UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

FILIPE TESTA DAROS

PLANEJAMENTO DE TRAJETÓRIA ROBÓTICA BASEADO EM RELEVO OCEÂNICO

CURITIBA

2022

FILIPE TESTA DAROS

PLANEJAMENTO DE TRAJETÓRIA ROBÓTICA BASEADO EM RELEVO OCEÂNICO

Robotic path planning based on ocean relief

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial (CPGEI) da Universidade Tecnológica Federal do Paraná como requisito para obtenção do título de Mestre em Ciências.

Orientador: Prof. Dr. Andre Schneider de Oliveira

Coorientador: Prof. Dr. Marco Antônio Simões Teixeira

CURITIBA

2022

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Campus Curitiba

FILIPE TESTA DAROS

PLANEJAMENTO DE TRAJETÓRIA ROBÓTICA BASEADO EM RELEVO OCEÂNICO

Trabalho de pesquisa de mestrado apresentado como requisito para obtenção do título de Mestre Em Ciências da Universidade Tecnológica Federal do Paraná (UTFPR). Área de concentração: Engenharia De Computação.

Data de aprovação: 01 de Dezembro de 2022

Dr. Marco Antonio Simoes Teixeira, Doutorado - Universidade Tecnológica Federal do Paraná

Dr. Ronnier Frates Rohrich, Doutorado - Universidade Tecnológica Federal do Paraná

Dra. Vivian Cremer Kalempa, Doutorado - Fundação Universidade do Estado de Santa Catarina (Udesc)

Documento gerado pelo Sistema Acadêmico da UTFPR a partir dos dados da Ata de Defesa em 01/12/2022.

Dedico este trabalho à minha família,
especialmente ao meu filho, pelos momentos
de ausência.

AGRADECIMENTOS

Agradeço à minha família pelo apoio durante todo o caminho. Agradeço ao meu orientador, André Schneider de Oliveira, e ao meu co-orientador, Marco Antônio Simões Teixeira, pela paciência, orientação e parceria. Agradeço ao programa e à universidade, pela oportunidade que me foi dada.

RESUMO

A separação de pedidos é um dos processos que tem impulsionado o aumento no número de pesquisas na área de logística. A robótica pode ajudar a reduzir o custo operacional de tal processo, um dos maiores do armazém, fazendo que o operador humano deixe de se deslocar para efetuar a coleta dos produtos, e sim que estes sejam movidos de suas posições até a estação de montagem do pedido com o auxílio de robôs. Dentre as técnicas de maior destaque aplicadas no cálculo de trajetórias desses agentes robóticos pode-se citar os campos potenciais e o algoritmo A*. Esse trabalho visa demonstrar uma nova abordagem, baseada no comportamento do relevo oceânico, para mapeamento de um ambiente que simula um armazém logístico e compará-la com as rotas calculadas a partir dos algoritmos de maior destaque, levando em consideração a distância, segurança e eficiência da trajetória proposta. Primeiramente, é apresentado o estado da arte e uma revisão teórica dos principais algoritmos envolvidos. Na sequência, os componentes do sistema proposto (ROS, RabbitMQ, CoppeliaSim) são introduzidos e seu uso justificado. Após isto, o algoritmo proposto é detalhado e a metodologia de comparação com A* e Campos potenciais é apresentada. Por fim, os resultados obtidos com a abordagem proposta são apresentados, mostrando como a técnica proposta obtém bom índice de segurança e precisão nas rotas calculadas, mesmo submetido à imprecisão do sensoriamento, além de mitigar alguns dos problemas dos algoritmos de referência citados.

Palavras-chave: robotica; relevo oceânico; mapeamento; a*; campos potenciais.

ABSTRACT

Order picking is one of the processes that has motivated the research increase in the field of logistics. Robotics can help reduce the operational cost of such a process, one of the biggest in the warehouse, in a way that the human operator can avoid moving around the warehouse to collect the products, and instead these products are moved from their storage positions to the static order assembly station with robots help. Among the most prominent techniques applied in robotics path planning, we can mention the potential fields and the A* algorithm. This paper aims to demonstrate a new approach, based on the behavior of ocean relief, for mapping an environment that simulates a logistics warehouse and comparing it with the paths calculated from the most frequently used algorithms, taking into account the distance, security, and efficiency of the proposed trajectory. At first, a theoretical review of the main algorithms involved is presented. Next, the components of the system (ROS, RabbitMQ, Coppelia) are introduced and their use is justified. After that, the proposed algorithm is detailed and the methodology of comparison with A* and Potential Fields is detailed. Finally, the results are presented, showing good numbers of safety and precision in the calculated routes, even when subjected to sensor imprecision, in addition to mitigating some of the problems of reference algorithms cited.

Keywords: robotics; ocean relief; mapping; a*; potential fields.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTAGEM DE CÓDIGOS FONTE

LISTA DE ABREVIATURAS E SIGLAS

Siglas

AFSA	Artificial Fish Swarm Algorithm (Algoritmo de Cardume de Peixes Aritificial)
CPU	Central Processing Unit (Unidade Central de Processsamento)
LIDAR	Light Detection and Ranging (Detecção e Alcance de Luz)
RAM	Random Access Memory (Memória de Acesso Randômico)
ROS	Robot Operating System (Sistema Operacional Robótico)

SUMÁRIO

1 INTRODUÇÃO

A redução de custos no processamento de mercadorias em armazéns de grandes centros de distribuição se tornou um fator importante para aumentar a competitividade das companhias no mercado atual. O valor gasto enquanto o produto está em triagem, até a entrega para o cliente, não agrega valor para o negócio, e por esse motivo, tem se tornado alvo de inúmeras pesquisas que visam à redução no tempo de trânsito e o aumento da eficiência operacional.

Em um armazém ou centro de distribuição, um dos fatores mais importantes no que diz respeito ao corte de custos é a redução do tempo gasto pelo profissional que monta os pedidos no deslocamento entre a área de coleta da mercadoria e as de montagem e conferência do pedido final. Conforme visto em Shiau e Liao (2013), além de crucial, esse é o processo mais oneroso do armazém. Para reduzir essa conta, uma das abordagens é manter o produto sempre o mais próximo possível do local de montagem dos pedidos, evitando o deslocamento em áreas que podem possuir o tamanho de vários campos de futebol. Apesar de evidente, essa é uma tarefa que tem demandado investimentos enormes e apresentado inúmeros desafios, especialmente para a robótica Parker (2009), mas também para outras áreas relacionadas Wang, Mcintosh e Brain (2010).

Uma das abordagens para reduzir a distância entre o produto e a área de montagem dos pedidos é inverter a lógica atual da movimentação, fazendo que as mercadorias, e não o operador, passem a se deslocar pelo armazém. Isso é possível através de agentes robóticos programados especificamente para essa tarefa. Eles recebem pedidos vindos de outros sistemas, geralmente um *Warehouse Management System* (Sistema de Gerenciamento de Armazéns - WMS), coletam os produtos necessários e levam até uma estação de empacotamento, onde o operador humano faz a montagem da embalagem final.

1.1 Motivação

A automação do processo de coleta e entrega das mercadorias dentro do armazém envolve inúmeros desafios. Como principais, podemos citar a construção do mapa do ambiente, determinação da posição atual do robô, desvio de obstáculos, planejamento de uma trajetória executável e preferentemente ótima, entre outros, conforme visto em Lavalle (2006). Além de todos esses pontos, acrescenta-se que tudo ocorre em um ambiente altamente dinâmico, com movimentação constante de pessoas e maquinários.

Além de atender os requisitos impostos pela própria operação de armazenagem, existe também a necessidade de integração com os inúmeros sistemas auxiliares necessários para o controle de todo fluxo de trabalho do armazém. Tais requisitos demandam não somente um algoritmo capaz de gerar um mapa e traçar uma trajetória, mas também um sistema que seja flexível o bastante para alterar seu comportamento de acordo com as necessidades que se apresentem.

A principal motivação desse trabalho é demonstrar como as características do relevo de fundo oceânico podem ser utilizadas para modelar um mapa que possa ser utilizado pelos agentes robóticos ao navegar em um armazém logístico, satisfazendo as restrições de domínio da área.

Esse trabalho se concentra na modelagem de um mapa a partir do sensoriamento do ambiente, e sua utilização para cálculo de trajetórias entre pontos distintos de um armazém simulado, e posterior apresentação dos resultados obtidos pela abordagem proposta quando comparada a dois métodos largamente utilizados, A* Duchon *et al.* (2014) e navegação por campos potenciais Sabudin, Omar e Melor (2016).

1.2 Objetivos

Nessa seção serão apresentados o objetivo geral do trabalho bem como os objetivos específicos, que visam flexibilizar o sistema desenvolvido para projetos futuros.

1.2.1 Objetivo geral

Desenvolver um algoritmo que, baseado na dinâmica do relevo de fundo oceânico, seja capaz de gerar uma rota de um ponto de origem até um destino aleatório, dentro de um armazém simulado, mapeado utilizando um sensor do tipo *Light Detection and Ranging* (Detecção e Alcance de Luz) LIDAR.

1.2.2 Objetivos específicos

1. Manter uma margem segura de distância entre o agente robótico e as estruturas do armazém. Isso é importante para evitar acidentes que possam danificar o robô ou a carga sendo transportada.

2. Ter uma tolerância a falhas de sensoriamento maior que o algoritmo A* tradicional, uma vez que sensores podem encontrar dificuldades em detectar com sucesso todas as estruturas dentro do armazém.

3. Obter trajetórias que não causem o bloqueio do agente robótico devido a ocorrência de mínimos locais, problema que ocorre principalmente em cenários com corredores em formato de 'U', quando da utilização da técnica de campos potenciais.

4. Gerar rotas com uma quantidade menor de curvas. Isso simplifica a operação do robô, economizando bateria e consequentemente estendendo o tempo de operação de cada unidade robótica no armazém.

5. Empregar o uso de um barramento de dados, tornando a solução flexível para alterações e trabalhos futuros. Além de aumentar a flexibilidade da solução, o uso do barramento de dados potencializa a capacidade do algoritmo de se integrar com outras plataformas.

6. Obter tempo de processamento da rota compatível com a operação de um armazém. Algoritmos que demandam muito tempo para processarem uma saída podem se tornar inviáveis em uma operação altamente dinâmica como a encontrada em armazéns logísticos.

1.3 Justificativa

Demonstrar como algoritmos amplamente conhecidos se comportam quando utilizados para resolver problemas de domínio específico, com suas peculiaridades e desafios. Além de desenvolver um sistema baseado em uma heurística própria e compará-lo com tais algoritmos, este trabalho busca também integrar ferramentas de uso cotidiano dentro da esfera de pesquisas relacionadas à robótica, como o *Robot Operating System* (Sistema Operacional Robótico) ROS, com plataformas utilizadas no mercado, como a linguagem C# e o barramento de mensagens *RabbitMQ*, tornando a solução implementada facilmente integrável com outros sistemas, bem como flexível o bastante para o desenvolvimento de trabalhos futuros.

1.4 Estrutura do trabalho

Esta dissertação foi dividida nos seguintes capítulos:

1. Capítulo 1: Introdução

2. Capítulo 2: Estado da arte e referencial teórico contendo uma revisão dos algoritmos A*, Campos potenciais e uma breve introdução sobre o relevo de fundo oceânico.

3. Capítulo 3: Apresenta o algoritmo proposto e as ferramentas utilizadas no desenvolvimento da proposta.

4. Capítulo 4: Resultados obtidos com o algoritmo proposto e comparação com A* tradicional e Campos potenciais.

5. Capítulo 5: Conclusão e trabalhos futuros.

2 REFERENCIAL TEÓRICO

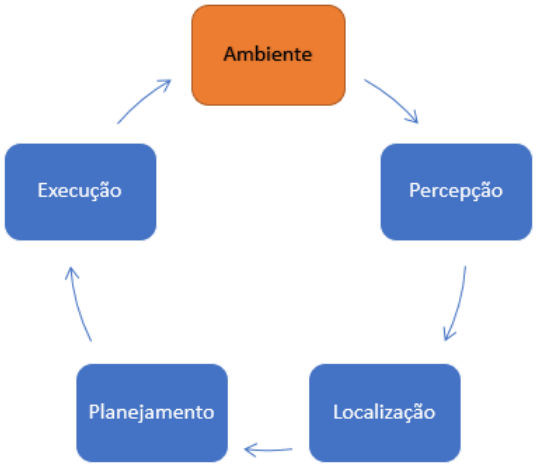
Nesse capítulo são apresentados uma revisão do estado da arte em mapeamento e navegação robótica, os aspectos teóricos dos algoritmos utilizados para o desenvolvimento desse estudo, e por fim uma breve introdução sobre algumas características do relevo oceânico.

2.1 Estado da arte

O mapeamento e a navegação autônoma são os principais atributos de um agente robótico que se proponha a automatizar determinada tarefa. A navegação robótica, conforme visto na Figura 1, é dividida em:

1. Percepção: Extração de informações importantes do ambiente através do sensoriamento.
2. Localização: Localizar o agente em relação ao ambiente em que está operando.
3. Planejamento: Calcular e planejar uma rota de um ponto de origem até um destino, evitando obstáculos.
4. Execução: Controle dos motores para movimentação do agente e execução da rota planejada.

Figura 1 – Passos envolvidos no processo de navegação robótica



Fonte: Adaptado de Mac2016.

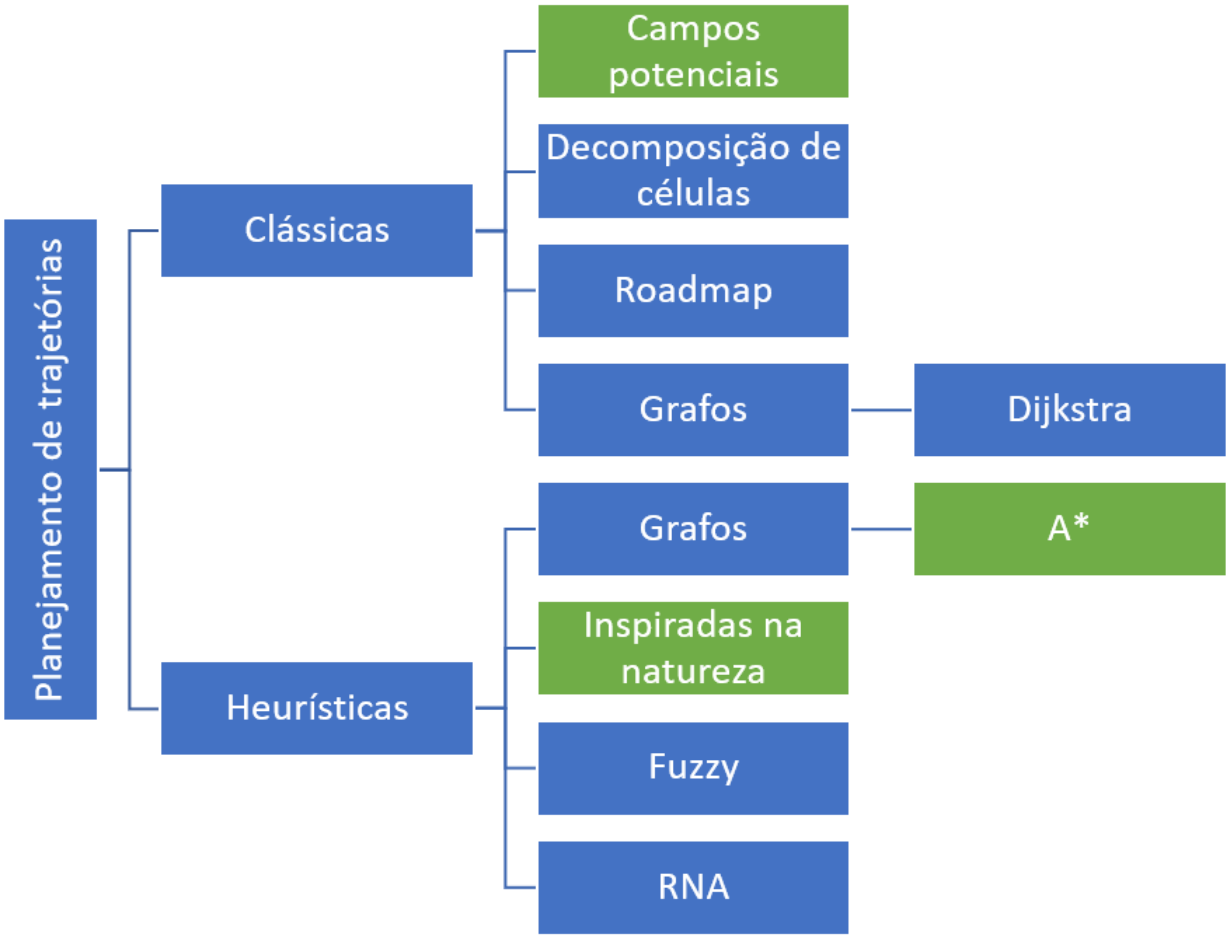
Além disso, conforme visto em Mac2016, existem dois tipos de planejamento de trajetória, e normalmente um conjunto dessas técnicas é utilizado para alcançar um bom resultado:

1. Global: Quando todo o ambiente é conhecido e o agente pode criar um mapa de baixa resolução da trajetória necessária para atingir o alvo. Geralmente atingem trajetórias ótimas, mas não se adaptam bem a ambientes altamente dinâmicos ou desconhecidos.

2. Local: Não necessita de informações prévias do ambiente e geralmente entrega uma trajetória de alta resolução, porém é ineficiente quando a rota a ser percorrida é excessivamente longa.

É possível encontrar na literatura uma grande quantidade de técnicas de planejamento de trajetória. Embora todas se disponham a resolver o mesmo problema, ou seja, levar o agente robótico do ponto de origem até o ponto de destino, cada uma possui uma gama de características distintas, e pode se comportar melhor ou pior dependendo do cenário proposto. Normalmente elas são caracterizadas, em um nível mais alto, em clássicas e baseadas em heurísticas. Essa classificação varia de autor para autor, conforme visto em Mac *et al.* (2016), Koubaa *et al.* (2018), mas pode-se ver alguns exemplos na Figura 2, onde as caixas em verde são de algoritmos que tem relação com o presente trabalho.

Figura 2 – Classificação dos algoritmos de planejamento de trajetória.

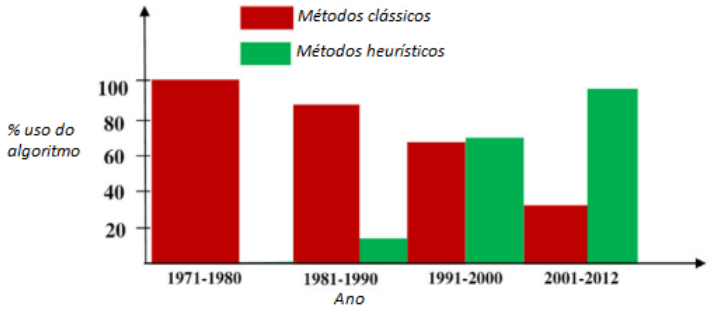


Fonte: Autoria própria (2023).

Em relação a sua utilização em pesquisas científicas, enquanto as abordagens clássicas dominaram a literatura no passado, é possível notar uma tendência maior no aparecimento de técnicas heurísticas a partir de meados dos anos 2000, conforme visto na Figura 3. Isso se

deve principalmente ao fato de algoritmos heurísticos terem um melhor desempenho quando submetidos ao teste de mínimos locais. Isso não significa que os algoritmos clássicos estejam caindo em desuso, e inúmeras pesquisas continuam evoluindo esses métodos. Porém métodos heurísticos tem mostrado um bom desempenho, especialmente em situações dinâmicas e com sensores mais simples, e por isso tem alavancado um grande número de trabalhos.

Figura 3 – Tendência de uso de algoritmos clássicos e heurísticos ao longo dos anos.



Fonte: Mac *et al.* (2016) .

Em Pradhan *et al.* (2006) é apresentada uma solução baseada em campos potenciais artificiais capaz de guiar um agente robótico de um ponto de inicio até um destino estático pré-determinado. Para mitigar o problema de mínimos locais presentes naturalmente nesse algoritmo, foi utilizada uma função de campo repulsivo que leva em consideração a distância da posição atual do robô até o destino, conforme visto na Figura 4;

A pesquisa de Pradhan *et al.* (2006) apresenta também o conceito de inúmeros robôs operando com o mesmo campo potencial, o que é uma grande vantagem do método, uma vez que o mapeamento do campo depende exclusivamente das posições dos obstáculos e do destino, desconsiderando a posição atual dos agentes. Os autores demonstram também um processo para viabilizar tal navegação, baseado em redes de Petri Peterson (1977), que pode ser visto em detalhes na Figura 5.

Os autores conseguiram resultados onde os alvos são atingidos mesmo sob condições normalmente associadas com mínimos locais, como corredores em formato de 'U'. Tais resultados podem ser vistos nas Figura 6, onde a operação de 15 agentes simultâneos é validada. Porém como também pode ser notado, as rotas geradas pelo campo não estão otimizadas, e apesar de atingirem com sucesso o objetivo proposto, dependeriam de melhorias para serem empregadas em uma situação no mundo real.

Na pesquisa desenvolvida em Zhang *et al.* (2013), é proposta uma abordagem para utilização de campos potenciais artificiais capaz de melhorar as rotas de agentes robóticos para ambientes dinâmicos, onde ocorre a predominância de obstáculos em movimento. Esse tipo de ambiente é comumente encontrado em pequisas que utilizam o cenário de futebol de robôs, mas pode ser facilmente transportada para cenários de uso industrial, como linhas de montagem e armazéns logísticos. O principal objetivo da pesquisa, além do cálculo do campo para objetos em movimento, foi reduzir a quantidade de desvios desnecessários de obstáculos, introduzindo

Figura 4 – Funções potenciais para mitigar problema de mínimos locais.

$$U_{\text{rep}}(\text{obs}_1) = \begin{cases} \frac{1}{2}\alpha_1 \left(\frac{1}{\rho(q, q_{\text{obs}_1})} - \frac{1}{\rho_0} \right)^2 \rho^n (q, q_{\text{target}}) & \text{if } \rho(q, q_{\text{obs}_1}) \leq \rho_0 \\ 0 & \text{if } \rho(q, q_{\text{obs}_1}) > \rho_0 \end{cases}$$

$$U_{\text{rep}}(\text{obs}_2) = \begin{cases} \frac{1}{2}\alpha_2 \left(\frac{1}{\rho(q, q_{\text{obs}_2})} - \frac{1}{\rho_0} \right)^2 \rho^n (q, q_{\text{target}}) & \text{if } \rho(q, q_{\text{obs}_2}) \leq \rho_0 \\ 0 & \text{if } \rho(q, q_{\text{obs}_2}) > \rho_0 \end{cases}$$

$$U_{\text{rep}}(\text{obs}_3) = \begin{cases} \frac{1}{2}\alpha_3 \left(\frac{1}{\rho(q, q_{\text{obs}_3})} - \frac{1}{\rho_0} \right)^2 \rho^n (q, q_{\text{target}}) & \text{if } \rho(q, q_{\text{obs}_3}) \leq \rho_0 \\ 0 & \text{if } \rho(q, q_{\text{obs}_3}) > \rho_0 \end{cases}$$

Fonte: Pradhan *et al.* (2006) .

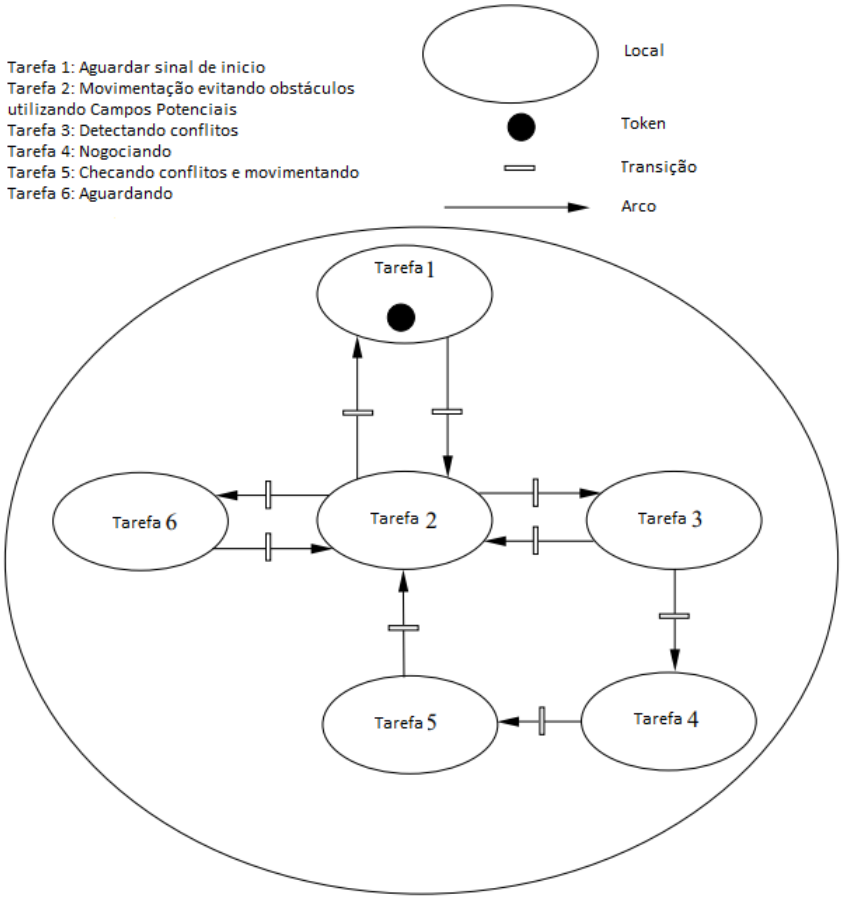
na função de cálculo do campo, além das posições do robô e dos obstáculos, a velocidade relativa e ângulo de colisão entre eles, conforme visto na Figura 7, onde pode-se notar:

1. R: Posição do robô.
2. Eixo y: Direção do robô.
3. O: Obstáculo.
4. V: Velocidade relativa do obstáculo em relação ao robô.
5. d_m: Distância mínima para desvio do obstáculo.

A partir dessas informações, os autores propuseram a fórmula para cálculo do campo potencial conforme Figura 8, onde $U_{rep}(P,V)$ representa o campo potencial repulsivo gerado pelo obstáculo; p_o é uma constante que define a área de influência do obstáculo; d_m é a soma da área do obstáculo, do robô, e da distância mínima de segurança entre os dois; e por fim $d_g = p_s(P, P_{goal})$ representa a distância do robô para o destino.

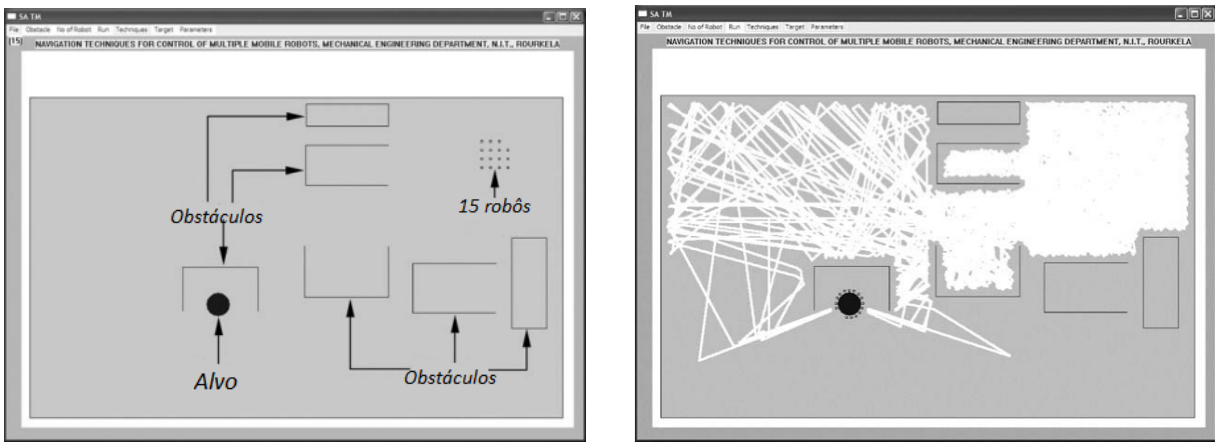
Os resultados obtidos pelos autores em Zhang *et al.* (2013) foram positivos, e em suas simulações os agentes robóticos foram capazes de desviar com segurança tanto de obstáculos estáticos quanto dinâmicos, inclusive sendo capazes de atingir destinos também em movimento,

Figura 5 – Modelagem de rede de Petri para operar sistema multi-agente.



Fonte: Pradhan *et al.* (2006) .

