SISTEMA COMPUTACIONAL PARA CONTROLE DE ROBÔS MÓVEIS

Lucas Schulze [1], Douglas Wildgrube Bertol[2]

[1] Acadêmico do Curso de Engenharia Elétrica - CCT bolsista PIVIC/UDESC
[2] Orientador, Departamento de Engenharia Elétrica CCT – douglas.bertol@udesc.br

Palavras-chave: Robótica. Controle Preditivo. ROS.

O uso cada vez mais disseminado da robótica no nosso cotidiano exige sistemas cada vez mais seguros e confiáveis, que necessitam de ferramentas que possibilitem o desenvolvimento ágil de software e hardware. Controladores preditivos, como o MPC (sigla para o termo em inglês *Model Predictive Control*), ganham cada vez mais notoriedade na robótica devido à possibilidade de trabalhar com sistemas multiváriaveis, tratar restrições e compensar atrasos, porém necessitam de modelos matemáticos mais representativos e recursos numéricos. O *Robot Operation System* (ROS) é um *framework* para modularidade e reutilização de software para robótica, disponibilizando e incentivando o compartilhamento de ferramentas e bibliotecas. Apesar de possuir muitos recursos para navegação e processamento de imagem, são poucos os mecanismos para implementação dentro do ROS de técnicas de controle além das abordagens clássicas, como o Proporcional Integral Derivativo (PID). Nesse contexto de atual consolidação do ROS, esse trabalho tem como objetivo desenvolver um modulo, uma biblioteca ou pacote para o ROS para implementação de controladores MPC para robótica.

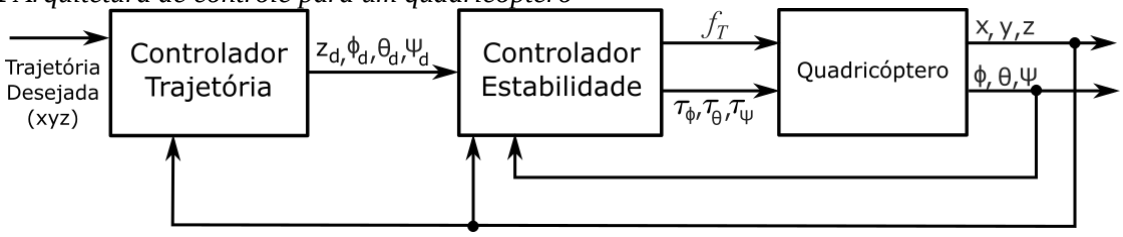
Após o estudo de ferramentas conhecidas na literatura e indústria, como o *roscontrol* e o *orocos*, foi definida a topologia dos controladores e desenvolvido um controlador de trajetória para um quadricóptero, sendo realizados testes utilizando o modelo matemático e o simulador Gazebo, ferramenta nativa do ROS.

Após alguns testes, o *roscontrol* demonstrou ser limitado apenas a controladores PID. Já o *orocos* apresentou uma gama maior de aplicações, porém, por não ser uma ferramenta nativa do ROS, a implementação no ROS é pouco flexível. Ao buscar outros pacotes e ferramentas para implementação de controladores preditivos no ROS, não foram encontrados pacotes disponíveis para o ROS com essa técnica que tivessem utilização suficientemente simples e que fossem genéricos para qualquer sistema. Um dos possíveis motivos é que o elemento principal para aplicação do MPC é o *solver*, algoritmo que resolve o problema de otimização, dessa forma, ao desenvolver um pacote, pode-se ficar limitado ao software desenvolvido por um terceiro. Algo comum em alguns *solver* é a possibilidade de gerar o código pronto para aplicação em uma linguagem de mais baixo nível, como C ou C++, a partir de ambientes de simulação ou linguagens de mais alto nível, como Matlab e Python. Um *solver* com essas características é o OSQP, que possui também características que aceleram o procedimento de cálculo, como possibilidade de *warm start* e algoritmos livres de divisão. Dessa forma, decidiu-se por

desenvolver um novo pacote para o ROS que implemente um MPC utilizando o gerador de código do OSQP.

O modelo matemático para um quadricóptero é obtido por meio da analise cinemática e dinâmica e com as equações de Newton-Euler. Por ser um sistema subatuado, seis graus de liberdade para serem controlados pelos quatro rotores, usualmente se projeta dois controladores em cascata, Figura 1, o primeiro com o objetivo de estabilização em uma altura e postura angular desejadas, que deve ter tempo de execução suficiente para a dinâmica do sistema. E outro externo que a partir de uma posição ou trajetória desejada em xyz, define os valores de referência para o primeiro.

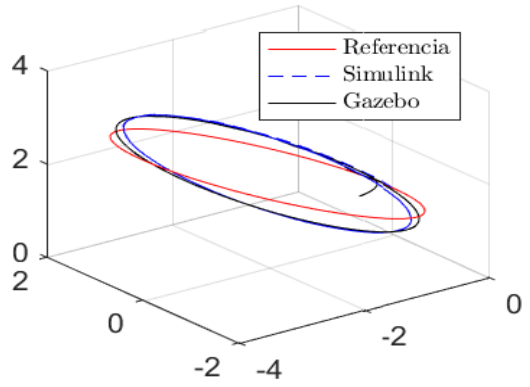
Fig. 1 *Arquitetura de controle para um quadricóptero*



Foi desenvolvido um controlador *Explicit* MPC para o controlador de estabilidade, sendo implementado por meio de um *plugin* diretamente no Gazebo. Para o controlador de trajetória, foco desse trabalho, foi desenvolvido um MPC utilizando o *solver* OSQP, permitindo a sua aplicação na solução proposta.

Dessa forma, foi desenvolvido o pacote *iris_traj_osqp* para o ROS, que implementa o controlador MPC a partir dos códigos gerados, dessa forma, não é necessário fazer qualquer alteração nos arquivos do pacote. É possível realizar toda a etapa de projeto dos controladores em outras linguagens como Python e Matlab e utilizá-los exatamente da mesma forma no Gazebo. Os resultados são apresentadas na Figura 2, em que é possível verificar a similaridade entre o modelo matemático no Simulink e o simulador utilizando o pacote.

Fig. 2 *Comparação dos resultados no Simulink e Gazebo*



Para trabalhos futuros, seria tornar o pacote mais flexível, sendo possível o usuário configurar alguns parâmetros da troca de informações e demais características do controlador sem necessidade de alterar o código fonte do pacote e/ou recompilá-lo.