Planejamento e Seguimento de Caminhos com Árvores de Exploração Rápida e Controle Preditivo Não-Linear ⋆

João Pedro B. Ferreira * **Renan P. Vieira** **
Téo C. Revoredo, IEEE Member ***

* *Programa de Pós-Graduação em Engenharia Eletrônica, Departamento de Engenharia Eletrônica e de Telecomunicações, Universidade do Estado do Rio de Janeiro, RJ (e-mail: jpuerj@gmail.com).*
** *Programa de Pós-Graduação em Engenharia Eletrônica, Departamento de Engenharia Eletrônica e de Telecomunicações, Universidade do Estado do Rio de Janeiro, RJ (e-mail: renan.porto.vieira@gmail.com).*
*** *Departamento de Engenharia Eletrônica e de Telecomunicações, Universidade do Estado do Rio de Janeiro, RJ (e-mail: teorevoredo@uerj.br).*

Abstract: Navigation by autonomous robots is an application of robotics that can revolutionize society. In situations where the map of the space to be navigated is known, some probabilistic and iterative algorithms can be used to generate paths. Rapidly-exploring random tree algorithms are effective even for robots with multiple degrees of freedom. This article uses and compares two rapidly-exploring random tree algorithms for path generation. Following these paths is performed by a non-linear predictive controller. The results indicate that it is possible to plan and execute a route, using the proposed methods, that takes an autonomous vehicle from the initial to the final position, avoiding any obstacles along the way.

Resumo: A navegação autônoma de robôs é uma aplicação da robótica que pode revolucionar a sociedade. Em situações que o mapa do local a ser navegado é conhecido, alguns algoritmos de caráter probabilístico e iterativo podem ser empregados para a geração de caminhos. Os algoritmos baseados em árvores de exploração rápida são eficazes até para robôs com diversos graus de liberdade. Este artigo utiliza e compara dois algoritmos de árvore de exploração para a geração de caminhos. O seguimento desses caminhos é realizado por um controlador preditivo não-linear. Os resultados indicam ser possível planejar e executar uma rota, utilizando os métodos propostos, que leva um veículo autônomo de uma pose inicial a uma final, desviando de eventuais obstáculos pelo caminho.

Keywords: Path planning; Predictive Control; Rapidly-exploring random tree
Palavras-chaves: Planejamento de caminho; Controle Preditivo; Árvore de exploração rápida

1. INTRODUÇÃO

Nos últimos anos, a navegação de robôs móveis têm sido abordada extensivamente, tanto na indústria quanto na academia. Por exemplo, uma aplicação promissora desta tecnologia consiste no desenvolvimento de veículos autônomos para o transporte de pessoas (Badue et al., 2021) ou de cargas variadas (Lee et al., 2021). Este problema se tornou mais importante com o avanço da tecnologia de carros de passeio completamente autônomos, alavancados, em parte, pelas descobertas recentes no campo da inteligência artificial (Ma et al., 2020), tornando esta área de estudo multidisciplinar.

Visto que autonomia é uma característica esperada em robôs móveis, é natural que estes consigam planejar caminhos no espaço por onde possam se locomover de modo seguro e otimizado. É um dos principais problemas da robótica autônoma, pois define uma sequência de ações a serem realizadas para que um robô, partindo de seu estado atual, possa alcançar o objetivo final, evitando colisões ao longo do percurso (Siegwart and Nourbakhsh, 2004). Devido a essa necessidade, diversas estratégias são elaboradas para solucionar este problema (Nascimento et al., 2019). Guimarães Batista (2020) sugere um planejamento de caminhos para um manipulador robótico utilizando campos potenciais artificiais (APF), mas só é eficaz se seus parâmetros de resolução estiverem bem ajustados e por

⋆ O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Os autores agradecem, também, à Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ) pelos seus suportes financeiros no desenvolvimento deste trabalho.

executar exploração pura faz com que o robô fique preso nos mínimos locais no APF (Khatib, 1986).

No contexto de veículos similares aos carros de passeio, Argento et al. (2020) e Vieira et al. (2022) propõem a realização do planejamento de caminhos para a manobra de estacionamento paralelo por meio de polinômios e utilizam algoritmos de busca para alcançar o melhor caminho. Apesar dos resultados, as soluções são limitadas por polinômios de quinto grau. Kuwata et al. (2009) adotam uma curva de Dubins, que é composta por linhas retas e arcos, para gerar um caminho, mas a curvatura do caminho gerado é descontínua. Lau et al. (2009) usam a curva de Bessel para gerar um caminho sem considerar a continuidade da curvatura.

Em geral, todas essas estratégias necessitam da utilização de técnicas de otimização, visto que os caminhos encontrados devem ser os melhores possíveis segundo os parâmetros desejados, como distância percorrida, segurança ou gasto energético. Por isso é muito comum a utilização de meta-heurísticas, como parte da estratégia adotada para se realizar o planejamento de escolher um caminho ótimo.

Um método de planejamento de caminhos pode ser baseado em amostragem, que apesar de não representar fielmente o espaço de interesse, possibilita uma representação mais compacta do mesmo, o que torna possível a aplicação de técnicas de planejamento em espaços de dimensão elevada (Sousa et al., 2017).

Entre os planejadores de caminhos baseados em amostragem, estão os algoritmos fundamentados em árvores de exploração rápida (do inglês, RRT, *rapidly-exploring random tree*). Essa classe de algoritmo é baseada em amostragem, portanto, probabilisticamente completa, que tende a encontrar uma solução conforme o aumento do número de amostras, além de determinar um plano de movimento viável de maneira rápida (LaValle, 1998). Apesar de sua praticidade, que baseia seu caminho através de uma árvore probabilística, se percebeu a necessidade de aperfeiçoar este processo por extensões (Karaman et al., 2011).

