Construção de um framework de planejamento e controle de trajetória em tempo real de múltiplos robôs terrestres

Johnathan Fercher da Rosa e Paulo Fernando Ferreira Rosa

Instituto Militar de Engenharia, IME, PGED, Brasil
johnathanfercher22.gmail.com,rpaulo@ime.eb.br
http://pged.ime.eb.br/

Resumo Nesse trabalho, é proposto um *framework* de planejamento de controle de trajetória em tempo real de múltiplos robôs terrestres. O *framework* é capaz de controlar um conjunto de robôs terrestres omnidirecionais, encontrando e guiando cada robô por uma trajetória que liga suas poses iniciais a suas poses finais, de maneira a evitar colisão com obstáculos fixos, móveis e entre robôs. São utilizadas duas etapas de processamento, uma offline, onde é calculada uma trajetória simples utilizando a biblioteca OMPL e outra online, onde os robôs são guiados pela trajetória utilizando a técnica campos potenciais. Foi obtido uma boa taxa de resolução de problemas, entretanto o desvio de obstáculos necessita ser aprimorado.

Keywords: path planning, multiple robots, ompl, potential fields, PID control

1 Introdução

O processamento de planejamento faz parte dos três pilares da robótica. Para que um robô esteja apto a realizar qualquer tarefa, o mesmo precisa sentir o ambiente com auxílio de sensores, planejar uma ação e atuar no ambiente. Quando trata-se de robótica móvel, a etapa de planejamento também é responsável pelo planejamento de trajetória. Com o crescimento dessa área, o problema começou a ser expandido, como pode ser visto em [1], considerando incertezas, múltiplos obstáculos, múltiplos robôs e dinâmica.

Visto que esse problema vem sendo estudado por décadas, propõe-se aqui um framework de planejamento de trajetória de múltiplos robôs em ambientes dinâmicos. O framework é capaz de controlar um conjunto de robôs terrestres omnidirecionais, encontrando e guiando cada robô por uma trajetória que ligue suas poses iniciais a suas poses finais, de maneira a evitar colisão com obstáculos fixos e móveis e entre os próprios robôs do conjunto. O framework não se preocupa em gerar as poses iniciais e finais, apenas em gerar um caminho entre duas poses.

Planejar pode ser definido como, encontrar uma sequência de estados que leve um estado inicial a um estado final. Diversas abordagens buscam planejar

todo o caminho que um robô deve executar antes do mesmo começar a se locomover, porém essa abordagem de resolução, conhecida como offline, demonstra-se intratável para um problema de múltiplos robôs, pois como pode ser visto em [2], tal abordagem apresenta um grande crescimento computacional. Também existe a questão de que o problema tratado refere-se a ambientes dinâmicos, o que impossibilita prever todos os estados transitórios entre um estado inicial e um final.

O framework descrito neste trabalho propõe abordar o problema de planejamento de trajetória de múltiplos robôs, isto é, planejar e guiar um conjunto de pelo menos 4 robôs por trajetórias que ligue suas poses iniciais e suas poses de interesse. Além de realizar essa tarefa, o framework foca em realizá-las em tempo real, isto é, o tempo necessário para planejar a trajetória do grupo de robôs, não deve atrasar a tarefa da etapa *online* de guiá-los.

1.1 Motivação

A utilização de múltiplos robôs em diversas tarefas vem se tornando algo comum, pois em diversas aplicações a utilização de múltiplos robôs se demonstra vantajosa ou necessária, por exemplo, em [3] a tarefa de vencer uma partida de futebol de robôs, não pode ser executada por apenas um único robô. Em [4], a utilização de dois robôs equipados com a plataforma FastSLAM (Mapeamento e Localização Simultâneos) executa a tarefa de mapeamento de forma mais eficiente em comparação a um único robô.

A execução do planejamento e controle de trajetória em tempo real gera um ganho para uma plataforma robótica, pois abre possibilidade de executar tarefas mais complexas em paralelo, além de possibilitar a construção de um sistema mais resiliente, que evite situações indesejáveis, como, colisões e deadlock.

