Trabalho de Conclusão de Curso

Prevenção de colisões por MPC-ORCA em robôs móveis diferenciais

de **Glauber Rodrigues Leite**

orientado por
Prof. Dr. Heitor Judiss Savino

Universidade Federal de Alagoas
Instituto de Computação
Maceió, Alagoas
5 de Fevereiro de 2020

UNIVERSIDADE FEDERAL DE ALAGOAS
Instituto de Computação

PREVENÇÃO DE COLISÕES POR MPC-ORCA EM ROBÔS MÓVEIS DIFERENCIAIS

Trabalho de Conclusão de Curso submetido ao Instituto de Computação da Universidade Federal de Alagoas como requisito parcial para a obtenção do grau de Engenheiro de Computação.

Glauber Rodrigues Leite

Orientador: Prof. Dr. Heitor Judiss Savino

Banca Avaliadora:

Thiago Damasceno Cordeiro	Prof. Dr., IC-UFAL
Ícaro Bezerra Queiroz de Araújo	Prof. Dr., IC-UFAL

Maceió, Alagoas
5 de Fevereiro de 2020

UNIVERSIDADE FEDERAL DE ALAGOAS
Instituto de Computação

PREVENÇÃO DE COLISÕES POR MPC-ORCA EM ROBÔS MÓVEIS DIFERENCIAIS

Trabalho de Conclusão de Curso submetido ao Instituto de Computação da Universidade Federal de Alagoas como requisito parcial para a obtenção do grau de Engenheiro de Computação.

Aprovado em 5 de Fevereiro de 2020:

Heitor Judiss Savino,
Prof. Dr., Orientador

Thiago Damasceno Cordeiro,
Prof. Dr., IC-UFAL

Ícaro Bezerra Queiroz de Araújo,
Prof. Dr., IC-UFAL

Dedicatória

À minha família

Agradecimentos

Gostaria de agradecer à minha família, a meus pais Geovane e Luciene por proporcionarem um ambiente de valorização às conquistas na educação e no conhecimento. Igualmente à minha noiva Elian, que participou tanto das melhores, quanto das piores partes dessa trajetória, sendo uma pessoa que não só foi influenciada, mas também influenciou na tarefa de aprender a tomar as escolhas corretas. Também gostaria de agradecer a meus irmãos Geovane Filho e Gabriel, pois ao crescerem comigo e viverem experiências que geram lembranças, me ajudaram a concluir esta etapa e me ajudarão nas que virão. Um agradecimento especial a meu tio Antônio, *in memoriam*, por todo apoio crucial e sacrifício que prestou para possibilitar a realização deste curso.

Gostaria de agradecer ao meu orientador Prof. Heitor Savino, pois ofereceu a mudança de paradigma que eu procurava, me guiando em temas que imaginava tão distantes e que hoje me sinto sortudo em ter a oportunidade de trabalhar. Não posso deixar de agradecer também ao corpo docente do Instituto de Computação, em especial aos professores do seguimento de Engenharia, Prof. Davi Bibiano, Prof. João Raphael, Prof. Ícaro Bezerra, Prof. Thiago Cordeiro e Prof. Tiago Vieira, que além de fornecerem o conhecimento em suas disciplinas, me ofereceram oportunidades e experiências além da sala de aula.

Gostaria de agradecer aos meus companheiros do laboratório EASY-SPARC, que começaram sendo parceiros de aulas e construíram uma amizade, me ensinando que é mais fácil do que eu imaginava tomar a frente das coisas, sejam projetos ou metas, Andressa Martins, Arthur Vangasse, Bruno Georgevich, Eduardo Miranda, Gregory Albertt, Luís Felipe e Maria Júlia.

24 de Janeiro de 2020, Maceió - AL

Desperto nos carros os instintos mais primitivos.

Relâmpago McQueen, Carros

Resumo

Este trabalho apresenta o desenvolvimento de uma solução distribuída utilizando controle preditivo (MPC) em conjunto com o algoritmo de prevenção de colisão ORCA aplicado a robôs móveis seguindo trajetórias individuais. São considerados robôs com rodas diferenciais se movendo em um plano, definidos pela sua posição e controlados por comandos de velocidade. A partir do modelo explícito do sistema, são computadas as ações de controle ótimas em um horizonte de predição de acordo com uma função custo, ou objetivo, a ser minimizada. Ainda nesse problema de otimização, são incorporadas restrições de velocidade resultantes do algoritmo de prevenção de colisão. O MPC é uma metodologia capaz de lidar com sistemas de controle com múltiplas variáveis eficientemente, uma vez que pode ser implementado utilizando um modelo representado em espaço de estados e resolvido por um programa quadrático convexo (QP). Os resultados das simulações mostram que a estratégia combinada MPC-ORCA implementada resultou em rotas suaves e livres de colisão em um ambiente constantemente variável, sem demandar replanejamento de trajetórias, nem comunicação direta entre os agentes.

***Palavras-chave*: *Model Predictive Control*; *Robótica distribuída*; Sistemas de controle multivariáveis.**

Abstract

This work describes the development of a distributed solution using predictive control (MPC) including the collision avoidance algorithm ORCA applied to mobile robots following individual trajectories. Differential drive robots are used, defined by their position on the plane and controlled by velocity commands. Based on a explicit system model, optimal control actions are computed over a prediction horizon according to a cost function to be minimized. This optimization problem is also subject to velocity constraints designated by the collision avoidance algorithm. MPC is a methodology that can handle multivariable control systems efficiently, since it can be implemented using state space model representation and be solved on a convex quadratic program (QP). Simulation results show that the developed combined strategy MPC-ORCA provided smooth and collision-free routes on a constantly changing environment, requiring no trajectory replanning, neither direct communication between agents.

***Keywords*: *Model Predictive Control*; *Cooperative Robots*; *MIMO Control Systems*.**

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

$\mathbb{N}$ Conjunto dos números naturais.

$\mathbb{R}$ Conjunto dos números reais.

$\dot{x}(t)$ Derivada de $x(t)$ no tempo.

$M^\top$ Transposta da matriz M.

I_n Matriz identidade de ordem n.

$\otimes$ Produto de Kronecker.

$\oplus$ Soma de Minkowski.

Lista de Abreviaturas

ADMM *Alternating Direction Method of Multipliers*

DMC *Dynamic Matrix Control*

LTI *Linear Time Invariant*

MIMO *Multiple Input Multiple Output*

MPC *Model Predictive Control*

ORCA *Optimal Reciprocal Collision Avoidance*

OSQP *Operator Splitting Quadratic Program*

QDMC *Quadratic Dynamic Matrix Control*

QP *Quadratic Programming*

RVO *Reciprocal Velocity Obstacles*

SISO *Single Input Single Output*

VO *Velocity Obstacles*

Sumário

Capítulo 1

Introdução

No ambiente industrial, é comum a presença de manipuladores robóticos para trabalhos de montagem, pintura e soldagem de componentes com grande precisão e velocidade. Todavia, grande parte desses robôs encontram-se fixados em uma posição específica tendo seu espaço de trabalho limitado pela falta de mobilidade [Roland, 2004]. Robôs móveis são capazes de relaxar essa restrição, expandindo a quantidade de aplicações que sistemas robóticos autônomos podem assumir, dentre elas, automação de armazéns [D'Andrea, 2012], operação em ambientes perigosos [Trevelyan et al., 2016] e *smart farming* na agroindústria [Hartanto et al., 2019].

Com o desenvolvimento de tecnologias computacionais cada vez mais poderosas, compactas e acessíveis, a utilização de robôs móveis tem deixado de se restringir exclusivamente a soluções industriais, crescendo também em outros setores, como o de serviços [Graetz and Michaels, 2018]. O projeto de sistemas robóticos móveis engloba um conjunto de áreas envolvendo aspectos da mecânica, eletrônica e computação. Dentro desse contexto, algoritmos de navegação de robôs foram desenvolvidos com o intuito de fornecer mecanismos computacionais que permitam que o robô percorra o ambiente para atingir seu objetivo. Para isso, relaciona-se um conjunto de tarefas interdependentes, como localização, mapeamento do ambiente, planejamento de trajetórias e controle do movimento.

Na maioria das aplicações, o robô não está sozinho, podendo existir em seu local de trabalho obstáculos, humanos ou mesmo outros agentes com objetivos próprios, caracterizando um sistema multi-agente. Um dos problemas fundamentais de sistemas robóticos distribuídos é a prevenção de colisão, uma vez que o desempenho do grupo não é programado diretamente, de forma que o desenvolvimento da solução é limitado à implementação individual de cada agente [Ota, 2006, Sycara, 1998]. Nesse cenário, o planejamento *offline* de uma rota não é capaz de garantir uma trajetória livre de colisões, exigindo o replanejamento eventual ou contínuo através da reexecução de algoritmos como **A***, **D*** e Campos Potenciais [Stentz, 1995, Mouad et al., 2012].

A categoria de algoritmos conhecida por algoritmos de controle local é composta por

técnicas que permitem ao robô reagir *online* aos demais agentes e obstáculos móveis de sua vizinhança, fornecendo desvios necessários na rota definida pelo planejador *offline* para assegurar uma trajetória livre de colisões. Os algoritmos consistem na demarcação de regiões de velocidades que colocariam o agente em situação de colisão com um obstáculo móvel em um momento futuro, devendo escolher o comando de velocidade subsequente fora dessa região, mas próximo da velocidade desejada definida pelo planejador *offline*.

Dentre os algoritmos de controle local, o mais fundamental é o *Velocity Obstacles* (VO) [Fiorini and Shiller, 2002], que manteve seu conceito no decorrer do tempo, mas sofreu alterações na sua definição matemática para suportar o *Reciprocal Velocity Obstacles* (RVO) [Van den Berg et al., 2008] e foi reformulada para a utilização no *Optimal Reciprocal Collision Avoidance* (ORCA) [Van Den Berg et al., 2011]. O *Reciprocal Velocity Obstacles* assume que existem outros agentes que realizarão um esforço recíproco para prevenir colisões, resultando em movimentos mais suaves e menos propensos a oscilações. Já o *Optimal Reciprocal Collision Avoidance* apresenta regiões com propriedades interessantes que facilitam a aplicação em problemas de otimização convexa, sendo eficiente mesmo em cenários mais densos.

Controle Preditivo baseado em Modelo (MPC, do inglês *Model Predictive Control*) é uma metodologia para projetar sistemas de controle ótimo utilizando predições a partir de um modelo explícito do processo, sendo bastante usado em problemas de controle multivariável que devem lidar com restrições [Camacho and Bordons, 2007]. Além de se destacar em aplicações de controle de processo na indústria química [Ricker and Lee, 1995, Ehlinger and Mesbah, 2017], estratégias MPC já foram utilizadas em aplicações robóticas como controle de trajetória para manipulador robótico [Poignet and Gautier, 2000] e robô móvel [Bouzoualegh et al., 2019]. Não obstante, podem ser encontrados trabalhos aplicando implementações distribuídas do MPC para controle descentralizado de formação de satélites [Carpenter, 2002], distorções harmônicas em sistemas de transmissão de energia [Skjong et al., 2019] e alocação inteligente de recursos energéticos [Barata et al., 2014].

Uma estratégia combinada MPC-ORCA foi proposta para movimentar um robô até uma posição objetivo usando controle preditivo enquanto evita situações de colisão com outros agentes a partir das restrições geradas pelo ORCA [Cheng et al., 2017]. O presente trabalho implementa e adaptada essa estratégia em robôs com mecanismo de locomoção por rodas diferenciais, sujeitos a restrições cinemáticas, e permite que sejam oferecidas trajetórias de referência genéricas.

1.1 Justificativa

O uso crescente de robôs em plantas industriais está diretamente relacionado ao aumento da produtividade, levando à densificação de sistemas robóticos compartilhando o mesmo

chão de fábrica [Graetz and Michaels, 2018]. No paradigma moderno de indústria, a Indústria 4.0, o desenvolvimento de sistemas ciber-físicos cada vez mais descentralizados e autônomos se tornou vital, principalmente no que se refere à escalabilidade das soluções empregadas [Lu, 2017]. Algumas vantagens podem ser apontadas como motivação para o desenvolvimento de soluções usando múltiplos agentes, incluindo o paralelismo inerente para execução mais rápida de um objetivo global, a simplificação de alguns problemas através de estratégias de decomposição e alocação de tarefas, além de existir situações em que podem ser usados múltiplos robôs mais simples com funcionalidades particulares ao invés de um único robô mais complexo [Arkin and Balch, 1998].

Sabe-se que o problema de controle local, ou planejamento dinâmico de movimentação, de um ponto no plano é NP-difícil em um ambiente com obstáculos móveis e restrições de velocidade sendo, portanto, um problema de relevância computacional [Canny and Reif, 1987]. Ainda assim, a utilização da estratégia combinada MPC-ORCA é devidamente qualificada para esse problema, uma vez que o MPC é capaz de lidar facilmente com cenários de controle com múltiplas variáveis e pode ser implementado usando técnicas de otimização matemática computacionalmente eficientes, suportando as regiões geradas pelo ORCA sob condições de otimalidade global.

1.2 Objetivos

1.2.1 Objetivos Gerais

Obter um sistema de controle em que cada agente tenha sua trajetória individual e que estes possam atualizar a trajetória em tempo real, a nível de controle, ou seja, sem replanejamento, para evitar a colisão. O sistema deve ser distribuído e não exigir comunicação direta entre os robôs, de forma que os algoritmos de navegação são executados baseando-se nas leituras locais de posição e velocidade dos demais agentes.

1.2.2 Objetivos Específicos

- Estudar um modelo matemático para robô móvel utilizando locomoção a partir de rodas diferenciais;
- Implementar o controle de movimento para um único robô utilizando controle preditivo (MPC);
- Desenvolver um sistema distribuído usando controle preditivo capaz de prevenir colisões com outros agentes;
- Analisar o comportamento do controlador.

1.3 Organização do Trabalho

O restante do estudo está organizado em capítulos que descrevem cada etapa do desenvolvimento do trabalho, fornecendo detalhes teóricos e aplicados da implementação da solução. O Capítulo 2 apresenta a definição do problema. Isto é, partindo das equações que descrevem a movimentação do robô, é estabelecido um modelo matemático que permite ao MPC realizar predições sobre os estados futuros das variáveis de interesse. O Capítulo 3 detalha o funcionamento do MPC e o projeto de um controlador preditivo usando otimização matemática. Com isso, é possível obter um sistema de controle de movimento para um único robô. O Capítulo 4 expõe as subáreas envolvidas na navegação de robôs e se concentra nos algoritmos *online* de prevenção de colisão entre agentes, concluindo na determinação da estratégia combinada MPC-ORCA. O Capítulo 5 discute os resultados adquiridos a partir dos cenários de simulação definidos, em conformidade com os objetivos propostos.

Capítulo 2

Definição do problema

Este capítulo apresenta uma base para a descrição de um robô móvel em aplicações de controle com modelo de sistema de segunda ordem, considerando restrições no movimento, linearização e discretização do problema. Com isso, é desenvolvido um modelo que permite estimar posições e velocidades futuras de um robô a partir de comandos de aceleração.

2.1 Configuração de um robô móvel

Define-se como configuração $\boldsymbol{q} \in \mathbb{Q}$ de um robô a especificação completa da posição de todos os pontos do sistema, sendo $\mathbb{Q}$ o conjunto de todas as possíveis configurações que o robô pode assumir. Assim, o número de *graus de liberdade* é a quantidade mínima de parâmetros necessários para descrever sua configuração [Choset et al., 2005]. De acordo com a aplicação, a especificação completa do robô pode ser inteiramente representada pela posição do centro de um robô circular no plano, considerando conhecido seu raio, ou pelos ângulos das juntas de um manipulador, considerando conhecidas as dimensões dos elos e as transformações dos eixos.

Um robô móvel precisa de mecanismos de locomoção para se mover num ambiente, levando em consideração algumas características como uniformidade, rigidez e atrito do terreno. Dentre as estratégias de locomoção, as caracterizadas por articulações requerem mais graus de liberdade e, consequentemente, maior complexidade mecânica do que as realizadas por rodas [Roland, 2004]. Entretanto, em um nível mais alto de assimilação de tarefas, ambas estratégias podem trabalhar em cima de um comando de velocidade ou aceleração da configuração do sistema.

Neste trabalho, os robôs são representados por entidades circulares de raio $\lambda \in \mathbb{R}$ se movendo no plano $\{\boldsymbol{x}_0, \boldsymbol{y}_0\}$, definido no sistema de coordenadas global $\mathcal{F}_0 = \{\boldsymbol{O}_0, \boldsymbol{x}_0, \boldsymbol{y}_0, \boldsymbol{z}_0\}$ com origem em $\boldsymbol{O}_0$, e eixos ortogonais unitários $\{\boldsymbol{x}_0, \boldsymbol{y}_0, \boldsymbol{z}_0\}$. Seja $\mathcal{F}_\mathrm{R} = \{\boldsymbol{O}_\mathrm{R}, \boldsymbol{x}_\mathrm{R}, \boldsymbol{y}_\mathrm{R}\}$ o sistema de coordenadas do robô, com origem em $\boldsymbol{O}_\mathrm{R}$, que nas coordenadas globais pode ser escrito como $\boldsymbol{O}_\mathrm{R} = (p_{\boldsymbol{x}_0}, p_{\boldsymbol{y}_0})$, sendo $p_{\boldsymbol{x}_0}, p_{\boldsymbol{y}_0} \in \mathbb{R}$ e $\{\boldsymbol{x}_\mathrm{R}, \boldsymbol{y}_\mathrm{R}, \boldsymbol{z}_\mathrm{R}\}$ eixos ortogonais unitários, como mostrado na Figura 2.1. Considere-se o

ângulo $\theta_{\boldsymbol{z}_0} \in \mathbb{S}^1$ que determina a rotação entre o sistema de referência global $\mathcal{F}_0$ e o sistema $\mathcal{F}_\mathrm{R}$ preso ao robô. Dessa forma, a configuração de um robô pode ser descrita como $\boldsymbol{q} = [p_{\boldsymbol{x}_0}\ p_{\boldsymbol{y}_0}\ \theta_{\boldsymbol{z}_0}]^\top \in \mathbb{Q} \subset \mathbb{R}^2 \times \mathbb{S}^1$ relacionada ao sistema de referencial global.

