基于模型预测控制的四足机器人仿真

李奥齐 郭骏德 江轶豪 田丰 杨博文

摘要：本项目复现了一种四足机器人模型预测控制方法，控制算法的核心是将机器人的动力学问题简化为二次型优化问题，并将求解二次凸优化模型得到的地面反作用力映射为各个关节的控制力矩。本项目选择麻省理工学院开发的 Mini Cheetah 作为四足机器人模型、Pybullet 作为仿真环境，并自行设计了机器人足端的参考轨迹，作为进一步研究控制方法的基础。本项目首先对机器人进行完全的位置控制以验证足端轨迹，然后独立完成了 MPC 控制代码的编写，在进一步修改权重参数和调整步态之后，实现了四足机器人 MPC 的仿真，并使机器人具有一定的抗干扰能力。

关键词： 行走机器人；四足机器人；模型预测控制；二次型优化

The Simulation of a Quadruped Robot Based on Model Predictive Control

LI Aoqi GUO Junde JIANG Yihao TIAN Feng YANG Bowen

Abstract: Our project reproduced a predictive control method for a quadruped robot model. The core of the control algorithm is to reduce the robot dynamics problem to a quadratic optimization problem, and to map the ground reaction forces obtained by solving the quadratic convex optimization model to the control moments of each joint. In this project, the Mini Cheetah developed by MIT was chosen as the quadruped robot model, Pybullet as the simulation environment, and the reference trajectory of the robot's foot end was designed by ourselves as the basis for further research on the control method. In this project, we first performed full position control of the robot to verify the foot end trajectory, then independently wrote the MPC control code, and after further modifying the weight parameters and adjusting the gait, we realized the simulation of the MPC of the quadruped robot and make the robot have a certain extent of anti-interference ability.

Key words: Walking robot; Quadruped robot; Model predictive control; Quadratic optimization

0 研究动机

目前，常见的行走机器人以双足、四足、六足较为常见。其中，四足机器人机构相对简单且灵活，承载能力强、稳定性好，在抢险救灾、探险及军事等各方面有很好的应用前景，其机构设计和控制方法研究一直受到国内外的重视[1]。四足行走机器人的研究从 20 世纪 60 年代开始起步，随着控制技术和计算机技术的发展，20 世纪 80 年代现代四足机器人的研究进入广泛开展阶段[2]。截至今天，科学家们对四足机器人技术研究做了大量工作并取得了一系列关键突破，使四足机器人在速度、稳定性、灵活性、智能性等方面的性能不断提高。2015 年左右，麻省理工学院的仿生机器人实验室（Biomimetic Robotics Lab）发表了关于四足机器人 Mini Cheetah 的研究，与其他四足机器人相比，其体型小巧、行动灵活、速度相当快，研究团队基于它完成了后空翻等高难度动作，其高速奔跑的速度可以达到 3.7 m/s，是目前 10 kg 以下的四足机器人中速度最快的。[3]

常见的四足机器人控制方法包括 PID 控制、基于零力矩点（Zero Moment Point，ZMP）稳定的控制方法、全身控制（Whole Body Control，WBC）方法、强化学习，以及本项目重点研究的模型预测控制（Model Predictive Control，MPC）方法。MPC 的核心是用求解优化问题的方法来解决控制问题，但是与最优估计不同，MPC 仅仅考虑未来的一个或几个时间步，这样虽然牺牲了最优性，但是减小了运算的复杂程度。与传统控制方法相比，MPC 具有善于处理多输入-多输出系统、可以处理约束条件、考虑未来情况而非仅根据当前情况进行决策的优势。凭借这些优势，MPC 已经在自动驾驶、机器人控制等领域中被广泛应用，并占据了重要地位。[4]

基于上述背景，我们小组对四足机器人和 MPC 等控制方法产生了极大的兴趣，我们希望能够通过自己的学习和研究，独立完成一个 MPC 框架的搭建，并用其实现对四足机器人的控制。我们计划通过 MPC 实现的任务是可以让机器人具有较强的抗干扰性，即当机器人受到外力或地形干扰时，MPC

