XXX Congresso de Iniciação Científica UNICAMP

Controle de Modelo Preditivo de Horizonte Finito Aplicado a um Veículo Robótico Autônomo

Aluno: Gabriel Barbosa Haj (FEM-Unicamp)
Orientador: André Ricardo Fioravanti (FEM-Unicamp)

28 de julho de 2022

Palavras-Chave— Robótica Móvel, Controle Numérico, Otimização

1 Introdução

1.1 Controle de Modelo Preditivo

A dinâmica de um veículo autônomo operando no mundo real é geralmente não-linear, sujeita a limitações físicas do controlador e a perturbações não previstas. A dinâmica desse sistema não-linear com entrada de controle $\mathbf{u} \in \mathbb{R}^{N_u}$ pode ser escrita de maneira generica como:

$$\dot{\mathbf{x}}(t) = g(\mathbf{x}(t), \mathbf{u}(t), t) \tag{1}$$

no qual $\mathbf{x} \in \mathbb{R}^{N_x}$ é o espaço de estados, que pode ser composto, por exemplo, pela posição e pela velocidade do veículo, e g é o campo vetorial que define a dinâmica do sistema.

Para se aplicar técnicas convencionais de controle, é necessário fazer simplificações no modelo (como linearizações), o que diminui a robustez do controlador e as situações nas quais ele apresentará um comportamento adequado. Para superar essas limitações, técnicas mais sofisticadas de controle podem ser aplicadas, como por exemplo, o *Model Predictive Control* (MPC), também conhecido como *Receding Horizon Control* (RHC). A estratégia desse tipo de controle é calcular um sinal de controle ótimo $\mathbf{u}$, através da formulação iterativa de um problema de otimização que minimiza uma função de custo J em um horizonte de tempo finito. Esse problema pode ser descrito como:

$$\begin{aligned} \underset{\mathbf{u}}{\text{minimize}}\ & J(\mathbf{x_{t_0}}, \mathbf{u}) \\ \text{s.a}\ & : \dot{\mathbf{x}}(t) = g(\mathbf{x}(t), \mathbf{u}(t), t), \\ & \mathbf{x}(t_0) = \mathbf{x_{t_0}}, \\ & \mathbf{u}(t) \in U, \forall t \in [t_0, t_0 + N - 1], \\ & \mathbf{x}(t) \in X, \forall t \in [t_0, t_0 + N] \end{aligned} \tag{2}$$

em que J é uma função de custo a ser minimizada, X e U são os conjuntos de estados e de esforços de controle adimissíveis.

A função de custo define a estratégia de controle, já que a escolha do que deve ser minimizado guia o funcionamento do controlador. Um exemplo de função de custo, bastante similar ao regulador quadrático, pode ser descrita por:

$$J(\mathbf{x_{t_0}}, \mathbf{u}) = \int_{t_0}^{t_0+N_t-1} (\mathbf{x} - \mathbf{x_r})^T \mathbb{Q} (\mathbf{x} - \mathbf{x_r}) dt + \int_{t_0}^{t_0+N_c-1} (\mathbf{u} - \mathbf{u_r})^T \mathbb{R} (\mathbf{u} - \mathbf{u_r}) dt \tag{3}$$

que calcula a distância quadrática do vetor de estados $\mathbf{x}$ e de controle $\mathbf{u}$ em relação a vetores de referência $\mathbf{x_r}$ e $\mathbf{u_r}$, respectivamente. Além disso, $\mathbb{Q} \in \mathbb{R}^{N_x \times N_x}$ é uma matrix semi-definida positiva que quantifica a penalização em torno dos estados e $\mathbb{R} \in \mathbb{R}^{N_u \times N_u}$ é uma matriz definida positiva que quantifica a penalização em torno do esforço de controle. N_t é o horizonte de predição e N_c é o horizonte de controle, sendo $N_t > N_c$ e o sinal de controle é mantido constante a partir de N_c

O problema descrito por (2) é chamado *Optimal Control Problem* (OCP). Discretizando o intervalo de interesse, temos que este problema se torna computacionalmente tratável, e a solução desse problema de otimização gera uma sequência de controle ótimo $\mathbf{u}^* = (u_{t_0}, ..., u_{t_0+N-1})$. É neste contexto que o Casadi [1] se faz útil. O Casadi é uma ferramenta open-source para otimização númerica. Essa ferramenta é, em geral, utilizada para facilitar a resolução de problemas de Programação Não-Linear(NLP). O Casadi não resolve as NLPs mas se utiliza de *plugins* (que, na realidade, são *solvers* de problemas de otimização), como por exemplo o IPOPT [9]. Convenientemente, podemos transformar a OCP descrita em (2) em um NLP e resolvê-la com auxilio do Casadi.