Figura 6 – Cenário proposto (esq) e resultados obtidos (dir)

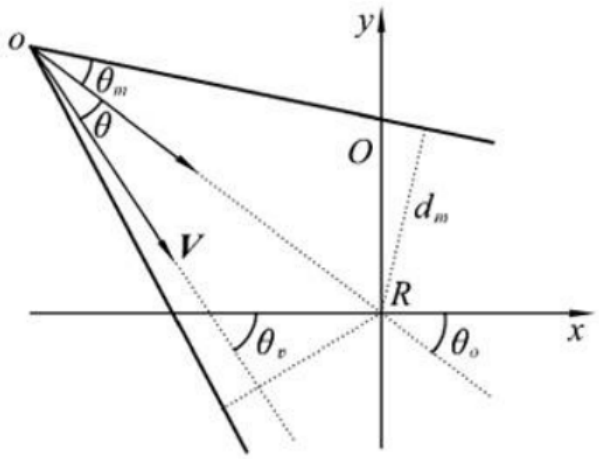


Fonte: Pradhan *et al.* (2006) .

como pode ser visto na Figura 9, onde R representa o robô, O1, O2, O3 e O4 representam os obstáculos e G representa o alvo, cada um com seus respectivos vetores de velocidade e coordenadas iniciais.

As pesquisas apresentadas em Pradhan *et al.* (2006) e Zhang *et al.* (2013) demonstram que, apesar da tendência de utilização de algoritmos heurísticos após meados de 2000, con-

Figura 7 – Proposta para considerar velocidade e ângulo de colisão do agente robótico e do obstáculo



Fonte: Zhang *et al.* (2013) .

Figura 8 – Fórmula de campo repulsivo proposta

$$U_{rep}(\mathbf{P},\mathbf{V}) = \begin{cases} not \quad defined \quad , & \rho_s(P,P_{obs}) - d_m < 0 \\ \varepsilon(e^{(\theta_m-\theta)}-1)(\dfrac{1}{\rho_s(P,P_{obs})-d_m} - \dfrac{1}{\rho_0})\rho^2{}_s(P,P_{goal})/2, & |\theta| < \theta_m \wedge 0 < \rho_s(P,P_{obs}) - d_m < \rho_0 \\ 0 \quad , & \text{other wise} \end{cases}$$

Fonte: Zhang *et al.* (2013) .

forme visto em Mac *et al.* (2016), estudos ainda expandem a capacidade de métodos clássicos, especialmente de campos potenciais, dada a sua elegância e eficiência matemática.

Conforme mencionado, a eficiência matemática dos métodos clássicos, especialmente o de campos potenciais, é um grande ponto positivo desses algoritmos. Isso viabiliza tomadas rápidas de decisão, pois como resultado do baixo custo de processamento computacional necessário, o programa pode emitir um resultado mais prontamente.

Em Wang *et al.* (2019) os autores utilizaram um campo potencial artificial para demonstrar a capacidade de um veículo autônomo efetuar manobras de desvio de obstáculos, tanto estáticos quanto dinâmicos, como visto anteriormente em Zhang *et al.* (2013). Porém além dos resultados simulados, a pesquisa também foi testada em um cenário no mundo real.

Pesquisas validadas em cenários do mundo real, especialmente as que dependem diretamente de leitura de sensores dos agentes robóticos, são especialmente importantes pois demonstram a capacidade do algoritmo proposto de lidar com recebimento de dados com baixa precisão ou até mesmo incorretos.

No trabalho apresentado em Wang *et al.* (2019), os autores utilizaram uma função gravitacional para construção do campo potencial. O destino da rota gera um campo gravitacional, que aumenta em intensidade a medida que o agente se aproxima de seu destino. Os obstácu-

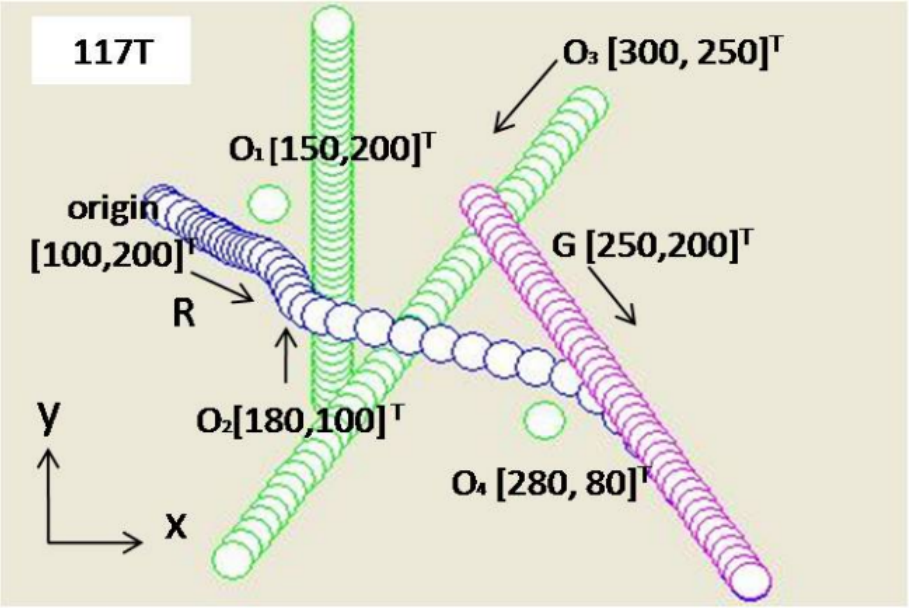
los no caminho geram campos repulsivos, conforme visto em pesquisas anteriores Zhang *et al.* (2013), Pradhan *et al.* (2006).

Ainda em Wang *et al.* (2019) os autores apresentam uma proposta para que o veículo seja capaz de escapar de mínimos locais. Por se tratar de um algoritmo desenvolvido para atuar em carros e ônibus percorrendo uma rodovia, existe a situação em que o veículo estará alinhado com o alvo e com eventuais obstáculos, uma vez que todos podem estar na mesma faixa de rodagem. Nesse cenário, a força resultante da ação dos campos repulsivos e atrativo pode ser zero, causando uma parada do veículo autônomo. Para resolver esse problema, os autores adicionaram destinos virtuais, fora da área de ação dos campos, para que o veículo consiga sair do mínimo local.

Os experimentos conduzidos no trabalho proposto por Wang *et al.* (2019) foram satisfatórios, inclusive no sentido de tornar a manobra para desvio do obstáculo confortável para passageiros humanos, uma vez que padrões de condução veicular foram analisados e imitados. Após executar os testes com sucesso em ambiente simulado, dois cenários no mundo real foram conduzidos, conforme Figura 10. Em um deles, o veículo desvia de um obstáculo estático, enquanto se move a uma velocidade de 20 km/h. No segundo teste, o veículo desvia de outro veículo em movimento que está trafegando em velocidade menor, de 10km/h.

Os resultados apresentados pelos autores em Wang *et al.* (2019) foram positivos dada a proposta apresentada, conforme visto na Figura 11. Nela pode-se notar as 3 linhas que representam as rotas dos experimentos realizados, no primeiro cenário conduzindo o desvio de um obstáculo estático, e no segundo mostrando o desvio de um obstáculo em movimento. Por fim os autores mencionam que o teste, apesar de empregar veículos reais, foi realizado em um ambiente controlado e em condições ideais, visto que o experimento é conduzido em uma es-

Figura 9 – Resultados obtidos para cálculo de campos potencias em ambientes dinâmicos



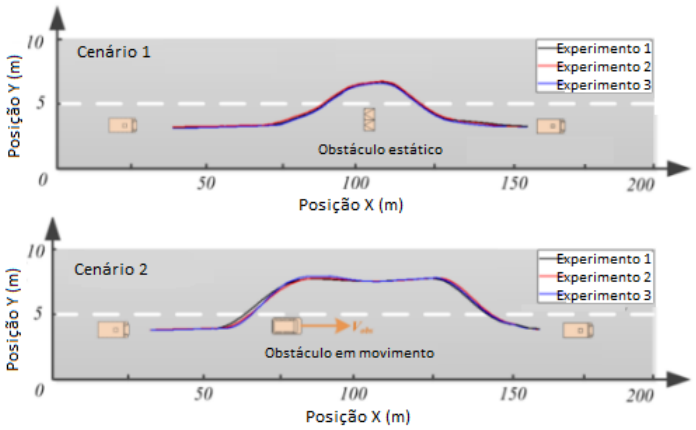
Fonte: Zhang *et al.* (2013) .

Figura 10 – Experimentos conduzidos no mundo real

Fonte: Wang *et al.* (2019) .

trada com boa pavimentação e largura, e a velocidade do veículo que atuou como obstáculo é constante, algo improvável de ocorrer. Portanto, como objeto de pesquisa futura, é mencionado que simulações que exponham o algoritmo a situações mais adversas devem ser conduzidas, visando expandir a capacidade do modelo proposto.

Figura 11 – Resultados dos experimentos



Fonte: Wang *et al.* (2019) .

Outro trabalho que utiliza campos potenciais artificiais, desta vez aplicado especificamente para o cenário logístico, pode ser visto em Tingbin *et al.* (2018). Neste trabalho, os autores se preocuparam em garantir que não somente uma rota entre a origem e o destino fosse calculada, mas também que essa rota considerasse alguns fatores que implicam em redução de custos para um armazém, como por exemplo consumo energético do agente robótico.

De modo a otimizar a rota calculada, uma técnica baseada no comportamento de cardumes de peixe é inserida no algoritmo. Esta técnica foi originalmente proposta em 2002 sob o nome de *Artificial Fish Swarm Algorithm* (Algoritmo de Cardume de Peixes Aritificial) AFSA, e é um algoritmo de otimização de busca estocástica, que se baseia no comportamento dos peixes de explorarem um universo de busca para encontrar abundância de alimentos. Esse algoritmo

vem sendo aplicado desde então em diversas vertentes, inclusive em áreas industriais, como pode ser visto em Zainal, Zain e Sharif (2015).

Os resultados apresentados em Tingbin *et al.* (2018), demonstram como a utilização de métodos bio-inspirados em conjunto com algoritmos clássicos pode trazer melhoras significativas, particularmente em cenários com restrições de uso de energia, segurança etc, como por exemplo a navegação dentro de um armazém logístico.

Como visto nas pesquisas apresentadas, é recorrente o problema de mínimos locais em algoritmos que utilizam campos potenciais artificiais para cálculo de trajetórias. A maioria dos autores se preocupa em prever em seus algoritmos mecanismos que façam com que o agente robótico seja capaz de sair dessas situações, continuando sua operação. No entanto, somente escapar de um mínimo local nem sempre é eficiente em termos de geração de rotas mais curtas. O ideal seria que rotas que possuam mínimos locais não fossem enviadas para os robôs.

Em Szczepanski, Bereit e Tarczewski (2021) os autores propõe um método capaz de prever mínimos locais que serão gerados ao longo da rota, aumentando a eficiência operacional do robô. Como mencionado pelos autores, o processo de evadir de um mínimo local consiste em 3 fases: (i) entrar em um mínimo local, (ii) estagnação do movimento do robô e (iii) algoritmo para escape do mínimo local. O método visto no trabalho propõe o processo em dois passos: (i) predição de onde estará um mínimo local e (ii) algoritmo para evitar que o robô chegue até o mínimo local.

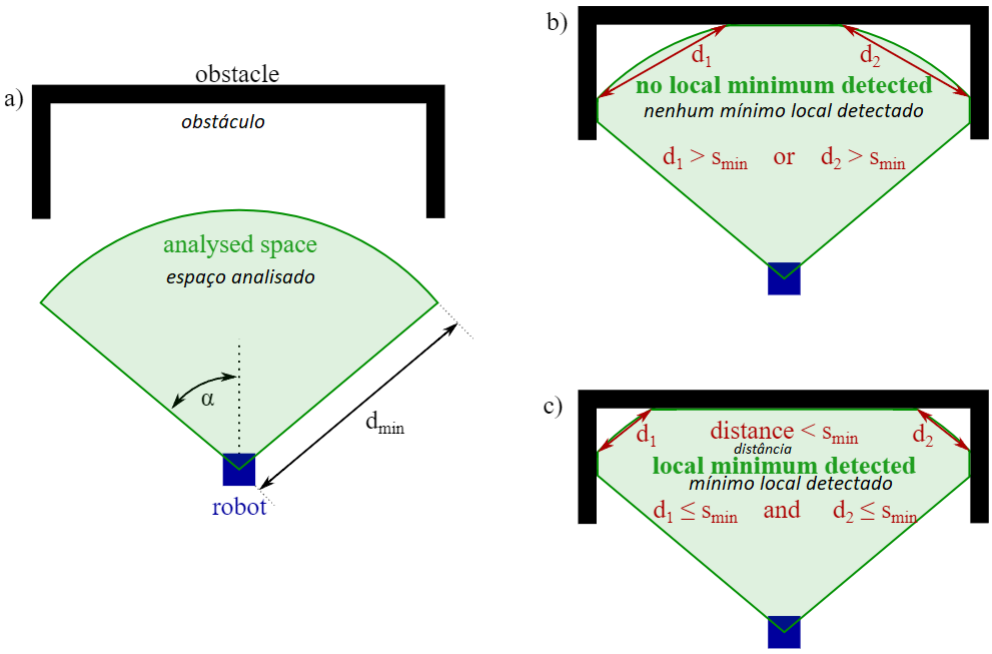
Os autores em Szczepanski, Bereit e Tarczewski (2021) usaram uma técnica para detectar sempre que o agente robótico estiver se encaminhando para um mínimo local. Essa técnica se baseia nos dados enviados pelo sensor LIDAR, e seu funcionamento pode ser visto em detalhes na Figura 12, que mostra em quadros o momento que o robô se aproxima de um cenário clássico de mínimo local (paredes em formato de ’U’).

Após detectar, através dos dados de sensoriamento, a ocorrência iminente de um mínimo local, o algoritmo proposto cria então uma parede virtual, visando alterar o campo potencial e gerar uma rota que evite a área que iria apresentar estagnação de movimento. Para redirecionar o agente robótico, um obstáculo virtual é então colocado à direita ou à esquerda do mesmo, de modo a gerar um campo repulsivo e "empurrar"o robô para a nova direção desejada. Esse processo pode ser visto na Figura 13, que selecionou o lado direito como mais interessante para a sequência da trajetória, colocando então um obstáculo virtual à esquerda.

Os autores apresentaram resultados em que o agente robótico conseguiu evitar com sucesso a estagnação de movimento entre paredes em forma de ’U’ e também em forma de arco. Esses resultados foram obtidos através de testes em laboratório, em um cenário estático e controlado, utilizando uma plataforma ROSbot 2.0 PRO. As rotas obtidas pelo sistema, bem como a comparação com o algoritmo clássico de campos potenciais pode ser visto na Figura 14.

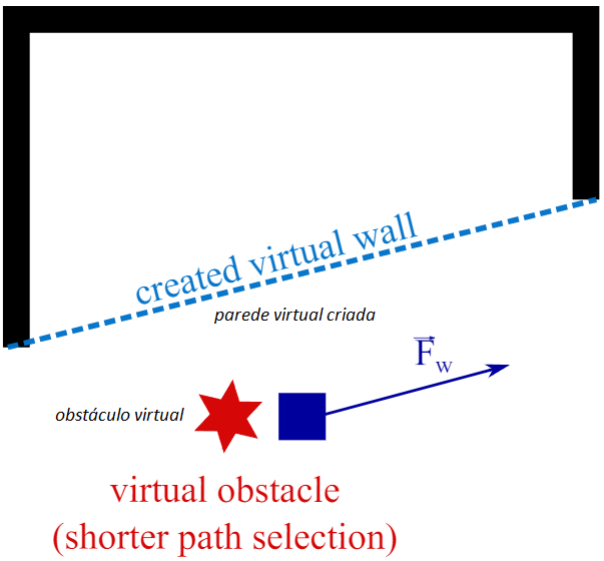
Os autores concluem em Szczepanski, Bereit e Tarczewski (2021) que o algoritmo apresentou bons resultados no que diz respeito a evitar mínimos locais, porém ressaltam que se

Figura 12 – Detecção de mínimo local



Fonte: Szczepanski, Bereit e Tarczewski (2021) .

Figura 13 – Exemplo de parede e obstáculo virtual criados para evitar mínimo local

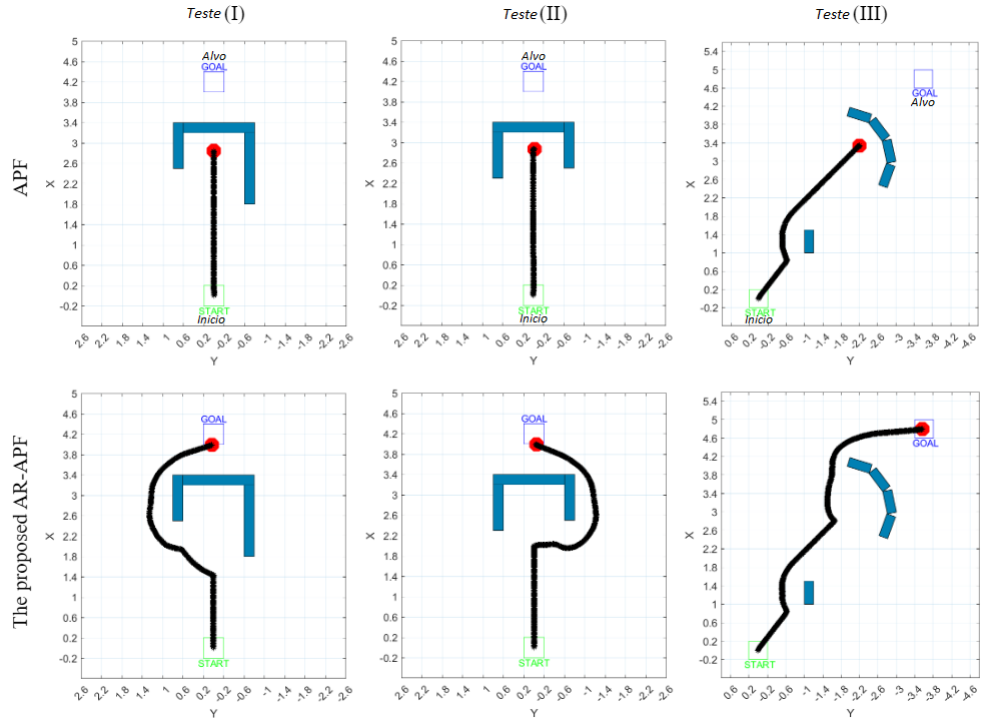


Fonte: Szczepanski, Bereit e Tarczewski (2021) .

trata de um algoritmo de planejamento de trajetória local, e que o sistema pode não ser capaz de detectar com antecedência todos os cenários, como por exemplo corredores mais longos em formato de “U” e com distância maior entre as paredes. Para melhorar os resultados neste tipo de cenário, os autores pretendem desenvolver novas pesquisas que incluam sistemas multi-agente e compartilhamento de dados.

No campo dos algoritmos de planejamento de trajetória baseados em heurísticas, um dos principais focos de trabalhos científicos nas últimas décadas tem sido abordagens baseadas

Figura 14 – Resultados obtidos

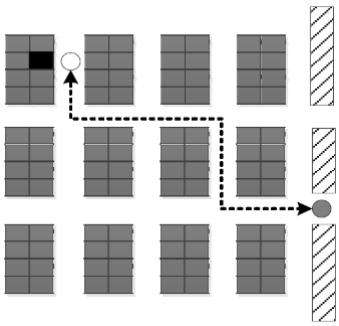


Fonte: Szczepanski, Bereit e Tarczewski (2021) .

no algoritmo A* (leia-se "A estrela", ou "A-star"). Esse algoritmo é uma evolução do algoritmo de Edsger Dijkstra, proposto originalmente em 1959.

Em Duan (2018) o autor utiliza um algoritmo A* modificado para executar a tarefa de coleta de mercadorias em um armazém logístico. O autor adicionou no cálculo da trajetória um fator que aumenta o custo da rota em caso de curvas, o que reduz o custo operacional e simplifica o caminho que deve ser percorrido. Essa diferença pode ser notada na Figura 15, que representa o cálculo com o algoritmo tradicional, e na Figura 16, com o resultado do sistema proposto.

Figura 15 – Rota gerada pelo algoritmo A* tradicional

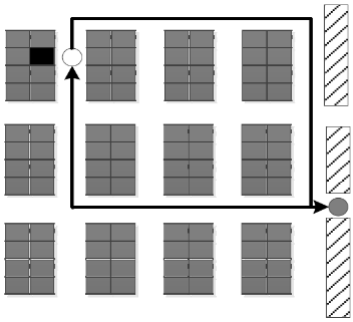


Fonte: Duan (2018) .

Como pode ser visto na Figura 15, o agente robótico teria que executar 3 curvas para sair do ponto de inicio até o destino, um giro de 360 graus, e mais 3 curvas para retornar até a origem. Ao analisar a mesma operação na Figura 16, pode-se notar que o mesmo processo de

coleta custaria ao robô a execução de 1 curva para atingir o ponto de destino e mais 2 curvas para retornar até a origem, além de dispensar o giro de 360 graus.

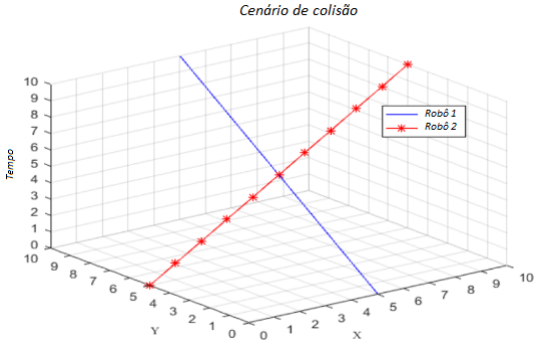
Figura 16 – Rota gerada pelo algoritmo proposto



Fonte: Duan (2018) .

Além das alterações do algoritmo A*, o autor em Duan (2018) também aplica o sistema para geração de trajetórias para múltiplos agentes, cenário comum em operações logísticas. Para isso, é necessário adicionar um mecanismo de resolução de conflitos, que tendem a ocorrer quando a rota de dois agentes se cruza, por exemplo, ao tentar coletar mercadorias em um mesmo corredor, como visto na Figura 17.

Figura 17 – Exemplo de colisão de rotas



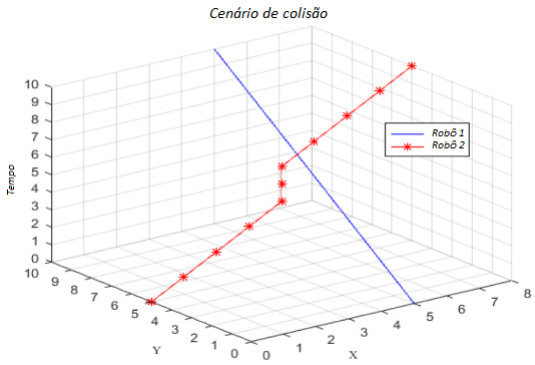
Fonte: Duan (2018) .

Para solucionar esse conflito, adiciona-se para cada rota um nível de importância, de modo a gerar uma hierarquia entre os agentes. O robô com menor nível hierárquico deve aguardar a resolução do conflito antes de prosseguir, como visto na Figura 18.

Dada a natureza do próprio algoritmo A*, é comum encontrar na literatura a sua combinação com outros algoritmos, visando resolver algum problema específico do cenário proposto ou a expansão da capacidade do sistema. Um exemplo de pesquisa com esse foco pode ser encontrada em Xiang *et al.* (2022), onde os autores propõe, além de melhorias de processamento, a combinação com um algoritmo míope para o cenário de operações multi-objetivo.

Conforme citado em Xiang *et al.* (2022) o algoritmo A* normalmente opera expandindo o espaço de busca em 8 direções distintas a partir do ponto atual. A vantagem desse comportamento é a maior adaptabilidade em ambientes complexos, porém isso aumenta o custo computacional do sistema. Os autores então efetuam a troca dessa característica por uma ex-

Figura 18 – Exemplo de resolução de conflito de rotas



Fonte: Duan (2018) .

ploração baseada no ângulo entre o ponto atual e o ponto de destino, conforme cálculo mostrado na Figura 19.

Figura 19 – Cálculo para definição do ângulo de exploração do algoritmo A*, onde (x_0,y_0) é a origem e (x_g,y_g) é o destino

$$\theta = \begin{cases} 90° & y_g > y_0 \cap x_g = x_0 \\ 270° & y_g < y_0 \cap x_g = x_0 \\ \arctan\frac{y_g - y_0}{x_g - x_0} & y_g > y_0 \cup x_g \neq x_0 \\ 180° + \arctan\frac{y_g - y_0}{x_g - x_0} & y_g < y_0 \cup x_g \neq x_0 \end{cases}$$

Fonte: Xiang *et al.* (2022) .

Após o ângulo ser obtido conforme Figura 19, o nó atual é expandido em um setor de 90 graus de ângulo e raio igual a $r = \sqrt{5}$. Se nesse setor não forem detectados obstáculos, o espaço de busca é então alterado de 8 direções para 3 direções, respeitando a Figura 20.

Figura 20 – Exploração do espaço de busca em 3 direções

θ	Manter 3 direções	Abandonar direção
[337. 5°, 360°) ∪ [0°, 22. 5°)	315 T, 000 T, 045 T	090 T, 135 T, 180 T, 225 T, 270 T
[22. 5°, 67. 5°)	000 T, 045 T, 090 T	135 T, 180 T, 225 T, 270 T, 315 T
[67. 5°, 112. 5°)	045 T, 090 T, 135 T	000 T, 180 T, 225 T, 270 T, 315 T
[112. 5°, 157. 5°)	090 T, 135 T, 180 T	045 T, 225 T, 270 T, 315 T, 000 T
[157. 5°, 202. 5°)	135 T, 180 T, 225 T	000 T, 045 T, 090 T, 270 T, 315 T
[202. 5°, 247. 5°)	180 T, 225 T, 270 T	000 T, 045 T, 090 T, 135 T, 315 T
[247. 5°, 292. 5°)	225 T, 270 T, 315 T	045 T, 090 T, 135 T, 180 T, 000 T
[292. 5°, 337. 5°)	270 T, 315 T, 000 T	045 T, 090 T, 135 T, 180 T, 225 T

Fonte: Xiang *et al.* (2022) .

Entretanto, se menos de 4 obstáculos forem detectados no espaço de busca delimitado pelo setor, então o espaço de busca é explorado conforme Figura 21, que exibe um modelo de 5 direções. Por fim, se mais de 4 obstáculos forem detectados no setor, o algoritmo segue o cálculo tradicional e expande em 8 direções.

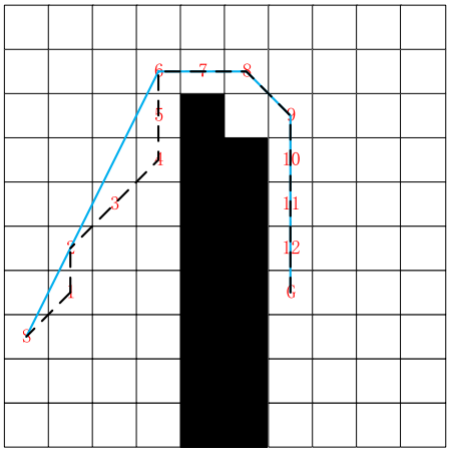
Os autores em Xiang *et al.* (2022) dão sequência ao processamento da rota aplicando uma técnica que visa reduzir o número de nós utilizado pelo algoritmo A*. O principal objetivo dessa etapa é a suavização da trajetória, ou seja, reduzir a quantidade de curvas necessárias para atingir o objetivo, e seu resultado pode ser visto na Figura 22.

Figura 21 – Exploração do espaço de busca em 5 direções

θ	Manter 5 direções	Abandonar direção
[337. 5°, 360°) U [0°, 22. 5°)	000 T, 045 T, 090 T, 270 T, 315 T	135 T, 180 T, 225 T
[22. 5°, 67. 5°)	000 T, 045 T, 090 T, 135 T, 315 T	180 T, 225 T, 270 T
[67. 5°, 112. 5°)	000 T, 045 T, 090 T, 135 T, 180 T	225 T, 270 T, 315 T
[112. 5°, 157. 5°)	045 T, 090 T, 135 T, 180 T, 225 T	270 T, 315 T, 000 T
[157. 5°, 202. 5°)	090 T, 135 T, 180 T, 225 T, 270 T	000 T, 045 T, 315 T
[202. 5°, 247. 5°)	135 T, 180 T, 225 T, 270 T, 315 T	000 T, 045 T, 090 T
[247. 5°, 292. 5°)	180 T, 225 T, 270 T, 315 T, 000 T	045 T, 090 T, 135 T
[292. 5°, 337. 5°)	225 T, 270 T, 315 T, 000 T, 045 T	090 T, 135 T, 180 T

Fonte: Xiang *et al.* (2022) .

