

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENG. DE CONTROLE E AUTOMAÇÃO

HENRIQUE SCHARLAU COELHO - 243627

MAPEAMENTO DE AMBIENTES...

Porto Alegre
2023

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENG. DE CONTROLE E AUTOMAÇÃO

HENRIQUE SCHARLAU COELHO - 243627

MAPEAMENTO DE AMBIENTES...

Trabalho de Conclusão de Curso (TCC-CCA)
apresentado à COMGRAD-CCA da Universi-
dade Federal do Rio Grande do Sul como parte
dos requisitos para a obtenção do título de *Ba-
charel em Eng. de Controle e Automação*.

ORIENTADOR:

Prof. Dr. Walter Fetter Lages

Porto Alegre
2023

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENG. DE CONTROLE E AUTOMAÇÃO

HENRIQUE SCHARLAU COELHO - 243627

MAPEAMENTO DE AMBIENTES...

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção dos créditos da Disciplina de TCC do curso *Eng. de Controle e Automação* e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. Walter Fetter Lages, UFRGS

Doutor pelo Instituto Tecnológico de Aeronáutica – São José dos Campos, Brasil

Banca Examinadora:

Prof. Dr. Walter Fetter Lages, UFRGS

Doutor pelo Instituto Tecnológico de Aeronáutica – São José dos Campos, Brasil

Prof. Dr. Renato Ventura Bayan Henriques, UFRGS

Doutor pela Universidade Federal de Minas Gerais – Belo Horizonte, Brasil

Profa. Dra. Rafael Antônio Comparsi Laranja, UFRGS

Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Alceu Heinke Frigeri

Coordenador de Curso

Eng. de Controle e Automação

Porto Alegre, Agosto 2023

RESUMO

Palavras-chave: Automação e Controle, Robótica.

LISTA DE ILUSTRAÇÕES

1	Mercado global de robôs autônomos de 2016 a 2021, com projeção até 2028.	8
2	Exemplo de configuração de camadas de um mapa de custo.	11
3	Arquitetura do <i>Navigation2</i>	12
4	Planta do 1º andar do prédio Centenário da EE-UFRGS	14
5	Ambiente Gazebo com modelo do prédio Centenário da EE-UFRGS ..	15
6	RViz com robô twil no ambiente do prédio Centenário da EE-UFRGS.	15
7	RViz mostrando o erro de odometria tirar essa?	16

LISTA DE TABELAS

1	Símbolos dos nós de uma árvore de comportamento.....	10
---	--	----

LISTA DE LISTAGENS

LISTA DE ABREVIATURAS

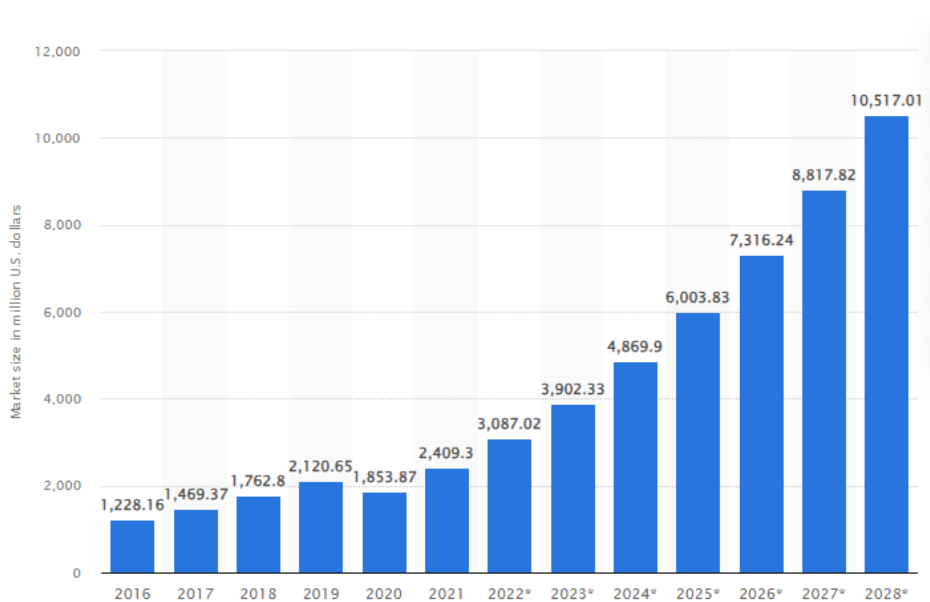
ROS	Robot Operating System
BT	Behavior Tree
SLAM	Simultaneous Localization and Mapping

