UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA

MARINA GAUTÉRIO BIZZOTTO

PLANEJAMENTO DE TRAJETÓRIAS PARA QUADRICÓPTEROS UTILIZANDO O MÉTODO RRT

UBERLÂNDIA
2023

MARINA GAUTÉRIO BIZZOTTO

PLANEJAMENTO DE TRAJETÓRIAS PARA QUADRICÓPTEROS UTILIZANDO O MÉTODO RRT

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia Elétrica, da Universidade Federal de Uberlândia, como requisito para a obtenção do título de Bacharel em Engenharia de Controle e Automação.

Universidade Federal de Uberlândia – UFU
Faculdade de Engenharia Elétrica

Orientadora: Profª. Drª. Gabriela Vieira Lima

UBERLÂNDIA
2023

AGRADECIMENTOS

Agradeço à Deus por me abençoar e me fazer capaz de superar todos os obstáculos.

Agradeço à minha família, em especial aos meus pais, que sempre me apoiaram e me deram as melhores oportunidades, as quais, possibilitaram que eu chegasse até aqui.

Agradeço ao meu namorado, Lucas Russignoli, que esteve ao meu lado durante toda essa trajetória, me incentivando e me ajudando a enfrentar todas as adversidades.

Agradeço à minha orientadora, Gabriela Vieira Lima, pela oportunidade de aprendizado e por me auxiliar durante o trabalho.

Por fim, agradeço à Universidade Federal de Uberlândia, por me permitir adquirir todos os conhecimentos que carregarei durante toda minha vida profissional.

Resumo

Em busca de facilitar muitas atividades que podem ser complexas, perigosas e até mesmo impossíveis para humanos, os veículos aéreos não tripulados (VANTs) vêm ganhando espaço. Os VANTs podem ser remotamente pilotados ou autônomos e são usados em diversas situações, como lazer, uso militar, na agricultura e nas indústrias, como a inspeção de equipamentos em áreas classificadas, uma vez que são áreas com atmosfera explosiva e não são indicadas para trabalho humano contínuo.

Os VANTs autônomos encontram algumas dificuldades, já que sem ter alguém pilotando e não conhecendo o ambiente, eles precisam ter a capacidade de desviar de obstáculos, então surge a necessidade do estudo sobre o planejamento de trajetórias. Teoricamente, existem vários métodos que resolvem esse problema, com diferentes vantagens e desvantagens, muitos foram desenvolvidos apenas para ambientes 2D, em outros é essencial ter um conhecimento completo do mapa do ambiente. Além disso, torna-se necessário a realização de testes práticos para validação dos métodos, o que ainda não é uma realidade para todos.

O presente trabalho tem como objetivo entender esses métodos e se aprofundar no método escolhido denominado *Rapidly-Exploring Random Tree*. Para isso, são feitas simulações e testes reais para diferentes cenários, considerando um ambiente pré-determinado e obstáculos estáticos para validar o método e, futuramente, ser possível utilizá-lo em situações cotidianas.

Palavras-chave: VANTs, Planejamento de trajetórias, *Rapidly-Exploring Random Tree*.

Abstract

In an attempt to facilitate many activities that can be complex, dangerous and even impossible for humans, unmanned aerial vehicles (UAVs) have been gaining ground. UAVs can be remotely piloted or autonomous and are used in a variety of situations, such as leisure, military use, agriculture and industry, such as the inspection of equipment in classified areas, as these are areas with an explosive atmosphere and are not suitable for continuous human work.

Autonomous UAVs encounter some difficulties, since without someone piloting and not knowing the environment, they need to be able to avoid obstacles, so there is a need to study trajectory planning. Theoretically, there are several methods that solve this problem, with different advantages and disadvantages, many were developed only for 2D environments, in others it is essential to have complete knowledge of the environment map. Furthermore, it is necessary to carry out practical tests to validate the methods, which is not yet a reality for everyone.

The present work aims to understand these methods and delve deeper into the chosen method called Rapidly-Exploring Random Tree. To this end, simulations and real tests are carried out for different scenarios, considering a pre-determined environment and static obstacles to validate the method and, in the future, be able to use it in everyday situations.

***Keywords**: UAV, planning of trajectories, Rapidly-Exploring Random Tree.*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

GPS	*Global Positioning System* – Sistema de Posicionamento Global
PSO	*Particle Swarm Optimization* – Otimização de Enxame de Partículas
RPA	*Remotely Piloted Aircraft* – Aeronave Remotamente Pilotada
RRT	*Rapidly-Exploring Random Tree* – Árvore Aleatória de Exploração Rápida
TWR	*Two Way Ranging* – Em dois sentidos
UAV	*Unmanned Aerial Vehicles* - Veículo Aéreo Não Tripulado
UWB	*Ultra-wideband* – Banda Ultralarga
VANT	Veículo Aéreo Não Tripulado

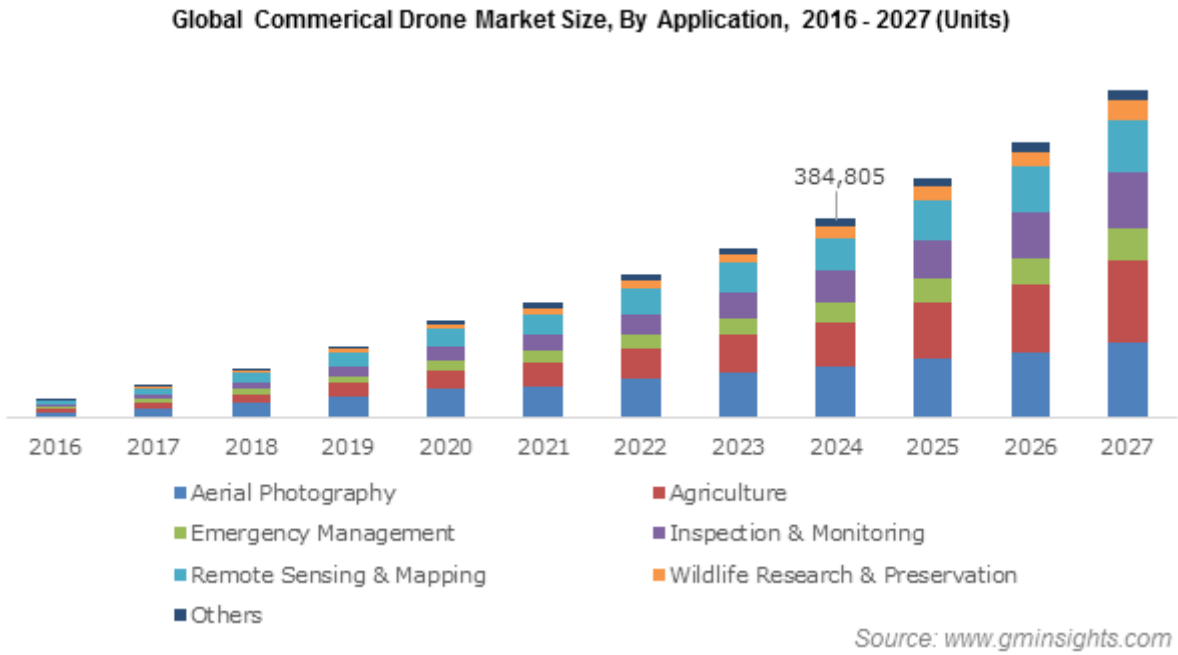
SUMÁRIO

1 INTRODUÇÃO

O termo VANT (veículo aéreo não tripulado) começou a aparecer durante a primeira guerra mundial, quando os austríacos enviaram balões explosivos não tripulados para atacar Veneza, porém, somente em 1951 surgiu o primeiro drone, Firebee, usado como alvo aéreo (UBIRATAN, 2015). No Brasil, o primeiro voo registrado de drone foi em 1983, com o BQM-1BR, construído pela CBT - Companhia Brasileira de Tratores, cujo objetivo também era de alvo aéreo (CALIXTO, 2023). Com o passar dos anos a tecnologia foi crescendo, possibilitando drones de tamanhos menores e autônomos.

Cada vez mais conhecidos, suas aplicações também estão se expandindo, podendo ser usados não somente para operações militares, mas também em indústrias, para sensoriamento e monitoramento em áreas de riscos ou de difícil acesso para humanos, em aplicações civis para buscas, resgates e perseguições, em mapeamento e monitoramento ambiental, agrícola, entre muitas outras áreas que surgem, isso pode ser visto na Fig. 1, assim como uma previsão para os próximos anos.

Figura 1 – Mercado global de drones



Fonte: Wadhwani e Loomba (2021).

A grande maioria dos VANTs ainda hoje são do tipo RPA (*Remotely-Piloted Aircraft*), ou seja, são remotamente pilotados, mas as aeronaves autônomas também vêm ganhando espaço, com uma dificuldade a mais que é planejar a trajetória de voo, já que possuem muitos obstáculos estáticos e

dinâmicos no ambiente. Existem seis níveis de autonomia para os VANTs, que podem ser vistos na Tabela 1:

Tabela 1 – Níveis de autonomia para os VANTs

Nível	Autonomia	Obstáculos	Piloto	Descrição
0	Não autônomo – RPA	Não consegue entender ou responder a obstáculos	Piloto está no comando 100% do tempo	Sem o piloto o VANT não irá funcionar
1	Baixa autonomia	Detecta e avisa	Piloto está no comando	Sem o piloto o VANT conseguirá permanecer no ar
2	Parcialmente autônomo	Detecta e avisa	Piloto está no comando	Sem o piloto o VANT consegue se estabilizar e detectar paredes
3	Condicionalmente autônomo	Detecta e evita os obstáculos	Piloto como reserva para o voo	Sem o piloto o VANT consegue voar e evitar paredes
4	Alta autonomia	Detecta e consegue navegar entre eles	Piloto consegue monitorar o voo remotamente	O VANT consegue explorar o ambiente e navegar em condições adversas
5	Total autonomia	Detecta e consegue navegar entre eles	Piloto irá definir o objetivo, mas não precisa monitorar o voo	O VANT consegue voar em qualquer ambiente sob todas as condições

Fonte: Adaptado de McNabb (2019).

Conforme mostrado na imagem, o nível 0 corresponde ao RPA, e vai aumentando a autonomia ao decorrer dos números chegando até o nível 5, que corresponde a total autonomia, o qual será usado para esse trabalho.

Os VANTs podem ser utilizados em diversos locais, alguns serão simples e só terão obstáculos estáticos, porém, alguns podem ser dinâmicos, como os que já foram mencionados, por isso é de extrema importância o planejamento de trajetórias.

Ao longo dos anos, muitos métodos foram criados com intuito de solucionar o problema de planejamento de trajetórias, muitos deles já foram testados e validados, no entanto, outras ainda estão somente na teoria. Como é um tema recente, até mesmo as teorias já validadas, continuam em estudo

para melhorias. Alguns métodos relevantes são, por exemplo, o algoritmo de otimização de enxame de partículas (PSO), o campo potencial artificial, o algoritmo A*, a perseguição e evasão orbital e, por fim, o método *Rapidly-exploring Random Tree* (RRT), o qual será mais explorado neste trabalho.

1.1 JUSTIFICATIVA

Diante do constante aumento no uso dos VANTs, é necessário o estudo dos drones em geral e do planejamento de trajetória, pois essa é uma área relativamente nova e que não tem uma solução ótima.

O planejamento de trajetórias é importante para que tarefas cotidianas possam ser realizadas por VANTs. Em sistemas de monitoramento, por exemplo, através da instalação de câmeras, o quadricóptero estará apto a inspecionar desde estoques industriais até verificar problemas em áreas de risco ou de difícil acesso humano. Outra atividade facilitada com o planejamento de trajetória são as tarefas de transporte que podem ser utilizadas em delivery, com auxílio de um sistema de posicionamento global.

Existem vários modelos matemáticos para resolver esse problema, portanto, é necessário estudá-los e testá-los, com o intuito de ampliar as documentações existentes para posteridade. Neste trabalho, o método utilizado para o planejamento de trajetórias será o método RRT. Com isso, os VANTs autônomos serão cada vez mais populares e disponíveis para realizar qualquer tipo de tarefa desejada.

1.2 OBJETIVO

O presente estudo propõe analisar os métodos de planejamento de trajetória aplicados a veículos aéreos não tripulados e aprofundar no método RRT, com objetivo de encontrar a trajetória entre um ponto inicial e final, desviando de obstáculo estáticos pré-determinados. Posteriormente, o método poderá ser utilizado para ambientes maiores e mais complexos como, por exemplo, em prédios, construções, indústrias e até mesmo em ambientes naturais como uma floresta.

O método RRT será implementado via simulações computacionais, utilizado o software Matlab, para buscar os pontos da trajetória entre o início e o objetivo final. Por fim, para validação, aplica-se a trajetória encontrada no VANT do modelo Crazyflie 2.1, dentro de um ambiente

determinado, onde o quadricóptero fará o teste de voo, cumprindo o trajeto e desviando dos obstáculos pré-determinados.

1.3 OBJETIVOS ESPECÍFICOS

Os objetivos específicos deste trabalho são:

- Realizar uma ampla revisão bibliográfica sobre os métodos de planejamento de trajetória;
- Estudar as técnicas de planejamento de trajetória utilizando o método RRT;
- Desenvolver simulações e validações utilizando o Matlab;
- Realizar ensaios experimentais utilizando o quadricóptero Crazyflie 2.1;
- Concluir as vantagens e desvantagens da utilização do método RRT para o planejamento de trajetórias.

