Pontifícia Universidade Católica
do Rio de Janeiro

Leonardo Garcia Moraes

Framework de Integração de Otimização de Trajetórias Off-line e Controle Preditivo On-line para Robôs com Pernas

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Engenharia Mecânica, do Departamento de Engenharia Mecânica da PUC-Rio .

Orientador : Prof. Marco Antonio Meggiolaro
Co-orientador: Prof[a]. Vivian Suzano Medeiros

Rio de Janeiro
Junho de 2024

Pontifícia Universidade Católica
do Rio de Janeiro

Leonardo Garcia Moraes

Framework de Integração de Otimização de Trajetórias Off-line e Controle Preditivo On-line para Robôs com Pernas

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós–graduação em Engenharia Mecânica da PUC-Rio . Aprovada pela Comissão Examinadora abaixo:

Prof. Marco Antonio Meggiolaro
Orientador
Departamento de Engenharia Mecânica – PUC-Rio

Prof[a]. Vivian Suzano Medeiros
Coorientadora
Escola de Engenharia de São Carlos - EESC-USP

Prof. Marcelo Becker
Escola de Engenharia de São Carlos - EESC-USP

Prof. Florian Alain Yannick Pradelle
Departamento de Engenharia Mecânica - PUC-Rio

Rio de Janeiro, 06 de Junho de 2024

Leonardo Garcia Moraes

Graduado em Engenharia Mecânica pela Universidade Federal do Rio de Janeiro

Ficha Catalográfica

Garcia Moraes, Leonardo

Framework de Integração de Otimização de Trajetórias Off-line e Controle Preditivo On-line para Robôs com Pernas / Leonardo Garcia Moraes; orientador: Marco Antonio Meggiolaro; co-orientador: Vivian Suzano Medeiros. – 2024.

77 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Mecânica, 2024.

Inclui bibliografia

1. Engenharia Mecânica – Teses. 2. Robôs com Pernas. 3. Planejamento de Trajetória. 4. Otimização de Trajetória. 5. Controle de Locomoção. 6. Controle Preditivo Baseado em Modelo. I. Meggiolaro, Marco Antonio. II. Suzano Medeiros, Vivian. III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Mecânica. IV. Título.

CDD: 621

Agradecimentos

Eu agradeço imensamente aos meus pais Joseane e Alexandre por sempre estarem ao meu lado me incentivando a continuar os estudos e ajudando em tudo que precisei.

Ao meu orientador Marco Antonio Meggiolaro por me aceitar no laboratório e me ajudar nos momentos precisos. A minha coorientadora Vivian Medeiros pelo extremo apoio e direcionamento ao longo de toda a pesquisa.

Aos colegas do LabRob pela ambientação, companhia e suporte principalmente no começo da pesquisa ainda durante a pandemia.

Aos meus colegas da equipe de Soluções para Efeitos Especiais da Globo pela flexibilidade para conciliar o mestrado com o trabalho.

Aos meus amigos próximos por apoiarem e entenderem a minha ausência durante este período.

A todo o corpo docente do Departamento de Engenharia Mecânica da PUC-Rio pela estrutura e aulas de alto nível.

Ao CNPq e a PUC-Rio, pelos auxílios concedidos, sem os quais este trabalho não poderia ter sido realizado.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

Garcia Moraes, Leonardo; Meggiolaro, Marco Antonio; Suzano Medeiros, Vivian. **Framework de Integração de Otimização de Trajetórias Off-line e Controle Preditivo On-line para Robôs com Pernas**. Rio de Janeiro, 2024. 77p. Dissertação de Mestrado – Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

Na última década, os robôs móveis com pernas ganharam notoriedade por sua capacidade de se movimentar com segurança em terrenos acidentados e superar obstáculos, como declives e escadas, podendo ser utilizados em mais aplicações em comparação com os robôs móveis com rodas. Novos desenvolvimentos que melhorem a robustez do planejamento de trajetória e o controle dinâmico de robôs com pernas são cruciais para o avanço desse campo. O objetivo deste trabalho é desenvolver um framework baseado em C++ e ROS Noetic que integre otimização de trajetória off-line para robôs com pernas com Model Predictive Control (MPC) on-line, considerando o mapa de elevação do terreno. A otimização de trajetória é baseada na biblioteca de código aberto TOWR (Trajectory Optimization for Walking Robots), que emprega uma função contínua para representar o mapa do terreno. Para torná-la mais genérica, foi implementada uma interface que permite que mapas de elevação 2,5D sejam usados como representação do terreno. Além disso, as trajetórias geradas pelo TOWR são fornecidas como referências para um controlador MPC baseado na biblioteca de código aberto OCS2. As trajetórias otimizadas pelo MPC são então rastreadas por um Whole-Body Controller (WBC), que calcula os torques de atuação das juntas do robô. A estrutura é validada em simulações usando a dinâmica completa do robô, com diferentes tipos de terreno e sob perturbação externa.

Palavras-chave

Robôs com Pernas; Planejamento de Trajetória; Otimização de Trajetória; Controle de Locomoção; Controle Preditivo Baseado em Modelo.

Abstract

Garcia Moraes, Leonardo; Meggiolaro, Marco Antonio (Advisor); Suzano Medeiros, Vivian (Co-Advisor). **Integration Framework for Offline Trajectory Optimization and Online Model Predictive Control for Legged Robots**. Rio de Janeiro, 2024. 77p. Dissertação de Mestrado – Departamento de Engenharia Mecânica, Pontifícia Universidade Católica do Rio de Janeiro.

In the last decade, legged mobile robots have gained notoriety for their ability to move safely over rough terrain and overcome obstacles such as slopes and stairs, opening up new applications compared to wheeled mobile robots. New developments that improve the robustness of trajectory planning and dynamic control of legged robots are crucial for the advancement of this field. The aim of this work is to develop a framework based in C++ and ROS Noetic that integrates offline trajectory optimization for legged robots with online Model Predictive Control (MPC) while taking into account the elevation map of the terrain. The trajectory optimization is based on the open-source library TOWR (Trajectory Optimization for Walking Robots), which employs a continuous function to represent the map of the terrain. To make it more generic, an interface was implemented to allow 2.5D elevation maps to be used as terrain representation. Furthermore, the trajectories generated by TOWR are provided as references for a MPC implemented based on the open-source library OCS2. The trajectories optimized by the MPC are then tracked by a weighted Whole-Body Controller (WBC), which computes the actuation torques for the robot's joints. The framework is validated in simulations using the full dynamics of the robot, with different terrain types and under external disturbance.

Keywords

Legged Robots; Trajectory Planning; Trajectory Optimization; Locomotion Control; Model Predictive Control.

Sumário

Lista de figuras

Lista de tabelas

Lista de Abreviaturas

CoM – Centro de Massa

EoM – Equações de Movimento

FCD – Dinâmica Centroidal Completa

FRBD – Dinâmica Completa

GDL – Graus de Liberdade

GRF – Forças de Reação do Solo

IK – Cinemática Inversa

IP – Pendulo Invertido

MPC – Model Predictive Control

OCS2 – Optimal Control for Switched Systems

ODE – Open Dynamics Engine

ROS – Robot Operating System

SCR – Sistema de Referência de Coordenadas

SRBD – Dinâmica Centroidal de Corpo Único

TOWR – Trajectory Optimizer for Walking Robots

WBC – Whole Body Control

Success is not for those who never stumble,
but for those who rise after every fall and
keep moving forward

, .

1
Introdução

No passado, os robôs eram especialmente utilizados em linhas de produção, realizando tarefas repetitivas e perigosas de forma eficiente. Com o tempo, a robótica evoluiu desempenhando um papel cada vez mais significativo em nossa sociedade moderna, revolucionando setores como manufatura, medicina, exploração espacial e nosso cotidiano. Com avanços tecnológicos e científicos constantes, os robôs estão se tornando mais sofisticados e capazes de realizar uma ampla gama de tarefas complexas. Eles podem colaborar com seres humanos, automatizar processos industriais, explorar ambientes hostis e até mesmo desempenhar papéis sociais.

Em termos de classificação de locomoção, representada na Figura 1.1, a maior parcela de robôs terrestres comercializados ainda são os de base fixa, especializados em tarefas precisas e repetitivas, porém o desenvolvimento de robôs de base móvel segue em constante evolução. Inicialmente representados pelos robôs com roda, os robôs móveis trouxeram uma série de novas possibilidades e aplicações envolvendo principalmente tarefas de navegação, mapeamento e transporte.

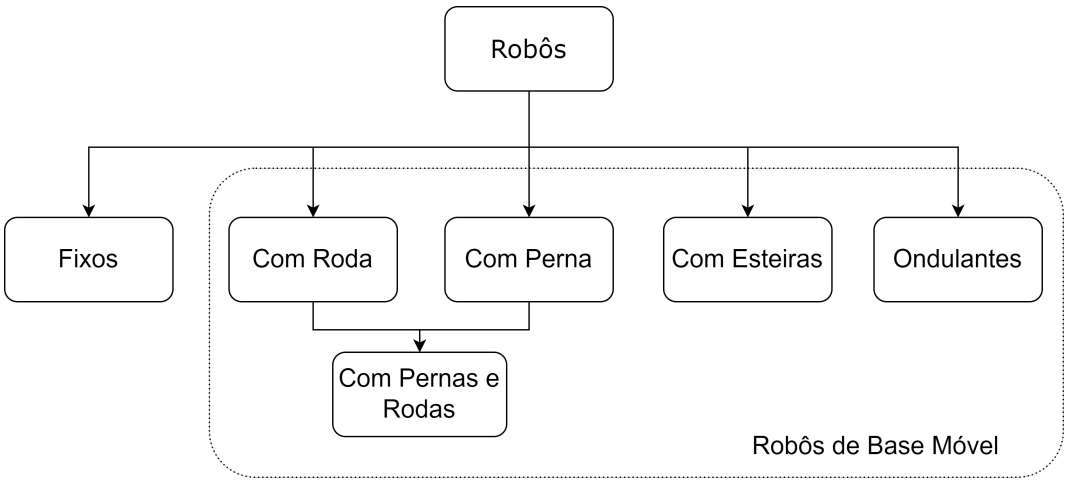


Figura 1.1: Classificação de robôs em relação ao mecanismo base de locomoção - Adaptado de (SICILIANO et al., 2010)

Nos últimos anos, buscando mais mobilidade, avanços notáveis na concepção e construção de robôs com pernas vem sendo apresentados pela industria e academia. A tecnologia de atuadores, sensores e algoritmos de controle tem sido aprimorada para permitir a locomoção estável e eficiente desses robôs em ambientes desafiadores e adaptação em terrenos variados como superfícies irregulares, inclinações e escadas. Hoje, unindo o melhor dos dois principais tipos

de robôs móveis, os robôs com pernas e rodas simultaneamente, vem ganhando espaço em frentes de pesquisa apresentando grande potencial e versatilidade.

Apesar da cinemática associada as pernas aumentar o nível de complexidade de atuação, controle e consumo energético em comparação com os robôs com roda, os robôs com perna elevam a tarefa de locomoção para um novo patamar de possibilidades, permitindo atuação em terrenos diversos e superação de novos obstáculos, incluindo rampas, degraus e vãos (Figura 1.2). Com isso, a motivação no estudo e desenvolvimento de robôs com pernas reside nas diversas novas abordagens de atuação na sociedade, agregando em áreas como:

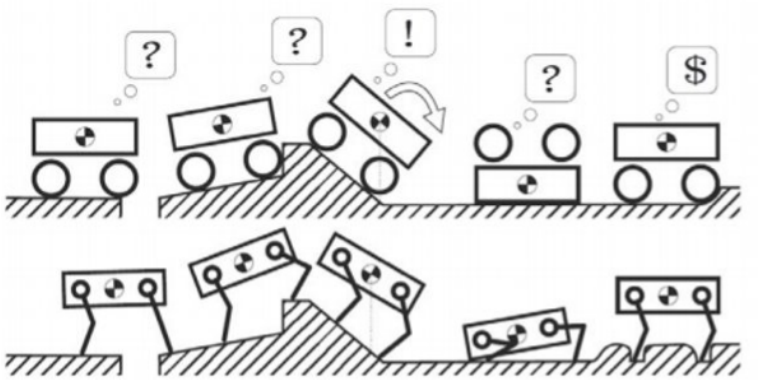


Figura 1.2: Comparação entre robôs com roda e robôs com pernas (HUTTER; SIEGWART; STASTNY, 2016)

- Exploração de ambientes desafiadores: Podem ser projetados para navegar por terrenos difíceis e complexos, como montanhas, selvas ou até mesmo outros planetas, contribuindo para a exploração de ambientes inexplorados e desafiadores.
- Aplicações de resgate e assistência: Potencial para serem usados em missões de resgate e assistência em áreas de difícil acesso ou após desastres naturais, contribuindo para o desenvolvimento de tecnologias que podem salvar vidas e ajudar comunidades em situações críticas.
- Avanços na medicina e reabilitação: Podem auxiliar na reabilitação de pessoas com deficiências motoras ou ajudar no desenvolvimento de próteses mais avançadas, contribuindo para o avanço dessas áreas e melhorando a qualidade de vida de muitas pessoas.

Atualmente, os robôs quadrupedes já possuem uma relevante posição tanto na academia quanto na indústria, tendo representantes de grandes empresas já comercializados em território mundial onde prometem aplicações e soluções diversas. Como exemplo tem-se o ANYmal (HUTTER et al., 2016), apresentado na Figura 1.3a, desenvolvido pela ANYbotics, empresa de

robótica especializada na industria de inspeção e que possui parceria com o laboratório de robôs com pernas da universidade ETH de Zurique, e o robô Spot (DYNAMICS, 2024) (Figura 1.3b) desenvolvido pela Boston Dynamics, empresa referência na área e famosa pelos seus robôs extremamente dinâmicos.

(a) ANYmal (b) Spot

Figura 1.3: Robôs comerciais de grandes empresas

1.1 Motivação

Devido a inerente complexidade da dinâmica de robôs com perna, planejar trajetórias de movimentação seguras e que respeitem condições cinemáticas e do ambiente não é uma tarefa simples, exigindo muitas vezes, abordagens de otimização. Com isso, o estudo de otimização de trajetórias é um importante tópico de estudo onde desenvolvimentos que agreguem versatilidade geram valor para a área.

Consequentemente, o rastreamento de trajetórias para robôs com pernas também é uma tarefa complexa envolvendo não só o movimento da base mas o movimento das pernas, o posicionamento das patas, condições de estabilidade e interferências externas. Com isso, abordagens que adicionem robustez ao controle de locomoção são importantes e necessárias para o avanço da robótica.

Portanto, essa maior versatilidade de locomoção vem atrelada a uma maior complexidade de problemas de planejamento e controle. Com mais graus de liberdade, definir trajetórias cartesianas para a base e pernas não é uma tarefa simples, especialmente se são incluídas as informações geométricas de terreno, a otimização de *gait* (cronograma que dita quais pernas estarão em contato com o solo e por quanto tempo) e outros parâmetros cinemáticos desejados. Existem hoje diversos frameworks que abordam o problema de planejamento de trajetórias, entretanto, estes frameworks não são comumente integrados com um módulo de controle capaz de rastrear estas trajetórias.

Logo, desenvolvimentos que integrem um planejamento de trajetórias ótimas com um sistema de controle robusto a perturbações externas, acrescentam para a evolução dessa área.

1.2 Revisão Bibliográfica

De forma geral, o estudo de robôs com pernas tem indícios que remontam a séculos atrás, sendo difícil afirmar seu início. Desenvolvimentos pontuais são observados em alguns registros ao longo do tempo, tendo um impulso nas décadas de 1980 e 1990 onde alguns projetos promissores começaram a ser apresentados. Dentre eles o robô Cheetah do Instituto de Tecnologia de Massachusetts (MIT) que explorou a locomoção dinâmica em quadrúpedes robóticos (SEOK et al., 2013) e a série ASIMO (HONDA, 2000) de robôs bípedes da Honda. A partir dos anos 2000, houve uma proliferação de estudos e projetos de robôs com pernas em instituições acadêmicas e laboratórios de pesquisa em todo o mundo. Os avanços na tecnologia de sensores, atuadores e controle permitiram o desenvolvimento de robôs cada vez mais ágeis e adaptáveis, como o Atlas (DYNAMICS, 2016), o Mini-Cheetah (BOSWORTH; KIM; HOGAN, 2015), o HyQ (SEMINI et al., 2011), o ANYmal (HUTTER et al., 2016) e o Spot (DYNAMICS, 2024).

Atualmente, as pesquisas em robôs com pernas estão divididas em diversas subáreas, como planejamento de trajetória, métodos de controle baseados em modelos, integração com aquisição de sensores, otimização de *gait*, movimentos altamente dinâmicos e até controle de aprendizagem por reforço e outras técnicas envolvendo inteligência artificial, indicando o potencial e as possibilidades de desenvolvimentos dessa área (KOTTEGE; SENTIS; KANOULAS, 2022).

Inicialmente, o planejamento de trajetória era normalmente simplificado para considerar interpolações simples entre pontos iniciais e finais e abordagens puramente reativas sem planejamento. Com os avanços no poder de processamento, o planejamento de trajetória começou a ser tratado como um problema de otimização para uma diversidade de abordagens. Como em Wang et al. (2020) que otimiza uma trajetória minimizando o tempo de percurso para um gait estático e posições de pata pré-definidas, Norby e Johnson (2020) asseguram um planejador rápido e eficiente em grandes horizontes de movimento. Já Sun, Wang e An (2020) propõem um replanejador em tempo real para uma locomoção cega baseado apenas em estimação de parâmetros. Esses e diversos outros trabalhos reforçam o poder da otimização de trajetórias em expandir os recursos de movimento e permitir a geração de novos compor-

tamentos. Como consequência, várias ferramentas e bibliotecas aplicáveis ao problema de locomoção foram desenvolvidas, cada uma com suas particularidades, como TOWR (WINKLER et al., 2018), Drake (TEDRAKE; TEAM, 2019), FROST (HEREID; AMES, 2017), QuadSDK (NORBY et al., 2022), KinoDynOpt (UNIVERSITY, 2020) e TrajOpt (SCHULMAN; LAB, 2013). A Tabela 1.1 representa um comparativo entre estes frameworks. Como pode-se observar, a biblioteca QuadSDK é a única das bibliotecas analisadas que possui integração com um módulo de controle, porém não possui a ferramenta de otimização de gait, enquanto a biblioteca TOWR é a única que possui otimização de gait para robôs quadrúpedes mas não possui integração com um módulo de controle.

	Integra Controle	Otimização de Gait	Quadrúpedes	Código Aberto	Integra Ros	Considera Mapa	Github Stars	Github Forks
TOWR	✗	✓	✓	✓	✓	✓	854	220
DRAKE	✗	✗	✗	✓	✗	✗	3.1k	1.2k
FROST	✗	✓	✗	✓	✗	✗	150	63
QuadSDK	✓	✗	✓	✓	✓	✓	654	128
KinoDynOpt	✗	✗	✓	✓	✗	✗	35	4
TrajOpt	✗	✗	✗	✓	✗	✗	360	158

Tabela 1.1: Tabela comparativa entre frameworks de planejamento de trajetórias com módulo para robôs com pernas baseados em otimização. Foi considerado o estado atual das *branchs* principais de cada biblioteca

Com os avanços na tecnologia de sensores e câmeras, a integração de um sistema de percepção e mapeamento com tarefas de planejamento se tornou cada vez mais popular, mostrando resultados promissores. Fahmi et al. (2019) mostram um inovador WBC passivo utilizando um mapeamento online de terrenos desafiadores. O sistema é validado experimentalmente com um robô hidráulico de grande porte, apresentando bons resultados em diversos cenários. Em Mastalli et al. (2020), os dados da câmera são convertidos em um mapa de custo, utilizado como entrada em um planejador de trajetórias que combina a posição das patas, o movimento horizontal do corpo e as informações de terreno. O planejador é posteriormente integrado com um WBC resultando em movimentos estáveis e bem precisos. Zhang et al. (2021) convertem também os dados dos sensores em um mapa de custo do terreno em tempo real, que é utilizado para realizar o planejamento de trajetórias em passagens estreitas, apresentando bons resultados em testes experimentais com um robô real. Gaertner et al. (2021) propõem um MPC que evita colisões enquanto realiza a navegação em terrenos estáticos e dinâmicos utilizando sensores exteroceptivos. Os resultados experimentais demonstram a capacidade do método de lidar com ambientes complexos, incluindo obstáculos suspensos e agentes dinâmicos.

Um tópico que vem ganhando espaço nos últimos anos é a otimização dos padrões de movimento das pernas durante o planejamento, denominado otimização de *gait*, abordado em Zhang et al. (2020), que otimiza o *gait* com base na conservação de energia; em Winkler et al. (2018), que usa parametrização baseada em fases para a trajetória das pernas e otimiza as durações das fases como variáveis contínuas; e em Ma, Hamed e Ames (2019), que executa a otimização de *gait* e da trajetória para robôs bípedes usando a biblioteca de otimização FROST (HEREID; AMES, 2017).

Ao longo das décadas, o desenvolvimento de métodos de controle e rastreamento de trajetória para robôs com pernas também seguiu um caminho evolutivo, ilustrado na Figura 1.4. Inicialmente, na década de 1980, a modelagem física dos robôs era baseada em modelos conceituais de movimento, como o Pêndulo Invertido (SHARBAFI; SEYFARTH, 2010). Ao longo da década de 1990, houve uma transição para a locomoção com pernas, em que os conceitos de estabilidade dinâmica começaram a ser considerados, embora sem fundamentos teóricos substanciais em relação a dinâmica completa do robô (VUKOBRATOVIĆ; BOROVAC, 2004). A partir dos anos 2000, houve uma mudança significativa para métodos de controle baseados na formulação dinâmica completa dos robôs, levando em conta a estrutura multi-corpos das pernas e os efeitos da aceleração na base do robô (FEATHERSTONE, 2014). Dai, Valenzuela e Tedrake (2014) exemplificam a utilização da modelagem de dinâmica centroidal no planejamento de movimentos para o robô bípede Atlas e Bellicoso et al. (2017) asseguram gaits dinâmicos para o robô ANYmal utilizando um controlador que considera a dinâmica completa do robô. Nos dias de hoje, com o avanço do planejamento e a utilização de robôs em diversos terrenos, a necessidade de métodos de controle de alto nível que sejam robustos a distúrbios (e.g. MPC) está cada vez mais presente e foi abordada por vários autores. Neunert et al. (2017) aplica o MPC utilizando a dinâmica completa do sistema aplicado no robô HyQ atingindo até 190 Hz em um horizonte de predição de 1 segundo, superando os outros modelos existentes na época, porém começa a apresentar desvios em *gaits* mais dinâmicos. Em Ding et al. (2021), é utilizado um MPC com representação livre para movimentos dinâmicos em robôs quadrúpedes onde a orientação é definida por matrizes de rotações invés dos convencionais ângulos de Euler, reduzindo alguns problemas de implementação. Os resultados experimentais asseguram alguns movimentos dinâmicos controlados, mas não leva em consideração informações do ambiente. Grandia et al. (2021) apresentam um MPC associado com funções de barreiras de controle para atingir movimentos dinâmicos com segurança para o robô ANYmal, considerando informações de terreno conhecidas. O sistema

consegue realizar movimentos seguros das patas e melhores avaliações de estabilidade dinâmica, porém não utiliza sensores para mapeamento do terreno limitando a escalabilidade de testes em cenários diversos.

O estudo de robôs com pernas também se beneficia significativamente das técnicas de aprendizado de máquina e inteligência artificial, resultando em novos comportamentos de locomoção e melhorias na interação homem-robô (MU; SHAO; ZHANG, 2021), contudo, a etapa de aprendizado ainda exige hoje grande esforço mecânico e muitas horas de treinamento.

