FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Planeamento Global de Trajetórias com Desvio de Obstáculos Híbrido para Embarcações Autónomas

Renato Jorge Moreira Silva

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Andry Maykol Gomes Pinto

Co-orientador: Daniel Filipe Barros Campos

17 de Julho de 2019

Resumo

A maioria do comércio mundial utiliza o ambiente marítimo como meio de transporte de carga, correspondendo a cerca de 80% deste mercado. Assim, a promoção de uma abordagem ecológica, com menores custos de operação, mais segura e eficiente torna-se uma necessidade. Além disso, considerando que atualmente a maioria dos acidentes marítimos são causados por erro humano, a inclusão de navios autónomos permitirá minimizar a probabilidade de ocorrência deste tipo de erros, reduzir os custos desnecessários com tripulação excessiva e, ainda, aumentar a eficiência dos espaços de carga.

Assim, para cumprir tais objetivos é necessário desenvolver e melhorar os métodos de planeamento de rota existentes possibilitando a redução dos consumos de combustível e dos tempos de missão. Para tal, esta dissertação propõe um algoritmo de planeamento de rotas que tem em consideração a presença de obstáculos estáticos para *Autonomous Surface Vehicles* (ASVs). Este permite caracterizar e descrever qualquer tipo de missão e planear diferentes rotas de acordo com a definição dos pesos de uma função de custo, fazendo recurso a um planeador global de trajetórias e a uma metodologia de otimização multicritério.

Desta forma, uma vez gerada a missão, torna-se necessário realizá-la de forma segura e eficiente. Apesar do trabalho significativo a ser realizado e do constante aparecimento de novos algoritmos, aplicados na sua grande maioria ao ambiente terrestre, estes não são ainda suficientemente robustos do que toca à segurança, autonomia e fiabilidade devido à complexidade do ambiente marítimo. Por isto esta dissertação apresenta, também, um novo método de execução de rotas com desvio de obstáculos híbrido, sendo estes tanto estáticos como dinâmicos. Introduzindo o conjunto das velocidades alcançáveis com os conceitos de cone de colisão, o método desenvolvido permite estimar uma velocidade livre de colisões com base em restrições internas e externas, como a velocidade, direção e raio de curvatura do veículo, posição e dimensões dos obstáculos circundantes, bem como as suas velocidades. Uma estimativa não restrita para a velocidade e direção do ASV é calculada mapeando uma zona de conflito, determinada pela rota do veículo e pela distância a obstáculos, que é utilizada para evitar situações eminentemente perigosas.

Este método de planeamento e execução de rotas com desvio de obstáculos híbrido otimizado para restrições globais e locais (HORIZON) demonstra, através de um conjunto de testes experimentais em ambiente simulado, a capacidade para planear e executar missões contornando múltiplos obstáculos e atingindo um erro de 1.02% relativamente à execução da rota planeada contra um erro de 19.59% recorrendo a uma metodologia do estado da arte.

Abstract

Most of the world's trade uses the maritime environment as a means of cargo transport, accounting for about 80% of this market. Thus, promoting an ecological approach with lower operating costs, safer and more efficient becomes a necessity. In addition, considering that most shipping insurance losses are caused by human error, the inclusion of autonomous ships will minimize the likelihood of such errors, reduce unnecessary costs with excessive crew, and increase the efficiency of cargo spaces.

So, in order to accomplish these objectives, it is necessary to develop and improve existing route planning methods to reduce fuel consumption and mission times. For this purpose, this thesis proposes a route planning algorithm that takes into account the presence of static obstacles for textit Autonomous Surface Vehicles (ASVs). It allows to characterize and describe any type of mission and to plan different routes according to the definition of the weights of a cost function, using of a global planner and a multicriteria optimization methodology.

Thus, once the mission is generated, it is necessary to carry it out safely and efficiently. Despite the significant work being done and the constant emergence of new algorithms, mostly focused on the terrestrial environment, they are not yet robust enough regarding safety, autonomy and reliability due to the complexity of the maritime environment. For this reason, this thesis also presents a new method of executing routes with a hybrid obstacle avoidance approach for both static and dynamic objects. Introducing the reachability with the protective zone concepts, this method estimates a collision-free velocity based on inner and outer constraints such as, current velocity, direction, and turning radius of the vehicle, position and dimensions of the surrounding obstacles as well as their speed. A non-restrictive estimative for the speed and direction of the ASV is calculated by mapping a conflict zone, determined by the course of the vehicle and the distance to obstacles that is used to avoid imminent dangerous situations.

This method of planning and execution of routes with hybrid obstacle avoidance optimized for global and local constraints (HORIZON) demonstrates, through a set of simulated experimental tests, the ability to plan and execute missions by circumventing multiple obstacles and reaching a 1.02% error relative to the execution of the planned route against a 19.59% error using a state-of-the-art methodology.

Agradecimentos

Ao Prof. Andry Pinto pela confiança, disponibilidade e apoio prestado durante a realização da dissertação fulcrais para o sucesso da mesma. Agradeço-lhe ainda o tema do trabalho, que sempre me aliciou, o que fez, na maioria das vezes, conseguir ultrapassar dificuldades surgidas.

Ao co-orientador Daniel Campos por todas contribuições dadas para o desenvolvimento do projeto e por toda a paciência demonstrada.

Pretendo deixar um especial obrigado aos meus amigos Pedro Leite, Vitor Sousa, José Caires, Francisca Pereira e Alexandra Oliveira pelos momentos magníficos que passámos longo dos últimos cinco anos.

A todos os elementos da C.O.P.A. que proporcionaram momentos de bem-estar e alívio de forma a manter a sanidade mental ao longo da realização desta dissertação.

Aos meus pais, Jorge Silva e Fátima Silva, um enorme agradecimento, por me darem a possibilidade de seguir os meus sonhos e por todo o suporte e carinho. Por fim ao meu irmão, Pedro Silva, por me aturar em dias maus e pelas parvoíces que tanto me ajudaram a descontrair.

Renato Jorge Moreira Silva

"It always seems impossible until it's done."

Nelson Rolihlahla Mandela

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

ACO	*Ant Colony Optimization*
AG	Algoritmo Genético
ASV	*Autonomous Surface Vehicle*
AV	Acelerações Viáveis
CC	Cone de Colisão
Cespaço	Espaço de Configuração
Cobstáculos	Configuração dos Obstáculos
COLREGS	*International Regulations for Preventing Collisions at Sea*
CRAS	Centro de Robótica e Sistemas Autónomos
DV	Diagrama Voronoi
FPS	*Frames* Por Segundo
IMO	*International Maritime Organization*
INESC TEC	Instituto de Engenharia de Sistemas e Computadores,Tecnologia e Ciência
LIDAR	*Light Detection and Ranging*
ODE	*Open Dynamics Engine*
ROS	*Robot Operating System*
RVIZ	*ROS visualizer*
SA	*Simulated Annealing*
TEB	*Timed-Elastic-Band*
TSP	*Travel Salesman Problem*
VA	Velocidades Alcançáveis
VAL	Velocidades Alcançáveis Livres
VG	*Visibility Graph*
VHF	*Vector Field Histogram*
VO	*Velocity Obstacles*
VRX	*Virtual RobotX*

Capítulo 1

Introdução

Este documento constitui todo o trabalho executado para a unidade de Dissertação e traduz todo o processo realizado ao longo do semestre para a realização do projeto final de curso no âmbito do Mestrado Integrado em Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto. Ao longo deste primeiro capítulo serão introduzidos o contexto e a motivação por detrás do tópico da dissertação e os objetivos da mesma. De seguida é apresentada a estrutura do documento bem como as suas contribuições científicas.

1.1 Contexto e Motivação

Atualmente é notável a evolução dos veículos autónomos, devido à redução do custo de hardware e ao constante aparecimento de novos algoritmos cada vez mais eficientes e seguros. Este tipo de veículos estão a tornar-se, gradualmente, parte das nossas vidas, pois as vantagens proporcionadas pela sua utilização tais como a redução dos acidentes de trânsito provocados por fatores humanos, a possibilidade de utilização de veículos por deficientes, tanto motores como visuais, o aumento da produtividade uma vez que o condutor pode realizar outras tarefas durante o processo de navegação, são de longe superiores às suas desvantagens. É do conhecimento comum que os veículos autónomos realizam tarefas, geralmente realizadas por humanos, com o objetivo de libertar o operador envolvido de tarefas que possam apresentar potencial perigo e que não retornam valor económico acrescentado.

Com o crescimento da robótica móvel terrestre aplicada à industria e o aumento da produtividade a ser cada vez mais evidente, o próximo passo será conseguir dotar a robótica marinha do mesmo nível de autonomia, pois de acordo com a *International Maritime Organization* (IMO) [1], mais de 80% do comércio mundial é transportado pelo mar. Tendo em consideração a revisão anual realizada pela *Allianz* [1] a maior parte dos acidentes marítimos, entre 75% a 96%, são devido a erro humano. É então possível tornar este método de transporte mais eficiente, rentável e

[1] http://www.imo.org/en/About/Pages/Default.aspx

seguro reduzindo consideravelmente a sua probabilidade de ocorrência. A incorporação de autonomia na robótica marinha permitirá também reduzir os gastos em tripulação excessiva e aumentar o aproveitamento dos espaços do navio para carga.

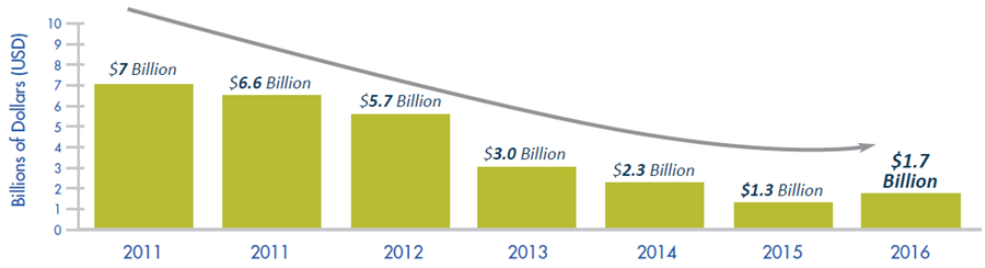


Figura 1.1: Custos monetários associados à pirataria na costa da Somália. Reportado por Jon Walker[2].

Outro fator, apesar de ser uma prática menos recorrente nos dias atuais, é a pirataria, que ainda existe e é responsável por consideráveis perdas monetárias. Como é possível observar na figura 1.1 os custos associados à pirataria na costa da Somália são bastante significativos, encontrando-se na ordem dos biliões de dólares e, apesar de estes terem diminuído ao longo dos anos, este problema ainda não se encontra completamente resolvido, pois verificou-se um aumento considerável de 2015 para 2016. Se considerarmos que as embarcações autónomas serão, num futuro próximo, não tripuladas e que, por isso, não existirá tripulação a bordo dos *Autonomous Surface Vehicles* (ASVs), a utilização de navios autónomos apresenta a vantagem de diminuir consideravelmente o risco de perdas humanas [1], já que nenhum refém poderá ser capturado e negociado por enormes quantias monetárias.

Por outro lado, para que uma embarcação se torne autónoma necessita de técnicas de perceção e navegação que lhe proporcione a capacidade de antecipar e evitar situações perigosas, de modo a levar a cabo todas as missões de forma eficiente e o mais seguro possível. Este tipo de técnicas deve ter como principal objetivo a identificação de situações onde possam estar em causa vidas humanas ou a possibilidade de causar sérios prejuízos ambientais. Portanto, estes sistemas deverão detetar tais situações perigosas atempadamente de forma a possibilitar aos sistemas de navegação a tomada de decisões preventivas de forma a tomar decisões quanto à alteração da rota e à velocidade do navio. Durante as últimas décadas, o planeamento de trajetórias foi uma área extensivamente estudada pela comunidade de robótica devido às diferentes áreas de aplicação onde este tipo de tecnologia poderia ser utilizada com sucesso.

> "*A reason for the success of motion planning research is that it is both narrow enough to make it easy to assess progress and deep enough that most results raise new issues motivating additional research.*"[2]

[2]https://emerj.com/ai-adoption-timelines/autonomous-ships-timeline/

Desta forma, na presente dissertação será realizado um estudo de um algoritmo capaz de planear rotas e executar trajetórias em tempo real num ambiente marítimo com configuração de obstáculos conhecida. Este deve também ser capaz de evitar colisões com os obstáculos (estáticos e dinâmicos), permitindo o replaneamento das trajetórias e apresentando um comportamento dinâmico e preditivo em função destes.

1.2 Objetivos

O principal objetivo desta dissertação é a idealização e implementação de um algoritmo que permite realizar o planeamento e execução de rotas num ambiente conhecido e dinâmico, evitando colisões e obter um comportamento preditivo de forma a realizar caminhos ótimos e seguros. Desta forma, os objetivos são:

- Desenvolver um algoritmo capaz de planear diferentes rotas de acordo com a definição de pesos de uma função de custo multicritério;
- Implementar um algoritmo de execução de rotas, de uma forma segura e eficiente, evitando colisões com obstáculos estáticos e dinâmicos;
- Obter uma representação do ambiente em que a embarcação se encontra, considerando a posição, velocidade e tamanho dos obstáculos;
- Gerar um método híbrido que combine um planeador global com uma representação local;
- Criar um método de seleção de uma velocidade dinâmica que assegura um caminho livre de colisões.

O sistema será primeiramente desenvolvido em ambiente de simulação, tendo como objetivo final a sua implementação e teste num protótipo de uma embarcação desenvolvido pelo Centro de Robótica e Sistemas Autónomos (CRAS) do Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência (INESC TEC).

1.3 Estrutura da dissertação

No capítulo 2 é descrito o estado da arte relativo às metodologias atualmente existentes para o planeamento de trajetórias. Serão então expostos os diversos sensores e de que forma são combinados de modo a obter uma representação do mundo em que o veículo se insere. De seguida é realizada uma introdução de conceitos sobre configuração de obstáculos e são enumerados e descritos os algoritmos de planeamento de trajetórias mais utilizados. Por fim, é efetuada uma análise de alguns métodos de pesquisa.

No capítulo 3 é exposta de forma sucinta a problemática a abordar por esta dissertação. Numa fase inicial é descrita a problemática da corrente dissertação e apresentada a metodologia a implementar para a concretização do projeto e de seguida é realizada uma breve introdução às ferramentas utilizadas, tais como o simulador e o modelo da embarcação utilizado.

No capítulo 4 é apresentado o método desenvolvido no decorrer desta dissertação. Está essencialmente dividido em três componentes. Em primeiro lugar é apresentado o método de planeamento de rota, posteriormente é abordado o algoritmo gestor de missão e por fim é abordado o algoritmo de execução de trajetórias.

No capítulo 5 são apresentados os resultados obtidos e a respetiva discussão para os métodos implementados para a resolução da problemática desta dissertação e comparados com algoritmos utilizados pela comunidade.

No capítulo 6 discute-se a satisfação dos objetivos propostos e o trabalho futuro sobre o qual se pode atuar futuramente.

1.4 Contribuições científicas

Este trabalho deu origem a dois artigos os quais foram apresentados em *IEEE International Conference on Autonomous Robot Systems and Competitions* (ICARSC) 2019:

- [3] Renato Silva, Pedro Leite, Daniel Campos, e Andry M Pinto. *Hybrid Approach to Estimate a Collision-Free Velocity for Autonomous Surface Vehicles*. página 6. *IEEE International Conference on Autonomous Robot Systems and Competitions*, 2019. Este artigo propõe uma heurística híbrida que combina um planeador global e local e possibilita estimar uma velocidade para o ASV que leva a um caminho livre de colisão;

- [4] Pedro Leite, Renato Silva, Aníbal Matos, e Andry Maykol Pinto. *An Hierarchical Architecture for Docking Autonomous Surface Vehicles*. página 6. *IEEE International Conference on Autonomous Robot Systems and Competitions*, 2019. Este artigo detalha uma versão inicial de uma arquitetura para atracagem de veículos autónomos de superfície e que foi expandida desde então.

Capítulo 2

Revisão Bibliográfica

O objetivo deste capítulo é apresentar uma visão geral dos algoritmos de planeamento de trajetórias, com recurso a desvio de obstáculos, utilizados em *Autonomous Surface Vehicles* e a vasta gama de aplicações em que estes veículos podem ser utilizados.

Assim, ao longo deste capítulo, serão primeiramente expostos os diversos sensores e de que forma são combinados de modo a obter uma representação do mundo em que o veículo se insere. De seguida é realizada uma introdução de conceitos sobre configuração de obstáculos e são enumerados e descritos os algoritmos de planeamento de trajetórias mais utilizados. Por fim, é efetuada uma análise de alguns métodos de pesquisa.

2.1 Introdução

A capacidade de um robô móvel em deslocar-se ao longo do ambiente de uma forma segura e fiável é um tópico de investigação que tem entusiasmado a comunidade científica nos últimos anos. Consequentemente, têm surgido diversas técnicas no estado da arte que tentam resolver alguns dos problemas que mais caracterizam o desafio que é a navegação autónoma de um veículo num ambiente real, nomeadamente, a existência de objetos estáticos de diferentes naturezas e dinâmicos com trajetórias inconstantes, limitação sensorial, condições meteorológicas desfavoráveis, etc. A evolução da tecnologia e o aumento da quantidade e qualidade dos sistemas de sensorização permitiram desenvolver sistemas de navegação sofisticados e que tentam mitigar alguns dos problemas supra indicados.

2.2 Sensores

Ao longo dos últimos anos tem existido um crescimento notável na quantidade de ASVs que são projetados e desenvolvidos, quer por empresas como por institutos de investigação devido às vantagens que acarretam a sua utilização no ambiente industrial. O primeiro passo para que uma embarcação autónoma consiga navegar de uma forma eficiente e segura é a correta perceção da cena, isto é, a aquisição de informação através de uma panóplia de sensores, deteção e a

identificação de todos os elementos cruciais (por exemplo outras embarcações, boias, portos) e a caracterização dos elementos dinâmicos e condições ambientais ou meteorológicas. Para isso, recorre-se à utilização de diversos sistemas de sensorização.

Atualmente, os tipos de sensores utilizados nestes veículos são bastante diversos tais como câmaras, radares, lasers, LIDARs (*Light Detection and Ranging*) e sonares, mas todos podem ser analisados em relação às seguintes caraterísticas: Gama medida, resolução, linearidade, largura de banda, precisão e repetibilidade, robustez a variações ambientais, custo, tamanho, peso e consumo energético.

Utilizando combinações destas é possível então complementar as suas vantagens e obter um sistema de sensorização substancialmente melhor. Por exemplo, J.Curcio, J. Leonard, e A. Patrikalakis (2005) [5] testaram uma embarcação equipada com radares e câmaras, enquanto que Hordur K. Heidarsson e Gaurav S. Sukhatme (2011) [6] utilizaram sonares e C. Jaques et al. (2013) [7] apresentaram um sistema capaz de mapear abaixo da superfície da água com recurso a sonares e acima da superfície usando LIDAR, câmara e radares.

Segundo Hordur K. Heidarsson e Gaurav S. Sukhatme (2011) [6] as tecnologias mais utilizadas em ASVs são as camaras, lasers e LIDARs.

2.3 Algoritmos de planeamento de trajetórias

Considerando que uma missão é composta por um conjunto indeterminado de pontos que se deseja visitar com o veículo, existem dois conceitos importantes a reter em primeiro lugar no que a esta problemática:

- Planeamento de rotas (*routing*);
- Planeamento de trajetórias (*trajectory planning*).

O planeamento de rotas tem como função descrever em termos sequenciais os pontos de destino desejados de forma a cumprir determinada missão. O planeamento de trajetórias tem como função descrever em termos geométricos e temporais o caminho a realizar para deslocar o veículo entre dois pontos.

De forma a se poder escolher qual o método ou algoritmo a utilizar é necessário ter em atenção alguns fatores, tais como, [8, 9]:

- As variáveis que definem as considerações a serem otimizadas, por exemplo, minimizar o comprimento da trajetória, energia consumida, ou outra variável;
- Requisitos computacionais ao nível do processamento em tempo real e a quantidade de memória que é necessária para executar os algoritmos;
- Se o resultado do algoritmo é determinístico ou probabilístico;
- A dinâmica do cenário envolvente, isto é, se os objetos são apenas estáticos ou também existem objetos dinâmicos;

- O tipo de cinemática do robô, isto é, um robô holonómico ou não holonómico;
- Se a reconstrução do cenário que é usado para estimar as trajetórias e/ou caminhos pode ser atualizada no decorrer da missão, com base na informação sensorial.

Como referido anteriormente, é necessário, em primeiro lugar, criar um mapa do ambiente onde o robô ou veículo se encontra inserido de forma a dar início a qualquer planeamento de trajetórias.

Ao conjunto de parâmetros que especificam a posição de um robô no sistema denomina-se configuração de um robô. O $C_{espaço}$, ou espaço de configuração, é o conjunto de todas as configurações possíveis do sistema. A transformação desde o espaço físico para o $C_{espaço}$ origina a possibilidade de poder tratar o robô como um ponto aumentando-se os obstáculos com o raio do robô [8].

Y. Hwang e N. Ahuja (1992) [10] consideraram que existem diversos métodos para calcular a configuração dos obstáculos, $C_{obstáculos}$. De seguida serão apresentados dois métodos, sendo o primeiro o mais simples e o seguinte o mais usual pela comunidade:

Método do ponto de avaliação

Este método é o mais simples e ineficaz e consiste em verificar para cada configuração do robô, se este interceta algum obstáculo. Em caso afirmativo, ou seja, caso haja interseção considera-se essa configuração como pertencente a $C_{obstáculos}$ [8, 9].