可以控制其做出反应以保持稳定，并在干扰消失之后回到参考步态。我们小组希望通过本项目加深对机器人动力学模型、机器人控制、关节轨迹规划等本课程内容的理解，学习并掌握 MPC 控制方法。同时，本项目还能让我们提高文献分析和研究能力、数学建模能力，进一步学习现代控制理论和优化问题求解的方法。

1 模型建立

基于对文献[5]的阅读和分析，我们对四足机器人做了如下建模。

1.1 机器人动力学模型

相比于传统的基于多刚体系统建立的各运动关节高频反馈控制的动力学模型，为了实现四足机器人运动和动力学模型的简化，我们将整个四足机器人看成一个单刚体模型，但是又保留了它的动力学特征，这一点在下图控制系统的蓝色框中得到了体现，即在单刚体模型下进行 MPC 的控制，但是这并不会输出各关节的力矩，而是把控制问题分割成先求解地面反力，再通过各个机械臂的力雅可比矩阵求得关节所需要的力矩，如下图

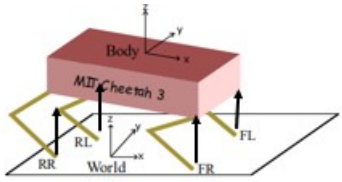

图 1 单刚体四足机器人模型（黑线箭头为地面反力）

其中，力雅可比矩阵可以通过电机位置姿态以及机械臂的力约束矩阵求得。同时 MPC 控制模块输出的地面反力是机体坐标系下的，需要对其进行坐标系转换，然后通过矩阵映射求得相应的电机扭矩，实现力控。

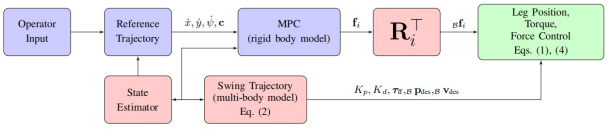

图 2 MPC 控制的四足机器人控制系统框图

首先是描述机器人状态的主要变量的选取。对于机器人自身运动状态的描述，姿态以及位置层面，是使用欧拉角（Roll, Pitch, Yaw）和质心位置 (x, y, z) 来描述，同时为了描述其运动信息层面，加入角速度以及机器人质心的运动速度。因此，在机器人状态描述层面，共有姿态角、质心位置、角速度、质心速度共 12 个量描述。首先是关于质心的牛顿动力学方程：

$$\ddot{\mathbf{p}} = \frac{\sum_{i=1}^{n} \mathbf{f}_i}{m} - \mathbf{g} \tag{1}$$

式(1)中，对于任何踩在地上的足式动物或者是足式机器人，他们所受到的外力包括重力和地面反力，通过对该整体进行受力分析，我们得到了加速度，即质心位置的二阶导数的等式。

对于角速度的动力学需要在一定程度上做近似处理。相比于 Yaw 偏航角，另外两个 Roll 翻滚角和 Pitch 俯仰角都作为小量作近似处理。对于欧拉角和旋转矩阵的处理，可以视作为通过 z 轴旋转 yaw 的角度，再通过当前坐标系的 y 轴旋转 pitch 的角度，再通过当前坐标系的 x 轴旋转 roll 的角度，从而得到欧拉角的旋转矩阵。通过将欧拉角坐标系下的单位向量进行逆变换，其中从欧拉角角速度映射到角速度的映射矩阵如下

$$\boldsymbol{\omega} = \begin{bmatrix} \cos\theta\cos\psi & -\sin\psi & 0 \\ \cos\theta\sin\psi & \cos\psi & 0 \\ -\sin\theta & 0 & 1 \end{bmatrix} \dot{\boldsymbol{\Theta}} \tag{2}$$

将式(2)求到的的映射矩阵求逆并作近似处理，从而得到关于从角速度到欧拉角角速度的映射关系

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \approx \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{\omega} = \mathbf{R}_z(\psi)\boldsymbol{\omega} \tag{3}$$

其中，$\begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T$ 是欧拉角角速度，$\boldsymbol{\omega}$ 为世界坐标系下的刚体转动的角速度。