Um framework que considere um ambiente dinâmico para realizar as ações de planejamento e controle de trajetória, está o tempo todo verificando o ambiente e considerando obstáculos móveis que podem por ventura se aproximar demais causando situações de perigo, assim, o framework fornece maior segurança na movimentação dos robôs.

1.2 Definição do Problema

Seja um espaço euclidiano W, denominado espaço de trabalho e representado por $\mathbb{R}^2$, seja a posição e orientação que um robô pode ocupar em W representada por uma pose $p = [x \quad y \quad \theta]^T$, seja uma trajetória contínua que ligue duas poses representada por uma sequência de poses $P = \{p_1, p_2, ..., p_n\}$, onde p_1 é uma pose inicial e p_n uma pose final, seja um robô representado por $r = [pose \quad raio]^T$, um obstáculo representado por $o = [x \quad y \quad raio]^T$ e um estado do espaço de trabalho representado por $E = \{R, O\}$ onde $R = \{r_1, r_2, ..., r_n\}$ é um conjunto de robôs controláveis e O o conjunto de obstáculos, assumindo que a forma de R e O são conhecidas e que O possui obstáculos fixos e dinâmicos. O problema pode ser definido como:

Para cada $r_n \in R$, encontrar uma trajetória P_n que ligue a pose atual p_n^a e a pose objetivo p_n^o, de maneira a evitar contato com B e com os demais robôs em R.

Este trabalho está organizado em mais 4 seções, na seção II é citado os trabalhos relacionados a este e como se relacionam, na seção III é apresentado o framework desenvolvido, na seção IV são apresentados os experimentos realizados e a seção V contém a conclusão e futuros trabalhos.

2 Trabalhos Relacionados

Em [3] é abordado o problema de construção de um time de futebol de robôs, onde é tratado o planejamento de trajetória de múltiplos robôs em tempo real, predição da localização dos agentes em campo, isto é, robôs do time, robôs adversários e a bola, além de uma estratégia cooperativa entre os robôs. O planejamento de trajetória utiliza uma variação do algoritmo *RRT* [5] (*Rapidly Exploring Random Trees*) denominada *BK-BGT* (*Behavioral Kinodynamic Balanced Growth Rree*) que considera os aspectos dinâmicos dos robôs utilizados, assim encontrando trajetórias que podem ser seguidas de maneira mais precisa, além disso as trajetórias consideram a arquitetura de inteligencia *STP* (*Skill, Tatic and Play*) para diminuir o espaço de busca. Apesar de eficiente em resolver o problema de planejamento de trajetória de múltiplos robôs em ambientes dinâmicos, o problema a ser tratado nesse artigo é mais geral do que o descrito na dissertação, o fato do trabalho considerar a dinâmica junto a arquitetura *STP* de um time de futebol, faz com que o trabalho seja válido somente para esse propósito, além de aumentar o custo computacional envolvido em encontrar trajetórias viáveis.

Em [6] é abordado o problema de planejamento de trajetória de um único robô em um ambiente altamente dinâmico, o ambiente possui 300 obstáculos, dentre eles fixos e móveis que executam movimentos lineares e circulares. A abordagem utilizada nesse artigo separa o planejamento em duas etapas, a primeira etapa em offline encontra uma trajetória a ser seguida e a segunda etapa utiliza a técnica campos potenciais para guiar o robô. São obtidos bons resultados, porém a abordagem não pode ser empregada na resolução do problema de planejamento de múltiplos robôs, pois é utilizado uma técnica de predição da posição dos objetos no espaço que impossibilita seu funcionamento em tempo real.

