



Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Programa de Pós-Graduação em Engenharia Elétrica

Rômulo Afonso Luna Vianna de Omena

**Sistema de Computação em Borda para Controle Preditivo de Veículos
Autoguiados em Redes sem Fio sujeitas à Degradação**

Campina Grande – PB
2023

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Programa de Pós-Graduação em Engenharia Elétrica

Sistema de Computação em Borda para Controle Preditivo de Veículos
Autoguiados em Redes sem Fio sujeitas à Degradação

Rômulo Afonso Luna Vianna de Omena

Tese de Doutorado submetida à Coordenação do Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para obtenção do grau de Doutor em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Processamento da Informação
Linha de Pesquisa: Engenharia da Computação

Angelo Perkusich, D.Sc.
Danilo Freire de Souza Santos, D.Sc.
(Orientadores)

Campina Grande, Paraíba, Brasil

©Rômulo Afonso Luna Vianna de Omena, Março de 2023

O55s

Omena, Rômulo Afonso Luna Vianna de.

Sistema de computação em borda para controle preditivo de veículos autoguiados em redes sem fio sujeitas à degradação / Rômulo Afonso Luna Vianna de Omena. – Campina Grande, 2023.

169 f.: il. color.

Tese (Doutorado em Engenharia Elétrica) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2023.

"Orientação: Prof. Dr. Angelo Perkusich; Prof. Dr. Danilo Freire de Souza Santos".

Referências.

1. Indústria 4.0. 2. Computação na Borda. 3. Controle Preditivo baseado em Modelo. 4. Veículos Autoguiados. I. Perkusich, Angelo Perkusich. II. Santos, Danilo Freire de Souza. III. Título.

CDU 681.51(043)



MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
POS-GRADUACAO EM ENGENHARIA ELETRICA
Rua Aprigio Veloso, 882, - Bairro Universitario, Campina Grande/PB, CEP 58429-900

REGISTRO DE PRESENÇA E ASSINATURAS

ATA DA DEFESA PARA CONCESSÃO DO GRAU DE DOUTOR EM CIÊNCIAS, NO DOMÍNIO DA ENGENHARIA ELÉTRICA, REALIZADA EM 17 DE MARÇO DE 2023(Nº359)

CANDIDATO: **RÔMULO AFONSO LUNA VIANNA DE OMENA**. COMISSÃO EXAMINADORA: ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG, Presidente da Comissão e Examinador interno, ANGELO PERKUSICH, DEE, UFCG, DANILLO FREIRE DE SOUZA SANTOS, Dr., UFCG, Orientadores, JAIDILSON JÓ DA SILVA, D.Sc., UFCG, Examinador interno, ALISSON VASCONCELOS DE BRITO, D.Sc., UFPB, CARLOS EDUARDO PEREIRA, Dr., UFRGS, DALTON CÉZANE GOMES VALADARES, Dr., IFPE, Examinadores externos. TÍTULO DA TESE: Sistema de Computação em Borda para Controle Preditivo de Veículos Autoguiados em Redes sem Fio sujeitas à Degradação. ÁREA DE CONCENTRAÇÃO: Processamento da Informação. HORA DE INÍCIO: **14h00** – LOCAL: **Sala Virtual, conforme Art. 5º da PORTARIA SEI Nº 01/PRPG/UFCG/GPR, DE 09 DE MAIO DE 2022**. Em sessão pública, após exposição de cerca de 45 minutos, o candidato foi arguido oralmente pelos membros da Comissão Examinadora, tendo demonstrado suficiência de conhecimento e capacidade de sistematização, no tema de sua tese, obtendo conceito APROVADO. Face à aprovação, declara o presidente da Comissão, achar-se o examinado, legalmente habilitado a receber o Grau de Doutor em Ciências, no domínio da Engenharia Elétrica, cabendo a Universidade Federal de Campina Grande, como de direito, providenciar a expedição do Diploma, a que o mesmo faz jus. Na forma regulamentar, foi lavrada a presente ata, que é assinada por mim, Filipe Emmanuel Porfírio Correia, e os membros da Comissão Examinadora presentes. Campina Grande, 17 de Março de 2023.

Filipe Emmanuel Porfírio Correia
Secretário

ANTONIO MARCUS NOGUEIRA LIMA, Dr., UFCG
Presidente da Comissão e Examinador interno

ANGELO PERKUSICH, DEE, UFCG
Orientador

DANILO FREIRE DE SOUZA SANTOS, Dr., UFCG
Orientador

JAIDILSON JÓ DA SILVA, D.Sc., UFCG
Examinador interno

ALISSON VASCONCELOS DE BRITO, D.Sc., UFPB
Examinador externo

CARLOS EDUARDO PEREIRA, Dr., UFRGS
Examinador externo

DALTON CÉZANE GOMES VALADARES, Dr., IFPE
Examinador externo

RÔMULO AFONSO LUNA VIANNA DE OMENA
Candidato

2 - APROVAÇÃO

2.1. Segue a presente Ata de Defesa de Tese de Doutorado do candidato RÔMULO AFONSO LUNA VIANNA DE OMENA, assinada eletronicamente pela Comissão Examinadora acima identificada.

2.2. No caso de examinadores externos que não possuam credenciamento de usuário externo ativo no SEI, para igual assinatura eletrônica, os examinadores internos signatários **certificam** que os examinadores externos acima identificados participaram da defesa da tese e tomaram conhecimento do teor deste documento.



Documento assinado eletronicamente por **FILIFE EMMANUEL PORFIRIO CORREIA, SECRETÁRIO (A)**, em 30/03/2023, às 15:24, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da [Portaria SEI nº 002, de 25 de outubro de 2018](#).



Documento assinado eletronicamente por **ANTONIO MARCUS NOGUEIRA LIMA, PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 30/03/2023, às 15:44, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da [Portaria SEI nº 002, de 25 de outubro de 2018](#).



Documento assinado eletronicamente por **JAIDILSON JO DA SILVA, PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 30/03/2023, às 15:44, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da [Portaria SEI nº 002, de 25 de outubro de 2018](#).



Documento assinado eletronicamente por **DANILO FREIRE DE SOUZA SANTOS, PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 30/03/2023, às 17:11, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da [Portaria SEI nº 002, de 25 de outubro de 2018](#).



Documento assinado eletronicamente por **Dalton Cézane Gomes Valadares, Usuário Externo**, em 30/03/2023, às 22:00, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da [Portaria SEI nº 002, de 25 de outubro de 2018](#).



Documento assinado eletronicamente por **ANGELO PERKUSICH, PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 03/04/2023, às 07:05, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da [Portaria SEI nº 002, de 25 de outubro de 2018](#).



Documento assinado eletronicamente por **Alisson Vasconcelos de Brito, Usuário Externo**, em 03/04/2023, às 08:22, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da [Portaria SEI nº 002, de 25 de outubro de 2018](#).



A autenticidade deste documento pode ser conferida no site <https://sei.ufcg.edu.br/autenticidade>, informando o código verificador **3262597** e o código CRC **F11FC1C9**.

Resumo

Os veículos autoguiados ou AGVs (*Automated Guided Vehicles*) são elementos essenciais para o transporte de materiais na indústria. No cenário da Indústria 4.0 e da Internet Industrial das Coisas, espera-se que a frota de AGVs esteja conectada e integrada ao sistema de gerenciamento da fábrica, sendo flexível e se adaptando às novas demandas. Os sistemas de controle de AGVs com navegação em rotas fixas podem não atender a esses requisitos. A computação na borda traz os recursos computacionais de nuvem para a borda da rede, portanto, estarão mais próximos dos usuários. Esses recursos podem ser acessados com rede de comunicação sem fio e assim aplicados em demandas industriais. Os AGVs podem se beneficiar disso ao transferir para o servidor de borda as atividades que demandam mais recursos computacionais. Entretanto, a rede sem fio no ambiente industrial está sujeita às degradações no sinal, que podem ser causadas por interferências, reflexões do sinal, efeitos de sombreamento, absorção das ondas eletromagnéticas, entre outros desafios. O AGV, como um robô móvel, pode atravessar áreas em que o sinal esteja degradado, podendo resultar em colisões e acidentes. Resultados de experimentos realizados sugerem que o Controle Preditivo baseado em Modelo (MPC - *Model Predictive Control*) executado no servidor de borda, aliado a uma estratégia de compensação de atrasos e perdas de pacotes implementada no robô, pode ser utilizado para mitigar essas degradações na rede de comunicação. Na sequência, uma arquitetura que inclui uma dupla camada de MPCs é proposta para o controle de múltiplos AGVs. A primeira camada, executada no servidor de borda, planeja a trajetória dos AGVs de modo global, prevenindo colisões dos AGVs com obstáculos fixos e entre si. No computador embarcado no AGV, o compensador utilizado nos experimentos anteriores dá lugar a outro MPC de rastreamento da trajetória, o qual deve receber do servidor de borda a trajetória do respectivo AGV e rastreá-la. Resultados de experimentos realizados em quatro cenários de validação indicam que, a partir da arquitetura proposta é possível conduzir os AGVs sem colisões, inclusive na ocorrência de atrasos e perdas de pacotes na rede de comunicação. Além disso, as atividades que demandam mais recursos computacionais são executadas no servidor de borda, de forma que o computador embarcado no AGV pode ser mais simples, reduzindo os custos e o consumo de bateria.

Abstract

Automated Guided Vehicles (AGVs) are essential for industry material transportation. In the Industry 4.0 and Industrial Internet of Things scenario, the AGV fleet is expected to be connected and integrated into the factory management system, being flexible and adapting to new demands. AGV control systems with fixed path navigation may not meet these requirements. Edge computing brings cloud resources to the network's edge, making them closer to users. These resources can be accessed through a wireless network and applied to industrial demands. The AGVs can benefit from this when offloading tasks that require more computing resources to the edge server. However, the wireless network in the industrial environment is subject to degradation due to interference, signal reflections, shadowing effects, and electromagnetic wave absorption, among other challenges. The AGV, as a mobile robot, may traverse areas where the signal is degraded, increasing risks of collisions and accidents. Results of experiments suggest that Model Predictive Control (MPC) executed at the edge server, combined with a delay and packet loss compensation strategy implemented in the robot, can be used to mitigate these network degradations. In sequence, a two-tier architecture with MPCs is proposed to control multiple AGVs. The first tier, executed on the edge server, plans the trajectory of the AGVs globally, preventing collisions of the AGVs with fixed obstacles and each other. In the computer embedded in the AGV, the compensator used in the previous experiments gives place to a trajectory-tracking MPC, which must receive the trajectory of the respective AGV from the edge server and track it. Results of experiments carried out in four validation scenarios indicate that from the proposed architecture, it is possible to drive the AGVs without collisions, even in the communication network's occurrence of delays and packet losses. In addition, tasks that demand more computational resources are offloaded to the edge server so that the computer embedded in the AGV can have more restricted resources, reducing costs and battery consumption.

Agradecimentos

Agradeço aos meus pais Reinaldo e Simone, e à minha tia Silviane, que ao longo do período do doutorado sempre procuravam saber do andamento do trabalho e também pelos incentivos.

À minha esposa Poliana Urtiga, pelo incentivo e por entender os momentos de ausência para dedicação ao doutorado. Agradeço também pela companhia e pelos momentos de descontração durante o compartilhamento do escritório em casa.

Aos meus irmãos Reinaldo e Manoela, pelo apoio e pelos momentos de descontração.

Aos meus sobrinhos Joaquim, Pedro e Nina, pelos momentos de alegria que tivemos no período das etapas finais deste trabalho, ajudando a desopilar e recarregar as energias para continuar a escrita da tese.

À família Nascimento Urtiga em Campina Grande-PB, por todo o suporte enquanto estive na cidade, passando pela graduação, mestrado e doutorado.

Aos meus orientadores, Professores Angelo Perkusich e Danilo Santos, pelas discussões, direcionamentos e sugestões durante o desenvolvimento deste trabalho. Dos tempos de confinamento da pandemia do Covid-19 até as etapas finais do doutorado, a nossa interação passou a ser remota, mesmo assim, sempre estiveram disponíveis quando precisei de suporte.

Aos professores da banca examinadora, Antonio Lima (UFCG), Jaidilson Silva (UFCG), Alisson Brito (UFPB), Carlos Pereira (UFRGS) e Dalton Valadares (IFPE) pelas críticas e sugestões.

Ao Professor Dalton Valadares e aos colegas de doutorado Marcus Bezerra e Daniel Macedo, pelo suporte e troca de conhecimentos nas nossas reuniões remotas das tardes de terça-feira.

Aos servidores da COPELE-UFCG, pelo apoio administrativo.

À Pró-Reitoria de Pesquisa, Pós-Graduação e Inovação do IFAL, por conceder o afastamento das atividades de docente para dedicação exclusiva ao doutorado.

A todos os meus familiares e amigos, pelo apoio e pela comemoração em cada etapa vencida.

Conteúdo

1	Introdução	1
1.1	Justificativa	3
1.2	Problemática	5
1.3	Objetivo geral	6
1.3.1	Objetivos específicos	7
1.4	Contribuição	7
1.4.1	Publicações	7
1.5	Metodologia	8
1.6	Organização do documento	10
2	Fundamentação teórica	11
2.1	Computação na borda	11
2.1.1	Computação na borda aplicada à indústria	14
2.2	Veículos autoguiados (AGVs)	15
2.3	Controle Preditivo Baseado em Modelo (MPC)	19
2.3.1	Estratégia do MPC	20
2.3.2	Função custo	21
2.3.3	O uso do MPC na compensação de atrasos e perdas de pacotes na rede de comunicação	23
2.4	Robô móvel de tração diferencial	24
2.5	ROS 2	26
2.6	Gazebo	28
2.7	Considerações finais	28

3	Revisão bibliográfica	30
3.1	Controle e coordenação de veículos autoguiados	30
3.1.1	Planejamento da rota e do movimento	30
3.1.2	Sistemas centralizados e descentralizados	35
3.1.3	Sistemas assistidos por computação em nuvem ou na borda	38
3.1.4	Discussão	43
3.2	O controlador industrial como um serviço de nuvem ou borda	45
3.2.1	Casos específicos com o MPC	47
3.2.2	Discussão	50
3.3	Considerações finais	51
4	Implementação do MPC	52
4.1	Solução do Problema de Controle Ótimo	52
4.1.1	Único Disparo	54
4.1.2	Múltiplos Disparos	55
4.1.3	Métodos de integração	57
4.2	Demonstração no Matlab e CasADi	58
4.2.1	Estabilização em um ponto	60
4.2.2	Rastreamento da trajetória	67
4.3	Considerações finais	72
5	Arquitetura proposta com dupla camada de MPCs para navegação de AGVs assistida por computação na borda	74
5.1	Arquitetura geral	74
5.2	Nós de borda	75
5.3	Nós locais	78
5.4	Demonstração no ROS 2	80
5.5	Considerações finais	85
6	Simulações e resultados	86
6.1	Recursos computacionais utilizados	87
6.1.1	Máquinas virtuais	87

6.2	<i>Workspaces</i> e pacotes do ROS 2	89
6.3	Supervisor	91
6.4	Ferramentas de simulação	92
6.5	Arquitetura da simulação	95
6.6	Cenários de simulação	97
6.6.1	Cenário 1	99
6.6.2	Cenário 2	117
6.6.3	Cenário 3	126
6.6.4	Cenário 4	133
6.7	Considerações finais	139
7	Conclusão	141
7.1	Sugestões de trabalhos futuros	143
A	Código de exemplo para execução do MPC no CasADi e Matlab	157
A.1	Estabilização em um ponto	157
A.2	Rastreamento da trajetória	163

Lista de Símbolos, Abreviaturas e Acrônimos

4G – Quarta geração das redes de telefonia e dados móveis

5G – Quinta geração das redes de telefonia e dados móveis

AGV – *Automated Guided Vehicle*

API – *Application Programming Interface*

CLP – Controlador Lógico Programável

CPS – *Cyber-Physical System*

ETSI – *European Telecommunications Standard Institute*

IIoT – *Industrial Internet of Things*

IoT – *Internet of Things*

IPOPT – *Interior Point Optimizer*

LiDAR – *Light Detection and Ranging*

MEC – *Multi-Access Edge Computing*

MPC – *Model Predictive Control*

NetEm – *Network Emulator*

NFV – *Network Function Virtualization*

PID – Proporcional Integral Derivativo

RAN – *Radio Access Network*

RNN – *Recurrent Neural Network*

ROS – *Robot Operating System*

SDN – *Software Defined Networks*

SLAM – *Simultaneous Localization and Mapping*

TCP – *Transmission Control Protocol*

TI – Tecnologia da Informação
UAV – *Unmanned Aerial Vehicle*
UDP – *User Datagram Protocol*
UE – *User Equipment*
XaaS – *Anything as a Service*
XML – *Extensible Markup Language*
YAML – *YAML Ain't Markup Language*

Lista de Figuras

1.1	Fluxograma da metodologia.	8
2.1	Representação de um comparativo da computação em nuvem e na borda. . .	12
2.2	Representação da arquitetura de integração em três camadas: computação em nuvem, MEC e dispositivos IoT.	14
2.3	Imagens de AGVs da <i>Amazon</i> para transporte de produtos em estoques. . .	17
2.4	Representação de sistemas de navegação comuns: (a) fixos e (b)-(e) navegação livre com guias virtuais.	18
2.5	Gráfico das variáveis x_{ref} , $x(k)$ e $u(k)$ durante a implementação do MPC no horizonte de predição N	21
2.6	Representação da analogia do MPC com o modo de dirigir um carro.	22
2.7	Representação da vista superior de um robô móvel genérico com duas rodas e tração diferencial.	24
2.8	Representação de nós do ROS 2 com troca de informações através de tópicos e serviços.	27
3.1	Diagrama do fluxo de trabalho dos AGVs.	31
3.2	Diagrama com os algoritmos de planejamento da rota.	32
3.3	Representação da arquitetura de controle (a) centralizada e (b) descentralizada.	36
4.1	Ilustração da discretização para o método dos Múltiplos Disparos: (a) estado e controle no tempo contínuo; (b) estado e controle discretizados; (c) integração das trajetórias do estado.	55
4.2	Representação gráfica de um exemplo de integração com o método de Runge-Kutta de quarta ordem.	58

4.3	Representações do robô e do obstáculo feitas por círculos para formulação da restrição de prevenção de colisões.	61
4.4	Trajetórias da simulação da estabilização em um ponto: (a) trajetória parcial do robô, com a trajetória prevista representada pelos pontos vermelhos; (b) trajetória completa.	65
4.5	Ângulo de orientação do robô em função do tempo na simulação da estabilização em um ponto.	66
4.6	Sinais de controle das velocidades linear (u_v) e angular (u_ω) na simulação da estabilização em um ponto.	67
4.7	Trajetórias da simulação do rastreamento da trajetória: (a) trajetória parcial do robô, com a trajetória prevista representada pelos pontos vermelhos; (b) trajetória completa.	70
4.8	Ângulo de orientação do robô em função do tempo na simulação do rastreamento da trajetória.	71
4.9	Sinais de controle das velocidades linear (u_v) e angular (u_ω) na simulação do rastreamento da trajetória.	72
5.1	Representação da arquitetura geral para uma aplicação com n AGVs.	75
5.2	Diagrama dos nós executados no servidor de borda em um exemplo com três AGVs.	76
5.3	Diagrama dos nós executados nos AGVs. O controlador de baixo nível é utilizado para assegurar que o chassi do AGV terá as velocidades linear e angular definidas pelo MPC de rastreamento.	79
5.4	Representação do estado inicial do cenário utilizado na demonstração com o ROS 2.	81
5.5	Trajetórias percorridas pelos AGVs na demonstração com o ROS 2.	83
5.6	Sinais de controle aplicados nos AGVs na demonstração com o ROS 2.	83
5.7	Trajetórias planejadas para os AGVs 2 e 3 em diferentes iterações (linhas contínuas) e as trajetórias de fato percorridas pelos AGVs (linhas tracejadas), na demonstração com o ROS 2.	85
6.1	Representação do acesso das máquinas virtuais aos <i>workspaces</i> do ROS 2.	90

6.2	Representação da interação do nó do Supervisor com os nós do MPC no servidor de borda.	91
6.3	Imagem da interface gráfica do supervisor.	92
6.4	Imagem da janela vinculada ao Supervisor para chamar o serviço de MPC individualmente para cada AGV presente no cenário.	94
6.5	Imagem da janela vinculada ao Supervisor para ajuste individual das prioridades de cada AGV presente no cenário.	94
6.6	Imagem da interface gráfica para utilização do <i>NetEm</i>	95
6.7	Arquitetura da simulação.	96
6.8	Modelo de AGV menor, utilizado nos Cenários 1, 2 e 4.	98
6.9	Modelo de AGV maior, utilizado no Cenário 3.	98
6.10	Imagem do mundo do Cenário 1 no Gazebo.	100
6.11	Representação gráfica dos nós e tópicos ativos referentes aos AGVs 1 e 2.	103
6.12	Imagem da tela do terminal com a lista de serviços do ROS 2 do Cenário 1, vinculados ao AGV ₁	104
6.13	Imagem da tela do terminal com outros serviços vinculados ao Cenário 1.	105
6.14	Trajetórias dos AGVs nas simulações com o Cenário 1 nos perfis de rede ‘Sem atraso’, ‘300ms’ e ‘200msPL50%’.	108
6.15	Sinais de controle para o Cenário 1 na simulação com o perfil ‘Sem atraso’.	110
6.16	Intervalo de tempo das iterações do MPC para o Cenário 1 na simulação com o perfil ‘Sem atraso’.	111
6.17	Uso de CPU e memória para o Cenário 1 na simulação com o perfil ‘Sem atraso’.	111
6.18	Sinais de controle para o Cenário 1 na simulação com o perfil ‘300ms’.	112
6.19	Intervalo de tempo das iterações do MPC para o Cenário 1 na simulação com o perfil ‘300ms’.	112
6.20	Uso de CPU e memória para o Cenário 1 na simulação com o perfil ‘300ms’.	113
6.21	Sinais de controle para o Cenário 1 na simulação com o perfil ‘200msPL50%’.	113
6.22	Intervalo de tempo das iterações do MPC para o Cenário 1 na simulação com o perfil ‘200msPL50%’.	114

6.23	Uso de CPU e memória para o Cenário 1 na simulação com o perfil ‘200msPL50%’	114
6.24	Imagem do mundo do Cenário 2 no Gazebo.	117
6.25	Trajетórias dos AGVs na simulação com o Cenário 2.	121
6.26	Sinais de controle na simulação com o Cenário 2.	122
6.27	Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória na borda, na simulação com o Cenário 2.	123
6.28	Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória no lado local, na simulação com o Cenário 2.	123
6.29	Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória na borda, na simulação com o Cenário 2, com limitação de trajetórias consideradas na prevenção de colisões.	124
6.30	Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória no lado local, na simulação com o Cenário 2, com limitação de trajetórias consideradas na prevenção de colisões.	125
6.31	Imagem do mundo do Cenário 3 no Gazebo.	127
6.32	Trajетórias dos AGVs na simulação com o Cenário 3.	130
6.33	Sinais de controle na simulação com o Cenário 3.	131
6.34	Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória na borda, na simulação com o Cenário 3.	132
6.35	Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória no lado local, na simulação com o Cenário 3.	132
6.36	Imagem do mundo do Cenário 4 no Gazebo.	134
6.37	Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória na borda, na simulação do o Cenário 4 com quatro AGVs.	136
6.38	Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória na borda, na simulação do o Cenário 4 com oito AGVs.	136
6.39	Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória na borda, na simulação do Cenário 4 com doze AGVs.	137

6.40 Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória na borda, na simulação do o Cenário 4 com doze AGVs, com limitação do MPC para considerar apenas os quatro AGVs mais próximos na prevenção de colisões.	137
--	-----

Lista de Tabelas

4.1	Parâmetros ajustados na simulação da estabilização em um ponto com o Matlab.	64
4.2	Parâmetros ajustados na simulação do rastreamento da trajetória com o Matlab.	69
5.1	Estados iniciais e referências para os AGVs 1, 2, 3 e 4 na demonstração com o ROS 2.	80
5.2	Posições dos obstáculos fixos na demonstração com o ROS 2.	81
5.3	Parâmetros configurados no MPC de planejamento da trajetória na demonstração com o ROS 2.	82
5.4	Pesos Q e R ajustados no MPC de rastreamento da trajetória na demonstração com o ROS 2.	82
6.1	Descrição dos recursos computacionais utilizados.	87
6.2	Descrição dos recursos alocados para as máquinas virtuais.	88
6.3	Parâmetros configurados no MPC de planejamento da trajetória nas simulações do Cenário 1.	107
6.4	Pesos Q e R ajustados no MPC de rastreamento da trajetória nas simulações do Cenário 1.	107
6.5	Médias do intervalo de tempo das iterações do MPC, do uso de CPU e do uso de memória nas simulações do Cenário 1.	110
6.6	Parâmetros configurados no MPC de planejamento da trajetória nas simulações do Cenário 2.	120
6.7	Parâmetros configurados no MPC de planejamento da trajetória nas simulações do Cenário 3.	129

6.8	Parâmetros configurados no MPC de planejamento da trajetória nas simulações do Cenário 4.	135
6.9	Somatório das médias do uso de CPU na borda em todas as simulações do Cenário 4.	138
6.10	Métricas do lado local relacionadas ao AGV_1 nas simulações do Cenário 4.	138

Capítulo 1

Introdução

Na era da Internet das Coisas (IoT – *Internet of Things*), a computação está espalhada em inúmeros dispositivos conectados à Internet e invisivelmente embutidos nos ambientes ao nosso redor [30], com aplicações distribuídas em diversos domínios, tais como, agricultura, saúde, transporte, indústria, compras, educação, veículos e casas inteligentes [4]. Na indústria o conceito da IoT se estende para a Internet Industrial das Coisas (IIoT – *Industrial Internet of Things*), onde as “coisas” industriais, como sensores, atuadores e máquinas estão conectadas à Internet [1]. A IIoT pode ser considerada como um subconjunto da IoT, e a intersecção da IIoT com os Sistemas Ciber-Físicos (CPS – *Cyber-Physical Systems*) são a base para a Indústria 4.0 [78]. A Indústria 4.0 foi inicialmente introduzida pelo governo da Alemanha em 2011 com o objetivo de impulsionar a indústria de manufatura para a quarta revolução industrial [98].

Nas aplicações de IIoT, a computação em nuvem pode falhar no atendimento aos requisitos de tempo real, segurança e confiabilidade, quando dados em massa são enviados para processamento, que, no caso das fábricas inteligentes, o nível de Gigabytes por segundo pode ser alcançado [63]. Em operações sensíveis ao tempo, como por exemplo, a parada de emergência de uma máquina ou a frenagem de emergência de um veículo, graves consequências podem ocorrer se os dados forem enviados para a nuvem e uma resposta não for recebida em tempo hábil [63]. Uma importante solução de pesquisa para o problema é a transferência dos dados para os recursos computacionais na borda da rede, a computação na borda. Esse novo modelo de computação torna possível a análise e o processamento dos dados através dos recursos computacionais e de rede distribuídos nos caminhos entre as fontes de dados

e as plataformas de computação em nuvem [63]. É importante notar que a computação na borda não exclui a computação em nuvem, mas funciona de modo complementar [39].

A quinta geração de comunicações móveis, o 5G, promoverá a computação na borda, visto que o acesso aos recursos computacionais poderá ser realizado através das redes de acesso por rádio (RAN – *Radio Access Network*) [40]. Essa forma de acesso era antes denominada *Mobile Edge Computing*, entretanto, o conceito foi estendido para outros tipos de acesso, tal como o Wi-Fi, assim, a denominação *Multi-Access Edge Computing* (MEC) tem sido adotada [63]. Potenciais casos de uso da MEC são: redes veiculares; orquestração e *caching* de serviços de vídeo; assistência cognitiva vestível; realidade virtual em lojas; serviços de IoT; e, até a Internet Tátil, que exige uma latência de aproximadamente um milissegundo para evitar a percepção de atrasos no controle remoto de atuadores [68].

Um servidor de MEC pode desempenhar um importante papel na indústria no que se refere ao modo de execução dos sistemas de controle industrial. O hardware dos controladores legados podem ser substituídos por instâncias de software e transferidos para um servidor de MEC, aumentando a flexibilidade e reduzindo os custos operacionais [99]. Nesse sentido, aplica-se o conceito do *XaaS* (*Anything as a Service*), termo em inglês que significa “Qualquer coisa como um Serviço”, onde os provedores podem oferecer serviços sob demanda, como infraestrutura de rede, funções de rede, software, recursos computacionais, entre outros [85]. Uma aplicação em potencial, que pode ser explorada nesse cenário de controladores baseados em software com computação na borda e redes 5G, são os Veículos Autoguiados (AGVs – *Automated Guided Vehicles*), tipos de robôs móveis utilizados para transporte de materiais em ambientes industriais. Se uma fábrica já tem um sistema de comunicação 5G instalado, o mesmo sistema pode então ser utilizado para o gerenciamento da frota de AGVs, controle, integração com o estoque e análises em tempo real [62]. Por exemplo, ao usar um sistema de MEC com 5G, os dados dos AGVs podem ser coletados e alguns dos veículos podem ser alocados baseado no volume de trabalho e no estado da produção, auxiliando na redução de custos e dinamizando a manufatura [62].

Ainda no caso dos AGVs, a computação na borda reduz a complexidade dos veículos, visto que as funcionalidades que demandam mais recursos computacionais podem ser transferidas para o servidor de borda. Nesse caso, os veículos podem ser embarcados com computadores mais simples, com recursos suficientes apenas para a execução de atividades que

devem ser executadas no próprio veículo. Essa estratégia auxilia na redução do custo da frota de AGVs e também na redução do consumo de energia [41]. O gerenciamento confiável da frota de AGVs, com um sistema de localização global efetivo, rastreamento robusto das posições e comunicação confiável com os veículos, são de extrema importância e podem ser alcançados utilizando-se plataformas de controle com servidor de MEC e redes 5G [62].

1.1 Justificativa

No contexto da Indústria 4.0, a diversidade e a personalização da demanda dependem fortemente da implementação de fábricas inteligentes altamente eficientes, de baixo consumo energético, flexíveis e reconfiguráveis, onde o manuseio de materiais é uma parte importante das fábricas e afeta diretamente os custos e a qualidade da produção [95]. Os métodos atuais de manuseio de materiais são implementados de várias formas, incluindo os robôs industriais (tais como os AGVs), esteiras transportadoras ou trabalho manual [95]. Os AGVs têm sido crescentemente demandados para aplicações em sistemas flexíveis de manufatura, de modo a aumentar a produtividade e reduzir custos. Para esse propósito, o sistema de controle implantado deve coordenar os veículos de modo eficiente, evitando conflitos, colisões e *deadlocks* [66].

As frotas de AGVs enquadradas na Indústria 4.0 precisam se adaptar às mudanças, serem escaláveis, coletarem informações úteis e tomarem decisões inteligentes [22]. Espera-se que a frota seja heterogênea, flexível e dinâmica, onde cada veículo tenha habilidades específicas e esteja apto a operar em um sistema de manufatura flexível no modo “*plug-and-produce*” [22]. Sistemas avançados de AGVs, que possam ser integrados de forma robusta e fácil à arquitetura de produção da fábrica, devem ser implementados para o sucesso da fábrica do futuro e da manufatura inteligente [62].

Técnicas já consolidadas para navegação de AGVs utilizam rotas fixas previamente delimitadas. Alguns exemplos são: (i) sistemas indutivos, onde os AGVs seguem uma rota definida pelo fluxo magnético gerado pela corrente elétrica que percorre fios instalados no chão; (ii) sistemas ópticos, nos quais os AGVs seguem uma fita ou linha pintada no chão; e (iii) sistemas magnéticos, similar ao sistema óptico, mas com fitas magnéticas. As fábricas do futuro são dinâmicas e podem mudar a configuração constantemente, trazendo desvanta-

gens para os sistemas de rotas fixas [22]. Nesses casos, além da navegação dos AGVs estar limitada a caminhos fixos, é necessário executar mudanças na infraestrutura para que possam percorrer novos caminhos [62].

Na mudança de um sistema de produção tradicional para um sistema de produção inteligente da Indústria 4.0, Rodriguez *et al.* [72] sugerem alguns passos a serem seguidos, o que envolve também mudanças no controle de robôs móveis para transporte de materiais:

- substituir parte dos cabos nas linhas de produção por enlaces de comunicação sem fio e configurar um servidor de controle de manufatura baseado em nuvem, de forma a alcançar um primeiro nível de flexibilidade, considerando que a reconfiguração é mais fácil quando comparada aos sistemas cabeados;
- migrar as instruções dos Controladores Lógicos Programáveis (CLPs) para a nuvem, adicionando um grau extra de flexibilidade ao sistema de produção e permitindo que as alterações possam ser feitas rapidamente via software, além de que o hardware local pode ser simplificado;
- no caso da frota de robôs móveis, os principais objetivos são relacionados à transferência de algumas funcionalidades para a nuvem, tais como a navegação e a localização, resultando em um melhor custo-benefício e gerenciamento mais fácil da frota.

Seguindo nessa abordagem da migração do controle para o servidor de borda ou de nuvem, algumas vantagens da transferência do gerenciamento de AGVs para a computação na borda consideradas por Oyekanlu *et al.* [62], são:

- o sistema de gerenciamento da frota de AGVs pode ser integrado ao sistema de gerenciamento corporativo da fábrica;
- com o gerenciamento concentrado no servidor de borda, a central de controle mantém o estado de toda a frota e determina o caminho que cada AGV deve seguir, evitando colisões e impasses;
- os AGVs podem se comunicar e trocar dados com o sistema de alocação de tarefas, estações de trabalho e outras máquinas para minimizar o tempo de transporte e utilizar o AGV mais próximo para executar a tarefa;

- o servidor de borda tem acesso à informação global da localização de todos os AGVs, o que pode ser aproveitado para otimização das rotas dos veículos, como também facilita a navegação livre.

Dada essa necessidade da Indústria 4.0 em se ter sistemas de produção flexíveis, reconfiguráveis e baseados em nuvem, encontra-se nesse cenário uma oportunidade de pesquisa no campo dos AGVs, no que se refere à transferência do sistema de navegação desses veículos para o servidor de borda. Com isso, busca-se a flexibilidade já mencionada por Rodriguez *et al.* [72], visto que as alterações do sistema de navegação podem ser feitas via software, aliado à possibilidade de incorporar esse sistema aos outros sistemas da fábrica e tirar proveito das informações globais disponíveis para determinação das rotas dos AGVs, conforme abordado por Oyekanlu *et al.* [62].

1.2 Problemática

Ao transferir o controle para o servidor de borda e utilizar uma rede sem fio para comunicação no cenário industrial, alguns problemas que podem ocorrer são: degradação do sinal da rede causada por interferências eletromagnéticas; reflexões do sinal em superfícies metálicas; e propagação em múltiplos percursos ou efeitos de sombreamento [14; 86]. Tais problemas podem degradar o desempenho, causando redução na vazão, atrasos e perdas de pacotes [91], inclusive nas redes 5G [36] e nos casos de uso do 5G com ondas milimétricas [84].

Atualmente os AGVs são embarcados com controladores para comando dos atuadores. Existem desvantagens nessa implementação, como a escalabilidade mais difícil e o desafio na coordenação de múltiplos AGVs. Essas desvantagens trazem motivação para a implementação dos controladores em servidores remotos ou *clusters* distribuídos [59]. Por outro lado, a transferência do controlador local para um servidor remoto impõe requisitos rigorosos no enlace de comunicação entre sensores/atuadores e controladores [59].

Deve-se assegurar que mesmo nas situações de degradação do sinal, os AGVs não colidam, ainda que seja utilizada uma rede 5G, conforme abordado por Oyekanlu *et al.* [62]. Os autores acrescentam que um controlador num servidor de MEC com rede 5G deve ser

reativo o suficiente e trabalhar em sintonia com o sinal de rádio, para evitar colisões no caso de degradação do sinal.

Nakimuli *et al.* [59] avaliaram o desempenho do controle de um AGV com rede 5G, sob diversas condições, incluindo atrasos e perdas de pacotes, com o controlador executado no servidor de borda. Os autores observaram que os atrasos e perdas de pacotes causam desvios na rota do AGV em relação à rota prevista, e que em determinadas condições o desempenho é inaceitável. Além disso, os desvios na rota demandam correções constantes feitas pelo controlador, o que resulta na elevação da corrente absorvida pelo AGV e conseqüentemente aumenta o consumo de bateria.

Conforme os resultados de experimentos realizados, o Controlador Preditivo baseado em Modelo (MPC – *Model Predictive Control*) é uma alternativa para tratar o problema dos atrasos e perdas de pacotes nas redes de comunicação, os quais podem ser resultantes da degradação do sinal da rede. Assim, baseado na problemática apresentada anteriormente, nesta tese é proposto um sistema de controle de AGVs com computação na borda, com utilização da arquitetura com a dupla camada de MPCs, de modo que a camada de MPC implementada no servidor de borda se aproveita do poder computacional desse recurso, para planejar trajetórias de múltiplos AGVs livres de colisões. Além disso, a arquitetura se beneficia das ações preditivas do MPC para manter os AGVs em suas trajetórias, evitando colisões, mesmo quando o sinal da rede de comunicação estiver degradado. A segunda camada do MPC é então implementada nos AGVs para mantê-los nas trajetórias planejadas pela camada de MPC no servidor de borda. Foram realizadas simulações em diferentes cenários, incluindo cenários que representam situações hipotéticas onde a arquitetura pode ser implementada.

1.3 Objetivo geral

O objetivo geral neste trabalho é investigar, projetar e desenvolver um sistema de controle de navegação de AGVs com duas camadas de MPC, uma aplicada num servidor de borda e a outra nos veículos, para possibilitar que os AGVs naveguem em caminhos livres e que ainda estejam aptos a navegar de forma independente, segura e sem colisões, quando houver atrasos ou perda de pacotes na rede de comunicação.

1.3.1 Objetivos específicos

Para alcance do objetivo geral, os objetivos específicos são:

- investigar como utilizar o MPC para planejar trajetórias livres de colisões para múltiplos AGVs, formando a primeira camada;
- desenvolver uma segunda camada de controle que deve obter a trajetória de um determinado AGV e mantê-lo na trajetória planejada;
- investigar como incorporar as camadas no ROS (*Robot Operating System*), de modo a formar uma arquitetura de controle assistida por computação na borda;
- criar cenários para validação da arquitetura;
- validar a arquitetura utilizando um simulador de robôs.

1.4 Contribuição

A principal contribuição nesta tese é o desenvolvimento de um sistema de navegação de AGVs assistido por computação na borda, que, provido das informações globais (posições dos AGVs e do ambiente em contexto), tem capacidade para planejar as trajetórias de múltiplos AGVs e auxiliá-los na navegação destas trajetórias, para que possam executar suas tarefas de transporte de materiais em ambientes industriais. Dado que as demandas que exigem mais recursos computacionais são transferidas para o servidor de borda, o sistema desenvolvido não só auxilia na redução de custos dos veículos, mas, a partir de um controlador preditivo e das informações globais, é capaz de manter a navegação dos AGVs livre de colisões, mesmo com degradações no sinal da rede de comunicação.

1.4.1 Publicações

Durante o desenvolvimento do sistema apresentado nesta tese, os seguintes trabalhos foram publicados:

1. Rômulo Omena, Danilo Santos, Angelo Perkusich, “*An Approach to Reduce Network Effects in an Industrial Control and Edge Computing Scenario*”, *Proceedings of the*

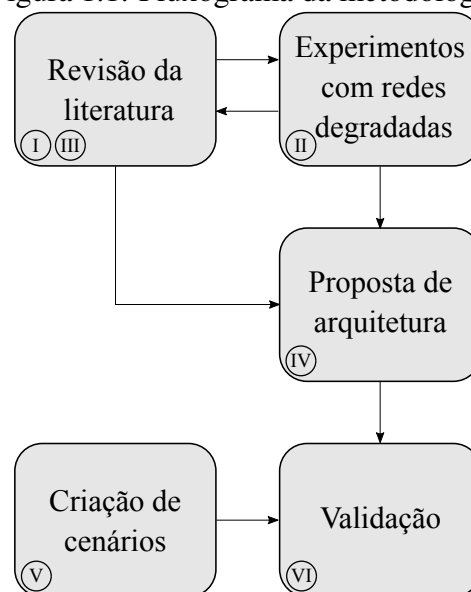
11th International Conference on Cloud Computing and Services Science - CLOSER, 2021.

2. Rômulo Omena, Danilo Santos, Angelo Perkusich, “Arquitetura com Computação na Borda e Controle Preditivo Baseado em Modelo para Controle de Veículos Autoguiados”, *Anais do XXIV Congresso Brasileiro de Automática - CBA, 2022.*
3. Rômulo Omena, Danilo Santos, Angelo Perkusich, Dalton Valadares, “*Two-tier MPC architecture for AGVs navigation assisted by edge computing in an industrial scenario*”, *Internet of Things, Volume 21, 2023.*

1.5 Metodologia

Na Figura 1.1 está apresentado o fluxograma da metodologia seguida. A partir da numeração do fluxograma, a metodologia foi dividida nas etapas descritas nos próximos parágrafos.

Figura 1.1: Fluxograma da metodologia.



Na etapa I buscou-se de identificar na revisão bibliográfica as soluções de controle que podem ser aplicadas em um cenário de computação na borda. Uma possível solução identificada foi o MPC, técnica de controle que pode ser aplicada nos casos em que o controlador e a planta trocam informações por uma rede de comunicação, junto com uma estratégia de compensação de atraso e perda de pacotes no lado local.

Com esse ponto de partida, na etapa II foram realizados experimentos para o controle de um robô móvel, onde o robô tinha o objetivo de estabilizar em uma posição de referência ou seguir uma trajetória, em um cenário com computação na borda e rede de comunicação com degradação. Com base nos resultados obtidos, o MPC foi adotado como a solução de controle a ser utilizada nesse cenário.

Retornando à revisão bibliográfica na etapa III, foram investigados os algoritmos de planejamento da rota de múltiplos robôs móveis (ou AGVs), que pudessem ser executados no servidor de borda e atuassem como provedores de referências para o MPC. Foi verificado que o MPC pode ser utilizado para obtenção de uma solução que contém a trajetória e os sinais de controle necessários para segui-la. Portanto, ao invés de receber uma trajetória de referência, o próprio MPC é capaz de planejá-la.

Ainda na etapa III, foi investigado como implementar uma camada de controle local para manter o AGV na trajetória planejada, num cenário em que o planejamento é implementado num servidor de nuvem ou de borda. Foi verificado que, ainda o MPC, atuando como um rastreador de trajetórias (recebendo uma trajetória de referência), pode ser adotado para esse fim. O MPC utilizado para rastreamento de trajetórias substituiu, portanto, a estratégia de compensação de pacotes da abordagem anterior, de modo que, em caso de atraso ou perda de pacotes na rede de comunicação, essa camada local do MPC pode continuar a rastrear a última trajetória recebida do servidor de borda.

Delimitado o problema, na etapa IV foi proposta a arquitetura de controle que engloba essas abordagens do MPC e as separa em duas camadas. Na camada presente no servidor de borda, o MPC planeja de modo global as trajetórias de múltiplos AGVs, as quais são enviadas à camada local de MPC em cada AGV para rastreamento das respectivas trajetórias.

A etapa V envolveu a criação de cenários para validação da arquitetura da etapa IV, incluindo desde cenários mais simples, até cenários que representam situações hipotéticas. A proposta de arquitetura foi por fim validada através de simulações na etapa VI. No processo de validação foi verificado se os AGVs têm capacidade de navegar sem colisões, quando degradações são induzidas na rede de comunicação. Além disso, foram analisadas algumas métricas no servidor de borda e no lado local, como o intervalo de tempo das iterações do MPC e o uso dos recursos computacionais.

1.6 Organização do documento

Neste capítulo foi apresentada a introdução desta tese, com a justificativa, a problemática, o objetivo, a contribuição e a metodologia.

No Capítulo 2 é apresentada a fundamentação teórica, com a descrição de ferramentas e conceitos abordados nesta tese.

No Capítulo 3 é apresentada a revisão bibliográfica, dividida em: controle e coordenação de veículos autoguiados; e o controlador industrial como um serviço de nuvem, abordando também os casos com o MPC.

No Capítulo 4 é apresentado o procedimento de obtenção da solução do MPC, incluindo uma demonstração utilizando ferramentas computacionais em duas aplicações associadas ao robô de tração diferencial: a estabilização em um ponto e o rastreamento da trajetória.

No Capítulo 5 é apresentada a arquitetura de controle com a dupla camada de MPCs, assistida por computação na borda e baseada no ROS 2. Ao final do capítulo, é apresentada uma demonstração com a aplicação da arquitetura em uma simulação com quatro AGVs.

No Capítulo 6 são apresentadas as simulações e os resultados da validação da arquitetura, a partir de quatro cenários.

No Capítulo 7 são apresentadas as conclusões e as sugestões para trabalhos futuros.

Capítulo 2

Fundamentação teórica

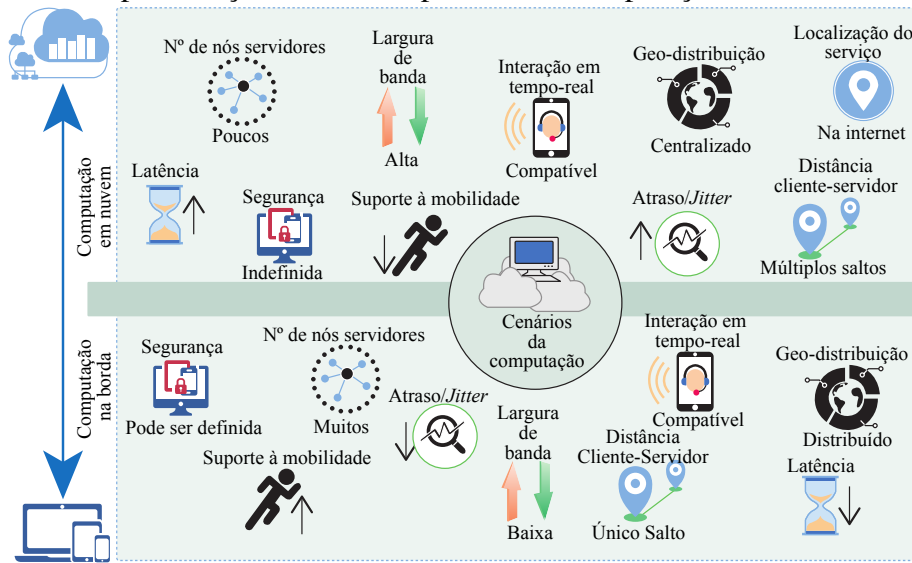
Neste capítulo está apresentada a fundamentação teórica, de forma a orientar o leitor com relação aos principais conceitos e tecnologias que são abordadas ao longo do trabalho. Inicialmente é apresentado o conceito da computação na borda, incluindo a sua utilização no cenário industrial. No restante do capítulo são apresentadas a teoria do MPC, com a estratégia, função custo e o seu uso na compensação de atrasos e perdas de pacotes na rede; a teoria dos AGVs, com o histórico, principais tipos e técnicas de navegação; o robô móvel de tração diferencial e o seu modelo cinemático; e por fim, o sistema operacional de robôs e o simulador de robôs utilizados neste trabalho, o ROS 2 e o Gazebo, respectivamente.

2.1 Computação na borda

Com o advento da IoT, um crescente número de dispositivos está conectado à internet. Utilizar os recursos computacionais da nuvem tem sido uma forma eficiente para processar os dados, considerando que a capacidade computacional da nuvem supera as dos dispositivos que estão na borda. Entretanto, a largura de banda da rede tem sido um fator limitador para a grande quantidade de dados gerados. Um veículo autônomo, por exemplo, pode gerar um Gigabyte de dados a cada segundo. O processamento em tempo real nesse caso é necessário para que as decisões corretas possam ser tomadas. Se todos os dados forem enviados para a nuvem, o tempo de resposta será muito elevado, e assim os dados precisam ser processados na borda da rede para que se tenha uma latência mais baixa, processamento mais eficiente e menor congestionamento da rede. A computação na borda surge como a tecnologia que

disponibiliza os recursos computacionais na borda da rede. A “borda” pode ser definida como quaisquer recursos de computação e rede ao longo do caminho entre as fontes de dados e os *data centers* em nuvem [77]. Na Figura 2.1 está ilustrado um comparativo entre a computação em nuvem e na borda [39], onde se pode verificar como algumas características são vistas nas duas abordagens. Comparando a computação na borda com a computação em nuvem a partir da figura, algumas características vistas são: menor latência, único salto na rede entre cliente e servidor, menor atraso/*jitter*, maior suporte à mobilidade, entre outras.

Figura 2.1: Representação de um comparativo da computação em nuvem e na borda.



Fonte: Adaptado de [39].

Quando o acesso aos recursos da computação na borda é realizado através da RAN, a computação na borda passa a ser chamada de MEC, disponibilizando aos dispositivos alto poder computacional e baixa latência na comunicação. O conceito da MEC foi inicialmente proposto pelo Instituto Europeu de Normas de Telecomunicações (ETSI – *European Telecommunications Standard Institute*). Os recursos da MEC são anexados neste caso nas estações base de comunicações móveis [50]. Ao transferir as tarefas intensivas para o servidor de borda e processar os dados nas proximidades dos usuários, as operadoras de redes móveis podem reduzir os gargalos de tráfego nas redes centrais e de *backhaul*, ao mesmo tempo que dão suporte computacional aos Equipamentos de Usuário (UE – *User Equipment*) que têm restrições no consumo de energia. Em geral, essa arquitetura de nuvem descentralizada constitui uma tecnologia base para os sistemas 5G, transformando as estações base móveis

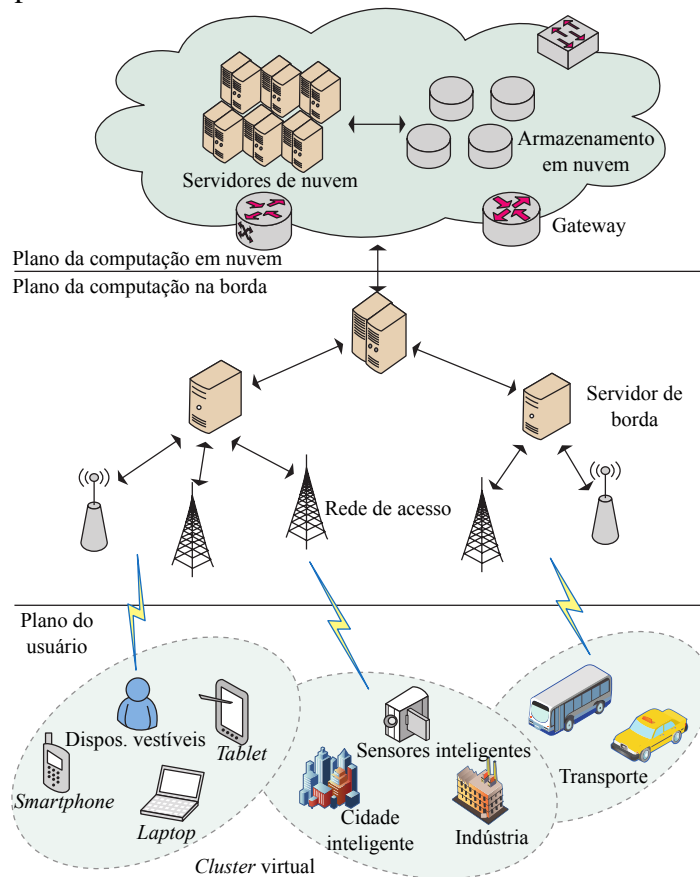
legadas, através da oferta de computação em nuvem e um ambiente de serviço de TI (Tecnologia da Informação) na borda da rede. Desde 2016, o ETSI retirou o termo “*Mobile*” do MEC e o renomeou como *Multi-access Edge Computing*, a fim de ampliar sua aplicabilidade em redes heterogêneas, incluindo Wi-Fi e tecnologias de acesso fixo [88].

