V SIINTEC INTERNATIONAL SYMPOSIUM ON INNOVATION AND TECHNOLOGY
SIMPÓSIO INTERNACIONAL DE INOVAÇÃO E TECNOLOGIA
CIRCULAR ECONOMY | ECONOMIA CIRCULAR

CONTROLE SIMPLES E ROBUSTO PARA MANIPULADORES ROBÓTICOS ATRAVÉS DO MOVEIT

Kaike Wesley Reis[1]; Rebeca Tourinho Lima[2]; Marco Antonio dos Reis[3]

[1] Universidade Federal da Bahia; Salvador/Bahia; kaikewesley@hotmail.com
[2] SENAI Cimatec; Salvador/Bahia
[3] SENAI Cimatec; Salvador/Bahia

Resumo: Os avanços tecnológicos se tornaram mais rápidos e constantes na última década. Neste cenário, a robótica vem desempenhando cada vez mais o papel de produzir autônomos capazes de substituir o homem em tarefas repetitivas ou perigosas. Dado o atual contexto a busca por soluções práticas e robustas se tornaram necessárias para acompanhar este ritmo. Esse artigo objetiva apresentar a ferramenta *MoveIt!* para a modelagem de manipuladores, utilizando o modelo *Manipulator-H*, através da criação de um pacote de controle inédito, de código aberto e documentado. Apresenta também um *benchmarking* para escolher a melhor opção de planejador. Ao final é disposta a melhor solução encontrada de acordo as métricas estabelecidas acompanhado do acesso para o pacote desenvolvido.

Palavras-Chave: Robótica; ROS; MoveIt; Manipulator-H.

AN EASY AND ROBUST CONTROL FOR ROBOTIC MANIPULATORS THROUGH MOVEIT

Abstract: Technological advances have become faster and more steady over the last decade. In this scenario, robotics is increasingly playing the role of producing autonomous devices capable of replacing man in repetitive or dangerous tasks. Given the current context, the search for practical and robust solutions has become necessary to keep up with this pace. This article aims to introduce the *MoveIt!* for manipulators control using the *Manipulator-H* model by creating an unpublished, open source and well documented control package. It also features a *benchmarking* to choose the best planner to this model. At the end, the best solution is found according to the established metrics followed by the package online access.

Keywords: Robotics; ROS; MoveIt; Manipulator-H.

1. INTRODUÇÃO

Na última década, avanços em diversos campos tecnológicos se tornaram cada vez mais crescentes: serviços de táxi aéreo sendo testados na Alemanha [1], carros autônomos sendo desenvolvidos por diversas companhias (com destaque a Tesla) e o grande avanço dos métodos de inteligência artificial, como o mais conhecido *Deep Fake* [2], exemplificam essa realidade que pode impactar profundamente a economia global. A robótica, por sua vez apresenta uma vanguarda mais direcionada para a criação de robôs autônomos para diferentes tipos de funções sejam elas industriais ou domésticas.

Dado esse intenso avanço, o objetivo desse artigo é apresentar uma modelagem computacional para manipuladores baseada no *MoveIt!* [3], *software* que apresenta uma solução simples e robusta para esta problemática. Além disso, será demonstrado a utilização do *benchmarking*, uma de suas ferramentas para escolher um planejador de trajetória para o *Manipulator-H*.

A criação deste artigo possui ainda o intuito de apresentar a facilidade proporcionada por tal aplicação, permitindo que novas pesquisas sejam voltadas para a criação de soluções para problemas ainda existentes. Além disso, existe o intuito de diminuir a ausência de materiais explicativos direcionados para essa área no idioma nacional ou até mesmo em inglês, que apresenta um conteúdo disperso em diferentes vídeos e fóruns. Outro ponto a se destacar, é a escolha da linguagem *Python* para o desenvolvimento do pacote, que apesar de ser uma linguagem mais simples e ainda robusta não apresenta grande aderência na comunidade do *MoveIt!* que ainda utiliza bastante de soluções com a linguagem *C++*.

1.1. Controle para manipuladores

O controle de manipuladores objetiva de modo geral mover o *end-effector*, ferramenta ou parte mais externa de um braço robótico, para uma posição desejada no espaço de trabalho através da manipulação das posições das juntas, articulações responsáveis pela movimentação do robô [4]. Para conquistar essa meta, a interação entre três processos principais: cinemática direta, cinemática inversa e planejamento de trajetória é necessária.