Apesar de muitas pesquisas terem sido realizadas no campo de planejamento e controle de movimento para robôs com pernas, ainda é muito desafiador criar uma estrutura de otimização de trajetória que inclua informações sobre o terreno, restrições dinâmicas e de atuação, que não apenas calcule o movimento da base e das pernas, mas que também possa otimizar o cronograma de contato das patas e ainda seja executada em tempo real. Para tarefas de planejamento de movimento tão complexas, o custo computacional geralmente é muito alto (WINKLER et al., 2018). Uma alternativa interessante é combinar uma otimização de trajetória offline com um controlador de rastreamento de trajetória online, empregando o controle preditivo baseado em modelo (MPC). Isso permitiria a adaptação online das trajetórias a distúrbios não previstos e incompatibilidades de modelos (MEDEIROS et al., 2020). Uma abordagem semelhante foi apresentada em (BJELONIC et al., 2022) para robôs com pernas e rodas, criando uma biblioteca de movimentos de trajetórias previamente otimizadas, que poderiam ser combinadas e fornecidas como entrada para um MPC online para rastreamento no robô real, apresentando resultados impressionantes.

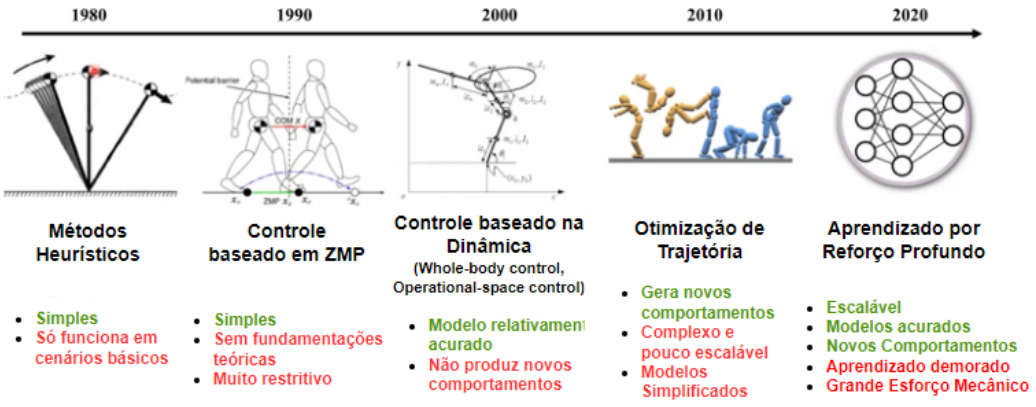


Figura 1.4: Marcos importantes na evolução do planejamento e controle de robôs com pernas - Adaptado de (HWANGBO, 2021)

1.3 Objetivos e Contribuições

O presente trabalho pretende atuar na facilitação e expansão das possibilidades de planejamento de trajetórias de robôs com pernas em terrenos acidentados, bem como validar dinamicamente as trajetórias geradas com a utilização de técnicas de controle preditivo robusto a perturbações externas.

De forma concreta, propõem-se o desenvolvimento de um framework em C++ baseado em ROS Noetic[1], cujo diagrama está apresentado na Figura 1.5, que integre uma biblioteca de otimização de trajetória offline para robôs com pernas com uma biblioteca de controle baseada em *Model Predictive Control* (MPC) online, considerando o mapa de elevação do terreno.

A otimização de trajetória é baseada na biblioteca de código aberto TOWR (Trajectory Optimization for Walking Robots) (WINKLER et al., 2018), que emprega uma função contínua para representar o mapa do terreno. Para torná-la mais genérica, foi implementada uma interface baseada na biblioteca de mapeamento `grid_map` (FANKHAUSER; HUTTER, 2016a) que permite que mapas de elevação 2,5D sejam usados como representação do terreno.

Além disso, as trajetórias geradas pelo TOWR foram fornecidas como referências para um controlador MPC baseado na biblioteca OCS2 (FARSHIDIAN et al., 2017a). As trajetórias otimizadas pelo MPC são então rastreadas por um Whole-Body Controller (WBC), que calcula os torques de atuação das juntas do robô considerando restrições de limites máximos dos atuadores e o cone de atrito para as forças de contato. Para a implementação do WBC, é utilizada a biblioteca de código aberto `legged_control` (LIAO, 2022), que também fornece uma interface entre o OCS2, o WBC e o simulador Gazebo (KOENIG; HOWARD, 2004).

Com estas implementações pretende-se obter uma estrutura unificada e robusta de planejamento e controle, facilitando e potencializando futuros estudos e trabalhos dentro do segmento de robôs com pernas. As performances de movimento são avaliadas em simulações física utilizando do software Gazebo, usando a dinâmica completa do robô, para diferentes tipos de terreno e sob perturbação externa. Os testes foram realizados em uma máquina virtual de Linux com Ubuntu 20.04, 12 Gb de memória RAM e 6 núcleos de processamento.

Em suma, as contribuições desse trabalho podem ser descritas como:

[1]Ambiente amplamente utilizado para desenvolvimentos de projetos de robótica com estrutura modular e sistema de comunicação integrado.

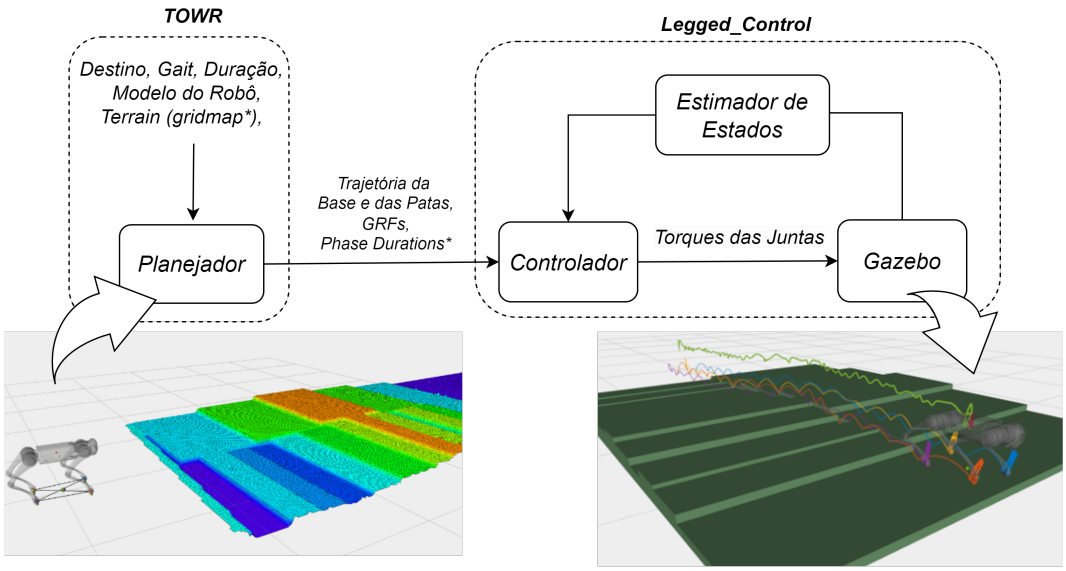


Figura 1.5: Diagrama representativo do framework proposto

- Implementação de uma interface simplificada baseada em interpolação linear que permita a utilização de um mapa de elevação 2,5D como entrada para o TOWR;
- A inclusão de restrições adicionais na formulação do TOWR para melhorar a robustez das trajetórias geradas, evitando colisão das patas com os obstáculos no terreno, baseadas na formulação discutida em (MEDEIROS, 2020).
- Implementação da biblioteca de integração `Towr2LeggedControl` que permite a validação dinâmica das trajetórias geradas pelo TOWR, fornecendo-as como referência para o sistema de controle da biblioteca `legged_control`, que integra um MPC com um WBC e simulações em Gazebo.
- Testes em simulação do framework proposto com o robô quadrúpede Go1 em diferentes terrenos, com e sem otimização de *gait*, em cenários onde robô está sujeito a perturbações externas constantes.

Um artigo contendo os principais resultados desenvolvidos ao longo desta pesquisa foi submetido e aceito para ser apresentado no Congresso Brasileiro de Automática (CBA 2024)[2]:

Garcia, L. M., Lopes, Medeiros, V. S., Meggiolaro, M. A. **Integration Framework for Offline Trajectory Optimization and Online Model Predictive Control for Legged Robots**. Em: XXV Congresso Brasileiro de Automática (CBA) (2024).

[2]<https://cba2024.vidaestudantil.com/>

1.4 Estrutura da Dissertação

No Capítulo 2, são apresentados alguns conceitos teóricos e bibliotecas importantes para o desenvolvimento da pesquisa. No Capítulo 3, é apresentado o desenvolvimento da biblioteca de integração `Grid2Towr` entre a biblioteca `Gridmap` e a biblioteca do TOWR. No Capítulo 4, é discutido o desenvolvimento da biblioteca de integração `Towr2LeggedControl` entre a saída do TOWR e a entrada no `legged_control`. No Capítulo 5, são apresentados os resultados e análises da framework em diferentes terrenos. Por último, no Capítulo 6, conclui-se a pesquisa destacando os principais pontos e possibilidades futuras de desenvolvimento.

2
Fundamentações e Conceitos

Neste capítulo, destacam-se alguns conceitos fundamentais dentro do estudo de planejamento e controle para robôs com pernas, bem como a descrição de algumas bibliotecas utilizadas neste trabalho. Além disso, algumas premissas são estabelecidas para o desenvolvimento da pesquisa.

2.1
Plataforma Experimental - Robô Go1

Atualmente, existem diversos robôs quadrúpedes comercialmente disponíveis sendo utilizados como base para pesquisa e desenvolvimento dentro de empresas e universidades. Um robô quadrúpede que vem ganhando espaço em grupos de pesquisa ao redor do mundo é o Go1, desenvolvido pela empresa chinesa Unitree (UNITREE, 2024), apresentado na Figura 2.1), destacando-se pelo seu alto custo-benefício, por ser um robô acessível, compacto, leve e capaz de realizar movimentos altamente dinâmicos em comparação com outros robôs de alto desempenho. A Tabela 2.1 destaca suas principais características. O Go1 será o robô utilizado como base para os testes da framework desenvolvida neste trabalho.

Figura 2.1: Robô Unitree Go1

	Preço	Peso	Dimensões	Vel. Máx.	Payload
Unitree Go1	US$ 2.700	12 kg	0.6 x 0.2 x 0.3 m	17 km/h	5 kg

Tabela 2.1: Principais Características do Go1

2.2
Modelagem Cinemática e Dinâmica

Em comparação com os manipuladores robóticos de base fixa, os robôs com pernas possuem o que se chama de base flutuante (*floating base*) onde a base não é diretamente atuada, sendo seu movimento consequência da reação das forças de contato entre as patas e o solo (*ground reaction forces*, GRF) (HUTTER et al., 2017).

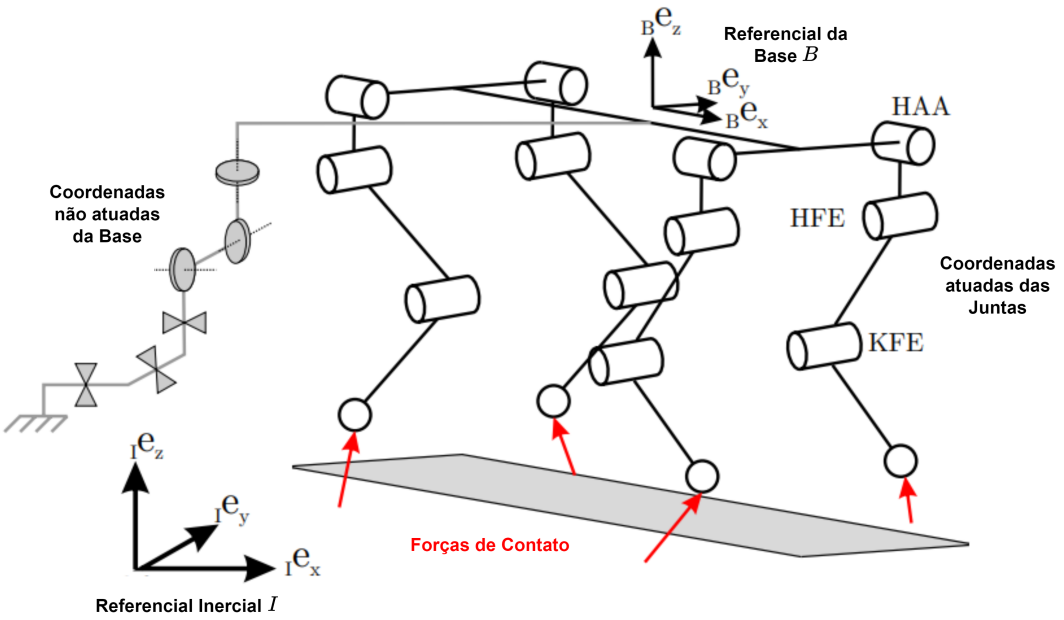


Figura 2.2: Modelagem Cinemática de um Robô Quadrúpede - Adaptado de (HUTTER et al., 2017)

Modelado na Figura 2.2, robôs de base flutuante podem ser totalmente descritos pelas n_b coordenadas de base não atuadas $\mathbf{q}_b$ e e n_j coordenadas de juntas atuadas $\mathbf{q}_j$

$$\mathbf{q} = \begin{pmatrix} \mathbf{q}_b \\ \mathbf{q}_j \end{pmatrix} \tag{2-1}$$

A base não atuada é livre em translação e rotação.

$$\mathbf{q}_b = \begin{pmatrix} \mathbf{q}_{bP} \\ \mathbf{q}_{bR} \end{pmatrix} \in \mathbb{R}^3 \times SO(3)^1, \tag{2-2}$$

onde a posição $\mathbf{q}_{bP}$ e a rotação $\mathbf{q}_{bR}$ podem ser parametrizadas usando os ângulos de Euler. Portanto, a dimensão do vetor de coordenadas generalizadas de um sistema de base flutuante, dado por $n_b + n_j$, depende da parametrização da rotação, sendo que o número mínimo de coordenadas generalizadas para a base é 6 (HUTTER et al., 2017).

[1]O grupo euclidiano ortogonal especial que representa rotações em três dimensões.

O vetor de velocidades generalizadas $\dot{\boldsymbol{q}}$ do sistema, parametrizando as rotações em ângulos de Euler é dado por

$$\dot{\boldsymbol{q}} = \begin{pmatrix} {}^{I}\boldsymbol{v}_{B} \\ {}^{B}\boldsymbol{\omega}_{IB} \\ \dot{\boldsymbol{q}}_{j} \end{pmatrix} \in \mathbb{R}^{n_v}, \tag{2-3}$$

onde ${}^{I}\boldsymbol{v}_{B} \in \mathbb{R}^3$ é o vetor de velocidade linear da base com relação ao referencial inercial I, ${}^{B}\boldsymbol{\omega}_{IB} \in \mathbb{R}^3$ é o vetor de velocidade angular do referencial I para o referencial B com relação ao referencial B e $\dot{\boldsymbol{q}}_j$ é o vetor de velocidade das juntas. A dimensão do vetor de coordenadas generalizadas é $n_v = 6 + n_j$.

As pernas de robôs quadrúpedes clássicos, como é o caso do Go1, possuem 3 graus de liberdade (GDL). Portanto para definir completamente um robô quadrúpede no espaço em relação a um referencial inercial precisa-se de um total de 18 GDL, 12 das pernas e 6 da base.

Uma vez definidas as posições generalizadas e os vetores de velocidade, o próximo passo é encontrar as EoM baseadas na dinâmica do sistema. Devido a alta complexidade envolvendo o sistema multi-corpos das pernas, pode-se encontrar algumas abordagens diferentes na literatura para representar a dinâmica de robôs com pernas. Em casos onde não é necessário uma extrema precisão na representação, alternativas simplificadas são utilizadas reduzindo o número de cálculos matemáticos e processamento computacional necessários (WINKLER, 2018a).

`Pinocchio` é uma biblioteca de código aberto amplamente utilizada para cálculos eficientes envolvendo a cinemática e dinâmica de corpos rígidos sendo uma boa opção para facilitar as operações e definições de modelagem de multi-corpos. Por conta disso, é frequentemente utilizada para realizar os cálculos de cinemática e dinâmica diretas e inversas de robôs com pernas como foi o caso desta pesquisa (CARPENTIER et al., 2019).

2.2.1 Dinâmica Completa (FRBD)

Uma vez definidas as posições e velocidades generalizadas, através do método de *Newton-Euler*, que aplica os princípios de conservação linear e angular em todos os elos do robô (SICILIANO et al., 2010), pode-se obter a equação de movimento que considera a dinâmica completa do sistema

$$\mathbf{M}(\mathbf{q})\ddot{\boldsymbol{q}} + \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q}) = \mathbf{S}^{\mathbf{T}}\tau + \mathbf{J}_{\mathbf{c}}^{\mathbf{T}}\mathbf{F}_{\mathbf{c}} \tag{2-4}$$

onde $\ddot{\boldsymbol{q}}$ é o vetor de acelerações generalizadas dadas as rotações expressas em ângulos de Euler, $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{n_v \times n_v}$ é a matriz de massa generalizada,

$\mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n_v}$ é o vetor de Coriolis e os termos centrífugos e $\mathbf{g}(\mathbf{q})$ é o vetor de termos gravitacionais. $\boldsymbol{\tau} \in \mathbb{R}^{n_j}$ é o vetor de torque generalizados atuando na direção das coordenadas generalizadas e $\mathbf{S} = [\mathbf{0}_{n_j \times 6} \quad \mathbf{1}_{n_j \times n_j}]$ é matriz de seleção que seleciona quais juntas são atuadas. Considerando n_c como o numero de end-effectors em contrato, a matriz Jacobiana $\mathbf{J}_c = [\mathbf{J}_{C_1}^T \ ... \ \mathbf{J}_{C_{n_c}}^T] \in \mathbb{R}^{3n_c \times n_v}$ mapeia as forças de contato $\mathbf{F}_c = [\mathbf{f}_1^T \ ... \ \mathbf{f}_{n_c}^T]^T \in \mathbb{R}^{3n_c}$ do espaço Cartesiano para o subespaço de coordenadas generalizadas.

Nesta pesquisa, este modelo é utilizado para representar a dinâmica do robô no Whole-Body Controller (WBC) implementado na biblioteca `legged_control` (Seção 2.9).

2.2.2 Dinâmica Centroidal de Corpo Único (SRBD)

Uma alternativa comum na representação da dinâmica de robôs com pernas, principalmente em quadrúpedes de pernas com massa pequena em relação a massa da base, é o modelo de Dinâmica Centroidal de Corpo Único (*Single Rigid Body Dynamics*). Este modelo considera a massa e a inércia fixas no centro de massa do robô, assumindo que a inércia das pernas é desprezível.

É possível obter as EoM utilizando as equações de Newton-Euler para um corpo rígido

$$\begin{aligned} m\ddot{\boldsymbol{r}}(t) &= \sum_{i=1}^{n_i} \boldsymbol{f_i}(t) - m\boldsymbol{g} \\ \boldsymbol{I}\dot{\boldsymbol{\omega}}(t) + \boldsymbol{\omega}(t) \times \boldsymbol{I}\boldsymbol{\omega}(t) &= \sum_{i=1}^{n_i} \boldsymbol{f_i}(t) \times (\boldsymbol{r}(t) - \boldsymbol{p_i}(t)) \end{aligned} \tag{2-5}$$

Onde $\boldsymbol{\omega}(t) \in \mathbb{R}^3$ é a velocidade angular da base, m é a massa do robô, $\boldsymbol{g} \in \mathbb{R}^3$ o vetor de gravidade, $\boldsymbol{I} \in \mathbb{R}^{3\times3}$ é a matriz de inércia do robô na posição padrão de *stance*, n_i é o número de pernas e $\boldsymbol{f_i}$ e $\boldsymbol{p_i}$ representam as forças e posições em cada uma delas e o vetor $\boldsymbol{r}(t)$ representa a posição do CoM.

Nesta pesquisa, este modelo é utilizado para representar a dinâmica do robô no planejamento de trajetórias do TOWR (Seção 2.6).

2.2.3 Dinâmica Centroidal Completa (FCD)

O modelo de Dinâmica Centroidal Completa (*Full Centroidal Dynamics*) é similar ao modelo SRBD, porém este leva em consideração a inércia dos elos das pernas. Sendo assim, é um pouco mais fidedigno a dinâmica real do robô que o SRBD e menos complexo computacionalmente que o FRBD.

$$\boldsymbol{A}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \dot{\boldsymbol{A}}(\boldsymbol{q})\dot{\boldsymbol{q}} = \begin{bmatrix} m\boldsymbol{g} + \sum_{i=1}^{n_i} \boldsymbol{f_i} \\ \sum_{i=1}^{n_i} \boldsymbol{f_i} \times (\boldsymbol{r}((q)) - \boldsymbol{p_i}(\boldsymbol{q}) \end{bmatrix} \tag{2-6}$$

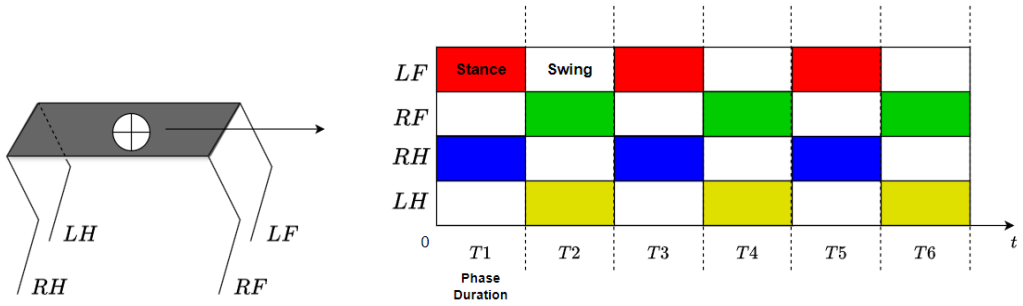
Onde $\boldsymbol{A} \in \mathbb{R}^{6\times(6+n)}$ mapeia as velocidades e as quantidades de movimento de cada elo em um referencial comum expresso no CoM, $\boldsymbol{g} \in \mathbb{R}^3$ o vetor de gravidade, n_i é o número de pernas e $\boldsymbol{f_i}$ e $\boldsymbol{p_i}$ representam as forças e posições em cada uma delas. O lado esquerdo da equação representa a variação na quantidade de movimento.

Nesta pesquisa, este modelo é utilizado para representar a dinâmica do robô no módulo de controle NMPC (Seção 2.8) da biblioteca OCS2 (Seção 2.9).

2.3 Gait e Estabilidade

No contexto de robôs com pernas, um *gait* refere-se a um padrão de movimento utilizado para a locomoção do robô. É uma sequência de passos e movimentos das pernas que define como o robô se desloca no ambiente. O termo está relacionado à maneira como diferentes pernas de um robô são coordenadas e sincronizadas durante a caminhada. Cada passo individual de uma perna é parte de um padrão maior de movimentos que permite ao robô se mover de maneira estável e eficiente. Dentre os principais gaits utilizados tem-se por exemplo a caminhada, o trote e o galope.

Diversos termos e parâmetros são utilizados para ajudar no processo de caracterização de um gait, destacando-se alguns dos principais tem-se: *phase*, referente as etapas de transição de cada pata, alternando entre *swing* e *stance*, onde o *swing* refere-se à etapa de movimento aéreo da pata enquanto o *stance* refere-se a etapa estática da pata; *phase duration*, que é o tempo de duração de um *phase*; *contact state*, que refere-se ao estado de contato da pata em um determinado momento, geralmente definido como uma variável booleana, ou seja, `true` no *stance* e `false` no *swing*. Os termos *lift-off* e *touch-down* referem-se ao momento no qual a pata inicia e termina o *swing* respectivamente. Com esses parâmetros é possível representar o gait em forma de diagrama de phases, ajudando na visualização do movimento, como mostra a Figura 2.3b. Outra notação importante ao se referir às patas é a convenção ilustrada na Figura 2.3a. Em relação ao sentido do movimento, LF é a pata frontal esquerda, RF é a pata frontal direita, LH é a pata posterior esquerda e RH é a posterior direta.



(a) Legenda das pernas

(b) Diagrama de phases. As cores sólidas representam o momento que a pata está em *stance* e as cores brancas, o momento que a pata está em *swing*

Figura 2.3: Representação de um gait de trote.