Método das diferenças de *Minkowski*

Considerando o robô um corpo rígido e sem rotação, e dois conjuntos de pontos A e B, sendo A o conjunto de pontos que representam o obstáculo e B o conjunto de pontos que representam o robô, este método calcula o $C_{obstáculos}$ através da diferença de *Minkowski*,

$$M_{diff}(A,B) = \{a - b \mid a \in A, b \in B\} \tag{2.1}$$

De seguida são enumerados vários algoritmos que foram desenvolvidos e idealizados pela comunidade robótica. Esses algoritmos podem ser divididos nas seguintes categorias, não sendo estas mutuamente exclusivas, de forma a ser possível realizar combinações entre elas.

2.3.1 Algoritmo Bug

Este tipo de algoritmo caracteriza-se por possuir apenas conhecimento local, fazendo recurso de diversos sensores, e um objetivo global. Desta forma os algoritmos Bug limitam-se a deslocar o veículo, desde o ponto inicial até ao ponto destino, contornando qualquer obstáculo que afete essa trajetória utilizando a menor quantidade de dados globais possíveis.

Os algoritmos Bug sofreram algumas modificações ao longo dos últimos anos surgindo assim os algoritmos Bug 1, Bug 2, Tangent Bug, DistBug e Bug 2+.

Bug 1

Este algoritmo apresenta a vantagem de ser um algoritmo completo, apesar de não garantir uma trajetória ótima entre dois pontos e caracteriza-se nos seguintes passos [8, 9]:

1. Ir em direção ao ponto destino;
2. Se encontrar um obstáculo que interrompa essa trajetória, contorna-o e guarda o ponto mais próximo do destino;
3. Verificar se o ponto final é atingível e caso seja verdade dirigir-se para o ponto guardado anteriormente;
4. Voltar ao 1º ponto.

O facto de ser necessário contornar completamente todo o obstáculo torna este algoritmo pouco eficiente.

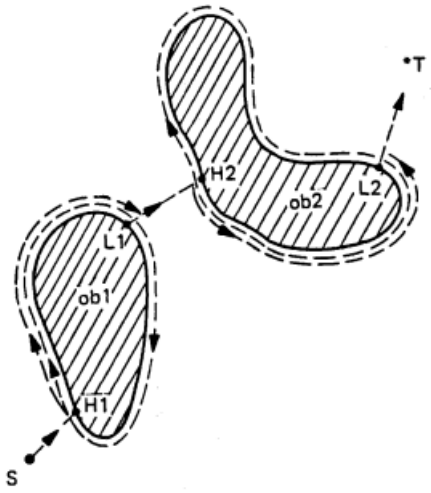


Figura 2.1: Exemplo de um caminho gerados pelo algoritmo Bug 1. Imagem retirada de [11].

Bug 2

O algoritmo Bug 2 é um melhoramento do método apresentado anteriormente e, tal como esse, é completo e não ótimo. A grande vantagem deste é não necessitar de contornar todo o obstáculo. Pode-se descrever este algoritmo através dos seguintes passos [8, 9]:

1. Definir um segmento de reta imaginário entre o ponto inicial e ponto final;
2. Ir em direção ao destino deslocando-se sobre o segmento de reta;
3. Se um obstáculo for encontrado, contorná-lo até encontrar o ponto destino ou um ponto pertencente ao segmento de reta;

(a) Se o robô voltar ao ponto que se encontrava antes de iniciar o contorno conclui-se que é impossível, tornando o alvo inacessível e terminando assim o processo.

4. Continuar em direção ao destino.

Bug 2+

Tal como é descrito por P. Costa (2011) [8], este algoritmo é uma evolução do algoritmo Bug 2 e foi apresentado por Antich (2009) [12].

A evolução consiste numa pequena alteração no algoritmo que passa por só se deslocar em direção ao destino se não tiver passado por pontos do segmento de reta mais próximos do mesmo.

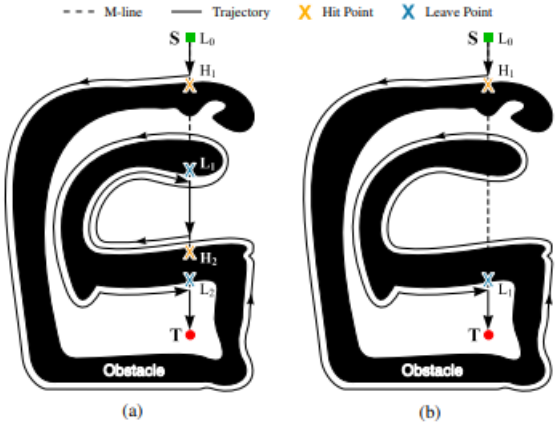


Figura 2.2: Comparação dos caminho gerados pelo algoritmo a) Bug 2 e pelo b) Bug 2+. Imagem retirada de [13].

Tangent Bug e DistBug

Estes dois algoritmos têm em comum o facto de utilizarem sensores de distância mas, enquanto que o Tangent Bug utiliza esses sensores para elaborar um grafo com os vértices do obstáculo e assim decidir para que vértice se dirigir, o DistBug dá uso a essa informação dos sensores para determinar quando abandonar o contorno e ir diretamente para o ponto de destino.

Ambos os algoritmos foram implementados em robôs reais. O Tangent Bug foi implementado por Kamon (1998) [14] e o DistBug por Rivlin (1997) [15].

2.3.2 Roadmap

Este tipo de algoritmo, apresentado por J. Latombe (1999) [16], permite representar nós e ligações entre nós através de curvas unidimensionais, que podem ter um significado físico. Desta forma é então gerado um grafo com todos os nós e as suas ligações, ficando a dificuldade reduzida a um problema de pesquisa em grafo.

Algumas das técnicas utilizadas para a construção do roadmap são *visibility graph* (VG) e diagrama de Voronoi (DV). Ambas as técnicas são completas, porém a utilização de diagramas de Voronoi gera caminhos mais seguros pois procura maximizar a distância aos obstáculos. Esta característica pode levar à perda de informação em sensores de curto alcance [9].

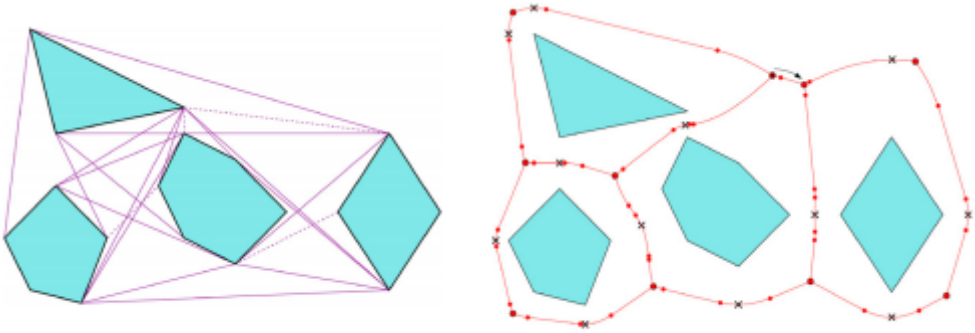


Figura 2.3: Comparação dos caminhos gerados pela utilização de *visibility graph* (à esquerda) e de diagrama Voronoi (à direita). Imagem retirada de [17].

2.3.3 Decomposição em células

Este método tem como objetivo representar o espaço em células, atribuindo a cada uma um valor representativo da sua condição livre ou obstruída, e gerar um grafo em que cada célula é um nó. O planeamento de trajetórias fica então, tal como o roadmap, reduzido a um problema de pesquisa em grafo.

A decomposição do espaço em células pode ser realizada gerando células exatas, as quais representam o meio em conformidade com o mundo real, isto é, as células representam espaços livres ou espaços ocupados, ou em células aproximadas. Para a decomposição em células exatas é comum dividir o espaço em polígonos convexos ou em trapézios devido à simplicidade que trás ao problema este tipo de separação, enquanto que para a decomposição em células aproximadas as formas mais utilizadas são célula fixa ou Quadtree, tal como é explicado por D. Campos (2014) [9].

2.3.4 Campos de potencial

Segundo D. Parhi e A. Jha (2012) [18] este método foi inicialmente sugerido por Khatib em 1985 e desde então tem sido bastante estudado devido à sua simplicidade matemática.

Em resumo, este algoritmo considera o robô como uma partícula de carga positiva onde são aplicadas forças imaginárias de repulsão e atração, causadas pelos obstáculos e pelo ponto destino, respetivamente.

Y. Koren et al. (1991) [19] apresentam os quatro principais problemas na sua implementação:

- Situações em que o robô fica preso num mínimo local;
- Não permite a passagem entre obstáculos próximos;
- Oscilações na presença de obstáculos;
- Oscilações em passagens estreitas.

Mais tarde, estes mesmos investigadores, desenvolveram um novo método chamado *vector field histogram* (VHF) [20] o qual apresenta comportamento suave e não oscilatório.

2.3.5 Timed-Elastic-Band (TEB)

Este algoritmo é um melhoramento do *Elastic Band*. Segundo S. Quinlan e O. Khatib (1993) [21], *elastic band* é um caminho deformável livre de colisões. Neste método é calculado primeiramente o caminho e de seguida a sua forma inicial é então sujeita a forças artificiais com o objetivo de o deformar, encurtando-o e suavizando-o.

Para além de ser online tem a grande vantagem de ser ótimo e ter em consideração as restrições cinemáticas e dinâmicas do veículo. Este pode ser dividido em dois passos. Inicialmente, é calculada uma trajetória inicial por um planeador global, que posteriormente é otimizada em tempo real de forma a minimizar o tempo de execução da trajetória, tendo em consideração as distâncias aos obstáculos e satisfazendo as velocidades e acelerações máximas proporcionadas pelo veículo. A trajetória ótima é obtida de forma bastante eficiente resolvendo um problema normalizado multiobjetivo. Foram efetuadas bastantes simulações e experiências em robôs reais e verificou-se que este algoritmo é bastante robusto [22].

Além desta melhoria, foi ainda implementada uma extensão que permite ao algoritmo transitar através dos obstáculos de forma a não ficar preso em trajetórias localmente ótimas. Esta extensão consiste na otimização em paralelo de um conjunto de trajetórias de diferentes topologias admissíveis. Estas topologias são obtidas recorrendo ao método descrito por C. Rösman, F. Hoffman, T. Bertram (2015, 2017) [23, 24].

Um exemplo de uma trajetória obtida recorrendo a este algoritmo pode ser observada na figura 2.4.

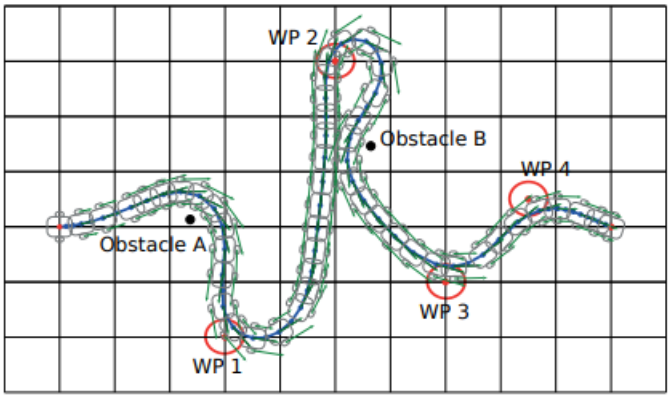


Figura 2.4: Trajetória obtida recorrendo ao algoritmo Timed-Elastic-Band. Imagem retirada de [25].

2.3.6 Velocity Obstacles (VO)

Ao longo dos últimos anos um dos algoritmos que tem evoluído é o *Velocity Obstacles*, tendo já sido simulado e testado em ASVs por D. Kufoalor, E. Brekke e T.Johansen (2018) [26] tendo em consideração as normas impostas pelo *International Regulations for Preventing Collisions at Sea* (COLREGS).

Neste método, primeiramente são definidos os obstáculos e de seguida são calculados os cones de colisão tendo em consideração a velocidade relativa dos mesmos. De seguida, de acordo com as restrições do ASV é definido o conjunto de todas as velocidades admissíveis, num espaço temporal definido previamente, e posteriormente é determinada uma velocidade, segundo uma ou várias heurísticas, que não pertença ao conjunto formado por todos os cones. Desta forma é assegurado um caminho livre de colisões.

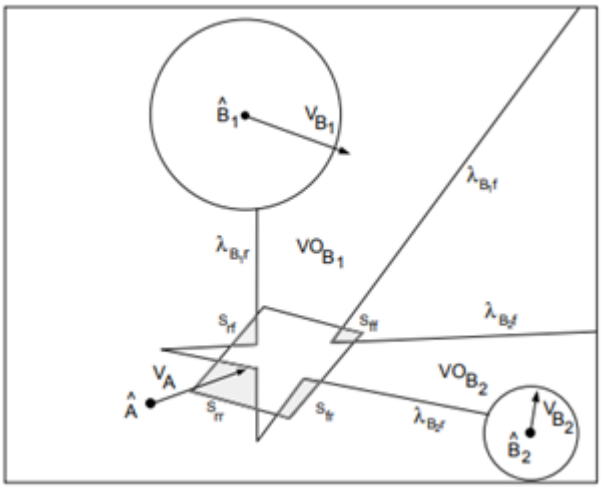


Figura 2.5: Imagem retirada de [27] representativa dos cones de colisão e do conjunto de velocidades acessíveis.

Algumas das heurísticas utilizadas por P. Fiorini e Z. Shiller (1993) [27] são:

- Escolher a maior velocidade, fora da zona de colisão, ao longo da linha para o destino;
- Escolher a maior velocidade, fora da zona de colisão, com um determinado ângulo máximo em relação à linha para o destino;
- Escolher a velocidade que evita os obstáculos de acordo com o seu risco associado.

É possível também combinar diferentes heurísticas para atingir determinados objetivos.

Foram publicadas inúmeras variações deste algoritmo tanto com utilização direta em ASVs, tais como *Dynamic Reciprocal Velocity Obstacles* [26] e *Adaptive Sampling Based COLREGs-Compliant Obstacle Avoidance for Autonomous Surface Vehicles* [28], como também noutros tipos de veículos como *Reciprocal Velocity Obstacles* [29] e *Reciprocal n-body Collision Avoidance* [30].

2.3.7 Outras abordagens

Muitos outros métodos têm sido originados e desenvolvidos ao longo dos últimos anos devido não só ao crescimento da complexidade dos problemas e à necessidade de obtenção de uma solução, mas também devido à curiosidade de muitos investigadores nesta área da robótica. Alguns desses métodos são:

Lógica *Fuzzy*

Em 2003, Saade e Khatib [31] desenvolveram uma abordagem *fuzzy* baseada em dados para fornecer uma estrutura para resolver o problema de movimento dinâmico de um robô tendo em consideração algumas restrições. Segundo a revisão e análise realizada por D. Parhi e A. Jha (2012) [18], a principal vantagem deste método em relação aos algoritmos genéticos *fuzzy* mais recentes, é que o robô consegue navegar com sucesso na presença de obstáculos dinâmicos e independentemente da sua quantidade. Este método apresentou também uma redução no tempo de viagem.

Mais recentemente, Hassanzadeh, Ghadiri e Dalayimilan (2008) [32] utilizaram um controlador *fuzzy* para desvio de obstáculos num robô móvel. Os resultados alcançados foram comparados com os resultados obtidos utilizando o método campos de potencial e verificou-se uma melhoria significativa comparativamente a este. Este método foi implementado no robô Khepera II e os resultados apresentados demonstraram a eficácia deste algoritmo e um tempo de viagem mais curto.

Redes neuronais

Meng e Nak (1993) [33] desenvolveram um sistema chamado NEURO-NAV dotado de um novo tipo de arquitetura de controlo para sistemas orientados por visão utilizando redes neuronais. Este tipo de controlo tem como objetivo fazer com que o robô seja mais "humano". Mais recentemente, esta problemática foi abordada por Wahab o qual utilizou duas redes neuronais distintas para a solucionar. A primeira tem como objetivo a determinação do espaço livre de forma a evitar os obstáculos e a segunda a navegação do robô para o destino.

Algoritmo genético (AG)

Este tipo de método foi utilizado pela primeira vez em planeamento de trajetórias, segundo P. Costa (2011) [8], por Arimoto em 1984 [34] e mais tarde foi apresentado por X. Yang e Jianping Tu (2003) [35] uma abordagem em algoritmos genéticos com um cromossoma de tamanho variável. Uma outra variante deste método foi desenvolvido por Sedighi et al. (2004) [36] e testado num robô real.

***Ant Colony Optimization* (ACO)**

Este método é um algoritmo de pesquisa, inspirado na forma como as formigas procuram o seu alimento em direções aleatórias e encontram a distância mínima entre a fonte de alimento e o

seu ninho através de comunicação indireta entre si, regulando a concentração de feromonas na sua proximidade.

Este método foi aplicado com sucesso por Cong e Ponnambalam em 2009 [37], para resolver o problema de planeamento de trajetórias de um robô móvel e em 2006, Zheng, Huan e Aaron [38], apresentaram uma nova abordagem para planeamento global de trajetórias em tempo real. Este novo método consiste nos seguintes passos [18]:

- Primeiramente é calculado o $C_{espaço}$ livre do robô móvel e gerado um grafo recorrendo à técnica MAKLINK [39];
- De seguida é aplicado o algoritmo Dijkstra, apresentado na secção seguinte, para encontrar um caminho livre de colisões;
- Por último é utilizado o algoritmo ACO para otimizar e gerar a trajetória global ótima.

2.4 Algoritmos de pesquisa em grafos

Nesta secção serão expostos alguns algoritmos de pesquisa utilizados em grafos. Tal como foi referido anteriormente, alguns dos métodos têm como objetivo gerar um grafo. Desta forma, para que seja encontrada a solução é necessário realizar uma pesquisa no mesmo. Devido às restrições implícitas no problema, tal como a restrição temporal, é necessário proceder a uma escolha cuidada do algoritmo.

Estes algoritmos podem ser separados em duas categorias. Os algoritmos sem informação, os quais realizam uma busca exaustiva e só concluem quando encontram a solução, e os algoritmos informados que têm em consideração informações sobre o sistema, utilizando funções de custo ou heurísticas.

Quanto aos métodos de pesquisa não informados, estes variam consoante a ordem de pesquisa no grafo. Dentro destes existem [8]:

- A pesquisa por profundidade – pesquisa os nós mais longe antes de recuar;
- A pesquisa por largura – pesquisa todas as ligações de um nó antes de passar ao próximo;
- A pesquisa por profundidade limitada – pesquisa com a mesma ordem que o primeiro, mas é-lhe aplicado um limite à profundidade máxima;
- A pesquisa por aprofundamento iterativo – pesquisa em profundidade, em que o limite de profundidade vai sendo aumentado, até encontrar o destino.

Dentro da comunidade robótica, no que toca à área do planeamento de trajetórias, a categoria normalmente utilizada é a categoria dos algoritmos informados devido às restrições referidas. Estes algoritmos permitem uma diminuição da quantidade de memória necessária e normalmente apresentam um tempo de processamento inferior.

Os algoritmos mais populares e reconhecidos são:

- Pesquisa Gulosa;
- Algoritmo Dijkstra;
- Algoritmo A*.

De seguida são apresentados os três algoritmos e as suas avaliações quanto à complexidade temporal e espacial, otimalidade e se é completo, com o intuito de se concluir a sua eficácia e eficiência.

Para medir a complexidade temporal e espacial define-se os seguintes termos:

- b – Número máximo de ligações para um nó;
- d – Profundidade da solução ótima;
- m – Máxima profundidade do grafo.

2.4.1 Pesquisa Gulosa

Este algoritmo é um método de pesquisa que tem como base a escolha do ótimo local em cada instante, de forma a encontrar a solução ótima global. Para isso, dá uso à informação de quanto falta para alcançar o destino e através dessa informação escolhe sempre, como próximo nó a explorar, o nó que apresenta a menor distância ao ponto final. Este algoritmo é completo, ou seja, origina sempre resultado, no caso em que o número máximo de ligações para um nó ser finito e a sua complexidade temporal e espacial é $O(b_m)$. É também não ótimo e o facto de ser necessário guardar todos os nós em memória, faz com que a sua eficiência seja relativamente baixa quando comparado com outros.

2.4.2 Algoritmo Dijkstra

Este algoritmo tem como base, de forma semelhante ao anterior, a escolha do ótimo local. Mas, contrariamente à Pesquisa Gulosa, este algoritmo tem em consideração as diferentes distâncias entre nós, ao invés da distância ao destino. Assim, neste método de pesquisa, para cada iteração, é escolhido como próximo nó, aquele que apresenta uma distância mais curta, ao partir do nó atual. É um algoritmo completo e ótimo, com complexidade espacial e temporal $O(b_m)$ e $O(b_d)$, respetivamente.

2.4.3 A*

Da mesma forma que o algoritmo Dijkstra utiliza informação acerca do nó que se encontra mais perto, o A* dá uso a essa mesma informação adicionando-lhe quanto falta para alcançar o ponto de destino.

É utilizada então uma heurística para determinar a distância restante e esses dois dados são combinados através de uma equação. A escolha da heurística é de extrema importância pois

as complexidades espacial e temporal dependem em grande parte desta. Assim, tanto se pode verificar uma complexidade exponencial $O(2^n)$, para a pior heurística possível ($h(n) = 0$), como uma complexidade linear $O(n)$, no caso em que $h(n) = h_m(n)$ sendo $h_m(n)$ o menor custo de n até ao destino [8].

Uma das grandes vantagens apresentadas por este método é a permitir realizar alterações ao $C_{espaço}$, tais como a utilização dos pontos de choque, entre o robô e os obstáculos móveis, como obstáculos ao invés de colocar a sua verdadeira posição, a criação de zonas de folga em torno destes pontos para evitar ou incentivar choques, a alteração do tamanho dos obstáculos em função da distância ao robô, entre outros.