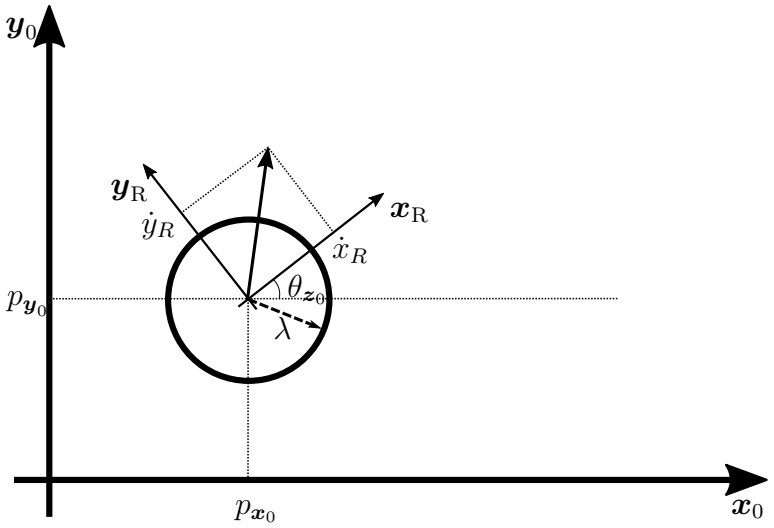

Figura 2.1: Robô circular com orientação, apresentando três graus de liberdade

A relação de orientação entre o sistema de coordenadas fixo do mundo $\mathcal{F}_0$ e o sistemas de coordenadas $\mathcal{F}_\mathrm{R}$ fixo no corpo do robô é dada pela matriz de rotação $\boldsymbol{R}^0_\mathrm{R}$, de forma que um ponto (ou vetor) descrito no referencial do robô pode ser representado no referencial global. A matriz de rotação pode ser utilizada para representar a velocidade local do robô $\boldsymbol{v}^\mathrm{R} = [v_{\boldsymbol{x}_\mathrm{R}}\ v_{\boldsymbol{y}_\mathrm{R}}]^\top \in \mathbb{R}^2$, sendo $v_{\boldsymbol{x}_\mathrm{R}} = \dot{p}_{\boldsymbol{x}_\mathrm{R}} \in \mathbb{R}$ e $v_{\boldsymbol{y}_\mathrm{R}} = \dot{p}_{\boldsymbol{y}_\mathrm{R}} \in \mathbb{R}$ as velocidades do robô nos eixos $\boldsymbol{x}_\mathrm{R}$ e $\boldsymbol{y}_\mathrm{R}$, respectivamente, no sistemas de coordenadas $\mathcal{F}_0$ fazendo

$$\boldsymbol{v}^0 = \boldsymbol{R}^0_\mathrm{R}\boldsymbol{v}^\mathrm{R} = \begin{bmatrix} \cos\theta_{\boldsymbol{z}_0} & -\sin\theta_{\boldsymbol{z}_0} \\ \sin\theta_{\boldsymbol{z}_0} & \cos\theta_{\boldsymbol{z}_0} \end{bmatrix} \boldsymbol{v}^\mathrm{R}, \tag{2.1}$$

sendo $\boldsymbol{v}^0 = [v_{\boldsymbol{x}_0}\ v_{\boldsymbol{y}_0}]^\top \in \mathbb{R}^2$ a velocidade do robô representada no referencial inercial, com $v_{\boldsymbol{x}_0} = \dot{p}_{\boldsymbol{x}_0} \in \mathbb{R}$ e $v_{\boldsymbol{y}_0} = \dot{p}_{\boldsymbol{y}_0} \in \mathbb{R}$ as velocidades do robô nos eixos $\boldsymbol{x}_0$ e $\boldsymbol{y}_0$, respectivamente.

2.1.1 Restrições cinemáticas de um robô móvel diferencial

Robôs móveis diferenciais são sujeitos restrições não-holonômicas (ou restrições cinemáticas) que limitam o movimento instantâneo, reduzindo o espaço de velocidades admissíveis a partir de determinada configuração. Ainda assim, é possível alcançar muitas configurações através de manobras apropriadas [Siciliano et al., 2009]. A movimentação do robô com rodas diferenciais é caracterizada por duas condições não-holonômicas [Ahmad Abu Hatab, 2013]:

- O robô não pode se mover lateralmente, ou seja, $v_{\boldsymbol{y}_\mathrm{R}} = 0$;

- Em cada roda, não há derrapagem no eixo longitudinal ($\boldsymbol{x}_\mathrm{R}$) ou deslize no eixo ortogonal ($\boldsymbol{y}_\mathrm{R}$).

Isto implica na definição de $\boldsymbol{v}^0 = v_{\boldsymbol{x}_\mathrm{R}}\boldsymbol{x}_\mathrm{R}$ como a velocidade linear e $\boldsymbol{\omega}^0 = \omega_{\boldsymbol{z}_\mathrm{R}}\boldsymbol{z}_\mathrm{R}$, sendo $\omega_{\boldsymbol{z}_\mathrm{R}} = \dot{\theta}_{\boldsymbol{z}_\mathrm{R}} = \dot{\theta}_{\boldsymbol{z}_0}$, como a velocidade angular.

Um conjunto de equações para o movimento do robô com rodas diferenciais pode ser obtido aplicando estas condições não-holonômicas na Equação (2.1). Considerando as componentes de segunda ordem $a_{\boldsymbol{x}_\mathrm{R}} = \dot{v}_{\boldsymbol{x}_\mathrm{R}}$ e $\alpha_{\boldsymbol{z}_\mathrm{R}} = \dot{\omega}_{\boldsymbol{z}_\mathrm{R}}$ como as entradas de aceleração linear e rotacional relacionadas a força e torque, respectivamente, tem-se

$$\begin{aligned}
\dot{p}_{\boldsymbol{x}_0} &= v_{\boldsymbol{x}_\mathrm{R}} \cos\theta_{\boldsymbol{z}_0}, \\
\dot{p}_{\boldsymbol{y}_0} &= v_{\boldsymbol{x}_\mathrm{R}} \sin\theta_{\boldsymbol{z}_0}, \\
\dot{\theta}_{\boldsymbol{z}_0} &= \omega_{\boldsymbol{z}_\mathrm{R}}, \\
\dot{v}_{\boldsymbol{x}_\mathrm{R}} &= a_{\boldsymbol{x}_\mathrm{R}}, \\
\dot{\omega}_{\boldsymbol{z}_\mathrm{R}} &= \alpha_{\boldsymbol{z}_\mathrm{R}}.
\end{aligned} \tag{2.2}$$

2.1.2 Aplicação do comando de entrada

Algumas interfaces de robôs móveis recebem um comando de velocidade na representação polar como entrada, ao invés da representação cartesiana. Portanto, um comando de velocidade $\boldsymbol{v}^0 = [v_{\boldsymbol{x}_0}\ v_{\boldsymbol{y}_0}]^\top$ ou aceleração $\boldsymbol{a}^0 = [a_{\boldsymbol{x}_0}\ a_{\boldsymbol{y}_0}]^\top$ usando a representação vetorial cartesiana podem ser transformados em um comando de velocidade usando a representação polar a partir das relações [Sharma K. et al., 2016]

$$\begin{aligned}
\nu = v_{\boldsymbol{x}_\mathrm{R}} &= \pm\sqrt{v_{\boldsymbol{x}_0}^2 + v_{\boldsymbol{y}_0}^2}, \\
\omega = \omega_{\boldsymbol{z}_\mathrm{R}} &= \frac{a_{\boldsymbol{y}_0}v_{\boldsymbol{x}_0} - a_{\boldsymbol{x}_0}v_{\boldsymbol{y}_0}}{(v_{\boldsymbol{x}_0}^2 + v_{\boldsymbol{y}_0}^2)}.
\end{aligned} \tag{2.3}$$

De acordo com a velocidade angular ω calculada, pode ser interessante definir uma velocidade linear negativa $\nu < 0$. Outra abordagem para calcular ω utiliza o valor atual da orientação $\theta_{\boldsymbol{z}_\mathrm{R}}$ e a orientação definida pelo vetor de velocidade, como

$$\omega = \Delta\theta_{\boldsymbol{z}_0} = \mathrm{atan2}(v_{\boldsymbol{y}_0}, v_{\boldsymbol{x}_0}) - \theta_{\boldsymbol{z}_0}. \tag{2.4}$$

A função $\mathrm{atan2} : \mathbb{R} \times \mathbb{R} \to \mathbb{S}$ define o arco tangente com valores não ambíguos nos quatro quadrantes do plano, resultando em um ângulo no intervalo fechado $[-\pi\ \pi]$.

2.1.3 Linearização

As restrições não-holonômicas do robô apresentam não linearidades que impossibilitam a construção de um modelo de sistema linear invariante no tempo (LTI, do inglês *Linear*

Time Invariant) a partir das Equações (2.2). Para evitar os problemas causados por essas restrições, define-se uma pequena distância d em no eixo central $\boldsymbol{x}_{\mathrm{R}}$, como mostrado na Figura 2.2, de forma que podem ser realizadas pequenas manobras para posicionar o robô nesse ponto de operação.

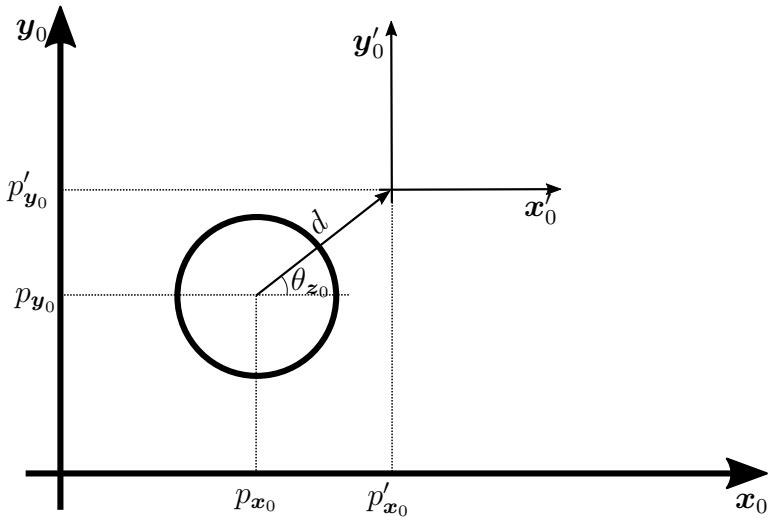

Figura 2.2: Linearização de robô com rodas diferenciais.

A relação entre o centro do robô e o referencial de linearização [Ren and Atkins, 2007] é dada por

$$\begin{bmatrix} p'_{\boldsymbol{x}_0} \\ p'_{\boldsymbol{y}_0} \end{bmatrix} = \begin{bmatrix} p_{\boldsymbol{x}_0} \\ p_{\boldsymbol{y}_0} \end{bmatrix} + d \begin{bmatrix} \cos\theta_{\boldsymbol{z}_0} \\ \sin\theta_{\boldsymbol{z}_0} \end{bmatrix}. \tag{2.5}$$

Tomando a primeira derivada no tempo da Equação (2.5), tem-se

$$\begin{bmatrix} \dot{p}'_{\boldsymbol{x}_0} \\ \dot{p}'_{\boldsymbol{y}_0} \end{bmatrix} = \begin{bmatrix} \dot{p}_{\boldsymbol{x}_0} \\ \dot{p}_{\boldsymbol{y}_0} \end{bmatrix} + d \begin{bmatrix} -\dot{\theta}_{\boldsymbol{z}_0} \sin\theta_{\boldsymbol{z}_0} \\ \dot{\theta}_{\boldsymbol{z}_0} \cos\theta_{\boldsymbol{z}_0} \end{bmatrix}, \tag{2.6}$$

e a segunda derivada se torna

$$\begin{bmatrix} \ddot{p}'_{\boldsymbol{x}_0} \\ \ddot{p}'_{\boldsymbol{y}_0} \end{bmatrix} = \begin{bmatrix} \ddot{p}_{\boldsymbol{x}_0} \\ \ddot{p}_{\boldsymbol{y}_0} \end{bmatrix} + d \begin{bmatrix} -\ddot{\theta}_{\boldsymbol{z}_0} \sin\theta_{\boldsymbol{z}_0} - \dot{\theta}^2_{\boldsymbol{z}_0} \cos\theta_{\boldsymbol{z}_0} \\ \ddot{\theta}_{\boldsymbol{z}_0} \cos\theta_{\boldsymbol{z}_0} - \dot{\theta}^2_{\boldsymbol{z}_0} \sin\theta_{\boldsymbol{z}_0} \end{bmatrix}. \tag{2.7}$$

A partir da Equação (2.2) é possível obter

$$\begin{aligned} \ddot{p}_{\boldsymbol{x}_0} &= \dot{v}_{\boldsymbol{x}_{\mathrm{R}}} \cos\theta_{\boldsymbol{z}_0} - v_{\boldsymbol{x}_{\mathrm{R}}} \dot{\theta}_{\boldsymbol{z}_0} \sin\theta_{\boldsymbol{z}_0}, \\ \ddot{p}_{\boldsymbol{y}_0} &= \dot{v}_{\boldsymbol{x}_{\mathrm{R}}} \sin\theta_{\boldsymbol{z}_0} + v_{\boldsymbol{x}_{\mathrm{R}}} \dot{\theta}_{\boldsymbol{z}_0} \cos\theta_{\boldsymbol{z}_0}. \end{aligned} \tag{2.8}$$

Substituindo (2.8) em (2.7):

$$\begin{bmatrix} \ddot{p}'_{\boldsymbol{x}_0} \\ \ddot{p}'_{\boldsymbol{y}_0} \end{bmatrix} = \begin{bmatrix} a_{\boldsymbol{x}_{\mathrm{R}}} \cos\theta_{\boldsymbol{z}_0} - v_{\boldsymbol{x}_{\mathrm{R}}} \omega_{\boldsymbol{z}_{\mathrm{R}}} \sin\theta_{\boldsymbol{z}_0} - d\alpha_{\boldsymbol{z}_{\mathrm{R}}} \sin\theta_{\boldsymbol{z}_0} - d\omega^2_{\boldsymbol{z}_0} \cos\theta_{\boldsymbol{z}_0} \\ a_{\boldsymbol{x}_{\mathrm{R}}} \sin\theta_{\boldsymbol{z}_0} + v_{\boldsymbol{x}_{\mathrm{R}}} \omega_{\boldsymbol{z}_{\mathrm{R}}} \cos\theta_{\boldsymbol{z}_0} + d\alpha_{\boldsymbol{z}_{\mathrm{R}}} \cos\theta_{\boldsymbol{z}_0} - d\omega^2_{\boldsymbol{z}_{\mathrm{R}}} \sin\theta_{\boldsymbol{z}_0} \end{bmatrix}, \tag{2.9}$$

$$\begin{bmatrix} \ddot{p}'_{\boldsymbol{x}_0} \\ \ddot{p}'_{\boldsymbol{y}_0} \end{bmatrix} = \begin{bmatrix} \cos\theta_{\boldsymbol{z}_0} & -d\sin\theta_{\boldsymbol{z}_0} \\ \sin\theta_{\boldsymbol{z}_0} & d\cos\theta_{\boldsymbol{z}_0} \end{bmatrix} \begin{bmatrix} a_{\boldsymbol{x}_\mathrm{R}} \\ \alpha_{\boldsymbol{z}_\mathrm{R}} \end{bmatrix} + \begin{bmatrix} -v_{\boldsymbol{x}_\mathrm{R}}\omega_{\boldsymbol{z}_\mathrm{R}}\sin\theta_{\boldsymbol{z}_0} - d\omega^2_{\boldsymbol{z}_0}\cos\theta_{\boldsymbol{z}_0} \\ v_{\boldsymbol{x}_\mathrm{R}}\omega_{\boldsymbol{z}_\mathrm{R}}\cos\theta_{\boldsymbol{z}_0} - d\omega^2_{\boldsymbol{z}_0}\sin\theta_{\boldsymbol{z}_0} \end{bmatrix}. \tag{2.10}$$