SUMÁRIO

1	INTRODUÇÃO	8
2	REVISÃO DA LITERATURA.....	9
2.1	Robot Operating System(ROS2) 2.....	9
2.2	Árvores de comportamento	9
2.3	Mapeamento	10
2.3.1	Mapas de custo	10
2.3.2	Sensores e SLAM.....	11
2.4	Navigation2	12
3	METODOLOGIA	14
3.1	Descrição do robô	14
3.2	Configuração do Nav2.....	14
3.2.1	Ambiente de simulação.....	14
3.2.2	Localização e Mapeamento	14
3.2.2.1	Mapas de custo	14
4	CONCLUSÃO	17
	REFERÊNCIAS	18
	APÊNDICES	19
	ANEXOS	20

1 INTRODUÇÃO

Figura 1: Mercado global de robôs autônomos de 2016 a 2021, com projeção até 2028.



Fonte: Statista (2023)

CITAR (AMAZON, 2022) nome: Proteus

2 REVISÃO DA LITERATURA

2.1 ROBOT OPERATING SYSTEM(ROS2) 2

O ROS 2 é a segunda geração do Robot Operating System, um *framework* para desenvolvimento de robôs. Ele foi desenvolvido a partir do zero para atender as necessidades de robôs modernos, com suporte para customização extensiva. É baseado no padrão Data Distribution Service(DDS), que é um padrão de comunicação utilizado em sistemas de infraestrutura crítica, como sistemas militares e financeiros (MACENSKI; FOOTE et al., 2022).

Uma mudança relevante a este trabalho do ROS 2 é nos padrões de comunicação. Existem três tipos de comunicação no ROS 2, *topics*, *services* e *actions*. *Topics* são canais de comunicação unidirecionais, onde um nó publica uma mensagem e outros nós podem se inscrever para receber essa mensagem. *Services* funcionam de forma cliente servidor, utilizando o padrão de requisição e resposta. *Topics* e *services* já existiam no ROS 1.

Actions, por outro lado, são únicos ao ROS 2. Este padrão de comunicação é utilizado para tarefas de longa duração, em que o cliente envia uma requisição e o servidor responde com um *feedback* periódico durante a execução, além do resultado da tarefa ao seu término, podendo ser falha ou sucesso. Durante a execução, é possível cancelar a tarefa.

Actions podem ser usados, por exemplo, em tarefas de navegação, em que um *action client* envia uma requisição com um ponto de destino para um *action server* que responde com um *feedback* periódico da posição atual do robô e com o resultado final. Sua definição a torna apropriada na utilização em árvores de comportamento.

2.2 ÁRVORES DE COMPORTAMENTO







Árvores de comportamento, em inglês *behavior trees*(BT), foram desenvolvidas na indústria de jogos para aplicação em inteligência artificial de personagens não jogáveis, substituindo máquinas de estado. Elas se destacam por sua modularidade e reatividade, porém mantendo as funcionalidades esperadas de uma máquina de estado (COLLEDANCHISE; ÖGREN, 2018).

Além de seu uso na indústria de jogos, árvores de comportamento também são utilizadas em projetos de robótica, como o robô JIBO ou o projeto iQmatic da Scania, que utiliza árvores de comportamento no sistema de navegação caminhões autônomos (COLLEDANCHISE; ÖGREN, 2018). Além disso, árvores de comportamento são uma peça fundamental do pacote de navegação *Navigation2* do ROS 2.