A cinemática direta constitui o processo mais simples e trata da solução da posição e orientação para o *end-effector* do manipulador no espaço dado a posição das juntas. Dado sua simplicidade, sua solução é padronizada através da convenção de Denavit-Hartenberg [4], função que apresenta como parâmetros de entrada as características construtivas do manipulador e retorna a cinemática do mesmo. Já a cinemática inversa é um problema oposto ao da cinemática direta e se trata do cálculo das posições das juntas para uma posição e orientação qualquer do *end-effector* no espaço [4]. Todavia, este último apresenta uma dificuldade maior para encontrar uma solução dado a inexistência de uma resposta singular (cada robô apresenta um cálculo diferente para este problema) e a infinidade ou ausência de soluções para determinadas configurações de posições e orientações.

O planejamento de trajetória atua no cálculo para encontrar o melhor caminho no espaço para um manipulador sair do ponto inicial para o destino, sendo responsável por especificar os pontos que o manipulador deve passar, o caminho propriamente dito, para alcançar tal objetivo. Este processo considera o ambiente em que o manipulador está inserido, sua geometria, capacidades motoras, métricas para encontrar o caminho, viabilidade de soluções para a cinemática direta e inversa e restrições definidas pelo pesquisador. Trata-se de um dos problemas mais complexos na área da ciência computacional [4], onde os algoritmos capazes de solucionar esse problema são chamados de planejadores.

1.2. Solução computacional para controle

O *MoveIt!* é um *framework* de código aberto para controle de manipuladores, reunindo um conjunto de técnicas para solucionar as problemáticas apresentadas anteriormente. Como exemplo, para o cálculo da cinemática inversa e direta o *MoveIt!* apresenta as bibliotecas *KDL* (*Kinematics and Dynamics Library*), *TRAC-IK* e *IKFast* (*Inverse Kinematics Fast*). Para o planejamento de trajetórias existem as bibliotecas *OMPL*(*Open Motion Planning Library*), *CHOMP* (*Covariant Hamiltonian Optimization for Motion Planning*), *SBPL* (*Search-Based Planning Library*) e *STOMP* (*Stochastic Trajectory Optimization for Motion Planning*) [5].

Inicialmente lançado em 2011, o *MoveIt!* trata-se de um projeto desenvolvido internacionalmente pela comunidade de robótica com o apoio de algumas corporações como a Franka Emika e PickNik Robotics [6]. Implementações utilizando esse framework já estão inclusive disponíveis para diversos manipuladores de renomadas empresas do setor industrial como a ABB, Fanuc e Kuka Robotics [6].

Sua implementação baseia-se no *ROS* (*Robot Operating System*) [7], um sistema flexível de código aberto desenvolvido e mantido pela comunidade internacional e por universidades para o desenvolvimento de *softwares* de controle e comunicação para robôs. Dada a presença de uma comunidade ativa e colaborativa além da facilidade promovida por este sistema, a criação de aplicações na área de manipuladores se tornam ainda mais simples.

1.3. Modelo do manipulador

O modelo utilizado para demonstrar o controle de um manipulador é o *Manipulator-H* criado pela ROBOTIS, empresa de tecnologia especializada na produção de atuadores, servomotores e manipuladores [8]. A solução desenvolvida, o pacote *ROS*, para este modelo em específico não está disponível na comunidade *open source*, sendo assim uma contribuição inédita. O *Manipulator-H* disposto na figura 1 apresenta seis graus de liberdade, sendo todas as juntas do tipo rotativa.

Figura 1.. Representação do manipulator-H

Fonte: ROBOTIS (2019)

2. METODOLOGIA

Foi desenvolvido inicialmente um pacote de controle para o *Manipulator-H* através do *MoveIt!*. O pacote apresenta um ambiente de teste e diversas formas de manipulação do robô sendo elas a movimentação através do acionamento posicional das juntas, ir para posições pré-determinadas e ir para pontos escolhidos no espaço tridimensional.

Além disso, foi elaborado um *benchmarking*, uma das funcionalidades implementadas no *MoveIt!*, para avaliar um conjunto de planejadores da biblioteca *OMPL* [9] utilizando o *KDL* como solução para o cálculo das cinemáticas, ambas bibliotecas padrão. Para o escopo deste artigo, define-se *benchmarking* como uma comparação de desempenho entre os algoritmos capazes de solucionar o planejamento de trajetória para movimentar o *Manipulator-H*.

Para melhor compreensão desse processo de comparação é necessário definir *start state* e *query*. O *start state* é a posição que o manipulador irá iniciar na simulação do *benchmarking* e *query* é um caminho determinado por dois pontos quaisquer do espaço de trabalho.