Na Figura 2.2 está ilustrada a proposta de Lyu *et al.* [47] para a integração das camadas de computação em nuvem, computação na borda e aplicações de IoT. As seguintes observações são levantadas:

- a camada do plano do usuário é a camada inferior e consiste de dispositivos móveis (por exemplo, *smartphones*, *tablets* e *laptops*) e dispositivos IoT, como atuadores industriais, dispositivos vestíveis e sensores inteligentes. Estes dispositivos podem ser agrupados em *clusters* virtuais.
- o plano da computação na borda está próximo aos usuários, entregando recursos similares ao da computação em nuvem através da rede de acesso por rádio, de forma a atender aos requisitos dos serviços de IoT. A geo-distribuição dos servidores na borda podem ser organizadas numa estrutura hierárquica para utilizar os recursos de forma eficiente, agregar os serviços e sustentar as cargas de trabalho durante as horas de pico.
- o plano da computação em nuvem está no núcleo da rede e é composto por múltiplos servidores de nuvem e *data centers*, os quais têm capacidade de processar e armazenar grandes volumes de dados.

Tecnologias importantes que darão suporte à criação da MEC são as Redes Definidas por Software (SDN – *Software Defined Networks*) e as Funções de Rede Virtualizadas (NFV – *Network Function Virtualization*). Ao desacoplar o plano de controle do plano de dados, a SDN introduz um controle centralizado lógico, que pode oferecer instâncias de rede virtuais, abstraindo a infra-estrutura de rede subjacente. Já a NFV permite que funções de rede executadas em *hardware* proprietário (*firewall*, por exemplo), sejam executadas por meio de *software*, trazendo maior flexibilidade e redução de custos [88].

Figura 2.2: Representação da arquitetura de integração em três camadas: computação em nuvem, MEC e dispositivos IoT.



Fonte: Adaptado de [47].

2.1.1 Computação na borda aplicada à indústria

No cenário industrial, algumas aplicações podem ser sensíveis ao tempo e catástrofes podem acontecer caso o atraso ultrapasse os limites aceitáveis, como por exemplo, na manutenção de linhas de alta tensão, inspeção de dutos subaquáticos, fabricação e monitoramento de aeronaves a jato, mineração, operação de grandes guindastes, entre outros. No contexto da IIoT e Indústria 4.0, uma camada intermediária, tal como a computação na borda, proporcionando alta largura de banda, baixa latência e baixo *jitter*, é necessária para processar tarefas urgentes e complexas em tempo hábil. Dispositivos e sensores com recursos restritos e alimentados por bateria, também se beneficiam da computação na borda, visto que as tarefas que demandam maior processamento são transferidas para uma camada superior [1].

Qiu *et al.* [63] elencam algumas vantagens da computação na borda aplicada à IIoT:

- melhorar o desempenho do sistema: além de coletar e transmitir dados para a plataforma de computação em nuvem, a maior contribuição da computação na borda para a IIoT é o alcance do nível de processamento em milissegundos. É eficiente para reduzir o atraso geral do sistema, reduzir a demanda pela banda de comunicação e melhorar o desempenho geral do sistema.
- segurança e privacidade: a computação na borda na IIoT permite que as empresas apliquem as soluções de segurança localmente, reduzindo o risco de vazamento dos dados tanto na transmissão como também dos dados armazenados nas plataformas de nuvem.
- redução dos custos operacionais: custos estão atrelados à migração de dados para a plataforma de nuvem. A computação na borda na IIoT pode diminuir o volume de dados transmitidos, ajudando a reduzir o consumo de banda e conseqüentemente ajudando a reduzir os custos operacionais.

Chen *et al.* [15] discutem a respeito do mecanismo de cooperação formado pela combinação da computação em nuvem e na borda. A computação na borda pode dar melhor suporte ao processamento em tempo real, reduzir o tráfego na rede e melhorar a segurança e privacidade dos dados. Por outro lado, a computação em nuvem não se concentra na análise de dados em tempo real. Seu foco é implementar grandes análises de *big data* a partir dos dados obtidos nas redes de computação na borda, desempenhando um papel importante na manutenção periódica, apoio à decisão e outras atividades que não necessariamente precisam ser realizadas em tempo real. A cooperação entre nuvem e borda também acontece quando se faz uso de algoritmos de inteligência artificial. A computação em nuvem pode desenvolver modelos baseados em *big data*, enquanto a computação na borda pode processar as tarefas de inteligência artificial para os dispositivos de campo [43].

2.2 Veículos autoguiados (AGVs)

Os sistemas de veículos autoguiados são componentes chave na intralogística para transporte de materiais, com aplicações em todos os ramos da indústria e áreas de produção [89]. Com

um histórico de aproximadamente 60 anos, a história dos AGVs pode ser dividida em quatro eras [89]:

1ª era: aconteceu nos Estados Unidos da América em 1953 e foi para a Europa nos anos seguintes. Os AGVs se guiam pelo campo magnético gerado por uma fita condutora de corrente elétrica montada sobre o chão. Esse princípio é conhecido como rastreamento indutivo. O sistema de guia óptico, onde os AGVs seguem um caminho definido por uma fita colorida, também foi introduzido nesse período.

2ª era: teve início na década de 70 e terminou no início da década de 90. Ficou marcada pela introdução da eletrônica através de computadores acoplados aos veículos e pelo uso de gabinetes de controle. O sistema de rastreamento indutivo era implementado com um fio condutor montado no chão. O mesmo fio era utilizado para transmissão de dados, que também era realizada por infravermelho ou sinais de rádio. Com o aumento da produção, um maior grau de automação era necessário para reduzir os custos.

3ª era: durou desde meados da década de 90 até cerca de 2010, período em que os padrões tecnológicos foram definidos e novos mercados foram estabelecidos. Os dispositivos tinham sistemas eletrônicos para guia com sensores que não precisavam de contato físico e eram controlados por computadores comuns. O sistema de guia através de cabos condutores não é mais uma regra. A navegação livre surge a partir da assistência por sistemas magnéticos e a laser. Redes de comunicação sem fio se tornaram o padrão para transmissão de dados. Com a evolução da tecnologia e o aprimoramento da computação e dos sensores, os AGVs passaram a poder transportar diferentes tipos de carga, incluindo paletes, contêineres, fardos, pacotes e coisas similares, fazendo com que a automação do transporte passasse a ser cada vez mais adotada pela indústria.

4ª era: se estende até os dias atuais, aproveitando as técnicas da 3ª era, porém, com sistemas de sensores mais inteligentes e softwares mais avançados. Integra também tecnologias dos veículos autônomos, como sistemas de navegação a laser, radar, LiDAR (*Light Detection and Ranging*), infravermelho, ondas ultrassônicas e vídeo.

Alguns tipos de AGVs, são [89]:

- empilhadeira ou *forklift*: utilizado no transporte de paletes;

- de carga única: transportam cargas individuais, paletes ou um único volume com vários itens internos;
- rebocador: análogo a uma locomotiva puxando os vagões, o rebocador pode realizar o transporte de uma ou mais carretas com materiais.

A *Amazon* utiliza um tipo de AGV para transporte de produtos em seus estoques, onde os AGVs são menores, mas são aplicados em grande quantidade, transportando estantes com vários itens. Esse sistema de transporte de materiais foi desenvolvido pela *Kiva Systems*, posteriormente adquirida pela *Amazon* e chamada de *Amazon Robotics* [10]. Algumas imagens desses AGVs podem ser vistas na Figura 2.3.

Figura 2.3: Imagens de AGVs da *Amazon* para transporte de produtos em estoques.

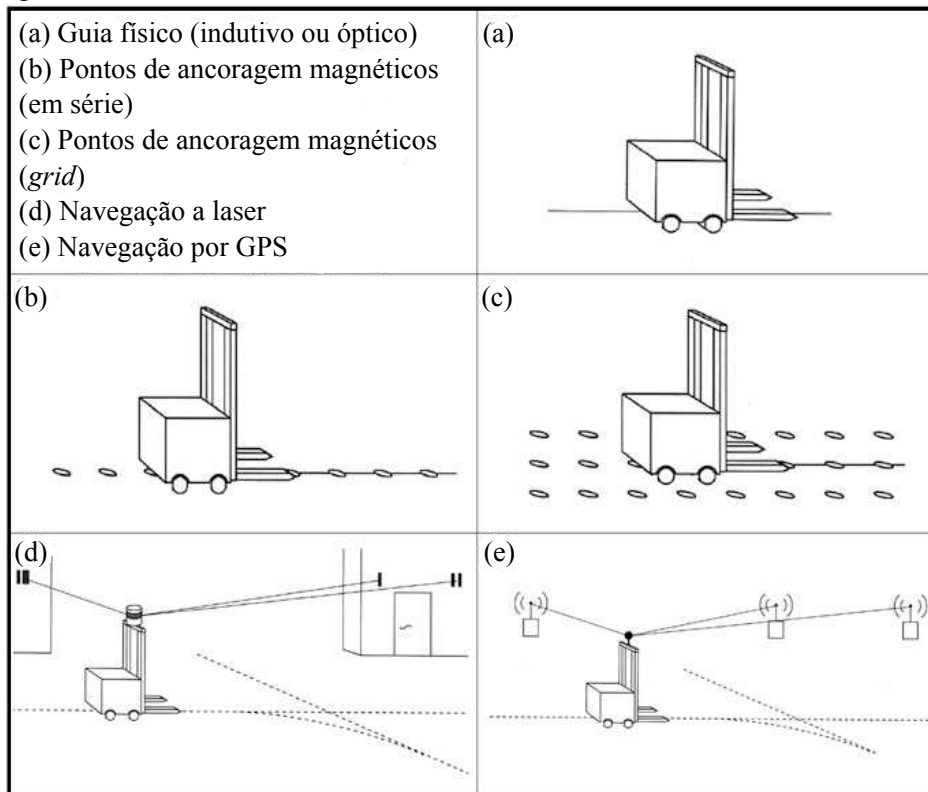


Fonte: [5].

Ullrich [89] divide os modos de navegação dos AGVs entre os que têm auxílio de guias físicos que formam rotas fixas, e os que têm rotas livres, auxiliados por pontos de ancoragem magnéticos, laser ou GPS (*Global Positioning System*), conforme ilustrados na Figura 2.4 e descritos a seguir:

- os guias físicos podem ser do tipo:
 - indutivo ativo: fios que conduzem corrente elétrica alternada são instalados no chão, induzindo corrente elétrica em duas bobinas instaladas no veículo. Um sistema usa como base a corrente diferencial nas bobinas para manter o AGV na rota fixada pelo fio no chão.
 - indutivo passivo: similar ao indutivo ativo, entretanto, os AGVs são guiados por fitas metálicas ou magnéticas fixadas no chão.

Figura 2.4: Representação de sistemas de navegação comuns: (a) fixos e (b)-(e) navegação livre com guias virtuais.



Adaptado de [89].

- óptico: uma fita colorida que tenha cor de alto contraste com o chão é fixada. Os AGVs possuem sensores para detecção e navegam pelo caminho formado pela fita.
- pontos de ancoragem magnéticos: a navegação é auxiliada por ímãs em série ou em *grid* instalados no chão.
- navegação a laser: materiais reflexivos são instalados nas paredes e pilares da instalação. Um dispositivo giratório emite feixes de laser e faz as leituras das reflexões, estimando a partir daí a posição do AGV. Dependendo do procedimento utilizado, são necessárias pelo menos duas ou três reflexões para a estimativa da posição do AGV.
- navegação por GPS: mais adequada para ambientes externos, faz a estimativa da posição do AGV a partir dos sinais captados de satélites de GPS.

Além dos sistemas de navegação relacionados anteriormente, os sistemas de odometria

têm sido aprimorados para auxiliar a navegação autônoma, tais como a odometria a laser ou odometria visual. A odometria a laser utiliza sensores LiDAR para captar informações do ambiente a partir das reflexões de laser, e assim estimar a posição do veículo. O mesmo acontece com a odometria visual, mas com a captação de informações do ambiente realizada por câmeras. As técnicas de Localização e Mapeamento Simultâneos (SLAM – *Simultaneous Localization and Mapping*) normalmente aplicam um algoritmo de odometria para obter a posição do veículo [57].

2.3 Controle Preditivo Baseado em Modelo (MPC)

O Controle Preditivo Baseado em Modelo, na língua inglesa chamado de *Model Based Predictive Control* ou *Model Predictive Control* (a sigla associada às iniciais do segundo termo é utilizada em todo o documento), começou a ser utilizado no final dos anos setenta e passou por consideráveis aprimoramentos desde então [11]. O MPC não se refere a uma estratégia de controle específica, mas a uma ampla variedade de métodos de controle que fazem o uso explícito do modelo do processo para obter o sinal de controle através da minimização de uma função custo [11]. As ideias gerais, que podem aparecer em maior ou menor grau na família do controle preditivo, são [11]:

- uso explícito do modelo para prever a saída do processo nos próximos instantes de tempo (horizonte);
- cálculo de uma sequência de controle que minimize uma função custo;
- estratégia retrocedente, portanto, em cada instante o horizonte é deslocado para o futuro, o que envolve a aplicação do primeiro sinal de controle da sequência calculada em cada iteração.

Algumas das vantagens do MPC em relação aos outros métodos de controle, são [11]:

- é atrativo para os que têm conhecimento limitado de controle, visto que os conceitos são intuitivos e ao mesmo tempo a sintonização é relativamente fácil;
- pode ser utilizado para controlar uma variedade de processos, desde os que tenham uma dinâmica simples até os mais complexos;

- pode lidar facilmente com múltiplas variáveis;
- tem compensação intrínseca para tempo morto;
- introduz o controle *feedforward* de maneira natural para compensar os distúrbios;
- o controlador resultante tem uma lei de controle fácil de implementar;
- o tratamento das restrições é simples;
- é útil quando as referências futuras (robótica ou processos em lote) são conhecidas.

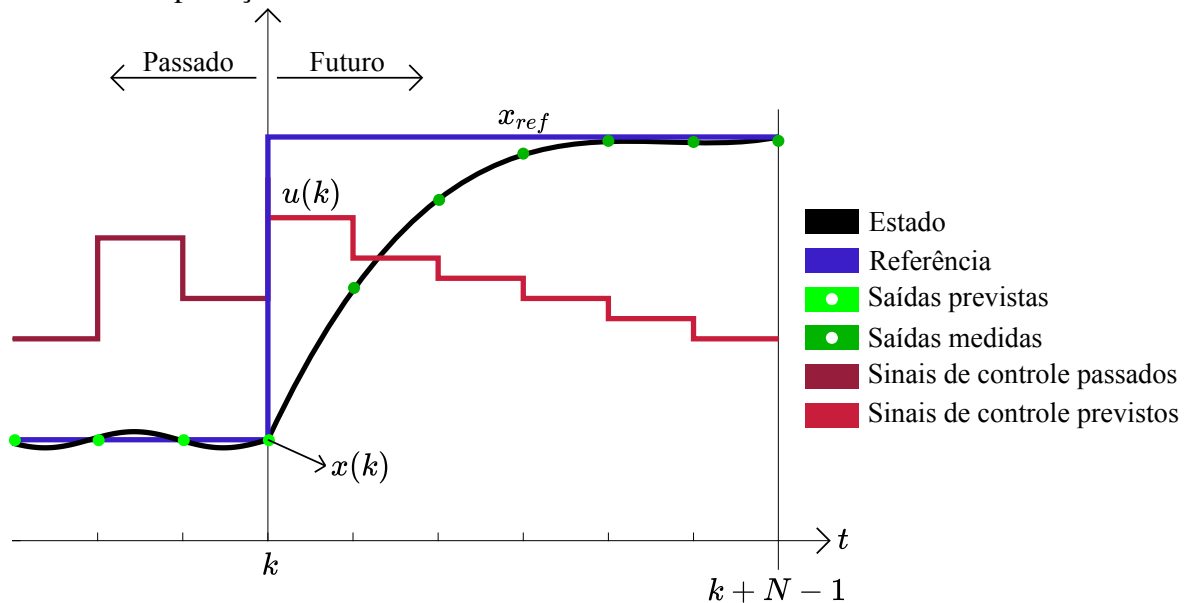
Uma das desvantagens é a necessidade de mais poder computacional em relação a outros controladores, principalmente quando restrições (relacionadas às limitações de estado do processo e limites das entradas de controle) são consideradas. Outra desvantagem é a dependência do modelo do processo, dado que os benefícios obtidos com o uso do MPC serão afetados pela discrepância entre o processo real e o modelo utilizado [11].

2.3.1 Estratégia do MPC

No MPC uma sequência de sinais de controle é resultado da minimização de uma função custo ao longo de um horizonte de predição N . Nesse processo, o estado atual é medido, enquanto os estados futuros são obtidos a partir do modelo do sistema. O primeiro sinal de controle é então aplicado na entrada do sistema e os demais são descartados. Na próxima iteração essas etapas são refeitas. Esse processo está ilustrado na Figura 2.5 e também está detalhado a seguir:

1. medir o estado estado atual $x(k)$;
2. utilizando a referência x_{ref} e o estado atual $x(k)$, calcular os sinais de controle $u(k), \dots, u(k + N - 1)$, minimizando a função custo, ou seja, dentro do horizonte de predição, minimizar a diferença entre o estado e a referência;
3. aplicar o primeiro sinal de controle $u(k)$;
4. atualizar o valor de k , fazendo $k = k + 1$;
5. retornar ao passo 1.

Figura 2.5: Gráfico das variáveis x_{ref} , $x(k)$ e $u(k)$ durante a implementação do MPC no horizonte de predição N .



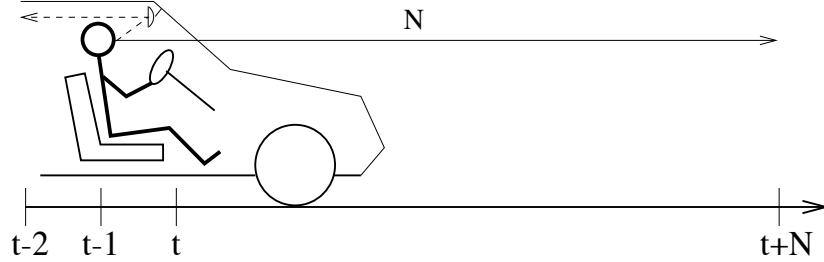
Fonte: Produzida pelo autor.

A estratégia do MPC é análoga à estratégia de controle utilizada ao se dirigir um carro. O motorista sabe a trajetória de referência desejada para um horizonte de controle finito e ao levar em consideração as características do carro (modelo mental do carro), decide qual ação de controle (acelerar, frear ou girar o volante) deve executar para seguir na trajetória desejada. Apenas a primeira ação de controle é considerada a cada instante e o procedimento é repetido para a próxima decisão de controle, numa abordagem de horizonte retrocedente. Ao utilizar esquemas clássicos de controle, como o controle PID, as ações de controle são baseadas nos erros passados. Se a analogia anterior for estendida para o controle PID, seria o equivalente a guiar o carro utilizando apenas o retrovisor [11]. A analogia com o modo de direção do carro está ilustrada na Figura 2.6.

2.3.2 Função custo

Conforme apresentado na subseção anterior, a estratégia do MPC está ligada a um problema de otimização, cuja solução é obtida através da minimização de uma função custo. Uma formulação genérica para a função custo J é dada a seguir [74]:

Figura 2.6: Representação da analogia do MPC com o modo de dirigir um carro.



Fonte: [11].

$$\min_u J = \sum_{k=0}^{N-1} \|x(k) - x_{ref}\|_Q^2 + \|u(k) - u_{ref}\|_R^2 \quad (2.1a)$$

$$\text{sujeito à} \quad x(k+1) = f(x(k), u(k)) \quad (2.1b)$$

$$x(0) = x_0 \quad (2.1c)$$

$$x_{min} \leq x(k) \leq x_{max} \quad (2.1d)$$

$$u_{min} \leq u(k) \leq u_{max} \quad (2.1e)$$

Na Equação 2.1a, N é o horizonte de predição, $x(k)$ é o estado no instante k , x_{ref} é a referência do estado, $u(k)$ é o sinal de controle a ser aplicado no instante k e u_{ref} é o sinal de controle de referência. A matriz Q é o peso que penaliza a diferença entre o estado e sua referência. O mesmo ocorre com a matriz R para o sinal de controle. Essas matrizes são diagonais e estão apresentadas a seguir:

$$Q = \begin{bmatrix} q_1 & 0 & \dots & 0 \\ 0 & q_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & q_n \end{bmatrix}, \quad R = \begin{bmatrix} r_1 & 0 & \dots & 0 \\ 0 & r_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & r_m \end{bmatrix} \quad (2.2)$$

As ordens das matrizes dependem da quantidade de variáveis de estado e controle envolvidas. Por exemplo, se o sistema tem três variáveis de estado, $n = 3$, portanto, temos uma matriz 3x3 e devemos preencher a diagonal com q_1 , correspondente ao peso da primeira variável, com q_2 , correspondente ao peso da segunda, e assim sucessivamente. Repete-se o procedimento para a matriz R de acordo com a quantidade m de variáveis de controle.

As restrições em que o problema de otimização está sujeito, são: o modelo do sistema na Equação 2.1b, o estado inicial na Equação 2.1c, e as restrições para o estado e o sinal de

controle nas Equações 2.1d e 2.1e, respectivamente, as quais delimitam essas variáveis entre um valor mínimo e máximo.

Mais detalhes da solução do problema associado ao MPC podem ser vistos no Capítulo 4, incluindo os métodos de discretização, integração, e uma demonstração no Matlab para solução de duas abordagens do MPC para o robô de tração diferencial.

2.3.3 O uso do MPC na compensação de atrasos e perdas de pacotes na rede de comunicação

Pelo fato de gerar uma sequência de controle dentro do horizonte de predição, o MPC é também explorado para mitigar os efeitos da rede de comunicação, tais como o atraso e a perda de pacotes, quando o processo é controlado via rede. Liu *et al.* [45] utilizaram um MPC modificado para lidar com o atraso na comunicação, tanto nos canais de envio como nos de recepção. Para compensar o atraso aleatório da rede, o controlador é capaz de gerar uma sequência de controle dentro de um horizonte de predição, na qual é enviada em um único pacote para o sistema. Sem atraso de comunicação, apenas o primeiro sinal da sequência de controle é aplicado no sistema e os demais sinais dentro da predição são descartados. Quando a rede está sujeita a atraso na comunicação, o pacote enviado através do canal de envio (do controlador para o atuador) estará atrasado ou será perdido. Portanto, as entradas de controle da última sequência recebida são aplicadas no sistema, até que um novo pacote chegue. Para compensar o atraso no canal de retorno ou *feedback* (do sensor para o controlador), um preditor é usado para prever o estado atual do sistema \hat{x} .

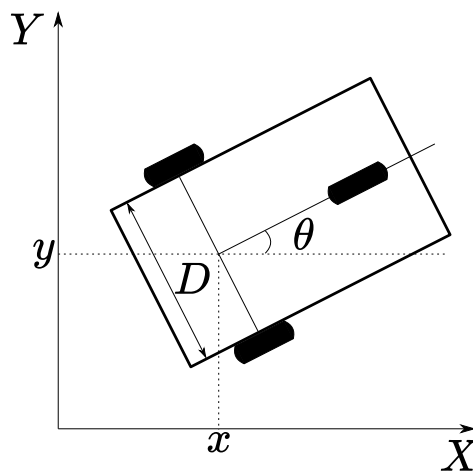
Da mesma forma, Findeisen e Varutti [26] utilizaram um controlador preditivo não-linear baseado em modelo para compensar o não determinismo da rede. Para compensar o atraso τ_{sc} do pacote de medição do sensor, o atraso é primeiramente estabelecido pela comparação do *timestamp* (contém informações de data e hora do envio do pacote) com o relógio interno do controlador. Em seguida, de posse do modelo do sistema, o estado atual da mesma pode ser estimado. A compensação do atraso no atuador é menos trivial, considerando que, dados os atrasos estocásticos no lado do atuador, não há garantia sobre a última entrada de controle aplicada ao sistema. Neste caso, supõe-se que o atraso τ_{ca} do controlador para o atuador é conhecido. A predição do estado pode agora ser calculada considerando τ_{sc} e τ_{ca} , e o

algoritmo de otimização do MPC é por fim executado. O *timestamp* do pacote de controle resultante do MPC é deslocado no tempo em τ_{ca} e enviado para o atuador. Este, é encarregado de aplicar a entrada de controle em que o *timestamp* corresponde ao seu relógio interno. Estes tipos de atuadores são chamados de atuadores inteligentes e têm a capacidade de armazenar sinais de controle futuros.

2.4 Robô móvel de tração diferencial

O robô móvel de tração diferencial é um robô móvel não holonômico, que possui uma roda em cada lateral traseira e um rodízio na frente, também conhecido como roda *caster*. A vista superior de um modelo genérico desse tipo de robô está apresentada na Figura 2.7. As rodas laterais se acoplam a motores, enquanto a roda *caster* tem o movimento livre. O robô pode rotacionar em relação ao eixo X conforme a rotação imposta ao seu chassi, a qual depende das velocidades angulares das rodas laterais [44].

Figura 2.7: Representação da vista superior de um robô móvel genérico com duas rodas e tração diferencial.



Fonte: Produzida pelo autor.

O estado do robô é definido pelo vetor $\mathbf{x} = [x \ y \ \theta]^T$, em que x e y são as coordenadas da posição do robô no espaço cartesiano e θ é o ângulo de orientação do robô em relação ao eixo x . O ângulo θ aumenta quando o robô gira no sentido anti-horário em relação ao eixo X e diminui quando gira no sentido horário. A entrada de controle aplicada ao robô é o vetor $\mathbf{u} = [v \ \omega]^T$, onde v é a velocidade linear e ω é a velocidade angular. O modelo cinemático

do robô no espaço de estados é então definido [74]:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) \rightarrow \begin{cases} \dot{x} = v \cos \theta \\ \dot{y} = v \sin \theta \\ \dot{\theta} = \omega \end{cases} \quad (2.3)$$

onde $f(\cdot)$ é uma função não linear dos estados e das entradas de controle que representa o movimento do robô. Da Equação 2.3 obtém-se o modelo cinemático discreto do robô:

$$\begin{aligned} x(k+1) &= x(k) + v(k) \cos(k)T \\ y(k+1) &= y(k) + y(k) \sin(k)T \\ \theta(k+1) &= \theta(k) + \omega(k)T \end{aligned} \quad (2.4)$$

onde T é o período de amostragem e k um índice que representa os intervalos de tempo, de forma que cada instante de tempo t é calculado por $t = kT$. O modelo discreto no espaço de estados é então representado da seguinte forma:

$$\mathbf{x}(k+1) = \mathbf{x}(k) + f(\mathbf{x}(k), \mathbf{u}(k))T \quad (2.5)$$

Os estados e as entradas de controle do robô podem estar sujeitos à restrições particulares. Por exemplo, o robô pode estar limitado a navegar em um determinado espaço, logo, o vetor de estado do robô deve estar sempre envolvido pelos limites \mathbf{x}_{min} e \mathbf{x}_{max} definido pelas coordenadas que limitam o espaço. Além disso, a entrada de controle pode ser restringida a permanecer numa faixa definida pelo limite mínimo \mathbf{u}_{min} e limite máximo \mathbf{u}_{max} . Se o robô é controlado por um MPC, essas restrições de estado e controle podem ser incluídas no algoritmo [74].

A partir do vetor do sinal de controle com a velocidade linear v e a velocidade angular w , as velocidades angulares ω_d e ω_e , que devem ser aplicadas respectivamente nas rodas esquerda e direita, são obtidas com a solução do seguinte sistema de equações [44]:

$$\begin{cases} v = \frac{r}{2}(\omega_d + \omega_e) \\ \omega = \frac{r}{2D}(\omega_d - \omega_e) \end{cases} \quad (2.6)$$

em que r é o raio das rodas e D é a distância entre elas.

2.5 ROS 2

O ROS 2 é uma plataforma de software para desenvolvimento de aplicações robóticas, também conhecido como um Kit de Desenvolvimento de Software (SDK – *Software Development Kit*) de robótica. O ROS 2 tem código aberto e distribuído sob a licença Apache 2.0, que concede aos usuários direitos amplos para modificar, aplicar e redistribuir o software, sem obrigação de contribuição recíproca [49].

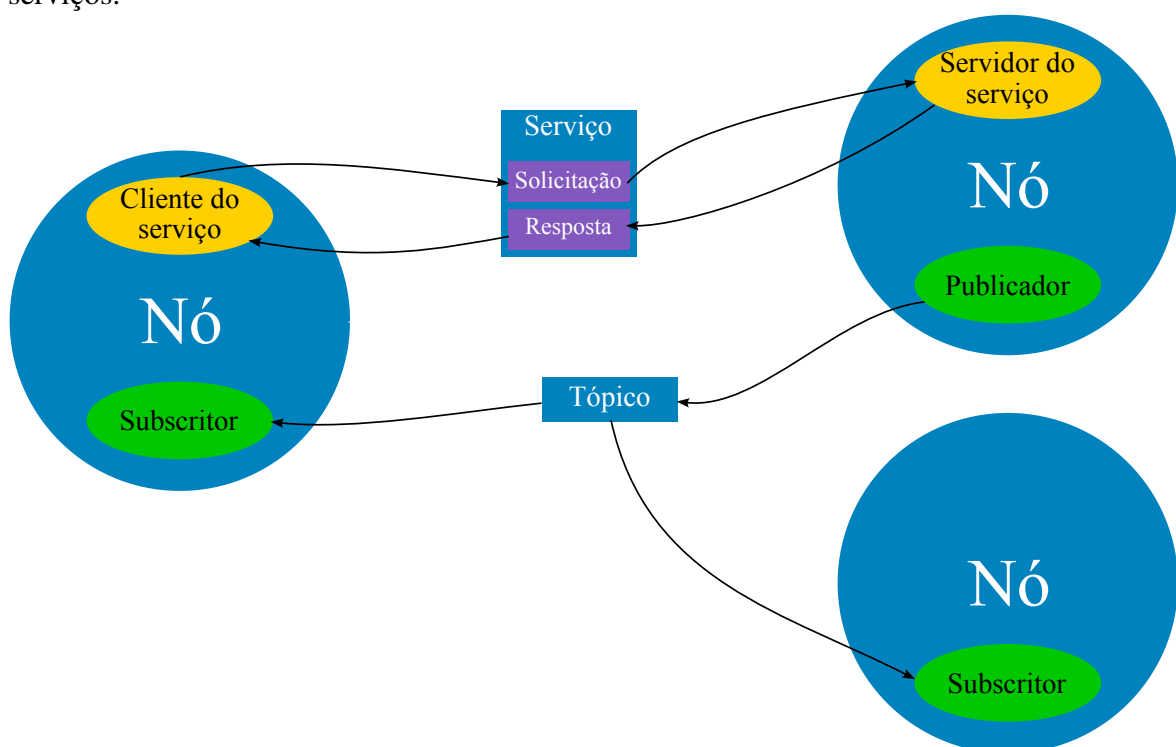
Diferentes aplicações de robótica, desde educação e pesquisa, até desenvolvimento e implantação de produtos, são suportadas pelo ROS 2. O sistema compreende um grande conjunto de componentes de software inter-relacionados que são comumente usados para desenvolver aplicações de robótica. O ecossistema de software é dividido em três categorias: (1) *middleware*, (2) algoritmos do ROS 2 e (3) ferramentas do desenvolvedor. O *middleware* engloba a comunicação entre os componentes, desde Interfaces de Programação de Aplicação (API – *Application Programming Interface*) até analisadores de mensagens. Algoritmos comumente usados em aplicações de robótica, como por exemplo, o SLAM, planejamento, e outros, são fornecidos pelo ROS 2. Inclui também ferramentas do desenvolvedor, como ferramentas gráficas e de linha de comando para configuração, inicialização, introspecção, visualização, depuração, simulação e registro [49].

O ROS 2 traz algumas melhorias em relação à sua primeira versão, o ROS 1. Uma delas é a adoção do DDS (*Data Distribution Service*), um padrão de comunicação aberto usado em infraestruturas críticas, tais como sistemas militares, espaciais e financeiros. O uso do DDS resolve alguns dos problemas ligados ao desenvolvimento de sistemas robóticos confiáveis e substitui o protocolo baseado no TCP/UDP do ROS 1. As versões também são diferentes na arquitetura da rede, onde o ROS 1 utiliza um servidor central (*roscore*), que dentre outras aplicações executa o ROS *Master*, para permitir que os nós individuais do ROS se localizem entre si. Já no ROS 2 a descoberta dos nós é feita ponto-a-ponto, sem a necessidade do servidor central. Outra diferença é que o ROS 1 pode ser executado apenas no Linux, enquanto o ROS 2 é multiplataforma e pode ser executado no Linux, Windows e macOS [49].

As aplicações do ROS 2 podem ser organizadas em pacotes, os quais podem possuir diversos nós. Cada nó pode ser alocado para um determinado propósito, como um nó para

o controlador dos motores do robô e outro nó para o SLAM, por exemplo. Os nós trocam informações através de tópicos, serviços, ações ou parâmetros. Na Figura 2.8 está a representação de um exemplo de aplicação com três nós, com troca de informações por tópicos e serviços. A comunicação por tópicos se dá através da subscrição e publicação de mensagens. O nó publicador publica a mensagem em um tópico, enquanto o subscritor se inscreve para receber a mensagem no mesmo tópico. É possível que mais de um nó publique ou se inscreva para receber em um mesmo tópico.

Figura 2.8: Representação de nós do ROS 2 com troca de informações através de tópicos e serviços.



Fonte: Adaptado de [71].

Os serviços são outro método de comunicação para os nós do ROS 2. São baseados em um modelo de chamada e resposta, diferente do modelo *publish/subscribe* dos tópicos. Enquanto os tópicos permitem que os nós se inscreva para receber fluxos de dados e obtenham atualizações contínuas, os serviços fornecem dados apenas quando são chamados por um cliente. Na representação da Figura 2.8, um nó é cliente de um serviço e está apto a fazer uma solicitação. O nó servidor desse serviço recebe a solicitação e retorna com uma resposta.

Os parâmetros são valores de configuração de um nó. Um nó pode ter parâmetros como

inteiros, flutuantes, booleanos, *strings* e listas. No ROS 2, cada nó mantém seus próprios parâmetros, os quais podem ser ajustados no momento de sua inicialização. Por fim, as ações destinam-se à tarefas de longa duração. Consistem em três partes: objetivo, *feedback* e resultado. Sua funcionalidade é semelhante a dos serviços, exceto que as ações são preemptivas (é possível cancelá-las durante a execução). As ações também fornecem *feedback* constante, ao contrário de serviços que retornam uma única resposta.

Outros recursos do ROS 2 podem ser vistos em sua documentação [71] e também no Capítulo 6, onde a execução da aplicação relacionada a este trabalho de tese está detalhada. A distribuição *Foxy Fitzroy* do ROS 2 foi a utilizada.

2.6 Gazebo

O Gazebo é um simulador de robótica de código aberto, onde ambientes de simulação, chamados de mundo, podem ser criados. Um mundo pode conter entidades dinâmicas, como os robôs, ou estáticas, representando diversos objetos do ambiente. O Gazebo pode ser integrado ao ROS 2 para interação com o ambiente de simulação. Por exemplo, comandos de velocidade para um robô móvel podem ser enviados ao Gazebo através de tópicos, enquanto o Gazebo também pode se utilizar dos tópicos para envio da posição do robô. Mais detalhes da utilização do Gazebo, incluindo a sua interação com o ROS 2, também podem ser vistos no Capítulo 6.

Neste trabalho foi a utilizada a versão 11 do Gazebo *Classic*, lançada em 2020 e com suporte até 2025. Versões mais recentes são oriundas de um projeto chamado *Ignition* e são chamadas apenas de Gazebo. Mais detalhes da ferramenta podem ser vistos na página do desenvolvedor [70].

2.7 Considerações finais

Neste capítulo foram apresentados alguns conceitos e tecnologias que são abordadas ao longo do trabalho. Dentre as tecnologias apresentadas, a computação na borda promove vários casos de uso, incluindo os da automação industrial. Na arquitetura proposta neste trabalho, a computação na borda é parte importante, dado que esse sistema computacio-

nal, provido de mais recursos computacionais em relação aos computadores embarcados nos AGVs, tem capacidade para planejar a trajetória de múltiplos veículos.

Para os AGVs foram apresentados o histórico, tipos e tecnologias de navegação. Na sequência foi apresentado o MPC, uma técnica de controle ótimo que realiza previsões em um horizonte a respeito do estado do sistema. O MPC é parte importante do trabalho e por isso no Capítulo 4 estão apresentados mais detalhes, incluindo a solução do problema com ferramentas computacionais.

Foi apresentado também o robô móvel de tração diferencial e o seu modelo cinemático. Este tipo de robô é utilizado para representar os AGVs nas simulações apresentadas neste trabalho. Por fim, foram apresentados o ROS 2, ferramenta utilizada para desenvolvimento de aplicações de robótica, e o Gazebo, utilizado neste trabalho para execução dos cenários de simulação. O ROS 2 foi uma ferramenta chave para a implementação da arquitetura apresentada no Capítulo 5. A partir do ROS 2 foi possível executar diversos nós em ambos os lados (borda e local), com troca de informações através de tópicos e uso do MPC como um serviço.

Capítulo 3

Revisão bibliográfica

A revisão bibliográfica apresentada neste capítulo está dividida em duas partes. Na primeira são explorados o controle e a coordenação dos veículos autoguiados, com foco no planejamento da rota e do movimento. Incluem-se nessa parte os sistemas centralizados e descentralizados, como também os que são assistidos por computação em nuvem ou na borda. A segunda parte generaliza o controle industrial no aspecto da sua execução como um serviço de nuvem. Os casos específicos com o MPC, tipo de controlador utilizado neste trabalho, também são abordados. Uma discussão a respeito dos trabalhos incluídos na revisão, abordando as suas similaridades e comparações com a proposta deste trabalho, é apresentada no final de cada uma das partes desta revisão.

3.1 Controle e coordenação de veículos autoguiados

Nas seções seguintes estão apresentados os principais algoritmos utilizados para o planejamento da rota e do movimento de AGVs ou robôs móveis em geral. Em seguida, os sistemas de controle de AGVs são divididos entre centralizados e descentralizados. Por fim, são apresentados os sistemas que têm assistência da computação em nuvem ou na borda.

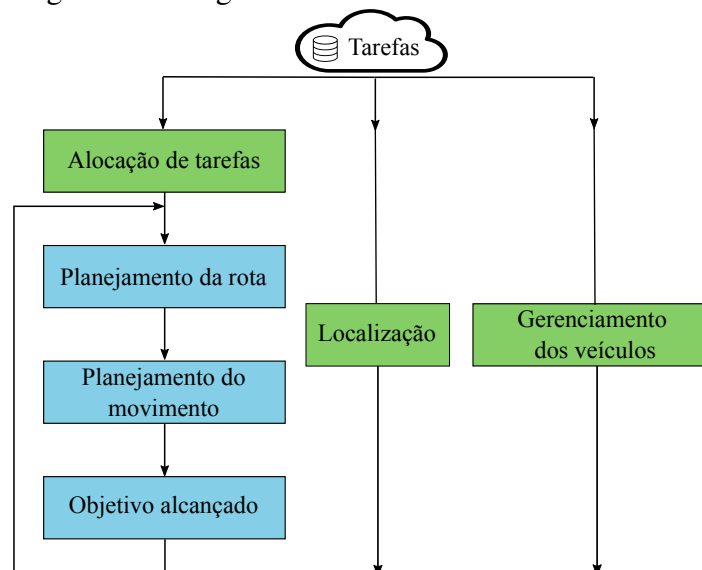
3.1.1 Planejamento da rota e do movimento

De Ryck *et al.* [22] consideram que um sistema completo de gerenciamento de AGVs pode ser decomposto em cinco atividades:

- alocação de tarefas: um conjunto de ordens a ser distribuído para a frota de AGVs deve ser alocado a um AGV específico de modo otimizado.
- planejamento da rota: uma vez que a tarefa foi alocada para um AGV, é necessário encontrar a rota mais curta para o destino final. Nesse caso, utiliza-se uma representação do ambiente para buscar uma sequência de segmentos que possa levar o AGV até o destino no menor tempo possível.
- localização: é necessário localizar o AGV para auxiliar na sua navegação.
- planejamento do movimento: planeja-se o movimento para evitar colisões dos AGVs com objetos ou pessoas, evitar *deadlocks* ou limitar a quantidade de AGVs em uma determinada área.
- gerenciamento dos veículos: monitoramento do estado dos AGVs, o que pode incluir o estado de carga da bateria, necessidade de manutenção, entre outros.

Na Figura 3.1 está ilustrado o fluxo dessas atividades, com adição do bloco de objetivo alcançado, onde é verificado se o AGV concluiu a tarefa e assim possa ser alocado para uma nova.

Figura 3.1: Diagrama do fluxo de trabalho dos AGVs.

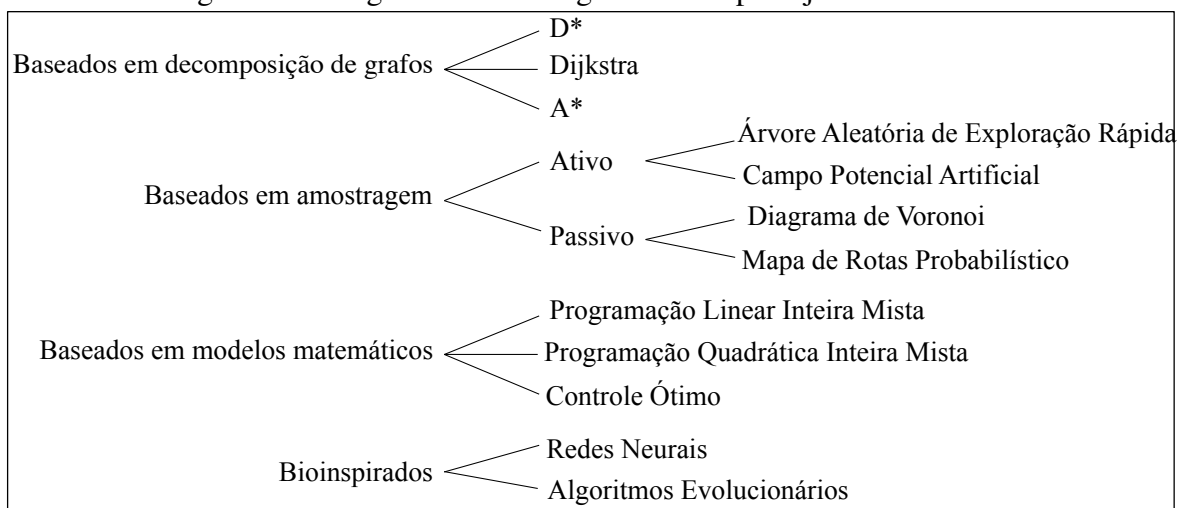


Fonte: Adaptado de [22].

Neste trabalho, apenas os blocos em azul da Figura 3.1 são explorados para o alcance do objetivo geral, o que também delimita esta revisão bibliográfica. Presume-se que as atividades foram alocadas para determinados AGVs e o sistema proposto aqui conduz o AGV para a conclusão de sua tarefa. A alocação de tarefas para AGVs é uma ampla área de pesquisa e é possível encontrar abordagens na literatura [22; 21].

Madridano *et al.* [101] distribuem os algoritmos de planejamento da rota em quatro categorias: baseados em decomposição de grafos, baseados em amostragem, baseados em modelos matemáticos, e bioinspirados. Os algoritmos incluídos em cada uma das categorias podem ser vistos no diagrama da Figura 3.2.

Figura 3.2: Diagrama com os algoritmos de planejamento da rota.



Fonte: Adaptado de [101].

Nos métodos baseados em decomposição de grafos, o ambiente é representado na forma de células que correspondem a obstáculos, espaços livres ou nós de início ou fim do caminho. A partir dessa decomposição, nós são estabelecidos nas células livres, as quais se conectam entre si através de arestas, formando uma estrutura conhecida como grafo. Dessa forma, os mapas do ambiente real são modelados como um conjunto de vértices (V) e arestas (A). A solução do problema de planejamento da rota é encontrar uma sequência de arestas consecutivas que conectam um nó de início a um nó final. Para encontrar uma solução otimizada, é preciso fazer uma varredura em todos os nós, calculando-se o custo a partir dos pesos dos nós e arestas que é necessário passar através. Dentre vários algoritmos, os mais utilizados são o Dijkstra, A^* e D^* [101].

Os métodos baseados em amostragem mapeiam o ambiente aleatoriamente na tentativa de encontrar um caminho entre dois pontos. Essa amostragem aleatória é normalmente conduzida na forma de nós ou células. A cada iteração, um novo ponto é acessado, e caso seja um espaço livre, é conectado a outras amostras próximas, de forma a criar conexões entre os nós que pertencem a espaços livres. Os métodos podem ser classificados em ativos ou passivos. No primeiro caso, os algoritmos por si só determinam o melhor caminho, enquanto no segundo, os algoritmos geram uma rede de caminhos para o alcance do destino, sendo necessário um algoritmo complementar para determinar o caminho ótimo. Alguns exemplos de algoritmos passivos são o Mapa de Rotas Probabilístico e o Diagrama de Voronoi, e de algoritmos ativos, são a Árvore Aleatória de Exploração Rápida e o Campo Potencial Artificial [101].

Métodos que são baseados em modelos matemáticos estabelecem restrições cinemáticas e dinâmicas para modelar o ambiente e o sistema. Busca-se obter uma solução ótima através da minimização de uma função custo, incluindo restrições de igualdade e desigualdade. Alguns exemplos são a Programação Linear Inteira Mista, a Programação Quadrática Inteira Mista e o Controle Ótimo. Neste último método, uma das técnicas mais populares é o MPC [101].

Os métodos bioinspirados buscam solucionar o problema de planejamento da rota com base nos comportamentos biológicos, imitando como os organismos vivos se comportam e agem, com o intuito de gerar um caminho ótimo. Os métodos não são totalmente determinísticos, apresentando estruturas paralelas e tornando-se adaptativos. Esses fatores fazem com que os métodos possam gerar soluções ótimas, sem precisar conhecer de maneira exaustiva o ambiente em que a missão está sendo conduzida. Alguns exemplos são os Algoritmos Evolucionários, os quais analisam o comportamento das espécies, e as Redes Neurais, que imitam as conexões e funcionamento dos neurônios para processar as informações. Dentro dos Algoritmos Evolucionários, alguns exemplos são: Algoritmo Genético, Otimização por Enxame de Partículas e Otimização por Colônia de Formigas. Para as Redes Neurais, exemplos são: Redes Neurais Profundas e Aprendizagem por Reforço [101].

Enquanto segue a rota planejada, o AGV pode se deparar com obstáculos, sejam eles obstáculos estáticos que não foram considerados pelo planejador, ou mesmo pessoas e outros AGVs que estejam navegando. Outra situação que pode ocorrer é o *deadlock*. Isso

acontece quando o AGV não tem ação e não se move para frente ou para trás. Portanto, a modificação da rota planejada para evitar colisões e *deadlocks* é denominada planejamento do movimento [22].

Para ambos os casos que envolvem a mudança da rota, existem abordagens centralizadas e descentralizadas. Quando centralizada, o computador central utiliza todas as informações disponíveis para prevenção de colisões, como as posições, objetivos e rotas previstas para encontrar uma solução global livre de colisões, o que pode resultar em uma trajetória totalmente diferente da anteriormente prevista, ou na redução da velocidade de um dos veículos, por exemplo. No modo descentralizado, o AGV reage com base no que se percebe localmente para evitar as colisões, podendo parar ou reduzir a velocidade até que o caminho esteja livre, ou também desviar do obstáculo. A prevenção de *deadlocks* centralizada pode ser incluída no processo de otimização de diferentes formas, uma delas é através das Redes de Petri. Na forma descentralizada os AGVs evitam os *deadlocks* de acordo com sua percepção local e existem regras que são estabelecidas para lidar com o problema. Outro tipo de controle utilizado para evitar colisões e *deadlocks* é o controle da zona, onde se limita a quantidade de AGVs em uma determinada área. Ao se deparar com zonas cheias, o AGV deve esperar ou replanejar sua rota [22].

Os termos planejamento global e planejamento local também são usados para se referir ao planejamento da rota e ao planejamento do movimento, respectivamente [18; 87]. Sun *et al.* [87] elencam alguns algoritmos para os dois tipos de planejamento. A maioria dos algoritmos de planejamento global (ou da rota) citados pelos autores, são os mesmos do diagrama da Figura 3.2. Para o planejamento local (ou do movimento), os algoritmos mencionados são: Campo Potencial Artificial, Recozimento Simulado, Lógica *Fuzzy*, Redes Neurais e Janela Dinâmica.

Planejamento acoplado e desacoplado

Mercy *et al.* [56] e Angelis *et al.* [19] dividem o planejamento da trajetória em abordagens acopladas e desacopladas. Um planejamento da trajetória acoplado soluciona um problema de otimização, tendo como resultado a trajetória otimizada e os correspondentes sinais de controle para conduzir o veículo ao longo da trajetória. A maioria dos métodos aplicam a abordagem desacoplada, dividindo o problema em planejamento da rota e em seguida o ras-

tratamento da trajetória. As técnicas para solucionar o problema do planejamento da rota são apresentadas na Figura 3.2. Para o rastreamento da trajetória, o MPC é comumente utilizado para conduzir o veículo na rota planejada, visto que esta técnica de controle pode levar em consideração as limitações do sistema. Embora as abordagens desacopladas geralmente têm problemas mais fáceis de resolver em relação às acopladas, elas retornam resultados subótimos, além de que podem ser impraticáveis pelo fato de que a cinemática dos veículos e as suas limitações não estão inclusas no problema de planejamento da rota. Portanto, os autores utilizam um algoritmo de controle ótimo para gerar a rota que conduz o veículo em um ambiente com obstáculos móveis. Junto com a rota, a solução do problema também retorna os sinais de controle para que o veículo possa trafegar na rota planejada.

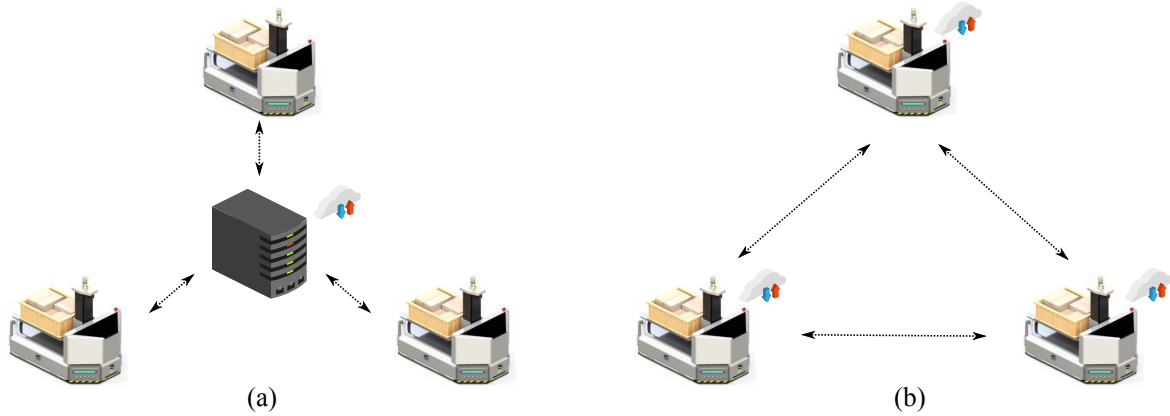
Seguindo o mesmo princípio, Filho *et al.* [55] apresentam um problema de controle ótimo para planejamento da trajetória, incluindo as restrições cinemáticas e dinâmicas dos robôs. As restrições cinemáticas estão diretamente relacionadas com a arquitetura do robô móvel, implicando em restrições tipicamente não holonômicas. As restrições dinâmicas vêm diretamente dos efeitos inerciais e interações entre diferentes corpos em contato. Os autores implementam um planejamento do movimento denominado Planejamento do Movimento com Horizonte Retrocedente (DRHMP – *Distributed Receding Horizon Motion Planning*), o que é similar a um MPC clássico, mas diferente pelo fato de que não apenas o primeiro valor da solução otimizada é aplicado na entrada do sistema. As trajetórias geradas pelo DRHMP incluem a cinemática do robô, mas não a dinâmica. Numa segunda etapa, os autores então utilizam o MPC para rastrear os robôs nas trajetórias planejadas, incluindo agora as restrições dinâmicas.