Fazendo

$$\begin{bmatrix} a_{\boldsymbol{x}_\mathrm{R}} \\ \alpha_{\boldsymbol{z}_\mathrm{R}} \end{bmatrix} = \begin{bmatrix} \cos\theta_{\boldsymbol{z}_0} & -d\sin\theta_{\boldsymbol{z}_0} \\ \sin\theta_{\boldsymbol{z}_0} & d\cos\theta_{\boldsymbol{z}_0} \end{bmatrix}^{-1} \begin{bmatrix} a_{\boldsymbol{x}_0} + v_{\boldsymbol{x}_\mathrm{R}}\omega_{\boldsymbol{z}_\mathrm{R}}\sin\theta_{\boldsymbol{z}_0} + d\omega^2_{\boldsymbol{z}_\mathrm{R}}\cos\theta_{\boldsymbol{z}_0} \\ a_{\boldsymbol{y}_0} - v_{\boldsymbol{x}_\mathrm{R}}\omega_{\boldsymbol{z}_\mathrm{R}}\cos\theta_{\boldsymbol{z}_0} + d\omega^2_{\boldsymbol{z}_\mathrm{R}}\sin\theta_{\boldsymbol{z}_0} \end{bmatrix}, \tag{2.11}$$

sendo $a_{\boldsymbol{x}_0}$ e $a_{\boldsymbol{y}_0}$ as entradas de aceleração das velocidades $\dot{p}_{\boldsymbol{x}_0} = \dot{p}'_{\boldsymbol{x}_0}$ e $\dot{p}_{\boldsymbol{y}_0} = \dot{p}'_{\boldsymbol{y}_0}$ no referencial inercial, a aplicação de um comando de velocidade ou aceleração em cima desse ponto de operação não precisa considerar as restrições cinemáticas, apresentando o comportamento de um robô holonômico, descrito por

$$\begin{aligned} \dot{\boldsymbol{p}}(t) &= \begin{bmatrix} \dot{p}'_{\boldsymbol{x}_0} \\ \dot{p}'_{\boldsymbol{y}_0} \end{bmatrix} = \boldsymbol{v}(t) = \begin{bmatrix} v_{\boldsymbol{x}_0} \\ v_{\boldsymbol{y}_0} \end{bmatrix}, \\ \dot{\boldsymbol{v}}(t) &= \begin{bmatrix} \dot{v}_{\boldsymbol{x}_0} \\ \dot{v}_{\boldsymbol{y}_0} \end{bmatrix} = \boldsymbol{a}(t) = \begin{bmatrix} a_{\boldsymbol{x}_0} \\ a_{\boldsymbol{y}_0} \end{bmatrix}. \end{aligned} \tag{2.12}$$

2.2 Modelo em variáveis de estados

A partir de um conjunto de equações diferenciais (ou equações de diferenças para sistemas de tempo discreto) de primeira ordem, o estado do sistema é representado pelo vetor de variáveis de estado $\boldsymbol{x}$ (geralmente em função do tempo t) que oferecem uma descrição interna do comportamento dinâmico do sistema. À medida que o sistema evolui, a variação do estado é descrita por uma combinação linear do estado $\boldsymbol{x}$ e da entrada $\boldsymbol{u}$ [Ogata, 1995]. Um sistema LIT de tempo contínuo pode ser modelado em variáveis de estado como

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t), \tag{2.13a}$$

$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t), \tag{2.13b}$$

de forma que $\boldsymbol{A}$ e $\boldsymbol{B}$ são matrizes que determinam a dinâmica interna do sistema, enquanto $\boldsymbol{C}$ e $\boldsymbol{D}$ definem como as variáveis são medidas ou observadas na saída. A Figura 2.3 mostra a implementação das Equações (2.13) em diagrama de blocos. No caso de equações de segunda ordem ou maior, é possível tratar alguns termos de ordem superior como variáveis de estados e expandir o conjunto de equações para conter apenas representantes de primeira ordem [Haykin and Veem, 1999].

Em aplicações digitais, um modelo discreto se faz necessário uma vez que a contraparte contínua contém componentes infinitesimais que, na prática, não são implementadas por computadores. No caso discreto, o vetor de estados evolui iterativamente, através de passos fixos relacionados a um tempo de amostragem $T_s \in \mathbb{R}$. A próxima iteração do vetor de estados, $\boldsymbol{x}(k+1)$, é calculada a partir da combinação linear entre o estado atual

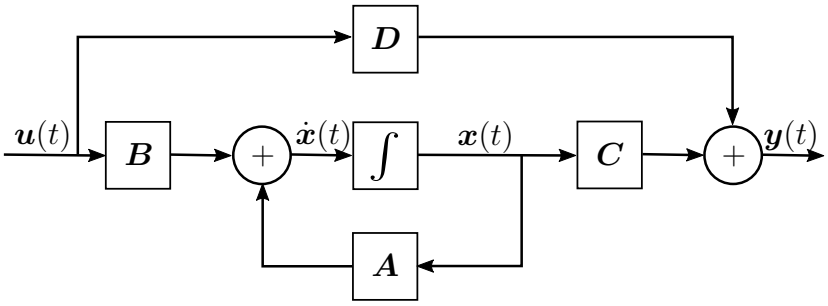

Figura 2.3: Diagrama de blocos de um sistema LIT representado em espaço de estados

$\boldsymbol{x}(k)$ e a entrada atual $\boldsymbol{u}(k)$, fazendo

$$\boldsymbol{x}(k+1) = \boldsymbol{F}\boldsymbol{x}(k) + \boldsymbol{G}\boldsymbol{u}(k). \tag{2.14}$$

Um modelo contínuo pode ser discretizado em relação a um tempo de amostragem T_s usando as relações [Ogata, 1995]

$$\boldsymbol{F} = e^{\boldsymbol{A}T_s}, \tag{2.15a}$$

$$\boldsymbol{G} = \int_0^{T_s} e^{\boldsymbol{A}t}\boldsymbol{B}\mathrm{d}t. \tag{2.15b}$$

Nota-se que a exponencial matricial pode ser calculada a partir da expansão em série de Taylor

$$e^{\boldsymbol{A}t} = \sum_{k=0}^{\infty} \frac{\boldsymbol{A}^k t^k}{k!} = \boldsymbol{I} + \boldsymbol{A}t + \frac{\boldsymbol{A}^2 t^2}{2!} + ... + \frac{\boldsymbol{A}^n t^n}{n!}. \tag{2.16}$$

2.3 Modelo do sistema

Usando a linearização, é possível descrever o movimento do robô com rodas diferenciais da mesma forma que o robô holonômico, desconsiderando as restrições holonômicas. Partindo da Equação 2.12, define-se o modelo contínuo

$$\begin{bmatrix} \dot{\boldsymbol{p}}(t) \\ \dot{\boldsymbol{v}}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{p}(t) \\ \boldsymbol{v}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{a}(t), \tag{2.17}$$

com vetor de estados $\boldsymbol{x}(t) = [\boldsymbol{p}(t)^\top \ \boldsymbol{v}(t)^\top]^\top$ e entrada $\boldsymbol{u}(t) = \boldsymbol{a}(t)$.

Comparando a Equação (2.17) com a Equação (2.13), são encontradas as matrizes $\boldsymbol{A}$ e $\boldsymbol{B}$. Para o modelo discreto, são aplicadas as equações (2.15a) e (2.15b), omitindo os

termos de segunda ordem ou superior da exponencial matricial, tal que

$$\boldsymbol{F} = e^{\boldsymbol{A}T_s} = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}, \tag{2.18a}$$

$$\boldsymbol{G} = \int_0^{T_s} e^{\boldsymbol{A}t}\boldsymbol{B}\mathrm{d}t = \int_0^{T_s} \begin{bmatrix} 1 & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathrm{d}t = \begin{bmatrix} 0.5T_s^2 \\ T_s \end{bmatrix}. \tag{2.18b}$$

Tomando as componentes bidimensionais dos vetores, é obtido o modelo discreto de segunda ordem

$$\begin{bmatrix} p_{\boldsymbol{x}_0}(k+1) \\ p_{\boldsymbol{y}_0}(k+1) \\ v_{\boldsymbol{x}_0}(k+1) \\ v_{\boldsymbol{y}_0}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_s & 0 \\ 0 & 1 & 0 & T_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{\boldsymbol{x}_0}(k) \\ p_{\boldsymbol{y}_0}(k) \\ v_{\boldsymbol{x}_0}(k) \\ v_{\boldsymbol{y}_0}(k) \end{bmatrix} + \begin{bmatrix} 0.5T_s^2 & 0 \\ 0 & 0.5T_s^2 \\ T_s & 0 \\ 0 & T_s \end{bmatrix} \begin{bmatrix} a_{\boldsymbol{x}_0}(k) \\ a_{\boldsymbol{y}_0}(k) \end{bmatrix}. \tag{2.19}$$

Capítulo 3

Controle Preditivo Baseado em Modelo

O Controle Preditivo Baseado em Modelo (MPC, do inglês *Model Predictive Control*), ou simplesmente Controle Preditivo, é uma metodologia de projeto de controladores para automação de sistemas através do uso conjunto de estratégias de predição e otimização com base no modelo explícito da planta [Camacho and Bordons, 2007], levando em conta as restrições que as variáveis do sistema devem atender. Geralmente, o problema de otimização considera um horizonte limitado de predições de estados futuros e ações de controle, sendo empregada apenas a primeira ação na planta real e descartando as demais. Na iteração seguinte o problema de otimização é atualizado e resolvido novamente com as novas medições adquiridas, uma característica conhecida como horizonte retrocedente.

O comportamento de um sistema utilizando Controle Preditivo pode ser exemplificado pela Figura 3.1 em uma aplicação SISO (do inglês *Single Input Single Output*). Partindo de um instante inicial, define-se um valor ou uma trajetória de referência $\boldsymbol{r}(k)$, $\boldsymbol{r}(k+1)$, ..., $\boldsymbol{r}(k+N)$ em um horizonte de predição de tamanho $N \in \mathbb{N}$, assumindo o período de amostragem $T_s \in \mathbb{R}$. A partir do modelo do processo e das leituras passadas, é designado um mecanismo que possibilite estimar saídas $\boldsymbol{y}(k)$, $\boldsymbol{y}(k+1)$, ..., $\boldsymbol{y}(k+N)$ e sinais de entrada $\boldsymbol{u}(k)$, $\boldsymbol{u}(k+1)$, ..., $\boldsymbol{u}(k+N)$ futuros. É construído um problema de otimização com base em um critério, ou função objetivo, para escolher o conjunto de sinais de entrada que mantém o comportamento do sistema próximo da trajetória de referência, podendo considerar ainda fatores como a variação da entrada (*control effort*). Geralmente, o problema de otimização é a tarefa que requer maior tempo no processamento da ação de controle e deve ser resolvido em tempo menor que T_s.

O conceito do MPC está presente na literatura desde meados da década de 1960, tornando-se bastante popular, principalmente, na indústria química, por apresentar sistemas com dinâmica mais lenta. Desde a década de 1980, tem sido utilizado em aplicações com múltiplas variáveis e sujeitas a restrições [Carlos et al., 1989]. Com o aumento no poder computacional, o MPC deixou de se limitar a processos com dinâmica lenta, sendo

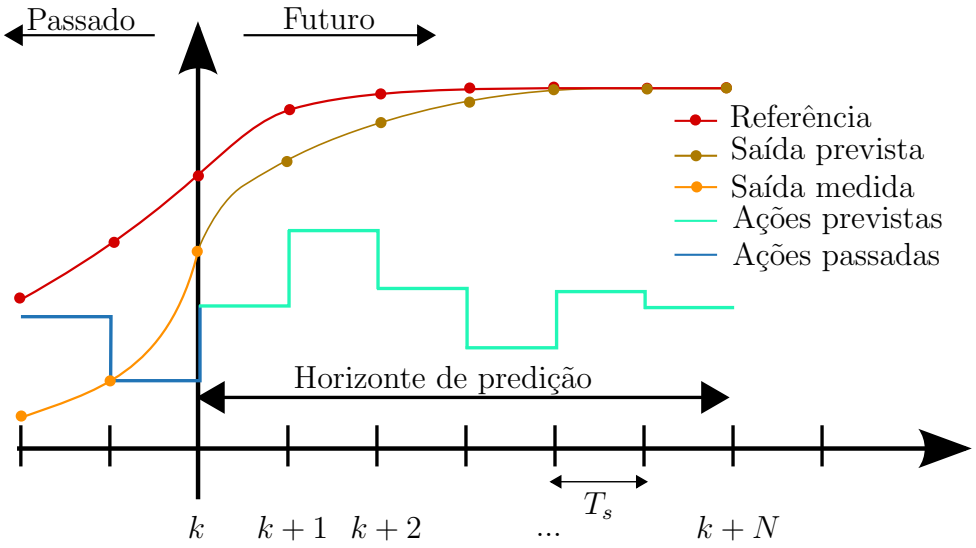

Figura 3.1: Esquema das componentes empregadas em controladores MPC.

empregadas técnicas mais rápidas e eficientes de otimização, o que tornou realizável utilizar o Controle Preditivo em aplicações cada vez mais ágeis.

Por se tratar de uma metodologia, existem diversas implementações de Controle Preditivo com diferentes combinações de modelagem, estratégias de predição e algoritmos de otimização. No decorrer dos anos, algumas implementações evoluíram e foram bastante difundidas. Um exemplo é o algoritmo QDMC (do inglês *Quadratic Dynamic Matrix Control*), que apresenta melhorias ao antecessor DMC (do inglês *Dynamic Matrix Control*) no tratamento de restrições com baixo custo computacional [Qin and Badgwell, 2003]. As características principais do QDMC são:

- Modelo linear da planta;
- Função objetivo quadrática em um horizonte finito de predição;
- O comportamento futuro das variáveis de interesse deve seguir a referência e é sujeito a um termo que pune alterações bruscas na entrada (*move suppression*);
- Entradas ótimas obtidas pela solução de um problema quadrático.

3.1 Determinação do problema MPC

Baseado no modelo apresentado na Equação (2.19) e adotando uma implementação QDMC, o problema MPC para o controle de um robô móvel é definido como

$$\begin{aligned} \text{minimizar} \quad & \sum_{k=0}^{k=N} (\boldsymbol{x}(k) - \boldsymbol{r}(k))^\top \boldsymbol{Q}(k)(\boldsymbol{x}(k) - \boldsymbol{r}(k)) + \sum_{k=0}^{k=N-1} \boldsymbol{u}(k)^\top \boldsymbol{R}(k)\boldsymbol{u}(k), \\ \text{sujeita a} \quad & \boldsymbol{x}(k+1) = \boldsymbol{F}\boldsymbol{x}(k) + \boldsymbol{G}\boldsymbol{u}(k), \\ & \boldsymbol{x}_{\min} \leq \boldsymbol{x}(k) \leq \boldsymbol{x}_{\max}, \\ & \boldsymbol{u}_{\min} \leq \boldsymbol{u}(k) \leq \boldsymbol{u}_{\max}, \\ & \boldsymbol{x}(0) = \boldsymbol{x}_0. \end{aligned} \tag{3.1}$$

Como todas as variáveis de estados são observáveis, considera-se que o vetor de estados seja a própria saída do sistema. A expressão a ser minimizada é composta por somas ponderadas de componentes quadráticas, em que $\boldsymbol{x}(k) - \boldsymbol{r}(k)$ definem o erro da trajetória de um robô móvel.

A função objetivo a ser minimizada é composta por somas ponderadas de componentes quadráticas, tal que as matrizes $\boldsymbol{Q}$ e $\boldsymbol{R}$ são positivas semi-definidas e podem variar no decorrer do horizonte de predição. As matrizes $\boldsymbol{Q}$ e $\boldsymbol{R}$ são parâmetros essenciais de sintonia do controlador por associarem pesos aos critérios de distanciamento da referência e aplicação de ação de controle, respectivamente. As restrições do problema delimitam a região de otimização de forma que respeite a dinâmica do sistema, os limites físicos do robô em relação a posição e velocidade, em $\boldsymbol{x}_{\min}$, $\boldsymbol{x}_{\max}$, e aceleração, em $\boldsymbol{u}_{\min}$ e $\boldsymbol{u}_{\max}$, além da imposição da condição inicial $\boldsymbol{x}_0$ dada pela leitura atual de posição e velocidade.

3.2 Otimização Convexa

Problemas de otimização convexa formam uma classe de problemas de otimização matemática com teoria bem estabelecida que permitem uma resolução numérica muito eficiente e confiável. Além disso, apresentam grande vantagem em aplicações de otimização global e combinatória de variáveis. Por esses motivos, tem sido empregada em aplicações como sistemas de controle, estimação e processamento de sinal, comunicações e redes, projeto eletrônico de circuitos, análise de dados, economia, entre outros [Boyd et al., 2004, Stellato et al., 2018].

De maneira geral, pode-se definir um problema de otimização convexa da seguinte forma:

$$\begin{aligned} \text{minimizar} \quad & J(\boldsymbol{z}), \\ \text{sujeita a} \quad & f_i(\boldsymbol{z}) \leq b_i, \text{ para } i = 1, ..., m, \end{aligned} \tag{3.2}$$

sendo $\boldsymbol{z} \in \mathbb{R}^n$ o vetor de variáveis a ser otimizado de acordo com a função de custo $J(\boldsymbol{z}) : \mathbb{R}^n \to \mathbb{R}$, sujeito às funções de restrição $f_i(\boldsymbol{z}) : \mathbb{R}^n \to \mathbb{R}$ e constantes $b_i \in \mathbb{R}$. Além disso, $J(\boldsymbol{z})$ e $f_i(\boldsymbol{z})$ satisfazem a propriedade de convexidade, ou seja, para $\boldsymbol{z}_1, \boldsymbol{z}_2 \in \mathbb{R}^n$ e $\alpha, \beta \in \mathbb{R}$:

$$\begin{aligned} &J(\alpha\boldsymbol{z}_1 + \beta\boldsymbol{z}_2) \leq \alpha J(\boldsymbol{z}_1) + \beta J(\boldsymbol{z}_2), \\ &f_i(\alpha\boldsymbol{z}_1 + \beta\boldsymbol{z}_2) \leq \alpha f_i(\boldsymbol{z}_1) + \beta f_i(\boldsymbol{z}_2), \text{ para } i = 1, ..., m. \end{aligned} \tag{3.3}$$

Particularmente, pode-se observar que funções lineares são também convexas. Algumas aplicações conhecidas em otimização, como o método dos mínimos quadrados, programação linear, programação semi-definida e programação quadrática são subclasses especiais de otimização convexa. Em resumo, o problema de otimização por mínimos quadrados apresenta como função objetivo a soma de quadrados e nenhuma restrição, programação linear assume função objetivo e restrições convexas, programação semi-definida otimiza uma função de custo linear de acordo com a interseção de cones de matrizes positivas semi-definidas e programação quadrática utiliza uma função objetivo quadrática e restrições convexas.

