INSTITUTO POLITÉCNICO DE BRAGANÇA
Escola Superior de Tecnologia e Gestão

Planeamento de Rotas em Veículos Aéreos não Tripulados

Kévin Luzio Gonçalves - 31903

Dissertação realizada sob a orientação de

Profª. Ana Isabel Pereira

Mestrado em Engenharia Industrial

2020 - 2021

Planeamento de Rotas em Veículos Aéreos não Tripulados

Dissertação do

Mestrado em Engenharia Industrial

Escola Superior de Tecnologia e Gestão

Kévin Luzio Gonçalves - 31903

2020 - 2021

Resumo

Os Veículos Aéreos Não Tripulados(VANTs) são aeronaves que não têm um piloto a bordo, mas necessitam de um controlador no solo, podendo ser totalmente autónomos.

Querendo que um ou múltiplos VANTs se desloquem de um ponto A a um ponto B num menor número de passos possível sem colidir com nenhum objeto ou com os outros VANTs surge a necessidade de programação de um algoritmo para fazer o planeamento da sua trajectória. Neste trabalho foi escolhido o algoritmo A* por ser um algoritmo bastante usado nesta aplicação em específico e tem-se demonstrado bastante eficiente. Propondo ambientes com diferentes complexidades e um diferente número de VANTs, pretende-se desenvolver um algoritmo com base no método A* para Matlab de forma a serem feitas simulações em diferentes ambientes simulados.

Os resultados obtidos são comparados e discutidos, verificando assim a eficiência do algoritmos nas diferentes simulações.

É possível concluir que o algoritmo obteve bons resultados em termos de custo e que foi capaz de atingir os seus objetivos em todas as simulações.

Palavras-chave: : VANT, Otimização, Algoritmo A*, MATLAB, Planeamento de Trajetórias

Abstract

Unmanned Aerial Vehicles (UAVs) are aircrafts that don't have a pilot on board, needing a controller on the ground, but can be totally autonomous.

Wanting one or multiple UAVs to travel from point A to point B in the less amount of steps possible without colliding with any object or with each other comes the need to program an algorithm to be in charge of it's path planning.

In this paper we choose A* algorithm because it is an algorithm well used in this application in specific and has proved to be quite efficient.

Proposing envoirements with different level of complexity and different number os UAVs, it is intended to develop an algorithm with A* as it's base for Matlab in order to make simulations with a set of UAVs.

The obatined results will be compared and discussed, thus checking the efficiency of the algorithm in the different simulations.

It is possible to conclude that the algorith had good results in terms of cust and was able to achieve it's goals in every simulations.

Keywords: UAV, Optimization, A* Algorithm, Matlab, Path Planning

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Neste capítulo é feita uma introdução ao tema estudado. Vão ser abordadas aplicações em diversas áreas e falar um pouco dos graus de autonomia dos VANTs e dos algoritmos de planeamento de trajetórias já conhecidos. De seguida serão indicados os objetivos da dissertação e a estrutura do relatório.

1.1 Introdução

Com a sociedade a evoluir a tecnologia evolui também. Hoje em dia são produzidos Veículos Aéreos Não Tripulados (VANTs) com tamanhos variados e com capacidade de nos fornecer informações que antes não era possível [7].

Tem-se verificado um aumento de desastres naturais, quer seja terramotos, furacões, cheias ou incêndios, havendo situações possíveis de evitar e outras em que os danos já são esperados. Com a evolução dos VANTs é possível ajudar na maioria dos casos, nomeadamente na monitorização das florestas, podendo detetar rapidamente ignições florestais ou aceder ao interior de escombros de um edifício e procurar vitimas dentro destes [23].

Os VANTs podem ser teleguiados por um utilizador ou serem autónomos. Para se deslocarem num ambiente desconhecido usam sensores para mapear potenciais obstáculos ou pontos de interesse. Quando são autónomos possuem procedimentos computacionais no seu sistema que lhe permita interpretar os dados obtidos por sensores de ultrasom,

câmaras, entre outros. Estes algoritmos vão permitir ao VANT navegar pelo ambiente, recolhendo os dados que precisa sem ficar bloqueado ou colidir com algum obstáculo [7] Estes procedimentos denominam-se algoritmos de planeamento de trajetórias. .

Alguns algoritmos já utilizados são, por exemplo, o *Glow-worm swarm* [8], A*, *Particle Swarm Optimization* (PSO) [29], *Penguin Search Optimization Algorithm* (PeSOA) [29], *greedy 2-opt* [17], entre outros [8].

1.2 Objetivo

O principal objetivo deste trabalho visou a optimização do planeamento de trajetórias utilizando o algoritmo A* com múltiplos VANTs. De modo a testar a eficácia do algoritmo foram criados diversos mapas de simulação contendo complexidades diferentes. Foram testadas três situações diferentes:

- Situação I: Apenas um VANT a mover-se no espaço;
- Situação II: Dois VANTs a mover-se no espaço sequencialmente;
- Situação III: Dois VANTs a mover-se no espaço paralelamente.

Como objetivo final pretende-se avaliar a eficácia e eficiência do algoritmo A* nas situações descritas anteriormente, para isso iremos verificar se o algoritmo encontra a distância mínima entre os dois pontos escolhidos sem colidir com os obstáculos ou com o outro objeto que se move no mesmo espaço e verificando o tempo que o algoritmo demora a executar o algoritmo.

1.3 Estrutura do Relatório

O presente relatório encontra-se estruturado em seis capítulos.

No primeiro capítulo é realizada uma contextualização do tema, expressa a motivação para realização deste trabalho e são apresentados os objetivos do estudo.

No segundo capítulo é apresentada uma contextualização acerca dos veículos aéreos não tripulados, incluindo a sua utilização, autonomia e os vários tipos de VANTs.

No terceiro capítulo é apresentada uma contextualização acerca do planeamento de trajetórias, incluindo os seus princípios básicos e os seus objetivos. Serão ainda descritas algumas aplicações em VANTs de alguns algoritmos já conhecidos.

No quarto capítulo é apresentada uma contextualização do algoritmo A*, incluindo as diversas variantes do algoritmo implementadas. Iremos também descrever o programa MATLAB que foi o utilizado para implementação deste algoritmo computacionalmente, onde iremos abordar da sua história, linguagem e interface.

No quinto capítulo são apresentados e discutidos os resultados obtidos pelos algoritmos nas diversas situações, bem como uma discussão desses resultados de forma a avalia o seu funcionamento. Por fim no sexto capítulo são apresentadas as principais conclusões e uma perspectiva para trabalhos futuros.

Capítulo 2

Veiculos Aéreos Não Tripulados (VANTs)

Neste capítulo são descritas características dos Veículos Aéreos Não Tripulados(VANT). Será apresentado o uso de VANTs ao longo da história, descrevendo da sua utilidade e capacidade de trabalhar autonomamente. Tal como o seu nome indica, os VANTs são aeronaves que não necessitam de um piloto a apoiar a sua tripulação. Assim podemos dizer que temos um sistema composto por três componentes, o controlador terrestre, o VANT e o sistema de comunicação entre eles. Os VANTs têm duas maneiras de ser comandados, por controlo remoto através de um operador no solo ou autonomamente através de procedimentos computacionais incorporados no seu sistema [19]. Apesar de nos dias de hoje os VANTs serem comuns no mercado civil estes foram inicialmente projectados para aplicações militares. Os VANTs têm muitas aplicações para a pessoa comum. Podem ser utilizados para actividades recreativas desde tirar fotografias e fazer filmagens ou apenas utiliza-los para aeromodelismo, usos agrícolas para apoio nas atividades agrícolas, uso cientifico para investigação nas mais diversas áreas e até mesmo em missões de busca e salvamento com recurso a câmaras térmicas.

2.1 Utilização de VANTs ao longo da história

A primeira utilização de VANTs foi em julho de 1849 num ataque da Áustria a Veneza. Neste caso foram utilizados balões que transportavam bombas que seriam largadas quando um temporizador chegasse ao fim. Pode-se dizer que teve pouco sucesso uma vez que poucas bombas acertaram na cidade, isto deveu-se exclusivamente à mudança de ventos [11].

Durante a primeira guerra mundial e através do uso do controlo de rádio foi possível inventar uma aeronave remotamente tripulada cujo propósito era embater nos *zeppelins* inutilizando-os [25].

Durante a segunda guerra mundial Reginal Denny propôs que aeronaves controladas remotamente fossem usadas para treinar o exército Norte Americano tendo por isso desenvolvido ainda mais as tecnologias de controlo por rádio.

Durante a guerra fria os VANTs foram utilizados para recolher informações acerca de explosões radioativas, tais como o impacto das ondas de choque destas. Estes VANTs eram comandados durante a descolagem e aterragem por controladores em *Jeeps*. Posteriormente o exército norte Americano decidiu utilizar VANTs equipados com câmaras para espiar países como o Vietname, China e Coreia do norte. Em 1982 a força área israelita começou a utilizar VANTs ao lado de aeronaves tripuladas de modo a obter uma vantagem nos céus da Síria. Enquanto os VANTs serviam de isco, ou funcionavam como *jammers* eléctricos, os pilotos das outras aeronaves conseguiam abater os pilotos Sírios [30].

Um dos problemas dos VANTs era a sua autonomia, sendo a sua maior fraqueza e o factor limitador da sua utilização. Em 1950 Raytheon sugeriu que fosse criada uma aeronave movida a microondas. Esta teria uma antena rectificadora com diversos díodos que converteria as microondas enviadas de terra em electricidade fornecendo assim electricidade que permitiria o VANT manter-se no ar. Esta ideia era uma alternativa aos satélites para estudar dados atmosféricos. Apesar de ser uma ideia apelativa não teve resultados

uma vez que a antena era muito pesada e como os satélites terrestres tinham mais entusiastas esta tecnologia deixou de ser estudada. Só em 1982 é que a NASA publicou um design muito mais leve da antena o que permitiu ao centro de pesquisa e comunicações canadiano alimentar um VANT integrado no projecto *SHARP* [10].

Na década de 1979, Paul B. MacCready e a sua companhia criaram o primeiro VANT movido a energia solar. Apesar das células fotovoltaicas serem bastante ineficientes e a energia que o sol produzia por unidade área ser pequena, eles criaram um VANT leve o suficiente para que os seus motores eléctricos lhe permitissem permanecer no ar [14].