Este algoritmo é ótimo e completo e é também o mais eficiente dos algoritmos até agora apresentados.

Devido ao facto de ser necessário refazer o planeamento desde o início quando ocorre uma alteração no grafo, este torna-se assim ineficiente ao ser aplicado em tempo real. Por esse motivo, muitos outros algoritmos foram desenvolvidos tendo como base este, sendo apenas necessário o reajuste da trajetória. Alguns desses algoritmos são o IDA*, ARA*, AD*, MA*, SMA*, D*, Focused D*, D* lite, E* e outros.

Alguns destes algoritmos apresentam vantagens relativamente ao tempo de execução, tais como o IDA*e o ARA*, enquanto outros na quantidade de memória, como é o caso do MA* e o SMA* mas nenhum apresenta uma solução melhor comparativamente ao A*.

O algoritmo IDA*, apresentado por Pearl (1984) [40], consiste em definir um limite k para a função representativa do custo de cada nó, executar o A*, não expandindo os nós com valor superior ao limite definido e, se não encontrar o destino, incrementar o valor de k e voltar a executar o algoritmo base.

O ARA* consiste em determinar uma solução sub-ótima e melhorá-la enquanto houver tempo disponível. Este método foi desenvolvido por Likhachev (2003) [41].

O método MA* adaptado por Chakrabarti (1989) [42], consiste em retirar da lista os nós com custo mais elevado, e por isso, menos promissores quando a memória se encontra esgotada. Rong Zhou e Eric A. Hansen (2002) [43], utilizando a base apresentada por Chakrabarti e aplicando-lhe uma estrutura de dados mais eficiente, criaram então o algoritmo SMA*.

2.5 Análise Crítica

Esta secção tem como objetivo apresentar as conclusões finais retiradas da pesquisa realizada e apresentada no estado da arte. Como referido são inúmeros os algoritmos de planeamento de trajetórias desenvolvidos mas é possível facilmente observar que, na sua grande maioria, estes não são ainda suficientemente robustos do que toca à segurança, autonomia e fiabilidade para serem aplicados em ambiente marítimo.

Apesar disso, algoritmos de trajetórias tais como, *Timed-Elastic-Band* e *Velocity Obstacles*, revelam-se bastante promissores pois são algoritmos que permitem ter em consideração a dinâmica

dos ASVs, instantes temporais e restrições geométricas da trajetória. Pelo estudo realizado, o método TEB apresenta-se como um dos algoritmos com mais destaque no que toca ao planeamento de trajetórias, embora seja um algoritmo de execução bastante instável em ambiente marítimo. Em relação ao conceito *Velocity Obstacles* a sua flexibilidade e adaptabilidade representada por uma heurística torna este método bastante propício a desenvolver aplicações nesta área, tendo já sido simulado e testado em ASVs. Dos restantes algoritmos apresentados, abordagens com campos de potencial poderiam ser também bastante promissores mas é ainda necessário a resolução de problemas de mínimos locais. Por outro lado, algoritmos como Bug e seus derivados poderiam também revelar-se favoráveis à implementação neste contexto mas estes são ainda bastante rudimentares e não promovem uma abordagem com baixos custos de operação, mais segura ou eficiente.

Capítulo 3

Formulação do Problema

Neste capítulo é exposta de forma sucinta a problemática a abordar por esta dissertação e apresentada a metodologia a implementar para a concretização do projeto. De seguida é realizada uma breve introdução às ferramentas utilizadas, tais como o simulador e o modelo da embarcação utilizado.

Com a evolução dos veículos autónomos terrestres e as vantagens da sua utilização a serem cada vez mais evidentes surge atualmente a necessidade de obter o mesmo grau de autonomia em veículos de superfície, sem esquecer as questões que acarretam a sua utilização, tais como a segurança e a eficiência. De modo a tornar o método de transporte marítimo mais eficiente, rentável e seguro é necessário desenvolver e implementar um algoritmo de planeamento de trajetórias e desvio de obstáculos em tempo real para ser futuramente utilizado pelas embarcações.

3.1 A problemática

A problemática da corrente dissertação consiste na implementação de um algoritmo de planeamento de trajetórias que é capaz de, tendo em consideração todos os obstáculos estáticos conhecidos no momento, resolver um problema do tipo *Travel Salesman Problem* (TSP) e assim determinar a rota que apresenta um menor custo para percorrer uma série de pontos de interesse. Para que posteriormente essa rota possa ser executada é também necessário a implementação de um algoritmo de execução de trajetórias. Este algoritmo deve considerar todo o tipo de obstáculos, estáticos e dinâmicos, e assim navegar a embarcação de um modo seguro para os pontos de destino desejados. Visto que o comportamento dos obstáculos dinâmicos são desconhecidos este algoritmo deve apresentar um comportamento dinâmico e preditivo, permitindo que o veículo se desloque a uma velocidade superior quando este se encontra longe de obstáculos e uma velocidade inferior em situações de potencial perigo.

Para este problema serão considerados obstáculos fixos, como por exemplo boias, imagem central da figura 3.2, paredões ou plataformas petrolíferas, e obstáculos dinâmicos, tais como, barcos a remo, navios e até mesmo cargueiros. Estes obstáculos serão obtidos através de um sistema

de *tracking* presente no veículo que fornecerá a informação de cada obstáculo relativamente à sua posição e velocidade.

3.2 Metodologia

Tendo em consideração a revisão bibliográfica apresentada no capítulo 2, foram escolhidos os algoritmos que servirão de base para a solução a desenvolver. De destacar que ao longo desta dissertação o termo rota referir-se-á à sequência de pontos a visitar de forma a concluir uma missão e o termo trajetória consistirá no caminho entre dois pontos de destino.

Por forma a solucionar o problema do tipo *Travel Salesman Problem* optou-se pelo algoritmo *simulated annealing* (SA) que consiste numa meta-heurística para otimização de uma determinada função de custo, que engloba a distância total da rota e variação total angular desta. Esta técnica de busca local probabilística permite encontrar a solução mínima global de uma função que poderá conter inúmeros mínimos locais [44]. De forma a possibilitar o cálculo de um valor de solução em cada iteração da meta-heurística foi utilizado o algoritmo *Timed-Elastic-Band* para se obter todas as trajetórias entre os diversos pontos definidos previamente pelo utilizador.

Depois de obtida a solução do TSP, que consiste na sequência de pontos a visitar, representada pela letra P na imagem 3.1, de modo a executar cada trajetória independentemente é novamente utilizado o algoritmo TEB juntamente com uma representação espacial baseada no conceito *Velocity Obstacles*.

Assim, é possível concluir que a proposta de solução se baseia num tipo de controlo híbrido. A primeira camada, apresentada a laranja, representa um sistema discreto e é responsável por atribuir à segunda camada qual o próximo ponto a ser visitado p_i, enquanto que esta última representa um sistema contínuo responsável pela execução da trajetória em questão e retorna à camada anterior a posição atual da embarcação p_r.

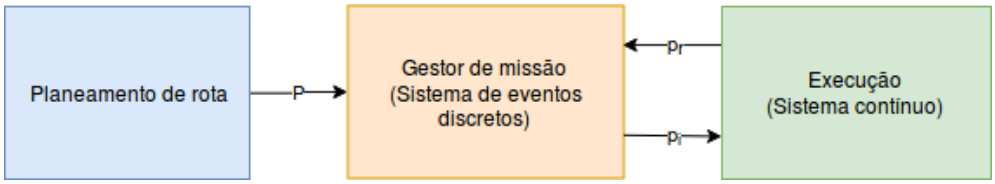


Figura 3.1: Modelos utilizados em ambiente de simulação.

3.3 Simulador

A introdução de um simulador neste projeto tem como principal objetivo obter um ambiente onde os todos os fatores envolventes sejam constantes e não se alterem, de modo a se poder retirar conclusões com os resultados obtidos, pois fatores como o desgaste mecânico do veículo, condições de visualização do sistema de *tracking* ou até mesmo a carga das baterias da embarcação podem perturbar os resultados obtidos.

Desta forma, foi utilizado o simulador da competição Virtual RobotX (VRX), [1], implementado em ambiente de simulação *Gazebo*, representativo de um cenário marítimo, com possibilidade de modulação de parâmetros da dinâmica do ambiente, tais como ondas, ventos e correntes, de representação de um modelo realístico do ASV descrito acima, imagem da esquerda da figura 3.2, tendo em consideração a configuração e os parâmetros aproximados dos *thrusters* reais e a sua massa e de simulação de interações entre o ambiente e a embarcação com recurso ao motor de física *Open Dynamics Engine* (ODE) [2]. É também possível representar modelos de boias e docas, tais como as representadas na imagem central e da direita da figura 3.2, a utilizar como obstáculos e pontos de destino respetivamente.

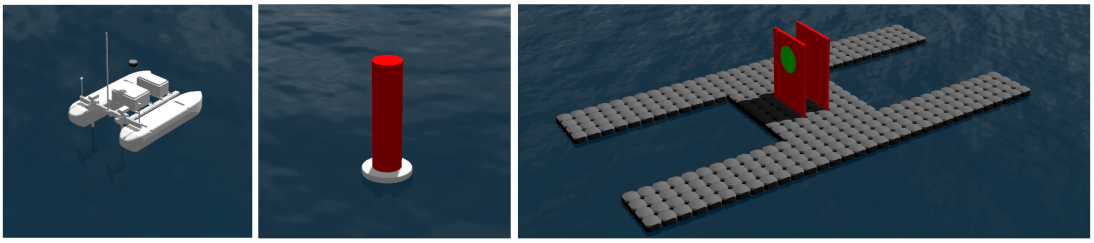


Figura 3.2: Modelos utilizados em ambiente de simulação.

3.3.1 ASV

Os algoritmos desenvolvidos têm como objetivo serem testados em ambiente de simulação *Gazebo* e posteriormente em ambiente real. Para isso, durante a fase de testes por simulador foi utilizado um modelo de um veículo de comprimento aproximadamente 1.5 metros e largura 1 metro, desenvolvido pelo INESC TEC, denominado por Zarco ASV [45], demonstrado na figura 3.3.

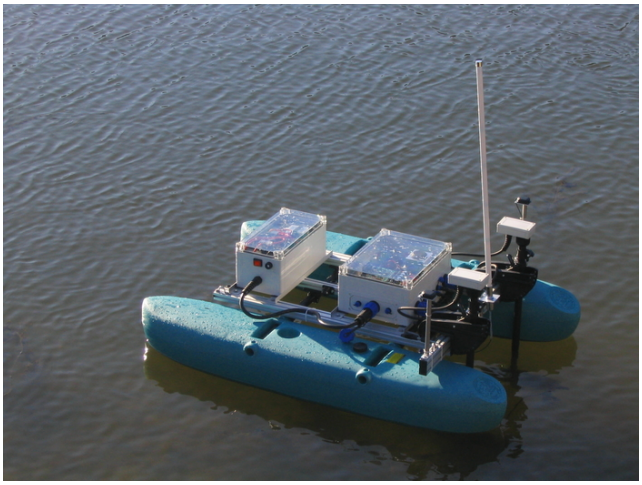


Figura 3.3: Embarcação proposta para ambiente de teste. Imagem retirada de Oceansys[3].

[1] https://bitbucket.org/osrf/vrx/src/default/
[2] http://gazebosim.org/blog/feature_physics

Este ASV é um veículo diferencial, possui uma autonomia de 4 a 6 horas navegando à velocidade nominal de $1\,\mathrm{m\,s^{-1}}$, e é composto por dois *thrusters* localizados atrás, ou seja possui tração traseira e um eixo de rotação recuado 0.3 m em relação ao centro do mesmo segundo o eixo das abcissas. Possui também um sistema de *tracking* que inclui [46]:

- 1 x MTi-30 Xsens IMU, 200 Hz, com 0.2°/0.5° de exatidão angular;
- 1 x recetor Swift Navigation GPS, 20 Hz, L1/L2 RTK, com 0.01 m de exatidão horizontal e 0.015 m de exatidão vertical;
- 2 x Mako G-125 com uma lente de 0.006 m a 30 *frames* por segundo (fps) e uma resolução de 1292x964;
- 1 x Velodyne VLP-16, 4 Hz, com 0.03 m de exatidão de alcance, distância de medição até 100 m e campo de visão de 360° horizontal e 30° vertical.

Na figura 3.4 encontra-se uma descrição da localização destes equipamentos no modelo do ASV implementado no simulador bem como os seus referenciais.

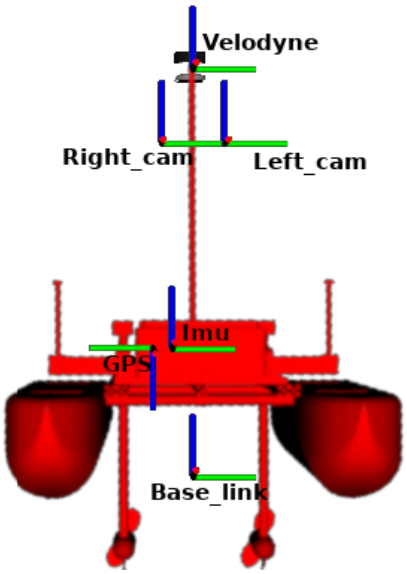


Figura 3.4: Localização dos diversos sensores no modelo do ASV utilizado na fase de testes.

A sua massa é aproximadamente 50 kg e contém ainda uma capacidade de carga adicional de 25 kg. Esta característica em junção com o efeito das ondas e do vento faz com que seja necessário ter em consideração a sua inércia e as restrições dinâmicas do veiculo.

[3] https://oceansys.fe.up.pt/?section=tech&item=#modal_zarco

Capítulo 4

O método de navegação HORIZON

O método HORIZON é um algoritmo de planeamento e execução de rotas com desvio de obstáculos híbrido otimizado para restrições globais e locais. Este consiste numa combinação de dois algoritmos que se complementam através de um gestor de missão. Assim o método HORIZON é capaz de:

- planear diferentes rotas de acordo com a definição dos pesos de uma função de custo, tendo em consideração todos os obstáculos estáticos conhecidos;
- executar a rota planeada, de uma forma segura e eficiente, evitando colisões com obstáculos estáticos e dinâmicos.

Assim este capítulo está essencialmente dividido em três componentes. Numa primeira fase é apresentado o método de planeamento de rota e analisadas as diferentes configurações da sua utilização. De seguida é exposto o algoritmo gestor de missão cujo objetivo é interagir com o controlador da execução de uma trajetória indicando o próximo ponto de destino. Por último é abordado o algoritmo de execução de uma trajetória.

4.1 Planeamento de rota

Tal como foi descrito no capítulo 3 um dos principais problemas a abordar nesta dissertação é a resolução de um TSP de forma a possibilitar ao utilizador a seleção de múltiplos pontos de destino para o ASV, permitindo uma maior flexibilidade de aplicações para este tipo de veículo.

Um TSP é um problema que tenta determinar a melhor rota para percorrer uma série de pontos inspirado na necessidade dos vendedores em realizar entregas em diversos locais percorrendo o menor caminho possível, reduzindo o tempo necessário para a viagem e os possíveis custos com transporte e combustível.

Assim, utilizando um algoritmo capaz de solucionar este tipo de problema, torna-se possível caracterizar e descrever, de uma forma bastante intuitiva, qualquer tipo de missão indicando apenas os pontos sem se ter de preocupar com questões de otimização em relação à ordem com que estes devem ser visitados.

De modo a que o utilizador possa selecionar a quantidade de pontos que pretende que a embarcação visite e a localização destes pontos no mundo foi criado um programa que tem como objetivo interagir com o mesmo utilizando o RVIZ, visualizador 3D para exibição de dados de sensores e informações de estado do ROS. Na figura 4.1 encontra-se um exemplo deste programa com 5 pontos interativos de destino.

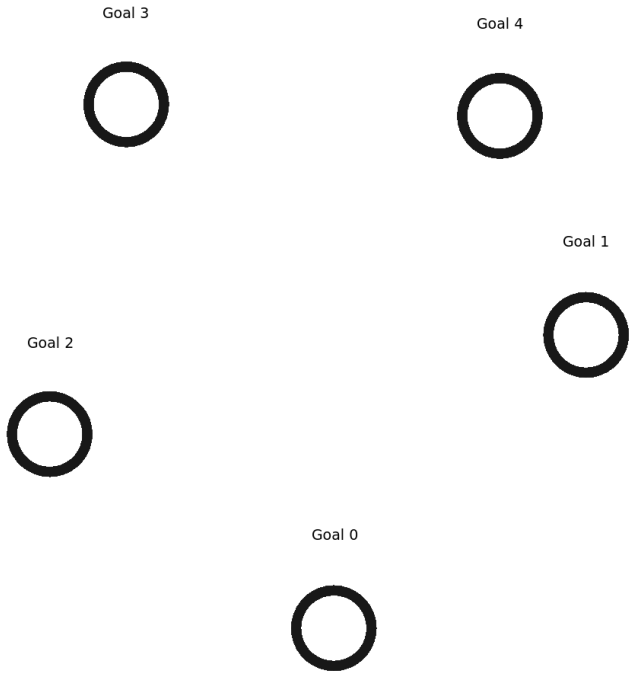


Figura 4.1: Exemplo da localização de 5 pontos de destino na aplicação interativa.

Depois de escolhidas as localizações destes pontos é então necessário selecionar a ordem com estes serão visitados. Para isso foi desenvolvido um algoritmo composto por três componentes, tal como é possível observar na figura 4.2:

- Em primeiro lugar, como os pontos se encontram em ambiente marítimo é necessário obter as melhores trajetórias que unem cada par de pontos tendo em consideração os obstáculos estáticos conhecidos. Para isso é utilizado o algoritmo TEB;
- De seguida procede-se uma formulação multicritério dos dados que tem como objetivo servir como meio de comunicação entre a primeira e a última fase;
- Por fim é utilizado o algoritmo *simulated annealing* como otimizador de rota, sendo capaz de retornar a melhor rota que minimiza a função de custo definida pelo utilizador.

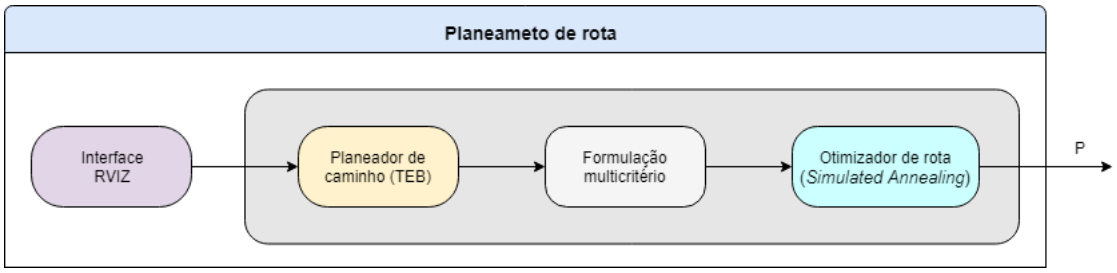


Figura 4.2: Diagrama representativo das interações entre blocos correspondentes ao planeamento de rota.

4.1.1 Planeamento de trajetórias globais

Para se obter todas as trajetórias entre os diversos pontos foi utilizado o algoritmo *Timed-Elastic-Band*. Tal como é referido na análise do estado da arte (secção 2.3.5), este algoritmo calcula primeiramente o caminho e de seguida a sua forma inicial é sujeita a forças artificiais com o objetivo de o deformar, encurtando-o e suavizando-o. Dado que apenas os obstáculos estáticos são conhecidos só estes podem ser considerados no cálculo das trajetórias.

O TEB gera então uma trajetória livre de colisão entre dois pontos. Portanto a figura 4.3 representa a geração de trajetórias com recurso a este método, e consequentemente, o planeamento das trajetórias bidirecionais entre dois pontos. É possível observar a azul as diferentes trajetórias geradas com recurso a este método e a vermelho os obstáculos estáticos.

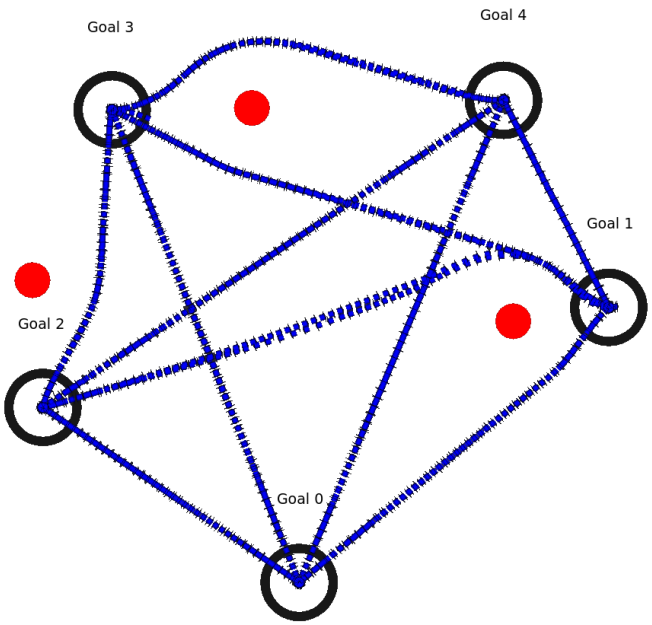


Figura 4.3: Trajetórias obtidas com o método TEB para o exemplo apresentado na figura 4.1.