Em [7] é demonstrado uma abordagem de otimização para resolução do problema de planejamento de trajetória de múltiplos robôs em um ambiente com obstáculos fixos, onde há a preocupação com a suavidade do caminho gerado. No artigo, é provado que o problema pode ser descrito como um problema de otimização onde apenas é necessário encontrar um caminho viável que ligue duas poses e respeite algum critério, como, a diferença entre $atan2(p_i - 1, p_i)$ e $atan2(p_i, p_i + 1)$ não pode ser maior do que θ_{max}. Apesar da formulação, o artigo não tem a pretensão de considerar ambientes dinâmicos que possuem obstáculos móveis, o que inviabiliza sua utilização no problema a ser tratado nesse artigo.

Em [8] é abordado o problema de planejamento de trajetória de múltiplos robôs em um ambiente discreto, onde o ambiente pode ser descrito como um grafo $G = \{V(G), E(G)\}$, onde $V(G)$ é o conjunto de vértices e $E(G)$ o conjunto de arestas, onde os robôs ocupam os vértices e se deslocam pelas arestas. Para garantir que não ocorrerá colisão entre o grupo de robôs são definidos duas regras. Somente um robô se movimenta por passo de simulação, com isso, é garantido que não ocorrerá colisões nas arestas do grafo, e a segunda regra, caso seja detectado que ocorrerá uma colisão entre dois robôs em um vértice de G é sorteado para um deles uma posição temporária fora da rota de colisão. O artigo obtém bons resultados, porém não resolve todas as simulações sem colisão, pois existem situações em que não é possível definir uma posição temporária fora de rota de colisão. A abordagem utilizada em [8] não seria viável para resolver o problema tratado nesse artigo, pois a existência de obstáculos móveis em um ambiente dinâmico, aumentaria muito o custo computacional e impossibilitaria a resolução do problema em tempo real.

3 O framework

A abordagem desse trabalho divide o problema em duas etapas, uma de planejamento *offline* simplificado e outra de correção de trajetória em tempo de execução, ou seja, *online*. Basicamente a grande questão de resolver o problema totalmente em offline é a necessidade de que haja uma quantidade grande de cálculos a serem executados para que as trajetórias sejam encontradas, forçando o computador a trabalhar muito durante um tempo e fazendo-o ficar ocioso durante o resto do tempo. Diminuindo a carga de processamento da primeira etapa e dividindo o processamento para ser executado com o passar do tempo em que os robôs estão seguindo suas trajetórias, é possível tornar o sistema executável em tempo real e resiliente a mudanças que podem ocorrer no espaço de trabalho.

3.1 Etapa de Processamento Offline

A tarefa a ser realizada na etapa offline pode ser definida da seguinte forma: Para cada $r_n \in R$, encontrar uma trajetória P_n que ligue a pose atual p_n^a e a pose objetivo p_n^o, de maneira a evitar contato com os obstáculos fixos de B.

Nessa etapa, não há preocupação que as trajetórias evitem contato com obstáculos móveis, pois além de aumentar o custo computacional do calculo, não há garantia de que os mesmos não mudem de posição, invalidando a trajetória. Também não se leva em consideração o desvio entre os robôs, pelo mesmo motivo dos obstáculos. Na figura 1 ilustra-se um exemplo de dois caminhos encontrados na etapa offline de processamento para dois robôs omnidirecionais distintos. Na figura, os quadrados em preto representam obstáculos fixos, os círculos em cinza representam obstáculos móveis e os caminhos em vermelho e azul são referentes aos robôs com rodas da mesma cor.