A 18 de Maio de 2006 a *FAA* emitiu um certificado que permite o uso de VANTs no espaço aéreo civil para efeitos de busca e salvamentos [5].

2.2 Autonomia

Como foi referido anteriormente os VANTs podem ser remotamente pilotados ou autónomos. Nos VANts autónomos é necessário que estejam equipados com sensores que me permitam mapear o meio exterior de modo a navegar nele [7].

2.2.1 Princípios básicos

Para se obter um robô autónomo temos de decompor cada uma das suas acções em estados. Assim, podemos criar códigos para máquinas de estado finitas, árvores de comportamentos ou planeamento de tarefas. Um exemplo disto é o controlo *PID* que permite a um quadcoptero manter-se nivelado no ar utilizando uma combinação de acelerómetros que permitem calcular a velocidade a que os motores devem trabalhar.

Alguns exemplos destes planeadores são algoritmos tais como planeamento de caminho que determinam o caminho óptimo para um robô cumprir com os seus objectivos, tais como viajar de um ponto inicial a um final, evitar obstáculos, mínimo consumo de combustíveis, entre outro [26].

2.2.2 Graus de autonomia

Segundo a Organização Internacional de Parametrização um robô autónomo define-se como um robô capaz de concretizar as suas tarefas baseando-se no seu estado e no meio em que está, sem necessitar de intervenção humana [24].

Nem todos os VANTs são iguais, e dependendo do seu propósito pode ser vantajoso que estes sejam totalmente autónomos ou remotamente controlados. Caso pretendamos mapear um terreno a partir do ar, não necessitamos de ter um operador a controlar o VANT, sabendo o robô os limites do terreno é capaz de o mapear todo sozinho. No caso de estarmos a utilizar um VANT para transmitir em direto, por exemplo, um desfile ou um concerto, queremos ter um VANT com pouca autonomia uma vez que o operador é que sabe onde focar e que imagens captar no momento.

A Tabela 2.1 descreve o grau de autonomia dos VANTs. Conforme a autoridade do piloto vai descendo, a autonomia do VANT vai sendo maior.

Tabela 2.1: Tabela que relaciona o controlo que o piloto com o controlo do VANT [6]

Controlo do piloto	**Controlo do VANT**
Total	Nula
Total	Aconselha quando pedido
Aceita ou nega os conselhos	Aconselha
Aceita ou nega os conselhos e autoriza ações	Aconselha e, se autorizado, toma ações
Nega os conselhos	Aconselha e toma ações a menos que seja negado
Interruptor	Total

Quando o piloto tem controlo total das ações do VANT este tem não tem autoridade e portanto não se considera autónomo. O piloto pode ter autoridade total e o VANT ter autonomia para aconselhar acerca de trajetórias, tempo de voo, entre outros, cabendo ao piloto tomar ações conforme esses conselhos. O VANT pode ter autonomia e dar informações ao piloto, tendo este apenas de os aceitar ou recusar e realizar as ações. O VANT pode ter autonomia para informar e tomar ações caso o piloto autorize. O VANT

pode ter autonomia para informar e tomar ações, cabendo ao piloto apenas negar essas ações e conselhos. O VANT pode ser totalmente autónomo e o piloto apenas serve para o desligar no caso de decisões que comprometam a sua função.

Na Figura 2.1 podemos ver um exemplo de um VANT sem grau de autonomia que é controlado por um utilizador no solo. Na Figura 2.2 temos o oposto, o exemplo de um VANT autónomo não controlado pelo seu utilizador.

Figura 2.1: DJI Mavic, exemplo de um VANT controlado pelo utilizador no solo sem grau de autonomia [1]

2.3 Tipos de VANTs

Os VANTs podem ser divididos em enumeras categorias, se pensarmos no seu uso, podemos falar de VANTs para fotografia/vídeo, mapeamento aéreo, inspeção de vias, vigilância de florestas, procura e salvamento, entre outros [23].

De modo a englobar todos os VANTs num pequeno número de categorias podemos categoriza-los através da base da sua plataforma aérea.

Nesta secção iremos falar acerca dos quatro grandes tipos de plataformas aéreas usados para os VANTs, são eles, VANTs de múltiplos rotores, VANTs de asas fixas, VANTs de

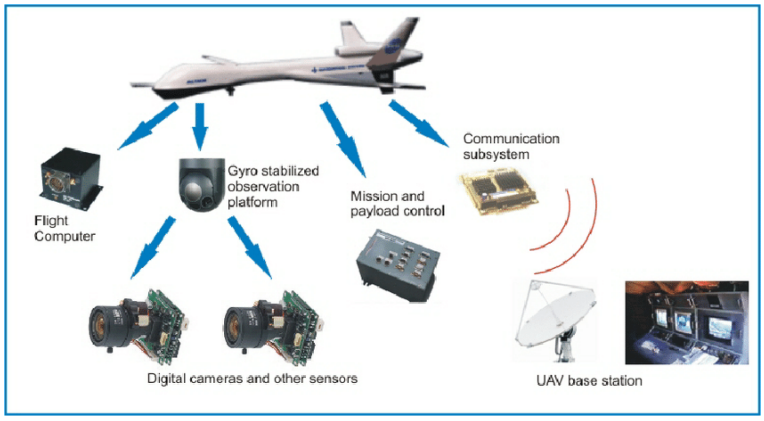

Figura 2.2: Exemplo de um VANT não controlado pelo utilizador com diagrama dos seus componentes [20]

rotor único e VANTs de asas fixas híbridos (VTOL).

2.3.1 VANTs de múltiplos rotores

Este tipo de VANTs é o mais comum a nível comercial pois são muito utilizados por profissionais e pelo utilizador comum. Isto deve-se à sua variada gama de preços. Dependo do uso pretendido, existem VANTs numa variada gama de preços de modo a satisfazer o utilizador comum ou o utilizador profissional. Como é óbvio para haver esta diferença de valores os componentes dos VANTs não são os mesmos, também não se podendo esperar a mesma vida útil ou a resistência do material. Este tipo de VANTs são os mais utilizados para aplicações como fotografia aérea ou videovigilância aérea.

Apesar de serem considerados VANTs de múltiplos rotores podemos ainda classifica-los conforme o número de rotores que possuem, se tiverem três rotores, chamam-se tricopteros, se tiverem quatro rotores chamam-se quadcopteros, se tiverem seis rotores chamam-se hexacopteros e se tiverem oito rotores chamam-se octopteros, destes quatro, os quadcopteros são os mais populares e mais utilizados.

Apesar de serem fáceis e baratos de produzir este tipo de VANTs tem várias desvantagens. As principais desvantagens são o tempo de voo, resistência e velocidade. Não costumam ser utilizados em projetos a larga escala como mapeamento aéreo pois gastam

a maioria da sua energia a tentar combater a gravidade e a manterem a sua estabilização no ar. Devido a isto a maioria destes VANTs costumam ter um tempo de voo de 20 a 30 minutos com uma carga de bateria [3]. A Figura 2.3 representa o modelo mais comum entre os VANTs de múltiplos rotores.

Figura 2.3: Exemplo de um quadcoptero, o tipo mais comum de VANTs de múltiplos rotores [22]

2.3.2 VANTs de asa fixa

Este tipo de VANTs têm um design e construção muito diferente dos VANTs de múltiplos rotores, em vez de terem braços com os rotores montados neles, têm asas e costumam ter um rotor atrás para propulsão. Ao contrário dos VANTs de múltiplos rotores os VANTs de asa fixa não se conseguem manter estacionários no ar, sendo necessário estarem sempre a mover-se no seu rumo predefinido ou a serem guiados pelo utilizador no solo.

A maioria dos VANTs de asa fixa possuem um tempo de voo de poucas horas, no entanto há versões com motores de combustão interna que lhe permitem estar até 16 horas no ar [3]. Como não conseguem estar estacionários não são boas opções para quem queira fazer fotografia aérea, no entanto são ideias para mapeamento aéreo. Uma desvantagem deste tipo de VANTs é que não são tão versáteis como os VANTs de múltiplos rotores, isto é, o utilizador necessita de uma "pista"ou catapulta para o colocar no ar e para o aterrar é necessário um pára-quedas ou uma rede para evitar causar danos neste.

Uma vantagem que um VANT de múltiplos rotores tem é que o utilizador aprende facilmente a comanda-lo a uma baixa altitude e sem risco de danos, um VANT de asa fixa é mais complexo de pilotar e o utilizador necessita de treino para adquirir as competências necessárias para o controlar e não danificar [3]. Podemos ver na Figura 2.4 um exemplo de um VANT de asa fixa.

Figura 2.4: Exemplo de um VANT de asa fixa [4]

2.3.3 VANTs de único rotor

Este tipo de de VANTs são os mais parecidos em termos de estrutura e design com os helicópteros modernos.

Ao contrario dos VANTs de múltiplos rotores, estes têm apenas um rotor maior, na sua parte superior, e um estabilizador com um rotor mais pequeno, na cauda. Os VANTs de único rotor são mais eficientes que os VANTs de múltiplos rotores, conseguem manter-se no ar durante mais tempo e até podem optar por ter um motor de combustão interna.

Em termos de aerodinâmica como não tem tantos braços e menos rotores é muito mais fácil de manter a sua estabilidade no ar. No entanto, estes VANTs vêm com um custo muito superior e são muito mais complexos que os VANTs de múltiplos rotores.

Como os VANTs de unico rotor necessitam de ter hélices maiores, há o risco de ferimentos no caso de má utilização do VANT ou em caso de acidente. Apesar de não precisarem de uma "pista"ou catapulta para serem postos no ar, o utilizador precisa de

treino para os controlar corretamente como acontece com os VANTs de asa fixa [3]. Um exemplo deste tipo de VANTs está representado na Figura 2.5.