Este padrão de movimento de passos possui uma relação direta com a estabilidade do robô. Enquanto no mínimo 3 pernas estiverem em contato com o chão, o centro de massa do robô pode ser posicionado de forma que a sua projeção se encontre sempre dentro do polígono formado pelos pontos em contato com o chão, garantindo que o robô esteja estaticamente estável a todo momento, como mostra a Figura 2.4a. No caso de duas ou menos patas em contato com o chão, não é possível definir mais um polígono base (Figura 2.4b), sendo necessário movimentos mais dinâmicos e métricas de estabilidade mais complexas para garantir a estabilidade do robô, como por exemplo o *Zero Moment Point* (ZMP) (VUKOBRATOVIĆ; BOROVAC, 2004).

(a) Gait estático

(b) Gait dinâmico

Figura 2.4: Relação do tipo de gait com a estabilidade do robô - Adaptado de (HUTTER et al., 2017)

Ao longo do tempo, uma abordagem muito comum foi definir previamente o gait e posteriormente realizar o planejamento de trajetórias. Ao escolher um gait fixo, define-se previamente os *phase duration* de cada pata, o que para planejamentos complexos, limita o tempo que cada pata tem para completar

uma *phase* restringindo consideravelmente as possibilidades de movimento. Por conta disso, uma linha de pesquisa que agrega muito valor a locomoção de robôs quadrúpedes é a otimização de gait, que inclui os *phase durations* como resultado de um processo de otimização, expandindo as possibilidades de movimento em terrenos adversos, como apresentado em (WINKLER et al., 2018).

2.4 Robotic Operating System (ROS)

O controle de sistemas robóticos envolve uma gama de tarefas, integrações com sensores, comando de atuadores, tomadas de decisões, planejamento, dentre diversas outras. O Robot Operating System (ROS) surgiu como um framework de código aberto robusto e versátil para desenvolvimentos em robótica. Sua arquitetura modular, ferramentas abrangentes e grande comunidade o tornam uma forte ferramenta para pesquisadores e profissionais que buscam construir robôs autônomos e inteligentes (QUIGLEY et al., 2009).

O uso de ROS permite a execução de diferentes processos simultaneamente em uma estrutura de nós que se comunicam entre si através de um protocolo TCP/IP. Cada nó pode receber, postar e multiplexar dados de sensores, controle, planejamento, atuadores e diversas outras mensagens.

As principais bibliotecas utilizadas nesse trabalho possuem integração nativa com ROS e todo o desenvolvimento aqui apresentado foi realizado dentro de um ambiente ROS Noetic[2]. Com isso, facilita-se a integração e comunicação entre diferentes bibliotecas, bem como possibilidades de futuros trabalhos dentro da comunidade acadêmica (QUIGLEY et al., 2009).

Neste contexto, alguns termos relevantes para esta dissertação são:

- *rostopic*: Espécie de canal de comunicação que facilita a interação entre diferentes programas (nós) por meio de mensagens (*rosmsgs*);
- *rosmsg*: Mensagens de diferentes formatos e conteúdos que transitam geralmente entre tópicos levando dados.
- *rosbag*: Formato de arquivo utilizado para armazenar dados de sensores e mensagens publicadas em tópicos (*rostopics*);
- *RViz*: Ferramenta para visualização 3D que permite observar e interagir com dados de sensores e robôs em tempo real, fornecendo uma interface gráfica intuitiva para monitorar e depurar sistemas robóticos. (KAM et al., 2015).

[2]Versão de ROS desenvolvida para Ubuntu 20.04.

2.5 Planejamento e Otimização de Trajetórias

O planejamento de trajetórias é uma etapa crucial da robótica móvel, onde um robô é designado para se mover de um local para outro de maneira eficiente e segura. O objetivo do planejamento de trajetórias é determinar a sequência de movimentos que o robô deve seguir para alcançar um objetivo específico, levando em consideração os obstáculos, restrições de movimento e outros fatores relevantes, resultando principalmente em pontos cartesianos (WINKLER, 2018a).

O planejamento de trajetórias em robôs com pernas apresenta desafios adicionais em comparação com robôs com rodas ou com sistemas de locomoção mais simples, sendo necessário coordenar o movimento das pernas respeitando os limites cinemáticos para se locomoverem, tendo como objetivo encontrar principalmente os pontos cartesianos e orientações de referência para a base e para todas as patas, como ilustrado na Figura 2.5.

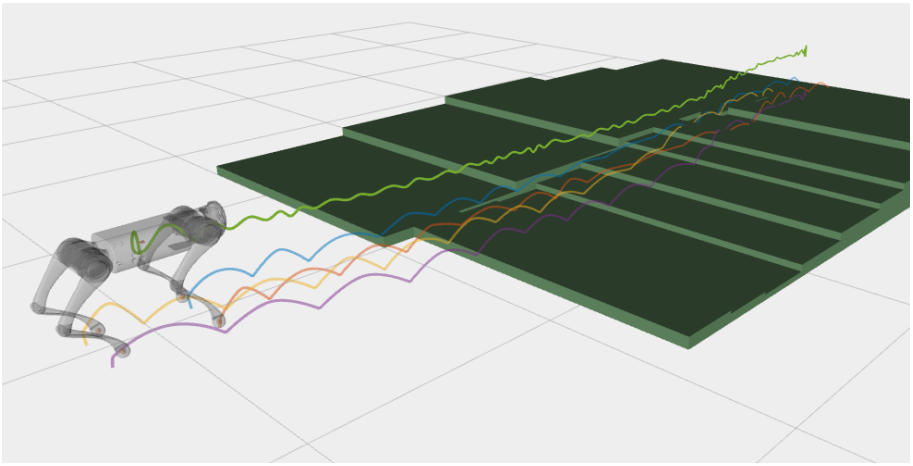


Figura 2.5: Planejamento de trajetória para robôs com pernas

Inicialmente, para a movimentação de robôs com pernas, o planejamento de trajetórias era normalmente simplificado para considerar interpolações simples entre pontos iniciais e finais ou abordagens puramente reativas sem planejamento eram utilizadas. Entretanto, para atuar em cenários mais complexos de locomoção autônoma, faz-se necessário abordagens mais sofisticadas.

Devido a complexidade do movimento e a necessidade de alcançar objetivos específicos (e.g. posição final e velocidade média) de forma eficiente atendendo diversas restrições (e.g. limites cinemáticos, modelo dinâmico, limites de juntas e atuadores, condições de atrito) e com o avanço na tecnologia de sensores e processamento computacional, o planejamento de trajetórias em robôs com pernas é frequentemente implementado como um problema de otimização, sendo abordado pelas mais diversas formulações existentes na literatura.

De maneira simplificada, o problema de otimização genérico tem a seguinte formulação

$$\begin{aligned} \min_{w} \quad & a(\boldsymbol{w}) \\ \text{subject to} \quad & \boldsymbol{b}(\boldsymbol{w}) = \mathbf{0}, \\ & \boldsymbol{c}(\boldsymbol{w}) \geq \mathbf{0}, \end{aligned} \tag{2-7}$$

onde o objetivo é encontrar o conjunto de variáveis $\boldsymbol{w}$ que minimizam a função de custo a enquanto satisfaz as igualdades $\boldsymbol{b} = \mathbf{0}$ e as desigualdades $\boldsymbol{c} \geq \mathbf{0}$. No caso de um problema de otimização de trajetórias para robôs com pernas, uma formulação comum é buscar as posições da base, das juntas e as forças de reação do solo que minimizem uma determinada função de custo, que pode envolver gasto energético, tempo de trajetória, suavidade de movimento ou estabilidade, incluindo restrições como limites cinemáticos e dinâmicos de juntas, força zero nas patas em *swing*, atrito com o terreno, estado inicial e final, modelo dinâmico do robô, dentre outras.

2.6 Biblioteca TOWR

A biblioteca TOWR (Trajectory Optimizer for Walking Robots) é uma biblioteca de código aberto desenvolvida para planejamento de trajetórias otimizadas para robôs com pernas (WINKLER, 2018b). Ela fornece uma estrutura flexível e modular para a geração de trajetórias eficientes e estáveis para robôs com pernas, como robôs quadrúpedes e bípedes, sendo possível adicionar um robô personalizado com poucas modificações.

O TOWR foi desenvolvido com o objetivo de simplificar e acelerar o processo de planejamento de trajetórias para robôs com pernas, oferecendo uma interface fácil de usar, permitindo aos usuários especificar as restrições e objetivos do sistema, como a posição final desejada, as limitações de velocidade e torque, restrições de contato e outros requisitos específicos do robô. Utiliza-se técnicas de otimização para encontrar trajetórias ideais, levando em consideração as características físicas e as limitações do robô, bem como a dinâmica do sistema. Ela utiliza uma formulação de otimização convexa, que permite a geração eficiente de trajetórias suaves e estáveis.

Uma importante ferramenta que o TOWR possui é a opção otimização de gait. Com isso os *phase durations* são incluídos como variáveis da otimização, resultando em uma maior possibilidade de movimentos gerados (WINKLER et al., 2018). A formulação completa da otimização realizada pelo TOWR está representada na Figura 2.6. O solver busca encontrar as posições lineares $\boldsymbol{r}(t)$ e angulares $\boldsymbol{\theta}(t)$ da base, a posição das patas $\boldsymbol{p_i}(t)$, a força nas patas $\boldsymbol{f_i}(t)$ e as *phase durations* $\boldsymbol{\Delta} T_i(t)$ das pernas que satisfaçam as restrições de estado

inicial, posição final desejada, o modelo cinemático e o modelo dinâmico SRBD do robô, duração total igual a soma dos *phase durations*, forças nulas nas patas no *swing*. Além de no *stance*, força de reação do solo sempre positiva em relação a normal do terreno, não escorregamento das patas, forças na direção tangente ao terreno menores que a força de atrito e altura da pata em Z igual a altura do terreno.

$$
\begin{aligned}
\text{find} \quad & \mathbf{r}(t) \in \mathbb{R}^3 && \text{(Posição linear do CoM)} \\
& \boldsymbol{\theta}(t) \in \mathbb{R}^3 && \text{(Ângulos de Euler da base)} \\
& \text{for every foot } i: \\
& \quad \Delta T_{i,1} \ldots, \Delta T_{i,2n_{s,i}} \in \mathbb{R} && \text{(phase durations)} \\
& \quad \mathbf{p}_i(t, \Delta T_{i,1}, \ldots) \in \mathbb{R}^3 && \text{(posição das patas)} \\
& \quad \mathbf{f}_i(t, \Delta T_{i,1}, \ldots) \in \mathbb{R}^3 && \text{(força na pata)} \\
\text{s.t.} \quad & [\mathbf{r}, \boldsymbol{\theta}](t{=}0) = [\mathbf{r}_0, \boldsymbol{\theta}_0] && \text{(estado inicial)} \\
& \mathbf{r}(t{=}T) = \mathbf{r}_g && \text{(posição final desejada)} \\
& [\ddot{\mathbf{r}}, \dot{\boldsymbol{\omega}}]^T = \mathbf{F}(\mathbf{r}, \mathbf{p}_1, \ldots, \mathbf{f}_1, \ldots) && \text{(modelo dinâmico)} \\
& \text{for every foot } i: \\
& \quad \mathbf{p}_i(t) \in \mathcal{R}_i(\mathbf{r}, \boldsymbol{\theta}), && \text{(modelo cinemático)} \\
& \quad \text{if foot } i \text{ in contact}: \\
& \quad\quad \dot{\mathbf{p}}_i(t \in \mathcal{C}_i) = \mathbf{0} && \text{(não escorregamento)} \\
& \quad\quad p_i^z(t \in \mathcal{C}_i) = h_{terrain}(\mathbf{p}_i^{xy}) && \text{(altura do terreno)} \\
& \quad\quad \mathbf{f}_i(t \in \mathcal{C}_i) \cdot \mathbf{n}(\mathbf{p}_i^{xy}) \geq 0 && \text{(força de impulsão)} \\
& \quad\quad \mathbf{f}_i(t \in \mathcal{C}_i) \in \mathcal{F}(\mu, \mathbf{n}, \mathbf{p}_i^{xy}) && \text{(cone de atrito)} \\
& \quad \text{if foot } i \text{ in air}: \\
& \quad\quad \mathbf{f}_i(t \notin \mathcal{C}_i) = \mathbf{0} && \text{(sem força no ar)} \\
& \quad \textstyle\sum_{j=1}^{2n_{s,i}} \Delta T_{i,j} = T && \text{(duração total)}
\end{aligned}
$$

Figura 2.6: Formulação do problema de otimização realizado pelo TOWR. Os termos em marrom estão relacionados ao contato com o ambiente, os demais são aplicados de forma geral a sistemas de base flutuante - Adaptado de (WINKLER, 2018a)

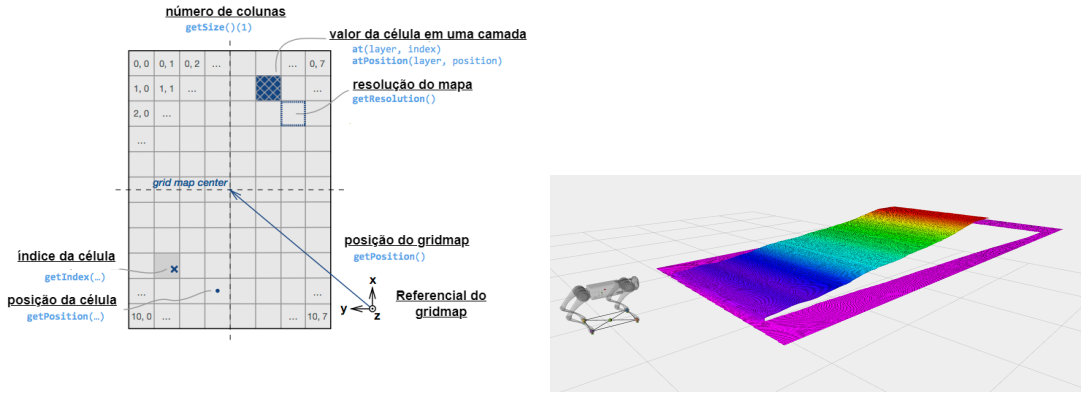
A biblioteca possui alguns terrenos pré-definidos simples para utilização no planejamento como degrau, rampa, vão e algumas pequenas variações. Para a definição do terreno, o TOWR utiliza funções analíticas contínuas para representar a altura ao longo do plano bem como para calcular as derivadas necessárias para a otimização. Apesar de personalizável e fácil de incluir um novo terreno dentro da plataforma, a definição de terrenos mais complexos por funções analíticas nem sempre é viável.

É importante ressaltar que, apesar do TOWR considerar a dinâmica de robôs com perna na formulação do problema de otimização, o modelo dinâmico utilizado é simplificado, e a biblioteca não fornece um mecanismo para testar as trajetórias encontradas em um simulador de corpos rígidos com o modelo completo do robô. Isso está fora do escopo do TOWR, que é simplesmente uma ferramenta de otimização de trajetórias. Utilizando o RViz como visualizador, as trajetórias geradas são mostradas cinematicamente na interface gráfica, sendo necessário portanto, utilizar de simulação física computadorizada ou testes com um protótipo real para validar as trajetórias encontradas. Alguns trabalhos anteriores mostraram a realização de trajetórias obtidas com algoritmos baseados no TOWR em robôs reais, indicando a viabilidade da sua utilização (MEDEIROS et al., 2020; BJELONIC et al., 2022).

2.7 Biblioteca Gridmap

A biblioteca `grid_map` é uma ferramenta utilizada para representar e manipular mapas em formato de grades (*gridmaps)* em aplicações de robótica e mapeamento. Os *gridmaps* são estruturas de dados bidimensionais que dividem o ambiente em células discretas, permitindo uma representação eficiente e compacta do espaço ocupado e livre (FANKHAUSER; HUTTER, 2016b).

A biblioteca oferece funcionalidades para a criação, atualização e manipulação de mapas em formato de grade (Figura 2.7a). Ela permite a especificação de diferentes tamanhos de células e resoluções de grade, o que possibilita a adaptação do mapa à escala e detalhe desejados. Além disso, a biblioteca suporta diferentes tipos de células. Além da célula simples, com um valor real associado, pode-se usar as células binárias (ocupado/livre) e células probabilísticas, que permitem representar a incerteza associada à ocupação de cada célula. Além disso, na mesma posição de uma célula pode-se ter diferentes valores relacionados a diferentes camadas (*layers*) de interesse como, por exemplo, uma camada de informações de elevação, uma de atrito e outra de temperatura, configurando o que se conhece na literatura como estrutura 2,5D.



(a) Mapeamento de células - Adaptado de (FANKHAUSER; HUTTER, 2016b)

(b) Mapa de elevação obtido a partir dos dados de uma câmera RGB-D.

Figura 2.7: Funcionalidades da biblioteca de gridmap

Uma das principais características da biblioteca é a capacidade de realizar operações eficientes de atualização e fusão de mapas. Isso permite que os robôs construam e atualizem seus mapas em tempo real, por exemplo, incorporando informações sensoriais de câmera (Figura 2.7b) ou LiDAR, por exemplo. Essa flexibilidade torna a biblioteca `grid_map` adequada também para tarefas de mapeamento em ambientes dinâmicos ou em sistemas multi-robôs (FANKHAUSER; HUTTER, 2016b).

2.8 Model Predictive Control (MPC)

O Controle Preditivo baseado em Modelo, mais conhecido como Model Predictive Control (MPC) se destaca como uma estratégia de controle avançada amplamente utilizada em diversos sistemas de controle, desde robótica e automação industrial até finanças e engenharia química. Sua principal característica reside na capacidade de prever o comportamento futuro do sistema e otimizar as ações de controle com base nesta previsão, resultando em um desempenho superior em comparação aos métodos de controle tradicionais (MACIEJOWSKI, 2000).

Dentro do contexto de robô com pernas, a utilização do MPC já se mostrou eficiente em diversas abordagens (HONG; KIM; PARK, 2020; DING et al., 2021; GRANDIA et al., 2021; CHO; PARK, 2022). Devido a dinâmica complexa envolvendo as pernas, perturbações externas e condições não previstas previamente afetam significativamente o rastreamento e a estabilidade do robô, tornando o MPC um importante módulo para qualquer framework de controle que objetiva atuar em ambientes complexos e não controlados.

Como os cálculos envolvendo o horizonte de predição necessitam das medições e estimações dos estados atuais, o MPC atua de forma on-line no sistema, sendo diretamente afetado pela velocidade de processamento e poder computacional embarcado. Por conta disso, com o avanço na tecnologia dos últimos anos, a utilização do MPC é cada vez mais frequente. Os principais termos associados a utilização do MPC são (MACIEJOWSKI, 2000):

- Modelo do Sistema: Um modelo matemático preciso do sistema a ser controlado é crucial para o sucesso do MPC. Esse modelo representa a dinâmica do sistema e permite prever seu comportamento futuro sob diferentes condições de entrada;
- Horizonte de Previsão: O MPC define um horizonte de previsão, que representa o intervalo de tempo futuro que o controlador considera ao tomar decisões. Um horizonte de previsão mais longo permite considerar eventos futuros com mais detalhes, mas pode aumentar a complexidade computacional;
- Problema de Otimização: A cada instante de tempo, o MPC resolve um problema de otimização que determina as ações de controle ideais para os próximos passos. Esse problema leva em consideração o modelo do sistema, o horizonte de previsão, os objetivos de controle e as restrições do sistema;
- Ações de Controle: As ações de controle otimizadas calculadas pelo MPC são então aplicadas ao sistema real. O controlador monitora a resposta do sistema e ajusta as ações de controle em futuras iterações, garantindo que o sistema siga a trajetória desejada.

O MPC se consolida como uma técnica de controle poderosa e versátil que oferece diversas vantagens em comparação a métodos tradicionais. Sua capacidade de prever o futuro e otimizar as ações de controle o torna ideal para diversos sistemas complexos, entregando alta performance e robustez. Apesar dos desafios relacionados à complexidade computacional e à necessidade de um modelo preciso, o MPC se destaca como uma ferramenta valiosa para diversos setores da robótica (KATAYAMA; TAZAKI, 2023).

2.9 Biblioteca OCS2

A biblioteca OCS2 (Optimal Control for Switched Systems) foi projetada especificamente para lidar com sistemas dinâmicos comutados. Ela fornece recursos avançados para a resolução de problemas de controle ótimo em sistemas

que alternam entre diferentes modos de operação, fornecendo implementações eficientes de algoritmos de otimização (FARSHIDIAN et al., 2017b).

Dentre os exemplos de formulação já incluídos na biblioteca, ela possui uma modelagem dinâmica de robôs com pernas, implementando uma abordagem com MPC não linear para o controle de movimento de um robô quadrúpede. O gait é definido pelo usuário e pode ser modificado durante a execução através de um módulo de sincronização e as trajetórias definidas por meio de um módulo gerenciador de referência. A função de custo utilizada é uma penalidade quadrática para rastrear a posição e o orientação da base e distribuir igualmente o peso do robô nas patas de apoio. O problema tem várias restrições, como força zero para as patas em *swing* e velocidade zero para as patas em *stance*. O cone de atrito é aplicado às forças de contato e as patas em *swing* acompanham um movimento pré-definido na direção Z.

A dinâmica do sistema é modelada de duas maneiras que podem ser escolhidas no arquivo de configuração: modelo dinâmico de corpo rígido único (SRBD), que assume que o robô é um único corpo rígido e as massas das pernas são desprezíveis em comparação com a base. A base é atuada através das forças de contato com o solo, aplicadas no ponto de contato de cada pata. A outra possibilidade é utilizar um modelo dinâmico centroidal (CD), que relaciona a mudança de momento linear e angular de todos os corpos rígidos projetado no centro-de-massa do sistema com as forças externas. Ao contrário do SRBD, esse modelo considera a cinemática completa do robô (FARSHIDIAN et al., 2017b).

2.10 Biblioteca LeggedControl

A biblioteca de código aberto `legged_control` (LIAO, 2022) fornece uma estrutura que faz a interface entre o OCS2 e o `ros_control` (CHITTA et al., 2017), o que permite a simulação no Gazebo (Seção 2.11) e testes no robô real usando ROS. A biblioteca é voltada para robôs com pernas e usa a formulação NMPC fornecida no OCS2 para locomoção com pernas, que emprega um Modelo Dinâmico Centroidal do robô como modelo de previsão. Para rastrear as trajetórias otimizadas pelo MPC, a biblioteca `legged_control` também fornece uma implementação de Whole-Body Control (WBC) que calcula os torques de atuação de cada junta resolvendo um problema de programação quadrática (QP) ponderada (GRANDIA et al., 2022). O diagrama de controle da biblioteca está ilustrado na Figura 2.8.