1.4 ORGANIZAÇÃO DO TRABALHO

O trabalho será organizado em quatro capítulos, onde:

- No Capítulo 2 será apresentada a base teórica sobre os VANTs, sua representação, terminologias e o planejamento de trajetórias;
- No Capítulo 3 será apresentada a metodologia para o desenvolvimento do método escolhido no software Matlab e a preparação para os ensaios experimentais;
- No Capítulo 4 será detalhado os resultados para os diferentes cenários analisados;
- No Capítulo 5 serão apresentadas as conclusões e as oportunidades identificadas para trabalhos futuros.

2 REFERENCIAL TEÓRICO

Como fundamentação teórica, será apresentado o conceito de planejamento de trajetórias, assim como os métodos já conhecidos, dentre eles o método RRT. Além disso, serão abordadas as terminologias, informações gerais do VANT e a representação do robô nas simulações.

2.1 PLANEJAMENTO DE TRAJETÓRIAS

Planejamento, segundo o neurocientista M.Owen (1977), é “organizar o comportamento de alguém no espaço e no tempo em uma situação específica em que uma meta/objetivo deve ser alcançada”. Ainda segundo M.Owen (1977), existem 3 níveis de planejamento: estratégico, operacional e tático. Sendo que, no nível estratégico, seria o planejamento de rotas, no operacional, o planejamento de controle e, por fim, o nível tático seria o planejamento de trajetórias (BENSRHAIR et al., 2021).

O termo planejamento de trajetórias é, de acordo com uma posição inicial e a posição objetivo final, gerar a melhor trajetória verificando todos os fatores para essa situação, evitando obstáculos, seguindo todas as restrições e sendo eficiente, seguro e viável. Essa etapa de planejamento de trajetórias envolve diversas técnicas e algoritmos matemáticos que podem ser simples ou complexos e podem ser usados em diversos campos, como em veículos autônomos, aeroespacial, drones, computação gráfica (realidade virtual) e na manufatura (automação de processos industriais, como, por exemplo, braços robóticos).

Na robótica, planejar a trajetória é equivalente a gerar uma sequência de posições e orientações que o robô precisa seguir para alcançar a posição final pré-estabelecida. Os fatores que influenciam essa decisão variam de acordo com o ambiente - aberto ou fechado, com obstáculos, sendo eles dinâmicos ou estáticos e grau de conhecimento. É um campo muito importante pois permite que objetos naveguem em ambientes complexos e com restrições específicas.

Especificando esse processo para VANTs, existe uma dificuldade a mais, visto que o planejamento precisa ser feito em 3D, uma vez que o deslocamento no ambiente não se dá somente em duas dimensões. Existem vários métodos para fazer esses planejamentos, os quais serão discutidos posteriormente. Para esse projeto, o foco é planejar a trajetória para veículos aéreos não tripulados, considerando ambiente fechado com obstáculos estáticos através do método RRT.

2.1.1 TERMINOLOGIA

Para melhor compreensão, neste tópico será apresentado algumas terminologias que serão utilizadas ao longo deste trabalho.

Existem dois tipos de planejamento de trajetórias: *offline* e *online*. O planejamento *offline*, basicamente, gera o caminho para uma situação completamente conhecida, supondo um ambiente estático, onde irá sair de um ponto inicial conhecido, passar por obstáculos estáticos e chegar ao ponto final pré-estabelecido. Portanto, antes do voo é preciso carregar esse caminho para o VANT, não sendo possível alterações. Já no planejamento *online*, é possível criar o caminho enquanto o VANT se desloca, só é necessário o ponto inicial. O ponto final pode ser modificado assim como os obstáculos, que nesse caso podem ser estáticos ou dinâmicos.

O espaço de trabalho é todo o ambiente no qual o VANT pode se deslocar, neste trabalho será um ambiente tridimensional. A configuração do drone é um vetor contendo sua posição, o espaço de configuração é um conjunto com todas as configurações possíveis. O mapeamento do ambiente serve para representar no espaço de trabalho as áreas navegáveis e não navegáveis (com obstáculos).

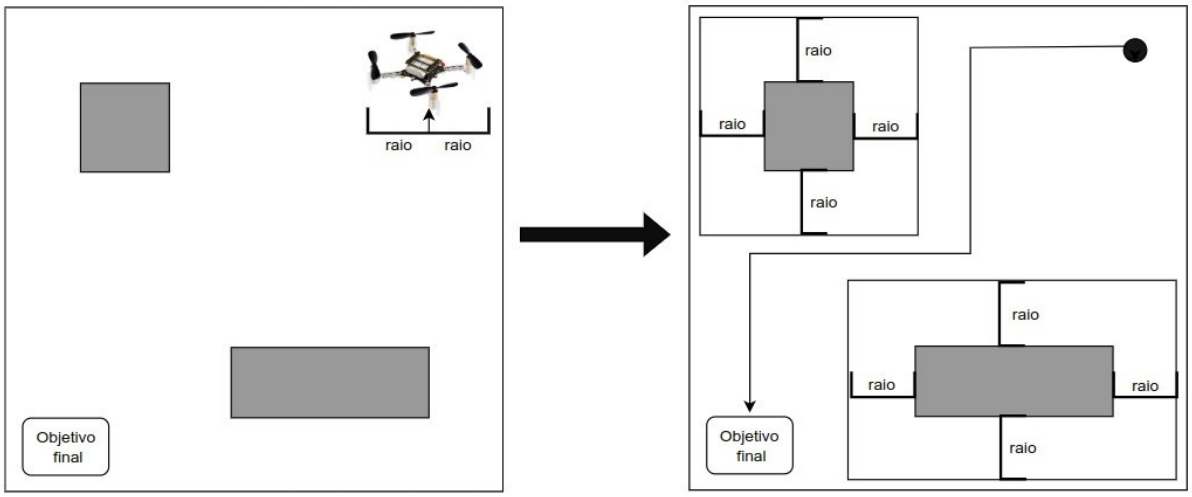
No planejamento de trajetórias, o termo caminho se refere a curva (sequência de posições) que o robô irá traçar no espaço de configuração. Já o termo trajetória, é mais completo e, além das posições, inclui velocidade, aceleração e outras variáveis de estado que variam ao longo do tempo.

2.1.2 REPRESENTAÇÃO DO ROBÔ

Importante ressaltar que, para simulações, o quadricóptero é representado como um objeto pontual, portanto é necessário considerar uma expansão dos obstáculos, uma vez que na realidade o drone não é pontual, para isso será aumentado pelo menos o tamanho do raio do drone aos obstáculos com o intuito de ter resultados mais confiáveis nas simulações e para os experimentos reais, pois assim levamos em conta qualquer erro de modelagem, e não corre o risco de o quadricóptero se colidir com os obstáculos nos testes reais.

Na Fig. 2, é possível observar como os objetos serão representados para não ocorrer nenhum erro entre os valores encontrados na simulação e nos testes reais. Nesse caso, foi feita uma borda do tamanho do raio do drone em volta dos obstáculos para evitar colisões.

Figura 2 - Representação do drone e obstáculos



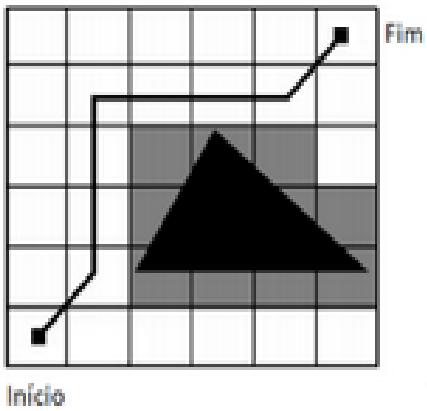
Fonte: A autora (2023).

2.2 MÉTODOS DE PLANEJAMENTO DE TRAJETÓRIA

2.2.1 DECOMPOSIÇÃO CELULAR APROXIMADA

No método da decomposição celular aproximada, o ambiente de trabalho (representado em 2D ou 3D) é dividido em formas geométricas para um espaço mais simples com células de mesmo tamanho, por exemplo quadrados. Após ter essa divisão, é possível fazer uma busca discreta, encontrar quais formatos estão disponíveis e encontrar a melhor trajetória por essas células livres. As células que tenham obstáculos, mesmo que ocupando uma pequena parte, são consideradas ocupadas, com isso, esse método perde informações já que elimina essas células.

Figura 3 – Decomposição Celular Aproximada



Fonte: Latombe (1991).

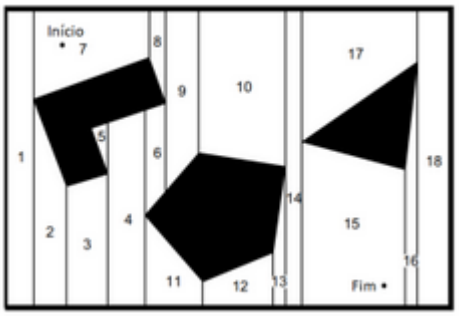
Primeiramente, é necessário representar o espaço de trabalho dividindo em uma grade de células, então identificar os obstáculos presentes e quais células estão ocupadas e quais estão livres, sendo possível utilizar algum outro método baseado em grade para planejar a trajetória e, por fim, otimizar e suavizar a solução encontrada.

Tem como vantagem ser um método que reduz a complexidade do problema de planejamento de trajetória e simplifica a busca pelo melhor trajeto. Como desvantagem, esse método simplifica o ambiente perdendo a precisão no trajeto, isso também é um problema quando tem dois obstáculos próximos e a passagem é estreita, ele pode entender que não há trajetória possível. Além disso, o método pode encontrar caminhos muito conservadores por ser um método discreto. Por fim, não é o indicado para usar em ambientes dinâmicos, visto que seria necessário refazer os cálculos constantemente, gerando uma complexidade computacional (LATOMBE,1991).

2.2.2 DECOMPOSIÇÃO CELULAR EXATA

No método da decomposição celular exata, o ambiente também é dividido em células, porém com mais precisão, para isso somente as partes sem obstáculos são divididas, garantindo a cobertura completa e não perdendo informações sobre o ambiente.

Figura 4 – Decomposição Celular Aproximada



Fonte: Latombe (1991).

A forma que esse método funciona é bem similar ao anterior, é feito a representação do ambiente de trabalho, os obstáculos são identificados, determinado quais células estão completamente ocupadas e o espaço restante é, novamente, divido em novas células chamadas de células de influência, por fim, é possível encontrar o melhor trajeto considerando todos os caminhos válidos sem perder espaço de atuação perto dos obstáculos.

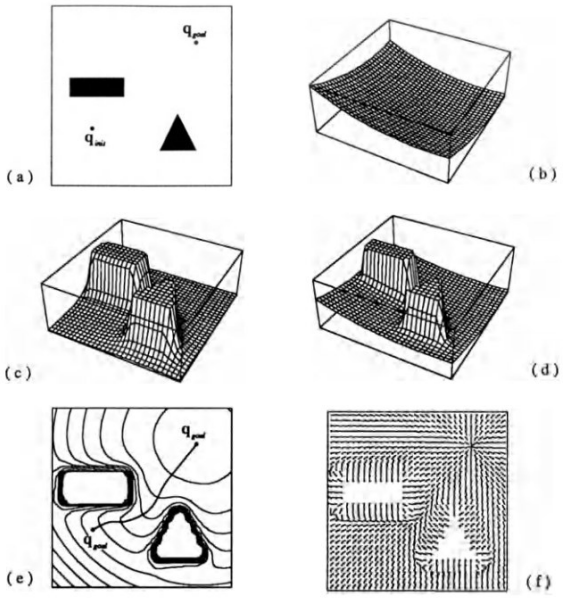
Esse é um método determinístico que é vantajoso, pois é muito eficiente quando se fala sobre representação do espaço de trabalho, porém, para realizar essa representação pode resultar em uma quantidade excessiva de células e gerar complexidade computacional para encontrar o melhor trajeto, o que pode se tornar uma desvantagem.

2.2.3 CAMPO POTENCIAL ARTIFICIAL

Esse método considera que as partículas estão sob influência de um campo potencial artificial, onde as forças atrativas aproximam o objeto do objetivo, pois a posição final apresenta um potencial atrativo. Já os obstáculos apresentam forças repulsivas, o que empurra o objeto para longe deles. O VANT irá simplesmente seguir do ponto menos atrativo ao mais atrativo, não sendo necessário uma criação prévia de nenhuma estrutura de dados para fazer essa análise no ambiente, podendo ser usado para ambientes dinâmicos. Se ocorrer qualquer mudança nos obstáculos esse ambiente deve ser atualizado (CHEN et al., 2005).

A Fig. 5(a) representa a configuração de espaço em duas dimensões para dois obstáculos. Na Fig. 5(b) mostra o campo potencial atrativo, na Fig. 5(c) mostra o campo potencial repulsivo, já na Fig. 5(d) é a soma dos dois campos potenciais. Por fim, a Fig. 5(e) exibe os contornos equipotenciais e o caminho gerado seguindo o gradiente negativo dessa função, e a Fig. 5(f) mostra a matriz de vetores gradientes negativos orientados ao espaço livre (LATOMBE, 1991).

Figura 5 – Campo potencial artificial



Fonte: Latombe (1991).

Tem como vantagem ser um método efetivo para ambientes dinâmicos que tenham poucos obstáculos e que seja possível encontrar caminho mais direto ao objetivo. Como desvantagem, o método é totalmente dependente dos parâmetros definidos inicialmente, como intensidade das forças atrativas e repulsivas e a influência dos obstáculos, portanto, em passagens estreitas pode ocorrer de só ter forças repulsivas, impedindo o VANT de atravessar. Além disso, é um método desenvolvido para ambientes mais simples, em ambientes complexos essa abordagem pode não ser suficiente para encontrar a trajetória. Por fim, outra desvantagem é que o VANT pode ficar preso em mínimos locais (DONG et al., 2012).