Figura 2.5: Exemplo de um VANT de rotor único [22]

2.3.4 VANTs de asa fixa hibridos (VTOL)

Este tipo de VANTs combina os benefícios dos modelos de asa fixa, que têm um tempo de funcionamento alto, com o modo de voo estático dos modelos com rotores, como apresentado na Figura 2.6. Apesar de no inicio dos testes este tipo de VANTs não terem tido muito sucesso, com o melhoramento dos sensores giroscópios e acelerómetros mais modernos este tipo de VANTs voltou a ser investido. Este tipo de VANTs tem uma mistura de automação e planar manual, isto é, o VANT levanta do solo verticalmente como o VANT de múltiplos rotores mas durante o voo funciona como o VANT de asa fixa, diz-se autónomo no sentido em que usa acelerómetros e giroscópios para manter o VANT estabilizado no ar [3]. A Figura 2.6 representa um exemplo de um VANT de asa fixa híbrido.

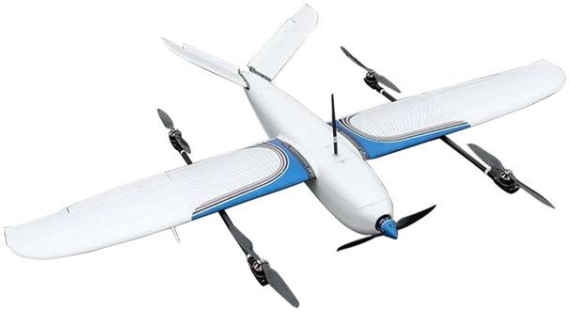

Figura 2.6: Exemplo de um VANT de asa fixa híbrido [28]

Capítulo 3

Planeamento de trajectórias

O planeamento de trajectórias define, o cálculo de rotas entre dois pontos. No entanto quando falamos de robôs, estes não têm capacidade intrínseca evitar obstáculos e calcular distâncias é algo que o nosso cérebro faz quase que autonomamente, já os robôs precisam de uma série de sensores e algoritmos para poderem localizar espacialmente esses obstáculos e evita-los. Só este aspecto já torna o trabalho de robôs teleguiados uma tarefa com alguma dificuldade, se estivermos a falar de robôs autónomos ainda mais difícil uma vez que é necessário um algoritmo que permita ao robô detetar e evitar obstáculos estáticos e muitas vezes dinâmicos.

3.1 Princípios básicos

O planeamento de trajectórias pode ajudar a resolver problemas em diversas áreas de trabalho, no entanto o seu funcionamento é semelhante, escolher o menor número de passos para ir do ponto A ao ponto B ou o tempo que se demora a chegar [12].

O planeamento de trajectórias pode ser aplicado a diversos ambientes, sejam eles ambientes totalmente conhecidos, parcialmente conhecidos ou totalmente desconhecidos. Dependendo do ambiente o robô pode precisar apenas de calcular a rota final, ou ir calculando a rota conforme vai obtendo informação do meio que o rodeia através de sensores [21]. Os obstáculos nestes ambientes podem ser de dois tipos, estáticos e dinâmicos, os

primeiros são aqueles que mantêm a sua localização e orientação ao longo do tempo e os segundos são aqueles nos quais a sua localização, orientação ou ambas variam ao longo do tempo [27].

Pode assim dizer-se que o planeamento de trajectórias se pode chamar local ou global. Local é quando o robô não tem dados do seu ambiente e obstáculos ou da localização de obstáculos dinâmicos e os vai adquirindo enquanto percorre o espaço, reajustando a sua trajectória ao longo do caminho. Global é quando se sabe o ambiente e a localização de todos os obstáculos e estes são estáticos, neste caso o robô já tem a sua trajectória quando começa a viajar e vai directamente do ponto inicial ao final [27].

3.2 Fatores a considerar no planeamento de trajetórias

O ambiente onde o robô viaja tem quatro principais componentes, o ponto inicial, o ponto final, a informação de obstáculos e o espaço livre. Os pontos inicial e final têm de estar localizados no espaço livre. Assim, o objectivo do planeamento de trajectórias é encontrar um caminho contínuo que leve o robô do ponto inicial ao ponto final viajando sempre no espaço livre. Este caminho depende do algoritmo utilizado uma vez que cada algoritmo dá prioridade a diferentes objectivos. Os objectivos mais populares são:

- Quando o caminho mais curto implica mudanças de direcção súbitas, é vantajoso ir por um caminho mais longo, mas que mantenha o momento linear do robô alto, sendo assim um caminho mais rápido;
- O robô deve manter o máximo de distância possível dos obstáculos, assim caso o robô tenha de fazer uma mudança de direcção súbita terá sempre espaço para o fazer;
- O caminho deve evitar curvas apertadas, curvas apertadas fazem o robô perder o seu momento linear, levando assim a uma perda de rapidez [12].

Além destes casos o planeamento de trajectórias pode ser utilizado para viajar do ponto inicial ao ponto final não pelo caminho mais rápido mas sim visitando o espaço todo. Isto implica visitar cada ponto do espaço pelo menos uma vez. Neste caso os objectivos do robô mudam. Em vez de se obter o caminho mais curto ou o mais rápido entre os dois pontos, o objectivo é obter o mínimo de pontos visitados, percorrendo o espaço todo. Pode ainda ser visitar todos os pontos o mais rápido possível mesmo que implique passar múltiplas vezes no mesmo ponto, neste caso o mais importante é o robô manter o seu momento o que permitirá este ser mais eficiente [27].

3.3 Estado da arte em planeamento de trajetórias

Nesta secção vão ser analisados vários trabalhos publicados por diferentes autores em diferentes situações, mas nos quais todos utilizam diferentes algoritmos para optimizar o planeamento de trajetórias de VANTs.

No estudo com o titulo *Decision Making Support of UAV Path Planning for Efficient Sensing in Radiation Dose Mapping* os autores tentam descobrir o algoritmo que mais eficientemente percorra uma série de pontos num mapa. Os algoritmos escolhidos foram o *greedy 2-opt* e o *flood fill* e testaram cada um deles separadamente. O método proposto pelos autores baseia-se em juntar ambos os algoritmos, o *flood fill* para agrupar os pontos em sub-rotas e o *greedy 2-opt* para determinar o caminho mais rápido entre as sub-rotas. A conclusão a que chegaram foi que apesar de o *greedy 2-opt* teve a menor distância total, no entanto demorava bastante tempo devido ao número de curvas de 90º. O *flood fill* sozinho teve a distância e o tempo maiores, sendo assim a pior opção. Já a junção dos dois algoritmos apesar de dar uma distância superior à do *greedy 2-opt* sozinho obteve um menor número de curvas de 90º tento reduzido o tempo para quase metade e sendo assim o algoritmo escolhido [17].

Usman et al. No estudo com o titulo *UAV Reconnaissance using Bio-Inspired Algorithms: Joint PSO and Penguin Search Optimization Algorithm (PeSOA) Attribute* os autores procuram o algoritmo que tenha menos probabilidade erro quando são enviados

vários grupos de *UAVs* para procurar a localização de diferentes quantidades de alvos. Os algoritmos utilizados são o *Particle Swarm Optimization(PSO)* básico e o *PSO* mas com atributos do *Penguins Search Optimisation Algorithm (PeSOA)* que são a divisão dos *UAVs* em grupos e a transferência de localização dos alvos entre cada *UAV* do grupo e quando estes voltam à base, a transferência dessas localizações para os outros grupos. A conclusão obtida foi que quanto maior o número de alvos mais o erro subia. No entanto, em todos os testes o algoritmo *PSO* com os atributos do algoritmo *PeSOA* obtiveram menor taxa de erro, sendo assim a melhor opção face à utilização de apenas o algoritmo *PSO* [29].

No estudo com o titulo *Three Dimensional Path Planning for UAVs in Dynamic Environment using Glow-worm Swarm Optimization* os autores propõem um método para optimização de *path planning* de *UAVs* baseado em *Glow-worm Swarm Optimization*. Foram feitas três experiências, a primeira num ambiente que tem obstáculos e pontos estáticos predefinidos. A segunda num ambiente que tem obstáculos e pontos estáticos predefinidos e são criados obstáculos de diferentes dimensões espalhados aleatoriamente pelo espaço. A terceira num ambiente que tem obstáculos estáticos predefinidos, os pontos movem-se aleatoriamente pelo espaço e são criados obstáculos de diferentes dimensões espalhados aleatoriamente pelo espaço. A conclusão chegada foi que o algoritmo funcionou mas que quanto maior é a complexidade da experiência mais altos são os custos de operação, bem como o tempo demorado[8].

No estudo com o titulo *A hybrid algorithm of particle swarm optimization,metropolis criterion and RTS smoother for path planning of UAVs* os autores tentam resolver os defeitos do algoritmo *PSO* em várias dimensões (duas e três). O método proposto foi misturar o *RTS smoother* com o *PSO* formando o *RPSO* e de seguida comprara-lo com outros algoritmos. Assim foi misturado o *Metropolis criterion* com o *PSO* original formando o *MPSO*, a este foi ainda aplicado um filtro *Kalman* tendo-lhe chamado *KPSO*. Para comparar com outro algoritmo que não fosse o *PSO* foi utilizado o *Artificial Bee Colony (ABC)* e misturado também com o *RTS smoother* formando o *RABC*. Apesar de o algoritmo poder ser aplicado a várias dimensões os autores propuseram um circuito

de obstáculos estáticos que se assemelham a montanhas por onde os *UAVs* teriam de navegar em três dimensões e com diferentes tamanhos de enxames e ao longo de várias iterações. A conclusão chegada foi que o *RTSO* foi o algoritmo com melhores resultados e melhor optimização ao longo das iterações, seguido pelo *KPSO*. O *MPSO* e o *RABC* tido resultados semelhantes seguidos de perto pelo *PSO*, já o *ABC* foi o único que teve piores resultados na optimização ao longo das iterações bem como os piores resultados desta experiência[31].

No estudo com o titulo *Modified Central Force Optimization (MCFO) algorithm for 3D UAV path planning* os autores propõem uma modificação ao algoritmo *Central Force Optimization (CFO)* tentando assim resolver alguns defeitos deste algoritmo optimizando assim o planeamento de trajetórias de *VANTs*. Esta modificação junta o algoritmo original do *CFO* com o *PSO operator* e o operador mutação do algoritmo genético. Depois disto tentaram ainda optimizar o tempo de voo suavizando as curvas, ou seja, fazendo com que o *UAV* curva-se perto de cada ponto em vez de ir directamente a ele e nesse ponto parar e virar para a nova direcção. Como poderia haver obstáculos nessa curva eles ainda aplicaram um algoritmo que permiti-se ao *UAV* saber quando havia um obstáculo e nesse caso não aplicar a suavização. Os algoritmos testados foram o *CFO*, o *MCFO*, o *Genetic Algorithm (GA)*, o *PSO*, o *Firefly Algorithm(FA)* e o *Random Search (RS)*. De todos estes algoritmos foram tirados os valores médio, óptimo e pior, tendo em todos eles obtidos os valores mais satisfatórios o *MCFO* seguido de perto pelo *PSO* e de seguida o *GA* [2].