Uma dessas extensões é o *RRT** que tem como principal vantagem a convergência para uma solução assintoticamente ótima (Karaman and Frazzoli, 2010). Tornando assim a possibilidade de um caminho mais cometido em relação ao método comum aplicado para robótica móvel.

Este artigo apresenta uma visão geral sobre o funcionamento dos algoritmos de geração de árvores exploratórias aleatórias (RRT e RRT*) e sua aplicação em robôs móveis que se assemelham a carros de passeio. O objetivo da aplicação é gerar caminhos que levem o robô de um ponto inicial ao um ponto final, desviando-o de eventuais obstáculos pelo caminho. Uma vez definido o trajeto, seu seguimento é realizado usando controle não-linear preditivo (Johansen, 2011), garantindo uma condução suave e contínua. Os algoritmos e o controlador são testados em um cenário único, em diferentes situações, e os resultados são comparados.

2. BACKGROUND

2.1 Modelo Cinemático do Veículo

A modelagem do veículo é puramente cinemática, sendo a dinâmica desconsiderada, algo válido em implementações de baixas velocidades (Zhu and Rajamani, 2006). O veículo é considerado como um corpo rígido com quatro rodas em contato com o solo, sendo as rodas dianteiras responsáveis pela orientação, como ostenta a Figura 1.

As equações cinemáticas são obtidas com base no modelo tipo bicicleta (Pinheiro, 2009). Tendo o ponto de referência que descreve o caminho efetuado pelo veículo localizado no centro do eixo traseiro, denotado por P na Figura 1. A escolha deste ponto se dá por serem suas coordenadas cartesianas uma saída diferencialmente plana para o sistema, ou seja, uma vez especificadas estas variáveis e suas derivadas de segunda ordem, todas as outras variáveis de estado e a entrada do sistema são também definidas (Revoredo et al., 2016).

O raio de curvatura (R) do caminho é dado pela Equação (1), na qual ϕ é o seu ângulo de esterçamento, limitado entre $\phi_{\min}$ e $\phi_{\max}$, e L é a distância entre os eixos do veículo. As especificações destes e dos outros parâmetros do modelo são resumidos na Tabela 1.

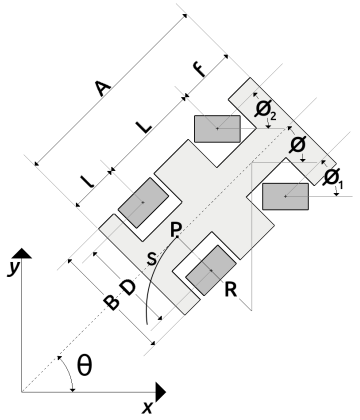

Figura 1. Modelo cinemático do veículo.

$$R = \frac{L}{\tan(\phi)} \tag{1}$$

Tabela 1. Parâmetros do modelo cinemático do veículo.

Parâmetro	Descrição
P	Ponto de referência entre eixo traseiro
R	Raio de curvatura instantâneo
L	Distância entre os eixos do veículo
ϕ	Ângulo de esterçamento do veículo
ϕ_i	Ângulo de esterçamento de cada roda (i = 1,2)
$\dot{\phi}$	Velocidade angular de esterçamento
D	Distância entre as rodas do eixo dianteiro
A	Comprimento do veículo
B	Largura do veículo
f	Distância entre o eixo traseiro e a traseira do veículo
l	Distância entre o eixo dianteiro e a dianteira do veículo
θ	Ângulo do veículo em relação ao eixo x
V	Velocidade linear

A distância (S) percorrida pelo ponto de guiamento e a sua derivada (V) são definidas por:

$$S = R \cdot \theta \tag{2}$$

$$V = R \cdot \dot{\theta} \tag{3}$$

Sendo a decomposição da velocidade, no plano x e y:

$$\dot{x}(t) = V \cdot \cos(\theta(t)) \tag{4}$$

$$\dot{y}(t) = V \cdot \operatorname{sen}(\theta(t)) \tag{5}$$

A inclinação do veículo (θ) e a sua velocidade angular podem ser expressas em função de x e y, da seguinte maneira:

$$\theta(t) = \arctan\left(\frac{\dot{y}(t)}{\dot{x}(t)}\right) \tag{6}$$

$$\dot{\theta}(t) = \frac{\dot{x} \cdot \ddot{y} - \ddot{x} \cdot \dot{y}}{\dot{x}^2 + \dot{y}^2} \tag{7}$$

Finalmente, a velocidade linear do automóvel e o seu ângulo de esterçamento são definidos:

$$V = \pm\sqrt{\dot{x}(t)^2 + \dot{y}(t)^2} \tag{8}$$

$$\phi = \arctan\left(\frac{L\dot{\theta}}{V}\right) \tag{9}$$

Devido à distância entre as rodas de um mesmo eixo, há raios de curvatura diferentes para cada lado do veículo. Para adaptar o modelo cinemático de maneira a representar adequadamente a sua movimentação, utiliza-se a geometria de Ackermann para o sistema de esterçamento (Gillespie, 1992), a qual pode assumir ângulos de esterçamento distintos para as rodas dianteiras, ϕ_1 e ϕ_2, dados pela Equação (10).