3.1.2 Sistemas centralizados e descentralizados

Os sistemas de controle de AGVs podem ser centralizados ou descentralizados. No primeiro caso, um único computador controla toda a frota de AGVs, implementando otimizações a partir das informações globais. A unidade central tem acesso às posições dos veículos e às informações das tarefas a serem executadas. Estes dados são usados pela entidade central para que as atividades sejam alocadas aos robôs e o caminho para conclusão das tarefas seja planejado de forma otimizada, evitando colisões com outros AGVs e *deadlocks* [23]. No controle descentralizado, a inteligência é distribuída aos AGVs para que operem de modo

independente na busca de alcançar os objetivos globais [22]. Nas Figuras 3.3(a) e 3.3(b) estão ilustradas as representações da arquitetura centralizada e descentralizada, respectivamente.

Figura 3.3: Representação da arquitetura de controle (a) centralizada e (b) descentralizada.



Fonte: Adaptado de [22].

Como exemplo de controle centralizado, Jose e Pratihar [37] propõem um sistema centralizado multi-robôs com alocação de tarefas e planejamento de rotas livre de colisões, para inspeção de uma planta industrial. O problema de alocação de tarefas é solucionado com o Algoritmo Genético, ao passo que o planejamento da rota é feito com o algoritmo A*. Guney e Raptis [31] apresentam um sistema centralizado de coordenação de AGVs, o qual utiliza uma lógica de prioridade dinâmica para resolver problemas de conflitos na mobilidade dos veículos. O foco do trabalho é na solução do problema do planejamento do movimento de multi-robôs em espaços confinados.

Draganjac *et al.* [25] destacam que apesar dos avanços significativos no campo dos veículos rodoviários autônomos e inteligentes, os AGVs industriais ainda contam com controladores centralizados e caminhos predefinidos, o que dificulta a escalabilidade e a flexibilidade quando é preciso fazer alterações no leiaute. Além disso, forçar os veículos a se moverem em rotas predeterminadas restringe a mobilidade e a autonomia, além de aumentar os custos de implantação e manutenção, já que um novo caminho deverá ser implantado sempre que houver alterações no leiaute. A solução para o problema é baseada na hipótese de que entregando total autonomia aos veículos, dando-lhes capacidade de planejar suas rotas livres de colisões e negociar situações de conflito, irá reduzir de forma significativa a complexidade computacional e consequentemente aumentar a escalabilidade e a flexibilidade do sistema.

Portanto, é apresentado um método que combina algoritmos para coordenação do movimento descentralizada e o planejamento da rota baseada em treliças. Qualquer situação de conflito é resolvida por negociação direta entre os veículos envolvidos, de modo a evitar *deadlocks*.

Chen *et al.* [17] apresentam um sistema descentralizado de alocação de tarefas e planejamento da rota multi-robô, em que a transição de cada robô é modelada com Cadeia de Markov. Realiza simulações com o ROS e experimentos com o robô *Turtlebot*. Nas simulações, o planejador de alto nível envia comandos para um controlador MPC de baixo nível, enquanto nos experimentos com o robô, o controlador de baixo nível é um PID.

Aplicado a quadrirotores, Zhu *et al.* [100] fizeram a junção das Redes Neurais Recorrentes (RNN – *Recurrent Neural Networks*) com o MPC para o planejamento do movimento multi-robôs de modo descentralizado em ambientes dinâmicos. Ao navegar em um espaço compartilhado, cada robô precisa das previsões do movimento dos robôs que estejam próximos para evitar colisões e tornar a navegação segura. Essas previsões do movimento podem ser obtidas entre os robôs através do compartilhamento das trajetórias futuras planejadas. Entretanto, a comunicação entre os robôs pode ser não confiável na prática. Nesse sentido, os autores propõem a combinação da previsão da trajetória dos obstáculos feita pela RNN, com o MPC, para planejar o movimento multi-robô de modo descentralizado. O MPC é utilizado em duas etapas, sendo a primeira para o treinamento da RNN, e a segunda para o planejamento do movimento dos robôs. Na etapa de treinamento, aplica-se um conjunto de dados que consiste de trajetórias geradas por um simulador multi-robô. O simulador utiliza um MPC centralizado para planejamento do movimento, onde a comunicação entre robôs é implementada para o compartilhamento das trajetórias e prevenção de colisões. Ao fazer o treinamento utilizando o conjunto de dados, o modelo imita o MPC centralizado e assim pode fazer a previsão do comportamento dos outros robôs. Enquanto os robôs estão navegando, a previsão da trajetória junto com o MPC é utilizada para o planejamento do movimento descentralizado. A previsão do movimento de pedestres também é incluída.

A abordagem de Rivière *et al.* [69] combina o planejamento centralizado, na tentativa de evitar o mínimo local, com a vantagem do controle descentralizado, no aspecto da escalabilidade e distribuição da computação. A abordagem é chamada de GLAS (*Global-to-Local Safe Autonomy Synthesis*) e é dividida em três etapas. Primeiro, são geradas trajetórias aleatórias multi-robô com um planejador global. Segundo, um modelo de observação local é

aplicado para gerar um conjunto de dados com pares de observação e ação. Terceiro, a imitação por aprendizagem profunda é utilizada para gerar uma política local, a qual imita a política global na prevenção do mínimo local.

Oyekanlu *et al.* [62] abordam os potenciais benefícios ao utilizar um servidor MEC e rede 5G, seja para um controle centralizado ou descentralizado. No modo centralizado, a central de controle mantém o estado de toda a frota e determina a rota que cada AGV deve percorrer, evitando colisões ou *deadlocks*. No modo descentralizado, os AGVs se comunicam e arbitram as tarefas entre si utilizando as antenas distribuídas da rede 5G. Com um servidor MEC e a rede 5G, os AGVs se comunicam e trocam dados com o alocador de tarefas, estações de trabalho e outras máquinas para minimizar o tempo de transporte ao utilizar o AGV mais próximo para executar uma tarefa. Em situações de *deadlock*, os AGVs trocam dados entre si para solucionar o problema caso o controle seja descentralizado. Se centralizado, os AGVs trocam dados com o controlador central para solucionar o conflito.

3.1.3 Sistemas assistidos por computação em nuvem ou na borda

Com a popularização da robótica e a expansão das áreas de aplicação, as tarefas a serem executadas pelos robôs estão tornando-se mais complexas, o que vai na oposição de seus recursos computacionais limitados. O conceito da robótica em nuvem tira dos robôs a necessidade de terem grande capacidade computacional e memória, mas os conectam à nuvem para execução de serviços sempre que necessário, além de reduzir seus custos [97]. Alguns trabalhos precursores da robótica em nuvem são o DAVinCi [7], *RoboEarth* [94], C²TAM [67] e Rapyuta [58]. O DAVinCi faz proveito da escalabilidade e paralelismo da nuvem para acelerar o algoritmo do SLAM com um *cluster Hadoop*. O *RoboEarth* é uma plataforma de código aberto no qual permite que os robôs conectados a uma rede possam gerar, compartilhar e reutilizar dados. O C²TAM transfere para a nuvem parte do algoritmo do SLAM, operando de modo cooperativo, onde os dados enviados pelos robôs são utilizados para formação de um único mapa do ambiente. O Rapyuta é uma plataforma de robótica em nuvem de código aberto baseado no *RoboEarth* e oferece um ambiente seguro e customizável para a transferência da computação para a nuvem.

Voltado à logística industrial, Cardarelli *et al.* [12] apresentam uma arquitetura de robótica em nuvem que dão suporte na coordenação de AGVs. O sistema de controle inclui

as seguintes funcionalidades: navegação global e planejamento da rota local, gerenciador de missões, sistema de percepção e visão global. A navegação global utiliza dados globais para controlar a navegação dos AGVs, entretanto, os veículos seguem rotas fixas e um planejamento da rota local é utilizado para desviar dos obstáculos. O gerenciador de missões aloca tarefas e planeja a rota dos AGVs. O sistema de percepção inclui não apenas sensores instalados nos AGVs para detecção de obstáculos, mas também um sistema fixo de percepção a laser, usado para monitorar interseções e pontos cegos, detectando, rastreando e classificando obstáculos. É incluído ainda na percepção um sistema semi-autônomo de exploração da planta, similar a um AGV, mas equipado com múltiplos escaneadores a laser capazes de processar o mapa em 3D, representando o leiaute geométrico da planta. Visto que as informações de obstáculos são disponibilizadas por várias fontes, computadores com desempenho suficiente para processamento desses dados precisariam ser instalados nos AGVs, o que aumentaria seus custos. Assim, um sistema de visão global centralizado coleta os dados de todos os sensores e os combinam para formar uma única e completa representação para ser compartilhada com os AGVs. Caso a comunicação com a central esteja deteriorada ou indisponível, a detecção de obstáculos é realizada unicamente pelos sensores do AGV, forçando o veículo a parar e aguardar a remoção do obstáculo.

Abbenseth *et al.* [2] citam duas principais desvantagens ao equipar robôs móveis com computadores de alto desempenho e sistemas de sensores: a) os robôs dependem de um mapa previamente conhecido, logo, atualizações no ambiente são percebidas apenas pelos sensores do robô, tornando as soluções de navegação limitadas a um horizonte reduzido; b) o custo no aumento da quantidade de robôs torna-se alto. Os autores implementam o conceito da robótica em nuvem para contornar essas desvantagens, através da centralização dos recursos computacionais. Essa abordagem então permite: a) a fusão dos dados de sensores em uma escala global, de tal forma que os obstáculos vistos por um agente móvel ou um sensor estacionário, são mesclados em um modelo de ambiente global e são comunicados aos agentes através de um mapa atualizado; b) a centralização da carga computacional e o uso de dados de sensores estacionários resultam na redução de custo dos AGVs. A rota é planejada na nuvem com um algoritmo baseado em grafos. Localmente, os robôs rastreiam a rota planejada e executam um planejamento de rota local reativo para desvio de obstáculos. Os robôs executam o SLAM e enviam atualizações do mapa para a nuvem. Lopez *et al.* [46]

utilizaram a mesma arquitetura em nuvem do trabalho de Abbenseth *et al.* [2], entretanto, o planejador local para prevenção de colisões é executado em três etapas. Um planejador denominado *elastic-band* procura um corredor livre para a passagem do robô e fornece os dados a um planejador de trajetórias baseado no MPC. A trajetória enfim planejada, levando em consideração a cinemática do robô e restrições cinodinâmicas, são repassadas a um controlador para rastreamento da trajetória, também do tipo MPC. As trajetórias são compartilhadas através da nuvem para a prevenção de colisões.

Considerando as degradações no sinal de rádio que podem acontecer no ambiente industrial, Nakimuli *et al.* [59] avaliaram o desempenho do controle de um AGV com computação na borda e rede 5G, na ocorrência de atrasos e perdas de pacotes. O controlador aplicado na borda é um PID. Inicialmente os autores fazem comparações com o controle realizado por meio das redes 4G e 5G. Na utilização da rede 4G, o desvio em relação à rota delimitada por uma fita magnética é maior, quando comparado com a rede 5G. Além disso, com a rede 4G o consumo de corrente também é maior (aumentando o uso de bateria), já que o controlador deve fazer constantes correções na velocidade angular para ajustar o AGV na rota. Posteriormente os autores induzem atrasos e perdas de pacotes na rede 5G. No primeiro caso, o atraso é variado de 0 a 250 milissegundos, com passos de 50 milissegundos, e observa-se que o desvio em relação à rota fixada aumenta em função do atraso. O AGV não completa o percurso quando o atraso está em 250 milissegundos. No segundo caso, a taxa de perda de pacotes é variada de 5 a 40%, com passos de 5%, e da mesma forma o desvio aumenta em função da perda de pacotes. Os autores observam que as taxas entre 10 e 30% têm um impacto severo nas correções, e que as perdas acima de 30% resultam em um desempenho inaceitável do controle. Utilizando a mesma infraestrutura de laboratório do trabalho de Nakimuli *et al.* [59] e uma arquitetura similar, Vakarak *et al.* [90] propõem a utilização da Aprendizagem Profunda para detectar o mau funcionamento do AGV num cenário da Indústria 4.0 com redes 5G. A detecção do problema é feita a partir das informações do tráfego da rede, sem a necessidade de coletar dados do AGV ou do CLP virtual aplicado no servidor de borda para controle do AGV. O treinamento do modelo é feito a partir das informações da rede e do desvio do AGV em relação à rota definida, visto que o AGV segue uma rota fixa delimitada por uma fita magnética. Durante o treinamento, algumas degradações são impostas na rede, como atraso, *jitter*, corrupção de pacotes e perda de pacotes. A ideia geral

do trabalho é prever o mau funcionamento do AGV quando perturbações forem detectadas na rede, de modo que ações corretivas sejam tomadas, como por exemplo, parar o AGV.

Envolvendo ainda o uso do 5G no ambiente industrial, Rodriguez *et al.* [73] exploram os desafios da integração das redes sem fio 5G com os sistemas industriais e apresenta uma estrutura para o alcance de uma integração estruturada e com sucesso. A aplicação da estrutura é apresentada a partir da integração do 5G com um caso de uso industrial: robôs móveis autônomos. Os resultados indicam que a tecnologia 5G pode ser usada para um gerenciamento confiável da frota em cenários industriais e pode suportar a migração do planejamento de rota local para o servidor de borda.

Lambrecht *et al.* [42] apresentam um estudo de caso que utiliza uma rede 4G junto com a computação na borda, para o controle de sistemas de transporte autônomos em um ambiente industrial. O SLAM e os planejamentos global e local são transferidos para o servidor de borda, enquanto o robô executa localmente apenas funções básicas de segurança. Nos testes, o robô navega em uma velocidade de 1 metro por segundo e o desempenho do controle é avaliado conforme a latência. Os autores fazem testes utilizando um servidor de borda e uma nuvem pública, e não recomendam a execução dos algoritmos de controle nessas plataformas caso a latência seja maior que 150 milissegundos. No mesmo cenário, Lambrecht e Funk [41] compararam o consumo de energia nas situações de execução dos serviços no computador acoplado ao robô e transferência dos serviços para o servidor de borda. Os autores concluem que no caso da transferência para o servidor de borda, o processamento local reduz de tal modo que o computador do robô pode ser substituído por um computador de placa única e baixo custo. Consequentemente, o consumo de energia é reduzido. O trabalho também inclui o uso de imagens de câmeras instaladas no robô, para reconhecimento de objetos com rede neural convolucional (CNN – *Convolutional Neural Networks*). O servidor de borda tem GPU (*Graphics Processing Unit*) e pode processar os algoritmos da CNN com maior desempenho, o que compensa, inclusive, a latência da rede na transferência das imagens, quando comparado ao processamento local desses algoritmos.

Também voltado ao cenário industrial, Harjuhahto *et al.* [32] realizaram simulações com um sistema híbrido, que inclui uma área virtual de estoque de uma fábrica com AGVs virtuais e um *cluster* real para processamento dos dados gerados pelos sensores LiDAR dos AGVs. O *cluster* atua como um nó de computação na borda (no trabalho, utiliza-se o termo *Fog*

Computing) próximo ao estoque. Os AGVs se conectam ao servidor de borda usando uma conexão TSN com rede 5G. Considerando que a planta tem paredes que formam áreas físicas diferentes e assim limitam a atuação do LiDAR, uma planta grande pode ter múltiplos nós de borda, cada um servindo uma ou mais de uma área distinta. Além de processar o LiDAR, os nós mantêm o estado de um *grid* de ocupação, o qual representa uma área dividida em células onde os estados de cada uma delas pode ser: ‘livre’, ‘desconhecido’, ‘ocupado’ ou ‘veículo’. Em caso de falhas no nó, os AGVs podem se conectar a outro nó.

Chen *et al.* [16] analisaram a sinergia entre a visão e a comunicação em uma aplicação com veículos aéreos não tripulados (UAVs – *Unmanned Aerial Vehicles*) assistidos por computação na borda. Em aplicações visuais com o UAV, como de vigilância ou monitoramento, por exemplo, a câmera é o item principal para realização da tarefa, como também fornece as imagens para a navegação autônoma com prevenção de colisões. Um voo em alta velocidade requer uma alta taxa de atualização das informações, impondo um desafio para o computador embarcado no UAV. Entretanto, na era 5G, as atividades que seriam processadas localmente podem ser transferidas para um servidor MEC. As informações do ambiente explorado podem ainda serem aproveitadas para aprimorar o desempenho da comunicação, de tal maneira que, através do mapa 3D, tem-se conhecimento a respeito dos obstáculos entre a estação base e o UAV que influenciam na propagação do sinal. O mapa 3D pode ser gerado através de algoritmos de processamento de vídeo, como o SLAM. Nesse contexto, a ideia principal do trabalho é aproveitar a imagem utilizada na construção do mapa 3D e fazer a previsão da comunicação sem fio ao redor do UAV. Assim, o UAV planeja sua trajetória em tempo real, tentando sempre manter a linha de visão com a estação base da rede de comunicação sem fio. O UAV pode, portanto, transferir as atividades computacionais para o servidor de borda para acelerar o processamento do vídeo e aquisição do mapa. Durante o processo, a percepção visual e a comunicação interagem e promovem uma à outra. Primeiro, o vídeo é transferido do UAV para o servidor de borda para processamento do SLAM. O mapa e a localização são devolvidos para o UAV, o qual utilizará esse mapa para construção de um mapa de rádio. A partir desse mapa, a trajetória é planejada com o MPC, sempre na tentativa de manter a linha de visão com a estação base e desviar dos obstáculos.

Groshev *et al.* [28] inicialmente analisaram o cenário atual da computação na borda no contexto de sistemas robóticos, com foco particular em arquiteturas de padrões e comitês

industriais. Em seguida, propõem uma arquitetura de robótica para o ambiente de borda onde informações a respeito do sinal de rádio estão disponíveis. A arquitetura é composta por três subsistemas: (i) o sistema robótico, (ii) o sistema de computação na borda, e (iii) o sistema de orquestração. O sistema robótico consiste de módulos distribuídos no robô e na borda. No robô estão embarcados apenas componentes básicos, como sensores e atuadores. O cérebro do robô, como é chamado no trabalho, se encontra no servidor de borda e é responsável pela coordenação e navegação dos robôs. O sistema da computação na borda integra as aplicações, além de executar outros serviços como informações de rádio e localização. Os movimentos do robô são coordenados e controlados considerando-se essas informações da rede e da localização. Por exemplo, baseado na intensidade do sinal da rede e em outros parâmetros, o cérebro do robô pode detectar a degradação no sinal de rede sem fio e adaptar a velocidade do robô. A arquitetura também prevê a comunicação direta entre os robôs (D2D – *Device-to-device*), a qual pode ser utilizada para coordenação dos robôs no instante em que o sinal da rede estiver degradado. Os autores fazem uma implementação da arquitetura com dois robôs móveis e por fim formulam os principais desafios e possibilidades de pesquisa identificados.

3.1.4 Discussão

Dos algoritmos apresentados na Figura 3.2, com exceção dos baseados em modelos matemáticos, as outras possibilidades não consideram as restrições cinemáticas e dinâmicas do robô [101], o que pode resultar em rotas impraticáveis a serem seguidas, dado que as limitações do veículo não são consideradas na fase de planejamento da rota [56]. As técnicas de controle ótimo, por sua vez, incluem as restrições cinemáticas e dinâmicas, funcionam como uma abordagem acoplada, além de que o problema das colisões pode ser incluído através de restrições no processo de otimização [56; 19]. Filho *et al.* [55], por exemplo, utilizam o controle ótimo para planejamento da trajetória livre de colisões, incluindo os modelos cinemáticos e dinâmicos, como também utilizam o MPC para rastrear os robôs na trajetória planejada na etapa anterior.

Os termos “planejamento da rota” e “planejamento da trajetória” foram mencionados nesta revisão sem nenhuma distinção. Entretanto, o planejamento da rota tem como resultado uma rota que é independente do tempo, ou seja, um conjunto de pontos que formam a rota

são distribuídos no espaço, mas sem associação com o tempo de alcance de cada um dos pontos [8]. O planejamento da trajetória tem relação com o tempo, portanto, estima-se o tempo que o robô deve alcançar cada ponto da trajetória planejada. O MPC se enquadra nesse caso, visto que os pontos da trajetória são previstos no horizonte de predição e separados no tempo de acordo com o intervalo ajustado no projeto do controlador.

Com relação à centralização e descentralização do controle e coordenação, muitas vezes os autores defendem a forma descentralizada, tal como De Ryck *et al.* [22], de forma a atender aos requerimentos futuros como a flexibilidade, robustez e escalabilidade. Entretanto, não consideram a centralização na borda ou na nuvem, considerando que essas plataformas podem até ser mais confiáveis para a execução das atividades que darão suporte na navegação, têm mais recursos computacionais, além de que podem contribuir na redução de custo dos AGVs e redução do consumo de energia, dado que o computador acoplado aos veículos pode ser mais simples nesse caso, conforme avaliado por Lambrecht e Funk [41]. Algumas abordagens são descentralizadas e tentam imitar um comportamento centralizado, como visto nos trabalhos de Zhu *et al.* [100] e Rivière *et al.* [69]. Os dados globais de um planejador de movimento central são utilizados para a criação de modelos de rede neural, que, posteriormente, de forma descentralizada, são aplicados para que os quadrirotores imitem o comportamento global.

Quando os sistemas de controle de robôs móveis são assistidos por computação em nuvem ou na borda, Cardarelli *et al.* [12] apresentam uma arquitetura de robótica em nuvem que utiliza dados globais para planejamento da navegação, entretanto, os AGVs são limitados a navegarem em rotas fixas. Matos *et al.* [51] apresentam um sistema de coordenação de múltiplos AGVs tolerante a falhas de comunicação, mas os AGVs também trafegam em rotas fixas e as falhas de comunicação não são detalhadas. Lopez *et al.* [46] propõem uma arquitetura em nuvem com a prevenção de colisões resolvida localmente entre os robôs, com o compartilhamento das trajetórias previstas realizada na nuvem, o que pode causar colisões em caso de problemas na comunicação. Nakimuli *et al.* [59] avaliaram o controle do AGV com computação na borda e rede 5G, sob diversas condições de atrasos e perdas de pacotes, entretanto, apenas analisam o desempenho do controle e não aplicam nenhuma estratégia para tratar dos problemas decorrentes dos efeitos da rede. Vakaruk *et al.* [90] utilizaram inteligência artificial para detectar um mau funcionamento do AGV a partir das informa-

ções da rede, contudo, não utilizam nenhuma estratégia para lidar com a degradação na rede. Lambrecht e Funk [41] fizeram a transferência de atividades de planejamento e SLAM para o servidor de borda, utilizando uma rede 4G, mas apenas avaliam o desempenho do controle e a redução do consumo de energia no robô quando o processamento das atividades é transferido para o servidor de borda, além de que não utilizam múltiplos robôs. Chen *et al.* [16] planejaram a rota de um UAV, de tal modo que o robô não perca a linha de visão com a estação base vinculada a um servidor de borda, e assim não trafegue em locais onde a qualidade da comunicação possa ser comprometida. Essa abordagem não é viável para aplicação em AGVs industriais, pelo fato de que uma trajetória planejada para manter uma boa qualidade na comunicação pode não ser a trajetória mais curta, e, conseqüentemente, levará mais tempo para ser concluída. Groshev *et al.* [28] apresentam uma arquitetura de borda com vários recursos, incluindo um serviço que fornece informações da qualidade do sinal de rádio da rede e um sistema de orquestração. O cérebro do robô, localizado no servidor de borda, é responsável pela coordenação e navegação dos robôs. Os autores, entretanto, não apresentam soluções para essa parte da arquitetura.

Como referências para a construção da arquitetura proposta no Capítulo 5, têm-se as abordagens de Mercy *et al.* [56] e Angelis *et al.* [19], na adoção do MPC para planejamento da trajetória com prevenção de colisões, como também a proposta de Lopez *et al.* [46], na adoção de um planejador global na nuvem e o MPC para o planejamento da trajetória livre de colisões. Inclui-se também a abordagem de Filho *et al.* [55], na qual um planejador com horizonte retrocedente é utilizado, similar ao MPC, junto ao MPC para rastreamento da trajetória.

3.2 O controlador industrial como um serviço de nuvem ou borda

Dentre os vários serviços que podem ser oferecidos pela computação em nuvem, Hegazy e Hefeeda [33] consideram a automação industrial como um serviço a ser oferecido. Ao virtualizar os controladores, os autores destacam que é possível alcançar significativas reduções de custo e de tempo. Quando as funcionalidades de automação são oferecidas a partir da nuvem, a internet pode colocar o processo controlado em risco, devido aos possíveis atrasos

na comunicação e falhas nas máquinas virtuais ou *links*. Portanto, os autores propõem um compensador de atrasos e um algoritmo para mitigar as falhas.

Abdelzaher *et al.* [3] consideram que as aplicações que têm suporte de inteligência na borda, fazem parte de uma categoria de aplicações em nuvem que estão em ascensão. Essas aplicações são impulsionadas pela confluência de três tendências globais: proliferação de dispositivos embarcados conectados; a crescente demanda por soluções inteligentes de detecção e controle para várias aplicações, que vão desde a automação residencial ao controle industrial; e a promessa da comunicação sem fio com velocidade de rede local, oferecida pelas emergentes infraestruturas sem fio das redes 5G, quebrando barreiras de comunicação e promovendo a transferência da computação para a nuvem. Os autores apontam no trabalho cinco desafios de pesquisa que surgem nas aplicações com inteligência e controle transferidos para a nuvem: (i) aprendizagem como serviço, (ii) garantia da qualidade do sensoramento, (iii) transferência da otimização e controle (*offloading*), (iv) garantia no fechamento da malha e (v) execução colaborativa.

A troca de controladores baseados em hardware, por instâncias de software, é denominada Funções Virtuais de Controle de Processos (VPFs – *Virtual Process Control Functions*) por Zhao e Dán [99]. As VPFs podem ser colocadas em servidores, podem ser provisionadas com flexibilidade sob demanda e são mais fáceis de atualizar em relação ao hardware de controladores legados. Junto com as redes 5G, as indústrias podem reduzir o custo operacional e flexibilizar o controle de processos utilizando a computação na borda para instalação das VPFs. No trabalho, os autores propõem uma estratégia para manter a resiliência das VPFs executadas no servidor de borda.

Kaneko *et al.* [38] apresentam uma arquitetura de sistema de controle industrial baseada em microsserviços utilizando nuvem e um servidor MEC. Os autores atribuem a dificuldade de se realizar o controle em tempo real nos sistemas de controle industriais baseados em nuvem, à imprevisibilidade da latência entre a nuvem pública e os dispositivos industriais. Em contrapartida, os sistemas de controle industriais executados no servidor MEC podem utilizar a baixa latência das redes 5G para realizar o controle em tempo real. Para reduzir o custo de operação do sistema de controle, a proposta da arquitetura apresentada é executar no servidor de borda apenas os serviços em tempo real, enquanto os demais serviços são executados na nuvem. A abordagem de microsserviços é então utilizada com execução dos

serviços em contêineres. Dado que um sistema de controle industrial é composto de um grande número de microsserviços, um algoritmo determina onde o serviço será executado, considerando o custo de operação e os requerimentos de processamento em tempo real.

Inaltekin *et al.* [35] investigaram as características da latência e da confiabilidade quando um controlador virtual é alocado em diferentes níveis. Nós de borda podem ter uma baixa latência na resposta, mas nem sempre são confiáveis. Por outro lado, nós remotos de computação em nuvem têm tempos de resposta mais longos, entretanto, são projetados para terem alta confiabilidade. Os autores então analisam a interação entre latência e confiabilidade, na intenção de fazer uma escolha ótima com relação à localização do controlador. Em testes realizados com um UAV, observa-se que a eficiência no rastreamento do caminho, decai mais rápido com o aumento da latência do que com a redução da confiabilidade, o que sugere a alocação do software de controle o mais próximo possível do UAV.

Ma *et al.* [48] propõem uma abordagem de comutação do controle em múltiplas camadas, de modo a otimizar o tempo de execução do controle em uma plataforma de computação de duas camadas. O controle é comutado dinamicamente entre a plataforma local e da borda, de acordo com as mudanças na confiabilidade da rede, buscando sempre manter a estabilidade. A abordagem proposta formula a seleção da plataforma como um problema de classificação baseado em dados, no qual pode ser resolvido a partir de modelos extraídos de simulações. O controlador local se refere ao controlador que está diretamente ligado aos sensores e atuadores por uma conexão confiável, exercendo o papel de um controlador de segurança e aplicando uma lei de controle simples. O controlador da borda é referido como um controlador de maior desempenho, aplicando uma lei de controle mais sofisticada, mas podendo ter perda de dados na rede sem fio.

3.2.1 Casos específicos com o MPC

Nos casos em que o controlador e a planta estão conectados por uma rede de comunicação, a adoção do MPC como controlador para compensar o atraso ou perda de pacotes em redes de comunicação já é uma área explorada. Por se tratar de uma estratégia de controle que realiza previsões dentro de um horizonte finito, o MPC pode ser utilizado para minimizar tais efeitos. A cada estado amostrado, a sequência de sinais de controle gerada pelo MPC é enviada à planta. Durante a operação normal, apenas o primeiro sinal é aplicado e o restante

é descartado. Em caso de atrasos ou perdas de pacotes, os sinais de controle dentro da última sequência recebida são utilizados até que um novo pacote com uma sequência mais recente seja recebido [45; 29].

Seguindo o conceito do controlador como um serviço de nuvem, Vick *et al.* [93] apresentam uma arquitetura de controle com compensação de atrasos da rede de comunicação, onde o MPC é executado na nuvem para controlar um manipulador robótico. Omena *et al.* [60] apresentam uma abordagem de compensação dos efeitos da rede de comunicação no controle industrial, incluindo o atraso e a perda de pacotes, utilizando o MPC executado no servidor de borda. A validação é realizada com um robô móvel de tração diferencial, onde o lado da borda executa o MPC e envia para o lado local toda a sequência de controle gerada no horizonte de predição. No caso de atraso ou perda de pacotes, o atuador no lado local avança na última sequência de controle até que uma nova seja recebida. A validação do sistema proposto é realizada em duas etapas. Na primeira, o robô deve estabilizar em uma referência determinada. Na segunda, a referência varia no tempo para formar uma trajetória circular ou na forma do número oito, onde o MPC deve manter o robô na trajetória. Nesse último caso, o algoritmo do MPC leva em consideração os próximos pontos da trajetória que o robô deverá passar. Dessa forma, o robô estará menos suscetível aos efeitos da rede, tendo em vista que, a trajetória futura, limitada ao horizonte de predição, é fornecida ao algoritmo de otimização do MPC. A sequência de controle é então ajustada conforme a trajetória, o que será aproveitada pelo atuador para manter o robô na trajetória desejada mesmo nas situações de atraso ou perda de pacotes.

Alguns trabalhos propõem arquiteturas que englobam computação na borda e MPC. Seisa *et al.* [76] apresentam uma arquitetura baseada na borda para UAVs autônomos, em que as tarefas de controle de alto nível relacionadas à trajetória são transferidas para o servidor de borda, de forma a aproveitar os recursos disponíveis e colocar o MPC em seus limites. Utilizam um contêiner para executar o MPC com o ROS e também ferramentas para orquestrar a aplicação. Seisa *et al.* [75] utilizaram um arcabouço experimental similar ao do artigo anterior, porém, aplicado a múltiplos robôs móveis que se comunicam com a borda por rede Wi-Fi. No servidor de borda, um MPC centralizado é responsável por enviar os sinais de controle aos robôs. Restrições que impõem distâncias mínimas e máximas entre os robôs são também incluídas no problema de otimização do MPC. Experimentos são realizados

com quatro robôs, utilizando diferentes horizontes de predição. No arcabouço utilizado, o tempo que um pacote com a informação da posição do robô alcança o servidor de borda, somado ao tempo de execução do MPC e ao tempo que o pacote com o sinal de controle alcança o robô, ultrapassa 300 milissegundos. Mesmo nessas condições, segundo os autores, os robôs mantêm o comportamento esperado quando se escolhe valores mais altos para o horizonte de predição,

O autor desta tese propõe em [61] uma arquitetura com computação na borda e MPC, baseada no ROS, para o controle de AGVs industriais. A arquitetura tem uma dupla camada de MPCs, aplicadas na borda e no lado local. No servidor de borda, o MPC planeja trajetórias, livres de colisões, para múltiplos AGVs. No lado local, cada AGV recebe sua respectiva trajetória e a rastreia utilizando um MPC de rastreamento da trajetória. A arquitetura também inclui uma camada de MPC para planejamento da trajetória no lado do AGV, que pode ser utilizada quando o AGV perder a conexão com a borda em intervalos de tempo maiores. Em [20] o mesmo autor utilizou uma arquitetura similar e apresentam os resultados de um experimento com quatro AGVs simulados no Gazebo. Os AGVs são capazes de seguirem em suas trajetórias, prevenindo colisões entre si e com obstáculos fixos, mesmo quando o sinal da rede está degradado. Nesses dois últimos trabalhos, foi adotada uma estratégia na implementação do MPC de planejamento da trajetória na borda. Ao invés de executar um único nó de MPC, o que dependendo do tamanho do problema a execução em tempo real não é possível, o planejamento da trajetória é dividido em diversos nós do ROS, um para cada AGV. Esses nós publicam suas respectivas trajetórias para subscrição no lado local, além de que são subscritas entre si para serem utilizadas nas restrições de prevenção de colisões.

Trabalhos conduzidos na Suécia pela Universidade de Lund, em conjunto com a empresa da área de telecomunicações, Ericsson, mesclaram o uso do MPC com computação na borda e redes 5G. Skarin *et al.* [82] avaliaram o desempenho do MPC como controlador de um processo de equilíbrio de uma esfera em uma viga que se movimenta em torno de um eixo, implementado em nós com diferentes capacidades de processamento, mas também hospedados em locais diferentes. Os nós incluem um *Raspberry Pi* como computador adjacente ao processo, um nó de borda ligado ao processo através de uma rede 5G, um *data center* localizado a poucos quilômetros de distância, e uma instância de máquina virtual cujo *data center* está localizado em Frankfurt, na Alemanha, e é mantido pela *Amazon Web Services* (AWS).

Em outro trabalho, Skarin *et al.* [79] implementaram o MPC no servidor de borda, porém, um controlador local também é aplicado. No modo assistido, o MPC executado na nuvem ou na borda, é responsável por controlar o processo da esfera e viga. Se os pacotes estiverem atrasados ou forem perdidos, o sistema é mudado para o modo local e é controlado por um controlador mais simples, o Regulador Linear Quadrático (LQR – *Linear Quadratic Regulator*). Implementando também os modos assistido e local, Skarin *et al.* [80] executaram em paralelo na nuvem MPCs com diferentes horizontes de predição, onde o lado local escolhe a melhor solução que foi recebida dentro do prazo. Skarin *et al.* [81] propõem ainda uma arquitetura de controle em duas camadas para o MPC baseado em nuvem. A camada da nuvem executa o controlador nominal, que atua nas condições normais e opera em uma frequência mais alta. O dispositivo local realiza a amostragem do estado, estimativa do estado, envia os dados para o MPC da nuvem e atua com o sinal de controle retornado. Assume-se que o dispositivo local não tem recursos suficientes para executar o mesmo MPC da nuvem. Assim, o MPC local opera com um período de amostragem maior, tendo mais tempo para obter uma solução entre o intervalo de amostragens. O desempenho dos serviços de nuvem para execução de sistema de controle críticos é também objeto de estudo de Skarin *et al.* [83]. Embora os recursos sejam abundantes, a nuvem é um ambiente com ruídos e sujeita à latência, prejudicial ao sistema de controle. Portanto, os autores avaliam um conjunto de plataformas e infraestruturas em nuvem com o intuito de hospedar sistemas de controle.

3.2.2 Discussão

Nos experimentos realizados por Seisa *et al.* [75], os autores afirmam que em caso de maiores latências, o horizonte de predição do MPC pode ser aumentado para que o robô mantenha o comportamento esperado. No entanto, nenhuma camada de controle é incluída no lado local para que o robô possa continuar em sua trajetória em caso de degradação do sinal da rede. Na arquitetura proposta no Capítulo 5, o MPC no servidor de borda planeja a trajetória de múltiplos AGVs, ao passo que o lado local (AGV) soluciona um problema mais simples, apenas rastreando a trajetória já planejada na borda. Conforme os resultados dos experimentos apresentados em [20], essa estratégia pode manter os AGVs em suas trajetórias, mesmo com a degradação do sinal na rede de comunicação.

3.3 Considerações finais

Nesta revisão bibliográfica, buscou-se apresentar um panorama a respeito dos AGVs, iniciando com a apresentação dos principais algoritmos utilizados no planejamento da rota e do movimento, apresentando também como o controle e a coordenação dos AGVs podem ser feitos de forma centralizada ou descentralizada, assim como os sistemas assistidos por computação na borda. Na seção seguinte, foram apresentados os casos de controladores industriais como um serviço de nuvem ou borda, incluindo os casos específicos com o MPC. No final de cada seção, foi realizada uma discussão de modo a evidenciar as oportunidades de pesquisa existentes.

A partir da revisão, foi verificado o potencial do MPC para o controle de robôs móveis, considerando a sua ação preditiva, além da possibilidade de inclusão das restrições cinemáticas, dinâmicas e de prevenção de colisões no problema de otimização. Observou-se que a ação preditiva do MPC é pouco explorada para lidar com os problemas relacionados ao uso das redes de comunicação, tais como os atrasos e perdas de pacotes, num cenário de transferência do controlador para a nuvem ou borda, principalmente em aplicações com robôs móveis, ou mais especificamente, com AGVs. Nesse sentido, a arquitetura já explorada em [61] e [20], é apresentada em detalhes no Capítulo 5.

Capítulo 4

Implementação do MPC

Este capítulo é uma extensão da teoria do MPC apresentada na Seção 2.3 do capítulo de Fundamentação Teórica. Na Seção 4.1 é apresentado brevemente o processo de conversão de um Problema de Controle Ótimo, tipo de problema onde o MPC se enquadra, em um problema de Programação Não-Linear. A Seção 4.2 contém uma demonstração utilizando ferramentas computacionais, em que dois tipos de problemas envolvendo o MPC aplicados ao robô de tração diferencial são implementados: a estabilização em um ponto e o rastreamento da trajetória.

4.1 Solução do Problema de Controle Ótimo

Modelos de sistemas não-lineares são tipicamente representados por modelos no tempo contínuo [64]. O controle desses sistemas a partir de uma técnica de controle ótimo resulta em um Problema de Controle Ótimo (OCP – *Optimal Control Problem*) [64]. O objetivo do OCP é encontrar uma trajetória para a entrada de controle $\mathbf{u}(t)$, de tal modo que a trajetória de estado resultante $\mathbf{x}(t)$, minimize a função custo J_{ocp} , dada por [64]:

$$J_{ocp}(\mathbf{x}(t), \mathbf{u}(t)) = l_f(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} l_s(\mathbf{x}(t), \mathbf{u}(t)) dt \quad (4.1)$$

A função custo J_{ocp} é normalmente dividida em duas partes, sendo o custo de estágio, resultante da integração de l_s (chamada de termo de Lagrange), e o custo final l_f (chamado de termo de Mayer). Um exemplo típico de OCP é representado da seguinte forma [64]:

$$\min_{\mathbf{x}(t), \mathbf{u}(t)} J_{ocp}(\mathbf{x}(t), \mathbf{u}(t)) \quad (4.2a)$$

$$\text{sujeito à } \dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \forall t \in [t_0, t_f] \quad (4.2b)$$

$$\mathbf{x}(t_0) = \mathbf{x}_0 \quad (4.2c)$$

$$\mathbf{h}(\mathbf{x}(t_f)) = 0 \quad (4.2d)$$

$$\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) \leq 0, \forall t \in [t_0, t_f] \quad (4.2e)$$

A função J_{ocp} está sujeita ao modelo do sistema (Equação 4.2b), ao estado inicial do sistema em $t = t_0$ (Equação 4.2c), às restrições de igualdade (Equação 4.2d), e às restrições de desigualdade da trajetória do estado e da entrada de controle (Equação 4.2e).

Existem diferentes métodos para a solução de um OCP [27; 65]. Para o caso do MPC, os métodos mais utilizados são os chamados métodos diretos, onde o problema é primeiro discretizado e depois otimizado [65]. Esses métodos desempenham um papel importante, pois a solução pode ser obtida dentro dos intervalos de tempo críticos de aplicações em tempo real [64].

Nos métodos diretos, o problema de controle ótimo é transcrito em um problema de Programação Não-Linear (NLP – *Nonlinear Programming*). Eles se subdividem nos métodos do (a) Único Disparo (*Single Shooting*), (b) Múltiplos Disparos (*Multiple Shooting*) e (c) Colocação (*Collocation*) [27; 64; 65]. A partir disso, o problema é resolvido utilizando métodos numéricos apropriados. Um NLP é normalmente formulado por [64]:

$$\begin{aligned} \min_{z \in \mathbb{R}^n} \quad & J(z) \\ \text{sujeito à} \quad & h(z) = 0 \\ & g(z) \leq 0 \end{aligned} \quad (4.3)$$

As variáveis de otimização são representadas por $z \in \mathbb{R}^n$, onde h define as restrições de igualdade e g as restrições de desigualdade.

Para a solução desse problema de otimização de dimensão finita, primeiro é necessário escolher uma representação da função contínua, processo normalmente chamado de discretização [65]. Isso envolve três partes do OCP: a trajetória de controle, a qual é constante entre

os intervalos de predição; a trajetória do estado, normalmente discretizada por um método de integração; e as restrições [65]. A seguir, são apresentados os passos de solução do NLP a partir dos métodos do Único Disparo e Múltiplos Disparos, dado que esses métodos são os mais utilizados.

4.1.1 Único Disparo

Nesse método, apenas a entrada de controle \mathbf{u} é discretizada e permanece constante entre os intervalos de predição. Essa é, portanto, a única variável a ser otimizada. O estado do sistema é propagado a partir de um método de integração, onde normalmente os métodos de Euler e Runge-Kutta são utilizados. A integração é realizada do instante inicial até o instante final em uma única varredura, motivo pelo qual o método é chamado de Único Disparo. O método pode ser comparado a um tiro ao alvo [92]. Primeiro, é feita uma estimativa de um bom ângulo de disparo. Em seguida, é verificado se o alvo foi atingido. Caso contrário, o ângulo é ajustado com base no resultado anterior para que o problema seja resolvido sequencialmente em várias iterações.

Essa propagação do estado é implementada da seguinte forma:

$$\begin{aligned}
 \mathbf{x}_{k+1} &= f(\mathbf{x}_0, \mathbf{u}_k) \\
 \mathbf{x}_{k+2} &= f(f(\mathbf{x}_0, \mathbf{u}_k), \mathbf{u}_{k+1}) \\
 \mathbf{x}_{k+3} &= f(f(f(\mathbf{x}_0, \mathbf{u}_k), \mathbf{u}_{k+1}), \mathbf{u}_{k+2}) \\
 &\vdots
 \end{aligned} \tag{4.4}$$

O estado do sistema nos instantes \mathbf{x}_{k+i} são calculados de forma recursiva usando a função f , partindo do estado inicial \mathbf{x}_0 . O NLP é finalmente formulado conforme a seguir:

$$\min_{\mathbf{u}} J_{NLP}(\mathbf{x}(k), \mathbf{u}(k)) \tag{4.5a}$$

$$\text{sujeito à } g(\mathbf{x}(k+i), \mathbf{u}(k+i)) \leq 0, i = 0, \dots, N-1 \tag{4.5b}$$

A vantagem do Único Disparo é o menor número de variáveis de otimização que o método requer. Entretanto, tem a desvantagem de que o problema pode se tornar altamente

não-linear, devido à recursividade na propagação do estado. Isso é ainda mais crítico para os sistemas instáveis [64].

4.1.2 Múltiplos Disparos

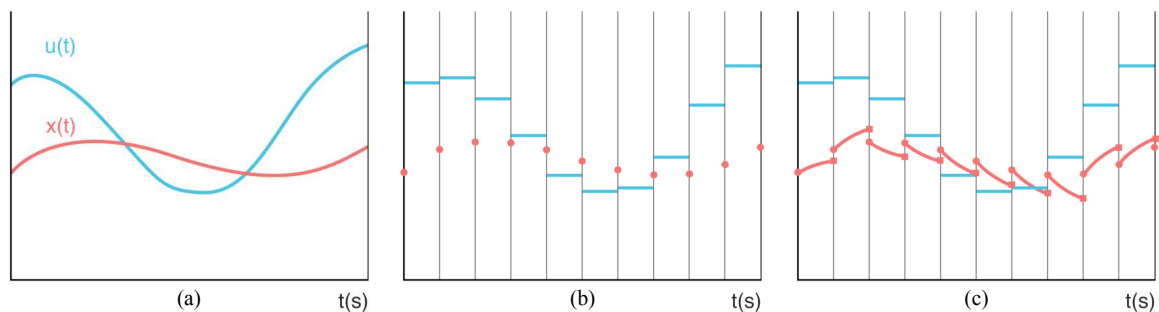
O método dos Múltiplos Disparos foi proposto por Bock e Plitt [9]. O procedimento geral é o mesmo do Único Disparo, com a diferença de que o estado também é discretizado e passa a ser uma variável de otimização. A cada intervalo $[t_k, t_{k+1}]$ o modelo do sistema é integrado, motivo pelo qual o método é chamado de Múltiplos Disparos. A previsão do estado ao longo do horizonte é implementada da seguinte forma:

$$\begin{aligned} \mathbf{x}_{k+1} &= f(\mathbf{x}_0, \mathbf{u}_k) \\ \mathbf{x}_{k+2} &= f(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) \\ \mathbf{x}_{k+3} &= f(\mathbf{x}_{k+2}, \mathbf{u}_{k+2}) \\ &\vdots \end{aligned} \quad (4.6)$$

Observe que não existe a recursividade como no método do Único Disparo, dado que o estado é obtido em cada um dos intervalos de previsão.

O procedimento do método pode ser melhor explicado com o auxílio das ilustrações da Figura 4.1.

Figura 4.1: Ilustração da discretização para o método dos Múltiplos Disparos: (a) estado e controle no tempo contínuo; (b) estado e controle discretizados; (c) integração das trajetórias do estado.



Fonte: [92].

Primeiro, as trajetórias no tempo contínuo do estado e da entrada de controle do gráfico da

Figura 4.1(a) são discretizadas, resultando no gráfico da Figura 4.1(b). Agora, cada intervalo k consiste de um sinal de controle \mathbf{u}_k e um estado inicial \mathbf{x}_k , como pode ser visto na Figura 4.1(b). Ao integrar ao longo de cada intervalo a partir do estado \mathbf{x}_k , o estado \mathbf{x}_{k+1} pode ser determinado, gerando os segmentos da trajetória do estado vistos na Figura 4.1(c). Observe que existem descontinuidades entre o estado resultante da integração e o início do estado no próximo intervalo. Portanto, no método dos Múltiplos Disparos a seguinte restrição de continuidade é aplicada:

$$\mathbf{x}_{k+1} - \tilde{\mathbf{x}}_{k+1} = 0 \rightarrow \mathbf{x}_{k+1} - f(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) = 0 \quad (4.7)$$

Por fim, podemos definir o NLP para o método dos Múltiplos Disparos:

$$\min_{\mathbf{x}, \mathbf{u}} \quad J_{NLP}(\mathbf{x}(k), \mathbf{u}(k)) \quad (4.8a)$$

$$\text{sujeito à} \quad \mathbf{x}_k - \mathbf{x}_0 = 0 \quad (4.8b)$$

$$\mathbf{x}_{k+i+1} = f(\mathbf{x}_{k+i}, \mathbf{u}_{k+i}), i = 0, \dots, N-1 \quad (4.8c)$$

$$g(\mathbf{x}(k+i), \mathbf{u}(k+i)) \leq 0, i = 0, \dots, N \quad (4.8d)$$

Observe que agora o NLP inclui as restrições da continuidade no estado inicial (Equação 4.8b) e no restante do horizonte (Equação 4.8c), além das restrições de desigualdade na Equação 4.8d, as quais também estão presentes no NLP do método do Único Disparo.

O método dos Múltiplos Disparos tem algumas características que tornam a sua convergência mais fácil quando comparado ao método dos Únicos Disparos, reduzindo o tempo para obtenção da solução. Algumas delas são [64]: (a) a divisão da solução em intervalos resulta em um comportamento mais linear; (b) enquanto no método do Único Disparo são feitas primeiramente as predições e depois a otimização, no método dos Múltiplos Disparos as predições e a otimização ocorrem simultaneamente; (c) o NLP tem uma estrutura mais esparsa.

Escolhido um método, o NLP pode ser finalmente formulado a partir de ferramentas como o CasADi ¹ [6] e o ACADO ² [34]. Essas ferramentas utilizam solucionadores que

¹<https://web.casadi.org/>

²<https://acado.github.io/>

podem resolver o NLP com métodos tais como o Método do Ponto Interior (IPM – *Interior Point Method*) e a Programação Quadrática Sequencial (SQP – *Sequential Quadratic Programming*).

4.1.3 Métodos de integração

Em ambos os métodos de discretização apresentados anteriormente, é necessário fazer previsões do estado do sistema a partir de seu modelo. Esse procedimento é implementado por métodos de integração, o qual faz o cálculo da trajetória de $\mathbf{x}(t)$, partindo de um valor inicial, de modo que se aproxime o melhor possível da dinâmica do sistema representada por uma equação diferencial ordinária.

Um método de integração comum e simples é o método de Euler, definido por [64]:

$$\tilde{\mathbf{x}}_{k+1} = \mathbf{x}_k + hf(\mathbf{x}_k, \mathbf{u}_k) \quad (4.9)$$

e por onde se obtém o estado aproximado $\tilde{\mathbf{x}}_{k+1}$, a partir do estado atual \mathbf{x}_k , da entrada de controle \mathbf{u}_k e do passo de tempo h . Outro método comum é o de Runge-Kutta de quarta ordem, normalmente chamado de RK4 [92]. A ordem do método define a quantidade de aproximações calculadas em um passo de tempo. Na Figura 4.2 está ilustrada uma representação gráfica de um exemplo de integração com o RK4. Primeiro são calculadas as inclinações das retas R_1 à R_4 , através das respectivas Equações 4.10a à 4.10d. A aproximação do estado é então a média ponderada de todas as inclinações, calculada pela Equação 4.11.

$$R_1 = f(\mathbf{x}_k, \mathbf{u}_k) \quad (4.10a)$$

$$R_2 = f\left(\mathbf{x}_k + \frac{h}{2}R_1, \mathbf{u}_k\right) \quad (4.10b)$$

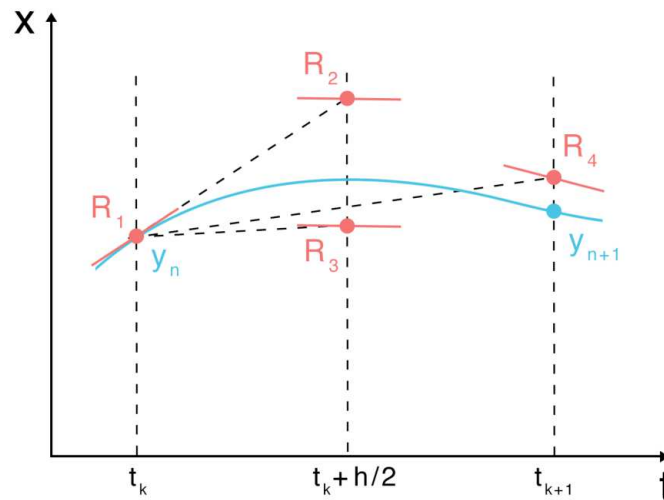
$$R_3 = f\left(\mathbf{x}_k + \frac{h}{2}R_2, \mathbf{u}_k\right) \quad (4.10c)$$

$$R_4 = f(\mathbf{x}_k + hR_3, \mathbf{u}_k) \quad (4.10d)$$

$$\tilde{\mathbf{x}}_{k+1} = \mathbf{x}_k + \frac{h}{6}(R_1 + 2R_2 + 2R_3 + R_4) \quad (4.11)$$

Seguindo o raciocínio de Rajasingham [64], no contexto do MPC, a escolha do método de integração é muito importante. Se a integração da trajetória do estado não é precisa, o

Figura 4.2: Representação gráfica de um exemplo de integração com o método de Runge-Kutta de quarta ordem.