Para realizar o calculo de trajetória é utilizado a biblioteca *Open Motion Planning Library (OMPL)* [9], que consiste em um apanhado de algoritmos do

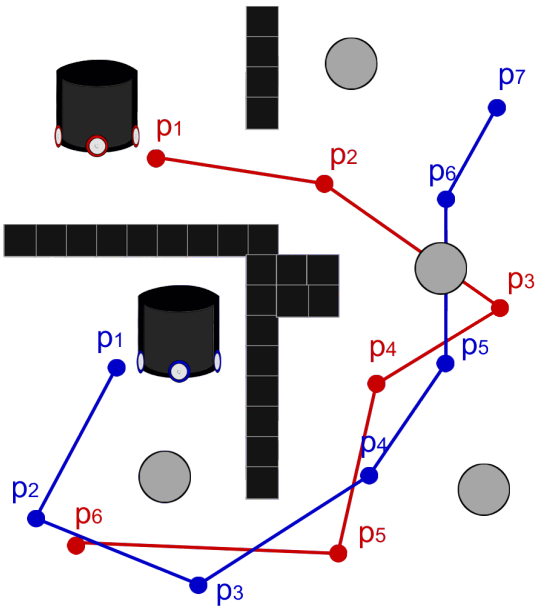

Figura 1. Exemplo de duas trajetórias calculadas na etapa offline. As sequências de pontos coloridos referem-se aos caminhos dos robôs com rodas da mesma cor.

estado da arte de planejamento de trajetória, como: *Fast Marching Trees (FMT)* [10], *Probabilistic Roadmap(PRM)* [11] e o próprio *RRT* utilizado. A escolha da biblioteca deve-se ao fato de além de implementar algoritmos uteis para o tratamento do problema, existir os seguintes benefícios:

- Feita para que possa ser adaptada facilmente a qualquer sistema;
- Pode-se utilizar tanto C++ quanto Python para o desenvolvimento;
- Benchmarking e otimização nativos;
- Documentação extensa gerada por Doxygen com demos e tutoriais;
- Trabalhar com diversos tipos de estados: 2D e 3D simples, carros e quadricópteros.

3.2 Etapa de Processamento Online

Durante a segunda etapa, é utilizado a técnica campos potenciais artificiais, como descrita em [12] para guiar os robôs de maneira reativa, evitando colisões e seguindo suas trajetórias definidas na etapa anterior. A técnica consiste em considerar o espaço de trabalho como um campo elétrico, onde robôs e obstáculos possuem a mesma carga, e poses objetivo possuem carga inversa, dessa maneira, como na realidade, os robôs se deslocam para as poses objetivo ao mesmo tempo que evitam contato entre sí e com obstáculos.

Formalmente, a técnica campos potenciais consiste em calcular uma força resultante $\boldsymbol{F} = \boldsymbol{F_A} + \boldsymbol{F_R}$ que define para onde um robô irá, a força resultante

é a soma do vetor de força atrativa $\boldsymbol{F_A}$ e o vetor da força repulsiva $\boldsymbol{F_R}$. A força atrativa $\boldsymbol{F_A}$ é proporcional à distância para uma pose objetivo, ou seja, quanto mais distante um robô de sua pose objetivo p_n, mais rápido é seu deslocamento para p_n, e quanto mais próximo, mais devagar. A força repulsiva $\boldsymbol{F_R}$ é inversamente proporcional a distância para um obstáculo, ou seja, quanto mais próximo de um obstáculo, maior é o vetor de repulsão, o que aumenta a reação do robô de evitar uma colisão. A força repulsiva $\boldsymbol{F_R}$ é acumulativa a quantidade de obstáculos próximos a um robô, ou seja, $\boldsymbol{F_R} = \boldsymbol{F_{r1}} + \boldsymbol{F_{r2}} + ... + \boldsymbol{F_{rn}}$.

Na figura 2 encontra-se a representação de um caminho e o funcionamento do campo potencial em guiar um robô pelo mesmo. O circulo em azul representa um robô e o circulo em cinza um obstáculo se aproximando. Como uma trajetória $P = \{p_1, p_2, ..., p_n\}$ é um conjunto de poses que partem da pose inicial de um robô p_1 para a pose final p_n, é calculado a força de atração da pose do robô para a pose da trajetória mais próxima do mesmo, sendo inicialmente p_2 e crescendo até p_n. O algoritmo define que a pose de atração de um robô r passa de p_i para p_i+1 quando o mesmo alcança uma determinada distância da pose p_i. Na figura, $\boldsymbol{F_A}$ em verde representa a força atrativa para p_2, $\boldsymbol{F_R}$ representa a força repulsiva gerada por um obstáculo cinza próximo e $\boldsymbol{F}$ representa a força resultante, que define para onde o robô irá.