Figura 22 – Suavização da rota



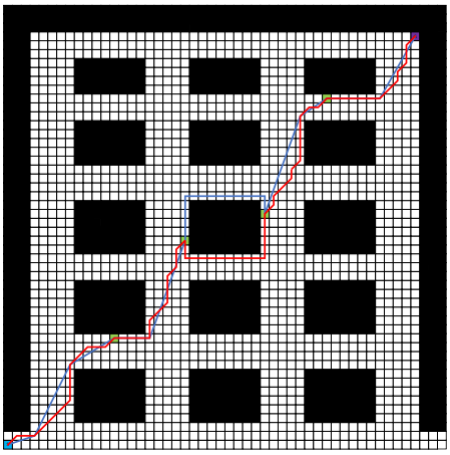
Fonte: Xiang *et al.* (2022) .

Utilizando o algoritmo A* modificado, os autores Xiang *et al.* (2022) obtiveram melhora significativa na distância total da rota, que segundo os mesmos, era a principal métrica avaliada. Além disso, também foram alcançadas melhoras na suavização da trajetória, tempo total de cálculo, número de nós e total de curvas.

Os autores em Xiang *et al.* (2022) finalizam o estudo apresentando uma proposta de uso combinado do algoritmo A* modificado com um algoritmo míope, visando a operação em um cenário multi-objetivo. Conforme mencionado no trabalho, normalmente em cenários como esse o algoritmo A* é executado sucessivas vezes, uma para cada ponto intermediário, até todos os pontos serem cobertos pela trajetória traçada. Essa abordagem é comparada pelos autores com a sua proposta, e os resultados obtidos podem ser visto na Figura 23. Nela pode-se ver a rota calculada pela proposta dos autores, em azul, em contraste com com a rota resultado da soma de várias execuções do A*. Pode-se notar tanto uma melhora em termos de distância percorrida quanto de suavidade da rota obtida em azul.

Outra pesquisa que explora a melhoria das rotas geradas pelo algoritmo A* pode ser visto em Wang *et al.* (2022). Os autores começam explicitando as diferenças entre o algoritmo de Dijkstra, BFS e A*. Essa comparação é bastante pertinente, pois dependendo do peso aplicado às parcelas da equação do A*, pode-se ter um comportamento mais parecido com o BFS ou mais parecido com a proposta de Dijkstra. Onde, no Dijkstra, existe a característica de uma

Figura 23 – Comparação para cenário multi-objetivo entre método proposto, em azul, com abordagem tradicional, em vermelho.

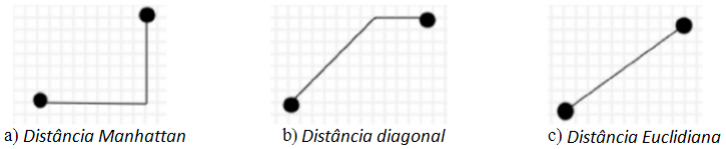


Fonte: Xiang *et al.* (2022) .

rota ótima, ao custo de mais nós no grafo sendo explorados, enquanto no BFS tem-se uma exploração de menos nós, porém não necessariamente uma rota ótima.

Os autores em Wang *et al.* (2022) também comparam as heurísticas mais comuns utilizadas no A* para estimar a distância do ponto atual até o destino, como visto na Figura 24. Como mencionado pelos autores, a escolha pela utilização do método de diagonal se deu pelos melhores resultados obtidos. Segundo os autores, a qualidade das rotas obtidas com a distância de Manhattan não é satisfatória e a distância Euclidiana, apesar de gerar boas trajetórias, apresenta maior custo computacional e complexidade.

Figura 24 – Métodos de estimativa de distância até o destino utilizados no algoritmo A*.



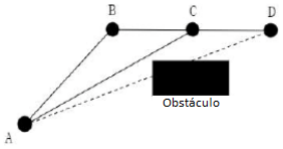
Fonte: Wang *et al.* (2022) .

Para melhorar os resultados obtidos pelo algoritmo A*, é adicionada uma nova parcela na equação do algoritmo, de modo a multiplicar a estimativa de distância restante até o destino por um peso determinado. A ideia dos autores em Wang *et al.* (2022) é que esse peso é maior quando a distância estimada até o destino é longa, fazendo com que o algoritmo A* tenha uma convergência mais rápida em direção ao seu alvo. A medida que o robô se aproxima do seu destino, esse peso é reduzido, fazendo que o algoritmo gere uma trajetória mais refinada, através de exploração de mais pontos.

O peso dinâmico introduzido em Wang *et al.* (2022) é uma maneira de alternar, em tempo de execução, entre uma trajetória mais refinada do algoritmo Dijkstra, através da exploração de mais nós, e um cálculo mais performático do BFS.

Para suavizar a trajetória gerada pelo algoritmo A*, os autores adicionam ao sistema uma última etapa, que utiliza o algoritmo de Floyd, ou método de interpolação, para reduzir a quantidade de pontos da rota calculada, conforme visto na Figura 25.

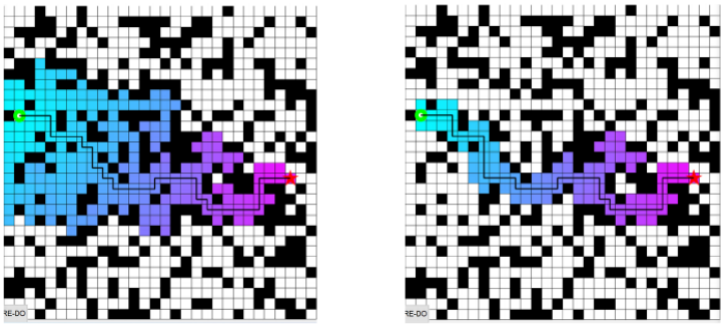
Figura 25 – Método de interpolação utilizado para suavizar as trajetórias do algoritmo A*.



Fonte: Wang *et al.* (2022) .

Os resultados obtidos pelos autores em Wang *et al.* (2022) mostram que o algoritmo A*, com as alterações propostas, apresentou redução na quantidade de nós explorados para concluir o cálculo da trajetória. Pode-se notar também uma redução no número de alterações de direção necessárias para um agente robótico atingir o seu destino, redução essa decorrente da utilização do método de interpolação.

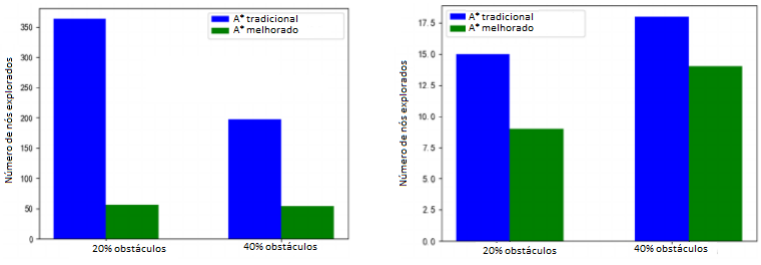
Figura 26 – Comparação das rotas calculadas. A* tradicional (à esquerda) e A* proposto (à direita)



Fonte: Wang *et al.* (2022) .

A melhora na eficiência do algoritmo proposto em Wang *et al.* (2022), em comparação com o método tradicional, pode ser visto em detalhes na Figura 27. No quadro da esquerda pode-se ver o número de nós explorados no cenário com 20% de ocupação do mapa por obstáculo e com 40% de ocupação por obstáculos, onde a barra azul representa o algoritmo A* de referência e em verde a proposta dos autores. Outro aspecto importante do algoritmo que pode ser visto na mesma imagem, no quadro a direita, é a redução no número de curvas, métrica importante especialmente em implementações com agentes robóticos no mundo real.

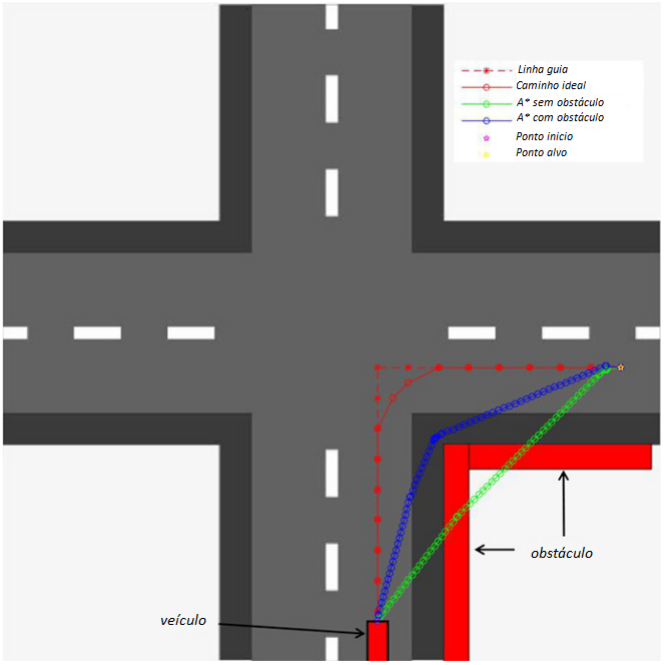
Figura 27 – Resultados obtidos



Fonte: Wang *et al.* (2022) .

A pesquisa em Erke *et al.* (2020) explora também algumas características do algoritmo A*, bem como pontos de melhoria, aplicado especificamente para o cenário de veículos autônomos terrestres. Segundo os autores, apesar de ser um algoritmo simples e rápido, especialmente quando comparado com o seu antecessor Dijkstra, ele apresenta alguns pontos negativos. Um desses pontos diz respeito ao comportamento do algoritmo ao calcular rotas que envolvam esquinas. Seu comportamento nesses casos difere bastante quando comparado a uma trajetória traçada por um humano. Isso se dá pelo fato da implementação clássica sempre procurar convergir o mais rapidamente para o destino. Essa diferença pode ser vista em detalhes na Figura 28, onde a linha tracejada em vermelho representa um rota “humana” e as demais representam trajetórias do algoritmo considerando obstáculos (azul) e sem considerar obstáculos do trajeto (verde).

Figura 28 – Diferença entre a rota do algoritmo A* e de uma trajetória feita por um humano



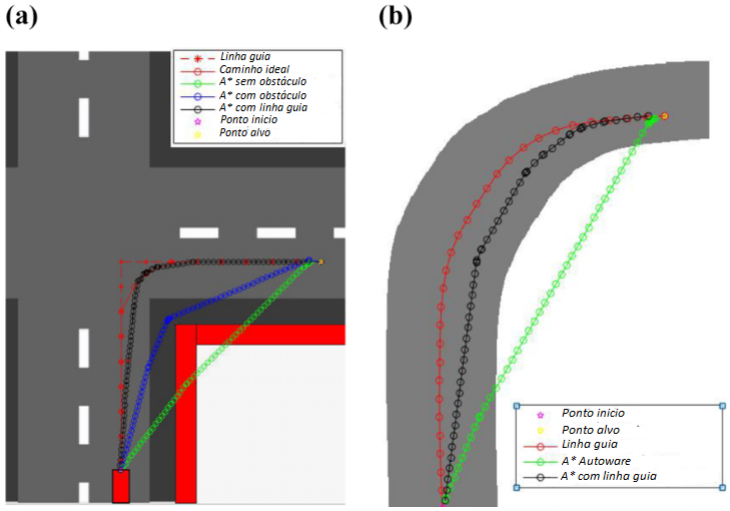
Fonte: Erke *et al.* (2020) .

Outra dificuldade mencionada pelos autores em Erke *et al.* (2020) relacionada ao uso do A* é a parametrização inicial do algoritmo. Essa parametrização pode trazer resultados distintos e tem relação direta com o local de uso do agente robótico. Essa característica, como mencionado também na pesquisa de Wang *et al.* (2022), pode alterar a trajetória final obtida, e fazer com com que o algoritmo balanceie seu comportamento entre o modelo de Dijkstra e o BFS.

Os autores em Erke *et al.* (2020) propõe então um algoritmo baseado no A* que leve em consideração o custo do agente robótico em fazer curvas, a detecção dos limites da rua onde o veículo está trafegando, uma rota guia que mostra o caminho ótimo que deveria ser seguido, e a existência de eventuais obstáculos no caminho. O trabalho compara então as trajetórias calculadas com o algoritmo clássico A*, bem como com rotas geradas pelo algoritmo de referência

Autoware (2022). Os resultados obtido podem ser vistos na Figura 29, que ilustra o cenário com a detecção dos limites da rua (a) e sem a detecção de onde a rua termina (b).

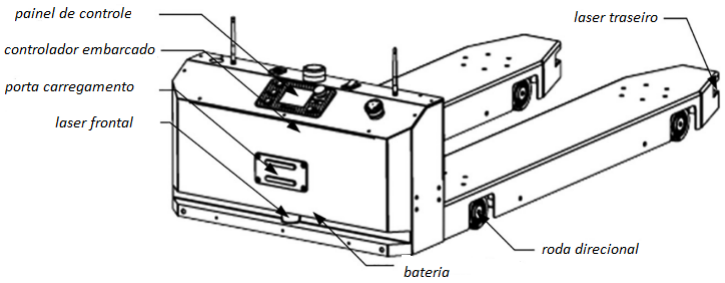
Figura 29 – Resultados obtidos



Fonte: Erke *et al.* (2020) .

Uma pesquisa aplicada diretamente no cenário logístico pode ser encontrada em Lin, Huang e Li (2021). Apesar de seu foco estar mais direcionado nos desafios da construção de um robô capaz de transportar uma carga de até 1000 quilos, o autor utiliza o algoritmo A* tradicional para o cálculo das trajetórias percorridas no armazém. O diagrama do robô construído pode ser visto na Figura 30.

Figura 30 – Base robótica construída



Fonte: Lin, Huang e Li (2021) .

Como visto na Figura 30, a base robótica possui dois sensores laser, um posicionado na parte dianteira, responsável por capturar os dados do ambiente para criação do modelo virtual do armazém, e outro na parte traseira, que é o responsável pelo rastreio da carga durante o carregamento. Além disso, a odometria do agente robótico é feita através da medição da rotação de cada uma das rodas.

Os autores em Lin, Huang e Li (2021) também utilizaram algumas restrições de domínio de negócio na modelagem do mapa, de modo a aproximar a solução com um cenário real. Essas restrições são:

1. Área restrita: O agente não deve passar na área.

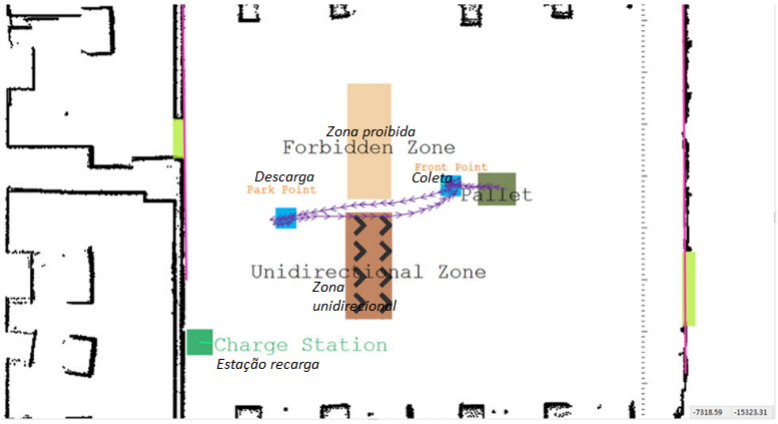
2. Linha restrita: O agente não deve cruzar a linha.

3. Zona unidirecional: Área em que somente um sentido de movimento é permitido.

4. Área de resistência: Aumenta o custo da estimativa do A* ao passar pela área.

Os resultados obtidos pelo processo de cálculo de trajetória e navegação de Lin, Huang e Li (2021) podem ser vistos na Figura 31. Os autores avaliaram a eficiência e eficácia do robô proposto através de execução de 200 trajetórias, com distância média entre a origem e o destino de aproximadamente 10 metros. Os dados mostraram que o erro médio ao descarregar a carga no ponto de destino foi de 11mm e o erro máximo foi de 26.18mm. Em 42,57% das rotas o erro ficou abaixo de 10mm e em 94,55% das vezes o erro foi menor que 20mm.

Figura 31 – Resultados obtidos

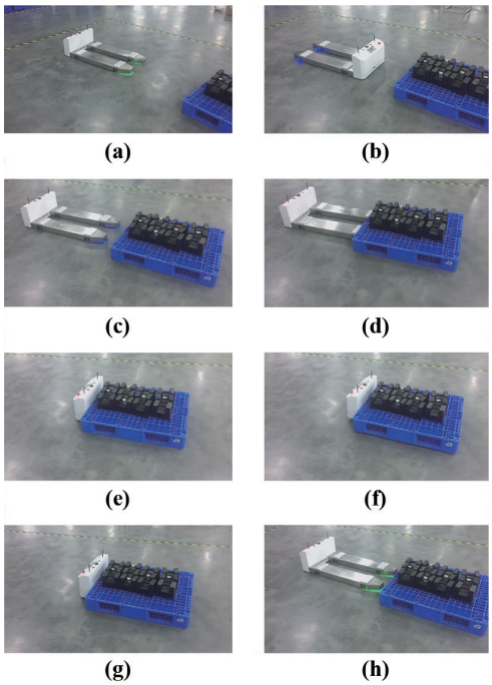


Fonte: Lin, Huang e Li (2021) .

Os autores em Lin, Huang e Li (2021) concluem que o robô desenvolvido tem inúmeras vantagens frente ao maquinário convencional operado por agentes humanos. Dentre essas vantagens, é citada a precisão do agente em efetuar o processo de coleta/descarga da carga paletizada, visto em detalhes na Figura 32, graças ao sensor colocado na parte traseira. Como o robô também é capaz de retornar automaticamente para uma estação de carga, a eficiência operacional também é destacada, pois viabiliza uma operação de 24 horas por dia e 7 dias por semana, desde que não sofra danos durante o processo. Outra característica importante é o seu tamanho, muito mais compacto se comparado a um carregador tradicional, fazendo com que os corredores do armazém possam ser reduzidos, aumentado a área útil de armazenagem de materiais.

Essa seção apresentou alguns trabalhos desenvolvidos ao longo dos últimos anos relacionados a área de navegação robótica e planejamento de trajetória. Nas próximas seções, além das considerações sobre as pesquisas mostradas, serão apresentados em detalhes os aspectos dos algoritmos de campos potenciais e A* relevantes ao presente trabalho, bem como um resumo rápido sobre as estruturas encontradas no relevo oceânico, fonte de inspiração para modelagem do mapa do presente estudo.

Figura 32 – Resultados obtidos



Fonte: Lin, Huang e Li (2021) .

2.2 Considerações

Conforme visto na seção anterior, alguns problemas aparecem de maneira recorrente quando da utilização dos métodos A* e campos potenciais na navegação robótica. Como mais evidentes é possível citar os mínimos locais dos campos potenciais Pradhan *et al.* (2006), Wang *et al.* (2019), Szczepanski, Bereit e Tarczewski (2021) e a geração de rotas que não se adéquam ao domínio do problema, por possuírem muitas curvas e/ou proximidade com obstáculos, obtidas pelo método A* Duan (2018), Xiang *et al.* (2022), Wang *et al.* (2022), Erke *et al.* (2020).

Naturalmente deve-se ressaltar os pontos positivos de ambos algoritmos, que também foram explorados nas pesquisas apresentadas, sendo os principais a possibilidade de compartilhamento do mapa de campo potencial para múltiplos agentes robóticos, e a capacidade do método A* de sempre encontrar uma rota até o destino, independente da existência de características como corredores em “U” e obstáculos, desde que tal rota exista.

O presente trabalho, como visto na seção 1.2, tem por finalidade potencializar a capacidade do método A* em sua característica de sempre encontrar uma rota atendendo as restrições de domínio específicos da logística, como segurança, redução do número de curvas e prevenção de mínimos locais. Para isso algumas características do relevo oceânico, explicadas na sequência do trabalho, serão usadas para gerar um mapa de modo computacionalmente eficiente, assim como ocorre com um campo potencial, e que também possa ser compartilhado entre inúmeros agentes robóticos.

Unindo os pontos positivos das abordagens exploradas, e eliminando a ocorrência de mínimos locais, acredita-se que a presente proposta pode não somente atender o problema

específico da navegação em ambiente logístico, mas também ser explorada em outras áreas da navegação robótica, através da alteração de parâmetros do sistema.

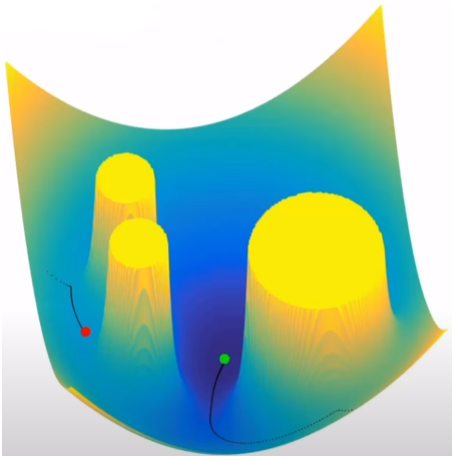
2.3 Campos potenciais

Como visto na seção anterior, campos potenciais são amplamente utilizados no planejamento de trajetórias para a área de robótica devido à sua simplicidade, elegância e principalmente sua eficiência computacional.

O método foi originalmente proposto em Khatib (1985), e baseia-se na ideia de que o agente robótico está constantemente operando sob influência de campos com força atrativa, usados para indicar o objetivo final do robô, e campos com força repulsiva, usados para representar obstáculos ao longo do caminho.

Uma maneira bastante intuitiva de visualizar o seu funcionamento é imaginar uma bola descendo uma ladeira Goodrich (2002), ou uma partícula carregada com carga positiva navegando por um campo, onde o agente robótico e os obstáculos possuem o mesmo sinal de carga, portanto repelindo-se, e o objetivo possui carga com sinal oposto ao do agente, portanto atraindo-o. Um exemplo em 3D pode pode ser visto na Figura 33 e em 2D na Figura 34.

Figura 33 – Exemplo de campo potencial com três obstáculos e dois agentes. Objetivo é marcado pela área mais escura

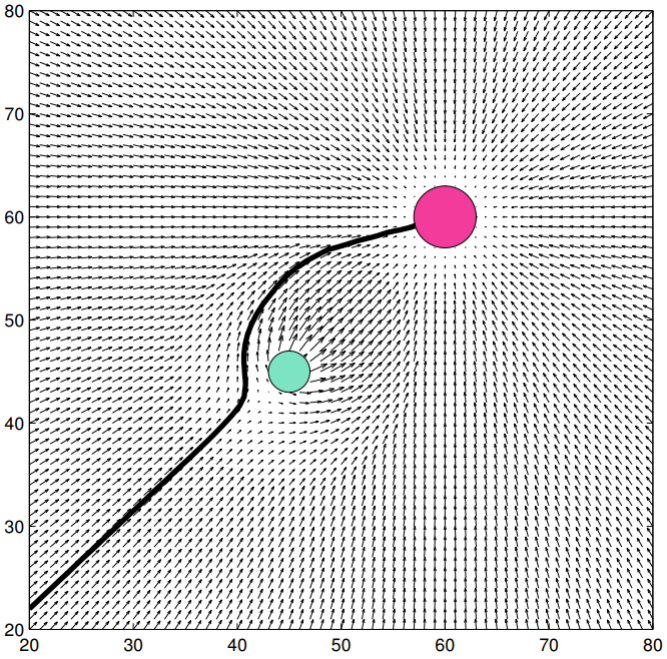


Fonte: Adaptado de Coulouris2013.

A direção que deve ser seguida pelo agente robótico ao navegar pelo ambiente é a soma das forças de todos os campos que influenciam o agente em determinado ponto. Um exemplo de campo potencial calculado pode ser visto na Figura 34, que representa um ambiente com um obstáculo e um objetivo. A área verde representa o centro do campo repulsivo do obstáculo e a área rosa representa o objetivo. A linha preta é a trajetória calculada pelo robô ao interagir com as forças dos campos em cada coordenada x e y.

Para chegar ao campo demonstrado na Figura 34, considere que a posição cartesiana do agente robótico no ambiente é definida por $q = (x,y)$. O campo é então calculado tal que

Figura 34 – Exemplo de campo potencial calculado representado em 2D.



Fonte: Notas de aula publicadas em Goodrich (2002).

$$U(q) = U_{at}(q) + U_{rep}(q) \tag{1}$$

onde

$U(q)$ = campo potencial resultante
$U_{at}(q)$ = campo potencial atrativo (objetivo)
$U_{rep}(q)$ = campo potencial repulsivo (obstáculos)

Inúmeras funções podem ser utilizadas para calcular a força exercida pelos campos, conforme visto em Sabudin, Omar e Melor (2016). Funções com aplicações específicas também são encontradas na literatura, como visto em Rasekhipour *et al.* (2016), onde os autores utilizaram funções hiperbólicas para obstáculos intransponíveis (pedestres) e funções exponenciais para representar campos de objetos que preferentemente devem ser evitados (buracos em uma rodovia). O formato final do campo é diretamente relacionado às funções utilizadas para cada componente do sistema.

Independentemente da função escolhida, o objetivo é calcular o mapeamento do vetor $v = [x,y]$ para um vetor de gradientes $\Delta = [\Delta x, \Delta y]$, através da definição da função v e do cálculo do seu gradiente. Outra opção é a definição de Δy e Δx em função de v de acordo com termos pré-definidos. Por exemplo, um campo atrativo conforme mostra a Figura 35 pode ser definido da seguinte maneira Goodrich (2002):

se $d < r$ então

$$\Delta x = \Delta y = 0 \tag{2}$$

se $r \leq d \leq s + r$ então

$$\Delta x = \alpha(d - r)\cos\theta \text{ e } \Delta y = \alpha(d - r)\sin\theta \tag{3}$$

se $d > s + r$

$$\Delta x = \alpha s\cos\theta \text{ e } \Delta y = \alpha s\sin\theta \tag{4}$$

sendo que:

- x_G, y_G são as coordenadas do objetivo.
- r é o raio do objetivo.
- $v = [x,y]$ são as posições x e y do agente.
- $d = \sqrt{(x_G - x)^2 + (y - y_G)^2}$ é a distância do agente até o objetivo.
- $\theta = \tan^{-1}(\frac{y_G - y}{x_G - x})$ é o ângulo entre o agente e o objetivo.
- s é a área de influência do campo.

Da mesma maneira que foi calculado o campo atrativo, pode-se também fazer o cálculo do seu campo repulsivo definindo Δx e Δy:

se $d < r$ então

$$\Delta x = -\text{sign}(\cos\theta)\infty \text{ e } \Delta y = \text{sign}(\sin\theta)\infty \tag{5}$$

se $r \leq d \leq s + r$ então

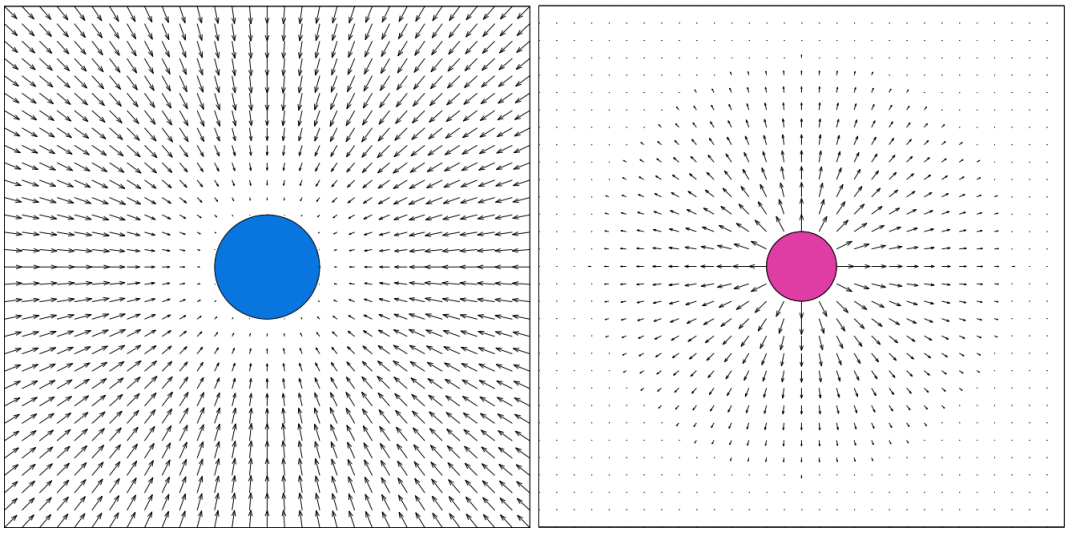
$$\Delta x = \beta(s + r - d)\cos\theta \text{ e } \Delta y = -\beta(s + r - d)\sin\theta \tag{6}$$

$$\text{se } d > s + r \text{ então } \Delta x = \Delta y = 0 \tag{7}$$

sendo que as variáveis das expressões são as mesmas utilizadas para o cálculo do campo atrativo, porém em relação ao obstáculo, e não ao objetivo.