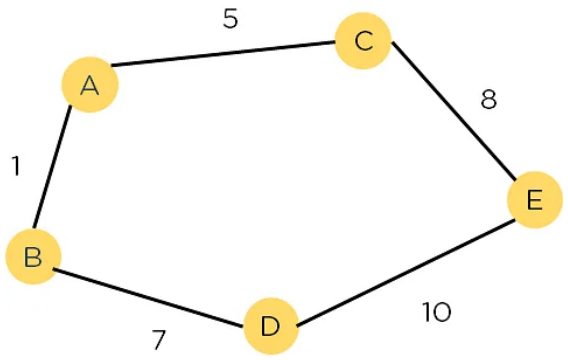
2.2.4 ALGORITMO A*

O algoritmo A* (A-*star*) é usado para encontrar o caminho mais curto entre o nó inicial e o nó final considerando o custo para essa trajetória, analisando distância e o tempo. A distância entre o ponto inicial e o ponto escolhido (n) é calculada e chamada de custo g(n). Então é feito o mesmo cálculo, mas, dessa vez entre o ponto escolhido (n) e o ponto final, esse será o custo heurístico, chamado de h(n). Portanto, o ponto que será eleito para fazer parte da trajetória é o que tiver o menor valor do custo final f(n), ou seja, o menor valor na soma dos custos g(n) e h(n) (PATEL, 1997).

Com os custos calculados, é criada uma fila de prioridade, que começa pelo nó inicial e, a cada nova iteração, um novo nó com menor custo é escolhido para ser adicionada a essa fila. Com todos esses nós na fila de prioridade, é utilizada a curva Reeds-Shepp (RS) para conectar esse nó com o nó destino, se for uma trajetória livre de colisão então o algoritmo A* irá escolher esse caminho e determinar a velocidade para todo o trajeto (SHENG et al., 2021).

Inicialmente, funciona como um problema de travessia de grafos, assim um robô poderia encontrar sozinho que caminho seguir. Para isso, ele utiliza grafos ponderados, ou seja, um grafo que possui uma função no qual relaciona o conjunto de vértices ou de arestas com um conjunto de números, e esses números irão representar o custo de seguir cada um dos caminhos encontrados (RAVIKIRAN et al., 2023).

Figura 6 – Grafo ponderado



Fonte: Ravikiran, A., S. (2023).

Esse algoritmo é muito utilizado para resolver problemas que envolvam encontrar caminhos em videogames, isso porque ele só segue adiante se tiver certeza de que aquele é o melhor caminho de acordo com suas funções, portanto, a grande vantagem desse algoritmo é que ele consegue encontrar a trajetória ideal (CHATTERJEE, 2023).

A maior desvantagem desse método é que por ser um algoritmo completo, que encontra todos os caminhos possíveis, necessita de muito espaço de armazenamento, além de gastar muito tempo nesta busca (RAVIKIRAN et al., 2023).

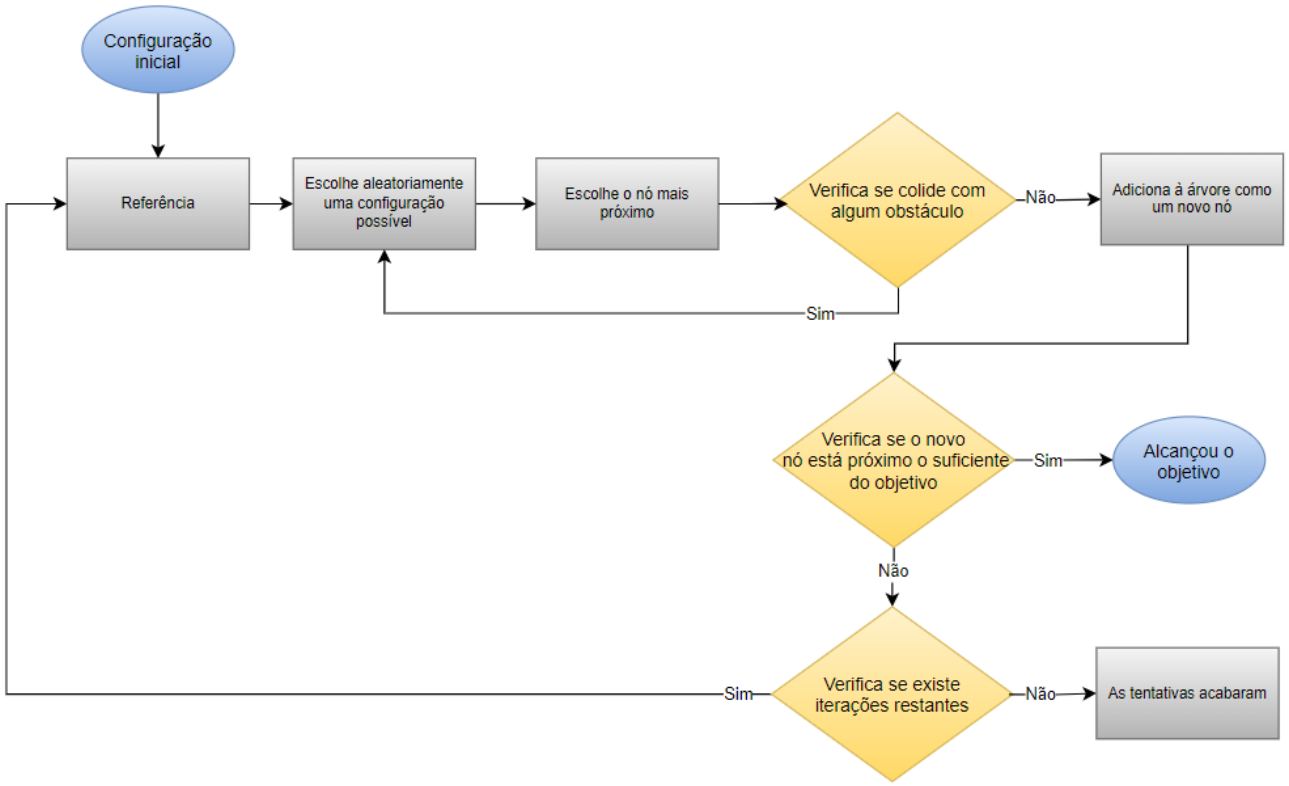
2.2.5 RAPIDLY-EXPLORING RANDOM TREE

O método *Rapidly-Exploring Random Tree* surgiu em 1998 no artigo "*Rapidly-exploring random trees: A new tool for path planning*" de Steven M. LaValle. Desde então, vem ganhando popularidade por ser um método que resolve problemas de alta complexidade do ambiente e consegue encontrar diversos caminhos diferentes. Nos anos seguintes, várias pesquisas foram feitas com intuito de implementar melhorias ou até de desenvolver variações desse método, como, por exemplo, para enfrentar algum problema específico ou para simplesmente melhorar algum aspecto como desempenho, eficiência, otimização ou convergência.

O *Rapidly-Exploring Random Tree* (RRT) é um método de planejamento de caminho probabilístico baseado em amostragem, foi criado para que o objeto consiga navegar em ambientes complexos e de alta dimensão. Para isso, ele gera uma trajetória do ponto inicial até o ponto final, olhando um pequeno espaço e gerando nós (como os ramos das árvores), a partir disso ele escolhe, pensando no objetivo final, qual melhor ramo seguir e esse processo se repete até ele chegar ao ponto

final. Ou seja, tende a se deslocar até a posição final, buscando o caminho mais curto evitando colisão com os obstáculos, um fluxograma com o detalhamento do funcionamento do método se encontra na Fig. 7.

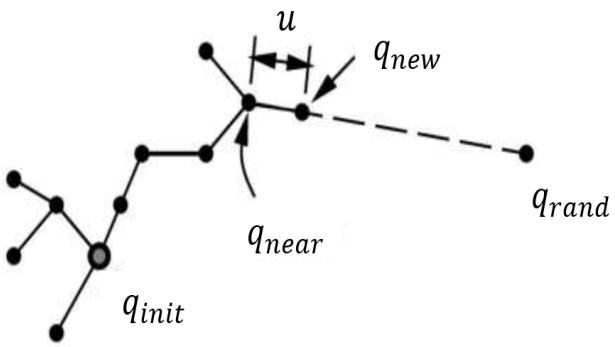
Figura 7 – Fluxograma método RRT



Fonte: A autora (2023).

Na Fig. 8 tem-se um exemplo no qual o ponto inicial (*qinit*), a cada iteração define-se um vértice (*qrand*) de forma aleatória e escolhe-se o nó mais próximo a este ponto (*qnear*). A distância entre o vértice e o nó mais próximo é minimizada pelo fator u (valor máximo de crescimento da árvore). Se o caminho escolhido estiver livre de obstáculos, um novo ponto é gerado (qnew). O processo se repete até que se execute o número máximo de iterações pré-estabelecidas ou até que se alcance o ponto final (*qend*) (LAVALLE, 2006). Os estados do RRT representam as configurações do VANT.

Figura 8 - Método RRT



Fonte: Latombe (1998).

A cada iteração, uma nova configuração é criada e, consequentemente, os nós da árvore. Essa configuração é gerada de acordo com um raio de busca e a quantidade de pontos desejados, essas informações precisam ser determinadas anteriormente, todos os nós gerados são configurações válidas para o objeto seguir e se aproximar do ponto final. Após o objeto estar em um ponto e já ter todas as opções que ele pode seguir (nós), o algoritmo precisa calcular qual melhor configuração seguir, para isso, ele compara a distância entre o ponto que ele se encontra até o destino com a nova distância entre o *qnew* e o objetivo final, esse processo é feito para todos os pontos escolhidos. O pseudocódigo desse processo é evidenciado na Fig. 9.

Figura 9 - Algoritmo RRT

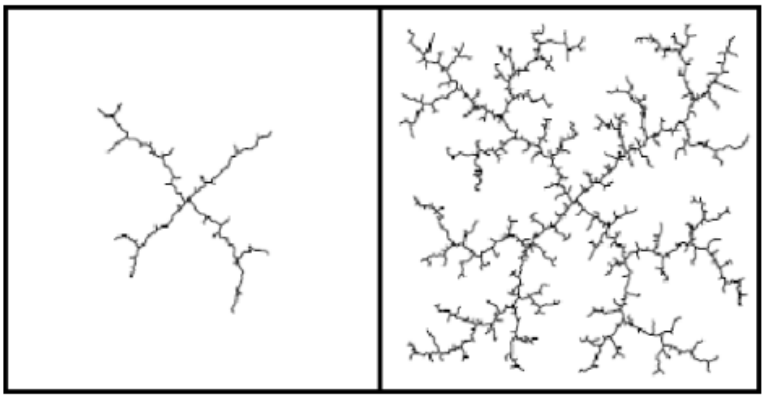
Algoritmo 1

```
1:  função GERA RRT(q_init, K, Δt)
2:      RRT.init(q_init)
3:      k = 1
4:      enquanto k ⩽ K faça
5:          q_rand ← Gera_Estado_Aleatorio()
6:          q_near ← Verifica_Mais_Proximo(q_rand, RRT)
7:          q_new ← Gera_Novo_Estado(q_near, Δt)
8:          se Verifica_Colisao = Livre então
9:              RRT.add_no(q_new)
10:         fim se
11:         k = k + 1
12:     fim enquanto
13:     devolve RRT
14: fim função
```

Fonte: Sousa (2017).

É um método muito útil, pois a cada iteração os ramos se espalham por todo ambiente, conseguindo explorar todos os cantos. Um exemplo de como funciona o crescimento da árvore de exploração pode ser observado na Fig. 10, considerando um ambiente 2D sem obstáculos (CHAIMOWICZ et al., 2007).

Figura 10 – Crescimento de uma RRT



Fonte: Chaimowicz (2007).

O intuito de utilizar um raio de busca auxilia o método a buscar pontos em uma área menor, aumentando a confiabilidade, além de que separa o ambiente em "vizinhanças" em torno de cada ponto referência, promovendo a diversidade, evitando a geração de pontos repetidos durante a expansão da árvore e que o algoritmo fique preso em alguma região do ambiente (CHENG et al., 2002).

Um exemplo prático é em uma rede rodoviária, utilizando o método RRT, ele irá evitar áreas de congestionamento ou áreas que estejam com algum problema e escolherá o melhor caminho, tendendo a melhorar também o tempo que será gasto por usar um método de planejamento dinâmico (ZHENGHUA et al., 2017).

Algumas desvantagens da utilização desse método estão relacionadas ao caminho traçado, por exemplo, mesmo alcançando o objetivo final, isso não é uma garantia que o melhor trajeto foi escolhido, ele pode escolher um caminho mais longo caso o ambiente esteja desorganizado, ou então não ser o caminho mais suave. É também dependente dos parâmetros colocados nele, podendo deixar um tempo de conversão maior, ou não chegar ao objetivo final por falta de iterações ou um número errado no raio de busca. Por fim, existe também a possibilidade de cair em mínimos locais dependendo do ambiente ou da situação.

Apesar disso, é um método eficaz, pois é muito flexível sendo possível utilizar para diferentes veículos, muito eficiente evitando obstáculos, o planejamento da trajetória ocorre em tempo real, além de ser probabilístico então a cada iteração a probabilidade de encontrar o objetivo aumenta.

Outra vantagem é que serve para problemas de alta aleatoriedade de nós, repetibilidade, espaços de alta dimensões, complexos e é rápido já que esse método não precisa encontrar todos os caminhos válidos (SOUSA, 2017). Portanto, perante todos os pontos levantados, o método escolhido para esse trabalho foi o método RRT.

