Aplicação do Algoritmo Rapidly-exploring Random Trees star ao planejamento automático de trajetórias para um robô Não-Holonômico

Taylane O. Santos[1], Luã Malco Da C. Souza[1], Luiz Carlos Simões S. Júnior[1], Gildeberto de S. Cardoso[1], José Valentim dos S. Filho[1]

[1]Centro de Ciências Exatas e Tecnologicas – Universidade Federal do Recôncavo da Bahia (UFRB)
Caixa Postal 44380-000 – Cruz das Almas – BA – Brazil

taytaylane13@hotmail.com, luamalco1@gmail.com, lcsimoes@ufrb.edu.br,

gildeberto@ufrb.edu.br, valentim@ufrb.edu.br,

Abstract. *The incessant search for technological devices has become a very growing factor in the current scenario. The use of technology applications in everyday activities can easily prove and justify that this advance is based on human needs, which in turn are increasingly linked to the ease and precision in performing previously inefficiently performed tasks. In view of this, the present work seeks to develop an automatic trajectory system based on the Rapidly-exploring Random Tree Star (RRT^*) planning technique, to be applied to a non-holonomic mobile robot, in order to results that can contribute to this evolution. For this purpose, the RGB camera of Kinect was used, as well as its depth sensor, to extract the characteristics of the environment and to allow the creation of the map to be used later in the algorithm of the RRT^*.*

Resumo. *A busca incessante por dispositivos tecnológicos tornou-se fator bastante crescente no cenário atual. O uso das aplicações da tecnologia em atividades cotidianas podem comprovar e justificar facilmente que esse avanço baseia-se nas necessidades humanas, que por sua vez estão cada vez mais ligadas a facilidade e precisão na execução de tarefas antes feitas de forma pouco eficazes. Visando isto, o presente trabalho busca desenvolver um sistema automático de geração de trajetórias baseado na técnica de planejamento Rapidly-exploring Random Tree Star (RRT^*), a ser aplicado a um robô móvel não-holonômico, a fim de obter resultados que possam contribuir para essa evolução. Para isso foi utilizada a câmera RGB do Kinect, bem como seu sensor de profundidade, para extração de características do ambiente e possibilitar a criação do mapa a ser utilizado posteriormente no algoritmo da RRT^*.*

1. Introdução

No âmbito da robótica móvel, mais especificamente a que se relaciona com veículos autônomos, existe uma gama de pesquisas ativas a fim de encontrar uma melhor forma de possibilitar a locomoção desses veículos em ambientes destruturados. Uma parte das técnicas usadas se refere ao planejamento prévio da trajetória no qual o veículo irá percorrer, que é o enfoque deste artigo.

De modo geral, planejadores de trajetória buscam, dado um espaço de configurações contendo obstáculos, encontrar um caminho factível entre uma

configuração inicial x_{init}, e uma final, x_{goal} para o robô sem que ele colida com os obstáculos. Entretanto, salvas algumas situações, o problema mostra-se complexo [Choset 2005]. Nesse artigo o sistema no qual será aplicado o planejamento de trajetória se restringe a um robô não-holonômico, ou seja, robô que se caracteriza por se locomover seguindo modelos cinemáticos com restrições em seu movimento.

Além disso, algumas técnicas de planejamento de trajetória são caracterizadas por serem baseadas em amostragem, e, por isso, possuem grande vantagem no que diz respeito a sua utilização em sistemas que apresentam alto grau de liberdade [Vaz 2011]. A RRT^* (Rapidly-exploring Random Trees Star) então está contida nessa classe, seu funcionamento se dá amostrando estados aleatórios no espaço de configurações, formando a "árvore de estados", e simultaneamente selecionando os estados que pertencerão ao caminho a ser seguido pelo robô.

Para que a RRT^* produza a trajetória final, é preciso que informações acerca do espaço de trabalho sejam conhecidas. Uma vantagem em seu uso como planejador de trajetória está no fato de não ser necessário que o espaço de configurações seja descrito fielmente ao ambiente, isso então desencadeia menor custo computacional na execução do algoritmo [Choset 2005].