O módulo do NMPC resolve o seguinte problema de otimização a cada ciclo através das interfaces de formulação e resolução fornecidas pelo OCS2

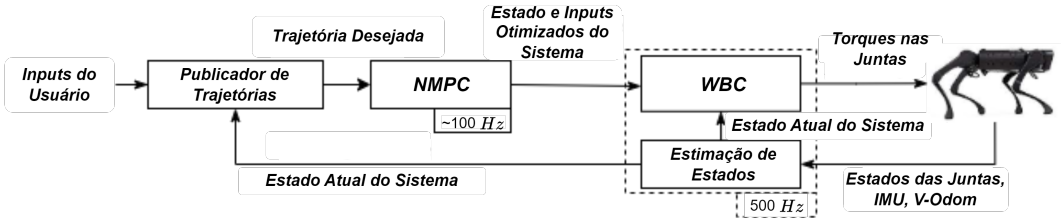


Figura 2.8: Diagrama de controle do framework legged_control - Adaptado de (LIAO, 2022)

$$
\begin{cases}
\min\limits_{\mathbf{u}(.)} \;\; \phi(\mathbf{x}(t_I)) + \int_{t_0}^{t_I} l(\mathbf{x}(t), \mathbf{u}(t), t), dt \\
\text{s.t.} \;\; \mathbf{x}(t_0) = \mathbf{x}_0, \qquad \text{estado inicial} \\
\quad \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \qquad \text{flow map do sistema} \\
\quad \mathbf{g}_1(\mathbf{x}(t), \mathbf{u}(t), t) = \mathbf{0} \qquad \text{restrições de igualdade entre estado e input} \\
\quad \mathbf{g}_2(\mathbf{x}(t), t) = \mathbf{0} \qquad \text{restrições de igualdade de estado} \\
\quad \mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), t) \geq \mathbf{0} \qquad \text{restrições de desigualdade}
\end{cases}
\tag{2-8}
$$

No framework, o estado $\boldsymbol{x}$ e o input $\boldsymbol{u}$ do sistema são definidos como

$$
\boldsymbol{x} = [\boldsymbol{h}_{com}^T, \boldsymbol{q}_b^T, \boldsymbol{q}_j^T]^T, \quad \boldsymbol{u} = [\boldsymbol{f}_c^T, \boldsymbol{v}_j^T]^T \tag{2-9}
$$

onde $\boldsymbol{h}_{com} \in \mathbb{R}^6$ é o vetor do momento centroidal normalizado, $\boldsymbol{q} = [\boldsymbol{q}_b^T, \boldsymbol{q}_j^T]^T$ é o vetor de coordenadas generalizadas. $\boldsymbol{f}_c \in \mathbb{R}^{12}$ consiste nas forças de contato das quatro patas (GRF). $\boldsymbol{q}_j$ e $\boldsymbol{v}_j$ são as posições e velocidades de juntas respectivamente. A função de custo é uma penalização quadrática no erro de rastreamento dos estados de referência. A dinâmica do sistema usa o CD com as restrições de cone de atrito, velocidade zero no pé em *stance* e a posição da pata em *swing* satisfaz no eixo Z a curva gerada por um polinômico cúbico.

Para resolver esse problema de controle ótimo, uma abordagem de *multiple shooting* é formulada para transcrever o problema de controle ótimo para um problema de programação não linear (NLP), que é resolvido usando a programação quadrática sequencial (SQP). O solver de QP utilizado nesta formulação é o HPIPM (FRISON; DIEHL, 2020). Mais detalhes podem ser obtidos em (GRANDIA et al., 2022).

Ao contrário do NMPC, o módulo de WBC só considera o momento atual de medição para seus cálculos, realizando um controle hierárquico de tarefas. As tarefas estão definidas da tabela 2.2, onde cada tarefa representa uma restrição de igualdade ou desigualdade para as variáveis de decisão. As

Prioridade	Tipo	Tarefa
0	$=$	EoM de base flutuante
	$\geq$	Limites de torque
	$\geq$	Restrição de cone de atrito
	$=$	Movimento zero nos pontos de contato
1	$=$	Aceleração linear e angular da base
	$=$	Rastreamento da trajetória da pata em swing
2	$=$	Rastreamento de força de contato.

Tabela 2.2: Tarefas e Prioridades para o Whole-Body Control separadas em igualdades e desigualdades

variáveis de decisão são

$$\boldsymbol{x}_{wbc} = [\ddot{\boldsymbol{q}}^T, \boldsymbol{f}_c^T, \tau^T]^T \tag{2-10}$$

onde $\ddot{\boldsymbol{q}}$ é a aceleração das coordenadas generalizadas e $\boldsymbol{\tau}$ é o vetor de torque das juntas. o WBC resolve o problema QP no espaço nulo (*null space*) das restrições lineares das tarefas de alta prioridade e tenta minimizar minimizar as variáveis das restrições de desigualdade. Essa abordagem pode considerar toda a dinâmica não linear do corpo rígido e garantir resultados hierárquicos rigorosos. Mais detalhes podem ser obtidos em Bellicoso et al. (2016).

2.11 Gazebo

O Gazebo se destaca como um simulador robótico 3D de código aberto amplamente utilizado na comunidade de robótica (KOENIG; HOWARD, 2004). Desenvolvido pela Open Robotics Foundation, o Gazebo oferece um ambiente virtual realista e robusto para testar, depurar e avaliar robôs. Ele utiliza como solver o ODE (Open Dynamics Engine), que permite simular as interações físicas entre o robô e o ambiente de forma precisa e realista. Isso permite aos desenvolvedores avaliar o desempenho do robô em diferentes condições, como terrenos acidentados, obstáculos e interações com outros objetos.

Além disso, o Gazebo possui ferramentas de integração com ROS, possuindo diversos *plugins* e bibliotecas já desenvolvidas para diferentes aplicações. Por conta disso, ele foi utilizado neste trabalho para validar dinamicamente as trajetórias geradas pelo processo de otimização de trajetória com um controlador NMPC + WBC em terrenos diversos e sob atuação de perturbações externas. A Figura 2.9 representa a interface gráfica do Gazebo, onde está destacado o sistema de coordenadas do referencial inercial utilizado para as

simulações.

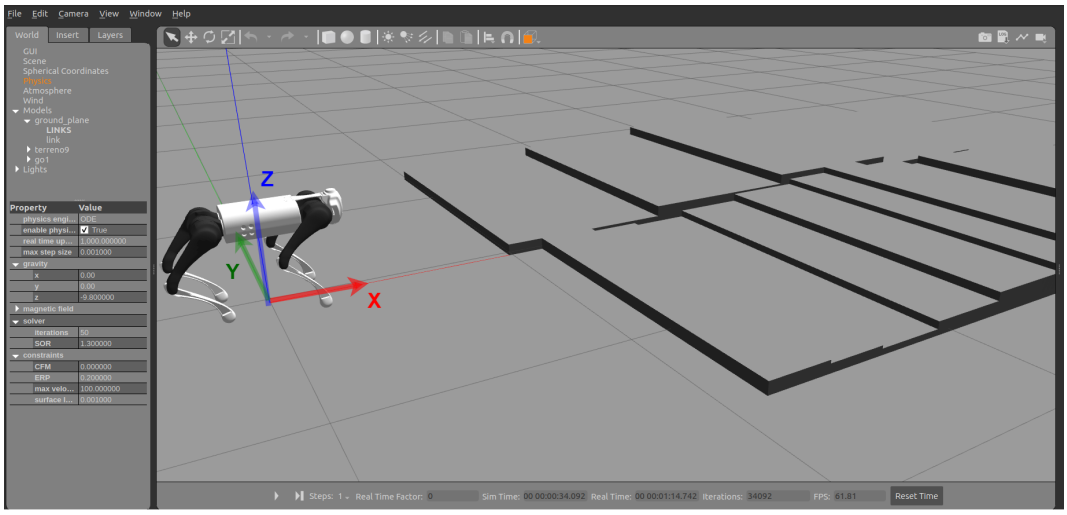


Figura 2.9: Interface Gráfica do Gazebo

Como o Gazebo é focado em simulação física, é muito comum utilizar em conjunto um framework de visualização. Neste trabalho utilizou-se o RViz para a visualização das trajetórias e adição de elementos gráficos que ajudem na análise qualitativa do sistema.

3
Biblioteca Grid2Towr

Para abordar o problema de locomoção de robôs com pernas em terrenos acidentados, o primeiro passo é utilizar o terreno como entrada para o planejador, etapa destacada em vermelho na Figura 3.1. Apesar da biblioteca padrão do TOWR possibilitar a escolha de alguns tipos de terrenos, suas modificações e personalizações podem ser inviáveis para terrenos mais complexos. Esta etapa do desenvolvimento documenta uma integração entre a biblioteca de otimização de trajetórias TOWR e a biblioteca de mapeamento `grid_map`, expandindo as opções de representação de terrenos que podem ser utilizadas e, com isso, permitindo a utilização de terrenos mais complexos. A biblioteca de `grid_map` já possui nativamente integração com ROS usando comunicação via tópicos e mensagens, além de já existem diversas aplicações que trabalham com a geração de gridmaps possibilitando, por exemplo, a geração de mapas através de imagens de câmeras RGB-D.

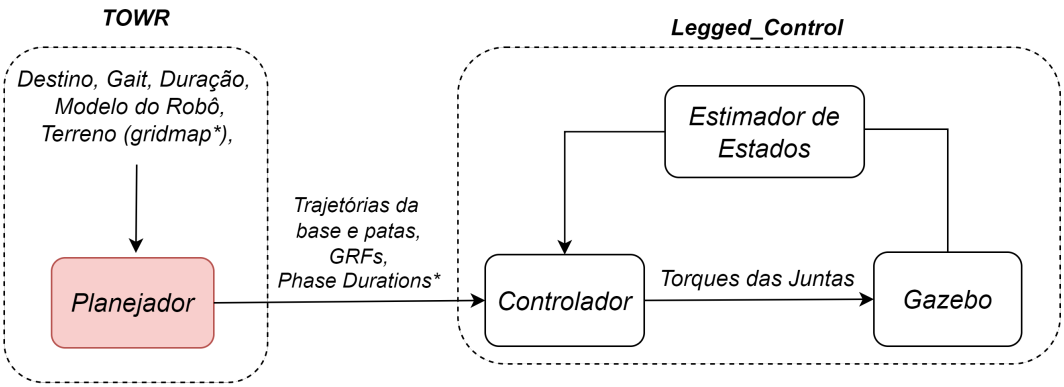


Figura 3.1: Fluxograma de locomoção - Etapa de planejamento

Os desenvolvimentos implementados nesta etapa estão destacados de vermelho e resumidos no diagrama da Figura 3.2. Compreende a criação de uma classe Grid2Towr, responsável por definir um novo terreno baseado em um gridmap dentro do TOWR, a adição de duas restrições na formulação do problema de otimização, e a modificação da rosbag de saída para incluir uma mensagem de *phase durations* de cada perna do robô.

3.1
Definições de Terreno no TOWR

Na biblioteca padrão do TOWR, é possível escolher alguns terrenos predefinidos para realizar a otimização de trajetória. São eles degrau, rampa, vão e outras variações (Figura 3.3).

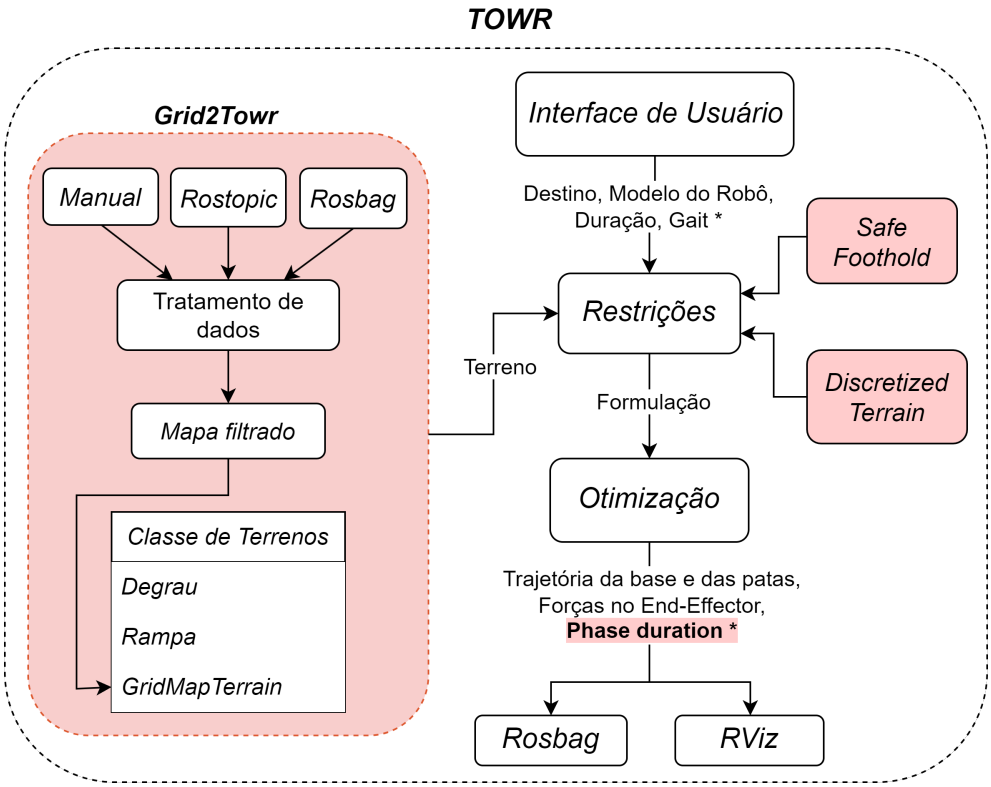


Figura 3.2: Diagrama de implementação do Grid2Towr

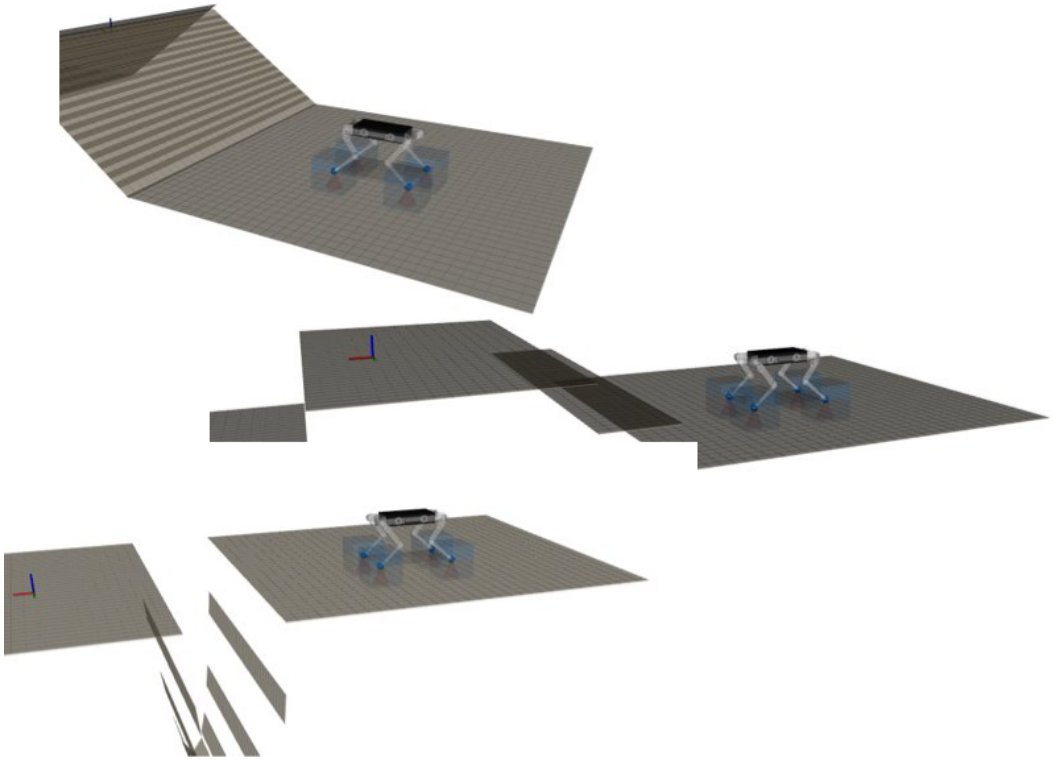


Figura 3.3: Exemplo dos principais terrenos disponíveis no TOWR

Na biblioteca TOWR, o terreno é definido como funções analíticas que retornam a sua altura em um determinado ponto no espaço e funções que

retornam as suas derivadas parciais, de primeira e segunda ordem, nas direções X e Y. Esses valores são usados como entradas no processo de otimização e são definidos manualmente dentro da biblioteca. O objetivo portanto é criar um novo tipo de terreno, chamado *GridMapTerrain*, que tem as suas posições e derivadas fornecidas pela nova classe Grid2Towr, que computa essas grandezas a partir de um gridmap genérico. Uma abordagem semelhante foi aplicada em (JELAVIC; FARSHIDIAN; HUTTER, 2021) para incluir representações de terreno em gridmap como entrada de um algoritmo de otimização de trajetória para robôs híbridos com pernas e com rodas. Os autores utilizam algoritmos de interpolação bicúbica e convolução bicúbica para gerar a representação analítica do terreno.

3.2 Implementação do Grid2Towr

Criou-se portanto uma nova classe *Grid2Towr* que fornece as funções necessárias para integrar um `grid_map` ao TOWR. A classe utiliza a biblioteca de mapeamento `grid_map` que oferece funcionalidades para a criação, atualização e manipulação de mapas em formato de grid. Ela permite a especificação de diferentes tamanhos de células e resoluções de grade, o que possibilita a adaptação do mapa à escala e detalhe desejados.

A definição do terreno como um gridmap dentro da Grid2Towr pode ser feita de três formas: definindo um gridmap manualmente pelas funções nativas da biblioteca; através de um *subscriber*[1] em um tópico no qual está sendo publicado um mapa na forma de um mapa de elevação (e.g. através de uma camêra) ou utilizar dados armazenados em uma rosbag que contém mensagens referentes a um mapa de elevação.

Para obter a altura de um determinado ponto no gridmap, foi usada uma função nativa da biblioteca `grid_map` e as derivadas foram calculadas numericamente considerando a variação entre o ponto de interesse, o ponto anterior e o ponto posterior separados por um distância personalizável.

Alguns tratamentos de dados foram implementados:

- Como o mapeamento da biblioteca define o grid em células discretas, utilizou-se de interpolação linear para tratar casos onde o ponto desejado esteja entre duas células;
- Aplicou-se uma correção de arredondamento para casos que a derivada calculada for muito próxima de zero;

[1]Em ROS, um subscriber é uma unidade de processamento que recebe mensagens publicadas em um determinado tópico, atuando como um ouvinte, aguardando fluxos de dados que correspondam ao tópico inscrito.

– Para o caso de células sem valor, problema comum em mapas gerados através de câmeras (i.e. NaN) atribui-se valor nulo;

– Em gridmaps mais complexos gerados por sensores é comum a presença de ruídos, para isso, adicionou-se um filtro de média móvel para remoção de *outliers*.

3.3 Implementação dentro do TOWR

No TOWR, é possível selecionar o terreno a ser utilizado na otimização direto pela interface de usuário. Para a nova implementação, modifica-se os arquivos de configuração para incluir a nova biblioteca e o novo terreno *GridMapTerrain* que utiliza a classe *Grid2Towr* para computar as funções de altura e derivadas. Com isso, é necessário apenas selecioná-lo na interface de usuário como um dos outros terrenos (Figura 3.4).

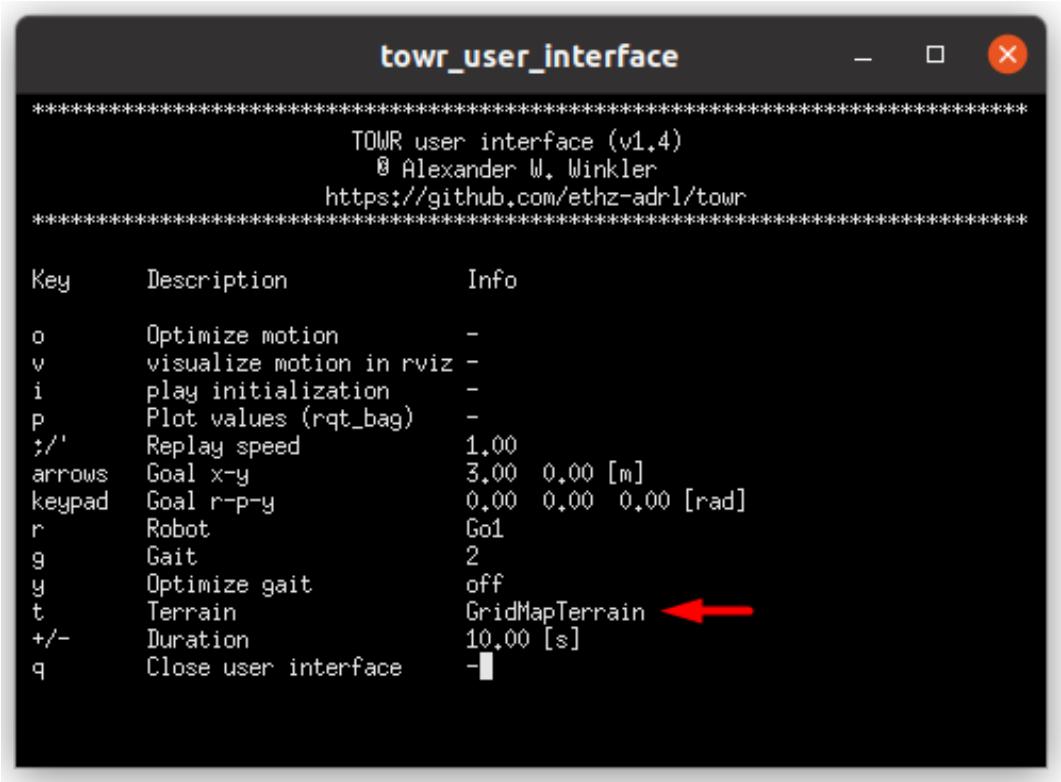
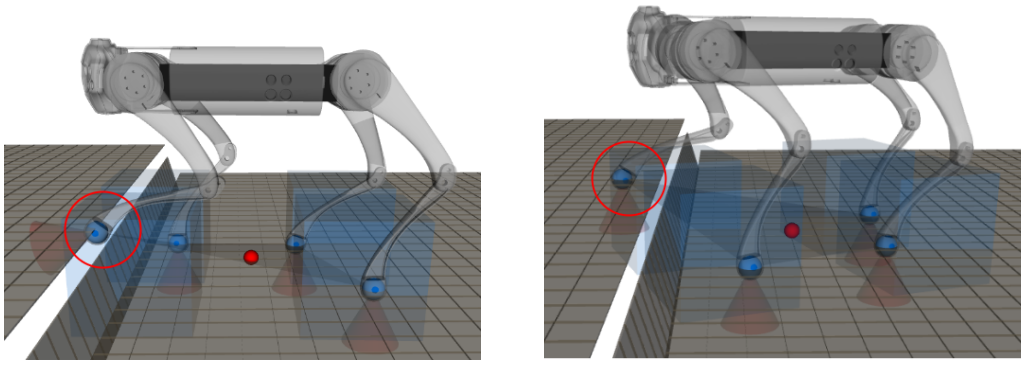


Figura 3.4: interface de usuário do TOWR

3.4 Melhorias na Otimização

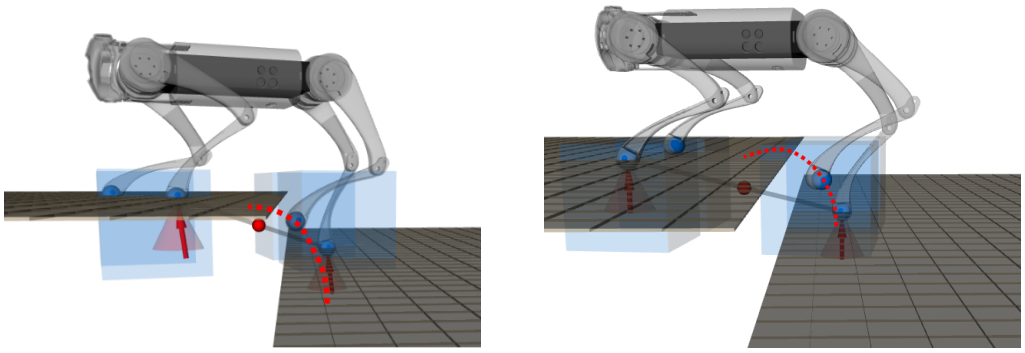
Com relação aos aprimoramentos na formulação e nos resultados de otimização do TOWR, duas novas restrições propostas por (MEDEIROS et al., 2020) foram incorporadas à formulação do TOWR com o objetivo de aprimorar

a trajetória gerada. A restrição de *safe foothold* força o planejador a colocar o pé em uma posição em que a altura do terreno seja constante em um determinado raio ao redor do ponto de apoio considerado (Figura 3.5b). A restrição de *discretized terrain* garante que não haja colisão com obstáculos durante os movimentos de *lift-off* ou *touch-down* dos pés, avaliando as restrições de desigualdade do terreno para etapa de *swing* em pontos discretizados em vez de apenas junções polinomiais (Figura 3.6b).