Na dissertação com o titulo *Application of a Mobile Robot to Spatial Mapping of Radioactive Substances in Indoor Environment* o autor pretende criar um método de mapeamento espacial de substâncias radioactivas. O autor aplicou dois métodos para ver o mais eficaz em termos de *path planning*, um método baseado na heurística e outro baseado no algoritmo genético. Para testar os algoritmos o autor propôs três cenários, o cenário A é um mapa 8x8 sem obstáculos, o cenário B é um mapa 8x8 com 2 obstáculos e o cenário C é um mapa 8x8 com 3 obstáculos. O objectivo é que o robô percorra todas as células possíveis nesse mapa no menor tempo possível, ou seja, visitar cada uma delas o menor número de vezes possível. A conclusão a que o autor chegou é que no cenário A o método

baseado na heurística é mais rápido face ao algoritmo genético, no entanto, conforme aumenta a complexidade (são testados os cenários B e C) o método baseado no algoritmo genético é muito mais rápido conforme o número de obstáculos aumenta [21].

Capítulo 4

Algoritmo A*

O algoritmo A* foi inicialmente criado para o projeto Shakey em 1963 [9], este projeto envolveu a construção de um robô autónomo capaz de planear as suas próprias ações. Este robô tinha como objetivo unir reconhecimento de padrões e as capacidades de memória de redes neuronais com programas de inteligência artificial desenvolvidos na época. O problema inicial era desenvolver uma sequência de pontos que o robô tinha de percorrer para evitar um único obstáculo no meio da sala onde estava e mais tarde múltiplos obstáculos. O robô percorreria a sala desde o ponto inicial até ao ponto final e guardava a localização dos obstáculos e a sua própria localização num modelo de grelha. Para esse propósito foi desenvolvido o algoritmo A* [18]. Mais tarde em 1968 Peter Hart et al, publicaram um livro onde explicam mais detalhadamente o funcionamento do algoritmo [9].

O algoritmo A* calcula o custo de um caminho tendo o utilizador definido os dois pontos, o inicial e o final, e os obstáculos.

O custo é calculado para cada ponto na matriz pelo número de movimentos necessários de um ponto inicial até um ponto final e é calculado pela soma da distância desse ponto ao ponto final mais a distância desse ponto ao ponto inicial, sem ter em conta os obstáculos.

Para controlar os pontos visitados o algoritmo guarda cada ponto em duas listas diferentes, a lista aberta e a lista fechada. À lista aberta são adicionados os pontos quando está a ser calculado o seu custo, depois o algoritmo analisa os vizinhos deste ponto e este é assim adicionado a lista fechada.

O cálculo do custo tem início no ponto inicial definido pelo utilizador, como neste caso estamos no ponto inicial, o custo será apenas a distância ao ponto final. De seguida o algoritmo verifica quais os pontos válidos, que não são obstáculos, directamente ligados ao primeiro ponto.

Para cada um desses pontos é calculado o custo e são adicionados à lista aberta por ordem decrescente de custo, ou seja, o ponto que tiver um custo menor será o primeiro da lista aberta e assim sucessivamente. Para o cálculo desse custo é utilizada a seguinte fórmula:

$$custo = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{4.1}$$

Onde (x_1, y_1) representam, respectivamente, as coordenadas no eixo dos xx e no eixo dos yy para o ponto que deu origem ao ponto atual e (x_2, y_2) representam, respectivamente, as coordenadas no eixo dos xx e no eixo dos yy para o ponto atual.

Quando todos os pontos adjacentes ao ponto inicial forem analisados este é adicionado à lista fechada. Para escolher quais os próximos pontos a analisar vamos escolher os vizinhos do ponto que estiver no topo da lista aberta que não sejam obstáculos nem estejam em nenhuma das listas anteriormente referidas.

Para o caso de os pontos da lista aberta terem o mesmo custo, é escolhido o que foi adicionado primeiro. E no caso de algum ponto não ter vizinhos válidos é adicionado à lista fechada.

São seguidos os passos referidos anteriormente até um dos vizinhos ser o ponto final. Neste caso, o algoritmo termina o procedimento.

O caminho é formado pelos vizinhos que lhe deram origem anteriormente. Como os pontos eram analisados conforme a ordem de custo, a soma dos custos garante que o caminho obtido é o mais curto [13].

4.1 Algoritmo A* aplicado a um VANT

Nesta secção será aplicado o algoritmo apresentado neste capítulo a um VANT. Visto ser utilizado apenas um VANT, o algoritmo utilizado é o A* descrito inicio deste capítulo.

Algorithm 1 : A* para um VANT

Adicionar o nó de partida a $Open$.
while $xNode$ e $yNode$ não correspondem às coordenadas do nó final **do**
(a) Adicionar os vizinhos sucessores do nó actual a $Open$
(b) Actualizar o antecessor, o $G(n)$, o $H(n)$ e o $F(n)$ do nós em $Open$
(c) Remover o nó actual de pen e adiciona-lo a $Closed$
(d) Seleccionar de $Open$ o nó que tenha menor $F(n)$.
(e) Calcular a distância do nó escolhido ao anterior e somar a Ct.
Escolher a sucessão de nós que origina um melhor valor para Ct.

Onde $(xNode, yNode)$ representam as coordenadas no eixo xx e yy, respectivamente, da posição do VANT, n representa o nó atual com as coordenadas $(xNode, yNode)$, $Open$ é uma matriz com duas colunas onde está contida a lista dos nós por analisar, $Closed$ é uma matriz com duas colunas onde está contida a lista dos nós analisados. $G(n)$ é uma função que calcula a distância do nó atual ao nó de partida, $H(n)$ é uma função que calcula a distância do nó atual ao nó final e Ct é o custo total mínimo entre o nó inicial e o nó final.

4.2 Algoritmo A* aplicado a dois VANTs no mesmo espaço movendo-se sequencialmente

Nesta secção será abordada a aplicação do algoritmo A* quando é introduzido um segundo VANTs no espaço. Quando dois VANTs se movem pelo mesmo espaço, um novo obstáculo é aí introduzido. Assim, é necessário ter em conta este obstáculo, para não haver colisões entre eles enquanto se movem. O algoritmo A* utilizado na secção anterior não é capaz de fazer o planeamento de trajectórias para dois VANTs por isso foi necessário modifica-lo para este funcionar com dois VANTs.

A ideia aplicada foi executar o algoritmo inicial da mesma maneira que este funcionou na secção anterior ao primeiro VANT. Obtida a rota do primeiro VANT, esta foi considerada como novos obstáculos. Tendo em conta os novos obstáculos foi aplicado novamente o algoritmo da secção anterior para o segundo VANT. Como a rota do primeiro VANT é considerada obstáculos, o segundo VANT vai evitá-la, percorrendo assim uma nova rota que evita a rota percorrida pelo primeiro VANT, evitando assim colisão entre ambos.

Algorithm 2 : A* para dois VANTs

Adicionar o nó de partida a $Open$.
while $xNode$ e $yNode$ não correspondam ás coordenadas do nó final **do**
(a)Adicionar os vizinhos sucessores do nó actual a pen
(b)Actualizar o antecessor, o $G(n)$, o (n) e o $F(n)$ do nó em $Open$
(c)Remover o nó actual de $Open$ e adiciona-lo a $Closed$
(d)Seleccionar de $Open$ o nó que tenha menor $F(n)$.
(e)Calcular a distância do nó escolhido ao anterior e somar a Ct.
Escolher a sucessão de nós que origina um melhor valor para Ct.
Definir a sucessão de nós anterior como obstáculos.
Adicionar o nó de partida a $Open_2$.
while $xNode_2$ e $yNode_2$ não correspondam ás coordenadas do nó final **do**
(a)Adicionar os vizinhos sucessores do nó actual a $Open_2$
(b)Actualizar o antecessor, o $G(n)$, o $H(n)$ e o $F(n)$ do nó em $Open_2$
(c)Remover o nó actual de $Open_2$ e adiciona-lo a $Closed_2$
(d)Seleccionar de $Open_2$ o nó que tenha menor $F(n)$.
(e)Calcular a distância do nó escolhido ao anterior e somar a Ct_2.
Escolher a sucessão de nós que origina um melhor valor para Ct_2.

Onde $(xNode, yNode)$ representam as coordenadas no eixo xx e yy, respectivamente, da posição do VANT 1, n representa o nó atual com as coordenadas $(xNode, yNode)$, $Open$ é uma matriz com duas colunas onde está contida a lista dos nós por analisar, $Closed$ é uma matriz com duas colunas onde está contida a lista dos nós analisados. $G(n)$ é uma função que calcula a distância do nó atual ao nó de partida, $H(n)$ é uma função que calcula a distância do nó atual ao nó final e Ct é o custo total mínimo entre o nó inicial e o nó final para o VANT 1.

Da mesma forma $(xNode_2, yNode_2)$ representam as coordenadas no eixo xx e yy, respectivamente, da posição do VANT 2, n representa o nó atual com as coordenadas

$(xNode_2, yNode_2)$, $Open_2$ é uma matriz com duas colunas onde está contida a lista dos nós por analisar, $Closed_2$ é uma matriz com duas colunas onde está contida a lista dos nós analisados e Ct_2 é o custo total mínimo entre o nó inicial e o nó final para o VANT 2.

4.3 Algotimo A* aplicado a dois VANTs no mesmo espaço movendo-se simultaneamente

Nesta secção será abordado o algoritmo A*, tal como foi descrito na secção anterior, mas com um novo objetivo, os dois VANTs moverem-se em simultâneo.