2.3 CRAZYFLIE 2.1

O VANT utilizado será o quadricóptero modelo Crazyflie 2.1 fabricado pela Bitcraze, que pode ser visto na Fig. 11:

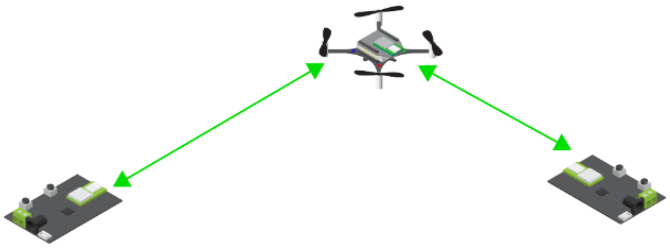
Figura 11 – Quadricóptero Crazyflie 2.1

Fonte: Bitcraze (2022).

A aeronave pesa 28 gramas, mede 92 mm de motor a motor e está equipado com rádio de baixa latência, longo alcance e bluetooth. Conta também com uma bateria LiPo de 3,7 V, acelerômetro, giroscópio e magnetômetro nos 3 eixos (IMC088) e sensor de pressão de alta precisão (BMP388). Opera com dois microcontroladores: STM32F405, o principal deles, utilizado para sensoriamento, controle e acionamento dos motores e o nRF51822, auxiliar, utilizado para gerenciamento de energia e comunicação via rádio. Possui 4 motores DC com velocidade máxima de aproximadamente 25000 rpm. Consegue fazer um voo de até 7 minutos com a bateria, com a capacidade de carga máxima de 15 gramas (BITCRAZE, 2022).

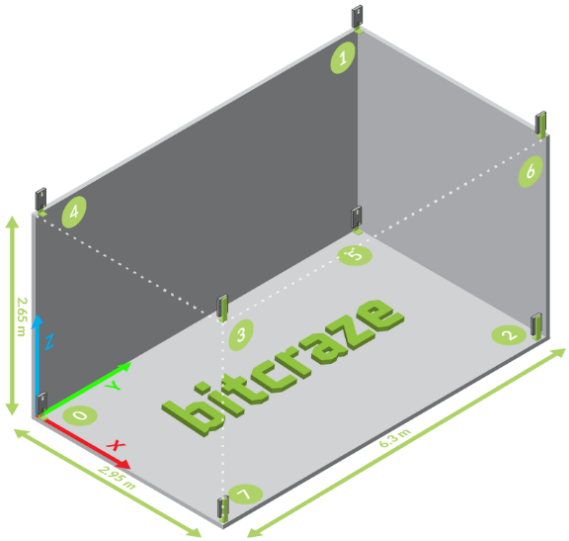
Para saber a posição do VANT no espaço tridimensional, mesmo ele possuindo sensores de bordo que dão uma estimativa da localização, não é a posição precisa. Para isso, é necessário adicionar um novo sistema, podendo ser GPS para ambientes abertos, ou então, para ambientes fechados, o sistema de posicionamento local de banda ultra larga (UWB) – o qual será utilizado nesse trabalho. Esse sistema funciona utilizando a troca de informação entre o VANT e as âncoras instaladas previamente no ambiente, o indicado é ter pelo menos 8 âncoras, que para esse trabalho foram fixadas em 4 hastes formando um paralelepípedo que pode ser visto na Fig. 12 e 13.

Figura 12 – Sistema de posicionamento local



Fonte: Bitcraze (2022).

Figura 13 – Posicionamento das âncoras



Fonte: Bitcraze (2022).

Portanto, as âncoras servem de referência e, de acordo com o conjunto de informações recebido pela troca de mensagens via rádio, é possível saber a posição do quadricóptero no espaço tridimensional – com um erro de 10 cm na medição, segundo o fabricante. É importante manter uma distância entre as âncoras e as paredes e teto para garantir uma melhor difusão do sinal de rádio, a

recomendação do fabricante Bitcraze é que essa distância seja de pelo menos 15 cm. As âncoras precisam ser alimentadas por uma fonte de alimentação externa e o receptor é acoplado a aeronave (BITCRAZE, 2022).

A comunicação via rádio possui um amplificador de potência de 20dBm, o qual possibilita um alcance de até 1 km. Vem com firmware compatível com o quadricóptero, de código aberto e tem uma interface de programação em python, o que facilita fazer modificações. Conectando essa antena ao USB do computador e instalando o driver Zadig, recomendado pelo site do fabricante, a comunicação via rádio está habilitada para ser utilizada.

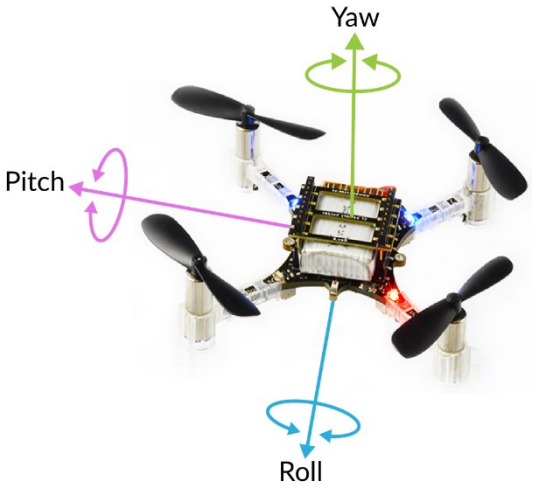
Figura 14 – Antena para comunicação via rádio

Fonte: Bitcraze (2022).

A orientação do quadricóptero se dá por seis principais graus de liberdade, sendo as três posições no espaço tridimensional (x, y, z) e os três ângulos de orientação: rolagem (*roll*), que é a rotação lateral em relação ao eixo longitudinal, arfagem (*pitch*), que é a inclinação frontal em relação ao eixo transversal e a guinada (*yaw*), rotação em torno do eixo vertical.

Além disso, a estimativa da posição do VANT no ambiente acontece pela comunicação TWR, o módulo receptor manda uma mensagem para as âncoras em sequência, assim, consegue calcular as distâncias até ela e se localizar no ambiente 3D (BITCRAZE, 2022).

Figura 15 – Ângulos de orientação



Fonte: Bitcraze (2022).

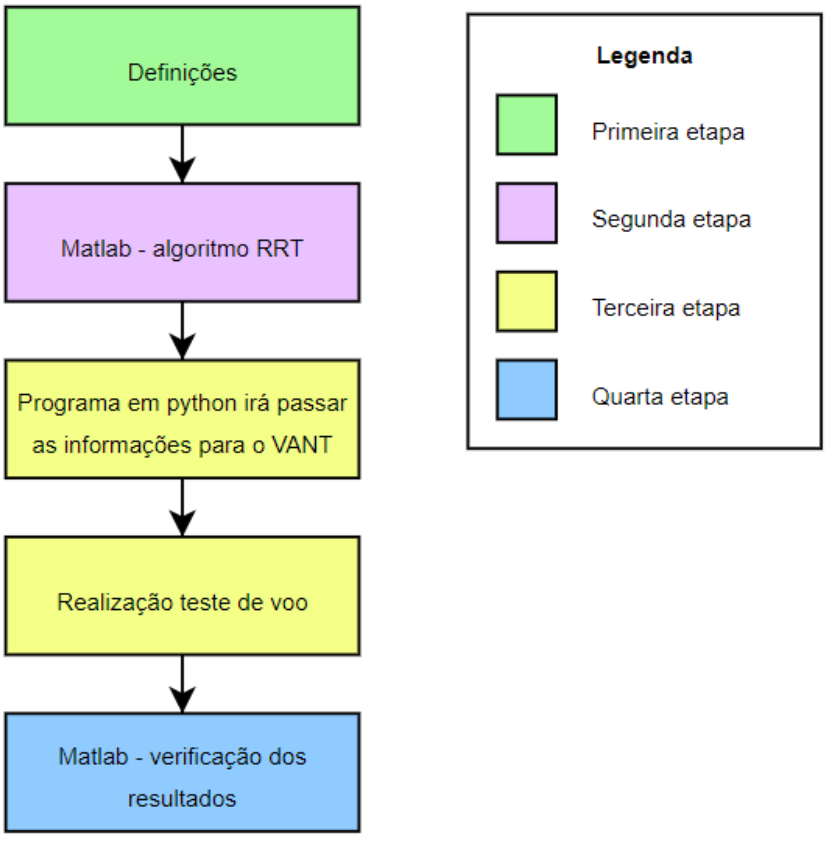
Esse quadricóptero é muito utilizado na parte de pesquisa, principalmente em universidades, tendo muitas vantagens para sua utilização, como por exemplo ser uma plataforma de código aberto (*open source*), ter atualizações recorrentes, além de seu tamanho e peso reduzidos o que permitem que essa aeronave seja utilizada também em ambientes indoor.

3 METODOLOGIA

O projeto foi desenvolvido em quatro etapas, a primeira foi a definição dos parâmetros gerais, como as dimensões do ambiente e informações sobre os obstáculos, quantos seriam usados, suas dimensões e localizações. A segunda etapa foi implementar o método RRT no Matlab, para isso foi necessário criar funções específicas para gerar pontos, calcular distâncias, e desenvolver o código principal, o qual junta todas essas funções e tem como objetivo achar uma lista de pontos correspondente a trajetória necessária para sair do ponto inicial e chegar ao ponto final, evitando os obstáculos presentes no ambiente.

Já a terceira etapa utiliza a lista de pontos encontrada no Matlab e um código em python, o qual transmite ao VANT a real trajetória a ser seguida, também é nessa etapa que acontece o teste de voo. Para a etapa final, foi feito uma verificação entre a trajetória encontrada pela simulação e a trajetória que o VANT realizou no teste de voo, essa comparação também foi realizada no Matlab. Todas essas etapas podem ser visualizadas no fluxograma do projeto na Fig. 16.

Figura 16 – Fluxograma do projeto



Fonte: A autora (2023).

3.1 IMPLEMENTAÇÃO DO ALGORITMO RRT

O Matlab é um software usado para desenvolver algoritmos, analisar dados e criar modelos e é muito utilizado na área de desenvolvimento. Muito útil para resolução de questões matemáticos e facilita a visualização de problemas através de gráficos. Uma das principais aplicações do uso do Matlab é na modelagem de controle, o que permite a representação de sistemas reais em equações matemáticas.

Nesse projeto, o software Matlab foi utilizado para encontrar a trajetória saindo de um ponto inicial até alcançar o ponto final, ambos pré-determinados, desviando de obstáculos através do método RRT. Para otimizar o desenvolvimento do projeto, foi necessário a criação de funções auxiliares, sendo elas:

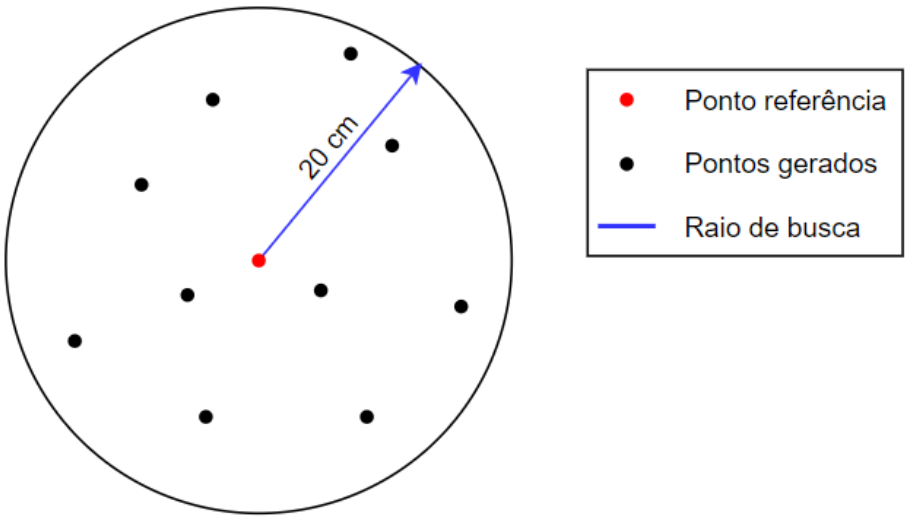
- Geração de pontos aleatórios;
- Cálculo da distância entre pontos;
- Verificação da existência de obstáculos na trajetória;
- Geração de gráficos com o obstáculo no ambiente.

Além dessas funções, também foram desenvolvidos o código principal e o código para verificar e comparar a trajetória da simulação e a do ensaio experimental de voo.

3.1.1 GERAÇÃO DE PONTOS ALEATÓRIOS

Esta função é responsável por gerar uma quantidade pré-selecionada de pontos aleatórios ao redor de um ponto qualquer escolhido, dentro de um raio determinado – raio de busca, que neste caso será igual a 20 cm, ou seja, os pontos gerados estarão dentro de uma circunferência com o raio igual ao raio de busca, Fig. 17. No projeto, serão gerados 10 pontos aleatórios a cada ponto de origem estabelecido. O primeiro ponto referência é o ponto inicial previamente escolhido para a trajetória e, dentre os pontos gerados, o algoritmo escolhe o mais próximo do ponto objetivo como eletivo para as subsequentes verificações e validações, caso não exista nenhuma restrição, este será o próximo ponto referência e assim por diante, até atingir o ponto final.

Figura 17 – Funcionamento da função de gerar pontos aleatórios



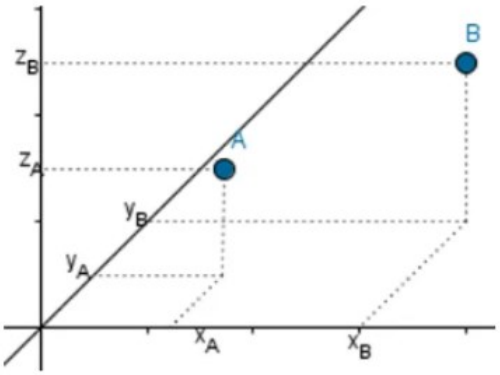
Fonte: A autora (2023).