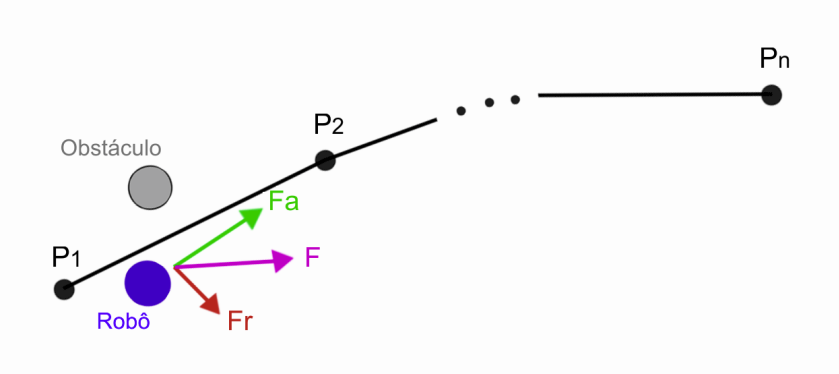

Figura 2. Exemplo do funcionamento dos campos potenciais em guiar um robô.

3.3 Conversão da força resultante e a plataforma utilizada

Após o calculo da força resultante que será aplicada nos robôs, é feito a conversão dessa informação para as velocidades que cada roda do robô deve empregar para que o mesmo siga o vetor resultante. Optou-se por utilizar uma plataforma robótica com movimentação omnidirecinal para realizar os experimentos, pois esse tipo de movimentação fornece maior manobrabilidade e estabilidade, assim facilitando o desenvolvimento. A plataforma utilizada é a mesma da equipe RoboIME do futebol de robôs Small Size League (SSL).

A conversão é feita com o auxílio de um modelo matemático referente a plataforma utilizada. Na figura 3 encontra-se a plataforma vista de cima, θ e φ represetam a angulação dos eixos do robô e $f = [F_1\ F_2\ F_3\ F_4]^T$ são as velocidades das rodas. Para definir as velocidades f é utilizado a equação 1, onde $v = [v_x\ v_y\ R_w]$ são as velocidades radial, tangencial e angular definidas pelo vetor resultante do campo potencial e D é a matriz de transformação representada na equação 2.

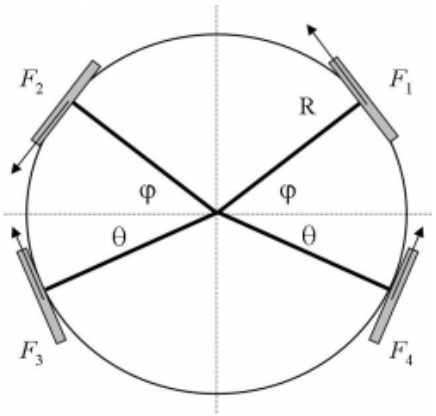

Figura 3. Disposição dos eixos do robô.

$$f = D \times v \tag{1}$$

$$D = \begin{bmatrix} -\sin\varphi & \cos\varphi & 1 \\ -\sin\varphi & -\cos\varphi & 1 \\ \sin\theta & -\cos\theta & 1 \\ \sin\theta & \cos\theta & 1 \end{bmatrix} \tag{2}$$

Após obter as velocidades que cada roda deve empregar para que o robô se direcione à pose objetivo, é necessário garantir que o robô consiga empregar essas velocidades, com esse objetivo é utilizado um controle PID [13]. O controle PID é uma técnica de controle conhecida por ser simples e eficiente, ela trata de utilizar ações proporcionais para diminuir o erro, ações derivativas para prever o erro e ações integrais para zerá-lo.

