou nó origem não são encontrados do
3         Usar algoritmo 2 a partir do nó objetivo (c1)
4     end
5     Usar algoritmo 2 a partir do nó origem até encontrar o último ponto de c1
6     Retornar ambos caminhos encontrados
7 fim
```

As duas melhorias citadas anteriormente conseguem evitar a maioria dos problemas de mínimo local existentes pelo APF. Porém, estamos pesquisando sobre ambientes não estruturados, então ainda há ambientes em que essas melhorias não encontrarão resposta, como quando há pouca distância entre o nó origem e objetivo, mas há um grande obstáculo entre eles.

Por essa razão, os autores deste trabalho também implementaram uma melhoria baseada nos pontos em LoS (*Line of Sight*), ou seja, visada direta, do nó origem e objetivo. O algoritmo faz uma busca desde o nó objetivo até o nó origem. Inicialmente, é criado um nó auxiliar que receberá o valor do nó objetivo. Em uma lista, serão adicionados todos os nós em LoS a partir

do nó auxiliar. O nó com maior valor é adicionado a rota e será o novo nó auxiliar. Esses passos se repetem até que o nó auxiliar seja o nó origem, significando que foi encontrado um caminho entre o nó origem e objetivo. O pseudo código da melhoria pode ser visto no Algoritmo 11.

Algoritmo 11: Evasão do Mínimo Local

```
1 início
2     nó auxiliar = nó objetivo while nó auxiliar != nó origem do
3         l1 = definir os pontos que estão em LoS a partir do nó auxiliar
4         n1 = definir o nó com maior distância
5         Adicionar n1 a rota a ser gerada
6         nó auxiliar = n1
7     end
8     Retornar rota gerada
9 fim
```

Por fim, visando diminuir o tempo de processamento do algoritmo o caminho que está sendo planejado é completo por uma linha reta caso o nó atual e o nó objetivo estejam em LoS. Caso o nó atual encontre o nó objetivo logo após um obstáculo, o caminho será suavizado para evitar o obstáculo e chegar no nó objetivo de maneira que não viole as restrições dinâmicas e aerodinâmicas no UAV.

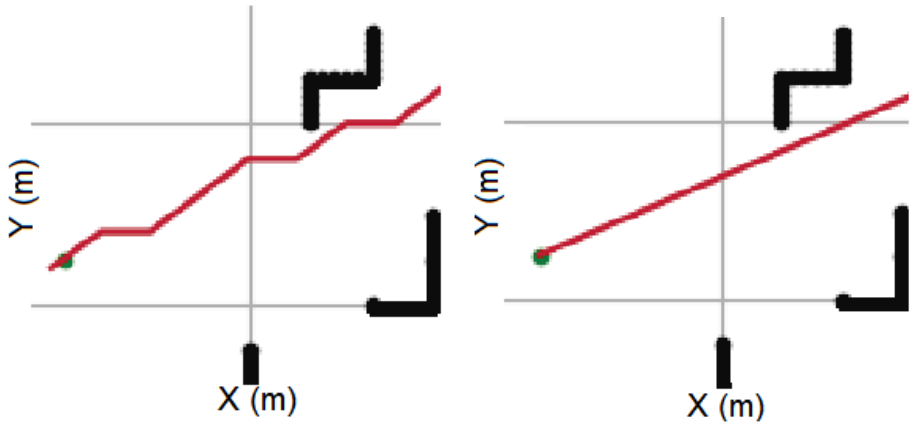
4.4.2 Minimizando Curvas

Os algoritmos podem não gerar o menor caminho entre o nó origem e o objetivo. Isso deve-se ao fato de que os algoritmos baseados no RRT geram caminhos aleatórios até o ponto objetivo, ou seja, em uma parte do ambiente sem obstáculo, o caminho é semelhante a forma de um Z, em vez de uma reta (WANG et al., 2019b). O algoritmo do A* também acaba não retornando o melhor caminho já que é baseado em grade, ou seja, só pode andar em vertical, horizontal ou diagonal a 45° (NOREEN et al., 2019), como mostrado na Figura 4.2. O mesmo pode ocorrer no *Probabilistic Roadmap* já que só considera alguns nós. Por isso, nesta dissertação, foi implementado uma melhoria na trajetória dos algoritmos, como mostrado em (NOREEN et al., 2019), visando torná-la mais objetiva até o destino final, minimizando o número de curvas realizadas.

A melhoria da trajetória funciona a partir da análise entre os pontos da mesma. São utilizados dois pontos de análise, ao início do algoritmo, esses pontos são definidos como os primeiros dois pontos da trajetória. É checado se há colisão entre os pontos. Caso não haja colisão o se-

gundo ponto de análise será trocado pelo próximo ponto da trajetória e a colisão será checada novamente. Caso haja colisão entre os dois pontos e for uma técnica clássica exata é adicionado a nova rota o valor do segundo ponto de análise, que estava sendo observado, e do ponto anterior a ele. Caso haja colisão entre os dois pontos e não for uma técnica clássica exata é adicionado apenas o segundo ponto de análise, que estava sendo observado. Além de, atualizar o ponto 1 de checagem para o valor do ponto 2 de checagem e o mesmo será atualizado com o valor do próximo ponto da trajetória. O pseudo código pode ser visto no Algoritmo 12.

FIGURA 4.2 – Diminuição do Número de Curvas.



(a) Caminho Restrito às Arestas (b) Caminho Melhorado

Fonte: Própria Autora

Algoritmo 12: Minimização de Curvas

```
1  início
2      p1 inicializa com o nó origem for i em caminho, começando pelo segundo nó do
3          p2 = i if Colisão entre o último nó de p1 e p2 then
4              if Se for uma técnica clássica exata then
5                  adicionar i e i − 1 a p1
6              end
7              else
8                  adicionar p2 a p1
9              end
10             p1 = p2 p2 = i + 1
11         end
12         else
13             p2 = i + 1
14         end
15     end
16     Retornar p1
17 fim
```

4.5 Considerações Finais

Neste capítulo foram apresentadas as técnicas de Planejamento de Trajetória que estão sendo usadas nessa dissertação. Além de demonstrar as melhorias implementadas em cada algoritmo e as que foram propostas, visando aprimorar o planejador de acordo com o que foi descrito na Seção 3.5.

Capítulo 5

PLANEJADOR PROPOSTO

Neste capítulo é apresentado a arquitetura do planejador, mostrando como ele atua para se movimentar em ambientes desconhecidos e se localizar no ambiente, como identifica e desvia dos obstáculos dinâmicos e realiza a suavização das trajetórias. Bem como, a atuação do tomador de decisões. Além de explicar como o planejador funciona em ambiente simulado e real.

5.1 Arquitetura Geral

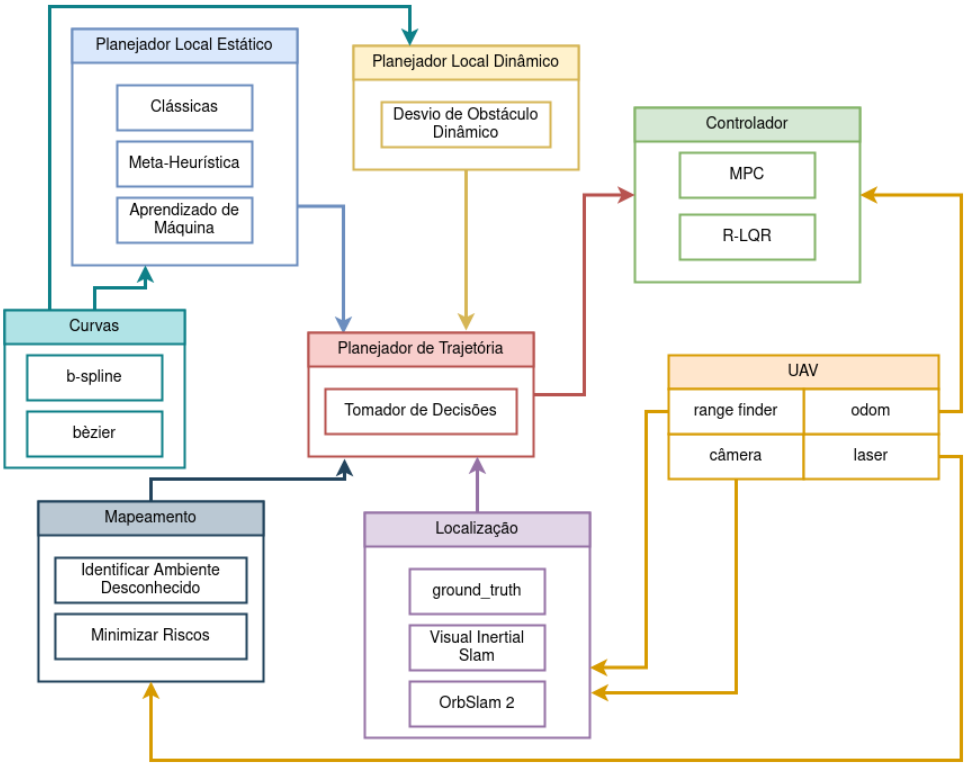
A arquitetura do planejador pode ser vista na Figura 5.1. Nela é apresentado os seguintes módulos:

- **UAV**: No módulo do UAV mostra-se as informações utilizadas que vem dos sensores do UAV.
- **Controle**: Nos testes com simulador e em ambiente real é utilizado o controlador mostrado na Seção 5.4 para movimentar o UAV.
- **Mapeamento**: O mapeamento será feito de acordo com o algoritmo descrito na Seção 5.5. Neste mesmo módulo foi inserido o algoritmo de minimização de riscos, descrito na Seção 5.6.
- **Localização**: O funcionamento do módulo de localização está sendo descrito na Seção 5.7.
- **Planejador Local Estático**: No planejador local estático foram inseridas as técnicas mostradas no Capítulo 4.

- **Planejador Local Dinâmico**: No planejador local dinâmico foi implementado a técnica mostrada na Seção 5.8, o Algoritmo 13.
- **Curvas**: No módulos das curvas está inserido o algoritmo descrito na Seção 5.9 e as curvas de Bèzier, caso seja preciso utilizá-la.
- **Planejador de Trajetória**: O modo de funcionamento do módulo do planejador de trajetória será demonstrado na Seção 5.10, mostrado no fluxograma da Figura 5.9.

Na Seção 5.2 é visto como é feito a integração do planejador com o UAV, e a descrição das plataformas de testes utilizadas. O mesmo *framework* é adotado para o ambiente simulado com o UAV F450, no Gazebo, e no ambiente real com o *Parrot Bebop* 2.

FIGURA 5.1 – Arquitetura do Planejador Proposto.



Fonte: Própria Autora

5.2 Plataforma de Testes

Os testes foram realizados em 3 ambientes, simulação em Python, simulação no Gazebo, e em ambiente real. Todas as simulações foram feitas em *Software-in-the-Loop* (SIL).

Os primeiros testes são realizados em Python para validar o planejador. Depois no simulador Gazebo, para validar o planejador com todos os outros algoritmos necessários, como controle, mapeamento e localização, que são utilizados da mesma maneira em ambiente real e simulado. Por último, os testes são feitos em ambiente real validando o comportamento do UAV para missões.

Em Python, as simulações foram feitas com a biblioteca *matplotlib* de maneira animada, para entender as reações do planejador ao decorrer da trajetória.

Em ambiente simulado foi utilizado o simulador Gazebo, o qual é um simulador de robótica 3D de código aberto e integra motor de física, renderização *OpenGL* e código para simulação de sensores e controle de atuadores (MICHAL, 2010; NOORI et al., 2017). Diversas pesquisas vem adotando-o como ferramenta de simulação (WANG et al., 2020; POPOVIĆ et al., 2020; Zhang et al., 2015; MEYER et al., 2012; KANCIR; DIGUET; SEVAUX, 2019; BáčA et al., 2020), pois oferece a capacidade de incluir diversos ambientes com acurácia e eficiência permitindo a realização de testes preliminares.

Nesta dissertação, o Gazebo foi utilizado em conjunto com o *Muti Robot Systems* (MRS) para fazer as simulações (BACA et al., 2020). Esse sistema oferece um conjunto de pacotes para facilitar a utilização dos UAVs, em ambiente simulado e real, incluindo pacotes de controle, estimação de estados, e vários sensores. Desse modo, é possível validar de maneira segura de abordagens em planejamento, controle, estimação de estados, visão computacional, rastreamento, entre outos.

Para realizar os testes em ambiente simulado e em real é necessário estabelecer uma comunicação com o UAV. Para isso, será utilizado o *Robot Operating System* (ROS), um *framework* para escrever códigos para robôs (QUIGLEY et al., 2009). Desta forma, o planejador baseado em ROS irá apenas importar os nós definidos pelas técnicas de Planejamento de Trajetória, e em seguida, executá-los.

5.3 UAVs

O *frame* selecionado para ambiente simulado foi o F450 (Fig. 5.2), que está implementado no sistema da MRS. Desse modo, é possível realizar a validação do algoritmo de planejamento de trajetória em ambiente simulado, facilitando os testes em ambiente real. Essa integração é possível pois a comunicação é feita baseada em ROS.

O sistema da MRS (BACA et al., 2020) possui diversos *drivers* de sensores que possibilitam

FIGURA 5.2 – F450 em ambiente simulado, no simulador Gazebo.

Fonte: Própria Autora

o desenvolvimento de algoritmos de localização e mapeamento, como câmeras, IMU (*Inertial Measurement Unit*) e medidor de distância à laser e GPS, tais como: *OPTFLOW*, LiDAR, VIO, *SLAM* e *Hector SLAM*.

O UAV implementado considera o modelo dinâmico de veículos aéreos multi rotores (LEE; LEOK; MCCLAMROCH, 2010). As coordenadas do *frame* são as mesmas das coordenadas do mundo (Gazebo), com exceção da velocidade angular ω. A comunicação do controle do UAV depende de variáveis de estado definidas como:

- $\mathbf{r} = [x, y, z]^T \longrightarrow$ posição do centro de massa do UAV em relação ao mundo;
- $\dot{\mathbf{r}} \in \mathbb{R}^3 \longrightarrow$ velocidade do centro de massa do UAV em relação ao mundo;
- $\ddot{\mathbf{r}} \in \mathbb{R}^3 \longrightarrow$ aceleração do centro de massa do UAV em relação ao mundo;
- $\mathbf{R} \in SO(3) \subseteq \mathbb{R}^{3x3} \longrightarrow$ matriz de rotação a partir do corpo do UAV para o mundo;
- $\omega = [\omega_1, \omega_2, \omega_3]^T \longrightarrow$ velocidade angular do *frame* do UAV.

Esses estados são conectados a um modelo não linear, que possuem a parte de translação (Eq. 5.1) e de rotação (Eq. 5.2). Nessas equações, Ω é o tensor da velocidade angular sob as condições: $\Omega v = \omega X v, \forall v \in \mathbb{R}^3$

$$m\ddot{r} = fR\hat{e}_3 - mg\hat{e}_3 \tag{5.1}$$

$$\dot{R} = R\Omega \tag{5.2}$$

O UAV possui aceleração gravitacional para baixo com magnitude $g \in \mathbb{R}$ junto com a força de empuxo f criada coletivamente pelas hélices na direção de $\hat{b}_3$. No entanto, os voos são não

acrobáticos, então é considerado e estimado o azimuth do eixo $\hat{b}_1$ no mundo como o do UAV *heading*. Sob as condições de $|\hat{e}_3^T \hat{b}_1| < 1$, o *heading* é definido como:

$$\eta = atan2(\hat{b}_1^T \hat{e}_2, \hat{b}_1^T \hat{e}_1) \tag{5.3}$$

O vetor *heading* também pode ser definido pelo eixo $\hat{b}_1$ como mostrado na Equação 5.4 e sua forma normalizada mostrada na Equação 5.5.

$$h = [R_{1,1}, R_{2,1}]^T = [b_1^T \hat{e}_1, b_1^T \hat{e}_2, 0] \tag{5.4}$$

$$\hat{h} = \frac{h}{||h||} = [cos\eta, sin\eta, 0]^T \tag{5.5}$$

Já para voos em ambiente real foi utilizado o drone comercial *Parrot Bebop* 2 (Fig. 5.3), disponível no Laboratório de Sistemas Inteligentes (LASI),que também é um UAV bastante difundido no mercado. Os sensores equipados são: uma câmera *fish eye* monocular de alta resolução, sensor ultrassônico, sensor de pressão, giroscópio de 3 eixos, acelerômetro, magnetômetro de 3 eixos, e GNSS (*Global Navigation Satellite System*).

FIGURA 5.3 – *Parrot Bebop* 2.

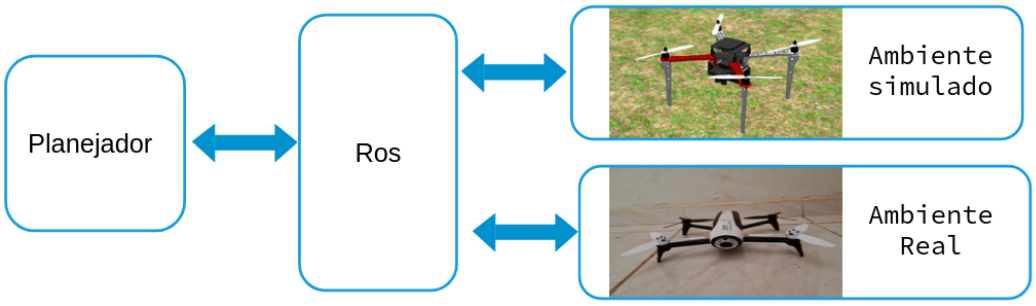
Fonte: (GIERNACKI et al., 2020)

A comunicação entre o *Parrot Bebop* 2 é feita através do ROS em conjunto com pacote *bebop_autonomy*, baseado no kit oficial de desenvolvimento da *Parrot*.

A comunicação com ambos os UAVs é feita da mesma maneira, utilizando a mesma estrutura do planejador, através do ROS, como visto na Figura 5.4. Como a comunicação é padronizada, ambos os UAVs responderam ao planejador como deveriam. A única mudança necessária foi a troca do nome dos tópicos, mas a maneira que são tratados é a mesma, pois é o mesmo formato da mensagem.

O modelo dinâmico simplificado para quadrotores é mostrado em (SANTANA et al., 2014). Apesar desse modelo não substituir a formulação não linear completa, ele ainda permite o controle de posição e velocidade com uma boa aproximação (BENEVIDES et al., 2019).

FIGURA 5.4 – Comunicação do F450 com o ROS em ambiente simulado e real.



Fonte: Própria Autora

Os comandos de velocidade são $v = [u_{v_x}, u_{v_y}, u_{v_z}, u_{v_\psi}]^T$ e são responsáveis por atuar nos eixos x, y, z, e ψ, sendo que cada elemento de v deve ser normalizado nos limites de [-1.0, +1.0]. Os controladores internos são responsáveis por manter a altitude.

O estado que coleta posição e orientação de guinada em relação as coordenadas globais pode ser definido como $q_g = [x_g, y_g, , z_g, \psi_g,]^T$. E, o modelo dinâmico mostrado em (SANTANA et al., 2014) pode ser descrito como mostrado na Equação 5.6.

$$\ddot{q}_g(t) = AR_t^T \dot{q}_g(t) + Bv(t) \tag{5.6}$$

Sendo R_t a matriz de rotação no eixo z e as matrizes A e B definidas na Equação 5.7 e 5.8, respectivamente.

$$A = \begin{bmatrix} -\gamma_2 \cos\psi & \gamma_4 \sin\psi & 0 & 0 \\ -\gamma_2 \sin\psi & -\gamma_4 \cos\psi & 0 & 0 \\ 0 & 0 & -\gamma_6 & 0 \\ 0 & 0 & 0 & -\gamma_8 \end{bmatrix} \tag{5.7}$$

$$B = \begin{bmatrix} \gamma_1 \cos\psi & -\gamma_3 \sin\psi & 0 & 0 \\ \gamma_1 \sin\psi & \gamma_3 \cos\psi & 0 & 0 \\ 0 & 0 & \gamma_5 & 0 \\ 0 & 0 & 0 & \gamma_7 \end{bmatrix} \tag{5.8}$$

5.4 Controle

O controle utilizado no F450 é baseado no MPC (*Model Predictive Control*), também já implementado pelo simulador da MRS, o qual é um conjunto de métodos de controle que engloba o conceito de predição e obtenção do sinal de controle através da minimização de uma determinada função objetivo e considera o erro futuro e as restrições nas variáveis de processo. Este controlador já está implementado pacote da MRS.

Para manter a estabilidade e controlar o *Parrot Bebop* 2 durante o voo e garantir que o UAV obedeça suas restrições foi utilizado a plataforma *drone_dev*, adotando o controle R-LQR (*Robust Linear Quadratic Regulator*) proposto por (BENEVIDES et al., 2019). Este controle consegue estabilizar e executar a trajetória proposta, inclusive em ambientes com vento.

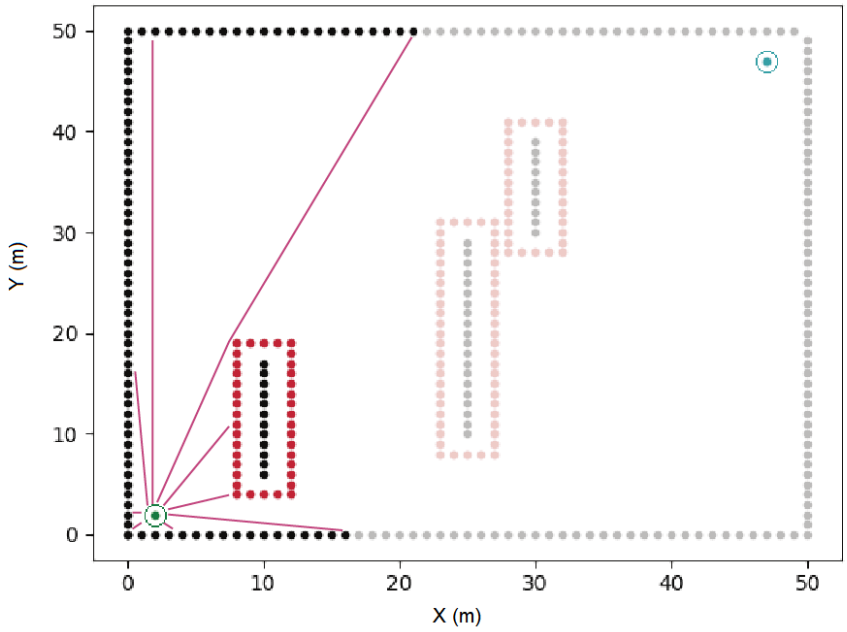
5.5 Ambientes Desconhecidos

Há missões em que o UAV não terá conhecimento do ambiente que irá se mover (CETIN; YILMAZ, 2016; HAYAT et al., 2017). Por isso, é importante que o mesmo consiga identificar os obstáculos que estão ao seu redor e fazer um mapa do local, que será atualizado constantemente.

Este trabalho utiliza técnicas de geometria analítica para identificar o ambiente ao redor do UAV enquanto vai se movimentando, visando simular uma situação real. Por isso, há uma variável com a informação completa do mapa, a qual o planejador não tem acesso direto. A única informação da posição dos obstáculos que o planejador recebe é de quais são os obstáculos em visada direta, a partir da sua posição atual.

Nas simulações realizadas no Python, para descobrir onde há obstáculos visíveis para o UAV no ambiente é checado se há colisão entre o nó atual e cada nó presente na variável com o mapa. Se houver colisão quer dizer que o UAV não está em visada direta com aquele obstáculo, então ele ainda não o verá quando for calcular a trajetória. Caso contrário, o nó será adicionado ao mapa do UAV, como visto na Figura 5.5.

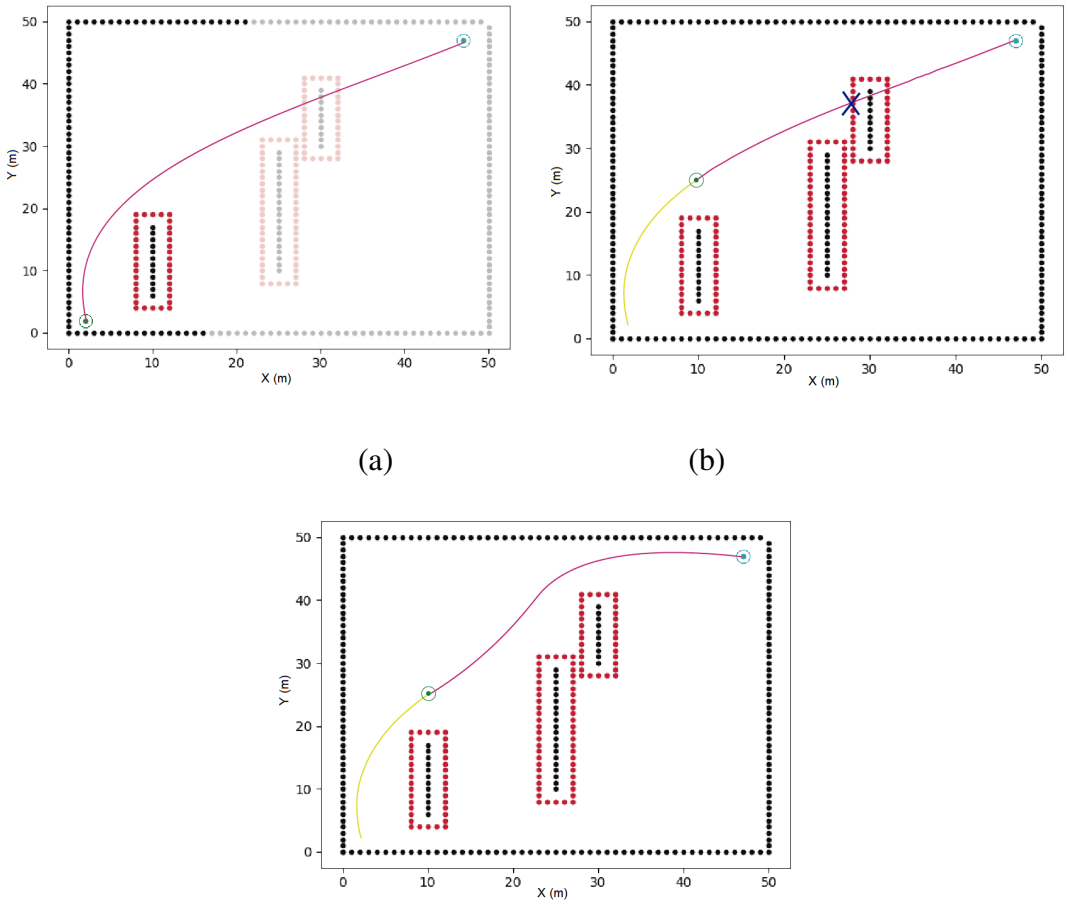
FIGURA 5.5 – Identificando Ambiente Desconhecido.



Fonte: Própria Autora

Quando os obstáculos em que o UAV está vendo são localizados, ou seja, quando o UAV faz o mapeamento do local, é aplicado uma das técnicas de Planejamento de Trajetória apresentadas no Capítulo 4. Desse modo, a trajetória a ser seguida será gerada, considerando apenas os obstáculos conhecidos até o momento, como visto na Figura 5.6 (a), sendo que o ponto verde é o UAV e o ponto ciano é o nó objetivo.

FIGURA 5.6 – Replanejando da Trajetória.



(a) (b)

(c)

Fonte: Própria Autora

A cada segundo a posição do UAV é atualizada em 1 metro de acordo a trajetória definida. A cada passo em que o UAV realiza o mapa vai sendo atualizado de acordo com o novo ambiente visível, ou seja, vai sendo inserido novos obstáculos ao mapa do UAV. Quando a trajetória atual colidir com um obstáculo e estiver a pelo menos 15 metros de distância do UAV, como visto na Figura 5.6 (b), será feito o replanejamento.

Nas simulações feitas no Gazebo, está sendo utilizado o Velodyne, visando obter dados mais precisos do ambiente, e validar melhor o algoritmo de planejamento de trajetória. O Velodyne emite uma nuvem de pontos, que é transformado em coordenadas (x, y, z) com auxílio da biblioteca *PointCloud2* do ROS, descrito na Seção 5.2.

Em ambiente real, está sendo utilizado o *Parrot Bebop* 2, disponível no LASI. Este UAV não é capaz de carregar o Velodyne e manter um voo estável. Por isso, o mapeamento foi realizado através da câmera monocular do *Parrot Bebop* 2 com o OrbSlam2 (NOBIS et al., 2020), o qual

emite uma nuvem de pontos, e utilizar a biblioteca *Statistical Outlier Removal* disponibilizado pela *Point Cloud Library* (PCL), para filtrar estatisticamente os dados mais importantes que estão sendo extraídos, convertendo para coordenadas (x, y, z).