O funcionamento de uma árvore de comportamento ocorre através de uma série de sinais enviados aos nós de uma árvore com uma frequência fixa. Este nó responde com o estado atual da execução, que pode ser *running*, se está em execução, *success*, se atingiu o

objetivo, ou *failure* nos demais casos. Na formulação clássica, existem quatro categorias de nós de controle (*Sequence*, *Fallback*, *Parallel* e *Decorator*) e duas categorias de nós de execução (*Action* e *Condition*). A Tabela 1 mostra os símbolos utilizados para representar os nós de uma árvore de comportamento.

Tabela 1: Símbolos dos nós de uma árvore de comportamento.

Tipo de nó	Símbolo
<i>Fallback</i>	
<i>Sequence</i>	
<i>Parallel</i>	
<i>Action</i>	
<i>Condition</i>	
<i>Decorator</i>	

Fonte: Elaborado pelo autor

O resultado dos nós de controle dependem dos resultados de seus nós filhos. Por exemplo, o nó *Sequence* executa seus filhos em ordem até encontrar um nó que retorna *failure* ou *running*. Caso não encontre, retorna *success*. O nó *Fallback* funciona de forma semelhante porém procura filhos que retornem *success* ou *running*, só retornando *Failure* caso contrário. O nó *Parallel* executa todos os filhos em paralelo, com o resultado dependendo do estado de execução dos filhos. O nó *Decorator* modifica o resultado de um nó filho de acordo com uma regra definida pelo usuário.

O nó de execução *Action* executa um comando, e retorna o resultado final deste comando, como *sucess* caso o objetivo seja atingido, ou *failure* caso contrário. Enquanto a tarefa está sendo executada, o nó retorna *running*. Nota-se que este nó tem definição semelhante ao *Action* do ROS 2. Finalmente, o nó *Condition* testa uma condição e retorna *success* ou *failure* caso a condição seja verdadeira ou falsa, respectivamente.

2.3 MAPEAMENTO

Para navegação autônoma, o robô deve ter conhecimento prévio do ambiente para planejamento de trajetórias. Existem diversas formas de representação do ambiente, como mapas de gradientes, mapas de custo e vetores de espaços. Neste trabalho, o foco será no mapa de custo.

O mapeamento também auxilia na localização do robô, comparando o mapa construído com os dados dos sensores em tempo real. Além disso, os dados dos sensores podem ser utilizados para atualizar o mapa de custo, em casos de ambientes pouco conhecidos ou dinâmicos.

É possível utilizar os dados de localização e dos sensor para construir um novo mapa. Esta técnica é conhecida como *Simultaneous Localization and Mapping (SLAM)*, que permite a criação de mapas para ambientes não conhecidos.

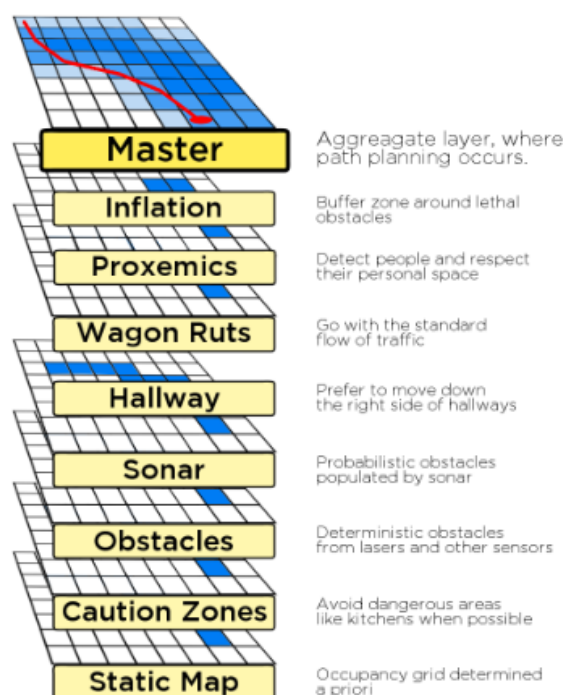
2.3.1 Mapas de custo

Um mapa de custo é uma representação de ambiente composta por uma grade de células que contém um custo, variando de desconhecido, livre, ocupado ou custo inflado.

