
Universidade Federal de Minas Gerais

Escola de Engenharia

EEE935 - Planejamento de Movimento de Robôs

---

# Trabalho Final

## RRT no espaço de configurações

---

Felipe Bartelt de Assis Pessoa 2016026841

18 de fevereiro de 2022

# Sumário

# 1 Introdução

O Rapidly Exploring Random Tree (RRT) é um algorimo baseado em amostragem bastante utilizado para planejamento de movimento. Dessa forma, propõe-se implementar o RRT bidirecional no espaço de configurações para um manipulador de 6 graus de liberdade, KUKA KR5, cujas juntas são todas rotativas. O objetivo é planejar um caminho de uma configuração inicial até uma configuração objetivo, desviando de todos os obstáculos, conhecidos a priori, enquanto se evita auto-colisão.

O trabalho é dividio em três seções. Na Seção 2 - Metodologia, apresenta-se modificações sobre o RRT bidirecional apresentado por Choset et al. [1], considerações feitas para se alterar o espaço topológico, a métrica adotada para o espaço topológico considerado, os oráculos de colisão e auto-colisão, além do método de pós-processamento. Na Seção 3 - Resultados, apresenta-se duas situações simuladas, uma simples e uma complexa, com discussões sobre os resultados e apresentações das árvores geradas. A Seção 4 - Conclusão conclui o exitoso trabalho e possibilidades de aplicação.

# 2 Metodologia

Para a implementação, tomou-se a abstração de que a configuração final é conhecida a priori, por meio de cinemática inversa ou não. Dessa forma, a configuração final é um parâmetro do algoritmo. Utilizou-se o JupyterBot[2] como simulador, que já contém funções auxiliares para o algoritmo proposto.

## 2.1 RRT bidirecional

Inicialmente, criam-se duas árvores (RRT), uma, $T_{\text{init}}$, com a raiz da configuração inicial $q_{\text{init}}$ e outra, $T_{\text{goal}}$, com raiz sendo a configuração do objetivo $q_{\text{goal}}$. A árvore $T_{\text{init}}$ tem como objetivo a configuração $q_{\text{goal}}$, enquanto o objetivo de $T_{\text{goal}}$ é $q_{\text{init}}$.

Então, uma das árvores é selecionada, $T_1$, é selecionada e gera-se uma configuração aleatória com distribuição uniforme $q_{\text{rand}}$ no espaço livre. Toma-se também um viés, há uma probabilidade de 20% da configuração $q_{\text{rand}}$ ser exatamente a configuração objetivo de $T_1$. Tenta-se gerar uma configuração $q_{\text{rand}}$ dentro de $\xi = 200$ iterações, caso não seja possível, nenhuma configuração é retornada.

Obtendo-se o ponto aleatório, procura-se, na árvore $T_1$, o nodo, $q_{\text{nearest}}$, com configuração mais próxima a $q_{\text{rand}}$. A partir de $q_{\text{nearest}}$, toma-se um passo de tamanho $L = 0.1$ na direção de $q_{\text{rand}}$, obtendo-se um novo ponto $q_{\text{new}}$. Se o caminho de $q_{\text{nearest}}$ até $q_{\text{new}}$ não colidir com nenhum obstáculo e não gerar auto-colisão, $q_{\text{new}}$ é adicionado aos filhos de $q_{\text{nearest}}$.

Sempre que exisir um $q_{\text{new}}$, procura-se na outra árvore, $T_2$, o nodo cuja configuração é a mais próxima de $q_{\text{new}}$ e, caso o caminho de $q_{\text{new}}$ até essa configuração não resulte em qualquer colisão, resultando numa distância menor ou igual a 0.25, as árvores são mescladas e o algoritmo é finalizado. Caso contrário, inverte-se a ordem das árvores ($T_1 = T_2$, $T_2 = T_1$) e repete-se o procedimento anterior, expandindo-se alternadamente ambas as árvores, até que as árvores sejam mescladas ou o número máximo de iterações seja atingido.