4.1.2 Formulação multicritério do planeamento de trajetórias

Devido ao facto de os caminhos retornados pelo TEB serem vetores de pontos no espaço é necessário reestruturar este conjunto de dados e obter informações acerca destes para que possam ser utilizadas pelo algoritmo do próximo bloco. Tendo em conta que o algoritmo seguinte a ser aplicado é o *simulated annealing* e que este necessita, não dos caminhos mas sim das distância dos mesmos e das variações de ângulos entre eles, o principal objetivo deste bloco é realizar os cálculos e as operações necessárias de forma a que o próximo possa ser executado de forma mais simples e objetiva possível.

Assim para obter a distância de cada caminho, a solução mais simples proposta, visto que a distância entre pontos é suficientemente curta, é somar as distâncias entre todos os pontos adjacentes. Uma formulação desta solução é apresentada nas equações 4.1 e 4.2.

$$d(i,i+1) = \sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2} \tag{4.1}$$

$$d(k) = \sum_{i=0}^{n-1} d(i,i+1) \tag{4.2}$$

onde $d(i,i+1)$ representa a distância euclidiana entre dois pontos, $d(k)$ a distância total da trajetória k entre dois pontos de destino, n o número de pontos de cada trajetória e x_i e y_i as coordenadas de cada ponto.

Para obter as variações de ângulos entre as trajetórias foram só consideradas as retas que unem pares de pontos. Dado que, para se saber a variação de ângulo entre duas trajetórias é necessário 3 pontos, ponto anterior, atual e final este tipo de dados foi então guardado num vetor de matrizes. Desta forma os ângulos foram obtidos com recurso à equação 4.3.

$$a(l,m,n) = \arctan(y_m - y_k, x_m - x_k) - \arctan(y_k - y_n, x_k - x_n) \tag{4.3}$$

onde $a(l,m,n)$ é a variação de ângulo entre a reta que une os pontos l e m e a reta que une os pontos l e n sendo l o ponto atual, m o ponto final e n o ponto anterior.

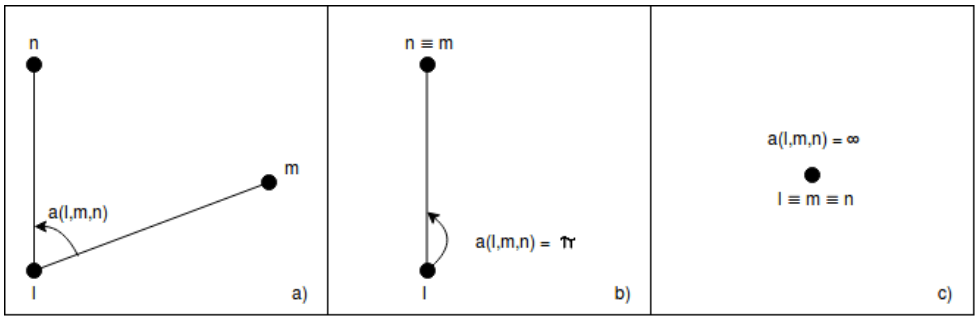


Figura 4.4: Figura representativa das diferentes possibilidades de cálculo para a variação de ângulo entre trajetórias.

De notar que nos casos em que m é coincidente a n foi considerado um ângulo π, sendo assim este o valor máximo de variações de ângulo entre trajetórias e em situações em que k é coincidente a m e n foi considerado um ângulo infinito pois é impossível calcular a variação de ângulo com 3 pontos coincidentes. Na figura 4.4 encontra-se uma breve demonstração de todas as diferentes possibilidades. Na situação a) encontra-se o caso normal, na b) a variação máxima de ângulo e na c) a questão de impossibilidade de cálculo.

Por último, para que se possa utilizar ambos os dados, distâncias e ângulos, juntamente numa única função de custo é necessário proceder a uma normalização destes. Para isso utilizaram-se as equações 4.4 e 4.5 para a normalização das distâncias e ângulos, respetivamente.

$$d_n(k) = d(k)/d_{max} \tag{4.4}$$

$$a_n(l,m,n) = a(l,m,n)/\pi \tag{4.5}$$

em que $d_n(k)$ é a distância da trajetória k normalizada, d_{max} é a distância máxima de todas as trajetórias e $a_n(l,m,n)$ é a variação de ângulo normalizada.

4.1.3 Otimização de rota multicritério

Após a formulação multicritério, torna-se relevante definir uma função de custo e a subsequente minimização. Tipicamente, estas formulações multicritério constituem problemas *NP-hard* e originam diversos mínimos locais que dificultam o processo de otimização. O método SA é uma meta heurística muito eficiente e permite abordar estas situações de uma forma muito intuitiva, tendo já sido aplicado em diversos problemas de robótica [47, 48].

Assim, este bloco tem como objetivo executar o algoritmo *simulated annealling* e obter à saída deste a sequência ordenada de pontos que minimize uma determinada função de custo ($min\ f(x)$) adaptada ao problema de planeamento de rota. Este método permite encontrar o mínimo global de uma função que contém vários mínimos locais simulando o arrefecimento de um corpo a alta temperatura. No processo o algoritmo substitui a solução atual por uma solução próxima de acordo com a função de custo e uma variável T representativa da temperatura. Quanto maior for T, maior a componente aleatória que será incluída na próxima solução escolhida. Uma das principais vantagens do SA é permitir testar soluções mais distantes da solução atual e dar mais independência ao ponto inicial da pesquisa.

O método é iniciado com uma temperatura T_0 elevada que é diminuída no final de cada ciclo por um fator de arrefecimento α. Desta forma, o algoritmo apresenta uma maior probabilidade de fugir a mínimos locais nos instantes iniciais e, à medida que este valor decresce, a probabilidade de aceitação de soluções com um maior custo diminui levando o sistema a convergir para a melhor solução encontrada até ao momento. Para cada valor de temperatura são geradas i_{max} soluções vizinhas da solução atual. Este valor representa o número de iterações para o sistema atingir o equilíbrio térmico em cada temperatura.

Assim, este método começa por partir de uma solução inicial, no qual se optou por se partir de um vetor com os índices dos pontos ordenados por ordem crescente. A cada ciclo um conjunto de novas soluções s_n, vizinhas à solução atual s_a, são calculadas e com cada uma destas é calculado o valor de Δ, sendo este a diferença do custo entre a solução calculada e a atual, como é demonstrado pela equação 4.6.

$$\Delta = f(s_n) - f(s_a) \tag{4.6}$$

onde f representa a função de custo.
Consoante o resultado obtido para o valor de Δ ocorre uma das seguintes condições:

- $\Delta < 0$: significa que há uma diminuição da energia e que por sua vez a nova solução é melhor que a atual. O método aceita a nova solução e s_n passa a ser solução atual;
- $\Delta = 0$: caso de estabilidade sendo indiferente aceitar ou não a nova solução;
- $\Delta > 0$: significa que há um aumento da energia e por isso a aceitação deste tipo de solução só ocorre com determinada probabilidade calculada tendo em consideração o fator de Boltzmann dado por $e^{\Delta/T}$.

Este algoritmo é caracterizado pelos seguintes passos descrito por Bertsimas et al. (1993) [44]:

1. *Inicialização das variáveis* $T = T_0$, T_{min}, α *e* i_{max}*;*
2. *Seleção de uma solução atual* s_a *informada ou aleatória;*
3. *Avaliação da solução atual* $f(s_a)$*;*
4. *Atribuição da solução atual* s_a *à melhor solução* s_m*;*
5. *Enquanto a temperatura T for maior que* T_{min}*:*
   (a) *Enquanto o número de iterações for menor que* i_{max}*:*
      i. *Seleção de uma nova solução* s_n *vizinha de* s_a*;*
      ii. *Avaliar a nova solução* $f(s_n)$*:*
         A. *Se* $f(s_n) < f(s_a)$ *então considerar a nova solução como solução atual e se* $f(s_n) < f(s_m)$ *então considerar a nova solução como melhor solução;*
         B. *Se* $f(s_n) \geq f(s_a)$ *gerar um número aleatório dentro do intervalo [0,1] e verificar se este é menor que* $e^{-\frac{f(s_n)-f(s_a)}{T}}$*. Em caso verdadeiro considerar a nova solução como solução atual.*
      iii. *Aumentar o número referente às iterações de uma unidade.*
   (b) *Diminuir o valor de temperatura pelo fator* α*.*

De forma a estabelecer uma função de custo definiu-se x como uma rota composta por uma a sequência de pontos a visitar, equação 4.7, d_x a distância total normalizada da rota 4.8, e a_x a variação angular total normalizada da rota 4.9.

$$x = \{x_0, x_1, \ \ldots \ , x_i\} \tag{4.7}$$

$$d_x = \sum d_n(x_i) \tag{4.8}$$

$$a_x = \sum a_n(x_i, x_{i+1}, x_{i-1}) \tag{4.9}$$

Assim a função de custo $f(x)$ utilizada, é composta por duas componentes: a distância total normalizada, d_x, da rota x e a soma das variações de ângulo, também normalizadas a_x, efetuadas nessa mesma rota. De forma a permitir uma maior flexibilidade de soluções esta função efetua uma média pesada com dois pesos, w_1 e w_2. A definição do valor dos pesos pode ser realizada através de um outro algoritmo que entre com o modelo da embarcação (por exemplo, o tipo de locomoção, a inércia e ainda, as dimensões). Assim a função de custo pode ser expressa pela equação 4.10.

$$f(x) = w_1 \times d_x + w_2 \times a_x \tag{4.10}$$

Para possibilitar a opção de escolha do ponto inicial e do ponto final todas as soluções calculadas pelo algoritmo mantêm como ponto inicial e final o ponto de menor índice, ou seja índice 0, e o ponto de maior índice, $n-1$ sendo n o número de pontos a visitar, respetivamente.

A título meramente exemplificativo do impacto dos valores de w_1 e w_2, a figura 4.5 demonstra duas rotas distintas que são obtidas quando esses valores são dispares, para os mesmos pontos e tendo em consideração os mesmos obstáculos. De modo a facilitar a visualização do sentido da rota foi utilizado um esquema de cores no qual a primeira trajetória a executar é representada a azul e a última a vermelho. À esquerda encontra-se a rota obtida utilizando o mesmo peso para o fator relativo à distância total e à variação de ângulo ou seja para $w_1 = 0.5$ e $w_2 = 0.5$ e à direita atribuindo a $w_1 = 0.25$ e a $w_2 = 0.75$. Comparando os dois resultados conclui-se que o aumento do valor de w_2 refletiu-se numa alteração de rota capaz de diminuir as variações de ângulo entre as trajetórias. No capítulo dos resultados (capítulo 5) encontra-se realizada de uma forma pormenorizada uma análise de sensibilidade destes pesos.

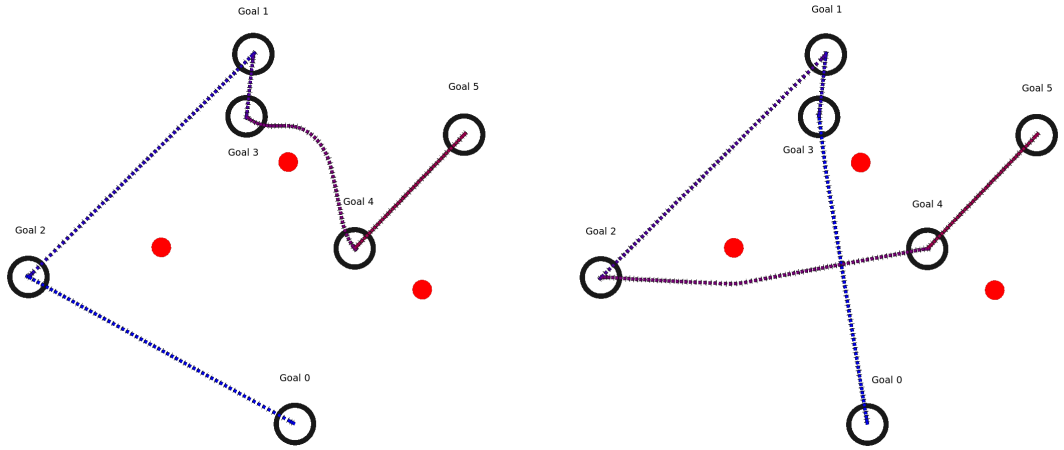


Figura 4.5: Rotas obtidas com diferentes valores nos pesos da função custo, à esquerda 50% para ambos e à direita 25% para w_1 e 75% para w_2.

4.2 Gestor de Missão

Este bloco representa a primeira camada do controlador híbrido descrito anteriormente na figura 3.1. O seu principal objetivo é interagir com o controlador da execução da trajetória indicando o ponto destino tendo sempre em consideração a posição atual da embarcação. Assim é possível descrever este bloco como uma máquina de estados, apresentada na figura 4.6, a qual recebe inicialmente um vetor de pontos p_i ordenados previamente pelo planeamento de rota e informa o bloco seguinte do próximo ponto desejado. Quando a distância atual ao ponto destino, d_a, for inferior a 0.2 m ocorre uma iteração do valor de i passando-se assim a publicar o próximo ponto. O algoritmo termina quando não existirem mais pontos para se publicar ou seja quando i for maior que n.

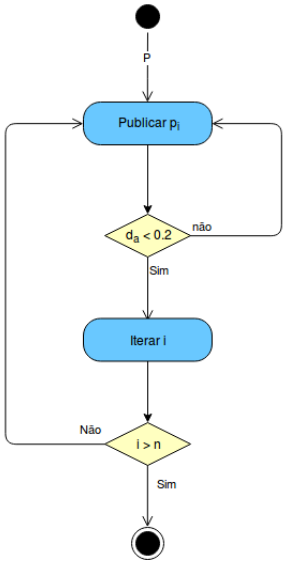


Figura 4.6: Diagrama de estados representativo do bloco gestor de missão.

4.3 Execução da trajetória

Depois de obtida a sequência de pontos e qual o próximo ponto a visitar é necessário um algoritmo capaz de deslocar a embarcação para esse ponto desviando o ASV dos obstáculos apresentando um comportamento preditivo e reativo aos diversos elementos estáticos e dinâmicos. Este algoritmo deve assim funcionar de forma completamente independente de todo o sistema à exceção do ponto de destino, o qual é fornecido pelo gestor de missão. Este deve também ser *online* na medida em que a velocidade a atribuir ao veículo é calculada à medida que o ASV executa a trajetória.

Para isso, o método desenvolvido faz recurso a um planeador global que permite a estimação de uma velocidade livre de colisão através da obtenção de uma trajetória, tendo em consideração os obstáculos estáticos, e de uma representação espacial do ambiente em torno da embarcação considerando os anteriores mais os dinâmicos. Com a trajetória delineada procede-se ao cálculo de uma velocidade estimada que é posteriormente analisada tendo em consideração essa representação, tal como está representado no diagrama da figura 4.7.

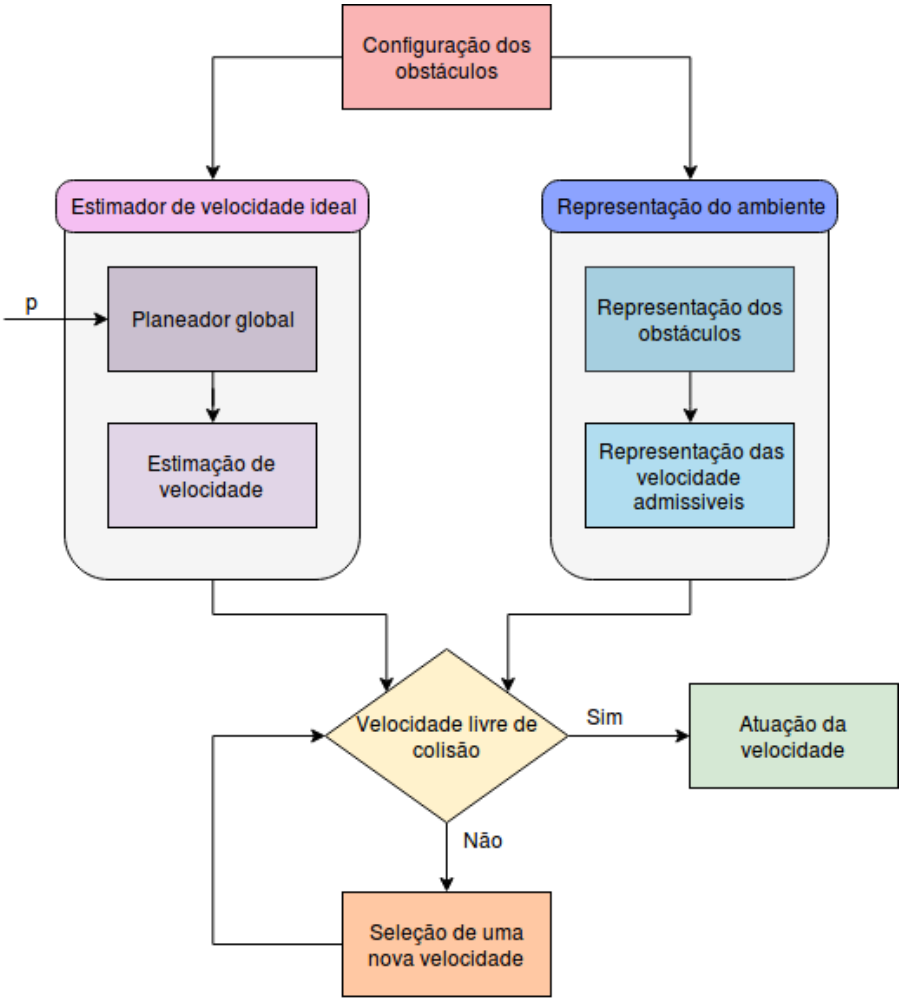


Figura 4.7: Diagrama de estados representativo do bloco de execução da trajetória.

4.3.1 Representação do ambiente

De modo a realizar uma representação do ambiente para posterior verificação da validade da velocidade ideal determinada utilizou-se o conceito *Velocity Obstacles*. Segundo esta teoria é possível caracterizar obstáculos e velocidades numa representação e, desta forma, calcular uma velocidade de movimento pretendida para o veículo que não origine colisões.

Representação dos obstáculos

O conceito VO define os obstáculos estáticos e dinâmicos através de círculos, que se podem movimentar ao longo do espaço de representação com determinada velocidade instantânea conhecida ou possível de ser calculada. O estado de cada objeto é representado por uma posição e um vetor de velocidade com ponto inicial no seu centro. Assim, em primeiro lugar este método assume que todos os obstáculos, estáticos ou dinâmicos, têm raio igual ao seu comprimento máximo somado ao comprimento máximo do veículo. Sendo A a embarcação e B um obstáculo qualquer, desta forma, a embarcação é então reduzida a um ponto $\hat{A}$, como é demonstrado na figura 4.8.

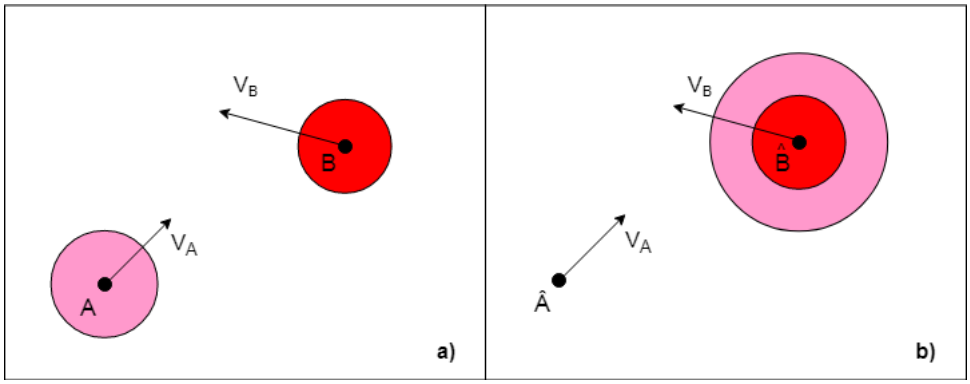


Figura 4.8: Figura representativa da soma do raio do obstáculo em consideração ao tamanho do veículo.

De seguida é definido o cone de colisão $CC_{A,B}$ como sendo o conjunto de velocidades de colisão relativas entre $\hat{A}$ e $\hat{B}$:

$$CC_{A,B} = \{V_A - V_B \mid \lambda_{A,B} \cap \hat{B} \neq \emptyset\} \tag{4.11}$$

onde $\lambda_{A,B}$ é a reta com direção igual à direção de $V_A - V_B$ e que passa pelo ponto $\hat{A}$.

Este cone com vértice em $\hat{A}$ é delimitado pelas duas retas, λ_d e λ_e, tangentes a $\hat{B}$ à direita e à esquerda, respetivamente, tal como na figura 4.9. Verifica-se então que qualquer velocidade relativa fora do cone de colisão é considerada uma velocidade livre de colisão pressupondo que nenhum objeto altere a sua forma ou velocidade.

Cada cone de colisão é especifico a um par obstáculo/veículo e de forma a considerar múltiplos obstáculos cada cone sofre de seguida uma translação com valor igual à velocidade do respetivo obstáculo o que é equivalente à equação 4.12.

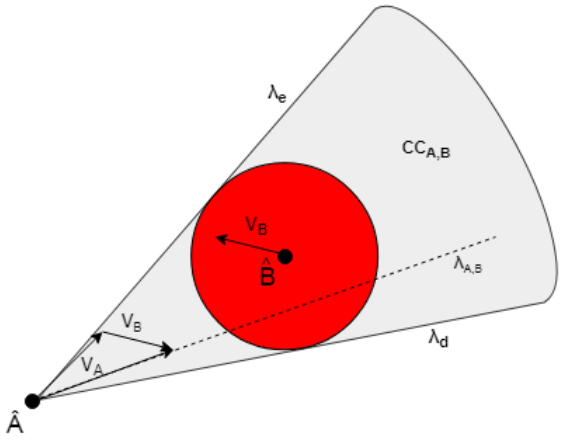


Figura 4.9: Figura representativa de um cone de colisão.

$$VO = CC_{A,B} \oplus V_B \tag{4.12}$$

onde $\oplus$ representa a soma de Minkowski.