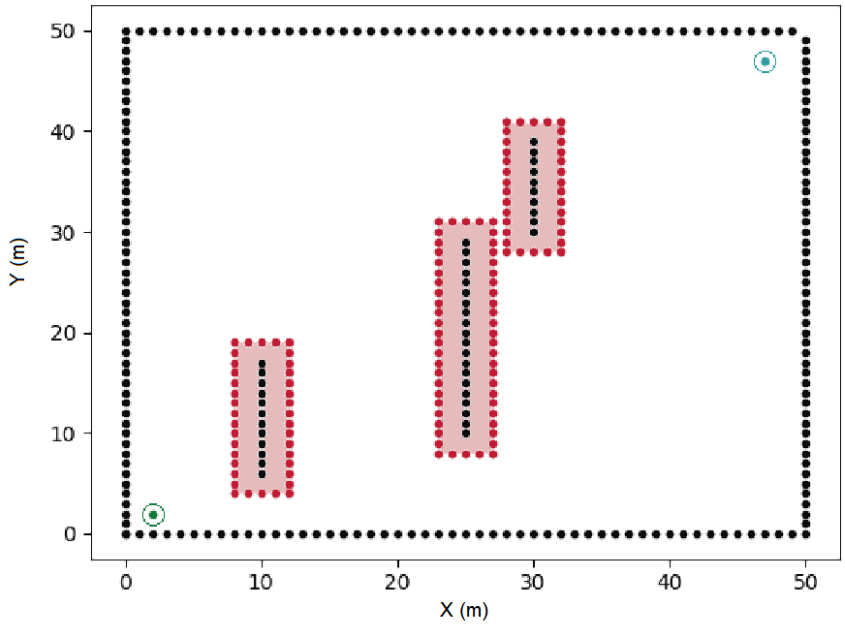
Na Figura 5.6 (c) mostra-se resultado de um replanejamento de trajetória. O replanejamento consiste em chamar o algoritmo que fez a primeira trajetória para fazer uma nova trajetória entre o nó atual e o objetivo, considerando os novos obstáculos. A trajetória atual será válida apenas até o nó atual e depois começará a ser considerado a nova trajetória, e elas serão unidas com a suavização de curva, como mostrado na Seção 5.9.

5.6 Minimização de Riscos

Para minimizar os riscos de colisão, visando não trazer danos ao UAV e ao obstáculo, que pode ser um ser vivo, foi implementado uma zona de risco ao redor dos obstáculos, como visto na Figura 5.7.

O UAV pode adentrar dentro da zona de risco, mas apenas em último caso. Ela é desenvolvida para evitar que o UAV colida com os obstáculos devido alguns fatores de risco. De acordo com Sankararaman e Goebel (2018), os fatores de risco podem ser incertezas e por restrições de desempenho. As incertezas são: clima incerto, obstáculos dinâmicos, sem sinal de GPS, sensores degradados, entre outros. As restrições de desempenho são: gerenciamento de bateria, falhas no sistema do UAV, estabilidade e restrições dinâmicas e aerodinâmicas.

FIGURA 5.7 – Minimizando Riscos.



Fonte: Própria Autora

5.7 Localização

Nas simulações em Python, a localização é feita utilizando a posição atual do ponto da matriz. Essa posição está sempre correta, como se fosse o uso do *ground_truth* de um UAV real.

Nas simulações no Gazebo está sendo usado o pacote *Visual Inertial Odometry* (FORSTER et al., 2016), implementado pelo sistema da MRS, em conjunto com o sensor *range finder*. E em ambiente real está sendo usado o OrbSlam 2 (NOBIS et al., 2020), tendo a frequência entre 10 e 13 Hz, em conjunto com a odometria do próprio UAV para definir a posição atual do UAV. Ambos modos de localização foram escolhidos pois já estavam sendo utilizados previamente em seus respectivos UAVs. Sendo assim, o uso dessas técnicas possui maior confiabilidade ao serem utilizadas novamente nas mesmas configurações.

5.8 Obstáculos Dinâmicos

Durante a trajetória de um UAV é possível que apareça obstáculos dinâmicos pelo caminho, como pessoas ou animais se movimentando. Por isso, foi desenvolvido uma estratégia para

evitar colisão com esses obstáculos em tempo real e mantendo a trajetória sempre suavizada.

O desvio de obstáculo dinâmico será diferente do desvio para obstáculo estático, já que precisa de uma resposta mais rápida. Como o módulo de visão computacional não foi implementado nessa dissertação, os obstáculos dinâmicos são descritos em outra variável. Desse modo, não há como confundir obstáculo dinâmico com obstáculo estático.

Inicialmente, é checado se há colisão entre o nó atual e algum obstáculo dinâmico nos próximos metros da trajetória. Essa distância é definida como 15% do tamanho total do cenário. Se esse obstáculo estiver dentro de um alcance de 10% do tamanho total do cenário, para qualquer direção, as coordenadas do obstáculo serão salvas. Se no próximo movimento do UAV continuar tendo um obstáculo dentro desse alcance de 10% do tamanho total do cenário, será identificado a direção em que o UAV está se locomovendo e o ângulo entre o nó atual e o obstáculo (ângulo 1) e entre o nó atual e o objetivo (ângulo 2).

Se a diferença entre o ângulo 1 e 2 for maior do que 50 graus significa que o obstáculo não está em visada direta entre o nó atual e o objetivo, sendo assim, não é preciso fazer o replanejamento. Caso o obstáculo já tenha passado da trajetória que o UAV está realizando também não será necessário fazer o replanejamento. É possível obter essa informação de acordo com a coordenada do obstáculo dinâmico e com a direção que está se movimentando.

Se a diferença entre o ângulo 1 e 2 for menor do que 50 graus significa que o obstáculo dinâmico irá interferir a trajetória, sendo necessário fazer o replanejamento. O replanejamento mantém a trajetória que o UAV já se locomoveu e a maior parte da trajetória após o obstáculo dinâmico. Para evitar o obstáculo dinâmico é feito um caminho utilizando a curva de *B-Spline* tendo como pontos de controle o nó atual, o nó em que haverá a colisão entre o obstáculo dinâmico e a trajetória (afastado em 2 metros para a direção oposta do movimento do obstáculo dinâmico), e o nó dentro da trajetória inicial que esteja mais próximo do eixo X do ponto de colisão, se o obstáculo se movimentar na vertical, ou do eixo Y, se o obstáculo se movimentar na horizontal.

O caminho gerado irá substituir os nós equivalentes na trajetória anterior. Os caminhos serão ligados através da curva de *B-Spline*, para o UAV fazer um movimento suave quando for trocar de caminho. O pseudo código para evitar obstáculos dinâmicos pode ser visto no Algoritmo 13.

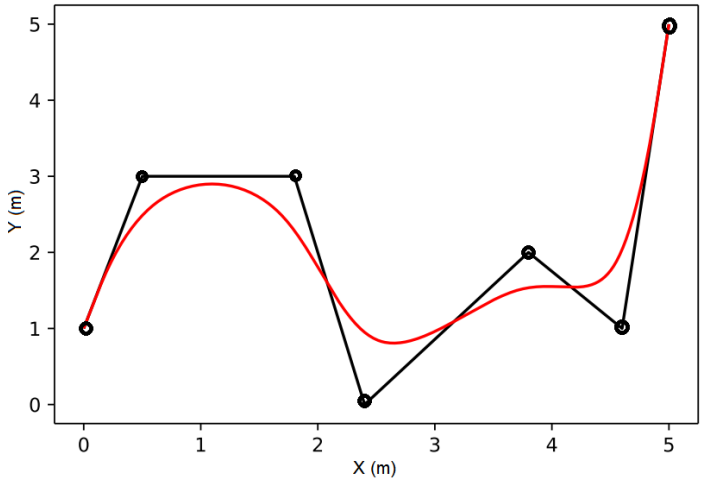
Algoritmo 13: Desvio de Obstáculos Dinâmicos

```
1  início
2      if houver colisão entre o nó atual e o nó que esteja a frente na trajetória a uma
         distância de 15% do tamanho total do cenário then
3          if Checar se o obstáculo está a pelo menos 10% do tamanho total do cenário
             de distância then
4              Flag = True
5          end
6      end
7      while Flag do
8          if Checar se ainda tem obstáculo dinâmico próximo then
9              Identificar a direção em que o obstáculo está se movendo
10             Definir o ângulo entre o nó atual e o objetivo e entre o nó atual e o obstáculo
11             if Caso o obstáculo não for atrapalhar a trajetória then
12                 Flag = False
13             end
14             else
15                 Definir o nó atual como nó de controle 1
16                 Definir o nó em que haverá colisão com a trajetória (o nó será
                     distanciado 2 metros na direção oposta ao do obstáculo) como nó de
                     controle 2
17                 O nó de controle 3 será o mesmo que o 2
18                 if Obstáculo na vertical then
19                     O nó que esteja a frente do nó atual na trajetória mais próximo do
                         eixo X do ponto de colisão será definido como nó de controle 4
20                 end
21                 else
22                     O nó que esteja a frente do nó atual na trajetória mais próximo do
                         eixo Y do ponto de colisão será definido como nó de controle 4
23                 end
24                 Aplicar a curva de B-Spline aos nós de controle Substituir o caminho
                     gerado na trajetória completa
25             end
26         end
27     end
28 fim
```

5.9 Suavização de Curva

A trajetória ideal para um UAV seguir precisa obedecer as restrições dinâmicas e aerodinâmicas do mesmo, ou seja, as curvas precisam ser o mais suaves possíveis (PANDEY; SHUKLA; TIWARI, 2018). Por isso, foi aplicado *splines* nas curvas das trajetórias geradas de cada algoritmo e para unir os replanejamentos de trajetória com o caminho gerado até o momento. Matematicamente, as curvas de *B-Spline* podem ser desenhadas como uma série de segmentos de linhas que unem os pontos de controle, como na Figura 5.8, que os pontos de controle são os círculos destacados nas vértices da trajetória.

FIGURA 5.8 – Exemplo de *B-Spline*.



Fonte: Própria Autora

Com o Algoritmo 12 os pontos de controle da trajetória será onde ocorre as curvas. Desse modo, é possível aplicar diretamente as curvas de *B-Spline* na trajetória gerada pelas técnicas de Planejamento de Trajetória para obter uma trajetória suavizada.

Para unir a trajetória com os replanejamentos ocorridos, devido obstáculos, é definido como ponto de controle apenas o nó que foi capaz de identificar a colisão pela primeira vez e os nós que ficam a 1.5 e 2 metros, respectivamente, a frente do nó de origem da nova trajetória. Dessa maneira, o UAV passará por uma curva suave para mudar da trajetória original para a nova trajetória gerada pelo replanejamento.

A interpolação por *spline* cúbico é uma técnica de aproximação que consiste em dividir o intervalo de interesse em vários subintervalos e interpolar, da forma mais suave possível, baseados em polinômios cúbicos (MCKINLEY; LEVINE, 1998). Dessa forma, a trajetória

gerada será mais suave, sendo mais fácil para o UAV se locomover.

Com $n+1$ pontos, cada ponto é definido por (x_i, y_i), em que i é a ordem das coordenadas e $a = x_0 < x_1 < ... < x_n = b$. O *spline* $S(x)$ satisfaz as seguintes propriedades (FRITSCH; CARLSON, 1980):

1. Cada subintervalo $[x_{i-1}, x_i], S(x)$ é um intervalo de grau 3, em que $i = 1, ..., n$
2. $S(x)$ é contínuo em $[a, b]$ e tem derivada contínua em $[a, b]$ até o grau 3

Esse método inicia selecionando N nós de interpolação $(x'_1, y'_1), (x'_n, y'_n), ..., (x'_nN, y'_nN)$ entre o ponto inicial e o objetivo, para determinar os intervalos. Os nós do *spline* são usados para obter os m pontos de interpolação $x_1, ..., x_m$ e $y_1, ..., y_m$ que quando conectados formam um caminho contínuo.

5.10 Tomador de Decisões

Inicialmente, o tomador de decisões atualiza a sua posição atual (obtido pelos métodos mostrados na Seção 5.7) e o mapa de acordo com a posição dos obstáculos ao redor (obtido pelos métodos explicados na Seção 5.5 e 5.6).

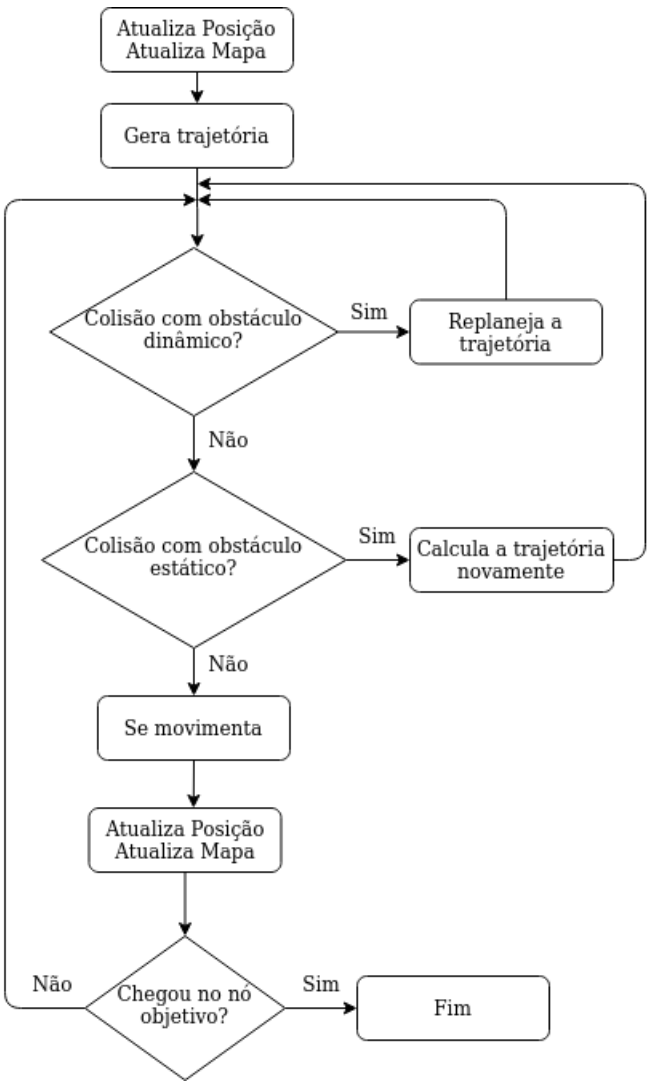
Desse modo, o UAV terá o nó inicial (sua posição), o nó objetivo e o cenário em que irá atuar, parcialmente. Com essas informações, é possível aplicar as técnicas de planejamento de trajetória mostradas no Capítulo 4, as quais retornam uma trajetória suavizadas (como mostrado na Seção 5.9) para o UAV seguir.

Essa trajetória é formada pelas coordenadas dos nós no mapa criado. Dessa forma, o UAV pode sempre enviar novas coordenadas para o controlador (mostrado na Seção 5.4).

Antes de enviar as coordenadas para o controlador a posição atual e o mapa de obstáculo é atualizado. Além de verificar se há colisão entre o UAV e um obstáculo, e em seguida, checar se há colisão entre o UAV e um obstáculo estático recém descoberto, quando o mapa foi atualizado. Caso haja colisão com um obstáculo dinâmico é feito o replanejamento de acordo com o Algoritmo 13 na Seção 5.8. Caso haja colisão com um obstáculo estático, a trajetória é recalculada com os algoritmos do Capítulo 4, como mostrado na Seção 5.5.

O fluxograma do tomador de decisões pode ser visto na Figura 5.9.

FIGURA 5.9 – Fluxograma do Tomador de Decisões.



Fonte: Própria Autora

5.11 Considerações finais

Neste capítulo mostrou-se o funcionamento do planejador e como o mesmo atua em cada ambiente. Bem como, será utilizado em ambiente simulado e real. Também foi descrito como a comunicação é feita entre o planejador e o UAV, em ambiente simulado e real, demonstrando que a arquitetura proposta consegue fazer o planejamento em diversos tipos de UAVs, com controles diferentes.

Capítulo 6

SIMULAÇÕES E ANÁLISES

Neste capítulo apresenta-se como serão realizados os testes, e as simulações de cada etapa. Assim como, o resultado e a análise de cada simulação.

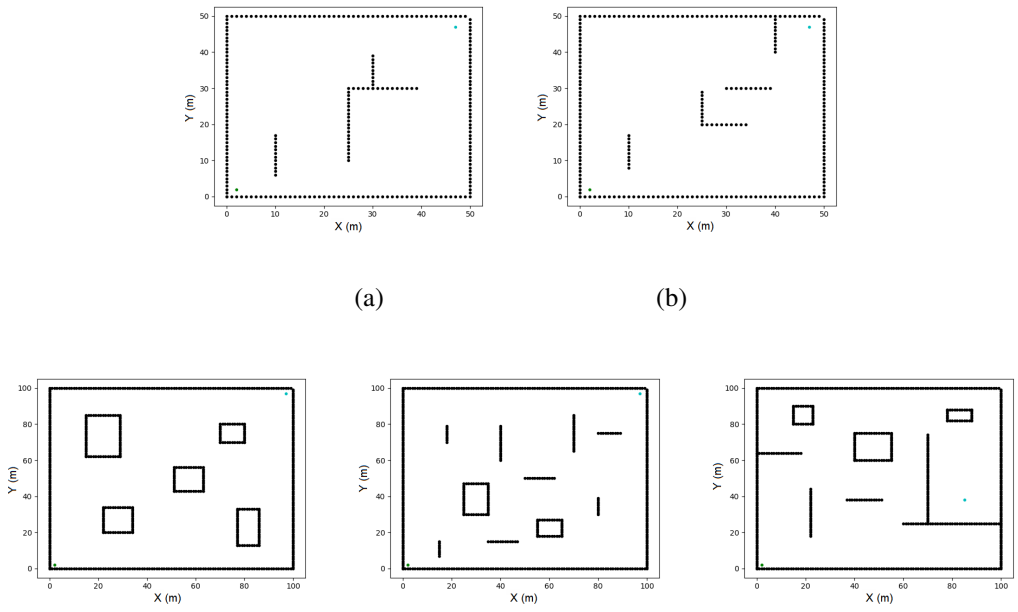
Na primeira etapa, os testes foram realizados avaliando como o planejador se comporta lidando com cada uma dessas características (não estruturado, desconhecido, e obstáculos dinâmicos) separadamente, em Python. Em seguida, ainda em Python, foram feitos cenários incluindo todas essas características para avaliar o tempo de processamento e tamanho da trajetória. Nestes cenários foi avaliado qual o melhor algoritmo a ser usado em cada ambiente e quais são suas principais dificuldades, podendo ser em tamanho do ambiente, número de obstáculos, ou complexidade do ambiente.

Na segunda etapa, foi simulado um cenário no Gazebo, com o objetivo de desenvolver e testar um algoritmo, que inclua o planejador de trajetória, que se comunique com o UAV. Por fim, na terceira etapa, esse cenário foi testado em ambiente real, para validar a técnica proposta. Com esses resultados validou-se o algoritmo de planejamento de trajetória desenvolvido capaz de se locomover em ambientes desconhecidos e não estruturados.

Nas simulações feitas em Python, o ambiente é um grid 2D ou 3D, em que cada obstáculo possui o tamanho de 0,5 metro. Na visualização dos mapas, irá parecer que há um espaço entre as paredes. Isso deve-se ao fato que a visualização do mapa mostra apenas 1 pixel, mas o que o UAV está vendo de fato é um obstáculo de 0,5 metro, que quando está do lado de outro obstáculo de mesma dimensão fecha o espaço entre as paredes.

Todas as etapas estão sendo realizadas em um notebook Samsung Odyssey Intel Core i7 7700HQ com 8 GB de RAM e placa de vídeo NVidia GTX 1050 4 GB. A linguagem adotada foi o Python 3.6.

FIGURA 6.1 – Ambientes. Em (a) Ambiente Pequeno e Simples, (b) Ambiente Pequeno e Não Estruturado, (c) Ambiente Grande e Simples, (d) Ambiente Grande e Não Estruturado - 1, e (e) Ambiente Grande e Não Estruturado - 2.



(a) (b)

(c) (d) (e)

Fonte: Própria Autora

6.1 Primeira Etapa

As simulações foram realizadas em 5 ambientes diferentes visando analisar ao máximo a capacidade de cada técnica.

Houve a análise de um ambiente pequeno e simples, (Figura 6.1 (a)), pequeno e não estruturado (Figura 6.1 (b), grande e simples (Figura 6.1 (c)), grande e não estruturado - 1 (Figura 6.1 (d)), grande e não estruturado - 2 (Figura 6.1 (e)).

O ambiente pequeno possui 50x50 pontos e o ambiente grande possui 100x100. Nesta dissertação cada ponto está sendo tratado como 1 metro, mas pode ser considerado como outros valores, por exemplo, cada ponto equivale a 10 centímetros, então o ambiente pequeno tem 5 metros e o grande 10 metros.

6.1.1 Simulações e Análise da Trajetória 2D

Esta seção apresenta os resultados obtidos utilizando as técnicas propostas em ambientes desconhecidos e não estruturados. A identificação do ambiente foi realizada como descrita na Seção 5.5 a partir da simulação de um sensor laser no Python. Nestas simulações o UAV realiza o mapeamento do ambiente enquanto completa a trajetória, ou seja, o ambiente é desconhecido.

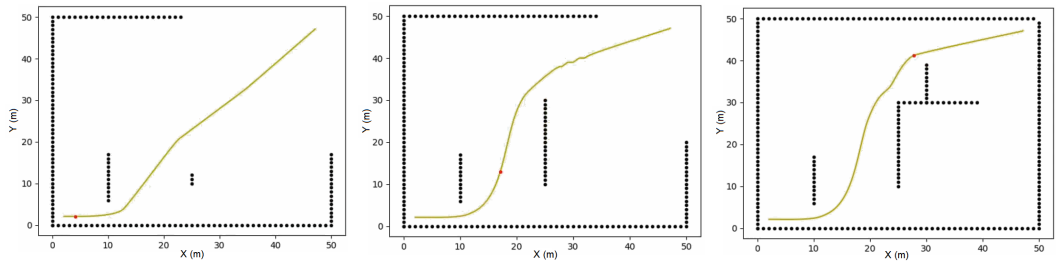
Além disso, realizou-se uma análise entre os algoritmos baseado no tempo para gerar a trajetória e comprimento da mesma, especificando para qual missão, ou situação, cada algoritmo obteve uma melhor performance. Desde a Tabela 6.1 até a Tabela 6.5, as técnicas foram avaliadas em 6 parâmetros, sendo eles:

- **1:** Melhor tempo, em segundos;
- **2:** Pior tempo, em segundos;
- **3:** Média do tempo, em segundos;
- **4:** Variância do tempo, em segundos;
- **5:** Desvio padrão do tempo, em segundos;
- **6:** Distância da trajetória, em metros;

6.1.1.1 Ambiente Pequeno e Simples

Entre a Figura 6.2 e 6.10 mostra-se os resultados de cada um dos algoritmos propostos em um ambiente pequeno e simples, Figura 6.1 (a), sendo desconhecido.

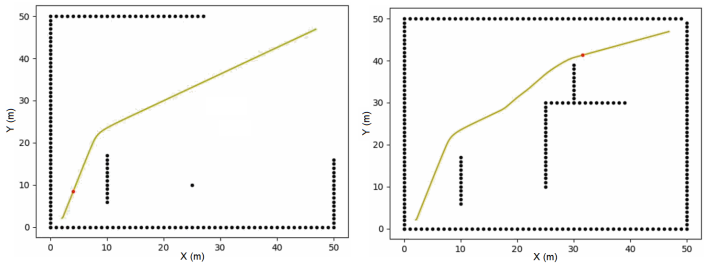
FIGURA 6.2 – A* em Ambiente Pequeno e Simples.



(a) (b) (c)

Fonte: Própria Autora

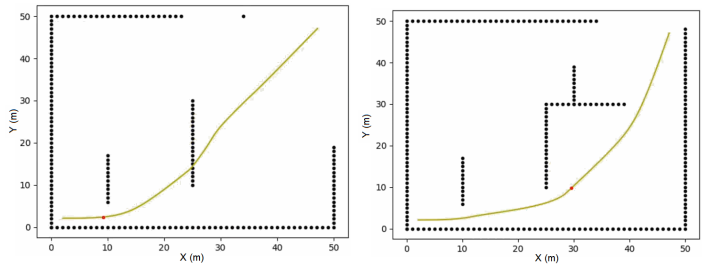
FIGURA 6.3 – APF em Ambiente Pequeno e Simples.



(a) (b)

Fonte: Própria Autora

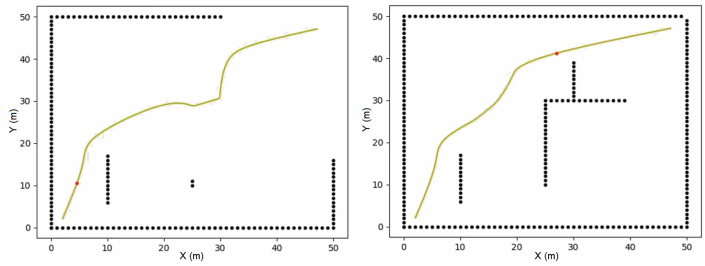
FIGURA 6.4 – PRM em Ambiente Pequeno e Simples.



(a) (b)

Fonte: Própria Autora

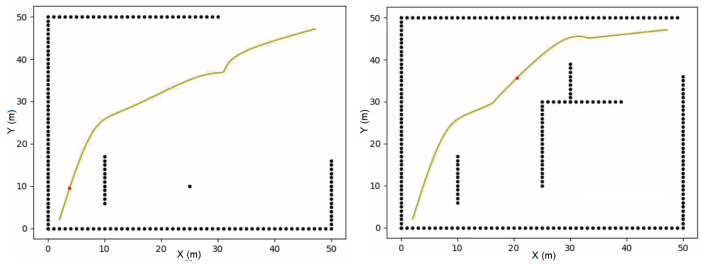
FIGURA 6.5 – RRT em Ambiente Pequeno e Simples.



(a) (b)

Fonte: Própria Autora

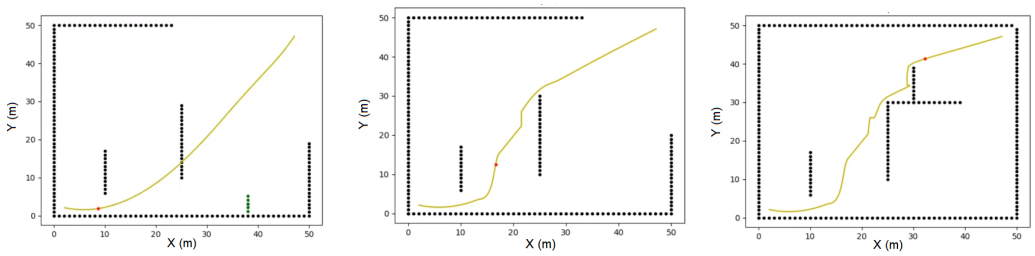
FIGURA 6.6 – RRT-C em Ambiente Pequeno e Simples.



(a) (b)

Fonte: Própria Autora

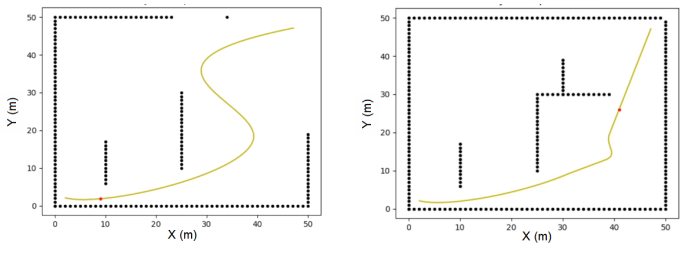
FIGURA 6.7 – PSO em Ambiente Pequeno e Simples.