## 2.2 Espaço Topológico e Métrica de Distância

Uma vez que o manipulador utilizado é composto de seis juntas rotativas, ter-se-ia um espaço topológico $T^6$, que não é euclidiano. Dessa forma, tomou-se limites para cada uma das juntas [3], apresentados na Tabela 1, induzindo-se um espaço de configurações euclidiano, além de tratar um problema mais realista.

| Junta | Ângulo Mínimo(°) | Ângulo Máximo(°) |
|---|---|---|
| 1 | -155 | 155 |
| 2 | -65 | 180 |
| 3 | -68 | 105 |
| 4 | -350 | 350 |
| 5 | -130 | 130 |
| 6 | -350 | 350 |

Tabela 1: Limite considerado para as juntas, em ordem $1 \mapsto$ base,... $6 \mapsto$ efetuador.

Garantindo-se um espaço euclidiano, permite-se utilizar a métrica euclidiana para o cálculo de distâncias aplicado no RRT. Preferiu-se tomar uma modificação da métrica euclidiana, conforme apresentado por Lavalle [4], de forma a ponderar cada junta com pesos diferentes, Uma vez que uma rotação na primeira junta, causa uma rotação em todas as demais, tomou-se essa como a de maior importância, com peso 1. Para a última junta, correspondente ao efetuador, considerada com menor importância para a distância entre nodos, tomou-se peso de 0.9. Os pesos intermediários são dados por uma progressão logarítmica. A motivação para tal abordagem é enviesar o algoritmo a evitar a rotação da base em demasia.

A métrica utilizada para o espaço topológico induzido $\tau$ é dada por (1) e mede a distância "angular" entre duas configurações quaisquer.

$$\rho(\tau) = \sqrt{c_1 \left(\rho_{\theta_1}(\theta_1, \theta_1')\right)^2 + c_2 \left(\rho_{\theta_2}(\theta_2, \theta_2')\right)^2 + \cdots + c_6 \left(\rho_{\theta_6}(\theta_6, \theta_6')\right)^2}, \text{ onde} \tag{1}$$

$$\rho(x, x') = \left(\sum_{i=1}^{n} |x_i - x_i'|^2\right)^{\frac{1}{2}} \tag{2}$$

## 2.3 Oráculos de colisões

A implementação tomada depende de dois oráculos, um para detecção de colisões com objetos externos e outro para detecção de auto-colisões. Ambos são baseados no algoritmo de projeções cíclicas de Von Neumann [5], já implementado no simulador utilizado.

O oráculo de colisões utiliza o algoritmo de Von Neumann para estimar a distância de obstáculos, cuja pose é conhecida a priori, a cada junta do manipulador. Assim, tomando-se um limiar de 0.01, quaisquer configurações a uma distância menor que esse limiar de um obstáculo, define uma configuração que gera colisão.

Para a implementação do oráculo de auto-colisão, tornou-se necessário modificar os modelos de colisão já implementados para o manipulador. Esses modelos correspondem a cilindros e prismas que permitem aproximar a geometria do manipulador, facilitando análise de colisões. Uma vez que esses modelos, por padrão do simulador, eram muito conservadores, fez-se algumas alterações conforme a Figura 1.

Analisando-se o modelo da Figura 1, nota-se que somente é necessário analisar auto-colisões entre superfícies de cores diferentes, uma vez que as de mesma cor auto-colidem por definição. Assim, a detecção de auto-colisão é dada pelo oráculo de colisões anterior, tomando-se a a distância entre as combinações dadas por cada par de cores.