Na figura 4.10 estão presentes dois obstáculos onde cada cone de colisão sofreu a sua respetiva translação. Para que o veículo $\hat{A}$ não apresente uma potencial colisão com nenhum dos obstáculos é necessário que o vetor da sua velocidade V_A não seja coincidente com nenhum VO, ou seja:

$$A(t) \cap B(t) = \emptyset \;\; se \;\; V_A(t) \notin VO(t) \tag{4.13}$$

sendo $VO = \cup_{i=0}^{m} VO_{B_i}$, onde m é o número total de obstáculos.

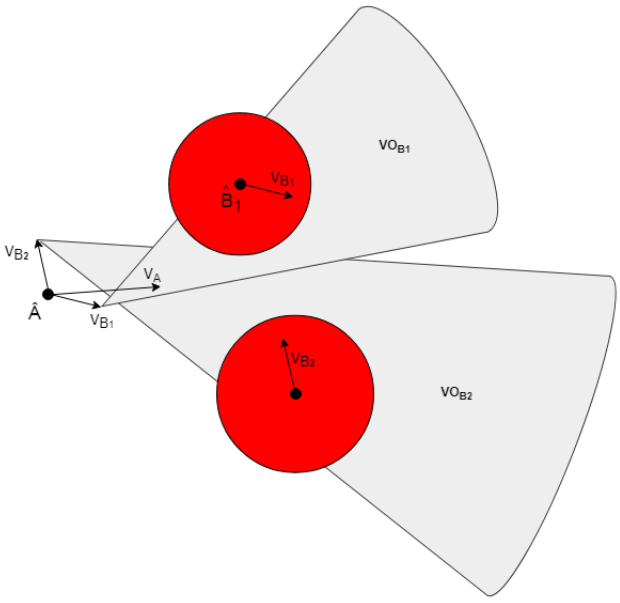


Figura 4.10: Conjunto VO de cada obstáculo B1 e B2.

Em situações onde existe uma grande quantidade de obstáculos é importante priorizar os obstáculos consoante a sua distância ao veículo. Desta forma, obstáculos que se encontrem mais perto terão maior influência pois apresentam um risco de colisão maior.

Desta forma os conjuntos VO foram modificados, subtraindo-lhes VO_h, componente de VO que não apresenta risco de colisão eminente, dado pela equação 4.14. Denomina-se colisão eminente a uma colisão que ocorrerá num tempo $t < T_h$, sendo que este valor depende do tempo de processamento e da dinâmica do sistema.

$$VO_h = \{v_A \mid v_A \in VO, \|v_{A,B}\| \leq \frac{d_m}{T_h}\} \tag{4.14}$$

onde d_m é a distância do obstáculo ao veículo.

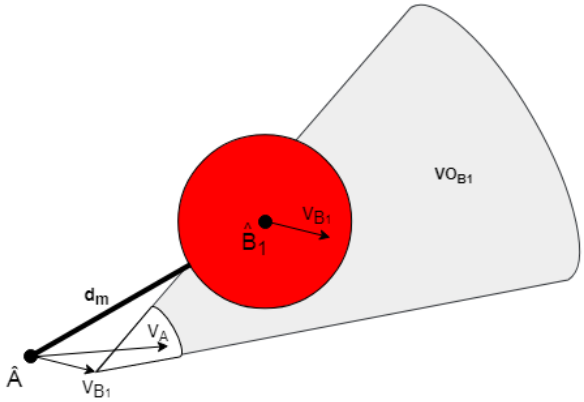


Figura 4.11: Novo conjunto VO relativo ao obstáculo B1 tendo em consideração o risco de colisão.

Recorrendo à biblioteca de visão computacional *OpenCV* foi então gerada uma imagem de 1000 linhas e 1000 colunas onde foram desenhados, com uma resolução de 0.05 m todos os círculos correspondentes aos obstáculos que se encontram ao redor do veículo a uma distância máxima de 50 metros. Assumiu-se para isso que a posição da embarcação na imagem é fixa e que esta se encontra no ponto central. De seguida foram utilizadas as equações [1] 4.15 e 4.16 para determinar os pontos pertencentes aos círculos, que pertencem também às retas tangentes a estes e que passam no ponto central da imagem, tal como demonstrado na figura 4.12.

$$x_{1,2} = \frac{r^2(x_p-a) \pm r(y_p-b)\sqrt{(x_p-a)^2+(y_p-b)^2-r^2}}{(x_p-a)^2+(y_p-b)^2} + a \tag{4.15}$$

$$y_{1,2} = \frac{r^2(y_p-b) \mp r(x_p-a)\sqrt{(x_p-a)^2+(y_p-b)^2-r^2}}{(x_p-a)^2+(y_p-b)^2} + b \tag{4.16}$$

Desta forma obtiveram-se então os pontos necessários para traçar as retas λ_d e λ_e indicadas da figura 4.9.

[1] http://www.ambrsoft.com/TrigoCalc/Circles2/CirclePoint/CirclePointDistance.htm

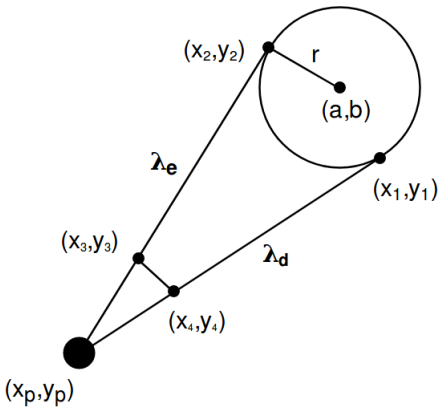


Figura 4.12: Esquema representativo do cálculo dos pontos tangenciais à circunferência.

Consoante o valor de T_h definido pelo utilizador foram de seguida calculados os dois restantes pontos, (x_3,y_3) e (x_4,y_4).

Por último foram somados a todos estes pontos o valor da velocidade instantânea referente ao obstáculo em questão. Na figura 4.13 encontra-se um exemplo do resultado obtido para um conjunto de quatro obstáculos estáticos e um obstáculo dinâmico com velocidade linear constante de $0.5\,\mathrm{m\,s^{-1}}$ e com direção da direita para a esquerda. No centro encontram-se os eixos representativos da direção da embarcação, sendo o vermelho relativo a x e o verde a y, e à sua volta, os círculos dentro dos cones a cinza claro representam obstáculos estáticos enquanto que o círculo preto representa o obstáculo dinâmico. Verifica-se então que os cones de colisão relativos aos obstáculos estáticos são tangentes aos seus círculos, enquanto que o do obstáculo dinâmico se encontra projetado para a esquerda devido à sua velocidade atual.

Representação das velocidades

Depois de obtida a imagem com a representação dos obstáculos e dos seus respetivos cones de colisão procedeu-se à representação da zona de velocidades alcançáveis. Este conjunto de velocidades é calculado em cada iteração do algoritmo e permite ter em consideração fatores como restrições da dinâmica do veículo e dos seus atuadores.

Segundo o conceito VO o conjunto das velocidades alcançáveis é definido como o conjunto de todas as velocidades que o veículo pode atingir, quando este se encontra num determinado estado, durante um intervalo de tempo Δt pré-definido. Para ser possível determinar este conjunto é necessário em primeiro lugar definir o conjunto das acelerações viáveis $AV(t)$, sendo este definido como:

$$AV(t) = \{\ddot{x} \mid \ddot{x} = f(x,\dot{x},u),\ u \in U\} \tag{4.17}$$

onde x representa o vetor posição, $f(x,\dot{x},u)$ a dinâmica do veículo, u o vetor das forças de atuação e U o conjunto de todas as forças de atuação possíveis.

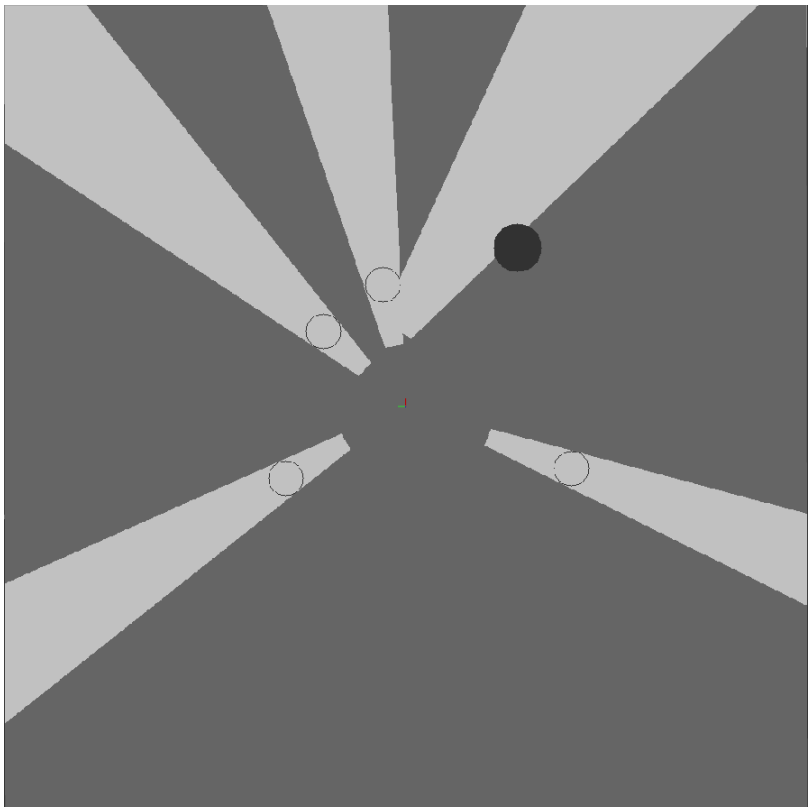


Figura 4.13: Representação dos obstáculos segundo conceito VO.

De modo a simplificar o problema foram então definidas como constantes as acelerações lineares e angulares do veículo para todos os instantes de tempo e todos os estados. Tendo em consideração que o tempo de execução do bloco de código é aproximadamente 50 ms, a aceleração linear foi definida como $1\,\mathrm{m\,s^{-2}}$ e a aceleração angular como $0.5\,\mathrm{rad\,s^{-2}}$.

Com estes valores atribuídos é então possível definir o conjunto das velocidades alcançáveis, $VA(t+\Delta t)$, como:

$$VA(t+\Delta t) = \{v \mid v = v_A(t) \oplus \Delta t \cdot AV(t)\} \tag{4.18}$$

Com recurso à equação 4.18 obtém-se então o conjunto das velocidades alcançáveis multiplicando o valor das acelerações possíveis pelo intervalo de tempo definido e somando à velocidade atual do ASV, como demonstrado esquematicamente na figura 4.14.

Por fim, realizou-se a interseção deste conjunto de velocidades com todos os cones de colisão obtendo-se o conjunto de velocidades alcançáveis livres de colisão eminente, *VAL*. Esta operação pode ser definida pela equação 4.19.

$$VAL(t+\Delta t) = VA(t+\Delta t) \ominus VO(t) \tag{4.19}$$

onde $\ominus$ é o operador de diferença de conjuntos.

Um exemplo da combinação destas duas representações pode ser observado na figura 4.15.

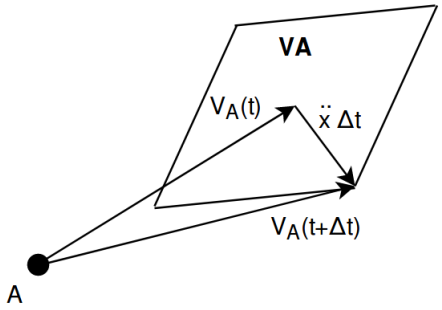


Figura 4.14: Esquema representativo da obtenção do conjunto das velocidades alcançáveis.

Através desta imagem é possível concluir que para não existir colisão deve-se escolher uma velocidade a atuar na embarcação pertencente ao conjunto VA representado pelo losango e fora de todos os cones de colisão VO, ou seja pertencente ao conjunto VAL.

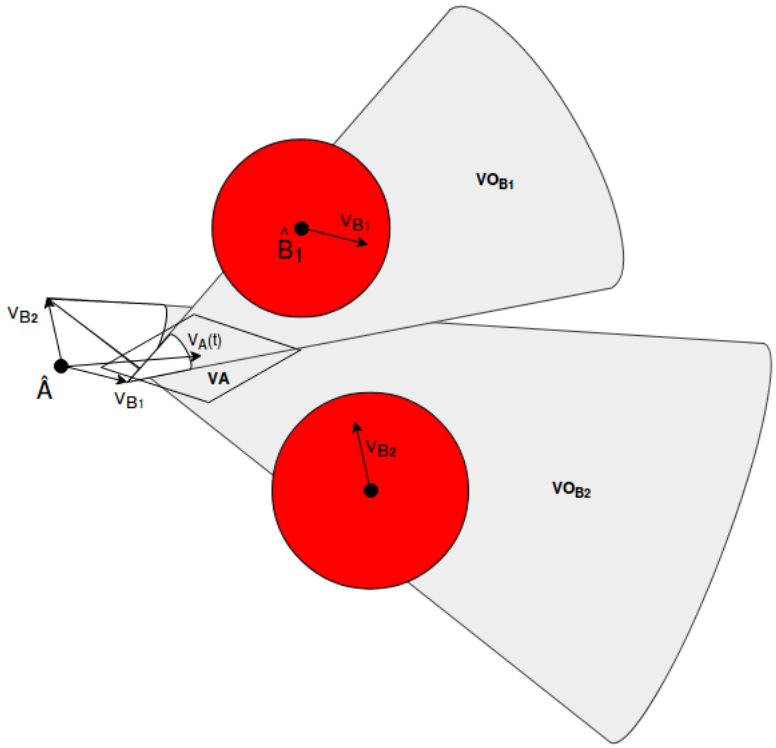


Figura 4.15: Figura representativa do conjunto VAL.

4.3.2 Estimador de velocidade ideal

Para a obtenção de uma velocidade ideal a atuar no ASV é utilizado um planeador global com o objetivo de traçar uma trajetória entre a posição atual do veículo e o ponto de destino, p_i, obtido através do gestor de missão. Assim, este planeador deve ter em consideração apenas

os obstáculos estáticos e gerar a cada iteração do algoritmo uma nova trajetória com recurso à informação obtido no momento. Como anteriormente, para isso foi utilizado o algoritmo TEB. Na figura 4.16 encontra-se um exemplo de uma trajetória obtida com recurso a este algoritmo.

Figura 4.16: Exemplo de trajetória entre a embarcação e um determinado ponto de destino obtida com recurso ao método TEB.

A velocidade instantânea ideal é então obtida através desta trajetória. Sabendo que a trajetória retornada é um conjunto de coordenadas com dada orientação, após vários testes experimentais, verificou-se que a maneira mais simples e rápida de obter essa velocidade é utilizando um ponto do conjunto próximo do ASV. Tendo isto em consideração, foi utilizado o décimo quinto ponto. Levando em conta a distância e o erro de orientação do veículo em relação à orientação desse ponto, a referência para a velocidade linear e angular é estimada com recurso às equações 4.20 e 4.21.

$$V_L = f(\sqrt{(x_P - x_R)^2 + (y_P - y_R)^2}) \tag{4.20}$$

$$V_A = g(\arctan(\theta_P - \theta_R)) \tag{4.21}$$

onde $f(x)$ e $g(x)$ são funções de normalização, x_P, y_P e θ_P são a posição e orientação do ponto alvo e x_R, y_R e θ_R são a posição e orientação reais do ASV, tal como demonstrado na figura 4.17 sendo x_m e y_m os eixos do mundo.

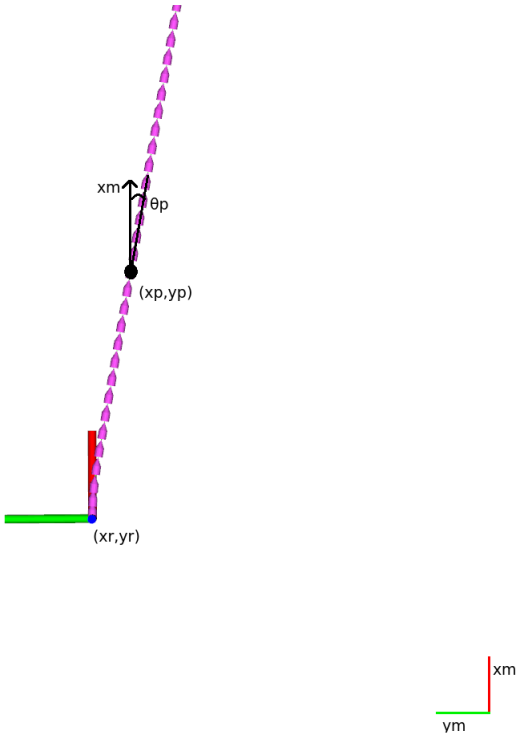


Figura 4.17: Esquema representativo do cálculo da velocidade ideal.

4.3.3 Seleção de uma nova velocidade

Segundo Fiorini [27] ao escolher uma velocidade do conjunto VAL garante-se automaticamente sobrevivência. De forma a utilizar o conceito VO em aplicações *online*, uma velocidade que garanta uma trajetória livre de colisão pode ser obtida utilizando uma heurística para encontrar uma solução local adequada.

Tal como explicado por R. Silva et al. (2019) [3] tendo em consideração a referência de velocidade ideal obtida anteriormente, em primeiro lugar procede-se à verificação da sua validade. Para isso verifica-se se essa potencial velocidade a atuar pertence ao conjunto das velocidades alcançáveis livres de colisão, VAL. Caso essa condição não se verifique procede-se então à escolha de uma nova velocidade utilizando para isso a seguinte heurística:

- Seleção da velocidade com menor erro em norma e direção, relativa à referência, pertencente ao conjuntos das velocidades alcançáveis livres de colisão.

Assim, a escolha da nova velocidade passa pelos seguintes três passos:

1. Representação da velocidade ideal no mapa das representações dos obstáculos e das velocidades;
2. Seleção do ponto pertencente a VAL mais próximo do ponto de referência gerado;
3. Conversão do novo ponto para velocidade utilizando as equações 4.20 e 4.21.

Capítulo 5

Resultados

Neste capítulo serão apresentados os resultados obtidos para os métodos implementados para a resolução da problemática desta dissertação e comparados com algoritmos utilizados pela comunidade.

5.1 Introdução

5.1.1 Estrutura dos resultados

Este capítulo começa por realizar, em primeiro lugar, uma descrição do ambiente de simulação utilizado durante a fase de testes. De seguida são apresentadas as diferentes rotas resultantes do algoritmo de planeamento HORIZON, para um caso específico, e posterior comparação com o resultado obtido utilizando o algoritmo Dijkstra referido no estado da arte 2.4.2. São também apresentados os resultados da execução de duas trajetórias utilizando o algoritmo HORIZON e o algoritmo TEB, sendo a primeira com um obstáculo estático e a segunda com um obstáculo dinâmico, de modo a realizar uma comparação. Por último encontra-se uma descrição dos resultados adquiridos ao executar uma missão por completo.

5.1.2 Ambiente de simulação

Como descrito na secção 3.3 o simulador utilizado permite representar um cenário marítimo com possibilidade de modulação dos parâmetros da dinâmica do ambiente de forma a tornar possível e a facilitar a transição do ambiente de simulação para o mundo real.

Segundo a descrição apresentada no manual [1] do simulador o campo de ondas é gerado simplesmente utilizando três diferentes funções de onda sobrepostas enquanto que o vento é constante e caracterizado por um vetor tridimensional representativo da sua velocidade, em $\mathrm{m\,s^{-1}}$, segundo x, y e z.

Desta forma o campo de ondas do ambiente de simulação foi gerado segundo a tabela 5.1 e o vento foi considerado nulo.

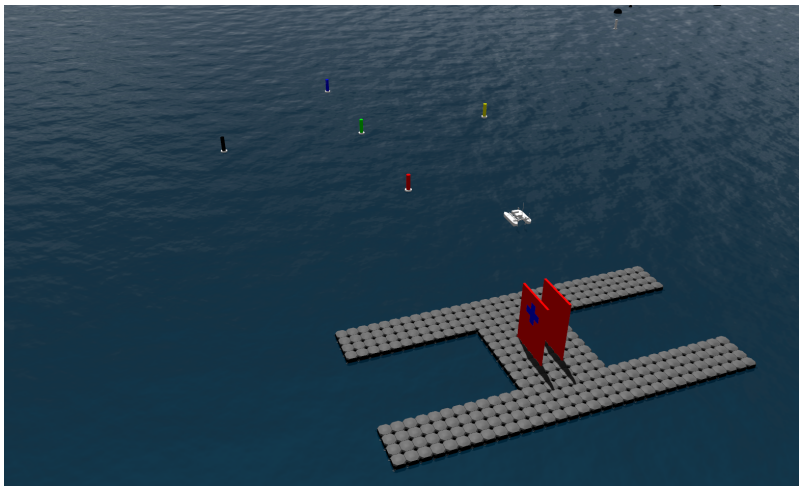
[1] https://github.com/bsb808/robotx_docs/blob/master/theoryofoperation/theory_of_operation.pdf

Tabela 5.1: Componentes do campo de ondas simulado.