(a) Sem a restrição (b) Com a restrição

Figura 3.5: Comparativo da restrição de *safe foothold*



(a) Sem a restrição (b) Com a restrição

Figura 3.6: Comparativo da restrição de *discretized terrain*

3.5 Gait Optimization

Uma importante ferramenta do TOWR é a possibilidade de incluir a otimização do gait no processo de planejamento de trajetória, expandindo a quantidade de movimentos possíveis. Caso o usuário opte por não otimizar o gait, a ferramenta planeja a trajetória com um gait fixo pré-definido onde as *phase durations* de cada pata são constantes.

Para realizar a otimização de gait, o TOWR realiza uma otimização de *phase durations*. Dado um gait inicial, o solver tem liberdade para variar o tamanho de cada *phase* em busca da solução do problema de otimização destacado na Seção 2.6.

Ao longo deste trabalho, utilizou-se o gait do tipo *standing trot* nos casos onde não se otimizou o gait. Trata-se de um gait semelhante ao trote mas que possui fases de *full stance* intermediárias entre os movimentos de *swing*, possibilitando uma maior estabilidade. Com as *phase durations* de cada pata definidas, é possível montar o diagrama do gait ilustrado na Figura 3.7. É possível observar que os *phase durations* de *stance* e *swing* são constantes independente da perna, e as pernas cruzadas tem comportamentos idênticos.

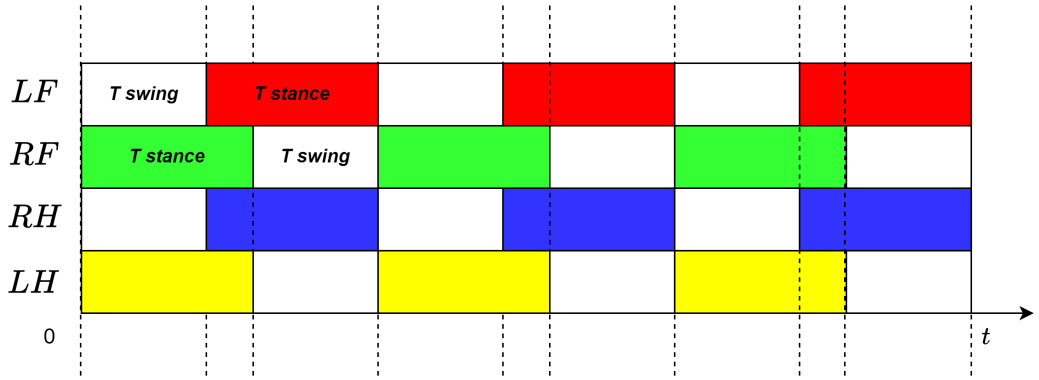


Figura 3.7: Diagrama de fases para o gait *standing trot*. Nos trechos em branco, a pata está no ar e nos trechos coloridos, a pata está em contato com o solo.

Utilizando a otimização de gait, as *phase durations* de cada pata são adicionadas como saídas da otimização, permitindo formas mais versáteis de ultrapassar terrenos mais complexos.

No TOWR, apesar das *phase durations* serem output da otimização, seus valores não são incluídos na rosbag final, e mesmo que a trajetória espacial de cada pata, represente indiretamente o gait do robô, em diversos frameworks de controle a tarefa de geração de gait é tratada de forma independente da tarefa de rastreamento (como é o caso do `legged_control`, utilizado neste trabalho). Para resolver este problema, criou-se uma nova função no TOWR que gera uma nova mensagem personalizada armazenada na rosbag, contendo as informações de *phase durations* e *contact states* de cada pata ao longo de toda a trajetória. Com essas informações é possível definir completamente o gait que foi otimizado. Com os dados da Tabela 3.1, é possível montar o diagrama de gait como mostra a Figura 3.8 exemplificando o resultado de um gait otimizado para um terreno senoidal com um destino de 2 metros e um tempo de trajetória de 5 segundos. Nota-se que o gait otimizado é bem mais complexo que o gait pré-definido e que não tem um padrão claro de repetição

como o caso do *standing trot*. Também é possível observar que o gait gerado possui momentos de *full flight phase* onde todas as patas estão em *swing* ao mesmo tempo.

LF	RF	LH	RH
0.37	0.21	0.20	0.36
0.20	0.25	0.20	0.20
0.22	0.20	0.20	0.27
0.20	0.23	0.28	0.21
0.27	0.20	0.21	0.28
0.21	0.27	0.27	0.20
0.28	0.21	0.21	0.27
0.21	0.21	0.27	0.20
0.21	0.20	0.20	0.22
0.20	0.27	0.21	0.20
0.27	0.21	0.21	0.27
0.21	0.27	0.27	0.21
0.21	0.21	0.21	0.28
0.20	0.27	0.27	0.20
0.27	0.20	0.20	0.21
0.21	0.21	0.29	0.20
0.26	0.20	0.20	0.26
0.20	0.28	0.21	0.20
0.20	0.215	0.20	0.20
0.20	0.35	0.35	0.20

Tabela 3.1: Tabela de *phase durations* resultantes do processo de otimização de gait para um terreno senoidal genérico

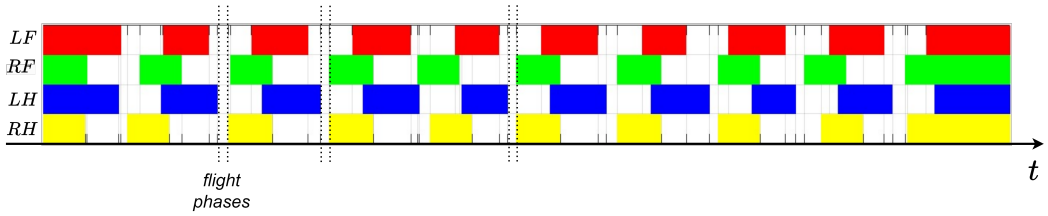


Figura 3.8: Diagrama de phases para o gait otimizado no terreno senoidal genérico

3.6 Visualização das Trajetórias Ótimas

Com estas implementações, foi possível encontrar trajetórias otimizadas para diferentes novos terrenos, exemplificados na Figura 3.9.

Após a otimização, é possível definir dentro do TOWR uma rosbag de destino para armazenamento da trajetória encontrada. Como saída padrão do

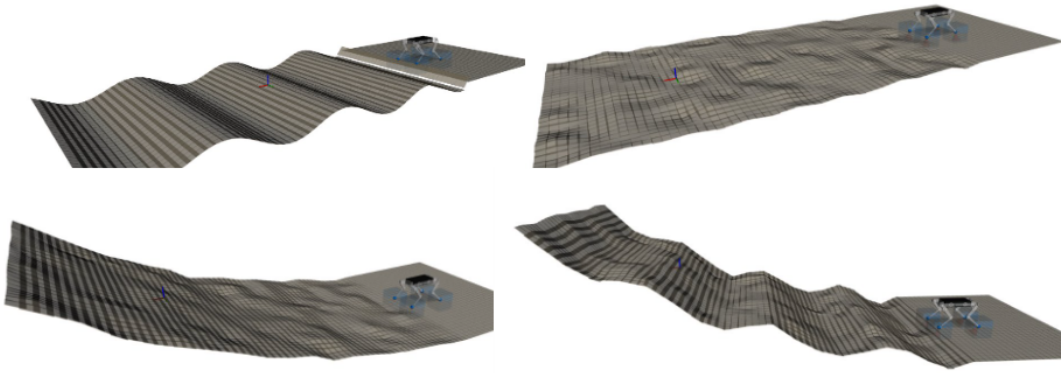


Figura 3.9: Exemplo de novos terrenos possíveis com o Grid2Towr.

processo de otimização, tem-se a trajetória no espaço da base, a trajetória no espaço de cada pata e a força de reação do solo em cada pata. Como comentado na Seção 3.5, foi adicionado também uma mensagem referente ao *phase duration* e o *contact state* de cada pata.

Pela rosbag de saída, é possível com alguma ferramenta gráfica, visualizar as informações armazenadas resultantes do processo de otimização. As Figuras 3.10, 3.11 e 3.12 exemplificam a visualização dos resultados de otimização para um terreno em forma de senoide definido manualmente dentro do Grid2Towr, uma ponto destino de 2 metros, 5 segundos de trajetória e otimizando o gait. São apresentados os gráficos da posição da base, as trajetória das patas no eixo X e as forças de contato. Note que, como esperado, as forças são nulas quando a pata não está em contato com o solo.

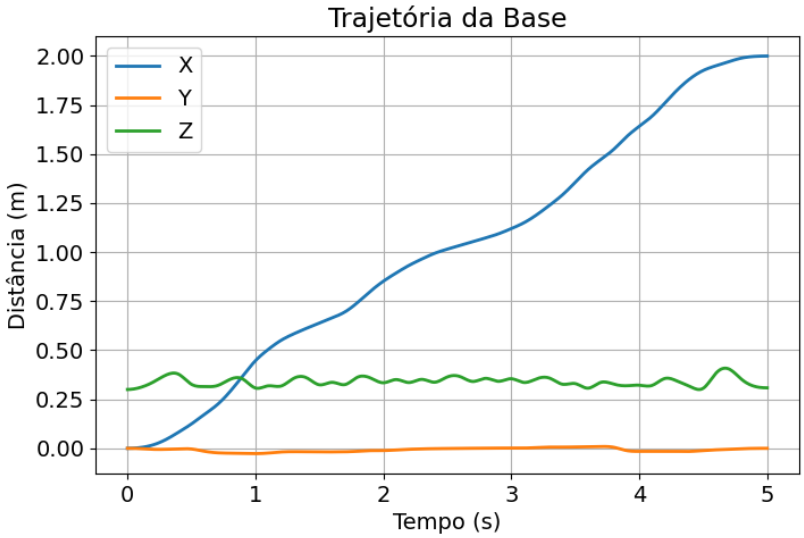


Figura 3.10: Trajetória da base em X,Y e Z para o terreno senoide

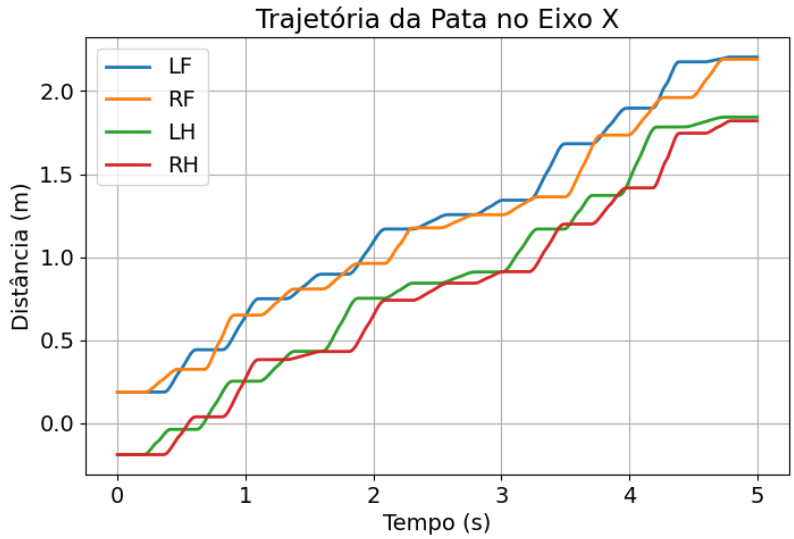


Figura 3.11: Trajetória das patas com o gait otimizado para o terreno senoide

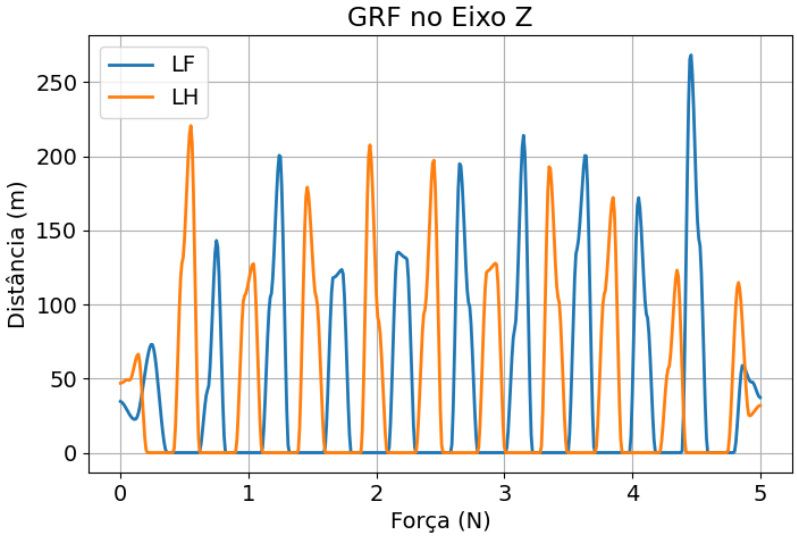


Figura 3.12: Forças de reação do solo nas patas LF e LH no eixo Z para o terreno senoide

4
Biblioteca Towr2LeggedControl

Apesar do TOWR gerar as trajetórias ótimas utilizando como uma das restrições a dinâmica do robô, ainda é importante aplicar essas trajetórias em um controlador em malha fechada e avaliar a performance do robô real ou em ambiente de simulação física, como mostra o diagrama na Figura 4.1.

A OCS2 (Optimal Control for Switched Systems) é uma biblioteca de código aberto projetada especificamente para lidar com sistemas dinâmicos comutados, fornecendo recursos para resolver problemas de controle ótimo em sistemas que alternam entre diferentes modos de operação (FARSHIDIAN et al., 2017a). Ela é particularmente útil para a implementação de MPC para diferentes plataformas robóticas, incluindo robôs com pernas. A biblioteca `ros_control`, por outro lado, fornece ferramentas de controle e interfaces de hardware para controlar robôs de base fixa e flutuante tanto na simulação quanto no hardware físico usando o ROS.

A biblioteca de código aberto `legged_control`, como comentado na Seção 2.10, fornece uma estrutura que faz a interface entre o OCS2 e o `ros_control`, o que permite a simulação no Gazebo e testes no robô real usando o ROS. A biblioteca é voltada para robôs com pernas e usa a formulação MPC fornecida no OCS2 para locomoção com pernas, que emprega um modelo dinâmico Centroidal do robô. Para rastrear as trajetórias geradas pelo MPC no robô, a biblioteca `legged_control` também fornece um sistema de Whole-Body Controller (WBC) que calcula os torques de atuação resolvendo um problema de programação quadrática (QP) ponderada (GRANDIA et al., 2022).

Esta etapa do trabalho tem como objetivo integrar as saídas do TOWR como entradas para o `legged_control` realizando as adaptações necessárias para utilizar as trajetórias computadas pelo TOWR como referência para o MPC e validá-las dinamicamente com o simulador Gazebo em cenários com terrenos não estruturados e perturbações externas constantes.

Os desenvolvimentos implementados nesta etapa estão destacados de vermelho e resumidos no diagrama da Figura 4.2. Compreende a criação do módulo de integração Towr2LeggedControl, responsável por ler a rosbag gerada pelo TOWR, converter a trajetória para os vetores de referência que o MPC necessita e converter as informações de *phase durations* para o formato de gait do controlador. Foi também realizada a implementação de um código de

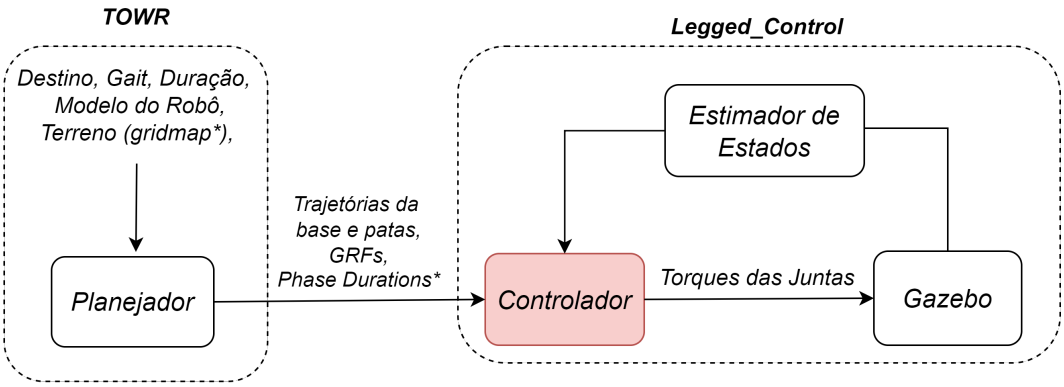


Figura 4.1: Fluxograma de locomoção - Etapa de controle

geração de *world*[1] no Gazebo utilizando um arquivo de malha do terreno e o desenvolvimento de uma função que aplica uma força externa personalizável durante a simulação.

Por padrão, para utilizar a biblioteca é necessário inicializar de forma independente a simulação, o controlador e o o gait (neste framework, a definição do gait é feita de forma independente ao rastreamento). Em seguida, é possível definir uma posição desejada no espaço e o framework calcula uma trajetória simplificada para a base considerando uma interpolação linear entre o ponto inicial e final desejado. Essa trajetória é utilizada como referência no MPC e o resultado segue é rastreado pelo WBC, fechando a malha com o módulo de estimação de estados. No caso da simulação, o módulo de estimação de estados simplesmente usa a posição real do robô gerado pelo Gazebo. Para modificar o terreno padrão da simulação é necessário criar um novo *world* no Gazebo adicionando os modelos de terreno gerados internamente ou externamente com alguma ferramenta CAD, por exemplo.

4.1 Implementação da Towr2LeggedControl

Dentro do `legged_control`, a trajetória de referência é fornecida ao MPC através de um tipo específico de mensagem composto de três elementos:

- Estados: Vetor composto do momento linear e angular em relação ao centro de massa, as coordenadas lineares e angulares da base e as posições das juntas;
- Inputs de Controle: Vetor composto das forças de reação do solo (GRF) e as velocidades das juntas;
- Tempos: Vetor de tempos para a referência ao longo de toda a trajetória.

[1]Formato de arquivo utilizado pelo Gazebo para definir o cenário da simulação.

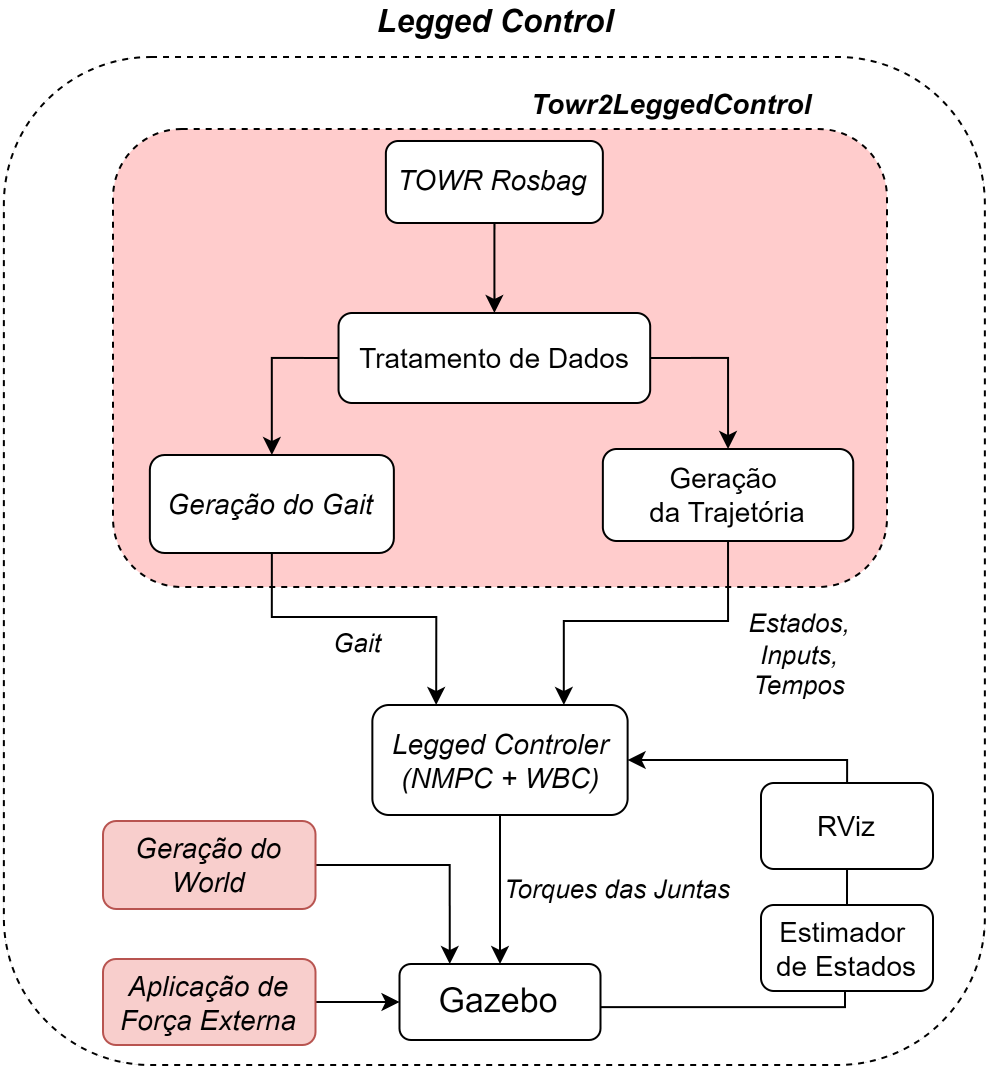


Figura 4.2: Diagrama de implementação do Towr2LeggedControl

O objetivo, portanto, é ler a rosbag emitida pelo TOWR e popular estes vetores nos formatos necessários. Para isso, diversos tratamentos de dados tiveram que ser realizados, ilustrados na Figura 4.3:

- Converter a posição angular da base de quatérnios para ângulos de Euler;
- Como o TOWR retorna apenas a posição das patas, utilizou-se da cinemática inversa calcular os posições das juntas referentes a cada nó da trajetória;
- Corrigir a ordem das pernas pois o TOWR e o Legged Control utilizam padrões de ordem diferentes;
- Ajustar os tempos de referência com o tempo da simulação para começarem quando a trajetória for de fato publicada;
- A trajetória das patas do TOWR considera a extremidade da perna como um ponto (*point feet*) mas o robô real tem uma esfera na extremidade,

logo adicionou-se à posição do end-effector mais 0.02 m na direção da normal do terreno para compensar o raio da pata.

– Os vetores de referência que o TOWR não gera na otimização foram considerados como zero, são eles os momentos lineares e angulares e as velocidades de junta. Isso implica com que o robô tente reduzir mudanças bruscas na orientação da base e minimizar as mudanças bruscas nos ângulos das juntas.

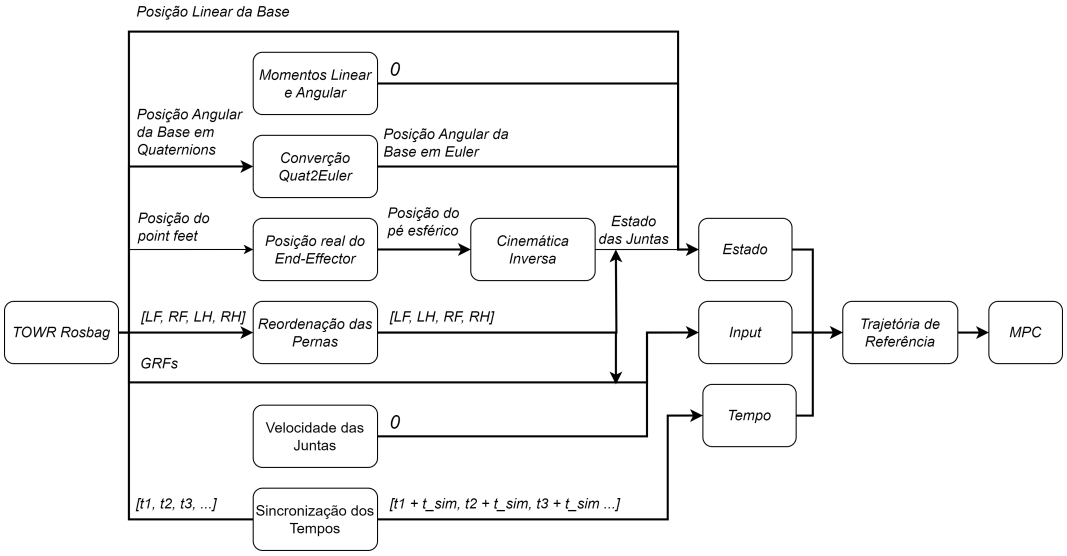


Figura 4.3: Tratamento de dados para geração da trajetória de referência

4.2 Geração do Gait

Dentro do `legged_control`, o módulo do gait atua de forma independente ao rastreamento de trajetória, sendo definido e executado de forma separada. Por padrão da biblioteca, para a definição do gait são necessários dois vetores, um que define quais patas estão em contato com o solo em cada estado e outro que define os tempos de cada mudança de estado das patas.