Foram utilizadas diversas ferramentas na concepção do pacote *ROS* que deu origem a este trabalho: utilizou-se a linguagem *Python* no desenvolvimento do pacote, o *Gazebo* (*software* de simulação) para simular o ambiente de teste e o *RViz* (ferramenta de visualização nativa do *ROS*) para definir graficamente as posições pré-determinadas para o manipulador e os pontos utilizados para compor um *query* e o *start state* do *benchmarking*.

O método de *benchmarking* proposto consiste em avaliar o desempenho dos planejadores através de um *query* definido por dois pontos aleatórios e válidos para a movimentação do manipulador. Ao total, foi testado um conjunto de nove planejadores cada um com vinte e cinco tentativas para realizar o planejamento com prazo limite de dez segundos para encontrar a solução. O cenário para aplicação da avaliação consiste em um ambiente de teste sem qualquer obstrução de caminho.

As métricas avaliadas para definir o melhor planejador foram duas: o tempo total para o planejador encontrar uma solução e o *solved*, razão entre quantas vezes o planejador conseguiu encontrar uma solução para o caminho e o total máximo de tentativas [10]. O tempo total é descrito pelo somatório do tempo de planejamento, interpolação, simplificação de caminho e processo [10]. Para alcançar o melhor resultado, o planejador precisa minimizar o tempo total e alcançar o valor de máximo de um para o *solved*.

Para apresentação dos resultados utilizou-se o Planner Arena [11], aplicação *web* desenvolvida também pela comunidade responsável pelo criação da ferramenta de *benchmarking* do *MoveIt!*.

3. RESULTADOS E DISCUSSÃO

O pacote *open source* que deu origem a este artigo se encontra em um repositório público do Senai CIMATEC [12]. Seu material consiste em um pacote completo, com códigos e instruções de instalação bem documentados além de um tutorial para a execução do processo de *benchmarking*. O pacote foi escrito utilizando a linguagem *Python*, de mais alto nível, que é mais acessível sem perder a robustez, mas não está muito presente na comunidade do *MoveIt!*.

O desenvolvimento do pacote *ROS* teve como base o uso da ferramenta auxiliar *Setup Assistant*, nativa do *MoveIt!*, que utilizou o *URDF* (*Unified Robot Description Format*) do *Manipulator-H*, formato de arquivo que apresenta a descrição física do robô para gerar o *SDRF* (*Semantic Robot Description Format*), arquivo que apresenta a descrição semântica do robô para o *MoveIt!*. Durante a aplicação dessa ferramenta, define-se a matriz de colisões do robô, posições pré-configuradas de interesse e se define os solucionadores para os problemas de cinemática e planejamento de trajetórias, sendo escolhidos para esse pacote, respectivamente, *KDL* e *RRT Connect,* presente na biblioteca *OMPL*. Essas escolhas são configuradas como padrões no *MoveIt!* sendo a última uma solução muito aceita e recomendada pela comunidade.

Os resultados para o *benchmarking* para a métrica de tempo total são apresentadas nas figuras 3 e 4.