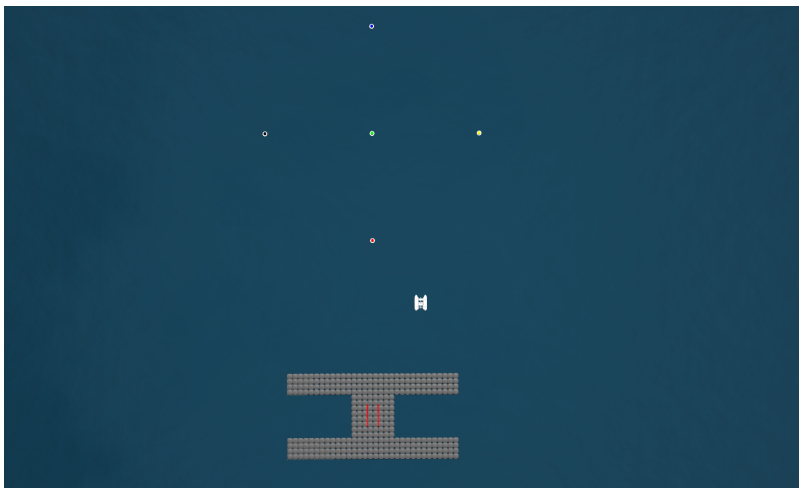
Componente	Amplitude	Período	Direção (x)	Direção (y)
1	0.06	12.6	-1	0
2	0.04	3.7	-0.7	0.7
3	0.03	6.3	0.7	0.7

5.2 Resultados do algoritmo de Planeamento

Nesta secção encontram-se os resultados obtidos para o planeamento de uma missão considerando o conjunto de pontos da figura 5.2, sendo o *Goal* 0 considerado o ponto inicial e *Goal* 9 o ponto final, e utilizando o ambiente de simulação descrito anteriormente em conjunto com a distribuição de bóias e doca apresentados na figura 5.1. Estes resultados são primeiramente obtidos utilizando o algoritmo Dijkstra e de seguida comparados com os diferentes resultados recorrendo ao algoritmo desenvolvido no decorrer desta dissertação.



(a) Vista em perspetiva.



(b) Vista aérea.

Figura 5.1: Ambiente de simulação utilizado na fase de testes do planeamento.

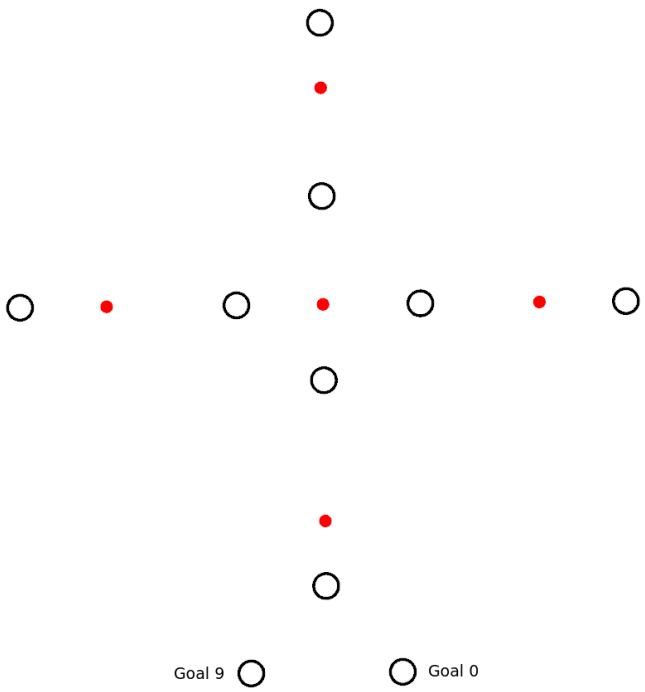


Figura 5.2: Conjunto de pontos a visitar e obstáculos no teste do algoritmo de planeamento.

5.2.1 Planeamento de rota através do algoritmo Dijkstra

O algoritmo Dijkstra é uma técnica de pesquisa que tenta resolver o problema realizando uma escolha localmente ótima em cada iteração com a esperança de encontrar um ótimo global. Assim a rota é gerada com a incorporação iterativa do vizinho mais próximo (que ainda não esteja incorporado na solução atual). Desta forma este método implementado consiste nos seguintes passos:

1. Inicialização de um vetor com tamanho igual ao número de pontos a visitar;
2. Inserção do nó inicial, nó 0, ao vetor;
3. Verificação do nó que se encontra mais próximo do último ponto adicionado ao vetor e que ainda não tenha sido adicionado ao mesmo;
4. Inserção do nó encontrado;
5. Voltar ao ponto 3 enquanto existirem pontos a visitar.

Assim para o conjunto de pontos a visitar demonstrado na figura 5.2 este algoritmo retornou a rota representada na figura 5.3. Esta rota apresenta uma distância total 101.57 metros e uma variação total de ângulo igual a 626.54°.

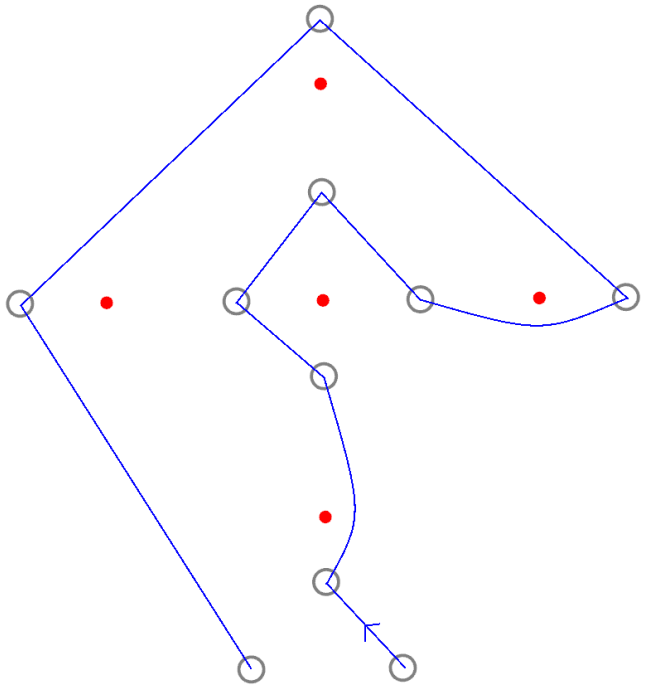


Figura 5.3: Rota obtida com o algoritmo Dijkstra.

5.2.2 Planeamento de rota através do algoritmo HORIZON

De forma a comparar com o método anterior, visto este só considerar as distâncias entre os diferentes pontos, utilizaram-se, em primeiro lugar, como pesos da função de custo de $w_1 = 1$ e $w_2 = 0$. Assim, tal como no algoritmo Dijkstra apenas as distâncias entre os pontos são consideradas pois w_1 representa o ganho relativo às distâncias das trajetórias e w_2 o ganho relativo à variação angular. Nesta configuração, a trajetória gerada encontra-se na figura 5.4. Tal como anteriormente, foi utilizado o mesmo esquema de cores para a representação do sentido da rota, sendo a primeira trajetória a executar representada a azul e a última a vermelho. A distância total da rota obtida foi 94.67 metros e a variação angular foi 612.89°.

Tabela 5.2: Tabela comparativa dos resultados obtidos para o algoritmo Dijkstra e o algoritmo HORIZON.

	Dijkstra	HORIZON ($w_1 = 1$ e $w_2 = 0$)	Variação	Variação (%)
Distância total (m)	101.57	**94.67**	6.9	6.79
Variação angular (°)	626.54	**612.89**	13.65	2.18

Na tabela 5.2 estão apresentados os resultados obtidos com recurso ao algoritmo Dijkstra e ao algoritmo desenvolvido (HORIZON), para o mesmo caso e levando em conta apenas as distâncias das trajetórias. Verifica-se então que este último retorna uma rota aproximadamente 6.79% mais curta e com uma variação angular 2.18% menor.

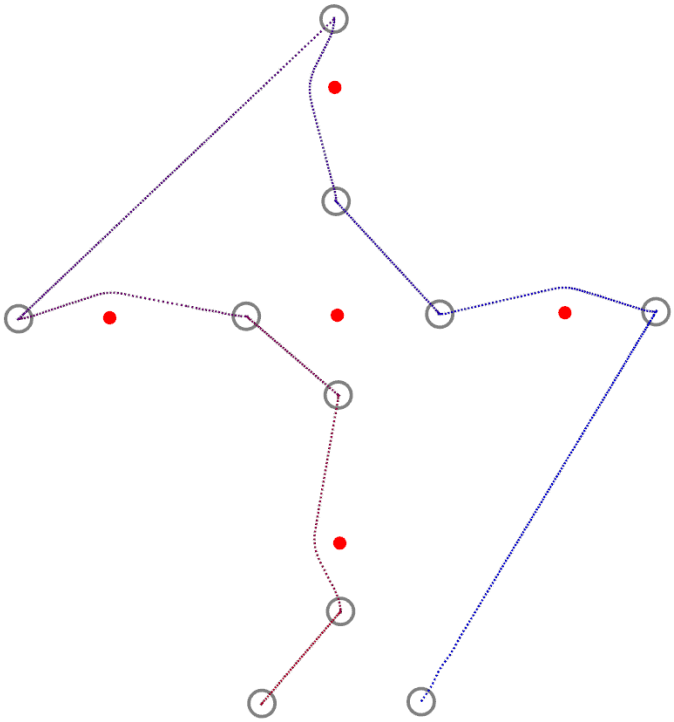
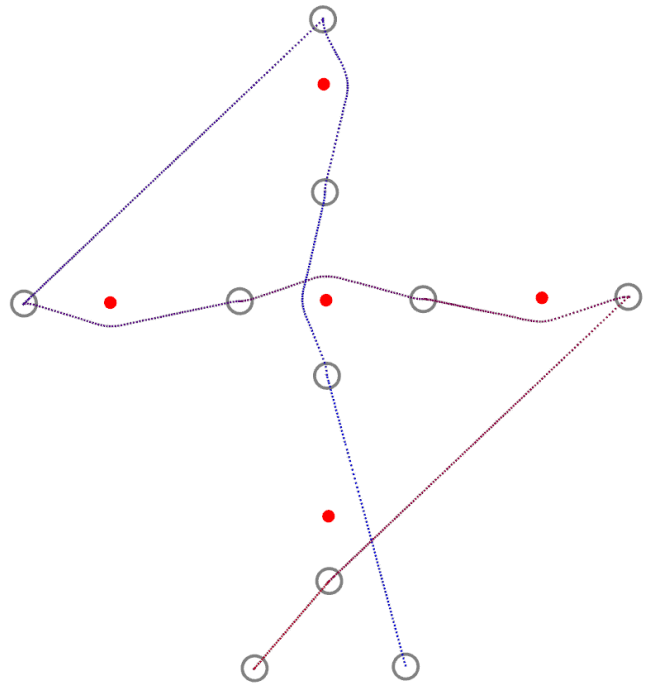


Figura 5.4: Rota obtida com o algoritmo HORIZON considerando apenas as distâncias das trajetórias.

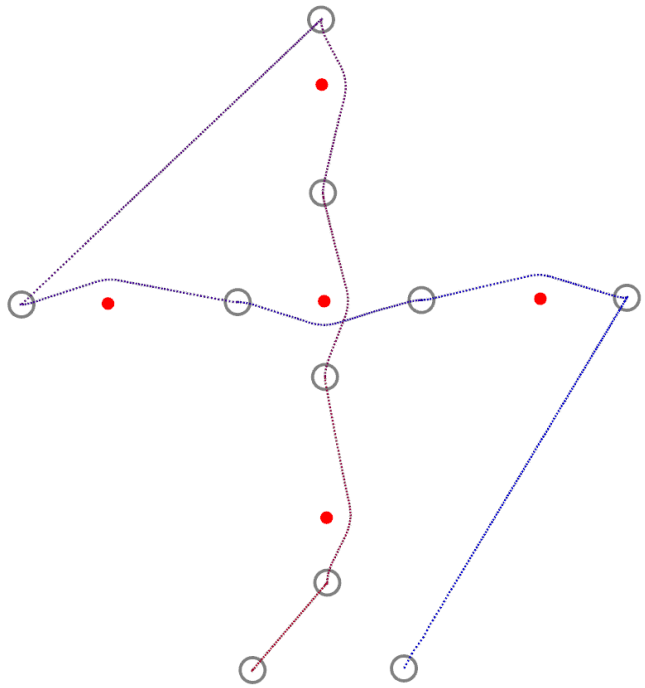
De seguida é apresentada uma breve análise de sensibilidade, para se estudar o impacto dos pesos w_1 e w_2 na estimação de diferentes rotas. Para isso variaram-se os valores da função de custo w_1 e w_2 de forma a demonstrar as diferentes soluções apresentadas pelo algoritmo consoante as preferências do utilizador. Assim, foram obtidas mais duas rotas, distintas das anteriores. A rota demonstrada na figura 5.5a foi obtida tendo apenas em consideração as variações angulares e, por isso, utilizando os pesos $w_1 = 0$ e $w_2 = 1$, enquanto que a rota 5.5b foi gerada com igual peso nestes fatores. Estas apresentam uma distância total e uma variação total de ângulo igual às representadas na tabela 5.3. Pela análise da tabela é possível concluir que o método desenvolvido responde de acordo com as preferências do utilizador ou seja, consoante os valores dos pesos da função de custo. Verifica-se também que a última trajetória, com iguais pesos, pode ser considerada a melhor solução pois, com um pequeno aumento de 5.57% do tamanho da rota, foi obtida uma enorme diminuição da variação angular, 29.37%.

Tabela 5.3: Resultados de distância total e variação angular para os diferentes valores de função custo.

	$w_1 = 1$ e $w_2 = 0$	$w_1 = 0$ e $w_2 = 1$	$w_1 = 0.5$ e $w_2 = 0.5$
Distância total (m)	**94.67**	103.51	100.26
Variação angular (°)	612.89	**427.58**	432.89



(a) Rota obtida com $w_1 = 0$ e $w_2 = 1$



(b) Rota obtida com $w_1 = 0.5$ e $w_2 = 0.5$

Figura 5.5: Rotas obtidas para diferentes valores dos pesos da função de custo.

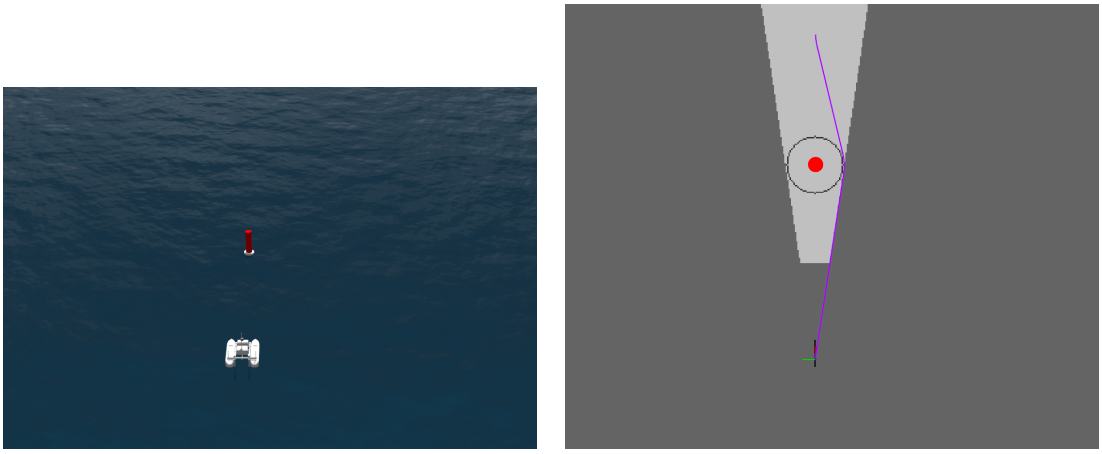
5.3 A execução de trajetórias com o desvio de obstáculos estáticos e dinâmicos.

Nesta secção encontram-se os resultados obtidos para o algoritmo de execução de trajetórias. Para isso esta secção encontra-se dividida em três componentes. Numa primeira parte são apresentados os resultados obtidos numa trajetória que consiste em desviar de um obstáculo estático. De seguida são demonstrados os resultados para o caso de um obstáculo dinâmico. E por último são apresentados os resultados obtidos na execução de uma missão gerada na secção anterior.

Estes resultados são primeiramente obtidos utilizando o algoritmo HORIZON e de seguida comparados com o resultado obtido utilizando o algoritmo TEB.

5.3.1 Comportamento apresentado com obstáculo estático

De modo a testar o algoritmo desenvolvido foi primeiramente inicializado um cenário de teste contendo o ambiente de simulação descrito em 5.1.2, um ASV e um obstáculo estático, bóia, posicionada no ponto (0,0), tal como representado na figura 5.6a. O objetivo do algoritmo é deslocar o veículo para um ponto atrás da bóia vermelha, posicionado a uma distância de 5 metros desta.



(a) Caso de teste com um obstáculo estático.

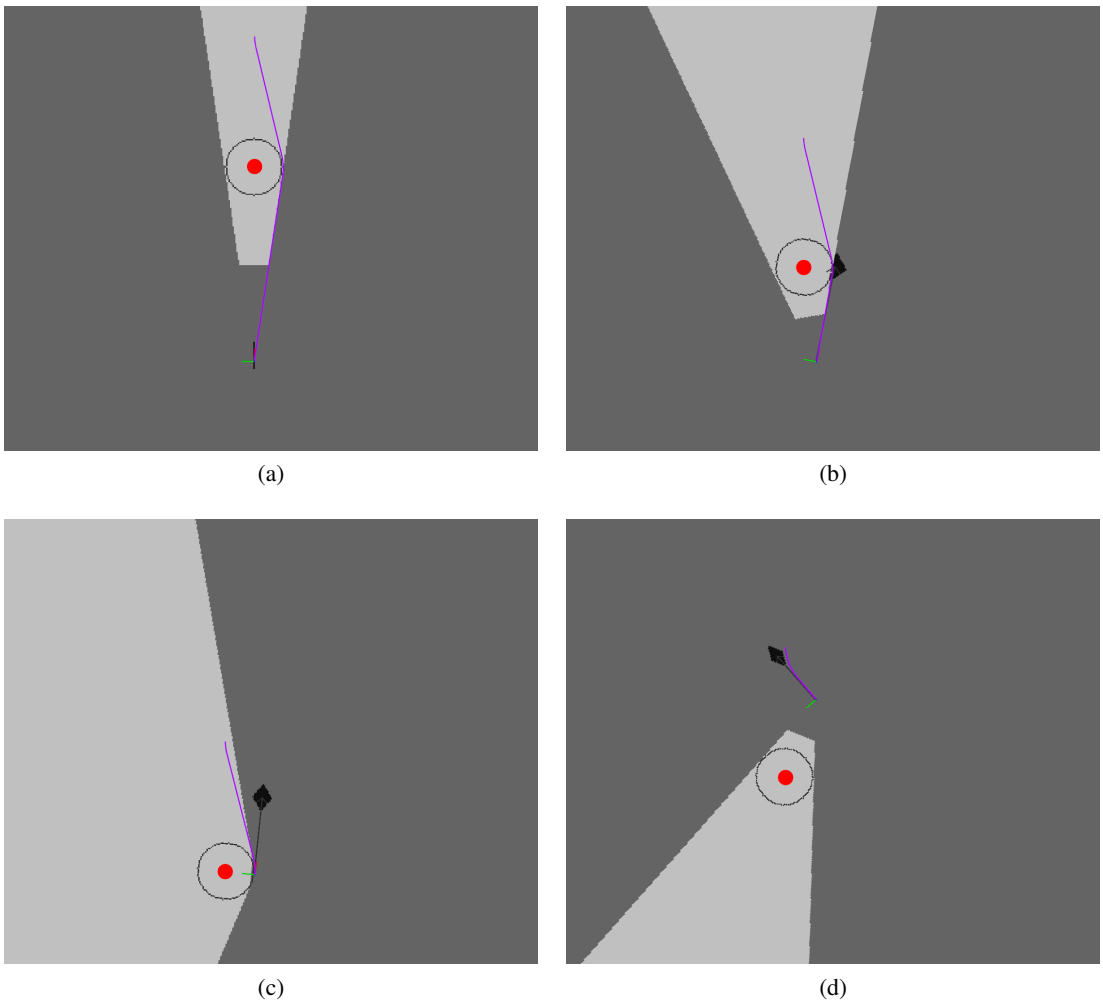
(b) Representação do ambiente obtida para o caso de teste com um obstáculo estático.

Figura 5.6: Figura representativa do caso de teste com um obstáculo estático e a sua representação do ambiente.

Utilizando o algoritmo Horizon obteve-se a representação do ambiente e trajetória do planeador global, no ponto inicial, representados na figura 5.6b.

A sequência de imagens da figura 5.7 tem como objetivo exemplificar a sucessão de passos que o algoritmo segue de forma a contornar o obstáculo e dirigir-se para o ponto de destino. Assim, o método HORIZON começa por realizar a representação do ambiente e por delinear uma trajetória, tendo em consideração o obstáculo (figura 5.7a). De seguida, tendo em consideração a trajetória planeada, o método começa por deslocar o veículo segundo esta. Na figura 5.7b é

possível observar que a zona de velocidades alcançáveis (a preto) nesse instante se encontra fora do cone de colisão com o obstáculo, garantindo, ao escolher uma velocidade desse conjunto, uma velocidade livre de colisão. Com isto o ASV desloca-se até chegar a um ponto ao lado direito do obstáculo (figura 5.7c). A partir deste ponto o cone de colisão encontra-se à esquerda do mesmo e o ASV pode assim começar a direcionar-se para o destino à medida que se move. Na figura 5.7d é possível então verificar a embarcação a dirigir-se para o ponto destino sem qualquer interferência de obstáculos.