Fonte: [92].

comportamento previsto difere significativamente do comportamento real do sistema, mesmo que se tenha um modelo que o represente fielmente. Por outro lado, métodos de integração mais precisos têm maior demanda computacional. Portanto, pode-se buscar um equilíbrio entre a precisão e a demanda computacional. Ambos os fatores são afetados pelo tamanho do passo de integração e a ordem do método. Se a execução em tempo real for um requerimento crítico, o passo de integração não pode ser escolhido arbitrariamente pequeno quando se quer uma alta precisão. Para aplicações com o MPC em tempo real, é conveniente escolher um método de maior ordem, como o RK4, o qual é mais apropriado para passos de integração maiores.

4.2 Demonstração no Matlab e CasADi

O CasADi [6] é uma ferramenta com software de código aberto para otimização numérica em geral, e controle ótimo em particular. Com a ferramenta, o usuário pode formular o problema e escolher um solucionador para obtenção da solução, dentre vários disponíveis, tal como o IPOPT [96], o mais popular e automaticamente incluído na instalação da ferramenta. O CasADi pode ser usado com C++, Python e Matlab/Octave, nos principais sistemas operacionais (Windows, Linux e macOS).

Nas subseções seguintes está demonstrado como formular o MPC para o robô móvel de tração diferencial (ver Seção 2.4) utilizando o CasADi. Embora o trabalho tenha sido desenvolvido em códigos Python executados no ROS 2, nessa demonstração foi utilizado o Matlab, de forma que o leitor possa compreender o processo de solução do MPC e reproduzir a simulação caso seja de interesse. Pode-se dizer que os códigos utilizados na demonstração apresentada funcionam como um núcleo do MPC. Quando se está no ambiente do ROS 2, o mesmo núcleo é utilizado, adicionando-se outros recursos do ROS 2, como a utilização dos tópicos para obter o estado do robô e enviar os sinais de controle, por exemplo. A demonstração foi implementada no CasADi v3.5.5, Matlab R2022b e sistema operacional macOS.

Utilizando as simulações do trabalho de Mehrez *et al.* [53] como referência, a demonstração é dividida em duas partes: estabilização em um ponto e rastreamento da trajetória. Na primeira, determina-se um ponto fixo onde o robô deve estabilizar. Na segunda, a referência deixa de ser um ponto fixo e passa a ser um conjunto de pontos que formam uma trajetória de referência. Os códigos utilizados são adaptações dos códigos disponibilizados no repositório de Mehrez [52]. Mais aplicações que utilizam o CasADi para implementação do MPC podem ser vistas em [54] e [74].

Antes da demonstração, vale relembrar a formulação genérica do MPC apresentada no Capítulo 2, representada pelas equações a seguir:

$$\min_{\mathbf{x}, \mathbf{u}} J = \sum_{k=0}^{N-1} \|\mathbf{x}(k) - \mathbf{x}_{ref}\|_Q^2 + \|\mathbf{u}(k) - \mathbf{u}_{ref}\|_R^2 \quad (4.12a)$$

$$\text{sujeito à} \quad \mathbf{x}(k+1) = f(\mathbf{x}(k), \mathbf{u}(k)) \quad (4.12b)$$

$$\mathbf{x}(0) = \mathbf{x}_0 \quad (4.12c)$$

$$\mathbf{x}_{min} \leq \mathbf{x}(k) \leq \mathbf{x}_{max} \quad (4.12d)$$

$$\mathbf{u}_{min} \leq \mathbf{u}(k) \leq \mathbf{u}_{max} \quad (4.12e)$$

Observe a adição de \mathbf{x} como variável a ser otimizada na Equação 4.12a, dado que o método dos Múltiplos Disparos é utilizado neste trabalho.

4.2.1 Estabilização em um ponto

A seguir, os procedimentos para implementação do MPC no problema da estabilização em um ponto estão apresentados e explicados por partes. No problema também é incluído um obstáculo fixo, do qual o robô deve desviar no trajeto até a referência. O desvio de obstáculos é implementado através da adição de uma restrição de desigualdade apresentada adiante. O leitor pode verificar o código escrito e comentado no Matlab no Apêndice A.1. Para execução do código, é necessário baixar o pacote do CasADi para Matlab e depois importá-lo. Instruções de utilização do CasADi podem ser vistas em sua documentação [13].

Seguindo o código do Apêndice A.1, inicialmente são declaradas as variáveis que definem o passo de tempo (T), o horizonte de predição (N) e os limites dos sinais de controle (velocidades linear e angular do robô). Para definição da função $f(\mathbf{x}, \mathbf{u})$, são declaradas as variáveis de estado e de controle. No CasADi, as expressões para formulação do NLP são definidas de forma simbólica. Por exemplo, o estado do robô é representado por três variáveis simbólicas do tipo SX:

```
x = SX.sym('x')
y = SX.sym('y')
theta = SX.sym('theta')
```

São declarados ainda os vetores que representam as variáveis de estado e controle que serão otimizadas (X e U no código, respectivamente), a variável que armazena a função custo, o vetor que contém as restrições, as matrizes Q e R , além do vetor de parâmetros P . Esse vetor contém os valores do estado inicial (e atual durante a simulação) e da referência a serem utilizados na função custo.

Após as declarações das variáveis, é possível prosseguir na construção da função custo. Antes disso, é necessário definir o estado inicial e a restrição de estado inicial inerente ao método dos Múltiplos Disparos, ou seja, fazemos $\mathbf{x}_0 - \tilde{\mathbf{x}}_0$ (Equação 4.7). A função custo é construída com um laço de repetição `for` com $k = 1:N$. Em cada repetição, é necessário fazer a integração (o método RK4 é utilizado) e depois implementar a subtração da Equação 4.7. As subtrações vão sendo incorporadas ao vetor g ao longo das repetições. As igualdades dessas restrições são inseridas em outra parte do código mencionada mais adiante. Nenhum sinal de controle de referência foi utilizado, ou seja, $\mathbf{u}_{ref} = 0$ e $\mathbf{u}(k)$ é multiplicado diretamente pela matriz R (Equação 4.12a).

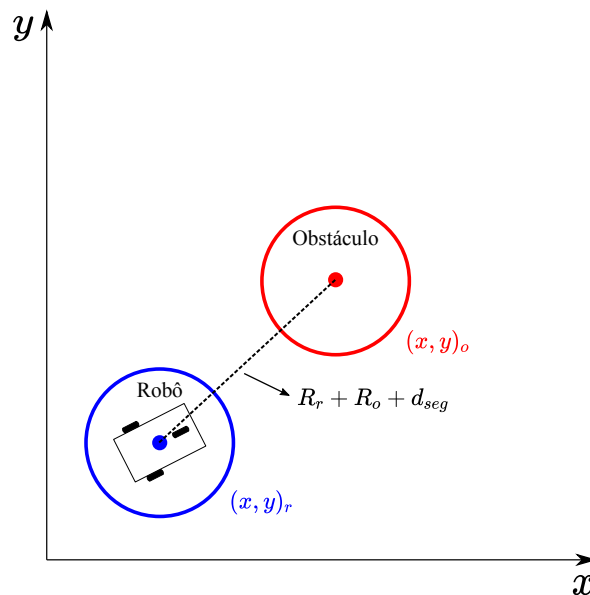
Se no problema não fosse incluído o obstáculo fixo, a construção do vetor de restrições g finalizaria no passo anterior. Caso existam os obstáculos, é necessário adicionar uma restrição de desigualdade, a qual prevenirá as colisões mantendo o robô distante dos obstáculos ao longo das predições. Portanto, é necessário utilizar outro laço de repetição `for` para incorporar as restrições no vetor g . Na Figura 4.3 apresenta-se uma ilustração para auxiliar no entendimento da formulação da restrição. O robô representado por um círculo de raio R_r e o obstáculo por um círculo de raio R_o , estão respectivamente posicionados nas coordenadas $(x, y)_r$ e $(x, y)_o$. A distância entre o robô e o obstáculo é calculada pela distância Euclidiana. Dado que a restrição é de desigualdade, é determinado que a distância entre os centros dos círculos que representam o robô e o obstáculo, seja maior ou igual à soma dos raios dos círculos, somada de uma distância de segurança. Teremos então a seguinte formulação:

$$\sqrt{(x_r - x_o)^2 + (y_r - y_o)^2} \geq (R_r + R_o + d_{seg}) \quad (4.13a)$$

$$\sqrt{(x_r - x_o)^2 + (y_r - y_o)^2} - (R_r + R_o + d_{seg}) \geq 0 \quad (4.13b)$$

$$-\sqrt{(x_r - x_o)^2 + (y_r - y_o)^2} + (R_r + R_o + d_{seg}) \leq 0 \quad (4.13c)$$

Figura 4.3: Representações do robô e do obstáculo feitas por círculos para formulação da restrição de prevenção de colisões.



Fonte: Produzida pelo autor.

No CasADi, essa restrição pode ser inserida da seguinte forma:

$$-\infty \leq -\sqrt{(x_r - x_o)^2 + (y_r - y_o)^2} + (R_r + R_o + d_{seg}) \leq 0 \quad (4.14)$$

Essa restrição é processada em todo o horizonte de predição, de modo que o termo da distância Euclidiana presente na Equação 4.14 é modificado para:

$$\sqrt{[x_r(k) - x_o]^2 + [y_r(k) - y_o]^2} \quad (4.15)$$

onde as variáveis $x_r(k)$ e $y_r(k)$ formam as coordenadas da posição do robô no passo k das predições. Seguindo o código, a expressão entre as desigualdades da Equação 4.15 é inserida no vetor g ao longo das predições, a partir do laço de repetição `for` com `k = 1:N+1`. Da mesma forma que as restrições anteriores, as desigualdades são inseridas posteriormente.

Os passos seguintes consistem em definir um vetor com as variáveis de otimização, declarar o NLP, ajustar os opcionais do IPOPT, declarar o solucionador e uma estrutura de dados. Essa estrutura contém uma série de argumentos que são passadas ao solucionador durante as iterações do MPC. Inicialmente são preenchidos os vetores lbx (mínimos) e ubx (máximos), com os mínimos e máximos do estado e do sinal de controle, de forma a atender às Equações 4.12d e 4.12e das restrições de desigualdade. Na sequência são preenchidos os vetores dos argumentos lbg e ubg com o valor 0, atendendo à restrição de igualdade do método dos Múltiplos Disparos. Vale notar que na criação da função custo, apenas a primeira parte da Equação 4.7 é construída. O valor zero então preenchido nos argumentos lbg e ubg completam a igualdade da equação. Ainda nos mesmos argumentos são inseridas as desigualdades da Equação 4.14.

Com os passos descritos anteriormente o NLP está totalmente formulado. Agora o solucionador pode ser chamado para obtenção da solução. Com apenas uma iteração, possivelmente o robô ainda não alcançará a referência (a menos que esteja muito próximo dela). Dessa forma, um laço de repetição `while` é utilizado para chamar o solucionador várias vezes. Nessa simulação, a condição para interrupção do laço, é que, o erro, definido pela norma da diferença do estado inicial e da referência, seja menor que um valor determinado, e que uma quantidade mínima de iterações seja executada.

A cada iteração do laço `while`, os valores da chave de parâmetros p da estrutura de dados são atualizados. Os parâmetros incluem o estado atual x_0 (na primeira iteração é o estado inicial) e a referência x_{ref} . Lembrando que na construção da função custo os parâmetros P

foram declarados de forma simbólica, entretanto, durante a simulação, esses parâmetros recebem valores numéricos através dos argumentos p . O solucionador é finalmente chamado, recebendo como entrada o estado atual e a estrutura de dados dos argumentos. Da solução obtida, são extraídos o sinal de controle e a trajetória. Visto que a simulação é realizada no Matlab, sem a aplicação do sinal de controle em um robô emulado ou real, a evolução do estado é implementada na função `shift`, também criada no Matlab, a qual recebe o estado atual e a entrada de controle para retornar o próximo estado através do modelo discreto do robô (Equação 2.4). Por fim, antes de iniciar a próxima iteração, os vetores que armazenam as soluções da trajetória e do sinal de controle, são atualizados, removendo-se o primeiro valor da sequência e repetindo-se o último. Os vetores atualizados são utilizados como um chute inicial da solução para a próxima iteração e faz a solução convergir mais rapidamente. Outras variáveis auxiliares não foram citadas aqui mas podem ser vistas no código do Apêndice A.1, assim como outros comentários.

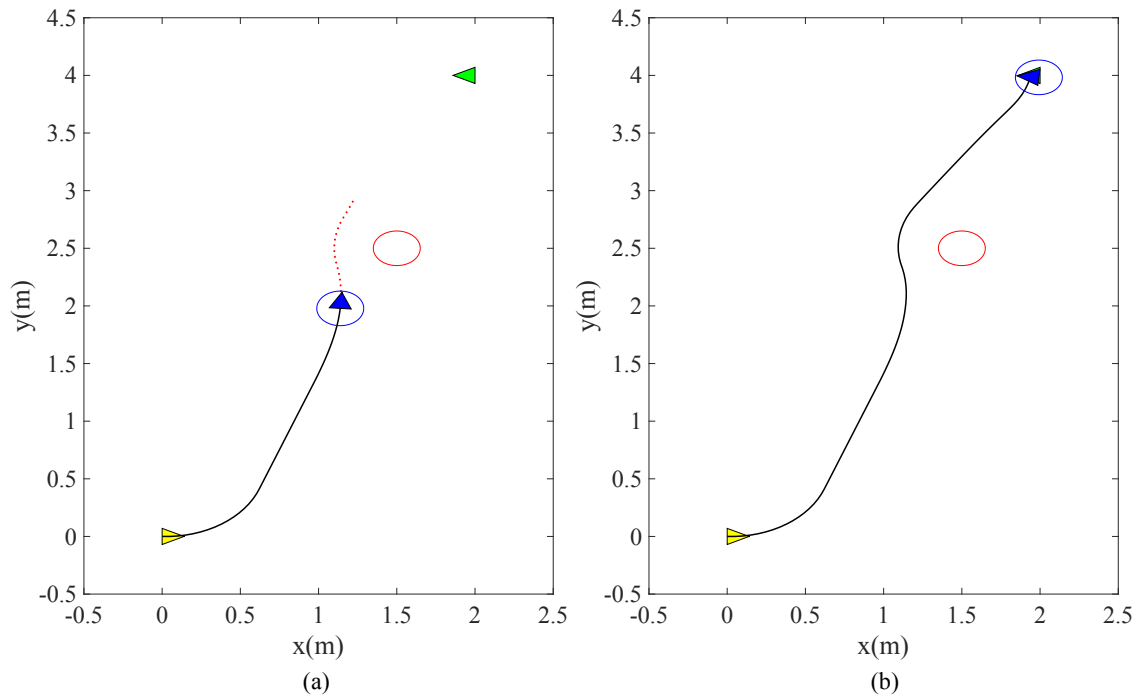
Uma simulação de exemplo foi realizada com os parâmetros listados na Tabela 4.1. Os vetores que definem \mathbf{x}_0 (estado inicial) e \mathbf{x}_{ref} (referência) contêm nessa ordem os valores de x , y e θ . O erro e as iterações são os critérios de parada, ou seja, o laço das iterações será interrompido quando o erro for menor que 0,1 e que no mínimo 120 iterações tenham sido executadas. No código, a quantidade de iterações é obtido pelo quociente do tempo de simulação desejado em segundos com o passo de tempo.

Tabela 4.1: Parâmetros ajustados na simulação da estabilização em um ponto com o Matlab.

T	100 ms
N	20
Q	$\text{diag}[10,0 \ 10,0 \ 0,1]$
R	$\text{diag}[0,5 \ 0,05]$
u_v^{min}	- 0,5 m/s
u_v^{max}	+ 0,5 m/s
u_ω^{min}	- $\pi/4$ rad/s
u_ω^{max}	+ $\pi/4$ rad/s
\mathbf{x}_0	$[0,0 \ 0,0 \ 0,0]^T$
\mathbf{x}_{ref}	$[2,0 \ 4,0 \ \pi]^T$
R_r	0,15 m
R_o	0,15 m
d_{seg}	0,1 m
Erro	$\leq 0,1$
Iterações	120

Na Figura 4.4(a) está plotado o gráfico da trajetória parcial do robô, antes do alcance da referência. Os triângulos representam as posições do robô, sendo o amarelo a posição inicial, o azul a posição atual e o verde a posição final. A linha contínua preta é a trajetória já percorrida, ao passo que a trajetória prevista (para $N = 20$) é representada pelos pontos vermelhos à frente do robô. Os círculos azul e vermelho representam o robô e o obstáculo fixo, respectivamente, onde seus raios são utilizados na restrição de prevenção de colisões. Na Figura 4.4(b) está plotado o mesmo gráfico com a trajetória completa até o término da simulação.

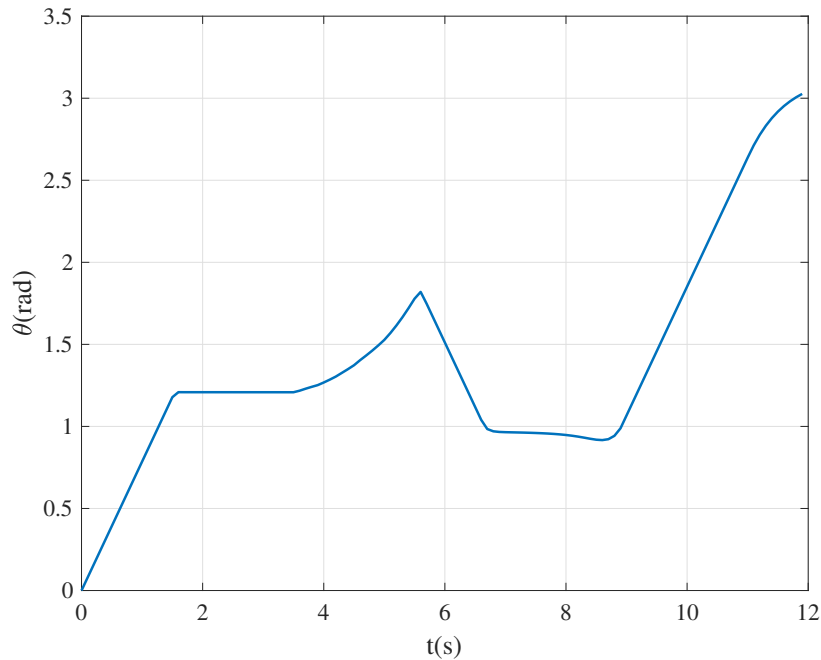
Figura 4.4: Trajetórias da simulação da estabilização em um ponto: (a) trajetória parcial do robô, com a trajetória prevista representada pelos pontos vermelhos; (b) trajetória completa.



Fonte: Produzida pelo autor.

A trajetória do robô inclui ainda o ângulo de orientação em relação ao eixo x , a qual é vista em função do tempo no gráfico da Figura 4.5. Observe que o ângulo parte de 0 rad, já que esse é o estado inicial para essa variável. Na sequência, o ângulo é ajustado ao longo da trajetória, passando por uma variação brusca no instante próximo aos seis segundos de simulação para desvio do obstáculo fixo. Aproximadamente em nove segundos de simulação, o ângulo começa a ser ajustado para que a referência de π rad seja alcançada.

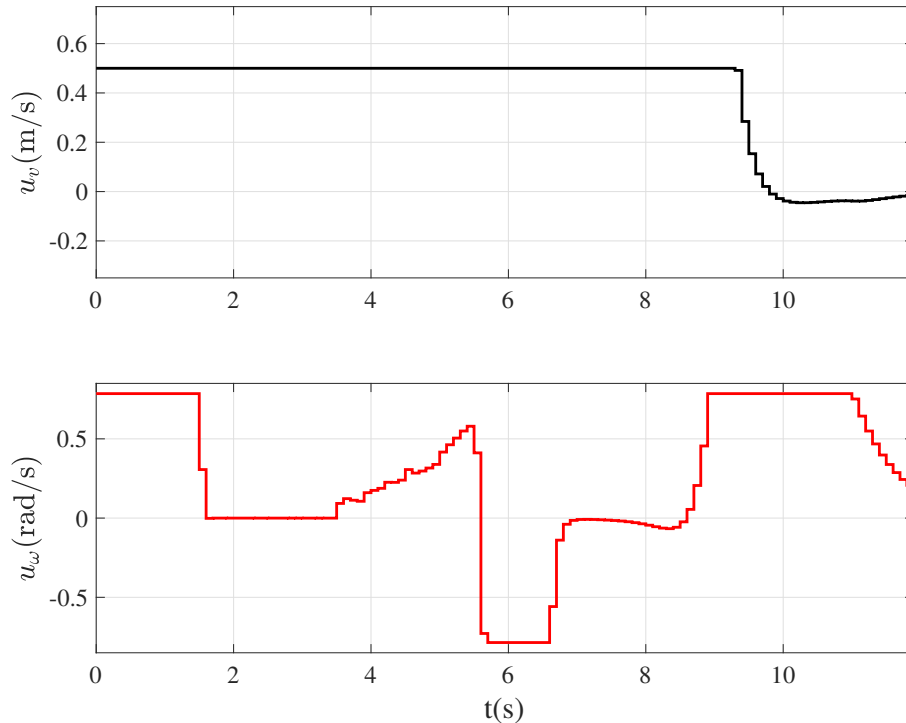
Figura 4.5: Ângulo de orientação do robô em função do tempo na simulação da estabilização em um ponto.



Fonte: Produzida pelo autor.

Na Figura 4.6 estão plotados os sinais de controle das velocidades linear e angular necessários para conduzir o robô do estado inicial até a referência. Observa-se que o sinal de controle da velocidade linear permanece em 0,5 m/s até próximo dos instantes finais da simulação, sendo esse o valor máximo ajustado para essa variável no MPC. Aproximadamente em dez segundos de simulação, o MPC reduz o sinal de velocidade do robô para próximo de zero, de modo que seja possível estabilizá-lo na referência especificada. As variações do ângulo de orientação do robô vistas na Figura 4.5 são associadas ao comportamento do sinal de controle da velocidade angular. Verifica-se que, quando o sinal de controle é diferente de zero, o ângulo de orientação é alterado, dado que o MPC estará impondo uma alteração nessa variável de estado através do sinal de controle. Quando o sinal de controle tem valor nulo ou está próximo disso, o ângulo de orientação permanece constante ou é levemente alterado.

Figura 4.6: Sinais de controle das velocidades linear (u_v) e angular (u_ω) na simulação da estabilização em um ponto.



Fonte: Produzida pelo autor.

4.2.2 Rastreamento da trajetória

Enquanto no problema da estabilização em um ponto é preciso definir apenas um ponto de referência, fazendo com que a função custo contenha uma única referência \mathbf{x}_{ref} ao longo do horizonte de predição, no rastreamento da trajetória a função custo recebe um conjunto de pontos de referência, de mesmo tamanho do horizonte de predição N . Dessa forma, a cada passo de tempo $k + i$, com $i = [0, N - 1]$, a função custo recebe uma referência $\mathbf{x}_{ref}(k + i)$. Suponha que queiramos que o robô siga uma trajetória circular. Nesse caso, teremos um conjunto de referências que inclui não somente o ponto onde o robô deve estar posicionado naquele instante, mas também os outros $N - 1$ pontos futuros que formam segmentos da trajetória circular. A função custo tem, portanto, o seguinte formato:

$$\min_{\mathbf{x}, \mathbf{u}} J = \sum_{k=0}^{N-1} \|\mathbf{x}(k) - \mathbf{x}(k)_{ref}\|_Q^2 + \|\mathbf{u}(k) - \mathbf{u}(k)_{ref}\|_R^2 \quad (4.16)$$

O código escrito no Matlab para demonstração do rastreamento da trajetória pode ser

visto no Apêndice A.2. Poucas diferenças existem em relação ao código da estabilização em um ponto. Uma delas é o vetor dos parâmetros P declarado simbolicamente no início do código, agora maior, visto que receberá mais dados de referência. A outra é a inserção dos argumentos p na estrutura de dados dos argumentos, no laço de repetição `while` durante as iterações do MPC. Além de conter o estado atual x_0 , p também inclui todos os pontos que formam segmentos da trajetória desejada. Outra diferença é que nessa demonstração não é incluído o obstáculo fixo, portanto, não existe a restrição da Equação 4.14 e os vetores `lbg` e `ubg` não incluem as desigualdades referentes a essa restrição.

Para demonstração do MPC de rastreamento da trajetória, foi feita uma simulação onde o robô segue uma trajetória na forma da Lemniscata, a qual tem forma similar ao algarismo 8. As coordenadas e o ângulo de orientação da trajetória em função do tempo são gerados pela seguinte expressão:

$$x_{ref}(k) = 3 \sin(0,1kT) \quad (4.17a)$$

$$y_{ref}(k) = 5 \cos(0,05kT) \quad (4.17b)$$

$$\theta_{ref}(k) = \arctan \left[\frac{y(k) - y(k-1)}{x(k) - x(k-1)} \right] \quad (4.17c)$$

A cada iteração do MPC, um laço de repetição `for` incrementa k e gera a trajetória que será utilizada naquela iteração. Na sequência os argumentos p são atualizados com a trajetória e passados ao solucionador. O restante do código segue o mesmo algoritmo da estabilização em um ponto.

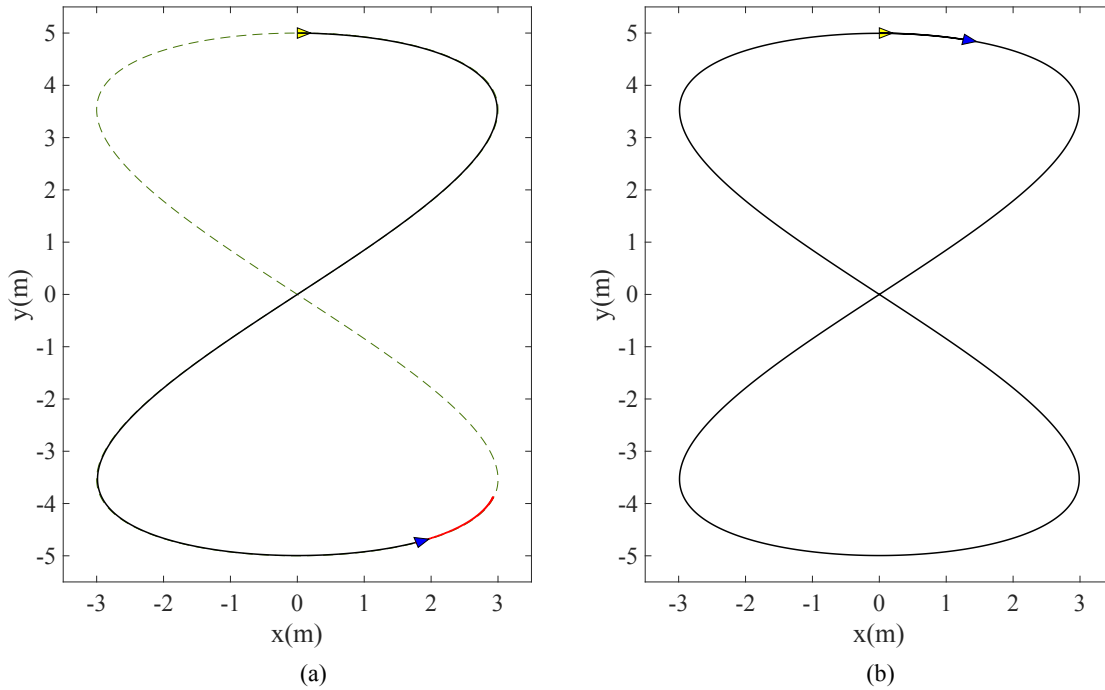
Na Tabela 4.2 estão apresentados os parâmetros utilizados na simulação. Comparando com a tabela da simulação anterior, observe que não existe uma referência fixa x_{ref} , dado que o parâmetro passa a ser variável no tempo e é definido pelas Equações 4.17a a 4.17c. Outros parâmetros ausentes são os ligados à prevenção de colisões, já que a simulação não tem obstáculo fixo, além do parâmetro do erro. Nesse caso a condição de interrupção do laço de repetição é apenas o número de iterações especificado.

Tabela 4.2: Parâmetros ajustados na simulação do rastreamento da trajetória com o Matlab.

T	100 ms
N	80
Q	$\text{diag}[1,0 \ 1,0 \ 0,1]$
R	$\text{diag}[0,5 \ 0,05]$
u_v^{min}	- 0,6 m/s
u_v^{max}	+ 0,6 m/s
u_ω^{min}	- $\pi/4$ rad/s
u_ω^{max}	+ $\pi/4$ rad/s
\mathbf{x}_0	$[0,0 \ 5,0 \ 0,0]^T$
Iterações	1300

Na Figura 4.7(a) está plotado o gráfico da trajetória parcial do robô. Da mesma forma que no gráfico da simulação anterior, os triângulos amarelo e azul, respectivamente, representam as posições inicial e atual do robô. A linha tracejada em verde é a Lemniscata gerada pelas Equações 4.17a e 4.17b. A partir dela é possível visualizar antecipadamente a trajetória que o robô deve fazer. A linha contínua preta é a trajetória já percorrida e a trajetória prevista (para $N = 80$) é representada pelos pontos vermelhos à frente do robô. Nessa simulação, o horizonte de predição foi propositalmente aumentado para que no gráfico as predições sejam melhor vistas. Dada a concentração dos pontos, as predições são vistas como uma linha no gráfico. Na Figura 4.7(b) está plotado o mesmo gráfico com a trajetória completa até o término da simulação. Perceba que nessa simulação o robô pode completar várias voltas da trajetória definida. Portanto, no resultado apresentado, foi definido um tempo de simulação de 130 segundos, de tal modo que, com o passo de tempo de 0,1 segundos, a simulação finaliza após 1300 iterações. Essa quantidade de iterações é suficiente para que o robô finalize uma volta e percorra o início da próxima.

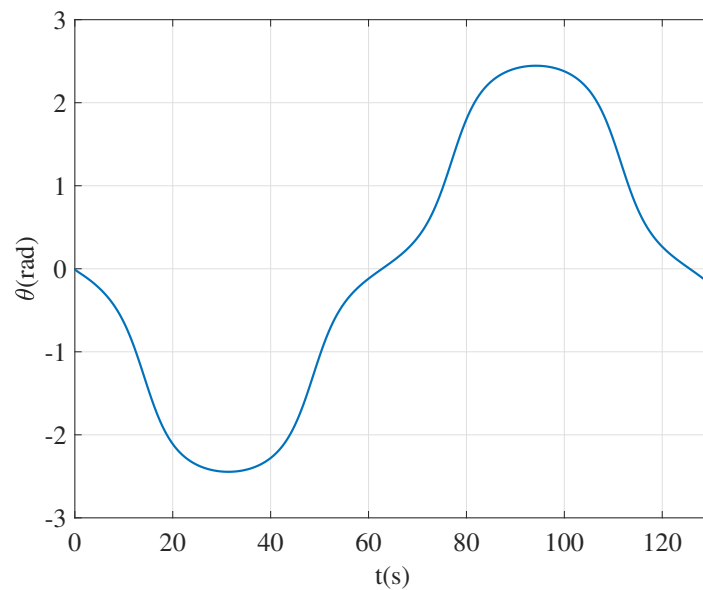
Figura 4.7: Trajetórias da simulação do rastreamento da trajetória: (a) trajetória parcial do robô, com a trajetória prevista representada pelos pontos vermelhos; (b) trajetória completa.



Fonte: Produzida pelo autor.

A trajetória do robô inclui também o ângulo de orientação em relação ao eixo x , vista em função do tempo no gráfico da Figura 4.8. O ângulo tem o estado inicial em 0 rad, tendo um pico negativo quando o robô atinge um quarto da trajetória completa, ou seja, quando x e y são iguais a zero (ver Figura 4.7). Quando o robô passa da metade da trajetória, o ângulo varia até atingir 0 rad novamente, alcançando um pico positivo em três quartos da trajetória. Nos instantes finais da simulação, o ângulo varia até 0 rad à medida que o robô se aproxima do estado inicial.

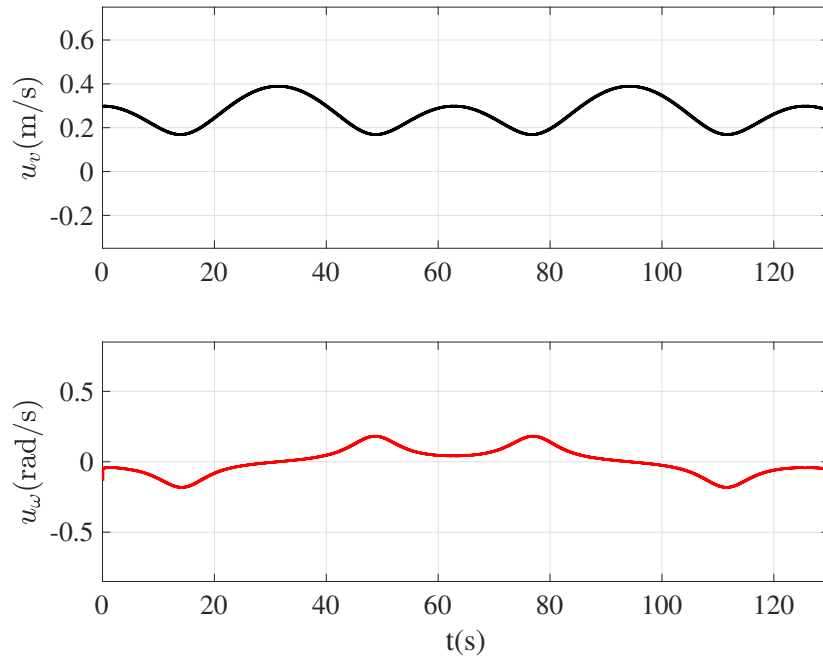
Figura 4.8: Ângulo de orientação do robô em função do tempo na simulação do rastreamento da trajetória.



Fonte: Produzida pelo autor.

Na Figura 4.9 estão plotados os sinais de controle das velocidades linear e angular necessários para que o robô percorra a trajetória especificada. Verifica-se na figura que o sinal de controle da velocidade linear tem um comportamento oscilante, visto que é necessário fazer sucessivas alterações na velocidade do robô para que a trajetória especificada seja rastreada. O sinal de controle da velocidade angular também é ajustado pelo MPC, de tal modo que o robô possa manter a orientação necessária para permanecer na trajetória especificada. A variação do ângulo de trajetória visto na Figura 4.8 é o resultado da aplicação desses sinais.

Figura 4.9: Sinais de controle das velocidades linear (u_v) e angular (u_ω) na simulação do rastreamento da trajetória.



Fonte: Produzida pelo autor.

É possível visualizar nas previsões representadas pelos pontos vermelhos no gráfico da Figura 4.7(a), o efeito de se utilizar essa outra formulação do MPC, onde a função custo leva em consideração os pontos futuros da trajetória. Observe que na posição onde o robô se encontra, o MPC já tem conhecimento da curva que está por vir, podendo se antecipar e ajustar as ações de controle para que o robô possa fazer a curva permanecendo na trajetória especificada. Essa abordagem do MPC pode ser adotada em cenários onde o computador que executa o MPC está separado do robô por uma rede de comunicação, conforme os resultados dos experimentos realizados em [60]. Se o MPC envia para o robô uma sequência de sinais de controle futuros, o atuador pode aplicar esses sinais na ocorrência de atrasos ou perdas de pacotes na rede de comunicação, o que mantém o robô na trajetória especificada.

4.3 Considerações finais

Nesse capítulo foi visto o processo de conversão de um Problema de Controle Ótimo (onde se enquadra o MPC) em um problema de Programação Não-Linear, que por fim é soluci-

onado através de métodos numéricos. Na sequência, foram realizadas demonstrações com duas abordagens do MPC para o robô móvel de tração diferencial, a estabilização em um ponto e o rastreamento da trajetória. Essas abordagens do MPC são elementos fundamentais da arquitetura proposta no próximo capítulo. A abordagem da estabilização em um ponto é utilizada como planejamento de trajetórias para múltiplos AGVs. Essas trajetórias são recebidas pelos AGVs e são utilizadas em uma segunda etapa, utilizando a abordagem do rastreamento da trajetória.

Capítulo 5

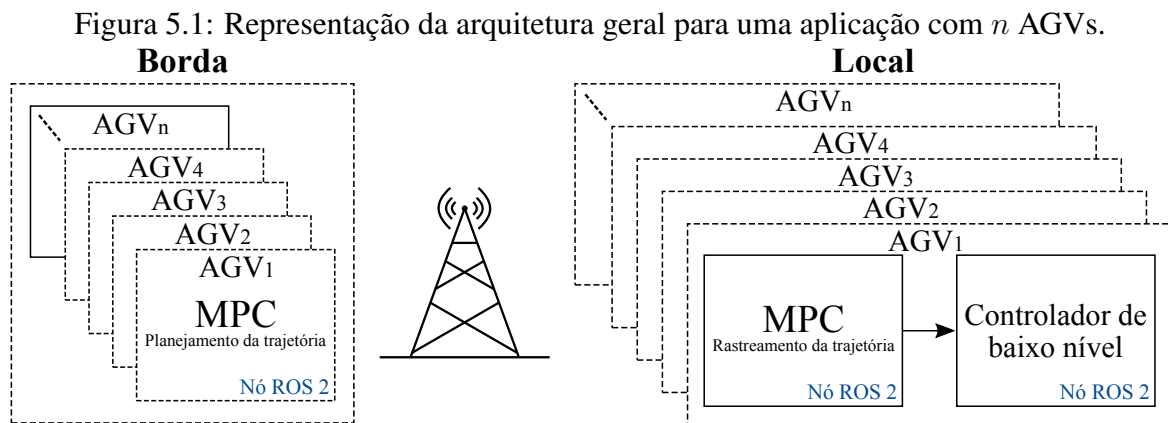
Arquitetura proposta com dupla camada de MPCs para navegação de AGVs assistida por computação na borda

Baseado no cenário e problemática apresentados no Capítulo 1, na revisão bibliográfica do Capítulo 3, e nos trabalhos apresentados por Omena *et al.* [60; 61; 20], nesta seção é proposta uma arquitetura de controle de AGVs com aplicação do MPC em duas camadas, sendo uma no servidor de borda e outra nos AGVs, de tal forma que o sistema esteja apto a lidar com situações em que a comunicação entre a borda e o AGV esteja sujeita a atrasos e perdas de pacotes. A arquitetura proposta está apresentada a seguir e é dividida entre a camada de MPC executada no servidor de borda e a camada executada no lado local. Essas camadas são implementadas em nós do ROS 2, chamados aqui de *Nós de Borda*, utilizados para o MPC de planejamento da trajetória no servidor de borda, e *Nós Locais*, utilizados para o MPC de rastreamento da trajetória no lado local ou do AGV. A última seção do capítulo contém uma demonstração no ROS 2 referente à aplicação da arquitetura em uma simulação com quatro AGVs.

5.1 Arquitetura geral

Na Figura 5.1 está apresentada a arquitetura geral para uma aplicação com n AGVs, envolvendo os nós de borda e os nós locais. Os blocos tracejados representam sistemas com-

putacionais diferentes. Existem, portanto, n nós de MPC para planejamento da trajetória executados no servidor de borda, enquanto o lado local está distribuído em n AGVs que executam o MPC para rastreamento da trajetória e o controlador de baixo nível em seus computadores embarcados. A antena ilustrada na figura representa o meio de comunicação entre os lados, realizada com uma rede sem fio. Na demonstração apresentada no final deste capítulo e nas simulações apresentadas no Capítulo 6, os nós locais, ou seja, os nós associados aos AGVs, não trocam informações entre si, entretanto, a comunicação entre esses nós é possível no ambiente de simulação desenvolvido, e caso necessário, pode ser utilizada em trabalhos futuros. Ressalta-se, ainda, que o ROS 2 foi adotado neste trabalho como meio de execução das camadas da arquitetura e de comunicação entre os nós, entretanto, a arquitetura pode ser implementada de outro modo, desde que a comunicação entre os processos na borda e locais seja possível para troca de informações das trajetórias e estados dos AGVs.



Fonte: Produzida pelo autor.

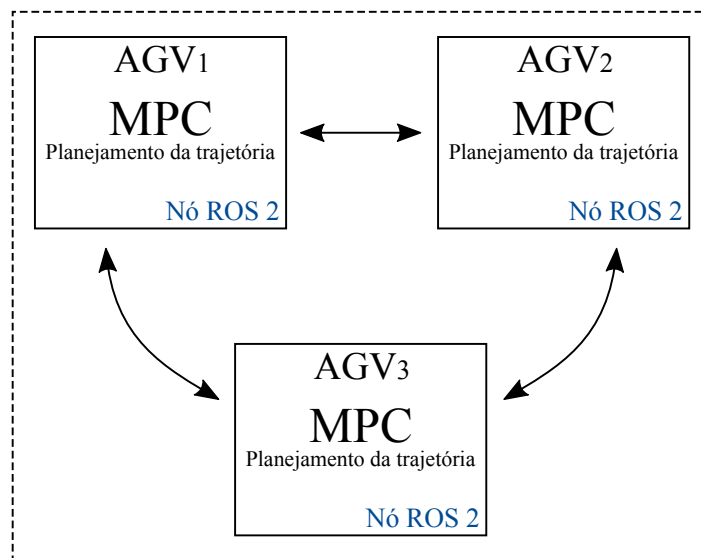
Nas seções seguintes, a apresentação da arquitetura é realizada por partes em um exemplo com três AGVs, através da separação dos nós de borda e dos nós locais.

5.2 Nós de borda

Os nós executados no servidor de borda para o exemplo com três AGVs estão representados pelos blocos da Figura 5.2. Cada um dos blocos é executado por meio de um processo do ROS 2, aqui, chamado de nó. Os nós, portanto, trocam informações através de tópicos e podem ser servidores ou clientes de serviços. No exemplo com três AGVs, temos respecti-

vamente três nós do ROS 2 para execução do MPC de planejamento da trajetória. Esses nós também são servidores do serviço do MPC, ou seja, sempre que for necessário mover um AGV, o seu respectivo serviço do MPC pode ser chamado para que a trajetória que o conduz até o ponto desejado possa ser gerada. O algoritmo do MPC utilizado no planejamento da trajetória tem os mesmos moldes do apresentado na demonstração da Subseção 4.2.1, no problema da estabilização em um ponto. Dado um ponto de referência, o MPC gera uma trajetória que é enviada ao respectivo AGV.

Figura 5.2: Diagrama dos nós executados no servidor de borda em um exemplo com três AGVs.



Fonte: Produzida pelo autor.

Foi visto na Subseção 4.1.2 que, no método de solução do MPC com os múltiplos disparos, otimizamos as variáveis do sinal de controle u e do estado x . Entretanto, nessa etapa de planejamento, apenas a trajetória do estado x ao longo do horizonte de predição é enviada para que seja rastreada no lado local. O procedimento de prevenção de colisões é também similar ao apresentado na demonstração da Subseção 4.2.1. Os obstáculos fixos são representados por círculos, onde seus raios e coordenadas são previamente estabelecidos. Além disso, é necessário prevenir as colisões entre os AGVs. Dado que as trajetórias são planejadas independentemente em cada um dos nós, essas trajetórias precisam ser compartilhadas entre os nós para que cada um dos nós faça o planejamento livre de colisões. Nesse procedimento, os AGVs também são representados por círculos, de forma que restrições são adicionadas para manter uma distância de segurança entre os AGVs, da mesma forma como

é feito entre um AGV e um obstáculo fixo. O compartilhamento da trajetória (representado por setas na Figura 5.2) é feito por tópicos, ou seja, cada nó publica a trajetória do respectivo AGV e se inscreve para receber as trajetórias dos demais. Tomando como exemplo o nó do AGV₁ na Figura 5.2, o nó publicará a trajetória do AGV₁ e se inscreve para receber as trajetórias do AGV₂ e do AGV₃. Uma desvantagem dessa abordagem de prevenção de colisões é que o problema de otimização aumenta junto com o número de obstáculos fixos e também de AGVs. Nas simulações apresentadas no capítulo seguinte, são apresentadas algumas estratégias para redução desse problema.

Embora os nós de MPC de planejamento estejam centralizados no servidor de borda, vale ressaltar que eles são independentes, o que se assemelha a uma abordagem em que a trajetória é planejada de forma descentralizada diretamente nos AGVs. Entretanto, na proposta apresentada, o servidor de borda, provido de mais recursos computacionais, é utilizado nessa etapa. Além disso, as trajetórias são compartilhadas dentro do mesmo sistema computacional e não estarão suscetíveis à degradação do sinal da rede. Outro ponto a ser destacado é que as trajetórias de todos os AGVs podem ser planejadas em um único problema de otimização executado em um único nó, com a desvantagem de que o problema aumenta com a quantidade de AGVs e pode se tornar inviável para execução em tempo real. Na arquitetura proposta foi adotado o modo de execução em nós separados, o que facilita a distribuição dos processos nos diferentes núcleos do processador, acelerando a execução.

Planejamento da trajetória

O nó de MPC de planejamento da trajetória para cada AGV deve solucionar o problema de otimização representados pela Equações 4.12a a 4.12e. Inclui-se a restrição de prevenção de colisões com obstáculos fixos da Equação 4.14, calculando-se a distância Euclidiana em todo o horizonte de predição com a Equação 4.15. A quantidade de restrições inseridas é definida pela quantidade de obstáculos fixos. Conforme já citado anteriormente, é necessário ainda prevenir as colisões dos AGVs entre si, utilizando uma restrição similar à da Equação 4.14 para a formulação do problema no CasADi:

$$-\infty \leq -\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} + (R_i + R_j + d_{seg}) \leq 0, \text{ para } i \neq j \quad (5.1)$$

em que x_i e y_i são as coordenadas da posição do AGV_{*i*}, x_j e y_j são as coordenadas da posição

do AGV_j , R_i e R_j são os raios dos círculos que representam os AGVs i e j , respectivamente, e d_{seg} é a distância de segurança entre os AGVs. Observe que a quantidade de vezes em que essa restrição é incluída depende da quantidade de AGVs no cenário. Por exemplo, com três AGVs, o AGV_1 deve prevenir colisões com os AGVs 2 e 3, de forma que a restrição é incluída duas vezes no MPC desse AGV, com o mesmo procedimento valendo para os demais. Essa restrição é processada para todo o horizonte de predição. Continuando com o exemplo do AGV_1 , o seu nó de MPC no servidor de borda se inscreve para receber as trajetórias dos outros AGVs, geradas para um horizonte de predição N . A partir de um laço de repetição `for` a restrição é processada para todo o horizonte de predição, executando-se esse procedimento duas vezes, dado que deve-se prevenir colisões com os outros dois AGVs. Utilizando apenas o termo referente à distância Euclidiana na restrição da Equação 5.1, a distância d entre os AGVs i e j é calculada com a seguinte equação:

$$d_{i,j}(k) = \sqrt{[x_i(k) - x_j(k)]^2 + [y_i(k) - y_j(k)]^2} \quad (5.2)$$

onde o índice k representa o passo da predição em que a distância está sendo calculada e que deve estar no intervalo $0 \leq k \leq N - 1$ (incrementado no laço de repetição `for`).

Ao final de cada iteração do algoritmo do MPC de planejamento da trajetória, a lista q de tamanho $N + 1$ com os vetores que formam a trajetória, representado pela Equação 5.3, é gerada. Cada elemento da lista contém as posições que definem a trajetória, com x , y e θ .

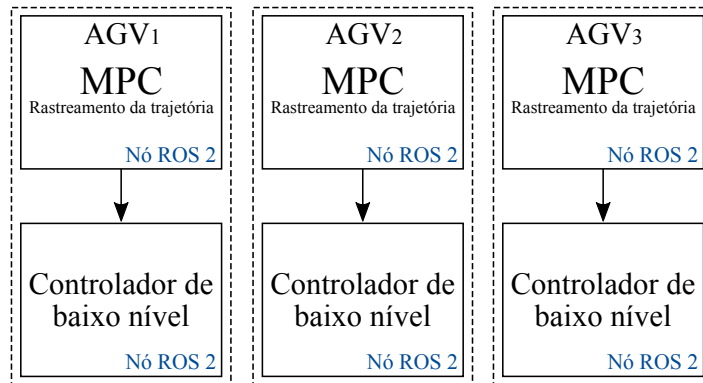
$$q_{0,\dots,N} = \left[\begin{bmatrix} x & y & \theta \end{bmatrix}_0, \begin{bmatrix} x & y & \theta \end{bmatrix}_1, \begin{bmatrix} x & y & \theta \end{bmatrix}_2, \dots, \begin{bmatrix} x & y & \theta \end{bmatrix}_N \right] \quad (5.3)$$

5.3 Nós locais

Na Figura 5.3 estão apresentados os nós locais, seguindo ainda o exemplo com três AGVs. Os nós do MPC de rastreamento da trajetória de cada AGV se inscrevem para receber suas respectivas trajetórias, as quais foram planejadas e publicadas pelos MPCs de planejamento no servidor de borda.

Observe que quando comparado ao algoritmo do MPC de planejamento da trajetória no servidor de borda, o MPC de rastreamento da trajetória soluciona um problema mais simples, dado que o problema lida apenas com a função do rastreamento, sem a inclusão das

Figura 5.3: Diagrama dos nós executados nos AGVs. O controlador de baixo nível é utilizado para assegurar que o chassi do AGV terá as velocidades linear e angular definidas pelo MPC de rastreamento.



Fonte: Produzida pelo autor.

restrições para prevenção de colisões com obstáculos fixos e outros AGVs. Isso permite que o computador embarcado no AGV seja mais simples, além de reduzir o consumo de bateria.

Rastreamento da trajetória

O algoritmo do MPC de rastreamento também segue o mesmo padrão do apresentado na demonstração da Subseção 4.2.2 e utiliza a função custo da Equação 4.16, ou seja, o algoritmo incluirá no problema de otimização todos os pontos da trajetória. Suponha que o servidor de borda planeja trajetórias com horizonte de predição $N = 80$, logo, o algoritmo do MPC de rastreamento utilizará 80 pontos de trajetória em seu problema de otimização. Em caso de atrasos ou perdas de pacotes na rede de comunicação, o algoritmo continuará rastreando a última trajetória até que uma mais recente seja recebida.

Os sinais de controle resultantes são por fim publicados e o nó do controlador de baixo nível se inscreve para recebê-los. Esse controlador, por sua vez, ajusta o torque dos motores do AGV de tal modo que o chassi do veículo se movimente com as velocidades linear e angular especificadas pelo MPC de rastreamento.

5.4 Demonstração no ROS 2

Embora a arquitetura proposta seja validada no capítulo seguinte em diferentes cenários de aplicação, nesta demonstração com o ROS 2 é apresentada a aplicação da arquitetura em uma simulação com quatro AGVs, de modo que se possa ter uma melhor compreensão a respeito dos elementos da arquitetura, incluindo ainda os detalhes do comportamento preditivo na geração das trajetórias.