(a) (b) (c)

Fonte: Própria Autora

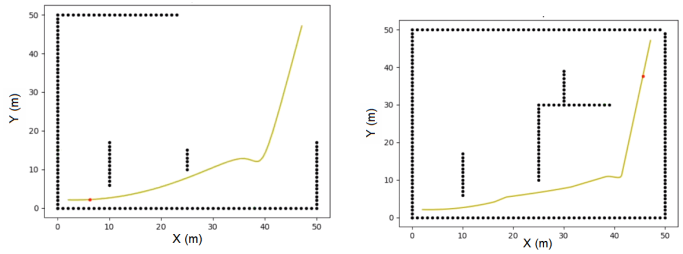
FIGURA 6.8 – GWO em Ambiente Pequeno e Simples.



(a) (b)

Fonte: Própria Autora

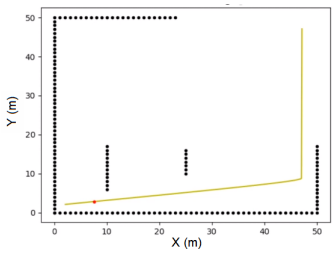
FIGURA 6.9 – GSO em Ambiente Pequeno e Simples.



(a) (b)

Fonte: Própria Autora

FIGURA 6.10 – RL em Ambiente Pequeno e Simples.



Fonte: Própria Autora

Tabela 6.1: Ambiente Pequeno e Simples.

	1 (s)	2 (s)	3 (s)	4 (s)	5 (s)	6 (m)
A*	0,82	1,53	1,18	0,03	0,19	72,37
APF	0,21	1,54	0,66	0,18	0,43	67,61
PRM	1,29	1,49	1,36	0,01	0,11	71,09
RRT	0,11	0,29	0,18	0,009	0,09	69,09
RRT-C	0,06	0,12	0,09	0,002	0,04	70,83
PSO	5,6	9,56	8,02	2,17	1,47	73,77
GWO	5,44	5,44	5,44	0	0	74,41
GSO	5,96	7,77	6,87	1,63	1,28	77,37
RL	4,17	4,17	4,17	0	0	83,77

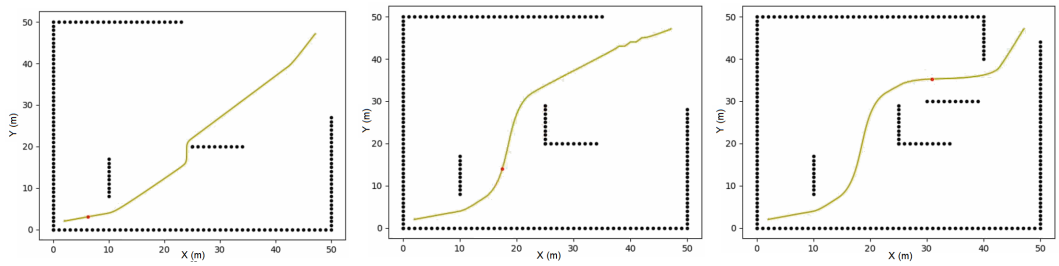
Na Tabela 6.1 podemos ver que a técnica RRT-C obteve melhor resultado em relação ao tempo. Porém, o APF obteve o menor comprimento da trajetória. Analisando as trajetórias realizadas pelas técnicas podemos perceber que tanto o APF quanto o RRT-C retornaram formatos de trajetórias semelhantes e que respeitam as restrições dinâmicas e aerodinâmicas do UAV, devido as poucas curvas no caminho e as que existem estão bastante suavizadas. A trajetória

gerada pelo RL é uma boa opção para o UAV seguir já que possui poucas curvas, minimizando o *Jerk*, permitindo que o UAV aumente a velocidade e chegue mais rápido ao seu objetivo.

6.1.1.2 Ambiente Pequeno e Não Estruturado

Entre a Figura 6.11 e 6.19 mostra-se os resultados de cada um dos algoritmos propostos em um ambiente pequeno e não estruturado, como mostrado na Figura 6.1 (b), sendo desconhecido.

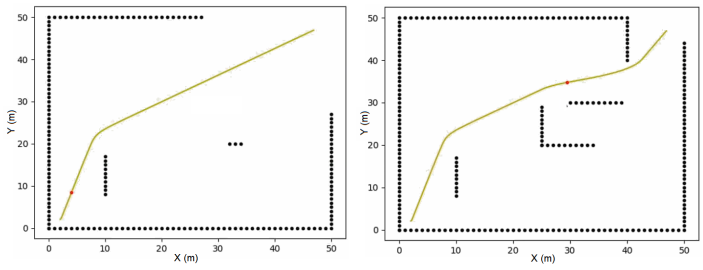
FIGURA 6.11 – A* em Ambiente Pequeno e Não Estruturado.



(a) (b) (c)

Fonte: Própria Autora

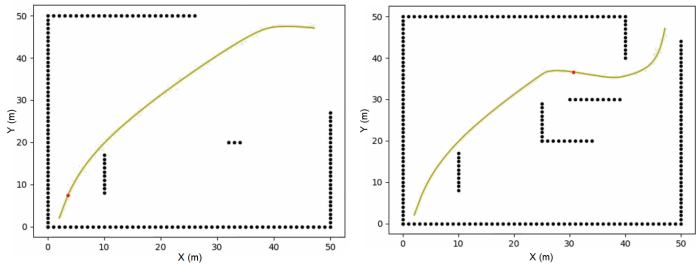
FIGURA 6.12 – APF em Ambiente Pequeno e Não Estruturado.



(a) (b)

Fonte: Própria Autora

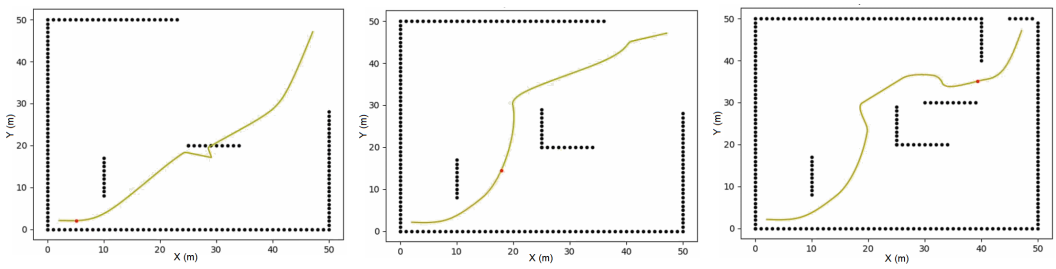
FIGURA 6.13 – PRM em Ambiente Pequeno e Não Estruturado.



(a) (b)

Fonte: Própria Autora

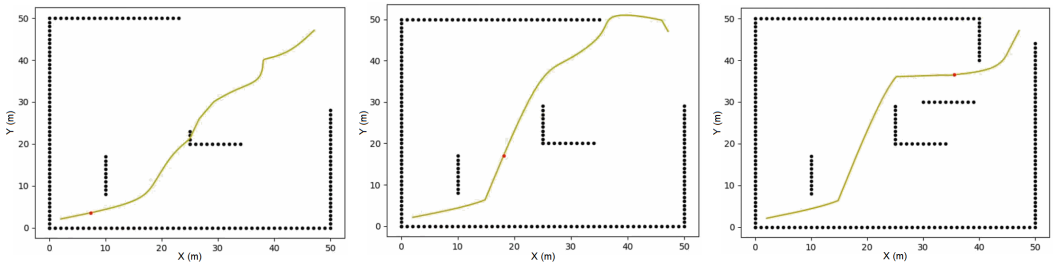
FIGURA 6.14 – RRT em Ambiente Pequeno e Não Estruturado.



(a) (b) (c)

Fonte: Própria Autora

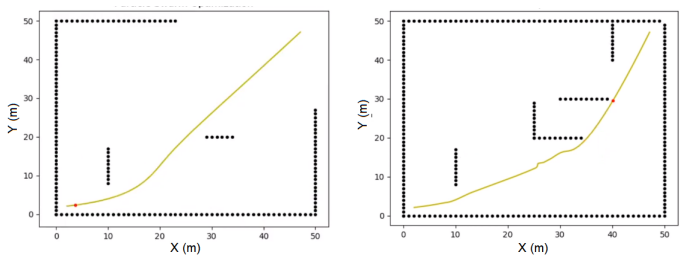
FIGURA 6.15 – RRT-C em Ambiente Pequeno e Não Estruturado.



(a) (b) (c)

Fonte: Própria Autora

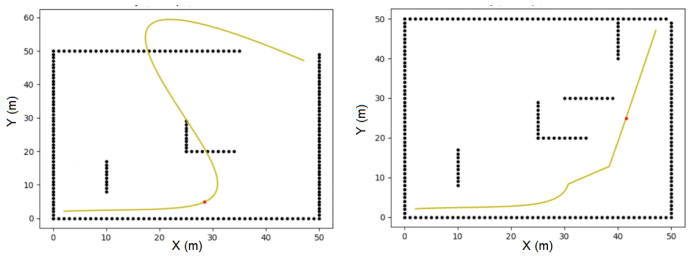
FIGURA 6.16 – PSO em Ambiente Pequeno e Não Estruturado.



(a) (b)

Fonte: Própria Autora

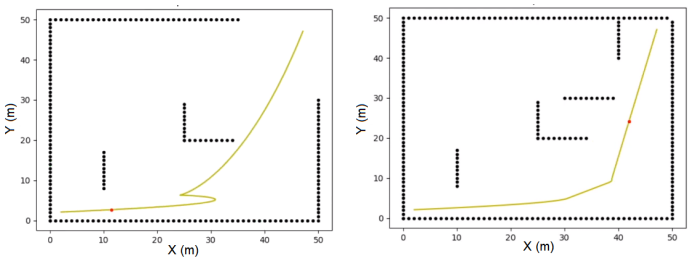
FIGURA 6.17 – GWO em Ambiente Pequeno e Não Estruturado.



(a) (b)

Fonte: Própria Autora

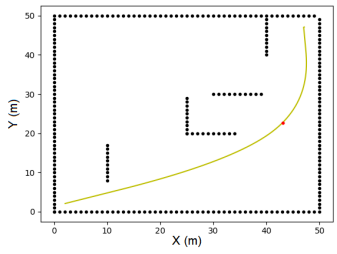
FIGURA 6.18 – GSO em Ambiente Pequeno e Não Estruturado.



(a) (b)

Fonte: Própria Autora

FIGURA 6.19 – RL em Ambiente Pequeno e Não Estruturado.



Fonte: Própria Autora

Tabela 6.2: Ambiente Pequeno e Não Estruturado.

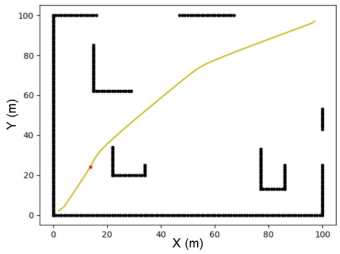
	1 (s)	2 (s)	3 (s)	4 (s)	5 (s)	6 (m)
A*	0,83	1,49	1,26	0,03	0,18	72,15
APF	0,23	0,46	0,28	0,004	0,06	67,74
PRM	1,32	1,49	1,40	0,002	0,05	73,23
RRT	0,09	0,56	0,28	0,02	0,14	77,29
RRT-C	0,08	0,50	0,20	0,01	0,13	75,96
PSO	5,81	8,03	7,17	1,03	1,01	68,32
GWO	5,75	10	7,88	9,04	3	75,23
GSO	6,41	9,8	8,1	5,74	2,39	76,64
RL	2,69	4,28	3,24	0,81	0,9	74,71

Na Tabela 6.2 podemos ver que a técnica RRT-C obteve melhor resultado em relação ao tempo, estando menos que 0,1 segundo mais rápido que o RRT. Porém, o APF obteve o menor comprimento da trajetória estando aproximadamente 0,2 segundos mais lento. Analisando as trajetórias realizadas pelas técnicas podemos perceber que o RRT-C não forneceu uma curva tão suavizada, e o APF se aproximou bastante do obstáculo. Então, considerando menor tempo, menor comprimento da trajetória e caminho mais suave a ser seguido, o A* apresentou melhores resultados. A trajetória gerada pelo GSO e pelo RL apresentaram um caminho com menos curvas e com um bom comprimento, permitindo que o UAV aumente a velocidade e chegue mais rápido ao seu objetivo.

6.1.1.3 Ambiente Grande e Simples

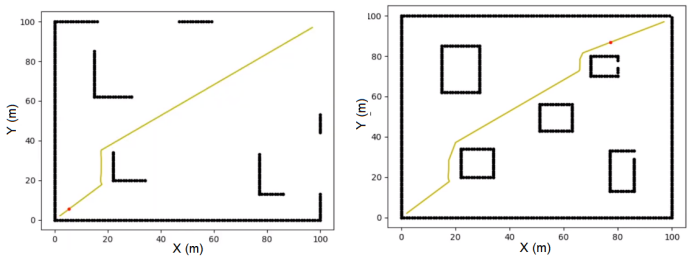
Entre a Figura 6.20 e 6.28 mostra-se os resultados de cada um dos algoritmos propostos em um ambiente grande e simples (Figura 6.1 (c)), sendo desconhecido.

FIGURA 6.20 – A* em Ambiente Grande e Simples.



Fonte: Própria Autora

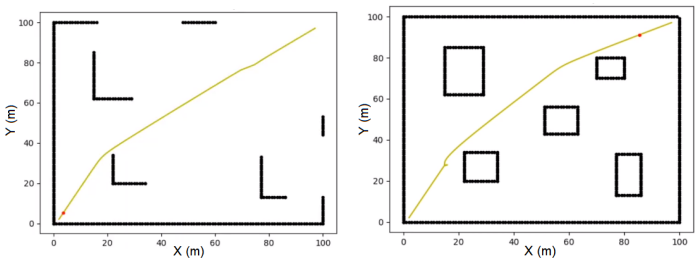
FIGURA 6.21 – APF em Ambiente Grande e Simples.



(a) (b)

Fonte: Própria Autora

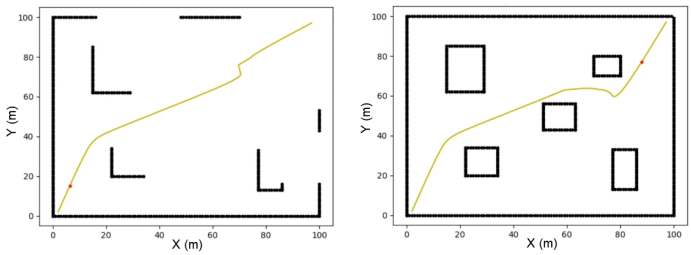
FIGURA 6.22 – PRM em Ambiente Grande e Simples.



(a) (b)

Fonte: Própria Autora

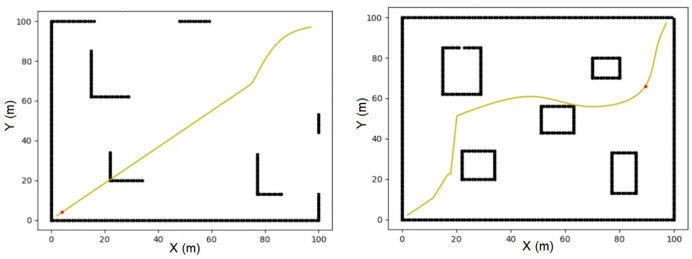
FIGURA 6.23 – RRT em Ambiente Grande e Simples.



(a) (b)

Fonte: Própria Autora

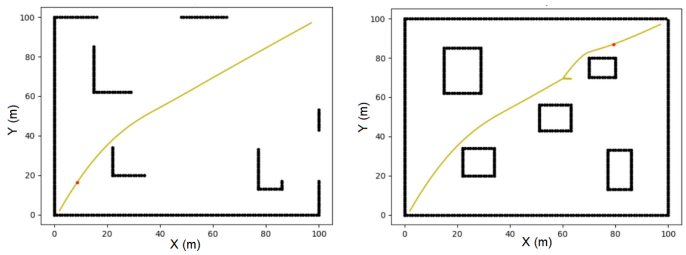
FIGURA 6.24 – RRT-C em Ambiente Grande e Simples.



(a) (b)

Fonte: Própria Autora

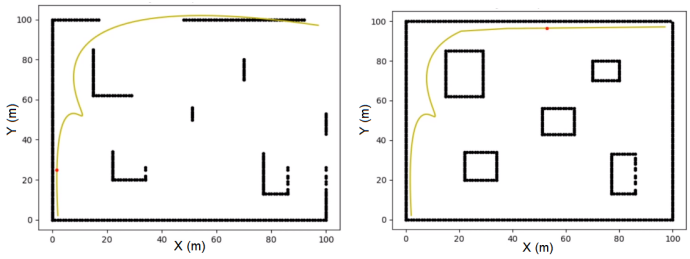
FIGURA 6.25 – PSO em Ambiente Grande e Simples.



(a) (b)

Fonte: Própria Autora

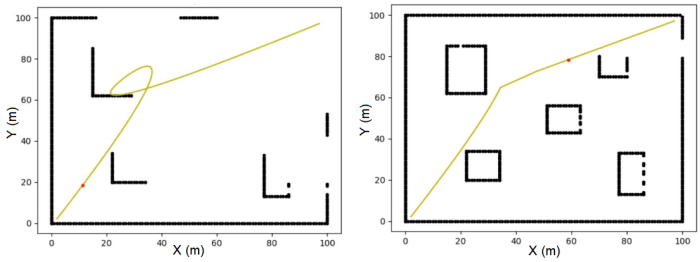
FIGURA 6.26 – GWO em Ambiente Grande e Simples.



(a) (b)

Fonte: Própria Autora

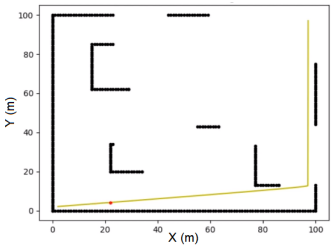
FIGURA 6.27 – GSO em Ambiente Grande e Simples.



(a) (b)

Fonte: Própria Autora

FIGURA 6.28 – RL em Ambiente Grande e Simples.



Fonte: Própria Autora

Na Tabela 6.3 podemos ver que o APF obteve melhor menor tempo, porém o RRT-C obteve melhor pior tempo e melhor média de tempo. Verificando o desvio padrão e variância do tempo de ambas as técnicas percebemos que o RRT-C possui dados mais homogêneos, tornando a

Tabela 6.3: Ambiente Grande e Simples.

	1 (s)	2 (s)	3 (s)	4 (s)	5 (s)	6 (m)
A*	0,57	0,57	0,57	0	0	138,37
APF	0,22	1,5	0,79	0,3	0,55	142,52
PRM	15,8	30	22,92	101,27	10,06	140,24
RRT	2,02	3,45	2,86	0,56	0,75	151,28
RRT-C	0,34	0,88	0,59	0,04	0,2	162,89
PSO	14,98	50,25	32,61	622,12	24,94	143,97
GWO	14,21	53,98	34,09	791,03	28,12	180,76
GSO	15,19	43,15	29,17	391	19,77	141,18
RL	19,89	19,89	19,89	0	0	179,84

técnica mais confiável. A técnica que obteve menor trajetória foi o A*, a qual também obteve a média do tempo melhor do que o APF e o RRT-C.

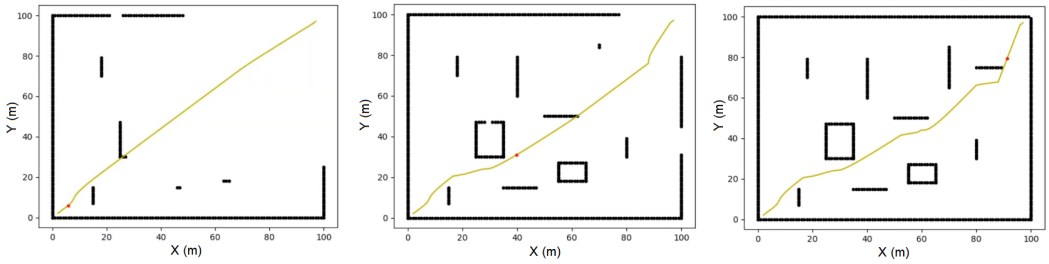
Analisando as trajetórias realizadas pelos algoritmos podemos perceber que o RRT-C retornou uma trajetória muito próxima aos obstáculos. A trajetória gerada pelo APF foi boa, com poucas curvas e sendo todas suavizadas. Porém, a trajetória gerada pelo A* foi melhor ainda pois alcançou o objetivo com uma grande curva aberta.

A trajetória gerada pelo PSO e pelo GSO apresentaram um caminho com menos curvas e com um bom comprimento, permitindo que o UAV aumente a velocidade e chegue mais rápido ao seu objetivo. Sendo que a trajetória fornecida pelo PSO além de ter bastantes retas, as curvas são bastante suavizadas. Apesar da técnica oferecer uma trajetória com um comprimento aceitável, ela demora demais para retornar a resposta.

6.1.1.4 Ambiente Grande e Não Estruturado - 1

Entre as Figuras 6.29 e 6.37 mostra-se os resultados de cada um dos algoritmos propostos em um ambiente grande e não estruturado (Figura 6.1 (d)), sendo desconhecido.

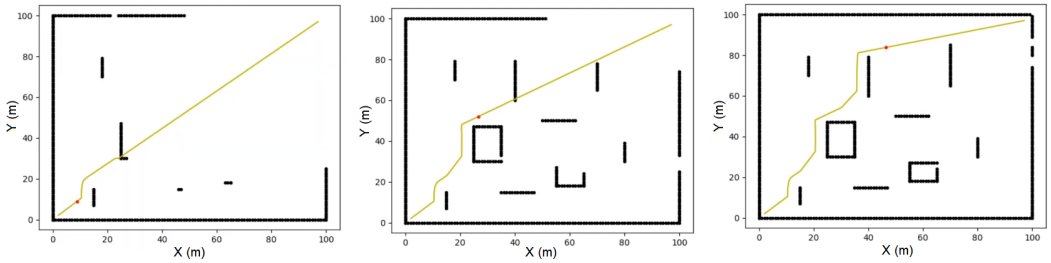
FIGURA 6.29 – A* em Ambiente Grande e Não Estruturado - 1.



(a) (b) (c)

Fonte: Própria Autora

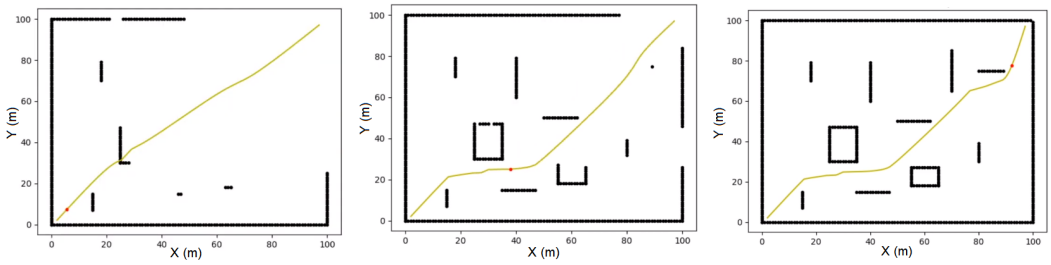
FIGURA 6.30 – APF em Ambiente Grande e Não Estruturado - 1.



(a) (b) (c)

Fonte: Própria Autora

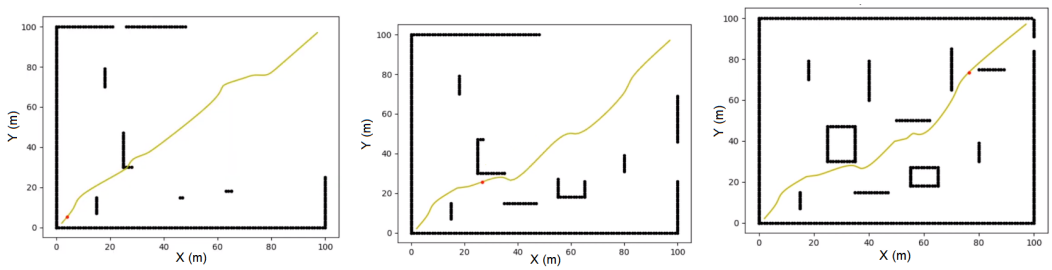
FIGURA 6.31 – PRM em Ambiente Grande e Não Estruturado - 1.



(a) (b) (c)

Fonte: Própria Autora

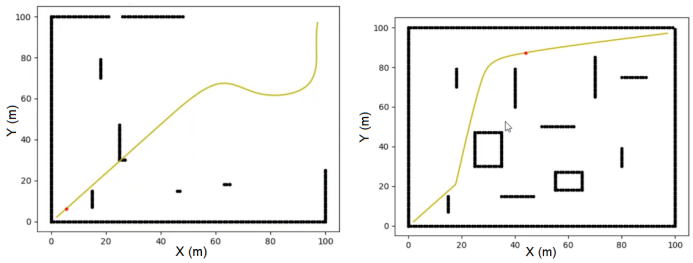
FIGURA 6.32 – RRT em Ambiente Grande e Não Estruturado - 1.



(a) (b) (c)

Fonte: Própria Autora

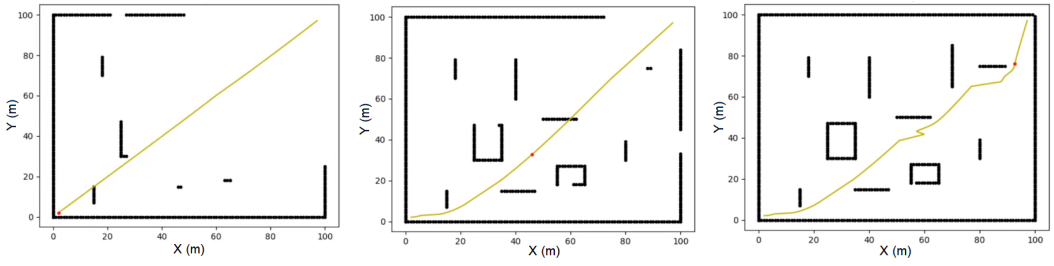
FIGURA 6.33 – RRT-C em Ambiente Grande e Não Estruturado - 1.



(a) (b)

Fonte: Própria Autora

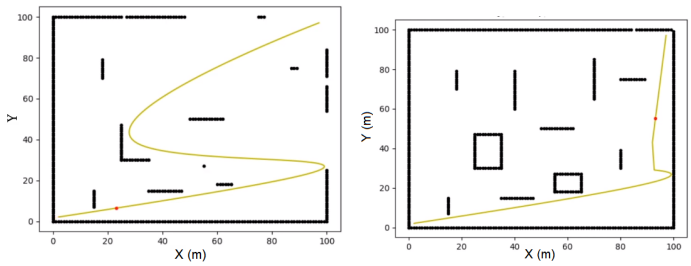
FIGURA 6.34 – PSO em Ambiente Grande e Não Estruturado - 1.



(a) (b) (c)

Fonte: Própria Autora

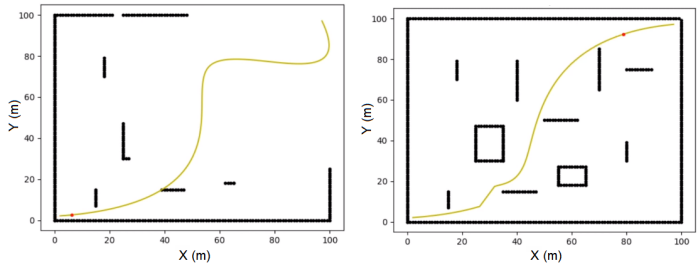
FIGURA 6.35 – GWO em Ambiente Grande e Não Estruturado - 1.



(a) (b)

Fonte: Própria Autora

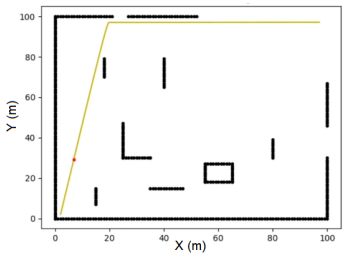
FIGURA 6.36 – GSO em Ambiente Grande e Não Estruturado - 1.



(a) (b)

Fonte: Própria Autora

FIGURA 6.37 – RL em Ambiente Grande e Não Estruturado - 1.



Fonte: Própria Autora

Na Tabela 6.4 podemos ver que o RRT obteve melhor menor tempo, o APF obteve melhor pior tempo e o A* melhor média de tempo. Verificando o desvio padrão e variância do tempo das técnicas percebemos que o A* possui dados mais homogêneos, tornando a técnica mais

Tabela 6.4: Ambiente Grande e Não Estruturado - 1.