Por meio da junção de ambos os oráculos, pode-se também analisar se caminhos que ligam duas configurações quaisquer, geram algum tipo de colisão. Para isso, faz-se uma pesquisa binária. Dado configurações $q_1$ e $q_2$ no espaço livre, pode-se analisar recursivamente se o caminho gera colisões, dividindo-se o intervalo inicial em dois subintervalos $\left(q_1, \frac{q_2-q_1}{2}\right)$ e $\left(\frac{q_2-q_1}{2}, q_1\right)$, caso a configuração $\frac{q_2-q_1}{2}$ esteja no espaço livre, analisa-se então os demais intervalos até que

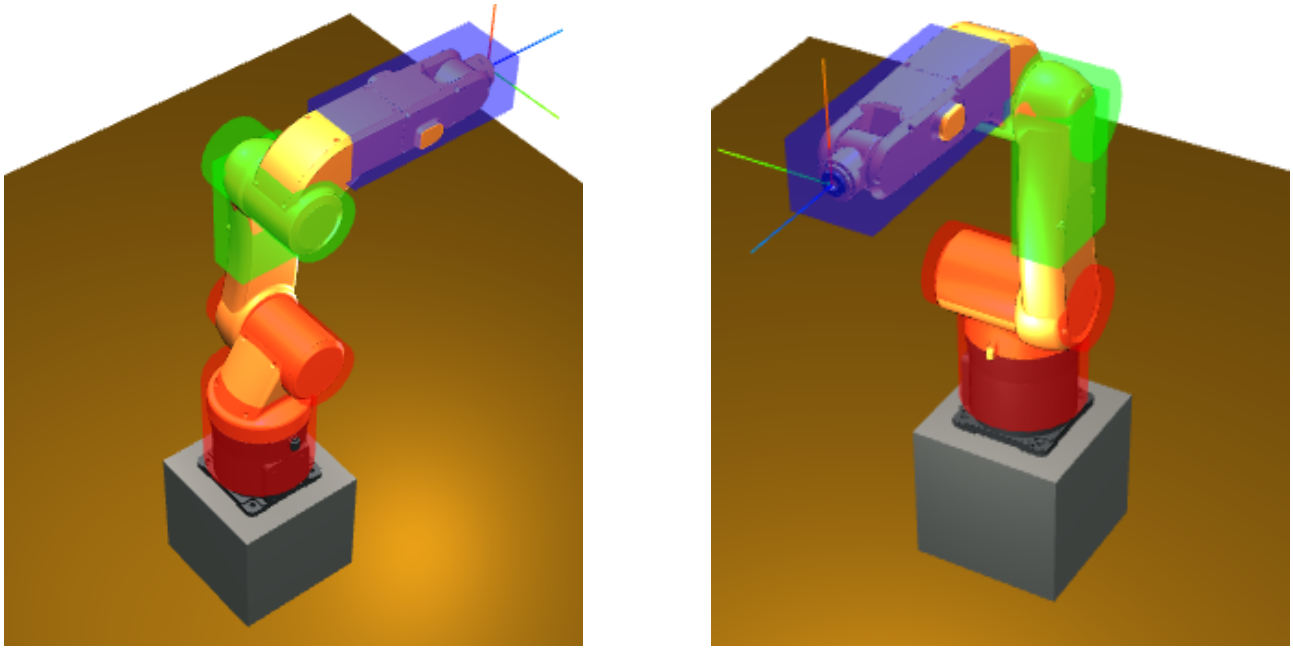


Figura 1: Modelos de colisão adotado

seja detectado algum tipo de colisão ou a variação da distância atinja certo limiar. Isso é, se o valor absoluto da diferença entre o tamanho de um novo intervalo e o tamanho do intervalo anterior, for menor que 0.1, o algoritmo retorna que o caminho é livre de colisões.

