Analysis of Local Trajectory Planners for Mobile Robot with Robot Operating System

Fabio Ugalde Pereira, Pedro Medeiros de Assis Brasil, Marco Antonio de Souza Leite Cuadros, Anselmo Rafael Cukla, Paulo Drews Junior e Daniel Fernando Tello Gamarra

***Abstract*—The goal of this work is to analyze and compare trajectory planners for a mobile robot in Robot Operating System (ROS), focusing on the performance of local planners on symmetric and asymmetric environments. In addition, two global planners, Dijkstra and A-star, are implemented in order to have a complete analysis and comprehension of the navigation architecture. Two local planning algorithms, Dynamic Window Approach and Timed Elastic Bands, are analyzed and compared more in depth using the mobile robot TurtleBot 3 Burger, an open-source and low-cost platform. The analyzed criteria were geometric and angular precision of the final position and orientation, time and distance of the complete trajectory, and usage of computational power. Experiments were carried out in two environments with different spatial arrangement of obstacles, with the intention of analyzing the behavior both in simulation with the Gazebo software and in the real robot. Both local planning algorithms enabled the robot to reach the target destination without any collisions, presenting the main difference in the usage of processing power.**

***Index Terms* — global planner, local planner, mobile robotics, navigation, ROS, TurtleBot.**

I. INTRODUÇÃO

Nas últimas décadas, tem-se visto grande avanço no campo da robótica móvel devido às tecnologias embarcadas cada vez mais potentes e ao desenvolvimento na área de software para robótica. Mesmo assim, ainda existe margem para melhoramentos no que diz respeito a problemas de navegação em ambientes com obstáculos, pois fatores como obstruções imprevistas ou mapas imprecisos podem limitar sua implementação.

A navegação autônoma é uma área da robótica móvel que estuda técnicas para que robôs possam se locomover de forma segura em um determinado ambiente, sendo um de seus principais desafios o planejamento de trajetórias dividido em planejamento global e local. O planejamento global precisa de um mapa do ambiente e projeta uma trajetória desde a posição inicial para chegar ao alvo, enquanto o planejador local trabalha com a informação atual e pode mudar a trajetória projetada a priori pelo planejador global, caso a dinâmica do ambiente demonstre essa necessidade. [1].

No entanto, existem diversas abordagens disponíveis para este problema tanto em nível de planejamento global quanto local. Diante disso, este trabalho objetiva comparar o desempenho de técnicas que funcionam como soluções desse problema. Fazendo experimentos em simulação e com o robô TurtleBot 3, o trabalho apresenta a comparação sistemática dos planejadores locais DWA (*Dynamic Window Approach*) [2] e TEB (*Timed Elastic Bands*) [3] em ambientes estáticos de disposição espacial simétrica e assimétrica para avaliação de seu desempenho e capacidade de tomada de decisão em ROS (Robot Operating System). Ainda, serão analisados os algoritmos de navegação global: Dijkstra [4] e A-star (ou A*) [5]. Outras variáveis foram mantidas constantes no decorrer dos testes de modo que apenas os planejadores de rotas gerassem diferenças no comportamento do robô.

Ambos os métodos locais levam em consideração a dinâmica de movimento do robô, respeitando suas velocidades máximas e a capacidade de frenagem, algo que os tornam versáteis para o uso em diversas plataformas. Assim, ao analisar dois métodos amplamente utilizados e de princípios operacionais fundamentalmente diferentes, é possível entender por meio dos dados de simulação e de experimentos reais suas características e funcionalidades em cada situação.

As principais contribuições do artigo são a comparação detalhada de dois planejadores locais através de simulações e da implementação em um robô real, realizando experimentos em ambientes simétricos e assimétricos para a avaliação de seu desempenho, em conjunto a um breve estudo de dois planejadores globais utilizando a mesma metodologia. Além disso, foi realizada a comparação de exigência computacional durante o percurso dos planejadores locais. A abordagem do tópico de planejamento de trajetórias proposta é apresentada de forma holística e integral com a utilização de planejadores globais e locais por meio de experimentos em simulação e em uma plataforma robótica.

O trabalho apresenta 6 seções, sendo a primeira uma breve introdução ao problema, a segunda seção apresenta os trabalhos relacionados, a terceira seção descreve o embasamento teórico, a quarta seção detalha a metodologia adotada, a quinta seção apresenta os principais resultados e a última seção resume as conclusões do trabalho.

II. TRABALHOS RELACIONADOS

Muitos estudos têm sido realizados em relação ao planejamento global de trajetórias para robôs móveis nos últimos tempos. Um exemplo é o trabalho realizado por

F. U. Pereira, Universidade Federal de Santa Maria, Santa Maria, Rio Grande do Sul, Brasil. (e-mail: ugaldepereira@gmail.com)

P. M. Assis Brasil. Universidade Federal de Santa Maria, Santa Maria, Rio Grande do Sul, Brasil. (e-mail: pedro_mabrasil@hotmail.com)

M. A. S. L. Cuadros. Instituto Federal do Espírito Santo, Serra, Espírito Santo, Brasil (email: marcoantonio@ifes.edu.br)

A. Cukla. Universidade Federal de Santa Maria, Santa Maria, Rio Grande do Sul, Brasil. (e-mail: anselmo.cukla@ufsm.br)

P. Drews Jr. Universidade Federal de Rio Grande, Rio Grande, Rio Grande do Sul, Brasil (email: paulodrews@furg.br)

D. F. T. Gamarra. Universidade Federal de Santa Maria, Santa Maria, Rio Grande do Sul, Brasil. (e-mail: daniel.gamarra@gmail.com)

Medeiros e Silva em [6], em que é aplicado o algoritmo de Dijkstra para o planejamento de movimento de Veículos Aéreos Não Tripulados (VANTs). Zhao e Zhang em [7] propõem em seu trabalho um algoritmo A-STAR-Dijkstra-Integrated baseado nos algoritmos Dijkstra e A* para fazer com que vários carros possam se deslocar paralelamente sem colisão ou *deadlock*. Fonseca em [8] desenvolveu um sistema computacional capaz de realizar a composição de mapas temáticos, aplicando o algoritmo de Dijkstra nestes mapas para determinar a rota de menor custo entre dois pontos. O trabalho de Brugnolli, Lopes e Lemos em [9] propõe um modelo de navegação reativo para o robô móvel Robotino baseado em lógica fuzzy, visando integrá-lo num sistema de navegação híbrido baseado também no algoritmo Dijkstra.