Em mapas de custo tradicionais, as informações de custo são armazenadas em mapas monolíticos, para utilização em planejamento de trajetórias. Esta implementação é utilizado com sucesso para caminhos curtos, mas pode apresentar dificuldade em lidar com ambientes dinâmicos maiores (LU; HERSHBERGER; SMART, 2014).

Uma solução para este problema são mapas de custo com camadas, que separam o processamento dos dados dos mapas de custos em camadas semanticamente distintas. Por exemplo, os dados dos sensores e o mapa estático previamente conhecido são processados em camadas separadas e depois combinados em um único mapa de custo. A Figura 2 mostra uma configuração possível de camadas de mapas de custo.

Figura 2: Exemplo de configuração de camadas de um mapa de custo.



Fonte: Lu, Hershberger e Smart (2014)

2.3.2 Sensores e SLAM

A escolha do sensor é importante para o mapeamento, pois afeta a qualidade e quantidade de informações obtida pelo robô, além de determinar a escolha das ferramentas utilizadas para o mapeamento do ambiente (CHONG et al., 2015).

Sensores acústicos, como sonares e sensor de distância a laser, são utilizados em ferramentas SLAM 2D tradicionais. Estes sistemas são robustos e bem estabelecidos, com fácil integração ao sistema de navegação do ROS 2.

Porém, com o avanço da tecnologia, sensores RGB-D e câmeras estéreo estão se tornando mais acessíveis, influenciando o desenvolvimento de sistemas de *Visual SLAM* (VSLAM). Dentre sistemas de VSLAM, destacam-se o ORB-SLAM3, OpenVSLAM e RTABMap, que possuem suporte a câmeras RGB-D e permitem localização pura.

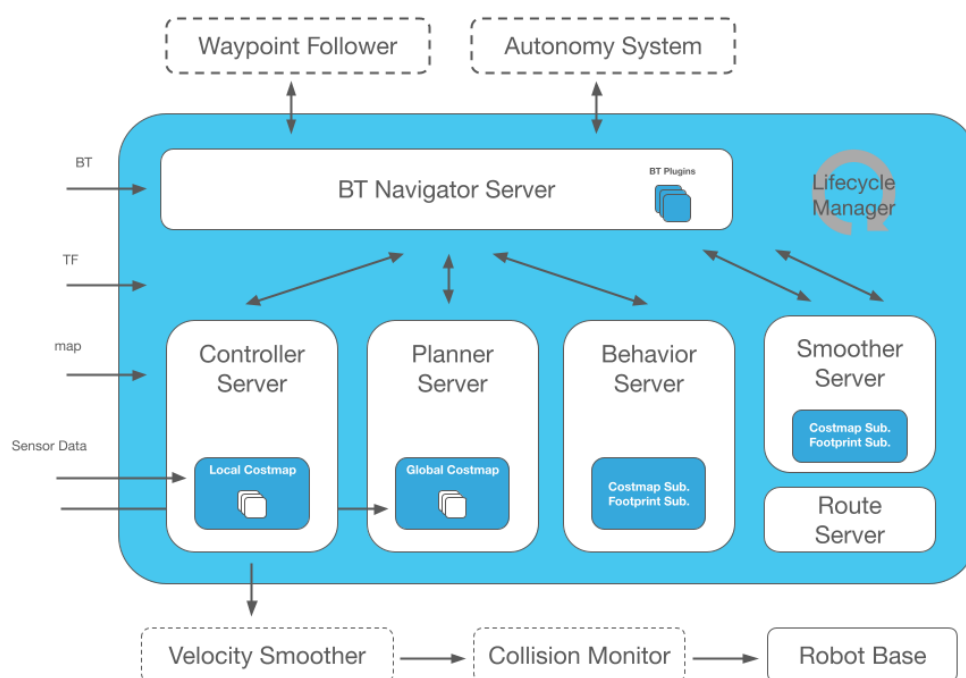
Em Merzlyakov e Macenski (2021), é feita uma comparação entre estes sistemas, mostrando que o OpenVSLAM é a técnica mais adequada para maioria dos casos. Porém, para ambientes internos com câmeras RGB-D, o RTABMap também teve um bom desempenho. Estes sistemas, porém, não são integrados nativamente ao *Navigation2*.

deveria me aprofundar mais nos sistemas de slam?