Como a posição de ambos os VANTs vai sendo diferente ao longo do espaço temos de garantir que estes não colidem, a opção escolhida para evitar isso foi, em cada iteração do algoritmo, considerar o primeiro VANT como um obstáculo para o segundo VANT e vice-versa. O primeiro VANT vai escolher a posição seguinte mais próxima do ponto final evitando os obstáculos do espaço e a posição em que se encontra o segundo VANT. O segundo VANT vai escolher a posição mais próxima seguinte mais próxima do ponto final evitando os obstáculos do espaço e a posição atual do primeiro VANT e libertando a sua posição anterior caso o primeiro VANT pretenda movimentar-se para essa posição. Assim, ao contrário do algoritmo da secção anterior, onde a rota toda do primeiro VANT é um obstáculo para o segundo VANT, apenas a posição atual de cada um dos VANTs é um obstáculo para o outro, garantindo que não colidem ao mesmo tempo que têm maior liberdade de movimentação pelo espaço.

Onde n_2 representa o nó atual com as coordenadas $(xNode, yNode)$ para o VANT 2.

4.4 MATLAB

MATLAB é um software de computação numérica desenvolvido pela Mathworks. Este programa permite manipulação de matrizes, desenhar gráficos de funções e de dados, implementação de algoritmos, criação de interfaces ao gosto de cada utilizador e interação com outros programas escritos em linguagens de programação diferentes [15]. Apesar deste

Algorithm 3 : Algoritmo A* para dois VANTs simultaneamente

Adicionar o nó de partida a $Open$.
Adicionar o nó de partida a $Open_2$.
while as coordenadas do nó final não estejam em $Closed$ ou $Closed_2$ **do**
(a)Adicionar os vizinhos sucessores de n a $Open$
(b)Adicionar os vizinhos sucessores de n_2 a $Open_2$
(c)Actualizar o antecessor, o $G(n)$, oH(n) e o $F(n)$ de n em $Open$
(d)Actualizar o antecessor, o $G(n)$, o $H(n)$ e o $F(n)$ de n_2 em $Open_2$
(e)Remover n de $Open$ e adiciona-lo a $Closed$
(f)Removemos n de $Open_2$ e removemos n_2 de $Open$.
(g)Seleccionar de $Open$ o nó que tenha menor $F(n)$.
(h)Seleccionar de $Open_2$ o nó que tenha menor $F(n)$.
(i)Adicionamos novamente n a $Open_2$ e n_2 a $Open$.
(j)Calcular a distância de n escolhido ao anterior e somar a Ct.
(k)Calcular a distância de n_2 escolhido ao anterior e somar a Ct_2.
Escolher a sucessão de nós que origina um melhor valor para Ct.
Escolher a sucessão de nós que origina um melhor valor para Ct_2.

programa estar voltado para a computação numérica, como foram criadas extensões que lhe permitem trabalhar e analisar dados em diferentes áreas como computação paralela, cálculo e optimização, estatística, inteligência artificial, entre outros[16].

4.4.1 Implementação do algoritmo

Nesta seção será descrito como cada algoritmo referido anteriormente foi implementado no programa MATLAB de modo a poder testá-lo computacionalmente e analizar os seus resultados.

Implementação do algoritmo no MATLAB

O algoritmo A* foi implementado em MATLAB tendo como base um código escrito por Paul Premakumar da empresa *The MathWorks, Inc.* Inicialmente o algoritmo pedia para o utilizador introduzir o tamanho do mapa, os obstáculos e os pontos de inicio e de fim e calculava o caminho para apenas um VANT. No sentido de simplificação foi removido tudo desse código que não fosse o algoritmo em si e foram adicionados os 3 mapas de estudo. Nesta fase foram definidos o ponto de inicio com coordenadas (1,1) e o ponto de

fim com as coordenadas (10,10).

O algoritmo implementado foi o descrito no Capítulo 4.1.

Implementação do algoritmo para dois VANTs sequencialmente

A implementação do algoritmo para dois VANTs sequenciais foi baseada no algoritmo anterior aplicado duas vezes, isto é, o segundo VANT vai aplicar o mesmo algoritmo considerado a rota do primeiro VANT como sendo obstáculos. Garantindo assim que não vão ser escolhidos os mesmo pontos para ambos os VANTs. Nesta fase foram definidos o ponto de inicio com coordenadas [1,1] e o ponto de fim com as coordenadas [10,10].

O algoritmo utilizado foi o descrito no Capítulo 4.2.

Implementação do algoritmo para dois VANTs simultaneamente

A implementação do algoritmo para dois VANTs implicou modificar o algoritmo para identificar o melhor caminho para ambos os VANTs ao mesmo tempo. Para isso foi adicionada uma condição dentro do algoritmo que fazia com que na mesma iteração o ponto onde estava o VANT 1 fosse removido temporariamente da lista aberta do VANT 2, evitando assim a escolha desse ponto pelo VANT 2. Na iteração seguinte o ponto onde estava o VANT 2 era removido da lista aberta do VANT 1, pois como este é o primeiro a mover-se não pode escolher a posição onde está o VANT 2, e o ponto onde o VANT 1 estava na iteração anterior fica novamente disponível na lista aberta do VANT 2, e vice-versa. Nesta fase foram definidos o ponto de inicio com coordenadas (1,1) e o ponto de fim com as coordenadas (10,10).

O algoritmo utilizado foi o descrito no Capítulo 4.3.

Capítulo 5

Resultados e discussão

Neste capítulo iremos analisar e discutir os dados e resultados obtidos pelo algoritmo apresentado para cada um dos mapas de simulação no caso de termos apenas um VANT a viajar no espaço, dois VANTs a viajar sequencialmente ou dois VANTs a viajar simultaneamente.

5.1 Mapas de simulação

Inicialmente foram criados três mapas com a mesma dimensão, 10x10, para análise e avaliação do algoritmo em estudo. Cada um deles com diferente número de obstáculos tornando-se cada vez mais complexos.

Estes mapas são representados no Matlab através de matrizes onde diferentes números correspondem aos obstáculos, espaço livre, ponto inicial e ponto final. Obstáculos correspondem ao número -1, o espaço livre corresponde ao número 2, o ponto inicial corresponde ao número 1 e o ponto final corresponde ao número 0.

Assim, podemos aplicar o algoritmo A* no programa Matlab, o algoritmo tem de encontrar a rota mais curta entre as posições da matriz com o número 1 e o número 0, evitando as posições da matriz com o número -1, ou seja, apenas podendo escolher as posições da matriz com o número 2.

5.1.1 Mapa 1

Este mapa, representado na Figura 5.1, tem dois obstáculos representados pelos círculos pretos que correspondem a doze pontos que o VANT não pode visitar, o ponto inicial é representado pelo losango azul e o ponto final é representado pelo pentágono vermelho, os restantes oitenta e oito quadrados correspondem aos pontos que o VANT pode visitar.

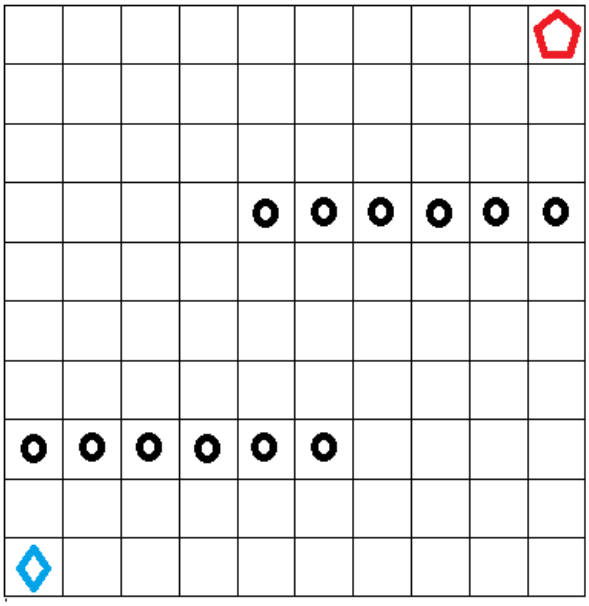

Figura 5.1: Primeiro mapa de simulação

5.1.2 Mapa 2

Este mapa, representado na Figura 5.2, tem quatro obstáculos representados pelos círculos pretos que correspondem a dezanove pontos que o VANT não pode visitar, o ponto inicial é representado pelo losango azul e o ponto final é representado pelo pentágono vermelho, os restantes oitenta e um quadrados correspondem aos pontos que o VANT pode visitar.

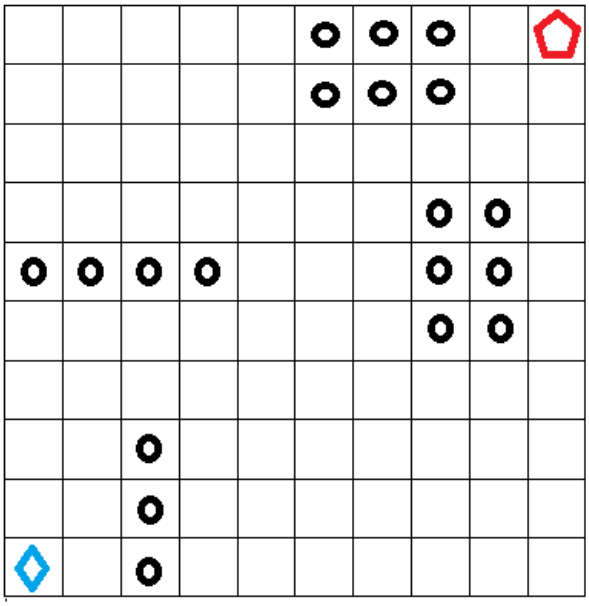

Figura 5.2: Segundo mapa de simulação

5.1.3 Mapa 3

Este mapa, representado na Figura 5.3, tem oito obstáculos representados pelos círculos pretos que correspondem a vinte e dois pontos que o VANT não pode visitar, o ponto inicial é representado pelo losango azul e o ponto final é representado pelo pentágono vermelho, os restantes setenta e oito quadrados correspondem aos pontos que o VANT pode visitar.