No âmbito de planejadores locais, deve-se mencionar a obra de Khatib em [10] na qual foi desenvolvida um dos primeiros algoritmos deste tipo. O método utiliza campos potenciais artificiais que "atraem e repelem" o robô em direção ao seu alvo, desviando dos obstáculos em tempo real. O método desenvolvido na obra de Borenstein e Koren em [11], chamado de Campo de Força Virtual, é capaz de utilizar dados de sensores imprecisos bem como fusão de sensores, permitindo que o robô possa navegar sem pausas em frente a obstáculos.

Na obra de Pimentel e Aquino em [12], foram avaliados em simulação a precisão e segurança de navegadores locais em diferentes situações avaliando a navegação social, a qual lida com a coexistência de humanos e robôs no mesmo ambiente. Ainda no âmbito de navegação social, Carmona *et al.* em [13] avaliam três planejadores locais em simulação para robôs manipuladores de pallets em ambientes fechados de depósitos.

Importante para o entendimento do funcionamento dos algoritmos de navegação modernos é o conceito de SLAM (Localização e Mapeamento Simultâneos). O trabalho de Durrant-Whyte e Bailey em [14] detalha os princípios de funcionamento do SLAM probabilístico no qual o robô computa sua leitura sensorial para a criação de um mapa em tempo real para navegação. Ainda, Naotunna e Wongratanaphisanem [15] avaliam os métodos DWA, TEB e *Eband* para aplicações em robôs de tração diferencial de carga mediante dois experimentos com rastreamento por câmeras externas, evidenciando que DWA e TEB são os mais adequados para este tipo de topologia de robô móvel. O referido trabalho não apresenta resultados em simulação.

Uma das principais obras em termos de metodologia para o desenvolvimento deste trabalho é o artigo de Cybulski *et al.* em [16]. Com o propósito de analisar a precisão e repetibilidade de três algoritmos de navegação local em um robô TurtleBot 3 Waffle Pi, um percurso com três pontos de coleta de informação foi analisado em simulação e experimento utilizando obstáculos dinâmicos e estáticos. Os resultados mostraram que cada um dos algoritmos atinge os objetivos específicos para o qual foram projetados, sem nenhum vencedor para aplicações gerais. A metodologia proposta deste trabalho se diferencia de [16] por apresentar também experimentos feitos em simulação e com robôs reais com planejadores globais. A abordagem seguida no artigo com relação aos ambientes também foi diferente, discriminado os mesmos em ambientes simétricos e assimétricos.

No trabalho de Pittner em [1], é descrito um método sistemático em ROS para a escolha de planejadores locais e globais para obtenção de trajetórias ideais. Dividido em 2 planejadores globais e 3 locais, são analisadas características como distância total percorrida, capacidade de desvio de obstáculos e uso computacional. Ao combinar estes algoritmos, é possível obter trajetórias otimizadas, não existindo uma solução geral otimizada para qualquer cenário. O trabalho de Zheng em [17] detalha no formato de manual os principais parâmetros para configuração da navegação no ambiente do ROS. Levando em consideração as restrições físicas do robô a ser simulado, é apontado como configurar parâmetros de velocidades translacional e rotacional. Outra referência que auxilia no entendimento de planejadores locais é a obra de Wen *et al.* [18] na qual foi estabelecido um *benchmark* unificado para escolha de planejadores locais baseado em simulações de situações variadas.

Na obra de Marin-Plaza *et al.* em [19], o método de planejamento local *Timed Elastic Band* é implementado em um veículo real baseado no modelo de Ackermann através do uso de ROS. Neste trabalho é evidenciado o fato de que o algoritmo TEB pode ser utilizado em situações em que não se tem um modelo preciso do veículo ou quando este encontra-se fora do trajeto previsto.

Em Cukla *et al.* [20] é apresentado uma interessante abordagem de controle para otimização de ganhos de um controlador por rastreamento de trajetória posicional de um cilindro hidráulico. Por fim, da Silva *et al.* em [21] e Jesus *et al.* em [22] foram cruciais para o desenvolvimento da metodologia deste projeto devido ao uso de ROS e de robôs e ambientes de testes semelhantes.

III. FUNDAMENTAÇÃO TEÓRICA

A principal diferença entre métodos globais e locais está no seu horizonte de análise e metodologia de planejamento (Fig. 1). Métodos globais utilizam um mapa prévio para calcular a trajetória do início ao fim, geralmente buscando as rotas mais curtas entre esses pontos. Os métodos locais limitam-se ao alcance das leituras sensoriais e modificam o trajeto global para evitar obstáculos e manter a segurança durante o percurso, levando em consideração a dinâmica e as restrições físicas do robô. Suas diferenças estão descritas na Tabela I.

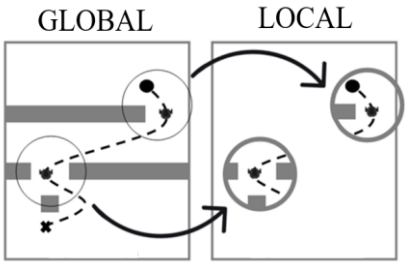

Fig. 1. Comparação entre análise global e local.

TABELA I
CARACTERÍSTICAS DE PLANEJADORES GLOBAIS E LOCAIS

Planejadores Globais	Planejadores Locais
Baseado em mapa	Baseado em sensores
Estima trajeto do início ao fim	Estima trajeto curto a frente
Não considera dinâmica do robô	Considera dinâmica do robô

É importante ressaltar que a palavra “trajetória” se refere a um conjunto de dados sobre a velocidade e a posição de um certo corpo, enquanto que “caminho” se trata apenas do trajeto em que esse corpo percorreu.

A. Dijkstra

O algoritmo Dijkstra [4] soluciona o problema do caminho mais curto num grafo dirigido ou não dirigido com arestas de peso não negativo. Escolhido um vértice como raiz da busca, esse algoritmo calcula o custo mínimo deste vértice para todos os demais vértices do grafo.