### 2.4 Pós-processamento

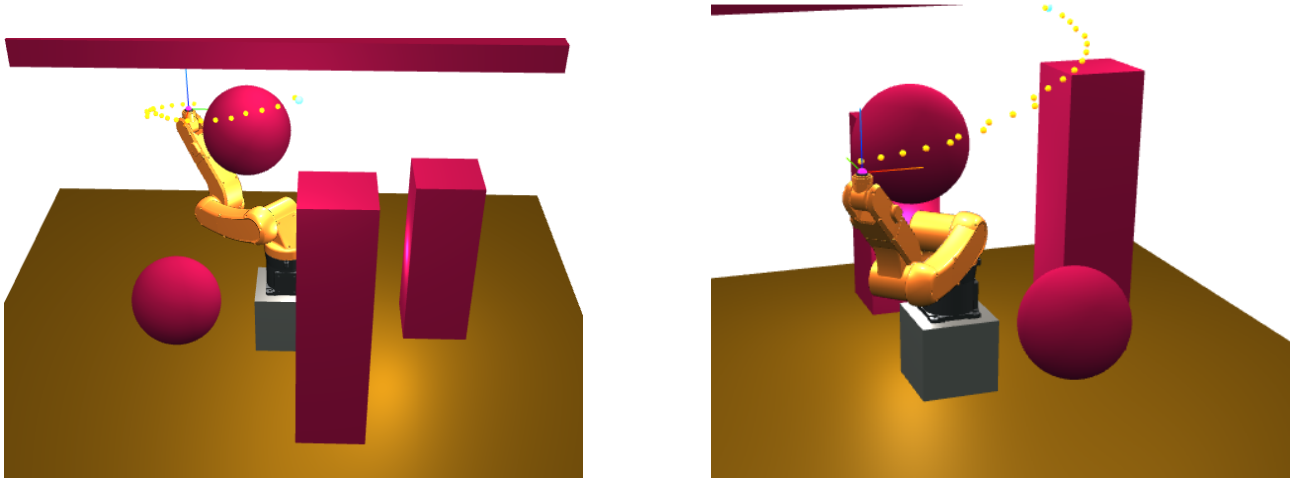
Após o RRT ter encontrado uma solução, realiza-se um pós-processamento para suavizar o caminho planejado. Para isso, toma-se o planejamento resultante e, para cada nodo $q_i$, checa-se se há um nodo $q_j$, $j > i + 1$, tal que a distância entre os dois seja menor ou igual a 0.25. Caso haja, todos os nodos $q_k$, $i < k < j$ são removidos do planejamento. O processo se encerra assim que se alcança o último nodo do planejamento inicial e então retorna-se o planejamento "podado".

## 3 Resultados

Primeiramente, definiu-se um ambiente de simulação mais simples, de forma a se corrigir eventuais erros de implementação. O caminho resultante do planejamento pode ser visto na Figura 2, onde as configurações são representadas por meio de um mapeamento para a pose do efetuador. Vê-se que para essa situação, o caminho planejado é bastante suave e, de certa forma, natural. Nota-se que somente foram necessárias 200 iterações para a convergência do RRT.

Traçou-se uma representação para ambas as árvores RRT dessa situação, que é demonstrada na Figura 3. Nota-se que, mesmo sem qualquer pós-processamento, as configurações das árvores indicam certo viés para seus respectivos objetivos, formando um caminho, quase suave, entre configuração inicial e final. Ao se observar as configurações $q_{\text{rand}}$ de ambas as árvores, não se nota qualquer viés, apenas uma distribuição uniforme de posições do efetuador no espaço livre.

Após, testou-se uma situação mais difícil, cujo caminho planejado resultado é mostrado na Figura 4. Nota-se que o manipulador não calcula um caminho por meio do espaço sem