$$\phi_1, \phi_2 = \arctan\left(\frac{L}{R \pm \frac{D}{2}}\right) \tag{10}$$

Diante da velocidade linear do veículo (V) e do seu ângulo de esterçamento (ϕ), as rodas devem ser capazes de assumir velocidades diferentes entre si (Hartani et al., 2009). As velocidades das rodas traseiras direita e esquerda, V_{td} e V_{te}, respectivamente, e das rodas dianteira direita e esquerda, V_{fd} e V_{fe}, são determinadas por:

$$V_{td}, V_{te} = V \cdot \left(1 \pm \frac{D}{2 \cdot R}\right) \tag{11}$$

$$V_{fd} = \frac{V_{td}}{cos(\phi_1)} \tag{12}$$

$$V_{fe} = \frac{V_{te}}{cos(\phi_2)} \tag{13}$$

2.2 RRT

Uma árvore de exploração aleatória é uma estrutura de dados que pode ser expandida de forma iterativa. Denotada por $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ é composta por um conjunto de vértices, $\mathcal{V} = \{x_0, x_1, \ldots\}$), que representam estados do robô, e um conjunto de arestas, $\mathcal{E} = \{(x_0, x_1), (x_0, x_2), (x_1, x_3), \ldots\}$, que denotam a conexão entre dois vértices. A distância entre vértices é calculada de acordo com uma função $\rho(x_i, x_j)$. Denota-se por $\mathcal{X}_{free}$ o espaço de estados que não incorrem em colisão com algum obstáculo físico.

A estrutura $\mathcal{V}$ é inicializada com apenas um vértice, x_0, que representa o estado inicial do robô. A cada iteração do algoritmo, uma amostra de estado x_{rand} é obtida de forma aleatória, de modo que $x_{rand} \in \mathcal{X}_{free}$. Com base na métrica ρ, verifica-se o vértice $x_k \in \mathcal{V}$ que minimiza a distância $\rho(x_k, x_{rand})$. Denomina-se este vértice de x_{near}. O controle $u_k \in U$ que leva x_{near} o mais próximo possível de x_{rand}, conforme a métrica ρ, é selecionado. Por fim, o novo vértice (x_{new}) adicionado à $\mathcal{V}$ é calculado a partir da função de transição de estado (14), desde que não haja colisão no percurso. O ciclo descrito se repete K vezes, dando origem a K vértices e arestas na estrutura $\mathcal{T}$.

$$\begin{aligned} \dot{x} &= f(x, u) \\ x_{new} &= x_{near} + \dot{x}\Delta t \end{aligned} \tag{14}$$

Um ponto crucial no algoritmo é a seleção do sinal de controle u_k. Considerando a velocidade do veículo constante, em um único sentido, a ação de controle é restrita à escolha do ângulo de direção. É possível utilizar um espaço de controle de Dubins (Dubins, 1957), onde $U = \{-\phi_{\max}, 0, \phi_{\max}\}$. Dessa forma, os caminhos gerados são retas ou curvas, nas quais o ângulo de direção seja o maior possível. Um espaço de controle de Reeds-Shepp (Reeds and Shepp, 1990) permite que o veículo se movimente na direção contrária, utilizando o mesmo conjunto de controle.

Nota-se que, dada uma função de transição de estado, fixar o espaço de controle permite com certa facilidade a definição de uma métrica ρ de distância entre estados. Por esta razão, alguns softwares como Matlab (MATLAB, 2018) agrupam esses elementos em um único objeto.

O processo de geração da árvore é apresentado em forma de pseudocódigo a seguir. Assume-se que existe uma função "Livre(x_a, x_b)" que retorna um indicador de não colisão no caminho de x_a até x_b.

Algoritmo 1 RRT

Require: x_0, K, Δt, f, ρ
$\mathcal{V} \leftarrow \{x_0\}$
$\mathcal{E} \leftarrow \emptyset$
$\mathcal{T} \leftarrow (\mathcal{V}, \mathcal{E})$
for $i = 1 \ldots K$ **do**
  $x_{rand} =$ EstadoAleatorio( )
  $x_{near} =$ EstadoMaisProximo(x_{rand})
  $u_k =$ SelecionarControle(x_{near}, x_{rand})
  $x_{new} = f(x_{near}, u, \Delta t)$
  if Livre(x_{near}, x_{new}) **then**
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{x_{new}\}$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_{near}, x_{new})\}$

*2.3 RRT**

O algoritmo RRT* busca otimizar a estrutura $\mathcal{T}$ ao remodelar as conexões existentes, buscando uma distância menor do vértice inicial aos outros.

De maneira semelhante ao RRT, o processo iterativo de construção da árvore se inicia com a amostragem aleatória de um estado $x_{rand} \in \mathcal{X}_{free}$. Dentre os vértices $x_k \in \mathcal{V}$, procura-se x_{near}, o estado mais próximo de x_{rand}, de acordo com uma métrica ρ. Busca-se o controle u_k que minimiza $\rho(x_{near}, x_{rand})$ e, em seguida, calcula-se x_{new} a partir da Equação (14).

A distinção entre às duas versões do RRT ocorre na etapa seguinte. Para isso, define-se como o custo de um estado, $J(x_i)$, a distância total de um caminho que leva de x_0 a x_i. O custo de navegação de um estado genérico x_a até x_b é denotado por $J(x_a, x_b)$. Devido à natureza unidirecional dos algoritmos baseados em árvores de amostragem, para cada estado há apenas um caminho correspondente.

No algoritmo RRT*, o vértice que antecede x_{new} não é x_{near}, mesmo que o percurso entre x_{near} e x_{new} esteja livre. De outro modo, obtêm-se um conjunto de estados $X_{near} \subseteq T$ na vizinhança de x_{new}. É realizada uma busca nesse conjunto pelo estado x_n que minimiza $J(x_{new})$. Então, o vértice x_n se torna o antecessor do estado x_{new}.