B. A* (A estrela)

A* (pronuncia-se "A-estrela") é um algoritmo de busca de caminho e passagem de gráfico [5], sendo visto como uma extensão do algoritmo de Dijkstra [4]. A* é um algoritmo de pesquisa formulado em termos de gráficos ponderados e alcança melhor desempenho usando heurísticas para orientar sua pesquisa. Esse algoritmo termina quando o caminho que escolhe estender é aquele que vai do ponto inicial até objetivo ou se não há caminhos qualificados para serem estendidos.

A* seleciona o caminho que minimiza a função f(n) em (1):

$$f(n) = g(n) + h(n) \tag{1}$$

Onde *n* é o próximo nó no caminho, *g(n)* é o custo do caminho do nó inicial para *n* e *h(n)* é uma função heurística que estima o custo do caminho mais barato de *n* até a meta. Quando *h(n)* possui o valor zero, as equações do algoritmo de Dijkstra e A* se tornam iguais.

C. Método de DWA

Este método de evitamento reativo de obstáculos proposto por Fox, Burgard e Thrun em [2] chamado de *Dynamic Window Approach* (Abordagem da Janela Dinâmica) – referido por DWA, funciona baseado diretamente a partir da dinâmica de robôs móveis. Seu princípio de funcionamento consiste na análise do ambiente apenas em um intervalo de tempo atingível a partir de velocidades que o robô consegue se mover e parar em segurança, por isso o nome “Janela Dinâmica”. Isto diminui a complexidade de processamento do cálculo de planejamento de rota, aumentando a eficiência e segurança.

Para entender seu funcionamento [2], deve-se levar em consideração que a cinemática de um robô de tração diferencial é descrita por equações que partem do princípio de que as velocidades translacionais e rotacionais podem ser controladas independentemente. Para isto, considera-se *x(t)* e *y(t)* as coordenadas do robô em um sistema global de referência, assim como *θ(t)* para sua orientação. Sendo t_0 o tempo inicial e t_n um tempo futuro, as coordenadas futuras do robô podem ser estimadas em (2) e (3).

$$x(t_n) = x(t_0) + \int_{t_0}^{t_n} v(t).\cos\theta(t)\, d(t) \tag{2}$$

$$y(t_n) = y(t_0) + \int_{t_0}^{t_n} v(t).\sin\theta(t)\, d(t) \tag{3}$$

O espaço de busca do robô é determinado em três etapas [4]: Inicialmente o DWA considera apenas trajetórias circulares pautadas pelo par *(v, ω)*, no qual *v* representa velocidade translacional e *ω* velocidade rotacional, seguido da restrição imposta pelas velocidades admissíveis as quais permitem que o robô pare em segurança antes de atingir qualquer obstáculo. Por fim, a janela dinâmica restringe as velocidades admissíveis para somente aquelas que podem ser atingidas dentro do intervalo determinado levando em consideração a aceleração limitada do robô.

Desta maneira, dentro do espaço de busca final, as possíveis rotas são estimadas de modo a maximizar o resultado de uma função objetivo, descrita em (4) [2]. Esta função leva em consideração três variáveis para estimar sua saída: o progresso do robô em direção ao ponto de chegada representado por *prog* em (4), a distância até o próximo obstáculo na trajetória estimada representado por *dist* e a velocidade translacional no sentido de movimento representado por *vel*. Os multiplicadores *α, β* e *γ* são utilizados para ponderar a influência de cada fator no resultado final, bem como *σ* é utilizado para suavizar o comportamento, resultando em afastamentos maiores dos obstáculos. Com base nestes dados, a função *G(v, ω)* altera o comportamento do robô, permitindo que a melhor rota seja utilizada.

$$G(v,\omega) = \sigma(\alpha\,.progr(v,\omega) + \beta\,.dist(v,\omega) + \gamma\,.vel(v,\omega)) \tag{4}$$

D. Método de TEB

O método de planejamento otimizado de trajetórias para evitamento de obstáculos chamado *Timed Elastic Band* (Banda Elástica Temporizada), referido como TEB, busca otimizar trajetórias sem conflito em direção ao objetivo, fundamentado na obra de Rösmann *et al.* em [3]. Esta abordagem é baseada no método das bandas elásticas de Quinlan e Khatib em [23] cuja ideia principal consiste em uma “banda elástica”, como se fosse um pedaço comprido de borracha, sujeito a forças que a deformam a partir do trajeto original para otimizar o percurso local conforme os requisitos da missão. O fator “temporizado” introduzido em TEB amplia o sistema de controle do robô com informações relevantes sobre suas restrições dinâmicas, como velocidades e acelerações, limitando a modificação da banda elástica àquilo viável de ser executado em segurança pelo robô.

O funcionamento deste algoritmo analisa uma curta distância à frente do robô na trajetória estimada pelo planejador global e cria uma sequência local de poses, otimizando a trajetória global e diminuindo o tempo total de execução do percurso [3]. As principais informações sobre a dinâmica do robô, como limite de aceleração translacional e rotacional e formato físico, são requeridas para que o algoritmo possa planejar rotas sem colisão.

Para este algoritmo, o robô é considerado uma massa pontual sem considerar possíveis deslizamentos laterais [3]. Uma banda elástica temporizada é composta de pares de localização $Q = (x_i, y_i)$ separados por intervalos constantes de tempo entre mudanças de direção $\tau = \Delta Ti$, representado pelo par $B = (Q, \tau)$. A banda otimizada B^* em (5) é obtida por meio do valor mínimo para a função objetivo descrita por (6) que representa a soma ponderada de múltiplos objetivos *γ* e as penalidades para violações das constrições impostas Γ. A velocidade, taxa de rotação e aceleração do robô são obtidas através do cálculo das diferenças finitas entre pares de pontos *(x, y)* consecutivos.

$$B^* = \underset{B}{\operatorname{argmin}} f(B) \quad (5)$$

$$f(B) = \sum_k \gamma_k \Gamma_k(B) \quad (6)$$

Outra funcionalidade do TEB é a capacidade de analisar múltiplas possíveis trajetórias simultaneamente [5]. Isto requer maior uso de poder computacional da máquina que estiver processando, não sendo sempre viável para aplicações de robótica móvel devido a possíveis restrições de hardware.