Dessa forma, para obter as informações a respeito do ambiente, foram utilizadas uma imagem RBG do ambiente em que estava o robô e uma "imagem de profundidade", contendo as distâncias do sensor até cada elemento presente na cena. Nessas imagens, técnicas de processamento de imagens, como a limiarização, foram utilizadas para que características - como os centróides dos obstáculos e robô - fossem obtidas.

No que diz respeito a sua eficiência, segundo [Karaman and Frazzoli 2010] a RRT^* se mostra como uma técnica de planejamento assintoticamente ótima, além de melhorar sua resposta ao longo do tempo por conseguir fazer reconexões dos estados pertencentes à arvore a fim de reduzir o custo de navegação.

Assim, o objetivo desse trabalho é o desenvolvimento de um sistema automático de planejamento de trajetórias baseado em visão computacional e no método *RRT**. Em outras palavras, o sistema deve identificar regiões livres de obstáculos no ambiente e o ponto de partida do robô utilizando métodos de visão computacional e a partir disso, gerar uma trajetória ótima até um outro ponto escolhido no ambiente. Dessa forma, esse sistema pode posteriormente ser parte do sistema de navegação de um robô móvel não-holonômico.

2. Materiais e Métodos

Nesta sessão serão apresentadas as estratégias utilizadas para a detecção do robô, obstáculos e planejamento da trajetória a ser seguinda.

2.1. Captura das imagens

Para captura das imagens RGB e de profundidade foi utilizado o Microsoft Kinect XBOX 360 fixado ao teto do ambiente de testes a uma altura de 2,5 m. O robô utilizado foi o Qbot 2 for QUARC, robô terrestre autônomo, equipado com sensores e um sistema de visão.

A câmera RGB do Kinect fornece ao ambiente de simulação utilizado, Matlab, uma matriz tridimensional onde cada posição dessa matriz (pixel da imagem), contém uma tupla composta por valores em R, G e B. Já o sensor de profundidade fornece uma matriz bidimensional onde cada posição da matriz contém a distância do sensor ao objeto.

A posição do centro geométrico do robô no ambiente (x_{init}) e orientação (θ_{init}), foram identificadas utilizando a imagem RBG. Para tanto, dois círculos de referência com cor pré-definida foram posicionados no robô, como mostra a Figura 1, onde o circulo vermelho encontra-se posicionado no centro geométrico do robô. Por inspeção, os intervalos de cor em R, G e B foram determinados para ambos os círculos e um filtro de cor retornou novas imagens com apenas os pixels que estavam contidos no intervalo. Foi possível perceber que essas imagens continham ruídos, e que eles apesentavam uma área menor que a do círculo procurado. Com isso, elas foram binarizadas, e através de um limiar de área, esses ruídos foram eliminados possibilitando determinar x_{init}, utilizando o círculo vermelho, x_{front}, o amarelo e θ_{init}, o coeficiente angular da equação da reta que os liga.

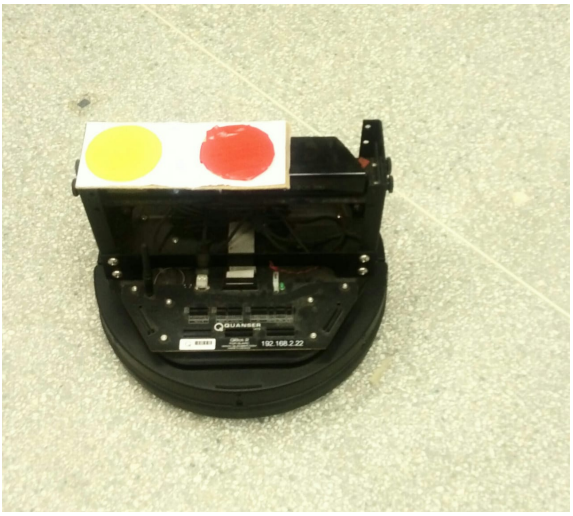


Figura 1. Robô (Qbot2 - Quanser) no ambiente de trabalho.

O sensor de profundidade foi utilizado para detecção dos obstáculos. Analisando a matriz de distâncias e conhecendo a distância do Kinect ao piso, foi definido um limite seguro de altura para o qual pode-se considerar uma região sem obstáculos, X_{free}, como proposto por [Santana et al. 2016]. Como já havia sido obtida a posição x_{init} e o raio do robô, a parcela da área que continha o robô na matriz de profundidade foi substituída por *NaN* para evitar possíveis conflitos. Pôde-se então construir uma matriz binária, mapa do ambiente, onde aos pontos livres de obstáculos foi atribuído o valor 0 e os pontos onde foram identificados obstáculos foi atribuído o valor 1. Os possíveis ruídos da matriz binarizada foram eliminados por uma filtragem por limiar de área.