Como o método permite o uso de qualquer função, pode-se deduzir que inúmeros formatos de campos podem ser obtidos. Apesar do campo atrativo geralmente utilizar funções menos

Figura 35 – Exemplo de campo potencial atrativo (esq) e repulsivo (dir).



Fonte: Notas de aula publicadas em Goodrich (2002).

complexas, uma vez que seu objetivo (atrair o agente robótico) é muitas vezes o mesmo, é nos obstáculos que pode-se explorar mais a fundo essa característica.

Através de alterações nessas funções pode-se obter alguns tipos de campos, cada um com sua função específica, conforme Goodrich (2002):

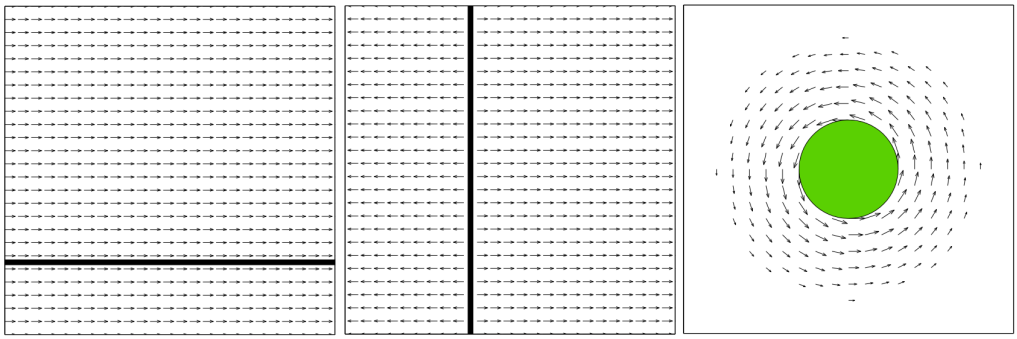
- Uniforme: obtido com $\Delta x = c$ e $\Delta y = 0$
- Perpendicular: obtido com $\Delta x = \pm c$ e $\Delta y = 0$
- Tangencial: calculado de forma semelhante ao campo repulsivo citado anteriormente, porém girando θ em $\pm 90º$

sendo c uma constante usada para definir a magnitude do campo potencial gerado.

Inúmeros exemplos práticos podem ser projetados com esses campos. O campo do tipo uniforme pode ser usado para um cenário em que deve-se seguir uma parede, ou se mover em direção a determinada área. Em contraste com o campo uniforme, o campo do tipo perpendicular pode ser utilizado para que o agente trace uma trajetória evitando uma parede, ou para que evite determinada área do ambiente. O campo do tipo tangencial pode ser utilizado para fazer com que o robô se mantenha em determinada área, como um cenário de patrulha. Esses exemplos de aplicação podem ser imaginados ao observar a Figura 36.

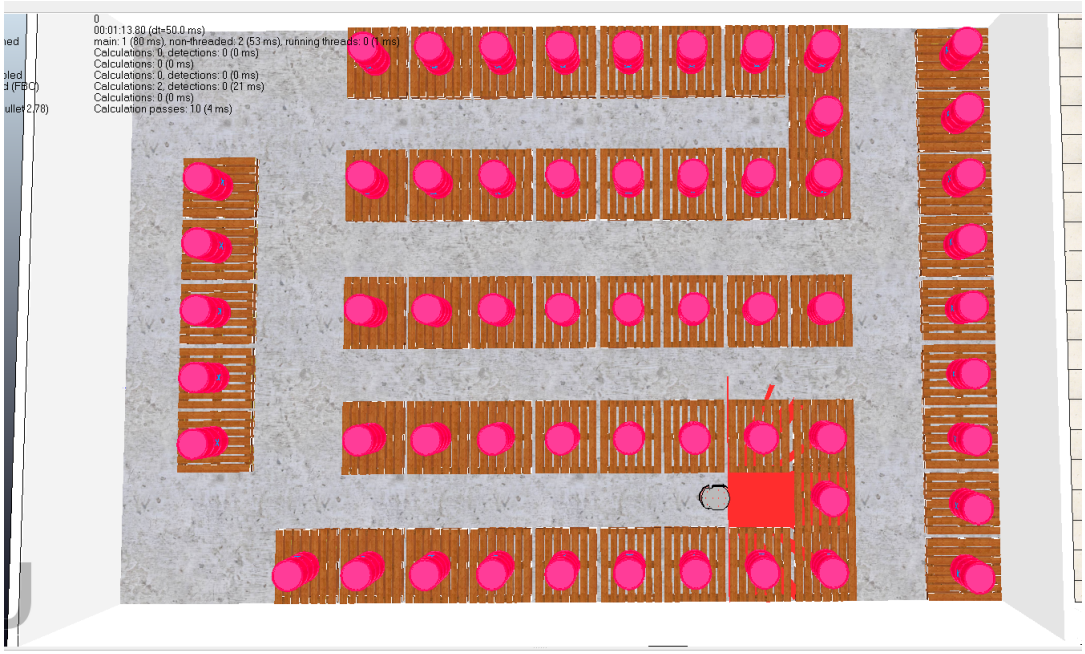
Como visto em outros trabalhos, um dos problemas que surge com a utilização da navegação por campos potenciais é a ocorrência de mínimos locais, especialmente considerando o cenário de um armazém logístico repleto de prateleiras e corredores. Inúmeras pesquisas se dedicam a resolver esse problema de maneira computacionalmente eficiente, conforme visto em Chiang *et al.* (2015), Guerra *et al.* (2016), Matoui, Boussaid e Abdelkrim (2015) entre outras citadas na seção anterior. Mesmo em cenários estáticos, o problema pode aparecer dependendo da configuração do ambiente, conforme visto na Figura 37, que exemplifica uma situação

Figura 36 – Da esquerda para a direita: o campo uniforme, perpendicular e tangencial.



Fonte: Notas de aula publicadas em Goodrich (2002).

Figura 37 – Situação em que o agente robótico está preso em um mínimo local. Modelado no CoppeliaSim.



Fonte: Autoria própria (2023).

facilmente encontrada no mundo real. Uma das preocupações ao desenvolver o algoritmo proposto nesse estudo é que o agente robótico não seja direcionado para mínimos locais dentro do armazém.

2.4 Método A*

Originalmente proposto em 1968 Hart, Nilsson e Raphael (1968), o algoritmo apresentou um método de incorporar restrições do domínio do problema em uma busca em grafo, unindo soluções antes puramente matemáticas com abordagens puramente heurísticas. Seu funcionamento se baseia no cálculo de uma função $f(v)$ para cada célula do grafo:

$$f(v) = g(v) + h(v) \tag{8}$$

onde

$h(v)$ = distância estimada de v até o nó de destino
$g(v)$ = distância do nó de origem até o nó v

Como um algoritmo de busca em grafo deve constantemente decidir qual célula deve ser avaliada na busca do melhor caminho, a ideia dos autores foi introduzir tal função relacionada ao domínio do problema, para que a decisão sobre qual célula expandir seja sempre a mais informada possível, evitando calcular células que não estejam em um caminho ótimo em detrimento das que se mostram mais promissoras respeitando as restrições do problema.

A função $g(v)$ é a que apresenta maior simplicidade em seu cálculo, uma vez que a medida que o algoritmo avança, a distância entre as células adjacentes passa a ser conhecida. A função $h(v)$ é onde geralmente é introduzido o domínio do problema ao algoritmo, uma vez que a distância de v até o destino pode ser desconhecida, sendo geralmente utilizada uma estimativa baseada em alguma heurística.

O algoritmo tem os seguintes passos:

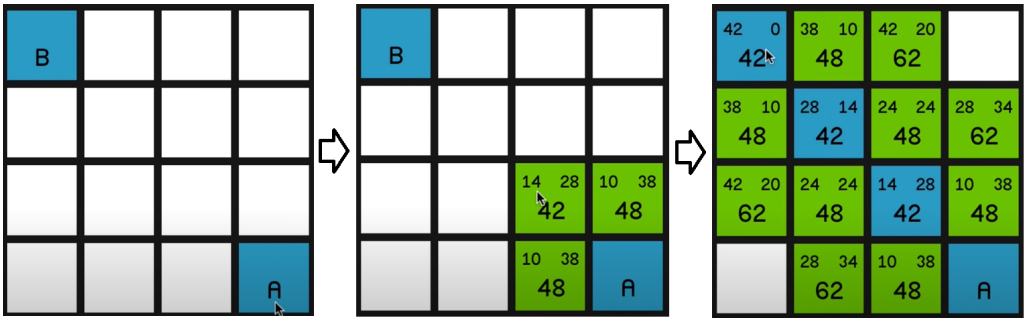
Algoritmo 1 – Exemplo A*

$A \leftarrow m$
enquanto $A \neq \emptyset \wedge a \neq t$ **faça**
  $a \leftarrow x$ onde x possui menor $f(x)$ e $x \in A$
  $A \leftarrow A - a$ Remove a do conjunto Aberto
  $F \leftarrow F + a$ Adiciona a na lista Fechada
  para v `vizinho de` a **faça**
    se $v \notin A$ **então**
      calcule custo $f(v)$
      guarde a como pai de v
      $A \leftarrow A + v$
    senão,
      $v \in A \wedge f(a) \leq f(v)$ // Custo da rota passando por a é menor que atual de v recalcule custo $f(v)$
      guarde a como pai de v
    finaliza se
  finaliza para
finaliza enquanto

onde:

m é a célula inicial.
a é a célula atual.
t é a célula alvo.
A é a lista de células a serem exploradas, chamada de lista ABERTA.
F é a lista de células exploradas, chamada de lista FECHADA.

Figura 38 – Exemplo de grid de tamanho 4x4.



Fonte: Autoria própria (2023).

Apesar de eficaz, a natureza de funcionamento do algoritmo pode apresentar problemas na navegação robótica, uma vez que seu desempenho está diretamente relacionado com o tamanho do grid. Como visto em Duchon *et al.* (2014)Tirumalapudi e Vedaraj (2020), é possível melhorar tal desempenho através de técnicas de pré-processamento das informações do grid, visando reduzir a quantidade de células a serem calculadas pelo algoritmo A*, e também através de modificação na própria fórmula apresentada em (8).

Como visto na equação (8), o algoritmo possui duas funções que devem ser calculadas para cada célula do grid. Através de alterações nessas funções é que o algoritmo passa a considerar as informações do domínio do problema no cálculo da trajetória, conforme citado anteriormente. No entanto, para fins de validação, pode-se considerar um modelo de cálculo bastante simplificado para determinar tal rota.

Considerando que cada célula do grid proposto na Figura 38 possui tamanho 10 por 10, pode-se afirmar que o custo de movimento horizontal ou vertical entre as células possui um total de 10, enquanto através do teorema de Pitágoras o custo de movimentos diagonais entre as células tem custo de aproximadamente 14.

Para cada célula pode-se ver o custo $g(h)$ representado no canto superior esquerdo e o custo $h(v)$ representado no canto superior direito. A soma de ambos, que representa $f(v)$, é mostrado no centro de cada célula. Naturalmente, esse exemplo tem pouca aplicabilidade no mundo real, uma vez que não traz alguma heurística do problema para dentro do cálculo, entretanto é de suma importância como critério de comparação entre soluções que trazem o domínio do problema para dentro do cálculo dessas funções, conforme apresentado no decorrer do trabalho.

Outro ponto de atenção do algoritmo A* clássico é que determinadas restrições do domínio do problema podem não ser atendidas, como, por exemplo, excesso de curvas na rota e proximidade com obstáculos. Esses fatores ganham ainda mais relevância a medida que a pesquisa avança para implementações no mundo real, visto que podem comprometer a integridade da carga transportada ou do próprio robô.

Figura 39 – Características do relevo oceânico.

Fonte: https://mundoeducacao.uol.com.br/geografia/relevo-submarino.htm.

2.5 Relevo oceânico

Conforme mencionado anteriormente, esse trabalho se baseou nas características do relevo oceânico para a construção de um mapa aplicável na navegação de um agente robótico. Para tal, as seguintes características foram consideradas com o objetivo de resolver o problema Tessler e Mahiques (2000):

- Plataforma continental: Região de águas rasas que circunda os continentes. É caracterizada por possuir profundidade média de 130 metros e baixo índice de declive, inferior a 1m/km. Possui uma largura média de 75km.
- Talude: Se estende a partir do término da plataforma continental. Relativamente é a porção mais íngreme do oceano, em média 25m/km, e profundidades que vão de 100m até 3200m. Possui largura de 10km até 200km.
- Planície abissal: Faz parte do assoalho de bacia oceânica e caracteriza-se por ser uma área bastante plana, inclusive podendo ser considerada a estrutura natural mais plana da Terra. Estão na faixa de 4000m a 6000m de profundidade.
- Ilha oceânica: Porções de terra de pequeno porte que afloram acima do nível do mar em regiões distantes do continente.

Essas características podem ser visualizadas em detalhes na Figura 39 , e sua utilização no algoritmo de mapeamento será detalhada no capítulo 3.

Outro aspecto importante da dinâmica dos oceanos incorporada na modelagem do mapa de navegação foi o surgimento e propagação de ondas. Na natureza, a ondulação é um processo que se origina com mais frequência através da fricção do vento com a água do mar. Essa

perturbação gerada na superfície possui um comprimento de onda estendido enquanto navega por águas profundas, e a medida que se aproxima da costa, seu comprimento reduz e sua altura aumenta. Ao se chocar contra o fundo a parte superior da onda acelera em relação a sua base, até o ponto em que a onda perde sustentação e arrebenta.

Para fins de modelagem, o presente trabalho usou as seguintes características da ondulatória oceânica:

- Ondas profundas: Ocorrem quando a profundidade é maior que metade do comprimento de onda.
- Ondas de transição: Acontecem quando a profundidade é inferior a metade do comprimento de onda mas maior que 1/20 do comprimento de onda.
- Ondas de águas baixas: São ondas cuja profundidade é inferior a 1/20 do comprimento de onda.
- Arrebentação: Apesar de não haver consenso na literatura sobre o momento exato da arrebentação, nesse artigo considerou-se o momento em que a profundidade é inferior à 1m.

3 MATERIAIS E MÉTODOS

Como visto no Capítulo 2, existem inúmeras pesquisas que apresentam soluções para mitigar os pontos fracos dos algoritmos A* e de campos potenciais. Dentre esses, destacam-se a tendência natural dos campos potenciais em mapear rotas com mínimos locais, bem como as dificuldades do A* em respeitar restrições ligadas ao domínio do problema, como segurança, eficiência operacional e performance.

Apoiado nos algoritmos vistos previamente nesse trabalho, procurou-se resolver os problemas mencionados dessas técnicas através da criação de uma nova abordagem que tenha como fundamento as características do relevo oceânico e propagação de ondas vistas na seção 2.5.

3.1 Seleção e instalação dos sistemas de apoio

Para viabilizar a criação do algoritmo proposto, bem como possibilitar as simulações necessárias para validação dos resultados, alguns sistemas de apoio foram selecionados e utilizados durante a construção desse trabalho. Esta seção descreve brevemente quais são esses sistemas. A lista abaixo enumera-os e detalhes sobre sua utilização serão descritos na sequência:

1. ROS: Sistema operacional utilizado na robótica.
2. CoppeliaSim: Simulador.
3. RabbitMQ: Barramento de mensagens.
4. Docker: Ferramenta de virtualização de outros sistemas.

A integração entre algoritmos de planejamento e atitude e as inúmeras opções de bases robóticas se apresenta como um desafio em inúmeras pesquisas. A necessidade de um sistema atuando como um *Middleware* entre o robô e os algoritmos de controle inspirou a criação do ROS. Ele possui uma grande quantidade de ferramentas que lidam com os mais diversos problemas que a robótica apresenta, como controle de sensores, motores, posicionamento, mapeamento, comunicação entre outros ROS (2022), e é distribuído através de uma licença 3-clause BSD Initiative (2022).

Para reduzir os custos e impulsionar a pesquisa nos estágios iniciais, é interessante validar o modelo proposto através de ambientes simulados. Isso evita lidar com a complexidade de bases robóticas enquanto o modelo de sistema ainda está sendo refinado. Para essa finalidade, um dos ambientes de desenvolvimento mais utilizados é o CoppeliaSim, da Coppelia Robotics Rohmer, Singh e Freese (2013). Ele possui uma grande variedade de sensores e robôs

e se integra facilmente com o ROS. Na próxima seção serão apresentados o ambiente virtual modelado bem como o agente robótico utilizado no trabalho.

Para tornar a solução mais flexível e desacoplada, as informações geradas pelos robôs na simulação, sendo executada no CoppeliaSim, são enviadas para um barramento de mensagens antes de chegarem ao software que calcula o modelo de mapa e a trajetória. Orientar a solução ao uso de mensagens permite que alterações, tanto nos robôs quanto no sistema de controle, possam ser feitas de maneira independente. Além disso, outra grande vantagem dessa abordagem é permitir que a solução se torne facilmente um sistema multi-agente. Para isso, basta introduzir novos robôs na simulação, e novos sistemas de controle podem ser executados para ler as mensagens dos robôs adicionados. Outra possibilidade valiosa é a execução de mais de um sistema de controle consumindo as mensagens do mesmo agente robótico. Dessa maneira, sistemas de controle diferentes podem ser testados em paralelo recebendo exatamente os mesmos dados da simulação, garantindo as mesmas condições para testes de abordagens distintas.

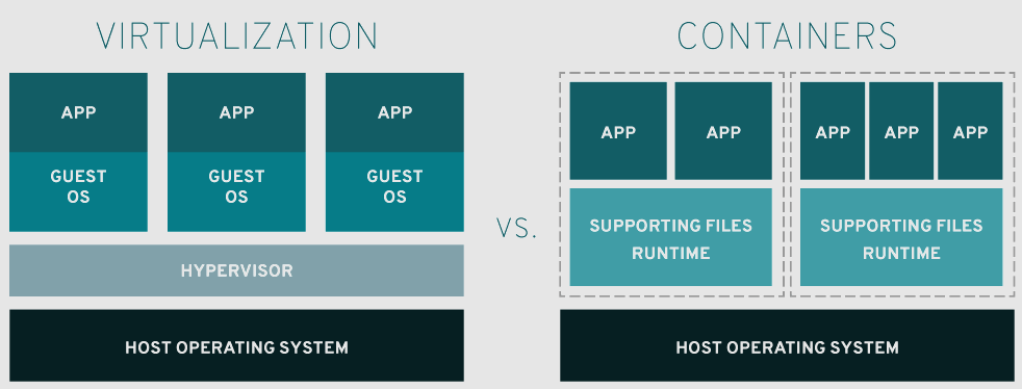
O barramento de mensagens escolhido foi o RabbitMQ. Os principais motivos dessa escolha foram a facilidade de instalação, principalmente combinado com a execução em containers do Docker, a baixa utilização de recursos da máquina em termos de memória RAM e CPU e principalmente a excelente performance quando submetido a um grande volume de mensagens, especialmente tendo em mente que um sensor LIDAR pode enviar milhares de mensagens por segundo, dependendo do estado da simulação. Modos de instalação podem ser encontrados diretamente no site da ferramenta.

Como pode ser visto no site do próprio barramento, existem inúmeras opções de instalação disponíveis. Para o presente trabalho, foi feita a instalação através da ferramenta de conteinerização Docker, mais especificamente através do arquivo de configuração mostrado na Listagem 1.

O Docker é uma ferramenta de virtualização de sistemas semelhante às soluções utilizadas para executar máquinas virtuais, como o Hyper-V ou o VMware. A principal diferença é que a execução de uma máquina virtual demanda também a execução de uma cópia do sistema operacional inteiro, causando um desperdício de recursos da máquina física. O Docker difere dessas soluções pois sua virtualização compartilha os recursos do sistema operacional onde está instalado, otimizando o consumo de CPU e memória RAM e simplificando a execução de inúmeras ferramentas distintas no mesmo computador RedHat (2022). Essa diferença entre os modelos pode ser vista em detalhes na Figura 40.

A instalação dessas ferramentas dentro do ambiente do Docker é normalmente baseada em arquivos de configuração, como o visto na Listagem 1, o que traz uma padronização para o processo, garantindo que o sistema se comporte da mesma maneira em computadores diferentes e assim facilitando a reprodução de resultados.

Figura 40 – Diferença entre os modelos de virtualização clássico e execução em containers do Docker.



Fonte: RedHat (2022).

Os sistemas vistos nessa seção são as principais ferramentas de apoio utilizadas. Nas seções seguintes será detalhado o desenvolvimento do algoritmo proposto.

3.2 Visão geral da abordagem proposta

O trabalho foi desenvolvido no ambiente virtual CoppeliaSim, visto na seção 3.1, através da modelagem de um armazém conforme previamente visto na Figura 37 . Os pontos de coleta de material ficarão distribuídos entre essas ruas, e as suas posições serão aleatórias.

Naturalmente um armazém real pode apresentar características diferentes das modeladas, como por exemplo ser dezenas de vezes maior, ou ser composto por prateleiras, ou não possuir nenhuma estrutura de armazenamento, e simplesmente guardar os materiais em áreas no chão, como nos casos de armazenagem e distribuição de grãos em *bags* de uma tonelada. O objetivo principal da modelagem proposta é criar uma boa quantidade de desafios para os algoritmos, incluindo imprecisões de sensoriamento e corredores sem saída. Além disso, ter um tamanho relativamente pequeno que viabilize a simulação em um computador de uso geral.

A leitura dos dados do ambiente simulado é feita pelo agente robótico Pioneer p3dx, disponível nativamente no CoppeliaSim. Esse agente, em detalhes na Figura 41, é responsável por informar o sistema sobre a sua posição e sobre a “visão” do ambiente onde está operando. Em relação ao modelo nativo da ferramenta, as seguintes alterações foram feitas:

1. LIDAR: Adicionado sensor laser Fast Hokuyo para detecção dos obstáculos presentes no armazém. O sensor possui raio de leitura de 180 graus e distância máxima de detecção de 2 metros.

2. Ultrassom: Removidos os sensores ultrassom para reduzir a carga de processamento do modelo.

3. Código: Adicionado código responsável por enviar as informações do sensor LIDAR para o ROS.

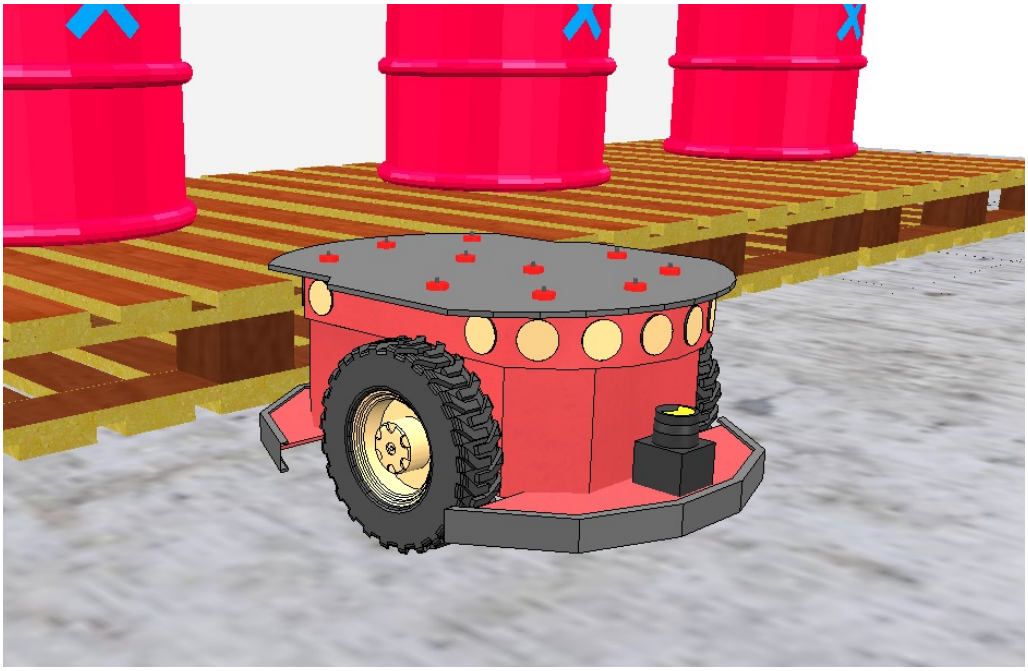
Listagem 1 – Arquivo docker-compose.yaml para configuração do RabbitMQ

```
version: '2.4'

###### Rede compartilhada por todos os servicos
networks:
  servicos:
    name: servicos
    driver: bridge
    ipam:
      config:
        - subnet: 172.18.0.0/16

volumes:
  rabbitmq: {}

services:

  rabbitmq:
    container_name: "rabbitmq"
    hostname: "rabbitmq"
    environment:
      RABBITMQ_DEFAULT_USER: "admin"
      RABBITMQ_DEFAULT_PASS: "1234"
    image: rabbitmq:3-management
    restart: always
    networks:
      - servicos
    ports:
      - "15672:15672"
      - "5672:5672"
      - "5671:5671"
      - "15671:15671"
    volumes:
      - rabbitmq:/var/lib/rabbitmq
```

Fonte: Autoria própria (2023).

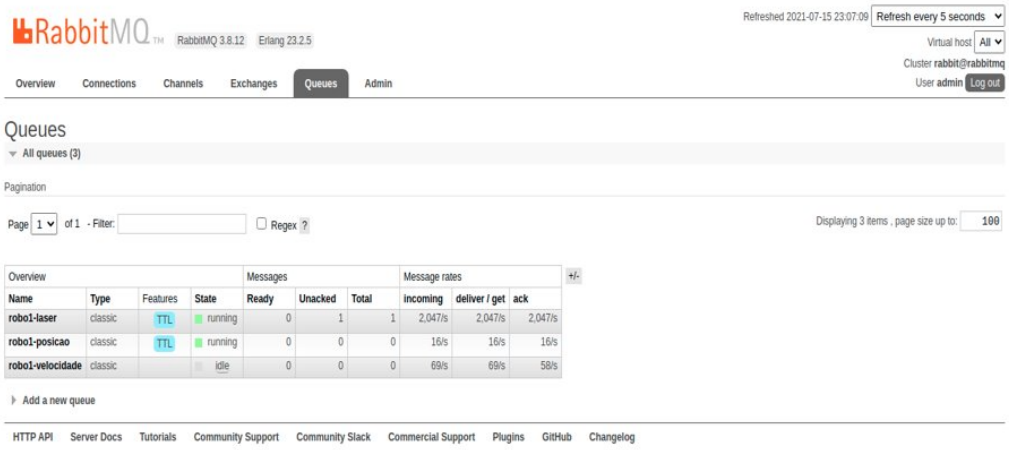
Como mencionado na Seção 3.1, o sensor LIDAR pode gerar um grande volume de dados, especialmente se em sua área de detecção estiver presente algum obstáculo. A Figura 42 demonstra o volume de mensagens transitados pelo barramento, em requisições por segundo, considerando o cenário proposto, com um agente robótico sendo simulado e um sistema de controle modelando o mapa e calculando as rotas. Na Figura 42, a fila “robo1-laser” contém as mensagens enviadas pelo sensor LIDAR. Essa fila é a que apresenta o maior volume de dados, tendo em vista que o sensor Fast Hokuyo possui uma área de leitura de 180 graus e inúmeros feixes de laser, cada um lendo uma informação de distância detectada para eventuais obstáculos. A fila “robo1-posicao” informa o sistema de controle sobre os dados de posição e orientação do agente e por fim a fila “robo1-velocidade” possui mensagens relacionadas às velocidades angular e linear do robô.

Figura 41 – Agente robótico modelado, com o sensor LIDAR Fast Hokuyo na parte frontal.



Fonte: Autoria própria (2023).

Figura 42 – Volume de mensagens transitado pelo barramento visto através da interface web do RabbitMQ.



Fonte: Autoria própria (2023).

Além do uso do barramento RabbitMQ, visto na seção 3.1, uma camada adicional, chamada de Listener, foi desenvolvida e está presente na proposta. Ela é responsável por traduzir os modelos de dados usados pelo ROS em pacotes padronizados para a própria solução, como pode ser visto na Figura 43. Desse modo, o acoplamento é reduzido ainda mais, viabilizando a troca do ROS por outro sistema, por uma base robótica real, ou simplesmente por uma versão mais nova do próprio ROS. Caso isso seja necessário, pode-se alterar somente essa camada de tradução, escrita na linguagem Python, sem necessariamente alterar as demais partes do sistema, responsáveis pela modelagem do mapa e cálculo das rotas. O código dessa camada pode ser visto no Apêndice F.

Em termos de macro arquitetura, o sistema é dividido nos seguintes componentes:

1. Cenário: Tanto armazém quanto o agente robótico e seus sensores simulados no CoppeliaSim, enviando e recebendo dados através do ROS.

2. Listener: Aplicação em Python responsável por traduzir os dados entre o formato usado pelo ROS e os modelos desenvolvidos, e enviar e receber esses dados para o RabbitMQ.