2.4 NAVIGATION2

O *Navigation2* (Nav2) é o sucessor do ROS *navigation stack*, permitindo a realização de tarefas complexas em diversos ambientes e classes de robôs cinemáticos. Baseando-se no legado do *navigation stack* do ROS 1, o Nav2 foi construído em cima do ROS2, implementando técnicas mais modernas para ter um sistema modular propício para ambientes dinâmicos com suporte a uma maior variedade de sensores (MACENSKI; MARTIN et al., 2020).

Figura 3: Arquitetura do *Navigation2*



Fonte: Navigation2 (2020)

Na Figura 3 é mostrada a arquitetura do Nav2. O *Behavior Tree(BT) Navigator Server* usa uma árvore de comportamento para orquestrar as tarefas de navegação, ativando os servidores de controle, planejamento e recuperação para navegar. Para executar nós de *actions*, normalmente são utilizados *Action servers* do ROS 2. Esta árvore de comportamento pode ser configurada pelo usuário através de um arquivo em XML, permitindo a descrição de comportamentos de navegação únicos sem necessidade de programação.

Além disso, todos estes servidores utilizam o conceito de *Managed Nodes*, também conhecidos como *Lifecycle Nodes*. Estes nós utilizam máquinas de estados para gerenciar seu ciclo de vida, utilizando transições de estado desde sua criação a destruição. No caso de falha ou desligamento, o nó vai do estado ativo ao estado finalizado seguindo a máquina de estados, permitindo que o sistema seja finalizado de forma segura.

Na arquitetura também pode-se notar a utilização de dois mapas de custo, um local e outro global. O mapa local, utilizado no servidor do controlador, é utilizado para planejamento a curto prazo e prevenção de colisão, enquanto o mapa global, utilizado no servidor de planejamento, é utilizado principalmente para planejamento a longo prazo.

3 METODOLOGIA

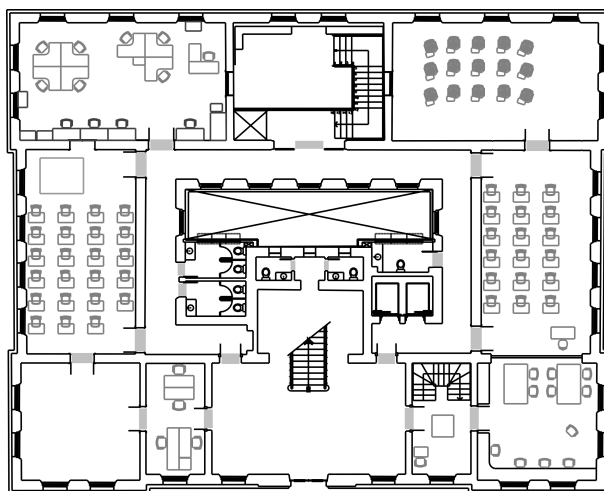
3.1 DESCRIÇÃO DO ROBÔ

falar do twil description falar do plugin da camera citar o pacote do intelrealsense
falar do controlador