Cada restrição representa um semi-espaço (*half-space*) $\boldsymbol{a}_i^\top \boldsymbol{z} \leq b_i$, limitado pelo hiperplano definido pelo vetor normal $\boldsymbol{a}_i$ e o ponto $\boldsymbol{z}_{0i}$, de forma que $\boldsymbol{a}_i^\top \boldsymbol{z}_{0i} = b_i$, como mostrado na Figura 3.2.

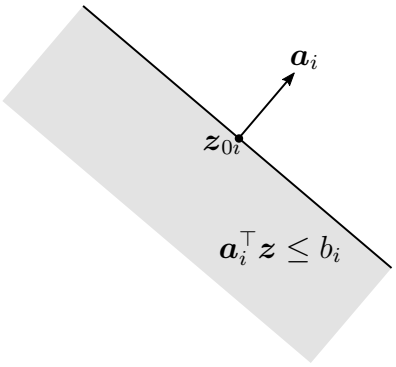

Figura 3.2: Representação de semi-espaço gerado por uma restrição linear.

A região delimitada pelas restrições lineares descrevem o politopo $\boldsymbol{A}\boldsymbol{z} \leq \boldsymbol{b}$, tal que $\boldsymbol{b} = \begin{bmatrix} b_1 & b_2 & ... & b_i & ... & b_m \end{bmatrix}^\top$ e $\boldsymbol{A}$ é uma matriz com m linhas consistindo dos vetores $\boldsymbol{a}_i^\top$ [Boyd et al., 2004]. O politopo tem dimensões baseadas no número de restrições e no número de variáveis de otimização, portanto, sendo $\boldsymbol{z} \in \mathbb{R}^n$ e, admitindo m restrições, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. A Figura 3.3 mostra um exemplo com politopo bidimensional construído a partir de cinco restrições.

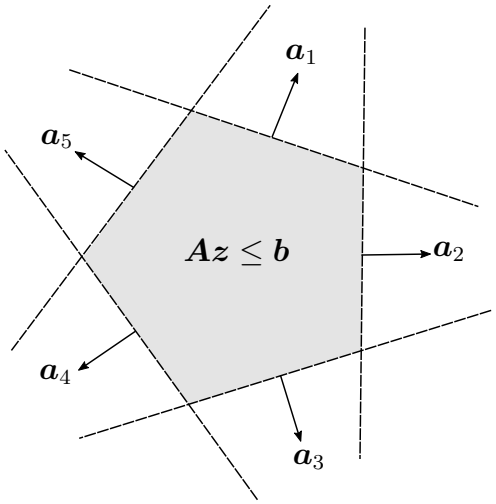

Figura 3.3: Politopo delimitado por restrições lineares

3.2.1 Programação Quadrática

O problema de otimização por programação quadrática (QP, do inglês *Quadratic Programming*) pode ser definido pela expressão

$$\begin{aligned} \text{minimizar} \quad & J(\boldsymbol{z}) = \frac{1}{2}\boldsymbol{z}^\top \boldsymbol{P}\boldsymbol{z} + \boldsymbol{c}^\top \boldsymbol{z}, \\ \text{sujeita a} \quad & \boldsymbol{z}_{\min} \leq \boldsymbol{A}\boldsymbol{z} \leq \boldsymbol{z}_{\max}, \end{aligned} \tag{3.4}$$

de forma que $\boldsymbol{P}$ é uma matriz positiva semi-definida e $\boldsymbol{c} \in \mathbb{R}^n$, assimilando pesos às componentes quadráticas e lineares, respectivamente, da função objetivo. Nas restrições, $\boldsymbol{z}_{\min}$ e $\boldsymbol{z}_{\max}$ determinam os limites máximos e mínimos da operação $\boldsymbol{A}\boldsymbol{z}$. Uma região descrita pelo politopo da Figura 3.3 pode ser delimitada nessa representação assumindo $\boldsymbol{z}_{\max} = \boldsymbol{b}$ e cada elemento de $\boldsymbol{z}_{\min}$ como $-\infty$. Além disso, é possível definir restrições de igualdade considerando $\boldsymbol{z}_{\min} = \boldsymbol{z}_{\max}$.

Uma das vantagens dessa abordagem é a garantia de um mínimo único e global, uma vez que o problema seja realizável. Em contraste, na programação linear a solução não tem unicidade assegurada, mas se encontra nos vértices (interseção entre hiperplanos) do politopo, enquanto na programação quadrática sabe-se que a solução está no interior do politopo.

Para solucionar um problema de programação quadrática, foram desenvolvidos algoritmos como os métodos do Conjunto Ativo (*Active Set*), do Ponto Interior (*Interior Point*) e de Primeira Ordem (*First Order*). Especificamente, os métodos de Primeira Ordem computam iterativamente uma solução ótima usando apenas informação de primeira ordem da função objetivo. Uma classe particular de técnicas de Primeira Ordem são as de Separação de Operadores (*Operator Splitting*), que modelam o problema de otimização como um problema de encontrar uma raiz de uma soma de operadores monotônicos.

Neste trabalho, foi utilizado o OSQP[1] (do inglês *Operator Splitting Quadratic Program*), um solucionador de propósito geral para problemas de otimização envolvendo programação quadrática, capaz de computar soluções com alta precisão e baseado no algoritmo ADMM (do inglês *Alternating Direction Method of Multipliers*), que aplica um método de separação de operadores [Stellato et al., 2018, Boyd et al., 2010].

3.3 Transformação do problema MPC em QP

Para que seja solucionado no OSQP ou em algum outro solucionador de programação quadrática, é necessário passar o problema MPC da Equação (3.1) para um problema QP como apresentado na Equação 3.4. O vetor $\boldsymbol{z}$ de variáveis de interesse consiste de todos os estados e entradas do horizonte de predição N, podendo ser escrito como

$$\boldsymbol{z} = [\boldsymbol{x}(0)^\top \ \boldsymbol{x}(1)^\top \ ... \ \boldsymbol{x}(N)^\top \ \boldsymbol{u}(0)^\top \ \boldsymbol{u}(1)^\top \ ... \ \boldsymbol{u}(N-1)^\top]^\top. \tag{3.5}$$

As componentes matriciais quadrática $\boldsymbol{P}$ e linear $\boldsymbol{c}$ da função objetivo são definidas considerando o horizonte de predição. Para isso, o produto de Kronecker é utilizado, onde são aplicadas as matrizes de pesos, $\boldsymbol{Q}$ e $\boldsymbol{R}$, e a trajetória de referência $\boldsymbol{r}$ apresentadas na formulação do problema MPC, obtendo

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{Q}(0) & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{Q}(1) & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{Q}(N) & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{R}(0) & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boldsymbol{R}(N-1) \end{bmatrix}, \tag{3.6a}$$

$$\boldsymbol{c} = \begin{bmatrix} -\boldsymbol{Q}(0)\boldsymbol{r}(0) \\ -\boldsymbol{Q}(1)\boldsymbol{r}(1) \\ \vdots \\ -\boldsymbol{Q}(N)\boldsymbol{r}(N) \\ \mathbf{0}_{N \cdot n_u \times 1} \end{bmatrix}. \tag{3.6b}$$

Para a aplicação desenvolvida neste trabalho, considera-se $n_x = 4$ a dimensão do vetor de estados $\boldsymbol{x}$ e $n_u = 2$ a dimensão do vetor de entradas $\boldsymbol{u}$.

O problema MPC apresenta restrições determinadas por funções convexas na forma de equações e inequações. Assim, as componentes $\boldsymbol{z}_{\min}$, $\boldsymbol{z}_{\max}$ e $\boldsymbol{A}$ do problema QP po-

[1]Disponível em `https://osqp.org/`

dem ser computadas a partir de submatrizes das parcelas de igualdades (subscrito eq) e desigualdades (subscrito ineq), como

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{\text{eq}} \\ \boldsymbol{A}_{\text{ineq}} \end{bmatrix}, \quad \boldsymbol{z}_{\min} = \begin{bmatrix} \boldsymbol{z}_{\min_{\text{eq}}} \\ \boldsymbol{z}_{\min_{\text{ineq}}} \end{bmatrix}, \quad \boldsymbol{z}_{\max} = \begin{bmatrix} \boldsymbol{z}_{\max_{\text{eq}}} \\ \boldsymbol{z}_{\max_{\text{ineq}}} \end{bmatrix}, \tag{3.7}$$

de forma que $\boldsymbol{A}_{\text{eq}}$, $\boldsymbol{z}_{\min_{\text{eq}}}$ e $\boldsymbol{z}_{\max_{\text{eq}}}$ são definidas a partir das matrizes que representam a dinâmica do sistema, $\boldsymbol{F}$ e $\boldsymbol{G}$, aplicadas no decorrer do horizonte de predição N. Como esses fatores representam condições de igualdade do problema de otimização, é necessário que $\boldsymbol{z}_{\min_{\text{eq}}} = \boldsymbol{z}_{\max_{\text{eq}}}$, gerando um conjunto de hiperplanos $\boldsymbol{A}_{\text{eq}}\boldsymbol{z} = \boldsymbol{z}_{\min_{\text{eq}}} = \boldsymbol{z}_{\max_{\text{eq}}}$. Além disso, $\boldsymbol{z}_{\min_{\text{eq}}}$ e $\boldsymbol{z}_{\max_{\text{eq}}}$ contêm a condição inicial $\boldsymbol{x}(0)$ em que o sistema se encontra. Assim, pode-se escrever

$$\begin{gathered} \boldsymbol{A}_{\text{eq}} = \left[(\boldsymbol{I}_{N+1} \otimes -\boldsymbol{I}_{n_x}) + (\boldsymbol{I}_{N+1}^{k=-1} \otimes \boldsymbol{F}) \quad [\boldsymbol{0}_{N\times 1}^{\top} \quad \boldsymbol{I}_N^{\top}]^{\top} \otimes \boldsymbol{G} \right], \\ \boldsymbol{z}_{\min_{\text{eq}}} = \begin{bmatrix} -\boldsymbol{x}(0) \\ \boldsymbol{0}_{N\cdot n_x} \end{bmatrix}, \qquad \boldsymbol{z}_{\max_{\text{eq}}} = \begin{bmatrix} -\boldsymbol{x}(0) \\ \boldsymbol{0}_{N\cdot n_x} \end{bmatrix}, \end{gathered} \tag{3.8}$$

em que o parâmetro k da matriz $\boldsymbol{I}_{N+1}^{k=-1}$ representa um deslocamento da diagonal, de forma que $k > 0$ representa um deslocamento para a direita e $k < 0$, para a esquerda.

As parcelas matriciais de inequações $\boldsymbol{A}_{\text{ineq}}$, $\boldsymbol{z}_{\min_{\text{ineq}}}$ e $\boldsymbol{z}_{\max_{\text{ineq}}}$ definem um conjunto de semi-espaços, resultando em uma região convexa que aplica as restrições de limites máximos e mínimos para os estados ($\boldsymbol{x}_{\min}$ / $\boldsymbol{x}_{\max}$) e entradas ($\boldsymbol{u}_{\min}$ / $\boldsymbol{u}_{\max}$). Por conta disso, em geral, $\boldsymbol{z}_{\min_{\text{ineq}}} \neq \boldsymbol{z}_{\max_{\text{ineq}}}$. Por fim, pode-se escrever

$$\boldsymbol{A}_{\text{ineq}} = \boldsymbol{I}_{(N+1)n_x+Nn_u}, \; \boldsymbol{z}_{\min_{\text{ineq}}} = \begin{bmatrix} \boldsymbol{1}_{N+1\times 1} \otimes \boldsymbol{x}_{\min} \\ \boldsymbol{1}_{N\times 1} \otimes \boldsymbol{u}_{\min} \end{bmatrix} \text{ e } \boldsymbol{z}_{\max_{\text{ineq}}} = \begin{bmatrix} \boldsymbol{1}_{N+1\times 1} \otimes \boldsymbol{x}_{\max} \\ \boldsymbol{1}_{N\times 1} \otimes \boldsymbol{u}_{\max} \end{bmatrix}. \tag{3.9}$$

Capítulo 4

Navegação de robôs

A navegação de robôs pode ser dividida em três grandes tarefas, são elas localização, planejamento de trajetória e controle de movimentação. Em um sistema autônomo, essas tarefas são interligada, de forma que uma etapa usa recursos computados pela outra, como a posição do agente, a representação do mapa e a rota a ser tomada.

A localização é responsável por mapear e modelar o ambiente em que o robô se encontra e determinar o estado do agente, seja através de um modelo predefinido do mundo, sensores eletrônicos ou processamento de imagem. Vários trabalhos nesse campo se preocupam em tratar imprecisões e incertezas na leitura dos sensores. Dentre os problemas de localização e mapeamento pode-se destacar o SLAM (do inglês *Simultaneous Localization And Mapping*) [Choset et al., 2005].

Conhecendo o ambiente ou mapa em que o robô se encontra, o planejamento de trajetória tem a função de estabelecer uma rota para que o robô se desloque de uma posição inicial a uma posição objetivo. Pode ser utilizada uma função matemática representando a trajetória desejada do robô ou mesmo a utilização de algoritmos que levam em conta aspectos como tempo e distância mínima para o objetivo. Algoritmos nessa categoria também são chamados de planejadores globais. Exemplos de técnicas que atacam esse problema são PRM (do inglês *Probabilistic Roadmaps*), variações do algoritmo de busca **A*** e Campos Potenciais (*Potential Fields*) [Sariff and Buniyamin, 2006, Mouad et al., 2012].

O sistema de controle do movimento tenta garantir que o robô possa rastrear uma referência de posição, velocidade ou aceleração, projetados pelo planejador global. Nesse contexto, o sistema deve ser capaz de responder a alguns tipos de pertubações, como obstáculos, inclinações de terreno e demais agentes móveis. Diferentemente da maioria dos algoritmos de planejamento de trajetória global, algoritmos de controle do movimento dever reagir local e dinamicamente no ambiente em tempo real [Van den Berg et al., 2008].

4.1 Planejamento da trajetória

A maioria dos planejadores consideram estáticos os obstáculos observados, de forma que obstáculos móveis ou agentes se movimentando no ambiente exigem o replanejamento contínuo da trajetória para evitar colisões. Neste trabalho, a etapa de planejamento de trajetória não considera obstáculos móveis ou agentes, sendo esses tratados localmente pelo sistema de controle de movimentação.

Neste estudo, a trajetória é definida por uma função $\boldsymbol{r}(t) = [\boldsymbol{p}_{\text{ref}}(t)^\top \ \boldsymbol{v}_{\text{ref}}(t)^\top]^\top$ qualquer, sendo $\boldsymbol{p}_{\text{ref}}(t)$ o vetor de referência de posições e $\boldsymbol{v}_{\text{ref}}(t)$ o vetor de referência de velocidades. Para assegurar movimentos suaves e realistas, determina-se $\boldsymbol{p}_{\text{ref}}(t)$ e $\boldsymbol{v}_{\text{ref}}(t)$ diferenciáveis, além da relação $\boldsymbol{v}_{\text{ref}}(t) = \dot{\boldsymbol{p}}_{\text{ref}}(t)$. A função logística

$$\sigma(t) = \frac{1}{1 + e^{-K(t - t_{\max})}} \tag{4.1}$$

é uma função sigmoide que apresenta essas características, sendo aplicada em diversos campos por apresentar a derivada na forma

$$\dot{\sigma}(t) = K\sigma(t)(1 - \sigma(t)), \tag{4.2}$$

em que $t_{\max}$ é o tempo em que a derivada da função atinge valor máximo e $K \in \mathbb{R}$ é o coeficiente de declividade. Utilizando a função logística em $\boldsymbol{p}_{\text{ref}}(t)$ obtém-se um movimento com aceleração inicialmente positiva e finalmente negativa ao chegar próximo do objetivo. Considerando que o robô parte de uma posição inicial $\boldsymbol{p}_{\text{start}}$ para uma posição objetivo $\boldsymbol{p}_{\text{goal}}$, as componentes de $\boldsymbol{r}$ são definidas através de uma combinação convexa e sua derivada, na forma

$$\boldsymbol{p}_{\text{ref}}(t) = \sigma(t)\boldsymbol{p}_{\text{goal}} + (1 - \sigma(t))\boldsymbol{p}_{\text{start}}, \tag{4.3a}$$

$$\boldsymbol{v}_{\text{ref}}(t) = \dot{\sigma}(t)\boldsymbol{p}_{\text{goal}} - \dot{\sigma}(t)\boldsymbol{p}_{\text{start}}. \tag{4.3b}$$

A Figura 4.1 mostra três exemplos de trajetórias usando função logística com coeficientes de declividade diferentes, mas mesmo $t_{\max}$, ou seja, todas as trajetórias apresentam pico de velocidade ao mesmo tempo. Pode-se observar que K é inversamente proporcional ao tempo necessário para se deslocar de $\boldsymbol{p}_{\text{start}}$ a $\boldsymbol{p}_{\text{goal}}$. Portanto, de acordo com o valor de K, pode-se adiantar $t_{\max}$ sem prejudicar a continuidade de $\boldsymbol{p}_{\text{ref}}(t)$, mas K deve ser escolhido de forma que restrições de velocidade e aceleração máximas do robô ainda sejam respeitadas pela trajetória gerada.