IV. Metodologia

Nesta seção será descrita as etapas seguidas nos experimentos no artigo, assim as ferramentas de software utilizadas, os ambientes nos quais foram feitos os testes, e o hardware usado.

A. Etapas dos Experimentos

Para todos os ambientes foram criados mapas utilizando o pacote *gmapping* [24]. Os ambientes foram discretizados em forma de *grid* e a estimação da pose do robô durante os experimentos foi feita baseada no método de *Adaptive Monte Carlo Localization* (AMCL). A metodologia de experimento foi baseada no trabalho de [16] e pode ser vista na Fig. 2. As coordenadas e pose iniciais do robô foram sempre as mesmas, bem como o objetivo enviado também possuía sempre os mesmos valores. Para cada experimento realizado, os dados foram armazenados e salvos para posterior análise caso o experimento fosse válido. Cada algoritmo foi testado 5 vezes em cada mapa em simulação e nos ambientes reais de modo a obter dados representativos de comportamento.

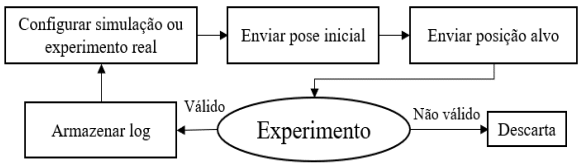

Fig. 2. Fluxograma da metodologia aplicada.

B. ROS

O Robot Operating System é um conjunto de ferramentas, bibliotecas e convenções para o desenvolvimento de software para robótica [25], [26]. É utilizado mundialmente em diferentes aplicações, mantendo um ecossistema *open-source* que facilita a programação na robótica.

Neste trabalho, o ROS foi utilizado para acessar variáveis de leitura dos sensores, como meio de implementar os algoritmos de navegação de planejamento local e para obter leituras de feedback da movimentação do robô. Seu sistema funciona por meio de "nós" e "tópicos" (serviços ou rotinas pré-definidas) que podem publicar seus resultados ou subscrever entradas em outros nós. Assim, todas as funcionalidades utilizadas em simulação e em experimentos reais foram idênticas, permitindo a validação e cruzamento dos dados.

C. TurtleBot 3 Burger

O TurtleBot 3 é uma plataforma de robótica móvel modular, compacta e customizável [27]. Em sua configuração padrão, possui diversos atributos para o uso em pesquisa no campo de robótica móvel. Além disso, o hardware e software da plataforma são *open-source*.

As especificações do robô relevantes para este trabalho encontram-se na Tabela II. Aqui vale ressaltar que os resultados deste trabalho são aplicáveis apenas a robôs de duas rodas, como o TurtleBot, no qual se controla velocidade translacional e velocidade rotacional independentemente a fim de atingir os pontos desejados no espaço.

TABELA II
ESPECIFICAÇÕES DO TURTLEBOT 3 BURGER

Itens	TurtleBot 3 Burger
Velocidade Translacional Máxima	0,22 m/s
Velocidade Rotacional Máxima	2,84 rad/s
Computador embarcado	Raspberry Pi 3
Sensores embarcados	LiDAR 360º, IMU

D. Gazebo e Rviz

Para a realização das simulações em software, optou-se pelo software *Gazebo* [28]. Facilmente integrável com o ROS e TurtleBot 3, o Gazebo é capaz de simular diversos ambientes, gerar sinais de sensores e gera uma representação gráfica de alta qualidade. Em conjunto, utilizou-se o RVIZ [29], um software do ROS que permite a visualização gráfica do que o robô está "sentindo" em seu ambiente, isto é, mostra as leituras de sensores, metas de deslocamento, etc., sendo utilizado para a identificação de problemas (bugs) ou para acompanhamento de missão.

E. Ambientes de Teste

Para a primeira parte do trabalho, realizou-se ensaios em simulação. Como mostrado na Fig. 3, optou-se pelo desenvolvimento de dois mapas de mesmo tamanho (3 metros de largura por 3 metros de comprimento), porém com disposições diferentes de seus obstáculos internos. Criou-se uma arena nestas dimensões para isolar o experimento e em ambos cenários o objetivo final encontra-se no mesmo local, de modo que o robô possa ter autonomia na navegação com comportamento reativo, independentemente de reconhecer o tipo objeto como é explorado em outros trabalhos [30].

No primeiro mapa, referido como "mapa simétrico" neste trabalho, possui uma disposição dos obstáculos de maneira simétrica (em formato da letra U) para a avaliação do funcionamento dos algoritmos quando confrontados com duas possíveis trajetórias de custo aproximado. O segundo mapa, referido como "mapa assimétrico", possui duas trajetórias possíveis de custo diferentes, permitindo que o comportamento do algoritmo possa ser avaliado em outras condições.

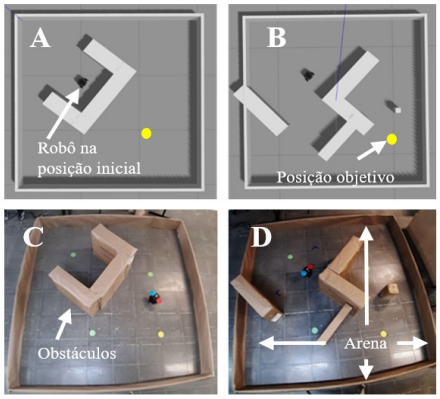

Fig. 3. Ambientes em simulação (A e B) e ambientes reais (C e D).

Ambos os cenários possuem locais de constrições de aproximadamente 40 centímetros, permitindo a análise do comportamento diante de ambientes estreitos e encontram-se disponíveis para acesso em um repositório online em [31]. Os ambientes reais foram criados seguindo as mesmas dimensões e disposição de obstáculos do ambiente simulado.

V. Resultados

Para a avaliação dos resultados, observou-se critérios primários como a capacidade de atingir o ponto de destino e o desvio dos obstáculos. Além disso, a dispersão geométrica e angular do ponto de chegada, o tempo e distância totais de percurso e o uso de CPU foram avaliados para a comparação entre os algoritmos locais.