Outra ferramenta que se faz importante nesse contexto de MPC, é o MPCTools [8], construido a partir do Casadi, que facilita a implementação e a solução de problemas de MPC. Neste trabalho optou-se por utilizar ambos pacotes utilizando-se a linguagem de programção Python e todos os códigos desenvolvidos estão disponíveis em um repositorio público [2].

1.2 O Projeto VERDE

O presente trabalho é parte de um projeto multi-disciplinar denominado VERDE - Veículo Robótico Elétrico de Diferencial Eletrônico - originado pela cooperação existente entre a FEM-UNICAMP, o Centro de Tecnologia da Informação Renato Archer CTI-Campinas e a FAPESP. Alguns trabalhos desse mesmo grupo de pesquisa que serão utilizados neste trabalho são [6], [3], [5] e [7]

Figura 1: Veículo VERDE.

O veículo possui suspensão independente nas quatro rodas, locomoção controlada por três atuadores, sendo dois deles motores de corrente contínua instalados de forma independente nas rodas traseiras, e um deles um servomotor instalado entre o sistema de direção. As rodas dianteiras apresentam uma configuração em Ackermann.

2 Resultados e Discussões

2.1 Modelo do Veículo VERDE

Neste trabalho utilizaremos o modelo cinemático linearizado variante no tempo de bicicleta do VERDE, e esse é descrito por:

$$\begin{bmatrix} \dot{y} \\ \dot{\psi} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0 & u_{ref}(t) & 0 \\ 0 & 0 & 1 \\ 0 & 0 & a_r \end{bmatrix} \begin{bmatrix} y \\ \psi \\ r \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ b_r \end{bmatrix} \delta \tag{4}$$

$$\dot{\mathbf{x}} = A(t)\mathbf{x} + B\delta$$

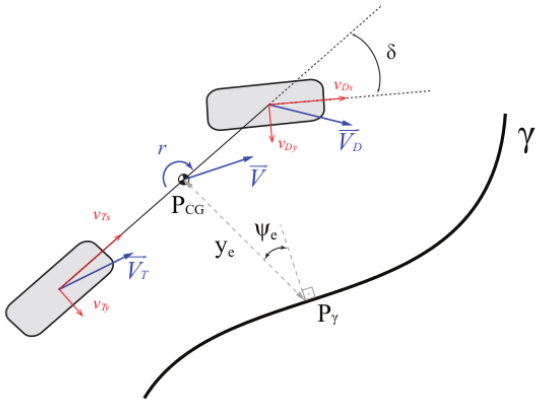

Figura 2: Modelo cinemático de bicicleta e referências de trajetória(retirado de [6])

Neste modelo, y é a posição lateral do carrinho, ψ é o angulo que o carrinho faz com o eixo X inercial, conhecido como ângulo de guinada, e r é a velocidade de rotação do ângulo de guinada.

2.2 *O Rastreamento de Trajetória Baseado no MPC*

O objetivo do rastreamento de trajetória, ou *Trajectory Tracking* é seguir uma velocidade e uma trajetória de referência. Um outro problema comum a esse meio é conhecido como *Path Following*, em que a trajetória deve ser seguida sem uma velocidade de referência [4]. Neste trabalho nos concentraremos no primeiro problema. O problema de *Trajectory Tracking* pode ser formulado em tempo discreto como:

$$\mathbf{x}(k) - \mathbf{x_{ref}}(k) = 0, \mathbf{x_{ref}} = \begin{bmatrix} y_{ref}(k) \\ \psi_{ref}(k) \\ r_{ref}(k) \end{bmatrix} \tag{5}$$

Desse modo, cria-se uma trajetória de referência virtual. Os sinal de controle vêm da estratégia conhecida como *Feedfoward Control* [4] que o modelo deve seguir, dada pela equação do modelo quando a entrada é o parâmetro de referência.