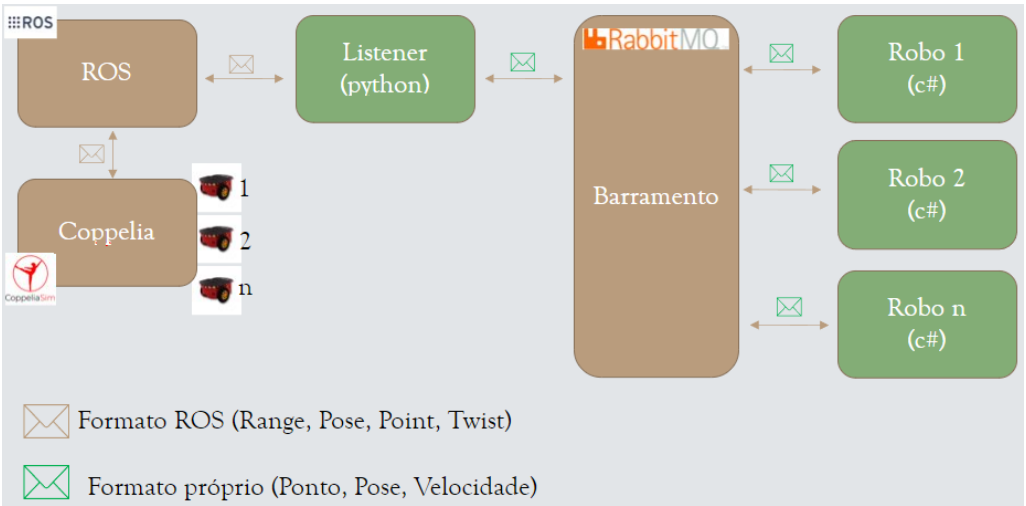
3. RabbitMQ: Barramento de mensagens utilizado para flexibilizar a solução proposta. Proporciona o desacoplamento entre os sistemas que controlam os agentes robóticos e o restante da aplicação.

4. Robos: Aplicação escrita em C# responsável por receber os dados vindos da simulação e calcular o mapa e as trajetórias. É a principal parte do algoritmo proposto neste trabalho.

Os componentes do sistema são vistos em seus lugares na Figura 43. A arquitetura proposta, através do uso do barramento, permite por exemplo que a solução seja capaz de operar inúmeros agentes simultaneamente. Todas as mensagens que transitam pela solução são publicadas em filas específicas que são lidas somente pelo sistema do agente robótico de destino. Além disso, mensagens especiais podem ser enviadas em *broadcast* para todos os agentes robóticos conectados, permitindo o compartilhamento de informações com toda a rede. Para esse trabalho, o caráter multi-agente do sistema não é importante, mas sem dúvidas pode ser explorado em propostas futuras.

A partir da simulação do cenário, são extraídas informações sobre a localização do agente e também de obstáculos detectados. Essas informações são então empacotadas e enviadas para o ROS. A partir dele, essas mensagens são lidas e padronizadas em formatos do modelo de dados do sistema, e então enviadas para suas filas específicas no barramento, como

Figura 43 – Componentes do sistema.



Fonte: Autoria própria (2023).

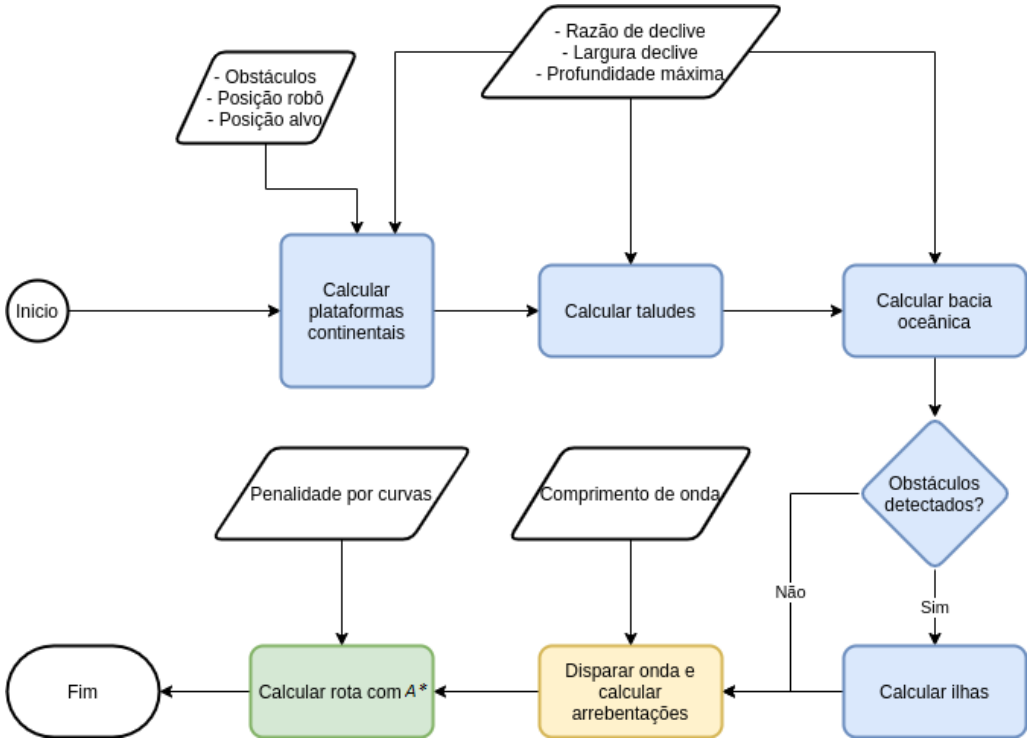
visto anteriormente na Figura 42. Os agentes robóticos conectados no barramento, cada um escutando sua própria fila, fazem então a leitura dessas mensagens e o posterior processamento de seu conteúdo. Mensagens que devem ser enviadas para a simulação seguem pelo mesmo caminho, mas em sentido inverso.

Das informações enviadas pela simulação, duas são fundamentais para o cálculo do mapa proposto neste trabalho:

1. Pose: Contém as informações sobre a posição e orientação atuais do agente robótico.
2. Point: Contém as informações sobre eventuais detecções de obstáculos feitas pelo sensor LIDAR do agente robótico.

O algoritmo proposto é subdividido em três etapas principais, conforme pode ser visto na Figura 44. A primeira fase do processo de cálculo, marcada em azul, consiste nos passos que geram o mapa do relevo oceânico. A fase seguinte, em amarelo, representa o disparo do processo ondulatório, e marca o término da modelagem do mapa. Por fim, em verde, o sistema encerra seu processamento com o cálculo da trajetória utilizando um algoritmo A* modificado. Esses passos serão detalhados nas seções seguintes.

Figura 44 – Processo do algoritmo proposto.



Fonte: Autoria própria (2023).

A próxima seção mostrará como as informações coletadas pelo agente robótico simulado, e transmitidas pela estrutura de sistemas apresentada até aqui, serão utilizadas para modelagem do mapa baseado no relevo oceânico.

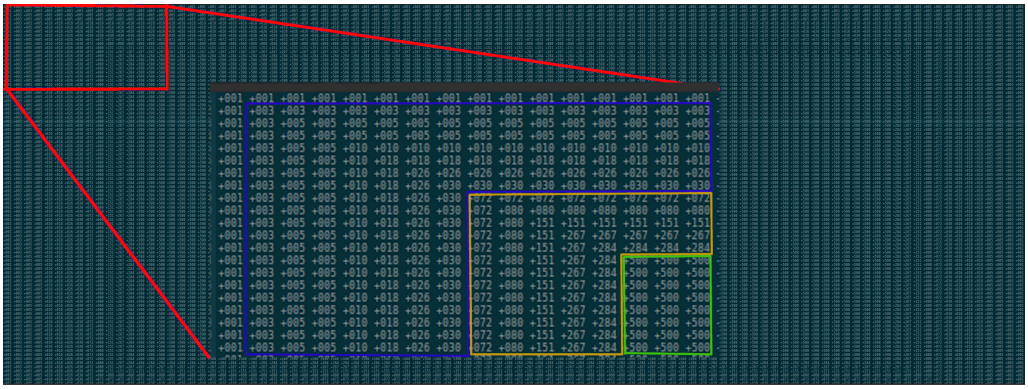
3.3 Modelagem do mapa baseado em relevo oceânico

Para iniciar o processo do algoritmo, conforme visto na Figura 44, o agente robótico precisa saber a sua localização atual e também as informações disponíveis relativas a existência de obstáculos no mapa. Naturalmente essas informações são atualizadas a medida que novas detecções são feitas, e nesses casos, o mapa pode ser atualizado e a rota pode ser recalculada para levar em consideração a posição dos novos objetos no *grid*.

A primeira fase do processo de cálculo, marcada em azul na Figura 44, consiste nos passos que geram o mapa do relevo oceânico. Ela é disparada com a inicialização do *grid* de profundidades, uma matriz bi-dimensional de inteiros maiores que zero. Na sequência, esse *grid* é povoado com as informações de profundidade de cada aspecto do relevo visto na Figura 39, considerando que os limites da matriz representam a costa, e a medida que se navega para o centro, a profundidade aumenta, respeitando as características de cada zona. O resultado final dessa fase é um mapa que se assemelha a um mar, no sentido de estar cercado por terra por vários lados. Essa matriz pode ser vista em detalhes na Figura 45. Todos os passos dessa etapa utilizam alguns parâmetros que podem ser definidos no algoritmo, de acordo com o ambiente e tipo de navegação desejados para o agente robótico.

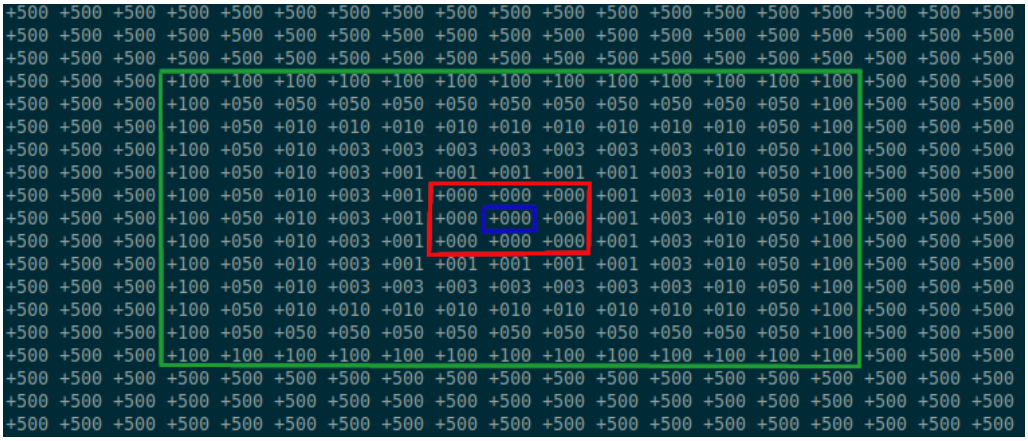
1. Profundidade máxima: Profundidade máxima que será atingida no cálculo da fase do relevo (plataforma, talude ou bacia)
2. Largura: Largura total da fase do relevo, em unidades.
3. Declive: Razão de declive para atingir a profundidade máxima desejada. Por exemplo, se 2 indica que a profundidade irá dobrar a cada incremento de x,y no *grid*.

Figura 45 – Corte em zoom da matriz após termino da fase 1. Cada posição [x,y] possui a profundidade do oceano no ponto dado, arredondado em escala 1/10.



Fonte: Autoria própria (2023).

Figura 46 – Representação de uma estrutura de ilha localizada dentro de uma área de bacia oceânica, em escala 1/10.



Fonte: Autoria própria (2023).

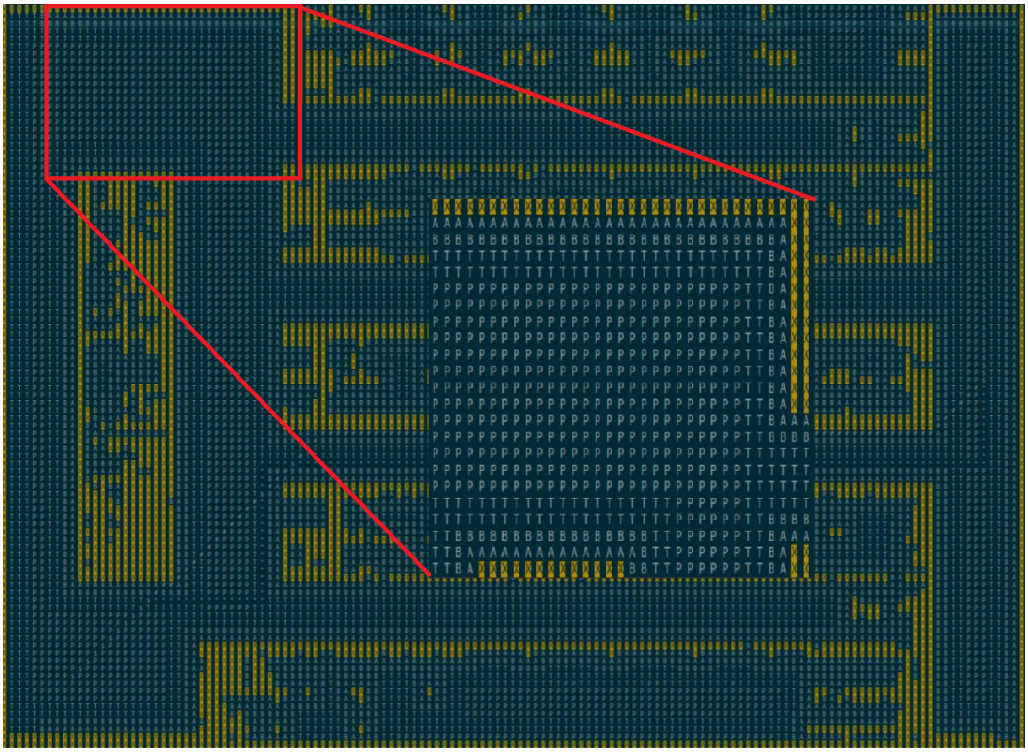
Esses parâmetros podem ser compreendidos através do recorte da Figura 45. Nela a plataforma continental possui largura de 7, razão de declive de 2 e profundidade máxima de 200, e está marcada em azul. Na sequência, temos o talude com largura de 5, razão de declive de 2 e profundidade máxima de 3000, marcado em amarelo. E para finalizar, em verde, a área que representa a bacia oceânica, com profundidade máxima de 5000 e que se estende pelo restante da matriz.

Ao alterar seus valores, pode-se definir comportamentos específicos de acordo com o ambiente em que o agente robótico está inserido. Por exemplo, em um ambiente em que se precise de uma margem de segurança maior em relação às paredes do armazém, e considerando que o algoritmo favorece a navegação em águas profundas, como será visto adiante, poderia-se estender as plataformas continentais de modo a manter as águas profundas mais afastadas dos limites da matriz.

Para finalizar a primeira fase do algoritmo, precisa-se incorporar ao mapa calculado os obstáculos detectados pelo sensor LIDAR do robô. Como dito anteriormente, esses obstáculos são traduzidos em estruturas que representam ilhas no oceano. Na posição indicada pelo sensor, é calculado então uma elevação que parte da profundidade naquele ponto do mapa até 0. Essa elevação possui parâmetros que podem ser modificados de acordo com o ambiente, visando principalmente manter uma área de segurança ao redor dos obstáculos detectados, chamada na literatura de área de inflação. Esses parâmetros são a lista de aclive $[1, 3, 10, 50, 200, 400, 800, 1600, 3000, 3600]$, a área de inflação do obstáculo e a largura da ilha, e podem ser visto em detalhes da Figura 46. Na imagem, a área de inflação de obstáculo, em vermelho, é igual a 1 e a largura da ilha, em verde, é igual a 5. O ponto marcado em azul representa a coordenada do obstáculo detectado pelo sensor.

A fase seguinte do algoritmo, marcada em amarelo na Figura 44, representa o processo ondulatório explicado na seção 2.5. Seu objetivo é a geração de um segundo *grid*, de tamanho idêntico ao primeiro, mas que possui informações referentes ao resultado do deslocamento

Figura 47 – Recorte do mapa gerado pelo processo ondulatório. Obstáculos estão destacados em amarelo.



Fonte: Autoria própria (2023).

da onda ao navegar pelo *grid* de profundidades calculado na fase anterior. Para essa fase, o parâmetro utilizado como comprimento inicial da onda foi de 200. Ao término do processo, tem-se uma mapa que indica, para cada coordenada $[x,y]$, se o ponto corresponde a uma zona de ondas profundas (P), de transição (T), baixas (B) ou a arrebentação (A) em uma região de costa ou próximo a uma ilha. Um exemplo desse mapa pode ser visto na Figura 47.

Como será visto na seção seguinte, tanto o mapa de profundidades do relevo oceânico quanto o gerado pelo processo ondulatório terão influência direta no cálculo da trajetória.

3.4 Planejamento da trajetória com mapa de relevo oceânico

A última fase do algoritmo, em verde na Figura 44, é a aplicação do algoritmo A* modificado, utilizando os *grids* calculados nas fases anteriores. Conforme visto na Equação 8, o cálculo da função $f(v)$ é resultado da soma de duas funções, a primeira, $g(v)$, representando a distância da origem até o nó v e a segunda, $h(v)$, indicando a distância estimada de nó v até o destino.

Para calcular a parcela $g(v)$ da equação, o algoritmo considera que movimentos efetuados na horizontal ou na vertical tem custo de 10, enquanto movimentos diagonais tem custo de 14. Além disso, para cada movimento em que a profundidade diminua, essa distância é penalizada por um fator k. Quanto maior o fator aplicado, maior a tendência do algoritmo calcular uma rota que se mantenha por águas profundas. Para finalizar o cálculo da função, o algoritmo

considera também as informações sobre a ondulatória do mapa, presente no grid calculado na fase anterior. Para isso, os seguintes acréscimos são feitos:

1. Ondas profundas: Nenhuma penalidade.
2. Ondas de transição: Nenhuma penalidade.
3. Ondas de águas baixas: 10% de penalidade.
4. Arrebentação: 100% de penalidade.

Já em relação a função $h(v)$, a base de cálculo é a soma das distâncias X e Y faltantes para atingir o objetivo da rota. Além dessa distância, conhecida como Manhattan, é acrescida uma penalidade, também na forma de um fator k, parametrizado em 60%, caso o algoritmo identifique necessidade de efetuar curvas para atingir o alvo. Essa análise é feita de modo simples, basicamente verificando se o alvo está na mesma coordenada X, Y ou na mesma diagonal da célula atualmente sendo calculada. Esse fator é adicionado uma vez que rotas com muitas curvas, além de mais complexas, aumentam a demanda de recursos do agente robótico para serem executadas.

Após o término da fase verde da Figura 44, o algoritmo proposto apresenta a sua rota calculada. Porém, para dar sequência ao trabalho e validar o resultado obtido, é feita uma comparação com dois outros algoritmos:

1. A*: Algoritmo padrão, com função $g(v)$ igual a 10 ou 14, de acordo com a direção do movimento, e com a função $h(v)$ igual à distância de Manhattan.
2. Campos potenciais: Algoritmo apresentado na sessão anterior, parametrizado com campos atrativos e repulsivos conforme Figura 35

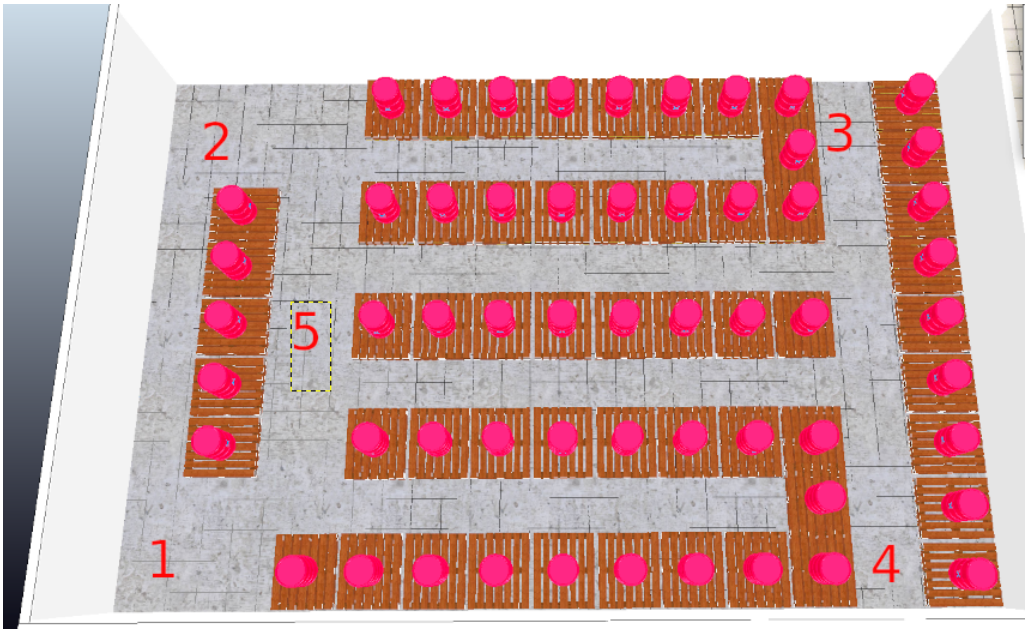
Os ensaios feitos para validar a proposta apresentada em comparação com esses algoritmos serão apresentados na Seção seguinte.

4 RESULTADOS DO MÉTODO DE RELEVO OCEÂNICO

Os ensaios para comparação dos algoritmos foram divididos em duas etapas, sendo que a primeira compara o método proposto com o algoritmo tradicional de campos potenciais da seção 2.3, e o segundo apresenta a comparação com o método A* demonstrado em 2.4.

Para todas as validações, visando gerar rotas sob as condições mais variadas, os pontos de origem do agente robótico foram definidos conforme Figura 48.

Figura 48 – Pontos de origem do agente robótico para execução dos ensaios



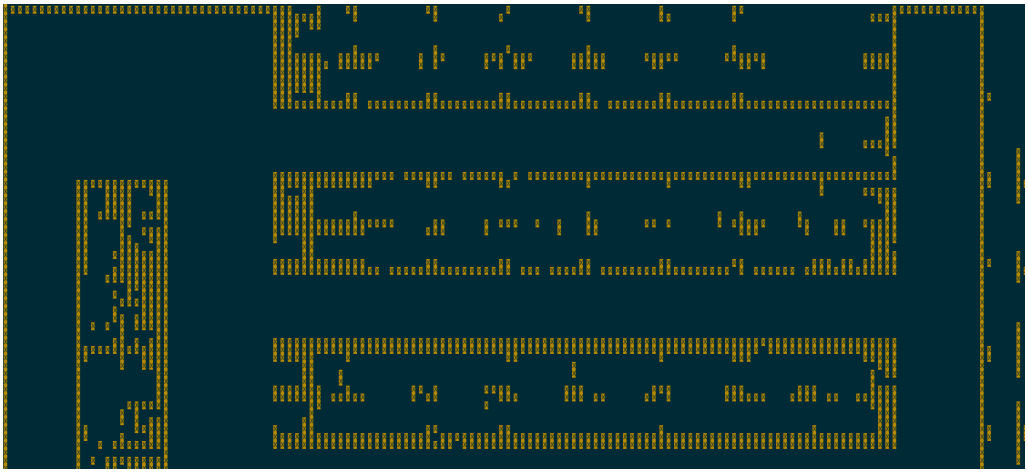
Fonte: Autoria própria (2023).

Além dos pontos de origem, outro aspecto comum para todos os ensaios é a matriz de obstáculos detectados pelo sensor LIDAR do agente robótico. Isso garante que todos os algoritmos tenham a mesma “visão de mundo” de onde estão operando. Importante ressaltar também que tal matriz não representa com perfeição todos os obstáculos presentes no cenário. Isso foi feito para garantir maior semelhança com um projeto no mundo real, onde podem ocorrer falhas de leitura ou mesmo no hardware embarcado no equipamento. Esses pontos de falha tiveram impacto direto em algumas rotas calculadas, principalmente com o algoritmo A* tradicional. Na Figura 49 pode-se reparar tais pontos de falha no sensoriamento:

4.1 Comparação com campos potenciais

Conforme visto na Seção 2.3, apesar de ser um método de implementação simples e de boa performance, o campo gerado por esse algoritmo pode apresentar problemas em ambientes que apresentem uma grande quantidade de obstáculos. Isso ocorre devido a geração de

Figura 49 – Recorte da matriz de obstáculos representando a metade superior do armazém. Pontos em amarelo representam os obstáculos. Inúmeras falhas de sensoriamento podem ser vistas.

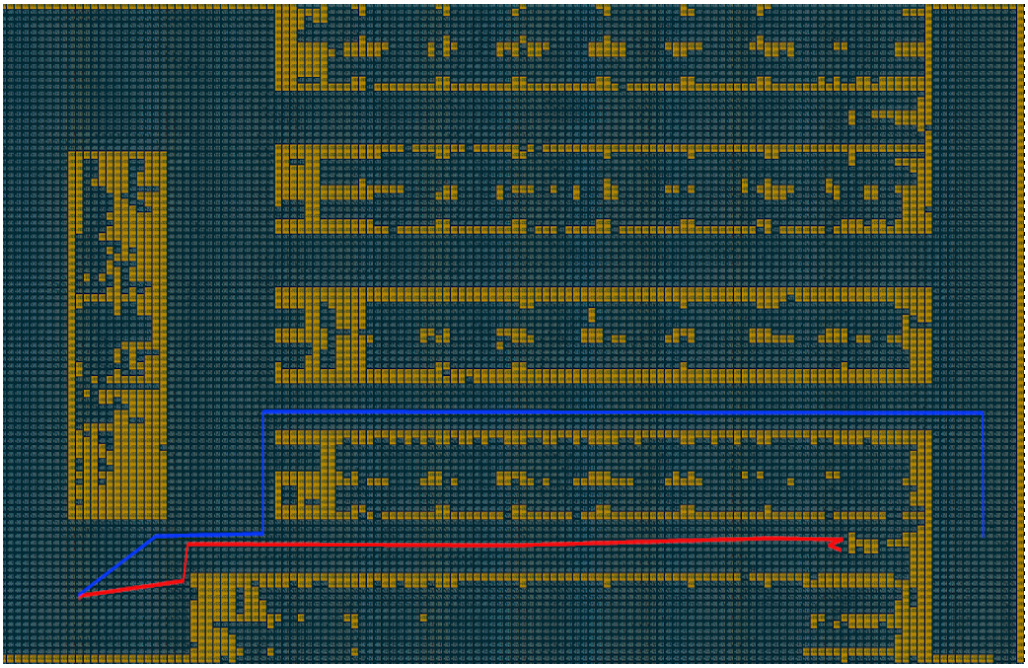


Fonte: Autoria própria (2023).

mínimos locais que podem direcionar o agente robótico para "becos sem saída", como exemplificado na Figura 37.

Considerando o cenário apresentado na Figura 48, o algoritmo de campos potenciais teve dificuldades particularmente quando a rota se iniciava nos pontos 1 ou 2 e tinha como alvo destinos próximos aos pontos 3 e 4. A tendência do algoritmo nesses casos foi de navegar até os mínimos locais situados dentro dos corredores sem saída nos cantos superior e inferior do armazém. Um exemplo de rota que terminou em um mínimo local pode ser visto na Figura 50.

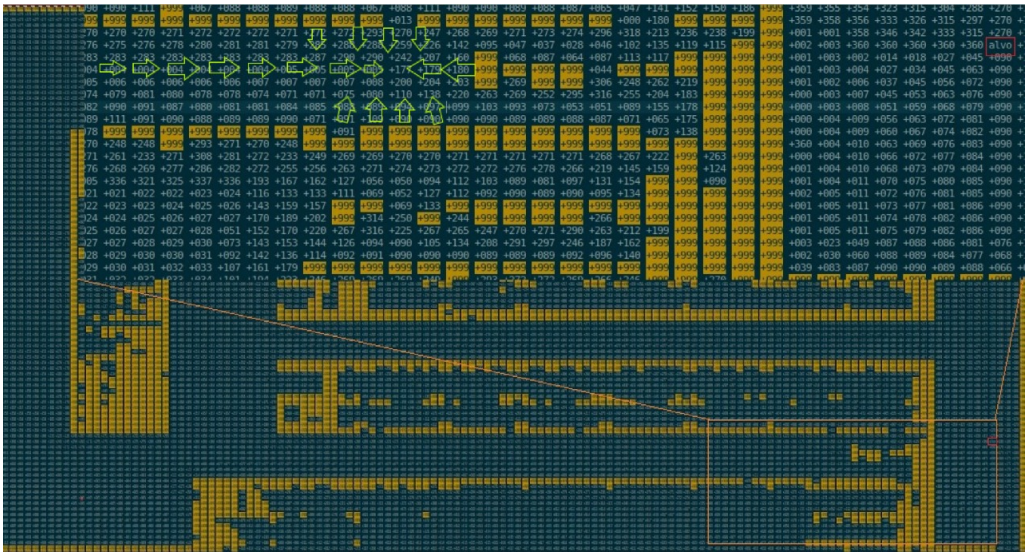
Figura 50 – Exemplo de rota iniciando na posição 1 e com alvo próximo à posição 4, presa em um mínimo local no final do corredor inferior. Campos potenciais em vermelho e algoritmo proposto em azul.