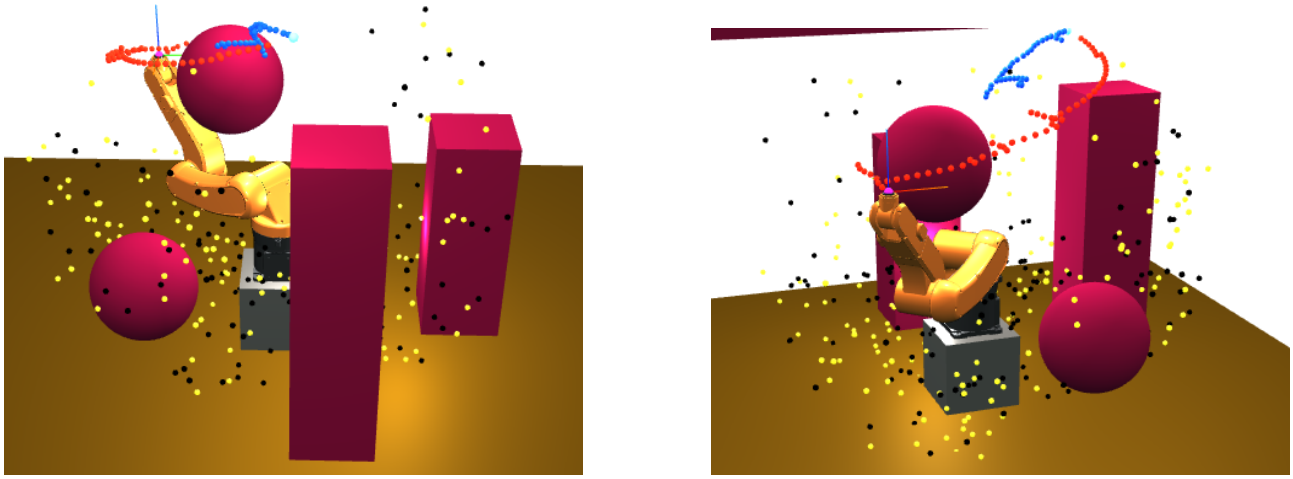
(a) Vista da configuração inicial

(b) Vista da configuração final

Figura 2: Caminho resultante do planejamento RRT para a primeira simulação. Configuração inicial demarcada com uma esfera em ciano, configuração final marcada com esfera magenta e caminho percorrido marcado em dourado. As configurações são representadas pelo mapeamento para a posição do efetuador.

obstáculos, o que é devido aos limites das juntas, que "forçam" o manipulador a percorrer um trajeto mais difícil, o que pode ser facilmente visualizado por meio da Figura 5c. Para essa situação, foram necessárias, aproximadamente, 5000 iterações até a convergência.

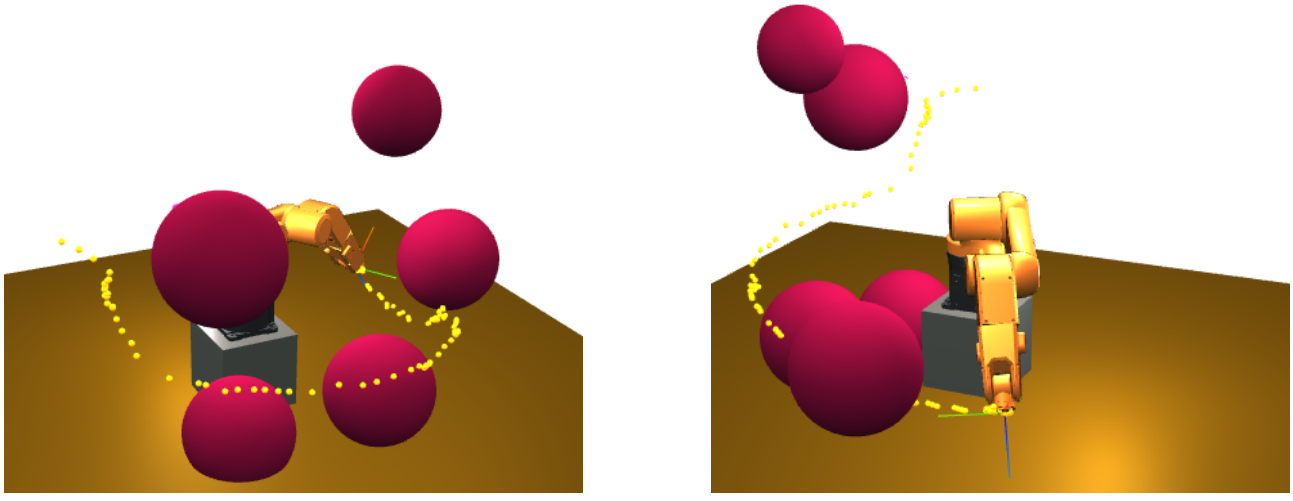
Também foi traçada uma representação para ambas as árvores RRT para essa situação, que é demonstrada na Figura 5. Percebe-se que essa situação é muito mais complexa, uma vez que as imagens são muito mais poluídas de informação. É interessante notar, novamente, que as árvores RRT, principalmente $T_{\text{init}}$, têm um viés para encontrar seu objetivo. Os nodos $q_{\text{next}}$ de $T_{\text{init}}$ tem o mesmo formato do caminho planejado final, o que foge do esperado. Novamente, esse viés não é notado nas configurações aleatórias $q_{\text{rand}}$.