O mapa do ambiente possibilitou a obtenção das propriedades acerca dos agrupamentos de 1's (obstáculos), como posições dos centros geométricos e raios.

2.2. Espaço de configurações - X

Considerando que o robô se locomove em um ambiente plano, R^2, esse ambiente pode ser descrito matematicamente como um conjunto de pontos possíveis que o robô pode

assumir no ambiente no qual está inserido. Dessa forma, $X \in R^2$ é chamado espaço de configurações do robô se, e somente se, o robô nesse espaço puder ser descrito por um ponto de modo a facilitar o planejamento de sua trajetória [LaValle 1998].

Dada uma região, X_{obs}, que descreve o espaço contendo obstáculos em X, X_{free} é referido como todos os pontos $x \in X$, tal que $x \notin X_{obs}$. É válido salientar que X é limitado, assim como seus subconjuntos X_{free} e X_{obs}. Também, em [Choset 2005], o espaço de trabalho é diferenciado do espaço de configurações para que se obtenha X_{obs}, assim, para que o robô seja descrito como um ponto x_{init}, uma fronteira é adicionada ao redor dos obstáculos com raio $R_{obs(i)}$ + R, onde $R_{obs(i)}$ é o raio da menor circunferência que circunscreve o obstáculo i e R o raio da menor circunferência que circunscreve o robô.

2.3. Planejamento de trajetória baseado em amostragem

Alguns dos métodos de planejamento de trajetória são baseados em amostragem. Assim, dado *X*, x_{init} e x_{obs}, busca-se encontrar um caminho entre esses dois estados a partir de pontos gerados aleatoriamente dentro do espaço de configurações. Essa forma de planejamento tem sido utilizada por trazer maior simplicidade ao problema visto que desse modo não é necessário explicitar todas as características do ambiente além de permitir o planejamento para problemas com altos graus de liberdade [Vieira 2014].

Planejadores de trajetória levam em consideração as informações necessárias do caminho a ser seguido por um robô específico, incluso restrições de movimento do próprio robô. Para tal, [LaValle 1998] propôs um método de planejamento cinemático-dinâmico de trajetória, *Rapidly exploring random tree* (RRT^*), onde pode-se levar em consideração tais restrições.

Assim como outros planejadores de trajetória baseados em amostragem, o algoritmo RRT se apresenta como uma variação do *Mapa de caminhos probabilísticos* (PRM). O PRM é um conjunto de estados, gerados aleatoriamente, dentro do espaço de configurações livre, onde todos estão conectados entre si. Essa técnica, abordada por [Kavraki et al. 1996], mostrou a possibilidade do uso de um mapa contendo estados aleatórios, onde, a partir dele, se conseguisse encontrar um caminho factível entre dois estados.

3. Rapidly exploring random tree star (RRT^*)

Dentre os vários métodos de planejamento de caminho baseado em amostragem, o algoritmo RRT também se destaca. Apresentado por [LaValle 1998] como um algoritmo de busca única, ou seja, retorna uma solução única para o problema, sua utilização atualmente é bem vista por apresentar soluções para problemas relativamente complexos utilizando um algoritmo simples. Outro fator está na simplicidade de dados do ambiente que serão entrada para o algoritmo: o ambiente no qual se quer descobrir a trajetória não precisa ser detalhado de forma a ser fiel ao ambiente real.

Proposta por [Karaman et al. 2011], a RRT^* apresenta uma melhoria em relação ao algoritmo anterior, RRT. No RRT, dado um espaço, X, contendo obstáculos e um estado inicial x_{init}, referente a posição inicial do robô, a árvore é inicializada, e x_{init} é adicionado ao grafo. Aleatoriamente um outro estado x_{rand} é amostrado de forma que esteja contido no espaço livre de obstáculos, $X_{free} \subset X$. Caso pertença, o nó mais próximo

de x_{rand}, x_{near}, é encontrado a partir de uma métrica, ρ, no espaço de estados. Tendo definido x_{near}, x_{new} é determinado utilizando uma função que limita a distância de x_{near} até x_{rand} tal que $x_{new} \in X_{free}$. Então, x_{new} é adicionado a árvore assim como a aresta que liga x_{near} a x_{new} também é adicionada.