Fonte: Autoria própria (2023).

No cenário da Figura 50 o campo gerado foi capaz de direcionar com sucesso o agente robótico na direção do alvo, propositalmente posicionado atrás do corredor inferior, próximo à posição 4. No entanto, ao chegar ao final do corredor, a rota ficou presa em um mínimo local. Decompondo os vetores do campo potencial em ângulos na escala de 0 graus até 359 graus, pode-se ver que os ângulos passam a apontar para direções que acabam prendendo o robô no local, como pode ser visto na Figura 51

Figura 51 – Recorte da rota com ângulos gerados no ponto de ocorrência de um mínimo local na rota calculada. +999 representa um obstáculo.



Fonte: Autoria própria (2023).

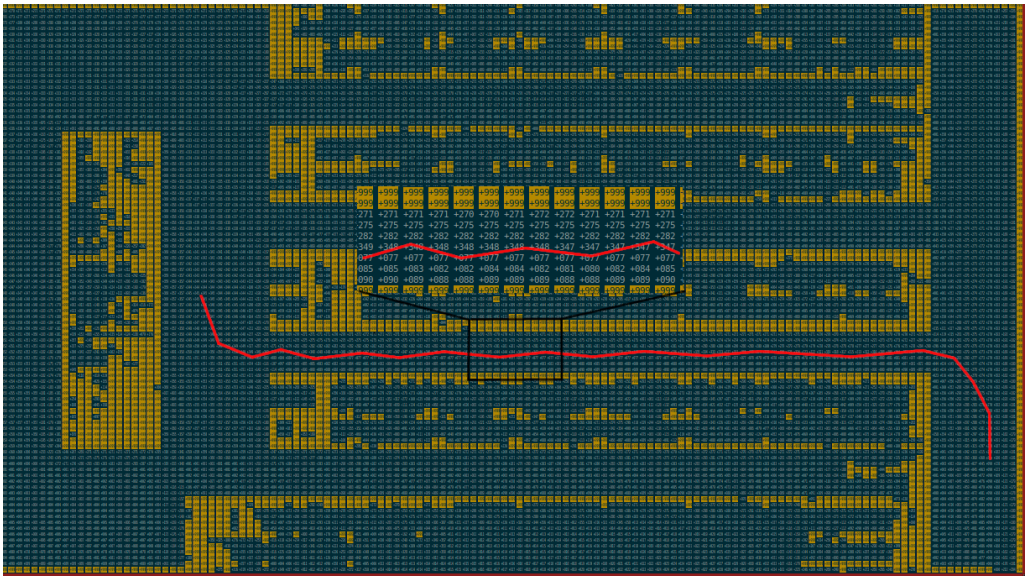
Outro aspecto que surgiu nas rotas calculadas com os campos potenciais foi a oscilação de direção em áreas de divisa com campos repulsivos dos obstáculos. Tal oscilação poderia causar inúmeras dificuldades para um agente robótico real, dada a quantidades de vezes em que mudanças de direção teriam que ser efetuadas em curtos espaços. Um exemplo de rota com tal oscilação pode ser vista na Figura 52.

4.2 Comparação com A*

A estrutura semelhante do algoritmo proposto e a implementação padrão do A* viabilizou o aumento do número de rotas simuladas e automatização da coleta dos resultados. Diferentemente do algoritmo de campos potenciais visto na seção anterior, o A* não apresenta o problema de mínimos locais nas rotas geradas. A comparação entre eles tomou como base os seguintes aspectos:

1. Diferença da distância percorrida, em pontos, da posição do agente até o alvo
2. Diferença entre o número de curvas encontradas na rota calculada
3. Diferença da soma dos graus de todas as curvas da rota

Figura 52 – Recorte da rota, em vermelho, com ângulos que gerariam oscilação na direção do agente robótico.



Fonte: Autoria própria (2023).

4. Diferença da distância mínima na rota para um obstáculo

5. Diferença do tempo de processamento da rota

Para cada um dos pontos de origem apresentados na Figura 48, foram simuladas 100 rotas, com pontos de destino aleatórios espalhados por toda a extensão do mapa. Para exemplificar melhor a coleta de resultados compilada na Tabela 2, será apresentado em detalhe a extração dos totais para uma dessas 500 rotas simuladas.

Tabela 1 – Totalizadores para rota número 68, com inicio no ponto 1, coordenada [9,9] e alvo na coordenada [75,78]. Distância euclidiana total de 95.48.

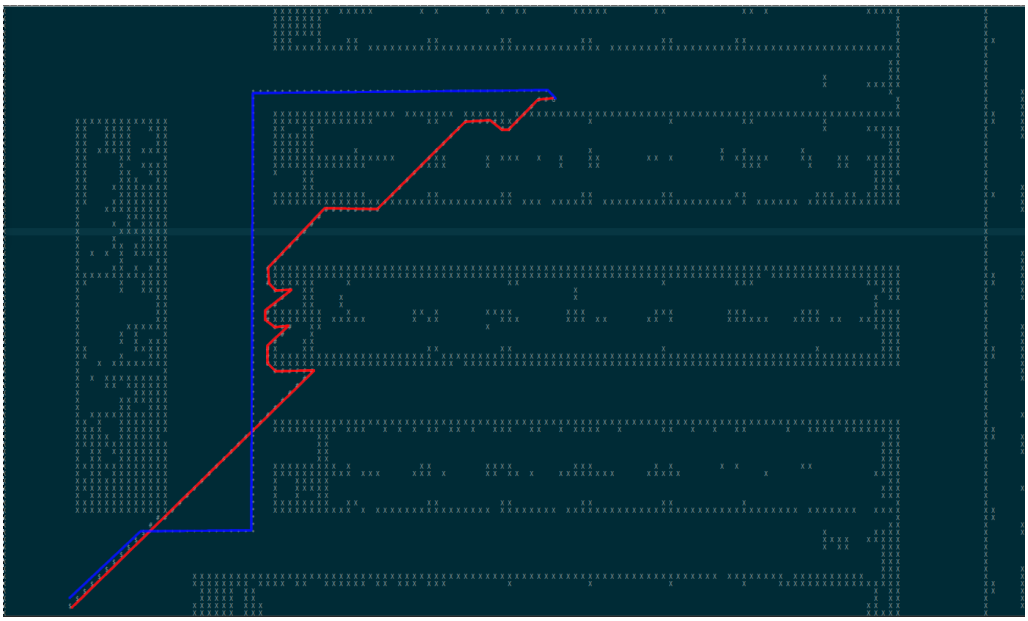
Métrica	A* oceano	A* tradicional	Diferença	Melhora em %
Pontos da rota	127	96	31	-32
Curvas	4	21	17	81
Soma graus	900	3105	2205	71
Tempo (ms)	162	69	-92	-134
Dist. min. para obstáculo	2	1	1	100

Fonte: Autoria própria (2023).

A rota número 68, apresentada na Tabela 1, é um exemplo onde, apesar do algoritmo proposto apresentar distância em pontos maior (cerca de 33%), ao analisar esse pontos gerados por cada um dos algoritmos, pode-se notar que o A* tradicional acabou sofrendo com a imprecisão simulada no sensor LIDAR, conforme comentado anteriormente. Outro indicador que mostrou o trabalho proposto inferior ao algoritmo de referência é o tempo de cálculo. Porém por se tratar de uma escala em milissegundos, isso não apresenta grande impacto na prática. Uma melhora significativa na quantidade de manobras que seriam necessárias para um agente

robótico executar a trajetória computada pode ser deduzida a partir da redução da quantidade de curvas (81%) e graus (71%). Por fim, o algoritmo proposto apresentou um aumento de 100% na distância mínima para obstáculos. Porém naturalmente quanto maior esse valor mais segura é a rota obtida, visto que esse é um dos objetivos da proposta.

Figura 53 – Rota de número 68, utilizada para exemplificar a metodologia e demonstrar falha no A* tradicional (em vermelho).



Fonte: Autoria própria (2023).

Ao analisar a Figura 53, fica nítido que a rota calculada pelo algoritmo proposto, em azul, apresenta maior segurança em relação à distância mínima para os obstáculos, bem como curvas mais facilmente executáveis por um robô. Apesar do custo em termos de tempo de processamento ser maior no algoritmo proposto, por se tratar de uma escala em milissegundos em uma rota calculada do lado do servidor, e não no agente robótico, tal diferença não traria grandes dificuldades em uma aplicação real, como já mencionado.

Ainda avaliando individualmente a rota de número 68, conforme Figura 53, pode-se notar que uma curva é feita logo antes do término da trajetória proposta. Avaliando o caminho como um todo, é natural imaginar que a penúltima curva, de 90 graus, antes de entrar no corredor, poderia ter sido feita com alguns pontos de antecedência, eliminando a necessidade desta última mudança na trajetória. Porém, esse comportamento reforça a característica do algoritmo proposto em se manter a maior quantidade possível de tempo afastado da estrutura de armazenagem. Caso a última curva não fosse feita, o agente robótico teria que percorrer uma distância maior navegando próximo ao obstáculo, reduzindo a segurança operacional. Outro aspecto que vale ressaltar é que o algoritmo optou por uma curva final de 45 graus, e não de 90, de modo a tornar o trajeto mais suave para a execução.

Os resultados compilados a partir da execução das 500 rotas, são vistos na Tabela 2. Essa tabela exibe as diferenças dos mesmos indicadores exemplificados na Tabela 1, porém agrupados por ponto de origem da rota. Ao analisar o compilado, pode-se notar a mesma ten-

dência vista ao analisar individualmente a rota de número 68 apresentada anteriormente, onde os caminhos calculados pelo algoritmo proposto apresentam mais pontos, porém com rotas mais seguras e com menos curvas, apesar de tempo de processamento maior. Os dados com todas as 500 rotas podem ser vistos nos apêndices A-E. Os valores entre parênteses representam a melhora em percentual, do mesmo modo que apresentado na Tabela 1.

Tabela 2 – Tabela com os resultados compilados das 500 rotas simuladas, 100 para cada ponto de inicio

Ponto inicio	1	2	3	4	5
Σ Pontos A* Oceano	8690	9326	13206	12830	6974
Σ Pontos A* Tradicional	8376	10031	12063	12093	6351
Melhora Pontos	-314 (-3%)	705 (7%)	-1143 (-9%)	-737 (-6%)	-623 (-9%)
Σ Curvas A* Oceano	301	307	516	472	259
Σ Curvas A* Tradicional	734	870	1410	869	571
Melhora Curvas	433 (59%)	563 (65%)	894 (63%)	397 (46%)	312 (55%)
Σ Graus A* Oceano	76140	26280	38610	35235	27180
Σ Graus A* Tradicional	123255	66420	80595	55395	64755
Melhora Graus	47115 (38%)	40140 (60%)	41985 (52%)	20160 (36%)	37575 (58%)
Σ Tempo A* Oceano (ms)	6467	3466	27542	20722	2055
Σ Tempo A* Tradicional (ms)	5637	15621	15268	7038	4481
Melhora Tempo	-838 (-15%)	12123 (78%)	-12253 (-80%)	-13669 (-194%)	2419 (54%)
Σ Dist. mínima A* Oceano	304	292	292	283	292
Σ Dist. mínima A* Tradicional	190	193	166	157	149
Melhora Dist. mínima	114 (60%)	99 (51%)	126 (76%)	126 (80%)	143 (96%)

Fonte: Autoria própria (2023).

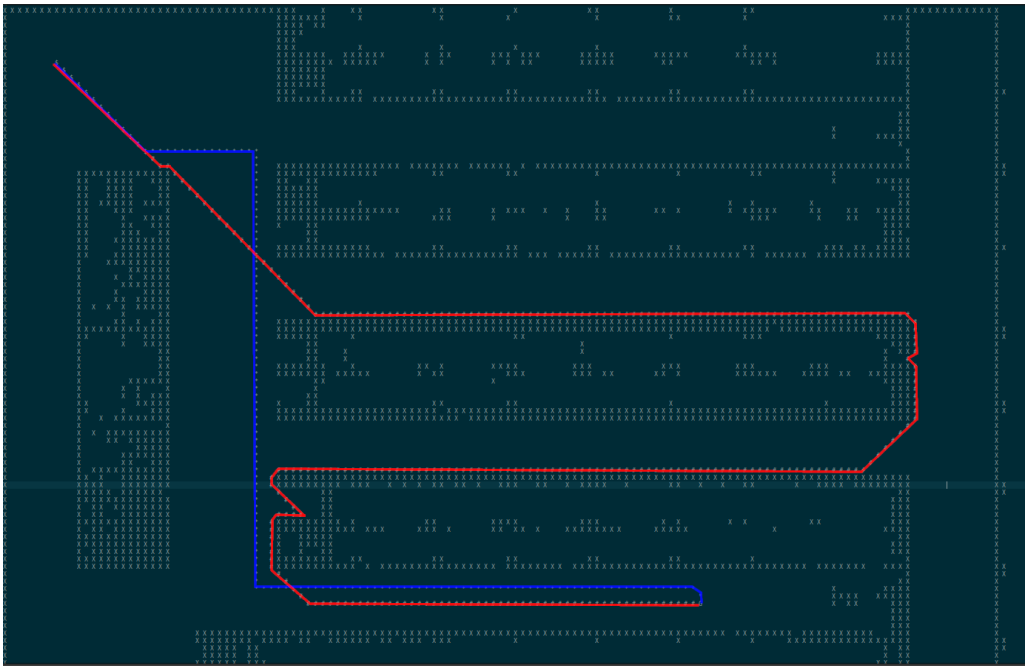
Avaliando a Tabela 2, pode-se notar uma quebra de tendência nas rotas com início em 2. As principais discrepâncias ocorreram na diferença total de pontos e no tempo de cálculo. Avaliando em detalhes as 100 rotas geradas com esse ponto de inicio, contatou-se que o algoritmo A* tradicional teve dificuldades para calcular uma sequência de rotas mais longas, que tinham como destino o corredor localizado na parte inferior do cenário. A geração dessa rota mais longa influenciou diretamente na diferença de pontos e também no tempo total. Um exemplo de rota com essas características pode ser visto na Figura 54. Todas as rotas iniciadas em 2 podem ser vistas no Apêndice B.

Em todas as 500 simulações de rotas, tanto o algoritmo proposto quanto o A* de conseguiram encontrar caminhos da origem até o destino. No entanto, conforme pontuado anteriormente, algumas imprecisões foram inseridas durante as leituras efetuadas pelo sensor LIDAR do agente robótico, como visto na Figura 49, de modo a possibilitar a comparação dos dois algoritmos no quesito de quantidade de rotas inválidas geradas.

A abordagem proposta apresenta desempenho consistentemente melhor que o algoritmo A*, especialmente para as rotas partindo das origens 1, 3 e 5, como pode ser visto na Tabela 3.

Para finalizar a apresentação dos resultados obtidos, foi feita a mesma comparação apresentada anteriormente na Tabela 2, porém desconsiderando todas as amostras em que ao

Figura 54 – Exemplo de cenário em que o A* tradicional (em vermelho) teve dificuldades para calcular uma rota eficiente, causando distorção nos resultados compilados.



Fonte: Autoria própria (2023).

Tabela 3 – Resultado da comparação de rotas inválidas entre o algoritmo proposto e o A*.

Ponto inicio	Falhas A*	Falhas A* Oceano	Melhora em %
1	15	0	100
2	5	5	0
3	76	14	81
4	13	13	0
5	21	0	100
Total	130	32	75

Fonte: Autoria própria (2023).

menos um dos algoritmos calculou uma rota impossível de ser executada. Como pode ser visto na Tabela 4, os resultados mantiveram a mesma tendência apresentada anteriormente, mesmo após a remoção das rotas inválidas.

Tabela 4 – Tabela com os resultados compilados desconsiderando rotas com falha.

Ponto inicio	1	2	3	4	5
Σ Pontos A* Oceano	6833	8692	1903	10886	5140
Σ Pontos A* Tradicional	6354	9361	1804	9964	4819
Melhora Pontos	-479 (-7%)	669 (7%)	-99 (-5%)	-922 (-9%)	-321 (-6%)
Σ Curvas A* Oceano	240	267	58	313	190
Σ Curvas A* Tradicional	371	694	108	566	383
Melhora Curvas	131 (35%)	427 (61%)	50 (46%)	253 (44%)	193 (50%)
Σ Graus A* Oceano	59895	19305	3690	22905	14400
Σ Graus A* Tradicional	73575	39420	5940	28665	27675
Melhora Graus	1368 (18%)	20115 (51%)	2250 (37%)	5760 (20%)	13275 (48%)
Σ Tempo A* Oceano (ms)	4040	3084	8069	13738	1254
Σ Tempo A* Tradicional (ms)	3535	15523	5247	4246	4237
Melhora Tempo	-515 (-14%)	12403 (80%)	-2829 (-48%)	-9484 (-223%)	2968 (70%)
Σ Dist. mínima A* Oceano	263	287	99	264	229
Σ Dist. mínima A* Tradicional	175	188	91	142	126
Melhora Dist. mínima	88 (50%)	99 (53%)	8 (8%)	122 (85%)	103 (81%)

Fonte: Autoria própria (2023).

.

5 CONCLUSÃO

O objetivo do trabalho foi desenvolver um sistema capaz de abstrair as características do relevo de fundo oceânico para a modelagem de um mapa e posterior utilização no cálculo de trajetórias entre pontos de origem e destino dentro de um armazém.

O estado da arte nas áreas de mapeamento e planejamento de trajetórias foi apresentado, tendo como foco pesquisas relacionadas aos algoritmos que utilizam o método A* e Campos Potenciais. Ainda no referencial teórico foram feitas as considerações buscando os pontos de interseção entre essas pesquisas e o algoritmo proposto neste trabalho. Por fim, maiores detalhes sobre os métodos de campos potenciais, A* e as características do relevo oceânico são explicados encerrando o capítulo.

O desenvolvimento da proposta foi apresentado no capítulo 3, e inclui todos os softwares de apoio necessários para atingir os objetivos inicialmente propostos, bem como a visão geral do algoritmo e o detalhamento do processo que monta o mapa baseado no relevo oceânico e o cálculo da trajetória.

No capítulo 4 os resultados obtidos com o presente trabalho foram apresentados e comparados com os dois métodos de referência. Os indicadores coletados são positivos quando equiparados com os algoritmos de campos potenciais e A*, inclusive com alguns cenários específicos mostrando o algoritmo proposto calculando rotas significativamente mais curtas e seguras (Figura 54).

Conforme visto ainda no capítulo de resultados, um dos principais problemas enfrentado ao mapear rotas em ambientes complexos usando campos potenciais é a incidência de mínimos locais. Não foi diferente no estudo apresentado, porém isso não inviabiliza o uso de tal algoritmo nesses ambientes. Técnicas podem ser usadas para identificar e informar ao algoritmo que ele está preso em um mínimo local, como visto em pesquisas do capítulo 2 Szczepanski, Bereit e Tarczewski (2021), de modo que as demais características positivas do algoritmo possam ser aplicadas com sucesso. Um exemplo de tal característica é a possibilidade do mesmo campo poder ser compartilhado e utilizado por inúmeros agentes simultaneamente, facilitando o processo de decisão sobre qual agente se encarregaria de executar tal rota. Tal característica também pode ser vista no algoritmo proposto, no que diz respeito a etapa de montagem do mapa anterior ao cálculo da rota. O mapa de relevo oceânico pode ser facilmente compartilhado entre inúmeros agentes, visto que as características do ambiente não mudam de acordo com a coordenada alvo de cada robô.

Outro problema que ocorreu ao utilizar o campo potencial para traçar a trajetória foi a oscilação em áreas de incidência simultânea de forças de atração e repulsão. Apesar de não ser impeditivo, essa é uma característica que deve ser atenuada para viabilizar a navegação no mundo real. Esse problema também é recorrente do algoritmo, e técnicas de pós-processamento do campo podem ser aplicadas para suavizar a rota obtida, como visto em Wang *et al.* (2022) e Xiang *et al.* (2022).

Quando comparado ao algoritmo de referência A*, o estudo proposto apresentou resultados interessantes, que seriam de grande utilidade na prática. Primeiramente no que diz respeito a consistência das rotas geradas quando o mapa do ambiente apresenta falhas de sensoriamento. Como vimos, o A* de referência apresentou uma quantidade significativamente maior de caminhos que não poderiam ser executados, conforme visto na Tabela 3, sendo que em alguns cenários uma melhora de 100% foi alcançada. Outro aspecto que pode ser visto nas tabelas 2 e 4 e que representa beneficio no mundo real é a redução no número de curvas necessárias para as rotas atingirem os alvos. Além de simplificar a operação do agente robótico reduzindo o número de mudanças de trajetória, isso também é positivo quando visto sob o aspecto de segurança, especialmente se o robô estiver carregando cargas sensíveis. A Tabela 2 mostra um desempenho consistentemente melhor nesse aspecto, com uma média de aproximadamente 58% menos curvas, e diferença média em graus de 48%.

Ainda sob a ótica da segurança, outra característica positiva da proposta apresentada é a distância mínima maior de obstáculos durante a trajetória. Navegar mais longe de obstáculos pode ter relação direta para evitar acidentes, e novamente, dependendo da natureza da carga carregada pelo robô, isso pode ter consequências ainda mais significativas. Como mostrado na Tabela 2, no cenário 5 o presente trabalho obteve uma melhora de 96% em relação ao algoritmo de referência, tendo alcançado um progresso médio de aproximadamente 73%.

Para melhorar a precisão do algoritmo quando os dados vindos do sensoriamento apresentam imprecisão, poderia-se adicionar uma fase de pré-cálculo da rota, responsável por adicionar áreas de inflação maiores quando ocorre suspeita de perda de informação do sensoriamento. Outra opção seria a alteração dos parâmetros do algoritmo proposto visando aumentar o custo da rota ao entrar em áreas mais próximas à costa.

Como trabalho futuro, poderia-se otimizar a fase de parametrização, atualmente estática, através de um algoritmo genético, visando o melhor acerto possível para o ambiente em que o agente esteja operando. Outra possibilidade que se abre é a combinação do algoritmo proposto com o de campos potenciais, visando o compartilhamento das informações do mapa em um sistema multi-agente utilizando a proposta apresentada para eliminar o problema dos mínimos locais.

REFERÊNCIAS

AUTOWARE. 2022. https://github.com/autowarefoundation/autoware.

CHIANG, H.-T. *et al.* Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments. In: IEEE. **2015 IEEE international conference on robotics and automation (ICRA)**. [*S.l.*], 2015. p. 2347–2354.

DUAN, L. M. Path planning for batch picking of warehousing and logistics robots based on modified a* algorithm. **International Journal of Online and Biomedical Engineering (iJOE)**, Kassel University Press GmbH, v. 14, p. 176–192, 11 2018. ISSN 2626-8493. Disponível em: https://online-journals.org/index.php/i-joe/article/view/9527.

DUCHON, F. *et al.* Path planning with modified a star algorithm for a mobile robot. **Procedia Engineering**, No longer published by Elsevier, v. 96, p. 59–69, 1 2014. ISSN 1877-7058.

ERKE, S. *et al.* An improved a-star based path planning algorithm for autonomous land vehicles. **International Journal of Advanced Robotic Systems**, SAGE Publications Inc., v. 17, 10 2020. ISSN 17298814. Disponível em: https://journals.sagepub.com/doi/full/10.1177/1729881420962263.

GOODRICH, M. A. Potential fields tutorial. **Class Notes**, v. 157, 2002.

GUERRA, M. *et al.* Avoiding local minima in the potential field method using input-to-state stability. **Control Engineering Practice**, Elsevier, v. 55, p. 174–184, 2016.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A formal basis for the heuristic determination of minimum cost paths. **IEEE Transactions on Systems Science and Cybernetics**, v. 4, n. 2, p. 100–107, 1968.

INITIATIVE, O. S. **Open Source Initiative**. 2022. Disponível em: https://opensource.org/licenses/BSD-3-Clause.

KHATIB, O. Real-time obstacle avoidance for manipulators and mobile robots. **In IEEE Int. Conf. on Robotics and Automation**, 1985.

KOUBAA, A. *et al.* Background on artificial intelligence algorithms for global path planning. **Studies in Computational Intelligence**, Springer Verlag, v. 772, p. 13–51, 2018. ISSN 1860949X. Disponível em: https://link.springer.com/chapter/10.1007/978-3-319-77042-0_2.

LAVALLE, S. M. Planning algorithms / motion planning. In: ____. [*s.n.*], 2006. Disponível em: http://lavalle.pl/planning/.

LIN, R.; HUANG, H.; LI, M. An automated guided logistics robot for pallet transportation. **Assembly Automation**, Emerald Group Holdings Ltd., v. 41, p. 45–54, 2 2021. ISSN 01445154.

MAC, T. T. *et al.* Heuristic approaches in robot path planning: A survey. **Robotics and Autonomous Systems**, North-Holland, v. 86, p. 13–28, 12 2016. ISSN 0921-8890.

MATOUI, F.; BOUSSAID, B.; ABDELKRIM, M. N. Local minimum solution for the potential field method in multiple robot motion planning task. In: IEEE. **2015 16th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)**. [*S.l.*], 2015. p. 452–457.

PARKER, L. E. Path planning and motion coordination in multiple mobile robot teams. **Encyclopedia of complexity and system science**, Citeseer, p. 5783–5800, 2009.

PETERSON, J. L. Petri nets. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 9, n. 3, p. 223–252, 1977.

PRADHAN, S. K. *et al.* Potential field method to navigate several mobile robots. **Applied Intelligence**, v. 25, p. 321–333, 12 2006. ISSN 0924669X.

RASEKHIPOUR, Y. *et al.* A potential field-based model predictive path-planning controller for autonomous road vehicles. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, v. 18, n. 5, p. 1255–1267, 2016.

REDHAT. **Containers vs VMs**. 2022. Disponível em: https://www.redhat.com/en/topics/containers/containers-vs-vms.

ROHMER, E.; SINGH, S. P. N.; FREESE, M. Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. In: **Proc. of The International Conference on Intelligent Robots and Systems (IROS)**. [*S.l.*: *s.n.*], 2013. Www.coppeliarobotics.com.

ROS. **ROS**. 2022. Disponível em: https://www.ros.org/core-components/.

SABUDIN, E.; OMAR, R.; MELOR, C. C. K. Potential field methods and their inherent approaches for path planning. **ARPN Journal of Engineering and Applied Sciences**, Asian Research Publishing Network (ARPN), v. 11, n. 18, p. 10801–10805, 2016.

SHIAU, J.-Y.; LIAO, T.-C. Developing an order picking policy for economical packing. In: IEEE. **Proceedings of 2013 IEEE International Conference on Service Operations and Logistics, and Informatics**. [*S.l.*], 2013. p. 387–392.

SZCZEPANSKI, R.; BEREIT, A.; TARCZEWSKI, T. Efficient local path planning algorithm using artificial potential field supported by augmented reality. **Energies 2021, Vol. 14, Page 6642**, Multidisciplinary Digital Publishing Institute, v. 14, p. 6642, 10 2021. ISSN 1996-1073. Disponível em: https://www.mdpi.com/1996-1073/14/20/6642/htmhttps://www.mdpi.com/1996-1073/14/20/6642.

TESSLER, M. G.; MAHIQUES, M. M. d. Processos oceânicos e a fisiografia dos fundos marinhos. In: ____. [*S.l.*]: Oficina de textos, 2000.

TINGBIN, C. *et al.* Research on the dynamic target distribution path planning in logistics system based on improved artificial potential field method-fish swarm algorithm. **Proceedings of the 30th Chinese Control and Decision Conference, CCDC 2018**, Institute of Electrical and Electronics Engineers Inc., p. 4388–4391, 7 2018.

TIRUMALAPUDI, R.; VEDARAJ, I. Mobile robots obstacle avoidance and shortest path planning using algorithms -a review. 06 2020.

WANG, P. *et al.* Obstacle avoidance path planning design for autonomous driving vehicles based on an improved artificial potential field algorithm. **Energies**, MDPI AG, v. 12, 2019. ISSN 19961073.

WANG, Q.; MCINTOSH, R.; BRAIN, M. A new-generation automated warehousing capability. **International Journal of Computer Integrated Manufacturing**, Taylor & Francis, v. 23, n. 6, p. 565–573, 2010.

WANG, R. *et al.* Application of a* algorithm in intelligent vehicle path planning. **Mathematical Models in Engineering**, v. 8, 08 2022.