3.1.2 CÁLCULO DA DISTÂNCIA ENTRE PONTOS

Para o funcionamento do método RRT, existem dois momentos no qual é necessário calcular a distância entre dois pontos. Primeiramente, quando gera pontos aleatórios, é calculada a distância entre esses pontos e o ponto final, com o intuito de escolher o mais próximo. Posteriormente, quando o próximo ponto já foi escolhido, as verificações finais foram realizadas e esse ponto vira o ponto referência, novamente é calculada a distância, agora entre esse novo ponto e o ponto final. Se essa distância for menor que o raio de captura, é entendido que o ponto final foi encontrado, se não, é necessário repetir o processo e encontrar os próximos pontos.

De acordo com os estudos da geometria analítica, a distância entre dois pontos é baseada no teorema de Pitágoras e se dá pelo comprimento do segmento que liga os dois no plano cartesiano, o mesmo acontece para pontos tridimensionais (GUZMAN, 2021).

Figura 18 – Fórmula para calcular distância entre dois pontos 3D



$$d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2}$$

Fonte: Adaptado de Silva (2016).

3.1.3 VERIFICAÇÃO DA EXISTÊNCIA DE OBSTÁCULOS NA TRAJETÓRIA

Para verificar se tem algum obstáculo entre o ponto que o VANT se encontra e o possível novo ponto foi necessário criar duas funções:

- Verificar se as coordenadas estão coincidindo com as coordenadas do obstáculo;
- Verificar toda a trajetória entre esses dois pontos.

Para verificar se há obstáculo, é determinado, de acordo com as coordenadas passadas, se a posição é menor que o ponto inicial do obstáculo ou maior que essa posição mais o seu tamanho, ou seja, posição final do obstáculo. Se essa condição for verdade, significa que esse ponto não está no obstáculo, já se for falso, significa que que esse ponto está dentro do obstáculo, portanto, é necessário escolher outro ponto. Na Fig. 19, está evidenciado o pseudocódigo desta etapa.

Figura 19 – Verificar se o ponto está no obstáculo

```
função ESTÁ_NO_OBSTÁCULO (coordenadas, obstáculo)
        enquanto i=1:1:3 (x, y e z)
                se coordenadas(i) não está no obstáculo(i)
                        posição válida
                        break
                fim se
        fim enquanto
fim função
```

Fonte: A autora (2023).

Já na segunda função, onde ocorre a verificação de colisão, retorna-se um vetor linha de M pontos, com M maior ou igual a 2, uniformemente espaçados entre n2 e n1, novo ponto eletivo e o ponto referência, respectivamente. A distribuição uniforme dos pontos é possível através da função *linspace*, que, no caso deste projeto, será utilizada nas três dimensões – x, y e z. O espaço – S – entre os pontos gerados é regido pela Eq. 1:

$$S = \frac{(n2 - n1)}{(M - 1)} \qquad (1)$$

Por exemplo, para M igual a 6, entre pontos situados nas posições 0 e 10, serão gerados 6 pontos com espaçamento igual a 2, onde o ponto inicial e final serão os pontos n1 e n2, respectivamente. O vetor linha gerado é utilizado para verificar a existência de algum obstáculo na posição dos pontos n1 e n2 e no trajeto entre eles, essa verificação ocorre ao chamar a função explicada acima, Fig. 19, para cada uma das posições encontradas em x, y e z. O pseudocódigo desta etapa é ilustrado na Fig. 20.

Figura 20 – Verificar se ocorre colisão

```
função VERIFICA_COLISÃO (p_new, p_goal, obstáculo)
        nº_pontos_gerados ← 20
        coordenadas_x ← nº_pontos_gerados entre p_init e p_goal em x
        coordenadas_y ← nº_pontos_gerados entre p_init e p_goal em y
        coordenadas_z ← nº_pontos_gerados entre p_init e p_goal em z

        enquanto i = 1:1: nº_pontos_gerados
                coordenadas ← [coordenadas_x(i), coordenadas_y(i), coordenadas_z(i)]
                se ESTÁ_NO_OBSTÁCULO (coordenadas, obstáculo)
                        ponto p_new não é válido
                fim se
        fim enquanto
fim função
```

Fonte: A autora (2023).

Se, após essas verificações, não tiver colisão com nenhum obstáculo, o ponto pode ser utilizado, passando a ser o novo ponto referência e é adicionado à lista de pontos da trajetória. Se ocorrer alguma colisão, o ponto não servirá, sendo necessário escolher outro ponto para ser o eletivo e refazer as mesmas verificações, até não acontecer colisões.

3.1.4 GERAÇÃO DE GRÁFICOS COM O OBSTÁCULO NO AMBIENTE

Para facilitar a visualização das simulações, foi criada uma função para representar os obstáculos, nela são declarados os vértices e as faces. Como mencionado anteriormente, para declarar um obstáculo, são necessários seis parâmetros, os três primeiros referentes a sua posição inicial e os outros referentes aos tamanhos em cada dimensão, conforme Eq. 2.

$$obstáculo = [x_{inicial} \;\; y_{inicial} \;\; z_{inicial} \;\; tamanho_x \;\; tamanho_y \;\; tamanho_z] \qquad (2)$$

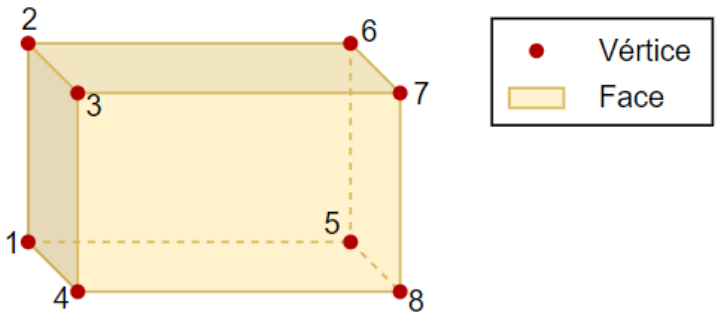
Para determinar os vértices, é usada a posição inicial do obstáculo e a posição final – que é a inicial mais o tamanho para cada dimensão. Portanto, o vértice 1, o qual é considerado a origem, contém os pontos x, y e z iniciais e nada mais, conforme Eq. 3. Para o vértice 2, conforme Fig. 21, mantem-se as coordenadas x e z, e adiciona-se o tamanho em y daquela face à coordenada y, conforme Eq. 4. Este padrão se mantém até que todos os 8 vértices sejam estabelecidos.

$$Vértice\ 01 = [x_{inicial} \;\; y_{inicial} \;\; z_{inicial}] \qquad (3)$$

$$Vértice\ 02 = [x_{inicial} \;\; (y_{inicial} + tamanho_y) \;\; z_{inicial}] \qquad (4)$$

Feito isso, foram constituídas as faces do obstáculo, onde determina os vértices pertencentes a cada face. Por exemplo, conforme Fig. 21, a primeira face é composta pelos vértices 1, 2, 3, e 4. E, por fim, define-se a cor para as faces como amarelo. Sendo assim, o obstáculo é finalizado, sendo possível evidenciá-lo no ambiente, juntamente com a trajetória percorrida, posteriormente.

Figura 21 – Representação do obstáculo



Fonte: A autora (2023).

3.1.5 ROTINA PRINCIPAL

Após a criação das funções auxiliares citadas acima, foi desenvolvido o código principal que, de fato, irá implementar o método RRT para calcular a trajetória no Matlab.

Primeiramente, foram declarados parâmetros iniciais, como tamanho do ambiente, que começa na origem e vai até as posições máximas para as três dimensões, a localização e tamanhos dos obstáculos. As posições máximas no ambiente foram configuradas de acordo com o tamanho real da sala onde o teste de voo será executado, os valores de x, y e z são referentes a largura, profundidade e altura, respectivamente, esses valores precisam estar em centímetros. Logo após essa etapa, é gerado um gráfico no qual é possível observar o ambiente e os obstáculos.

Para finalizar as configurações iniciais, é necessário definir o ponto inicial, o ponto final e os raios de busca e captura. O ponto inicial começa sendo o ponto referência e à medida que outros pontos forem escolhidos, esse ponto será substituído. Além disso, define-se o número de pontos aleatórios que serão gerados e o número de iterações.

Tabela 2 – Inserção dos parâmetros

Variável	**Valor atribuído**	**Unidade de medida**
Ponto inicial	[30, 30, 30]	cm
Ponto final	[240, 400, 250]	cm
Raio de captura	20	cm
Raio de busca	20	cm
Número de pontos aleatórios	10	-
Número de iterações	200	-

Fonte: A autora (2023).

Após tudo configurado, o algoritmo começa o planejamento de trajetórias. Primeiramente, irá gerar os pontos aleatórios dentro do raio de busca ao redor do ponto referência, calcular a distância para cada ponto gerado até o ponto final, e ordená-los para escolher o ponto mais perto do objetivo, esse será o ponto eletivo, p_new, o qual passará pelas verificações.

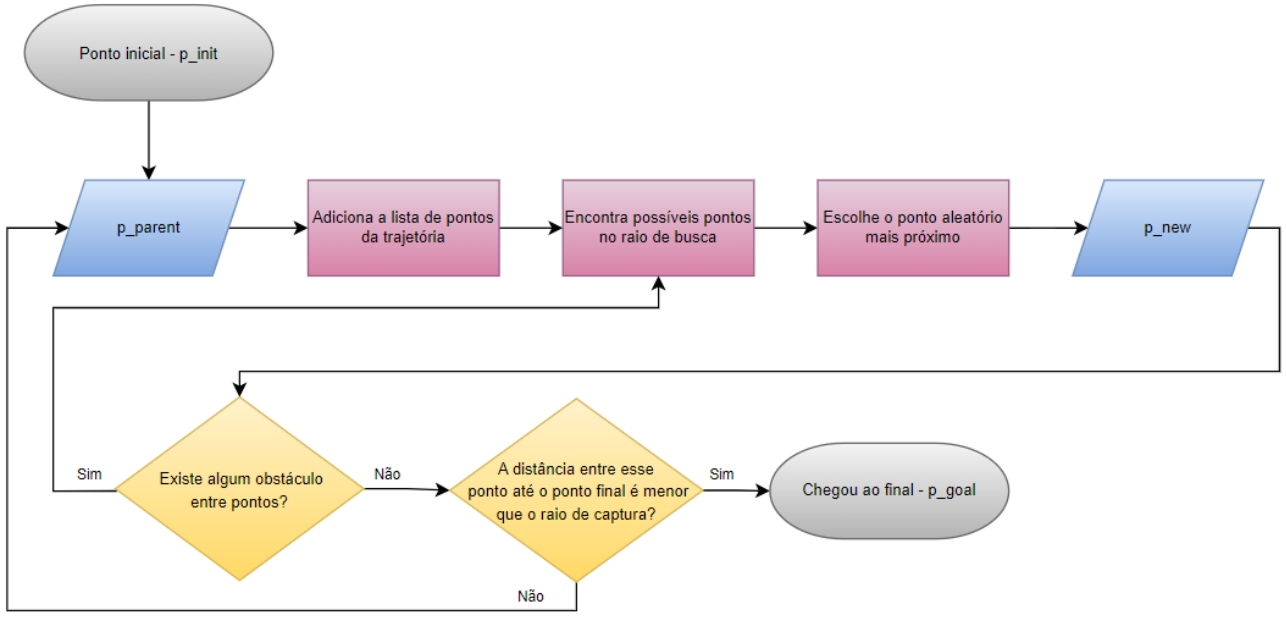
São feitas as verificações para decidir se o novo ponto está apto para substituir o ponto referência. Primeiramente, verifica se existe obstáculo entre esse ponto e o ponto referência, se existir, esse ponto não serve para ser o novo ponto referência e é necessário gerar novos pontos e refazer as verificações. Caso não exista, esse ponto continua elegível e passará pela segunda verificação, calcula-

se a distância entre o ponto e o ponto final e verifica se é menor que o raio de captura, se for menor significa que o ponto objetivo foi encontrado, se não for menor significa que o ponto é valido, porém não encontrou o objetivo, portanto esse ponto é adicionado à lista de pontos da trajetória, e ele passa a ser o ponto referência, e todo processo é refeito até que encontre o ponto final.

Ao final dessa etapa, foi adicionada uma função do software Matlab chamada TicToc, que funciona como um cronômetro. Com essa função é possível saber quanto tempo foi necessário para que o algoritmo encontrasse a trajetória entre os dois pontos.

Por fim, a trajetória já foi traçada e armazenada na lista (lista_parent), basta colocar isso em um gráfico para facilitar visualização, para isso é adicionado ao gráfico os pontos inicial e final, e a lista da trajetória. Todas essas etapas realizadas no Matlab estão ilustradas no fluxograma da Fig. 22.

Figura 22 – Fluxograma do projeto no Matlab



Fonte: A autora (2023).

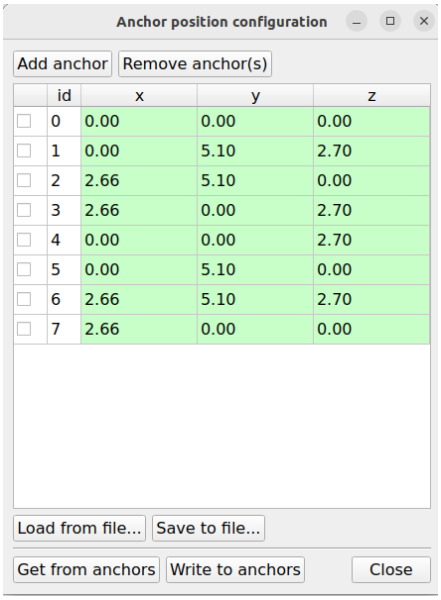
3.2 CONFIGURAÇÃO DO SISTEMA DE POSICIONAMENTO LOCAL

Para a realização dos ensaios experimentais, foi necessário fazer a preparação de um ambiente adequado, para isso foram decididas as dimensões, 2,66 metros de largura, 5,10 metros de profundidade e 2,70 metros de altura.