A. Planejadores Globais

Os experimentos com planejadores globais, tanto em simulação como no robô real, seguiram uma metodologia de dez testes para cada mapa, sendo cinco utilizando Dijkstra e cinco utilizando A*. Os testes em simulação foram feitos para validar o funcionamento dos algoritmos de busca de caminho mínimo. São mostradas na Fig. 4 sequências de quadros do robô indo em direção ao objetivo, utilizando o algoritmo de Dijkstra para calcular esta rota em simulação no ambiente simétrico. Os pontos amarelos representam o objetivo final que o robô deve alcançar.

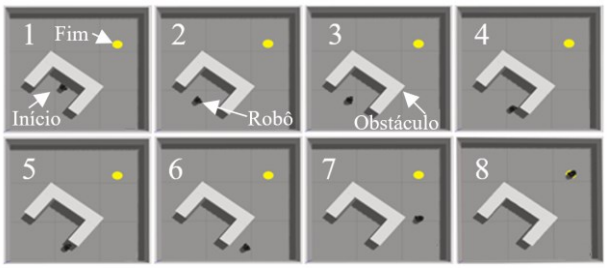

Fig. 4. Imagens sequenciais (de 1 a 8) do software Gazebo utilizando algoritmo de Dijkstra no ambiente simulado em formato de "U".

Para os ensaios reais, foram registrados a odometria do TurtleBot 3 Burger, a coordenada do alvo e o caminho percorrido pelo robô a partir de um sistema de captura de posição para se obter uma maior confiabilidade dos dados. Os algoritmos de Dijkstra e A* foram capazes de alcançar o objetivo em todos os mapas reais e simulados [32] [33] e seus resultados podem ser vistos na Tabela III. Para os ensaios seguintes de planejadores locais, utilizou-se Dijkstra como planejador global da navegação.

TABELA III
RESULTADOS PARA PLANEJADORES GLOBAIS

Mapa	Valor Médio	Simulação		Experimento	
		Dijkstra	A*	Dijkstra	A*
Simétrico	Tempo	20,9 s	21,2 s	29 s	39 s
	Erro geo.	± 0,03 m	± 0,03 m	± 0,02 m	± 0,03 m
	Erro ang.	± 2,88°	± 3,48°	± 2°	± 4°
Assimétrico	Tempo	23,9 s	25,2 s	21 s	22 s
	Erro geo.	± 0,07 m	± 0,07 m	± 0,04 m	± 0,04 m
	Erro ang.	± 4,31°	± 5,28°	± 3°	± 7°

B. Planejadores Locais em Simulação

Previamente à coleta de dados, ambos algoritmos de planejamento local tiveram seus parâmetros ajustados conforme o guia de Zheng em [17]. Para aqueles parâmetros que mesmo assim ainda apresentavam comportamento errático, alguns ensaios foram realizados para que o ajuste fosse feito.

Inicialmente, pode-se afirmar que o funcionamento dos algoritmos em simulação foi comprovado eficaz. Em nenhum dos casos de testes ocorreram colisões, assim como em todos os ensaios o objetivo geométrico e angular foi atingido de maneira razoável (Fig. 5). Nota-se que para que o robô possa cumprir a missão e achar uma rota através da constrição, é necessário que sua margem de segurança ao redor dos obstáculos seja reduzida.

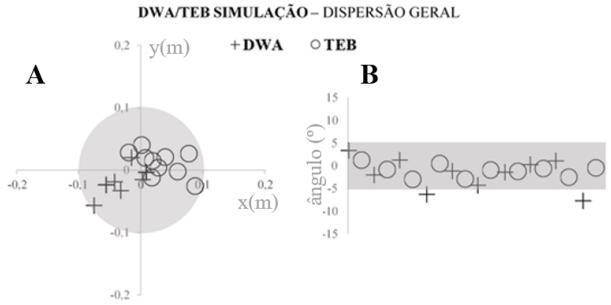

Fig. 5. Gráfico de dispersão geométrica (A) e angular (B) da pose final em simulação. Área em cinza representa margem de erro permitida pelo algoritmo.

C. Planejadores Locais Robô Real

Pôde-se notar inicialmente que o comportamento de cada planejador local foi condizente com as simulações realizadas previamente, porém houve uma diferença para o TEB que não conseguiu completar um dos ensaios em cada mapa devido à incapacidade de encontrar uma rota viável. Ambos os ensaios em que o algoritmo TEB não foi capaz de concluir a missão, a falha ocorreu em uma área de constrição do mapa. Uma missão do experimento real pode ser vista em Fig. 6.

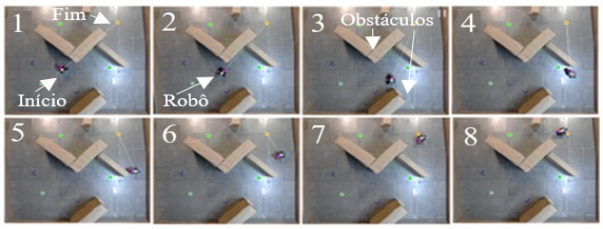

Fig. 6. Imagens sequenciais (de 1 a 8) do trajeto em experimento real em mapa assimétrico.

Ao analisar a dispersão para o robô real na Fig. 7, é possível notar as variações entre simulação e experimento real. Analisando somente simulação, tendo a margem de erro como limite, ambos algoritmos são precisos e acurados geometricamente. Ainda, o TEB apresentou alta acurácia angular, ao passo que DWA apresentou comparativamente uma leve dispersão angular com 20% dos pontos além da margem de erro.

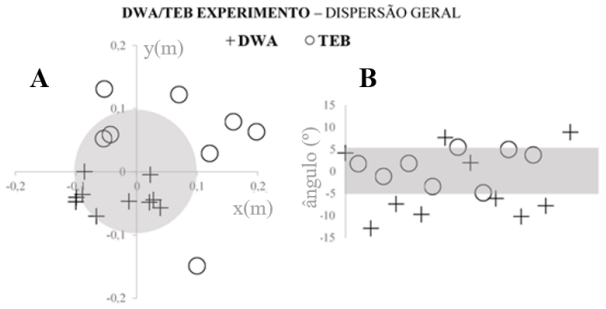

Fig. 7. Gráfico de dispersão geométrica (A) e angular (B) da pose final para robô real. Área em cinza representa margem de erro permitida pelo algoritmo.