Como um dos objetivos do trabalho é validar a ferramenta de otimização de gait do TOWR em ambiente dinâmico, adaptou-se a rosbag de saída do TOWR para fornecer os dados de *phase durations* de cada pata, como já comentado na Seção 3.5. Utilizando os dados dessa rosbag, desenvolveu-se uma nova função para converter a informação de *phase durations* para o formato necessário pelo módulo de gait do `legged_control`, como mostra a Figura 4.4.

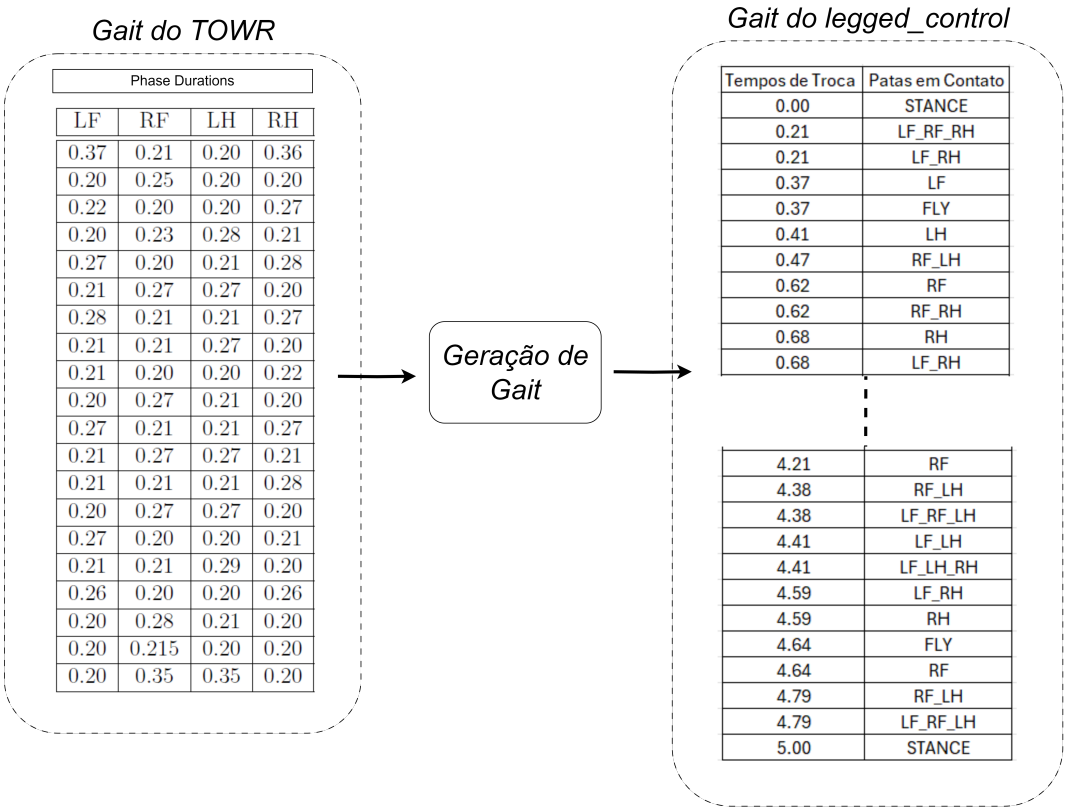


Figura 4.4: Módulo de conversão de formato de gait. Neste exemplo foi utilizado o gait gerado no exemplo da Seção 3.5

4.3 Geração dos Terrenos

Para geração dos terrenos utilizados no trabalho, adotou-se o fluxograma apresentado na Fig 4.5. Devido a facilidade de criar geometrias parametrizadas e com base polinomial, utilizou-se o Solidworks (SYSTèMES, 2023) para as modelagens CAD[2] do terreno e geração dos arquivos de geometria triangular universal (STL[3]). Com o software Blender (COMMUNITY, 2018), redefine-se a origem e orientação do arquivo STL e converte-se o modelo para um arquivo de malhas (DAE[4]) necessário para utilização no Gazebo. Em seguida, o arquivo de malhas é utilizado para a geração do arquivo *world* onde define-se a posição no espaço do terreno e é gerado arquivo YAML necessário para visualização do terreno no RViz.

[2] *Computer-Aided Design*, refere à utilização de softwares e ferramentas digitais para criar modelos 3D precisos e detalhados de objetos, peças ou produtos

[3] *Standard Triangle Language*, formato de arquivo que carrega descrição precisa da superfície de um objeto 3D, convertida para série de triângulos

[4] *Digital Asset Exchange* formato de arquivo contendo informações geométricas usadas para descrever objetos e malhas 3D

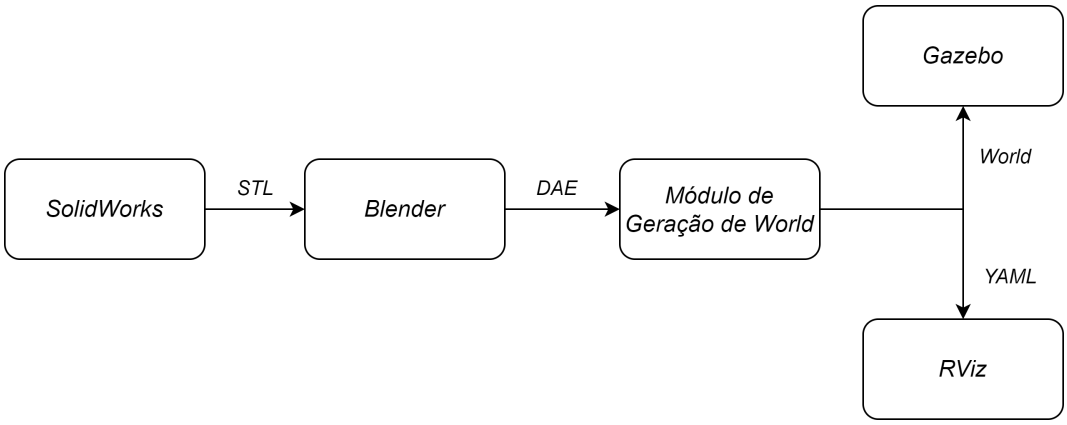


Figura 4.5: Fluxograma para geração dos terrenos

4.4 Módulo de Aplicação de Força Externa

Como um dos objetivos da pesquisa é validar as trajetórias geradas em cenários com perturbações externas, uma função de aplicação de força foi implementada. Desta forma, é possível adicionar uma força externa aplicada no robô durante a simulação, com direção, módulo e tempo de aplicação definidos pelo usuário (Figura 4.6).

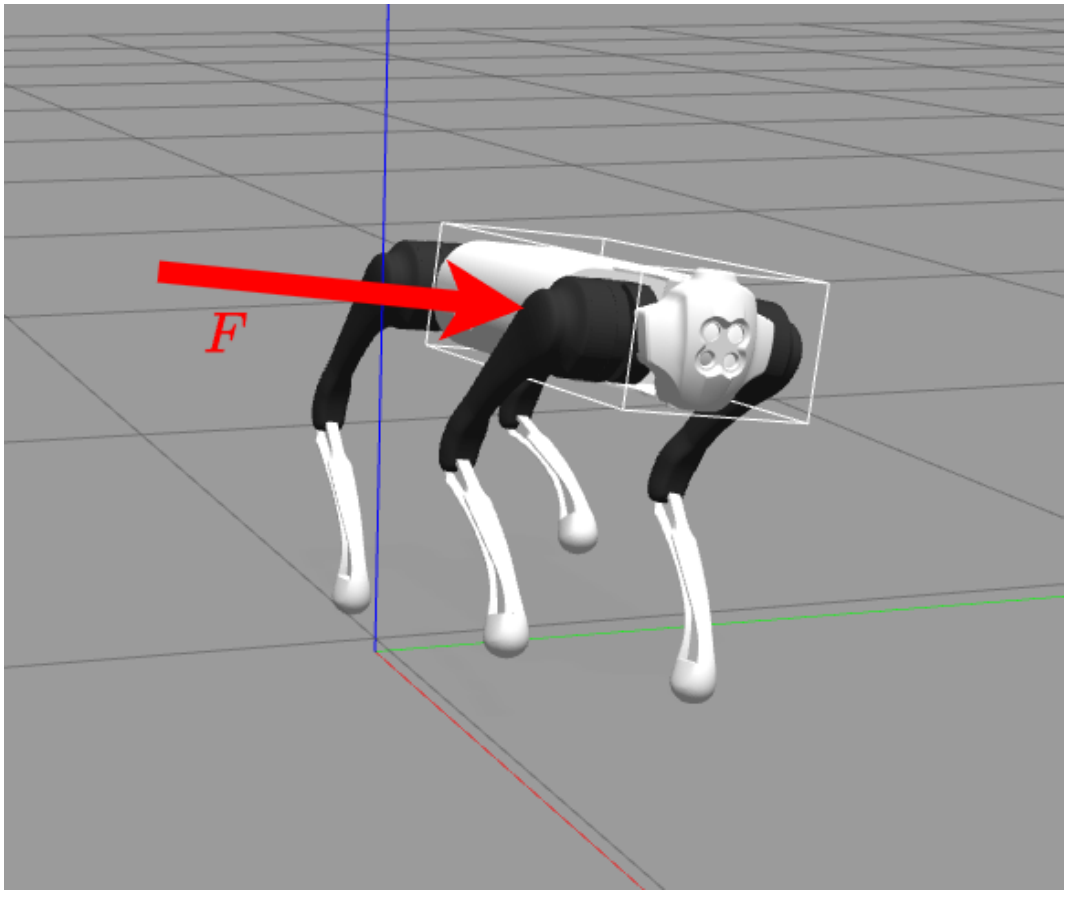


Figura 4.6: Força externa sendo aplicada durante a simulação

4.5 Calibração das Constantes de Controle

Após as implementações descritas nas seções anteriores, foi possível utilizar a trajetória do TOWR como referência para o MPC na biblioteca `legged_control`. Posteriormente, iniciou-se uma análise de sensibilidade para a calibração dos parâmetros de controle e aprimorar o rastreamento de trajetória. Utilizando os parâmetros padrão, variou-se um deles por vez observando os resultados na simulação até atingir resultados satisfatórios. As Tabelas 4.1, 4.2, 4.3, 4.4, 4.5, 4.6 destacam os principais parâmetros alterados no framework.

	Frequência	Horizonte de predição
MPC	100 Hz	3 s

Tabela 4.1: Parâmetros principais utilizados no MPC

	pos x	pos y	pos z	θ_x	θ_y	θ_z
Matriz Q	5000	5000	7000	500	1500	1500

Tabela 4.2: Matriz de pesos referente ao rastreamento da trajetória da base na função custo do MPC.

Matriz Q	HAA	HFE	KFE
LF	3000	3000	1500
LH	3000	3000	1500
RF	3000	3000	1500
RH	3000	3000	1500

Tabela 4.3: Matriz de pesos referente ao rastreamento das posição das juntas na função custo do MPC.

Matriz R	Fx	Fy	Fz
LF	7	7	7
LH	7	7	7
RF	7	7	7
RH	7	7	7

Tabela 4.4: Matriz de pesos referente ao rastreamento das forças de contato na função custo do MPC.

	kp	kd
Swing WBC	1000	100

Tabela 4.5: Ganho proporcional e ganho derivativo da tarefa de rastreamento da trajetória de swing das patas no WBC.

	Vel. de lift off	Vel. de touchdown	altura no swing
Swing Config	1 m/s	-0.7 m/s	0.11m

Tabela 4.6: Configurações dos principais parâmetros da trajetória de swing para o MPC.

4.6 Análise de Resultados

É possível utilizar a interface gráfica do Gazebo para acompanhar a simulação física do movimento do robô, sendo possível analisar seu desempenho de forma qualitativa. Adaptou-se também o visualizador RViz para mostrar ao longo do movimento as trajetórias de referência, como exemplificado na Figura 4.7, onde é possível visualizar as trajetórias de referência fornecidas pelo TOWR e as trajetórias realizadas de fato pelo robô. De forma quantitativa, utilizando uma ferramenta de geração de gráficos, é possível exportar os dados e plotar os principais gráficos para comparar a trajetória de referência gerada na etapa de otimização no TOWR com a trajetória executada pelo robô na simulação, usando como métrica para análise do erro de rastreamento o RMSE (erro quadrático médio), calculado como

$$RMSE = \sqrt{\frac{\Sigma_{i=1}^{N}(y_i - \hat{y}_i)^2}{N}}, \tag{4-1}$$

onde N é o número de pontos, y_i é o valor medido e $\hat{y}_i$ é a referência, ambos em relação a amostra i.

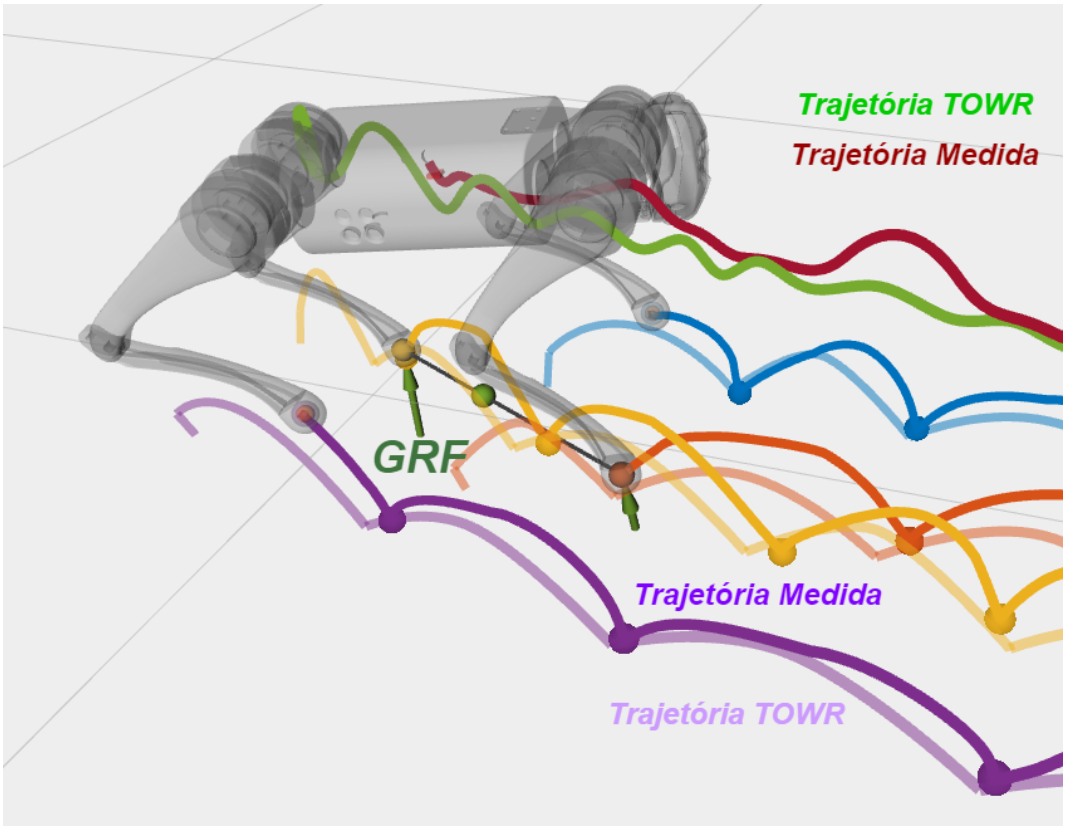


Figura 4.7: Visualização das trajetórias no RViz. Para a base, a curva verde representa a trajetória obtida no TOWR e a vermelha representa a trajetória medida dentro do horizonte de predição do MPC. Para as patas, as cores sólidas são as trajetórias medidas e as cores com transparência são as trajetórias de referência do TOWR. Ainda é possível visualizar as GRFs nas patas que estão em contato com o solo.

5
Resultados e Discussão

Após implementados os desenvolvimentos descritos nos Capítulos anteriores, utilizou-se o robô Go1 para navegar por diferentes terrenos e avaliar a performance do sistema de controle de rastreamento de trajetória. Nas próximas seções, está descrito detalhadamente o procedimento completo desde o planejamento de trajetórias, o rastreamento em simulação e a análise de resultados para quatro diferentes terrenos.

O objetivo desta etapa é validar a viabilidade das trajetórias geradas no TOWR como referência prévia para o controlador. Como o robô está submetido a perturbações, o distanciamento da trajetória de referência em relação a trajetória realizada não representa diretamente a qualidade do controle, mas sim a liberdade de atuação do MPC de corrigir e atuar no sistema durante a execução.

Para fins comparativos, utiliza-se do erro quadrático médio (RMSE) como métrica principal de análise quantitativa porém, uma ausência de metodologias similares na literatura dificulta uma análise mais profunda de qualidade do controle.

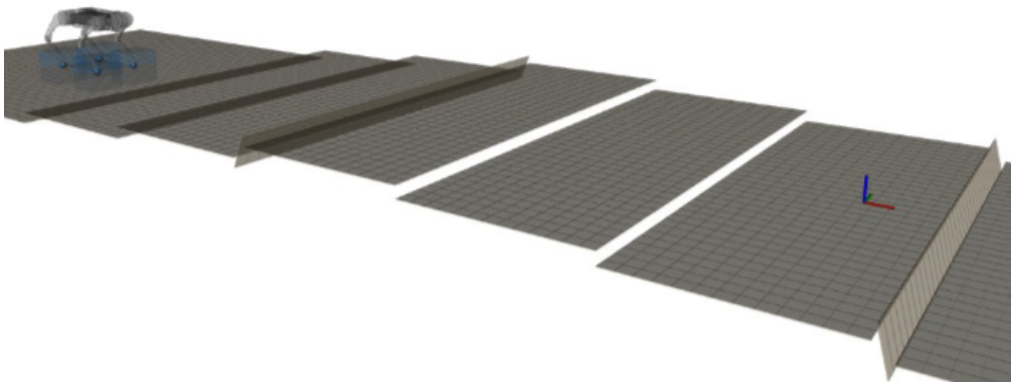
5.1
Terreno: Degraus Consecutivos

Iniciando com um terreno simples de forma a exemplificar a criação do terreno manualmente dentro do Grid2Towr, o primeiro terreno utilizado para testes é composto de 5 degraus (3 subindo e 2 descendo), posicionados a 0,8 metros de distância entre si e com altura de 0,05 metros cada, como mostra a Figura 5.1a.

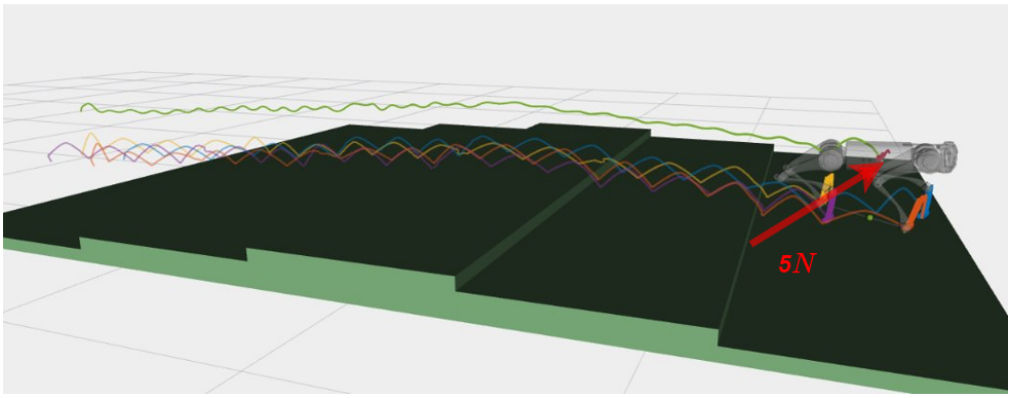
Primeiramente, cria-se um gridmap manualmente que represente o terreno dentro do TOWR utilizando a biblioteca Grid2Towr, o qual é fornecido como entrada para a otimização. A trajetória otimizada pelo TOWR recebeu como parâmetros de entrada um gait fixo de *standing trot*, um ponto de destino na direção longitudinal ao terreno (eixo X) de 4,6 metros e uma duração de 8 segundos (velocidade média de 0,575 m/s). O tempo de duração e a velocidade foram definidos de forma que o sistema conseguisse achar uma solução viável e dentro de um custo computacional aceitável.

Uma vez encontrada uma solução para a trajetória, os dados são armazenados em um rosbag para serem rastreados pela estrutura NMPC+WBC. O mesmo terreno foi exportado para o Gazebo para realizar as simulações. O

robô executa a trajetória com uma força constante de 5 N aplicada na direção perpendicular à base (direção Y) durante toda a simulação. A trajetória executada pelo robô é mostrada na Figura 5.1b. As principais linhas demarcadas referem-se à trajetória gerada pelo TOWR, a linha verde refere-se à trajetória da base e as linhas azul, amarela, laranja e roxa referem-se às trajetórias de cada perna.



(a) Terreno de degraus consecutivos utilizado para otimização de trajetória (TOWR).



(b) Terreno de degraus consecutivos no Gazebo e a trajetória de referência do robô pós rastreamento

Figura 5.1: Visualizações do terreno de degraus consecutivos

A performance do controlador de rastreamento pode ser analisada utilizando os gráficos de movimento linear da base (Figuras 5.2 e 5.3) e o movimento das patas na direção x (Figura 5.4).

O rastreamento da base na direção X e direção Z demonstrou resultados com o erro médio quadrático (RMSE) de 0,231 m and 0,039 m para X e Z, respectivamente. Para as pernas, há uma pequena disparidade entre o movimento de *swing* do TOWR e o medido pelo estimador de estados, que pode

ser atribuída à influência da perturbação externa, mostrando que o MPC tem a flexibilidade de otimizar as posições dos pés para se adaptar às perturbações.