3.2 CONFIGURAÇÃO DO NAV2

3.2.1 Ambiente de simulação

Figura 4: Planta do 1º andar do prédio Centenário da EE-UFRGS



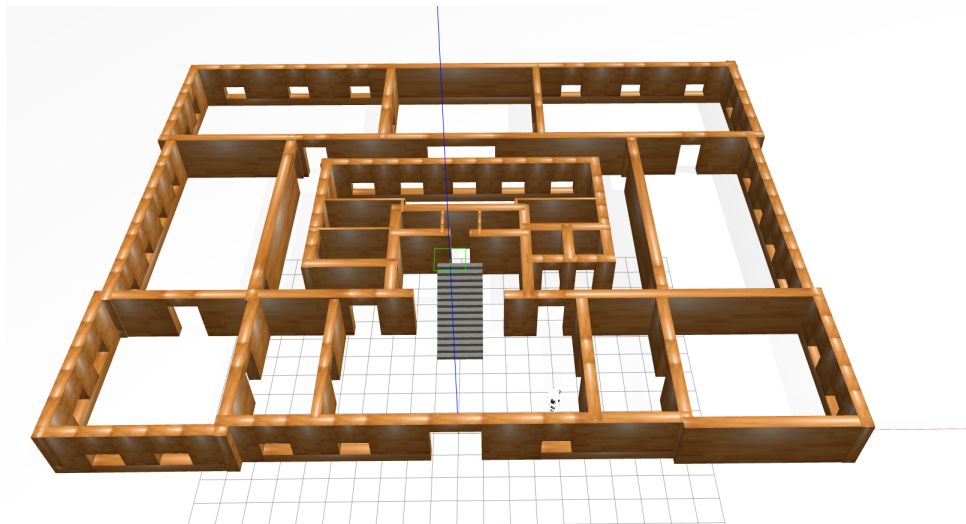
Fonte: citar o petry aqui

3.2.2 Localização e Mapeamento

3.2.2.1 Mapas de custo

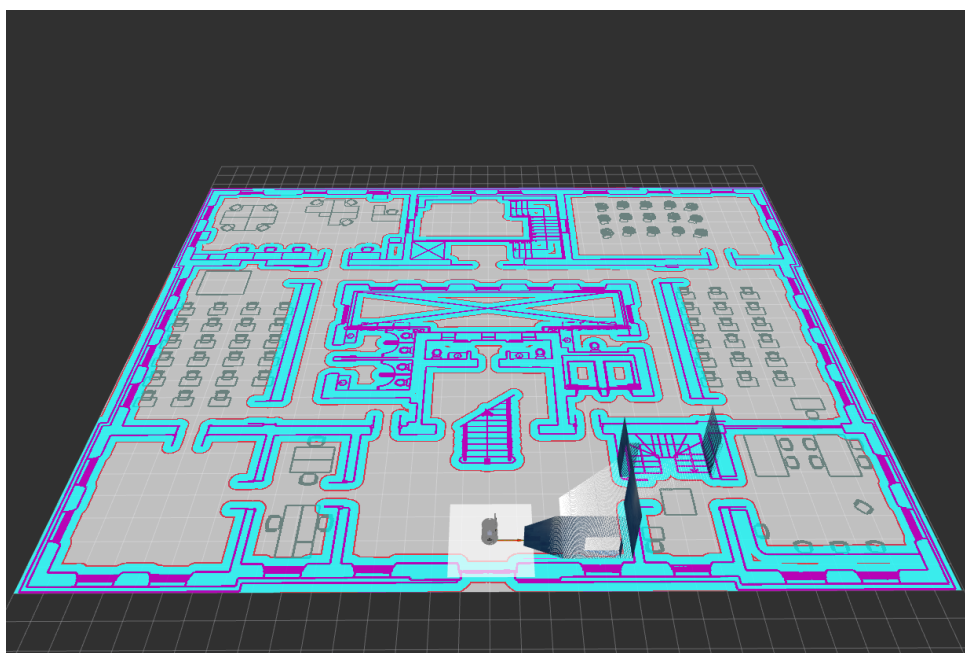
Citar (ZHENG, 2019) para falar da inflation layer
nav2_amcl e slam_toolbox são slam 2d e só funcionam com mensagens do tipo LaserScan (de sensores LIDAR, etc..)
para usar a câmera RGB-D na transformação map -> odom, teria que utilizar um pacote VSLAM