	1 (s)	2 (s)	3 (s)	4 (s)	5 (s)	6 (m)
A*	0,48	1,38	1,03	0,08	0,29	143,65
APF	0,7	1,25	2,1	0,35	0,59	155,35
PRM	2,88	3,34	3,07	0,04	0,21	145,32
RRT	0,46	2,3	1,35	0,85	0,92	143,48
RRT-C	0,68	2,35	1,52	1,39	1,18	155,28
PSO	12,29	43,74	21,05	172,23	13,12	148,53
GWO	12,26	35,94	24,1	280,36	16,74	176,03
GSO	12,3	18,74	15,52	20,77	4,55	148,43
RL	18,92	18,92	18,92	0	0	173,87

confiável. O RRT apresentou o menor caminho, sendo apenas 0,17 metros menor do que a trajetória fornecida pelo A*.

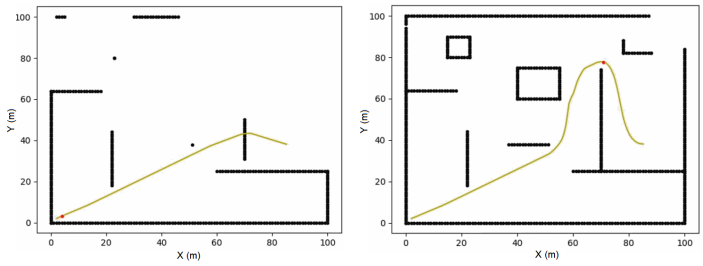
Analisando as trajetórias realizadas pelos algoritmos podemos perceber que o APF e o A* apresentaram melhor resultado, já que retornaram menos curvas e mais suavizadas. Entre ambas as técnicas, o A* obteve uma distância maior até os obstáculos, gerando uma trajetória mais segura.

A trajetória gerada pelo GSO e PRM apresentaram caminhos com curvas mais suaves, facilitando a locomoção do UAV. A trajetória do GSO foi uma curva aberta em que o UAV pode manter uma velocidade constante sem muitas modificações na rotação. E o PRM gerou uma trajetória apenas com curvas essenciais e bastante suavizadas.

6.1.1.5 Ambiente Grande e Não Estruturado - 2

Entre a Figura 6.38 e 6.43 mostra-se os resultados de cada um dos algoritmos propostos em um ambiente grande e não estruturado (Figura 6.1 (e)), sendo desconhecido. Os algoritmos meta-heurísticos (PSO, GWO, e GSO) não conseguiram concluir o percurso, por isso não há resultados deles.

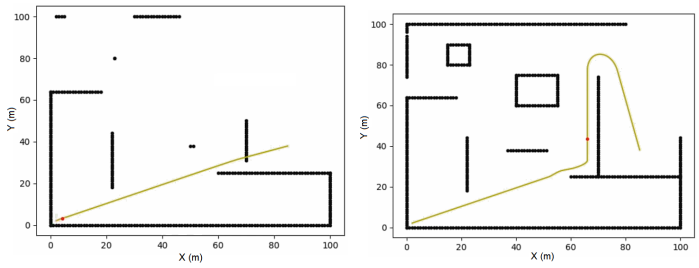
FIGURA 6.38 – A* em Ambiente Grande e Não Estruturado - 2.



(a) (b)

Fonte: Própria Autora

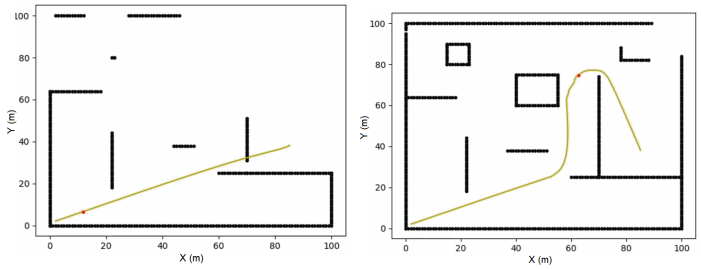
FIGURA 6.39 – APF em Ambiente Grande e Não Estruturado - 2.



(a) (b)

Fonte: Própria Autora

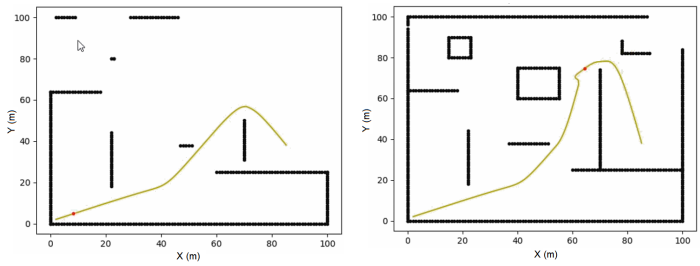
FIGURA 6.40 – PRM em Ambiente Grande e Não Estruturado - 2.



(a) (b)

Fonte: Própria Autora

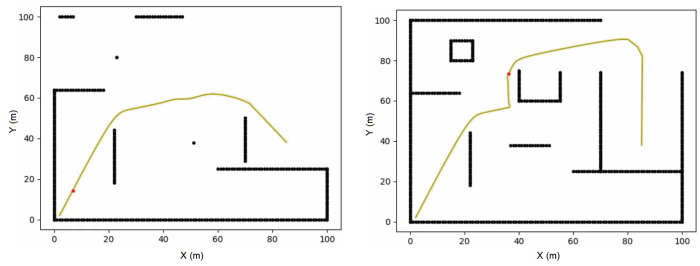
FIGURA 6.41 – RRT em Ambiente Grande e Não Estruturado - 2.



(a) (b)

Fonte: Própria Autora

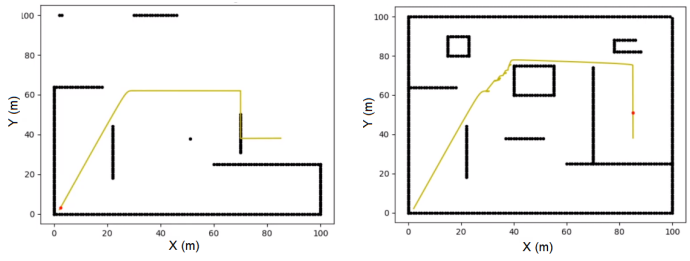
FIGURA 6.42 – RRT-C em Ambiente Grande e Não Estruturado - 2.



(a) (b)

Fonte: Própria Autora

FIGURA 6.43 – RL em Ambiente Grande e Não Estruturado - 2.



(a) (b)

Fonte: Própria Autora

Na Tabela 6.5 podemos ver que o RRT-C obteve melhor menor e maior tempo, e o PRM conseguiu a melhor média de tempo. Verificando o desvio padrão e variância do tempo de

Tabela 6.5: Ambiente Grande e Não Estruturado - 2.

	1 (s)	2 (s)	3 (s)	4 (s)	5 (s)	6 (m)
A*	9,56	21,44	17,92	0,63	0,79	153,86
APF	1,87	4,20	2,56	0,5	0,7	176,72
PRM	2,05	3,59	2,40	0,1	0,32	163,90
RRT	0,79	7,03	2,97	2,46	1,57	156,25
RRT-C	0,6	4,12	2,72	2,52	1,58	176,94
PSO	∞	∞	∞	∞	∞	∞
GWO	∞	∞	∞	∞	∞	∞
GSO	∞	∞	∞	∞	∞	∞
RL	16.34	24.47	20.24	3.87	1.96	182.30

ambas as técnicas percebemos que o PRM possui dados mais homogêneos, tornando a técnica mais confiável. O A* apresentou o menor caminho, porém, o tempo para obtê-lo foi muito acima de um tempo aceitável. Analisando as trajetórias realizadas pelos algoritmos podemos perceber que o A*, APF e PRM apresentaram rotas suavizadas, com poucas curvas e bem semelhantes entre si.

O RRT apresentou uma trajetória com tamanho aceitável em um bom tempo. No entanto, seu pior tempo, assim como a variância e desvio padrão, estão demasiadamente elevado, não sendo uma técnica confiável para uso em tempo real. O PRM se mostrou ser a melhor técnica para esse ambiente, pois apesar de apresentar uma trajetória 10 metros do que a menor, o seu tempo de resposta é mais rápido e possui dados homogêneos, tornando-o mais confiável.

Neste cenário todas as técnicas, com exceção do RRT-C e do RL que obteviram uma trajetória demasiadamente próxima aos obstáculos, apresentaram caminhos suaves com pouco *Jerk*, sendo possível acelerar a velocidade do UAV em certos momentos da trajetória, visando chegar mais rápido ao objetivo.

6.1.1.6 Análise da Trajetória

Para fazer essa análise foram escolhidos 5 ambientes distintos, mostrado anteriormente na Figura 6.1, variando em tamanho, número de obstáculos e em complexidade, ou seja, quanto menos estruturado mais complexo. O cenário mostrado na Figura 6.1 (a) é o menor cenário e que possui menos obstáculos. Já o cenário na Figura 6.1 (b) é pequeno mas não estruturado, visando analisar qual a consequência de um cenário não estruturado para cada técnica.

A Figura 6.1 (c) mostra um cenário grande e relativamente simples, visando entender como a complexidade entre as técnicas é afetada em relação ao tamanho do ambiente. A Figura 6.1

(d) mostra um cenário grande com muitos obstáculos, com a intenção de entender o impacto que o número de obstáculos oferece às técnicas. Por fim, na Figura 6.1 (e) tem-se um cenário não estruturado, utilizado para identificar as principais dificuldades em cada técnica e identificar quanto tempo computacional será gasto para resolver tais dificuldades.

Podemos observar que, entre as Seção 6.1.1.1 e 6.1.1.5 apresentam-se qual técnica obteve a menor trajetória e em um menor tempo em tais ambientes. Em todos os casos, a melhor técnica para ambos os casos foram as técnicas clássicas. Porém, as técnicas meta-heurísticas e de aprendizado máquina demostraram um caminho melhor para o UAV percorrer, principalmente, se for necessário velocidade na missão, pois em diversos resultados foram retornadas trajetórias demasiadamente retas, possibilitando o aumento da velocidade. Como apresentado na Figura 6.20, em que a trajetória foi melhor do que a demonstrada pela Figura 6.21, porém, nesta trajetória é possível acelerar o UAV.

Após esta análise podemos observar como o ambiente afeta o desempenho dos algoritmos. Todos as técnicas são afetadas pela complexidade do ambiente, mas as que mais sofrem alteração com isso é o A* e as técnicas meta-heurísticas. A complexidade do APF aumentou mais com o número de obstáculos. Isso é notável averiguando a Figura 6.1 (a) e (b), que apesar da Figura 6.1 (b) ser um ambiente não estruturado, teve um tempo de resposta mais rápido do que a Figura 6.1 (a), já que havia menos obstáculos entre o nó origem e objetivo.

O PRM, RRT, PSO, GWO, GSO e RL, tiveram mais impacto relacionado à dimensão do ambiente. O tempo cresce exponencialmente junto com o tamanho do ambiente, e aumenta cada vez mais de acordo com sua complexidade. Já o RRT-C sofre influência por número de obstáculos, dimensão do ambiente e complexidade. Pode-se reparar isso, pois seu tempo de resposta aumentou a cada ambiente em que foi testado. Apesar da técnica sofrer influência por todos os fatores, ainda foi a técnica que apresentou um dos menores tempo de resposta. Apesar disso, vale ressaltar, que o PRM foi a técnica que apresentou menor variância e desvio padrão, mostrando ter resultados mais estáveis para voos em tempo real, que requeira certa confiança.

Considerando um ambiente desconhecido, em que não se tenha ideia de como será percurso, ou se for um ambiente em que o UAV terá pouco espaço para se locomover, como ambientes *indoor*, a melhor escolha será utilizar o A*. Pois esta técnica consegue retornar as trajetória rapidamente, para os replanejamentos do percurso, e possui baixa variância e desvio padrão se mostrando confiável. Além de sempre apresentar uma das menores trajetórias.

Caso seja alguma missão em que o UAV precise completar rapidamente e se tenha o conhecimento de que haverá poucos obstáculos no percurso, como campos, técnicas de aprendizado de máquina ou meta-heurísticas são a melhor opção pois a trajetória possui várias retas, possi-

bilitando o aumento da velocidade. Mas, caso haja muitos obstáculos pode-se optar pelo uso do APF, pois esta técnica também retorna trajetórias com longos espaços retos e tem um tempo de replanejamento mais rápido.

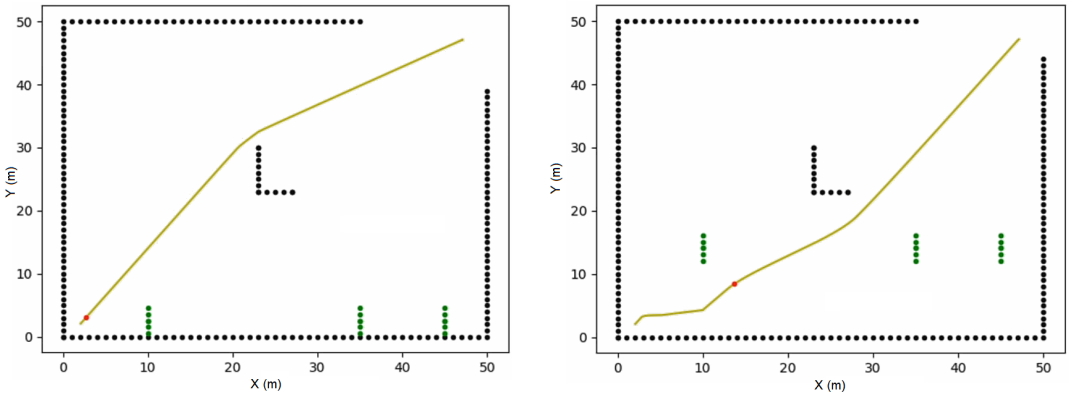
Se a missão for algo que precise explorar dentro de florestas ou montanhas, é melhor o uso do PRM, RRT ou RRT-C. Pois esses ambientes com certeza terão muitos obstáculos pequenos e perto um do outro. Essas técnicas, por serem baseadas em amostragem, conseguem investigar bem todos esses espaços e retornar a melhor trajetória, além de sempre retornarem curvas pequenas muito bem suavizadas. Se for necessário mais estabilidade no algoritmo o PRM será melhor, mas se precisar de mais velocidade será o RRT-C, e o RRT é o meio termo entre eles.

Pensando em um cenário urbano, completamente não estruturado, como para realizar *delivery*, as técnicas A* ou PRM são as mais indicadas. Pois a segurança será a prioridade e essas duas técnicas foram as que retornaram as trajetórias com menor desvio padrão e variância. Sendo assim, essas técnicas são as mais confiáveis para desviar de um humano caso haja algum imprevisto. O A* geralmente retorna as melhores trajetórias, porém, seu tempo de resposta aumenta exponencialmente em ambientes não estruturados. Então, se for uma rota mais simples vale a pena utilizar o A*, caso contrário, o PRM será a melhor opção.

6.1.2 Desvio de Obstáculo Dinâmico

O desvio de obstáculo dinâmico foi realizado conforme descrito no Algoritmo 13. Nas Figuras de 6.44 até 6.47 mostra-se como ocorre o desvio caso o obstáculo se locomova de baixo para cima, de cima para baixo, da direita para esquerda, e da esquerda para direita, respectivamente.

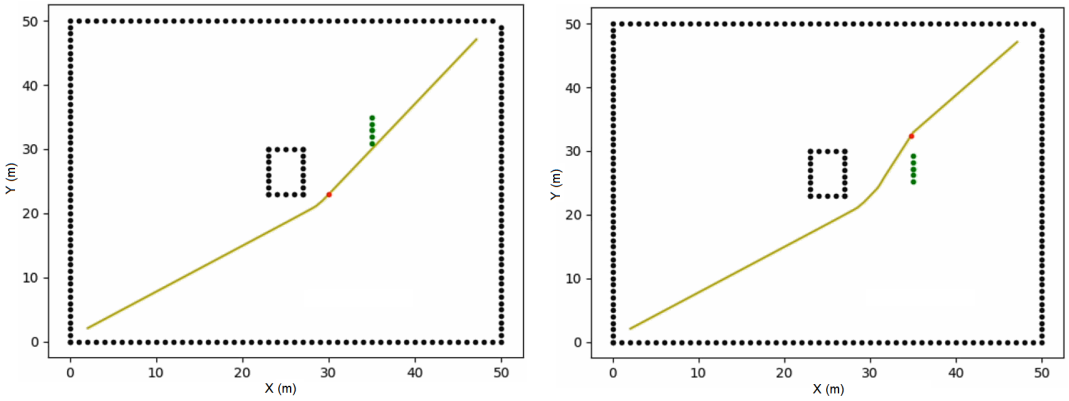
FIGURA 6.44 – Desvio de Obstáculo Dinâmico de Baixo para Cima.



(a) (b)

Fonte: Própria Autora

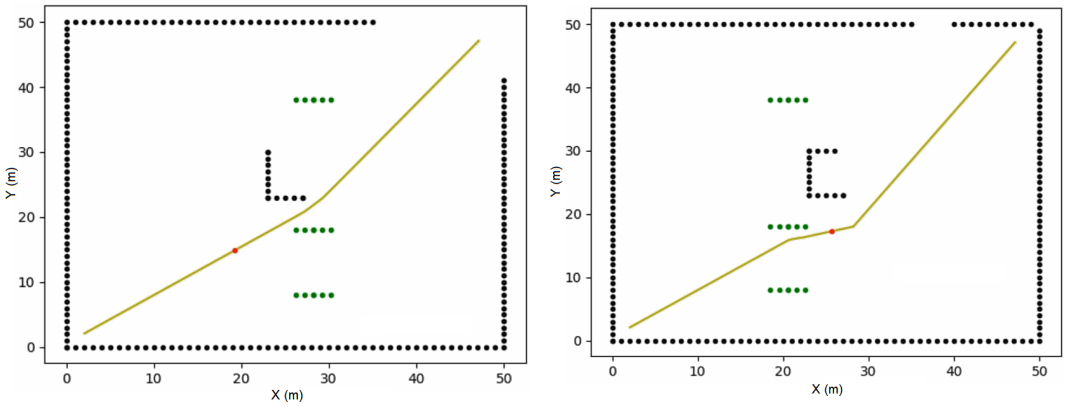
FIGURA 6.45 – Desvio de Obstáculo Dinâmico de Cima para Baixo.



(a) (b)

Fonte: Própria Autora

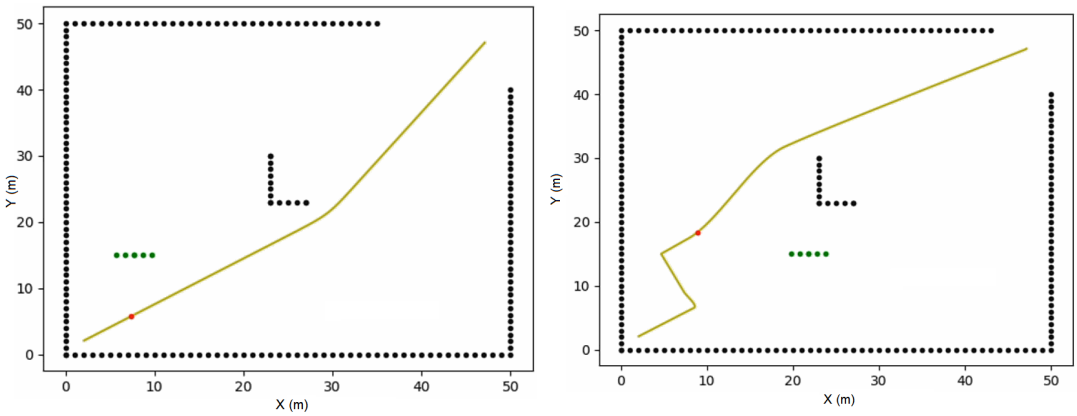
FIGURA 6.46 – Desvio de Obstáculo Dinâmico da Direita para Esquerda.



(a) (b)

Fonte: Própria Autora

FIGURA 6.47 – Desvio de Obstáculo Dinâmico da Esquerda para Direita.



(a) (b)

Fonte: Própria Autora

Em todos os movimentos em que os obstáculos dinâmicos realizaram o algoritmo foi capaz de gerar uma trajetória suavizada para evitar a colisão. Além disso, também conseguiu detectar quando um obstáculo já tivesse passado ou não estivesse em seu caminho até o nó objetivo.

6.2 Segunda Etapa

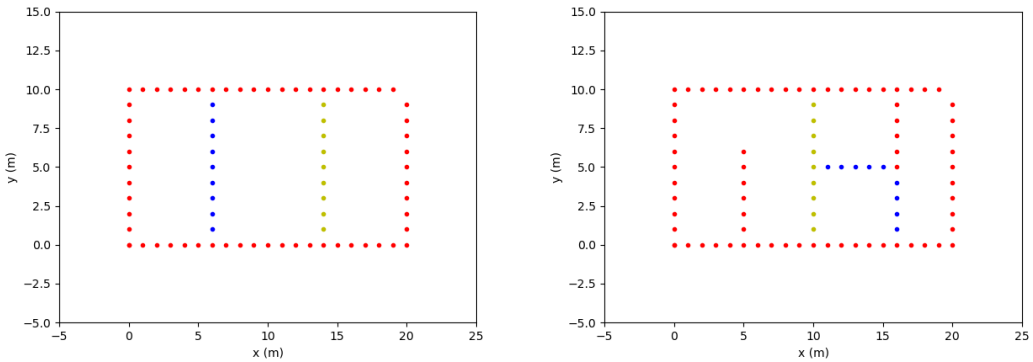
Os cenários escolhidos para fazer os testes em ambiente simulado foram *indoor* de 20x10 metros.Esse cenário foi escolhido pois é possível validar o algoritmo considerando um ambi-

ente 3D, não estruturado, e desconhecido. Por ser um ambiente com tamanho relativamente pequeno, será possível validar o quanto de segurança, em relação a distância até os obstáculos, o planejador pode oferecer.

Nessa etapa, foi utilizado dois cenários. O primeiro é um ambiente simples em que só é possível chegar ao objetivo se o UAV desviar dos obstáculos estáticos indo para cima e para baixo, visando perceber as principais dificuldades dos algoritmos ao interagir com ambientes 3D. O segundo é um ambiente não estruturado em 3D, visando obter dados significativos sobre o planejador de trajetórias.

Nos cenários, há espaços em que o UAV não pode passar (círculos vermelhos), em que só pode passar por baixo (círculos amarelos) e que só pode passar por cima (círculos azuis). Desse modo, foi possível analisar o desempenho do planejamento em 3D. Os cenários são mostrados nas Figuras 6.48 e 6.49.Na Figura 6.48 é mostrado a visualização em Python, em que pode-se observar onde o UAV pode passar por cima, baixo ou não pode passar. Na Figura 6.49 é mostrada a visualização superior do ambiente no Gazebo. Em ambos os ambientes as coordenadas iniciais são (2, 4, 1) e finais (19, 9, 3).

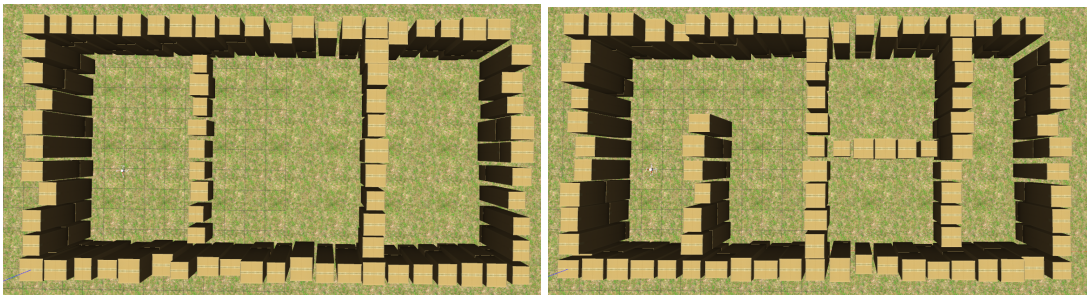
FIGURA 6.48 – Ambiente Simulado - Python. (a) Ambiente simples (b) Ambiente Não estruturado.



(a) (b)

Fonte: Própria Autora

FIGURA 6.49 – Ambiente Simulado - Gazebo. (a) Ambiente simples (b) Ambiente Não estruturado.



(a) (b)

Fonte: Própria Autora

Antes de fazer as simulações 3D no Gazebo, foi testado no Python, para verificar a performance dos algoritmos. Os testes em Python foram realizados com o ambiente já conhecido, dessa forma, é possível validar os algoritmos sem considerar o erro dos sensores. Cada simulação em Python foi rodada 100 vezes.

Em seguida, as simulações foram feitas no Gazebo, para emular uma missão similar a de um ambiente real. Nessas simulações foram considerados os algoritmos do planejador de trajetória (A*, RRT-C, PSO, RL), de localização (VIO), de mapeamento (Velodyne), e de controle (MPC). Cada algoritmo rodou 1 vez no Gazebo.

O A*, RRT-C, PSO, e RL, foram os algoritmos escolhidos para esta etapa pois apresentaram melhores resultados na primeira etapa de cada sub grupo, considerando os mais diversos ambientes.

Os resultados são apresentados em tabelas. As tabelas referentes às simulações em Python seguem a seguinte legenda:

- T_m **(s):** Melhor tempo do algoritmo, em segundos;
- T_p **(s):** Pior tempo do algoritmo, em segundos;
- T **(s):** Média de tempo do algoritmo, em segundos;
- D_p **(m):** Menor trajetória, em metros;
- D_m **(m):** Maior trajetória, em metros;
- D **(m):** Média do comprimento das trajetória, em metros.

E as referentes às simulações no Gazebo:

- T_v **(s):** Tempo de voo, sem segundos;
- T **(s):** Média do tempo do algoritmo e desvio padrão, sem segundos;
- D **(m):** Comprimento da trajetória, em metros;
- CPU **(MHz):** CPU, em megahertz;
- M **(MB):** Memória, em megabyte;
- B **(%):** Bateria do UAV.

O algoritmo utilizado para medir a bateria no simulador considera apenas o tempo de voo e o uso do motor.

6.2.1 Ambiente Simples

Esta subseção apresenta as simulações em Python e no Gazebo em 3D para o ambiente simples.

6.2.1.1 Simulação em Python

Nas Tabelas 6.6 e 6.7 são mostradas as métricas referentes às simulações 3D feitas em Python para o ambiente simples. Nas tabelas é possível perceber que a taxa de prevenção de colisão (P_c) de todos os algoritmos é de 100%, com exceção do PSO, que foi de 80%. Além disso, todas as técnicas obtiveram o *completeness* de 100%.

Tabela 6.6: Ambiente Simples em Python - 3D (1).

	CPU (MHz)	Memória (MB)	P_c (%)	*Completeness* (%)
A*	823,34±169,18	6,36±0,8	100	100
RRT-C	449,2±269,52	6,42±0,79	100	100
PSO	527,88±344,44	6,28±0,73	80	100
RL	444,27±339,99	6,29±0,73	100	100

A análise de custo computacional é feita baseada na biblioteca psutil [1] do Python. Esta biblioteca possui a função de recuperar informações sobre processos em execução e utilização do sistema, como: CPU, memória, discos, rede, sensores, entre outros.

[1] https://psutil.readthedocs.io/en/latest/

Na Tabela 6.6 também é apresentado a análise de custo computacional dos algoritmos. A técnica de aprendizado por reforço apresentou a menor média de processamento, apesar de ter um dos maiores desvios padrão, seguido pelo RRT-C. O A* obteve o maior custo computacional, pois esta técnica analisa todo o ambiente antes de retornar uma resposta, sendo mais custoso do que os demais. O uso da memória foi similar para todos os algoritmos, assim como o desvio padrão da mesma.

Tabela 6.7: Ambiente Simples em Python - 3D (2).