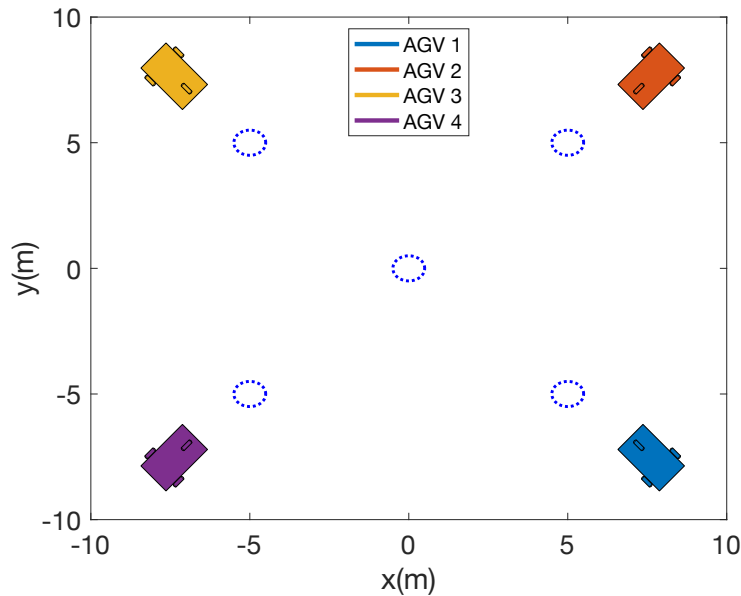
Dois computadores conectados entre si com cabo Ethernet foram utilizados, um representado o lado da borda, e o outro o lado local. O computador do lado da borda tem o sistema operacional Ubuntu 22.04 LTS e o ROS 2 com a distribuição *Foxy Fitzroy*. O computador do lado local tem quatro máquinas virtuais, onde cada uma delas representa um computador embarcado no AGV. As máquinas virtuais locais têm as mesmas distribuições do sistema operacional e do ROS 2 utilizadas no servidor de borda, porém, com recursos mais restritos. Da mesma forma que na demonstração da Seção 4.2, um simulador de robôs não foi utilizado, de modo que a evolução dos estados dos AGVs no lado local é também obtida a partir do modelo discreto do robô de tração diferencial (Equação 2.4). Nesse caso o controlador de baixo nível presente na arquitetura da Figura 5.1 ainda não é utilizado.

Na Figura 5.4 está ilustrado o estado inicial do cenário utilizado, com as posições iniciais dos AGVs descritas na Tabela 5.1, além de cinco obstáculos fixos de raio igual a 0,5 metros, com suas posições descritas na Tabela 5.2. Na Tabela 5.1 também estão as referências para todos os AGVs. Nessa simulação, cada AGV deve seguir para a posição onde inicialmente está o outro AGV no sentido diagonal, evitando colisões entre si e com os obstáculos fixos.

Tabela 5.1: Estados iniciais e referências para os AGVs 1, 2, 3 e 4 na demonstração com o ROS 2.

AGV	Estado inicial	Referência
1	$\mathbf{x}_1 = \begin{bmatrix} 7,5 & -7,5 & 3\pi/4 \end{bmatrix}^T$	$\mathbf{x}_1^{ref} = \begin{bmatrix} -7,5 & 7,5 & 3\pi/4 \end{bmatrix}^T$
2	$\mathbf{x}_2 = \begin{bmatrix} 7,5 & 7,5 & -3\pi/4 \end{bmatrix}^T$	$\mathbf{x}_2^{ref} = \begin{bmatrix} -7,5 & -7,5 & -3\pi/4 \end{bmatrix}^T$
3	$\mathbf{x}_3 = \begin{bmatrix} -7,5 & 7,5 & -\pi/4 \end{bmatrix}^T$	$\mathbf{x}_3^{ref} = \begin{bmatrix} 7,5 & -7,5 & -\pi/4 \end{bmatrix}^T$
4	$\mathbf{x}_4 = \begin{bmatrix} -7,5 & -7,5 & \pi/4 \end{bmatrix}^T$	$\mathbf{x}_4^{ref} = \begin{bmatrix} 7,5 & 7,5 & \pi/4 \end{bmatrix}^T$

Figura 5.4: Representação do estado inicial do cenário utilizado na demonstração com o ROS 2.



Fonte: Produzida pelo autor.

Tabela 5.2: Posições dos obstáculos fixos na demonstração com o ROS 2.

Obstáculo	Posição (x, y)
1	(0, 0)
2	(5, 5)
3	(-5, 5)
4	(-5, -5)
5	(5, -5)

Na Tabela 5.3 estão os parâmetros utilizados pelos nós de MPC de planejamento da trajetória. Os parâmetros da tabela já foram vistos na demonstração do capítulo anterior, exceto os referentes às distâncias de segurança entre os AGVs. Na tabela, $diam_{AGV}$ é o diâmetro referente ao círculo que representa o AGV, d_{seg}^{obs} e d_{seg}^{agv} se referem às distâncias de segurança para os obstáculos fixos e para os AGVs entre si, respectivamente. Os nós de MPC de rastreamento da trajetória utilizam os mesmos parâmetros, exceto os relacionados à prevenção de colisões (não aplicados nessa camada) e os parâmetros Q e R . Estes dois últimos parâmetros estão apresentados na Tabela 5.4.

Tabela 5.3: Parâmetros configurados no MPC de planejamento da trajetória na demonstração com o ROS 2.

T	100 ms
N	100
Q	$\text{diag}[5,0 \ 5,0 \ 0,5]$
R	$\text{diag}[0,5 \ 0,5]$
u_v^{min}	- 1,0 m/s
u_v^{max}	+ 1,0 m/s
u_ω^{min}	- $\pi/4$ rad/s
u_ω^{max}	+ $\pi/4$ rad/s
$diam_{AGV}$	0,75 m
d_{seg}^{obs}	1,0 m
d_{seg}^{agv}	1,0 m

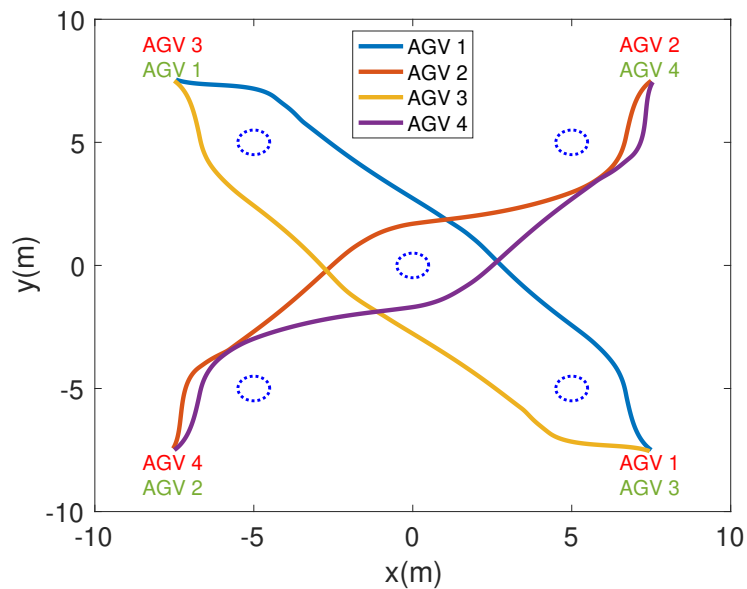
Tabela 5.4: Pesos Q e R ajustados no MPC de rastreamento da trajetória na demonstração com o ROS 2.

Q	$\text{diag}[10 \ 10 \ 20]$
R	$\text{diag}[10 \ 20]$

Na Figura 5.5 está plotado o gráfico das trajetórias percorridas pelos AGVs. Os círculos vermelhos no gráfico têm seus centros localizados na posições iniciais dos AGVs, e os verdes nas posições finais. Na Figura 5.6 estão plotados os gráficos dos sinais de controle das velocidades linear e angular, gerados pelos nós locais de MPC de rastreamento da trajetória, para conduzir os AGVs nas trajetórias planejadas pelo servidor de borda. Observam-se nos gráficos da Figura 5.6, que os pares de AGVs 1-3 e 2-4 têm um mesmo padrão nos sinais de controle, dado que esses pares seguem trajetórias similares (em sentidos opostos). Os sinais de controle da velocidade linear se aproximam da velocidade máxima permitida em alguns

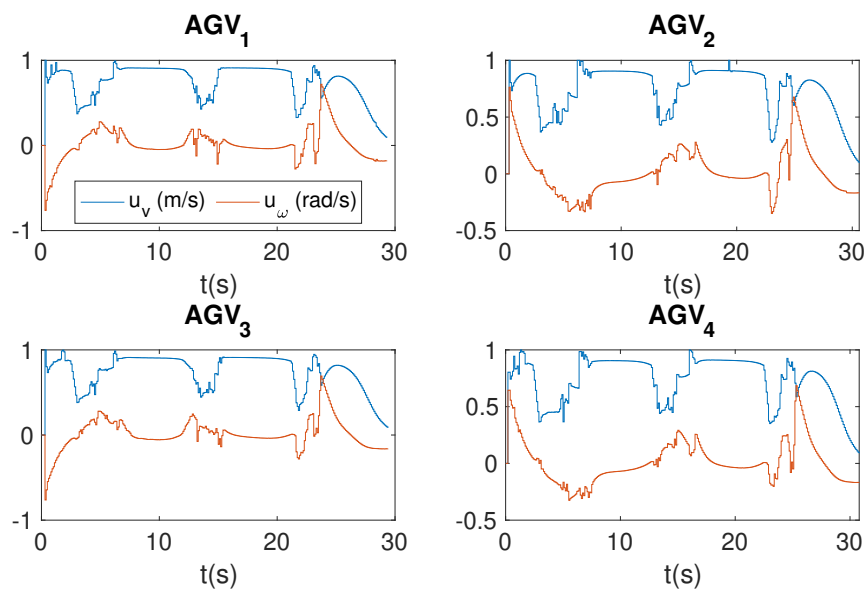
instantes, mas são reduzidos nos instantes em que os AGVs devem desviar dos obstáculos fixos ou quando estão próximos da referência. Diferentemente, nos instantes de desvio dos obstáculos, os sinais de controle da velocidade angular são ajustados para valores menores ou maiores que zero, de modo que as orientações dos AGVs sejam alteradas para os desvios.

Figura 5.5: Trajetórias percorridas pelos AGVs na demonstração com o ROS 2.



Fonte: Produzida pelo autor.

Figura 5.6: Sinais de controle aplicados nos AGVs na demonstração com o ROS 2.



Fonte: Produzida pelo autor.

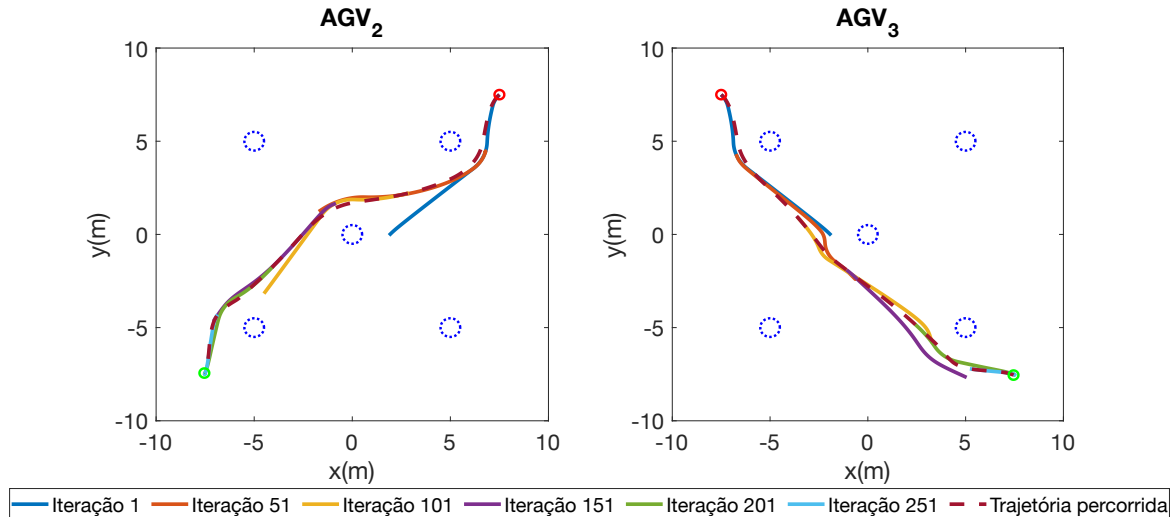
Para que não haja confusão com as trajetórias planejadas, vale ressaltar que as trajetórias ilustradas na Figura 5.5 são as trajetórias que foram de fato percorridas pelos AGVs, após a aplicação dos sinais de controle gerados pelo MPC de rastreamento da trajetória. Os pontos que formam a trajetória percorrida são obtidos a partir do modelo discreto do robô, após cada iteração do MPC de rastreamento.

Na simulação cada iteração do MPC de planejamento em todos os quatro nós é disparada por um temporizador do ROS 2, a cada 100 milissegundos (mesmo valor do passo de tempo T do MPC), cerca de trezentas vezes. Essa quantidade aproximada de iterações foi a necessária para que os AGVs alcançassem as referências. Em cada iteração, gera-se uma trajetória com N pontos, com um passo de tempo também de 100 milissegundos entre um ponto e outro. Essas trajetórias são alteradas enquanto os AGVs seguem em direção às suas referências. Isso acontece pelo fato de que, com os parâmetros N e T utilizados, os quais resultam em previsões de 10 segundos, os nós de MPC no servidor de borda não têm capacidade para gerar em uma única iteração, a trajetória completa para conduzir o AGV até a referência. Algum obstáculo ou outros AGVs que antes não foram alcançados pelo horizonte de previsão, farão com que a trajetória seja alterada assim que uma possível colisão seja detectada.

De modo a detalhar o processo de planejamento da trajetória e a situação destacada no parágrafo anterior, foram escolhidos os AGVs 2 e 3 para que sejam ilustradas as trajetórias geradas em diferentes iterações do MPC, vistas na Figura 5.7. Na figura estão ilustradas as trajetórias resultantes de seis iterações para cada AGV (linhas contínuas), iniciando pela primeira iteração, com passos de cinquenta iterações. Esses passos foram escolhidos para que se possa visualizar as trajetórias geradas em diferentes etapas, na condução do AGV do estado inicial até a referência, incluindo trajetórias geradas até aproximadamente a metade do percurso, depois da metade e próximo do fim.

As trajetórias de fato percorridas pelos AGVs, também ilustradas na Figura 5.5, são vistas nas linhas tracejadas da Figura 5.7. Observe que, na iteração 1 as trajetórias planejadas não consideram o obstáculo que está no centro. Na iteração 51 o obstáculo já é alcançado pelo horizonte de previsão e as trajetórias incluem o desvio. Ao mesmo tempo, o MPC do AGV₂ detecta uma possível colisão com o AGV₃ e altera a direção da trajetória. Na iteração 251 os AGVs já se encontram próximos às suas referências.

Figura 5.7: Trajetórias planejadas para os AGVs 2 e 3 em diferentes iterações (linhas contínuas) e as trajetórias de fato percorridas pelos AGVs (linhas tracejadas), na demonstração com o ROS 2.



Fonte: Produzida pelo autor.

5.5 Considerações finais

Foi apresentada nesse capítulo a arquitetura proposta, com a dupla camada de MPCs e computação na borda, para conduzir os AGVs até as suas referências, inclusive quando houver atrasos e perdas de pacotes na rede de comunicação. A arquitetura é baseada em nós do ROS 2 e contém os nós de borda e os nós locais. Os nós de borda, onde cada um deles representa um AGV, planejam as trajetórias e as compartilham entre si, para serem utilizadas no procedimento de prevenção de colisões. Os nós locais, também cada um associado a um AGV, estão distribuídos nos veículos e se inscrevem para receber as respectivas trajetórias, as quais serão utilizadas pelo algoritmo do MPC de rastreamento da trajetória. O nó do controlador de baixo nível finalmente se inscreve para receber os sinais de controle gerados nessa segunda camada de MPC. Uma demonstração de funcionamento da arquitetura foi realizada, com quatro AGVs, incluindo detalhes das previsões realizadas pelo MPC de planejamento da trajetória, ao longo da condução do AGV do estado inicial até a referência. Mais simulações com a arquitetura proposta em diferentes cenários de aplicação estão apresentadas no próximo capítulo.

Capítulo 6

Simulações e resultados

No Capítulo 5 foi apresentada a arquitetura com a dupla camada de MPCs, incluindo uma demonstração no ROS 2. Nesse capítulo a mesma arquitetura é utilizada em simulações com AGVs emulados no Gazebo, em quatro cenários diferentes utilizados para validação.

Na Seção 6.1 está a descrição dos recursos computacionais utilizados nas simulações, incluindo as máquinas virtuais instaladas nesses recursos. Na Seção 6.2 está detalhado como é feita a organização das aplicações desenvolvidas no ROS 2, utilizando *workspaces* e pacotes. Na Seção 6.3 está apresentado o Supervisor, um elemento que funciona junto à arquitetura para supervisionar a navegação dos AGVs nos cenários. Nas Seções 6.4 e 6.5 estão apresentadas as ferramentas de simulação e a arquitetura utilizadas nas simulações, respectivamente. Na Seção 6.6 estão apresentados os cenários utilizados nas simulações realizadas com a arquitetura. De início, está demonstrado o processo de criação dos mundos e dos AGVs no Gazebo, os quais são elementos que compõem os cenários. Na sequência, o primeiro cenário é utilizado para demonstrar os procedimentos necessários para execução das simulações no ambiente do ROS 2, bem como os nós, tópicos e serviços do ROS 2 envolvidos. Esse primeiro cenário não está associado a uma aplicação real, mas, além do processo de simulação ser demonstrado a partir dele, serve também para validação da arquitetura, inclusive com degradação do sinal na rede de comunicação.

Os resultados das simulações com o cenário incluem as trajetórias dos AGVs, os sinais de controle, o intervalo de tempo das iterações do MPC, além de como os recursos computacionais são utilizados em cada um dos lados, tais como o uso do processador e da memória. Nos cenários seguintes os resultados são apresentados diretamente, sem o detalhamento do

processo de execução dos nós do ROS 2. No segundo cenário, a simulação é feita em uma linha de produção hipotética, com o uso dos AGVs para transporte de materiais e produtos acabados. O terceiro é um centro logístico também hipotético, em que os AGVs transportam materiais entre armazéns e docas. Por último, o quarto cenário é utilizado para verificar a escalabilidade da arquitetura, analisando do uso dos recursos computacionais quando uma quantidade de AGVs maior é inserida.

6.1 Recursos computacionais utilizados

Na Tabela 6.1 estão descritos os recursos computacionais utilizados para a realização de simulações com a arquitetura apresentada na Figura 5.1. Dois computadores são utilizados: um representando o servidor de borda; e outro representando o lado local ou dos AGVs, ou seja, emulam computadores que em uma aplicação real estariam embarcados nos veículos. Ambos os computadores têm processadores baseados na arquitetura ARM. O computador com maior memória RAM e armazenamento (incluindo o externo) foi escolhido para representar o lado local, visto que nesse recurso são colocadas várias máquinas virtuais para execução do Gazebo e representação dos computadores vinculados aos AGVs.

Tabela 6.1: Descrição dos recursos computacionais utilizados.

Item	Computador de borda	Computador local
Processador	Apple M1 (8 CPU/7 GPU)	Apple M1 (8 CPU/8 GPU)
Memória RAM	8 GB	16 GB
Sistema operacional	macOS 13.1	macOS 13.1
Armazenamento	256 GB	256 GB + 256 GB (externo)

6.1.1 Máquinas virtuais

Dado que o trabalho envolve diferentes sistemas computacionais distribuídos nos lados da borda e local, a implementação da arquitetura proposta através de máquinas virtuais (MVs) foi a solução adotada para a execução das simulações. A partir dos recursos descritos na

Tabela 6.1, foram utilizadas várias instâncias do Ubuntu na distribuição 20.04 LTS. A virtualização em ambos os computadores é gerenciada pelo software *Parallels*¹, executado sobre o sistema operacional desses computadores. Devido à arquitetura ARM dos computadores utilizados, a distribuição do Ubuntu utilizada também é específica para essa arquitetura.

Os computadores trocam informações em rede e foram conectados via Ethernet em um roteador com portas Gigabit. As interfaces virtuais de rede das máquinas virtuais foram ajustadas no *Parallels* para o modo ponte, ou seja, as máquinas virtuais recebem um endereço IP (*Internet Protocol*) da rede, tal qual estivessem conectadas fisicamente ao roteador. Esse modo de conexão permite que as máquinas virtuais de qualquer lado se comuniquem entre si e assim as simulações possam ser executadas.

Para dar agilidade no processo de criação das máquinas virtuais, elas podem ser replicadas a partir de uma máquina virtual base, com todos as ferramentas necessárias instaladas, como o ROS 2 e o CasADi. As máquinas virtuais têm seus recursos computacionais ajustados individualmente e para as simulações foram usadas as configurações descritas na Tabela 6.2.

Tabela 6.2: Descrição dos recursos alocados para as máquinas virtuais.

Item	MVs de borda		MVs locais	
	MPC	Supervisor	Gazebo	AGVs (uma por AGV)
Processador	4 núcleos	2 núcleos	4 núcleos	1 núcleo
Memória RAM	3 GB	2 GB	6 GB	512 MB
Armazenamento	Variável até 64 GB			

Os recursos para cada máquina foram ajustados conforme a necessidade. No servidor de borda existe uma máquina virtual dedicada à execução dos nós de MPC para planejamento das trajetórias. Esses nós solucionam os problemas de otimização do MPC, incluindo as restrições de prevenção de colisões, o que vai demandar mais processamento. Essa máquina então conta com 4 núcleos de processador e 3 GB de memória RAM. O armazenamento é alocado por demanda e tem o limite de 64 GB, com a mesma regra valendo para todas as máquinas. Existe ainda outra máquina virtual no servidor de borda que é utilizada para

¹<https://www.parallels.com/br/>

execução do nó do Supervisor, com 2 núcleos e 2 GB de memória RAM. Percebeu-se que o software de virtualização limita o uso do processador para uma única máquina virtual, mesmo que parte dele esteja ocioso. Dessa forma, os nós do MPC e Supervisor executados no computador de borda foram distribuídos em diferentes máquinas virtuais, com o intuito de aproveitar melhor o processador desse computador. O Supervisor aqui é utilizado para gerenciar a navegação dos AGVs e será detalhado adiante.

As demais máquinas virtuais da Tabela 6.2 são as executadas no computador que representa o lado local. O Gazebo executa diversas atividades, tais como os motores de física (*physics engines*), o processamento gráfico e os nós do ROS 2 atrelados a cada entidade da simulação, necessários para a comunicação através dos tópicos. Com isso, uma máquina virtual foi alocada exclusivamente para o Gazebo, com 4 núcleos e 6 GB de memória RAM. Para o caso do AGV, utiliza-se uma máquina virtual mais simples, que possa representar um computador de baixo custo embarcado no veículo. Existe uma máquina virtual para cada AGV utilizado na simulação, com 1 núcleo de processador e 512 MB de memória RAM. Embora existam computadores de baixo custo que poderiam ser embarcados em AGVs reais e possuem processadores com múltiplos núcleos, no software de virtualização não se tem controle sobre o poder de processamento da máquina virtual, exceto a quantidade de núcleos. Assim, uma forma de se ter um computador mais simples vinculado ao AGV foi reduzir o número de núcleos do processador virtual a um único núcleo.

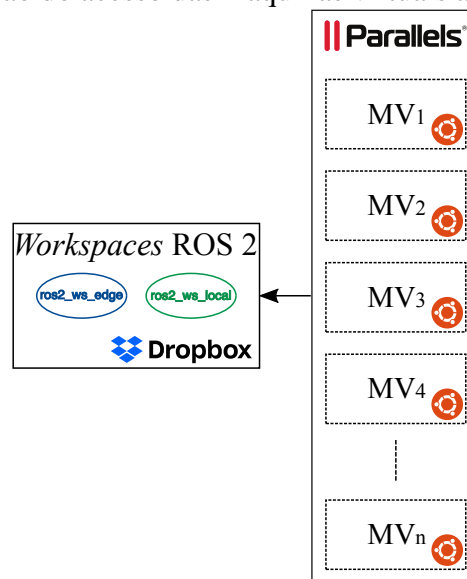
6.2 Workspaces e pacotes do ROS 2

O desenvolvimento de aplicações no ROS 2 é realizada dentro de diretórios, chamados de *workspaces*. Dentro de um *workspace* estão os pacotes, os quais podem ser escritos em Python ou C++. Com isso, é possível organizar as aplicações, distribuindo-se as tarefas em diferentes pacotes. Neste trabalho foram criados dois *workspaces*, um para o lado da borda e outro para o lado local, nomeados respectivamente de *ros2_ws_edge* e *ros2_ws_local*. Dentro de seus respectivos *workspaces*, os códigos do servidor de borda foram organizados num pacote chamado de *edge_pkg*, enquanto os locais foram organizados no pacote *local_pkg*. Após a escrita do código, esses pacotes são construídos utilizando-se o *colcon*².

²<https://colcon.readthedocs.io/en/released/>

Uma estratégia adotada para facilitar o acesso aos *workspaces* foi utilizar uma plataforma de armazenamento em nuvem, o *Dropbox*. O *Parallels*, automaticamente configura uma unidade de rede nas máquinas virtuais, que aponta para o diretório do *Dropbox* na máquina hospedeira. Com isso, é necessário apenas manter os *workspaces* dentro do *Dropbox*, que automaticamente eles podem ser acessados pelas máquinas virtuais. Esse processo está ilustrado na Figura 6.1 e vale para as máquinas virtuais de borda e locais.

Figura 6.1: Representação do acesso das máquinas virtuais aos *workspaces* do ROS 2.



Fonte: Produzida pelo autor.

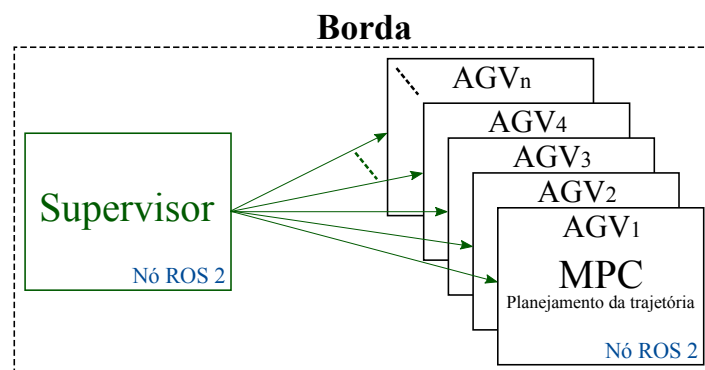
Para que o ROS 2 possa ser utilizado, é necessário executar no terminal o comando `source` na instalação da ferramenta. O mesmo deve ser feito no diretório do *workspace* para que os pacotes inseridos nele possam ser reconhecidos e executados por linha de comando no terminal. Esse processo pode ser automatizado pela inserção dos comandos do `source` no arquivo `~/.bashrc`, de modo que os comandos são executados sempre que uma janela do terminal é aberta. Na aplicação deste trabalho, as máquinas virtuais da borda fazem o `source` no *workspace* `ros2_ws_edge`, utilizando o caminho da unidade de rede que dá acesso ao *Dropbox*. O mesmo vale para as máquinas virtuais locais com o *workspace* `ros2_ws_local`.

6.3 Supervisor

Um agente supervisor foi adicionado no servidor de borda para supervisionar a navegação dos AGVs nos cenários de aplicação da arquitetura proposta, tal como feito por Demasure *et al.* [24], onde um supervisor gerencia conflitos, colisões e *deadlocks*. Criar ou avaliar estratégias de supervisão não está no escopo do trabalho, entretanto, a inclusão do supervisor facilita a avaliação da arquitetura em um determinado cenário de aplicação. Por exemplo, considere uma linha de produção de uma fábrica, em que os AGVs trazem materiais do estoque para a linha de produção ou carregam produtos semiacabados ou acabados. Nesse contexto, uma das tarefas do supervisor pode ser o gerenciamento da ocupação das vagas onde os AGVs podem estacionar para aguardar que a carga seja adicionada ou retirada do veículo, evitando que mais de um AGV tente estacionar ao mesmo tempo em uma vaga. Neste trabalho, o supervisor executa diferentes algoritmos de acordo com os cenários que serão explorados na Seção 6.6.

Na Figura 6.2 está representada a interação do Supervisor com a camada de MPC da arquitetura geral apresentada no capítulo anterior. Vale ressaltar que o funcionamento da arquitetura proposta não é dependente do Supervisor. Esse agente deve apenas supervisionar os AGVs de acordo com o cenário e interagir com a camada de MPC no servidor de borda. Desse modo, o Supervisor faz solicitações ao serviço do MPC sempre que for necessário mover um AGV.

Figura 6.2: Representação da interação do nó do Supervisor com os nós do MPC no servidor de borda.

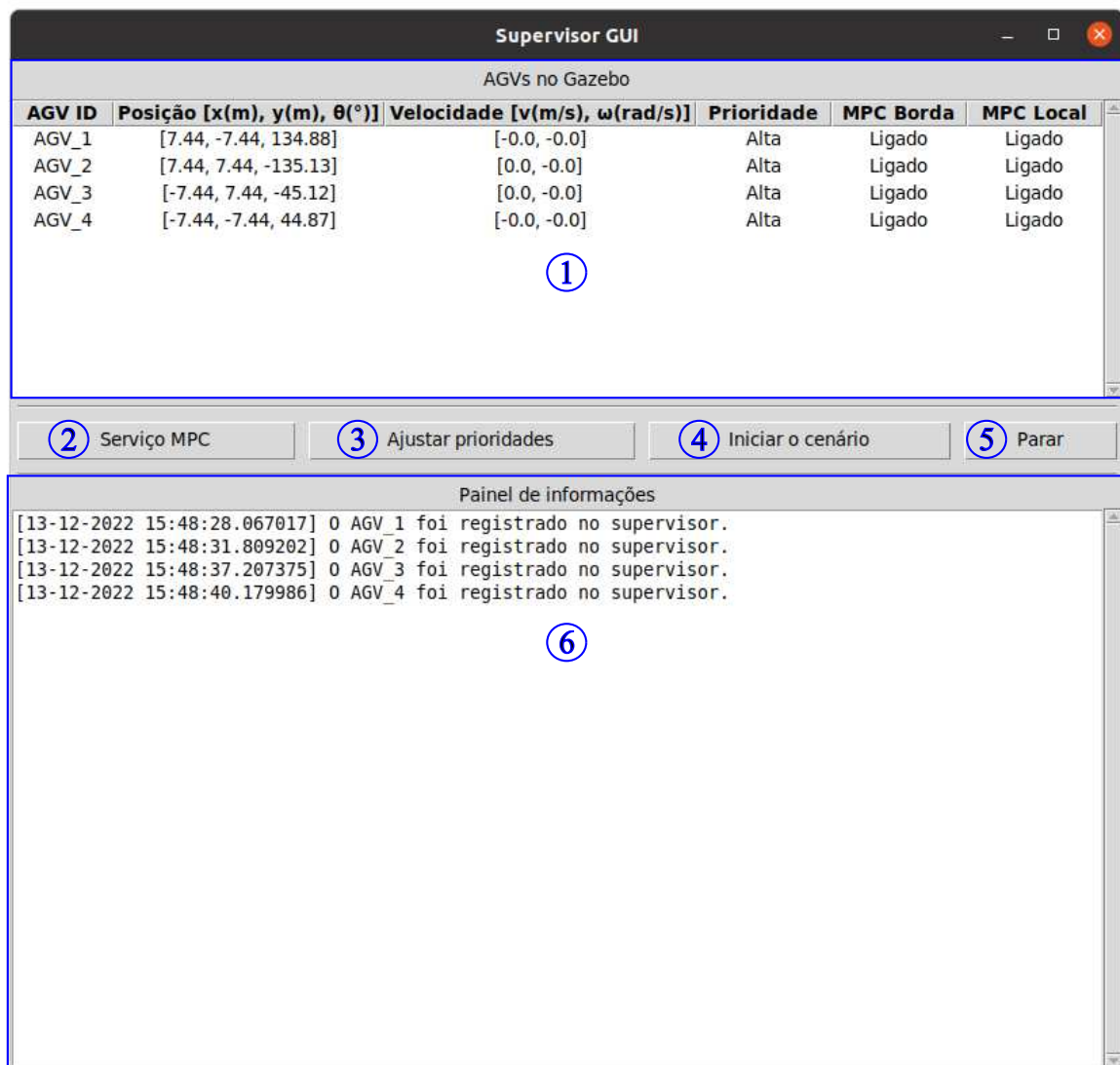


Fonte: Produzida pelo autor.

6.4 Ferramentas de simulação

Foi desenvolvida uma interface gráfica associada ao nó do Supervisor, que é aberta sempre que o nó é executado. A interface gráfica é utilizada para visualização dos estados dos AGVs e de seus MPCs, visualização de informações e botões para interação com a simulação. A interface utilizada em uma simulação com quatro AGVs é vista na Figura 6.3.

Figura 6.3: Imagem da interface gráfica do supervisor.



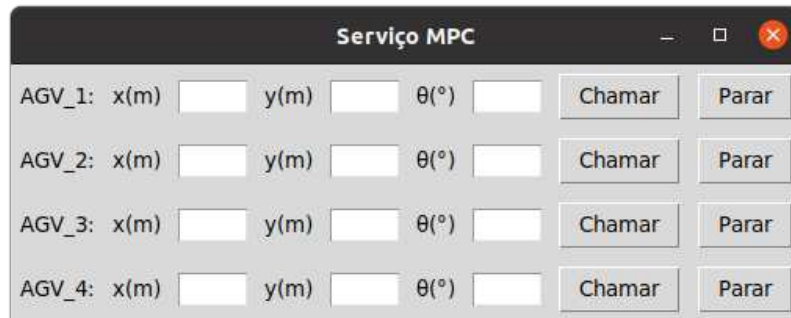
Fonte: Produzida pelo autor.

Seguindo a numeração presente na figura, cada parte da interface tem a seguinte função:

1. Tabela para visualização dos AGVs que estão no Gazebo, na coluna 'AGV_ID', junto com suas posições na coluna 'Posição', velocidade na coluna 'Velocidade', prioridade

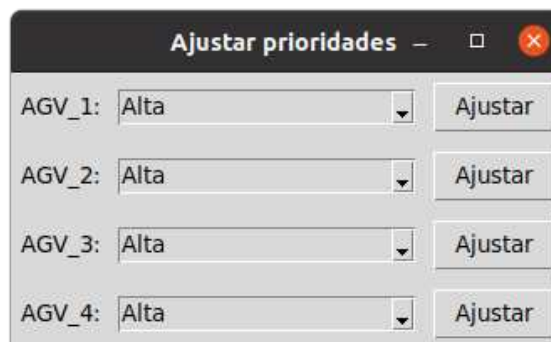
- na coluna ‘Prioridade’, estados dos nós dos MPCs da borda (planejamento) e locais (rastreamento) nas colunas ‘MPC Borda’ e ‘MPC Local’, respectivamente.
2. Botão utilizado para chamar o serviço de MPC de planejamento da trajetória no servidor de borda. Pode ser utilizado quando se deseja que um AGV se mova para uma determinada posição. Quando clicado, a janela vista na Figura 6.4 é aberta e são carregados os campos para preenchimento dos dados de posição para onde se deseja que o AGV se mova. Após inserir os dados de x e y em metros e de θ em graus, clica-se no botão ‘Chamar’ para chamar o serviço. Caso se deseje parar o AGV, clica-se no botão ‘Parar’.
 3. Botão utilizado para ajustar as prioridades dos AGVs. Após clicado, a janela vista na Figura 6.5 é aberta para que as prioridades sejam ajustadas individualmente. Neste trabalho, as prioridades são ‘Alta’, ‘Média’ ou ‘Baixa’ e apenas definem as velocidades dos AGVs. Um AGV que esteja executando uma tarefa que não tenha exigência de tempo para execução, por exemplo, pode ter sua prioridade ajustada para ‘Baixa’, de modo que poderá dar preferência na passagem a outros AGVs que têm prioridade maior, além de reduzir o consumo de energia.
 4. Botão utilizado para iniciar a simulação de um cenário.
 5. Botão utilizado para encerrar a simulação. Quando clicado, todos os AGVs param na posição que se encontram.
 6. Painel de informações onde são apresentadas as ações do Supervisor, como o registro de um AGV, chamadas do serviço do MPC e ajustes de prioridades.

Figura 6.4: Imagem da janela vinculada ao Supervisor para chamar o serviço de MPC individualmente para cada AGV presente no cenário.



Fonte: Produzida pelo autor.

Figura 6.5: Imagem da janela vinculada ao Supervisor para ajuste individual das prioridades de cada AGV presente no cenário.



Fonte: Produzida pelo autor.

O *NetEm*³ é utilizado para emular degradações do sinal da rede de comunicação, através da aplicação de atrasos e perdas de pacotes. Para uso do *NetEm* nas diversas máquinas virtuais utilizadas, foi criado um nó do ROS 2 que é executado junto aos MPCs da borda e aos MPCs locais. Esses nós são servidores de um serviço, os quais recebem os parâmetros de atraso e perda a serem aplicados nos pacotes que saem das interfaces de rede das máquinas virtuais. Para interagir com os serviços, foi desenvolvido um outro nó do ROS 2 com uma interface gráfica associada, vista na Figura 6.6. Os campos a serem preenchidos na interface são o atraso em milissegundos, a distribuição do atraso em milissegundos (a Distribuição Normal é utilizada) e a porcentagem da perda de pacotes. Caso se deseje adicionar apenas o atraso, é necessário preencher os campos de atraso e distribuição, preenchendo com o valor

³<https://wiki.linuxfoundation.org/networking/netem>

zero o campo das perdas. O inverso deve ser feito caso se deseje adicionar apenas perdas de pacotes. Após preencher os dados, clica-se no botão ‘Aplicar’, para que os serviços do *NetEm* da borda e locais sejam chamados. É possível também fazer com que o *NetEm* fique ativo apenas em um intervalo de tempo. Nesse caso, preenche-se o campo do tempo com os segundos em que o *NetEm* deve permanecer ativo e depois clica-se no botão ‘Aplicar c/ tempo’. O botão ‘Remover’ é utilizado para desabilitar o *Netem*.

Figura 6.6: Imagem da interface gráfica para utilização do *NetEm*.



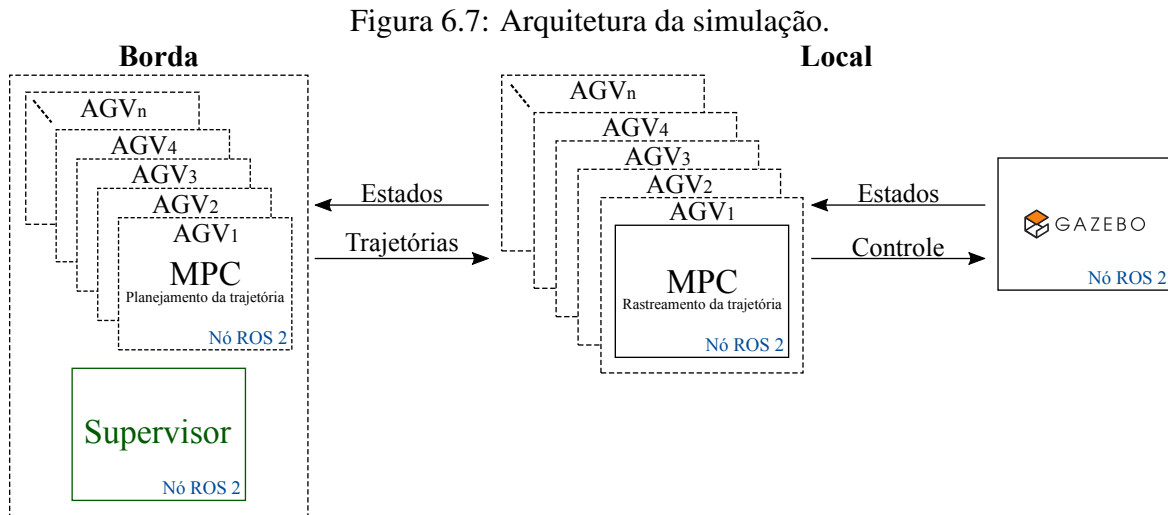
Fonte: Produzida pelo autor.

Vale observar que o *NetEm* é aplicado em todas as máquinas virtuais utilizadas para os pacotes que saem das interfaces de rede. Considere a aplicação de um atraso de 100 milissegundos com distribuição de 10 milissegundos e perda de pacotes de 10%. A saída de um pacote de uma máquina virtual local para o servidor de borda será então atrasada em torno de 100 milissegundos e tem chance de 10% de ser perdido, o mesmo acontecendo no sentido inverso. Para que as máquinas virtuais locais não tenham atrasos ou perdas em relação à máquina a qual executa o Gazebo, foi utilizado um filtro para que o *NetEm* seja aplicado apenas para os pacotes que têm o servidor de borda como destino.

6.5 Arquitetura da simulação

Na Figura 6.7 está apresentada a arquitetura utilizada nas simulações. Da mesma forma que na arquitetura geral da Seção 5.1, no servidor de borda existe um nó de MPC de planeja-

mento da trajetória para cada AGV. Esses nós enviam suas respectivas trajetórias para o lado local e recebem os estados dos respectivos AGVs. O Supervisor também foi adicionado na arquitetura da simulação, visto que o respectivo nó deve interagir com os nós de MPC nas simulações para atender às demandas do cenário.



Fonte: Produzida pelo autor.

No lado local cada AGV tem sua máquina virtual executando o nó de MPC de rastreamento da trajetória. Dado que não são utilizados AGVs reais nas simulações, todos os AGVs estão concentrados nos cenários criados no Gazebo, executado em outra máquina virtual. Portanto, os nós de MPC do lado local enviam para o Gazebo os sinais de controle e recebem os estados dos AGVs. Estes são depois repassados para o servidor de borda. O Gazebo também executa nós do ROS 2, através da habilitação do *plugin* do robô de tração diferencial. Esse *plugin* atua como um controlador de baixo nível e existe um para cada AGV no Gazebo. O *plugin* também se inscreve para receber os sinais de controle e publica os estados dos AGVs. A partir dos sinais de controle de velocidade linear e angular, o *plugin* determina o torque a ser aplicado em cada uma das rodas do AGV.

Em todos os cenários de simulação, os nós do MPC de planejamento da trajetória chamam o solucionador do CasADi a cada 100 milissegundos, através de um temporizador do ROS 2. Assim que o resultado é obtido, as trajetórias são publicadas e os nós se inscrevem para receberem as trajetórias uns dos outros, para que sejam utilizadas no procedimento de prevenção de colisões. Os nós do MPC de rastreamento da trajetória também se inscrevem para receber a trajetória do respectivo AGV. O solucionador desses nós locais também são

disparados a cada 100 milissegundos por um temporizador, de modo a rastrear a trajetória recebida do servidor de borda. Se o solucionador for chamado e uma trajetória mais recente não seja recebida (em caso de atrasos ou perdas de pacotes), o nó local do MPC continuará rastreando a última trajetória recebida. Nesse caso é feito um deslocamento das posições na lista dos vetores que formam a trajetória (ver Equação 5.3). O primeiro vetor da lista é então eliminado e o segundo é deslocado para a primeira posição, repetindo-se o último vetor para que o tamanho da lista seja preservado. Caso o solucionador seja chamado novamente e uma trajetória mais recente ainda não tenha sido recebida, esse procedimento é refeito.

6.6 Cenários de simulação

Para a implementação da arquitetura proposta em situações que emulem um cenário real de aplicação, foram criados quatro cenários de simulação no Gazebo. O primeiro é um cenário básico e não está vinculado a uma aplicação real, servindo apenas para a validação da arquitetura. Nele é possível verificar se as camadas utilizadas na arquitetura podem conduzir os AGVs de um estado inicial até a referência, evitando colisões entre si e com obstáculos fixos, inclusive na ocorrência de atraso ou perda de pacotes na rede. O segundo cenário representa uma aplicação de uma pequena linha de produção de uma fábrica, com uso dos AGVs para transporte de materiais. O terceiro representa um cenário logístico, sendo uma aplicação de maior proporção, com AGVs maiores em relação aos utilizados nos outros cenários, transportando materiais entre docas e armazéns. O quarto cenário, assim como o primeiro, não representa uma aplicação real, mas é empregado para avaliar a escalabilidade do sistema e o uso dos recursos computacionais quando a quantidade de AGVs é expandida.

A seguir, é apresentado como ocorre a criação dos mundos no Gazebo, os quais representam os cenários de aplicação, além da criação dos modelos de AGV utilizados.

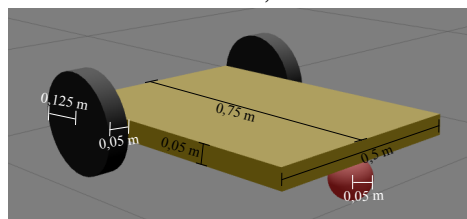
Criação dos mundos e dos AGVs no Gazebo

Um mundo no Gazebo contém uma coleção de robôs, objetos e parâmetros globais, os quais incluem o céu, luz ambiente e propriedades físicas. Os elementos do mundo são incluídos em um arquivo SDF (*Simulation Description Format*), escrito com a linguagem XML (*Extensible Markup Language*) e salvo com a extensão `.world`. A criação do mundo pode

ser iniciada no Gazebo e quaisquer alterações posteriores podem ser feitas diretamente no arquivo SDF. Neste trabalho, os mundos no Gazebo foram criados conforme o cenário de aplicação e incluem apenas os elementos estáticos, onde a partir do ROS 2, o Gazebo é iniciado com o mundo criado. Os elementos dinâmicos, aqui os AGVs, são incluídos posteriormente também com o ROS 2. Mais detalhes desse processo estão apresentados juntos com o Cenário 1.

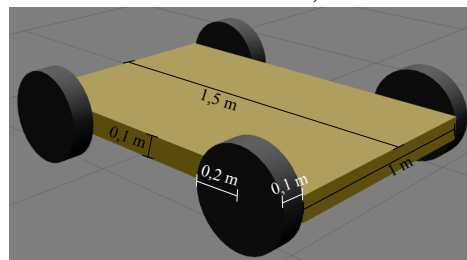
A criação do AGV foi realizada também no Gazebo, utilizando o seu editor de modelos. Nesse processo foram utilizadas formas geométricas básicas, como cubos, esferas e cilindros. O modelo do AGV é salvo no formato SDF (igual para a extensão) e também pode ser editado diretamente via código. Os Cenários 1, 2 e 4 utilizam um modelo de AGV menor, enquanto o Cenário 3 utiliza um AGV maior, já que representa uma aplicação de transporte de maiores cargas. Nas Figuras 6.8 e 6.9 estão apresentados o modelo menor e o maior, respectivamente, incluindo suas dimensões. O modelo menor é um robô de tração diferencial, nos mesmos moldes do apresentado na Seção 2.4, onde apenas as duas rodas traseiras têm tração. A roda dianteira é do tipo *caster*, com formato de esfera e serve apenas para guiar o robô na direção imposta pelo torque aplicado nas rodas traseiras. De forma diferente, o modelo maior contém quatro rodas tracionadas.

Figura 6.8: Modelo de AGV menor, utilizado nos Cenários 1, 2 e 4.



Fonte: Produzida pelo autor.

Figura 6.9: Modelo de AGV maior, utilizado no Cenário 3.



Fonte: Produzida pelo autor.

Após a criação do modelo, inclui-se manualmente no código as linhas que contêm os parâmetros do *plugin* do Gazebo para robôs de tração diferencial. O *plugin* utilizado é o `libgazebo_ros_diff_drive.so`. Alguns dos parâmetros informados são as juntas do robô que devem ter tração, a distância que separa as rodas, o diâmetro das rodas e o torque máximo. Quando o modelo é inserido no Gazebo, o *plugin* habilita um nó do ROS 2 que será responsável por publicar a odometria do robô e por se inscrever para receber os comandos de velocidade. O mesmo *plugin* atua como o controlador de baixo nível apresentado da arquitetura proposta apresentado na Figura 5.1.

6.6.1 Cenário 1

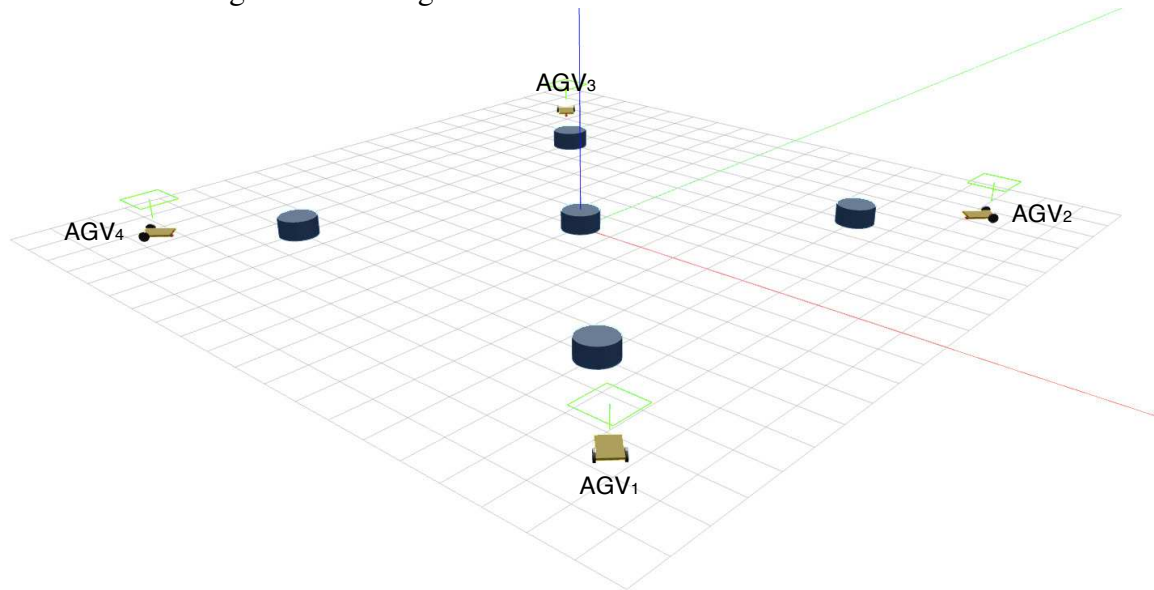
O Cenário 1 não está vinculado a um cenário real de uma aplicação e aqui serve para validação da arquitetura, tal como feito no trabalho publicado em [20]. Com a validação, tem-se o objetivo de verificar se a arquitetura possibilita a navegação dos AGVs, sem colisões com os obstáculos fixos e com os AGVs entre si, mesmo quando houver degradação do sinal da rede de comunicação.

No cenário os AGVs devem sair de um estado inicial até a referência, evitando colisões entre si e com obstáculos fixos. Também é utilizado aqui para demonstrar o passo a passo do processo de execução das simulações no ROS 2 e no Gazebo. Este cenário é o mesmo utilizado na demonstração com o ROS 2 na Seção 5.4, entretanto, os sinais de controle gerados pelo MPC de rastreamento da trajetória no lado local são aplicados nos AGVs de um mundo criado no Gazebo, contendo os quatro veículos e os 5 obstáculos fixos. O mundo criado para este cenário está apresentado na Figura 6.10.

Execução dos nós do ROS 2

Conforme visto anteriormente, os nós do ROS 2 são executados por meio de pacotes, os quais estão inseridos em *workspaces*. Na forma mais simples, após construídos, os pacotes são executados por linha de comando no terminal. A forma mais viável de executar aplicações que utilizam diversos nós é a utilização dos *launch files*. Com esses arquivos é possível executar diversos nós com apenas um comando e inclusive passar parâmetros para esses nós no momento das suas inicializações. Para fins de organização, neste trabalho foram criados

Figura 6.10: Imagem do mundo do Cenário 1 no Gazebo.