Ao analisar o experimento, nota-se que DWA diminuiu sua precisão, mas manteve grande parte dos pontos dentro do limite estipulado. O TEB apresentou grande dispersão geométrica, com apenas 25% de taxa de sucesso. O comportamento oposto ocorre para a análise angular, na qual o TEB mantém relativa precisão, porém o DWA apresenta-se muito disperso com apenas 20% dos pontos inferiores à margem de erro. Os dados comparativos para desvio padrão e margem de erro geométricos e angulares estão evidenciados na Tabela IV.

TABELA IV
ANÁLISE DE DISPERSÃO GEOMÉTRICA E ANGULAR

Característica	Simulação		Experimento	
	DWA	TEB	DWA	TEB
σ geométrico	0,09 m	0,05 m	0,08 m	0,15 m
Casos inferiores a margem de erro geo.	90%	100%	80%	25%
σ angular	3,72º	0,67º	8,27º	3,72º
Casos inferiores a margem de erro ang.	80%	100%	20%	88%

Outro fator importante para a comparação dos algoritmos é o tempo de execução da missão e distância total percorrida. Para estas variáveis, foram calculados a média aritmética e o desvio padrão, presentes na Tabela V para o tempo total de missão. Nota-se que, em simulação, TEB apresenta menores tempos ao passo que em experimento foi DWA que teve um desempenho melhor. Mesmo assim, TEB apresentou desvios padrão significativamente menores em ambas as situações, sendo mais consistente de maneira geral.

TABELA V
TEMPO DE CONCLUSÃO DA MISSÃO EM SEGUNDOS

Mapa	Valor	Simulação		Experimento	
		DWA	TEB	DWA	TEB
Simétrico	$\bar{x}$	31,48 s	25,5 s	27 s	33,82 s
	σ	5,62 s	1,01 s	3,93 s	3 s
Assimétrico	$\bar{x}$	25,24 s	16,43 s	25,76 s	31,46 s
	σ	5,47 s	0,47 s	12,23 s	1,56 s

No quesito distância, as variações entre os algoritmos são menores. Em todas as situações, a distância média do DWA é inferior ao TEB como visto na Tabela VI. TEB novamente apresentou maior precisão, mantendo baixo desvio padrão em todas as situações. Isso demonstra que o TEB é um algoritmo que possui boa repetibilidade de desempenho, ao passo que DWA é capaz de gerar resultados de menor valor.

TABELA VI
DISTÂNCIA DE CONCLUSÃO EM METROS

Mapa	Valor	Simulação		Experimento	
		DWA	TEB	DWA	TEB
Simétrico	$\bar{x}$	3,31 m	4,26 m	3,5 m	3,67 m
	σ	0,3 m	0,17 m	0,47 m	0,49 m
Assimétrico	$\bar{x}$	3,22 m	3,38 m	2,65 m	3,18 m
	σ	0,39 m	0,06 m	0,37 m	0,14 m

Algo de notável diferença é o tempo de processamento de cada algoritmo. A Fig. 8 mostra o uso de CPU do tópico */move_base* em simulação no computador utilizado de especificação Intel® CoreTM i-7 5500U @ 2,4GHz. Este computador não possuía GPU e processos como a simulação em Gazebo, visualização no RVIZ e outros serviços em ROS estavam rodando simultaneamente à coleta dos dados apresentados. Ao analisar os dados, nota-se que TEB consome significativamente mais poder computacional do que o DWA, principalmente em zonas de constrição do mapa. Isto pode ser a razão de TEB ter apresentado maiores tempos de percurso nos experimentos e até mesmo casos sem rotas viáveis, caso o limite de processamento da máquina para este processo tenha sido atingido durante o procedimento.

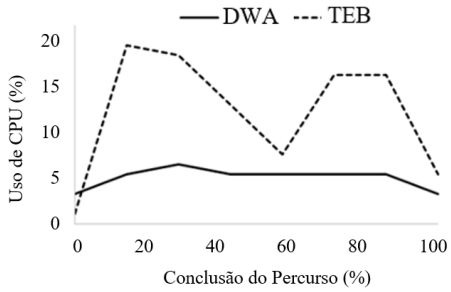

Fig. 8. Uso de CPU de planejadores locais durante percurso em simulação.

VI. Discussão dos Resultados

Após conduzir todos os ensaios e analisar os dados obtidos, não é possível indicar um algoritmo de navegação local que tenha um desempenho melhor em todas as situações. Ambos foram capazes de atingir o objetivo primário desejado – atingir o objetivo sem colisões. O DWA é intrinsecamente mais leve computacionalmente e isto pode ser um fator decisivo na escolha. No entanto, o TEB chega com maior precisão angular no objetivo, e quando poder computacional não é um fator crítico, pode ser usado com a função de estimar múltiplas trajetórias otimizando ainda mais o percurso. As principais características podem ser comparadas lado a lado na Tabela VII.

Uma vez estipulados os objetivos de navegação para um robô móvel, pode-se avaliar os resultados aqui apresentados para fundamentar sua escolha. Para casos em que precisão angular

possa ser mais relevante, TEB torna-se mais interessante, porém, caso a capacidade computacional e a capacidade da bateria sejam reduzidos, DWA consegue ainda assim suprir as principais necessidades de navegação. Como ponto forte de ambos, os algoritmos estão disponíveis para uso do público nos repositórios de ROS.

Para a pose final, algumas diferenças devem ser notadas, como o fato de DWA apresentar melhor desempenho geométrico no alcance da pose final ao passo que TEB teve melhores resultados para a orientação final.