Foram colocadas quatro hastes no ambiente, uma em cada vértice, em suas extremidades foram posicionadas as âncoras, uma na parte inferior e a outra na superior, sistema de posicionamento local.

Por fim, foram feitas as configurações do sistema dentro do software do próprio Bitcraze, foi necessário colocar todas as dimensões em metros para cada vértice, conforme mostrado na Fig. 23:

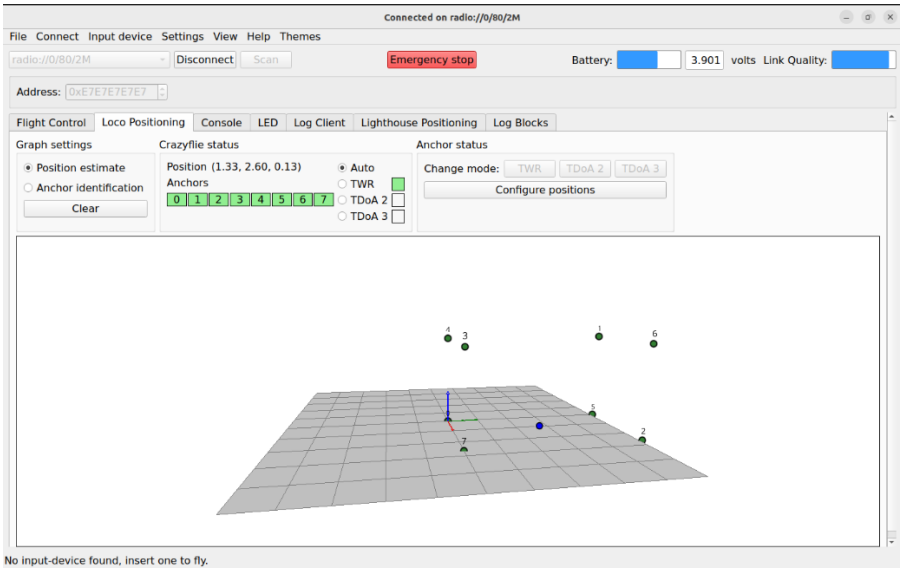
Figura 23 – Posicionamento das âncoras



Fonte: A autora (2023).

Após preenchimento das dimensões e clicar em "*Write to anchors*", uma nova tela aparece onde é possível verificar se todas as âncoras estão funcionando corretamente e se estão na posição correta.

Figura 24 – Visualização das âncoras e do VANT



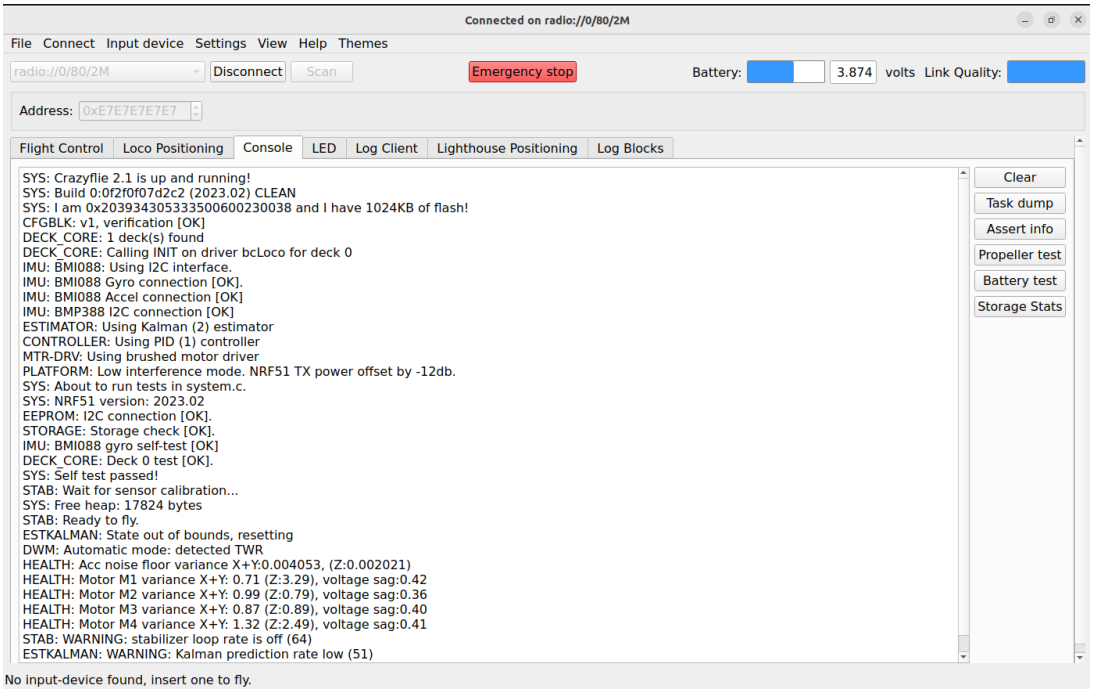
Fonte: A autora (2023).

Na figura anterior, os pontos verdes são as âncoras que estão numeradas de zero a sete e o ponto azul representa o VANT na posição que ele se encontrava nesse momento. Caso haja algum erro, qualquer uma das indicações verdes seria representada pela cor vermelha, que poderia ser algum problema de comunicação, cabo com mal contato, algum problema com a comunicação via rádio, ou então algum erro nas configurações.

Na próxima etapa, foram feitas as verificações do VANT. Após ligá-lo e ativar a comunicação via rádio, aparece informações como a qualidade da comunicação, mostrado no canto superior direito da figura anterior, e o nível de bateria do VANT (Volts). Existem dois botões nessa tela, um com a função de parada de emergência e outro para desconectar a comunicação.

Para maior confiabilidade nos testes de voo, é possível que sejam feitas verificações prévias na aeronave, como a constatação do funcionamento das hélices, motores e bateria do quadricóptero. Todas as verificações feitas são evidenciadas na Fig. 25, assim como os botões no lado direito para refazer os testes:

Figura 25 – Verificações de funcionamento do VANT



Fonte: A autora (2023).

Nessa tela também são ilustradas algumas informações extras do VANT, como, por exemplo, a utilização de um controlador PID e um filtro de Kalman.

Com o ambiente configurado e o VANT com a liberação de voo, são feitas as modificações no algoritmo em python, o qual irá passar as coordenadas que o VANT precisa seguir. Essas coordenadas foram encontradas por meio da simulação no Matlab e são colocadas no código em python para validação da trajetória com um ensaio experimental.

A base para esse programa em python é disponibilizada pelo próprio fabricante do Crazyflie 2.1. Primeiramente, as bibliotecas necessárias foram importadas, as variáveis globais foram criadas e foi configurado a comunicação via rádio, para isso foi criado um identificador de recurso uniforme (URI), especificando que é para uma interface de rádio, utilizando a porta USB número zero, o canal de rádio na porta 80 e por último a velocidade de 2Mbit/s. Após essas configurações iniciais, são adicionadas as posições que o VANT precisa seguir e a sua posição é salva a cada 500 milissegundos, que é o tempo de amostragem.

As demais configurações e funções desse algoritmo seguem o padrão do Bitcraze e não foram necessárias modificações. Por fim, as posições reais que o VANT realmente percorrer serão salvas em 3 arquivos de texto, um para cada dimensão contendo os as posições para x, y e z.

4 RESULTADOS E DISCUSSÕES

Para validação do método RRT para o planejamento de trajetória foram analisados quatro cenários.

Primeiro, um ambiente sem obstáculos, para validar se é possível encontrar a trajetória entre os dois pontos, em um ambiente tridimensional sem interferências externas. Em seguida, um cenário contendo um obstáculo flutuante, para verificar se é possível encontrar um caminho desviando dele. Então, um novo cenário, também com obstáculo, mas simulando um ambiente real, para isso foi adicionado dois obstáculos. Por fim, um ambiente com múltiplos obstáculos, com intuito de encontrar situações mais complexas e observar como o algoritmo resolve o planejamento de trajetórias nessa situação.

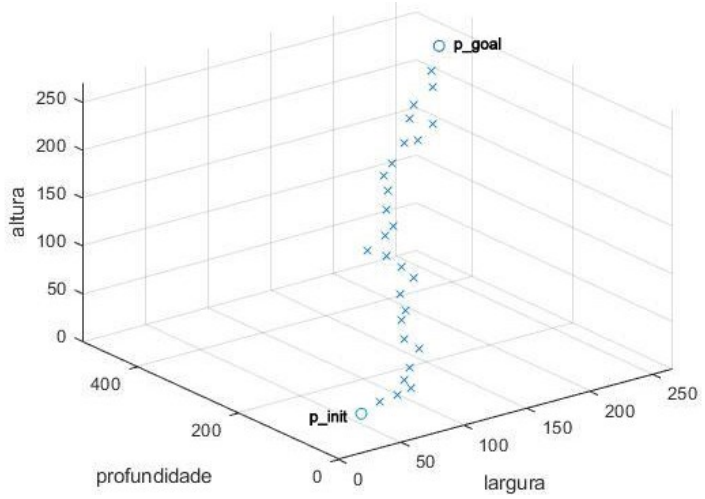
4.1 CENÁRIO SEM OBSTÁCULO

Primeiramente, foi verificado que é possível sair de um ponto inicial e chegar ao ponto final, utilizando o método RRT, para essa validação foi realizado somente a simulação no Matlab em um cenário sem obstáculos.

Foi utilizado o mesmo algoritmo, mas o obstáculo foi colocado na origem e com tamanho zero em todas as dimensões. De tal modo, os cálculos foram realizados sem que nenhum obstáculo fosse encontrado no caminho e foi possível descobrir uma trajetória que o VANT poderia seguir sem nenhuma interferência, comprovando assim que o método consegue realizar seu papel inicial e traçar a trajetória entre os dois pontos pré-determinados. O algoritmo levou 0,42 segundos para encontrar a trajetória.

Os pontos representados pelo ‘x’ são os pontos necessários para a trajetória e os representados pelo círculo são referentes aos pontos do início (p_init) e do fim (p_goal).

Figura 26 – Trajetória encontrada na simulação para ambiente sem obstáculos

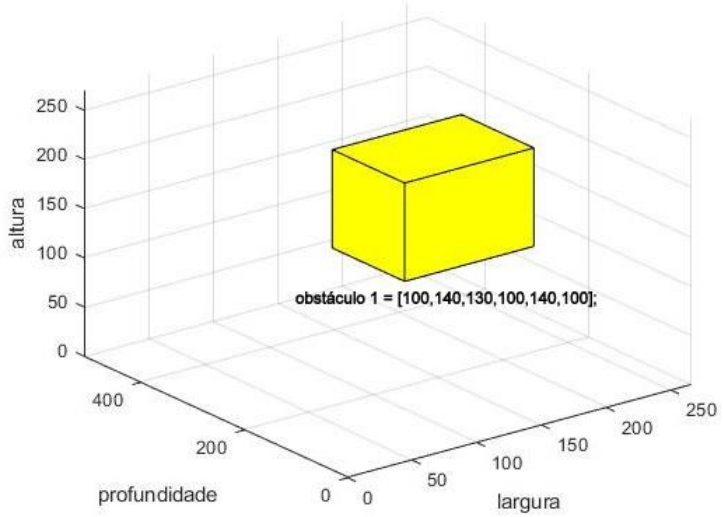


Fonte: A autora (2023).

4.2 CENÁRIO COM UM OBSTÁCULO

Após a validação inicial do método, foi inserido nas simulações um obstáculo flutuante, com posições 100, 140 e 130, para x, y e z, respectivamente, os quais terão 100, 140 e 100 de comprimento, conforme Fig. 27.

Figura 27 – Ambiente configurado com um obstáculo

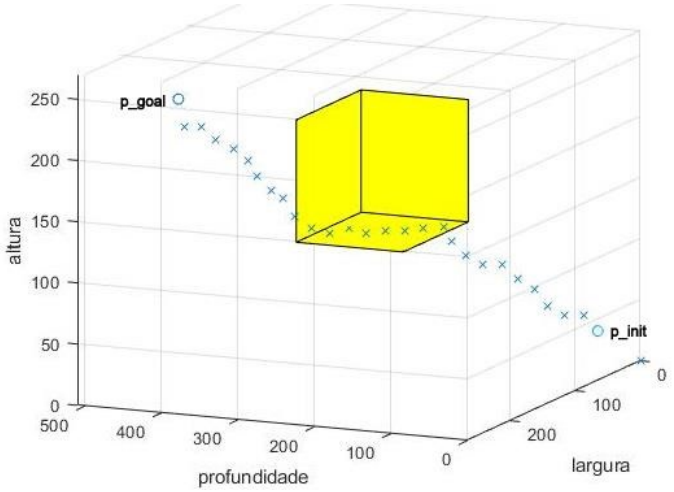


Fonte: A autora (2023).

As três primeiras posições colocadas na declaração do obstáculo são referentes as coordenadas nas três dimensões, e as últimas três posições se referem ao tamanho em cada eixo.

Com o ambiente e o obstáculo definidos, foi calculada a trajetória pelo software Matlab, Fig. 28. Para melhor visualização da trajetória, a imagem foi rotacionada. O algoritmo levou 0,56 segundos para encontrar a trajetória.