Figura 2. Boxplot do tempo total para o primeiro grupo de planejadores

Fonte: Autoria própria

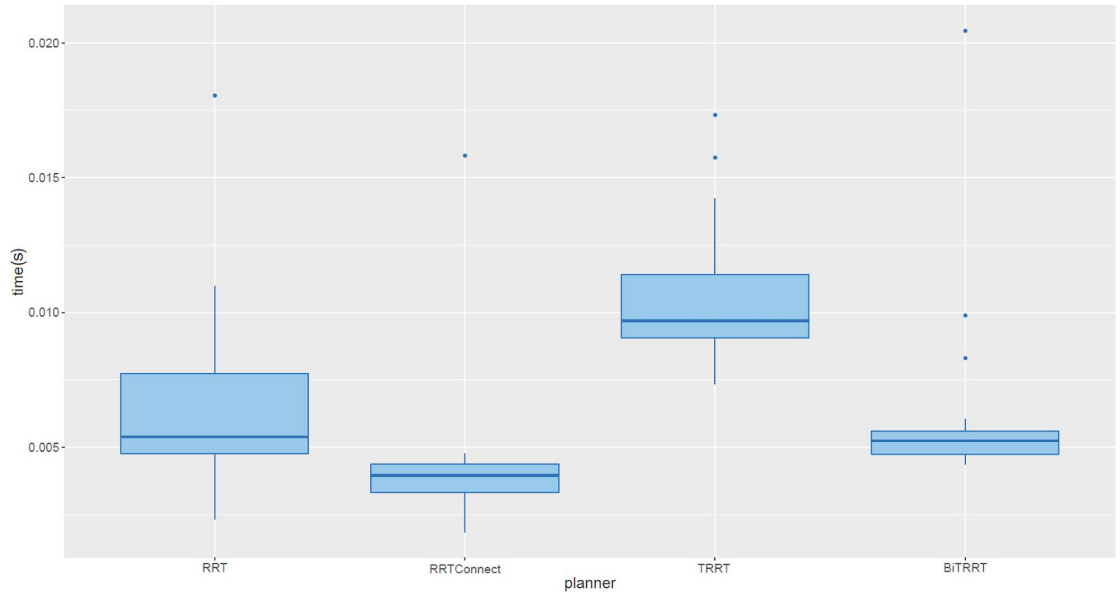
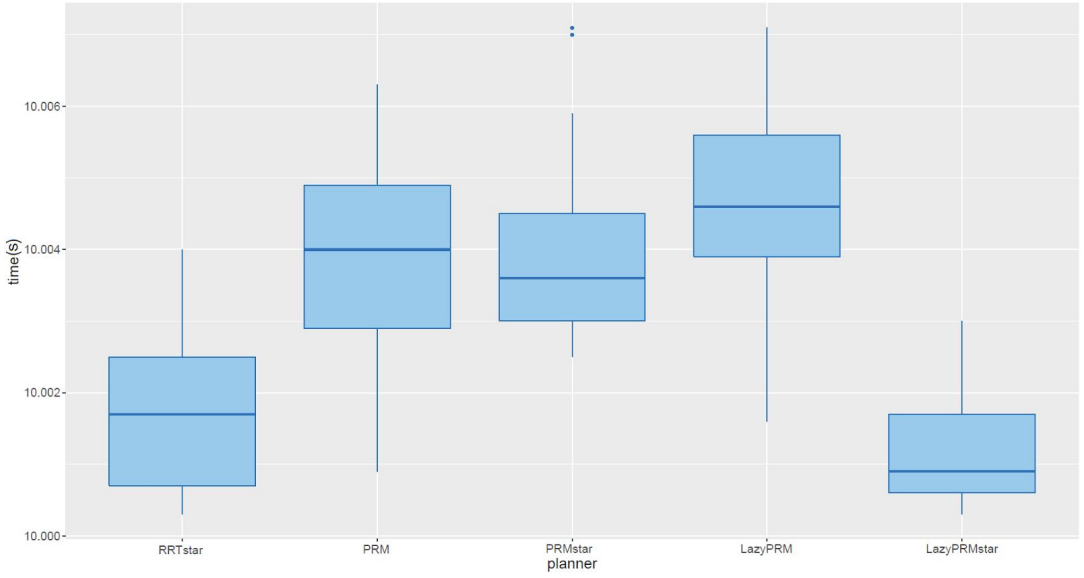


Figura 3. Boxplot do tempo total para o segundo grupo de planejadores

Fonte: Autoria própria



Os gráficos das figuras 2 e 3 são diagramas de caixa referente ao tempo total calculado de cada tentativa para cada planejador. Essa representação é conhecida por *boxplot* e se trata de uma ferramenta gráfica para visualizar a variação do conjunto amostral (representado pelo tamanho da caixa) em torno da média (representado pela linha central e horizontal da caixa).

Os resultados foram divididos em dois grupos, distintos pelo tempo total de solução. Na figura 2 o primeiro grupo é composto por quatro planejadores que apresentam um tempo de resposta inferior a 15 milissegundos, já o segundo grupo de planejadores, representado na figura 3, apresentam um tempo total de resposta em torno de 10 segundos, tempo máximo estabelecido para cada tentativa. Portanto, para aplicações onde o tempo é um fator importante, os planejadores do primeiro grupo demonstram ser uma melhor escolha em comparação aos do segundo para o planejamento de trajetórias para o *Manipulator-H*.

Os resultados para a métrica *solved* todavia, identifica todos os planejadores como capazes de encontrar uma solução para o caminho sem obstrução, visto que conseguiram solucionar o caminho em todas as tentativas, alcançando o valor máximo de um.

Portanto, considerando as métricas definidas, o planejador *RRT Connect* demonstra ser a melhor escolha para o controle do *Manipulator-H*. Apresentado no segundo *boxplot* da esquerda para a direita na figura 2, ele mostra uma menor variação no tempo total em comparação aos outros planejadores do primeiro grupo e a menor média também, sendo abaixo de 5 milissegundos. Isso mostra que seu algoritmo é preciso no encontro das soluções para trajetórias, mesmo sendo testado diversas vezes para o mesmo caminho.