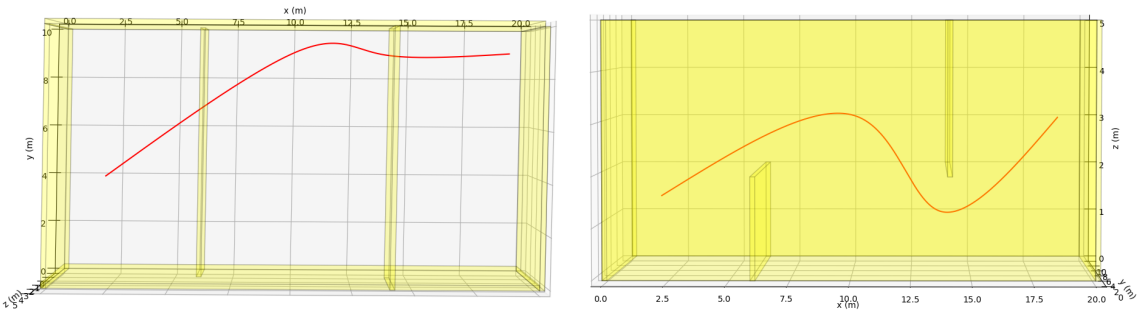
	T_m (s)	T_p (s)	T_s (s)	D_p (m)	D_m (m)	D (m)
A*	0.54	0.72	0.57±0.03	21.76	21.76	21.76±0
RRT-C	0.13	0.76	0.3±0.13	18.62	35.15	21.39±3.04
PSO	17.4	20.71	18±0.34	18.59	19.58	18.92±0.21
RL	10.51	15.6	12.65±0.97	19.35	31.22	22.06±2.07

Na Tabela 6.7 é possível analisar o tempo para gerar a trajetória e seu comprimento. O RRT-C apresentou a menor média do tempo para gerar a trajetória e a segunda menor média entre os comprimentos das trajetórias, apesar de indicar o maior desvio padrão para a trajetória. O PSO apresentou a menor média do comprimento da trajetória.

O A* obteve resultados próximos ao do RRT-C, apresentando um desvio padrão menor em ambas as métricas, mostrando ser mais confiável para voos em ambientes real. Analisando unicamente o tamanho das trajetórias, todos os algoritmos obtiveram bons resultados. Porém, o PSO e RL levaram mais tempo para gerar uma trajetória.

Da Figura 6.50 até 6.53 é apresentado as trajetórias geradas pelos algoritmos A*, RRT-C, PSO, e RL, respectivamente, no ambiente simples.

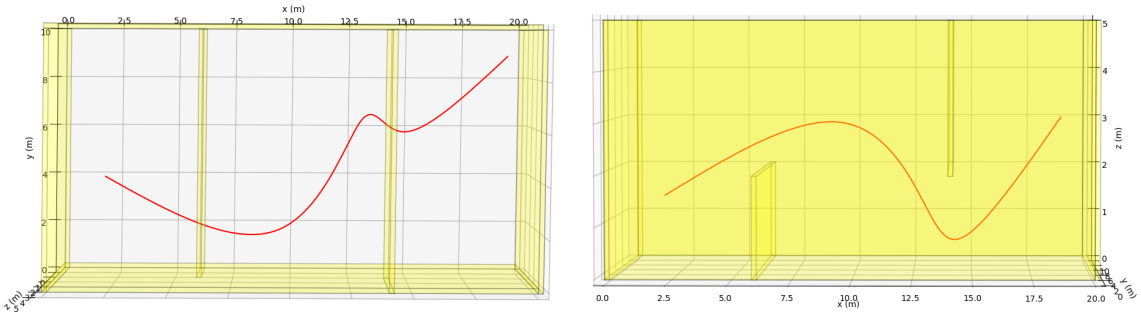
FIGURA 6.50 – A* em Ambiente 3D Simples - Python. Em (a) vista superior e (b) visão lateral.



(a) (b)

Fonte: Própria Autora

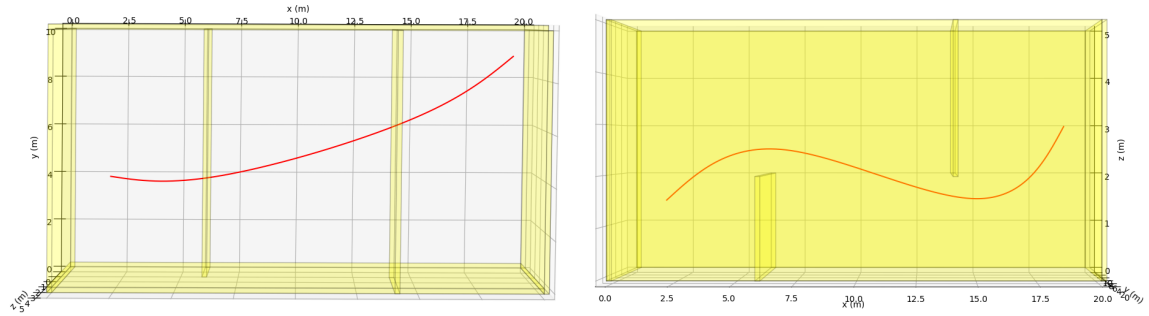
FIGURA 6.51 – RRT-C em Ambiente 3D Simples - Python. Em (a) vista superior e (b) visão lateral.



(a) (b)

Fonte: Própria Autora

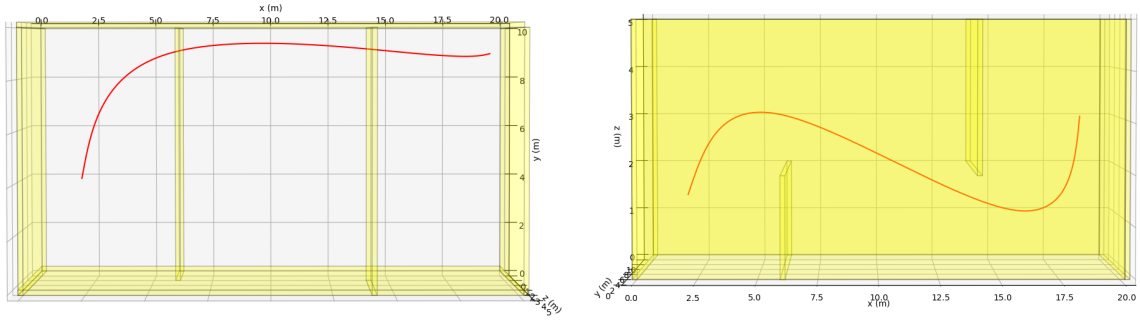
FIGURA 6.52 – PSO em Ambiente 3D Simples - Python. Em (a) vista superior e (b) visão lateral.



(a) (b)

Fonte: Própria Autora

FIGURA 6.53 – RL em Ambiente 3D Simples - Python. Em (a) vista superior e (b) visão lateral.



(a) (b)

Fonte: Própria Autora

Neste ambiente, o desafio era gerar um percurso em que o trajeto seja executado por baixo e por cima, dado o ambiente. Todos os algoritmos retornaram trajetórias boas para serem seguidas pelo UAV, fazendo curvas apenas quando necessário. Com exceção, do RRT-C, que por ser baseado em nós aleatórios a trajetória resultando é sinuosa.

6.2.1.2 Simulação no Gazebo

Na Tabela 6.8 é apresentado as métricas obtidas pela simulação no Gazebo no ambiente simples. Nessa tabela, todos os desvios padrões são 0 pois não houve replanejamento. O resultado das simulações do Gazebo devem variar bastante para as de Python, pois em Python o planejador conhece todo o ambiente previamente, e no Gazebo o ambiente é descoberto de acordo com a movimentação do UAV.

O cálculo da bateria foi feito utilizando um plugin de bateria do Gazebo [2]. Está sendo considerado a tensão de 14,8 V, e a corrente mácima de 60000 mAH, sendo que o UAV precisa de 1000 mAH sem utilizar os motores. A corrente necessária com os motores é de 75000 mAH.

Tabela 6.8: Ambiente Simples no Gazebo - 3D.

	T_v (s)	T (s)	D (m)	CPU (MHz)	M (MB)	B (%)
A*	101.6	1.17±0	23.1	30.8	3.69	1
RRT-C	112.22	1.25±0	20.93	70	4.7	2
PSO	118.64	11.05±0	18.28	36.4	3.94	2
RL	189.36	18.7±0	21.76	131.59	14.09	4

[2]https://github.com/robotizandoufsc/battery_simulator

O gasto da bateria depende unicamente do tempo de voo e da corrente necessária para o motor funcionar, sendo igual para todos os algoritmos. Por isso, o uso da bateria é proporcional ao tempo de voo.

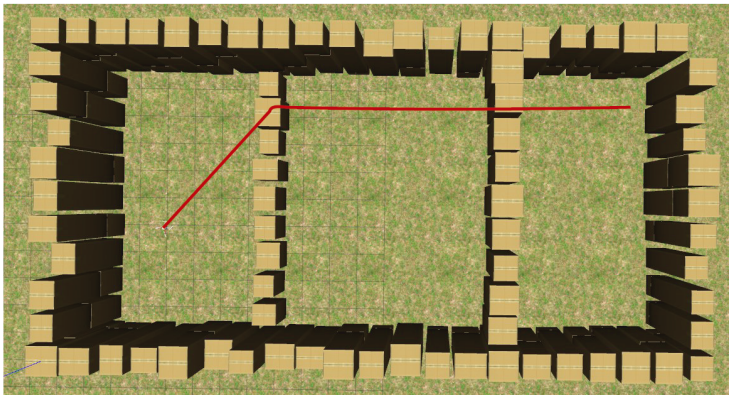
Como a velocidade para todos os algoritmos é constante, o tempo de voo também é proporcional ao tempo do planejador de trajetória. O A* foi quem teve o menor tempo de voo e gasto de bateria, já que planejou a trajetória em menos tempo. Os resultados do comprimento de trajetória foram semelhantes aos obtidos pelas simulações em Python, apesar de que no Gazebo o cenário não é conhecido, ou seja, o PSO retornou a menor trajetória seguido pelo RRT-C.

O que teve uma grande variação dos resultados em Python foi o custo computacional e o tempo para retornar uma trajetória. Em Python, o PSO era o algoritmo que mais demorava seguido do RL. No Gazebo a situação se inverte, mostrando que o PSO demora mais tempo para retornar uma trajetória de acordo com a quantidade de obstáculos no ambiente, e o RL mantém a mesma média de tempo independente da quantidade de obstáculos.

Em Python, o custo computacional do A* era superior aos demais, mas no Gazebo foi o menor, mostrando também ser dependente ao número de obstáculos no ambiente. O custo computacional de todos os algoritmos diminuiu, com exceção do RL, mostrando que o processamento necessário para executá-lo independe do cenário. Assim como ocorreu com o uso de memória, que diminuiu em todos os algoritmos, exceto o RL.

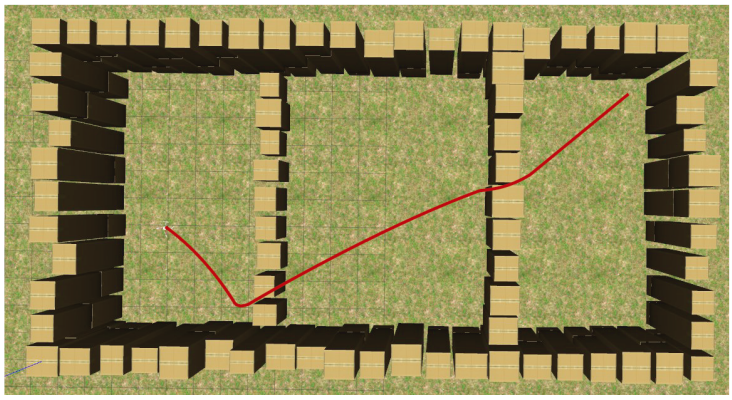
Da Figura 6.54 até 6.57 é mostrada as trajetórias retornadas, no Gazebo, pelos algoritmos A*, RRT-C, PSO, e RL, respectivamente, no ambiente simples.

FIGURA 6.54 – A* em Ambiente 3D Simples - Gazebo.



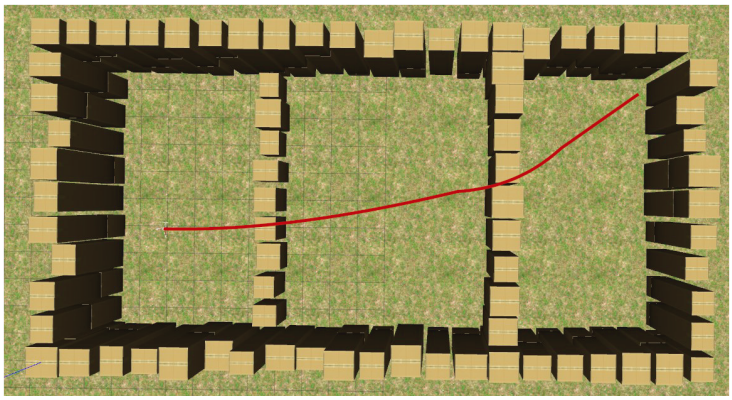
Fonte: Própria Autora

FIGURA 6.55 – RRT-C em Ambiente 3D Simples - Gazebo.



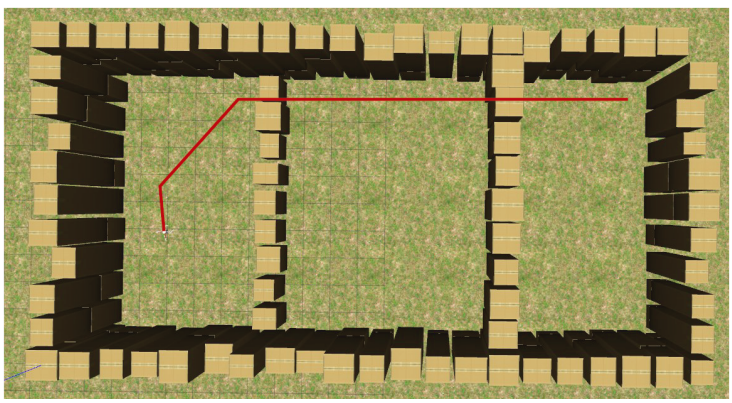
Fonte: Própria Autora

FIGURA 6.56 – PSO em Ambiente 3D Simples - Gazebo.



Fonte: Própria Autora

FIGURA 6.57 – RL em Ambiente 3D Simples - Gazebo.



Fonte: Própria Autora

O A* e o PSO retornaram boas trajetórias para serem seguidas pelo UAV, tendo o A* priorizado minimizar o *Jerk* e o torque constante. O PSO realizou uma trajetória com curva aberta, suavizando-a. O RRT-C também retornou uma boa trajetória a ser seguida, porém os seus movimentos iniciais vão para o lado oposto do objetivo, sendo contraprodutivo em uma análise geral. Essas trajetórias podem acontecer com o RRT-C por ser baseado em aleatoriedade. A trajetória gerada pelo RL apesar de manter o torque do UAV constante durante boa parte da trajetória, realiza uma curva desnecessária no início.

6.2.2 Ambiente Não Estruturado

Nesta subseção apresenta as simulações em Python e no Gazebo em 3D para o ambiente não estruturado.

6.2.2.1 Simulação em Python

Nas Tabelas 6.9 e 6.10 são mostradas as métricas referentes às simulações 3D feitas em Python para o ambiente não estruturado. Nas tabelas é possível perceber que o *completeness* de todos os algoritmos foi de 100%, com exceção do PSO, que foi de 16%. Como o PSO gera uma trajetória inicial e a otimiza até ficar sem colisão, é possível obter um alto *completeness* e uma baixa taxa de prevenção de colisão. Devido esses motivos, pode-se concluir que o PSO é ineficaz para realizar voos em missões reais pois a probabilidade de gerar uma trajetória com colisão é pequena, mesmo considerando os replanejamentos.

Tabela 6.9: Ambiente Não Estruturado em Python - 3D (1).

	CPU (MHz)	Memória (MB)	P_c (%)	*Completeness* (%)
A*	969.5±251.59	6.32±0.71	100	100
RRT-C	549.64±554.55	6.53±0.85	100	100
PSO	422.91±291.58	6.3±0.73	16	100
RL	625.4±487.29	6.32±0.77	100	100

Na Tabela 6.9 também é apresentado o quanto de CPU e memória cada algoritmo utilizou para gerar a trajetória. O PSO apresentou a menor média de processamento, apesar de ter um dos maiores desvios padrão, seguido pelo RRT-C. O A* obteve o maior custo computacional, como mostrado pela análise feita em ambiente simples. O uso da memória foi simular para todos os algoritmos, assim como o desvio padrão da mesma.

Na Tabela 6.10 é possível analisar o tempo para gerar a trajetória e seu comprimento. O RRT-C apresentou a menor média do tempo para gerar a trajetória e desvio padrão. A técnica de

Tabela 6.10: Ambiente Não Estruturado em Python - 3D (2).

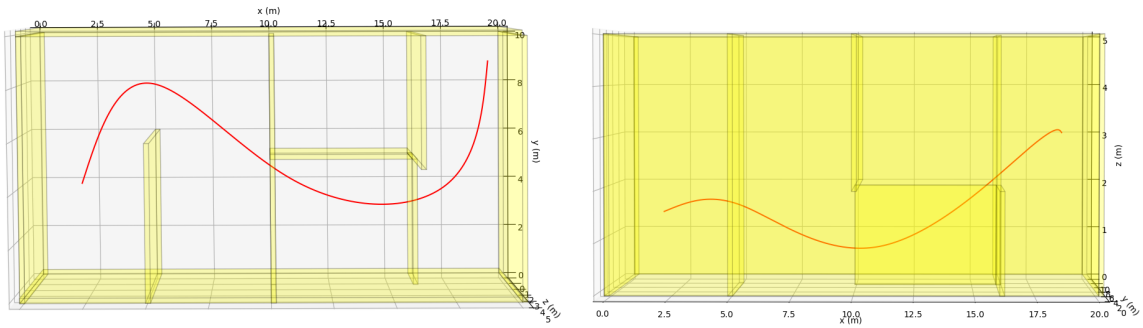
	T_m (s)	T_p (s)	T_s (s)	D_p (m)	D_m (m)	D (m)
A*	0,44	0,72	0,51±0,04	30,43	30,43	30,43±0
RRT-C	0,16	0,86	0,43±0,02	26,65	42,25	31,69±3,84
PSO	20,02	28,61	21,87±3,02	21,45	32,82	28,91±0,24
RL	11,72	18,88	13,54±1,27	25,69	43,84	30,26±3,21

aprendizado por reforço e o A* tiveram a média do comprimento de trajetória similares, ficando atrás apenas do PSO, sendo que o A* apresenta o menor desvio padrão de todas as técnicas já que sua trajetória é determinística.

Analisando unicamente o tamanho das trajetórias, todos os algoritmos obtiveram bons resultados. E apesar do PSO e RL terem retornado a menor média entre os algoritmos, eles demoram muito para gerar uma trajetória.

Das Figuras 6.58 até 6.61 são apresentadas as trajetórias geradas pelos algoritmos A*, RRT-C, PSO, e RL, respectivamente, no ambiente não estruturado.

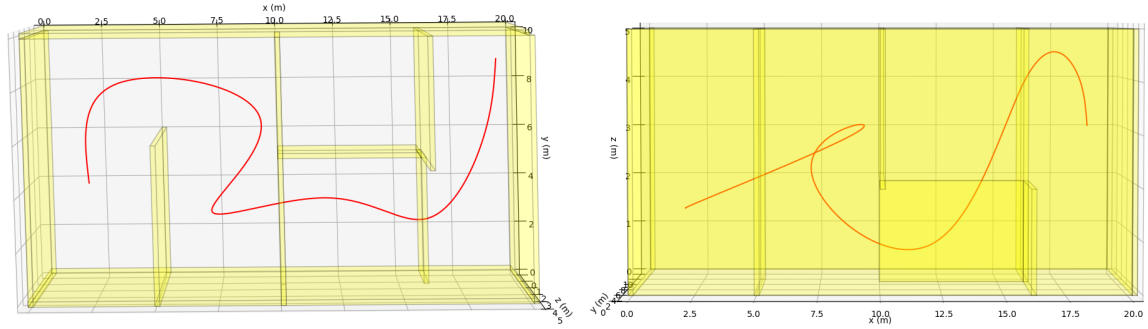
FIGURA 6.58 – A* em Ambiente 3D Não Estruturado - Python. Em (a) vista superior e (b) visão lateral.



(a) (b)

Fonte: Própria Autora

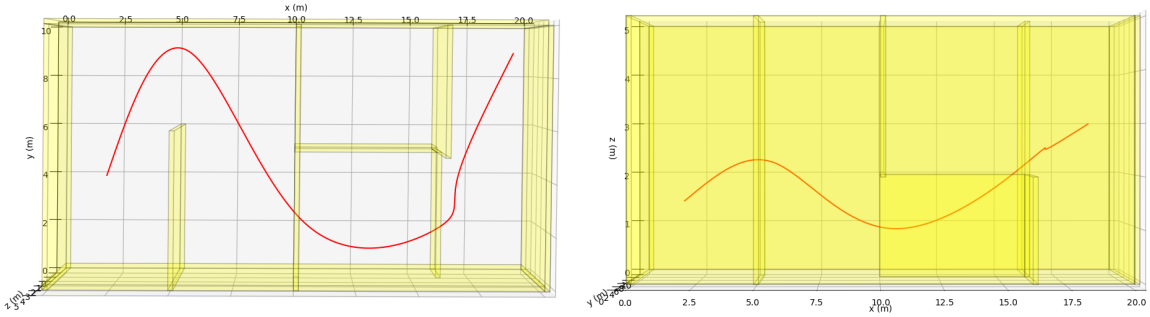
FIGURA 6.59 – RRT-C em Ambiente 3D Não Estruturado - Python. Em (a) vista superior e (b) visão lateral.



(a) (b)

Fonte: Própria Autora

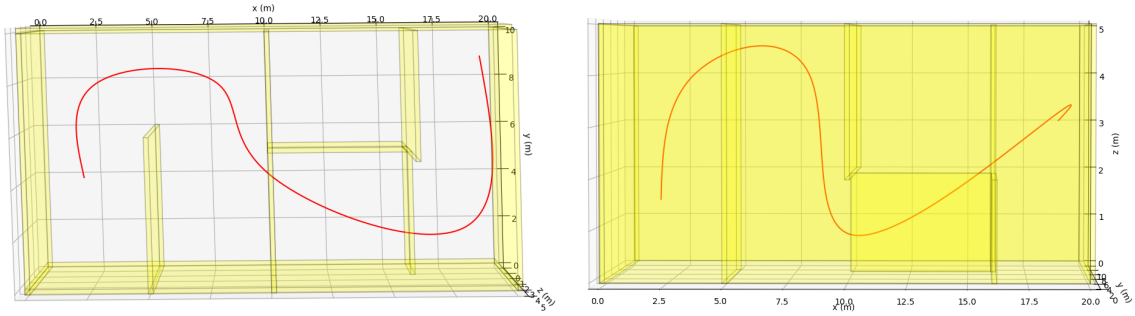
FIGURA 6.60 – PSO em Ambiente 3D Não Estruturado - Python. Em (a) vista superior e (b) visão lateral.



(a) (b)

Fonte: Própria Autora

FIGURA 6.61 – RL em Ambiente 3D Não Estruturado - Python. Em (a) vista superior e (b) visão lateral.



(a) (b)

Fonte: Própria Autora

Neste ambiente, o desafio para o UAV é maior, pois o percurso possui vários obstáculos, sendo necessário a execução de movimentos mais complexos. Além de ser indispensável que os movimentos sejam feitos por cima ou por baixo dos obstáculos, e requerer que o planejador identifique qual o melhor caminho durante os replanejamentos. Com este ambiente, todos os algoritmos retornaram trajetórias boas para serem seguidas. Além de terem escolhido a trajetória com menos curvas para chegar no objetivo. A trajetória mais arriscada a ser seguida foi a gerada pela técnica de aprendizado por reforço pois ficou próxima aos obstáculos.

Após realizar a análise desses algoritmos em ambientes com diferentes complexidades foi possível perceber que o A* retorna as trajetórias mais confiáveis em um tempo aceitável, porém o custo computacional é superior aos demais algoritmos. Em contrapartida, o RRT-C possui um custo computacional pequeno, retornando boas trajetórias no menor tempo entre os algoritmos.

O PSO e RL levam muito tempo para retornar uma trajetória, então não poderiam ser utilizados para um voo em tempo real. Porém, pode ser utilizado em missões que o planejamento seja feito antes do voo e adote algoritmos mais rápidos para o replanejamento.

O PSO retornou a menor média do comprimento de trajetórias em todos os ambientes, além de ter baixo custo computacional. Por isso, é uma ótima técnica para ser utilizada para calcular a trajetória inicial, e depois usar o A* ou RRT-C para fazer o replanejamento.

A técnica de aprendizado por reforço retornou trajetórias boas a serem seguidas, e com custo computacional baixo. Porém, o custo computacional cresceu de acordo com a complexidade do ambiente, assim como o do RRT-C. Por isso, se a ideia for usar algum desses algoritmos *on-board* será necessário fazer uma análise do tamanho do ambiente da missão e do custo com-

putacional.

Como o custo computacional do PSO se manteve constante e possui boas trajetórias, é uma boa escolha o utilizar para realizar o primeiro planejamento da trajetória. E utilizar os outros algoritmos para fazer o replanejamento, consumindo menos processamento.

6.2.2.2 Simulação no Gazebo

Na Tabela 6.11 é apresentado as métricas obtidas pela simulação no Gazebo no ambiente não estruturado. Nessa tabela, os desvios padrões são diferente de 0, ou seja, no meio da trajetória foi necessário realizar replanejamento pois havia colisão no trajeto. Nesse caso, os resultados também variaram das simulações em Python, já que os obstáculos são descobertos de acordo que o UAV se movimenta.

Tabela 6.11: Ambiente Não Estruturado no Gazebo - 3D.

	T_v (s)	T (s)	D (m)	CPU (MHz)	M (MB)	B (%)
A*	163	0,8±0,35	24,78	50,39	3,89	4
RRT-C	176	0,94±0,5	24,95	42	6,09	4
PSO	196,36	20,75±1,14	27,4	67,19	13,19	5
RL	211,22	23,26±1,83	29,28	187,6	21	5

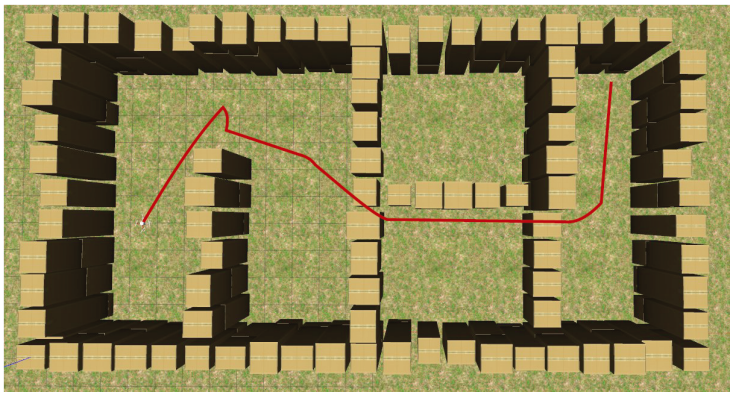
Como mostrado anteriormente, o gasto da bateria é proporcional ao tempo de voo, assim como ao tempo para gerar a trajetória. O A* foi o algoritmo que planejou a trajetória em menor tempo, assim como obteve o menor desvio padrão. Além disso, o A* também retornou a menor trajetória, seguido pelo RRT-C.

O tempo para gerar a trajetória do A* e do RRT-C foi semelhante ao encontrado em Python, provavelmente porque está considerando replanejamentos, ou seja, o número de obstáculos no ambiente é similar. Mas mesmo assim, o tempo do PSO diminuiu, mostrando que o tempo para o PSO gerar uma trajetória é exponencialmente dependente ao número de obstáculos no ambiente.

O custo computacional e o uso da memória de todos os algoritmo também diminuiu drasticamente, ou seja, também são exponencialmente dependentes ao número de obstáculos no ambiente. Com exceção do RL que manteve o mesmo custo computacional independentemente do cenário.

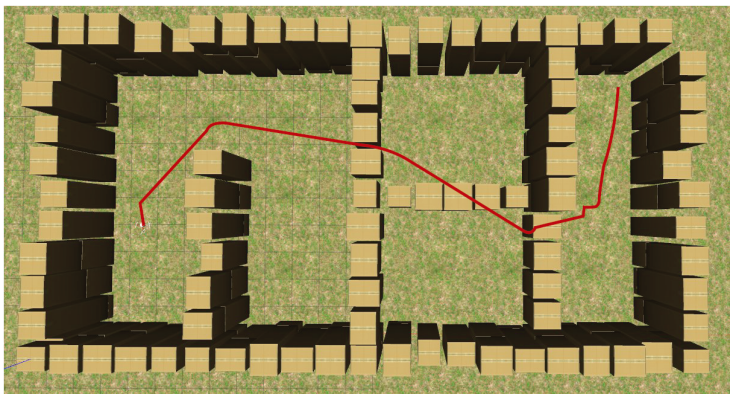
Das Figuras 6.62 até 6.65 são mostradas as trajetórias retornadas, no Gazebo, pelos algoritmos A*, RRT-C, PSO, e RL, respectivamente, no ambiente não estruturado.