Figura 28 – Trajetória encontrada na simulação para ambiente com um obstáculo flutuante



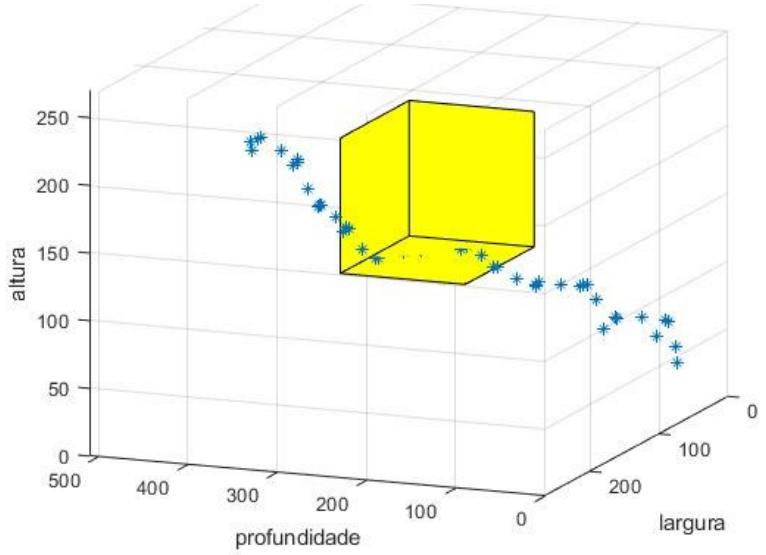
Fonte: A autora (2023).

De acordo com a simulação, para esse cenário, seria necessário realizar a trajetória mostrada na figura, a qual contém 26 posições. As coordenadas dessas posições estão em centímetros, portanto foi necessário converter esses valores para metro, por fim, a lista final de pontos para o VANT seguir que foi colocada no código em python.

Após todos esses ajustes foi possível realizar o teste de voo, foram realizados dois testes com o objetivo de validar a trajetória e não ter problemas por alguma situação externa, como algum problema no VANT. Como resultado desses testes, uma nova lista de posições é encontrada, nesse caso contém as posições que o VANT realmente percorreu, essa lista de pontos vai ser maior, visto que o drone armazena a posição a cada meio segundo.

Retornando ao Matlab foi possível visualizar a trajetória real do VANT com o obstáculo, verificando se teve alguma falha no teste de voo, como por exemplo algum problema de localização no sistema de posicionamento local ou com o VANT. Caso algum problema fosse encontrado o teste seria refeito.

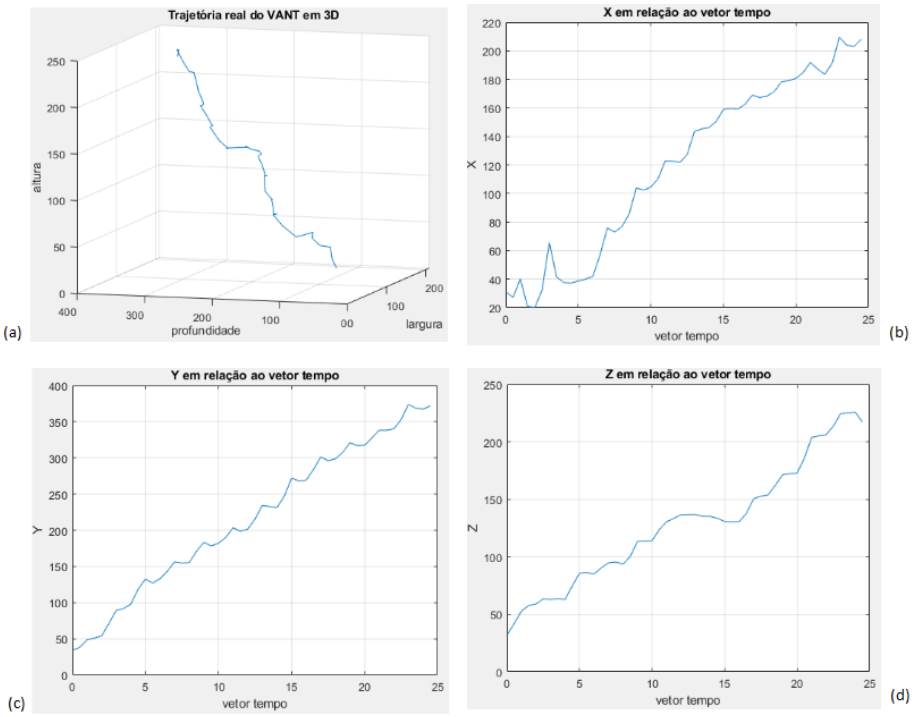
Figura 29 – Trajetória real do VANT



Fonte: A autora (2023).

Após essa investigação, ainda no Matlab, é interessante observar outros gráficos representando a real trajetória percorrida, na Fig. 30(a) é mostrado a trajetória em 3D, na Fig. 30(b) a representação de x em relação ao tempo, na Fig. 30(c) a representação de y e por fim na Fig. 30(d) a representação de z.

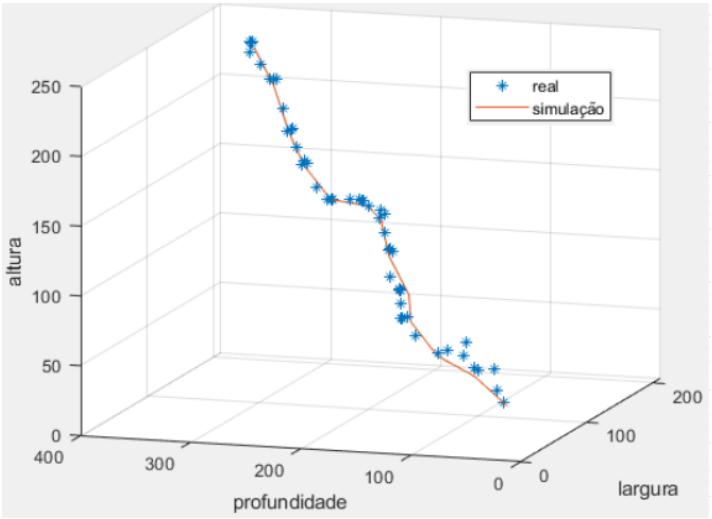
Figura 30 – Trajetória do VANT



Fonte: A autora (2023).

Por fim, foi comparado a trajetória percorrida no teste de voo com a trajetória encontrada na simulação no Matlab.

Figura 31 – Comparação da trajetória da simulação com o teste de voo



Fonte: A autora (2023).

Após todos esses testes e verificações, foi validado que o método cumpre com o esperado e que a simulação e o teste de voo foram um sucesso para um cenário com um obstáculo flutuante.

4.3 CENÁRIO COM DOIS OBSTÁCULOS

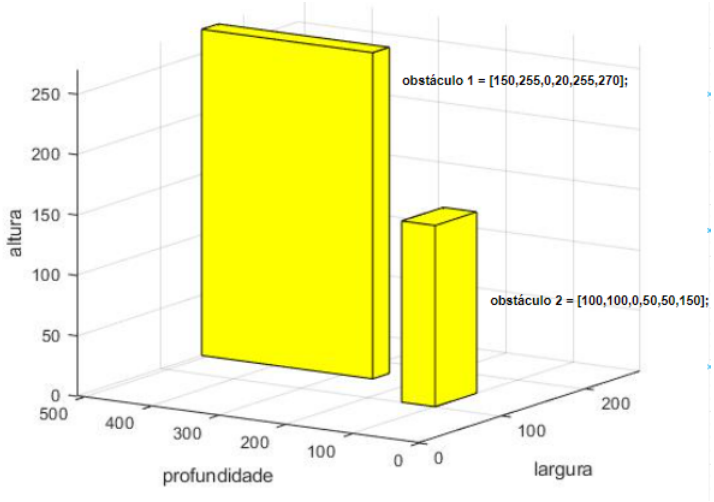
Para validação final, foi considerado um terceiro cenário com dois obstáculos, esse cenário pode ser comparado com situações reais do dia a dia, já que nesse caso um dos obstáculos será uma parede, que está ligado ao chão, ao teto e a uma das paredes do ambiente, e o outro obstáculo se encontra no chão, perto do primeiro, ou seja, a passagem pro VANT agora está bem menor, podendo somente passar entre os dois obstáculos ou por cima do segundo.

Esse é um cenário que corresponde com a realidade, visto que os obstáculos não são flutuantes, e também demostra uma situação bem comum, a qual podemos comparar, por exemplo, com um ambiente industrial, uma sala de máquinas dividida por uma parede e perto da passagem tem um tanque de água que alimenta uma caldeira. O VANT nesse caso poderia atravessar essa sala com o intuito de verificar algum problema, posteriormente, uma câmera ou algum sensor poderia ser instalada nele, e não seria necessário mover uma equipe até o local para essa verificação inicial.

A realização dos testes para essa situação mencionada foi semelhante ao cenário com um obstáculo, primeiramente foi criado o ambiente com os obstáculos citados. O primeiro está localizado

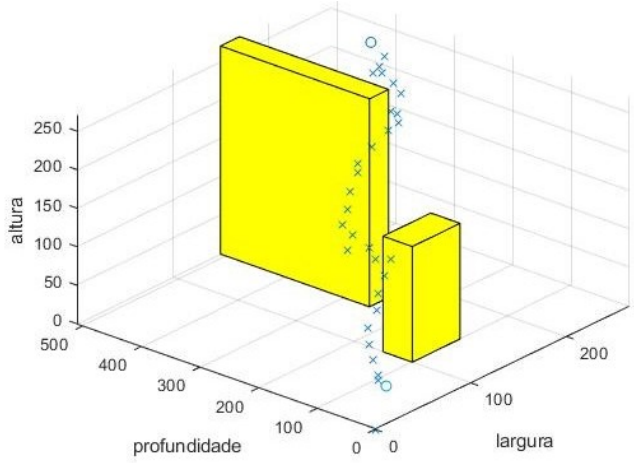
nas posições 150, 255 e 0, x, y e z, respectivamente, com os tamanhos 20, 255 e 270. Já o segundo se encontra nas posições 100, 100 e 0, x, y e z, respectivamente, com os tamanhos 50, 50 e 150, conforme Fig. 32. Então, a trajetória foi calculada pelo software Matlab, Fig. 33. O algoritmo levou 0,45 segundos para encontrar a trajetória.

Figura 32 - Ambiente configurado com dois obstáculos



Fonte: A autora (2023).

Figura 33 - Trajetória encontrada na simulação para ambiente com dois obstáculos reais

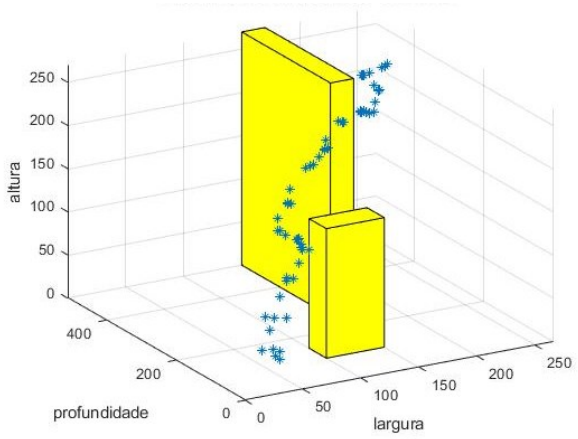


Fonte: A autora (2023).

Como mencionado nó tópico 3.2, essas coordenadas encontradas no Matlab estão em centímetros, portanto foram convertidas para metro e a lista final de pontos foi colocada no código em python.

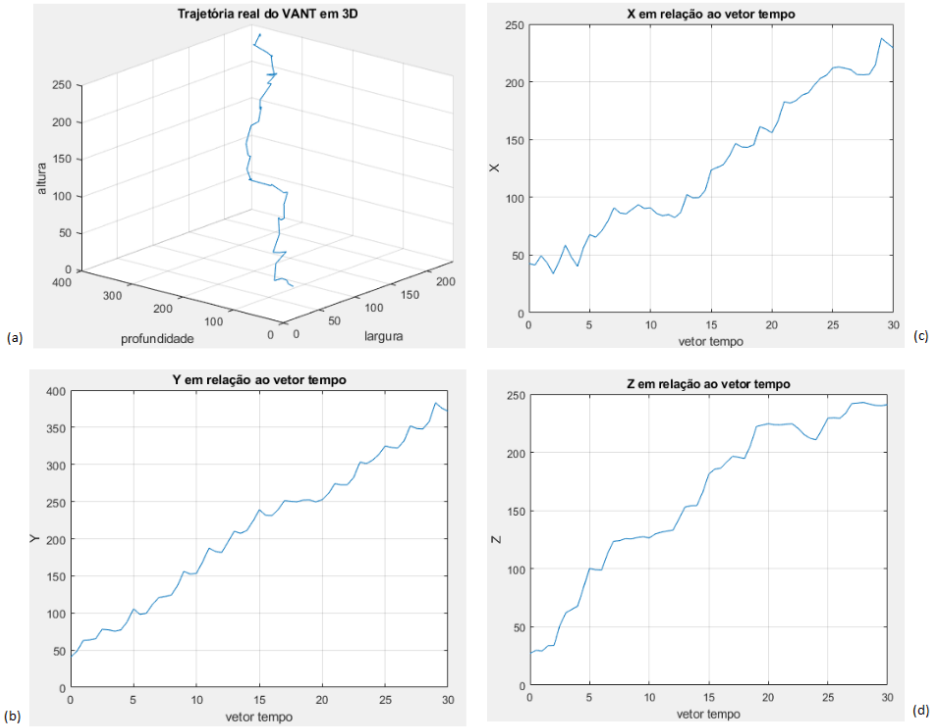
Foi realizado dois testes de voo, e uma nova lista de pontos foi encontrada com as posições que o VANT realmente percorreu. Do mesmo modo que para o cenário com somente um obstáculo, foram feitas as validações dos testes no Matlab, primeiramente foi verificado se teve algum problema no teste de voo, em seguida foram gerados os gráficos representando a real trajetória percorrida, a trajetória em 3D e, então, a representação de x, y e z em relação ao tempo.