5.2 Resultados com apenas um VANT

Nesta secção iremos analisar os dados obtidos pelo algoritmo original com apenas um VANT a viajar no espaço. Tendo o algoritmo apenas de evitar obstáculos estáticos e encontrar o caminho mais curto entre o ponto inicial e o ponto final. Na Tabela 5.1 apresentamos o custo e o tempo de execução do algoritmo para apenas um VANT para todos os mapas de simulação

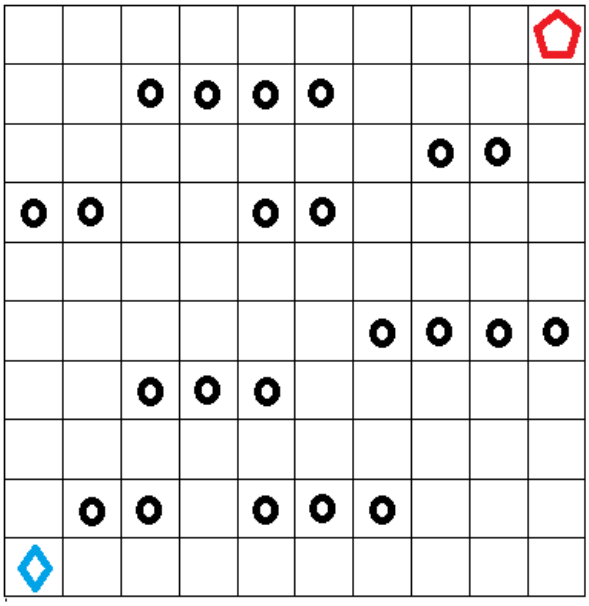

Figura 5.3: Terceiro mapa de simulação

Tabela 5.1: Resultados para um VANT

	Custo	Tempo de execução (s)
Mapa 1	19.31	4.71
Mapa 2	13.31	3.22
Mapa 3	14.49	3.72

A Figura 5.4 mostra a trajectória do VANT obtida pelo algoritmo que corresponde ao caminho mais curto no primeiro mapa de simulação.

A Figura 5.5 mostra a trajetória do VANT obtida pelo algoritmo como sendo o caminho mais curto no segundo mapa de simulação.

A Figura 5.6 mostra a trajetória do VANT obtida pelo algoritmo que corresponde ao caminho mais curto no terceiro mapa de simulação.

O algoritmo encontrou o caminho mais curto entre os dois pontos para cada um dos mapas de simulação e foi determinado o tempo de execução deste. Podemos considerar o tempo de execução aceitável e verificamos que está diretamente ligado ao custo, isto

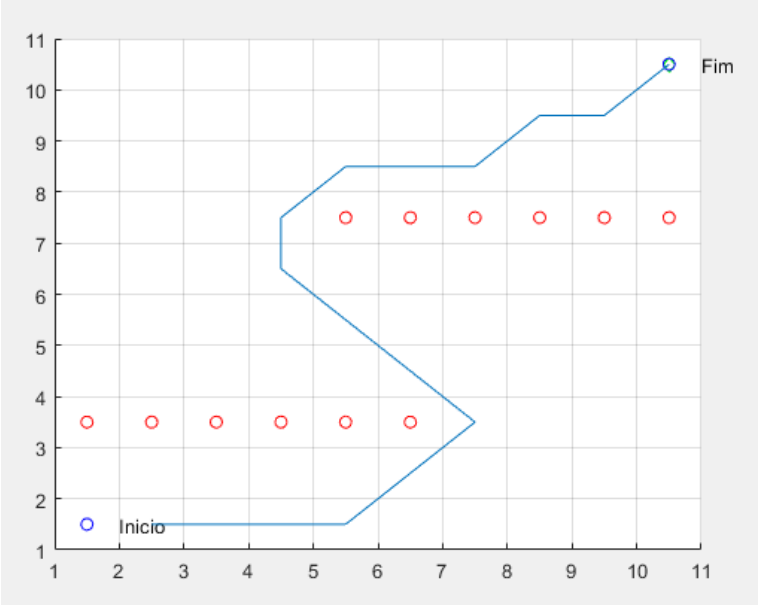

Figura 5.4: Trajetória para um VANT no primeiro mapa de simulação

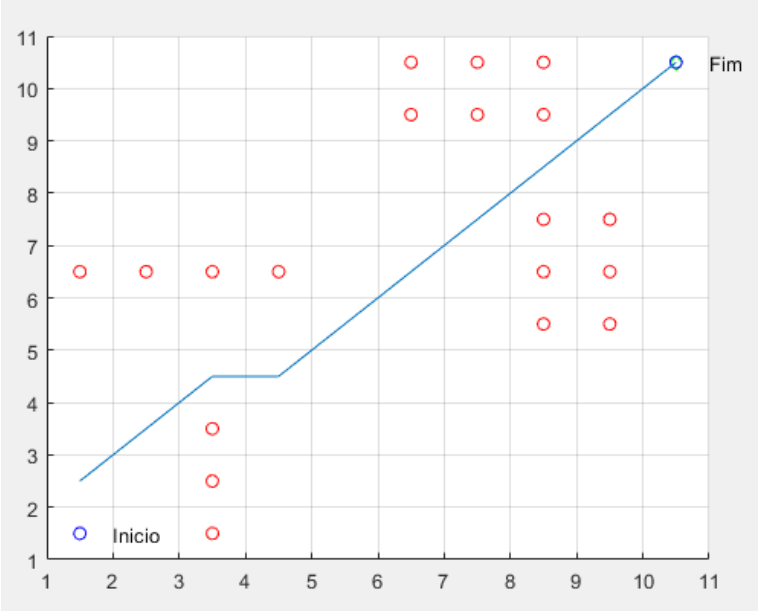

Figura 5.5: Trajetória para um VANT no segundo mapa de simulação

é, quanto maior o custo, maior o tempo de execução, como esperado. Estes dados serão usados para comparação com os algoritmos seguintes.

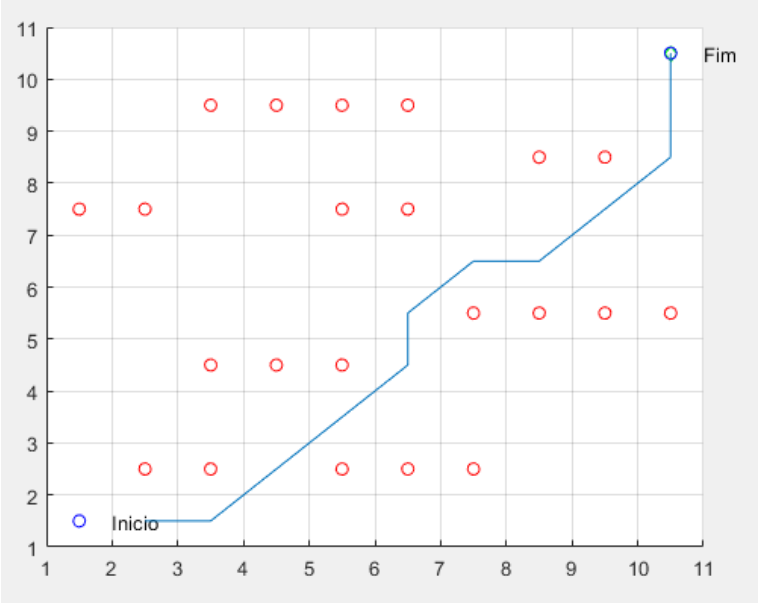

Figura 5.6: Trajetória para um VANT no terceiro mapa de simulação

5.3 Resultados com dois VANTs a deslocar-se sequencialmente

Nesta secção iremos analisar os dados obtidos pelo algoritmo modificado com dois VANTs a deslocar-se sequencialmente.

Nesta situação o caminho percorrido pelo primeiro VANT é considerado obstáculos estáticos para o segundo VANT, tendo o algoritmo de os evitar para o segundo VANT e encontrar o caminho mais curto entre o ponto inicial e o ponto final para ambos os VANTs.

Na Tabela 5.2 apresentamos o custo e o tempo de execução do algoritmo para dois VANTs movendo-se sequencialmente para todos os mapas de simulação.

Tabela 5.2: Resultados para dois VANTs movendo-se sequencialmente

	Custo		
	VANT 1	VANT 2	Tempo de execução (s)
Mapa 1	19.31	24.97	9.68
Mapa 2	13.31	18.49	7.41
Mapa 3	14.49	14.49	6.63

A Figura5.7 mostra as trajetórias de ambos os VANTs movendo-se sequencialmente obtidas pelo algoritmo como sendo o caminho mais curto para cada um deles no primeiro mapa de simulação.

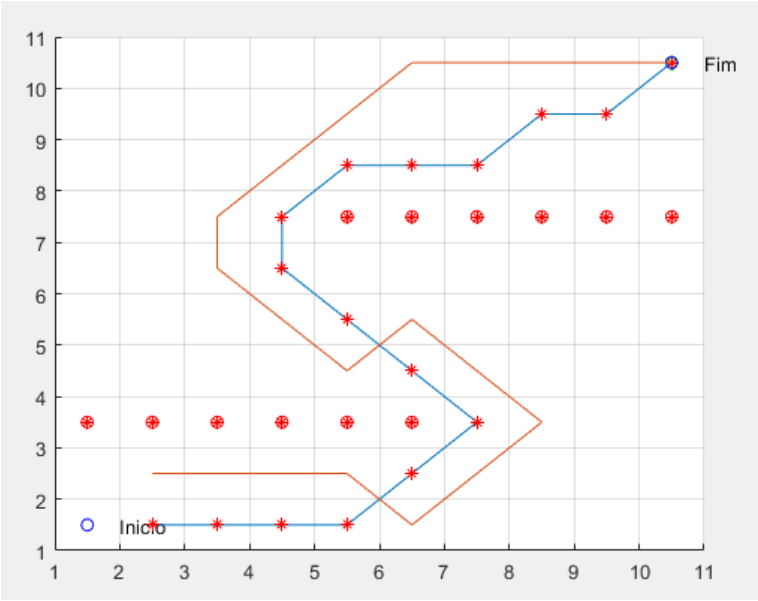

Figura 5.7: Caminho mais curto para dois VANTs movendo-se sequencialmente no primeiro mapa de simulação

No primeiro mapa de simulação verificou-se que o VANT 1 percorreu o mesmo caminho que anteriormente como esperado, o VANT 2 por consequência teve de percorrer um caminho mais longo uma vez que as posições onde passou o VANT 1 estavam bloqueadas, notando-se num incremento do custo.