Além desta otimização, o algoritmo RRT* verifica, em cada vértice $x_{n_i} \in X_{near}$, se $J(x_{n_i})$ é menor quando x_{new} é o antecessor de x_{n_i}. Denota-se o antecessor de um vértice x_i por x_i'. Caso isto ocorra, o algoritmo substitui o vértice antecessor a x_{n_i}. Portanto, ocorre uma otimização não só do caminho que leva a x_{new}, mas também de todos os percursos destinados aos vértices na vizinhança de x_{new}, ou seja, os estados do conjunto X_{near}. O algoritmo em pseudocódigo é apresentado a seguir.

Algoritmo 2 RRT*

Require: x_0, K, Δt, f, ρ
$\mathcal{V} \leftarrow \{x_0\}$
$\mathcal{E} \leftarrow \emptyset$
$\mathcal{T} \leftarrow (\mathcal{V}, \mathcal{E})$
for $i = 1 \dots K$ **do**
  $x_{rand} \leftarrow$ EstadoAleatorio( )
  $x_{near} \leftarrow$ EstadoMaisProximo(x_{rand})
  $u_k \leftarrow$ SelecionarControle(x_{near}, x_{rand})
  $x_{new} \leftarrow f(x_{near}, u, \Delta t)$
  if Livre(x_{near}, x_{new}) **then**
    $\mathcal{X}_{near} \leftarrow$ EstadosProximos(x_{new})
    $V \leftarrow V \cup \{x_{new}\}$
    $x_{min} \leftarrow x_{near}$
    $J_{min} \leftarrow J(x_{near}) + J(x_{near}, x_{new})$
    for all $x_{n_i} \in \mathcal{X}_{near}$ **do**
      $J_i \leftarrow J(x_{n_i}) + J(x_{n_i}, x_{new})$
      if $J_i < J_{min}$ **and** Livre(x_{n_i}, x_{new}) **then**
        $x_{min} \leftarrow x_{n_i}$
        $J_{min} \leftarrow J_i$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_{min}, x_{new})\}$
    for all $x_{n_i} \in \mathcal{X}_{near}$ **do**
      $J_i = J(x_{new}) + J(x_{new}, x_{n_i})$
      if $J_i < J(x_{n_i})$ **and** Livre(x_{new}, x_{n_i}) **then**
        $\mathcal{E} \leftarrow \mathcal{E} \setminus \{(x'_{n_i}, x_{n_i})\}$
        $\mathcal{E} \leftarrow \mathcal{E} \cup \{(x_{new}), x_{n_i}\}$

2.4 Controle Preditivo de Modelo Não-Linear

O controle preditivo de modelo não-linear (ou NLMPC, do inglês *non-linear model predictive control*) é um método que calcula a ação de controle a cada intervalo de tempo, utilizando uma combinação de predição baseado em um modelo não-linear de um sistema e a otimização de equações com restrições. Já que o estado é atualizado a cada instante de tempo, um problema de otimização deve ser resolvido a cada iteração.

O objetivo do controlador é minimizar a Equação (15), sujeita a restrições de desigualdade (Equação (16)), para todo $t \in [0, T]$, e à evolução do sistema (Equação (17)).

$$J(u[0,T],\, x[0,T]) \triangleq \int_0^T l\,(x(t),\, u(t),\, t)\, dt + S(x(T),\, T) \tag{15}$$

$$\begin{aligned} u_{\min} \leq u(t) \leq u_{\max} \\ g(x(t),\, u(t),\, t) \leq 0 \end{aligned} \tag{16}$$

$$\frac{d}{dt}x(t) = f(x(t),\, u(t)) \tag{17}$$

A Equação (15) é composta por dois termos: o custo de estágio, l, e o custo terminal, S. O termo T é o horizonte de predição. A evolução das variáveis de estado do sistema é dado pela função f, que depende do sinal de controle u, restrito à faixa $[u_{\min}, u_{\max}]$. Outras restrições podem ser impostas através de uma função g que depende, no caso geral, do estado e do controle. É necessário que as funções f, g e l sejam contínuas e suaves. O problema de otimização formulado pode ser resolvido a partir dos chamados *métodos diretos* (Audet, 2014), caracterizados pela discretização e parametrização do problema.

É possível utilizar diferentes equações de estado, restrições e custo. No contexto de veículos similares a carros de passeio, Wang et al. (2019) utiliza um modelo dinâmico para descrever a evolução do sistema e uma função de custo com pesos adaptativos para as diferentes variáveis de estado. Tang et al. (2020) emprega um controlador preditivo, baseado no modelo cinemático do veículo, com um controlador PID (Åström et al., 2006), para corrigir a derrapagem lateral do veículo em altas velocidades. Bai et al. (2018) também utiliza pesos adaptativos para a função de custo, mas considera apenas o modelo cinemático do veículo. Vale notar que outras estratégias de controle não-preditivo são implementadas com sucesso para o problema em questão (Chen and Zhu, 2017; Snider et al., 2009; Aguilar et al., 1997; Thrun et al., 2006).

3. MÉTODOS

A seguir são apresentados os detalhes do experimento realizado, que utiliza os algoritmos RRT e RRT* para a geração de caminhos e o controle preditivo baseado em modelo para o seguimento do caminho.