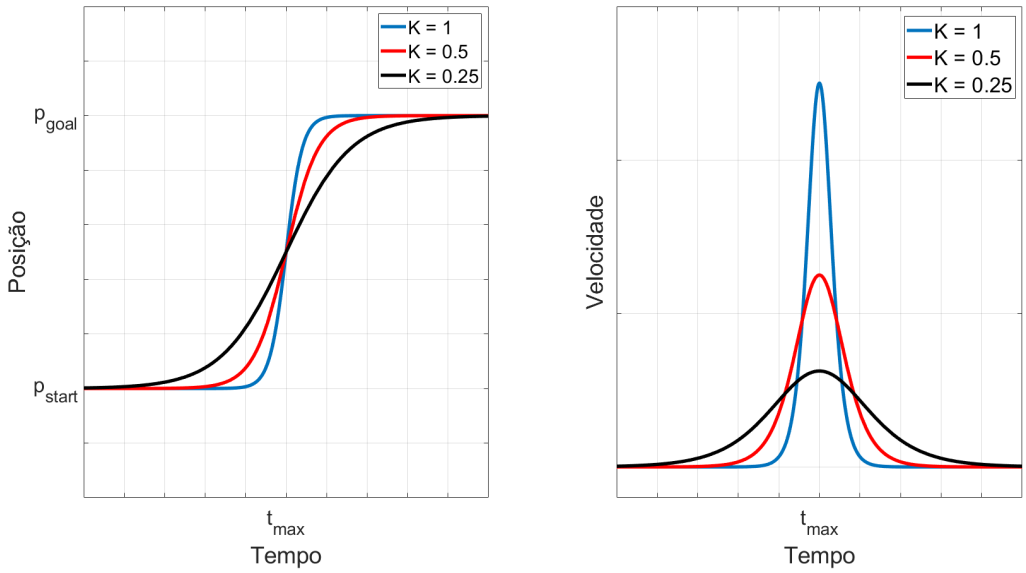

Figura 4.1: Trajetórias utilizando função logística apresentando velocidade máxima em $t_{\max}$, mas coeficientes de declividade diferentes.

4.2 Controle local para prevenção de colisão entre robôs

A seguir são apresentados métodos de primeira ordem, presentes na literatura, dedicados a prevenir colisões entre agentes móveis descentralizados, conhecendo algumas características do ambiente, como as posições e velocidades atuais dos outros agentes. Não é necessário comunicação explícita entre os robôs, podendo as posições e velocidades serem estimadas por sensores em cada agente.

A Figura 4.2 mostra o cenário em que os métodos serão aplicados, onde são considerados os robôs circulares A e B, descritos através dos respectivos centros, $\boldsymbol{p}_A$ e $\boldsymbol{p}_B$, e raios r_A e r_B, além das velocidades absolutas $\boldsymbol{v}_A$ e $\boldsymbol{v}_B$. As técnicas são apresentadas pelo ponto de vista do robô A, de forma que $\boldsymbol{v}_B$ é considerado um vetor fixo e $\boldsymbol{v}_A$ é o vetor da velocidade de interesse, podendo considerar ainda a velocidade de referência determinada pelo planejador global $\boldsymbol{v}_{\text{ref}A}$.

4.2.1 Velocity Obstacles

O conceito de *Velocity Obstacles* (VO) é baseado no mapeamento do comportamento variável do ambiente no espaço de velocidades do robô [Fiorini and Shiller, 2002, Minguez et al., 2016]. Define-se o cone de colisão, que determina as velocidades $\boldsymbol{v}_{A,B}$, sendo as velocidades de A relativas a B, que resultam na colisão entre os agentes em

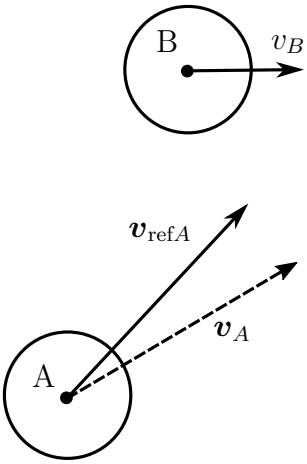

Figura 4.2: Cenário base com dois agentes móveis.

algum momento futuro se as velocidades se mantiverem constantes, como

$$\boldsymbol{CC}(\boldsymbol{v}_B) = \{\boldsymbol{v}_{A,B} \mid \boldsymbol{\lambda}(\boldsymbol{p}_A, \boldsymbol{v}_A - \boldsymbol{v}_B) \cap \boldsymbol{B} \neq \emptyset\}, \tag{4.4}$$

sendo $\boldsymbol{\lambda}(\boldsymbol{p}_A, \boldsymbol{v}_A - \boldsymbol{v}_B)$ a linha na direção de $\boldsymbol{v}_A - \boldsymbol{v}_B$ passando pelo ponto $\boldsymbol{p}_A$, e sendo limitado pelas linhas $\boldsymbol{\lambda}_1$ e $\boldsymbol{\lambda}_2$ tangentes a $\boldsymbol{B}$ que representa todos os pontos do agente B, como mostrado na Figura 4.3.

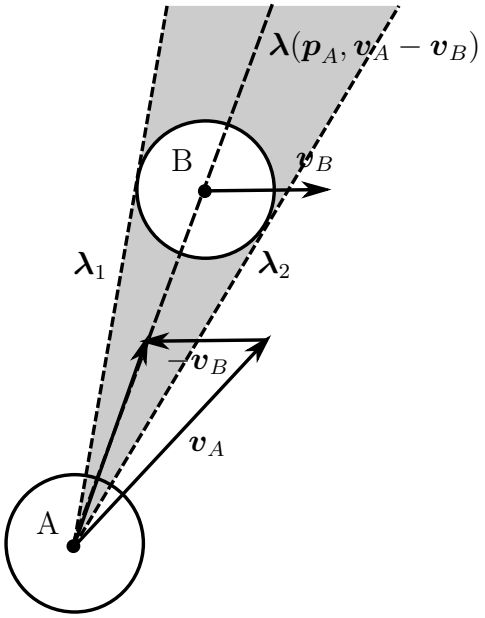

Figura 4.3: Determinação do cone de colisão $\boldsymbol{CC}(\boldsymbol{v}_B)$.

Considerando um sistema com múltiplos colisores, deve-se considerar a região em termos de velocidades absolutas, ou em relação a um referencial inercial fixo. Utilizando a adição de Minkowski, é definida a região $\boldsymbol{VO}_{A|B}(\boldsymbol{v}_B)$ como

$$\boldsymbol{VO}_{A|B}(\boldsymbol{v}_B) = \boldsymbol{CC}(\boldsymbol{v}_B) \oplus \boldsymbol{v}_B. \tag{4.5}$$

A Figura 4.4 mostra a região resultante da desta soma.

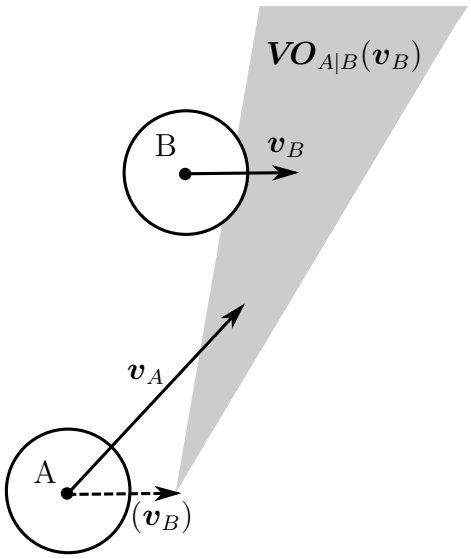

Figura 4.4: Região *Velocity Obstacles* $\boldsymbol{VO}_{A|B}(\boldsymbol{v}_B)$.

Definindo o conjunto de obstáculos, é possível construir regiões de *Velocity Obstacles* de A em relação a cada obstáculo. A união de todas essas regiões como

$$\boldsymbol{VO}_A = \bigcup_{B \neq A} \boldsymbol{VO}_{A|B}(\boldsymbol{v}_B) \tag{4.6}$$

define a região combinada de velocidades de A que resultam em colisão com algum agente em um momento futuro.

Portanto, sendo $\boldsymbol{v}_{\text{ref}A}$ a velocidade desejada do agente A computada pelo planejador global, deve ser encontrado $\boldsymbol{v}_A^* \notin \boldsymbol{VO}_A$, de forma que $\boldsymbol{v}_A^*$ esteja próximo de $\boldsymbol{v}_{\text{ref}A}$.

Seguindo uma descrição alternativa, pode ser obtida uma região VO levando em consideração uma janela de tempo τ, como mostrado na Figura 4.5. Assim, a região é limitada de forma que velocidades que resultam em colisão em um momento posterior a τ são desprezadas [Van Den Berg et al., 2011]. Sendo $\boldsymbol{B}(\boldsymbol{p}, r)$ uma bola aberta com raio r e centrada em $\boldsymbol{p}$, define-se o cone de colisão e a região VO no espaço de velocidades:

$$\boldsymbol{CC}^{\tau}_{A|B} = \{\boldsymbol{v} \mid \exists t \in [0, \tau] :: \boldsymbol{v}t \in \boldsymbol{B}(\boldsymbol{p}_B - \boldsymbol{p}_A, r_A + r_B)\}, \tag{4.7a}$$

$$\boldsymbol{VO}^{\tau}_{A|B}(\boldsymbol{v}_B) = \boldsymbol{CC}^{\tau}_{A|B} \oplus \boldsymbol{v}_B. \tag{4.7b}$$

Apesar de $\boldsymbol{v}_B$ ser apenas um vetor nesse cenário, essa definição permite aplicar a soma de Minkowski a um conjunto de velocidades admissíveis $\boldsymbol{V}_B$, podendo resultar não apenas no deslocamento do cone de colisão, mas também na dilatação da região para oferecer uma margem de segurança.

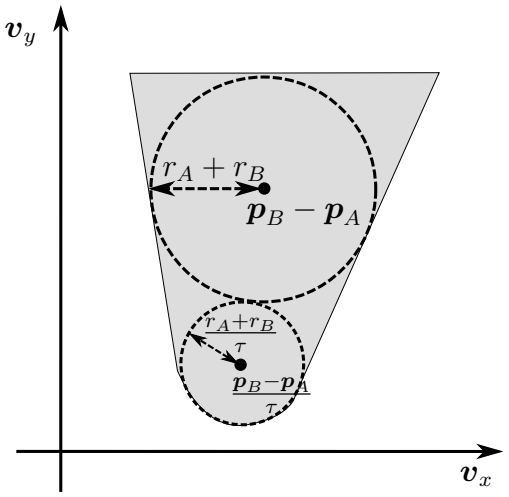

Figura 4.5: Cone de colisão considerando uma janela de tempo.

4.2.2 Reciprocal Velocity Obstacles

Utilizando VO, quando cada agente não leva em consideração que existem outros agentes no ambiente que também executam a rotina de prevenção de colisão, o movimento resultante é suscetível a oscilações indesejadas. Nesse cenário, a abordagem de *Reciprocal Velocity Obstacles* (RVO) apresenta uma extensão do VO levando a movimentos livres de colisões e oscilações [Van den Berg et al., 2008].

Utilizando RVO, ao invés de escolher uma velocidade fora da região VO, é escolhida a média da velocidade atual $\boldsymbol{v}_A$ e a velocidade que está imediatamente fora da região, dividindo a responsabilidade de prevenir a colisão entre os dois agentes. A região RVO entre dois agentes A e B com janela de tempo τ é definida como

$$\boldsymbol{RVO}^{\tau}_{A|B}(\boldsymbol{v}_A, \boldsymbol{v}_B) = \{\boldsymbol{v}^*_A \mid 2\boldsymbol{v}^*_A - \boldsymbol{v}_A \in \boldsymbol{VO}^{\tau}_{A|B}(\boldsymbol{v}_B)\}. \tag{4.8}$$

Geometricamente, a região pode ser interpretada como um deslocamento de $\boldsymbol{VO}^{\tau}_{A|B}(\boldsymbol{v}_B)$ pelo fator $(\boldsymbol{v}_A - \boldsymbol{v}_B)/2$.

A estratégia RVO apresenta vantagens no cenário em que são avaliados dois agentes com potencial de reação ao ambiente. No caso de obstáculos passivos, i.e. que não têm percepção e reação ao ambiente, é mantida a estratégia do *Velocity Obstacles*. Assumindo o conjunto de obstáculos passivos $\mathbb{O}$, a região combinada de velocidades que colocam o agente A em rota de colisão com demais objetos é definida como

$$\boldsymbol{RVO}^{\tau}_{A} = \bigcup_{B \neq A} \boldsymbol{RVO}^{\tau}_{A|B}(\boldsymbol{v}_A, \boldsymbol{v}_B) \cup \bigcup_{O \in \mathbb{O}} \boldsymbol{VO}^{\tau}_{A|O}(\boldsymbol{v}_O). \tag{4.9}$$

4.2.3 Optimal Reciprocal Collision Avoidance

A proposta do *Optimal Reciprocal Collision Avoidance* (ORCA) consiste em uma reformulação das técnicas utilizadas em VO e RVO com o intuito de gerar condições suficientes para navegação livre de colisão de múltiplos robôs a partir de regiões definidas por semi-espaços no plano [Van Den Berg et al., 2011]. Essa abordagem facilita a resolução do problema utilizando algoritmos de otimização convexa, principalmente em cenários com maior densidade de robôs.

Primeiramente, considerando que a velocidade atual $\boldsymbol{v}_A$ está contida na região $\boldsymbol{VO}^{\tau}_{A|B}(\boldsymbol{v}_B)$, é computado o vetor $\boldsymbol{u}$ que representa a menor alteração que leva a velocidade do robô A para a borda da região.

$$\boldsymbol{u} = \left(\underset{\boldsymbol{v}^*_A \in \partial \boldsymbol{VO}^{\tau}_{A|B}(\boldsymbol{v}_B)}{\operatorname{argmin}} ||\boldsymbol{v}^*_A - \boldsymbol{v}_A|| \right) - \boldsymbol{v}_A, \tag{4.10}$$

sendo $\partial \boldsymbol{VO}^{\tau}_{A|B}(\boldsymbol{v}_B)$ um ponto no limite da região $\boldsymbol{VO}^{\tau}_{A|B}(\boldsymbol{v}_B)$.

Considera-se ainda o vetor $\boldsymbol{n}$ como o vetor normal ao contorno de $\boldsymbol{VO}^{\tau}_{A|B}(v_B)$ no ponto $\boldsymbol{v}_A + \boldsymbol{u}$ apontando para o interior da região VO. A Figura 4.6 descreve os vetores no cenário exemplo.

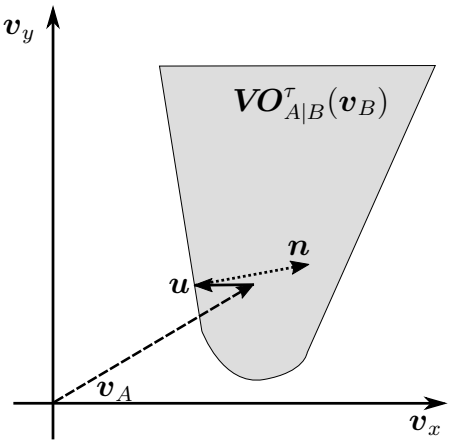

Figura 4.6: Determinação dos vetores $\boldsymbol{u}$ e $\boldsymbol{n}$ de acordo com a velocidade $\boldsymbol{v}_A$ e a região de *Velocity Obstacles* $\boldsymbol{VO}^{\tau}_{A|B}(v_B)$.

Define-se a região $\boldsymbol{ORCA}^{\tau}_{A|B}$ como o conjunto de velocidades de A que contém mais velocidades mais próximas de $\boldsymbol{v}_A$ que previnem a colisão entre os robôs do que qualquer outro conjunto, partindo do princípio da divisão do esforço entre os dois agentes, similarmente à abordagem RVO. Portanto, a região $\boldsymbol{ORCA}^{\tau}_{A|B}$ é o semi-plano determinado pelo ponto $\boldsymbol{v}_0 = \boldsymbol{v}_A + \frac{1}{2}\boldsymbol{u}$ e o vetor normal $\boldsymbol{n}$, sendo definida como

$$\boldsymbol{ORCA}^{\tau}_{A|B} = \{\boldsymbol{v}^*_A \mid (\boldsymbol{v}^*_A - (\boldsymbol{v}_A + \frac{1}{2}\boldsymbol{u})) \cdot \boldsymbol{n} \leq 0\} \tag{4.11}$$

e representado na Figura 4.7.