(a) Vista da configuração inicial

(b) Vista da configuração final

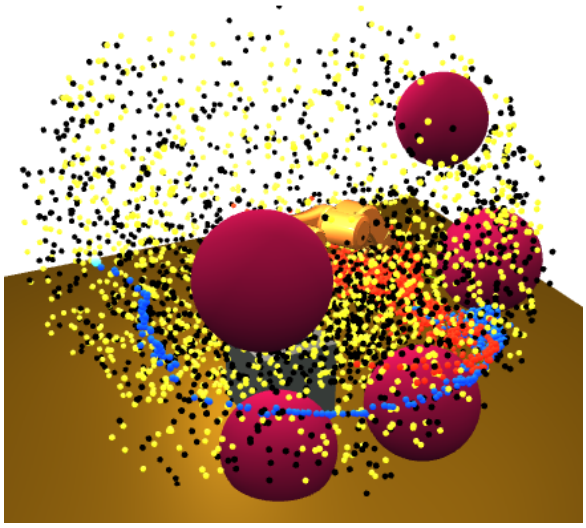
Figura 3: Representação de ambas as árvores RRT para a primeira simulação. Configuração inicial demarcada com uma esfera em ciano e configuração final marcada com esfera magenta. Esferas pretas indicam configurações $q_{\text{rand}}$ de $T_{\text{goal}}$, esferas amarelas as configurações $q_{\text{rand}}$ de $T_{\text{init}}$, esferas vermelhas as configurações $q_{\text{next}}$ de $T_{\text{goal}}$ e esferas azuis as configurações $q_{\text{next}}$ de $T_{\text{init}}$. As configurações são representadas pelo mapeamento para a posição do efetuador.



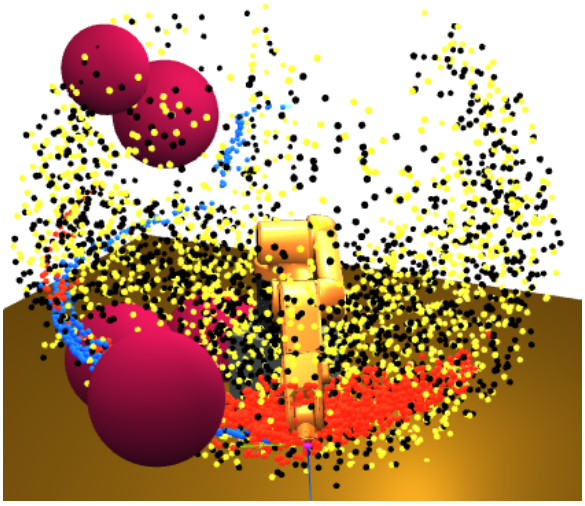
(a) Vista da configuração inicial

(b) Vista da configuração final

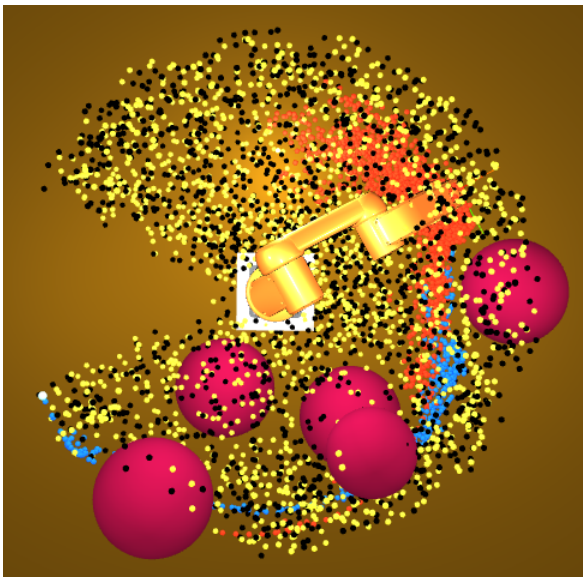
Figura 4: Caminho resultante do planejamento RRT para a segunda simulação. Caminho percorrido marcado em dourado. As configurações são representadas pelo mapeamento para a posição do efetuador. As configurações inicial e final não foram representadas, entretanto o manipulador se encontra exatamente na configuração final e é possível visualizar a configuração inicial por meio do início do caminho planejado.