Em termos de tempo de execução, como o algoritmo desta simulação foi aplicado

sequencialmente a cada VANT verificamos que o tempo de execução foi o dobro em relação à simulação feita com um VANT.

A Figura 5.8 mostra as trajetórias de ambos os VANTs movendo-se sequencialmente obtidas pelo algoritmo como sendo o caminho mais curto para cada um deles no segundo mapa de simulação.

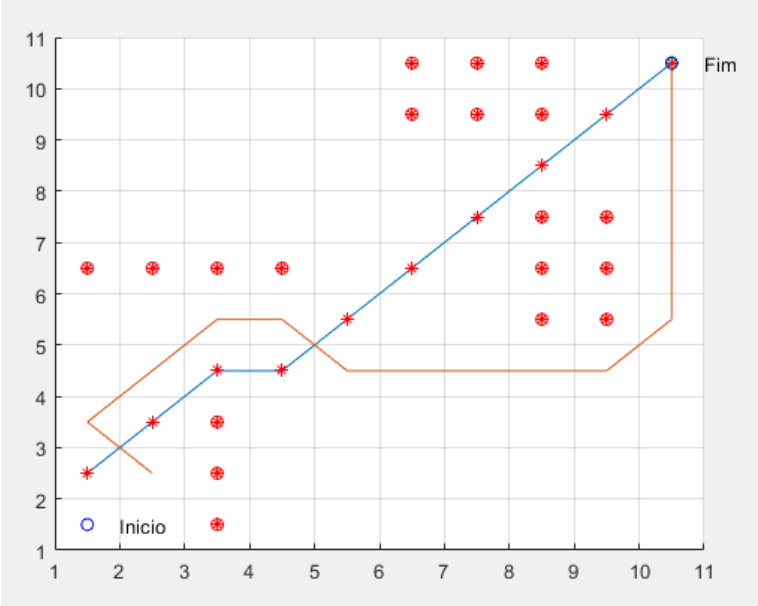

Figura 5.8: Caminho mais curto para dois VANTs movendo-se sequencialmente no segundo mapa de simulação

No segundo mapa de simulação verificou-se que o VANT 1 percorreu o mesmo caminho que anteriormente como esperado, já o VANT 2 teve de contornar outro obstáculo uma vez que o VANT 1 bloqueou essa passagem, tendo o custo para o VANT2 aumentado bastante.

Em termos de tempo de execução, como o algoritmo desta simulação foi aplicado sequencialmente a cada VANT verificamos que o tempo de execução foi o dobro em relação à simulação feita com um VANT.

A Figura 5.9 mostra as trajetórias de ambos os VANTs movendo-se sequencialmente obtidas pelo algoritmo como sendo o caminho mais curto para cada um deles no terceiro mapa de simulação.

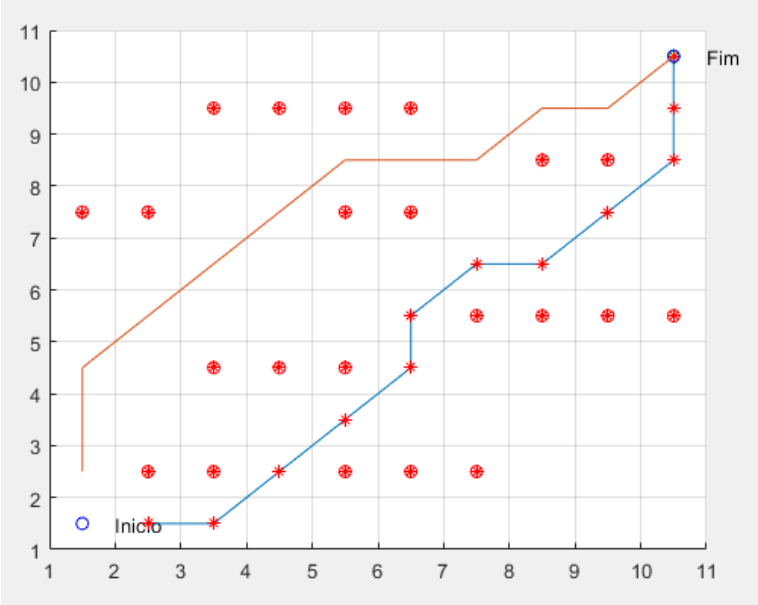

Figura 5.9: Caminho mais curto para dois VANTs movendo-se sequencialmente no terceiro mapa de simulação

No terceiro mapa de simulação verificou-se que o VANT 1 percorreu o mesmo caminho que anteriormente como esperado, já o VANT 2 encontrou um outro caminho alternativo que tem o mesmo custo do caminho do VANT 1, tido por isso obtido o melhor custo possível.

Em termos de tempo de execução, como o algoritmo desta simulação foi aplicado sequencialmente a cada VANT verificamos que o tempo de execução foi o dobro em relação à simulação feita com um VANT.

5.4 Resultados com dois VANTs a viajar simultaneamente

Nesta secção iremos analisar os dados obtidos pelo algoritmo modificado com dois VANTs a viajar simultaneamente. Nesta situação os VANTs são considerados obstáculos dinâmicos cuja posição muda em cada iteração. Assim o algoritmo teve de evitar os obstáculos

estáticos e evitar a posição de cada um dos VANTs em cada iteração.

Na Tabela 5.3 apresentamos o custo e o tempo de execução do algoritmo para dois VANTs movendo-se simultaneamente para todos os mapas de simulação.

Tabela 5.3: Resultados para dois VANTs movendo-se paralelamente

	Custo		
	VANT1	VANT 2	Tempo de execução (s)
Mapa 1	19.31	19.90	8.89
Mapa 2	13.31	13.31	5.53
Mapa 3	14.49	16.49	7.03

A Figura 5.10 mostra as trajetórias de ambos os VANTs movendo-se simultaneamente obtidas pelo algoritmo como sendo o caminho mais curto para cada um deles no primeiro mapa de simulação.

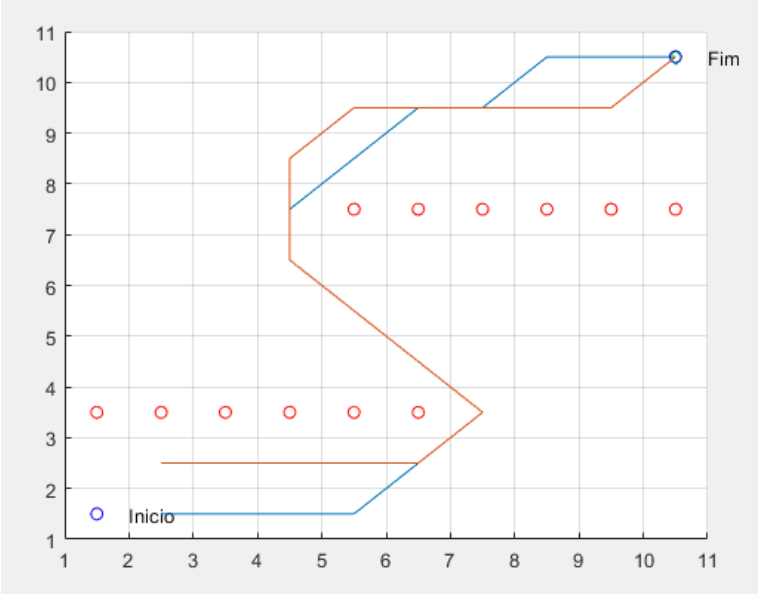

Figura 5.10: Caminho mais curto para dois VANT movendo-se simultaneamente no primeiro mapa de simulação

Neste mapa verificou-se uma alteração do caminho tomado pelo VANT 1, isto deveu-se a alteração do algoritmo o que levou o VANT 1 a evitar colidir com o VANT 2 naquele

instante. O custo manteve-se o mesmo para o VANT 1 e desceu bastante para o VANT2 o que é satisfatório e esperado.

Em relação ao tempo de execução, como foram menos pontos a calcular para o VANT 2 o tempo de execução foi menor comparado ao mesmo mapa de simulação mas com os dois VANTs a viajar sequencialmente.

A Figura 5.11 mostra as trajetórias de ambos os VANTs movendo-se simultaneamente obtidas pelo algoritmo como sendo o caminho mais curto para cada um deles no segundo mapa de simulação.

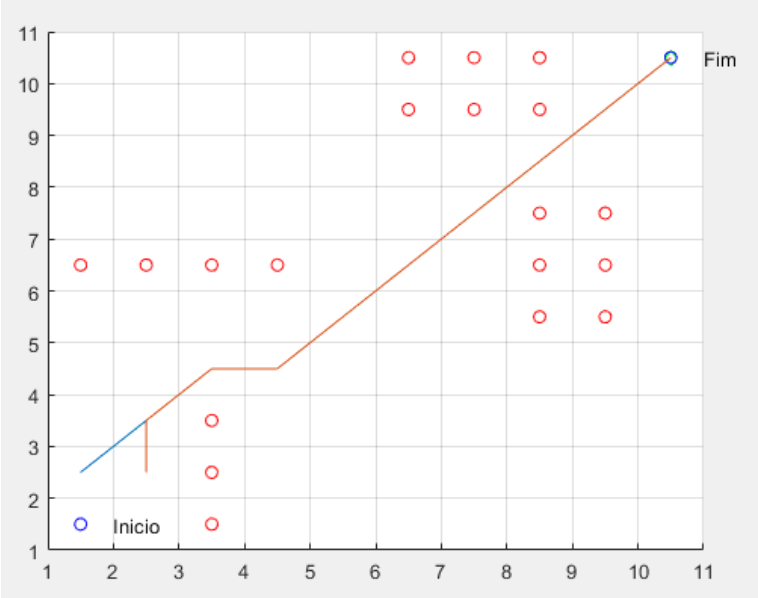

Figura 5.11: Caminho mais curto para dois VANT movendo-se simultaneamente no segundo mapa de simulação

Neste mapa verificou-se que o VANT 1 seguiu o mesmo caminho que anteriormente. Já o VANT 2, como teve de contornar o obstáculo anterior limitou-se a seguir o caminho que o VANT 1 fez descendo assim o seu custo significativamente.