3.1 Veículo

Este artigo considera parcialmente as dimensões e limitações de um veículo autônomo terrestre, semelhante a um carro de passeio miniaturizado, criado por alunos do curso de engenharia elétrica, na Universidade do Estado do Rio de Janeiro (UERJ). A Figura 2 apresenta o robô construído e a Tabela 2 as suas dimensões e limitações.

Tabela 2. Dimensões e limitações do veículo autônomo de pequeno porte.

Parâmetro	Valor
Comprimento do veículo (A)	235,5 mm
Largura do veículo (B)	170 mm
Distância entre os eixos (L)	150 mm
Ângulo máximo ($\phi_{\max}$)	45°
Velocidade máxima ($V_{\max}$)	1 m/s

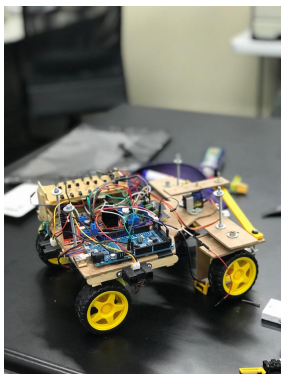

Figura 2. Veículo construído por alunos do curso de engenharia elétrica.

3.2 Mapa com Obstáculos e Objetivo

O mapa ocupado por polígonos representando obstáculos pode ser visto na Figura 3. Os obstáculos são escalonados por um fator de $1,1$ para não haver colisões no caminho ao considerar as dimensões do veículo.

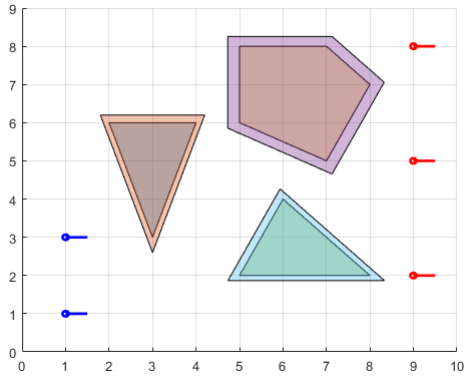

Figura 3. Os pontos azuis e vermelhos com traços indicam a posição e orientação inicial e final, respectivamente. Os polígonos em suas dimensões reais e escaladas podem ser observados.

As simulações realizadas consideram todos as pares de poses iniciais e finais da Figura 3, de forma que as finais indicam a posição e orientação desejadas. Os algoritmos RRT e RRT* realizam a busca até que um dos caminhos obtidos inclua um estado final x'_f, desde que $\rho(x'_f, x_d) < 0,01$, onde x_d representa a pose final desejada. Portanto, a tolerância utilizada na geração dos caminhos é de $0,01$ unidades.

3.3 Geração dos Caminhos

Os caminhos são gerados pelos algoritmos descritos nas Seções 2.2 e 2.3. Um método simples de diminuir o custo computacional associado à obtenção dos caminhos até um destino definido é associar uma probabilidade baixa (neste trabalho, $0,05\%$) à amostragem da pose de destino. Alguns parâmetros adicionais comuns entre às duas abordagens são descritos a seguir:

(a) Número máximo de nós: representa o número máximo de estados (ou vértices) que podem fazer parte da árvore de caminhos.
(b) Distância máxima de conexão: indica a maior distância possível entre dois vértices.
(c) Raio mínimo de curva: um veículo que se movimenta de acordo com a Equação (1) realiza um caminho circular ao se deslocar com velocidade constante e ângulo ϕ com valor máximo absoluto. Este parâmetro indica o raio deste caminho circular.

A Tabela 3 apresenta os valores desses parâmetros na simulação realizada.

Tabela 3. Parâmetros utilizados na geração dos caminhos.

Parâmetro	Valor
Número máximo de nós (N_v)	1×10^4
Distância máxima de conexão ($\rho_{\max}$)	2×10^{-1}
Raio mínimo de curva ($R_{\min}$)	3×10^{-1}

Tipicamente, os algoritmos encontram um caminho plausível até o objetivo utilizando um número de nós bem abaixo do máximo especificado. É então escolhido a interrupção das iterações, nos dois algoritmos, assim que o objetivo fosse encontrado, não permitindo otimizações adicionais nos caminhos gerados, ou a criação de novos caminhos possivelmente mais curtos.

Já que a distância entre dois estados em um caminho pode ser grande em demasia, até para um controle preditivo, utiliza-se uma interpolação dos estados que compõe os caminhos gerados. Dessa forma, cada caminho entre o estado inicial e o objetivo passa a apresentar 1000 estados em sua extensão.

3.4 Seguimento dos Caminhos

O controle preditivo minimiza o erro acumulado em um momento atual até um instante futuro (Bordons and Camacho, 2007). Portanto, é necessário definir o horizonte de predição do controlador, isto é, o horizonte de tempo no qual a otimização será realizada. Por utilizar um modelo da cinemática do robô, a predição do futuro é realizada por emulações computacionais. Deve-se, portanto, estipular um passo de tempo para essa simulação. O horizonte de predição utilizado neste trabalho é de $5\,\mathrm{s}$ e o passo de simulação $0,1s$.

A função de custo (Equação (15)) é calculada a partir dos desvios dos estados de referência, neste caso, os que compõe o caminho gerado. Utiliza-se um peso de $1,4$ para o cálculo do custo em relação às variáveis que indicam a posição, isto é, x e y, enquanto o peso atribuído para os desvios da orientação é $3,0$.

4. RESULTADOS

Com base nos métodos apresentados na Seção 3 são obtidos os caminhos da Figura 4 utilizando os algoritmos RRT e RRT*.