XIANG, D. *et al.* Combined improved a* and greedy algorithm for path planning of multi-objective mobile robot. **Scientific Reports 2022 12:1**, Nature Publishing Group, v. 12, p. 1–12, 8 2022. ISSN 2045-2322. Disponível em: https://www.nature.com/articles/s41598-022-17684-0.

ZAINAL, N.; ZAIN, A. M.; SHARIF, S. Overview of artificial fish swarm algorithm and its applications in industrial problems. **Applied Mechanics and Materials**, Trans Tech Publications Ltd, v. 815, p. 253–257, 11 2015. ISSN 1662-7482. Disponível em: https://www.scientific.net/AMM.815.253.

ZHANG, Q. *et al.* **LNAI 7996 - Dynamic Obstacle-Avoiding Path Planning for Robots Based on Modified Potential Field Method**. 2013. 332-342 p.

APÊNDICE A – Resultados detalhados. Origem ponto 1

Amostra	Origem	Destino	Distância	Pontos proposta	Pontos A*	Curvas Proposta	Curvas A*	Soma º Proposta	Soma º A*	Tempo Proposta	Tempo A*	Dist. min. Proposta	Dist. min. A*
1	[9,9]	[129,67]	133,28	169	140	4	8	1125	1935	120	16	3	1
2	[9,9]	[55,16]	46,53	47	47	1	1	315	315	2	8	3	3
3	[9,9]	[43,80]	78,72	95	92	5	24	1485	2295	21	19	3	1
4	[9,9]	[57,38]	56,08	67	49	5	3	1485	945	10	9	3	1
5	[9,9]	[3,37]	28,64	29	29	1	1	45	45	1	7	3	3
6	[9,9]	[129,32]	122,18	144	122	4	9	945	1755	194	14	3	1
7	[9,9]	[48,17]	39,81	40	40	1	1	315	315	2	8	4	4
8	[9,9]	[48,79]	80,13	100	97	3	27	855	4275	39	28	3	1
9	[9,9]	[86,39]	82,64	97	78	5	3	1215	945	35	10	3	1
10	[9,9]	[4,47]	38,33	39	39	1	1	45	45	2	8	4	4
11	[9,9]	[126,50]	123,98	149	123	4	8	1125	1935	85	11	3	1
12	[9,9]	[51,18]	42,95	43	43	1	1	315	315	2	9	4	4
13	[9,9]	[61,81]	88,81	114	90	5	20	1215	2295	94	44	3	1
14	[9,9]	[44,36]	44,20	53	36	3	3	855	945	3	7	2	1
15	[9,9]	[4,54]	45,28	46	46	1	1	45	45	2	8	4	4
16	[9,9]	[129,88]	143,67	190	161	4	8	1125	1935	98	23	3	1
17	[9,9]	[98,15]	89,20	91	90	4	1	900	315	5	10	3	2
18	[9,9]	[60,78]	85,80	112	88	4	20	900	2790	67	34	2	1
19	[9,9]	[80,38]	76,69	90	72	5	3	1485	945	28	10	3	1
20	[9,9]	[4,84]	75,17	76	76	1	1	45	45	7	10	4	4
21	[9,9]	[125,19]	116,43	153	321	4	21	945	3465	504	958	3	1
22	[9,9]	[92,18]	83,49	84	84	1	1	315	315	6	9	4	4
23	[9,9]	[42,80]	78,29	94	90	5	24	1485	2295	15	18	3	1
24	[9,9]	[67,39]	65,30	78	59	5	3	1215	945	18	10	3	1
25	[9,9]	[3,68]	59,30	60	60	1	1	45	45	2	11	3	3
26	[9,9]	[128,86]	141,74	187	159	4	8	1125	1935	85	20	3	1
27	[9,9]	[47,17]	38,83	39	39	1	1	315	315	2	8	4	4
28	[9,9]	[40,80]	77,47	92	84	5	24	1485	1980	13	14	3	1
29	[9,9]	[65,37]	62,61	75	57	3	3	855	945	14	9	3	1
30	[9,9]	[4,60]	51,24	52	52	1	1	45	45	3	10	4	4
31	[9,9]	[127,81]	138,23	181	154	4	8	1125	1935	86	20	3	1
32	[9,9]	[89,16]	80,31	81	81	1	1	315	315	6	11	3	3
33	[9,9]	[56,81]	85,98	109	91	5	21	1215	2385	49	33	3	1
34	[9,9]	[75,38]	72,09	85	67	5	3	1485	945	27	9	3	1
35	[9,9]	[3,26]	18,03	18	18	1	1	45	45	0	7	3	3
36	[9,9]	[129,78]	138,42	180	151	4	8	1125	1935	87	20	3	1
37	[9,9]	[76,16]	67,36	68	68	1	1	315	315	4	9	3	3
38	[9,9]	[36,81]	76,90	89	74	5	3	1215	675	10	9	3	1
39	[9,9]	[47,36]	46,62	56	39	3	3	855	945	5	7	2	1
40	[9,9]	[4,17]	9,43	9	9	1	1	45	45	0	6	4	4
41	[9,9]	[126,34]	119,64	139	118	4	7	945	1665	132	15	3	1
42	[9,9]	[39,17]	31,05	31	31	1	1	315	315	1	7	4	4
43	[9,9]	[82,80]	101,83	134	193	5	27	1485	3285	230	356	3	1
44	[9,9]	[38,37]	40,31	48	30	3	2	855	630	2	6	3	1
45	[9,9]	[3,11]	6,32	7	7	1	1	45	45	0	6	3	3
46	[9,9]	[127,15]	118,15	161	330	10	22	2565	3105	746	938	3	1
47	[9,9]	[93,15]	84,21	86	85	4	1	900	315	10	12	2	2
48	[9,9]	[99,79]	114,02	151	199	3	26	855	3915	317	275	3	1
49	[9,9]	[32,38]	37,01	43	31	2	3	585	675	2	7	3	1
50	[9,9]	[4,33]	24,52	25	25	1	1	45	45	0	7	4	4

Amostra	Origem	Destino	Distância	Pontos proposta	Pontos A*	Curvas Proposta	Curvas A*	Soma º Proposta	Soma º A*	Tempo Proposta	Tempo A*	Dist. min. Proposta	Dist. min. A*
51	[9,9]	[127,47]	123,97	147	120	4	8	1125	1935	83	14	3	1
52	[9,9]	[72,15]	63,29	65	64	4	1	900	315	5	9	2	2
53	[9,9]	[66,81]	91,83	119	90	5	19	1215	2295	112	52	3	1
54	[9,9]	[37,39]	41,04	48	32	5	3	1215	675	4	7	3	1
55	[9,9]	[3,88]	79,23	80	80	1	1	45	45	5	14	3	3
56	[9,9]	[128,59]	129,08	160	132	4	8	1125	1935	66	13	3	1
57	[9,9]	[32,15]	23,77	24	24	1	1	315	315	1	7	2	2
58	[9,9]	[30,81]	75,00	84	74	2	3	585	675	7	11	3	1
59	[9,9]	[43,39]	45,34	54	35	5	3	1215	945	5	7	3	1
60	[9,9]	[3,36]	27,66	28	28	1	1	45	45	1	7	3	3
61	[9,9]	[125,74]	132,97	172	147	4	8	1125	1935	74	15	3	1
62	[9,9]	[59,15]	50,36	52	51	4	1	900	315	2	7	2	2
63	[9,9]	[100,81]	116,04	153	198	5	31	1215	4185	359	263	3	1
64	[9,9]	[31,39]	37,20	43	32	2	3	585	675	2	6	3	1
65	[9,9]	[4,18]	10,30	10	10	1	1	45	45	0	6	4	4
66	[9,9]	[127,24]	118,95	150	143	4	16	945	2655	509	815	3	1
67	[9,9]	[33,16]	25,00	25	25	1	1	315	315	1	6	3	3
68	[9,9]	[75,78]	95,48	127	96	4	21	900	3105	162	69	2	1
69	[9,9]	[76,37]	72,62	86	68	3	3	855	945	29	11	3	1
70	[9,9]	[3,44]	35,51	36	36	1	1	45	45	2	8	3	3
71	[9,9]	[125,42]	120,60	140	117	4	8	1125	2205	123	14	3	1
72	[9,9]	[81,17]	72,44	73	73	1	1	315	315	4	8	4	4
73	[9,9]	[46,79]	79,18	98	99	3	27	855	4275	24	35	3	1
74	[9,9]	[44,37]	44,82	54	36	3	3	855	945	4	7	3	1
75	[9,9]	[4,22]	13,93	14	14	1	1	45	45	0	6	4	4
76	[9,9]	[128,52]	126,53	153	125	4	8	1125	1935	78	13	3	1
77	[9,9]	[48,15]	39,46	41	40	4	1	900	315	2	6	2	2
78	[9,9]	[86,80]	104,74	138	197	5	27	1485	3285	237	302	3	1
79	[9,9]	[34,37]	37,54	44	30	2	3	585	675	1	10	3	1
80	[9,9]	[3,64]	55,33	56	56	1	1	45	45	3	10	3	3
81	[9,9]	[128,35]	121,81	139	120	5	3	945	945	111	12	3	1
82	[9,9]	[78,16]	69,35	70	70	1	1	315	315	5	9	3	3
83	[9,9]	[47,78]	78,77	98	98	3	27	855	4275	24	34	2	1
84	[9,9]	[67,36]	63,98	77	59	4	3	900	945	17	8	2	1
85	[9,9]	[4,15]	7,81	7	7	1	1	45	45	0	6	4	4
86	[9,9]	[129,29]	121,66	147	136	4	12	945	1935	268	16	3	1
87	[9,9]	[61,18]	52,77	53	53	1	1	315	315	3	8	4	4
88	[9,9]	[92,80]	109,22	144	203	5	24	1485	3105	264	318	3	1
89	[9,9]	[37,39]	41,04	48	32	5	3	1215	675	3	6	3	1
90	[9,9]	[3,79]	70,26	71	71	1	1	45	45	6	10	3	3
91	[9,9]	[128,55]	127,58	156	128	4	8	1125	1935	98	12	3	1
92	[9,9]	[53,16]	44,55	45	45	1	1	315	315	2	7	3	3
93	[9,9]	[55,78]	82,93	106	87	3	20	855	2295	49	30	2	1
94	[9,9]	[54,37]	53,00	64	46	3	3	855	945	7	8	3	1
95	[9,9]	[3,78]	69,26	70	70	1	1	45	45	7	9	3	3
96	[9,9]	[129,43]	124,72	145	121	4	8	1125	2205	73	11	3	1
97	[9,9]	[89,16]	80,31	81	81	1	1	315	315	4	9	3	3
98	[9,9]	[102,79]	116,40	154	196	3	26	855	3915	400	229	3	1
99	[9,9]	[84,38]	80,41	94	76	5	3	1485	945	26	10	3	1
100	[9,9]	[3,49]	40,45	41	41	1	1	45	45	2	7	3	3

APÊNDICE B – Resultados detalhados. Origem ponto 2

Amostra	Origem	Destino	Distância	Pontos proposta	Pontos A*	Curvas Proposta	Curvas A*	Soma º Proposta	Soma º A*	Tempo Proposta	Tempo A*	Dist. min. Proposta	Dist. min. A*
1	[7,90]	[128,35]	132,91	165	137	4	5	315	225	35	21	3	1
2	[7,90]	[84,16]	106,79	140	280	4	18	315	945	168	1480	3	1
3	[7,90]	[75,81]	68,59	69	69	1	1	45	45	3	9	4	4
4	[7,90]	[89,38]	97,10	122	164	5	10	315	495	76	383	3	1
5	[7,90]	[3,85]	6,40	6	6	1	1	45	45	0	6	3	3
6	[7,90]	[128,60]	124,66	139	122	5	3	315	135	18	16	3	1
7	[7,90]	[59,16]	90,44	115	111	4	23	315	1485	45	134	3	1
8	[7,90]	[34,80]	28,79	28	28	1	1	45	45	0	7	5	5
9	[7,90]	[44,39]	63,01	77	67	3	15	225	945	10	17	3	1
10	[7,90]	[3,48]	42,19	43	43	1	1	45	45	2	7	3	3
11	[7,90]	[128,61]	124,43	139	122	3	3	225	135	17	19	3	1
12	[7,90]	[94,17]	113,57	148	290	5	18	315	945	184	1173	3	1
13	[7,90]	[31,80]	26,00	25	25	1	1	45	45	1	6	5	5
14	[7,90]	[69,36]	82,22	105	92	4	15	315	945	36	187	2	1
15	[7,90]	[4,20]	70,06	71	71	1	1	45	45	5	9	4	4
16	[7,90]	[128,39]	131,31	161	133	4	5	315	225	25	14	3	1
17	[7,90]	[56,17]	87,92	110	108	5	23	315	1485	45	109	3	1
18	[7,90]	[96,79]	89,68	90	90	1	1	45	45	6	12	3	3
19	[7,90]	[74,39]	84,20	107	97	3	15	225	945	38	270	3	1
20	[7,90]	[3,24]	66,12	67	67	1	1	45	45	5	10	3	3
21	[7,90]	[126,32]	132,38	166	140	4	5	315	225	22	16	3	1
22	[7,90]	[40,18]	79,20	93	87	5	24	315	1440	12	13	3	1
23	[7,90]	[68,78]	62,17	63	62	4	1	630	45	3	9	2	2
24	[7,90]	[31,38]	57,27	65	54	2	3	135	135	4	10	3	1
25	[7,90]	[4,43]	47,10	48	48	1	1	45	45	2	8	4	4
26	[7,90]	[128,36]	132,50	164	136	4	5	315	225	19	15	3	1
27	[7,90]	[99,16]	118,07	155	295	4	18	315	945	209	1299	3	1
28	[7,90]	[67,78]	61,19	62	61	4	1	630	45	3	8	2	2
29	[7,90]	[88,36]	97,35	124	165	4	10	315	495	67	407	2	1
30	[7,90]	[4,47]	43,10	44	44	1	1	45	45	1	8	4	4
31	[7,90]	[125,36]	129,77	161	136	4	5	315	225	21	16	3	1
32	[7,90]	[89,17]	109,79	143	285	5	18	315	945	154	1073	3	1
33	[7,90]	[95,80]	88,57	89	89	1	1	45	45	6	11	4	4
34	[7,90]	[104,36]	111,02	140	149	4	10	315	495	107	109	2	1
35	[7,90]	[3,35]	55,15	56	56	1	1	45	45	5	10	3	3
36	[7,90]	[127,70]	121,66	127	138	8	62	1395	9990	68	20	1	1
37	[7,90]	[85,18]	106,15	138	281	5	18	315	945	147	1303	3	1
38	[7,90]	[38,81]	32,28	32	32	1	1	45	45	1	7	4	4
39	[7,90]	[40,38]	61,59	73	61	5	15	315	900	8	11	3	1
40	[7,90]	[3,85]	6,40	6	6	1	1	45	45	0	6	3	3
41	[7,90]	[126,24]	136,08	174	148	4	5	315	225	24	23	3	1
42	[7,90]	[102,17]	119,81	156	298	5	18	315	945	227	1579	3	1
43	[7,90]	[34,78]	29,55	28	28	1	1	45	45	1	7	3	3
44	[7,90]	[68,39]	79,51	101	91	3	15	225	945	48	180	3	1
45	[7,90]	[3,80]	10,77	11	11	1	1	45	45	0	7	3	3
46	[7,90]	[129,69]	123,79	128	139	8	47	1395	7425	103	19	1	1
47	[7,90]	[44,15]	83,63	101	96	4	23	315	1485	19	29	2	1
48	[7,90]	[61,80]	54,92	55	55	1	1	45	45	3	11	4	4
49	[7,90]	[102,39]	107,82	135	151	3	10	225	495	96	162	3	1
50	[7,90]	[3,69]	21,38	22	22	1	1	45	45	1	6	3	3

Amostra	Origem	Destino	Distância	Pontos proposta	Pontos A*	Curvas Proposta	Curvas A*	Soma º Proposta	Soma º A*	Tempo Proposta	Tempo A*	Dist. min. Proposta	Dist. min. A*
51	[7,90]	[126,71]	120,51	126	130	8	28	1395	4410	54	20	1	1
52	[7,90]	[99,15]	118,70	156	295	4	18	315	945	200	1326	2	1
53	[7,90]	[57,79]	51,20	51	51	1	1	45	45	3	9	3	3
54	[7,90]	[47,36]	67,20	83	70	4	15	315	945	13	17	2	1
55	[7,90]	[3,14]	76,11	77	77	1	1	45	45	5	11	3	3
56	[7,90]	[129,59]	125,88	141	123	5	3	315	135	19	15	3	1
57	[7,90]	[34,16]	78,77	90	76	2	3	135	135	9	13	3	1
58	[7,90]	[97,79]	90,67	91	91	1	1	45	45	8	11	3	3
59	[7,90]	[59,38]	73,54	92	82	5	15	315	945	25	77	3	1
60	[7,90]	[4,23]	67,07	68	68	1	1	45	45	5	9	4	4
61	[7,90]	[125,15]	139,82	182	157	4	5	315	225	33	18	3	1
62	[7,90]	[89,15]	111,13	146	285	4	18	315	945	183	1049	2	1
63	[7,90]	[69,79]	62,97	63	63	1	1	45	45	3	10	3	3
64	[7,90]	[86,38]	94,58	119	167	5	10	315	495	56	550	3	1
65	[7,90]	[4,72]	18,25	19	19	1	1	45	45	0	6	4	4
66	[7,90]	[129,68]	123,97	129	141	8	26	1395	3510	117	24	1	1
67	[7,90]	[31,15]	78,75	88	77	2	3	135	135	7	12	2	1
68	[7,90]	[53,79]	47,30	47	47	1	1	45	45	3	7	3	3
69	[7,90]	[30,39]	55,95	63	53	2	3	135	135	4	8	3	1
70	[7,90]	[4,89]	3,16	4	4	1	1	45	45	0	6	4	4
71	[7,90]	[126,73]	120,21	124	122	8	13	1395	1665	40	15	1	1
72	[7,90]	[72,16]	98,49	128	124	4	23	315	1485	90	443	3	1
73	[7,90]	[62,78]	56,29	57	56	4	1	630	45	3	9	3	2
74	[7,90]	[75,36]	86,83	111	98	4	15	315	945	46	351	2	1
75	[7,90]	[3,86]	5,66	5	5	0	0	0	0	0	6	3	3
76	[7,90]	[127,56]	124,72	143	121	4	3	315	135	15	15	3	1
77	[7,90]	[49,15]	85,96	106	101	4	23	315	1485	25	48	2	1
78	[7,90]	[40,78]	35,11	34	34	1	1	45	45	1	9	2	2
79	[7,90]	[30,39]	55,95	63	53	2	3	135	135	4	11	3	1
80	[7,90]	[3,18]	72,11	73	73	1	1	45	45	4	10	3	3
81	[7,90]	[129,33]	134,66	168	139	4	5	315	225	19	14	3	1
82	[7,90]	[31,16]	77,79	87	76	2	3	135	135	6	15	3	1
83	[7,90]	[101,79]	94,64	95	95	1	1	45	45	8	13	3	3
84	[7,90]	[62,37]	76,38	96	85	5	15	315	945	28	115	3	1
85	[7,90]	[3,77]	13,60	14	14	1	1	45	45	0	6	3	3
86	[7,90]	[127,34]	132,42	165	138	4	5	315	225	27	16	3	1
87	[7,90]	[66,17]	93,86	120	118	5	23	315	1485	56	266	3	1
88	[7,90]	[79,79]	72,84	73	73	1	1	45	45	7	10	3	3
89	[7,90]	[63,36]	77,79	99	86	4	15	315	945	27	106	2	1
90	[7,90]	[4,86]	5,00	5	5	1	1	45	45	0	8	4	4
91	[7,90]	[126,16]	140,13	182	156	4	5	315	225	26	19	3	1
92	[7,90]	[72,15]	99,25	129	124	4	23	315	1485	93	492	2	1
93	[7,90]	[82,80]	75,66	76	76	1	1	45	45	4	9	4	4
94	[7,90]	[44,38]	63,82	77	67	5	15	315	945	9	14	3	1
95	[7,90]	[3,77]	13,60	14	14	1	1	45	45	0	7	3	3
96	[7,90]	[127,22]	137,93	177	150	4	5	315	225	30	16	3	1
97	[7,90]	[55,18]	86,53	108	107	5	23	315	1485	35	85	3	1
98	[7,90]	[41,79]	35,74	35	35	1	1	45	45	2	7	3	3
99	[7,90]	[36,36]	61,29	72	56	4	3	315	135	7	8	3	1
100	[7,90]	[4,23]	67,07	68	68	1	1	45	45	7	9	4	4

APÊNDICE C – Resultados detalhados. Origem ponto 3

Amostra	Origem	Destino	Distância	Pontos proposta	Pontos A*	Curvas Proposta	Curvas A*	Soma º Proposta	Soma º A*	Tempo Proposta	Tempo A*	Dist. min. Proposta	Dist. min. A*
1	[128,91]	[127,20]	71,01	72	72	1	1	45	45	11	16	5	5
2	[128,91]	[98,18]	78,92	228	221	8	24	585	1395	2548	1892	3	1
3	[128,91]	[88,81]	41,23	51	47	11	11	765	585	58	7	1	1
4	[128,91]	[79,38]	72,18	99	188	4	24	225	1395	45	442	3	1
5	[128,91]	[3,13]	147,34	201	152	5	17	405	855	32	15	3	1
6	[128,91]	[125,32]	59,08	60	60	1	1	45	45	4	9	3	3
7	[128,91]	[96,16]	81,54	229	303	7	29	585	2115	2439	1744	3	1
8	[128,91]	[50,81]	78,64	127	85	12	11	855	585	166	12	1	1
9	[128,91]	[63,39]	83,24	115	161	2	23	135	1305	76	118	3	1
10	[128,91]	[4,80]	124,49	170	131	6	9	405	495	60	13	3	1
11	[128,91]	[128,21]	70,00	71	71	0	0	0	0	5	10	6	6
12	[128,91]	[101,16]	79,71	234	224	7	24	585	1395	2668	1615	3	1
13	[128,91]	[34,81]	94,53	142	101	3	11	225	585	50	11	3	1
14	[128,91]	[76,38]	74,25	102	182	4	26	225	1575	79	369	3	1
15	[128,91]	[3,28]	139,98	204	152	5	18	405	900	266	21	3	1
16	[128,91]	[125,13]	78,06	79	79	1	1	45	45	6	13	3	3
17	[128,91]	[55,18]	103,24	185	175	8	32	585	1935	757	105	3	1
18	[128,91]	[52,78]	77,10	126	83	9	3	1080	135	118	10	1	1
19	[128,91]	[45,38]	98,48	133	126	4	23	225	1305	113	19	3	1
20	[128,91]	[3,33]	137,80	209	157	5	19	405	945	704	29	3	1
21	[128,91]	[125,63]	28,16	29	29	1	1	45	45	1	8	3	3
22	[128,91]	[31,18]	121,40	168	136	3	12	225	585	21	16	3	1
23	[128,91]	[33,81]	95,52	143	102	3	11	225	585	43	13	3	1
24	[128,91]	[53,39]	91,26	125	142	2	24	135	1395	98	37	3	1
25	[128,91]	[4,23]	141,42	198	151	5	19	405	945	111	17	3	1
26	[128,91]	[125,83]	8,54	9	9	1	1	45	45	0	8	3	3
27	[128,91]	[44,18]	111,29	174	153	8	30	585	1755	302	35	3	1
28	[128,91]	[64,78]	65,31	111	71	13	3	900	135	139	8	1	1
29	[128,91]	[95,38]	62,43	83	170	4	21	225	1665	64	480	3	1
30	[128,91]	[4,49]	130,92	198	177	5	18	405	855	561	118	3	1
31	[128,91]	[126,15]	76,03	77	77	1	1	45	45	5	11	4	4
32	[128,91]	[34,15]	120,88	168	139	3	12	225	585	21	16	2	1
33	[128,91]	[67,80]	61,98	109	68	17	9	1035	495	183	8	1	1
34	[128,91]	[63,37]	84,50	116	161	4	23	225	1305	68	112	3	1
35	[128,91]	[3,64]	127,88	184	147	5	18	405	855	321	29	3	1
36	[128,91]	[128,66]	25,00	26	26	0	0	0	0	0	8	6	6
37	[128,91]	[43,16]	113,36	176	152	7	30	585	1755	300	28	3	1
38	[128,91]	[58,81]	70,71	116	77	14	11	945	585	195	11	1	1
39	[128,91]	[36,37]	106,68	147	117	8	12	585	585	25	12	3	1
40	[128,91]	[3,28]	139,98	204	152	5	18	405	900	249	22	3	1
41	[128,91]	[128,12]	79,00	80	80	0	0	0	0	4	9	6	6
42	[128,91]	[52,18]	105,38	182	169	8	31	585	1845	710	81	3	1
43	[128,91]	[57,78]	72,18	122	78	13	3	1215	135	185	12	1	1
44	[128,91]	[80,37]	72,25	99	183	4	18	225	945	48	877	3	1
45	[128,91]	[3,71]	126,59	177	159	5	24	405	2205	185	24	3	1
46	[128,91]	[126,65]	26,08	27	27	1	1	45	45	1	7	4	4
47	[128,91]	[33,16]	121,04	168	138	3	12	225	585	22	15	3	1
48	[128,91]	[42,78]	86,98	147	93	4	3	495	135	138	12	2	1
49	[128,91]	[30,38]	111,41	149	116	3	12	225	585	20	12	3	1
50	[128,91]	[4,60]	127,82	187	155	5	16	405	765	369	48	3	1

Amostra	Origem	Destino	Distância	Pontos proposta	Pontos A*	Curvas Proposta	Curvas A*	Soma º Proposta	Soma º A*	Tempo Proposta	Tempo A*	Dist. min. Proposta	Dist. min. A*
51	[128,91]	[125,72]	19,24	20	20	1	1	45	45	0	8	3	3
52	[128,91]	[52,17]	106,08	183	169	8	31	585	1845	671	69	3	1
53	[128,91]	[45,78]	84,01	150	90	4	3	495	135	167	15	2	1
54	[128,91]	[32,36]	110,64	149	118	3	12	225	585	12	13	3	1
55	[128,91]	[3,86]	125,10	178	132	5	24	405	1125	41	16	3	1
56	[128,91]	[125,77]	14,32	15	15	1	1	45	45	0	7	3	3
57	[128,91]	[82,18]	86,28	212	286	8	24	585	1395	1198	1433	3	1
58	[128,91]	[94,80]	35,74	44	41	10	9	675	495	23	10	1	1
59	[128,91]	[48,39]	95,41	130	132	2	23	135	1305	129	21	3	1
60	[128,91]	[3,51]	131,24	197	173	5	18	405	855	512	96	3	1
61	[128,91]	[127,25]	66,01	67	67	1	1	45	45	5	10	5	5
62	[128,91]	[78,16]	90,14	211	216	7	32	585	1935	1234	745	3	1
63	[128,91]	[49,79]	79,91	127	86	12	4	855	225	164	15	1	1
64	[128,91]	[33,37]	109,27	147	117	3	12	225	585	17	12	3	1
65	[128,91]	[4,78]	124,68	169	131	4	3	315	135	61	16	3	1
66	[128,91]	[125,14]	77,06	78	78	1	1	45	45	6	15	3	3
67	[128,91]	[38,17]	116,52	169	141	8	34	585	1755	63	19	3	1
68	[128,91]	[91,80]	38,60	47	44	10	9	675	495	32	8	1	1
69	[128,91]	[32,39]	109,18	146	115	3	12	225	585	13	16	3	1
70	[128,91]	[4,34]	136,47	209	158	5	19	405	945	789	32	3	1
71	[128,91]	[127,88]	3,16	4	4	1	1	45	45	0	8	5	5
72	[128,91]	[54,18]	103,95	184	173	8	32	585	1935	754	100	3	1
73	[128,91]	[93,80]	36,69	45	42	10	9	675	495	28	7	1	1
74	[128,91]	[30,38]	111,41	149	116	3	12	225	585	18	11	3	1
75	[128,91]	[3,71]	126,59	177	159	5	24	405	2205	184	23	3	1
76	[128,91]	[126,49]	42,05	43	43	1	1	45	45	3	9	4	4
77	[128,91]	[40,18]	114,34	170	146	8	30	585	1710	88	17	3	1
78	[128,91]	[72,78]	57,49	100	63	12	3	765	135	87	10	1	1
79	[128,91]	[84,36]	70,43	97	175	3	16	225	945	40	698	2	1
80	[128,91]	[4,82]	124,33	173	131	5	8	405	405	47	17	3	1
81	[128,91]	[128,28]	63,00	64	64	0	0	0	0	3	9	6	6
82	[128,91]	[100,17]	79,12	231	223	8	24	585	1395	2590	1525	3	1
83	[128,91]	[89,79]	40,80	49	46	8	4	585	225	50	8	1	1
84	[128,91]	[91,38]	64,64	87	177	4	20	225	1665	23	497	3	1
85	[128,91]	[3,74]	126,15	174	138	5	16	405	1215	123	20	3	1
86	[128,91]	[128,72]	19,00	20	20	0	0	0	0	0	9	6	6
87	[128,91]	[41,17]	114,21	172	148	8	30	585	1710	103	21	3	1
88	[128,91]	[34,80]	94,64	141	101	3	9	225	495	35	13	3	1
89	[128,91]	[61,37]	86,05	118	157	4	22	225	1305	125	97	3	1
90	[128,91]	[3,47]	132,52	201	175	5	21	405	1035	885	97	3	1
91	[128,91]	[129,29]	62,01	63	63	1	1	45	45	4	13	5	5
92	[128,91]	[78,15]	90,97	212	216	7	32	585	1935	1176	750	2	1
93	[128,91]	[45,79]	83,86	151	90	4	4	495	225	186	11	3	1
94	[128,91]	[37,39]	104,81	144	115	7	22	495	1035	38	12	2	1
95	[128,91]	[4,10]	148,11	203	151	5	17	405	855	37	18	3	1
96	[128,91]	[125,44]	47,10	48	48	1	1	45	45	2	8	3	3
97	[128,91]	[52,18]	105,38	182	169	8	31	585	1845	653	68	3	1
98	[128,91]	[67,78]	62,37	107	68	14	3	1260	135	160	9	1	1
99	[128,91]	[47,37]	97,35	132	130	4	23	225	1305	124	21	3	1
100	[128,91]	[3,67]	127,28	181	159	5	20	405	1035	245	22	3	1