欧拉公式的近似转换和空间惯量的张量在坐标系下的变换需要建立转动惯量与角速度的导数和地面反力产生的力矩的数学关系，并以此推导出角加速度

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{I}\boldsymbol{\omega}) = \sum_{i=1}^{n} \mathbf{r}_i \times \mathbf{f}_i = \sum_{i=1}^{n} \begin{bmatrix} 0 & -r_{zi} & r_{yi} \\ r_{zi} & 0 & -r_{xi} \\ -r_{yi} & r_{xi} & 0 \end{bmatrix} \mathbf{f}_i$$

$$= \mathbf{I}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{I}\boldsymbol{\omega}) \approx \mathbf{I}\dot{\boldsymbol{\omega}} \tag{4}$$

$$\Leftrightarrow \dot{\boldsymbol{\omega}} = \sum_{i=1}^{n} \mathbf{I}^{-1} \begin{bmatrix} 0 & -r_{zi} & r_{yi} \\ r_{zi} & 0 & -r_{xi} \\ -r_{yi} & r_{xi} & 0 \end{bmatrix} \mathbf{f}_i$$

由于角速度作为小量是可以被忽略，并且如果角速度这一项被添加，可能会导致求解的发散。同时，该式还需要将各个坐标系下的转动惯量转换为世界坐标系下的空间惯量

$$\hat{\mathbf{I}} = \mathbf{R}_z(\psi)_{\mathcal{B}}\mathbf{I}\mathbf{R}_z(\psi)^{\top} \tag{5}$$

1.2 状态空间模型

将式(1)～式（5）所得公式进行联立，从而得到状态空间模型

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\hat{\boldsymbol{\Theta}}\\ \hat{\mathbf{p}}\\ \hat{\boldsymbol{\omega}}\\ \hat{\dot{\mathbf{p}}}\end{bmatrix}=\begin{bmatrix}\mathbf{0}_3 & \mathbf{0}_3 & \mathbf{R}_z(\psi) & \mathbf{0}_3\\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{1}_3\\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{\Theta}}\\ \hat{\mathbf{p}}\\ \hat{\boldsymbol{\omega}}\\ \hat{\dot{\mathbf{p}}}\end{bmatrix}+\begin{bmatrix}\mathbf{0}_3 & \cdots & \mathbf{0}_3\\ \mathbf{0}_3 & \cdots & \mathbf{0}_3\\ \hat{\mathbf{I}}^{-1}[\mathbf{r}_1]_\times & \cdots & \hat{\mathbf{I}}^{-1}[\mathbf{r}_n]_\times\\ \mathbf{1}_3/m & \cdots & \mathbf{1}_3/m\end{bmatrix}\begin{bmatrix}\mathbf{f}_1\\ \vdots\\ \mathbf{f}_n\end{bmatrix}+\begin{bmatrix}0\\0\\0\\ \mathbf{g}\end{bmatrix} \tag{6}$$

进一步将式(6)进行离散化处理，我们可以得到下列离散化关系：

$$\begin{bmatrix}\hat{\boldsymbol{\Theta}}^{k+1}\\ \hat{\mathbf{p}}^{k+1}\\ \hat{\boldsymbol{\omega}}^{k+1}\\ \hat{\dot{\mathbf{p}}}^{k+1}\end{bmatrix}=\begin{bmatrix}\mathbf{1}_3 & \mathbf{0}_3 & \mathbf{R}_z(\psi) & \mathbf{0}_3\\ \mathbf{0}_3 & \mathbf{1}_3 & \mathbf{0}_3 & \mathbf{1}_3\Delta t\\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{1}_3 & \mathbf{0}_3\\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{1}_3\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{\Theta}}^{k}\\ \hat{\mathbf{p}}^{k}\\ \hat{\boldsymbol{\omega}}^{k}\\ \hat{\dot{\mathbf{p}}}^{k}\end{bmatrix}+\begin{bmatrix}\mathbf{0}_3 & \cdots & \mathbf{0}_3\\ \mathbf{0}_3 & \cdots & \mathbf{0}_3\\ \hat{\mathbf{I}}^{-1}[\mathbf{r}_1]_\times\Delta t & \cdots & \hat{\mathbf{I}}^{-1}[\mathbf{r}_n]_\times\Delta t\\ \mathbf{1}_3\Delta t/m & \cdots & \mathbf{1}_3\Delta t/m\end{bmatrix}\begin{bmatrix}\mathbf{f}_1\\ \vdots\\ \mathbf{f}_n\end{bmatrix}+\begin{bmatrix}0\\0\\0\\ \mathbf{g}\end{bmatrix} \tag{7}$$