Para comparar o comprimento dos caminhos obtidos, executa-se cada algoritmo de planejamento um total de dez vezes, com diferentes sementes aleatórias, em cada um dos seis cenários propostos. Os resultados deste experimento podem ser vistos na Figura 5. O algoritmo RRT* apresenta desempenho melhor, como distância total do percurso, nos cenários *(c)*, *(d)* e *(f)*, enquanto o RRT é superior nos caminhos *(b)* e *(e)*. Este resultado indica que a solução do algoritmo RRT* é superior, apesar da interrupção antecipada quando a pose final é encontrada. Entretanto, a otimização caracterizada pelo processo de religação resulta em um tempo de processamento $0,09\times$ maior, como mostra a Figura 6.

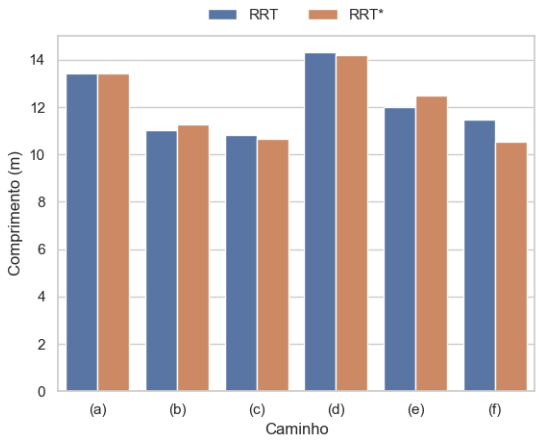

Figura 5. Comparação do comprimento dos caminhos gerados.

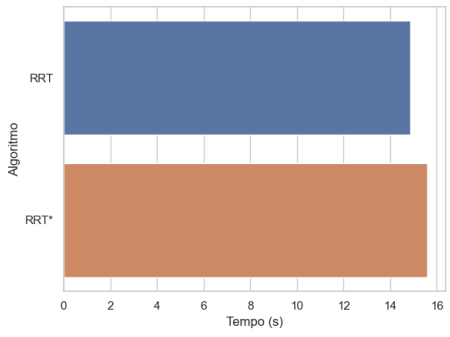

Figura 6. Tempo médio de processamento dos algoritmos em 10 execuções independentes.

O seguimento de caminho por controle preditivo é simulado utilizando os parâmetros discutidos na Seção 2.4. O desempenho do controlador em dois dos caminhos gerados é apresentado na Figura 7.

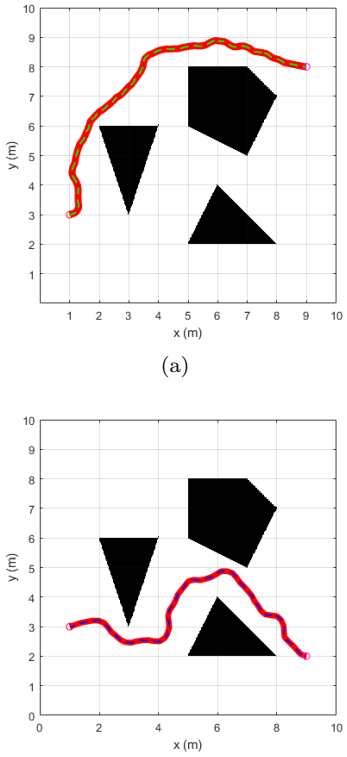

Figura 7. Controlador preditivo nos caminhos da Figura 4a e Figura 4c.

A Figura 8 apresenta a diferença entre os valores planejados para as variáveis da pose (x, y, θ), no caminho da Figura 7(a), e os valores obtidos pelo controlador. Pode-se notar que o controlador é capaz de realizar o seguimento dos caminhos com sucesso, a Tabela 4 ostenta o erro absoluto médio de seguimento de todos os caminhos gerados por cada algoritmo.

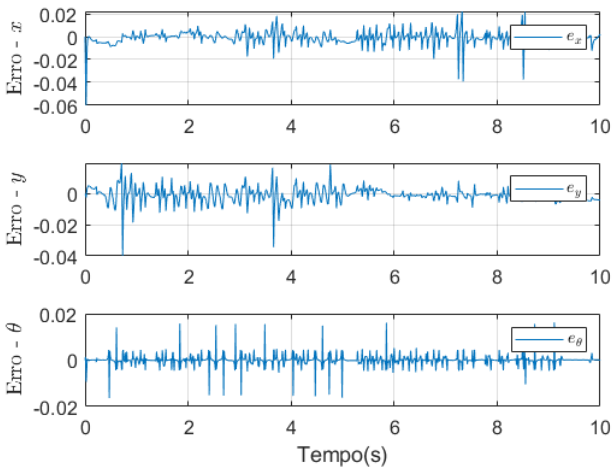

Figura 8. Erro no seguimento no caminho da Figura 7(a).