APÊNDICE D – Resultados detalhados. Origem ponto 4

Amostra	Origem	Destino	Distância	Pontos proposta	Pontos A*	Curvas Proposta	Curvas A*	Soma º Proposta	Soma º A*	Tempo Proposta	Tempo A*	Dist. min. Proposta	Dist. min. A*
1	[128,5]	[128,80]	75,00	76	76	0	0	0	0	14	13	4	4
2	[128,5]	[44,18]	85,00	152	140	6	11	405	585	154	50	3	1
3	[128,5]	[94,78]	80,53	105	134	5	21	315	1845	599	325	1	1
4	[128,5]	[50,39]	85,09	109	103	4	3	225	135	12	14	3	1
5	[128,5]	[3,15]	125,40	177	150	5	5	405	225	65	36	3	1
6	[128,5]	[126,35]	30,07	31	31	1	1	45	45	1	7	4	4
7	[128,5]	[58,17]	71,02	167	154	6	11	405	585	224	74	3	1
8	[128,5]	[85,81]	87,32	139	125	15	24	1305	2025	818	327	1	1
9	[128,5]	[87,37]	52,01	71	66	2	3	135	135	4	11	3	1
10	[128,5]	[3,64]	138,22	210	175	5	12	405	585	670	33	3	1
11	[128,5]	[126,43]	38,05	39	39	1	1	45	45	1	10	4	4
12	[128,5]	[50,17]	78,92	159	146	6	11	405	585	170	54	3	1
13	[128,5]	[32,79]	121,21	168	154	3	8	225	405	22	19	3	1
14	[128,5]	[50,37]	84,31	108	103	2	3	135	135	9	17	3	1
15	[128,5]	[4,15]	124,40	176	149	5	5	405	225	49	32	3	1
16	[128,5]	[126,62]	57,04	58	58	1	1	45	45	3	9	4	4
17	[128,5]	[49,18]	80,06	157	145	6	11	405	585	181	57	3	1
18	[128,5]	[72,80]	93,60	128	173	12	21	765	2205	273	76	1	1
19	[128,5]	[72,38]	65,00	86	81	4	3	225	135	6	11	3	1
20	[128,5]	[3,33]	128,10	187	166	5	12	405	585	257	27	3	1
21	[128,5]	[129,32]	27,02	28	28	1	1	45	45	1	7	4	4
22	[128,5]	[36,18]	92,91	142	133	7	5	495	225	48	28	2	1
23	[128,5]	[86,80]	85,96	114	126	11	24	765	2025	891	308	1	1
24	[128,5]	[55,36]	79,31	103	98	3	3	180	135	10	13	2	1
25	[128,5]	[4,60]	135,65	213	179	5	12	405	585	1028	52	3	1
26	[128,5]	[129,57]	52,01	53	53	1	1	45	45	3	10	4	4
27	[128,5]	[95,17]	35,11	204	191	6	11	405	585	901	1206	3	1
28	[128,5]	[42,80]	114,11	174	167	8	27	1305	1845	303	21	3	1
29	[128,5]	[45,39]	89,69	114	108	4	3	225	135	11	17	3	1
30	[128,5]	[3,53]	133,90	207	187	5	14	405	675	991	100	3	1
31	[128,5]	[127,23]	18,03	19	19	1	1	45	45	0	6	4	4
32	[128,5]	[41,15]	87,57	153	139	5	11	405	585	111	34	2	1
33	[128,5]	[81,80]	88,51	135	121	12	24	1125	2025	627	298	1	1
34	[128,5]	[100,38]	43,28	58	53	4	3	225	135	2	8	3	1
35	[128,5]	[3,48]	132,19	202	195	5	17	405	855	599	124	3	1
36	[128,5]	[127,88]	83,01	84	84	1	1	45	45	6	14	4	4
37	[128,5]	[84,16]	45,35	195	180	5	11	405	585	613	146	3	1
38	[128,5]	[34,81]	120,88	168	156	3	8	225	405	24	21	3	1
39	[128,5]	[93,37]	47,42	65	60	2	3	135	135	4	8	3	1
40	[128,5]	[3,46]	131,55	200	191	5	17	405	855	464	112	3	1
41	[128,5]	[125,51]	46,10	47	47	1	1	45	45	3	8	3	3
42	[128,5]	[38,17]	90,80	147	135	6	12	405	585	68	30	3	1
43	[128,5]	[32,80]	121,82	169	155	3	8	225	405	26	17	3	1
44	[128,5]	[104,37]	40,00	54	49	2	3	135	135	3	9	3	1
45	[128,5]	[3,81]	146,29	199	170	5	10	405	495	37	20	3	1
46	[128,5]	[129,47]	42,01	43	43	1	1	45	45	2	7	4	4
47	[128,5]	[66,17]	63,15	175	162	6	11	405	585	462	91	3	1
48	[128,5]	[89,79]	83,65	111	129	9	23	675	2025	689	316	1	1
49	[128,5]	[98,37]	43,86	60	55	2	3	135	135	3	12	3	1
50	[128,5]	[4,87]	148,66	204	169	5	10	405	495	33	19	3	1

Amostra	Origem	Destino	Distância	Pontos proposta	Pontos A*	Curvas Proposta	Curvas A*	Soma º Proposta	Soma º A*	Tempo Proposta	Tempo A*	Dist. min. Proposta	Dist. min. A*
51	[128,5]	[128,23]	18,00	19	19	0	0	0	0	0	6	4	4
52	[128,5]	[56,15]	72,69	168	152	5	11	405	585	211	63	2	1
53	[128,5]	[72,79]	92,80	127	172	12	21	765	2205	256	81	1	1
54	[128,5]	[49,38]	85,62	109	104	4	3	225	135	12	11	3	1
55	[128,5]	[4,32]	126,91	185	163	5	11	405	495	258	38	3	1
56	[128,5]	[129,55]	50,01	51	51	1	1	45	45	2	8	4	4
57	[128,5]	[44,16]	84,72	155	140	5	11	405	585	146	42	3	1
58	[128,5]	[32,80]	121,82	169	155	3	8	225	405	21	14	3	1
59	[128,5]	[62,38]	73,79	96	91	4	3	225	135	7	15	3	1
60	[128,5]	[3,23]	126,29	177	150	5	9	405	405	128	26	3	1
61	[128,5]	[126,89]	84,02	85	85	1	1	45	45	7	12	4	4
62	[128,5]	[71,15]	57,87	183	167	5	11	405	585	494	107	2	1
63	[128,5]	[104,81]	79,70	97	106	11	26	765	2295	338	253	1	1
64	[128,5]	[37,38]	96,80	121	116	4	3	225	135	14	14	3	1
65	[128,5]	[4,74]	141,90	199	169	5	12	405	585	140	15	3	1
66	[128,5]	[127,17]	12,04	13	13	1	1	45	45	0	8	4	4
67	[128,5]	[38,16]	90,67	149	136	5	12	405	585	66	35	3	1
68	[128,5]	[54,79]	104,65	148	165	10	20	765	1665	322	31	1	1
69	[128,5]	[94,37]	46,69	64	59	2	3	135	135	3	16	3	1
70	[128,5]	[4,20]	124,90	171	149	4	5	315	225	54	36	2	1
71	[128,5]	[129,17]	12,04	13	13	1	1	45	45	0	7	4	4
72	[128,5]	[48,18]	81,05	156	144	6	11	405	585	152	44	3	1
73	[128,5]	[74,81]	93,23	138	174	14	21	1485	2205	323	67	1	1
74	[128,5]	[75,38]	62,43	83	78	4	3	225	135	6	9	3	1
75	[128,5]	[3,36]	128,79	190	171	5	12	405	585	329	50	3	1
76	[128,5]	[128,64]	59,00	60	60	0	0	0	0	3	9	4	4
77	[128,5]	[47,18]	82,04	155	143	6	11	405	585	194	47	3	1
78	[128,5]	[83,81]	88,32	137	123	15	24	1305	2025	676	336	1	1
79	[128,5]	[71,38]	65,86	87	82	4	3	225	135	6	10	3	1
80	[128,5]	[4,20]	124,90	171	149	4	5	315	225	57	23	2	1
81	[128,5]	[125,58]	53,08	54	54	1	1	45	45	3	9	3	3
82	[128,5]	[68,15]	60,83	180	164	5	11	405	585	578	96	2	1
83	[128,5]	[92,80]	83,19	108	132	11	21	765	1845	629	295	1	1
84	[128,5]	[32,36]	100,88	126	121	4	3	225	135	10	14	3	1
85	[128,5]	[3,57]	135,38	211	182	5	12	405	585	1031	59	3	1
86	[128,5]	[125,47]	42,11	43	43	1	1	45	45	3	10	3	3
87	[128,5]	[80,18]	49,73	188	176	6	11	405	585	640	150	3	1
88	[128,5]	[35,78]	118,23	165	153	6	8	450	405	34	16	3	1
89	[128,5]	[55,39]	80,53	104	98	4	3	225	135	8	10	3	1
90	[128,5]	[3,77]	144,25	196	170	6	10	405	495	35	20	3	1
91	[128,5]	[128,63]	58,00	59	59	0	0	0	0	5	9	4	4
92	[128,5]	[44,16]	84,72	155	140	5	11	405	585	141	43	3	1
93	[128,5]	[64,78]	97,08	137	164	13	18	900	1620	389	50	1	1
94	[128,5]	[34,37]	99,30	124	119	2	3	135	135	14	15	3	1
95	[128,5]	[4,75]	142,39	198	169	5	12	405	585	56	16	3	1
96	[128,5]	[129,88]	83,01	84	84	1	1	45	45	6	13	4	4
97	[128,5]	[89,18]	41,11	197	185	6	11	405	585	700	456	3	1
98	[128,5]	[34,78]	119,02	165	153	3	8	225	405	20	18	3	1
99	[128,5]	[48,38]	86,54	110	105	4	3	225	135	11	13	3	1
100	[128,5]	[4,66]	138,19	207	173	5	12	405	585	689	29	3	1

APÊNDICE E – Resultados detalhados. Origem ponto 5

Amostra	Origem	Destino	Distância	Pontos proposta	Pontos A*	Curvas Proposta	Curvas A*	Soma º Proposta	Soma º A*	Tempo Proposta	Tempo A*	Dist. min. Proposta	Dist. min. A*
1	[26,48]	[128,46]	102,02	110	113	2	12	315	1665	24	15	3	1
2	[26,48]	[59,15]	46,67	59	55	3	9	225	585	15	26	2	1
3	[26,48]	[99,79]	79,31	97	74	2	7	315	1665	68	10	3	1
4	[26,48]	[40,39]	16,64	15	15	1	1	45	45	0	7	3	3
5	[26,48]	[3,27]	31,14	60	47	3	8	225	405	35	9	3	1
6	[26,48]	[128,36]	102,70	103	103	3	1	135	45	17	15	3	2
7	[26,48]	[56,16]	43,86	55	52	3	9	225	585	10	26	3	1
8	[26,48]	[31,80]	32,39	33	33	1	1	45	45	1	8	4	4
9	[26,48]	[39,38]	16,40	14	14	1	1	45	45	0	7	4	4
10	[26,48]	[4,89]	46,53	63	47	3	3	225	135	4	7	3	1
11	[26,48]	[127,12]	107,22	127	120	2	3	135	135	17	11	3	1
12	[26,48]	[53,16]	41,87	52	49	3	9	225	585	7	24	3	1
13	[26,48]	[90,81]	72,01	89	65	4	7	675	1665	58	8	3	1
14	[26,48]	[98,38]	72,69	73	73	1	1	45	45	7	9	4	4
15	[26,48]	[3,30]	29,21	63	49	3	8	225	405	50	11	3	1
16	[26,48]	[125,73]	102,11	117	135	3	15	585	1935	13	28	3	1
17	[26,48]	[46,16]	37,74	45	42	3	9	225	585	6	14	3	1
18	[26,48]	[103,80]	83,38	101	78	4	7	945	1665	85	12	3	1
19	[26,48]	[75,36]	50,45	50	50	2	1	90	45	6	9	2	2
20	[26,48]	[3,51]	23,19	80	66	3	5	225	225	129	62	3	1
21	[26,48]	[128,39]	102,40	103	103	1	1	45	45	9	13	3	3
22	[26,48]	[46,17]	36,89	43	42	4	9	225	585	4	10	3	1
23	[26,48]	[101,79]	81,15	99	76	2	7	315	1665	76	11	3	1
24	[26,48]	[96,36]	71,02	71	71	2	1	90	45	6	11	2	2
25	[26,48]	[3,66]	29,21	65	51	3	5	225	225	46	11	3	1
26	[26,48]	[129,44]	103,08	109	105	2	10	315	1485	18	11	3	1
27	[26,48]	[92,15]	73,79	92	88	3	9	225	585	41	1086	2	1
28	[26,48]	[72,80]	56,04	70	50	4	7	945	1395	26	8	3	1
29	[26,48]	[48,38]	24,17	23	23	1	1	45	45	1	7	4	4
30	[26,48]	[4,20]	35,61	50	46	2	6	135	315	3	7	2	1
31	[26,48]	[125,36]	99,72	100	100	2	1	90	45	9	14	3	2
32	[26,48]	[34,15]	33,96	34	34	1	1	45	45	1	7	2	2
33	[26,48]	[59,81]	46,67	58	45	4	8	675	1485	16	9	3	1
34	[26,48]	[48,36]	25,06	23	23	2	1	90	45	1	7	2	2
35	[26,48]	[3,11]	43,57	60	47	3	6	225	315	3	7	3	1
36	[26,48]	[129,51]	103,04	113	114	3	9	405	1305	21	12	3	1
37	[26,48]	[32,16]	32,56	33	33	1	1	45	45	1	8	3	3
38	[26,48]	[94,79]	74,73	92	69	2	7	315	1665	51	10	3	1
39	[26,48]	[63,37]	38,60	38	38	1	1	45	45	2	8	3	3
40	[26,48]	[3,13]	41,88	58	47	3	6	225	315	4	8	3	1
41	[26,48]	[129,18]	107,28	123	114	2	3	135	135	16	12	3	1
42	[26,48]	[81,16]	63,63	80	77	3	9	225	585	28	165	3	1
43	[26,48]	[100,79]	80,23	98	75	2	7	315	1665	104	10	3	1
44	[26,48]	[85,37]	60,02	60	60	1	1	45	45	4	10	3	3
45	[26,48]	[4,80]	38,83	53	47	4	3	225	135	3	6	3	1
46	[26,48]	[128,42]	102,18	106	103	2	8	315	1395	13	11	3	1
47	[26,48]	[50,16]	40,00	49	46	3	9	225	585	8	17	3	1
48	[26,48]	[45,80]	37,22	43	54	4	13	945	2835	6	13	3	1
49	[26,48]	[79,36]	54,34	54	54	2	1	90	45	3	9	2	2
50	[26,48]	[3,77]	37,01	53	48	4	3	225	135	3	8	3	1

Amostra	Origem	Destino	Distância	Pontos proposta	Pontos A*	Curvas Proposta	Curvas A*	Soma º Proposta	Soma º A*	Tempo Proposta	Tempo A*	Dist. min. Proposta	Dist. min. A*
51	[26,48]	[128,81]	107,21	128	103	3	17	585	3105	14	12	3	1
52	[26,48]	[101,18]	80,78	97	97	4	9	225	585	51	1247	3	1
53	[26,48]	[58,78]	43,86	56	45	3	8	360	1215	18	9	2	1
54	[26,48]	[38,36]	16,97	13	13	0	0	0	0	0	6	2	2
55	[26,48]	[3,81]	40,22	56	48	3	3	225	135	3	7	3	1
56	[26,48]	[125,54]	99,18	106	100	3	5	405	1035	18	12	3	1
57	[26,48]	[55,16]	43,19	54	51	3	9	225	585	8	18	3	1
58	[26,48]	[60,81]	47,38	59	45	4	7	675	1395	19	8	3	1
59	[26,48]	[95,37]	69,87	70	70	1	1	45	45	6	11	3	3
60	[26,48]	[3,83]	41,88	58	48	3	3	225	135	2	6	3	1
61	[26,48]	[126,32]	101,27	106	101	2	3	135	135	8	9	3	1
62	[26,48]	[67,15]	52,63	67	63	3	9	225	585	18	42	2	1
63	[26,48]	[58,80]	45,25	56	45	4	7	945	900	13	7	3	1
64	[26,48]	[45,38]	21,47	20	20	1	1	45	45	0	6	4	4
65	[26,48]	[4,22]	34,06	54	46	3	8	225	405	8	9	3	1
66	[26,48]	[129,63]	104,09	111	104	3	1	585	315	12	10	3	1
67	[26,48]	[71,18]	54,08	67	67	4	9	225	585	18	55	3	1
68	[26,48]	[90,79]	71,11	88	65	2	7	315	1665	58	8	3	1
69	[26,48]	[43,37]	20,25	18	18	1	1	45	45	0	6	3	3
70	[26,48]	[3,89]	47,01	64	48	3	3	225	135	4	7	3	1
71	[26,48]	[126,76]	103,85	121	101	3	10	585	2295	10	10	3	1
72	[26,48]	[80,17]	62,27	77	76	4	9	225	585	26	140	3	1
73	[26,48]	[36,80]	33,53	34	33	3	1	405	45	2	7	3	1
74	[26,48]	[57,36]	33,24	32	32	2	1	90	45	1	8	2	2
75	[26,48]	[3,47]	23,02	80	67	3	10	225	495	129	66	3	1
76	[26,48]	[125,42]	99,18	103	100	2	8	315	1395	10	8	3	1
77	[26,48]	[70,18]	53,25	66	66	4	9	225	585	23	61	3	1
78	[26,48]	[85,81]	67,60	84	60	4	7	675	1665	49	11	3	1
79	[26,48]	[91,39]	65,62	66	66	1	1	45	45	5	13	3	3
80	[26,48]	[4,16]	38,83	54	46	3	6	225	315	3	8	3	1
81	[26,48]	[125,15]	104,36	122	117	2	3	135	135	16	14	3	1
82	[26,48]	[53,17]	41,11	50	49	4	9	225	585	7	17	3	1
83	[26,48]	[42,81]	36,67	41	40	4	9	675	855	4	10	3	1
84	[26,48]	[78,37]	53,15	53	53	1	1	45	45	3	10	3	3
85	[26,48]	[3,44]	23,35	77	63	3	8	225	405	112	54	3	1
86	[26,48]	[127,72]	103,81	118	133	3	14	585	1935	10	24	3	1
87	[26,48]	[73,17]	56,30	70	69	4	9	225	585	25	58	3	1
88	[26,48]	[52,78]	39,70	49	43	2	10	315	2070	11	9	2	1
89	[26,48]	[100,38]	74,67	75	75	1	1	45	45	6	9	4	4
90	[26,48]	[3,14]	41,05	57	47	3	6	225	315	4	7	3	1
91	[26,48]	[127,15]	106,25	124	117	2	3	135	135	14	10	3	1
92	[26,48]	[90,15]	72,01	90	86	3	9	225	585	42	503	2	1
93	[26,48]	[68,80]	52,80	66	50	4	8	945	1485	24	8	3	1
94	[26,48]	[56,39]	31,32	31	31	1	1	45	45	1	6	3	3
95	[26,48]	[3,30]	29,21	63	49	3	8	225	405	38	11	3	1
96	[26,48]	[129,31]	104,39	110	104	2	3	135	135	11	10	3	1
97	[26,48]	[59,18]	44,60	55	55	4	9	225	585	16	25	3	1
98	[26,48]	[92,81]	73,79	91	67	4	7	675	1665	69	10	3	1
99	[26,48]	[75,39]	49,82	50	50	1	1	45	45	2	7	3	3
100	[26,48]	[4,26]	31,11	58	46	3	8	225	405	25	8	3	1

APÊNDICE F – Programa para tradução dos modelos de dados do ROS e do sistema proposto

Listagem 2 – Programa em Python para tradução entre modelos e publicação nas filas do barramento. P1

```
#!/usr/bin/python3
import rospy
import pika
import json
from sensor_msgs.msg import Range
from geometry_msgs.msg import Pose
from geometry_msgs.msg import Point
from geometry_msgs.msg import Twist
from threading import Thread
import functools
import datetime
import threading

def inicio():

    def enviarSensorRange(sensor, distancia, channel):

        sensor = sensor.replace("/", "-")

        envio = {
            "message": {
                "sensor": sensor,
                "distancia": distancia
            },
            "messageType": [
                "urn:message:Robo.Entidades:Range"
            ]
        }
        channel.basic_publish(exchange='', routing_key=sensor,
                                                body=json.dumps(envio))

    def enviarSensorLaser(sensor, point, channel):

        sensor = sensor.replace("/", "-")

        envio = {
            "message": {
                "x": point.x,
                "y": point.y,
                "z": point.z
            },
            "messageType": [
                "urn:message:Robo.Entidades:Ponto"
            ]
        }
        channel.basic_publish(exchange='', routing_key=sensor,
                                                body=json.dumps(envio))
```

Fonte: Autoria própria (2023).

Listagem 3 – Programa em Python para tradução entre modelos e publicação nas filas do barramento. P2

```
48
49      def enviarPosicao(sensor, pose, channel):
50          sensor = sensor.replace("/", "-")
51
52          envio = {
53              "message": {
54                  "dataLeitura": datetime.datetime.now()._str_(),
55                  "sensor": sensor,
56                  "posicao": {
57                      "x": pose.position.x,
58                      "y": pose.position.y,
59                      "z": pose.position.z,
60                  },
61                  "orientacao": {
62                      "x": pose.orientation.x,
63                      "y": pose.orientation.y,
64                      "z": pose.orientation.z
65                  }
66
67              },
68              "messageType": [
69                  "urn:message:Robo.Entidades:Pose"
70              ]
71          }
72          channel.basic_publish(exchange='', routing_key=sensor,
73                                                  body=json.dumps(envio))
74
75
76      def receberVelocidade(ch, method, properties, body):
77          ch.basic_ack(delivery_tag=method.delivery_tag)
78
79          filaRos =  "/" + method.exchange.replace("-", "/")
80          jsonStr = body.decode()
81          rospy.loginfo(" [x] Publicando velocidade na fila ROS %s", filaRos)
82
83          # Enviar para o ROS
84          pub = rospy.Publisher(filaRos, Twist, queue_size=10)
85
86          vel_msg = Twist()
87
88          msgRabbit = json.loads(jsonStr)
89
90          if(msgRabbit["linear"] is not None):
91              linear = msgRabbit["linear"]
92              if("x" in linear):
93                  vel_msg.linear.x = float(linear["x"])
94              if("y" in linear):
95                  vel_msg.linear.y = float(linear["y"])
96              if("z" in linear):
97                  vel_msg.linear.z = float(linear["z"])
```

Fonte: Autoria própria (2023).

Listagem 4 – Programa em Python para tradução entre modelos e publicação nas filas do barramento. P3

```
98
99          if(msgRabbit["angular"] is not None):
100             angular = msgRabbit["angular"]
101             if("x" in angular):
102                 vel_msg.angular.x = float(angular["x"])
103             if("y" in angular):
104                 vel_msg.angular.y = float(angular["y"])
105             if("z" in angular):
106                 vel_msg.angular.z = float(angular["z"])
107
108         pub.publish(vel_msg)
109
110     def callbackUltrassom(data: Range, args):
111         connection = args[2]
112         cb = functools.partial(enviarSensorRange,args[0],data.range,args[1])
113         connection.add_callback_threadsafe(cb)
114         cb()
115
116     def callbackLaser(data: Point, args):
117         connection = args[2]
118         cb = functools.partial(enviarSensorLaser, args[0], data, args[1])
119         connection.add_callback_threadsafe(cb)
120         cb()
121
122     def callbackPosicao(data: Pose, args):
123         connection = args[2]
124         cb = functools.partial(enviarPosicao, args[0], data, args[1])
125         connection.add_callback_threadsafe(cb)
126         cb()
127
128     def receberVelocidades():
129         #Declara os listeners de velocidade para todos os robos encontrados
130         connection2 = pika.BlockingConnection(parameters)
131         channel2 = connection2.channel()
132         for r in robosEncontrados:
133             channel2.queue_declare(queue=r+"-velocidade", durable=False)
134             channel2.basic_qos(prefetch_count=100)
135             rospy.loginfo("Consumindo fila %s", r+"-velocidade")
136             channel2.basic_consume(queue=r+"-velocidade",
137                                    on_message_callback=receberVelocidade)
138
139         channel2.start_consuming()
140
141     credentials = pika.PlainCredentials('admin', '1234')
142     parameters = pika.ConnectionParameters('localhost',5672,'/',credentials)
143
144     connections = {}
145     channels = {}
```

Fonte: Autoria própria (2023).

Listagem 5 – Programa em Python para tradução entre modelos e publicação nas filas do barramento. P4

```
146
147     # In ROS, nodes are uniquely named. If two nodes with the same
148     # name are launched, the previous one is kicked off. The
149     # anonymous=True flag means that rospy will choose a unique
150     # name for our 'listener' node so that multiple listeners can
151     # run simultaneously.
152     rospy.init_node('leitura_robos', anonymous=True)
153     robosEncontrados = []
154
155     topics = rospy.get_published_topics()
156     for t in topics:
157         nome: str = t[0]
158         if(nome.startswith('/robo')):
159
160             if nome[1:6] not in str(robosEncontrados):
161                 robosEncontrados.append(nome[1:6])
162
163             tipo: str = t[1]
164             queueName=nome[1:].replace("/", "-")
165             nome = nome[1:]
166             rospy.loginfo("Topico %s do tipo %s", nome, t[1])
167             if(tipo == 'sensor_msgs/Range'):
168                 connections[queueName] = pika.BlockingConnection(parameters)
169                 channels[queueName] = connections[queueName].channel()
170                 channels[queueName].queue_declare(queue=queueName, arguments=
171                                                   {'x-message-ttl' : 1000})
172
173                 rospy.Subscriber(nome, Range, functools.partial(
174                                     callbackUltrassom, args=(queueName,
175                                     channels[queueName], connections
176                                     [queueName])))
177
178             if(tipo == 'geometry_msgs/Pose'):
179                 connections[queueName] = pika.BlockingConnection(parameters)
180                 channels[queueName] = connections[queueName].channel()
181                 channels[queueName].queue_declare(queue=queueName, arguments=
182                                                   {'x-message-ttl' : 1000})
183
184                 rospy.Subscriber(nome, Pose, functools.partial(
185                     callbackPosicao, args=(queueName, channels[queueName],
186                     connections[queueName])))
187
188             if(tipo == 'geometry_msgs/Point'):
189                 connections[queueName] = pika.BlockingConnection(parameters)
190                 channels[queueName] = connections[queueName].channel()
191                 channels[queueName].queue_declare(queue=queueName, arguments=
192                                                   {'x-message-ttl' : 1000})
```

Fonte: Autoria própria (2023).

Listagem 6 – Programa em Python para tradução entre modelos e publicação nas filas do barramento. P5

```
193                 rospy.Subscriber(nome, Point, functools.partial(callbackLaser,
194                                                 args=(queueName, channels[queueName],
195                                                 connections[queueName])))
196
197     t1 = threading.Thread(target=receberVelocidades, args=[])
198     t1.start()
199
200     # spin() simply keeps python from exiting until this node is stopped
201     rospy.spin()
202
203 if __name__ == '__main__':
204     inicio()
```

Fonte: Autoria própria (2023).