Fonte: Produzida pelo autor.

pacotes por cenário, exclusivos para os *launch files*. O Python foi utilizado para criação dos *launch files*, embora possam também serem escritos em XML ou YAML (*YAML Ain't Markup Language*).

A inicialização dos nós nos lados da borda e local são realizadas de modo independente e não existe uma ordem para isso. No servidor de borda, é necessário executar o nó do Supervisor e os nós dos MPCs para planejamento da trajetória. Destes, quatro nós são executados, dado que existem quatro AGVs no cenário. O mesmo *launch file* também executa um nó utilizado no servidor de borda para aplicação de atraso e perda de pacotes com o *NetEm*. Mais detalhes da utilização desse nó podem ser vistos adiante. Após a execução dos nós, algumas informações são exibidas nos terminais, sendo algumas delas mensagens do próprio ROS 2 e outras que podem ser inseridas via código do nó para exibição.

No lado local, na máquina virtual exclusiva do Gazebo, é executado um *launch file* que faz a abertura do Gazebo com o mundo criado para o cenário, além de que o mesmo *launch file* executa um nó que está incluído no pacote *spawn_pkg*. Esse nó, por sua vez, chama o serviço */SpawnEntity*, o qual é um serviço do ROS 2 em que o Gazebo é o servidor. Nas simulações o serviço é utilizado para lançar no mundo do Gazebo todos os AGVs necessários para o cenário. O nó responsável pela função é finalizado após o lançamento de todos os AGVs.

Ainda no lado local, utilizam-se quatro máquinas virtuais, onde cada uma delas está vinculada a um AGV e executa o nó de MPC para rastreamento da trajetória. Para facilitar a execução dos nós locais, as máquinas virtuais são acessadas pela máquina virtual do Gazebo, através do protocolo SSH (*Secure Shell*).

Vale ressaltar que além de os *launch files* possibilitarem a execução de vários nós em uma única linha de comando, o repasse de parâmetros para os nós permite que apenas um código que realiza uma determinada função seja mantido. Na execução do Cenário 1 no servidor de borda, por exemplo, o *launch file* correspondente apontará quatro vezes para o código referente ao MPC de planejamento de trajetória, mas com repasse de diferentes parâmetros para cada nó executado. Um dos parâmetros repassados aos nós são os identificadores dos AGVs, que no cenário corrente são as palavras ‘AGV_1’, ‘AGV_2’, ‘AGV_3’ e ‘AGV_4’. Essa abordagem facilita o desenvolvimento e os ajustes no projeto, dado que a alteração em um único código resultará em mudanças em todos os nós que o utilizarão.

Nós, Tópicos e Serviços

Para cada AGV no cenário existem quatro nós associados. Tomando o AGV₁ como exemplo, tem-se:

- */agv_1/differential_drive_controller*: nó associado ao *plugin libgazebo_ros_diff_drive.so*;
- */agv_1/mpc*: nó do MPC de planejamento da trajetória no servidor de borda;
- */agv_1/mpc_local_tracking*: nó do MPC de rastreamento da trajetória no lado local;
- */agv_1/netem_node*: nó para aplicação dos comandos do *NetEm* na máquina virtual associada ao AGV.

Existem ainda o nó */gazebo* associado ao Gazebo, o nó */netem_gui_node* da interface gráfica para aplicação do *NetEm* e o nó */supervisor_node* referente ao Supervisor.

Continuando com o AGV₁, tem-se os seguintes tópicos associados:

- */agv_1/cmd_vel*: tópico criado pelo nó */agv_1/differential_drive_controller*, o qual se inscreve para receber os comandos de velocidade a serem aplicados no AGV. O nó */agv_1/mpc_local_tracking* é responsável por publicar esses comandos.

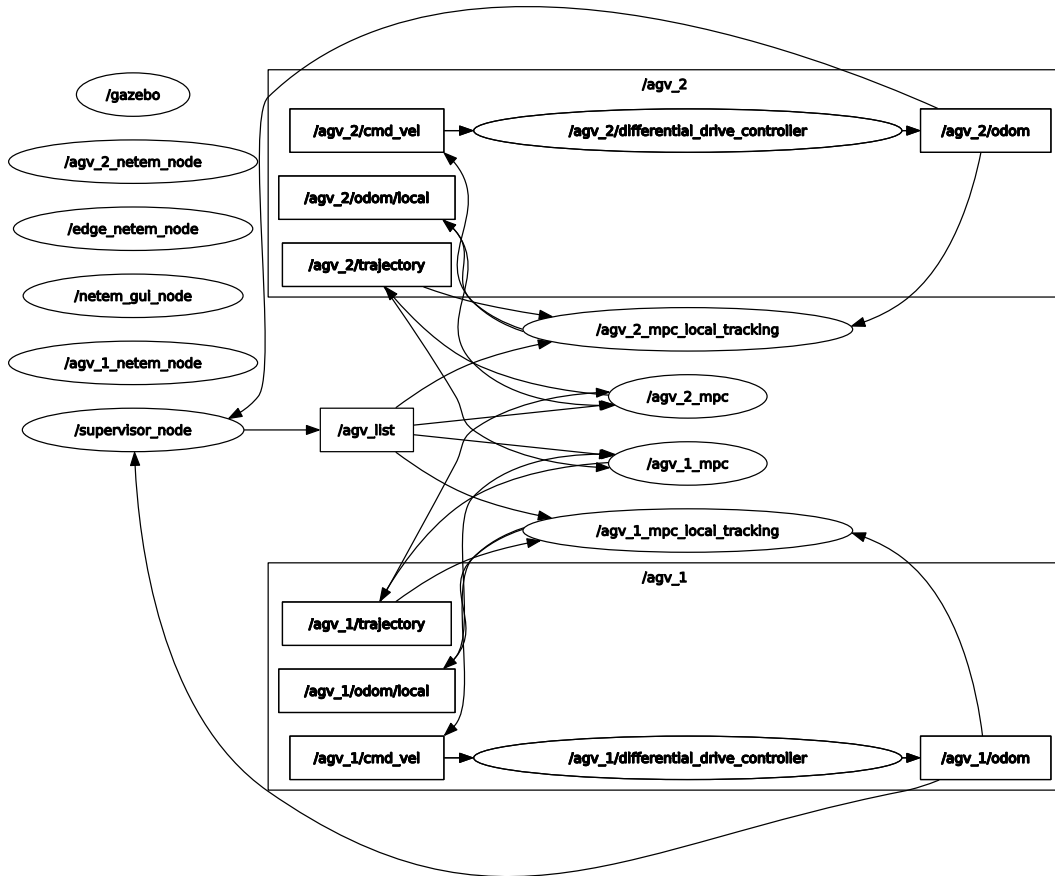
- */agv_1/odom*: também gerado pelo mesmo nó citado acima e é por onde o nó */agv_1/mpc_local_tracking* se inscreve para receber os dados de odometria do AGV.
- */agv_1/odom/local*: gerado pelo nó */agv_1/mpc_local_tracking* para publicação dos dados de odometria, onde o nó */agv_1/mpc* no servidor de borda se inscreve para receber. Esse desvio da odometria por esse tópico é implementado para que quando o *NetEm* for aplicado, os pacotes que circulam dos nós locais para o servidor de borda sofram a intervenção. Caso o nó de borda se inscrevesse para receber diretamente no tópico */agv_1/odom*, não existiria nenhuma intervenção, visto que não há aplicação do *NetEm* na máquina virtual do Gazebo.
- */agv_1/trajjectory*: gerado pelo nó */agv_1/mpc* no servidor de borda para publicação da trajetória. O nó */agv_1/mpc_local_tracking* no lado local se inscreve para receber a trajetória. Os nós de MPC dos outros AGVs no servidor de borda também se inscrevem para receber a trajetória, a fim de utilizá-la no procedimento de prevenção de colisões.
- */agv_list*: tópico criado pelo nó do Supervisor, sendo responsável por publicar uma lista com todos os AGVs que estão ativos. Os nós de MPC do servidor de borda e locais se inscrevem para receber essa lista.

Existem ainda outros tópicos não citados e que são de uso interno do ROS 2.

Através do comando `rqt_graph` no terminal, o ROS 2 cria uma representação gráfica dos nós e tópicos que estão em execução, incluindo setas que indicam o fluxo das informações através dos tópicos. Executando o comando para o Cenário 1, a representação é extensa e se torna inviável para inclusão neste documento. O comando foi então executado mantendo-se ativos apenas os nós referentes aos AGVs 1 e 2, o que resulta na ilustração da Figura 6.11.

Na Figura 6.11, os nós estão representados por elipses e os tópicos por retângulos. Toda a interação entre os nós e tópicos explicadas anteriormente pode ser vista. Embora os nós dos AGVs 3 e 4 não estejam na representação gráfica, todo o processo acontece da mesma forma. Os nós de MPC (borda e local) dos AGVs 3 e 4, por exemplo, também se inscrevem para receber a lista de AGVs publicada pelo Supervisor. Os nós de MPC do servidor de borda ainda se inscrevem para receber as trajetórias dos outros AGVs, de modo a utilizá-las no procedimento de prevenção de colisões.

Figura 6.11: Representação gráfica dos nós e tópicos ativos referentes aos AGVs 1 e 2.

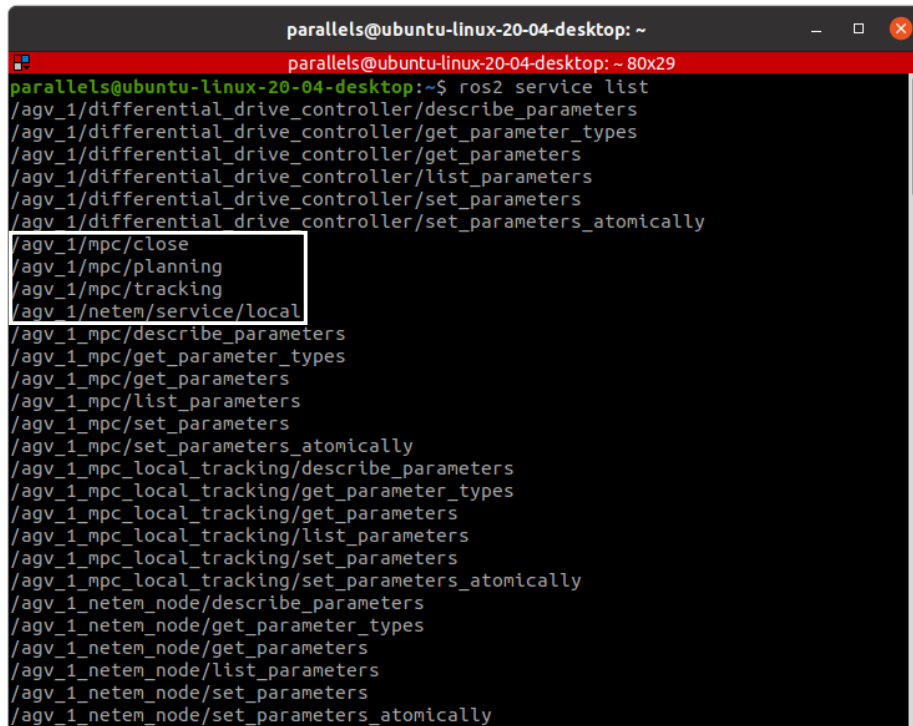


Fonte: Produzida pelo autor.

Por fim, podemos visualizar também no terminal a lista de serviços disponíveis. Na Figura 6.12 está a lista parcial resultante da execução do comando `ros2 service list`. Na mesma figura estão destacados alguns serviços vinculados ao AGV₁. Estes serviços, são:

- `/agv_1/mpc/close`: o nó do MPC de planejamento da trajetória é o servidor e tem como o cliente o nó de MPC de rastreamento da trajetória no lado local. Quando o AGV alcança a referência, o serviço é chamado pelo nó local para que o nó do servidor de borda possa encerrar as iterações do MPC.
- `/agv_1/mpc/planning`: o nó do MPC de planejamento da trajetória também é o servidor e tem como o cliente o nó do Supervisor. Este último chama o serviço para que o MPC do servidor de borda possa gerar as trajetórias sempre que for necessário mover o AGV.
- `/agv_1/mpc/tracking`: é um serviço vinculado ao nó do MPC de rastreamento da trajetória no lado local. Sempre que o serviço anterior é chamado, o nó de MPC no servidor

Figura 6.12: Imagem da tela do terminal com a lista de serviços do ROS 2 do Cenário 1, vinculados ao AGV₁.



```
parallels@ubuntu-linux-20-04-desktop: ~
parallels@ubuntu-linux-20-04-desktop: ~ 80x29
parallels@ubuntu-linux-20-04-desktop:~$ ros2 service list
/agv_1/differential_drive_controller/describe_parameters
/agv_1/differential_drive_controller/get_parameter_types
/agv_1/differential_drive_controller/get_parameters
/agv_1/differential_drive_controller/list_parameters
/agv_1/differential_drive_controller/set_parameters
/agv_1/differential_drive_controller/set_parameters_atomically
/agv_1/mpc/close
/agv_1/mpc/planning
/agv_1/mpc/tracking
/agv_1/netem/service/local
/agv_1_mpc/describe_parameters
/agv_1_mpc/get_parameter_types
/agv_1_mpc/get_parameters
/agv_1_mpc/list_parameters
/agv_1_mpc/set_parameters
/agv_1_mpc/set_parameters_atomically
/agv_1_mpc_local_tracking/describe_parameters
/agv_1_mpc_local_tracking/get_parameter_types
/agv_1_mpc_local_tracking/get_parameters
/agv_1_mpc_local_tracking/list_parameters
/agv_1_mpc_local_tracking/set_parameters
/agv_1_mpc_local_tracking/set_parameters_atomically
/agv_1_netem_node/describe_parameters
/agv_1_netem_node/get_parameter_types
/agv_1_netem_node/get_parameters
/agv_1_netem_node/list_parameters
/agv_1_netem_node/set_parameters
/agv_1_netem_node/set_parameters_atomically
```

Fonte: Produzida pelo autor.

de borda chama o serviço de rastreamento da trajetória para que o rastreamento seja iniciado e o AGV siga na trajetória planejada.

- */agv_1/netem/service/local*: é servido pelo nó do *NetEm* vinculado ao AGV. O cliente é o nó vinculado à interface gráfica do *NetEm*, o qual faz a aquisição dos dados inseridos na interface e chama o serviço para que os comandos do *NetEm* sejam executados na máquina virtual vinculada ao AGV.

Na Figura 6.13 está apresentado outro trecho da lista de serviços.

Figura 6.13: Imagem da tela do terminal com outros serviços vinculados ao Cenário 1.

```

parallels@ubuntu-linux-20-04-desktop: ~
parallels@ubuntu-linux-20-04-desktop: ~ 80x35
/delete_entity
/edge_netem_node/describe_parameters
/edge_netem_node/get_parameter_types
/edge_netem_node/get_parameters
/edge_netem_node/list_parameters
/edge_netem_node/set_parameters
/edge_netem_node/set_parameters_atomically
/gazebo/describe_parameters
/gazebo/get_parameter_types
/gazebo/get_parameters
/gazebo/list_parameters
/gazebo/set_parameters
/gazebo/set_parameters_atomically
/get_model_list
/netem/service/edge
/netem_gui_node/describe_parameters
/netem_gui_node/get_parameter_types
/netem_gui_node/get_parameters
/netem_gui_node/list_parameters
/netem_gui_node/set_parameters
/netem_gui_node/set_parameters_atomically
/pause_physics
/register_agv
/reset_simulation
/reset_world
/spawn_entity
/start_stop_process
/supervisor_node/describe_parameters
/supervisor_node/get_parameter_types
/supervisor_node/get_parameters
/supervisor_node/list_parameters
/supervisor_node/set_parameters
/supervisor_node/set_parameters_atomically
/unpause_physics
parallels@ubuntu-linux-20-04-desktop:~$

```

Fonte: Produzida pelo autor.

Os serviços em destaque na figura, são:

- */agv_1/netem/service/edge*: igual ao serviço */agv_1/netem/service/local*, mas tem como servidor a borda e como cliente a interface gráfica do *NetEm*, de modo que os comandos sejam também aplicados na máquina virtual do servidor de borda.
- */register_agv*: tem o nó do Supervisor como servidor e o nó de MPC de planejamento da trajetória como cliente. Sempre que o AGV é inserido no Gazebo e inicia a publicação de sua odometria, o nó de MPC solicita o registro do AGV no Supervisor através do serviço.
- */spawn_entity*: serviço em que o Gazebo é o servidor e é utilizado para inserção de uma entidade no Gazebo. O cliente é um nó vinculado ao cenário a ser simulado e chama o serviço para inserção dos AGVs necessários para a simulação.

Resultados

Nesse cenário foram realizadas três simulações, cada uma com diferentes condições na rede de comunicação entre a borda e as máquinas virtuais locais. As condições são impostas por diferentes perfis de rede identificados por (a) ‘Sem atraso’, (b) ‘300ms’ e (c) ‘200msPL50%’. No perfil ‘Sem atraso’ nenhum atraso e perda de pacotes são aplicados. Nessa situação, o tempo para que um pacote deixe um dos lados, alcance o outro e retorne, medido através do comando `ping`, tem em torno de 1 milissegundo. No perfil ‘300ms’ a interface gráfica do *NetEm* é utilizada para induzir um atraso de 300 milissegundos com desvio de 30 milissegundos em Distribuição Normal, enquanto no perfil ‘200msPL50%’ um atraso de 200 milissegundos com desvio de 20 milissegundos em Distribuição Normal é somado à perda de pacotes numa taxa de 50%. Nos dois últimos perfis as condições são impostas aos pacotes que deixam as interfaces de rede em ambos os lados.

Os estados iniciais e referências dos AGVs, bem como as posições dos obstáculos, são as mesmas das Tabelas 5.1 e 5.2, respectivamente. Na Tabela 6.3 estão listados os parâmetros utilizados no MPC de planejamento da trajetória. Os parâmetros do MPC de rastreamento da trajetória são os mesmos, sem a inclusão do diâmetro do AGV e da distância de segurança, parâmetros estes utilizados apenas no servidor de borda nas restrições de prevenção de colisões. Os pesos Q e R nesse MPC também são diferentes e ajustados conforme as matrizes da Tabela 6.4.

Tabela 6.3: Parâmetros configurados no MPC de planejamento da trajetória nas simulações do Cenário 1.

T	100 ms
N	60
Q	$\text{diag}[5 \ 5 \ 0,5]$
R	$\text{diag}[0,5 \ 0,5]$
u_v^{min}	- 1,0 m/s
u_v^{max}	+ 1,0 m/s
u_ω^{min}	- $\pi/4$ rad/s
u_ω^{max}	+ $\pi/4$ rad/s
$diam_{AGV}$	0,75 m
d_{seg}^{obs}	1,0 m
d_{seg}^{agv}	1,0 m

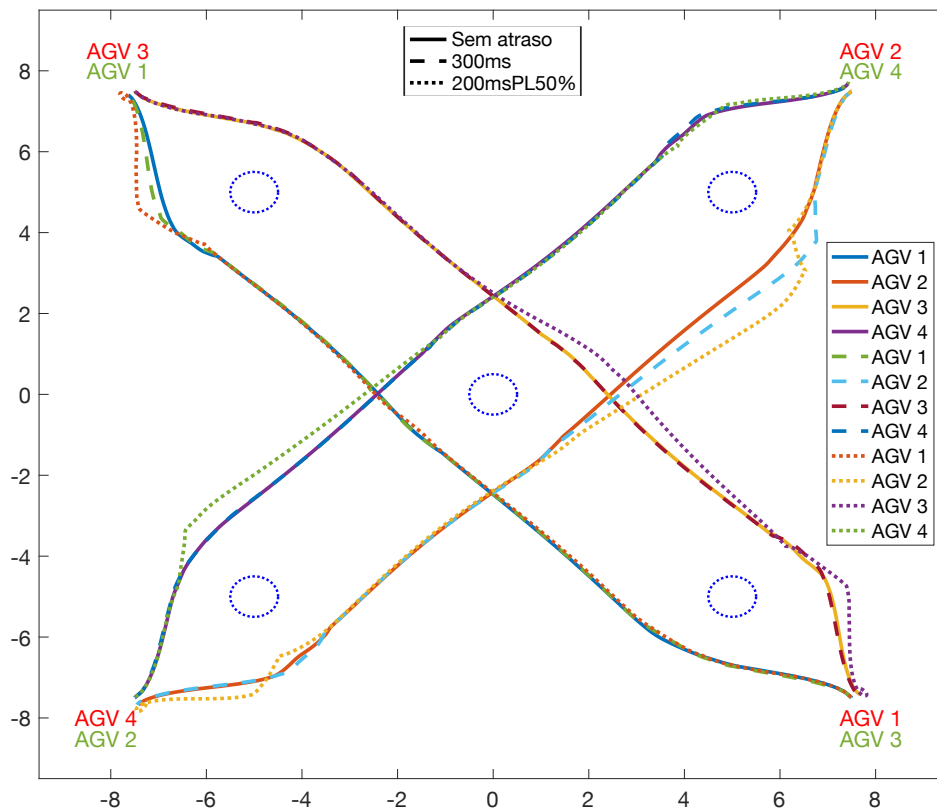
Tabela 6.4: Pesos Q e R ajustados no MPC de rastreamento da trajetória nas simulações do Cenário 1.

Q	$\text{diag}[10 \ 10 \ 20]$
R	$\text{diag}[10 \ 20]$

Os códigos dos nós executados foram programados para armazenar e salvar os dados da simulação em arquivos texto. Os arquivos são posteriormente carregados no Matlab para geração dos gráficos. Na Figura 6.14 está ilustrado o gráfico com as trajetórias percorridas pelos AGVs, resultantes das simulações com todos os perfis de rede, após serem iniciadas através da interface gráfica do Supervisor. As trajetórias ilustradas são formadas a partir dos dados de posição dos AGVs obtidos dos tópicos de odometria. Cada quina do gráfico tem a identificação do AGV que sai de sua posição inicial, em vermelho, e a identificação do AGV que chega em sua posição final, em verde. As trajetórias em linha contínua são referentes à

simulação do perfil ‘Sem atraso’, as com linhas tracejadas do perfil ‘300ms’ e as com linhas pontilhadas do perfil ‘200msPL50%’, conforme a legenda no canto superior da figura. Na legenda do lado direito é possível verificar as cores correspondentes às trajetórias dos AGVs em cada um dos perfis de rede.

Figura 6.14: Trajetórias dos AGVs nas simulações com o Cenário 1 nos perfis de rede ‘Sem atraso’, ‘300ms’ e ‘200msPL50%’.



Fonte: Produzida pelo autor.

A gravação da tela da máquina virtual que executa o Gazebo, durante a execução das simulações com o Cenário 1, pode ser vista no vídeo em <https://youtu.be/fRGGw2nOpAU>. No caso da simulação com o perfil ‘200msPL50%’, os parâmetros do *NetEm* são inseridos na interface gráfica depois da inicialização da simulação. Ainda nesse perfil, o *NetEm* é ajustado para se manter ativo durante trinta segundos. Com esses procedimentos evita-se que as perdas de pacotes interfiram nas chamadas de serviço.

Os demais resultados extraídos das simulações incluem os sinais de controle resultantes

do MPC de rastreamento da trajetória e aplicados aos AGVs, os intervalos de tempo em milissegundos das iterações dos MPCs no servidor de borda e local (planejamento e rastreamento da trajetória, respectivamente) e as porcentagens de uso de CPU (*Central Processing Unit*) e memória pelos processos referentes aos nós da borda e local. O intervalo de tempo das iterações do MPC é calculado pela diferença entre o instante de tempo em que o solucionador do CasADi é chamado para o cálculo da solução e o instante em que a solução é obtida. Já os dados associados ao uso dos recursos computacionais foram extraídos em intervalos de um segundo com o módulo *psutil*⁴, desenvolvido para Python. Para o caso do uso de CPU, o valor obtido é a soma das porcentagens de utilização de todos os núcleos do processador. Desse modo, os valores obtidos foram divididos pelo número de núcleos utilizados na máquina virtual em questão.

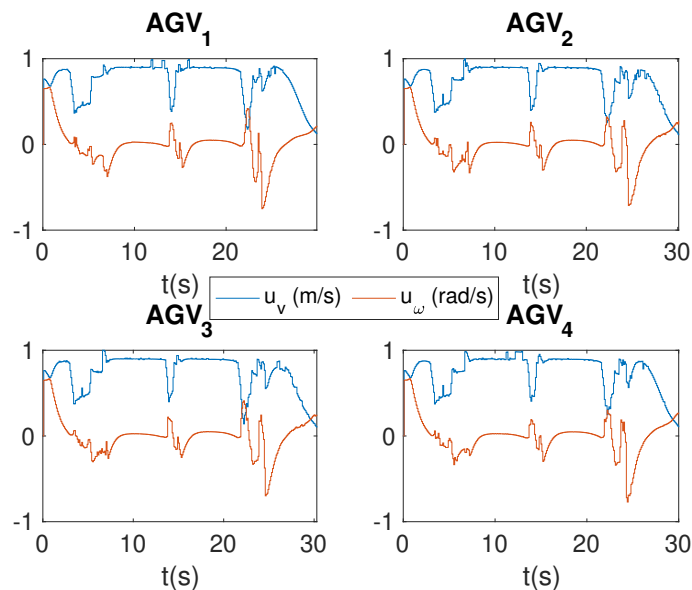
Na Tabela 6.5 estão listados os valores médios das métricas do intervalo de tempo das iterações do MPC, uso de CPU e uso de memória. Os gráficos em função do tempo dos sinais de controle e das métricas citadas, estão distribuídos entre as Figuras 6.15 e 6.17 para a simulação com o perfil ‘Sem atraso’, entre as Figuras 6.18 e 6.20 para a simulação com o perfil ‘300ms’ e entre as Figuras 6.21 e 6.23 para a simulação com o perfil ‘200msPL50%’. Ao analisar a tabela e os gráficos, vale lembrar que o servidor de borda é representado por uma única máquina virtual, ou seja, os nós de MPC na borda disputam os mesmos recursos. Por outro lado, as máquinas virtuais locais são independentes e cada uma executa o nó de MPC para o AGV que está vinculado a ela.

⁴<https://psutil.readthedocs.io/en/latest/>

Tabela 6.5: Médias do intervalo de tempo das iterações do MPC, do uso de CPU e do uso de memória nas simulações do Cenário 1.

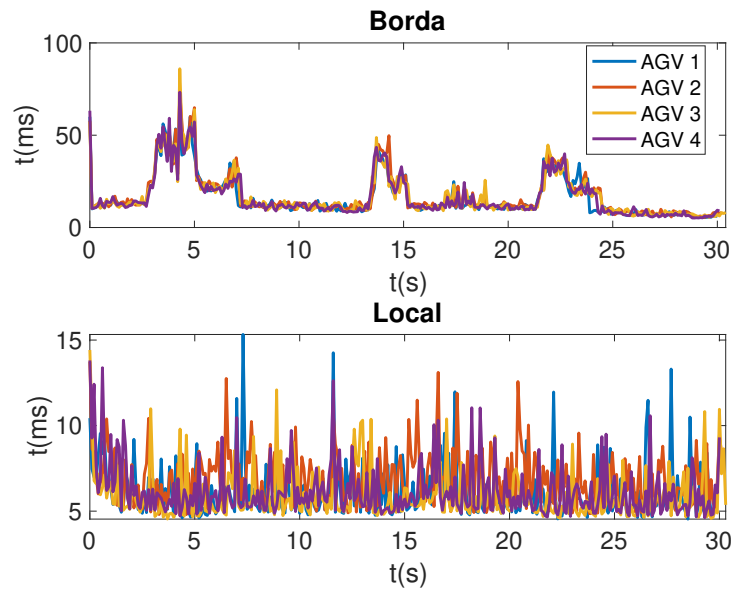
Métrica	AGV	Perfil de rede					
		Sem atraso		300ms		200msPL50%	
		Borda	Local	Borda	Local	Borda	Local
Intervalo de tempo das iterações do MPC (ms)	AGV 1	16,39	6,07	15,93	5,58	15,81	5,84
	AGV 2	17,42	6,88	14,54	5,33	14,25	6,07
	AGV 3	16,88	5,84	16,36	5,36	15,72	6,68
	AGV 4	16,51	6,15	16,60	5,35	14,84	6,71
Uso de CPU (%)	AGV 1	9,11	18,16	8,74	16,84	8,73	16,73
	AGV 2	8,74	18,87	8,43	16,79	7,82	17,20
	AGV 3	9,03	17,87	8,66	16,39	8,53	17,75
	AGV 4	9,03	17,93	8,71	16,55	8,36	17,82
Uso de memória (%)	AGV 1	2,81	16,01	2,84	16,84	2,84	16,07
	AGV 2	2,81	16,03	2,83	16,79	2,83	16,03
	AGV 3	2,81	15,93	2,84	16,39	2,83	16,01
	AGV 4	2,80	15,93	2,81	16,55	2,83	16,04

Figura 6.15: Sinais de controle para o Cenário 1 na simulação com o perfil ‘Sem atraso’.



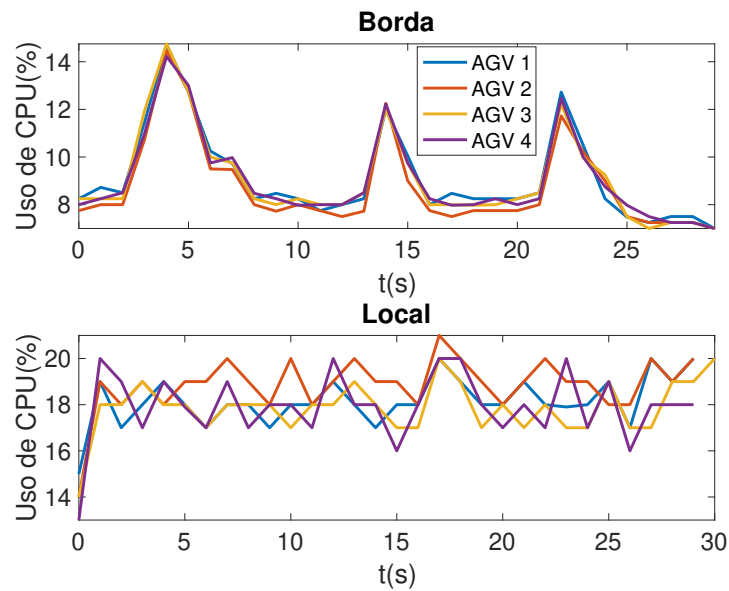
Fonte: Produzida pelo autor.

Figura 6.16: Intervalo de tempo das iterações do MPC para o Cenário 1 na simulação com o perfil ‘Sem atraso’.



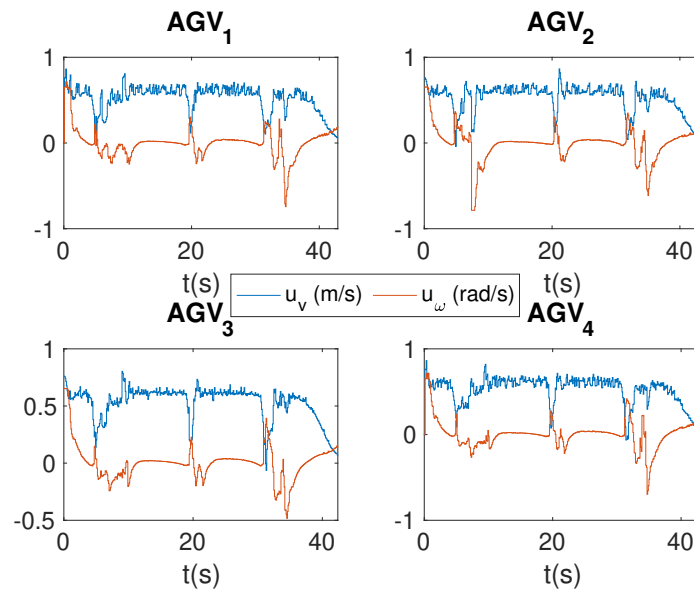
Fonte: Produzida pelo autor.

Figura 6.17: Uso de CPU e memória para o Cenário 1 na simulação com o perfil ‘Sem atraso’.



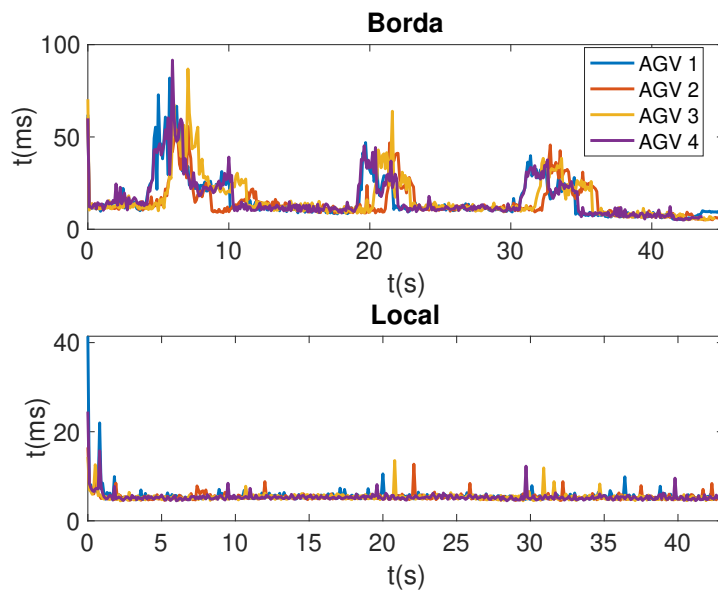
Fonte: Produzida pelo autor.

Figura 6.18: Sinais de controle para o Cenário 1 na simulação com o perfil '300ms'.



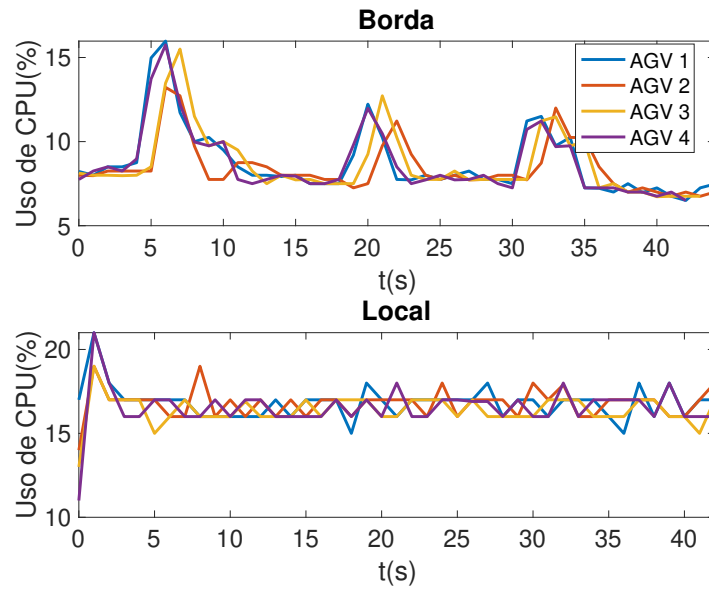
Fonte: Produzida pelo autor.

Figura 6.19: Intervalo de tempo das iterações do MPC para o Cenário 1 na simulação com o perfil '300ms'.



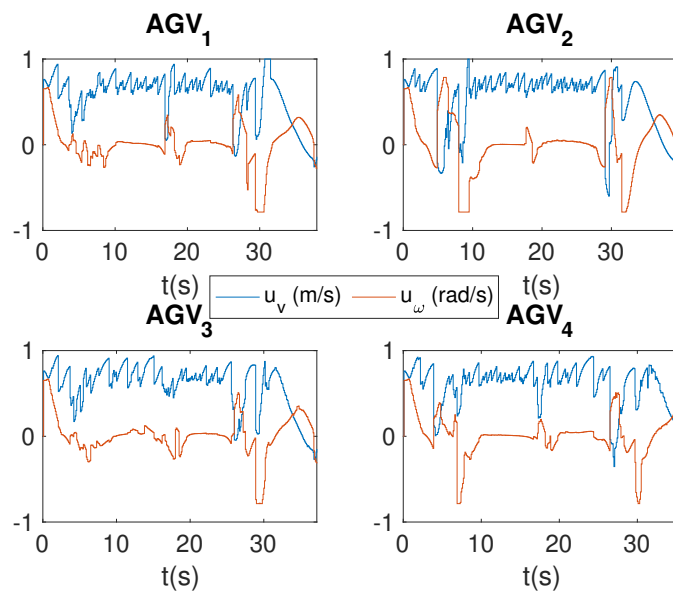
Fonte: Produzida pelo autor.

Figura 6.20: Uso de CPU e memória para o Cenário 1 na simulação com o perfil '300ms'.



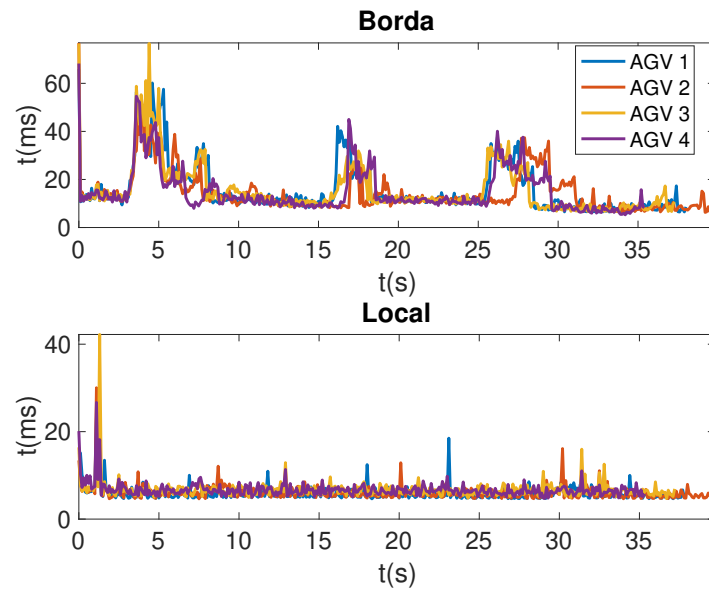
Fonte: Produzida pelo autor.

Figura 6.21: Sinais de controle para o Cenário 1 na simulação com o perfil '200msPL50%'.



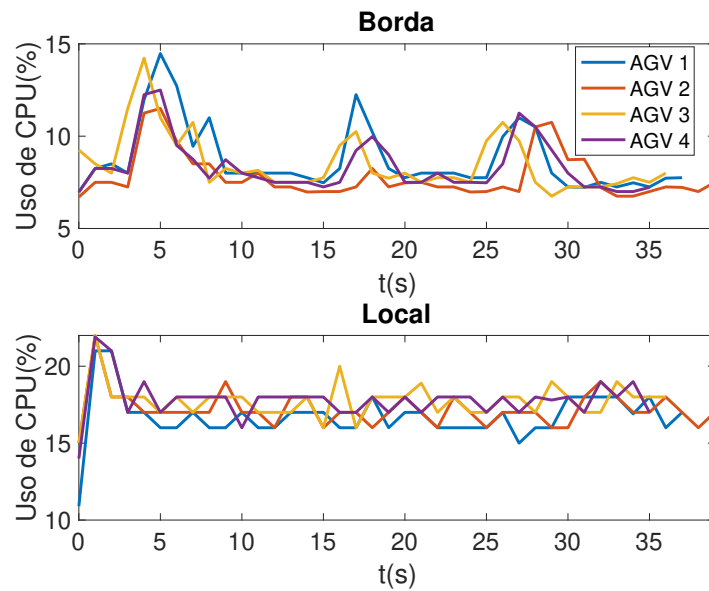
Fonte: Produzida pelo autor.

Figura 6.22: Intervalo de tempo das iterações do MPC para o Cenário 1 na simulação com o perfil '200msPL50%'.



Fonte: Produzida pelo autor.

Figura 6.23: Uso de CPU e memória para o Cenário 1 na simulação com o perfil '200msPL50%'.



Fonte: Produzida pelo autor.

Discussão

A partir das trajetórias vistas na Figura 6.14 e do vídeo com endereço disponibilizado anteriormente, verifica-se que a arquitetura com a dupla camada de MPCs conduz os AGVs até as suas referências, prevenindo colisões, inclusive nos perfis de rede com atraso e perdas de pacotes, tendo em vista as ações preditivas dos nós de MPC de planejamento da trajetória no servidor de borda. A partir das métricas apresentadas nos gráficos e das suas respectivas médias listadas na Tabela 6.5, verifica-se também que dado os recursos computacionais utilizados, o intervalo de tempo das iterações do MPC, nos lados da borda e local, são suficientes para uma execução em tempo real, visto que as iterações são disparadas a cada 100 milissegundos, mas, levam um tempo significativamente menor para serem executadas. Quando analisados o uso dos recursos computacionais, constata-se que os nós executados nos lados da borda e local não utilizam a capacidade total de CPU e memória das máquinas virtuais, não comprometendo todo o recurso e dando espaço para execução de outras tarefas.

Ainda que os AGVs naveguem em sentidos diferentes, o padrão de trajetória que cada um deve seguir é o mesmo, ou seja, os veículos devem percorrer distâncias equivalentes, desviar de três obstáculos fixos no caminho e ao final devem estabilizar nas referências. Por esse motivo, verifica-se nas Figuras 6.15, 6.18 e 6.21 que, nas simulações de cada perfil, o comportamento dos sinais de controle entre os AGVs é similar, onde nos instantes de desvio e estabilização os sinais das velocidades linear e angular são adequadamente ajustados. Outro fator que pode ser observado no vídeo e nos gráficos dos sinais de controle das Figuras 6.18 e 6.21, é a diminuição e a oscilação da velocidade linear dos AGVs nas simulações dos perfis '300ms' e '200msPL50%', respectivamente. Isso acontece devido ao atraso na recepção da posição do AGV no servidor de borda, o que leva a trajetória a ser planejada a partir dessa posição. Assim, quando essa trajetória é recebida no lado local, o AGV apresenta um desvio em relação àquela posição considerada no planejamento, levando o MPC de rastreamento a fazer constantes correções na trajetória. O problema pode ser considerado em trabalhos futuros, de modo que os AGVs possam manter suas velocidades mesmo com a degradação do sinal da rede.

Analisando os gráficos das Figuras 6.16, 6.19 e 6.22, vê-se que as iterações do MPC de planejamento da trajetória no servidor de borda têm maior intervalo de tempo de execução durante o desvio dos obstáculos fixos, o que levam essa métrica a ter três picos durante a

simulação, correspondendo aos desvios dos três obstáculos no trajeto, por AGV. Nos gráficos do uso de CPU na borda nas Figuras 6.17, 6.20 e 6.23, vê-se que existe uma relação entre o intervalo de tempo de execução do MPC e o uso de CPU, uma vez que os picos em ambas as métricas acontecem nos mesmos instantes da simulação. Nas simulações com os perfis ‘300ms’ e ‘200msPL50%’, esses picos acontecem em diferentes instantes, para cada AGV, dado que os veículos se aproximam dos obstáculos também em diferentes instantes da simulação.

Para o MPC de rastreamento da trajetória executado no lado local, os intervalos de tempo das iterações têm pequenas oscilações durante a simulação, vistos nos gráficos das Figuras 6.16, 6.19 e 6.22. Isso se deve ao fato de que os nós de MPC do lado local solucionam um problema mais simples, já que não deverão lidar com a prevenção de colisões e apenas rastrearão a trajetória de tamanho fixo recebidas do servidor de borda. O uso de CPU no lado local visto nos gráficos das Figuras 6.17, 6.20 e 6.23, nesse caso também tem relação com o tempo de execução do MPC e assim apresenta pequenas oscilações durante a simulação. O uso de memória pelos nós locais não apresenta variações significantes e por esse motivo os gráficos não foram gerados, entretanto, seus valores médios podem ser vistos na Tabela 6.5. Também foram extraídos os dados de uso da memória *swap*. Esses dados se mantiveram com valor zero durante todas as simulações, dado que a memória RAM alocada para as máquinas virtuais foi suficiente.

Vale ressaltar que embora a porcentagem de uso de CPU e memória sejam maiores nos nós locais, os recursos alocados para as máquinas virtuais que executam esses nós são menores (ver Tabela 6.2). Ainda assim, os recursos são suficientes para a execução dos nós de MPC de rastreamento da trajetória, sugerindo que a utilização de um computador mais simples embarcado nos AGVs é suficiente para a implementação da camada local da arquitetura proposta, reduzindo custos e o consumo de bateria.

Dos resultados vistos nas simulações com o Cenário 1, pode-se concluir que, graças às ações preditivas dos nós de MPC de planejamento da trajetória no servidor de borda, os AGVs navegam no ambiente, evitando colisões com obstáculos fixos e entre si, mesmo com as degradações. Nas métricas do cenário em relação ao intervalo de tempo das iterações do MPC e ao uso dos recursos computacionais, observa-se que não há diferença significativa entre as obtidas nas simulações com os perfis ‘Sem atraso’, ‘300ms’ e ‘200msPL50%’,

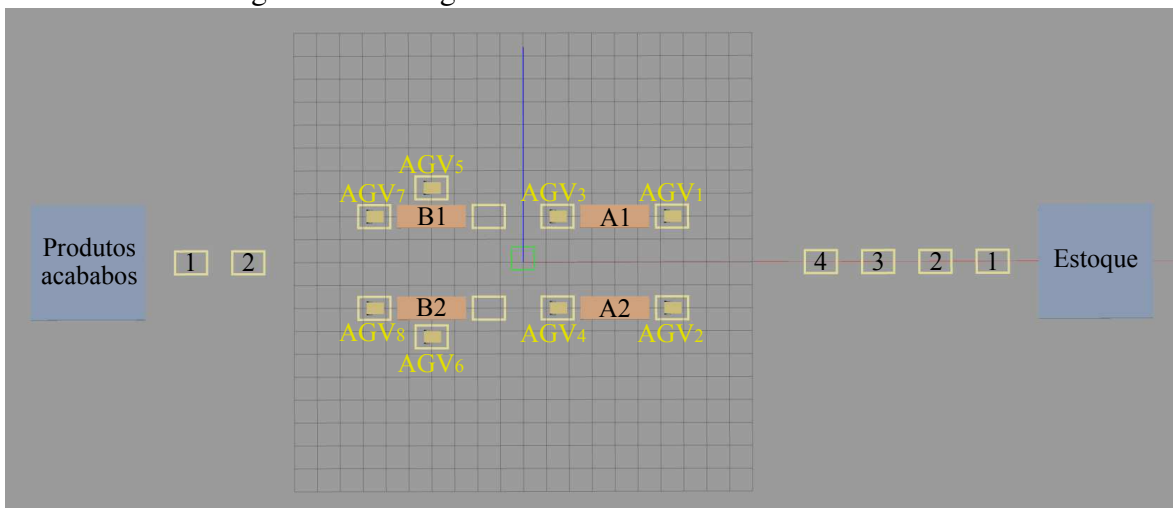
portanto, o sistema é capaz de operar mantendo um comportamento similar nessas métricas, mesmo com as degradações impostas na rede de comunicação.

6.6.2 Cenário 2

No cenário anterior, foi possível validar a arquitetura em diferentes condições da rede de comunicação. No entanto, a simulação é breve e os AGVs apenas devem completar uma única missão. Nesse sentido, o Cenário 2 foi criado para demonstrar a utilização da arquitetura proposta em uma situação hipotética de uma pequena linha de produção. Nesse cenário os AGVs transportam materiais do estoque para as estações de trabalho, entre as estações de trabalho, e destas para para a área de produtos acabados; repetindo as atividades enquanto a linha está em funcionamento. Desse modo, pretende-se realizar a validação em uma aplicação em que os elementos da arquitetura são solicitados diversas vezes para atender à demanda da linha de produção, verificando se os AGVs navegam sem colisões, inclusive quando atrasos na rede de comunicação são induzidos.

Supõe-se que os materiais podem ser inseridos e retirados dos AGVs por humanos ou robôs articulados, e que o processamento dos materiais nas estações de trabalho são realizados por humanos ou máquinas (não incluídos na simulação). Na Figura 6.24 é apresentado o mundo construído no Gazebo para o Cenário 2.

Figura 6.24: Imagem do mundo do Cenário 2 no Gazebo.



Fonte: Produzida pelo autor.

Nas estações de trabalho A1 e A2, são executadas as mesmas tarefas a partir dos materiais

trazidos do estoque pelos AGVs. O material processado segue de A1 para B1 e de A2 para B2. Nas estações B1 e B2, são executadas tarefas que processam o material resultante das estações anteriores (A1 e A2) e outro material vindo do estoque. Por fim, os produtos acabados, resultantes dos processamentos das estações B1 e B2, são transportados pelos AGVs até a área de produtos acabados. Estabelecidas as regras de funcionamento da linha de produção, o Supervisor deve alocar as seguintes tarefas para os AGVs:

- os AGVs 1 e 2 trazem materiais do estoque para as respectivas estações trabalho A1 e A2;
- o AGV₃ carrega o material processado na estação A1 e o entrega na estação B1;
- o AGV₄ carrega o material processado na estação A2 e o entrega na estação B2;
- os AGVs 5 e 6 trazem materiais do estoque para as respectivas estações de trabalho B1 e B2;
- os AGVs 7 e 8 transportam os produtos processados pelas estações B1 e B2, respectivamente, até a área de produtos acabados.

No estoque a coleta do material é realizada apenas na vaga 1, indicada pela numeração à direita na ilustração da Figura 6.24. Os AGVs 1, 2, 5 e 6, portanto, disputarão essa vaga. Para evitar conflitos, o Supervisor gerencia uma fila de espera com mais três vagas que os AGVs podem ocupar até alcançarem a vaga 1. A ocupação das vagas é feita de acordo com a ordem de alocação das tarefas. Suponha que foi enviado ao AGV₅ a ordem para coleta de material no estoque. Em seguida, o mesmo aconteceu para o AGV₁. Ainda que o AGV₁ esteja mais próximo do estoque, a vaga 1 ou a vaga mais próxima que esteja livre, será ocupada pelo AGV₅ e o AGV₁ deve aguardar na vaga anterior. O mesmo acontece com os AGVs 7 e 8 nas duas vagas da área de produtos acabados. Esses deslocamentos entre as vagas da linha de produção são inicializados através das chamadas de serviço do MPC de planejamento da trajetória, do nó correspondente ao AGV que deve executar a tarefa.

O procedimento de execução da simulação é realizado da mesma forma que o Cenário 1. No entanto, no cenário corrente existem mais nós no servidor de borda e no lado local, visto que existem mais AGVs envolvidos. Dados o maior número de veículos e os recursos computacionais disponíveis para simulação, foi utilizada mais uma máquina virtual no

computador da borda com as mesmas configurações da máquina virtual associada ao MPC na Tabela 6.2, para executar os nós do MPC de planejamento da trajetória. Foram distribuídos quatro nós por máquina virtual, executados por *launch files* específicos para cada máquina, de forma a atender aos oito AGVs do cenário. Esse procedimento permite que o software utilizado na virtualização das máquinas virtuais possa utilizar partes dos recursos da máquina hospedeira que antes estavam ociosos. Para as simulações isso implicará em um menor intervalo de tempo das iterações do MPC.

Resultados

Em única simulação com esse cenário, os perfis de rede são alterados durante a execução. Visto que o funcionamento da arquitetura com o perfil de rede com perda de pacotes já foi demonstrado no cenário anterior, apenas os perfis de rede ‘Sem atraso’ e ‘300ms’ são aplicados. Em testes realizados, a aplicação do perfil ‘200msPL50%’ em algumas vezes comprometeu as chamadas de serviço do MPC de rastreamento da trajetória e por essa razão não foi aplicado nas simulações com o cenário. Uma solução para esse problema é adicionar no nó do MPC do servidor de borda, um procedimento que faça chamadas do serviço de MPC local até que uma resposta seja obtida, ou ainda, no caso da implementação com o ROS 2, ajustar as políticas da qualidade de serviço⁵ existentes na ferramenta.