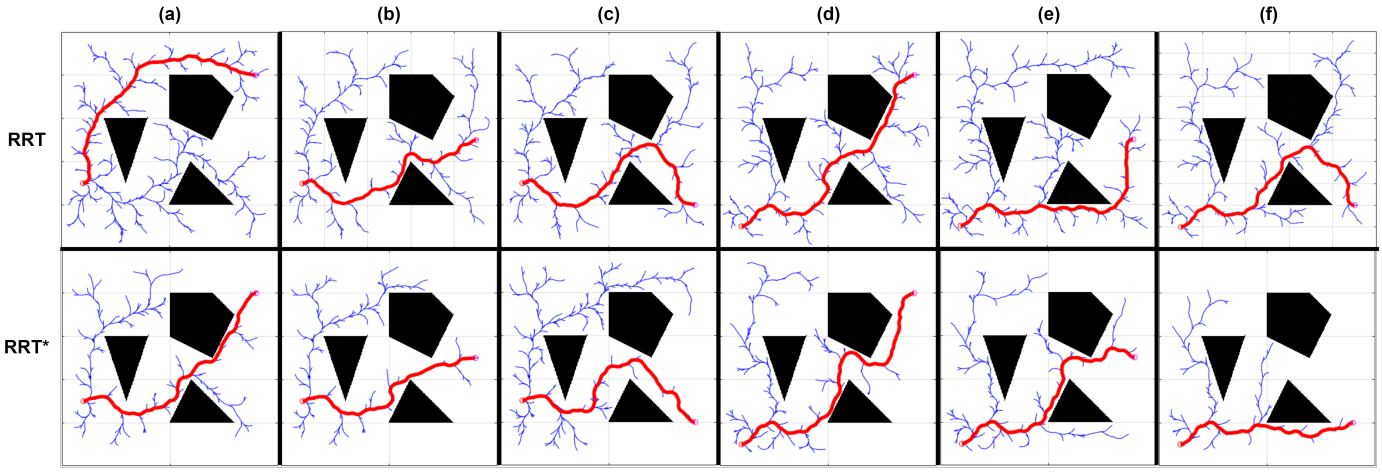

Figura 4. Caminhos gerados pelos algoritmos RRT e RRT* em diferentes posições iniciais e finais.

Tabela 4. Erro absoluto médio de seguimento dos caminhos.

	RRT			RRT*		
	x (cm)	y (cm)	θ (rad)	x (cm)	y (cm)	θ (rad)
(a)	0,3565	0,3157	0,1301	**0,2709**	**0,2170**	**0,1044**
(b)	0,3742	0,2544	0,0794	**0,2640**	**0,1833**	**0,0639**
(c)	0,3610	0,2837	0,1131	**0,2966**	**0,2335**	**0,0947**
(d)	**0,3380**	**0,2978**	0,1253	0,3593	0,3367	**0,1164**
(e)	0,3603	0,2587	0,1065	**0,2970**	**0,2340**	**0,0928**
(f)	0,3031	0,2518	0,1067	**0,2425**	**0,2120**	**0,0561**

Por fim, a Tabela 5 mostra a diferença absoluta entre a pose final desejada e o que é obtido por cada algoritmo de planejamento e seguimento de caminho. É possível notar que os resultados obtidos estão próximos dos objetivos propostos. Quanto ao critério de menor erro absoluto médio de seguimento dos caminhos e a diferença com a pose final desejada, o algoritmo RRT* obtém geralmente o melhor desempenho.

Tabela 5. Diferença entre a pose final desejada e a obtida pelo planejamento e seguimento do caminho.

	RRT			RRT*		
	x (cm)	y (cm)	θ (°)	x (cm)	y (cm)	θ (°)
(a)	0,7390	0,3855	1,8566	**0,3212**	**0,0168**	**0,0650**
(b)	0,6914	**0,0278**	1,7494	**0,3121**	0,2116	**0,1068**
(c)	0,6995	**0,0921**	1,8109	**0,4101**	0,2514	**0,0424**
(d)	0,6597	0,4950	1,7518	**0,2644**	**0,4345**	**0,0771**
(e)	0,6584	0,2252	1,7226	**0,3043**	**0,1086**	**0,0662**
(f)	0,7534	0,4678	1,9124	**0,4533**	**0,1098**	**0,0500**

5. CONCLUSÃO

Este trabalho realiza o planejamento de caminhos a partir de dois métodos que utilizam árvores de exploração aleatória. A modelagem do veículo é baseada no tipo bicicleta e adaptado para a geometria de Ackermann para evitar derrapagem. É feita uma comparação do algoritmo RRT com o RRT* em comprimento total de diversos caminhos e observa-se uma ligeira vantagem para o método RRT*, mesmo quando não se consideram otimizações seguintes ao descobrimento da pose de destino. O seguimento dos caminhos gerados é realizado por ação de controle preditivo, que permite o rastreamento preciso dos caminhos produzidos. Pode-se concluir que o método proposto neste trabalho é eficaz para problemas em que se considera a navegação de robôs em locais com obstáculos. Recomenda-se que investigações futuras incluam o detalhamento na utilização do modelo dinâmico e a validação experimental.

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001. Os autores agradecem, também, à Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro (FAPERJ) pelos seus suportes financeiros no desenvolvimento deste trabalho.

REFERÊNCIAS

Aguilar, L.E., Hamel, T., and Soueres, P. (1997). Robust path following control for wheeled robots via sliding mode techniques. In *Proceedings of the 1997 IEEE/RSJ International Conference on Intelligent Robot and Systems. Innovative Robotics for Real-World Applications. IROS'97*, volume 3, 1389–1395. IEEE.

Argento, E.V., Vieira, R.P., and Revoredo, T.C. (2020). Estacionamento paralelo autônomo de veículos leves de passeio. *Anais da Sociedade Brasileira de Automática*, 2(1).

Åström, K.J., Hägglund, T., and Astrom, K.J. (2006). *Advanced PID control*, volume 461. ISA-The Instrumentation, Systems, and Automation Society Research Triangle Park.

Audet, C. (2014). A survey on direct search methods for blackbox optimization and their applications. *Mathematics without boundaries*, 31–56.

Badue, C., Guidolini, R., Carneiro, R.V., Azevedo, P., Cardoso, V.B., Forechi, A., Jesus, L., Berriel, R., Paixao, T.M., Mutz, F., et al. (2021). Self-driving cars: A survey. *Expert Systems with Applications*, 165, 113816.