4 Experimentos

Foram realizados 5 mil experimentos em um ambiente simulado simples onde robôs e obstáculos não possuíam limitações quanto á sua movimentação e 100 experimentos no simulador grSim que incorpora as limitações de movimento da plataforma utilizada.

4.1 Simulação I

O framework conseguiu resolver 96.88% dos experimentos e foi executado em tempo real. Na tabela 1 em relação à quantidade de robôs, encontra-se o tempo médio de execução de toda a simulação, isto é, do planejamento offline e o tempo de execução para levar todos os robôs de uma pose inicial a uma pose final, a porcentagem de sucesso e a média de colisões ocorridas.

Tabela 1. Relação do funcionamento por quantidade de robôs.

	Tempo	**Sucesso**	**Colisões**
5 robôs	0.096s	96.4%	0.30
6 robôs	0.121s	97.6%	0.46
7 robôs	0.165s	97.6%	0.62
8 robôs	0.188s	95.6%	0.76
9 robôs	0.221s	97.2%	0.87
10 robôs	0.265s	94.1%	1.00

Foi utilizado o algoritmo *RRT* [5] como implementado na biblioteca *OMPL* para gerar as trajetórias durante a etapa offline. O espaço de trabalho possuía 1000x1000 pixels, a velocidade máxima dos obstáculos era de 5.65px/st (pixels por passo de simulação) e a velocidade máxima dos robôs era de 8px/st. A média global de robôs utilizados foi de 7.5016, variando de 4 a 9, foi considerado que cada robô possuía 3 pixels de raio. A média global de obstáculos utilizados foi de 48.9674, variando de 1 á 99, o raio dos obstáculos variavam de 2 á 25 pixels. A média global de colisões ocorridas foi de 0.673, variando de 0 á 6.

Na figura 4 é apresentado um exemplo de experimento criado, com 9 robôs e 99 obstáculos, as trajetórias geradas na etapa de processamento offline possuem tonalidade mais escura e as trajetórias executadas em online possuem a mesma cor de suas em offline, porém com tonalidade mais clara. Esse experimento foi resolvido pelo framework e ocorreram apenas 3 colisões.

4.2 Simulação II

O simulador grSim foi criado para a categoria de futebol de robôs Small Size League (SSL). Nele existem duas equipes controláveis de 6 robôs omnidirecionais. Na figura 5 encontra-se o simulador.

O framework conseguiu resolver 98% dos experimentos e foi executado em tempo real. Na tabela 2 encontra-se a porcentagem de sucesso e a média de colisões ocorridas.

Foram utilizados 6 robôs omnidirecionais e 6 obstáculos móveis. A média de colisões foi de 0.52, maior do que a média na simulação I para 6 robôs, devido as limitações de movimentação.

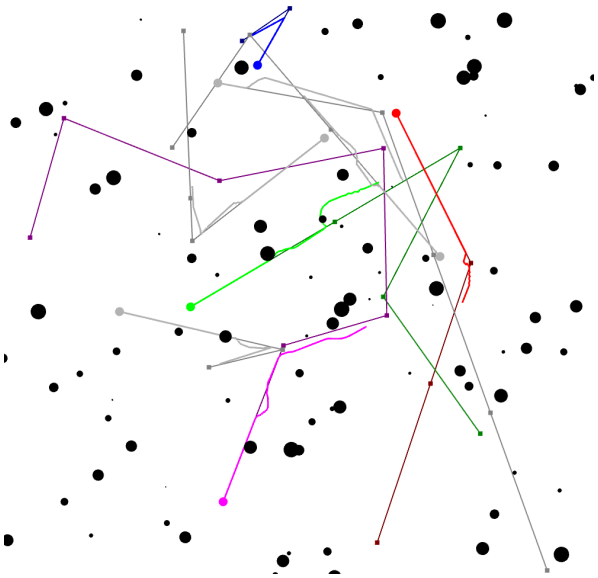

Figura 4. Exemplo de experimento.