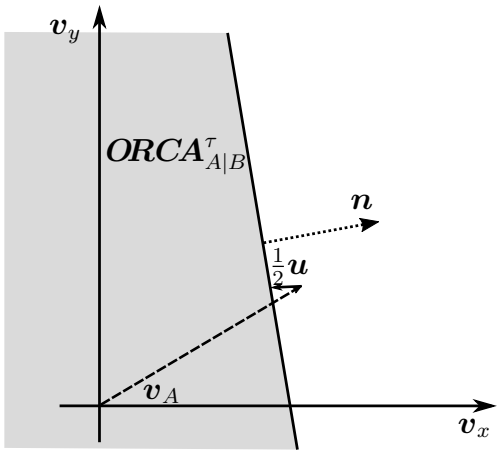

Figura 4.7: Região $\boldsymbol{ORCA}^{\tau}_{A|B}$ de velocidades permitidas para o robô A.

4.3 MPC-ORCA

Os semi-planos resultantes da aplicação do ORCA em cada robô de um cenário com múltiplos agentes são regiões convexas, de forma que a interseção dessas regiões determina um politopo como o apresentado na Figura 3.3. Assim, é possível construir um problema de otimização convexa para determinar um comando de velocidade livre de colisão próximo da velocidade de referência $\boldsymbol{v}_{\mathrm{ref}A}$ em um determinado momento.

A utilização de uma estratégia combinada de MPC e ORCA fornece diversas vantagens sobre a aplicação de um algoritmo de otimização construído puramente a partir do ORCA [Cheng et al., 2017]. A característica de horizonte retrocedente presente no MPC oferece trajetórias mais seguras, reagindo antecipadamente a prováveis cenários de colisão e reduzindo o número de situações em que o problema de otimização se torna infactível (*infeasible*). Além disso, ao incorporar a dinâmica do modelo de segunda ordem proposto na Equação (2.19), as trajetórias computadas são mais suaves, por não assumirem grandes mudanças instantâneas de velocidade dos agentes.

A descrição do problema MPC presente na expressão (3.1) pode ser expandida incorporando as restrições ORCA. Para isso, define-se $\boldsymbol{g}(\boldsymbol{x}(k), \{\boldsymbol{x}_{\mathrm{neigh}}(k)\}) \leq 0$ como as restrições de velocidade resultantes da aplicação do ORCA relacionando o robô atual com os demais robôs de sua vizinhança. Além disso, restrições ORCA para estados futuros são geradas aplicando o algoritmo no decorrer do horizonte de predição N, considerando que os agentes manterão velocidades constantes. Portanto, pode-se escrever o problema

MPC-ORCA como

$$\begin{aligned}
\text{minimizar} \quad & \sum_{k=0}^{N}(\boldsymbol{x}(k)-\boldsymbol{r}(k))^{\top}\boldsymbol{Q}(k)(\boldsymbol{x}(k)-\boldsymbol{r}(k))+\sum_{k=0}^{N-1}\boldsymbol{u}(k)^{\top}\boldsymbol{R}(k)\boldsymbol{u}(k), \\
\text{sujeita a} \quad & \boldsymbol{x}(k+1)=\boldsymbol{F}\boldsymbol{x}(k)+\boldsymbol{G}\boldsymbol{u}(k), \\
& \boldsymbol{x}_{\min}\leq\boldsymbol{x}(k)\leq\boldsymbol{x}_{\max}, \\
& \boldsymbol{u}_{\min}\leq\boldsymbol{u}(k)\leq\boldsymbol{u}_{\max}, \\
& \boldsymbol{g}(\boldsymbol{x}(k),\{\boldsymbol{x}_{\text{neigh}}(k)\})\leq 0, \\
& \boldsymbol{x}(0)=\boldsymbol{x}_0.
\end{aligned} \tag{4.12}$$

Outra particularidade da estratégia combinada MPC-ORCA é a possibilidade de implementar um controlador que opere em modo servo ou regulador. No modo servo é recebida uma trajetória de referência $\boldsymbol{r}(k)$, $\boldsymbol{r}(k+1)$, ..., $\boldsymbol{r}(k+N)$ do planejador global e o controlador coordena comandos de velocidade que mantém o robô próximo da trajetória desejada ou realizando pequenos desvios que previnam a colisão com demais agentes. No modo regulatório, descarta-se o planejador global considerando $\boldsymbol{r}(k)=\boldsymbol{r}(k+1)=...=\boldsymbol{r}(k+N)=\boldsymbol{x}_{\text{goal}}$, onde o objetivo é geralmente descrito por $\boldsymbol{x}_{\text{goal}}=[\boldsymbol{p}_{\text{goal}}\ \boldsymbol{0}]^{\top}$. Dessa forma, o controlador deve levar o robô ao objetivo fixado evitando colisão com os demais. Neste trabalho, o MPC-ORCA é utilizado no modo servo por possibilitar que o robô realize trajetórias genéricas e complexas, como movimentos circulares, senoidais, com velocidade constante ou variável.

4.3.1 Aplicação do ORCA no problema QP

Seguindo a metodologia empregada na Seção 3.3, é necessário transformar o problema MPC da Equação (4.12) na formulação QP da Equação (3.4) para que seja utilizado um solucionador de programação quadrática como o OSQP. Para isso, são adicionadas linhas a $\boldsymbol{A}$, $\boldsymbol{z}_{\min}$ e $\boldsymbol{z}_{\max}$ refazendo a expressão (3.7) da seguinte forma:

$$\boldsymbol{A}=\begin{bmatrix}\boldsymbol{A}_{\text{eq}}\\ \boldsymbol{A}_{\text{ineq}}\\ \boldsymbol{A}_{\text{ORCA}}\end{bmatrix},\quad \boldsymbol{z}_{\min}=\begin{bmatrix}\boldsymbol{z}_{\min,\text{eq}}\\ \boldsymbol{z}_{\min,\text{ineq}}\\ \boldsymbol{z}_{\min,\text{ORCA}}\end{bmatrix},\quad \boldsymbol{z}_{\max}=\begin{bmatrix}\boldsymbol{z}_{\max,\text{eq}}\\ \boldsymbol{z}_{\max,\text{ineq}}\\ \boldsymbol{z}_{\max,\text{ORCA}}\end{bmatrix}. \tag{4.13}$$

Considerando uma quantidade m de agentes na vizinhança do robô, $\boldsymbol{A}_{\text{ORCA}}$, $\boldsymbol{z}_{\min,\text{ORCA}}$ e $\boldsymbol{z}_{\max,\text{ORCA}}$ contêm $N\times m$ linhas descrevendo as restrições ORCA. Cada linha é construída através dos valores de $\boldsymbol{v}_0$ e $\boldsymbol{n}$ resultantes da aplicação do algoritmo, descrevendo um semi-plano de valores de velocidade admissíveis para o estado $\boldsymbol{x}(k)$, sendo $k=0,1,...,N$.

Um semi-plano que aplica restrições para apenas um estado $\boldsymbol{x}(k)$ pode ser definido

como

$$\begin{bmatrix} \mathbf{0} \\ \boldsymbol{n} \end{bmatrix}^{\top} \boldsymbol{x}(k) \leq \boldsymbol{v}_0 \cdot \boldsymbol{n}. \tag{4.14}$$

Todavia, para que essas restrições sejam aplicadas à variável de otimização $\boldsymbol{z}$ (Equação (3.5)), determina-se a matriz $\boldsymbol{A}_{\text{ORCA}}$ em uma estrutura esparsa. Os valores de $\boldsymbol{A}_{\text{ORCA}}$, $\boldsymbol{z}_{\text{min,ORCA}}$ e $\boldsymbol{z}_{\text{max,ORCA}}$ são dados por

$$\boldsymbol{A}_{\text{ORCA}} = \begin{bmatrix} 0 & n_1 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ 0 & n_2 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & n_m & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & n_{m+1} & \cdots & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & n_{Nm} & \cdots & 0 & 0 \end{bmatrix},$$

$$\boldsymbol{z}_{\text{min,ORCA}} = \begin{bmatrix} -\infty \\ -\infty \\ \vdots \\ -\infty \end{bmatrix}, \qquad \boldsymbol{z}_{\text{max,ORCA}} = \begin{bmatrix} v_{0_1} \cdot n_1 \\ v_{0_2} \cdot n_2 \\ \vdots \\ v_{0_{Nm}} \cdot n_{Nm} \end{bmatrix}. \tag{4.15}$$

A cada iteração, o problema MPC com restrições ORCA é transformado no problema QP, mantendo apenas a matriz $\boldsymbol{P}$, que não sofre alternações no decorrer do tempo.

4.3.2 Estratégia para problema inviável

Em cenários densos ou com várias trajetórias conflitantes, pode acontecer do problema de otimização ficar infactível (*infeasible*) devido às restrições. Algumas alternativas podem ser utilizadas nestes casos, considerando o contexto de robôs móveis:

1. Considerar aceleração nula, de forma que o agente vai manter a velocidade atual;
2. Forçar uma velocidade nula, parando completamente o agente;
3. Realizar um amortecimento na velocidade atual, ou seja, aplicar uma aceleração de sinal contrário ao da velocidade atual, mas com módulo próximo de zero.

Todas estas estratégias têm potencial de lidar com a inviabilidade do problema de otimização por conta do ambiente, que é não estático, com outros agentes em movimento. Entretanto, a primeira estratégia é mais propensa a colisões, caso o problema de infactibilidade não for resolvido logo. A segunda estratégia apresenta maior segurança neste quesito, mas corre o risco de estagnar completamente o sistema. Além disso, nem sempre

é possível de se realizar por conta das restrições de aceleração máxima e mínima do robô. A terceira estratégia é um híbrido que mantém a dinamicidade da primeira e a segurança da segunda, pois a característica preditiva do MPC permite antecipar a situação inviável, possibilitando que o robô mantenha alterações suaves e realizáveis de velocidade. Esta última estratégia foi empregada neste trabalho.

Capítulo 5

Resultados

Neste capítulo são apresentados os detalhes de implementação, parâmetros de sintonia dos controladores e simulações para avaliar o cumprimento dos objetivos estabelecidos. Foram desenvolvidas duas soluções, um controlador MPC em um cenário com apenas um robô, para avaliar seu funcionamento, e um controlador baseado na estratégia combinada MPC-ORCA instanciado em cada robô de um sistema multi-agente. Apesar do modelo utilizado pelo controlador ser de segunda ordem, os robôs recebem comandos de velocidade, portanto, são consideradas como entradas as componentes de velocidade (linear e angular) e como saídas as componentes de posição (no plano $\{\boldsymbol{x}_0, \boldsymbol{y}_0\}$). Os códigos implementados e vídeos das simulações estão disponíveis no repositório do grupo EASY-SPARC[1].

5.1 Implementação

A implementação dos algoritmos foi realizada na linguagem Python em cima do ecossistema ROS[2]. O ROS (do inglês *Robot Operating System*) é um *framework* voltado para o desenvolvimento de software para robôs, contando com uma coleção de bibliotecas, ferramentas e interfaces com simuladores, como Gazebo, Stage e V-Rep. Neste trabalho é utilizado o Gazebo, pois suas simulações contam com o motor de física ODE[3] (do inglês *Open Dynamics Engine*), sendo bastante empregado em desafios como o *DARPA Robotics Challenge* e o *NASA Space Robotics Challenge*. Utilizando as funcionalidades do Gazebo, foi configurado um robô com rodas diferenciais na linguagem de descrição URDF (do inglês, Unified Robot Description Format) e foi possível configurar a aplicação de uma força e torque de perturbação atuando sobre o veículo, como pode ser observado na Figura 5.1.

Cada aplicação independente ou rotina executada em ROS é definida como um *nó*, e.g. sistema de controle, simulador, etc.. A comunicação entre nós pode ser realizada a partir

[1]Disponível em `https://github.com/EASY-SPARC/TCC-Glauber`
[2]Disponível em `https://www.ros.org`
[3]Disponível em `https://www.ode.org`

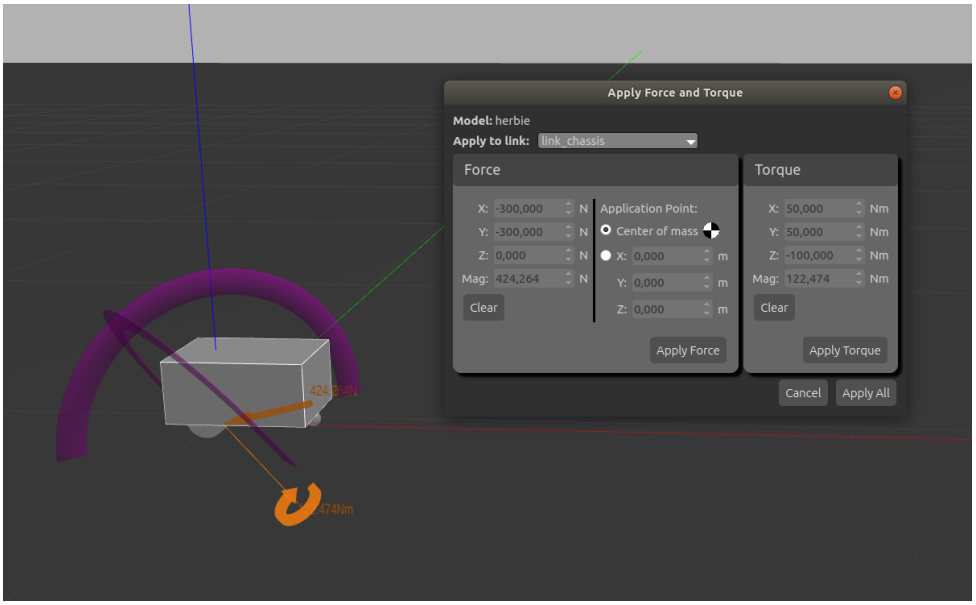

Figura 5.1: Aplicação de força e torque no robô através da interface do Gazebo.

de canais de mensagem, conhecidos como *tópicos*. A Figura 5.2 apresenta a estrutura de nós e tópicos utilizados em uma das simulações. Os robôs são controlados na simulação por um comando de velocidade publicado no tópico */cmd_vel* de cada agente. Pode-se observar que os controladores não têm comunicação direta entre si, mas podem realizar leituras do ambiente (através do tópico */gazebo/model_states*).

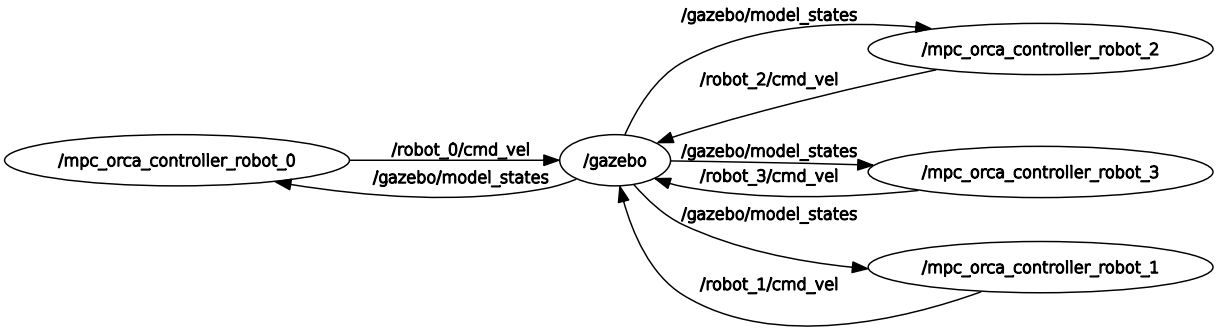

Figura 5.2: Grafo dos principais tópicos e nós ativos nas simulações multi-agentes.

O solucionador empregado para resolver o problema de programação quadrática foi o OSQP, que executa um algoritmo ADMM. No problema de otimização, as matrizes $\boldsymbol{P}$ (Equação (3.6)) e $\boldsymbol{A}$ (Equação 4.15) apresentam uma grande quantidade de valores iguais a zero, podendo ocupar bastante espaço de memória e prejudicar o funcionamento do sistema. Para resolver isso, essas matrizes são alocadas em estruturas CSC (do inglês *Compressed Sparse Column*) de dados esparsas compatíveis com OSQP e disponíveis na biblioteca de computação científica SciPy[4].

[4]Disponível em `https://www.scipy.org`

5.2 Controle de trajetória de um único robô

As simulações iniciais foram projetadas para estudar o comportamento de um controlador MPC em um robô diferencial utilizando o modelo linearizado apresentado na Seção 2.3. A Tabela 5.1 apresenta os parâmetros utilizados no controlador.

Parâmetro	Valor
$\boldsymbol{Q}(1)$	$\text{diag}(3.0, 3.0, 0.0, 0.0)^5$
$\boldsymbol{Q}(k)$	$\text{diag}(1.5, 1.5, 0.0, 0.0)^5$
$\boldsymbol{R}(k)$	$\text{diag}(0.55, 0.55)^5$
N	10
T_s	0.1 s

Tabela 5.1: Parâmetros do controlador MPC-ORCA

A escolha de $\boldsymbol{Q}$ penaliza apenas o erro de posição, desconsiderando as variáveis de velocidade, desde que sejam respeitados os limites de velocidade do robô. $\boldsymbol{Q}(1)$ apresenta valores maiores para que o controlador procure uma solução que tente reduzir o erro de posição logo na próxima interação, mas ainda considere o erro proveniente nas demais predições.