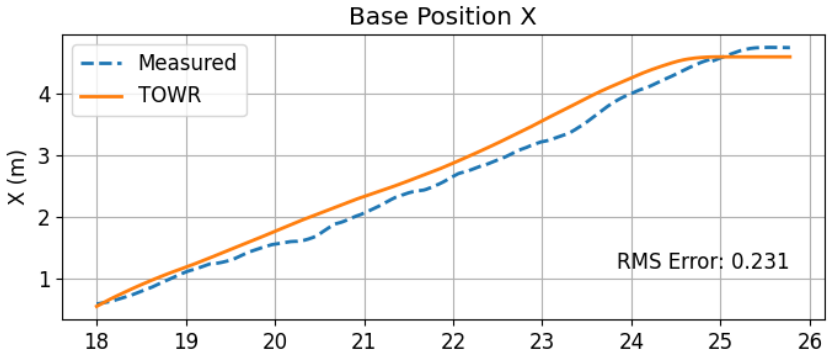


Figura 5.2: Degraus Consecutivos - Movimento Linear da Base em X

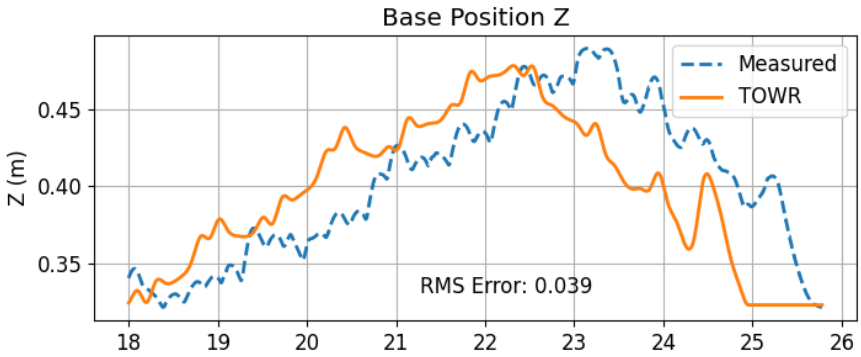


Figura 5.3: Degraus Consecutivos - Movimento Linear da Base em Z

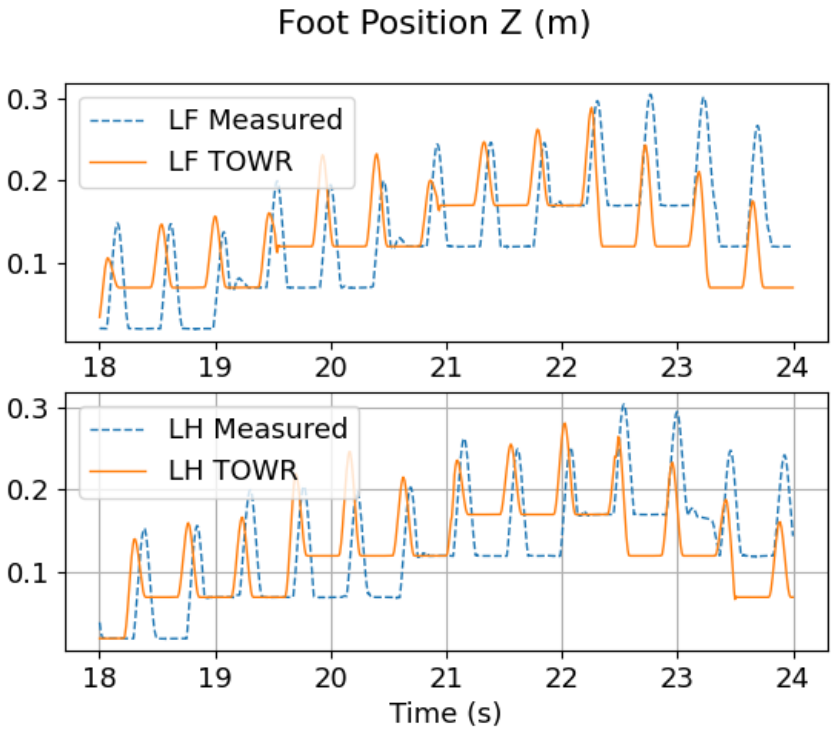


Figura 5.4: Degraus Consecutivos - Movimento das patas na direção Z

5.2 Terreno: Splines Variáveis

O segundo terreno utilizado para validar o framework, usufruindo das funcionalidades matemáticas do fluxograma de geração de terrenos comentado na Seção 4.3, foi criado utilizando múltiplas splines[1] com distâncias e alturas variáveis. Para este teste, o modelo do terreno foi primeiramente criado e adicionado no Gazebo. Na simulação, o robô foi equipado com uma câmera RGB-D Intel Realsense d435 e um plug-in foi incluído para gerar as imagens simuladas e a nuvem de pontos do ambiente. Utilizando a biblioteca de código aberto `Elevation Mapping` (FANKHAUSER; BLOESCH; HUTTER, 2018), a nuvem de pontos obtida pela câmera foi utilizada para gerar o mapa de elevação do terreno, mostrado na Figura 5.5 e utilizado como entrada para o TOWR.

A trajetória otimizada para esse terreno teve uma duração de 9 segundos e um ponto de destino de 4,6 metros na direção X e 0,3 metros na direção Y (0,51 m/s de velocidade média) com gait fixo de *standing trot*. Depois que uma solução de trajetória foi encontrada pelo TOWR (Figura 5.6a), os dados da trajetória são armazenados em uma rosbag para serem lidos pelo Towr2LeggedControl. O robô também executa o rastreamento com uma força

[1]Matematicamente, uma spline é uma função definida precisamente por polinômios

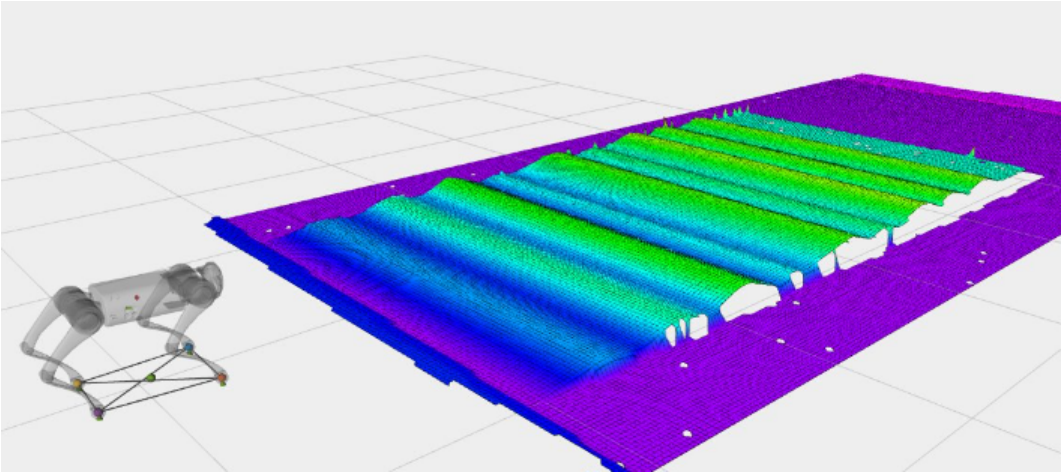
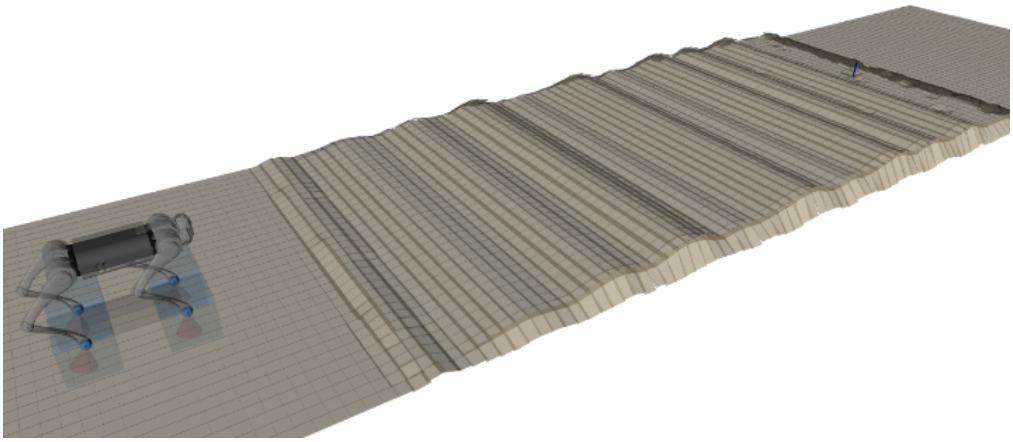
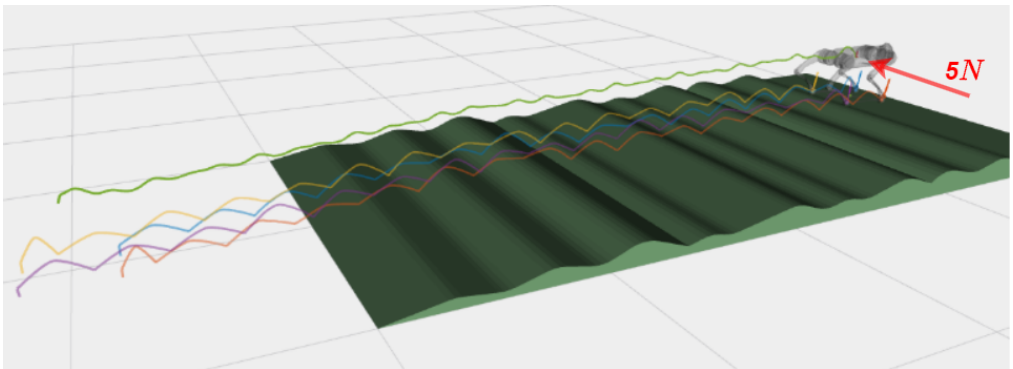


Figura 5.5: Terreno de Splines - Geração do gridmap

constante de 5 N aplicada na direção perpendicular à trajetória (direção Y), conforme mostrado na Figura 5.6b.



(a) Terreno de splines, visualização do terreno equivalente analítico no TOWR.



(b) Terreno de splines no Gazebo e a trajetória de referência do robô pós rastreamento

Figura 5.6: Visualizações do terreno de splines

Os resultados do rastreamento são mostrados para as posições da base e dos pés nas Figuras 5.7, 5.8 e 5.9. O rastreamento da base em X e Z demonstrou

resultados de RMSE de 0,204 m e 0,021 m, respectivamente, mesmo com uma trajetória que oscila de forma mais significativa. Para as pernas, foi observada uma disparidade mais acentuada nos resultados, que pode ser atribuída à maior necessidade de o MPC estabilizar o robô nesse terreno oscilatório com a presença de uma força lateral.

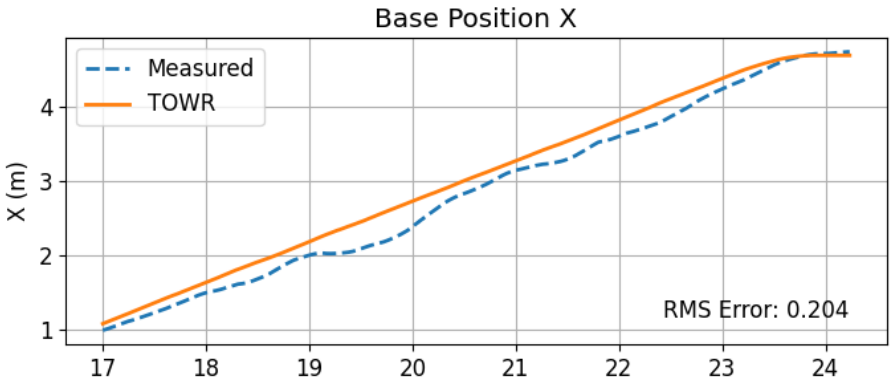


Figura 5.7: Terreno de Splines - Movimento Linear da Base em X

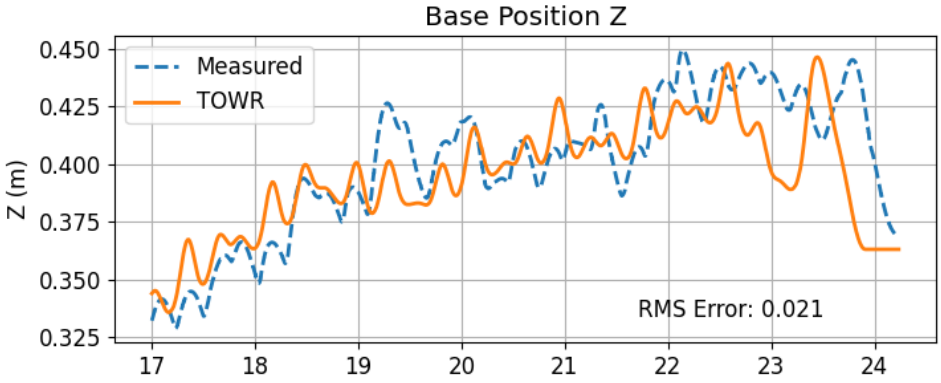


Figura 5.8: Terreno de Splines - Movimento Linear da Base em Z

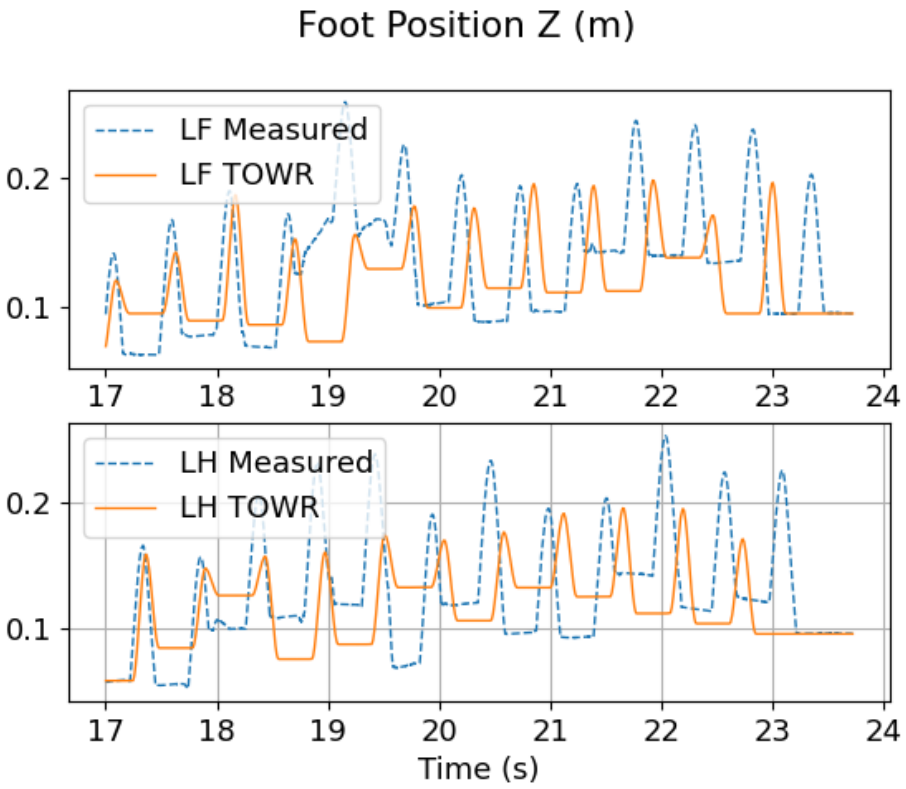


Figura 5.9: Terreno de Splines - Posição das patas na direção Z

5.3 Terreno Assimétrico

Neste exemplo, para avaliar a robustez do framework e a funcionalidade da ferramenta de otimização de gait, foi criado um terreno mais complexo, assimétrico entre os lados esquerdo e direito. O lado esquerdo é composto de 6 degraus (3 degraus subindo e 3 degraus descendo) colocados a 0,8 metros de distância com 0,05 metros de altura cada, enquanto o terreno no lado direito é composto de 11 degraus, variando entre subidas e descidas, com 0,2 a 0,4 metros de distância e alturas de até 0,06 metros cada.

Similar ao processo da Seção 5.2, o mapa de elevação do terreno (Figura 5.10) foi utilizado como input para o Grid2Towr e utilizado na otimização. Para esse terreno, a trajetória otimizada foi solicitada com duração de 11 segundos, um ponto de destino de 4,7 m em X e 0,3 m em Y (0,43 m/s de velocidade média) e com otimização do gait. O gait resultante está representado pelo diagrama da Figura 5.12. A trajetória otimizada computada pelo TOWR foi rastreada na simulação com uma força constante de 5 N aplicada na direção perpendicular à trajetória (direção y), como mostra a Figura 5.11b. É importante destacar que sem a otimização do gait, o TOWR não conseguiu encontrar uma solução viável para este terreno, mesmo variando os outros

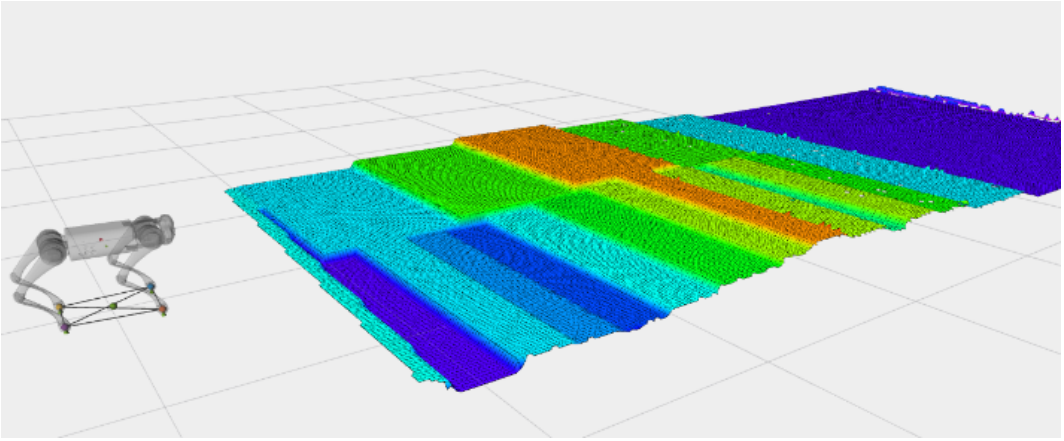
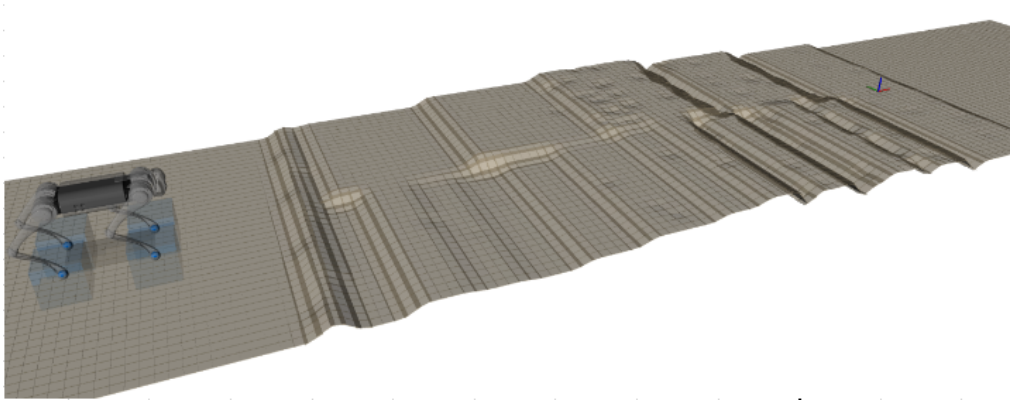
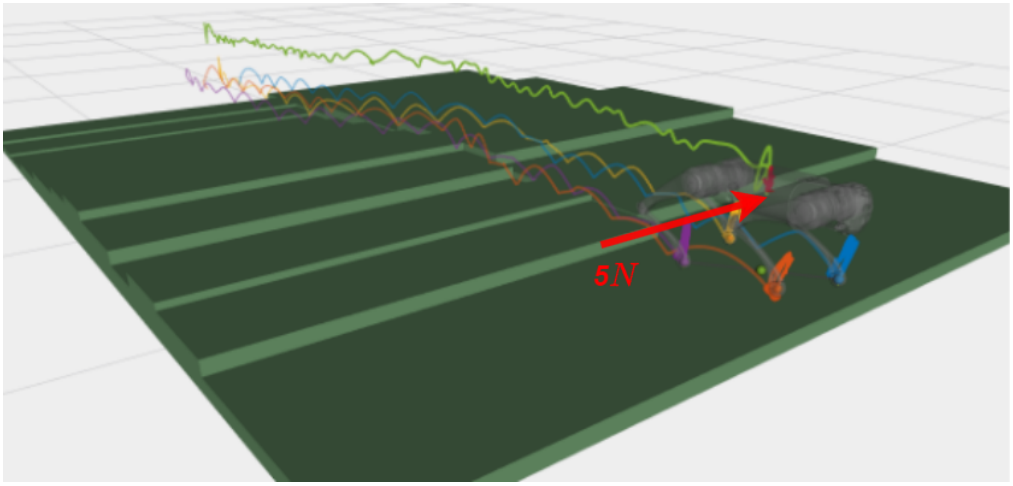


Figura 5.10: Terreno Assimétrico - Geração do gridmap

parâmetros de entrada, reforçando a importância desta ferramenta.



(a) Terreno assimétrico, visualização do terreno equivalente analítico no TOWR.



(b) Terreno assimétrico no Gazebo e a trajetória de referência do robô pós rastreamento

Figura 5.11: Visualizações do terreno assimétrico

As Figuras 5.13, 5.14 e 5.15 mostram os resultados do rastreamento da trajetória obtida pelo TOWR. A base em X foi rastreada com um resultado

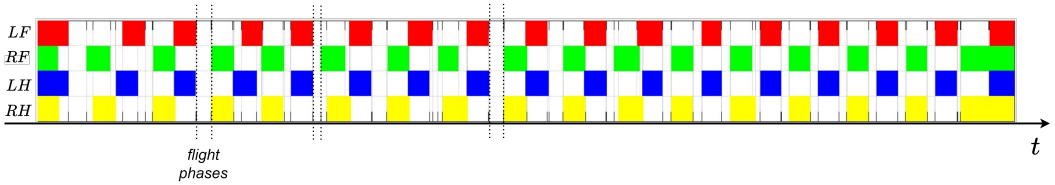


Figura 5.12: Diagrama de phases para o gait otimizado no terreno assimétrico

de RMSE de 0,221 m. Devido à maior complexidade do terreno, a base oscila consideravelmente no eixo Z, dificultando o rastreamento principalmente na segunda metade da trajetória, apesar do RMSE permanecer semelhante aos anteriores em 0,033 m. O processo de otimização do gait resultou em um movimento mais complexo que os anteriores, passado pelo módulo de conversão de gait seu resultado é representado na Figura 5.16. Essa complexidade é evidenciada pela maior dificuldade em rastrear a posição das pernas, conforme observado em determinados pontos do gráfico, podendo também ter sido influenciado pelos parâmetros fixos de *swing* da Seção 4.5.

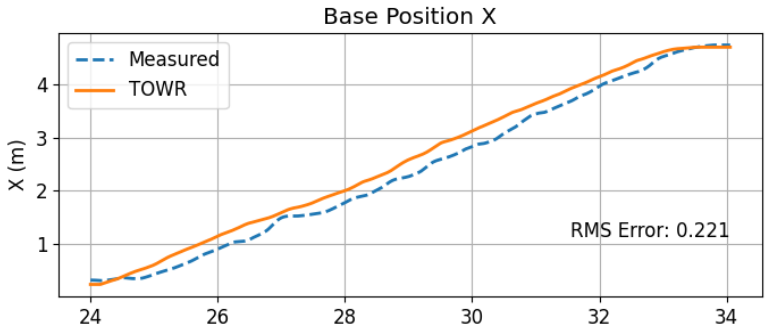


Figura 5.13: Terreno assimétrico - Movimento linear da base em X

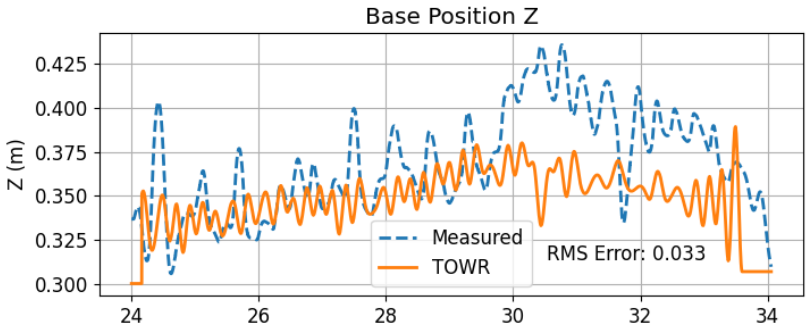


Figura 5.14: Terreno assimétrico - Movimento linear da base em Z

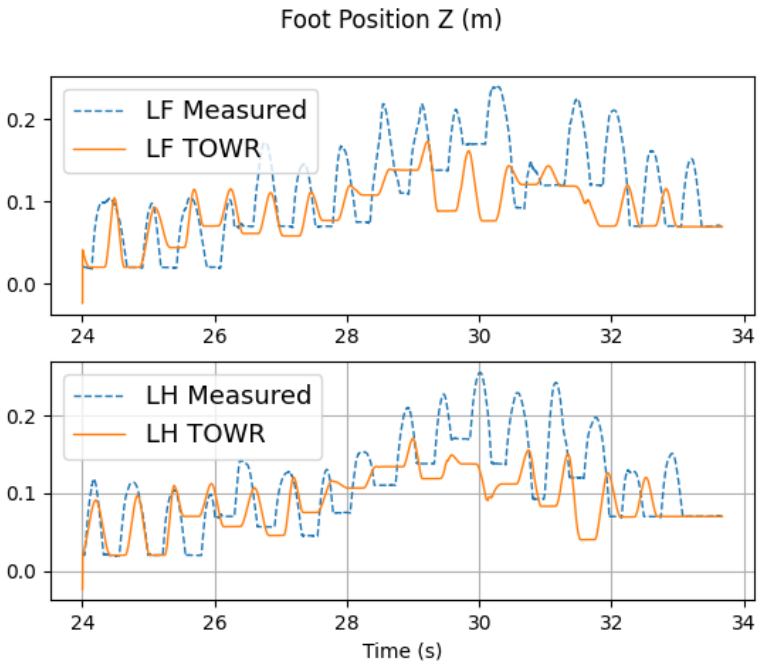


Figura 5.15: Terreno assimétrico - Movimento das patas na direção Z

5.4 Compilado de Resultados

Resumindo os resultados obtidos neste capítulo, a Tabela 5.1 destaca as principais informações quantitativas referentes aos testes realizados. É possível concluir que, apesar da dificuldade de realizar o rastreamento devido a perturbações externas e trajetórias complexas, o robô chega ao ponto alvo dentro do tempo desejado em todos os casos cumprindo seu objetivo principal.