4. CONCLUSÃO

Utilizando o *Manipulator-H* como objeto de estudo e o *MoveIt!* como ferramenta de controle, foi desenvolvida uma solução simples e eficiente para o controle. Além disso foi possível avaliar o planejador mais eficiente considerando as métricas escolhidas através do *benchmarking*: *RRT Connect*. Vale lembrar que apesar deste planejador apresentar melhor resultado, não existe de modo geral uma supremacia para um algoritmo em específico comparado aos outros, mas sim situações em que um pode prevalecer em detrimento de diversas condições a serem especificadas como o tempo máximo para cada tentativa, a presença ou não de obstáculos no ambiente e até mesmo a estrutura do manipulador.

Através desse artigo, é possível verificar que o *MoveIt!* atua como um facilitador para problemas comuns existentes no desenvolvimento de aplicações robóticas que utilizem manipuladores. Dada a sua facilidade de implementação, ferramentas disponíveis e uma comunidade ativa o pesquisador roboticista através do uso desse aparato pode concentrar seus esforços na solução de outros problemas mais complexos e ainda sem solução. Além disso, vale ressaltar a importância da comunidade internacional de robótica no desenvolvimento das principais ferramentas presentes nesse artigo, apresentando soluções de código aberto e, portanto, gratuitas como o *ROS* e o *MoveIt!*.

O pacote desenvolvido utilizando este ferramental, busca propiciar um material para iniciantes e entusiastas nessa área através de um conteúdo explicativo desde a instalação até o uso, algo ausente na comunidade atualmente. Utiliza-se ainda de uma linguagem mais simples, robusta e fácil de compreender como o

Python. No futuro, planeja-se incrementar as funcionalidades existentes neste pacote. É almejado acrescentar a possibilidade do uso de uma solução mais robusta para cinemática inversa e direta através do *IKFast* e a visualização do espaço de trabalho do robô.

5. REFERÊNCIAS

[1]VINHOLES, Thiago. Lilium apresenta protótipo de táxi aéreo urbano. **Airway**, Disponível em: <https://airway.uol.com.br/lilium-apresenta-prototipo-de-taxi-aereo-urbano/>. Acesso em: 11 ago. 2019.

[2]HUI, Jonathan. How deep learning fake videos (Deepfake) and how to detect it?. **Medium**, Disponível em: <https://medium.com/@jonathan_hui/how-deep-learning-fakes-videos-deepfakes-and-how-to-detect-it-c0b50fbf7cb9>. Acesso em: 11 ago. 2019.

[3]COLEMAN, David et al. Reducing the Barrier to Entry of Complex Robotic Software: a MoveIt! Case Study. **ArXiv**, 2014.

[4]SPONG Mark W. et al. **Robot Modeling and Control**. New York: John Wiley & Sons Inc., 2006.

[5]PICKNIK, Consulting. MoveIt! Concepts. **PickNik Robotics**, Disponível em: <https://moveit.ros.org/documentation/concepts/>. Acesso em 12 set. 2019.

[6]SUCAN, Ioan; CHITTA, Sachin. **MoveIt**, 23 mai. 2016. Disponível em: <http://moveit.ros.org>. Acesso em: 04 ago. 2019.

[7]MORGAN Quigley et al. ROS: an open-source Robot Operating System. **ICRA Workshop on Open Source Software**, 2009.

[8]ROBOTIS. About Us. **Robotis**, Disponível em: <http://www.robotis.us/about-us/>. Acesso em: 10 set. 2019.

[9]SUCAN, Ioan et al. The Open Motion Planning Library. **IEE Robotics & Automation Magazine**, v.19, n.4, p. 72–82, 2012.

[10]SHADOW ROBOT, Company. Planners Benchmarking Documentation. **Shadow Robot Company**, Disponível em: <https://buildmedia.readthedocs.org/media/pdf/planners-benchmarking/latest/planners-benchmarking.pdf>. Acesso em: 13 ago. 2019.

[11]MOLL, Mark et al. Benchmarking Motion Planning Algorithms: An Extensible Infrastructure for Analysis and Visualization. **IEE Robotics & Automation Magazine**, v.22, n.3, p. 96–102, 2015.

[12]REIS, Kaike Wesley. MoveIt! Package for Manipulator-H. **Brazilian Institute of Robotics (BIR)**, Disponível em: <https://github.com/Brazilian-Institute-of-Robotics/manipulator_h_moveit>. Acesso em: 10 ago. 2019.