Os valores assumidos por $\boldsymbol{R}$ aplicam uma pequena penalidade na aceleração com o objetivo de que a velocidade se comporte de maneira mais suave. Com o tempo de amostragem $T_s = 0.1$ segundos, um horizonte de predição $N = 10$ permite realizar predições de 1 segundo e foi definido de forma que um aumento nesse valor não resultava em grandes contribuições no resultado.

5.2.1 Cenário 1

O primeiro cenário de simulação consiste em uma trajetória com função sigmoide partindo de $\boldsymbol{p}_{\text{start}} = [0\ 0]^\top$ para $\boldsymbol{p}_{\text{goal}} = [7\ 7]^\top$, pico de velocidade em $t_{\max} = 10$ segundos e coeficiente de declividade $K = 0.5$. As Figuras 5.3 e 5.4 expõem os resultados do primeiro cenário, apresentando respectivamente as saídas e as entradas aplicadas, assim como o vídeo *cenario_1* disponível no repositório[6].

Na Figura 5.3, as linhas sobrepostas indicam que as variáveis de posição do robô seguiram a trajetória de referência. Analisando em conjunto com as ações de controle apresentadas na Figura 5.4, a característica preditiva do controlador pode ser observada, uma vez que, mesmo com pesos nulos na variável velocidade, o controlador foi capaz de considerar as alterações dos sinais de referência ajustando a velocidade do robô de maneira suave.

[5] A função $\text{diag} : \mathbb{R}^n \to \mathbb{R}^{n \times n}$ retorna uma matriz diagonal com elementos definidos pelos argumentos.
[6] Disponível em `https://github.com/EASY-SPARC/TCC-Glauber/tree/master/videos`

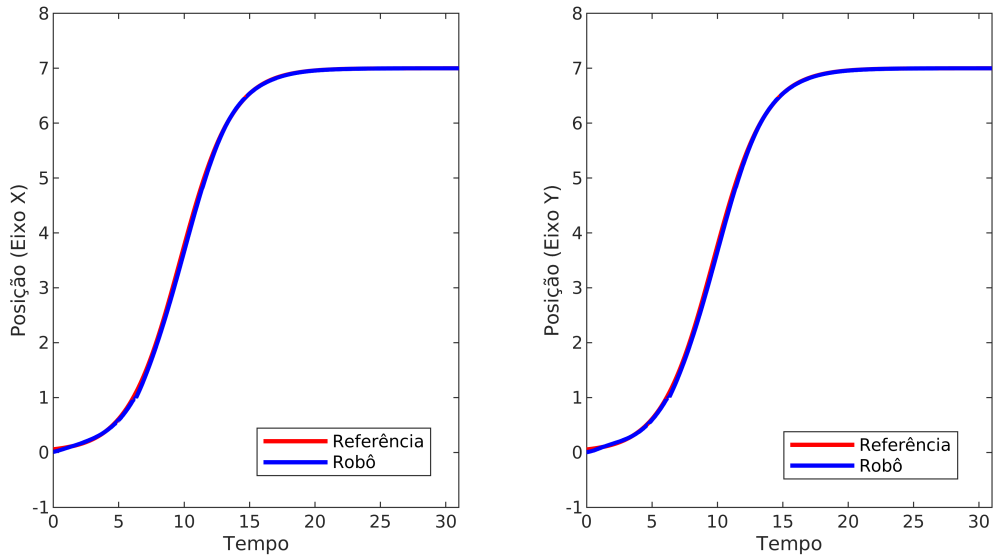

Figura 5.3: Acompanhamento de trajetória usando MPC no cenário 1.

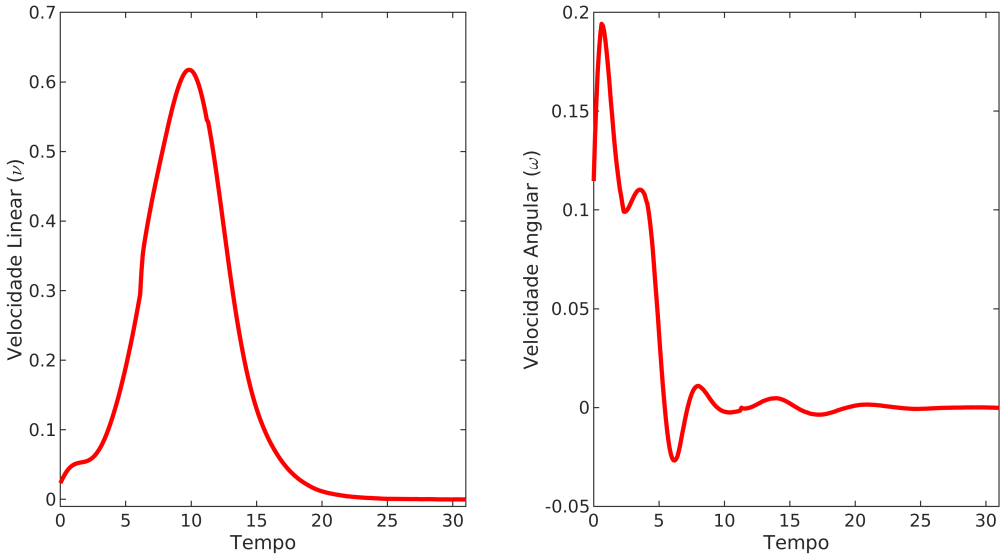

Figura 5.4: Ação de controle usando MPC no cenário 1.

Em aproximadamente 15 segundos, quando o robô está próximo de $\boldsymbol{p}_{\text{goal}}$, o controlador reduz a velocidade a uma taxa que faça a posição do robô acompanhar uma trajetória que passa a variar cada vez menos. Aos 20 segundos, $\boldsymbol{r}(k) \cong \boldsymbol{r}(k+1) \cong ... \cong \boldsymbol{r}(k+N)$ e o controlador mantém o robô completamente alinhado ao *setpoint*, por não prever alterações.

5.2.2 Cenário 2

O segundo cenário de simulação foi projetado com o objetivo de observar a reação do controlador MPC quando o sistema é sujeito a uma perturbação não modelada, representando, no caso, a aplicação inesperada de forças ou torques no robô, ou uma porção de terreno com inclinação. A perturbação é caracterizada, como na Figura 5.1 por uma força $F_p = [-300 \ \ -300 \ 0]^\top N$ e torque $\tau_p = [50 \ 50 \ \ -100]^\top Nm$ incapazes de fazer o veículo tombar, mas suficientes para deslocar o robô de sua trajetória a uma distância considerável.

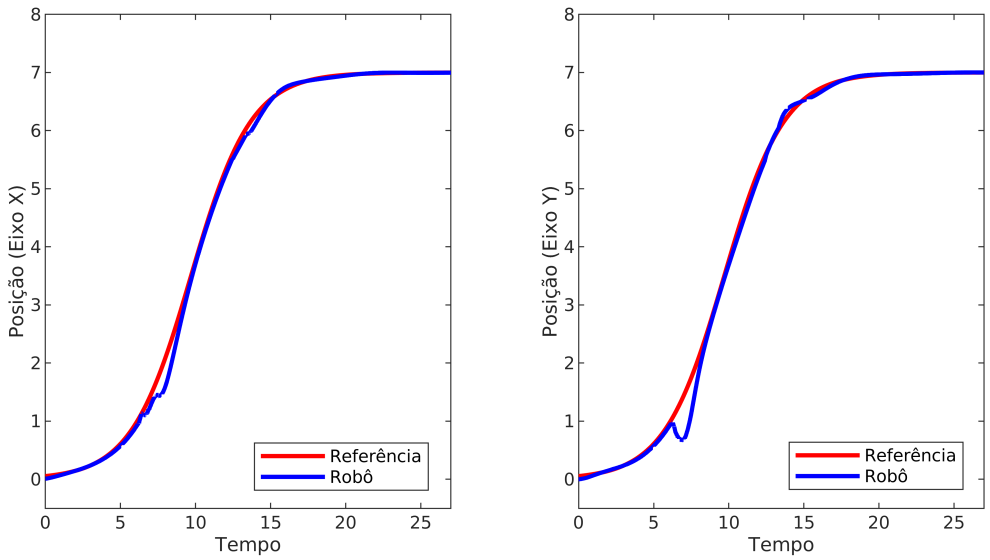

Figura 5.5: Rejeição à perturbação do controlador MPC implementado no cenário 2.

As Figuras 5.5 e 5.6, assim como o vídeo *cenario_2* disponível no repositório[6], mostram os resultados desse segundo cenário. Utilizando a interface apresentando na Figura 5.1, foram realizadas aplicações instantâneas de força e torque entre 6 e 7 segundos, provocando desvios de trajetória. Entretanto, o MPC foi capaz de manter o robô em uma posição próxima de acordo com os limites de atuação e conseguiu retomar sua trajetória.

O comportamento do agente também pode ser observado na Figura 5.7, que apresenta uma visão superior do plano em que o robô se move. A trajetória de referência impõe um percurso retilíneo com velocidade variável, onde os marcadores representam a posição do robô ou de sua referência em momentos linearmente espaçados. A perturbação pode ser observada na região em que o robô se encontra no ponto $[1 \ 1]^\top$ ocasionando um

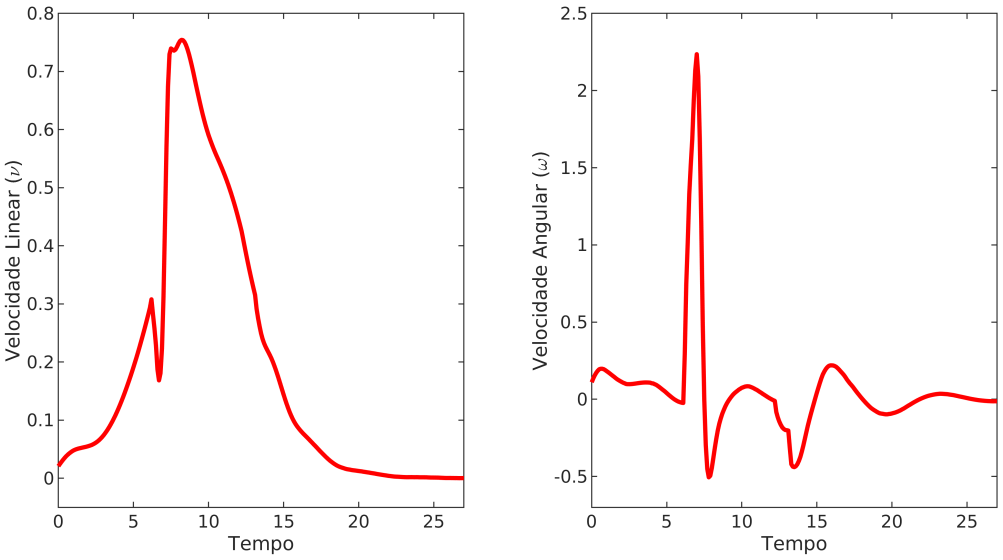

Figura 5.6: Ação de controle do MPC sob perturbação no cenário 2.

deslocamento indesejado de sua trajetória. A reação do controlador possibilitou que o robô retornasse ao seu percurso evitando movimentos excessivamente bruscos por conta da penalidade nas variáveis de aceleração da função de custo.

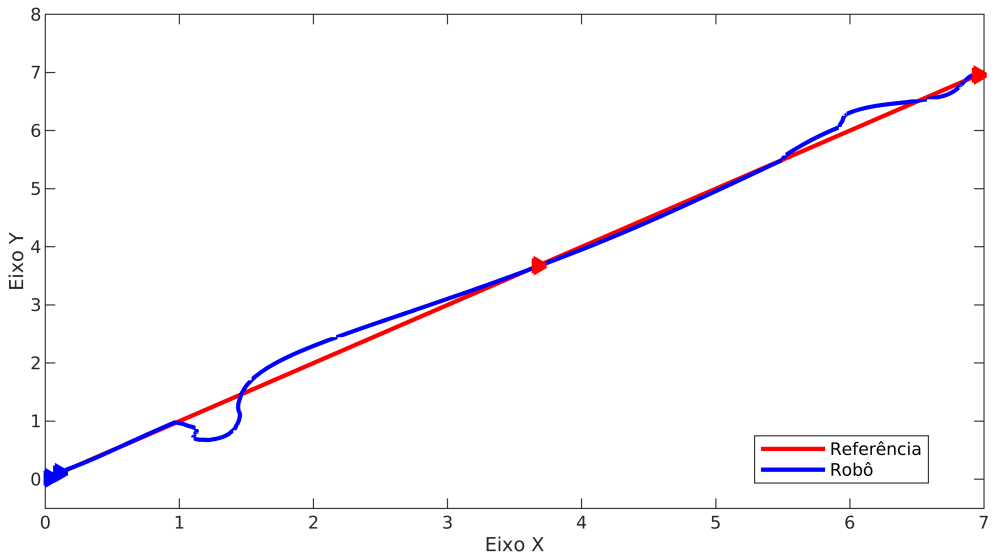

Figura 5.7: Trajetória do robô utilizando MPC diante de perturbação.

5.3 Controle de um sistema com múltiplos robôs distribuídos

Foram designados dois cenários multiagentes para validar a implementação da estratégia combinada MPC-ORCA. Desta vez, quatro robôs foram posicionados inicialmente em locais opostos e equidistantes entre si, sendo atribuídas trajetórias de referência que intencionalmente resultariam na colisão de todos os agentes aproximadamente no ponto $p = [0\ 0]^\top$.

Os parâmetros utilizados pelo controlador foram os mesmos relacionados na Tabela 5.1, com a adição do parâmetro de truncamento do cone de colisão $\tau = 5s$ para a computação do algoritmo ORCA. A Tabela 5.2 sumariza as configurações iniciais e finais utilizadas para computar as trajetórias dos cenários multiagentes.

Robô	**Cenário 3**		**Cenário 4**	
	$\boldsymbol{p}_{\text{start}}$	$\boldsymbol{p}_{\text{goal}}$	$\boldsymbol{p}_{\text{start}}$	$\boldsymbol{p}_{\text{goal}}$
1	$[-7\ 0]^\top$	$[0\ 7]^\top$	$[-7\ -7]^\top$	$[7\ 7]^\top$
2	$[7\ 0]^\top$	$[-7\ 0]^\top$	$[-7\ 7]^\top$	$[7\ -7]^\top$
3	$[0\ -7]^\top$	$[0\ 7]^\top$	$[7\ -7]^\top$	$[-7\ 7]^\top$
4	$[0\ 7]^\top$	$[0\ -7]^\top$	$[7\ 7]^\top$	$[-7\ -7]^\top$

Tabela 5.2: Valores iniciais e finais definidos para o terceiro e quarto cenários.

5.3.1 Cenário 3

A Figura 5.8 apresenta uma visão geral da movimentação dos robôs durante a terceira simulação, relacionada ao vídeo *cenario_3* disponível no repositório[6]. Apesar da trajetória de referência se tratar de uma reta que direciona cada robô a sua respectiva posição final, as restrições impostas pelo MPC-ORCA foram capazes de realizar desvios antecipados para evitar colisão entre os agentes, principalmente no ponto $p = [0\ 0]^\top$, em que todos robôs apresentam mesma variável de posição desejada ao mesmo tempo.

Como pode ser observado na Figura 5.9, o robô 1 realizou um desvio em sua trajetória prevendo a colisão com outros agentes, de maneira mais evidente na variável de posição relacionada ao eixo Y, que, na trajetória de referência, permanecia inalterado. Entretanto, também podem ser observados desvios na variável de posição relacionada ao eixo X, uma vez que o gráfico à esquerda na Figura 5.9 apresenta pequenas diferenças do gráfico à esquerda na Figura 5.3.

Percebe-se que o controlador procurou assumir velocidades que evitasse o impacto enquanto mantivesse o robô o mais próximo possível da referência para que ele pudesse voltar para a trajetória desejada quando não houvesse mais riscos de colisão.

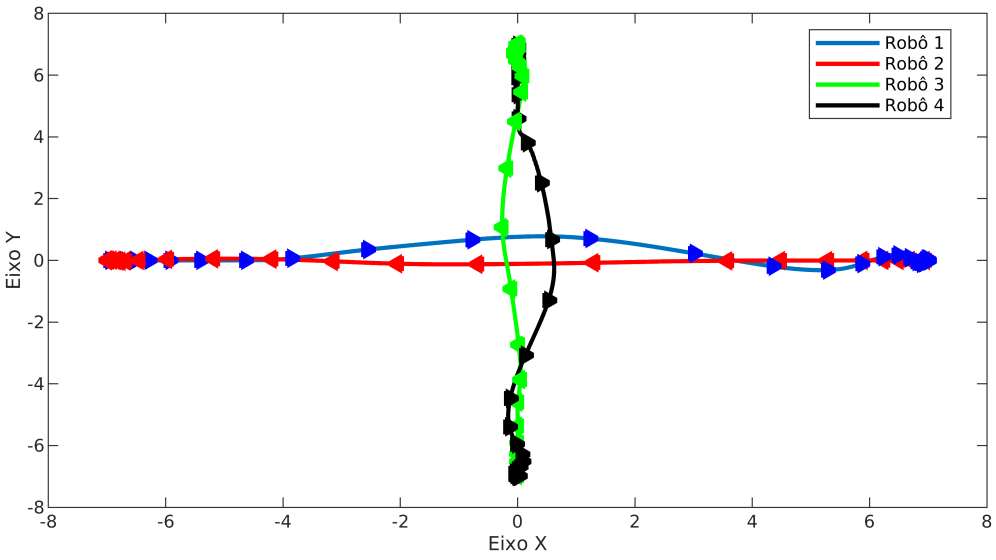

Figura 5.8: Trajetórias resultantes da aplicação do MPC-ORCA no cenário 3.