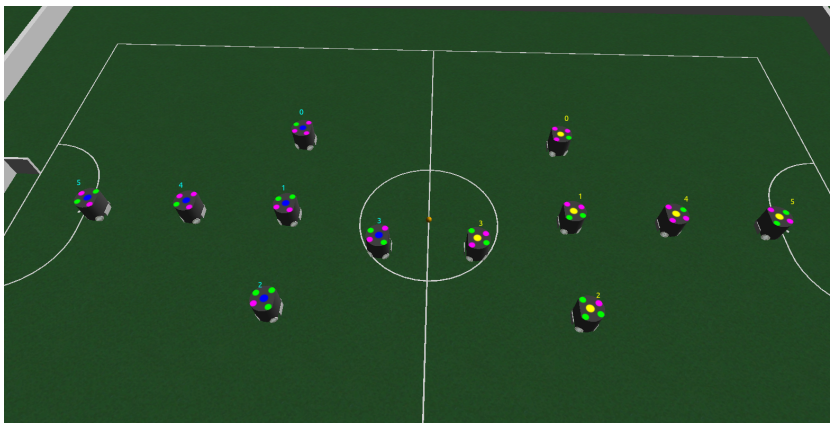

Figura 5. Simulador grSim.

Tabela 2. Porcentagem de Sucesso e Média de colisões no grSim.

	Sucesso	**Colisões**
6 robôs	98.0%	0.52

5 Conclusão e Trabalhos Futuros

A abordagem proposta para a construção do framework atende aos requisitos em ambientes simulados, possibilitando o controle de múltiplos robôs em tempo real em um ambiente dinâmico, porém o mesmo ainda apresenta colisões ocorrendo durante as resoluções. Visto que o framework resolve ambas etapas, offline e online, em tempo muito baixo, será investido na segurança contra colisões da etapa online. Também será testado alternativas ao algoritmo RRT na etapa planejamento offline, afim de encontrar melhores caminhos.

Referências

1. Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.
2. Pavel Surynek. An optimization variant of multi-robot path planning is intractable. In *AAAI*, page 6, 2010.
3. Stefano H Rodrigues. *Sistemas Autônomos e Inteligentes para Robôs Cooperativos no Ambiente Small Size League*. PhD thesis, Instituto Militar de Engenharia, Rio de Janeiro, 2013.
4. Adão M Neto. *Mapeamento de Ambiente Interno Semi-Estruturado com Múltiplos Veículos Utilizando Sensor Visual Embarcado*. PhD thesis, Instituto Militar de Engenharia, Rio de Janeiro, 2012.
5. Steven M LaValle. Rapidly-exploring random trees a new tool for path planning. 1998.
6. Hao-Tien Chiang, Nick Malone, Kendra Lesser, Meeko Oishi, and Lydia Tapia. Path-guided artificial potential fields with stochastic reachable sets for motion planning in highly dynamic environments. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2347–2354. IEEE, 2015.
7. Nader Motee, Ali Jadbabaie, and George Pappas. Path planning for multiple robots: An alternative duality approach. In *American Control Conference (ACC), 2010*, pages 1611–1616. IEEE, 2010.
8. Jun-Han Oh, Joo-Ho Park, and Jong-Tae Lim. Centralized decoupled path planning algorithm for multiple robots using the temporary goal configurations. In *Computational Intelligence, Modelling and Simulation (CIMSiM), 2011 Third International Conference on*, pages 206–209. IEEE, 2011.
9. Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012.
10. Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International journal of robotics research*, page 0278364915577958, 2015.
11. Lydia E Kavraki and Jean-Claude Latombe. Probabilistic roadmaps for robot path planning. 1998.
12. Michael A Goodrich. Potential fields tutorial. *Citeseer*, 1, 2002. 22 mar. de 2016.
13. Katsuhiko Ogata. *Modern control engineering*. Prentice Hall PTR, 2003.