A simulação foi inicializada através do botão ‘Iniciar o cenário’ da interface gráfica do Supervisor e finalizada através do botão ‘Parar’ após aproximadamente seis minutos. A gravação da tela da máquina virtual que executa o Gazebo, durante a execução das simulações com o Cenário 2, pode ser vista no vídeo em <https://youtu.be/27ysYtpTtc4>. Os parâmetros dos nós de MPC de planejamento da trajetória estão listados na Tabela 6.6, e os do MPC de rastreamento da trajetória são os mesmos do cenário anterior, listados na Tabela 6.4. A simulação ocorre na maior parte do tempo no perfil de rede ‘Sem atraso’. O perfil ‘300ms’ é aplicado durante trinta segundos, aproximadamente nos minutos um, três e cinco da simulação. Ao aplicar os atrasos nesses instantes é possível verificar o comportamento da arquitetura quando a rede é comutada entre os perfis ‘Sem atraso’ e ‘300ms’.

Na Figura 6.25 está ilustrado o gráfico com as trajetórias percorridas pelos AGVs na

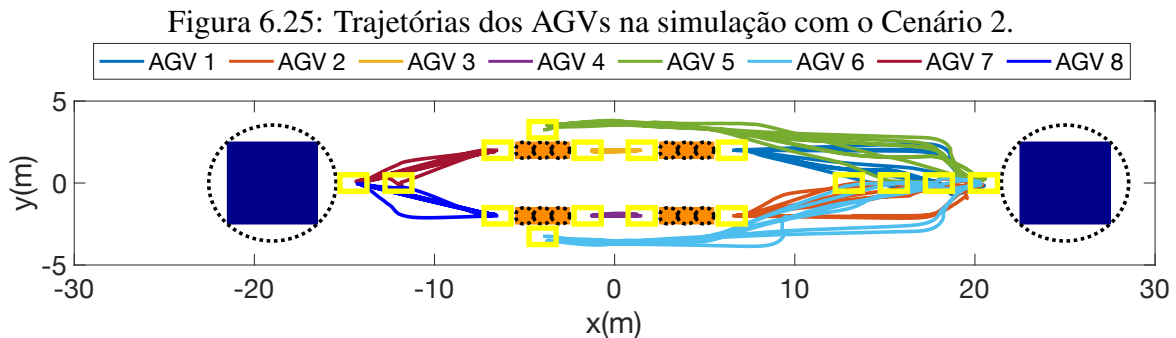
⁵<https://docs.ros.org/en/foxy/Concepts/About-Quality-of-Service-Settings.html>

Tabela 6.6: Parâmetros configurados no MPC de planejamento da trajetória nas simulações do Cenário 2.

T	100 ms
N	60
Q	$\text{diag}[5 \ 5 \ 0,5]$
R	$\text{diag}[0,5 \ 0,5]$
u_v^{min}	- 1,0 m/s
u_v^{max}	+ 1,0 m/s
u_ω^{min}	- $\pi/4$ rad/s
u_ω^{max}	+ $\pi/4$ rad/s
$diam_{AGV}$	0,75 m
d_{seg}^{obs}	1,0 m
d_{seg}^{agv}	0,5 m

simulação do Cenário 2. Os círculos pontilhados em preto, os quais envolvem as áreas do estoque, dos produtos acabados e das estações de trabalho, são as representações circulares desses obstáculos utilizados na restrição de prevenção de colisões do MPC de planejamento da trajetória. Para diminuir a quantidade de obstáculos fixos a serem considerados por esse MPC e conseqüentemente diminuir o tempo das iterações e o uso de CPU, foi incluído no código do respectivo nó, uma função disparada periodicamente por um temporizador do ROS 2, responsável por calcular a distância entre o AGV e os obstáculos fixos. No mesmo código determina-se a quantidade dos obstáculos mais próximos a serem considerados pelo MPC. Nessa simulação foram considerados apenas os cinco obstáculos mais próximos. Visto que cada estação de trabalho é representada por três círculos, temos no cenário um total de catorze obstáculos, dos quais apenas os cinco mais próximos são considerados pelo MPC. Com relação aos raios dos círculos que representam os obstáculos, a divisão da estação de trabalho em círculos menores resulta em mais espaço no ambiente para circulação dos AGVs. Por exemplo, se essas estações fossem representadas por um único círculo maior, ao respeitar

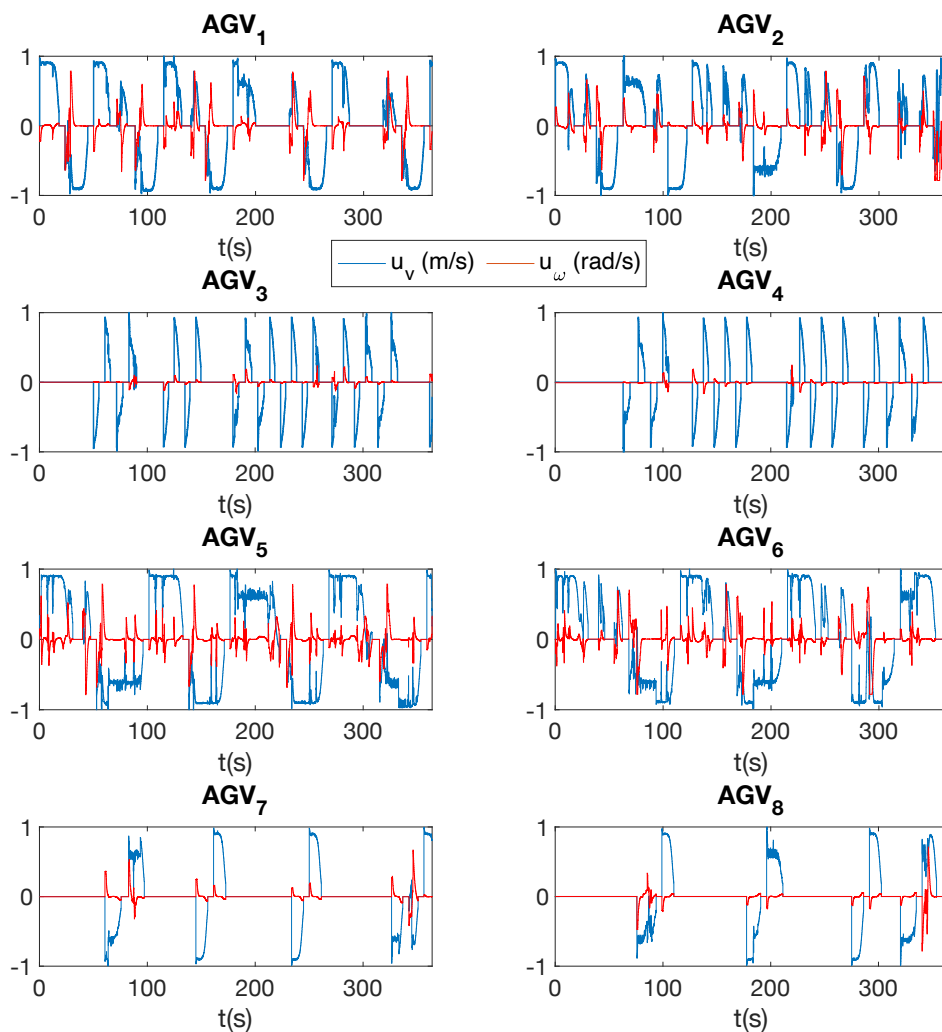
a distância mínima de segurança com o círculo, o MPC limitaria a área de circulação dos AGVs naquele ambiente.



Fonte: Produzida pelo autor.

Na Figura 6.26 estão apresentados os gráficos dos sinais de controle das velocidades linear e angular da simulação. Em alguns instantes esses sinais podem permanecer com valor zero ou próximos de zero. O primeiro caso acontece nos instantes em que os AGVs estão parados, ou seja, aguardando que o material seja colocado ou retirado. O segundo caso acontece quando os AGVs não necessitam de maiores atuações em alguma das variáveis. É o caso dos AGVs 3, 4, 7 e 8, os quais têm picos positivos e negativos nos sinais de controle da velocidade linear, de modo a moverem-se para frente e para trás, entretanto, apresentam pequenas alterações nos sinais de controle da velocidade angular, já que os ângulos de orientação não necessitam de maiores ajustes para que os AGVs possam estabilizar nas referências.

Figura 6.26: Sinais de controle na simulação com o Cenário 2.

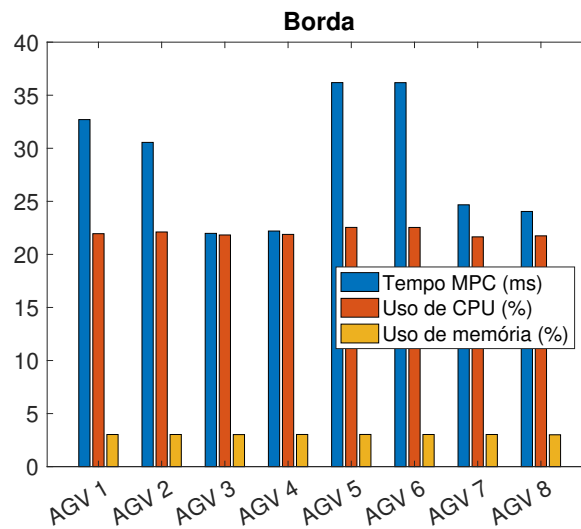


Fonte: Produzida pelo autor.

Quando comparado ao Cenário 1, tem-se uma maior quantidade de dados gerados na simulação do Cenário 2, dada a existência de mais AGVs. Por esse motivo a tabela foi substituída por gráficos em barra. Na Figura 6.27 é apresentado o gráfico com as métricas relacionadas à borda, e na Figura 6.28 o gráfico com as métricas relacionadas ao nós do lado local.

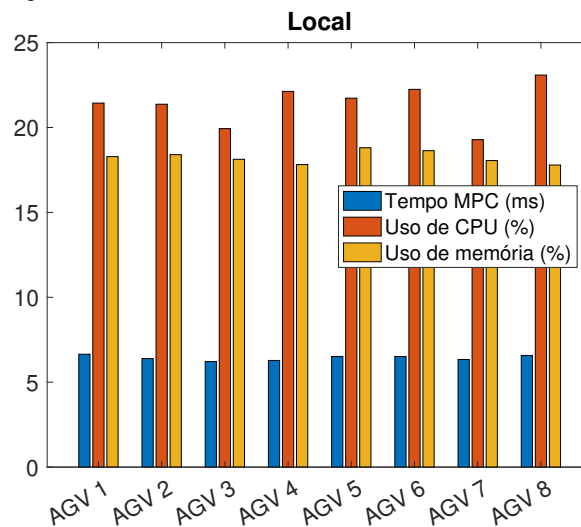
Ainda nesse cenário, foi executada uma segunda simulação, com a adoção de uma estratégia em que se limita a quantidade de trajetórias que serão consideradas na prevenção de colisões. Similar à estratégia aplicada com os obstáculos fixos, uma função disparada por

Figura 6.27: Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória na borda, na simulação com o Cenário 2.



Fonte: Produzida pelo autor.

Figura 6.28: Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória no lado local, na simulação com o Cenário 2.

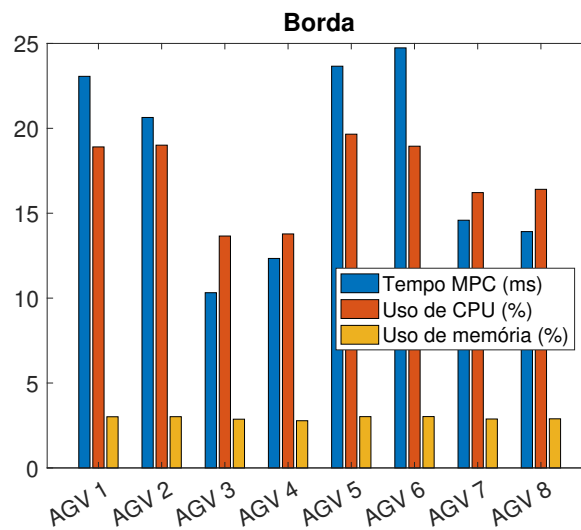


Fonte: Produzida pelo autor.

um temporizador no nó do MPC no servidor de borda calcula a distância de um respectivo AGV com os outros AGVs. Apenas as trajetórias de uma determinada quantidade de AGVs serão consideradas. Na simulação, os AGVs 1, 2, 5 e 6, não têm nenhuma limitação, ou seja, consideram as trajetórias de todos os AGVs. Os AGVs 3 e 4 incluem apenas as trajetórias dos dois AGVs mais próximos. Por último, os AGVs 7 e 8 consideram as trajetórias dos

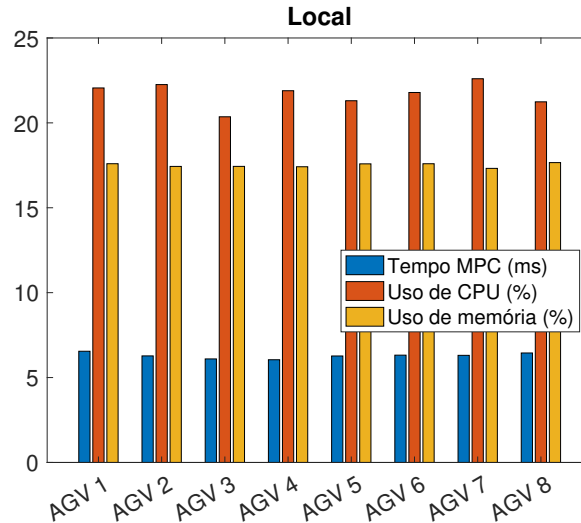
quatro AGVs mais próximos. Dessa maneira é possível verificar as médias dos intervalos de tempo das iterações do MPC quando diferentes quantidades de trajetórias são consideradas. Nessa segunda simulação foi utilizado apenas o perfil de rede ‘Sem atraso’, de modo a apenas avaliar as métricas quando aplicada a estratégia mencionada. Nas Figuras 6.29 e 6.30 estão apresentados os gráficos com as métricas relacionadas à borda e ao lado local, respectivamente.

Figura 6.29: Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória na borda, na simulação com o Cenário 2, com limitação de trajetórias consideradas na prevenção de colisões.



Fonte: Produzida pelo autor.

Figura 6.30: Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória no lado local, na simulação com o Cenário 2, com limitação de trajetórias consideradas na prevenção de colisões.



Fonte: Produzida pelo autor.

Discussão

No gráfico da Figura 6.27, referente à primeira simulação do Cenário 2, observa-se um aumento na média do intervalo de tempo das iterações do MPC quando comparado ao Cenário 1. Ainda que nesse cenário o MPC esteja considerando apenas os cinco obstáculos fixos mais próximos, igualando à quantidade de obstáculos fixos do Cenário 1, cada nó de MPC deve considerar uma quantidade maior de trajetórias para a prevenção de colisões, e assim espera-se esse aumento. A média do uso de CPU também se eleva nesse cenário, mas tem valores próximos em todos os AGVs, enquanto a média do uso de memória é ligeiramente maior em relação ao Cenário 1.

No gráfico da Figura 6.28, também da primeira simulação, verifica-se que a média do intervalo de tempo das iterações do MPC de rastreamento da trajetória se mantém muito próximas das obtidas no Cenário 1, e que existe um aumento na média do uso de CPU e memória. Conforme já discutido anteriormente, os nós de MPC de rastreamento da trajetória não inclui em seu problema as restrições de prevenção de colisões com obstáculos fixos e dos AGVs entre si, fazendo com que o intervalo de tempo das iterações tenha maior uniformidade entre os AGVs.

Na segunda simulação percebe-se que, os AGVs que têm a mesma limitação na quantidade de trajetórias a serem consideradas, têm as médias dos intervalos de tempo das iterações do MPC similares, mas com médias do uso de CPU ainda mais próximas, conforme visto no gráfico da Figura 6.29. Isso acontece pelo fato de que os MPCs com mesma configuração devem solucionar um problema de mesma magnitude, e por isso, tendem a apresentar médias muito próximas do uso de CPU. Particularidades das trajetórias de cada AGV farão com que se tenham apenas pequenas diferenças nas médias de tempo do MPC. Para o lado local, conforme esperado, as métricas têm comportamento similar em relação à primeira simulação.

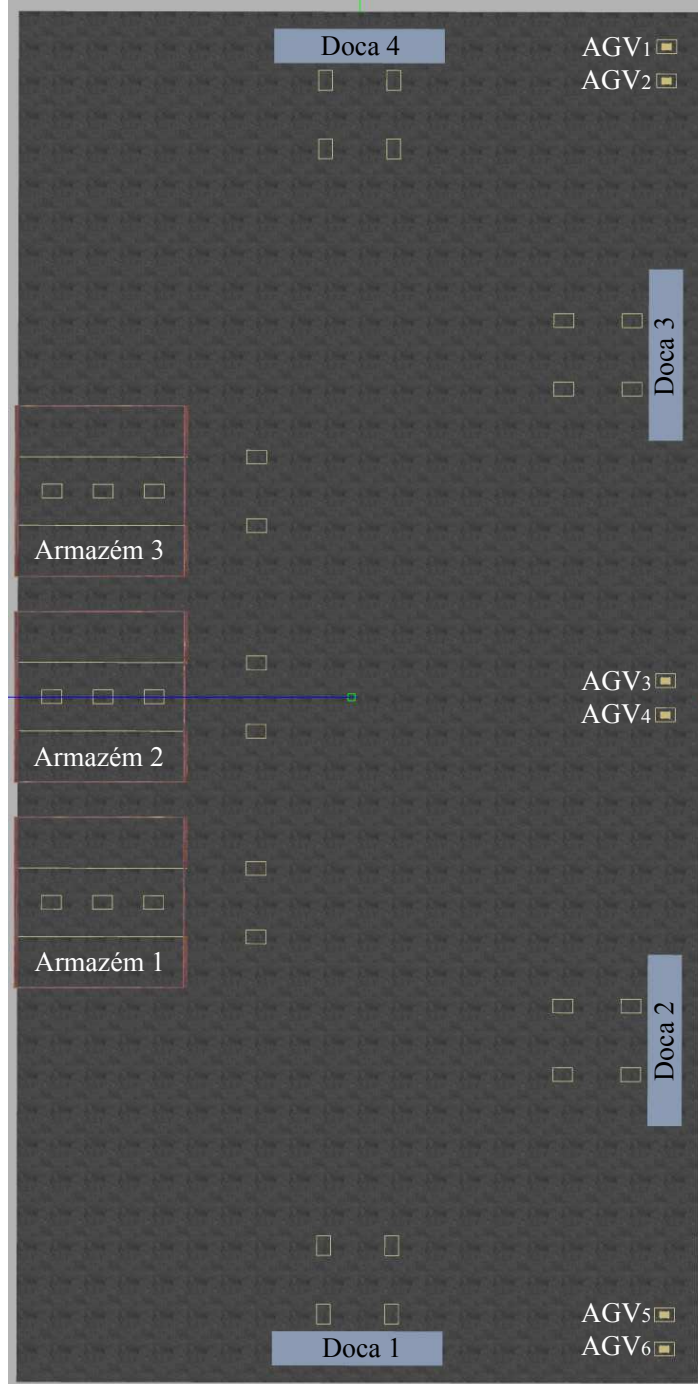
O Cenário 2 representa uma pequena linha de produção hipotética e conta com um algoritmo no Supervisor para assistir o funcionamento da linha. Numa única simulação com o cenário, foram aplicados os perfis de rede ‘Sem atraso’ e ‘300ms’, e os AGVs puderam navegar sem colisões em toda a simulação. Quanto às métricas obtidas na borda, verifica-se que os AGVs que navegam em maiores percursos têm uma maior média no intervalo de tempo das iterações do MPC de planejamento da trajetória. Entretanto, o uso de CPU é muito próximo entre os AGVs. Isso indica que a magnitude do problema é o fator principal que definirá o uso de CPU, já que o MPC associado a todos os AGVs previnem colisões com a mesma quantidade de obstáculos e também consideram a mesma quantidade de trajetórias de outros AGVs na prevenção de colisões. Numa segunda simulação no mesmo cenário, as trajetórias a serem consideradas por cada AGV foram limitadas em diferentes quantidades. Observa-se que quanto maior o número de trajetórias consideradas, maiores as médias de tempo do MPC e do uso de CPU, mas que essa segunda métrica se mantém muito próxima entre os AGVs que incluem a mesma quantidade de trajetórias. Isso confirma a relação já mencionada entre a magnitude do problema e o uso de CPU.

6.6.3 Cenário 3

O Cenário 3 também representa uma situação hipotética onde a arquitetura proposta pode ser implementada, sendo voltado a uma aplicação logística que pode ser parte da estrutura de uma transportadora de mercadorias ou de uma indústria, por exemplo. Diferente do cenário anterior, os AGVs não executam tarefas repetitivas, mas recebem diferentes missões durante o funcionamento do cenário. Pretende-se, portanto, validar a arquitetura nessas condições.

Considerando que os AGVs transportarão cargas de maior volume, o modelo utilizado é o do AGV maior, já visto na Figura 6.9, com a inclusão de seis deles no cenário. Na Figura 6.31 é apresentado o mundo construído no Gazebo para o Cenário 3.

Figura 6.31: Imagem do mundo do Cenário 3 no Gazebo.



Fonte: Produzida pelo autor.

Para a simulação o Supervisor cria uma tarefa aleatória a cada trinta segundos e a aloca

ao AGV que esteja livre e mais próximo do ponto de coleta da carga. Um intervalo de tempo menor que trinta segundos poderia gerar um acúmulo de tarefas em espera e por esse motivo esse intervalo foi estabelecido. Caso alguma tarefa esteja na espera, o Supervisor aguarda pelo próximo AGV livre para alocá-la.

Os locais de coleta e entrega da carga são armazéns, que são locais de armazenamento de cargas, ou docas, locais onde as cargas chegam ou saem do centro logístico. Uma tarefa então envolve a coleta da carga em uma doca para ser entregue no armazém ou vice-versa. O Supervisor também gerencia a ocupação das vagas nos armazéns e nas docas. No armazém, a tarefa deve especificar em qual das três vagas o AGV deve estacionar e apenas um AGV por vez é autorizado a entrar. Caso o armazém esteja ocupado, outros AGVs que estejam se encaminhando para o mesmo armazém devem aguardar em duas vagas de espera disponíveis. Nas docas existem dois pontos de coleta ou entrega do material, também especificado na tarefa, com uma vaga de espera para cada uma delas. O AGV fica estacionado por cinco segundos toda vez que for realizar a coleta ou entrega do material, considerando que esse é o tempo que o material será entregue ou retirado por um robô articulado, humano ou qualquer outro agente.

Resultados

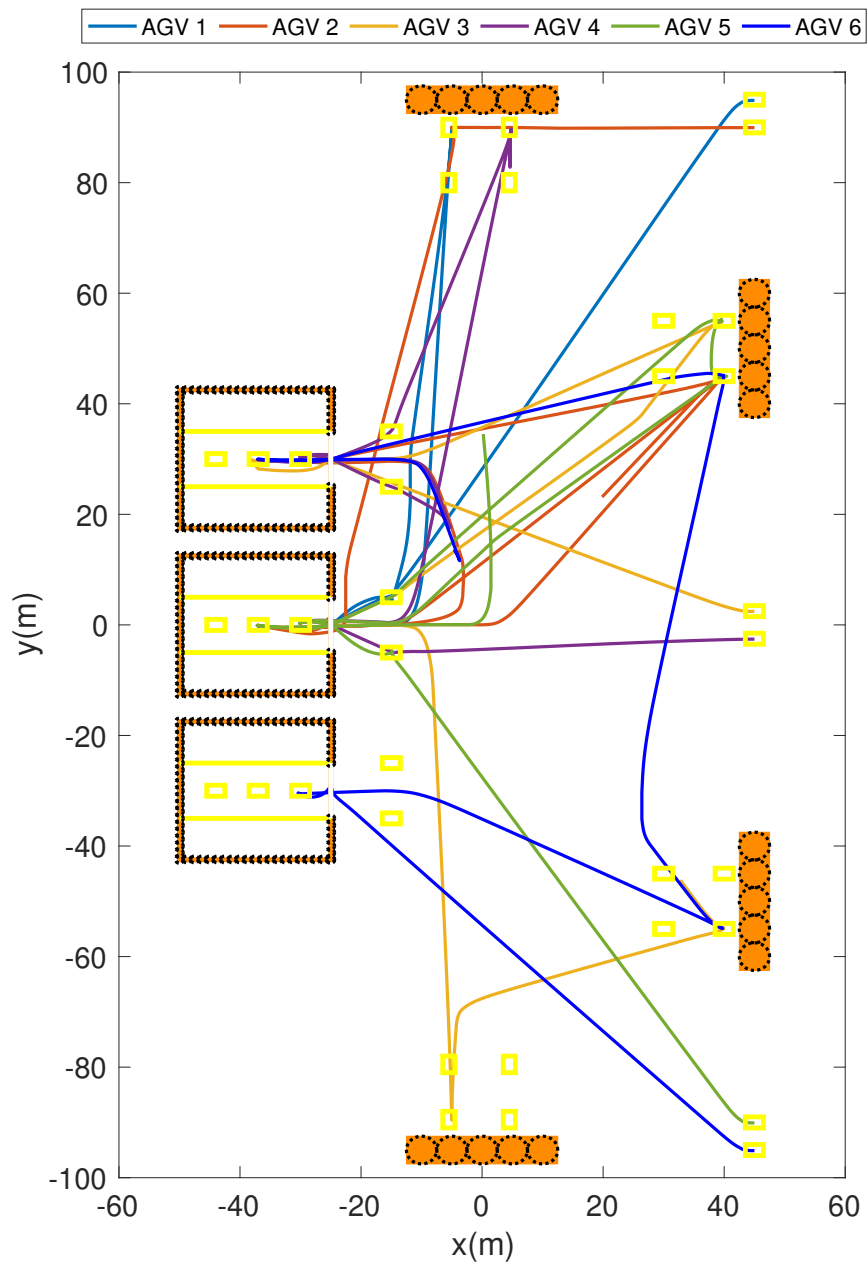
Para o Cenário 3 foi executada uma simulação de aproximadamente oito minutos, apenas utilizando o perfil de rede ‘Sem atraso’, visto que a utilização da arquitetura em diferentes condições de rede já foi abordada nos cenários anteriores. A gravação da tela da máquina virtual que executa o Gazebo, durante a execução das simulações com o Cenário 3, pode ser vista no vídeo em <https://youtu.be/jErzw1GzCz4>. A fim de facilitar a visualização, a gravação do vídeo foi realizada com a inversão dos eixos x e y no Gazebo. Na Tabela 6.7 estão listados os parâmetros do MPC de planejamento da trajetória. Os parâmetros do MPC de rastreamento da trajetória são os mesmos do Cenário 1, listados na Tabela 6.4.

Tabela 6.7: Parâmetros configurados no MPC de planejamento da trajetória nas simulações do Cenário 3.

T	100 ms
N	60
Q	$\begin{bmatrix} 5 & 5 & 0,5 \end{bmatrix}$
R	$\begin{bmatrix} 0,5 & 0,5 \end{bmatrix}$
u_v^{min}	- 1,5 m/s
u_v^{max}	+ 1,5 m/s
u_ω^{min}	- $\pi/4$ rad/s
u_ω^{max}	+ $\pi/4$ rad/s
$diam_{AGV}$	1,5 m
d_{seg}^{obs}	1,5 m
d_{seg}^{agv}	1,5 m

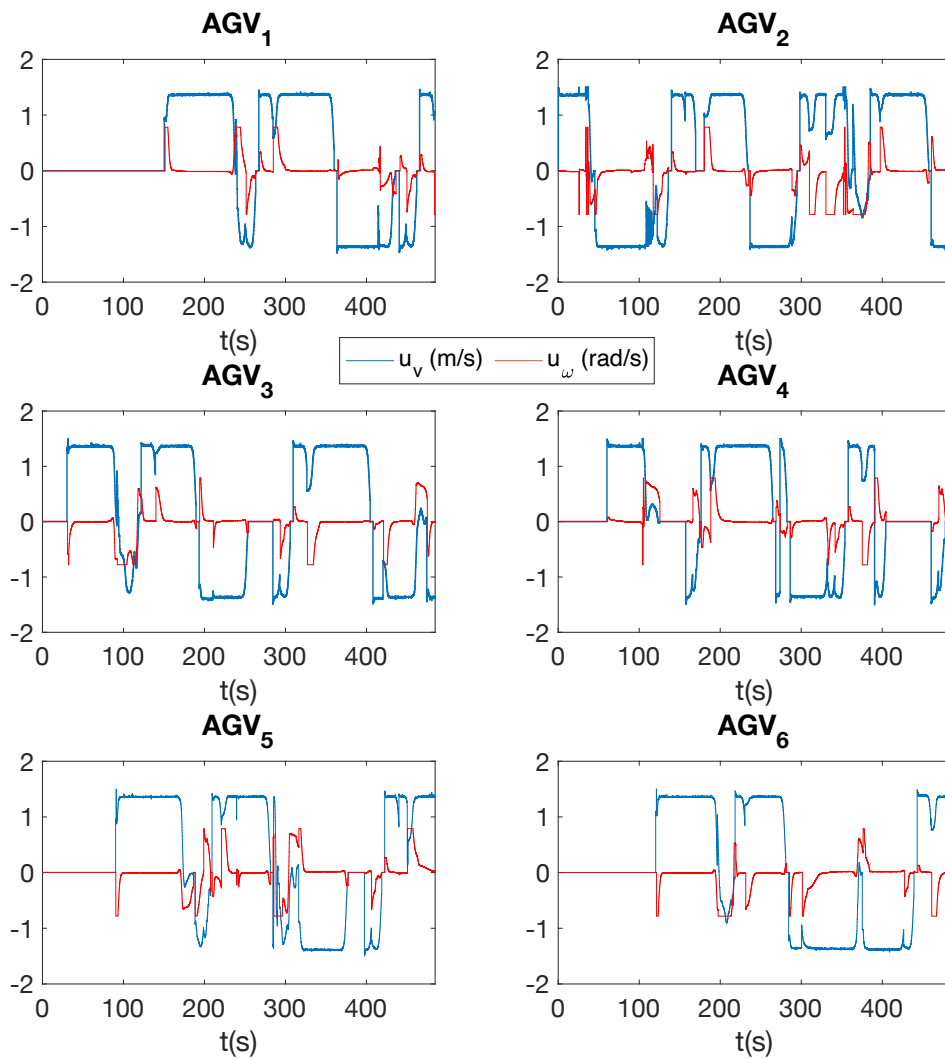
Na Figura 6.32 é apresentado o gráfico com as trajetórias dos AGVs na simulação. Os círculos pontilhados em preto na figura também são a representação dos obstáculos para o MPC, e nessa simulação também foram considerados apenas os cinco mais próximos. Na Figura 6.33 estão apresentados os gráficos com os sinais de controle aplicados nos AGVs na simulação executada. Similar ao Cenário 2, em alguns instantes esses sinais podem permanecerem em zero ou próximos de zero, seja pelo fato dos AGVs estarem ociosos (aguardando a alocação de uma tarefa), aguardando que o material seja retirado/colocado, ou por não precisarem de atuação em uma das variáveis. Em alguns instantes, por exemplo, os AGVs não precisam ajustar o ângulo de orientação, mantendo o valor do sinal da velocidade angular em zero (ou próximo), enquanto o sinal da velocidade linear está no valor mínimo ou máximo.

Figura 6.32: Trajetórias dos AGVs na simulação com o Cenário 3.



Fonte: Produzida pelo autor.

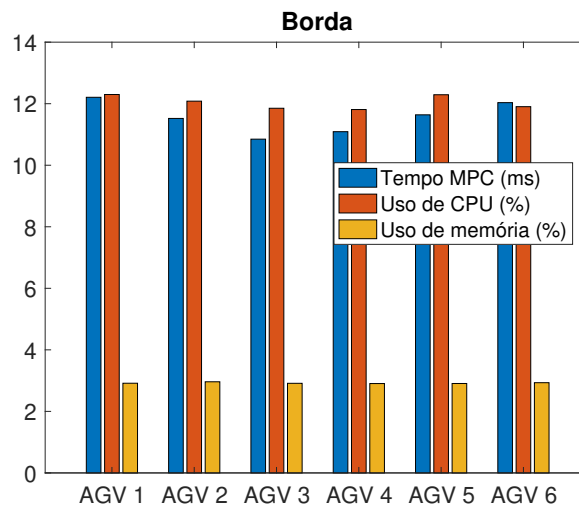
Figura 6.33: Sinais de controle na simulação com o Cenário 3.



Fonte: Produzida pelo autor.

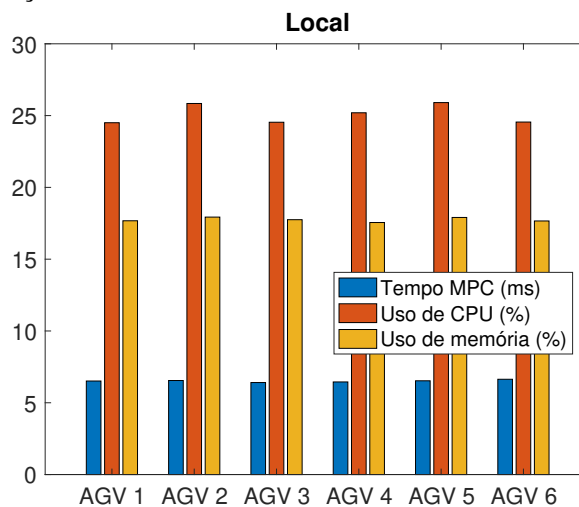
Por fim, na Figura 6.34 é apresentado o gráfico em barra com as métricas relacionadas à borda, enquanto na Figura 6.35 é apresentado o gráfico com as métricas do lado local.

Figura 6.34: Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória na borda, na simulação com o Cenário 3.



Fonte: Produzida pelo autor.

Figura 6.35: Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória no lado local, na simulação com o Cenário 3.



Fonte: Produzida pelo autor.

Discussão

Com a redução do número de AGVs do cenário, apenas uma máquina virtual para execução dos nós de MPC foi utilizada. Quando comparado ao Cenário 2, verifica-se no gráfico da Figura 6.34 uma redução na média do intervalo de tempo das iterações do MPC e no uso

de CPU. Além de existir menos AGVs (reduzindo o número de trajetórias na prevenção de colisões), o cenário tem mais espaço e possibilita que os nós de MPC no servidor de borda possam encontrar trajetórias livre de colisões de forma mais fácil, resultando na redução das métricas de tempo do MPC e uso de CPU. O uso de memória tem valores aproximados em relação ao Cenário 2. Quanto às métricas do lado local, apresentadas no gráfico da Figura 6.35, os valores têm comportamento similar aos do Cenário 2, tendo apenas uma pequena elevação no uso de CPU.

O Cenário 3 é voltado para aplicações logísticas e se utiliza de AGVs de dimensões maiores (em relação aos outros cenários) para transporte de cargas. O cenário também conta com um algoritmo do Supervisor para alocar as tarefas e gerenciar a ocupação das vagas. Diferente do Cenário 2, em que cada AGV executa sempre a mesma tarefa, uma tarefa de transporte de carga é alocada ao AGV que esteja mais próximo do ponto de retirada. Os AGVs, portanto, navegam em trajetórias de tamanho similar ao longo da simulação, o que leva as métricas da borda a terem mais uniformidade entre os AGVs. Comparado ao Cenário 2, as médias dos intervalos de tempo das iterações do MPC são menores, assim como o uso de CPU. Isso está relacionado à maior facilidade na geração das trajetórias livre de colisões, visto que o cenário tem mais espaço, como também à menor quantidade de AGVs (menos trajetórias a serem consideradas na prevenção de colisões).

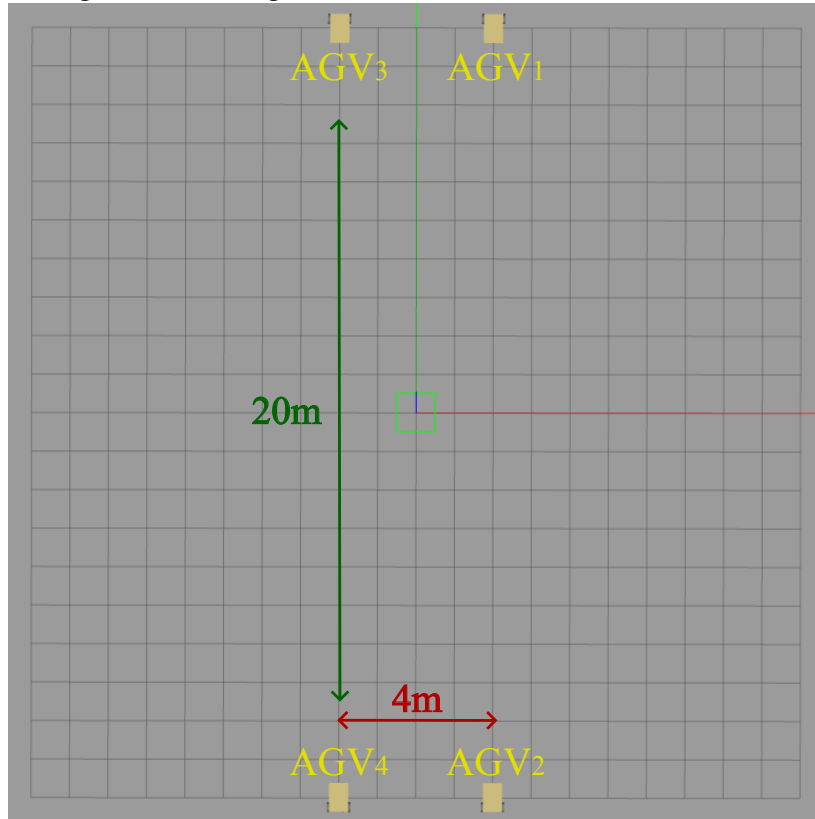
6.6.4 Cenário 4

O Cenário 4 é o último apresentado e tem uma proposta diferente em relação aos anteriores. Pretende-se com o cenário, avaliar a escalabilidade da arquitetura proposta, analisando o comportamento das métricas já utilizadas à medida que o número de AGVs é expandido. Nesse cenário também foi adotado o mesmo procedimento do Cenário 2 com relação à distribuição dos nós de MPC no servidor de borda em duas máquinas virtuais.

O mundo utilizado no Gazebo não tem outros elementos, apenas os AGVs, com a quantidade a ser inserida definida via *launch file*. Foram realizadas simulações utilizando quatro, oito e doze AGVs. Com o recurso computacional utilizado nas simulações, quantidades maiores de AGVs passam a comprometer o fator de tempo real do Gazebo, e por isso um máximo de doze AGVs foi o limite definido. Na Figura 6.36 é apresentado o mundo no Gazebo utilizado na simulação com quatro AGVs. Os veículos se separam por uma distância de

quatro metros no eixo x e de vinte metros no eixo y . Os mundos utilizados nas simulações com oito e doze AGVs seguem o mesmo padrão.

Figura 6.36: Imagem do mundo do Cenário 4 no Gazebo.



Fonte: Produzida pelo autor.

Resultados

Nas simulações com esse cenário, os AGVs opostos em relação ao eixo y trocam suas posições, mantendo a orientação inicial. A gravação da tela da máquina virtual que executa o Gazebo, durante a execução das simulações com o Cenário 4, pode ser vista no vídeo em <https://youtu.be/6kiVB1qm37U>. Na Tabela 6.8 estão listados os parâmetros do MPC de planejamento da trajetória. Os parâmetros do MPC de rastreamento da trajetória também seguem o do Cenário 1 (Tabela 6.4).

Devido ao maior número de AGVs utilizados nesse cenário (na simulação com doze), a distribuição dos recursos computacionais no lado local foi implementada de modo diferente. Apenas o AGV₁ utiliza a mesma máquina virtual dos cenários anteriores, enquanto os nós locais dos demais AGVs são executados por *launch file* em uma única máquina virtual, com

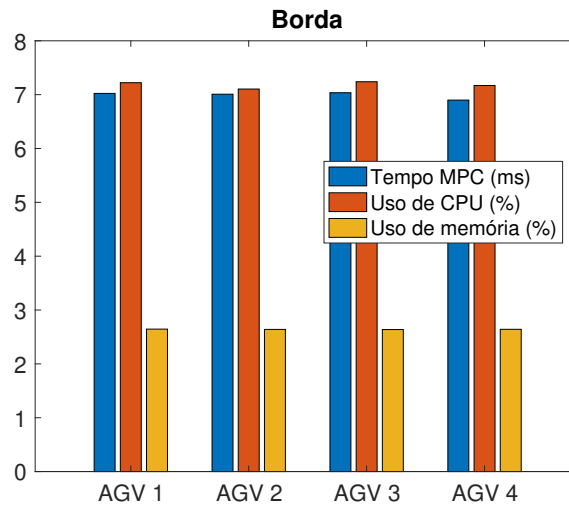
Tabela 6.8: Parâmetros configurados no MPC de planejamento da trajetória nas simulações do Cenário 4.

T	100 ms
N	60
Q	$\text{diag}[5 \ 5 \ 0,5]$
R	$\text{diag}[0,5 \ 0,5]$
u_v^{min}	- 1,0 m/s
u_v^{max}	+ 1,0 m/s
u_ω^{min}	- $\pi/4$ rad/s
u_ω^{max}	+ $\pi/4$ rad/s
$diam_{AGV}$	0,75 m
d_{seg}^{obs}	1,0 m
d_{seg}^{agv}	1,0 m

4 núcleos de processador e 4 GB de memória RAM. Com essa estratégia é possível analisar as métricas associadas ao AGV_1 , em uma máquina virtual que tem as mesmas configurações das utilizadas nas simulações dos cenários anteriores. Ao mesmo tempo, os nós relacionados aos outros AGVs são concentrados em uma única máquina virtual, eliminando a necessidade de lidar com mais máquinas virtuais em simulações com mais AGVs.

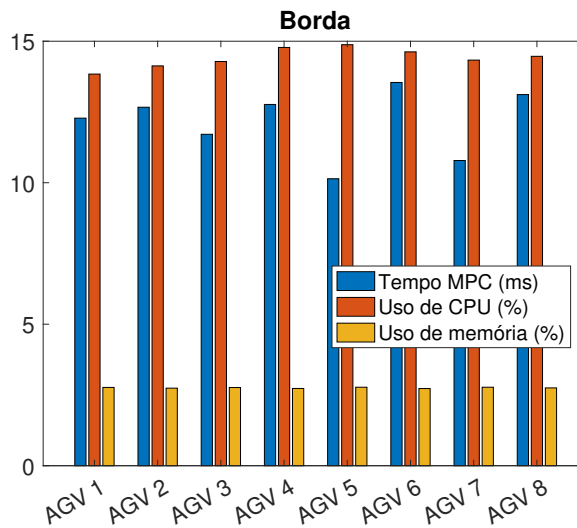
Nas Figuras 6.37, 6.38 e 6.39 estão apresentados os gráficos em barra com as métricas relacionadas à borda, nas simulações com quatro, oito e doze AGVs, respectivamente. Foi realizada outra simulação com doze AGVs, onde cada nó correspondente no servidor de borda considera apenas as trajetórias dos quatro AGVs mais próximos, implementando a mesma estratégia já adotada no Cenário 2. Na Figura 6.40 é apresentado o gráfico em barra com as métricas obtidas na borda com essa segunda simulação. No vídeo e nas tabelas apresentadas a seguir, a mesma simulação é representada pelo identificador ‘12 AGVs (4)’.

Figura 6.37: Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória na borda, na simulação do o Cenário 4 com quatro AGVs.



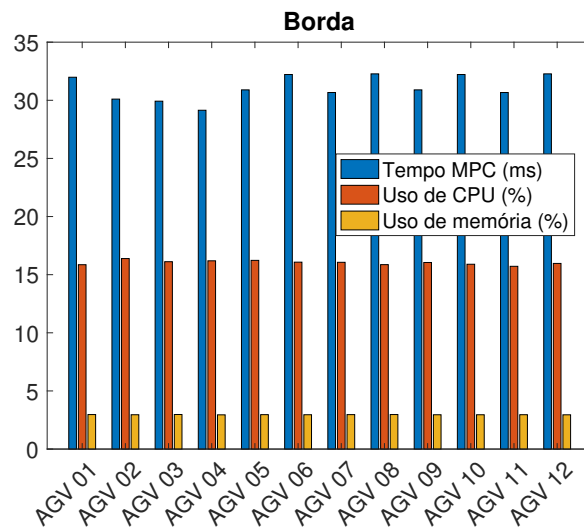
Fonte: Produzida pelo autor.

Figura 6.38: Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória na borda, na simulação do o Cenário 4 com oito AGVs.



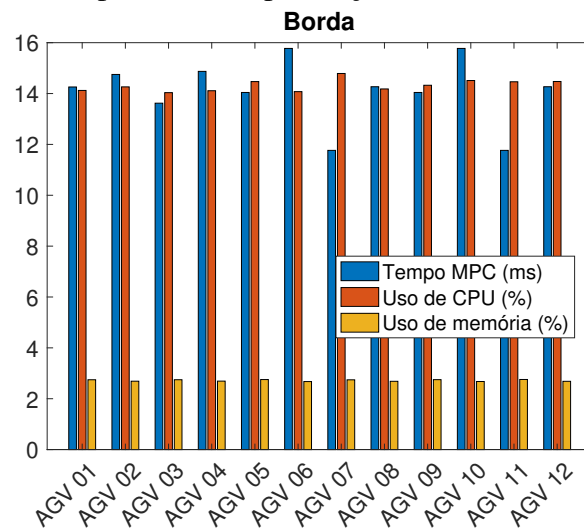
Fonte: Produzida pelo autor.

Figura 6.39: Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória na borda, na simulação do Cenário 4 com doze AGVs.



Fonte: Produzida pelo autor.

Figura 6.40: Médias do intervalo de tempo das iterações do MPC, uso de CPU e memória na borda, na simulação do o Cenário 4 com doze AGVs, com limitação do MPC para considerar apenas os quatro AGVs mais próximos na prevenção de colisões.



Fonte: Produzida pelo autor.

Na Tabela 6.9 estão apresentados os somatórios das médias do uso de CPU na borda, por simulação. Para o lado local, conforme já mencionado, apenas a máquina virtual associada ao AGV₁ mantém as configurações dos cenários anteriores. Portanto, apenas as métricas para esse AGV, em todas as simulações, foram extraídas e apresentadas na Tabela 6.10.

Tabela 6.9: Somatório das médias do uso de CPU na borda em todas as simulações do Cenário 4.

Simulação	Máquina virtual	Somatório do uso de CPU (%)
4 AGVs	1	14,46
	2	14,27
8 AGVs	1	57,33
	2	58,00
12 AGVs	1	96,04
	2	96,38
12 AGVs (4)	1	86,20
	2	85,61

Tabela 6.10: Métricas do lado local relacionadas ao AGV₁ nas simulações do Cenário 4.

Simulação	Intervalo de tempo das iterações do MPC (ms)	Uso de CPU (%)	Uso de memória (%)
4 AGVs	5,69	19,83	16,01
8 AGVs	6,02	22,76	16,07
12 AGVs	6,69	25,92	16,41
12 AGVs (4)	7,03	26,92	16,45

Discussão

Conforme esperado, verifica-se a partir das Figuras 6.37, 6.38 e 6.39, que as médias de tempo das iterações do MPC e do uso de CPU no servidor de borda aumentam na medida em que há uma expansão na quantidade de AGVs. Na simulação com oito AGVs, existe um aumento já esperado nas métricas pelo fato de que o problema do MPC torna-se maior (mais trajetórias na prevenção de colisões). No entanto, conforme visto na Tabela 6.9, o somatório do uso de CPU por máquina virtual no servidor de borda ainda não se encontra no seu limite máximo (100%). Quando a quantidade de AGVs se expande para doze, o somatório se torna

próximo de seus limites, já que existem mais nós de MPC no servidor de borda (seis por máquina virtual), solucionando problemas de MPC ainda maiores. Nesse caso, vários nós disputam pelo mesmo recurso que já se encontra próximo da saturação, o que contribuirá para o aumento do tempo médio do MPC. Quando se considera apenas as trajetórias dos quatro veículos mais próximos, observa-se na Tabela 6.9 uma redução do somatório do uso de CPU, assim como uma redução de aproximadamente 50% na média de tempo do MPC, visto na Figura 6.40.

Dos resultados das métricas locais relacionadas ao AGV_1 , vistas na tabela 6.10, observa-se que ocorrem pequenos aumentos nas métricas conforme a quantidade de AGVs é elevada. Com essa elevação, os nós de MPC no servidor de borda fazem constantes alterações nas trajetórias, as quais serão tratadas posteriormente pelo MPC no lado local e resultam no aumento dessas métricas nesse lado. O mesmo acontece na última simulação, onde a quantidade de trajetórias consideradas pelo MPC no servidor de borda para prevenção de colisões é limitada a quatro. Nesse caso há constantes correções na trajetória pelo fato de que os AGVs mais próximos vão se alterando ao longo da simulação, resultando em constantes alterações na trajetória, que por consequência causarão um pequeno aumento nas métricas do lado local.

A partir das simulações com o Cenário 4, conclui-se que a arquitetura pode ser escalada com a adição de nós de planejamento da trajetória no servidor de borda. Entretanto, deve ser verificado se as médias do intervalo de tempo do MPC não ultrapassarão os limites da execução em tempo real, visto que, com mais nós, maior será o problema do MPC (mais trajetórias na prevenção de colisões), assim como o uso de CPU será mais concorrido entre os nós. Isso pode ser atenuado através da disponibilização de mais recursos computacionais, ou mesmo pela adoção de medidas que possam reduzir o problema do MPC, tal como a estratégia de reduzir a quantidade de trajetórias consideradas na prevenção de colisões. Nesse último caso é possível incluir apenas as trajetórias dos AGVs mais próximos.

6.7 Considerações finais

Nesse capítulo foram realizadas várias simulações que implementam a arquitetura proposta na Seção 5.1. As simulações foram distribuídas em cenários apenas de validação da arqui-

tetura, e outros cenários que não só validam, mas também podem representar um cenário real de aplicação. Nesse segundo caso, um algoritmo executado junto ao Supervisor faz a alocação das tarefas e supervisiona a ocupação das vagas em que os AGVs devem estacionar para concluir a tarefa de coleta ou entrega da carga.

No Cenário 1 o sistema foi submetido a degradações no sinal da rede e mesmo nessas condições os AGVs atingiram suas referências, desviando dos obstáculos fixos e entre si. No Cenário 2 os AGVs foram alocados para executar tarefas repetitivas em uma linha de produção hipotética, com suporte do Supervisor, e puderam realizá-las mesmo quando foram induzidos atrasos na rede. No mesmo cenário foi visto que estratégias podem ser adotadas para reduzir o uso de CPU e o tempo médio das iterações do MPC. No Cenário 3 a arquitetura é colocada em teste em uma situação em que os AGVs são alocados para diferentes atividades em um centro logístico hipotético, também com suporte do Supervisor. Os AGVs puderam completar tarefas que envolve inclusive a entrada em ambientes fechados, como os armazéns. No Cenário 4 foi visto que é possível escalar o sistema e que as estratégias também aplicadas no Cenário 2 podem ser adotadas para reduzir o uso de CPU e o tempo médio das iterações do MPC.

Quanto ao uso dos recursos computacionais no servidor de borda, foi visto nos resultados das simulações que o uso dos recursos, com maior impacto no uso de CPU, tem relação direta com a magnitude do problema de MPC a ser resolvido. Entretanto, o uso de CPU pode ser diminuído através da adoção de técnicas para redução do problema do MPC, como visto nos Cenários 2 e 4.