TABELA VII
AVALIAÇÃO QUALITATIVA ENTRE ALGORITMOS

Característica	DWA	TEB
Maior acurácia geométrica	x	
Maior precisão geométrica	x	
Maior acurácia angular		x
Maior precisão angular		x
Menor distância total	x	
Menor tempo total	-	-
Menor uso de CPU	x	

No quesito distância total de percurso, DWA apresentou trajetos mais curtos provavelmente devido à sua natureza de funcionamento que prioriza as trajetórias mais otimizadas apenas dentro da janela dinâmica. O TEB deforma o trajeto global dentro do seu horizonte de análise como se fosse uma banda elástica, gerando trajetórias otimizadas de acordo com a dinâmica e cinemática do robô, não sendo necessariamente as rotas mais curtas.

Ao analisar os tempos totais de percursos, nota-se que em simulação TEB apresentou os menores tempos ao passo que em experimentos reais foi DWA. Algumas das possíveis causas para esta inconsistência são a falta de poder computacional suficiente e divergências não conhecidas entre modelos de simulação e reais.

Quando analisado o processamento da navegação sob a perspectiva de uso de CPU, TEB apresentou um consumo mais elevado de recursos durante o percurso, com picos nas zonas de constrição do mapa onde há maior dificuldade para o sistema de controle em encontrar uma rota viável.

VII. Conclusões

Conclui-se que ambos os algoritmos globais e locais foram capazes de percorrer as missões de maneira satisfatória. Após analisar os resultados levando em conta os princípios de funcionamento de cada algoritmo, é possível compreender melhor seus comportamentos perante as situações propostas. Ressalta-se que ambos cumpriram o objetivo primário e que suas características específicas devem ser analisadas pelo projetista de modo a se adequar ao robô móvel utilizado e o tipo de ambiente a ser navegado.

O artigo apresenta uma avaliação de planejadores locais em ambientes de simulação, que posteriormente foram construídos e feitos experimentos com o robô real. Ainda, são apresentados experimentos em simulação e com o robô real dos planejadores globais. Baseado nos resultados observados, o artigo consegue oferecer uma abordagem holística no quesito de planejamento de trajetórias, evidenciado pela análise de características espaciais e temporais dos trajetos, bem como uso de poder computacional de cada algoritmo.

Como sugestão de trabalhos futuros, seria interessante realizar experimentos similares em ambientes externos e com desnível altimétrico. Ainda, fusão de métodos também apresenta interessantes possibilidades de estudo.

Agradecimentos

Agradecemos aos colegas Junior Costa de Jesus e Jair Bottega pela grande ajuda neste trabalho.

Referências

[1] M. Pittner, M. Hiller, F. Particke, L. Patino-Studencki, and J. Thielecke, "Systematic Analysis of Global and Local Planners for Optimal Trajectory Planning," in *ISR 2018; 50th International Symposium on Robotics*, Jun. 2018, pp. 1–4.

[2] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997, doi: 10.1109/100.580977.

[3] C. Roesmann, W. Feiten, T. Woesch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *ROBOTIK 2012; 7th German Conference on Robotics*, May 2012, pp. 1–6.

[4] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, Dec. 1959, doi: 10.1007/BF01386390.

[5] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968, doi: 10.1109/TSSC.1968.300136.

[6] F. L. L. Medeiros and J. D. S. Silva, "Aplicação de Algoritmo Dijkstra ao Planejamento de Movimento de VANTs," São José dos Campos - SP, 2010, p. 8.

[7] Z. Zhang and Z. Zhao, "A Multiple Mobile Robots Path planning Algorithm Based on A-star and Dijkstra Algorithm," *Int. J. Smart Home*, vol. 8, pp. 75–86, May 2014, doi: 10.14257/ijsh.2014.8.3.07.

[8] A. R. Fonseca, "Composição de Mapas Planares e Planejamento de Rotas Aplicados à Navegação de Robôs Móveis e Linhas de Transmissão," Master's Thesis, Universidade Federal de Minas Gerais, Belo Horizonte, MG, Brazil, 2006.

[9] M. M. Brugnolli, G. C. Lopes, and M. A. D. Lemos, "Uma Solução Híbrida Para Navegação e Desvio De Obstáculos Baseada No Algoritmo Dijkstra E Lógica Fuzzy," *Olimpo Sorocaba*, p. 6, 2013.

[10] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *1985 IEEE International Conference on Robotics and Automation Proceedings*, Mar. 1985, vol. 2, pp. 500–505. doi: 10.1109/ROBOT.1985.1087247.

[11] J. Borenstein and Y. Koren, "High-speed obstacle avoidance for mobile robots," in *Proceedings IEEE International Symposium on Intelligent Control 1988*, Aug. 1988, pp. 382–384. doi: 10.1109/ISIC.1988.65461.

[12] F. Pimentel and P. Aquino, "Performance Evaluation of ROS Local Trajectory Planning Algorithms to Social Navigation," in *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*, Oct. 2019, pp. 156–161. doi: 10.1109/LARS-SBR-WRE48964.2019.00035.

[13] M. Fernandez Carmona, T. Parekh, and M. Hanheide, "Making the Case for Human-Aware Navigation in Warehouses," in *Towards Autonomous Robotic Systems*, vol. 11650, K. Althoefer, J. Konstantinova, and K. Zhang, Eds. Cham: Springer International Publishing, 2019, pp. 449–453. doi: 10.1007/978-3-030-25332-5_38.

[14] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006, doi: 10.1109/MRA.2006.1638022.

[15] I. Naotunna and T. Wongratanaphisan, "Comparison of ROS Local Planners with Differential Drive Heavy Robotic System," in *2020 International Conference on Advanced Mechatronic Systems (ICAMechS)*, Dec. 2020, pp. 1–6. doi: 10.1109/ICAMechS49982.2020.9310123.