Logo, (3) pode ser reescrita como:

$$J(\mathbf{x_0}, \delta) = \sum_{k=0}^{N_t} (\mathbf{x}[k] - \mathbf{x_{ref}}[k])^T \mathbb{Q} (\mathbf{x}[k] - \mathbf{x_{ref}}[k]) + \sum_{k=0}^{N_c - 1} (\delta[k] - \delta_{ref}[k])^T R (\delta[k] - \delta_{ref}[k]) \tag{6}$$

e assim, sem perdas de generalidade considerando $t_0 = 0$, a OCP (2) em tempo discreto se torna:

$$\begin{aligned} &\min_{\mathbf{u}} \ J(\mathbf{x_0}, \delta) \\ s.a. : &\ \mathbf{x}(k+1) = A_k(k)\mathbf{x} + B_k\delta, \\ &\ \mathbf{x}(0) = \mathbf{x_0}, \\ &\ \delta(k) \in U, \forall k \in [0, N_c - 1], \\ &\ \mathbf{x}(k) \in X, \forall k \in [0, N_t] \end{aligned} \tag{7}$$

Sendo A_k e B_k as versões discretizadas a uma taxa de $\Delta = 0.05s$ das matrizes A e B que compõe o sistema.

2.3 Análise dos Resultados

A trajetória escolhida para ser seguida é conhecida como mudança de faixa e então, foram variados os parâmetros de peso da matriz $\mathbb{Q}$ e o peso R, N_t e N_u. Além disso, foram utilizadas duas métricas para avaliar a trajetória efetuada pelo sistema,através da distância entre os pontos da trajetória realizada e da trajetória de referência, sendo a primeira métrica, a média das distâncias, e a segunda, a distância máxima no trajeto:

$$erro_1 = \frac{\sum_{k=0}^{N_t} \sqrt{(\mathbf{x}^2[k] - \mathbf{x_{ref}}^2[k])}}{N_t + 1} \qquad \text{e} \qquad erro_{max} = \max(\sqrt{(\mathbf{x}^2[k] - \mathbf{x_{ref}}^2[k])}) \tag{8}$$

Com essa métrica, verificou-se que o parâmetro N_t e os pesos relativos aos estados y e ψ eram os que mais contribuiam para o desempenho dos controladores. Além disso, pode-se perceber que o controlador não foi capaz de seguir simultâneamente, e de maneira fiel, a posição lateral e o ângulo de guinada, respectivamente, y e ψ.

Após testar alguns valores de parâmetros, chegou-se aos parâmetros que apresentou um resultado satisfatório nas duas métricas simultâneamente, respectivamente de 0.254 m para a média das distâncias e 0.405 m para a distância máximo:

$$\mathbb{Q} = \begin{bmatrix} 20 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \; R = 0.01, \; N_t = 5, \; N_c = 1$$

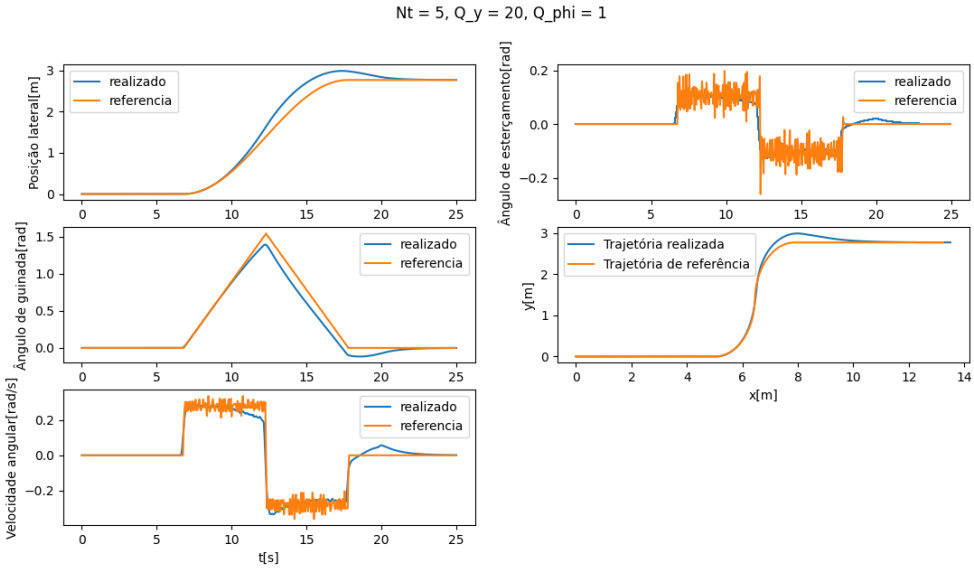

Figura 3: Evolução dos estados do sistema(a esquerda), esforço de controle e trajetória realizada (a direita), $erro_1 = 0.254\ m$ e $erro_{max} = 0.405\ m$