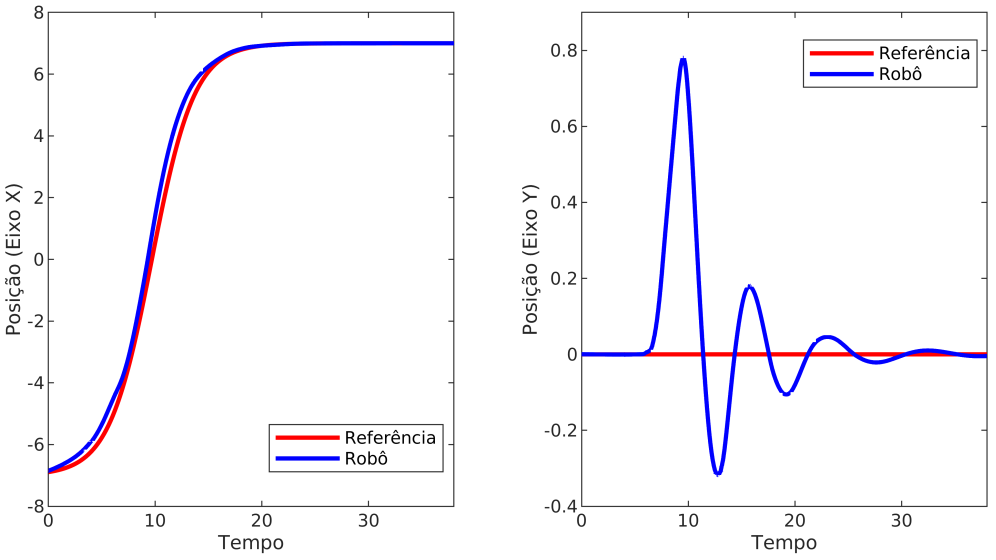

Figura 5.9: Comportamento do robô 1 utilizando MPC-ORCA no cenário 3.

5.3.2 Cenário 4

No quarto cenário, foram designadas configurações diferentes para mostrar a generalidade da solução proposta. A Figura 5.10 apresenta a visão planar das trajetórias assumidas pelos robôs nesse cenário de simulação, em que trajetórias livres de colisão também podem ser observadas. A simulação pode ser conferida no vídeo *cenario_4* disponível no repositório[6].

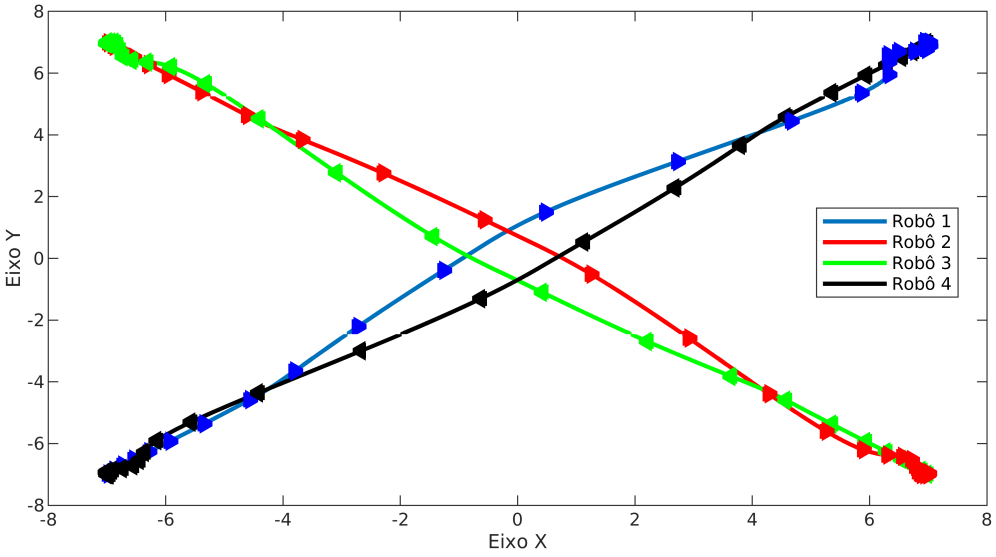

Figura 5.10: Trajetórias resultantes da aplicação do MPC-ORCA no cenário 4.

Os resultados desse experimento, para o robô 1, podem ser conferidos na Figura 5.11. Apesar da aplicação de restrições de prevenção de colisão, o caminho percorrido pelo robô não apresentou grande divergência da trajetória de referência.

5.4 Discussão

Algumas características puderam ser percebidas no processo de sintonia dos parâmetros do controlador e como elas afetaram os resultados obtidos. Nos cenários 1 e 2, a região de velocidades admissíveis em cada iteração é definida pelas restrições do modelo utilizado e os limites de operação (velocidade máxima $\boldsymbol{v}_{max}$ e velocidade mínima $\boldsymbol{v}_{min}$). Já nos cenários 3 e 4, o problema de otimização se tornou ainda mais restritivo nessa variável por conta da aplicação do algoritmo ORCA no horizonte de predição, levando a casos em que o problema se tornava inviável.

A possibilidade de reagir antecipadamente e a aplicação da estratégia discutida na Seção 4.3.2 permitiram menor ocorrência de problemas inviáveis. Entretanto, limites de velocidade muito estreitos dificultavam que o controlador fizesse o robô retornar à trajetória desejada, depois de um caso de problema inviável decorrente do algoritmo

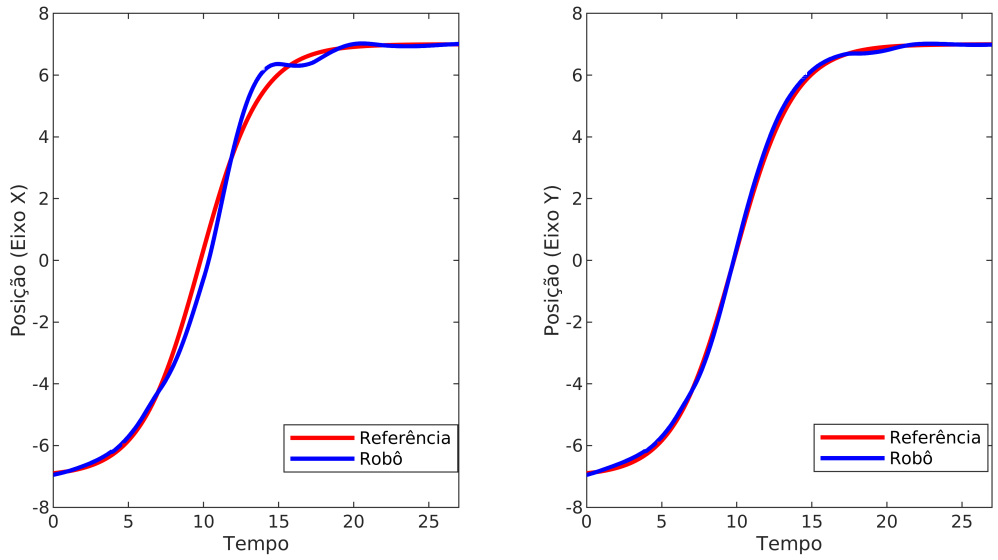

Figura 5.11: Comportamento do robô 1 utilizando MPC-ORCA no quarto cenário.

ORCA, uma vez que $\boldsymbol{v}_{max}$ e $\boldsymbol{v}_{min}$ são restrições rígidas em todo horizonte de predição. Diante disso, um relaxamento dessas restrições se tornou necessário, no caso, aumentando o intervalo de operação das variáveis de velocidade $[\boldsymbol{v}_{min}\ \boldsymbol{v}_{max}]$.

É observado também que, durante a execução das simulações multiagentes, a aplicação de uma penalidade na aceleração na função de custo (através da matriz $\boldsymbol{R}$) resulta em desvios de prevenção de colisão mais suaves, que são mais fáceis de corrigir posteriormente, tanto do ponto de vista de menor variação na entrada do sistema como no algoritmo de otimização.

Em ambas simulações de cenários multiagentes, os veículos conseguem realizar movimentos rápidos, uma vez que se posicionam em configurações livres de colisões antecipadamente, o que indica que o MPC-ORCA pode ser considerado uma técnica aplicável em ambientes com obstáculos em movimento.

Conclusão

Este trabalho apresentou uma aplicação de controle preditivo baseado em modelo (MPC) combinado com o algoritmo para prevenção de colisão ORCA em um sistema distribuído de robôs móveis com objetivos individuais, sem a necessidade de replanejamento recorrente de trajetória nem comunicação direta entre os agentes. Para cada iteração de um horizonte de predição, foram computadas restrições de velocidade que resultariam na colisão entre o robô controlado e outros agentes, sendo inseridas em um problema de otimização quadrática juntamente com as restrições determinadas pelo MPC, como limites de velocidade e aceleração.

Foram utilizados robôs com rodas diferenciais, apresentando restrições cinemáticas, o que demandou a linearização do modelo para a aplicação do controlador implementado. No entanto, os robôs conseguiram estabelecer trajetórias suaves e livres de colisão, realizando pequenos desvios das trajetórias de referência. Um robô empregando essa estratégia consegue lidar com ambientes altamente mutáveis, onde os demais agentes podem se movimentar em alta velocidade. Uma vantagem da solução desenvolvida em comparação a proposta por outros estudos é a utilização de uma trajetória de referência ao invés de uma posição objetivo, pois permite que o robô se movimente de acordo com uma rota descrita por uma função genérica, definindo variações de posição e velocidade no decorrer do tempo.

Apesar deste estudo abordar robôs se movendo no plano, as técnicas utilizadas podem ser expandidas para o controle e prevenção de colisão em robôs operando em espaços de configurações com dimensões superiores, e.g. veículos aéreos e manipuladores robóticos, sendo necessário adaptar o modelo do sistema e a determinação das restrições ORCA. Além disso, podem ser desenvolvidos trabalhos voltados à implementação do MPC-ORCA em um nível de controle ainda mais fundamental do robô, com comandos de torque ou aceleração das rodas.

Bibliografia

[Ahmad Abu Hatab, 2013] Ahmad Abu Hatab, R. D. (2013). Dynamic Modelling of Differential-Drive Mobile Robots using Lagrange and Newton-Euler Methodologies: A Unified Framework. *Advances in Robotics & Automation*, 02(02).

[Arkin and Balch, 1998] Arkin, R. C. and Balch, T. (1998). *Cooperative Multiagent Robotic Systems*, page 277–296. MIT Press, Cambridge, MA, USA.

[Barata et al., 2014] Barata, F. A., Igreja, J. M., and Neves-Silva, R. (2014). Distributed MPC for thermal comfort and load allocation with energy auction. *International Journal of Renewable Energy Research*, 4(2):371–383.

[Bouzoualegh et al., 2019] Bouzoualegh, S., Guechi, E.-H., and Kelaiaia, R. (2019). Model Predictive Control of a Differential-Drive Mobile Robot. *Acta Universitatis Sapientiae Electrical and Mechanical Engineering*, 10(1):20–41.

[Boyd et al., 2004] Boyd, S., Boyd, S., Vandenberghe, L., and Press, C. U. (2004). *Convex Optimization*. Berichte über verteilte messysteme. Cambridge University Press.

[Boyd et al., 2010] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2010). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122.

[Camacho and Bordons, 2007] Camacho, E. F. and Bordons, C. (2007). *Model Predictive control*. Advanced Textbooks in Control and Signal Processing. Springer London, London, 2nd edition.

[Canny and Reif, 1987] Canny, J. and Reif, J. (1987). New lower bound techniques for robot motion planning problems. pages 49–60.

[Carlos et al., 1989] Carlos, E. G. A. R. A., Preti, M., and Morari, M. (1989). Model Predictive Control : Theory and Practice a Survey. *Automatica*, 25(3):335–348.

[Carpenter, 2002] Carpenter, R. J. (2002). Decentralized control of satellite formations. *International Journal of Robust and Nonlinear Control*, 12(2-3):141–161.

[Cheng et al., 2017] Cheng, H., Zhu, Q., Liu, Z., Xu, T., and Lin, L. (2017). Decentralized navigation of multiple agents based on ORCA and model predictive control. *IEEE International Conference on Intelligent Robots and Systems*, 2017-Septe:3446–3451N.

[Choset et al., 2005] Choset, H., Lynch, K., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., and Thrun, S. (2005). *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press.

[D'Andrea, 2012] D'Andrea, R. (2012). Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead. *IEEE Transactions on Automation Science and Engineering*, 9(4):638–639.

[Ehlinger and Mesbah, 2017] Ehlinger, V. and Mesbah, A. (2017). *Model Predictive Control of Chemical Processes: A Tutorial*, pages 367–402.

[Fiorini and Shiller, 2002] Fiorini, P. and Shiller, Z. (2002). Motion planning in dynamic environments using the relative velocity paradigm. pages 560–565.

[Graetz and Michaels, 2018] Graetz, G. and Michaels, G. (2018). Robots at Work. *The Review of Economics and Statistics*, 100(5):753–768.

[Hartanto et al., 2019] Hartanto, R., Arkeman, Y., Hermadi, I., Sjaf, S., and Kleinke, M. (2019). Intelligent Unmanned Aerial Vehicle for Agriculture and Agroindustry Intelligent Unmanned Aerial Vehicle for Agriculture and Agroindustry. {*IOP*} *Conference Series: Earth and Environmental Science*, 335:012001.

[Haykin and Veem, 1999] Haykin, S. and Veem, B. V. (1999). *Signals and Systems*.

[Lu, 2017] Lu, Y. (2017). Industry 4.0: A survey on technologies, applications and open research issues. *Journal of Industrial Information Integration*, 6:1–10.

[Minguez et al., 2016] Minguez, J., Lamiraux, F., and Laumond, J.-P. (2016). *Motion Planning and Obstacle Avoidance*, pages 1521–1548.

[Mouad et al., 2012] Mouad, M., Adouane, L., Khadraoui, D., and Martinet, P. (2012). *Mobile robot navigation and obstacles avoidance based on Planning and Re-Planning algorithm*, volume 45. IFAC.

[Ogata, 1995] Ogata, K. (1995). *Discrete-time control systems*, volume 2. Prentice Hall Englewood Cliffs, NJ.

[Ota, 2006] Ota, J. (2006). Multi-agent robot systems as distributed autonomous systems. *Advanced Engineering Informatics*, 20(1):59–70.

[Poignet and Gautier, 2000] Poignet, P. and Gautier, M. (2000). Nonlinear model predictive control of a robot manipulator. *International Workshop on Advanced Motion Control, AMC*, (2):401–406.

[Qin and Badgwell, 2003] Qin, S. and Badgwell, T. A. (2003). A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764.

[Ren and Atkins, 2007] Ren, W. and Atkins, E. (2007). Distributed multi-vehicle coordinated control via local information exchange. *International Journal of Robust and Nonlinear Control*, 17(10-11):1002–1033.

[Ricker and Lee, 1995] Ricker, N. L. and Lee, J. H. (1995). Nonlinear model predictive control of the Tennessee Eastman challenge process. *Computers and Chemical Engineering*, 19(9):961–981.

[Roland, 2004] Roland, S. (2004). *Introduction to Autonomous Mobile Robots*. London.

[Sariff and Buniyamin, 2006] Sariff, N. and Buniyamin, N. (2006). An overview of autonomous mobile robot path planning algorithms. *SCOReD 2006 - Proceedings of 2006 4th Student Conference on Research and Development Towards Enhancing Research Excellence in the Region*, (June):183–188.

[Sharma K. et al., 2016] Sharma K., R., Honc, D., and Dusek, F. (2016). Predictive Control Of Differential Drive Mobile Robot Considering Dynamics And Kinematics. pages 354–360.

[Siciliano et al., 2009] Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2009). *Robotics*. Advanced Textbooks in Control and Signal Processing. Springer London, London.

[Skjong et al., 2019] Skjong, E., Johansen, T. A., and Molinas, M. (2019). Distributed control architecture for real-time model predictive control for system-level harmonic mitigation in power systems. *ISA Transactions*, 93(February):231–243.

[Stellato et al., 2018] Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S. (2018). OSQP: An Operator Splitting Solver for Quadratic Programs. *2018 UKACC 12th International Conference on Control, CONTROL 2018*, (1):339.

[Stentz, 1995] Stentz, A. (1995). The focussed D* algorithm for real-time replanning. *14th International Joint Conference on Artificial intelligence (IJCAI)*, 95(August):1652–1659.

[Sycara, 1998] Sycara, K. P. (1998). Multiagent Systems. 19(2):79–92.

[Trevelyan et al., 2016] Trevelyan, J., Kang, S., and Hamel, W. (2016). *Robotics in Hazardous Applications*, pages 1177–1201.

[Van Den Berg et al., 2011] Van Den Berg, J., Guy, S. J., Lin, M., and Manocha, D. (2011). Reciprocal n-body collision avoidance. In Pradalier, C., Siegwart, R., and Hirzinger, G., editors, *Robotics Research*, pages 3–19, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Van den Berg et al., 2008] Van den Berg, J., Ming Lin, and Manocha, D. (2008). Reciprocal Velocity Obstacles for real-time multi-agent navigation. In *2008 IEEE International Conference on Robotics and Automation*, volume 23, pages 1928–1935. IEEE.