FIGURA 6.62 – A* em Ambiente 3D Não Estruturado - Gazebo.



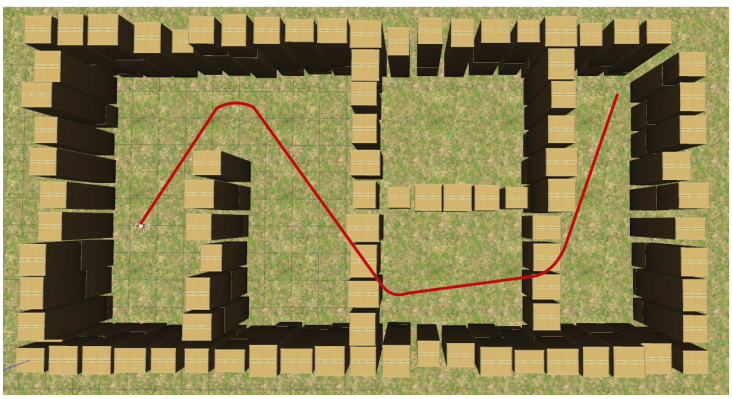
Fonte: Própria Autora

FIGURA 6.63 – RRT-C em Ambiente 3D Não Estruturado - Gazebo.



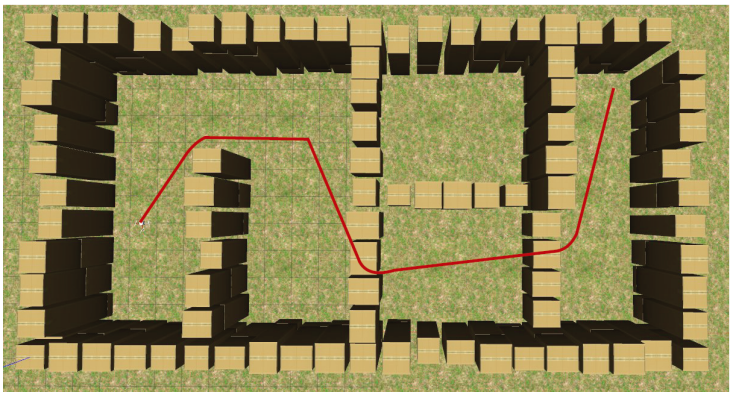
Fonte: Própria Autora

FIGURA 6.64 – PSO em Ambiente 3D Não Estruturado - Gazebo.



Fonte: Própria Autora

FIGURA 6.65 – RL em Ambiente 3D Não Estruturado - Gazebo.



Fonte: Própria Autora

O PSO e o RL retornaram boas trajetórias para serem seguidas pelo UAV. O RRT-C retornou uma trajetória com pequenas curvas que dificultam a locomoção do UAV. Além de ter passado pela parte do cenário que faz o UAV realizar mais movimentos bruscos. O A* também retornou uma trajetória com pequenas curvas fechadas que dificultam a locomoção do UAV, além de estar próximo aos obstáculos, o que poderia ocasionar colisão dependendo da robustez do planejador.

Como mostrado na Seção 6.2.2.1, o PSO é um bom algoritmo para ser utilizado para planejar a primeira trajetória. Pois além do que foi mostrado na Seção 6.2.2.1, a seção atual mostrou que o custo computacional do PSO depende do número dos obstáculos, ou seja, para o primeiro planejamento o custo será baixo como quase não terá obstáculos descobertos no cenário.

Porém, devido o baixo *completeness* da técnica, a técnica de aprendizado por reforço (priorizar segurança) ou o A* (priorizar tempo) devem apresentar resultados melhores, caso saiba que o cenário será muito complexo. Pois como será o primeiro planejamento, o tempo não será o fator mais importante, e sim o que deve ser priorizado na trajetória.

Um bom algoritmo para ser utilizado para o replanejamento da trajetória é o A*, pois é o algoritmo que apresenta trajetórias mais confiáveis, possui baixo custo computacional quando não se considera todo o ambiente. Além disso, retornou a trajetória em menor tempo, nas simulações realizadas no Gazebo.

6.3 Terceira Etapa

A terceira etapa ocorreu realizando testes em ambiente real para validar a robustez do planejador e avaliar como um UAV real se locomove com o algoritmo desenvolvido. Para isso,

as métricas analisadas foram: comprimento da trajetória (m) e tempo de voo (s), em diferentes zonas de risco. O ambiente escolhido para esta etapa é o mesmo utilizada na etapa 2, mostrado na Figura 6.49 (b).

Os voos foram realizados na área de Convivência do Departamento de Computação (DC) da UFSCar, sendo uma área de $5x2,5$ m. O ambiente no simulador possui $20x10$ m, por isso, a escala da trajetória foi reduzida para os testes realizados neste ambiente.

Na Figura 6.66, o *Parrot Bebop* 2 pode ser visto na local de testes, antes e durante o voo. Na área adotada nos testes não havia obstáculos. O mapa utilizado já era conhecido pelo UAV, pois havia sido pré-planejado pelo algoritmo A*, utilizando diferentes zonas de risco.

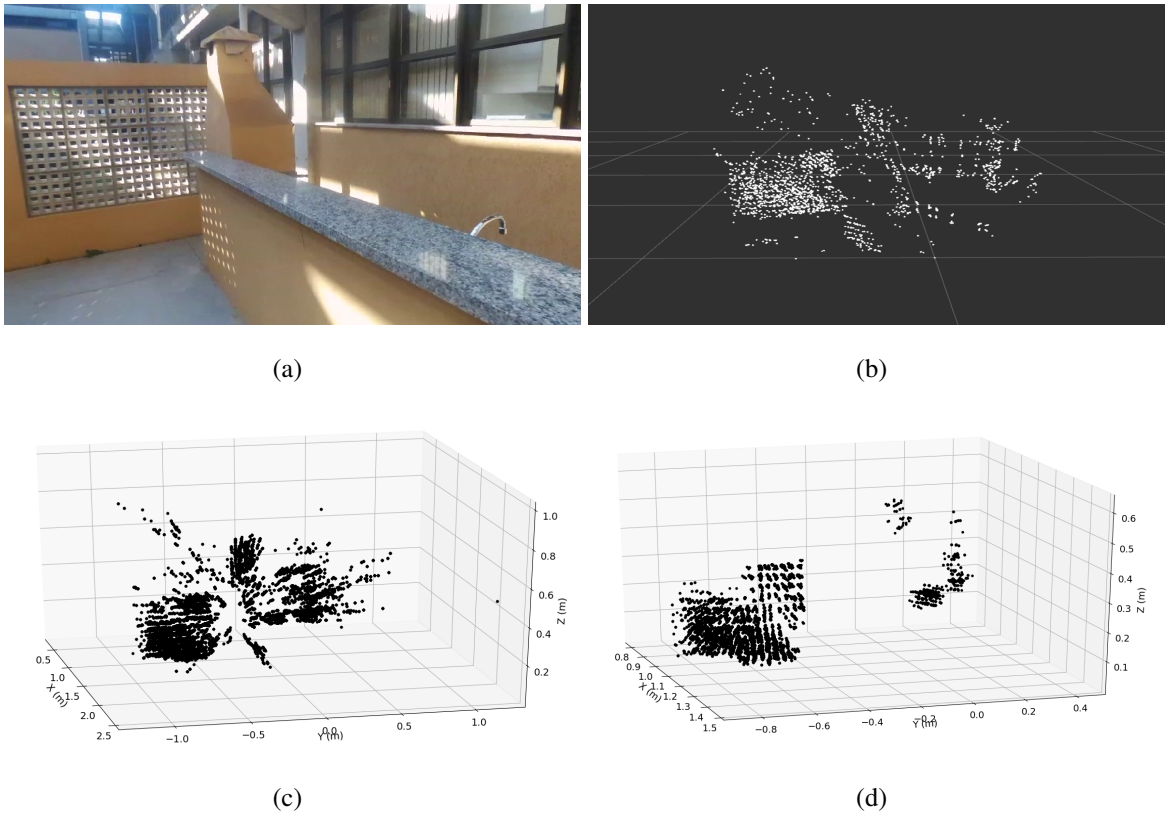
FIGURA 6.66 – Bebop na área de Convivência - DC/UFSCar. (a) Antes de iniciar o voo (b) Voando.

(a) (b)

Fonte: Própria Autora

Com a câmera do *Parrot Bebop* 2 não foi possível realizar o mapeamento do ambiente pois a nuvem de pontos retornada pela câmera monocular retorna mais obstáculos do que realmente há no ambiente, ou não detecta alguns obstáculos essenciais para o mapeamento. Mesmo com o filtro estatístico da PCL não foi possível filtrar para remover os ruídos. Um exemplo da *Point Cloud* retornada é mostrado na Figura 6.67. Neste exemplo, a parede é identificada, mas a churrasqueira não aparece na *Point Cloud*.

FIGURA 6.67 – *Point Cloud* retornada. Em (a) Ambiente real, (b) Visualização do Rviz, (c) Visualização em Python, sem filtro e (d) Visualização em Python com filtro estatístico.



Fonte: Própria Autora

Por este motivo, os voos em ambiente real foram feitos apenas para validar a robustez do planejador, comprovando que o mesmo é válido para realizar voos em ambiente real.

Como o mapeamento do ambiente não pode ser feito em tempo real, utilizamos o mapeamento realizado previamente e os dados dos obstáculos passados para o planejador. O controlador usado foi o R-LQR, e para evitar que o *optical flow* [3] tenha problemas ao se movimentar foram inseridas diversas *features* no chão (Fig. 6.68 (a)). A localização foi feita com o OrbSlam2, e para isso foram necessárias várias *features* no ambiente para ajudar o algoritmo a definir a coordenada atual. No local do teste havia várias *features* na parede, semelhantes ao tabuleiro de calibração da câmera, e foram adicionadas algumas extras no chão (Fig. 6.68 (b)). Um exemplo de como o OrbSlam 2 identifica o cenário é mostrado na Figura 6.69. A maneira como é feito a movimentação do UAV em ambiente real é mostrada na Figura 6.70.

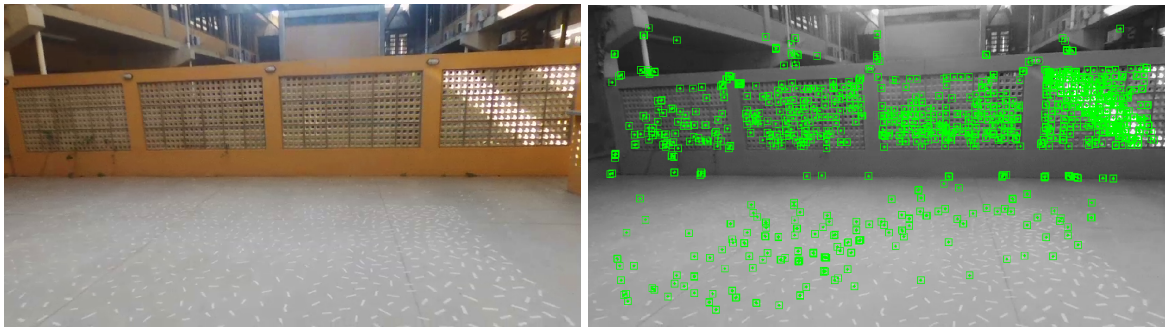
[3] Permite a navegação sem GPS, mas para detectar movimento é necessário que a textura da superfície abaixo do UAV não seja uniforme

FIGURA 6.68 – *Features* no ambiente. Em (a) *Features* para auxiliar o *Optical Flow* e (b) *Features* para auxiliar o OrbSlam 2.

(a) (b)

Fonte: Própria Autora

FIGURA 6.69 – Como o OrbSlam 2 identifica o cenário. Em (a) O cenário e (b) As *features* detectadas no cenário.



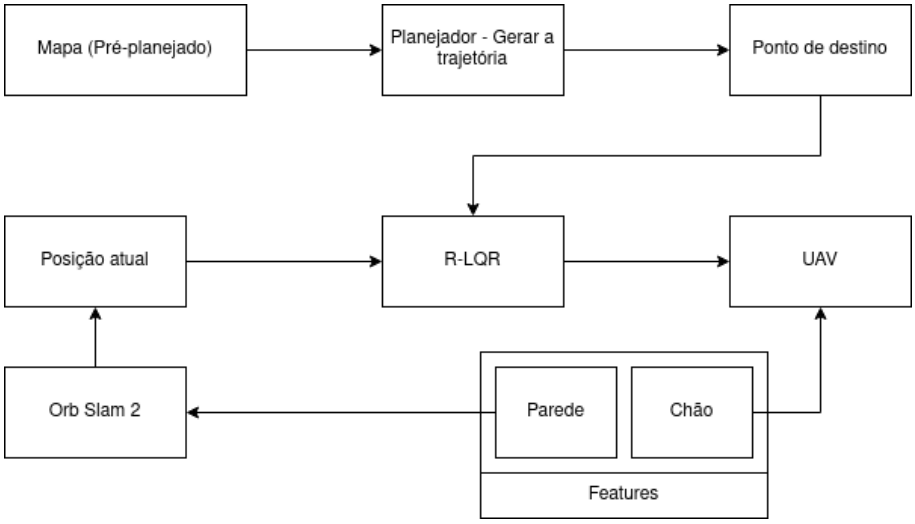
(a) (b)

Fonte: Própria Autora

6.3.1 Voo em Ambiente Não Estruturado

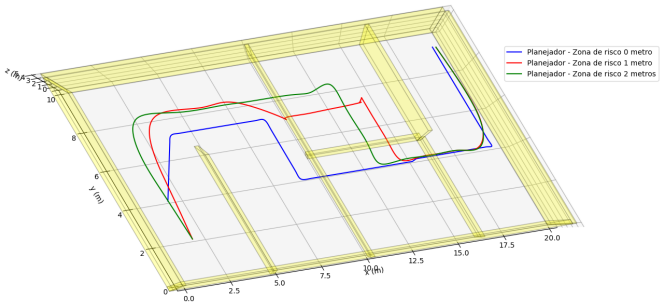
As trajetórias que deveriam ser executadas no ambiente real podem ser vista na Figura 6.71. A Figura 6.72 apresenta as trajetórias realizadas em ambiente real com diferentes tamanhos de zonas de risco. A trajetória em azul não possui zona de risco, a trajetória em vermelho possui zona de risco de 1 metro, e a trajetória em verde possui zona de risco de 2 metros. Cada voo foi realizado 3 vezes e o desempenho foi semelhante em todos os voos.

FIGURA 6.70 – Fluxograma de como o UAV se movimenta em ambiente real.



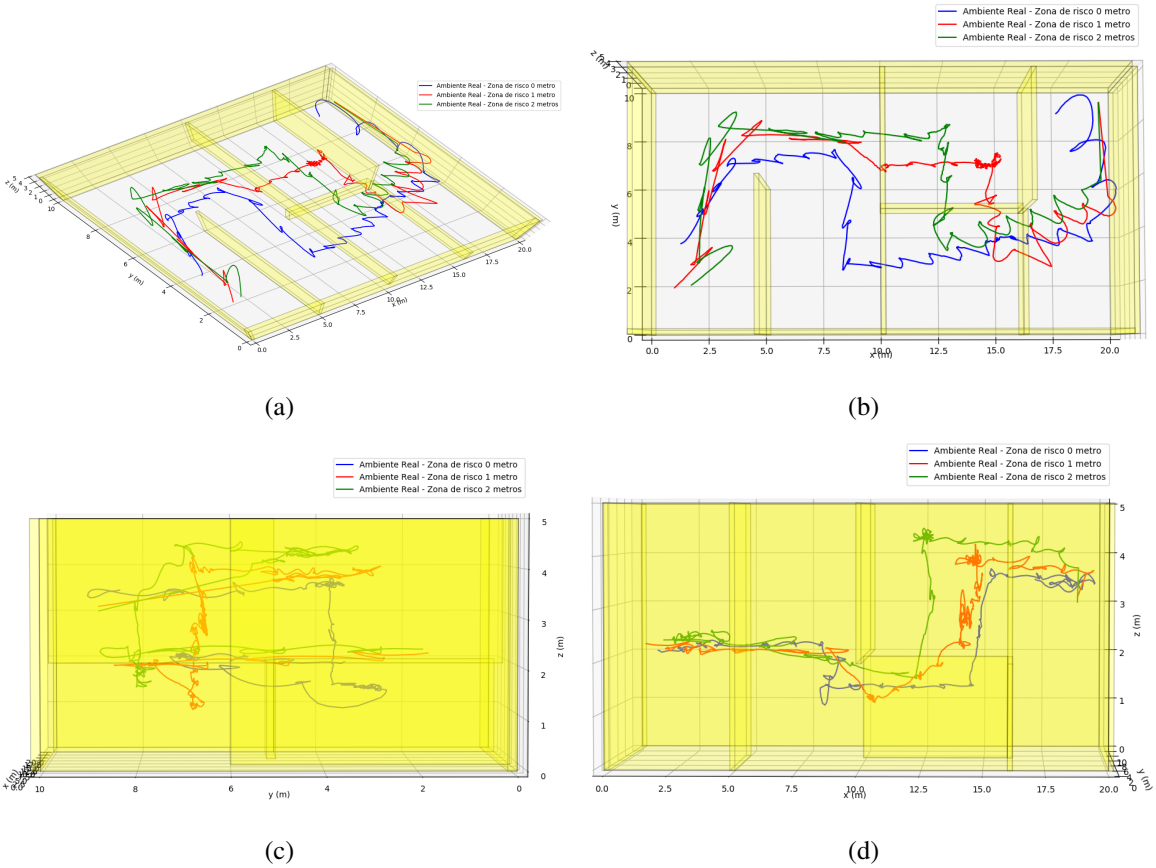
Fonte: Própria Autora

FIGURA 6.71 – Trajetórias retornadas pelo planejador. Em azul a trajetória sem zona de risco. Em vermelho a trajetória com zona de risco de 1 metro. Em verde a trajetória com zona de risco de 2 metros.



Fonte: Própria Autora

FIGURA 6.72 – Comparação dos os voos reais com diferentes zonas de risco. (a) Visão superior em diagonal (b) Visão superior (c) Visão lateral (d) Visão frontal.



(a) (b)

(c) (d)

Fonte: Própria Autora

A trajetória sem zona de risco tem o comprimento de 54,77 metros, a trajetória com zona de risco de 1 metro tem o comprimento de 65,06 metros, e a trajetória com zona de risco de 2 metros possui o comprimento de 77,37 metros.

O aumento do comprimento da trajetória é proporcional ao aumenta da zona de risco, pois necessita realizar uma trajetória maior para se distanciar dos obstáculos.

Ao analisar as trajetórias, principalmente pela visão superior, pode-se perceber que a trajetória sem zona de risco está bem próxima aos obstáculos, então caso haja algum erro no controle ou na localização há a probabilidade de colisão. Essa trajetória também retornou um caminho com alturas menores, sendo assim, o local de onde o OrbSlam 2 encontra as *features* se manteve constante e a localização foi precisa durante todo o trajeto. O que pode ser notado na reta final, aos 18 metros em X, que a trajetória sem zona de risco conseguiu manter a estabilidade mais do que as demais trajetórias.

As trajetórias com zona de risco de 1 e 2 metros chegaram a alturas de até 4 metros, próximo

ao obstáculo de 16 metros em X, então o local onde o OrbSlam 2 estava se baseando para definir o ponto atual mudou e a trajetória posterior não teve tanta estabilidade. Como pode ser visto na Figura 6.72 (b), a curva final, a partir de 13 metros em X, foi realizada em diagonal se juntando a reta final.

Apesar do erro na localização no final, não houve colisão, já que estava sendo considerado zonas de risco. Percebe-se que a trajetória com zona de risco de 2 metros chegou mais próximo a colisão, pois perdeu a sua localização antes da trajetória com zona de risco de 1 metro. A perda da localização em ambas as trajetórias ocorre quando os UAVs sobem de mais e trocam a referência do OrbSlam 2. Na trajetória com zona de risco de 1 metro ocorreu em 15 metros em X e com zona de risco de 2 metros ocorreu em 14 metros em X, sendo assim, o erro só foi aumentando até o fim da trajetória.

Na Tabela 6.12 é mostrado o tamanho da trajetória e o tempo de voo obtidos durante os voos. A análise não foi feita considerando a bateria pois o medidor de bateria do *Parrot Bebop* 2 é pouco preciso. Também não teve análise de memória e CPU pois os testes realizados em ambiente real foram para validar a robustez do planejador, então as trajetórias foram geradas previamente pelo A* (algoritmo que sempre retorna a mesma trajetória, caso o ambiente não mude).

Tabela 6.12: Análise Estatística dos Voos em Ambiente Real.

Zona de Risco (m)	Tamanho da Trajetória (m)	Tempo de Voo (s)
0	54.77	168.58
1	65.06	177.27
2	77.37	187.41

O tempo das trajetórias executadas em ambiente real foi similar ao tempo das trajetórias executadas no simulador mostradas na Etapa 2. Porém, o tempo de voo na etapa 2 considera o mapeamento e replanejamento da trajetória, o que não ocorre no voo em ambiente real. O tempo de voo, considerando apenas seguir a trajetória, no simulador é de 48,49, 52,76 e 57,74 segundos, respectivamente para cada zona de risco. O tempo de voo em ambiente real foi demasiadamente maior do que no simulador pois em ambiente real é necessário ter pausas entre cada ponto da trajetória para garantir que o controle se mantenha estável. Além disso, no simulador, a velocidade média é de 1 m/s, e em ambiente real é de 0,15 m/s.

O tamanho da trajetória e o tempo de voo são proporcionais ao tamanho da zona de risco, pois quanto maior a zona de risco, maior terá que ser a trajetória, e consequentemente o tempo para executá-la.

A trajetória sem zona de risco conseguiu seguir melhor a trajetória retornada pelo simulador, já que não precisou voar tão alto, então a referência para a localização se manteve a mesma, fazendo com que a trajetória seguisse corretamente o planejado. Mas mesmo assim, em diversos pontos a trajetória esteve bem perto dos obstáculos, já que a cada ponto que o UAV se movimenta há um certo erro na trajetória para parar. Caso ocorresse o controle gerasse um erro maior entre cada ponto da trajetória poderia ocasionar colisão.

Por isso, foi feito testes com trajetórias com zonas de risco de 1 e 2 metros. Porém para essas trajetórias serem completas era necessário voar até 4 metros, e consequentemente trocar o ponto de referência do OrbSlam 2 (diminuindo o número de *features*). Então, no final dessas trajetórias o percurso executado foi bem diferente do proposto, mas se ocasionar colisão.

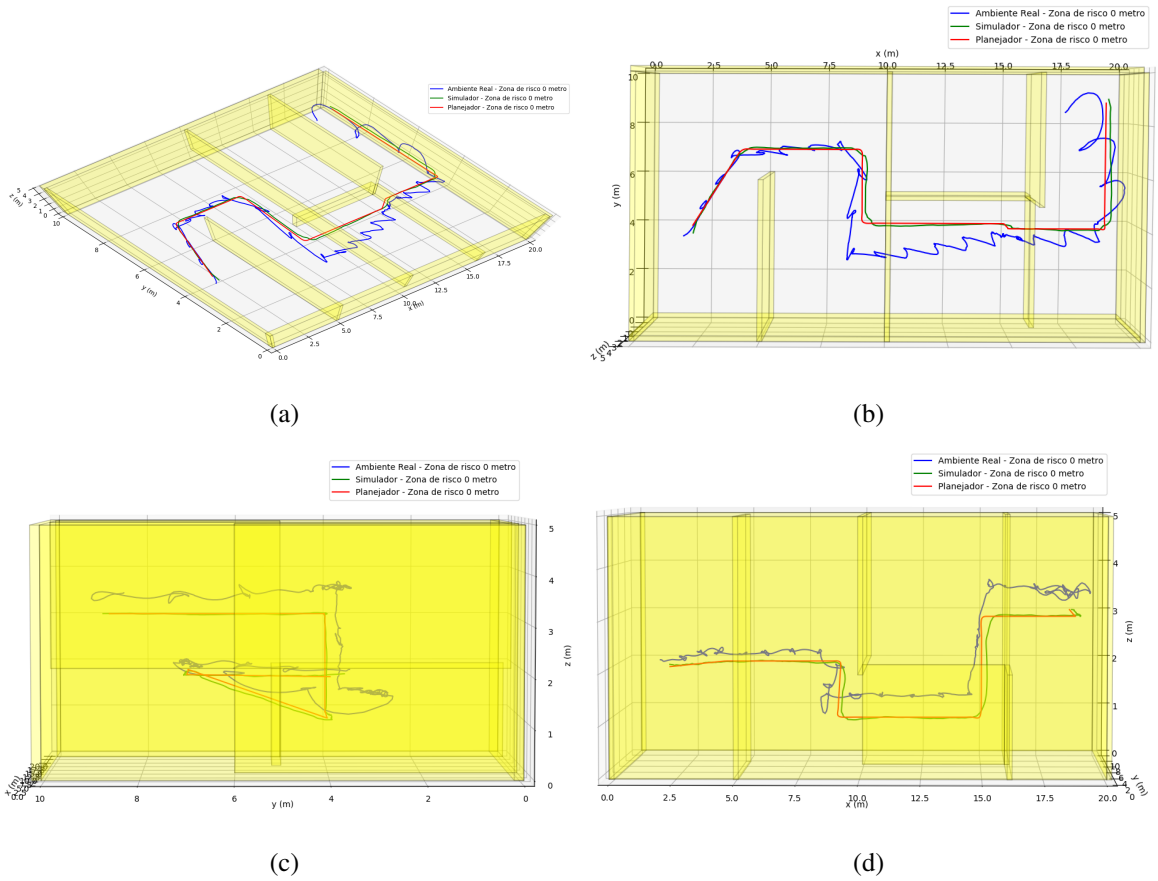
A trajetória com zona de risco de 1 metro começou a perder a referência da localização em 15 metros em X e com zona de risco de 2 metros ocorreu em 14 metros em X. O erro se mantém constante com o decorrer da trajetória, então a trajetória com zona de risco de 2 metros foi mais prejudicada, já que perdeu a localização primeiro.

Por estes motivos, a trajetória com zona de risco de 1 metro é a melhor escolha para ser utilizada em ambientes reais. Pois não se aproxima tanto dos obstáculos e não se distancia muito da trajetória ideal, e caso haja algum erro, como foi o caso desse teste, será menor do que outras zonas de risco. Além de que, o tempo para concluir a trajetória e o comprimento da trajetória são os menores dentre os que consideram uma zona de risco.

6.3.2 Análise da Robustez

A análise de robustez foi feita considerando 3 zonas de risco diferentes para determinar qual é a melhor zona de risco para realizar voos reais, devido ao erro entre a rota planejada e a executada. Todas as trajetórias utilizadas foram geradas a partir do algoritmo A*. Nas Figuras 6.73 até 6.75 é mostrado a comparação entre as trajetórias executadas pelo simulador, ambiente real, e as retornadas pelo planejador, com diferentes zonas de risco. As trajetórias em azul são do voo em ambiente real com o *Parrot Bebop* 2 e o controle R-LQR, as trajetórias em verde são as do ambiente simulado com o F450 e o controle MPC, e as trajetórias em vermelho são as retornadas pelo planejador.

FIGURA 6.73 – Análise da Robustez do UAV em ambiente real - A* - sem zona de risco. (a) Visão superior em diagonal (b) Visão superior (c) Visão lateral (d) Visão frontal.