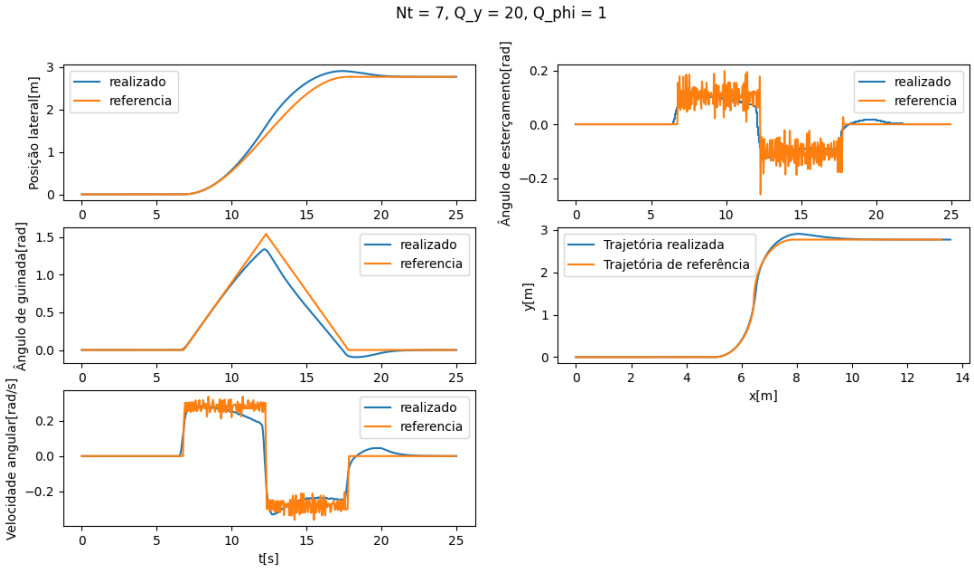

Figura 4: Aumento no horizonte de predição N_t $\quad erro_1 = 0.268\ m$ e $erro_{max} = 0.387\ m$

Embora a trajetória do segundo controlador pareça mais fiel à trajetória de referência, sendo seu $erro_1$

maior que o do primeiro controlador, e o seu $erro_{max}$ menor, percebe-se que ele desenvolveu a trajetória com velocidade maior que a desejada, e assim também manteve-se distante da posição esperada para aquele instante de tempo. Para o caso no qual a trajetória a ser seguida não possui uma velocidade esperada, o caso de *Path Following*, este controlador teria performado melhor que o anterior.

3 Conclusão

A partir dos trabalhos realizados até agora, pode-se concluir que os pacotes Casadi e MPCTools suprem as necessidades para a implementação de diversos algorítimos de MPC. Com eles, pôde-se aplicar a estratégia de *Trajectory Tracking* no modelo cinemático de bicicleta do VERDE, encontrando um controlador que possui erro médio de 0.254 m e erro máximo de 0.405 m.

Ainda deseja-se implementar esse controlador no sistema físico real de forma a validar os resultados experimentalmente e comparará-los com outros resultados desenvolvidos por trabalhos do mesmo grupo de pesquisa, como [5], [6] e [7].

Referências

[1] Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[2] Gabriel Barbosa Haj. Model predictive control usando casadi e mpctools aplicado ao verde, 2022. URL: https://github.com/gabrielhaj/mpc-verde.

[3] Rafael de Angelis Cordeiro et al. Modelagem e controle de trajetória de um veículo robótico terrestre de exterior. *(UNICAMP)*, 2018.

[4] Mohamed W. Mehrez, Karl Worthmann, George K.I. Mann, Raymond G. Gosine, and Timm Faulwasser. Predictive path following of mobile robots without terminal stabilizing constraints. *IFAC-PapersOnLine*, 50(1):9852–9857, 2017. 20th IFAC World Congress. URL: https://www.sciencedirect.com/science/article/pii/S2405896317313733, doi:https://doi.org/10.1016/j.ifacol.2017.08.907.

[5] Alexandre M Ribeiro, Alexandra Moutinho, André R Fioravanti, and Ely C de Paiva. Estimation of tire–road friction for road vehicles: a time delay neural network approach. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 42(1):1–12, 2020.

[6] Alexandre Monteiro Ribeiro. Desenvolvimento de uma abordagem de diferencial eletrônico para um veículo elétrico robótico multitração. 2016.

[7] AM Ribeiro, AR Fioravanti, A Moutinho, and EC de Paiva. Nonlinear state-feedback design for vehicle lateral control using sum-of-squares programming. *Vehicle System Dynamics*, pages 1–27, 2020.

[8] Rawlings J.B. Risbeck, M.J. Mpctools: Nonlinear model predictive control tools for casadi, 2016. URL: https://bitbucket.org/rawlings-group/octave-mpctools.

[9] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.