(a) Vista da configuração inicial



(b) Vista da configuração final



(c) Vista superior

Figura 5: Representação de ambas as árvores RRT para a segunda simulação. Configuração inicial demarcada com uma esfera em ciano e configuração final marcada com esfera magenta. Esferas pretas indicam configurações $q_{\text{rand}}$ de $T_{\text{goal}}$, esferas amarelas as configurações $q_{\text{rand}}$ de $T_{\text{init}}$, esferas vermelhas as configurações $q_{\text{next}}$ de $T_{\text{goal}}$ e esferas azuis as configurações $q_{\text{next}}$ de $T_{\text{init}}$. As configurações são representadas pelo mapeamento para a posição do efetuador.

# 4 Conclusão

Por meio das simulações feitas, nota-se que as restrições impostas ao manipulador não só possibilitaram situações mais difíceis, como mais realistas, além de permitir o tratamento do espaço como euclidiano. O evitamento de auto-colisão se mostrou um problema mais interessante que o esperado, uma vez que essa restrição impede diversas configurações no espaço, restringindo ainda mais o problema. O pós-processamento se mostrou bastante necessário, principalmente ao visualizar as RRTs finais da segunda simulação.

Mesmo tomando bastantes iterações e, devido à capacidade do computador utilizado, bastante tempo, o RRT se mostrou excepcional. Dado um número suficiente de iterações, sendo o caminho possível, então o RRT é um excelente algoritmo para se realizar planejamentos a priori tanto no espaço de trabalho quanto no espaço de configurações. Ao se tomar diversas restrições holonômicas para um problema, especula-se que o RRT, ou variações deste, seja um excelente algoritmo para planejamento.

Outro fato que também fica evidente, é o uso do algoritmo RRT para verificar se um planejamento é possível. Após se confirmar o funcionamento do algoritmo proposto por meio da primeira simulação, passou-se a utilizar o próprio algoritmo para criar um ambiente mais complexo, adicionando-se obstáculos e alterando as configurações, enquanto se verificava a possibilidade de um planejamento por meio da convergência do RRT ou pela dispersão de seus nodos.

Investigou-se o fato de haver algum viés para a convergência dos nodos das RRTs, porém nada pôde ser concluído. A probabilidade de escolha da configuração objetivo como configuração aleatória, não demonstrou qualquer relação com o viés. Todas as amostragens são uniformes. Retirar os limites das juntas não implicou em uma dispersão dos nodos, somente possibilitou caminhos novos. Tomar pesos iguais para a métrica utilizada surtiu em uma dispersão maior, porém o viés ainda era claro. Dessa forma, nada se pôde concluir quanto ao viés e esse é então um objeto de estudo para trabalhos futuros.

# Referências


[1] H. Choset, K. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, May 2005. [Online]. Available: https://www.scholars.northwestern.edu/en/publications/principles-of-robot-motion-theory-algorithms-and-implementations

[2] V. M. Gonçalves, "Jupyterbot," https://github.com/viniciusmgn/jupyterbot_vinicius, 2022.

[3] A. A. Hayat, R. O. M. Sadanand, and S. K. Saha, "Robot manipulation through inverse kinematics," *Proceedings of the 2015 Conference on Advances In Robotics - AIR '15*, 2015.

[4] S. M. Lavalle, *Planning algorithms*. Cambridge University Press, , Cop, 2014. [Online]. Available: http://lavalle.pl/planning/

[5] J. V. Neumann, *Functional operators. Volume 2, The geometry of orthogonal spaces*. Princeton University Press, 1950.