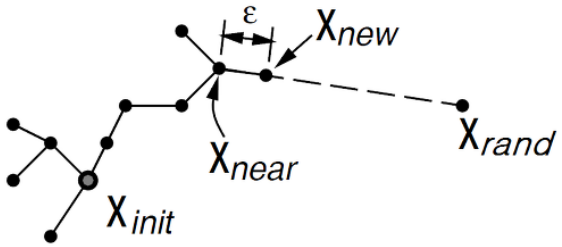


Figura 2. Expansão da árvore.
Fonte: [LaValle and Kuffner Jr 2000]

Neste ponto o algoritmo RRT^* difere do RRT. No RRT^*, após a inserção de x_{new} à arvore, outra verificação é adicionada ao algoritmo. O ponto x_{new} é atribuído como o centro de uma região $V(x_{new}, r)$ circular (ou esférica, se X $\in R^3$), sendo r o raio dessa região, definido por uma função que o diminui gradativamente a cada iteração.

A região V é utilizada para gerar um novo estado, x_{min}, definido como "pai"de x_{new} tal que a distância do trajeto que liga x_{init} até x_{new}, passando por x_{min}, seja a menor quando comparada a todos os outros nós $v_j \in V$. Para isso, uma função custo é utilizada. Essa verificação é feita a cada novo nó x_{new} adicionado a árvore, e com isso, a cada iteração, o grafo é atualizado, e os nós antecessores de x_{new}, nós pais, são substituídos por outros que retornem menor custo quando comparada à iteração anterior, o que fornece uma convergência maior dos pontos para x_{goal}.

Vale ressaltar que o critério a parada do algoritmo depende de duas condições: (1) atingir o número de iterações estipuladas como entrada ou (2) um nó, x_{dest}, atingir uma região próxima suficientemente de x_{goal}. Caso essas duas condições não aconteçam, o algoritmo repete todo o processo até que o software finalize retornando a solução.

Segundo [Lumelsky and Stepanov 1987], um algoritmo pode ser dito completo quando, em tempo finito, retorna uma solução (nesse caso, um caminho), ou apresenta uma resposta de falha, caso não haja solução. Sabendo isso, é notório que o algoritmo RRT (e, por sua vez, o RRT^*) não pode estar nessa classificação por haver possibilidade de não retornar solução em tempo finito e entrar em um ciclo infinito caso não haja solução. Analisando isso, [Karaman et al. 2011] propôs que o RRT é probabilisticamente completo, ou seja, dado um problema, a chance do algoritmo retornar uma solução tende para 1 quando o seu tempo de execução, ou melhor, o número de iterações tende para o infinito. Além disso, no RRT^*, quanto maior for o tempo de execução , mais aproximada da trajetória ideal será a resposta, motivo pelo qual [Karaman et al. 2011] classificaram como assintoticamente ótimo.

4. Resultados e discussões

4.1. Mapa do ambiente

A RRT^* necessita de informações a respeito do ambiente para que as trajetórias sejam geradas na parcela do espaço que não apresenta obstáculos. Dessa forma, a imagem RGB do Kinect foi utilizada, mostrada na figura 3, bem como uma matriz de profundidade.

Figura 3. Imagem RGB do ambiente.

Inicialmente, um filtro de cor foi utilizado para retirar da imagem qualquer outra cor que não possuísse valor dentro do intervalo no qual a cor do círculo vermelho está definida.

Após a aplicação do filtro, a imagem resultante foi binarizada, como mostrada na Figura 4.a. Observando-a, foi perceptível que além dos pixels que representavam o círculo procurado, a imagem também possuía outros pixels esparsos que representavam ruídos. Sabendo que o círculo procurado apresentava na imagem um valor fixo de área, e fazendo dele um limiar de área, esses ruídos foram retirados, como visto na na Figura 4.b.

(a) Binarizada (b) Após filtragem por área (c) Em verde x_{init}

Figura 4. Filtragem da imagem RGB.