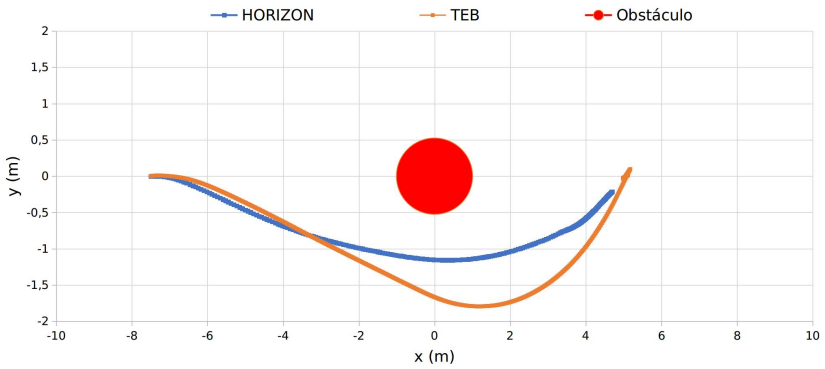
(a) (b) (c) (d)

Figura 5.7: Evolução temporal da representação do ambiente e da trajetória no desvio a um obstáculo estático.

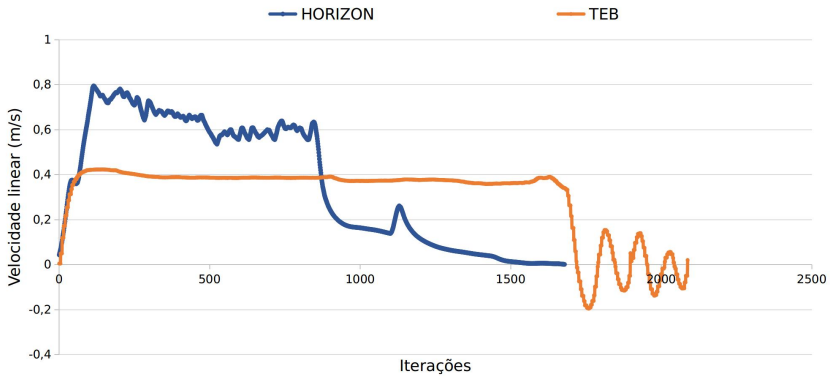
Na figura 5.8 estão representados os gráficos dos caminhos obtidos, das velocidades lineares e das velocidades angulares aquando da utilização do algoritmo desenvolvido e do TEB. É possível verificar que a trajetória executada pelo algoritmo TEB é substancialmente maior. Isto deve-se ao facto de este realizar uma deslocação de encontro ao ponto final num instante de tempo posterior.

Em relação à velocidade linear verifica-se que o algoritmo HORIZON permite que o veículo se desloque a uma velocidade maior, diminuindo ligeiramente à medida que o ASV se aproxima do

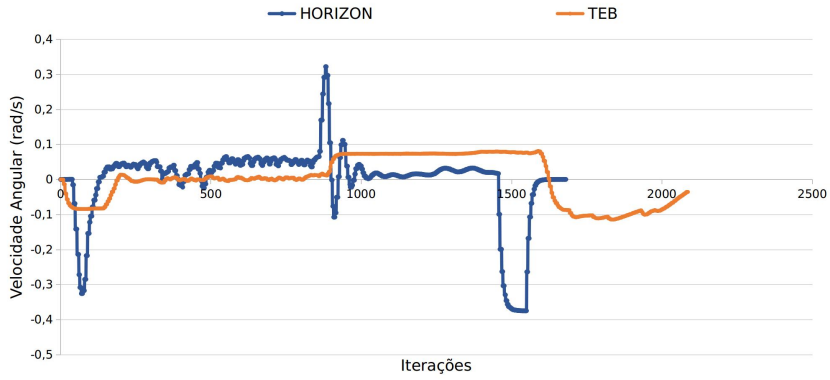
obstáculo, e apresenta nos últimos instantes uma transição suave para zero conforme este alcança o ponto de destino. Com recurso ao TEB, o ASV move-se a uma velocidade constante, independentemente da distância ao obstáculo, e na fase final ocorre uma transição rápida provocando um *overshooting* de $0.2\,\mathrm{m\,s^{-1}}$.



(a) Trajetórias executadas com recurso aos algoritmos HORIZON (azul) e TEB (laranja).



(b) Velocidades lineares apresentadas pelos algoritmos HORIZON (azul) e TEB (laranja).



(c) Velocidades angulares apresentadas pelos algoritmos HORIZON (azul) e TEB (laranja).

Figura 5.8: Gráfico das trajetórias executadas e das velocidades lineares e angulares no comportamento apresentado com um obstáculo estático.

Quanto ao gráfico das velocidades angulares o algoritmo desenvolvido apresenta três grandes picos. O primeiro tem como objetivo direcionar o veículo para uma trajetória livre de colisão. O segundo ocorre depois do ASV ultrapassar o obstáculo e tem como objetivo direcioná-lo para o ponto destino. Por fim, o último, que acontece quando o ASV alcança o ponto de destino, tem como propósito direcionar o barco segundo o eixo do deslocamento.

Na tabela 5.4 encontram-se representados os diferentes resultados obtidos em relação à distância total percorrida, distância mínima ao obstáculo e o tempo de execução da trajetória. Pela análise desta verifica-se que o algoritmo HORIZON apresenta um melhoramento em todos os aspetos analisados, sendo uma diminuição de 17.51% no que toca à distância total e uma redução de 19.45% do tempo de execução.

Tabela 5.4: Tabela demonstrativa das distâncias totais percorridas, das distâncias mínimas ao obstáculo e dos tempos de execução obtidos pelos algoritmos HORIZON e TEB.

	HORIZON	TEB
Distância Total (m)	**14.32**	17.36
Tempo de Execução da Trajetória (s)	**33.62**	41.74

5.3.2 Comportamento apresentado com obstáculo dinâmico

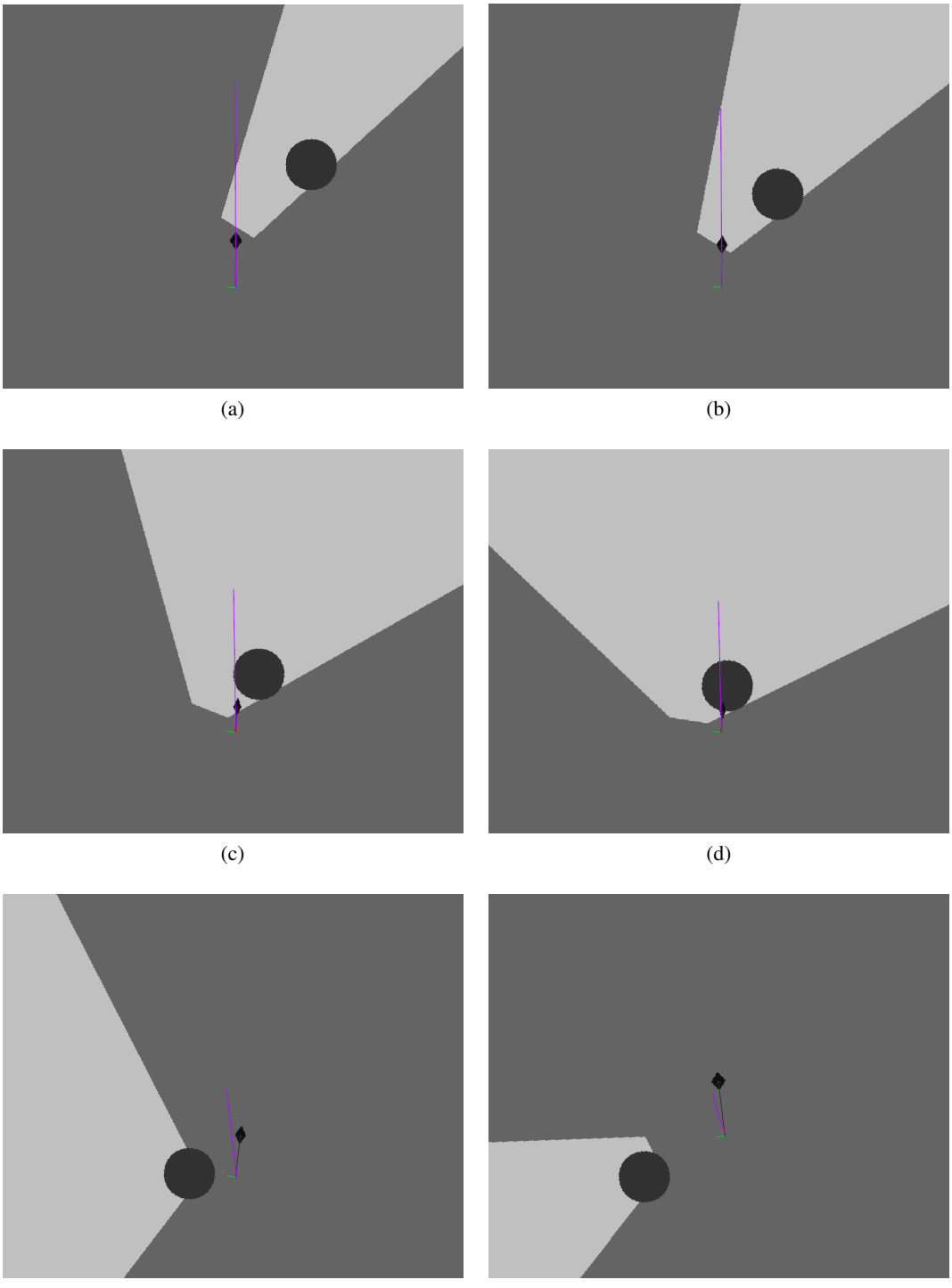
Da mesma forma que no caso de teste anterior, primeiramente, inicializou-se um cenário contendo o mesmo ambiente de simulação e um ASV. De seguida uma outra embarcação foi adicionada de modo a recriar uma situação de colisão entre dois veículos, como é possível verificar na figura 5.9. Esta embarcação descolar-se-á com velocidade constante e igual a $0.2\,\mathrm{m\,s^{-1}}$ segundo o eixo dos *xx* (eixo vermelho), intercetando a trajetória do ASV a ser controlado. O objetivo continua a ser o mesmo, ou seja, deslocar o ASV para o ponto (5,0).

(a) Caso de teste com um obstáculo dinâmico.

(b) Representação do ambiente obtida para o caso de teste com um obstáculo dinâmico.

Figura 5.9: Figura representativa do caso de teste com um obstáculo dinâmico e a sua representação do ambiente.

Utilizando o algoritmo HORIZON obteve-se a representação do ambiente e trajetória do planeador global, representadas na figura 5.9b, no instante de tempo t_2 da figura 5.11a.

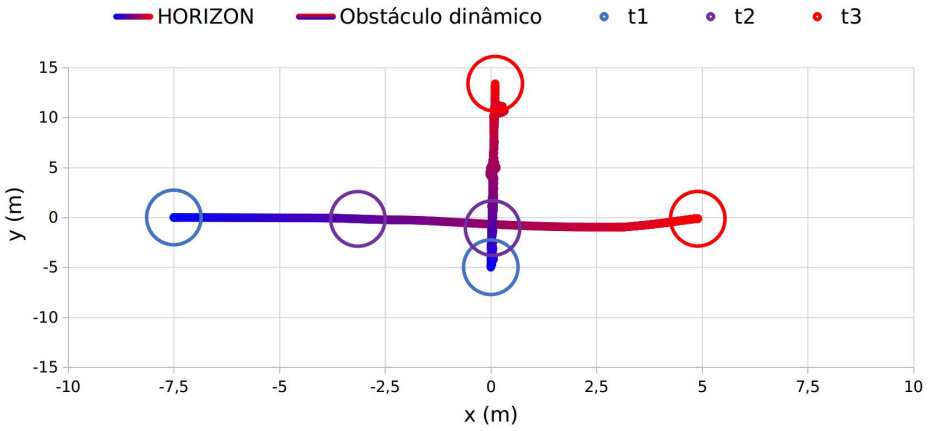


(a) (b) (c) (d) (e) (f)

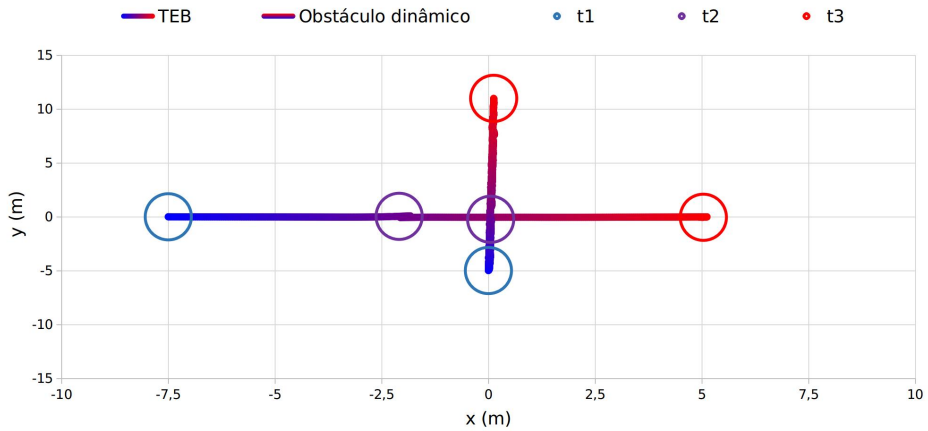
Figura 5.10: Evolução temporal da representação do ambiente e da trajetória no desvio a um obstáculo dinâmico.

A sequência de imagens da figura 5.10 tem como objetivo exemplificar a sucessão de passos que o algoritmo segue, desta vez, de forma a evitar a colisão com o obstáculo dinâmico e a dirigir-se para o ponto de destino. Em primeiro lugar o algoritmo começa por deslocar-se em linha reta, segundo a trajetória planeada (figura 5.10a). De seguida, à medida que se aproxima do obstáculo, o cone de colisão aproxima-se também da zona das velocidades alcançáveis, até que estes dois conjuntos se intercetam, como é possível observar na figura 5.10b. Desta forma, o algoritmo vai então selecionando velocidades cada vez menores de forma a evitar que a zona de velocidades alcançáveis se sobreponha ao cone de colisão. Atentando às figuras 5.10c e 5.10d denota-se então um abrandamento do ASV a ser controlado pois a zona de velocidades a negro encontra-se cada vez mais próxima do referencial do veículo. Após a passagem do obstáculo dinâmico o veículo segue então a sua rota normalmente, figuras 5.10e e 5.10f.

Assim como no teste anterior, obtiveram-se resultados utilizando o algoritmo HORIZON e o método TEB.

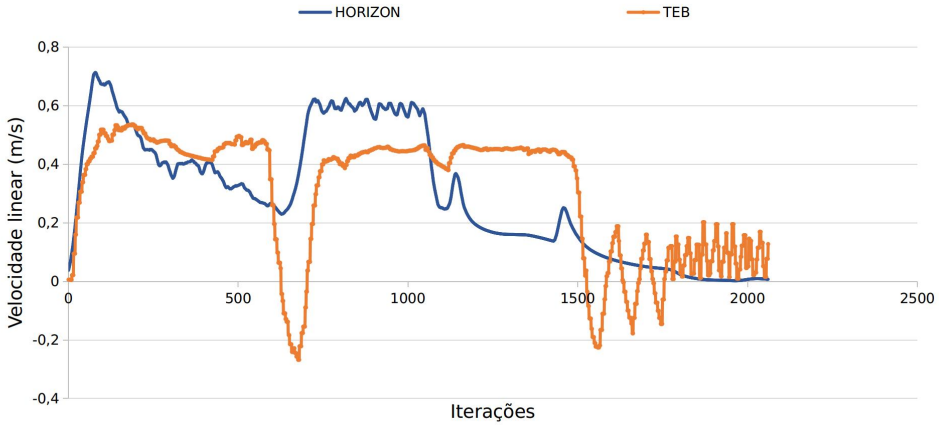


(a) Trajetórias executadas pelo ASV controlado com o algoritmo HORIZON (esquerda para a direita) e pelo obstáculo dinâmico (baixo para cima).

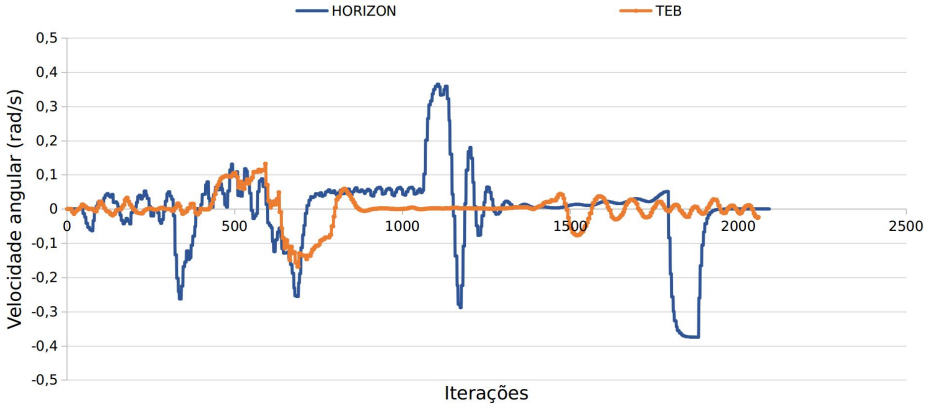


(b) Trajetórias executadas pelo ASV controlado com o algoritmo TEB (esquerda para a direita) e pelo obstáculo dinâmico (baixo para cima).

Figura 5.11: Trajetórias executadas pelos dois algoritmos em resposta a um obstáculo dinâmico.



(a) Velocidades lineares apresentadas pelos algoritmos HORIZON (azul) e TEB (laranja).



(b) Velocidades angulares apresentadas pelos algoritmos HORIZON (azul) e TEB (laranja).

Figura 5.12: Gráfico das velocidades lineares e angulares no comportamento apresentado com um obstáculo dinâmico.

Desta forma, na figura 5.11 estão representas as trajetórias executadas e na figura 5.12 as velocidades obtidas com estes dois algoritmos. Na primeira foi utilizado um esquema de cores semelhante ao do planeamento de rotas de modo a possibilitar uma melhor compreensão dos instantes temporais. Assim, as trajetórias começam no instante t_1 a azul e acabam no instante t_3 a vermelho. O instante t_2 representa o instante de tempo em que o veículo apresentou a primeira reação em função ao obstáculo dinâmico.

Como é possível observar pela análise dos gráficos da figura 5.11, o algoritmo implementado apresenta uma primeira reação ao obstáculo num instante temporal t_2 que é aproximadamente 6 segundos antes do instante temporal em que se inicia o procedimento de desvio calculado pelo TEB. Assim pode-se concluir que este apresenta assim uma capacidade preditiva maior. Além disso, nesse mesmo instante, verifica-se que o segundo método tentou ultrapassar o obstáculo

dinâmico pela sua frente, ou seja deslocar-se para a esquerda, e só depois percebeu que tal não seria possível alterando a sua trajetória, recuando, de forma a deixar passar o obstáculo. Com recurso ao gráfico das velocidades lineares 5.12a confirma-se esta situação ao observar que a velocidade linear utilizando o algoritmo TEB atingiu valores negativos, ou seja, o ASV deslocou-se para trás.

Em relação ao algoritmo HORIZON verifica-se que este apresenta uma redução da velocidade linear em função da proximidade ao obstáculo. Assim, esta capacidade de adaptação da velocidade permite que o ASV continue a sua trajetória sem que seja necessário parar ou deslocar-se para trás, reduzindo assim o número de manobras necessárias e evitando situações potencialmente perigosas.

Quanto às velocidades angulares observa-se que o método desenvolvido apresenta um comportamento de modo a evitar a colisão com o obstáculo. Sabendo a sua velocidade atual e a velocidade do veículo detetado o algoritmo determina que a melhor trajetória é por trás deste, realizando assim um pequeno desvio da sua trajetória ideal, não sendo assim necessário diminuir em totalidade a sua velocidade linear.

5.4 Resultados obtidos ao executar uma missão

De forma a apresentar um exemplo completo da aplicação do algoritmo Horizon procedeu-se à execução de uma missão anteriormente planeada por este. Assim, tendo em consideração o conjunto de pontos e obstáculos anteriormente selecionados, figura 5.2, foi executada a rota planeada utilizando como valores de função custo $w_1 = 0.5$ e $w_2 = 0.5$. A rota resultante do planeamento está uma vez mais explicita na figura 5.13.

De modo a retirar conclusões acerca das vantagens do algoritmo implementado, esta mesma rota foi executada, não só com o algoritmo HORIZON, mas também utilizando o algoritmo TEB, resultando nas trajetórias presentes na figura 5.14. Neste gráfico encontram-se três rotas distintas: a verde a rota planeada e estimada pela fase de planeamento do algoritmo HORIZON como sendo a rota ideal a executar, a laranja a rota executada com recurso a esse mesmo algoritmo e a azul a rota executada pelo método TEB.

Ao comparar as diferentes rotas facilmente se verifica que a execução da missão com o algoritmo HORIZON resultou numa rota mais próxima da planeada. Observa-se também que, por duas vezes, a rota executada com recurso a este método origina um desvio pelo lado oposto do obstáculo ao planeado. Este comportamento resulta da dinâmica provocada pela maré e capacidade de adaptação do algoritmo de delinear uma nova trajetória melhorada tendo em consideração este fator. Quanto à rota executada pelo algoritmo TEB verifica-se que esta, em certos instantes de tempo, encontra-se bastante deslocada da estimada. Este comportamento deve-se ao facto de o algoritmo não realizar uma aproximação suave a cada ponto de destino, resultando em largas curvas devido ao seu momento de inércia.

Tendo em consideração que a distância total da rota planeada pelo algoritmo desenvolvido é de $100.27m$, é possível observar, na tabela 5.5, a distância total percorrida pelo ASV utilizando quer o método HORIZON, quer o método TEB e os seus respetivos erros.