Bai, G., Meng, Y., Gu, Q., Luo, W., and Liu, L. (2018). Path tracking of car-like vehicles based on variable weight model predictive control. In *2018 Chinese Automation Congress (CAC)*, 1943–1947. IEEE.

Bordons, C. and Camacho, E. (2007). *Model predictive control*. Springer Verlag London Limited.

Chen, Y. and Zhu, J.J. (2017). Car-like ground vehicle trajectory tracking by using trajectory linearization control. In *Dynamic Systems and Control Conference*, volume 58288, V002T21A014. American Society of Mechanical Engineers.

Dubins, L.E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3), 497–516.

Gillespie, T.D. (1992). Fundamentals of vehicle dynamics. *SAE International*.

Guimarães Batista, J. (2020). Trajectory planning using artificial potential fields with metaheuristics. *IEEE Latin America Transactions*, 18(5), 914–922. URL `https://latamt.ieeer9.org/index.php/transactions/article/view/2430`.

Hartani, K., Bourahla, M., Miloud, Y., and Sekour, M. (2009). Electronic differential with direct torque fuzzy control for vehicle propulsion system. *Turk J Elec Eng end Comp Sci*, 17(1).

Johansen, T.A. (2011). Introduction to nonlinear model predictive control and moving horizon estimation. *Selected topics on constrained and nonlinear control*, 1, 1–53.

Karaman, S. and Frazzoli, E. (2010). Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104(2).

Karaman, S., Walter, M.R., Perez, A., Frazzoli, E., and Teller, S. (2011). Anytime motion planning using the rrt*. In *2011 IEEE International Conference on Robotics and Automation*, 1478–1483. doi:10.1109/ICRA.2011.5980479.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1), 90–98. doi:10.1177/027836498600500106. URL `https://doi.org/10.1177/027836498600500106`.

Kuwata, Y., Karaman, S., Teo, J., Frazzoli, E., How, J., and Fiore, G. (2009). Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5), 1105–1118. doi:10.1109/tcst.2008.2012116. URL `https://doi.org/10.1109/tcst.2008.2012116`.

Lau, B., Sprunk, C., and Burgard, W. (2009). Kinodynamic motion planning for mobile robots using splines. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. doi:10.1109/iros.2009.5354805. URL `https://doi.org/10.1109/iros.2009.5354805`.

LaValle, S.M. (1998). Rapidly-exploring random trees: A new tool for path planning. *California State University, Stanislaus*.

Lee, W., Alkouz, B., Shahzaad, B., and Bouguettaya, A. (2021). *Package Delivery Using Autonomous Drones in Skyways*, 48–50. Association for Computing Machinery, New York, NY, USA. URL `https://doi.org/10.1145/3460418.3479289`.

Ma, Y., Wang, Z., Yang, H., and Yang, L. (2020). Artificial intelligence applications in the development of autonomous vehicles: a survey. *IEEE/CAA Journal of Automatica Sinica*, 7(2), 315–329.

MATLAB (2018). *(R2018a)*. The MathWorks Inc., Natick, Massachusetts.

Nascimento, L.B., Santos, V.G., Pereira, D.S., Fernandes, D.H., and Alsina, P.J. (2019). Planejamento de caminho para sistemas robóticos autônomos. *Sociedade Brasileira de Computação*.

Pinheiro, B.C. (2009). Sistema de controle tempo real embarcado para automação de manobra de estacionamento. *Dissertação de Mestrado em Automação e Sistemas ,Universidade Federal de Santa Catarina*.

Reeds, J. and Shepp, L. (1990). Optimal paths for a car that goes both forwards and backwards. *Pacific journal of mathematics*, 145(2), 367–393.

Revoredo, T.C., Mora-Camino, F., and Slama, J. (2016). A two-step approach for the prediction of dynamic aircraft noise impact. *Aerospace Science and Technology*, 59, 122–131. doi:10.1016/j.ast.2016.10.017. URL `https://doi.org/10.1016/j.ifacol.2016.03.121`.

Siegwart, R. and Nourbakhsh, I.R. (2004). *Introduction to Autonomous Mobile Robots*. Intelligent robots and autonomous agents. MIT Press, London, England.

Snider, J.M. et al. (2009). Automatic steering methods for autonomous automobile path tracking. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*.

Sousa, S.K.A.d. et al. (2017). Planejamento de movimento para robôs móveis baseado em uma representação compacta da rapidly-exploring random tree (rrt). *Repositório Institucional da Universidade Federal de Sergipe*.

Tang, L., Yan, F., Zou, B., Wang, K., and Lv, C. (2020). An improved kinematic model predictive control for high-speed path tracking of autonomous vehicles. *IEEE Access*, 8, 51400–51413.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., et al. (2006). Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, 23(9), 661–692.

Vieira, R., Argento, E., and Revoredo, T. (2022). Trajectory planning for car-like robots through curve parametrization and genetic algorithm optimization with applications to autonomous parking. *IEEE Latin America Transactions*, 20(2), 309–316. doi:10.1109/tla.2022.9661471. URL `https://doi.org/10.1109/tla.2022.9661471`.

Wang, H., Liu, B., Ping, X., and An, Q. (2019). Path tracking control for autonomous vehicles based on an improved mpc. *IEEE Access*, 7, 161064–161073.

Zhu, C. and Rajamani, R. (2006). Global positioning system-based vehicle control for automated parking. *I MECH E Part D Journal of Automobile Engineering*, 220(1), 37–52.