Quando se compara as métricas locais entre os cenários, observa-se o mesmo padrão em todos resultados, ou seja, independente do problema de MPC que o servidor de borda deve solucionar, as métricas do lado local sempre têm um comportamento similar. Dessa forma, os recursos computacionais na borda podem ser alocados de acordo com a complexidade imposta pelo cenário de aplicação, ao passo que um determinado recurso computacional embarcado no AGV pode ser utilizado em diferentes cenários. Nesse contexto, o servidor de borda, provido de mais recursos computacionais, pode suprir a aplicação no que se refere às atividades que têm mais demandas computacionais, enquanto um computador mais simples pode ser embarcado no AGV e atender a diversos cenários.

Capítulo 7

Conclusão

Nesta tese foi apresentada uma arquitetura de controle de AGVs com computação na borda e comportamento preditivo. Na abordagem, a maior parte da demanda computacional para a navegação dos AGVs é transferida para o servidor de borda, de modo que os computadores embarcados nos veículos executam tarefas mais simples, reduzindo os custos e o consumo de bateria. O controle preditivo é a solução adotada para mitigar os efeitos do uso da rede de comunicação, uma vez que esse meio de troca de informações é intrínseco ao uso da computação na borda.

Foi verificado na revisão bibliográfica que o MPC, no caso do controle de robôs móveis, funciona como um sistema de planejamento acoplado. Nessa modalidade, a trajetória e os sinais de controle necessários para conduzir o AGV na trajetória são gerados em conjunto. Além disso, foi verificado que a sequência dos sinais de controle gerados pelo MPC pode ser utilizada para compensar os atrasos e as perdas de pacotes na rede de comunicação. A partir daí, foram realizados experimentos utilizando o MPC no servidor de borda junto com um compensador no lado local, para o controle de um robô móvel de tração diferencial. O compensador aplica os sinais de controle da última sequência recebida do MPC do servidor de borda, até que um novo pacote seja recebido. Nos experimentos foi constatado que o robô seguiu a trajetória especificada, inclusive na ocorrência de atrasos e perdas de pacotes na rede de comunicação.

Ainda na revisão bibliográfica foi verificado que normalmente aplica-se uma camada de controle junto ao AGV, para que seja possível mantê-lo na trajetória planejada. O MPC também pode ser utilizado nesse caso, atuando como uma camada de rastreamento da trajetória.

A revisão bibliográfica e os resultados obtidos serviram, portanto, como base para a construção da arquitetura com a dupla camada de MPCs e computação na borda. Esta, provida de mais recursos computacionais, planeja a trajetória de múltiplos AGVs com a primeira camada de MPC, incluindo as restrições de prevenção de colisões. As trajetórias computadas nessa primeira camada são enviadas aos AGVs. Para que os AGVs sigam as trajetórias planejadas, uma segunda camada de MPC para rastreamento da trajetória é aplicada no lado local, junto de cada AGV. Desse modo, no servidor de borda são computadas as trajetórias de maneira global, ao passo que, no lado local, essas trajetórias são rastreadas, onde finalmente os sinais de controle gerados são repassados ao controlador de baixo nível.

Foram construídos quatro cenários para validação da arquitetura, conforme apresentado no Capítulo 6. A partir do Cenário 1, foi possível verificar que os AGVs seguem suas trajetórias até suas referências, sem colisões entre si e com os obstáculos fixos, inclusive quando são induzidos atrasos e perdas de pacotes na rede de comunicação. O Cenário 2 representa uma linha de produção hipotética onde a arquitetura pode ser aplicada, contando ainda com um Supervisor para alocação das tarefas e gerenciamento da ocupação das vagas. Na simulação com o cenário foram induzidos atrasos na rede de comunicação, entretanto, o funcionamento do cenário não foi prejudicado, assim como não houve colisões. Num espaço físico maior, o Cenário 3 representa um centro logístico, onde AGVs maiores carregam materiais entre docas e armazéns. O cenário também conta com o Supervisor para cumprir o mesmo papel exercido no Cenário 2.

Quando se analisa as métricas extraídas nas simulações, que são a média do intervalo de tempo das iterações do MPC e o uso dos recursos computacionais, como CPU e memória, no servidor de borda percebe-se que a média de tempo do MPC e o uso de CPU estão diretamente ligados ao tamanho do problema. Quanto mais AGVs no cenário, maiores os valores para essas métricas, visto que, com mais AGVs, cada nó de MPC no servidor de borda terá uma quantidade maior de trajetórias a serem consideradas na prevenção de colisões. Com o Cenário 4, criado exclusivamente para avaliar a escalabilidade da arquitetura, verificou-se o uso dos recursos computacionais quando se insere mais AGVs no cenário. Uma das estratégias para redução dessas métricas foi a limitação da quantidade de trajetórias a serem incluídas, implementadas nos Cenários 2 e 4, onde apenas uma determinada quantidade de AGVs mais próximos têm suas trajetórias consideradas no procedimento de prevenção de

colisões.

Em relação ao uso de memória dos recursos computacionais da borda, todos os cenários tiveram valores muito próximos, indicando que a magnitude do problema tem impacto significativo apenas no uso de CPU. No caso das métricas relacionadas ao lado local, não se observou alterações significativas entre os cenários, indicando que o computador embarcado no AGV pode ser de fato mais simples e de baixo custo, podendo ter a mesma configuração independente do cenário a ser aplicado.

Conforme já mencionado, foi evidenciado que as métricas relacionadas à borda são impactadas conforme a magnitude do problema. Portanto, em uma determinada aplicação os recursos computacionais na borda podem ser alocados conforme o cenário. Ainda assim, essas métricas podem ser reduzidas a partir da adoção de estratégias que limitam as quantidades de obstáculos fixos e trajetórias, conforme verificado nos resultados das simulações. Nesse sentido, um supervisor inteligente pode ser agregado à arquitetura, exercendo um papel que vai além do exercido pelo supervisor aplicado neste trabalho. Por exemplo, uma das medidas adotadas pode ser a divisão em áreas, onde o supervisor determina que os MPCs na borda dos AGVs que se localizam em uma determinada área, considerarão apenas os obstáculos e as trajetórias dos AGVs que estão na mesma área. Pode-se também determinar que um AGV que tem prioridade mais alta não precisará considerar as trajetórias dos que têm prioridade mais baixa, precisando ocorrer apenas o inverso. Estas e outras estratégias podem ser aplicadas para um uso mais inteligente dos recursos computacionais disponíveis.

Na perspectiva da IIoT e da Indústria 4.0, a arquitetura proposta pode ser utilizada como um serviço para controle de AGVs, juntamente com um supervisor para o gerenciamento da frota. Além disso, tendo sua parte principal funcionando no servidor de borda, a arquitetura pode ser integrada a outros serviços, como por exemplo, um sistema de alocação de tarefas. Considerando ainda o requisito da flexibilidade imposta pela Indústria 4.0, com a arquitetura é possível adaptar o funcionamento dos AGVs em qualquer leiaute, com baixo custo, dado que as configurações principais serão realizadas via software.

7.1 Sugestões de trabalhos futuros

Algumas sugestões para continuidade do trabalho, são:

- utilizar estratégias para detectar quando uma posição recebida do AGV está desatualizada, não gerando uma trajetória a partir dessa posição. Com isso evita-se o envio de trajetórias que contenham posições que AGV já percorreu, evitando consequentemente a diminuição da velocidade linear quando ocorrem atrasos na rede de comunicação entre a borda e o AGV.
- desenvolver um supervisor inteligente que possa informar aos nós do MPC de planejamento da trajetória quais obstáculos fixos e trajetórias serão consideradas na prevenção de colisões. Além de reduzir o problema do MPC, os nós não precisarão calcular periodicamente as distâncias com os obstáculos e outros AGVs para determinar com quais deles deverá se prevenir as colisões, reduzindo o uso de CPU por esse nós. A seleção das trajetórias dos AGVs a serem consideradas pode ser baseada na proximidade ou na prioridade, por exemplo.
- incorporar um sistema de odometria para que a arquitetura possa ser aplicada em um AGV real. Caso seja utilizada alguma técnica que exija o processamento de um maior volume de dados, como o SLAM ou a odometria visual, pode-se utilizar o servidor de borda para a execução dessa tarefa.
- incorporar um método de prevenção de colisões com obstáculos móveis, como pessoas ou outros agentes, de acordo com a percepção local realizada com sensores acoplados no AGV ou mesmo com o SLAM.
- projetar e montar amostras de AGVs reais ou utilizar robôs móveis comerciais que tenham capacidade de executar o ROS, a fim de realizar testes com a arquitetura proposta.

Bibliografia

- [1] M. Aazam, S. Zeadally, and K. A. Harras. Deploying fog computing in industrial internet of things and industry 4.0. *IEEE Transactions on Industrial Informatics*, 14(10):4674–4682, 2018.
- [2] Jannik Abbenseth, Felipe Garcia Lopez, Christian Henkel, and Stefan Dörr. Cloud-based cooperative navigation for mobile service robots in dynamic industrial environments. In *Proceedings of the Symposium on Applied Computing, SAC '17*, page 283–288, New York, NY, USA, 2017. Association for Computing Machinery.
- [3] Tarek Abdelzaher, Yifan Hao, Kasthuri Jayarajah, Archan Misra, Per Skarin, Shuochao Yao, Dulanga Weerakoon, and Karl-Erik Årzén. Five challenges in cloud-enabled intelligence and control. *ACM Trans. Internet Technol.*, 20(1), February 2020.
- [4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys Tutorials*, 17(4):2347–2376, 2015.
- [5] Amazon. Amazon robotics. <https://www.amazon.science/research-areas/robotics>. Acesso em: 24/01/2023.
- [6] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, In Press, 2018.
- [7] Rajesh Arumugam, Vikas Reddy Enti, Liu Bingbing, Wu Xiaojun, Krishnamoorthy Baskaran, Foong Foo Kong, A. Senthil Kumar, Kang Dee Meng, and Goh Wai Kit. Davinci: A cloud computing framework for service robots. In *2010 IEEE International Conference on Robotics and Automation*, pages 3084–3089, 2010.

-
- [8] Ning Bo, Xiangmin Li, Jinjin Dai, and Jiayu Tang. A hierarchical optimization strategy of trajectory planning for multi-uavs. In *2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, volume 1, pages 294–298, 2017.
- [9] H.G. Bock and K.J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems*. *IFAC Proceedings Volumes*, 17(2):1603–1608, 1984. 9th IFAC World Congress: A Bridge Between Control Science and Technology, Budapest, Hungary, 2-6 July 1984.
- [10] Robert Bogue. Growth in e-commerce boosts innovation in the warehouse robot market. *Industrial Robot: An International Journal*, 43:583–587, 10 2016.
- [11] E. F. Camacho and C. Bordons. *Introduction to Model Predictive Control*, pages 1–11. Springer London, London, 2007.
- [12] Elena Cardarelli, Valerio Digani, Lorenzo Sabattini, Cristian Secchi, and Cesare Fantuzzi. Cooperative cloud robotics architecture for the coordination of multi-agv systems in industrial warehouses. *Mechatronics*, 45:1–13, 2017.
- [13] CasADi. Documentation. <https://web.casadi.org/docs/>. Acesso em: 26/09/2022.
- [14] Michael Cheffena. Propagation channel characteristics of industrial wireless sensor networks [wireless corner]. *IEEE Antennas and Propagation Magazine*, 58(1):66–73, 2016.
- [15] B. Chen, J. Wan, A. Celesti, D. Li, H. Abbas, and Q. Zhang. Edge computing in iot-based manufacturing. *IEEE Communications Magazine*, 56(9):103–109, 2018.
- [16] Quan Chen, Hai Zhu, Lei Yang, Xiaoqian Chen, Sofie Pollin, and Evgenii Vinogradov. Edge computing assisted autonomous flight for uav: Synergies between vision and communications. *IEEE Communications Magazine*, 59(1):28–33, 2021.
- [17] Yuxiao Chen, Ugo Rosolia, and Aaron D. Ames. Decentralized task and path planning for multi-robot systems. *IEEE Robotics and Automation Letters*, 6(3):4337–4344, 2021.

-
- [18] João Paulo de Almeida, Renan Nakashima, Flávio Jr, and Lúcia Arruda. A global/local path planner for multi-robot systems with uncertain robot localization. *Journal of Intelligent & Robotic Systems*, 100:311–333, 2020.
- [19] Emanuele L. de Angelis, Fabrizio Giulietti, Goele Pipeleers, Gianluca Rossetti, and Ruben Van Parys. Optimal autonomous multirotor motion planning in an obstructed environment. *Aerospace Science and Technology*, 87:379–388, 2019.
- [20] Rômulo A. L. V. de Omena, Danilo F. S. Santos, Angelo Perkusich, and Dalton C. G. Valadares. Two-tier mpc architecture for agvs navigation assisted by edge computing in an industrial scenario. *Internet of Things*, 21:100666, 2023.
- [21] M. De Ryck, D. Pissoort, T. Holvoet, and E. Demeester. Decentral task allocation for industrial agv-systems with resource constraints. *Journal of Manufacturing Systems*, 59:310–319, 2021.
- [22] M. De Ryck, M. Versteyhe, and F. Debrouwere. Automated guided vehicle systems, state-of-the-art control algorithms and techniques. *Journal of Manufacturing Systems*, 54:152 – 173, 2020.
- [23] M. De Ryck, M. Versteyhe, and K. Shariatmadar. Resource management in decentralized industrial automated guided vehicle systems. *Journal of Manufacturing Systems*, 54:204–214, 2020.
- [24] Guillaume Demesure, Michael Defoort, Abdelghani Bekrar, Damien Trentesaux, and Mohamed Djemai. Decentralized motion planning and scheduling of agvs in an fms. *IEEE Transactions on Industrial Informatics*, 14(4):1744–1752, 2018.
- [25] Ivica Draganjac, Tamara Petrović, Damjan Miklič, Zdenko Kovačić, and Juraj Oršulić. Highly-scalable traffic management of autonomous industrial transportation systems. *Robotics and Computer-Integrated Manufacturing*, 63:101915, 2020.
- [26] R. Findeisen and P. Varutti. ‘ *Nonlinear Predictive Control over Nondeterministic Communication Networks*, pages 167–179. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

-
- [27] Sébastien Gros and Moritz Diehl. Numerical optimal control (draft), 2022.
- [28] Milan Groshev, Gabriele Baldoni, Luca Cominardi, Antonio de la Oliva, and Robert Gazda. Edge robotics: are we ready? an experimental evaluation of current vision and future directions. *Digital Communications and Networks*, 9(1):166–174, 2023.
- [29] Lars Grüne, Jürgen Pannek, and Karl Worthmann. A prediction based control scheme for networked systems with delays and packet dropouts. In *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 537–542, 2009.
- [30] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645 – 1660, 2013.
- [31] Mehmet Ali Guney and Ioannis A. Raptis. Dynamic prioritized motion coordination of multi-agv systems. *Robotics and Autonomous Systems*, 139:103534, 2021.
- [32] Jaakko Harjuhahto, Anton Debner, and Vesa Hirvisalo. Processing LiDAR Data from a Virtual Logistics Space. In Anton Cervin and Yang Yang, editors, *2nd Workshop on Fog Computing and the IoT (Fog-IoT 2020)*, volume 80 of *OpenAccess Series in Informatics (OASICs)*, pages 4:1–4:12, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.
- [33] Tamir Hegazy and Mohamed Hefeeda. Industrial automation as a cloud service. *IEEE Transactions on Parallel and Distributed Systems*, 26(10):2750–2763, 2015.
- [34] B. Houska, H.J. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.
- [35] Hazer Inaltekin, Maria Gorlatova, and Mung Chiang. Virtualized control over fog: Interplay between reliability and latency. *IEEE Internet of Things Journal*, 5(6):5030–5045, 2018.

- [36] Tao Jiang, Jianhua Zhang, Pan Tang, Lei Tian, Yi Zheng, Jianwu Dou, Henrik Asplund, Leszek Raschkowski, Raffaele D’Errico, and Tommi Jämsä. 3gpp standardized 5g channel model for iiot scenarios: A survey. *IEEE Internet of Things Journal*, 8(11):8799–8815, 2021.
- [37] Kelin Jose and Dilip Kumar Pratihar. Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods. *Robotics and Autonomous Systems*, 80:34–42, 2016.
- [38] Yu Kaneko, Yuhei Yokoyama, Nobuyuki Monma, Yoshiki Terashima, Keiichi Teramoto, Takuya Kishimoto, and Takeshi Saito. A microservice-based industrial control system architecture using cloud and mec. In Ajay Katangur, Shih-Chun Lin, Jinpeng Wei, Shuhui Yang, and Liang-Jie Zhang, editors, *Edge Computing – EDGE 2020*, pages 18–32, Cham, 2020. Springer International Publishing.
- [39] K. Kaur, S. Garg, G. S. Aujla, N. Kumar, J. J. P. C. Rodrigues, and M. Guizani. Edge computing in the industrial internet of things environment: Software-defined-networks-based edge-cloud interplay. *IEEE Communications Magazine*, 56(2):44–51, 2018.
- [40] A. Kiani and N. Ansari. Edge computing aware noma for 5g networks. *IEEE Internet of Things Journal*, 5(2):1299–1306, 2018.
- [41] Jens Lambrecht and Eugen Funk. Edge-enabled autonomous navigation and computer vision as a service: A study on mobile robot’s onboard energy consumption and computing requirements. In Manuel F. Silva, José Luís Lima, Luís Paulo Reis, Alberto Sanfeliu, and Danilo Tardioli, editors, *Robot 2019: Fourth Iberian Robotics Conference*, pages 291–302, Cham, 2020. Springer International Publishing.
- [42] Jens Lambrecht, Ernst Joachim Steffens, Marc Geitz, Axel Vick, Eugen Funk, and Wolfgang Steigerwald. Cognitive edge for factory: a case study on campus networks enabling smart intralogistics. In *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1325–1328, 2019.

- [43] X. Li, J. Wan, H. Dai, M. Imran, M. Xia, and A. Celesti. A hybrid computing solution and resource scheduling strategy for edge computing in smart manufacturing. *IEEE Transactions on Industrial Informatics*, 15(7):4225–4234, 2019.
- [44] Zhijun Li, Jun Deng, Renquan Lu, Yong Xu, Jianjun Bai, and Chun-Yi Su. Trajectory-tracking control of mobile robot systems incorporating neural-dynamic optimized model predictive approach. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46(6):740–749, 2016.
- [45] G. P. Liu, J. X. Mu, D. Rees, and S. C. Chai. Design and stability analysis of networked control systems with random communication time delay using the modified mpc. *International Journal of Control*, 79(4):288–297, 2006.
- [46] Felipe Garcia Lopez, Jannik Abbenseth, Christian Henkel, and Stefan Dörr. A predictive online path planning and optimization approach for cooperative mobile service robot navigation in industrial applications. In *2017 European Conference on Mobile Robots (ECMR)*, pages 1–6, 2017.
- [47] X. Lyu, H. Tian, L. Jiang, A. Vinel, S. Maharjan, S. Gjessing, and Y. Zhang. Selective offloading in mobile edge computing for the green internet of things. *IEEE Network*, 32(1):54–60, 2018.
- [48] Yehan Ma, Chenyang Lu, Bruno Sinopoli, and Shen Zeng. Exploring edge computing for multitier industrial control. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(11):3506–3518, 2020.
- [49] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.
- [50] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief. A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys Tutorials*, 19(4):2322–2358, 2017.
- [51] Diogo Matos, Pedro Costa, José Lima, and Paulo Costa. Multi agv coordination tolerant to communication failures. *Robotics*, 10(2), 2021.

- [52] Mohamed W. Mehrez. MPC and MHE implementation in Matlab using CasADi. <https://github.com/MMehrez/MPC-and-MHE-implementation-in-MATLAB-using-Casadi>. Acesso em: 22/09/2022.
- [53] Mohamed W. Mehrez, George K. I. Mann, and Raymond G. Gosine. Stabilizing nm-pc of wheeled mobile robots using open-source real-time software. In *2013 16th International Conference on Advanced Robotics (ICAR)*, pages 1–6, 2013.
- [54] Mohamed W. Mehrez, Karl Worthmann, Joseph P.V. Cenerini, Mostafa Osman, William W. Melek, and Soo Jeon. Model predictive control without terminal constraints or costs for holonomic mobile robots. *Robotics and Autonomous Systems*, 127:103468, 2020.
- [55] José M. Mendes Filho, Eric Lucet, and David Filliat. Real-time distributed receding horizon motion planning and control for mobile multi-robot dynamic systems. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 657–663, 2017.
- [56] Tim Mercy, Ruben Van Parys, and Goele Pipeleers. Spline-based motion planning for autonomous guided vehicles in a dynamic environment. *IEEE Transactions on Control Systems Technology*, 26(6):2182–2189, 2018.
- [57] Sherif A. S. Mohamed, Mohammad-Hashem Haghbayan, Tomi Westerlund, Jukka Heikkonen, Hannu Tenhunen, and Juha Plosila. A survey on odometry for autonomous navigation systems. *IEEE Access*, 7:97466–97486, 2019.
- [58] Gajamohan Mohanarajah, Dominique Hunziker, Raffaello D’Andrea, and Markus Waibel. Rapyuta: A cloud robotics platform. *IEEE Transactions on Automation Science and Engineering*, 12(2):481–493, 2015.
- [59] Winnie Nakimuli, Jaime Garcia-Reinoso, J. Enrique Sierra-Garcia, Pablo Serrano, and Isaac Quintana Fernández. Deployment and evaluation of an industry 4.0 use case over 5g. *IEEE Communications Magazine*, 59(7):14–20, 2021.

-
- [60] Rômulo Omena, Danilo Santos, and Angelo Perkusich. An approach to reduce network effects in an industrial control and edge computing scenario. In *Proceedings of the 11th International Conference on Cloud Computing and Services Science - CLOSER*, pages 296–303. INSTICC, SciTePress, 2021.
- [61] Rômulo Omena, Danilo Santos, and Angelo Perkusich. Arquitetura com computação na borda e controle preditivo baseado em modelo para controle de veículos autoguiados. In *Anais do XXIV Congresso Brasileiro de Automática*, 2022.
- [62] Emmanuel A. Oyekanlu, Alexander C. Smith, Windsor P. Thomas, Grethel Mulroy, Dave Hitesh, Matthew Ramsey, David J. Kuhn, Jason D. Mcghinnis, Steven C. Buonavita, Nickolus A. Looper, Mason Ng, Anthony Ng’oma, Weimin Liu, Patrick G. McBride, Michael G. Shultz, Craig Cerasi, and Dan Sun. A review of recent advances in automated guided vehicle technologies: Integration challenges and research areas for 5g-based smart manufacturing applications. *IEEE Access*, 8:202312–202353, 2020.
- [63] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu. Edge computing in industrial internet of things: Architecture, advances and challenges. *IEEE Communications Surveys Tutorials*, 22(4):2462–2488, 2020.
- [64] Albin Rajasingham. *Nonlinear model predictive control of combustion engines*. Springer, 2021.
- [65] J.B. Rawlings, D.Q. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2022.
- [66] Sarmad Riazi, Kristofer Bengtsson, and Bengt Lennartson. Energy optimization of large-scale agv systems. *IEEE Transactions on Automation Science and Engineering*, 18(2):638–649, 2021.
- [67] L. Riazuelo, Javier Civera, and J.M.M. Montiel. C2tam: A cloud framework for cooperative tracking and mapping. *Robotics and Autonomous Systems*, 62(4):401–413, 2014.

- [68] B. P. Rimal, D. P. Van, and M. Maier. Mobile edge computing empowered fiber-wireless access networks in the 5g era. *IEEE Communications Magazine*, 55(2):192–200, 2017.
- [69] Benjamin Rivière, Wolfgang Hönig, Yisong Yue, and Soon-Jo Chung. Glas: Global-to-local safe autonomy synthesis for multi-robot motion planning with end-to-end learning. *IEEE Robotics and Automation Letters*, 5(3):4249–4256, 2020.
- [70] Open Robotics. Gazebo. <https://gazebo.org/home>. Acesso em: 24/01/2023.
- [71] Open Robotics. ROS 2 Foxy Fitzroy documentation. <https://docs.ros.org/en/foxy/index.html>. Acesso em: 24/01/2023.
- [72] Ignacio Rodriguez, Rasmus S. Mogensen, Allan Schjørring, Mohammad Razzaghpour, Roberto Maldonado, Gilberto Berardinelli, Ramoni Adeogun, Per H. Christensen, Preben Mogensen, Ole Madsen, Charles Møller, Guillermo Pocovi, Troels Kolding, Claudio Rosa, Brian Jørgensen, and Simone Barbera. 5g swarm production: Advanced industrial manufacturing concepts enabled by wireless automation. *IEEE Communications Magazine*, 59(1):48–54, 2021.
- [73] Ignacio Rodriguez, Rasmus Suhr Mogensen, Andreas Fink, Taus Raunholt, Søren Markussen, Per Hartmann Christensen, Gilberto Berardinelli, Preben Mogensen, Casper Schou, and Ole Madsen. An experimental framework for 5g wireless system integration into industry 4.0 applications. *Energies*, 14(15), 2021.
- [74] Amir Salimi Lafmejani and Spring Berman. Nonlinear mpc for collision-free and deadlock-free navigation of multiple nonholonomic mobile robots. *Robotics and Autonomous Systems*, 141:103774, 2021.
- [75] Achilleas Santi Seisa, Björn Lindqvist, Sumeet Gajanan Satpute, and George Nikolakopoulos. E-cnmpc: Edge-based centralized nonlinear model predictive control for multiagent robotic systems. *IEEE Access*, 10:121590–121601, 2022.

- [76] Achilleas Santi Seisa, Sumeet Gajanan Satpute, Björn Lindqvist, and George Nikolopoulos. An edge-based architecture for offloading model predictive control for uavs. *Robotics*, 11(4), 2022.
- [77] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, 3(5):637–646, 2016.
- [78] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund. Industrial internet of things: Challenges, opportunities, and directions. *IEEE Transactions on Industrial Informatics*, 14(11):4724–4734, 2018.
- [79] Per Skarin, Johan Eker, Maria Kihl, and Karl-Erik Årzén. Cloud-assisted model predictive control. In *2019 IEEE International Conference on Edge Computing (EDGE)*, pages 110–112, 2019.
- [80] Per Skarin, Johan Eker, and Karl-Erik Årzén. Cloud-based model predictive control with variable horizon. *IFAC-PapersOnLine*, 53(2):6993–7000, 2020. 21th IFAC World Congress.
- [81] Per Skarin, Johan Eker, and Karl-Erik Årzén. A cloud-enabled rate-switching mpc architecture. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 3151–3158, 2020.
- [82] Per Skarin, William Tärneberg, Karl-Erik Årzen, and Maria Kihl. Towards mission-critical control at the edge and over 5g. In *2018 IEEE International Conference on Edge Computing (EDGE)*, pages 50–57, 2018.
- [83] Per Skarin, William Tärneberg, Karl-Erik Årzén, and Maria Kihl. Control-over-the-cloud: A performance study for cloud-native, critical control systems. In *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, pages 57–66, 2020.
- [84] Dmitrii Solomitckii, Antonino Orsino, Sergey Andreev, Yevgeni Koucheryavy, and Mikko Valkama. Characterization of mmwave channel properties at 28 and 60 ghz in factory automation deployments. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, 2018.

- [85] B. Sonkoly, R. Szabó, B. Németh, J. Czentye, D. Haja, M. Szalay, J. Dóka, B. P. Gerő, D. Jocha, and L. Toka. 5g applications from vision to reality: Multi-operator orchestration. *IEEE Journal on Selected Areas in Communications*, 38(7):1401–1416, 2020.
- [86] Peter Stenumgaard, Jose Chilo, Javier Ferrer-Coll, and Per Angskog. Challenges and conditions for wireless machine-to-machine communications in industrial environments. *IEEE Communications Magazine*, 51(6):187–192, 2013.
- [87] Huihui Sun, Weijie Zhang, Runxiang Yu, and Yujie Zhang. Motion planning for mobile robots—focusing on deep reinforcement learning: A systematic review. *IEEE Access*, 9:69061–69081, 2021.
- [88] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella. On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys Tutorials*, 19(3):1657–1681, 2017.
- [89] Günter Ullrich. *Automated Guided Vehicle Systems: A Primer with Practical Applications*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.
- [90] Stanislav Vakaruk, J. Enrique Sierra-García, Alberto Mozo, and Antonio Pastor. Forecasting automated guided vehicle malfunctioning with deep learning in a 5g-based industry 4.0 scenario. *IEEE Communications Magazine*, 59(11):102–108, 2021.
- [91] Dalton Valadares, Joseana Araújo, Angelo Perkusich, Marco Spohn, Elmar Melcher, Alexandre Costa, Felipe Ramos, and Natália Albuquerque. Performance evaluation of an ieee 802.11g network in an industrial environment. *IEEE Latin America Transactions*, 18(05):947–953, 2020.
- [92] Gijs van Lookeren Campagne. A nonlinear model predictive control based evasive manoeuvre assist function. Master’s thesis, Faculty of Mechanical, Maritime and Materials Engineering (3mE) - Delft University of Technology, 2019.
- [93] Axel Vick, Jan Guhl, and Jorg Krüger. Model predictive control as a service — concept and architecture for use in cloud-based robot control. In *2016 21st Internatio-*

- nal Conference on Methods and Models in Automation and Robotics (MMAR)*, pages 607–612, 2016.
- [94] Markus Waibel, Michael Beetz, Javier Civera, Raffaello D’Andrea, Jos Elfring, Dorian Gálvez-López, Kai Häussermann, Rob Janssen, J.M.M. Montiel, Alexander Perzylo, Björn Schieble, Moritz Tenorth, Oliver Zweigle, and René Van De Molengraft. Roboearth. *IEEE Robotics Automation Magazine*, 18(2):69–82, 2011.
- [95] Jiafu Wan, Shenglong Tang, Qingsong Hua, Di Li, Chengliang Liu, and Jaime Lloret. Context-aware cloud robotics for material handling in cognitive industrial internet of things. *IEEE Internet of Things Journal*, 5(4):2272–2281, 2018.
- [96] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.
- [97] Binhuai Xu and Jing Bian. A cloud robotic application platform design based on the microservices architecture. In *2020 International Conference on Control, Robotics and Intelligent System*, CCRIS 2020, page 13–18, New York, NY, USA, 2020. Association for Computing Machinery.
- [98] Li Da Xu, Eric L. Xu, and Ling Li. Industry 4.0: state of the art and future trends. *International Journal of Production Research*, 56(8):2941–2962, 2018.
- [99] P. Zhao and G. Dán. A benders decomposition approach for resilient placement of virtual process control functions in mobile edge clouds. *IEEE Transactions on Network and Service Management*, 15(4):1460–1472, 2018.
- [100] Hai Zhu, Francisco Martinez Claramunt, Bruno Brito, and Javier Alonso-Mora. Learning interaction-aware trajectory predictions for decentralized multi-robot motion planning in dynamic environments. *IEEE Robotics and Automation Letters*, 6(2):2256–2263, 2021.
- [101] Ángel Madridano, Abdulla Al-Kaff, David Martín, and Arturo de la Escalera. Trajectory planning for multi-robot systems: Methods and applications. *Expert Systems with Applications*, 173:114660, 2021.

Apêndice A

Código de exemplo para execução do MPC no CasADi e Matlab

A.1 Estabilização em um ponto

```
1 % Importar a versão Matlab do CasADi para o sistema operacional utilizado
2 addpath('~/casadi-osx-matlabR2015a-v3.5.5')
3 import casadi.*
4
5 %---Início da formulação do problema---%
6
7 T = 0.1; % passo de tempo em segundos
8 N = 20; % horizonte de predição
9
10 % Valores máximos e mínimos das variáveis de controle
11 v_max = 0.5; v_min = -v_max;
12 omega_max = pi/4; omega_min = -omega_max;
13
14 % Variáveis de estado
15 x = SX.sym('x'); y = SX.sym('y'); theta = SX.sym('theta');
16 estados = [x;y;theta]; n_estados = length(estados);
17
18 % Variáveis de controle
19 v = SX.sym('v'); omega = SX.sym('omega');
20 controle = [v;omega]; n_controle = length(controle);
```

```

21
22 % Modelo do sistema  $f(x,u)$ 
23 f = Function('f',{ estados , controle }, ...
24     {[v*cos(theta);v*sin(theta);omega]});
25
26 % Vetor que representa as variáveis de controle ao longo das predições
27 U = SX.sym('U',n_controle,N);
28
29 % Parâmetros (estado inicial e referência)
30 P = SX.sym('P',n_estados + n_estados);
31
32 % Vetor que representa os estados ao longo das predições, formando a
33 % trajetória
34 X = SX.sym('X',n_estados,(N+1));
35
36 f_custo = 0; % Inicialização da função custo
37 g = []; % Vetor de restrições
38
39 % Matriz Q (estados)
40 Q = zeros(3,3); Q(1,1) = 10; Q(2,2) = 10; Q(3,3) = 0.1;
41 % Matriz R (controle)
42 R = zeros(2,2); R(1,1) = 0.5; R(2,2) = 0.05;
43
44 st = X(:,1); % Estado inicial
45 g = [g;st-P(1:3)]; % Restrição no estado inicial (múltiplos disparos)
46 for k = 1:N
47     st = X(:,k); con = U(:,k);
48     f_custo = f_custo+(st-P(4:6))'*Q*(st-P(4:6)) + con'*R*con;
49     estado_prox = X(:,k+1);
50
51     % Início da integração RK4
52     k1 = f(st, con);
53     k2 = f(st + T/2*k1, con);
54     k3 = f(st + T/2*k2, con);
55     k4 = f(st + T*k3, con);
56     estado_proximo_RK4=st +T/6*(k1 +2*k2 +2*k3 +k4);
57     % Fim da integração RK4

```

```
58
59     % Preenchimento do vetor de restrições em cada passo de tempo
60     % (múltiplos disparos)
61     g = [g; estado_prox - estado_proximo_RK4];
62 end
63
64 % Adicionar as restrições para prevenção de colisões com obstáculos
65 % fixos
66 obs_x = 1.5;
67 obs_y = 2.5;
68 rob_diam = 0.3;
69 obs_diam = 0.3;
70 d_seg = 0.1;
71 for k = 1:N+1
72     g = [g ; -sqrt((X(1,k)-obs_x)^2+(X(2,k)-obs_y)^2) + ...
73         (rob_diam/2 + obs_diam/2 + d_seg)];
74 end
75
76 % Preencher o vetor das variáveis de otimização (X e U)
77 var_opt = [reshape(X,3*(N+1),1); reshape(U,2*N,1)];
78
79 % Criar o NLP
80 nlp = struct('f', f_custo, 'x', var_opt, 'g', g, 'p', P);
81
82 % Opcionais do IPOT
83 opts = struct;
84 opts.ipopt.max_iter = 2000;
85 opts.ipopt.print_level = 0;
86 opts.print_time = 0;
87 opts.ipopt.acceptable_tol = 1e-8;
88 opts.ipopt.acceptable_obj_change_tol = 1e-6;
89
90 % Criar o solucionador
91 solver = nlpso('solver', 'ipopt', nlp, opts);
92
93 % Iniciar o preenchimento dos argumentos das restrições
94 args = struct;
```

```

95
96 % Preenchimento dos vetores lbx e ubx com os limites máximos e mínimos
97 % dos estados e sinais de controle
98 args.lbx(1:3:3*(N+1),1) = -5; % x mínimo (m)
99 args.ubx(1:3:3*(N+1),1) = 5; % x máximo (m)
100 args.lbx(2:3:3*(N+1),1) = -5; % y mínimo (m)
101 args.ubx(2:3:3*(N+1),1) = 5; % y máximo (m)
102 args.lbx(3:3:3*(N+1),1) = -inf; % theta mínimo (rad)
103 args.ubx(3:3:3*(N+1),1) = inf; % theta máximo (rad)
104 args.lbx(3*(N+1)+1:2:3*(N+1)+2*N,1) = v_min; % v mínimo (m/s)
105 args.ubx(3*(N+1)+1:2:3*(N+1)+2*N,1) = v_max; % v máximo (m/s)
106 args.lbx(3*(N+1)+2:2:3*(N+1)+2*N,1) = omega_min; % omega mínimo (rad/s)
107 args.ubx(3*(N+1)+2:2:3*(N+1)+2*N,1) = omega_max; % omega máximo (rad/s)
108
109 % Preenchimento dos vetores lbg e ubg com zero. No processo de otimização
110 % esses valores são igualados ao vetor g preenchido acima, de modo a
111 % satisfazer a restrição do método dos múltiplos disparos
112 %  $x(k+1)-f(x(k+1),u(k+1)) = 0$ 
113 args.lbg(1:3*(N+1)) = 0; % Restrições de igualdade
114 args.ubg(1:3*(N+1)) = 0; % Restrições de igualdade
115
116 % O mesmo é feito para a restrição de prevenção de colisões
117 args.lbg(3*(N+1)+1 : 3*(N+1)+ (N+1)) = -inf;
118 args.ubg(3*(N+1)+1 : 3*(N+1)+ (N+1)) = 0;
119
120 %---Fim da formulação do problema---%
121
122 %---Início da simulação---%
123
124 t0 = 0; % Instante de tempo inicial
125 x0 = [0.0 ; 0.0 ; 0.0]; % Condição inicial [x0,y0,theta0]'
126 xs = [2.0 ; 4.0 ; pi]; % Referência [x,y,theta]'
127
128 % Armazenar a evolução da posição (estado) do robô no vetor xx
129 xx(:,1) = x0;
130
131 t(1) = t0; % Armazenar o instante t0 (utilizado nos gráficos)

```

```
132
133 % Inicializar o vetor das variáveis de otimização do sinal de controle
134 u0 = zeros(N,2);
135
136 % Inicializar o vetor das variáveis de otimização do estado
137 X0 = repmat(x0,1,N+1)';
138
139 tempo_sim = 12; % Tempo da simulação em segundos
140
141 % Inicializar a variável que armazena o número de iterações
142 iter_mpc = 0;
143
144 % Inicializar o vetor traj para armazenar as trajetórias geradas
145 traj = [];
146
147 % Inicializar o vetor u_cl para armazenar os sinais de controle aplicados
148 u_cl = [];
149
150 % Iniciar a contagem do intervalo de tempo de execução do loop
151 inicio_loop = tic;
152
153 % Início do loop.
154 % Enquanto a diferença entre o estado atual e a referência for menor que
155 % 0.01 e a quantidade de iterações for menor que tempo_sim/T, manter a
156 % execução do loop.
157 while(norm((x0-xs),2) > 1e-2 && iter_mpc < tempo_sim / T)
158     args.p = [x0;xs]; % parâmetros (estado inicial e referência)
159
160     % Valores iniciais das variáveis de otimização
161     args.x0 = [reshape(X0',3*(N+1),1);reshape(u0',2*N,1)];
162
163     % Chamar o solucionador e preencher o vetor sol com a solução (x e u)
164     sol = solver('x0', args.x0, 'lbx', args.lbx, 'ubx', args.ubx, ...
165         'lbg', args.lbg, 'ubg', args.ubg, 'p', args.p);
166
167     % Extrair a sequência de sinais de controle da solução em cada
168     % iteração
```

```

169     u = reshape( full( sol.x(3*(N+1)+1:end) )', 2, N)';
170
171     % Extrair e armazenar a trajetória da solução em cada iteração
172     traj(:, 1:3, iter_mpc+1) = reshape( full( sol.x(1:3*(N+1)) )', 3, N+1)';
173
174     % Armazenar o sinal de controle aplicado, ou seja, apenas o primeiro
175     % da sequência u
176     u_cl = [u_cl ; u(1,:)];
177
178     t(iter_mpc+1) = t0; % Armazenar o passo de tempo no vetor t
179
180     % Aplicar o sinal de controle no modelo do robô e atualizar as
181     % variáveis t0, x0 e u0 com a função shift.
182     % t0: atualização do passo de tempo.
183     % x0: recebe agora o estado atual do robô e é o estado inicial para a
184     % próxima iteração.
185     % u0: atualizado para ser o valor inicial da próxima iteração e
186     % acelerar a convergência. O primeiro valor da sequência u é
187     % descartado e passado para u0. O último valor é então repetido
188     % para manter a dimensão do vetor.
189     [t0, x0] = shift(T, t0, x0, u, f);
190
191     xx(:, iter_mpc+2) = x0; % Adicionar x0 no vetor xx (estado atual)
192
193     % Extrair a trajetória de estado da solução e armazenar em X0
194     X0 = reshape( full( sol.x(1:3*(N+1)) )', 3, N+1)';
195
196     % O vetor X0 é atualizado para ser o valor inicial da próxima
197     % iteração.
198     % O procedimento é o mesmo do u0, onde remove-se o primeiro valor e
199     % repete-se o último.
200     X0 = [X0(2:end, :); X0(end, :)];
201     u0 = [u(2:size(u,1), :); u(size(u,1), :)];
202
203     iter_mpc = iter_mpc + 1; % Atualizar o número de iterações
204 end
205 % Fim do loop

```

```

206
207 % Fim da simulação
208
209 % Registrar o intervalo de tempo de execução do loop
210 fim_loop = toc(inicio_loop);
211
212 % Calcular a média do tempo de execução do loop
213 media_tempo_loop = fim_loop/(iter_mpc+1);
214
215 % Calcular o erro (estado_final - ref)
216 erro = norm((x0-xs),2);
217
218 % Obs.: o código utilizado na geração dos gráficos não está incluído.

```

A.2 Rastreamento da trajetória

```

1 % Importar a versão Matlab do CasADi para o sistema operacional utilizado
2 addpath('~\casadi-osx-matlabR2015a-v3.5.5')
3 import casadi.*
4
5 %---Início da formulação do problema---%
6
7 T = 0.1; % passo de tempo em segundos
8 N = 80; % horizonte de predição
9
10 % Valores máximos e mínimos das variáveis de controle
11 v_max = 0.6; v_min = -v_max;
12 omega_max = pi/2; omega_min = -omega_max;
13
14 % Variáveis de estado
15 x = SX.sym('x'); y = SX.sym('y'); theta = SX.sym('theta');
16 estados = [x;y;theta]; n_estados = length(estados);
17
18 % Variáveis de controle
19 v = SX.sym('v'); omega = SX.sym('omega');
20 controle = [v;omega]; n_controle = length(controle);
21

```



```

22 % Modelo do sistema  $f(x,u)$ 
23 f = Function('f',{ estados , controle }, ...
24     {[v*cos(theta);v*sin(theta);omega]});
25
26 % Vetor que representa as variáveis de controle ao longo das predições
27 U = SX.sym('U',n_controle,N);
28
29 % Os parâmetros agora incluem o estado inicial e toda a referência
30 % ao longo do horizonte de predição
31 P = SX.sym('P',n_estados + N*(n_estados));
32
33 % Vetor que representa os estados ao longo das predições, formando a
34 % trajetória
35 X = SX.sym('X',n_estados,(N+1));
36
37 f_custo = 0; % Inicialização da função custo
38 g = []; % Vetor de restrições
39
40 % Matriz Q (estados)
41 Q = zeros(3,3); Q(1,1) = 1; Q(2,2) = 1; Q(3,3) = 0.1;
42 % Matriz R (controle)
43 R = zeros(2,2); R(1,1) = 0.5; R(2,2) = 0.05;
44
45 st = X(:,1); % Estado inicial
46 g = [g;st-P(1:3)]; % Restrição no estado inicial (múltiplos disparos)
47 for k = 1:N
48     st = X(:,k); con = U(:,k);
49     % A função custo agora considera todas as referências inseridas em P
50     f_custo = f_custo+(st-P(3*k+1:3*k+3))'*Q*(st-P(3*k+1:3*k+3)) + ...
51         con'*R*con;
52     estado_proximo = X(:,k+1);
53
54     % Início da integração RK4
55     k1 = f(st , con);
56     k2 = f(st + T/2*k1 , con);
57     k3 = f(st + T/2*k2 , con);
58     k4 = f(st + T*k3 , con);

```

```

59     estado_proximo_RK4=st +T/6*(k1 +2*k2 +2*k3 +k4);
60     g = [g;estado_proximo-estado_proximo_RK4];
61 end
62
63 % Preencher o vetor das variáveis de otimização (X e U)
64 var_opt = [reshape(X,3*(N+1),1);reshape(U,2*N,1)];
65
66 % Criar o NLP
67 nlp = struct('f', f_custo, 'x', var_opt, 'g', g, 'p', P);
68
69 % Opcionais do IPOT
70 opts = struct;
71 opts.ipopt.max_iter = 2000;
72 opts.ipopt.print_level =0;
73 opts.print_time = 0;
74 opts.ipopt.acceptable_tol =1e-8;
75 opts.ipopt.acceptable_obj_change_tol = 1e-6;
76
77 % Criar o solucionador
78 solver = nlpso('solver', 'ipopt', nlp ,opts);
79
80 % Iniciar o preenchimento dos argumentos das restrições
81 args = struct;
82
83 % Preenchimento dos vetores lbx e ubx com os limites máximos e mínimos
84 % dos estados e sinais de controle
85 args.lbx(1:3:3*(N+1),1) = -20; % x mínimo (m)
86 args.ubx(1:3:3*(N+1),1) = 20; % x máximo (m)
87 args.lbx(2:3:3*(N+1),1) = -20; % y mínimo (m)
88 args.ubx(2:3:3*(N+1),1) = 20; % y máximo (m)
89 args.lbx(3:3:3*(N+1),1) = -inf; % theta mínimo (rad)
90 args.ubx(3:3:3*(N+1),1) = inf; % theta máximo (rad)
91 args.lbx(3*(N+1)+1:2:3*(N+1)+2*N,1) = v_min; % v mínimo (m/s)
92 args.ubx(3*(N+1)+1:2:3*(N+1)+2*N,1) = v_max; % v máximo (m/s)
93 args.lbx(3*(N+1)+2:2:3*(N+1)+2*N,1) = omega_min; % omega mínimo (rad/s)
94 args.ubx(3*(N+1)+2:2:3*(N+1)+2*N,1) = omega_max; % omega máximo (rad/s)
95

```

```
96 % Preenchimento dos vetores lbg e ubg com zero. No processo de otimização
97 % esses valores são igualados ao vetor g preenchido acima, de modo a
98 % satisfazer a restrição do método dos múltiplos disparos
99 %  $x(k+1) - f(x(k+1), u(k+1)) = 0$ 
100 args.lbg(1:3*(N+1)) = 0; %  $-1e-20$  % Restrições de igualdade
101 args.ubg(1:3*(N+1)) = 0; %  $1e-20$  % Restrições de igualdade
102
103 %---Fim da formulação do problema---%
104
105 %---Início da simulação---%
106
107 t0 = 0; % Instante de tempo inicial
108 x0 = [0 ; 5 ; 0.0]; % Condição inicial [x0,y0,theta0]'
109
110 xx(:,1) = x0; % Armazenar a evolução da posição (estado) do robô no vetor
      xx
111 t(1) = t0; % Armazenar o instante t0 (utilizado nos gráficos)
112
113 % Inicializar o vetor das variáveis de otimização do sinal de controle
114 u0 = zeros(N,2);
115
116 % Inicializar o vetor das variáveis de otimização do estado
117 X0 = repmat(x0,1,N+1)';
118
119 tempo_sim = 130; % Tempo da simulação em segundos
120
121 % Inicializar a variável que armazena o número de iterações
122 iter_mpc = 0;
123
124 % Inicializar o vetor traj para armazenar as trajetórias geradas
125 traj = [];
126
127 % Inicializar o vetor u_cl para armazenar os sinais de controle aplicados
128 u_cl = [];
129
130 % Inicializar os vetores que armazenarão a referência ao longo do
131 % horizonte de predição
```

```
132 x_ref = zeros(1,N);
133 y_ref = zeros(1,N);
134 theta_ref = zeros(1,N);
135
136 % Iniciar a contagem do intervalo de tempo de execução do loop
137 inicio_loop = tic;
138
139 % Início do loop.
140 % Enquanto a quantidade de iterações for menor que tempo_sim/T, manter a
141 % execução do loop.
142 while(iter_mpc < tempo_sim / T)
143     args.p(1:3) = x0; % Estado inicial
144
145     % Início da geração da trajetória a ser utilizada como referência na
146     % iteração corrente
147     instante_atual = iter_mpc*T;
148     for k = 1:N
149
150         t_predicao = instante_atual + (k-1)*T; % Instante de tempo no
151         % horizonte
152         x_ref(k) = 3*sin(0.1*t_predicao); % x_ref
153         y_ref(k) = 5*cos(0.05*t_predicao); % y_ref
154
155         % Procedimento para obtenção do theta_ref
156         if iter_mpc == 0
157             theta_ref(k)=0;
158         elseif iter_mpc > 0 && k == 1
159             theta_ref(k) = atan2(y_ref(k)-y_ref_1, x_ref(k)-x_ref_1);
160         else
161             theta_ref(k) = atan2(y_ref(k)-y_ref(k-1), x_ref(k)-x_ref(k-1))
162             ;
163         end
164         u_ref = 0; omega_ref = 0;
165
166         % Armazenar a referência nos argumentos
167         args.p(3*k+1:3*k+3) = [x_ref(k), y_ref(k), theta_ref(k)];
168     end
169 end
```

```

167
168 % Armazenar o primeiro valor de posição da trajetória de referência
169 x_ref_1 = x_ref(1);
170 y_ref_1 = y_ref(1);
171 % Fim da geração da trajetória de referência
172
173 % Valores iniciais das variáveis de otimização
174 args.x0 = [reshape(X0',3*(N+1),1);reshape(u0',2*N,1)];
175
176 % Chamar o solucionador e preencher o vetor sol com a solução (x e u)
177 sol = solver('x0', args.x0, 'lbx', args.lbx, 'ubx', args.ubx,...
178 'lbg', args.lbg, 'ubg', args.ubg,'p',args.p);
179
180 % Extrair a sequência de sinais de controle da solução em cada
181 % iteração
182 u = reshape(full(sol.x(3*(N+1)+1:end))',2,N)';
183
184 % Extrair e armazenar a trajetória da solução em cada iteração
185 traj(:,1:3,iter_mpc+1)= reshape(full(sol.x(1:3*(N+1)))',3,N+1)';
186
187 % Armazenar o sinal de controle aplicado, ou seja, apenas o primeiro
188 % da sequência u
189 u_cl= [u_cl ; u(1,:)];
190
191 t(iter_mpc+1) = t0; % Armazenar o passo de tempo no vetor t
192
193 % Aplicar o sinal de controle no modelo do robô e atualizar as
194 % variáveis t0, x0 e u0 com a função shift.
195 % t0: atualização do passo de tempo.
196 % x0: recebe agora o estado atual do robô e é o estado inicial para a
197 % próxima iteração.
198 % u0: atualizado para ser o valor inicial da próxima iteração e
199 % acelerar a convergência. O primeiro valor da sequência u é
200 % descartado e passado para u0. O último valor é então repetido
201 % para manter a dimensão do vetor.
202 [t0, x0] = shift(T, t0, x0, u,f);
203

```

```
204     xx(:, iter_mpc+2) = x0; % Adicionar x0 no vetor xx (estado atual)
205
206     % Extrair a trajetória de estado da solução e armazenar em X0
207     X0 = reshape(full(sol.x(1:3*(N+1))))', 3, N+1);
208
209     % O vetor X0 é atualizado para ser o valor inicial da próxima
210     % iteração.
211     % O procedimento é o mesmo do u0, onde remove-se o primeiro valor e
212     % repete-se o último.
213     X0 = [X0(2:end, :); X0(end, :)];
214     u0 = [u(2:size(u,1), :); u(size(u,1), :)];
215
216     iter_mpc = iter_mpc + 1; % Atualizar o número de iterações
217 end
218
219 % Fim do loop
220
221 % Fim da simulação
222
223 % Registrar o intervalo de tempo de execução do loop
224 fim_loop = toc(inicio_loop);
225
226 % Calcular a média do tempo de execução do loop
227 media_tempo_loop = fim_loop/(iter_mpc+1);
228
229 % Obs.: o código utilizado na geração dos gráficos não está incluído.
```