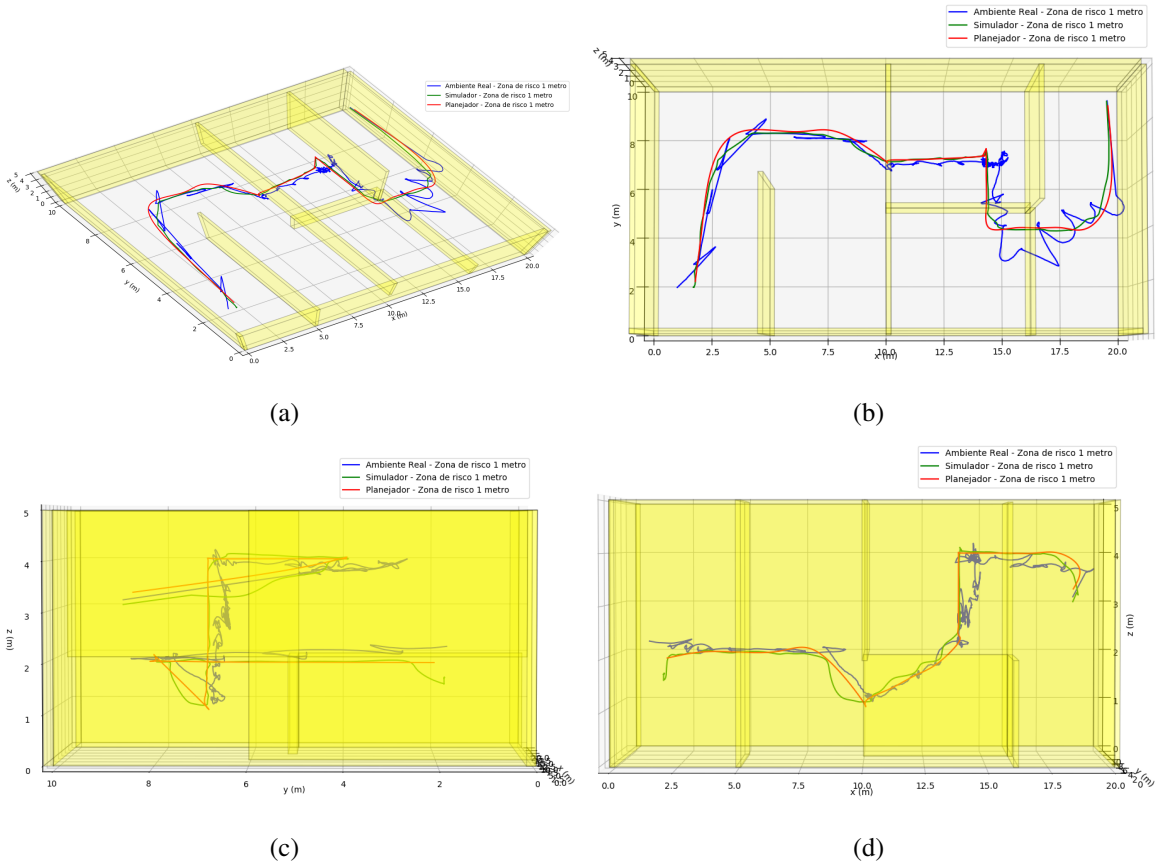
(a) (b)

(c) (d)

Fonte: Própria Autora

Na Figura 6.73 é apresentado as trajetórias sem zona de risco. Nesse caso, houve uma grande discrepância entre o voo em ambiente real e o retornado pelo planejador no eixo Z. No eixo X quase não houve variação, já no eixo Y houve uma variação na metade do percurso. Porém, nenhuma dessas variações da trajetória ocorrida em ambiente real teria resultado em colisão.

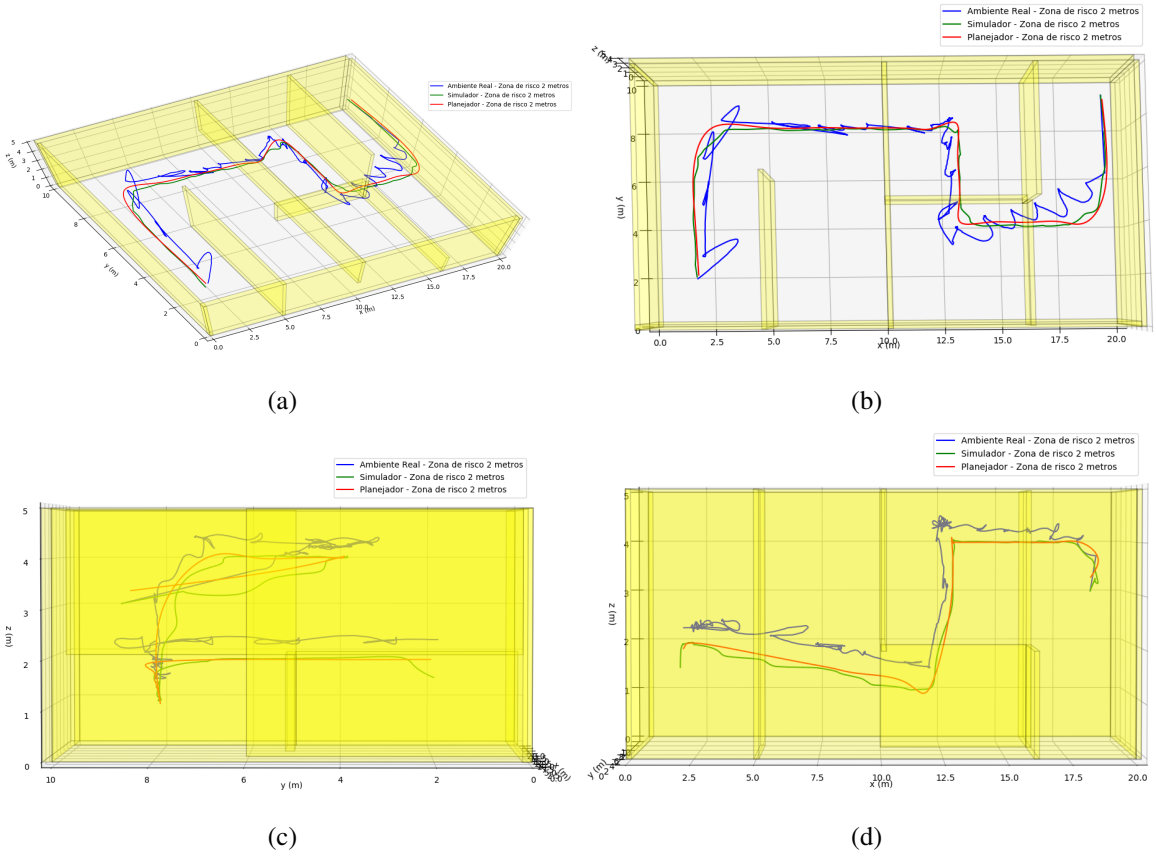
FIGURA 6.74 – Análise da Robustez do UAV em ambiente real - A* - com zona de risco de 1 metro. (a) Visão superior em diagonal (b) Visão superior (c) Visão lateral (d) Visão frontal.



Fonte: Própria Autora

Na Figura 6.74 é apresentado as trajetórias com zona de risco de 1 metro. Neste caso, o erro entre a trajetória retornada pelo planejador e a executada em ambiente real foi quase nulo até o momento em que o UAV atingiu 4 metros de altura. Nesse ponto, a referência utilizada para o OrbSlam 2 mudou, então o erro da localização aumentou, gerando grande variância entre a trajetória retornada pelo simulador e a executada em ambiente real no percurso final, a partir dos 15 metros em X.

FIGURA 6.75 – Análise da Robustez do UAV em ambiente real - A* - com zona de risco de 2 metros. (a) Visão superior em diagonal (b) Visão superior (c) Visão lateral (d) Visão frontal.



Fonte: Própria Autora

Na Figura 6.75 é apresentado as trajetórias com zona de risco de 2 metros. Esse caso foi semelhante ao de zona de risco com 1 metro, pois ao subir até os 4 metros de altura, o erro no eixo Y aumentou bastante, deixando a parte final da trajetória distante do planejado. Mas mesmo com o aumento do erro, não haveria colisão no cenário. Antes dos 12,5 metros em X, ou seja, antes da altura aumentar, a trajetória em ambiente real foi semelhante a trajetória retornada pelo simulador, apenas com um pequeno erro no eixo X.

Na Figura 6.73 até 6.75, analisando apenas as trajetórias retornadas pelo planejador e as executadas pelo simulador (trajetórias vermelhas e verde, respectivamente) é possível perceber que a variação existente no eixo X e Y são ínfimas, sendo a do eixo X um pouco maior no final da trajetória. Também quase não houve variação do eixo Z, o único momento que houve variação foi no início e final das trajetórias, devido a decolagem e pouso do UAV.

Na Tabela 6.13 é mostrado a distância percorrida por cada uma das trajetórias, em cada ambiente e zonas de risco. A variação da trajetória do simulador e a trajetória retornada pelo

planejador foi de 1,04, 1,71, e 2,25 metros, considerando os voos sem zona de risco, com 1 e 2 metros, respectivamente. Com isso, podemos perceber que o erro entre as trajetórias aumenta de acordo com o comprimento da trajetória.

Tabela 6.13: Análise de Robustez com Diferentes Zonas de Risco - A*.

Ambiente	Zona de Risco (m)	Tamanho da Trajetória (m)
Planejador	0	27,17
	1	28,13
	2	28,66
Simulador	0	28,21
	1	29,84
	2	30,91
Ambiente Real	0	54,77
	1	65,06
	2	77,37

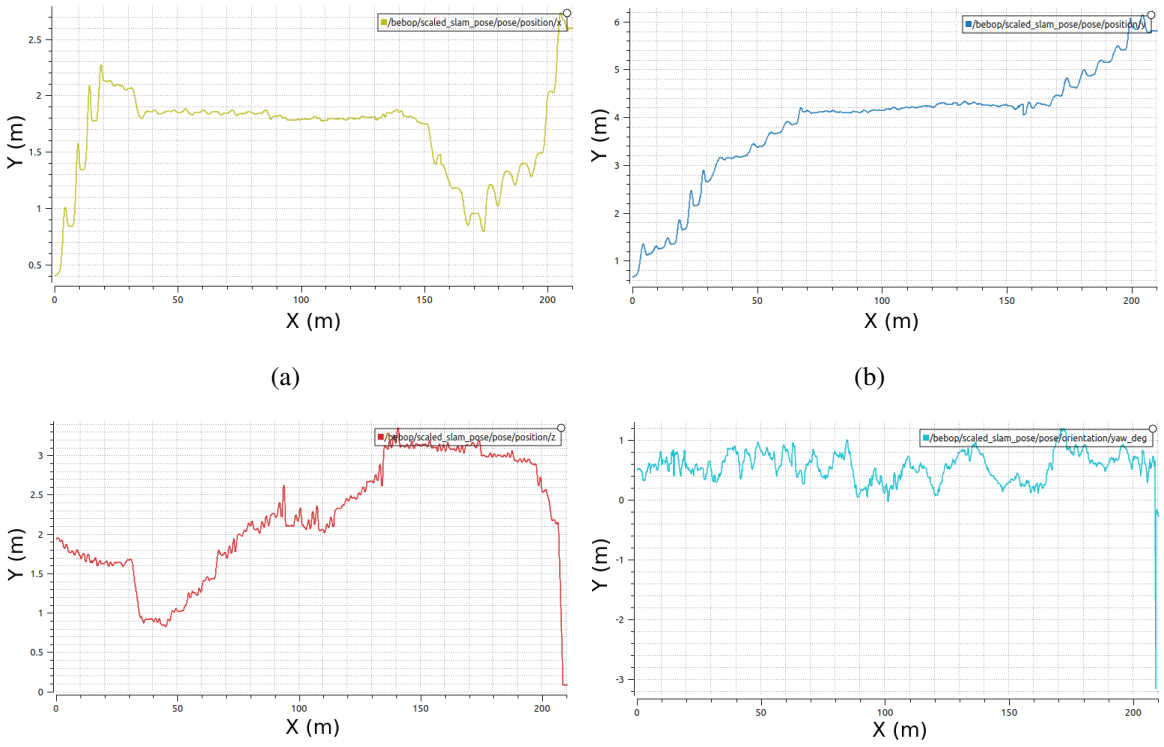
A variação da trajetória realizada em ambiente real e a trajetória retornada pelo planejador foi de 27,6, 36,93, e 48,71 metros, considerando os voos sem zona de risco, com 1 e 2 metros, respectivamente. O erro também aumentou de acordo com o comprimento da trajetória, só que em uma proporção bem maior do que houve com o ambiente simulado. Apesar da variação entre as trajetórias ter sido grande, não houve colisão durante o percurso.

Essa grande variação deve-se ao fato de que o planejador proposto retorna trajetórias discretas, e o controle R-LQR considera cada ponto desses como se fosse uma trajetória completa. Por isso que a cada ponto que o UAV se movimenta, o início e o final, possuem um erro maior, já que o UAV para nesses pontos e acaba por ir um pouco a frente e depois voltar para o ponto requisitado. Por consequência, as trajetórias em ambientes reais são maiores.

A variação entre as trajetórias executadas no simulador e as retornadas pelo planejador é de aproximadamente 5%, considerando as trajetórias executadas em ambiente real a variação é de quase 50%. Apesar da variação ser bem alta, não houve colisão em nenhuma trajetória, então pode-se afirmar que a robustez do planejador é boa.

A Figura 6.76 mostra as variações da posição do *Parrot Bebop* 2 ao longo dos eixos na trajetória com margem de erro de 1 metro, o qual obteve melhores resultados. Em cada eixo é possível ver que entre cada *waypoint* há um pico na posição do UAV, principalmente nos eixos X e Y. Esses picos na posição produzem o erro entre os *waypoints* mostrado na Figura 6.72.

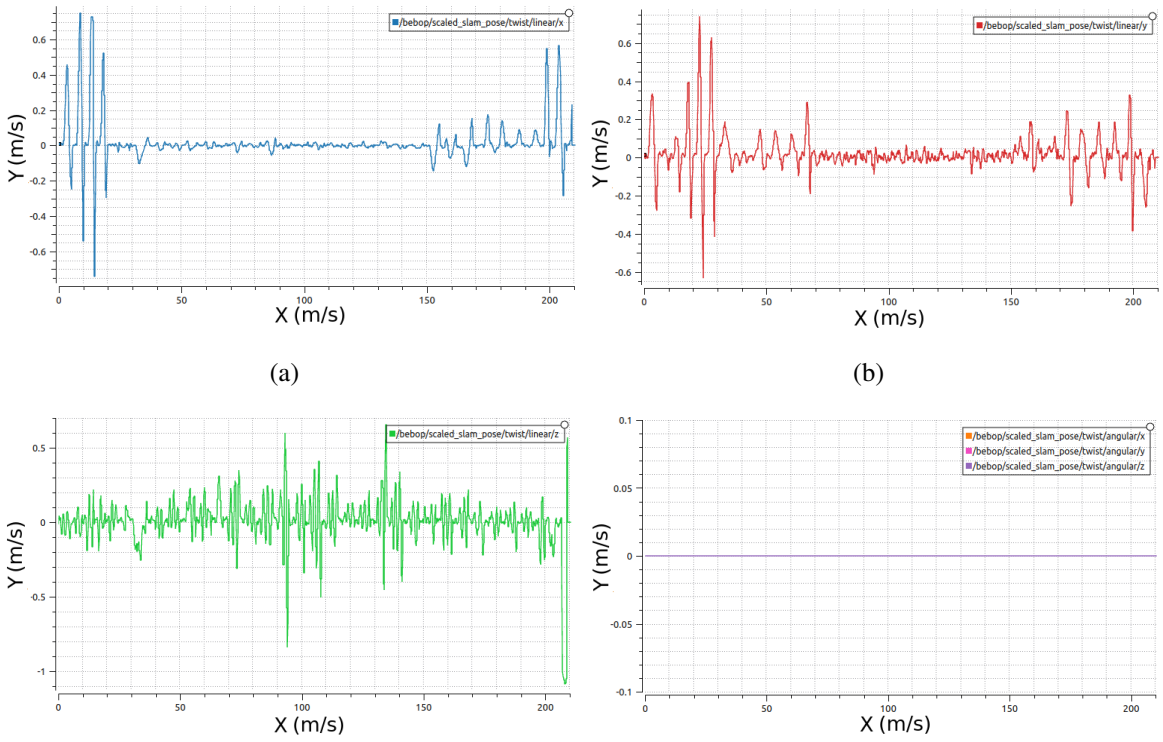
FIGURA 6.76 – As posições do *Parrot Bebop* 2 ao longo dos eixos. (a) Eixo X (b) Eixo Y (c) Eixo Z (d) Yaw.



Fonte: Própria Autora

A Figura 6.76 mostra as variações da velocidade do *Parrot Bebop* 2 ao longo dos eixos, também na trajetória com margem de erro de 1 metro. Nas Figuras 6.76 (a) e (b) é possível ver a velocidade máxima do UAV em cada eixo. A velocidade máxima definida é de 0.15 m/s, e foi a velocidade mantida na maior parte trajetória, porém na mudança entre *waypoints* a velocidade chega a 0.6 m/s ocasionando os picos na posição do UAV e os erros nas trajetórias entre cada *waypoint*.

FIGURA 6.77 – As velocidades do *Parrot Bebop* 2 ao longo dos eixos. (a) Eixo X (b) Eixo Y (c) Eixo Z (d) Todas as trajetórias de velocidade angular.



Fonte: Própria Autora

No eixo Z é perceptível que a velocidade nunca fica estável já que o UAV está sempre tentando se estabilizar no ar. E no eixo do Yaw está explícito que não houve variação da velocidade informados pelo planejador proposto ou pelo controle. Porém, há variação da posição no eixo Yaw, demonstrando as restrições físicas dos veículo e a necessidade do uso da margem de segurança para diminuir as chances de ter colisões.

6.4 Considerações finais

Neste capítulo apresentou-se as simulações de cada técnica proposta, assim como foi feita uma análise dos resultados explicando qual técnica melhor se encaixa para diversas situações. Além de mostrar o desvio de obstáculo dinâmico em 2D, o qual pode também pode ser implementado em conjunto com o algoritmo de ambiente desconhecido e não estruturado. Mas para manter os resultados mais estáveis visando validá-los, optou-se apresentá-los separadamente.

Então, foi apresentado as simulações feitas em ambiente 3D, realizando uma análise com

simulações feitas em Python e no Gazebo, considerando todas as métricas apresentadas na Seção 3.5. Além de apresentar os resultados dos voos em ambiente real, fazendo uma análise de robustez em relação a trajetória retornada pelo planejador, a executada no simulador, e a executada em ambiente real.

Capítulo 7

CONCLUSÕES E TRABALHOS FUTUROS

Neste capítulo apresenta-se as considerações finais, assim como o cronograma com as atividades realizadas até o momento e o planejamento para as atividades a serem realizadas.

7.1 Considerações Finais

Foram feitos testes em Python, em 2D, utilizando técnicas clássicas, meta heurísticas, e de aprendizado de máquina em ambientes não estruturados e desconhecidos, bem como testes utilizando um algoritmo simples, na Seção 5.8, para desvio de obstáculos dinâmicos.

Com isso, mostrou-se que cada técnica de Planejamento de Trajetória é melhor implementada em certas missões do que em outras. Este trabalho analisou cada uma dessas técnicas e verificou qual seria o melhor ambiente para implementar cada uma delas. Além de mostrar como a complexidade de cada uma é aumentada de acordo com o ambiente. E foi evidente o formato mais comum que cada técnica costuma retornar a trajetória.

As técnicas meta-heurísticas costumam retornar trajetória com uma grande curva aberta, facilitando a trajetória do UAV. Assim como as técnicas de aprendizado de máquina tendem a retornar trajetórias com mais retas, possibilitando que o UAV aumente a velocidade durante o percurso. As técnicas clássicas tendem retornar trajetórias retilíneas, mas sempre seguem pelo meio do cenário, então precisam desviar mais vezes de obstáculos, ocasionando diversas curvas pequenas e fechadas, mas suavizadas, para chegar até o objetivo. Já as técnicas de aprendizado por reforço chegam ao seu objetivo com o menor número de curvas possível, sendo o restante da trajetória retilíneo.

Descobriu-se que o tempo de resposta do A* é mais influenciado pelo complexidade do ambiente. Já as técnicas PRM, RRT, PSO, GWO, GSO e RL sofreram mais impacto com a

dimensão do ambiente. E o APF com a quantidade de obstáculos presentes em cada ambiente. O RRT-C foi o único que sofreu grandes mudanças com a complexidade, número de obstáculos e dimensão do ambiente. Mas mesmo assim, foi um dos algoritmos que retornou a trajetória em menor tempo, em todos os casos.

Pode-se perceber que a técnica A* é a melhor a ser usada em ambientes completamente desconhecidos. E as técnicas de aprendizado de máquina e meta-heurísticas, ou o APF se tiver muitos obstáculos, são a melhor opção caso necessite que a missão seja rapidamente cumprida. Se for uma missão a ser feita em florestas ou montanhas será melhor utilizar as técnicas clássicas aproximadas, já que conseguem explorar melhor o ambiente. Caso seja uma missão em ambiente urbano o A* ou PRM são as técnicas mais indicadas devido sua alta confiabilidade, demonstrada pela variância e desvio padrão, e velocidade para lidar com incertezas.

Os testes realizados em 3D foram feitos em Python, no simulador Gazebo, e em ambiente real. Por estar sendo utilizado o sistema MRS, com o ROS, no ambiente simulado e real, foi possível aproveitar a validação da arquitetura do planejador utilizado no ambiente simulado durante os testes em ambiente real.

As análises dos algoritmos em ambiente 3D mostraram que o PSO é o melhor algoritmo para ser utilizado para realizar o primeiro planejamento, já que o seu custo computacional é menor quando tem menos obstáculos para considerar. Mesmo sendo um dos algoritmos que retornou em maior tempo, por ser o primeiro planejamento, não irá atrapalhar um voo em tempo real. Além disso, o PSO retornou as menores trajetórias e mais suaves. Porém, devido seu baixo *completeness*, é melhor utilizar o RL (priorizar segurança) ou o A* (priorizar velocidade), caso saiba que o ambiente é muito complexo.

Como algoritmo de replanejamento, o A* é uma boa escolha, pois retornou a trajetória em menor tempo nas simulações realizadas no Gazebo. Além de que todos os seus planejamentos apresentam baixo desvio padrão, em tempo e distância, demonstrando confiabilidade.

Por fim, foram realizados os voos em ambiente real para validar a robustez do planejador. Foram feitos voos com 3 zonas de risco diferente, e cada um deles foi realizado 3 vezes, confirmando a confiabilidade do planejador em obter os mesmos resultados, independente do momento do voo.

Ao utilizar o *Parrot Bebop* 2 com sua câmera monocular e o *optical flow* percebe-se que ao perder a referência da localização a trajetória se torna instável, e que a cada ponto da trajetória que é executado ocorre um pequeno erro na trajetória para começar e terminar o voo. Pensando nessas características, para esse UAV, definiu-se que um voo com zona de risco de 1 metro seria

a melhor opção para voos em ambiente real.

Para voos com outros UAVs a zona de risco pode mudar de acordo com a instabilidade de movimento do mesmo, e a precisão da localização. Com menos precisão do movimento e da localização é necessário aumentar a zona de risco, e com mais precisão pode ser possível até realizar voos sem zona de risco.

Quando a trajetória não utiliza zona de risco é possível haver colisão caso o erro na movimentação do UAV entre um ponto e outro da trajetória aumente. Ao utilizar zonas de risco maiores a trajetória se distância bastante da trajetória ideal. E caso haja um erro de localização, o erro presente até o final do percurso será menor do que com zonas de risco maiores, já que a trajetória é menor.

7.2 Trabalhos Futuros

Para os trabalhos futuros pretende-se realizar os mesmos testes da etapa 3 no UAV F-450. Além de utilizar uma técnica de aprendizado por reforço contínuo com auxílio de técnicas de otimização para desenvolver um planejador de trajetória para altas velocidades. Além de realizar o mapeamento do cenário através de câmeras, sem a necessidade de uma nuvem de pontos.

7.3 Artigos

Esta seção apresenta os trabalhos que estão sendo desenvolvidos e para quais congressos e revistas estão sendo enviados.

O artigo "*Comparison between Meta Heuristic Algorithms for Path Planning*"foi aceito no *Workshop on MSc and PhD Works in Robotics* presente no *17th IEEE Latin American Robotics Symposium* - LARS 2020 (ROCHA; VIVALDINI, 2020), sendo os autores Lidia Rocha, e Kelen Vivaldini.

O artigo "*A UAV Global Planner to Improve Path Planning in Unstructured Environments*"foi aceito no *International Conference on Unmanned Aircraft Systems* (ROCHA et al., 2021), sendo os autores Lidia Rocha, Marcela Aniceto, Igor Araújo, e Kelen Vivaldini.

REFERÊNCIAS

AGGARWAL, S.; KUMAR, N. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Computer Communications*, Elsevier, 2019.

AGGARWAL, S.; KUMAR, N. Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges. *Computer Communications*, Elsevier, v. 149, p. 270–299, 2020.

ALEJO, D. et al. Optimal reciprocal collision avoidance with mobile and static obstacles for multi-UAV systems. In: IEEE. *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*. [S.l.], 2014. p. 1259–1266.

ALOTAIBI, E. T.; ALQEFARI, S. S.; KOUBAA, A. Lsar: Multi-UAV collaboration for search and rescue missions. *IEEE Access*, IEEE, v. 7, p. 55817–55832, 2019.

BACA, T. et al. The MRS UAV system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles. *arXiv preprint arXiv:2008.08050*, 2020.

BAYERLEIN, H. et al. Multi-uav path planning for wireless data harvesting with deep reinforcement learning. *IEEE Open Journal of the Communications Society*, IEEE, 2021.

BENCY, M. J.; QURESHI, A. H.; YIP, M. C. Neural path planning: Fixed time, near-optimal path generation via oracle imitation. *arXiv preprint arXiv:1904.11102*, 2019.

BENEVIDES, J. R. et al. Ros-based robust and recursive optimal control of commercial quadrotors. In: IEEE. *15th Int. Conf. on Automation Science and Engineering (CASE)*. [S.l.], 2019. p. 998–1003.

BISWAS, S. et al. A particle swarm optimization based path planning method for autonomous systems in unknown terrain. In: IEEE. *2019 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT)*. [S.l.], 2019. p. 57–63.

BLÖSCH, M. et al. Vision based mav navigation in unknown and unstructured environments. In: IEEE. *2010 IEEE International Conference on Robotics and Automation*. [S.l.], 2010. p. 21–28.

BORTOFF, S. A. Path planning for UAVs. In: IEEE. *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*. [S.l.], 2000. v. 1, n. 6, p. 364–368.

BREEN, J. et al. Early warning model of dangerous road pavement condition using UAV. 2020.

BáčA, T. et al. The MRS UAV system: Pushing the frontiers of reproducible research, real-world deployment, and education with autonomous unmanned aerial vehicles. In: . [S.l.: s.n.], 2020.

CARRIO, A. et al. A review of deep learning methods and applications for unmanned aerial vehicles. *Journal of Sensors*, Hindawi, v. 2017, 2017.

CETIN, O.; YILMAZ, G. Real-time autonomous UAV formation flight with collision and obstacle avoidance in unknown environment. *Journal of Intelligent & Robotic Systems*, Springer, v. 84, n. 1-4, p. 415–433, 2016.

CHEN, J. et al. An improved probabilistic roadmap algorithm with potential field function for path planning of quadrotor. In: IEEE. *2019 Chinese Control Conference (CCC)*. [S.l.], 2019. p. 3248–3253.

CHEN, S. et al. An improved artificial potential field based path planning algorithm for unmanned aerial vehicle in dynamic environments. In: IEEE. *2017 International Conference on Security, Pattern Analysis, and Cybernetics (SPAC)*. [S.l.], 2017. p. 591–596.

CHEN, Z.; LUO, F.; ZHAI, C. Obstacle avoidance strategy for quadrotor UAV based on improved particle swarm optimization algorithm. In: IEEE. *2019 Chinese Control Conference (CCC)*. [S.l.], 2019. p. 8115–8120.

CHENG, P.; KELLER, J.; KUMAR, V. Time-optimal UAV trajectory planning for 3d urban structure coverage. In: IEEE. *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.], 2008. p. 2750–2757.

CHOSET, H. M. et al. *Principles of robot motion: theory, algorithms, and implementation*. [S.l.]: MIT press, 2005.

COMBA, L. et al. Unsupervised detection of vineyards by 3d point-cloud UAV photogrammetry for precision agriculture. *Computers and Electronics in Agriculture*, Elsevier, v. 155, p. 84–95, 2018.

CUI, J. et al. Path planning algorithms for power transmission line inspection using unmanned aerial vehicles. In: IEEE. *2017 29th Chinese Control And Decision Conference (CCDC)*. [S.l.], 2017. p. 2304–2309.

CURIAC, D.-I.; VOLOSENCU, C. Path planning algorithm based on arnold cat map for surveillance UAVs. *Defence Science Journal*, v. 65, n. 6, p. 483–488, 2015.

DERAY, J. et al. Timed-elastic smooth curve optimization for mobile-base motion planning. In: IEEE. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2019. p. 3143–3149.