Em relação ao tempo de execução volta-se a verificar a mesma situação da situação anterior, o VANT 2 teve um custo menor por isso o tempo de execução do algoritmo desceu quando compadrado ao mesmo mapa de simulação mas dois VANT a viajar sequencialmente

A Figura 5.12 mostra as trajetórias de ambos os VANTs movendo-se simultaneamente obtidas pelo algoritmo como sendo o caminho mais curto para cada um deles no terceiro mapa de simulação.

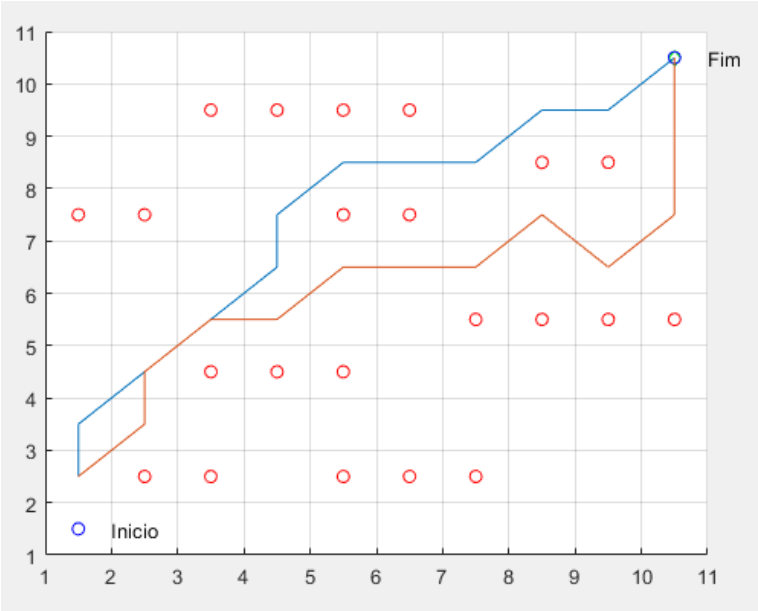

Figura 5.12: Caminho mais curto para dois VANTs movendo-se simultaneamente no terceiro mapa de simulação

Neste mapa verificou-se que tanto o VANT 1 como o VANT 2 mudaram a sua trajectória, tendo o VANT 1 mantendo o mesmo custo mas tendo o VANT 2 optado por um caminho mais caro e com uma trajectória não tão linear como se esperava. Em relação ao tempo de execução, como o custo para o VANT 2 aumentou, o tempo de execução do algoritmo também aumentou quando comparado com o mesmo mapa de simulação mas com dois VANTs a viajar sequencialmente.

Capítulo 6

Conclusões e Trabalhos futuros

Com a evolução da tecnologia os VANTs tornaram-se cada vez mais versáteis e mais úteis. Podem ser utilizados em missões de salvamento e resgate, mapeamento aéreo ou até mesmo como hobby. Apesar destes poderem ser controlados remotamente vê-se cada vez mais as vantagens da sua automação nas diversas áreas, como tal, o desenvolvimento de algoritmos para o planeamento das suas trajetórias ganha mais importância originando assim novas ideias e novos métodos para chegar aos resultados pretendidos.

Nesta dissertação o objetivo principal era otimizar o planeamento de trjetórias utilizando um algoritmo que permitisse a dois VANTs deslocarem-se de um ponto inicial a um ponto final no percurso mais curto. Tendo usado um algoritmo base modificado para ter valores para comparação procedeu-se à implementação deste em três mapas de simulação com diferente número de obstáculos.

Tendo obtido os percursos mais curtos e o tempo de execução do algoritmo procedeu-se à introdução de um segundo VANT no ambiente. Nesta situação o algoritmo procedeu ao cálculo do percurso mais curto para cada um dos VANTs sequencialmente, tendo o primeiro VANT viajado pelo mesmo caminho do primeiro algoritmo e o segundo VANT evitado o percurso do primeiro originando assim um custo mais alto para este e duplicando o tempo de execução do algoritmo.

Com o objetivo de melhorar os custos e o tempo de execução foi modificado o segundo algoritmo de modo a permitir aos dois VANTs deslocarem-se paralelamente, introduzindo

assim obstáculos dinâmicos que os VANTs teriam de evitar em cada iteração. Como esperado foi conseguido encontrar o caminho mais curto para ambos os VANTs na maioria das situações e reduzido o tempo de execução do algoritmo graças a isso.

Podemos concluir que o algoritmo obteve bons resultados em termos de custo e que foi capaz de atingir os seus objetivos em todas as simulações. Em termos de tempo de execução podemos concluir que quanto maior o número de VANTs e maior o custo, maior vai ser o tempo de execução, podendo esperar o dobro do custo por cada VANT presente na simulação.

Em suma, o objetivo da dissertação foi atingido e abriu a possibilidade de trabalhar com mais VANTs no futuro ou aplicar diretamente este algoritmo a outras simulações.

Bibliografia

[1] Amazon. *DJI Mavic Air Quadcopter with Remote Controller - Onyx Black*. URL: https://www.amazon.com/DJI-Mavic-Quadcopter-Remote-Controller/dp/B078WQ9SN3.

[2] Y. Chen et al. "Modified centralforceoptimization(MCFO) algorithm for 3D UAV path planning". Em: *Neurocomputing171(2016)878–888* (2015).

[3] Circuitstoday. *Types of Drones – Explore the Different Models of UAV's*. URL: https://www.circuitstoday.com/types-of-drones.

[4] Delair. *Boost your mapping productivity with a fixed-wing drone*. URL: https://delair.aero/delair-commercial-drones/professional-mapping-drone-delair-ux11/.

[5] FAA. *Aircraft Certification*. URL: https://www.faa.gov/licenses_certificates/aircraft_certification/.

[6] D. Floreano e R. Wood. "Science, technology and the future of small autonomous drones". Em: *Nature* (2015). DOI: 10.1038/nature14542.

[7] D. Floreano e R. Wood. "Science, technology and the future of small autonomous drones". Em: *Nature, 521, 460-466* 521 (2015), pp. 460–466. DOI: https://doi.org/10.1038/nature14542.

[8] U. Goel et al. "Three Dimensional Path Planning for UAVs in Dynamic Environment using Glow-worm Swarm Optimization". Em: *International Conference on Robotics*

and Smart Manufacturing (RoSMa2018) 133 (2018), pp. 230–239. DOI: https://doi.org/10.1016/j.procs.2018.07.028.

[9] P. Hart, N. Nilsson e B. Raphael. *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*. IEEE, 1968. DOI: 10.1109/TSSC.1968.300136.

[10] G. Jull. "An Overview of SHARP". Em: *Friends of the CRC* (1997).

[11] P. Kaplan. *Naval Aviation in the Second World War*. Pen Sword Aviation, 2013. ISBN: 1781593698.

[12] G. Klančar e I. Škrjanc. *Wheeled Mobile Robotics*. Butterworth-Heinemann, 2017. ISBN: 0128042044.

[13] L. Leça. "Implementação de Algoritmos dePlaneamento de Trajectórias de Robôs Móveis". Tese de mestrado. Faculdade de Engenharia da Universidade do Porto, 2015.

[14] D. Martin. *P. B. MacCready, 81, Inventor, Dies*. URL: https://www.nytimes.com/2007/08/31/us/31maccready.html.

[15] Mathworks. *MATLAB*. URL: https://www.mathworks.com/products/matlab.html.

[16] Mathworks. *Products and Services*. URL: https://www.mathworks.com/products.html?s_tid=gn_ps.

[17] T. Morita et al. "Decision Making Support of UAV Path Planning for Efficient Sensing in Radiation Dose Mapping". Em: *42nd IEEE International Conference on Computer Software Applications* 01 (2018), pp. 333–338. DOI: 10.1109/COMPSAC.2018.00053.

[18] N. Nilsson. *The quest for artificial intelligence a history of ideas and achievements*. Cambridge University Press, 2009. ISBN: 0521122937.

[19] International Civil Aviation Organization. *Unmanned Aircraft Systems (UAS)*. ICAO, 2011. ISBN: 978-92-9231-751-5.

[20] E. Pastor, J. Lopez e P. Royo. "A hardware/software acrhitecture for UAV oayload and mission control". Em: *IEEE Xplore* (2006), pp. 1–8. DOI: 10.1109/DASC.2006.313738.

[21] L. Piardi. "Application of a Mobile Robot to Spatial Mapping of Radioactive Substances in Indoor Environment". Tese de mestrado. Instituto Politécnico de Bragança, 2018.

[22] Prodrone. *PRODRONE Develops the "Speed Delivery"*. URL: https://www.prodrone.com/release-en/2874/.

[23] P. Soares. "Utilização de drones na floresta: muito além do óbvio". Em: *AgroIn* (2016). DOI: 10.13140/RG.2.2.25185.15207.

[24] International Organization for Standardization. "Robots and robotic devices— vocabulary". Em: *ISO* 2 (2012), p. 38.

[25] A. Taylor. *Jane's Book of Remotely Piloted Vehicles*. Collier Books, 1977. ISBN: 002080640X.

[26] J. Tisdale, Z. Kim e J. Hedrick. "Autonomous UAV path planning and estimation". Em: *IEEE Robotics Automation Magazine* 16.2 (2009), pp. 35–42. DOI: 10.1109/MRA.2009.932529.

[27] S. Tzafestas. *Introduction to Mobile Robot Control*. Elsevier, 2014. ISBN: 0124170498.

[28] UnmannedRC. *Dragon VTOL Plane For Drone Mapping and Topography*. URL: https://unmannedrc.com/products/dragon-vtol-plane-uav.

[29] M. R. Usman et al. "UAV Reconnaissance using Bio-Inspired Algorithms: Joint PSO and Penguin Search Optimization Algorithm (PeSOA) Attribute". Em: *2019 16th IEEE Anual Consumer Communications Netwrok Conference* (2019), pp. 1–6. DOI: 10.1109/CCNC.2019.8651831.

[30] W. Wagner. *Lightning Bugs, and other Reconnaissance Drone*. Aero Publishers, 1982. ISBN: 0816866546.

[31] X. Wu et al. “A hybrid algorithm of particle swarm optimization, metropolis criterion and RTS smoother for path planning of UAVs”. Em: *Applied Soft Computing Journal 73 (2018) 735–747* (2018).