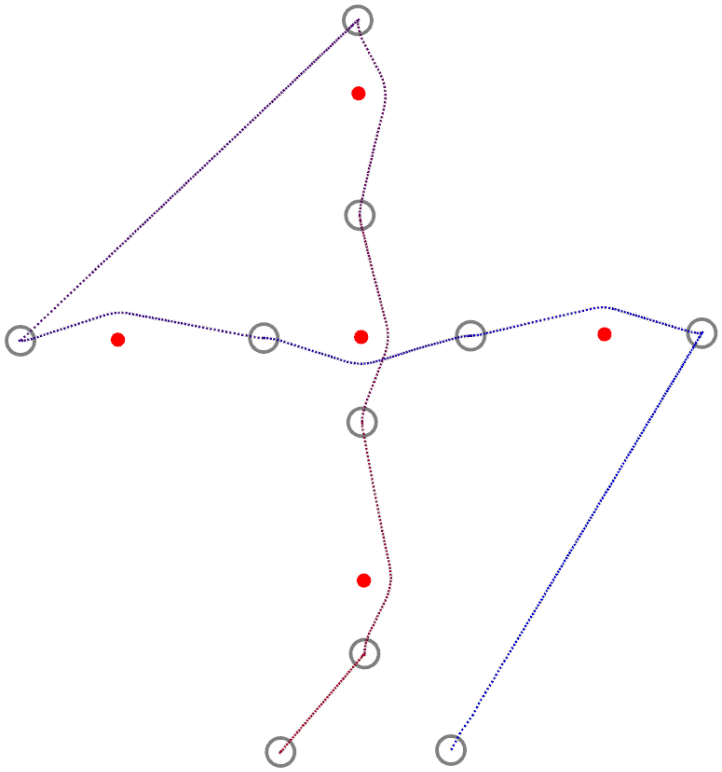


Figura 5.13: Rota planeada com recurso ao método HORIZON e com valores de função de custo $w_1 = 0.5$ e $w_2 = 0.5$.

Tabela 5.5: Tabela comparativa das distâncias percorridas pelo algoritmo HORIZON e pelo algoritmo TEB.

	HORIZON	TEB
Distância (m)	**101.3**	119.92
Erro (m)	**1.03**	19.65
Erro (%)	**1.02**	19.59

Através da análise da tabela conclui-se que ao se executar a missão com recurso ao algoritmo HORIZON obtém-se uma distância de rota 1.02% maior à planeada, enquanto que utilizando o método TEB obtém-se uma rota 19.59% maior que a esperada. Denota-se então uma enorme vantagem ao utilizar este novo método como algoritmo de execução de trajetória e desvio de obstáculos, visto este ter apresentado uma diminuição de 18.57% à distância total da rota face à executada pelo segundo algoritmo.

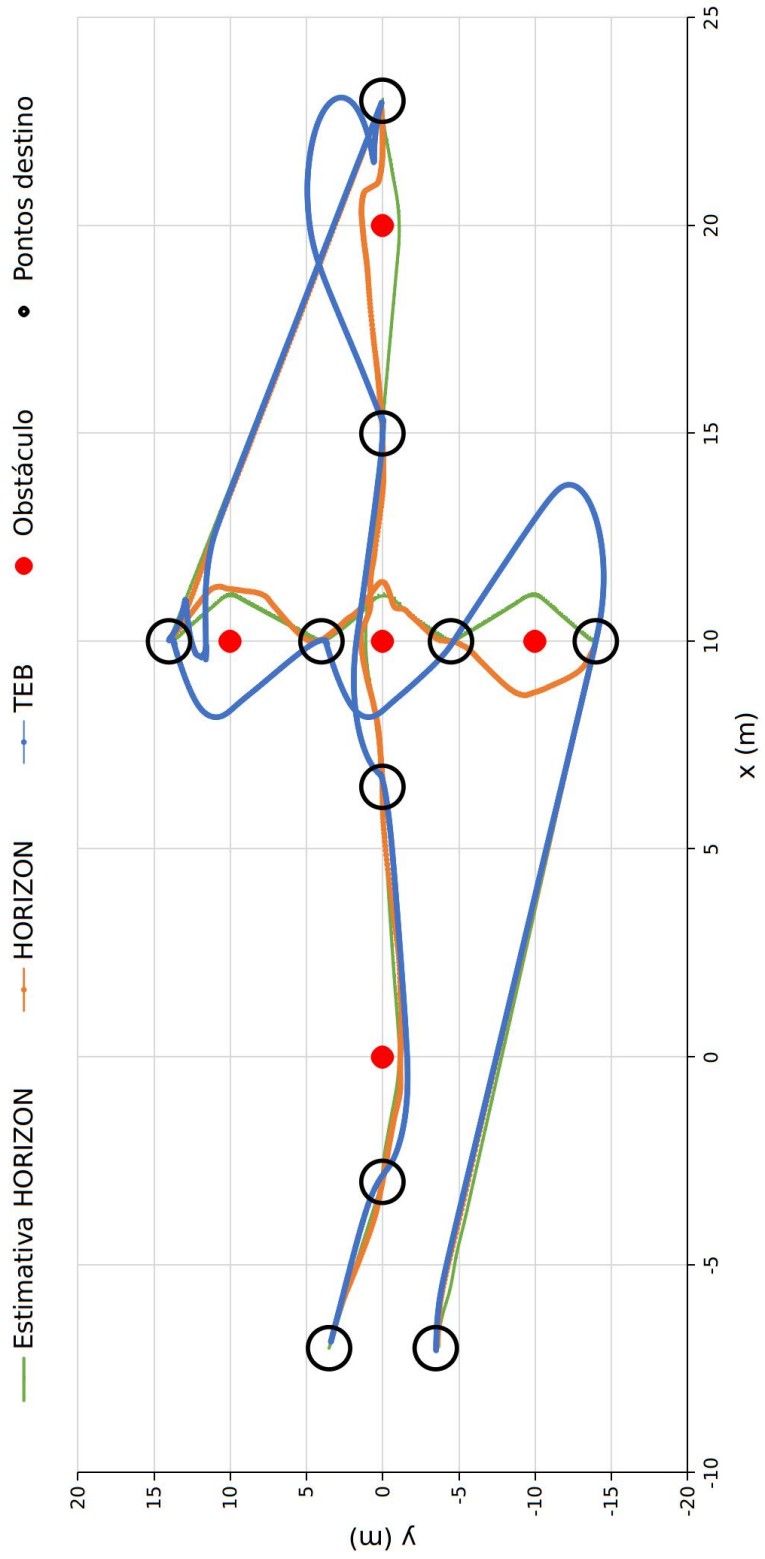


Figura 5.14: Trajetória planeada (verde) e executada (laranja) com recurso ao algoritmo HORIZON e comparação com a executada (azul) pelo método TEB.

Capítulo 6

Conclusões e Trabalho Futuro

Nesta dissertação foi abordada a problemática do planeamento de rotas e trajetórias para embarcações autónomas, recorrendo a um simulador 3D realista de um cenário com diversos obstáculos (estáticos e dinâmicos), tendo em atenção a dinâmica do ambiente marítimo para recriar o *setup* experimental. Por forma a apresentar as metodologias utilizadas foram expostas nos capítulos iniciais as descrições dos algoritmos implementados e os processos em que se baseiam, sendo no fim demonstrados os resultados obtidos para cada um dos métodos.

Tal como referido no capítulo 1 o objetivo principal desta dissertação é implementar um algoritmo que permite realizar o planeamento e execução de rotas num ambiente conhecido e dinâmico, evitando colisões e obter um comportamento preditivo e reativo de forma a realizar caminhos ótimos e seguros.

Para esse fim, foram implementados diversos processos e metodologias. Primeiramente foi desenvolvido um algoritmo capaz de solucionar um problema do tipo TSP e de planear diferentes rotas de acordo com a definição dos pesos de uma função de custo, tendo em consideração todos os obstáculos estáticos conhecidos. Este algoritmo permite também caracterizar e descrever qualquer tipo de missão indicando apenas os pontos que se deseja visitar, sem que seja necessário ter em consideração questões de otimização em relação à ordem com que são inseridos. Para isso, foi desenvolvida uma interface em RVIZ capaz de interagir com o utilizador permitindo a seleção dos pontos desejados.

Através dos testes realizados foi possível observar que para o conjunto de dez pontos distribuídos no cenário experimental, composto por cinco obstáculos estáticos, obteve-se uma rota 6.79% mais curta e com uma variação total angular 2.18% menor, quando comparado com o resultado obtido através do algoritmo Dijkstra.

De seguida foi desenvolvido um algoritmo de execução de rotas planeadas, de uma forma segura e eficiente, evitando colisões com obstáculos estáticos e dinâmicos. O método desenvolvido faz recurso a um planeador global que permite a estimação de uma velocidade livre de colisão através da obtenção de uma trajetória, tendo em consideração os obstáculos estáticos, e de uma representação espacial do ambiente em torno da embarcação considerando os obstáculos estáticos e dinâmicos.

Pela análise dos comportamentos apresentados ao desvio de obstáculos estáticos e dinâmicos verifica-se que o algoritmo foi capaz de executar as trajetórias pretendidas apresentando zero colisões, uma velocidade dinâmica e uma capacidade preditiva superior, comprovada pelo facto de o algoritmo proposto apresentar uma primeira reação ao obstáculo num instante temporal, aproximadamente 6 segundos antes do instante temporal em que se inicia o procedimento de desvio calculado pelo TEB. Esta última propriedade permitiu reduzir assim o número de manobras necessárias para evitar situações potencialmente perigosas.

Por fim, é de referir que ao longo deste projeto para além dos algoritmos para atacar a problemática, foi desenvolvido um outro algoritmo capaz de interligar as duas componentes de planeamento e execução, tendo sido obtidos resultados que demonstram a viabilidade do recurso a este método. A experiência realizada da execução de uma missão permite concluir que o método desenvolvido foi capaz de planear e executar a rota, passando por todos os pontos, evitando colisões, e obter uma distância total percorrida apenas 1.02% superior à estimada.

6.1 Trabalho Futuro

Com vista a dar continuidade ao trabalho desenvolvido ao longo da dissertação e testar os algoritmos em meios alternativos são sugeridos como trabalhos futuros os seguintes pontos:

- Desenvolvimento de um algoritmo que entre com o modelo da embarcação (por exemplo, o tipo de locomoção, a inércia e ainda, as dimensões) para a definição do valor dos pesos da função custo;
- Criação de um modelo de embarcação para ambiente marítimo tendo consideração as restrições cinemáticas e dinâmicas impostas ao veículo;
- Estudo de uma heurística de seleção de nova velocidade considerando, por exemplo, tempo de missão;
- Testar com ambiente de simulação com condições marítimas extremas;
- Testar em ambiente real.

Referências

[1] Allianz Global Corporate & Specialty. Safety and Shipping Review 2015. (December 2014):36, 2015. URL: http://www.agcs.allianz.com/assets/PDFs/Reports/Shipping-Review-2015.pdf, doi:10.1109/TSTE.2010.2050348.

[2] J.-C. Latombe. Motion Planning: A Journey of Robots, Molecules, Digital Actors, and Other Artifacts. *International Journal of Robotics Research*, 18(11):1119–1128, 1999.

[3] Renato Silva, Pedro Leite, Daniel Campos, e Andry M Pinto. Hybrid Approach to Estimate a Collision-Free Velocity for Autonomous Surface Vehicles. *IEEE International Conference on Autonomous Robot Systems and Competitions*, 2019.

[4] Pedro Leite, Renato Silva, Aníbal Matos, e Andry Maykol Pinto. An Hierarchical Architecture for Docking Autonomous Surface Vehicles. *IEEE International Conference on Autonomous Robot Systems and Competitions*, 2019.

[5] Joseph Curcio, John Leonard, e Andrew Patrikalakis. SCOUT - A low cost autonomous surface platform for research in cooperative autonomy. *Proceedings of MTS/IEEE OCEANS*, 2005. doi:10.1109/OCEANS.2005.1639838.

[6] Hordur K. Heidarsson e Gaurav S. Sukhatme. Obstacle detection and avoidance for an autonomous surface vehicle using a profiling sonar. *Proceedings - IEEE International Conference on Robotics and Automation*, 2011. doi:10.1109/ICRA.2011.5980509.

[7] C Jacques, Maurice F Fallon, J John, Jacques C Leedekerken, e John J Leonard. Mapping Complex Marine Environments with Autonomous Surface Craft. 2013.

[8] Pedro Luís Cerqueira Gomes da Costa. *Planeamento cooperativo de tarefas e trajectórias em múltiplos robôs*. Tese de doutoramento, 2011. URL: http://hdl.handle.net/10216/62107.

[9] Daniel Campos. Planeamento Simultâneo de Trajetórias para Múltiplos Robôs Autónomos num Ambiente Industrial. Tese de mestrado, 2014.

[10] Yong K. Hwang e Narendra Ahuja. Gross motion planning—a survey. *ACM Computing Surveys*, 24(3):219–291, 1992. URL: http://portal.acm.org/citation.cfm?doid=136035.136037, doi:10.1145/136035.136037.

[11] A. A. Stepanov Vladimir J Lumelsky. Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE Transactions on Automatic Control*, 31(11):1058–1063, 1986. doi:10.1109/TAC.1986.1104175.

[12] J. Antich, A. Ortiz, e J. Mínguez. A bug-inspired algorithm for efficient anytime path planning. Em *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, páginas 5407–5413, Oct 2009. doi:10.1109/IROS.2009.5354182.

[13] Javier Antich, Alberto Ortiz, e M Javier. A Bug -Inspired Algorithm for Efficient Anytime Path Planning.

[14] Ishay Kamon, Elon Rimon, e Ehud Rivlin. TangentBug: A Range-Sensor-Based Navigation Algorithm. *The International Journal of Robotics Research*, 17(9):934–953, 1998. URL: https://doi.org/10.1177/027836499801700903, doi:10.1177/027836499801700903.

[15] I. Kamon e E. Rivlin. Sensory-based motion planning with global proofs. *IEEE Transactions on Robotics and Automation*, 13(6):814–822, Dec 1997. doi:10.1109/70.650160.

[16] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.

[17] Ron Wein, Jur P. Van Den Berg, e Dan Halperin. The visibility-Voronoi complex and its applications. *Computational Geometry: Theory and Applications*, 36(1):66–87, 2007. doi:10.1016/j.comgeo.2005.11.007.

[18] D R Parhi e Alok Kumar Jha. Review and Analysis of Different Methodologies Used in mobile robot navigation. 4(1):1–18, 2012.

[19] Y Koren, Senior Member, J Borenstein, e Ann Arbor. Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation. 1991.

[20] J. Borenstein e Y. Koren. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, June 1991. doi:10.1109/70.88137.

[21] S. Quinlan e O. Khatib. Elastic bands: connecting path planning and control. Em *[1993] Proceedings IEEE International Conference on Robotics and Automation*, páginas 802–807 vol.2, May 1993. doi:10.1109/ROBOT.1993.291936.

[22] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, e T. Bertram. Efficient trajectory optimization using a sparse model. Em *2013 European Conference on Mobile Robots*, páginas 138–143, Sep. 2013. doi:10.1109/ECMR.2013.6698833.

[23] C. Rösmann, F. Hoffmann, e T. Bertram. Planning of multiple robot trajectories in distinctive topologies. Em *2015 European Conference on Mobile Robots (ECMR)*, páginas 1–6, Sep. 2015. doi:10.1109/ECMR.2015.7324179.

[24] Christoph Rösmann, Frank Hoffmann, e Torsten Bertram. Integrated online trajectory planning and optimization in distinctive topologies. *Robot. Auton. Syst.*, 88:142–153, Fevereiro 2017. URL: https://doi.org/10.1016/j.robot.2016.11.007, doi:10.1016/j.robot.2016.11.007.

[25] C. Roesmann, W. Feiten, T. Woesch, F. Hoffmann, e T. Bertram. Trajectory modification considering dynamic constraints of autonomous robots. Em *ROBOTIK 2012; 7th German Conference on Robotics*, páginas 1–6, May 2012.

[26] D K M Kufoalor, E F Brekke, e T A Johansen. Proactive Collision Avoidance for ASVs using A Dynamic Reciprocal Velocity Obstacles Method. páginas 2402–2409, 2018.

[27] P. Fiorini e Z. Shiller. Motion planning in dynamic environments using the relative velocity paradigm. *[1993] Proceedings IEEE International Conference on Robotics and Automation*. URL: http://ieeexplore.ieee.org/document/292038/, doi:10.1109/ROBOT.1993.292038.

[28] Petr Švec, Brual C. Shah, Ivan R. Bertaska, Wilhelm Klinger, Armando J. Sinisterra, Karl Von Ellenrieder, Manhar Dhanak, e Satyandra K. Gupta. Adaptive Sampling Based COLREGs-Compliant Obstacle Avoidance for Autonomous Surface Vehicles. *IEEE*, 2014.

[29] Jur van den Berg, Ming Lin, e Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. páginas 1928–1935, 05 2008. doi:10.1109/ROBOT.2008.4543489.

[30] Jur Van Den Berg, Stephen J. Guy, Ming Lin, e Dinesh Manocha. Reciprocal n-body collision avoidance. *Springer Tracts in Advanced Robotics*, 70(STAR):3–19, 2011. doi:10.1007/978-3-642-19457-3_1.

[31] Mohannad Al-Khatib e Jean J. Saade. An efficient data-driven fuzzy approach to the motion planning problem of a mobile robot. *Fuzzy Sets and Systems*, 134:65–82, 2003. doi:10.1016/S0165-0114(02)00230-0.

[32] Iraj Hassanzadeh, Hamid Ghadiri, e R. Dalayimilan. Design and implemention of a simple fuzzy algorithm for obstacle avoidance navigation of a mobile robot in dynamic environment. *2008 5th International Symposium on Mechatronics and Its Applications*, páginas 1–6, 2008.

[33] M. Meng e A. C. Kak. Neuro-nav: a neural network based architecture for vision-guided mobile robot navigation using non-metrical models of the environment. Em *[1993] Proceedings IEEE International Conference on Robotics and Automation*, páginas 750–757 vol.2, May 1993. doi:10.1109/ROBOT.1993.291944.

[34] F Miyazaki, S Arimoto, M Takegaki, e Y Maeda. Sensory Feedback Based on the Artificial Potential for Robot Manipulators. *IFAC Proceedings Volumes*, 17(2):2381–2386, 1984. URL: http://dx.doi.org/10.1016/S1474-6670(17)61338-7, doi:10.1016/S1474-6670(17)61338-7.

[35] Jianping Tu e S. X. Yang. Genetic algorithm based path planning for a mobile robot. Em *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 1, páginas 1221–1226 vol.1, Sep. 2003. doi:10.1109/ROBOT.2003.1241759.

[36] K. H. Sedighi, K. Ashenayi, T. W. Manikas, R. L. Wainwright, e Heng-Ming Tai. Autonomous local path planning for a mobile robot using a genetic algorithm. Em *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 2, páginas 1338–1345 Vol.2, June 2004. doi:10.1109/CEC.2004.1331052.

[37] Yee Zi Cong e Ponnambalam S.G. Mobile Robot Path Planning using Ant Colony Optimization. páginas 851–856, 2009. doi:10.1109/AIM.2009.5229903.

[38] Guan-zheng Tan, Huan He, e Sloman Aaron. Global optimal path planning for mobile robot based on improved Dijkstra algorithm and ant system algorithm. *Journal of Central South University of Technology*, 13:80–86, 2006. doi:10.1007/s11771-006-0111-8.

[39] M.K. Habib e H. Asama. Efficient method to generate collision free paths for an autonomous mobile robot based on new free space structuring approach. *Proceedings IROS '91:IEEE/RSJ International Workshop on Intelligent Robots and Systems '91*, (December):563–567, 1991. URL: http://ieeexplore.ieee.org/document/174534/, doi:10.1109/IROS.1991.174534.

[40] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984.

[41] Maxim Likhachev, Geoffrey J. Gordon, e Sebastian Thrun. Ara*: Anytime a* with provable bounds on sub-optimality. Em *NIPS*, 2003.

[42] P P Chakrabarti, Sujoy Ghose, Arup Acharya, e S C de Sarkar. Heuristic Search in Restricted Memory. *Artificial Intelligence*, 41:197–221, 1989. doi:10.1016/0004-3702(89)90010-6.

[43] Rong Zhou e Eric A. Hansen. Memory-bounded a* graph search. páginas 203–209, 01 2002.

[44] Dimitris (MIT) Bertsimas e John (MIT) Tsitsiklis. Simulated annealing, 1993.

[45] Nuno Alexandre Cruz, Aníbal Matos, Sérgio Cunha, e Sérgio Silva. Zarco - An Autonomous Craft for Underwater Surveys. Em *Proceedings of the 7th Geomatic Week*, 2007.

[46] Ana Rita Gaspar, Alexandra Nunes, Andry Maykol Pinto, e Aníbal Matos. Urban@CRAS dataset: Benchmarking of visual odometry and SLAM techniques. *Robotics and Autonomous Systems*, 109:59–67, 2018. doi:10.1016/j.robot.2018.08.004.

[47] A. Figueroa, E. Montero, e M. Riff. An effective simulated annealing for off-line robot motion planning. Em *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*, páginas 967–971, Nov 2017. doi:10.1109/ICTAI.2017.00148.

[48] S. Hayat e Z. Kausar. Mobile robot path planning for circular shaped obstacles using simulated annealing. Em *2015 International Conference on Control, Automation and Robotics*, páginas 69–73, May 2015. doi:10.1109/ICCAR.2015.7166004.