DEWANGAN, R. K.; SHUKLA, A.; GODFREY, W. W. Three dimensional path planning using grey wolf optimizer for UAVs. *Applied Intelligence*, Springer, v. 49, n. 6, p. 2201–2217, 2019.

DOKEROGLU, T. et al. A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering*, Elsevier, v. 137, p. 106040, 2019.

DONG, S. Application research of quadrotor uav motion planning system based on heuristic search and minimum-snap trajectory optimization. In: IEEE. *2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. [S.l.], 2021. v. 5, p. 2270–2274.

DOSOVITSKIY, A. et al. Carla: An open urban driving simulator. In: PMLR. *Conference on robot learning*. [S.l.], 2017. p. 1–16.

DUCHOĖ, F. et al. Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering*, v. 96, p. 59–69, 2014.

DUTTA, S. et al. Automatic re-planning of lifting paths for robotized tower cranes in dynamic bim environments. *Automation in Construction*, Elsevier, v. 110, p. 102998, 2020.

EBERHART, R.; KENNEDY, J. Particle swarm optimization. In: CITESEER. *Proceedings of the IEEE international conference on neural networks*. [S.l.], 1995. v. 4, p. 1942–1948.

EVERETT, M.; CHEN, Y. F.; HOW, J. P. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In: IEEE. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2018. p. 3052–3059.

FAESSLER, M. et al. Autonomous, vision-based flight and live dense 3D mapping with a quadrotor micro aerial vehicle. *Journal of Field Robotics*, Wiley Online Library, v. 33, n. 4, p. 431–450, 2016.

FORSTER, C. et al. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, IEEE, v. 33, n. 1, p. 1–21, 2016.

FRANCIS, G. et al. *Occupancy Map Building through Bayesian Exploration*. 2017.

FREIMUTH, H.; KÖNIG, M. Planning and executing construction inspections with unmanned aerial vehicles. *Automation in Construction*, Elsevier, v. 96, p. 540–553, 2018.

FRITSCH, F. N.; CARLSON, R. E. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, SIAM, v. 17, n. 2, p. 238–246, 1980.

GALVEZ, R. L.; DADIOS, E. P.; BANDALA, A. A. Path planning for quadrotor UAV using genetic algorithm. In: IEEE. *2014 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*. [S.l.], 2014. p. 1–6.

GIERNACKI, W. et al. Bebop 2 quadrotor as a platform for research and education in robotics and control engineering. In: IEEE. *2020 International Conference on Unmanned Aircraft Systems (ICUAS)*. [S.l.], 2020. p. 1733–1741.

GODOY, J. E. et al. Adaptive learning for multi-agent navigation. In: *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. [S.l.: s.n.], 2015. p. 1577–1585.

GOEL, U. et al. Three dimensional path planning for UAVs in dynamic environment using glow-worm swarm optimization. *Procedia computer science*, Elsevier, v. 133, p. 230–239, 2018.

GUTIERREZ, M. A.; D'HARO, L. F.; BANCHS, R. A multimodal control architecture for autonomous unmanned aerial vehicles. In: *Proceedings of the Fourth International Conference on Human Agent Interaction*. [S.l.: s.n.], 2016. p. 107–110.

HAYAT, S. et al. Multi-objective UAV path planning for search and rescue. In: IEEE. *2017 IEEE International Conference on Robotics and Automation (ICRA)*. [S.l.], 2017. p. 5569–5574.

HEIDARI, A.; ABBASPOUR, R. Autonomous UAV path planning for search and rescue missions in post natural disaster assessment based on novel g-bfoa algorithm. *Journal of Geomatics Science and Technology*, Journal of Geomatics Science and Technology, v. 3, n. 4, p. 41–52, 2014.

HU, H. et al. Cuckoo search-based method for trajectory planning of quadrotor in an urban environment. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, SAGE Publications Sage UK: London, England, v. 233, n. 12, p. 4571–4582, 2019.

HWANG, Y. K.; AHUJA, N. et al. A potential field approach to path planning. *IEEE Transactions on Robotics and Automation*, Citeseer, v. 8, n. 1, p. 23–32, 1992.

JUAN, V. S.; SANTOS, M.; ANDÚJAR, J. M. Intelligent UAV map generation and discrete path planning for search and rescue operations. *Complexity*, Hindawi, v. 2018, 2018.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. *Journal of artificial intelligence research*, v. 4, p. 237–285, 1996.

KAIPA, K. N.; GHOSE, D. *Glowworm swarm optimization: theory, algorithms, and applications*. [S.l.]: Springer, 2017. v. 698. 1–18 p.

KANCIR, P.; DIGUET, J.; SEVAUX, M. Development of tools for multi vehicles simulation with robot operating system and ardupilot. In: . [S.l.: s.n.], 2019.

KAUFMANN, E. et al. Deep drone racing: Learning agile flight in dynamic environments. *arXiv preprint arXiv:1806.08548*, 2018.

KAVRAKI, L. E. et al. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, IEEE, v. 12, n. 4, p. 566–580, 1996.

KAVRAKI, L. E. et al. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, IEEE, v. 12, n. 4, p. 566–580, 1996.

KIM, I. et al. Obstacle avoidance path planning for UAV using reinforcement learning under simulated environment. In: *IASER 3rd International Conference on Electronics, Electrical Engineering, Computer Science, Okinawa*. [S.l.: s.n.], 2017. p. 34–36.

KLEMM, S. et al. RRT-connect: Faster, asymptotically optimal motion planning. In: IEEE. *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. [S.l.], 2015. p. 1670–1677.

KOCH, T.; KÖRNER, M.; FRAUNDORFER, F. Automatic and semantically-aware 3d UAV flight planning for image-based 3d reconstruction. *Remote Sensing*, Multidisciplinary Digital Publishing Institute, v. 11, n. 13, p. 1550, 2019.

KOREN, Y.; BORENSTEIN, J. et al. Potential field methods and their inherent limitations for mobile robot navigation. In: *ICRA*. [S.l.: s.n.], 1991. v. 2, p. 1398–1404.

KOSLOSKY, E. et al. *Geração de trajetória em espiral e navegação com desvio de obstáculos para veículos aéreos não-tripulados*. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2018.

KRELL, E. et al. Collision-free autonomous robot navigation in unknown environments utilizing PSO for path planning. *Journal of Artificial Intelligence and Soft Computing Research*, Sciendo, v. 9, n. 4, p. 267–282, 2019.

KUFFNER, J. J.; LAVALLE, S. M. RRT-connect: An efficient approach to single-query path planning. In: IEEE. *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*. [S.l.], 2000. v. 2, p. 995–1001.

KULKARNI, S. et al. UAV aided search and rescue operation using reinforcement learning. *arXiv preprint arXiv:2002.08415*, 2020.

KUMAR, S. et al. Path planning and control of mobile robots using modified tabu search algorithm in complex environment. *Available at SSRN 3539922*, 2020.

KUMAR, V.; KUMAR, D. An astrophysics-inspired grey wolf algorithm for numerical optimization and its application to engineering design problems. *Advances in Engineering Software*, Elsevier, v. 112, p. 231–254, 2017.

LAUMOND, J.-P. et al. *Robot motion planning and control*. [S.l.]: Springer, 1998. v. 229.

LAVALLE, M.; KUFFNER, S. J. J. Rapidly-exploring random trees: Progress and prospects. In: *Proc. Workshop on the Algorithmic Foundations of Robotics*. San Francisco: [s.n.], 2000.

LAVALLE, S. *Planning algorithms. Cambridge, UK: Cambridge University Press*. 2006.

LAVALLE, S. M. Rapidly-exploring random trees: A new tool for path planning. Citeseer, 1998.

LAVALLE, S. M. Rapidly-exploring random trees: A new tool for path planning. Citeseer, 1998.

LAVALLE, S. M. *Planning algorithms*. [S.l.]: Cambridge university press, 2006.

LEE, T.; LEOK, M.; MCCLAMROCH, N. H. Geometric tracking control of a quadrotor uav on se(3). In: *49th IEEE Conference on Decision and Control (CDC)*. [S.l.: s.n.], 2010. p. 5420–5425.

LEE, T.-K.; BAEK, S.; OH, S.-Y. Sector-based maximal online coverage of unknown environments for cleaning robots with limited sensing. *Robotics and Autonomous Systems*, Elsevier, v. 59, n. 10, p. 698–710, 2011.

LEE, W.; PARK, G.; JOE, I. UAV path planning based on reinforcement learning for fair resource allocation in UAV-relayed cellular networks. In: *Information Science and Applications*. [S.l.]: Springer, 2020. p. 53–63.

LEI, W. et al. An improved artificial potential field for unmanned aerial vehicles path planning. *DEStech Transactions on Computer Science and Engineering*, n. cst, 2017.

LETIZIA, N. A.; SALAMAT, B.; TONELLO, A. M. A novel recursive smooth trajectory generation method for unmanned vehicles. *IEEE Transactions on Robotics*, IEEE, 2021.

LI, B. Y. et al. On 3d autonomous delivery systems: Design and development. In: IEEE. *2017 International Conference on Advanced Robotics and Intelligent Systems (ARIS)*. [S.l.], 2017. p. 1–6.

LI, D.; LU, M. Classical planning model-based approach to automating construction planning on earthwork projects. *Computer-Aided Civil and Infrastructure Engineering*, Wiley Online Library, v. 34, n. 4, p. 299–315, 2019.

LI, J.; YANG, S. X.; XU, Z. A survey on robot path planning using bio-inspired algorithms. In: IEEE. *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. [S.l.], 2019. p. 2111–2116.

LI, K. et al. Path planning of multiple UAVs with online changing tasks by an orpfoa algorithm. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 94, p. 103807, 2020.

LI, R. C. et al. Collision avoidance with deep reinforcement learning. 2019.

LI, W. An improved artificial potential field method based on chaos theory for UAV route planning. In: IEEE. *2019 34rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. [S.l.], 2019. p. 47–51.

LI, W. et al. A cubic spline method combing improved particle swarm optimization for robot path planning in dynamic uncertain environment. *International Journal of Advanced Robotic Systems*, SAGE Publications Sage UK: London, England, v. 17, n. 1, p. 1729881419891661, 2020.

LI, Y. An rrt-based path planning strategy in a dynamic environment. In: IEEE. *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*. [S.l.], 2021. p. 1–5.

LIFEN, L. et al. Path planning for UAVS based on improved artificial potential field method through changing the repulsive potential function. In: IEEE. *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*. [S.l.], 2016. p. 2011–2015.

LIN, N. et al. A novel improved bat algorithm in UAV path planning. *Journal of Computers, Materials & Continua*, 2019.

LISON, P. An introduction to machine learning. *Language Technology Group: Edinburgh, UK*, 2015.

LLASAG, R. et al. Human detection for search and rescue applications with UAVs and mixed reality interfaces. In: IEEE. *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)*. [S.l.], 2019. p. 1–6.

LONG, W. et al. An exploration-enhanced grey wolf optimizer to solve high-dimensional numerical optimization. *Engineering Applications of Artificial Intelligence*, Elsevier, v. 68, p. 63–80, 2018.

LONG, W. et al. Inspired grey wolf optimizer for solving large-scale function optimization problems. *Applied Mathematical Modelling*, Elsevier, v. 60, p. 112–126, 2018.

LOPEZ, J. L. S. et al. A real-time 3d path planning solution for collision-free navigation of multirotor aerial robots in dynamic environments. *Journal of Intelligent & Robotic Systems*, Springer, v. 93, n. 1-2, p. 33–53, 2019.

LU, Y. et al. A survey on vision-based UAV navigation. *Geo-spatial information science*, Taylor & Francis, v. 21, n. 1, p. 21–32, 2018.

LUMELSKY, V. J.; STEPANOV, A. A. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, Springer, v. 2, n. 1-4, p. 403–430, 1987.

MALYUTA, D. et al. Long-duration fully autonomous operation of rotorcraft unmanned aerial systems for remote-sensing data acquisition. *Journal of Field Robotics*, Wiley Online Library, v. 37, n. 1, p. 137–157, 2020.

MARKET, T. *Transparency Market - Inspection UAVs*. [S.l.], 2021. Disponível em: ⟨https://www.transparencymarketresearch.com/⟩.

MCINTYRE, D.; NAEEM, W.; XU, X. Cooperative obstacle avoidance using bidirectional artificial potential fields. In: IEEE. *2016 UKACC 11th International Conference on Control (CONTROL)*. [S.l.], 2016. p. 1–6.

MCKINLEY, S.; LEVINE, M. Cubic spline interpolation. *College of the Redwoods*, v. 45, n. 1, p. 1049–1060, 1998.

MEYER, J. et al. Comprehensive simulation of quadrotor UAVs using ros and gazebo. In: . [S.l.: s.n.], 2012. v. 7628, p. 400–411. ISBN 978-3-642-34326-1.

MICHAL, D. S. *A comparison of development environments for mobile autonomous robots: Player/Stage/Gazebo vs. Microsoft robotics developer studio*. [S.l.]: The University of Alabama in Huntsville, 2010.

MOKHTARI, K.; WAGNER, A. R. *The Pedestrian Patterns Dataset*. 2020.

MOKHTARI, K.; WAGNER, A. R. Don't get yourself into trouble! risk-aware decision-making for autonomous vehicles. *arXiv preprint arXiv:2106.04625*, 2021.

MOKHTARI, K.; WAGNER, A. R. *Pedestrian Collision Avoidance for Autonomous Vehicles at Unsignalized Intersection Using Deep Q-Network*. 2021.

MONDAL, S. et al. Autonomous architecture for UAV-based agricultural survey. In: *AIAA Scitech 2020 Forum*. [S.l.: s.n.], 2020. p. 2298.

NACHUM, O.; NOROUZI, M.; SCHUURMANS, D. Improving policy gradient by exploring under-appreciated rewards. *arXiv preprint arXiv:1611.09321*, 2016.

NACHUM, O. et al. Bridging the gap between value and policy based reinforcement learning. In: *Advances in Neural Information Processing Systems*. [S.l.: s.n.], 2017. p. 2775–2785.

NGUYEN, A. et al. A visual real-time fire detection using single shot multibox detector for uav-based fire surveillance. In: IEEE. *2020 IEEE Eighth International Conference on Communications and Electronics (ICCE)*. [S.l.], 2021. p. 338–343.

NOBIS, F. et al. Persistent map saving for visual localization for autonomous vehicles: An orb-slam 2 extension. In: IEEE. *2020 Fifteenth International Conference on Ecological Vehicles and Renewable Energies (EVER)*. [S.l.], 2020. p. 1–9.

NOORI, F. M. et al. On 3d simulators for multi-robot systems in ros: Morse or gazebo? In: IEEE. *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. [S.l.], 2017. p. 19–24.

NOREEN, I. et al. A path-planning performance comparison of RRT*-AB with MEA* in a 2-dimensional environment. *Symmetry*, Multidisciplinary Digital Publishing Institute, v. 11, n. 7, p. 945, 2019.

NOREEN, I.; KHAN, A.; HABIB, Z. A comparison of RRT, RRT* and RRT*-smart path planning algorithms. *International Journal of Computer Science and Network Security (IJCSNS)*, International Journal of Computer Science and Network Security, v. 16, n. 10, p. 20, 2016.

NOSRATI, M.; KARIMI, R.; HASANVAND, H. A. Investigation of the*(star) search algorithms: Characteristics, methods and approaches. *World Applied Programming*, Citeseer, v. 2, n. 4, p. 251–256, 2012.

PAN, J.-S.; LIU, J.-L.; LIU, E.-J. Improved whale optimization algorithm and its application to ucav path planning problem. In: SPRINGER. *International Conference on Genetic and Evolutionary Computing*. [S.l.], 2018. p. 37–47.

PANDA, M.; DAS, B. Grey wolf optimizer and its applications: A survey. In: SPRINGER. *Proceedings of the Third International Conference on Microelectronics, Computing and Communication Systems*. [S.l.], 2019. p. 179–194.

PANDEY, P.; SHUKLA, A.; TIWARI, R. Three-dimensional path planning for unmanned aerial vehicles using glowworm swarm optimization algorithm. *International Journal of System Assurance Engineering and Management*, Springer, v. 9, n. 4, p. 836–852, 2018.

PATLE, B. et al. Path planning in uncertain environment by using firefly algorithm. *Defence technology*, Elsevier, v. 14, n. 6, p. 691–701, 2018.

PATLE, B. et al. A review: On path planning strategies for navigation of mobile robot. *Defence Technology*, Elsevier, 2019.

PEREZ-GRAU, F. J. et al. An architecture for robust UAV navigation in gps-denied areas. *Journal of Field Robotics*, Wiley Online Library, v. 35, n. 1, p. 121–145, 2018.

POPOVIĆ, M. et al. Online informative path planning for active classification using UAVs. In: IEEE. *2017 IEEE international conference on robotics and automation (ICRA)*. [S.l.], 2017. p. 5753–5758.

POPOVIĆ, M. et al. Multiresolution mapping and informative path planning for UAV-based terrain monitoring. In: IEEE. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [S.l.], 2017. p. 1382–1388.

POPOVIĆ, M. et al. An informative path planning framework for UAV-based terrain monitoring. *Autonomous Robots*, Springer Nature BV, v. 44, n. 6, p. 889–911, 2020.

PRIMATESTA, S.; GUGLIERI, G.; RIZZO, A. A risk-aware path planning strategy for UAVs in urban environments. *Journal of Intelligent & Robotic Systems*, Springer, v. 95, n. 2, p. 629–643, 2019.

PULITI, S. et al. A comparison of UAV laser scanning, photogrammetry and airborne laser scanning for precision inventory of small-forest properties. *Forestry: An International Journal of Forest Research*, Oxford University Press, v. 93, n. 1, p. 150–162, 2020.

QU, C. et al. A novel hybrid grey wolf optimizer algorithm for unmanned aerial vehicle (UAV) path planning. *Knowledge-Based Systems*, Elsevier, p. 105530, 2020.

QU, C. et al. A novel reinforcement learning based grey wolf optimizer algorithm for unmanned aerial vehicles (UAVs) path planning. *Applied Soft Computing*, Elsevier, v. 89, p. 106099, 2020.

QUIGLEY, M. et al. Ros: an open-source robot operating system. In: KOBE, JAPAN. *ICRA workshop on open source software*. [S.l.], 2009. v. 3, n. 3.2, p. 5.

RAJA, P.; PUGAZHENTHI, S. Optimal path planning of mobile robots: A review. *International journal of physical sciences*, Citeseer, v. 7, n. 9, p. 1314–1320, 2012.

ROCHA, L. et al. A uav global planner to improve path planning in unstructured environments. In: *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. [S.l.: s.n.], 2021. p. 688–697.

ROCHA, L.; VIVALDINI, K. Comparison between meta-heuristic algorithms for path planning. In: *Anais do VIII Workshop de Teses e Dissertações em Robótica/Concurso de Teses e Dissertações em Robótica*. Porto Alegre, RS, Brasil: SBC, 2020. p. 1–10. ISSN 0000-0000. Disponível em: ⟨https://sol.sbc.org.br/index.php/wtdr_ctdr/article/view/14950⟩.

SAHA, O.; DASGUPTA, P. Real-time robot path planning around complex obstacle patterns through learning and transferring options. In: IEEE. *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. [S.l.], 2017. p. 278–283.

SAM, L. et al. UAV imaging of small caves in icelandic lava field as possible mars analogues. *LPICo*, v. 2197, p. 1053, 2020.

SANKARARAMAN, S.; GOEBEL, K. Computational architecture for autonomous decision-making in unmanned aerial vehicles. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *Micro-and Nanotechnology Sensors, Systems, and Applications X*. [S.l.], 2018. v. 10639, p. 106391Y.

SANTANA, L. V. et al. A trajectory tracking and 3d positioning controller for the ar. drone quadrotor. In: IEEE. *2014 international conference on unmanned aircraft systems (ICUAS)*. [S.l.], 2014. p. 756–767.

SAXENA, P. et al. Three dimensional route planning for multiple unmanned aerial vehicles using salp swarm algorithm. *arXiv preprint arXiv:1911.10519*, 2019.

SCHMID, L. et al. An efficient sampling-based method for online informative path planning in unknown environments. *IEEE Robotics and Automation Letters*, IEEE, v. 5, n. 2, p. 1500–1507, 2020.

SHARMA, V. D.; TOKEKAR, P. Risk-aware path planning for ground vehicles using occluded aerial images. *arXiv preprint arXiv:2104.11709*, 2021.

SHE, R.; OUYANG, Y. Efficiency of uav-based last-mile delivery under congestion in low-altitude air. *Transportation Research Part C: Emerging Technologies*, Elsevier, v. 122, p. 102878, 2021.

SHI, Z.; NG, W. K. A collision-free path planning algorithm for unmanned aerial vehicle delivery. In: IEEE. *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. [S.l.], 2018. p. 358–362.

SICHKAR, V. N. Reinforcement learning algorithms in global path planning for mobile robot. In: IEEE. *2019 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM)*. [S.l.], 2019. p. 1–5.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. *Introduction to autonomous mobile robots*. [S.l.]: MIT press, 2011.

SONG, J. et al. The high-speed rotorcraft unmanned aerial vehicle path planning based on the biogeography-based optimization algorithm. *Advances in Mechanical Engineering*, SAGE Publications Sage UK: London, England, v. 11, n. 5, p. 1687814019847863, 2019.

SONG, Q. et al. Dynamic path planning for unmanned vehicles based on fuzzy logic and improved ant colony optimization. *IEEE Access*, IEEE, 2020.

SOTNIK, S. et al. Some features of route planning as the basis in a mobile robot. *International Journal*, v. 8, n. 5, 2020.

SOUSA, S. K. A. d. et al. Planejamento de movimento para robôs móveis baseado em uma representação compacta da rapidly-exploring random tree (RRT). Universidade Federal de Sergipe, 2017.

STEINMETZ, J. A. *Mooring dirigible aircraft*. [S.l.]: Google Patents, 1927. US Patent 1,634,964.

SUDHAKAR, S. et al. Unmanned aerial vehicle (UAV) based forest fire detection and monitoring for reducing false alarms in forest-fires. *Computer Communications*, Elsevier, v. 149, p. 1–16, 2020.

SUTTON, R. S. et al. Policy gradient methods for reinforcement learning with function approximation. In: *Advances in neural information processing systems*. [S.l.: s.n.], 2000. p. 1057–1063.

TAN, J. et al. The 3d path planning based on A* algorithm and artificial potential field for the rotary-wing flying robot. In: IEEE. *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*. [S.l.], 2016. v. 2, p. 551–556.

TENEZACA, B. D. et al. Implementation of dubin curves-based RRT* using an aerial image for the determination of obstacles and path planning to avoid them during displacement of the mobile robot. In: *Developments and Advances in Defense and Security*. [S.l.]: Springer, 2020. p. 205–215.

THARWAT, A. et al. Intelligent bézier curve-based path planning model using chaotic particle swarm optimization algorithm. *Cluster Computing*, Springer, v. 22, n. 2, p. 4745–4766, 2019.

THRUN, S. et al. A system for volumetric robotic mapping of abandoned mines. In: IEEE. *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*. [S.l.], 2003. v. 3, p. 4270–4275.

TOMA, A.-I. et al. Pathbench: A benchmarking platform for classical and learned path planning algorithms. *arXiv preprint arXiv:2105.01777*, 2021.

TOMA, A.-I. et al. Waypoint planning networks. *arXiv preprint arXiv:2105.00312*, 2021.

TSIATSIS, V. et al. *Internet of Things: technologies and applications for a new age of intelligence*. [S.l.]: Academic Press, 2018. 299-305 p.

TUBA, E. et al. Optimal path planning in environments with static obstacles by harmony search algorithm. In: SPRINGER. *International Conference on Harmony Search Algorithm*. [S.l.], 2019. p. 186–193.

VALLEJO, D. et al. A multi-agent architecture for multi-robot surveillance. In: SPRINGER. *International Conference on Computational Collective Intelligence*. [S.l.], 2009. p. 266–278.

VANDENBERG, D. J.; SWEARS, B. S. *Systems and methods for autonomous vehicle operator vigilance management*. [S.l.]: Google Patents, 2020. US Patent 10,611,384.

VEMPATI, A. S. et al. Paintcopter: An autonomous UAV for spray painting on three-dimensional surfaces. *IEEE Robotics and Automation Letters*, IEEE, v. 3, n. 4, p. 2862–2869, 2018.

VEMPATI, A. S. et al. A virtual reality interface for an autonomous spray painting UAV. *IEEE Robotics and Automation Letters*, IEEE, v. 4, n. 3, p. 2870–2877, 2019.

VIVALDINI, K. et al. UAV route planning for active disease classification. *Autonomous Robots*, v. 43, 07 2018.

VIVALDINI, K. C. et al. Robotic forklifts for intelligent warehouses: Routing, path planning, and auto-localization. In: IEEE. *2010 IEEE International Conference on Industrial Technology*. [S.l.], 2010. p. 1463–1468.

WANG, C. et al. Efficient autonomous exploration with incrementally built topological map in 3d environments. *IEEE Transactions on Instrumentation and Measurement*, IEEE, 2020.

WANG, C. et al. Autonomous navigation of UAV in large-scale unknown complex environment with deep reinforcement learning. In: IEEE. *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. [S.l.], 2017. p. 858–862.

WANG, G. et al. *Unmanned Aerial Vehicles Path Planning Based on Deep Reinforcement Learning*. [S.l.]: Springer Nature, 2019. v. 1. 81-88 p.

WANG, H. et al. An improved RRT based 3-d path planning algorithm for UAV. In: IEEE. *2019 Chinese Control And Decision Conference (CCDC)*. [S.l.], 2019. p. 5514–5519.

WASKOW, S. J. Reinforcement learning using tile coding in multi agent scenarios. 2010.

WEISS, S. et al. Intuitive 3d maps for mav terrain exploration and obstacle avoidance. *Journal of Intelligent & Robotic Systems*, Springer, v. 61, n. 1-4, p. 473–493, 2011.

WU, C. et al. UAV autonomous target search based on deep reinforcement learning in complex disaster scene. *IEEE Access*, IEEE, v. 7, p. 117227–117245, 2019.

XIE, R. et al. Unmanned aerial vehicle path planning algorithm based on deep reinforcement learning in large-scale and dynamic environments. *IEEE Access*, IEEE, v. 9, p. 24884–24900, 2021.

XIN, J. et al. An improved genetic algorithm for path-planning of unmanned surface vehicle. *Sensors*, Multidisciplinary Digital Publishing Institute, v. 19, n. 11, p. 2640, 2019.

XU, Z.; DENG, D.; SHIMADA, K. Autonomous uav exploration of dynamic environments via incremental sampling and probabilistic roadmap. *IEEE Robotics and Automation Letters*, IEEE, v. 6, n. 2, p. 2729–2736, 2021.

YAN, C.; XIANG, X.; WANG, C. Towards real-time path planning through deep reinforcement learning for a UAV in dynamic environments. *Journal of Intelligent & Robotic Systems*, Springer, p. 1–13, 2019.

YAO, Z. Rimjump*: Tangent based shortest path planning for cluttered 3d environment. Preprints, 2021.

YERSHOVA, A. et al. Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. In: IEEE. *Proceedings of the 2005 IEEE international conference on robotics and automation*. [S.l.], 2005. p. 3856–3861.

YUE, X.; ZHANG, W. UAV path planning based on k-means algorithm and simulated annealing algorithm. In: IEEE. *2018 37th Chinese Control Conference (CCC)*. [S.l.], 2018. p. 2290–2295.

ZHANG, D.; XU, Y.; YAO, X. An improved path planning algorithm for unmanned aerial vehicle based on RRT-connect. In: IEEE. *2018 37th Chinese Control Conference (CCC)*. [S.l.], 2018. p. 4854–4858.

Zhang, M. et al. A high fidelity simulator for a quadrotor UAV using ros and gazebo. In: *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*. [S.l.: s.n.], 2015. p. 002846–002851.

ZHANG, T.; ZHANG, M.; ZOU, Y. Time-optimal and smooth trajectory planning for robot manipulators. *International Journal of Control, Automation and Systems*, Springer, v. 19, n. 1, p. 521–531, 2021.

ZHANG, Z. et al. A novel real-time penetration path planning algorithm for stealth UAV in 3d complex dynamic environment. *IEEE Access*, IEEE, v. 8, p. 122757–122771, 2020.

ZHOU, X. et al. Improved bat algorithm for uav path planning in three-dimensional space. *IEEE Access*, IEEE, v. 9, p. 20100–20116, 2021.