A partir da imagem resultante do filtro de área foi possível obter a posição do centro geométrico do conjunto de pixels brancos que é aproximadamente o mesmo do centro do robô.

O mesmo procedimento foi reaplicado a imagem original, e x_{front} foi determinado como mostrado na figura 5, para encontrar a posição do centro geométrico do círculo de

cor amarela. Então, com esses dois pontos o ângulo da reta que passa por eles é θ_l e $\theta_{init} = 90^o - \theta_l$, que é aproximadamente o mesmo ângulo de orientação do robô com relação ao eixo das abscissas.

Figura 5. Em verde, "x"que marca x_{front}.

Para obter as informações a respeito dos obstáculos, foi utilizado o sensor de profundidade do Kinect. Com ele é possível obter uma matriz do ambiente, onde cada posição dessa matriz é a mesma posição no R^2 da imagem RGB, mas, contendo valores da distância do sensor até cada objeto contido naquele pixel.

Observando as características do ambiente, foi definido então um limiar de distância, e a partir de um filtro por limiarização, os valores de distância acima desse limiar foram retirados da matriz. Para uma parcela dessa matriz de profundidade, foi atribuído valor nulo nas posições que eram as mesmas dos pixels da imagem do robô, utilizando para tal, a posição de seu centróide e raio. Assim, a imagem resultante está mostrada na figura 6.

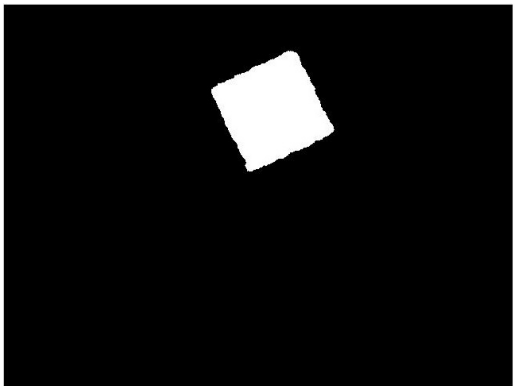
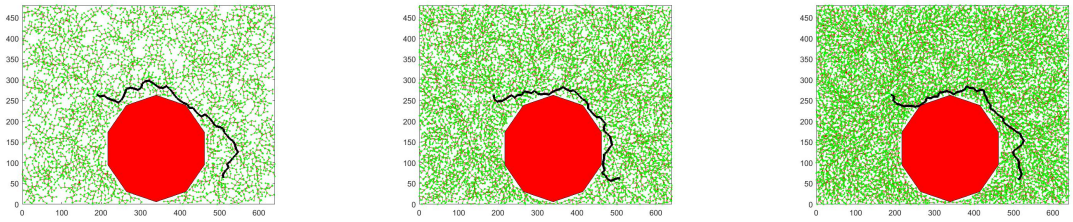


Figura 6. Imagem binarizada após a filtragem.

Analogamente a forma feita na imagem RGB, propriedades acerca dos agrupamentos de pixels dos objetos foram encontradas, sendo elas: suas posições e também o raio da menor circunferência que os circunscreve. Dessa forma, o mapa do ambiente foi descrito como uma estrutura contendo as informações obtidas (e necessárias) a respeito do ambiente, podendo então ser utilizado no algoritmo da RRT^*.

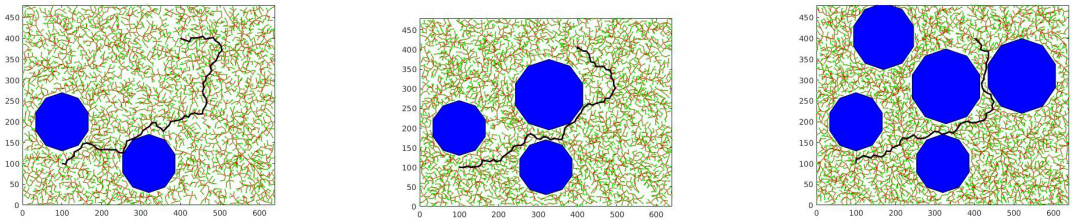
4.2. Aplicação do algoritmo RRT^*

Com o mapa do ambiente, pôde-se então executar o algoritmo da RRT^*. Para isso, o algoritmo, além de fazer o planejamento, também expande a área dos obstáculos acrescendo ao raio deles, o raio do robô. A figura 7 mostra o resultado após a execução do algoritmo. Em verde, foi plotada as arestas dos vértices gerados; em vermelho, o obstáculo do ambiente; e em preto, a trajetória escolhida como ótima.