Nota-se em alguns momentos durante as simulações que, ao perder a estabilidade momentaneamente devido a algum mal posicionamento de pata ou devido a atuação da força externa, há uma discrepância entre as trajetórias de referência e as reais. Nesses casos, o MPC consegue atuar dinamicamente no sistema reposicionado o robô de forma a retornar à trajetória de referência. Pode-se observar este comportamento nos gráficos do terreno de splines aos 22 segundos e nos gráficos do terreno assimétrico aos 30 segundos.

De forma geral, apesar do robô completar as trajetórias lineares da base no eixo x com RMSE máximo de 0.231 m, existe espaço para melhorias. As trajetórias de referência otimizadas pelo TOWR apresentam uma oscilação não desprezível no eixo Z agregando complexidade ao movimento, como pode ser visto nos três gráficos de movimento linear da base no eixo Z. Novas trajetórias que variem a coordenada Z de forma mais suave melhorariam

Tempo de Troca	*Patas em Contato*
0.000	STANCE
0.208	LF_LH_RH
0.217	LF_RH
0.315	LF
0.316	FLY
0.500	RF
0.566	RF_LH
0.746	LH
0.799	LH_RH
0.801	RH
0.868	LF_RH
1.028	LF
1.100	FLY
1.175	LH
1.182	RF_LH
1.394	LF_RF_LH
1.395	STANCE
1.409	LF_RF_RH
1.409	LF_RH
1.618	LF
1.624	FLY
1.777	LH
1.782	RF_LH
1.996	RF_LH_RH
2.007	RF_RH
2.009	RH
2.088	LF_RH
2.243	LF
2.290	LF_LH
2.292	LF_RF_LH
2.305	RF_LH
2.516	RF
2.525	FLY
2.593	LF
2.593	LF_RH
2.819	RH
2.820	FLY
2.900	RF
2.960	RF_LH
3.148	LH
3.191	LF_LH
3.191	LF_LH_RH

Tempo de Troca	*Patas em Contato*
3.206	LF_RH
3.417	LF
3.423	FLY
3.575	LH
3.582	RF_LH
3.792	LF_RF_LH
3.792	STANCE
3.807	LF_LH_RH
3.808	LF_RH
4.014	LF
4.043	FLY
4.100	RF
4.145	RF_LH
4.311	LH
4.392	LF_LH
4.394	LF_LH_RH
4.404	LF_RH
4.617	RH
4.623	FLY
4.773	RF
4.774	RF_LH
4.996	LF_RF_LH
4.997	STANCE
5.008	LF_RF_RH
5.011	LF_RH
5.218	RH
5.234	FLY
5.381	LH
5.382	RF_LH
5.594	LF_RF_LH
5.595	STANCE
5.607	LF_RF_RH
5.609	LF_RH
5.821	RH
5.827	FLY
5.900	RF
5.948	RF_LH
6.137	LF_RF_LH
6.166	LF_LH
6.193	LF_LH_RH
6.202	LF_RH
6.401	LF

Tempo de Troca	*Patas em Contato*
6.401	FLY
6.486	LH
6.491	RF_LH
6.709	RF
6.712	FLY
6.799	LF
6.799	LF_RH
7.002	RH
7.004	FLY
7.090	RF
7.092	RF_LH
7.319	LH
7.353	FLY
7.399	RH
7.399	LF_RH
7.613	LF
7.615	FLY
7.692	RF
7.694	RF_LH
7.908	RF
7.913	FLY
7.990	RH
7.993	LF_RH
8.208	LF
8.208	FLY
8.286	LH
8.290	RF_LH
8.506	LH
8.507	FLY
8.588	RH
8.589	LF_RH
8.808	LF
8.809	FLY
8.887	LH
8.895	RF_LH
9.107	LH
9.109	FLY
9.452	LF
9.739	LF_RH
9.745	STANCE
10.000	

Figura 5.16: Resultado da otimização de gait no formato do legged_control

potencialmente os resultados. Da mesma forma, para a trajetória das pernas, verifica-se em alguns momentos um mal posicionamento das patas obrigando ao controlador reposicionar o robô, o que afeta consideravelmente o rastreamento. Um aprimoramento na resolução do gridmap gerado, melhorando a percepção das quinas, por exemplo, pode diminuir estes efeitos. Além disso, apesar da funcionalidade de otimização de gait ter sido aplicada com sucesso, o rastreamento das trajetórias de *swing* pode ser aperfeiçoado. Os parâmetros de controle de configuração de *swing* comentados na Seção 4.5 afetam diretamente o comportamento das pernas, a altura fixa de *swing* pode restringir trajetórias otimizadas onde a altura do gait varia, por exemplo. Neste caso, o ideal é alterar a formulação do MPC para gerar as trajetórias de referência de swing

Terreno	Degrau Consecutivo	Splines	Assimétrico
Ponto alvo (x,y) [m]	(4.60,0.0)	(4.6,0.30)	(4.7,0.30)
Pos. final da base (x,y) [m]	(4.62,0.0)	(4.62,0.32)	(4.71,0.29)
Tempo desejado [s]	8.0	9.0	10.0
Tempo até o ponto final [s]	8.11	9.13	10.08
Vel. Méd. [m/s]	0.575	0.51	0.47
Gait	*standing trot*	*standing trot*	gait otimizado
RMSE da Base(x,y) [m]	(0.231,0.039)	(0.204,0.021)	(0.221,0.033)

Tabela 5.1: Resumo dos resultados obtidos para os três terrenos apresentados

com base nos pontos gerados pelo TOWR.

Como comentado na Seção 4.2 a tarefa de rastreamento e a tarefa de gait são independentes e para implementar o gait otimizado do TOWR, uma função que monta um cronograma de posicionamento das patas ao longo de toda a trajetória foi criada. Com isso, qualquer atraso no sistema, seja na publicação das trajetórias (em relação a inicialização do gait) ou no próprio rastreamento, afeta completamente o resultado além de, evidentemente, a posição inicial de cada pata.

A Tabela 5.2 destaca o custo computacional representado pelo tempo total utilizado e o número de iterações necessárias para o TOWR encontrar uma solução viável para o problema de otimização. Os dados foram obtidos utilizando uma máquina virtual de Linux com Ubuntu 20.04, 12 Gb de memória RAM e 6 núcleos de processamento. Apesar do tempo de processamento estar diretamente relacionado com o poder computacional, é interessante analisar a diferença relativa entre as três soluções. Para o primeiro terreno, geometricamente mais simples, utilizou-se o gridmap gerado manualmente pela Grid2Towr e para a otimização ser concluída foi necessário um tempo de 40 segundos. Para o terreno de splines, utilizou-se do mapa de elevação gerado pela câmera para a otimização, e foi necessário um tempo de 70 segundos. Para o terreno de assimétrico, o geometricamente mais complexo incluindo o mapa de elevação e otimização de gait, foi necessário um tempo de 126 segundos. Alguns fatores que afetam diretamente o custo computacional são a resolução do gridmap utilizado, o tempo total de trajetória, a quantidade de restrições utilizadas e evidentemente a complexidade do terreno.

Terreno	Degrau Consecutivo	Splines	Assimétrico
Tempo total (s)	40	70	126
Número de Iterações	21	40	78

Tabela 5.2: Custo computacional do processo de otimização de trajetórias

6
Conclusão e Trabalhos Futuros

O trabalho apresentado propõem uma integração desenvolvida em C++ dentro do ambiente ROS entre a biblioteca de planejamento off-line de trajetórias TOWR e a biblioteca de controle de locomoção baseada em NMPC+WBC, legged_control. Os resultados discutidos na Capítulo 5 demonstram a aplicabilidade e performance da integração entre os dois sistemas, validando dinamicamente em simulações físicas as trajetórias ótimas encontradas na etapa de planejamento. Utilizando o robô Go1, o framework foi testado utilizando três terrenos diferentes, onde foi possível avaliar alguns pontos positivos e negativos da ferramenta.

Com o desenvolvimento da biblioteca Grid2Towr, foi possível estender a aplicabilidade do TOWR. Por padrão, a biblioteca utiliza de terrenos definidos por funções analíticas, dificultando a implementação de cenários mais genéricos. A proposta apresentada viabiliza terrenos mais complexos utilizando das ferramentas da biblioteca de mapeamento `grid_map`. Com isso, é possível, por exemplo, aplicar ferramentas de percepção com câmeras de profundidade e, através de mapas de elevação, realizar o planejamento de trajetórias otimizadas em terrenos mais genéricos como demonstrado nos testes realizados nesse trabalho.

Com o desenvolvimento da biblioteca Towr2Gridmap, foi possível estender a aplicabilidade do `legged_control`. Por padrão a biblioteca interpola uma trajetória simplificada utilizando o ponto de destino e o ponto inicial da base do robô. A proposta apresentada utiliza a trajetória otimizada gerada pelo TOWR como entrada para o módulo de controle e simulação em ambiente físico, validando as trajetórias obtidas na otimização frente a terrenos diversos e perturbações externas.

Como analisado na Seção 5.4, o robô conseguiu chegar ao ponto de destino no tempo desejado nos três cenários apresentados validando o framework posposto. Dito isso, foi possível observar diversos pontos de melhoria para aumentar a aplicabilidade da ferramenta. Aumentar a restrição de movimento do centro de massa no planejamento de trajetórias para diminuir a oscilação da base. Aumentar a resolução dos mapas de elevação para melhorar a precisão dos gridmaps gerados. Aprofundar a implementação do módulo de rastreamento das patas para aumentar a fidedignidade à trajetória de referência. Aprofundar na implementação no módulo de gait para reduzir os problemas de sincronismo entre o rastreamento de trajetórias e a inicialização do gait.

Como extensão desta pesquisa, algumas novas abordagens podem ser seguidas como aplicar o framework em um robô real utilizando das funcionalidades da `legged_control` e avaliar sua robustez, expandir as formas de terrenos possível como entradas para o TOWR (e.g. Octomap (HORNUNG et al., 2013) e ampliar a Towr2LeggedControl para receber entradas de diferentes planejadores de trajetórias, incluindo planejadores online. Com o framework proposto, espera-se aumentar as possibilidades de utilização de ambas as bibliotecas abrindo espaço para novos desenvolvimentos na área.

7
Referências bibliográficas

BELLICOSO, C. D. et al. Perception-less terrain adaptation through whole body control and hierarchical optimization. In: **2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)**. 2016. p. 558–564.

BELLICOSO, C. D. et al. Dynamic locomotion and whole-body control for quadrupedal robots. In: IEEE. **2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. 2017. p. 3359–3365.

BJELONIC, M. et al. Offline motion libraries and online mpc for advanced mobility skills. **The International Journal of Robotics Research**, v. 41, n. 9-10, p. 903–924, 2022.

BOSWORTH, W.; KIM, S.; HOGAN, N. The mit super mini cheetah: A small, low-cost quadrupedal robot for dynamic locomotion. In: IEEE. **2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)**. 2015. p. 1–8.

CARPENTIER, J. et al. The pinocchio c++ library – a fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In: **IEEE International Symposium on System Integrations (SII)**. 2019.

CHITTA, S. et al. ros_control: A generic and simple control framework for ros. **The Journal of Open Source Software**, 2017. Disponível em: <http://www.theoj.org/joss-papers/joss.00456/10.21105.joss.00456.pdf>.

CHO, J.; PARK, J. H. Model predictive control of running biped robot. **Applied Sciences**, MDPI, v. 12, n. 21, p. 11183, 2022.

COMMUNITY, B. O. **Blender - a 3D modelling and rendering package**. Stichting Blender Foundation, Amsterdam, 2018. Disponível em: <http://www.blender.org>.

DAI, H.; VALENZUELA, A.; TEDRAKE, R. Whole-body motion planning with centroidal dynamics and full kinematics. In: IEEE. **2014 IEEE-RAS International Conference on Humanoid Robots**. 2014. p. 295–302.

DING, Y. et al. Representation-free model predictive control for dynamic motions in quadrupeds. **IEEE Transactions on Robotics**, IEEE, v. 37, n. 4, p. 1154–1171, 2021.

DYNAMICS, B. **ATLAS**. 2016. <https://bostondynamics.com/atlas/>. Acessado em: 29-05-2024.

DYNAMICS, B. **Spot - The Agile Mobile Robot**. 2024. Acessado em: 29-05-2024. Disponível em: <https://bostondynamics.com/products/spot/>.

FAHMI, S. et al. Passive whole-body control for quadruped robots: Experimental validation over challenging terrain. **IEEE Robotics and Automation Letters**, IEEE, v. 4, n. 3, p. 2553–2560, 2019.

FANKHAUSER, P.; BLOESCH, M.; HUTTER, M. Probabilistic terrain mapping for mobile robots with uncertain localization. **IEEE Robotics and Automation Letters (RA-L)**, v. 3, n. 4, p. 3019–3026, 2018.

FANKHAUSER, P.; HUTTER, M. A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation. In: KOUBAA, A. (Ed.). **Robot Operating System (ROS) – The Complete Reference (Volume 1)**. Springer, 2016. cap. 5. ISBN 978-3-319-26052-5. Disponível em: <http://www.springer.com/de/book/9783319260525>.

FANKHAUSER, P.; HUTTER, M. A universal grid map library: Implementation and use case for rough terrain navigation. **Robot Operating System (ROS) The Complete Reference (Volume 1)**, Springer, p. 99–120, 2016.

FARSHIDIAN, F. et al. An efficient optimal planning and control framework for quadrupedal locomotion. In: **2017 IEEE International Conference on Robotics and Automation (ICRA)**. 2017. p. 93–100.

FARSHIDIAN, F. et al. **OCS2: An open source library for Optimal Control of Switched Systems**. 2017. [Online]. Disponível em: <https://github.com/leggedrobotics/ocs2>.

FEATHERSTONE, R. **Rigid body dynamics algorithms**. : Springer, 2014.

FRISON, G.; DIEHL, M. **HPIPM: a high-performance quadratic programming framework for model predictive control**. 2020.

GAERTNER, M. et al. Collision-free mpc for legged robots in static and dynamic scenes. In: IEEE. **2021 IEEE International Conference on Robotics and Automation (ICRA)**. 2021. p. 8266–8272.

GRANDIA, R. et al. **Perceptive Locomotion through Nonlinear Model Predictive Control**. 2022.

GRANDIA, R. et al. Multi-layered safety for legged robots via control barrier functions and model predictive control. In: IEEE. **2021 IEEE International Conference on Robotics and Automation (ICRA)**. 2021. p. 8352–8358.

HEREID, A.; AMES, A. D. Frost: Fast robot optimization and simulation toolkit. In: IEEE/RSJ. **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. Vancouver, BC, Canada, 2017.

HONDA. **Asimo**. 2000. <https://global.honda/innovation/robotics/ASIMO/history.html>. Acessado em: 29-05-2024.

HONG, S.; KIM, J.-H.; PARK, H.-W. Real-time constrained nonlinear model predictive control on so(3) for dynamic legged locomotion. In: **2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. 2020. p. 3982–3989.

HORNUNG, A. et al. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. **Autonomous Robots**, 2013. Software Disnponível em <http://octomap.github.com>. Disponível em: <http://octomap.github.com>.

HUTTER, M. et al. Anymal-a highly mobile and dynamic quadrupedal robot. In: IEEE. **2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)**. 2016. p. 38–44.

HUTTER, M.; SIEGWART; STASTNY, R. **Robot dynamics lecture notes**. 2016. <https://ethz.ch/content/dam/ethz/special-interest/mavt/robotics-n-intelligent-systems/rsl-dam/documents/RobotDynamics2016/7-leggedrobots.pdf>. Acessado em: 2024-05-18.

HUTTER, M. et al. **Robot dynamics lecture notes**. 2017. <https://ethz.ch/content/dam/ethz/special-interest/mavt/robotics-n-intelligent-systems/rsl-dam/documents/RobotDynamics2017/RD_HS2017script.pdf>. Acessado em: 2024-05-18.

HWANGBO, J. M. **Learning Walking Control with Reinforcement Learning and Introduction to RaiSim Physics Engine**. 2021. <https://www.youtube.com/watch?v=RoTYz_cqK7w>. Acessado em: 2024-05-18.

JELAVIC, E.; FARSHIDIAN, F.; HUTTER, M. Combined sampling and optimization based planning for legged-wheeled robots. In: **2021 IEEE International Conference on Robotics and Automation (ICRA)**. 2021. p. 8366–8372.

KAM, H. R. et al. Rviz: a toolkit for real domain data visualization. **Telecommun. Syst.**, Kluwer Academic Publishers, USA, v. 60, n. 2, p. 337–345, oct 2015. ISSN 1018-4864.

KATAYAMA, M. M. S.; TAZAKI, Y. Model predictive control of legged and humanoid robots: models and algorithms. **Advanced Robotics**, Taylor & Francis, v. 37, n. 5, p. 298–315, 2023.

KOENIG, K.; HOWARD, A. **Design and analysis of simulation software for autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems***. 2004. <https://gazebosim.org/home>. Acessado em: 2024-05-06.

KOTTEGE, N.; SENTIS, L.; KANOULAS, D. Editorial: Towards real-world deployment of legged robots. **Frontiers in Robotics and AI**, v. 8, 2022. ISSN 2296-9144. Disponível em: <https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2021.829403>.

LIAO, Q. **Legged Control Library**. 2022. <https://github.com/qiayuanl/legged_control>. Acessado em: 2024-05-06.

MA, W.-L.; HAMED, K. A.; AMES, A. D. First steps towards full model based motion planning and control of quadrupeds: A hybrid zero dynamics approach. In: IEEE. **2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. 2019. p. 5498–5503.

MACIEJOWSKI, J. **Predictive Control with Constraints**. Prentice Hall, 2000. ISBN 0201398230,9780201398236. Disponível em: <http://gen.lib.rus.ec/book/index.php?md5=acd9a6158ed2bd9d54a4a0feaa6d3d4c>.

MASTALLI, C. et al. Motion planning for quadrupedal locomotion: Coupled planning, terrain mapping, and whole-body control. **IEEE Transactions on Robotics**, IEEE, v. 36, n. 6, p. 1635–1648, 2020.

MEDEIROS, V. S. **Trajectory Optimization for Hybrid Wheeled-Legged Robots in Challenging Terrain**. Tese (PhD thesis) — Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, Brazil, July 2020. Disnponível em <https://www.maxwell.vrac.puc-rio.br/colecao.php?strSecao=resultado&nrSeq=50271@2>.

MEDEIROS, V. S. et al. Trajectory optimization for wheeled-legged quadrupedal robots driving in challenging terrain. **IEEE Robotics and Automation Letters**, v. 5, n. 3, p. 4172–4179, 2020.

MU, X.; SHAO, S.; ZHANG, D. Adaptive locomotion control of sixteen-legged robot based on deep reinforcement learning. In: IEEE. **2021 IEEE International Conference on Robotics and Biomimetics (ROBIO)**. 2021. p. 992–997.

NEUNERT, M. et al. Whole-body nonlinear model predictive control through contacts for quadrupeds. **CoRR**, abs/1712.02889, 2017. Disponível em: <http://arxiv.org/abs/1712.02889>.

NORBY, J.; JOHNSON, A. M. Fast global motion planning for dynamic legged robots. In: IEEE. **2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. 2020. p. 3829–3836.

NORBY, J. et al. Quad-SDK: Full stack software framework for agile quadrupedal locomotion. In: **ICRA Workshop on Legged Robots**. 2022.

QUIGLEY, M. et al. Ros: an open-source robot operating system. In: **Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics**. Kobe, Japan: , 2009.

SCHULMAN, J.; LAB the R. L. **trajopt: Trajectory Optimization for Motion Planning**. 2013. Acessado em: 29-05-2024. Disponível em: <https://github.com/joschu/trajopt>.

SEMINI, C. et al. Design of hyq–a hydraulically and electrically actuated quadruped robot. **Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering**, SAGE Publications Sage UK: London, England, v. 225, n. 6, p. 831–849, 2011.

SEOK, S. et al. Design principles for highly efficient quadrupeds and implementation on the mit cheetah robot. In: IEEE. **2013 IEEE International Conference on Robotics and Automation**. 2013. p. 3307–3312.

SHARBAFI, M. A.; SEYFARTH, A. **Bioinspired Legged Locomotion**. Springer London, 2010. (Advanced Textbooks in Control and Signal Processing). ISBN 9780128037669. Disponível em: <https://books.google.com.br/books?id=jPCAFmE-logC>.

SICILIANO, B. et al. **Robotics: Modelling, Planning and Control**. Springer London, 2010. (Advanced Textbooks in Control and Signal Processing).

ISBN 9781846286414. Disponível em: <https://books.google.com.br/books?id=jPCAFmE-logC>.

SUN, H.; WANG, C.-h.; AN, H. Real-time replanning and control of quadruped robots for blind locomotion on uneven terrain. In: **2020 3rd International Conference on Unmanned Systems (ICUS)**. 2020. p. 401–406.

SYSTèMES, D. **Solidworks**. 2023. Disponível em: <https://www.solidworks.com/>.

TEDRAKE, R.; TEAM the D. D. **Drake: Model-based design and verification for robotics**. 2019. Acessado em: 29-05-2024. Disponível em: <https://drake.mit.edu>.

UNITREE. **Go1**. 2024. <https://www.unitree.com/go1/>. Acessado em: 03-20-2024.

UNIVERSITY, M. in Motion Laboratory at N. Y. **Kino-dynamic Trajectory Optimization for Multiped Robots**. 2020. <https://github.com/machines-in-motion/kino_dynamic_opt>. Acessado em: 29-05-2024.

VUKOBRATOVIĆ, M.; BOROVAC, B. Zero-moment point—thirty five years of its life. **International journal of humanoid robotics**, World Scientific, v. 1, n. 01, p. 157–173, 2004.

WANG, Y. et al. A static gait generation for quadruped robots with optimized walking speed. In: **2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)**. 2020. p. 1899–1906.

WINKLER, A. **Optimization-based motion planning for legged robots**. Tese (Doutorado) — ETH, 05 2018.

WINKLER, A. **TOWR–An open-source Trajecetory Optimizer for Legged Robots in C++**. 2018.

WINKLER, A. W. et al. Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. **IEEE Robotics and Automation Letters**, IEEE, v. 3, n. 3, p. 1560–1567, 2018.

ZHANG, D. et al. Design and gait optimization of quadruped robot based on energy conservation. In: IEEE. **2020 3rd International Conference on Control and Robots (ICCR)**. 2020. p. 112–119.

ZHANG, Z. et al. Efficient motion planning based on kinodynamic model for quadruped robots following persons in confined spaces. **IEEE/ASME Transactions on Mechatronics**, IEEE, v. 26, n. 4, p. 1997–2006, 2021.