Figura 34 - Trajetória real do VANT



Fonte: A autora (2023).

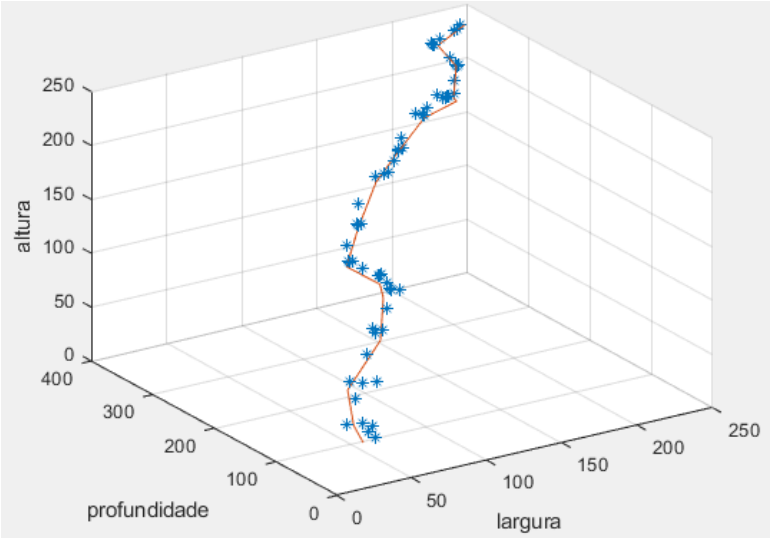
Figura 35 – Trajetória do VANT



Fonte: A autora (2023).

Por fim, foi comparado a trajetória percorrida no teste de voo com a trajetória encontrada na simulação no Matlab.

Figura 36 - Comparação da trajetória da simulação com o teste de voo



Fonte: A autora (2023).

Finalizando os testes, foi visto que método RRT também consegue atingir seu objetivo em um cenário real e que, mesmo com uma passagem estreita, foi possível encontrar a trajetória, o que torna possível levar o VANT para o ambiente industrial, como o exemplo citado no começo desse tópico. Apesar dos obstáculos no ambiente, se forem previamente mapeados e estiverem no ambiente de configuração do VANT, é possível encontrar a trajetória que corresponda as necessidades e chegue ao ponto final, facilitando a verificação primária de problemas sem que haja necessidade de mover um time de manutenção ao local até que saibam qual o problema e já consigam ir com a solução, ganhando tempo, além de evitar riscos desnecessários, caso haja perigo a vida humana.

Conclui-se que o todos os testes para um cenário com dois obstáculos foram um sucesso e o método RRT está validado, podendo até ser usado em cenários reais.

4.4 CENÁRIO SIMULADO COM MÚLTIPLOS OBSTÁCULOS

O último cenário conta com um ambiente com múltiplos obstáculos e, por ser um experimento mais complexo, só foi feita a etapa de simulação. O tamanho do ambiente e os parâmetros iniciais foram modificados, conforme Tabela 3:

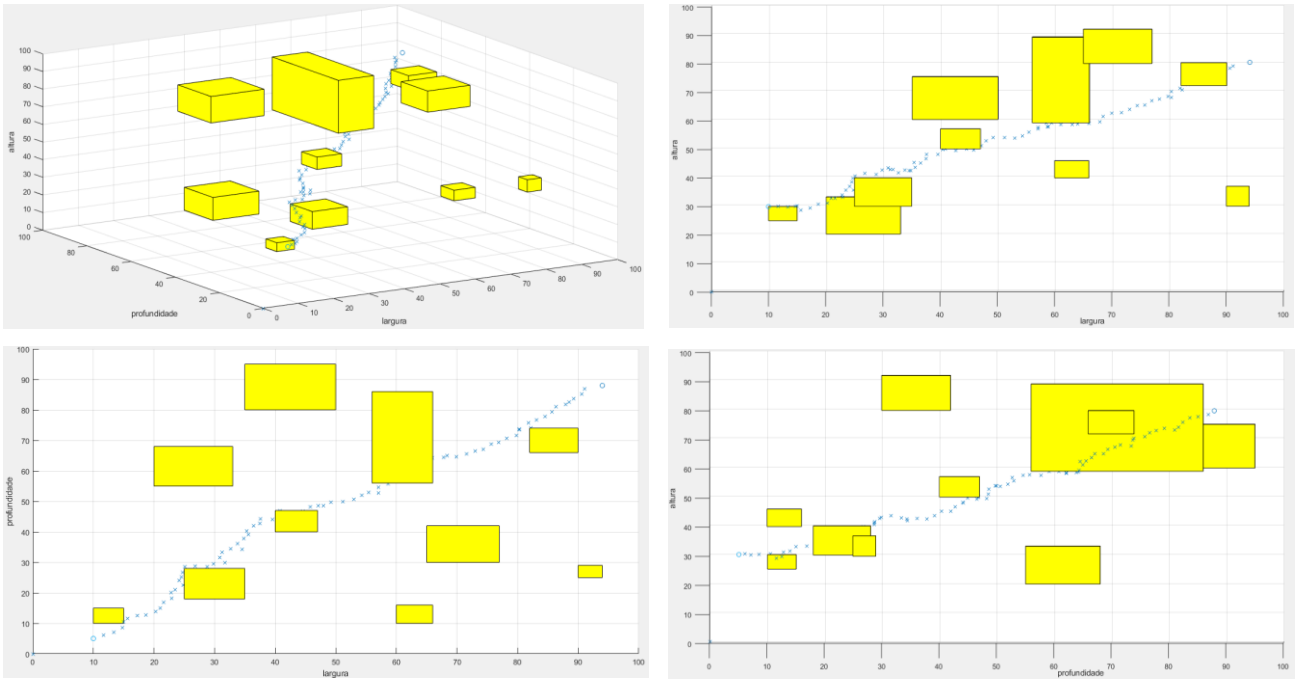
Tabela 3 – Modificação dos parâmetros

Variável	Valor atribuído	Unidade de medida
X máximo	100	Cm
Y máximo	100	Cm
Z máximo	100	Cm
Ponto inicial	[10, 5, 30]	Cm
Ponto final	[94, 88, 80]	Cm
Raio de captura	2	Cm
Raio de busca	2	Cm
Número de pontos aleatórios	10	-
Número de iterações	200	-

Fonte: A autora (2023).

Para o cálculo da trajetória, foi modificada algumas funções para que fosse possível adicionar mais obstáculos, nesse caso foram colocados dez obstáculos aleatoriamente dispostos no ambiente e com diferentes tamanhos. Para essa simulação, o algoritmo levou 0,55 segundos para encontrar a trajetória e o resultado obtivo é ilustrado na Fig. 37:

Figura 37 – Trajetória simulada com múltiplos obstáculos



Fonte: A autora (2023).

5 CONCLUSÃO

Após a validação para todos os cenários, foi concluído que o método RRT é capaz de encontrar uma trajetória entre o ponto inicial e o ponto final, desviando de obstáculos pré-determinados e que é possível utilizar o VANT em trabalhos reais, colaborando com o ser humano.

O método tem como pontos positivos ser eficiente em diferentes ambientes, desde os mais simples até os de alta complexidade. Além disso, um ponto importante é que, com esse método, o VANT sempre conseguirá desviar dos obstáculos, já que, os cálculos são refeitos, à medida que o ponto referência avança, fazendo com que o planejamento seja feito em tempo real. Portanto, será possível encontrar a solução, caso ela exista e o número de iterações for suficiente. Outro ponto positivo a destacar é que o método RRT se adapta para diferentes tipos de VANT.

Por outro lado, os pontos negativos também existem, como o fato de que não é garantido que o caminho ótimo será encontrado e suas configurações são dependentes dos parâmetros iniciais, o que pode ser complexo de configurar. Além disso, durante a trajetória, o VANT pode se deparar com mínimos locais, isso pode acontecer se a passagem for muito estreita ou até mesmo no próprio obstáculo.

Os pontos negativos levantados podem ser melhorados em trabalhos futuros. Com a adição de sensoriamento ao VANT, como câmeras e sensores de identificação de movimento e objetos, tornaria possível detectar a utilização deste método em ambientes com obstáculos dinâmicos, além de implementação do algoritmo em tempo real. Outro ponto importante é a implementação de softwares visando a diminuição do efeito negativo da questão de mínimos locais, onde há a dificuldade de trajetórias em caminhos estreitos e com obstáculos complexos.

Finalmente, torna-se visível que os objetivos propostos neste trabalho foram atingidos e que conceitos importantes na área de controle e automação foram aprofundados. Fica claro que o método carrega consigo importantes funcionalidades que propiciarão a popularidade dos VANTs em diversos setores e, com melhorias, esse avanço na utilização será cada vez maior.

REFERÊNCIAS BIBLIOGRÁFICAS

ABOU EL MAJD, Y.; EL GHAZI, H.; NAHHAL, T., 2017, **PaRRT: Parallel rapidly exploring random tree (RRT) based on MapReduce**. IEEE. 1-5.

ADIYATOV, O.; VAROL, H. A., 2013, **Rapidly-exploring random tree based memory efficient motion planning**. IEEE. 354-359.

BENSRHAIR, A.; BAPIN, T., 2021, **From AI to Autonomous and Connected Vehicles: Advanced Driver-Assistance Systems (ADAS)**.

BITCRAZE. **Crazyflie 2.1**. p. Site do fabricante do VANT Crazyflie 2.1. Disponível em: https://www.bitcraze.io

BROTERO, M., MARTINS, A., 2023. **Guerra na Ucrânia: Entenda por que ataques com drones têm sido cada vez mais comuns.**

CALIXTO, F., 2023, **História dos drones: como surgiram? Para que servem?** Disponível em: https://itarc.org/historia-dos-drones/

CHAIMOWICZ, L., SOUSA, S., 2007, **Utilizando Rapidly-Exploring Random Trees (RRTs) para o planejamento de caminhos em jogos.**

CHATTERJEE, M., 2023. **What is A* Search Algorithm?**

CHEN, L.; SHAN, Y.; TIAN, W.; LI, B. *et al.* **A fast and efficient double-tree RRT* -like sampling-based planner applying on mobile robotic systems.** IEEE/ASME transactions on mechatronics, 23, n. 6, p. 2568-2578, 2018.

CHEN, Y.-C.; QI, H.; LIU, X., 2005, **Mas-based pursuit-evasion algorithm under unknown environment**. IEEE. 265-269.

CHENG, P., LAVALLE, S. M., 2002, **Resolution Complete Rapidly-Exploring Random Trees.**

DONG, J.; ZHANG, X.; JIA, X., 2012, **Strategies of pursuit-evasion game based on improved potential field and differential game theory for mobile robots**. IEEE. 1452-1456.

FAN, J.; RUAN, J.; LIANG, Y.; TANG, L., 2009, **A pso solution for pursuit-evasion problem of randomly mobile agents**. IEEE. 4032-4035.

GUZMAN, J., 2021 **Distância entre dois pontos – Fórmula e exemplos.**

HONG-YAN, S.; ZHI-QIANG, S., 2010, **Study on a solution of pursuit-evasion differential game based on artificial fish school algorithm**. IEEE. 2092-2096.

LATOMBE, J.-C., 1991, **Robot Motion Planning. Springer Science + Business Media New York**.

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

LAVALLE, S. M., 2006, **Planning algorithms.**

LIMA, V. G., 2019, **Planejamento de trajetórias para quadricópteros em tarefas de perseguição.**

MASHAYEKHI, R.; IDRIS, M. Y. I.; ANISI, M. H.; AHMEDY, I., 2020, **Hybrid rrt: A semi-dual-tree rrt-based motion planner.** IEEE Access, 8, p. 18658-18668.

MCNABB, M., 2019, **DRONEII: Tech Talk – Unraveling 5 Levels of Drone Autonomy.** Disnponível em: https://dronelife.com/2019/03/11/droneii-tech-talk-unraveling-5-levels-of-drone-autonomy/

PATEL, A., 1997, **Amit's A* Pages.** Disponível em: http://theory.stanford.edu/~amitp/GameProgramming/index.html

RAVIKIRAN, A., S., 2023. **A* Algorithm Concepts and Implementation.**

SHENG, W.; LI, B.; ZHONG, X., 2021. **Autonomous Parking Trajectory Planning with Tiny Passages: A Combination of Multistage Hybrid A-Star Algorithm and Numerical Optimal Control**.

SHOLES, E.; 2007. **Evolution of a UAV Autonomy Classification Taxonomy**.

SILVA, L.; 2016. **Distância entre dois pontos no espaço.**

SOUSA, S. K. A., 2017, **Planejamento de movimento para robôs móveis baseado em uma representação compacta da *rapidly-exploring random tree* (RRT).**

UBIRATAN, E., 2015, **A origem dos VANTs.**

WADHWANI, P.; LOOMBA, S., 2021, **Commercial Drone Market size worth over $55 BN by 2027**. Global Market Insights.

XUE, Y.; ZHANG, X.; JIA, S.; SUN, Y. *et al.*, 2017, **Hybrid bidirectional rapidly-exploring random trees algorithm with heuristic target graviton**. IEEE. 4357-4361.

ZENG, X.; CAI, W.; YANG, L.; ZHU, Y., 2018, **On the Orbital Pursuit-Evasion Games with Low Constant Thrust-to-Mass Ratio**. IEEE. 1655-1660.

ZHENGHUA, H.; QIUBO, Z.; CHUNYA, T.; KUAI, W., 2017, **Application of rapidly exploring random tree algorithm in the scene of road network with scheduled tracks**.