[16] B. Cybulski, A. Wegierska, and G. Granosik, "Accuracy comparison of navigation local planners on ROS-based mobile robot," in *2019 12th*

International Workshop on Robot Motion and Control (RoMoCo), Jul. 2019, pp. 104–111. doi: 10.1109/RoMoCo.2019.8787346.
[17] K. Zheng, "ROS Navigation Tuning Guide," *arXiv:1706.09068*, p. 23, 2016.
[18] J. Wen *et al.*, "MRPB 1.0: A Unified Benchmark for the Evaluation of Mobile Robot Local Planning Approaches," *ArXiv201100491 Cs*, Nov. 2020, Accessed: Mar. 07, 2021. [Online]. Available: http://arxiv.org/abs/2011.00491
[19] P. Marin-Plaza, A. Hussein, D. Martin, and A. de la Escalera, "Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles," *Journal of Advanced Transportation*, Feb. 22, 2018.
[20] A. R. Cukla, R. C. Izquierdo, F. A. Borges, E. A. Perondi, and F. J. Lorini, "Optimum Cascade Control Tuning of a Hydraulic Actuator Based on Firefly Metaheuristic Algorithm," *IEEE Lat. Am. Trans.*, vol. 16, no. 2, pp. 384–390, Feb. 2018, doi: 10.1109/TLA.2018.8327390.
[21] D. F. T. Gamarra, A. P. Legg, M. A. de S. L. Cuadros, and E. S. da Silva, "Sensory Integration of a Mobile Robot Using the Embedded System Odroid-XU4 and ROS," in *2019 Latin American Robotics Symposium (LARS), 2019 Brazilian Symposium on Robotics (SBR) and 2019 Workshop on Robotics in Education (WRE)*, Oct. 2019, pp. 198–203. doi: 10.1109/LARS-SBR-WRE48964.2019.00042.
[22] J. C. Jesus, J. A. Bottega, M. A. S. L. Cuadros, and D. F. T. Gamarra, "Deep Deterministic Policy Gradient for Navigation of Mobile Robots in Simulated Environments," in *2019 19th International Conference on Advanced Robotics (ICAR)*, Dec. 2019, pp. 362–367. doi: 10.1109/ICAR46387.2019.8981638.
[23] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, May 1993, pp. 802–807 vol.2. doi: 10.1109/ROBOT.1993.291936.
[24] "gmapping - ROS Wiki." http://wiki.ros.org/gmapping (accessed Jun. 11, 2020).
[25] C. Fairchild and Dr. T. L. Harman, *ROS Robotics by Example*, Second. Livery Place, 25 Livery Street, Birmingham B3 2PB, UK: Packt Publishing.
[26] "ROS.org | About ROS." https://www.ros.org/about-ros/ (accessed May 08, 2020).
[27] "DYNAMIXEL SYSTEMS - TurtleBot 3 - ROBOTIS." https://www.robotis.us/turtlebot-3/ (accessed Apr. 07, 2020).
[28] "Gazebo." http://www.gazebosim.org/ (accessed Mar. 21, 2020).
[29] "rviz - ROS Wiki." http://wiki.ros.org/rviz (accessed Apr. 16, 2020).
[30] D. H. D. Reis, D. Welfer, M. A. D. S. L. Cuadros, and D. F. T. Gamarra, "Mobile Robot Navigation Using an Object Recognition Software with RGBD Images and the YOLO Algorithm," *Appl. Artif. Intell.*, vol. 33, no. 14, pp. 1290–1305, Dec. 2019, doi: 10.1080/08839514.2019.1684778.
[31] "Github repository with simulation worlds." https://github.com/fupbot/sym_asym_maps (accessed May 31, 2020).
[32] P. M. de Assis Brasil, F. U. Pereira, M. A. de Souza Leite Cuadros, A. R. Cukla, and D. F. Tello Gamarra, "A Study on Global Path Planners Algorithms for the Simulated TurtleBot 3 Robot in ROS," in *2020 Latin American Robotics Symposium (LARS), 2020 Brazilian Symposium on Robotics (SBR) and 2020 Workshop on Robotics in Education (WRE)*, Nov. 2020, pp. 1–6. doi: 10.1109/LARS/SBR/WRE51543.2020.9307003.
[33] P. M. de Assis Brasil, F. U. Pereira, M. A. de Souza Leite Cuadros, A. R. Cukla, and D. F. T. Gamarra, "Dijkstra and A* Algorithms for Global Trajectory Planning in the TurtleBot 3 Mobile Robot," in *Intelligent Systems Design and Applications*, Cham, 2021, pp. 346–356.

Fabio Ugalde Pereira Cursa Engenharia de Controle e Automação na Universidade Federal de Santa Maria. Fez intercâmbio acadêmico para a University of Southern California (2015-2016) focado em sistemas aeroespaciais e software para engenharia. ORCID https://orcid.org/0000-0003-1367-0962.

Pedro Medeiros de Assis Brasil Cursa Engenharia de Controle e Automação pela Universidade Federal de Santa Maria. Atualmente faz estágio na empresa WEG S.A. na área de Tecnologia da Informação. ORCID https://orcid.org/0000-0002-1717-036X

Marco Antonio de Souza Leite Cuadros Possui graduação em Engenharia Elétrica - Universidad Nacional del Centro del Perú (1998), mestrado em Engenharia Elétrica pela Universidade Federal do Espírito Santo (2004) e doutorado em Engenharia Elétrica pela Universidade Federal do Espírito Santo (2011). Professor Titular do Instituto Federal do Espirito Santos (IFES). ORCID https://orcid.org/0000-0003-4191-1794

Anselmo Rafael Cukla É Engenheiro Eletrônico pela UNaM (2010), Mestre (2012) e Doutor (2016) em Engenharia Mecânica pela UFRGS. Fez ainda um Doutorado em regime de cotutela na Universidade Nova de Lisboa (UNINOVA), Portugal (2016). Atualmente é professor no DPEE, na UFSM, Brasil. ORCID https://orcid.org/0000-0002-5313-4593.

Paulo Drews Jr. Doutor e mestre em Ciência da Computação pela UFMG em visão computacional e robótica. Engenheiro de Computação pela FURG. É professor adjunto da FURG e Coordenador de Planejamento da Unidade Embrapii iTec/FURG em Sistemas Robóticos e Automação. ORCID http://orcid.org/0000-0002-7519-0502.

Daniel Fernando Tello Gamarra Possui graduação em Engenharia Mecánica-Universidad Nacional Del Centro Del Perú (1998), e mestrado em Engenharia Elétrica pela Universidade Federal do Espírito Santo (2004), mestrado em Informática pela Sussex University (2006) na Inglaterra, Doutorado em Informatica pela Scuola Superiore Sant´Anna em Pisa Italia (2009). ORCID https://orcid.org/0000-0002-4714-7849.