(a) 4000 iterações (b) 8000 iterações (c) 12000 iterações

Figura 7. Trajetórias geradas pela RRT^*.



(a) 2 obstáculos (b) 3 obstáculos (c) 5 obstáculos

Figura 8. Trajetórias geradas pela RRT^* com 5000 iterações.

Como já citado, o algoritmo da RRT^* é classificado como assintóticamente ótimo por tender a solução do problema a medida que há um aumento no número de iterações. Isso pôde ser percebido, como mostra o gráfico da Figura 9, que com o aumento do número de iterações, o custo (nesse caso, o tamanho do percurso) aproximou-se de sua linha assintótica.

Na figura 8 pode-se perceber também que a aplicabilidade do sistema desenvolvido é válida não apenas para situações triviais, mas também para aquelas em que há vários obstáculos no ambiente, ou ainda quando esses obstáculos estão bem próximos uns dos outros.

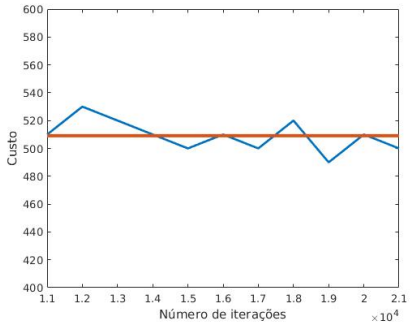


Figura 9. Gráfico: Custo x Número de iterações

É perceptível também nos gráficos das Figuras 10 e 9 que é válido para esse sistema sua caracteristica de completude. Ao passo que o algoritmo tendia para a solução de menor custo, crescia seu tempo de execução, fato que o reafirma como probabilisticamente completo.

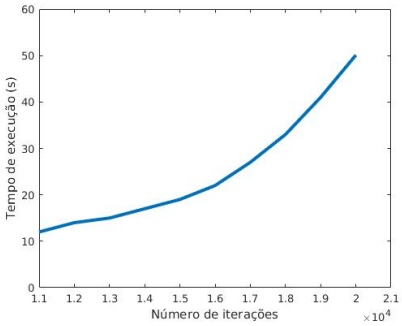


Figura 10. Gráfico: Tempo de execução x Número de iterações

5. Conclusões

Durante este trabalho foi notório que a técnica de planejamento utilizada foi eficiente quando aplicada para um robô móvel. Além disso, a utilização do Kinect como dispositivo de aquisição de dados, bem como a etapa de manipulação desses dados, mostrou-se simples e de rápido tempo execução.

Pretende-se, como trabalho futuro, utilizar o resultado da etapa de planejamento para um controlador baseado em lógica Fuzzy a fim de possibilitar de fato a locomoção do robô no ambiente real através da trajetória gerada.

Agradecimentos

GEAR: grupo de energia, automação e robótica; FAPESB: fundação de amparo à pesquisa no estado da Bahia.

Referências

Choset, H. M. (2005). *Principles of robot motion: theory, algorithms, and implementation*. MIT press.

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. Institute of Electrical and Electronics Engineers.

Kavraki, L. E., Kolountzakis, M. N., and Latombe, J.-C. (1996). Analysis of probabilistic roadmaps for path planning. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 4, pages 3020–3025. IEEE.

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

LaValle, S. M. and Kuffner Jr, J. J. (2000). Rapidly-exploring random trees: Progress and prospects.

Lumelsky, V. J. and Stepanov, A. A. (1987). Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2(1-4):403–430.

Santana, T., d Cardoso, Gildeberto, F., and Valentim, J. (2016). Detecção de obstáculos a partir do kinect no planejamento de trajetórias do rapidly-exploring random tree star (rrt*).

Vaz, D. A. B. d. O. (2011). *Planejamento de movimento cinemático-dinâmico para robôs móveis com rodas deslizantes*. PhD thesis, Universidade de São Paulo.

Vieira, H. L. (2014). *Redução do custo computacional do algoritmo RRT através de otimização por eliminação*. PhD thesis, Universidade de São Paulo.