Figura 5: Ambiente Gazebo com modelo do prédio Centenário da EE-UFRGS



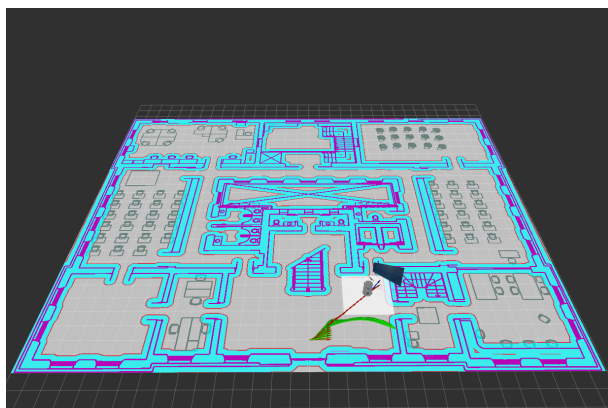
Fonte: Autor

Figura 6: RViz com robô twil no ambiente do prédio Centenário da EE-UFRGS



Fonte: Autor

Figura 7: RViz mostrando o erro de odometria *tirar essa?*



Fonte: Autor

4 CONCLUSÃO

falar do robô real rosdeps/containerização

REFERÊNCIAS

- AMAZON. *10 years of Amazon robotics: how robots help sort packages, move product, and improve safety*. 2022. Disponível em: <<https://www.aboutamazon.com/news/operations/10-years-of-amazon-robotics-how-robots-help-sort-packages-move-product-and-improve-safety>>. Acesso em: 8 abr. 2023. Acesso em: 04 de ago. de 2023.
- CHONG, T. et al. Sensor Technologies and Simultaneous Localization and Mapping (SLAM). *Procedia Computer Science*, v. 76, p. 174–179, 2015. 2015 IEEE International Symposium on Robotics and Intelligent Sensors (IEEE IRIS2015). ISSN 1877-0509. DOI: <https://doi.org/10.1016/j.procs.2015.12.336>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1877050915038375>>.
- COLLEDANCHISE, M.; ÖGREN, P. *Behavior Trees in Robotics and AI*. [S.l.]: CRC Press, jul. 2018. DOI: 10.1201/9780429489105. Disponível em: <<https://doi.org/10.1201/2F9780429489105>>.
- LU, D. V.; HERSHBERGER, D.; SMART, W. D. Layered costmaps for context-sensitive navigation. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. [S.l.: s.n.], 2014. P. 709–715. DOI: 10.1109/IR0S.2014.6942636.
- MACENSKI, S.; MARTIN, F. et al. The Marathon 2: A Navigation System. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). [S.l.]: IEEE, out. 2020. DOI: 10.1109/iros45743.2020.9341207. Disponível em: <<https://doi.org/10.1109/2Firos45743.2020.9341207>>.
- MACENSKI, S.; FOOTE, T. et al. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, v. 7, n. 66, eabm6074, 2022. DOI: 10.1126/scirobotics.abm6074. Disponível em: <<https://www.science.org/doi/abs/10.1126/scirobotics.abm6074>>.
- MERZLYAKOV, A.; MACENSKI, S. *A Comparison of Modern General-Purpose Visual SLAM Approaches*. [S.l.: s.n.], 2021. arXiv: 2107.07589 [cs.R0].
- NAVIGATION2. *Nav2 Overview*. 2020. Disponível em: <<https://navigation.ros.org/index.html>>. Acesso em: 30 de jul. de 2023.
- STATISTA. *Size of the global market for autonomous mobile robots (AMR) from 2016 to 2021, with a forecast through 2028*. 2023. Disponível em: <<https://www.statista.com/statistics/1285835/worldwide-autonomous-robots-market-size/>>. Acesso em: 8 abr. 2023. Acesso em: 04 de ago. de 2023.
- ZHENG, K. *ROS Navigation Tuning Guide*. [S.l.: s.n.], 2019. arXiv: 1706.09068 [cs.R0].

Apêndices

Anexos