将式(7)的重力加速度g转换到A矩阵中并简化表示，可以表示为

$$x(k+1)=A_k x(k)+B_k f(k) \tag{8}$$

1.3 MPC 模型

式(7)中的 $f(k)$ 是机器人足端受到的地面反力，也是该系统的输入量。MPC 的设计目标就是通过一个合适的地面反力，来达到一个最优的轨迹跟踪效果。这与 LQR 控制非常相似，但是相比于 LQR 系统，MPC 控制添加了一个给定预测时域的预测环节。对于该预测部分，有

$$\begin{bmatrix}x_{k+1}\\ x_{k+2}\\ x_{k+3}\\ \vdots\\ x_{k+h}\end{bmatrix}=A_{qp}x_k+B_{qp}\begin{bmatrix}f_k\\ f_{k+1}\\ f_{k+2}\\ \vdots\\ f_{k+h}\end{bmatrix} \tag{9}$$

$$A_{qp}=\begin{bmatrix}A & A^2 & A^3 & \cdots & A^h\end{bmatrix}^T$$

$$B_{qp}=\begin{bmatrix}B & 0 & 0 & \cdots & 0\\ AB & B & 0 & \cdots & 0\\ A^2B & AB & B & \cdots & 0\\ \vdots & \vdots & \vdots & \vdots & \vdots\\ A^{h-1}B & A^{h-2}B & A^{h-3}B & \cdots & 0\end{bmatrix}$$

2 动态方程及求解

2.1 地面反力求解

基于上述的模型预测控制模型，我们将求解地面反力的方程转换为一个二次型优化问题，目标为最小化下方的代价函数

$$\min_{\mathbf{x},\mathbf{u}}\sum_{i=0}^{k-1}\|\mathbf{x}_{i+1}-\mathbf{x}_{i+1,\mathrm{ref}}\|_{\mathbf{Q}_i}+\|\mathbf{u}_i\|_{\mathbf{R}_i} \tag{10}$$

并且上述代价函数具有以下的力约束条件

$$\begin{aligned}f_{\min}\le f_z\le f_{\max}\\ -\mu f_z\le\pm f_x\le\mu f_z\\ -\mu f_z\le\pm f_y\le\mu f_z\end{aligned} \tag{11}$$

式(10)主要考虑的因素是轨迹之间的跟踪误差，该式的物理意义在于最小化参考轨迹偏离量的加权最小二乘，并且兼顾最小的代价，即使得计算轨迹尽可能与参考轨迹重合的同时，所使用到的力最小。

其中 Q 为状态偏移量（实际状态与参考状态的偏差）的对角权重矩阵，R 为足底地面反力大小的对角权重矩阵。当调大任意一个权重矩阵的参数，代表着对某个物理量控制的“看重”。

要求得一个最优的输入值（即地面反力），需要把 x 等非输入项进行替换，将问题转换为标准二次型优化问题，将式(10)与式(9)进行联立，将 x 等变量进行代换，得到如下的标准二次型优化问题

$$\min_{\mathbf{f}}\frac{1}{2}\mathbf{f}^T\mathbf{H}\mathbf{f}+\mathbf{R}^T\mathbf{f} \tag{12}$$

其中，有

$$\begin{aligned}\mathbf{H}&=2\left(\mathbf{B}_{qp}{}^T L\mathbf{B}_{qp}+K\right)\\ \mathbf{R}&=2\mathbf{B}_{qp}{}^T L\left(\mathbf{A}_{qp}x_k-\mathbf{X}^{\mathrm{eff}}\right)\end{aligned} \tag{13}$$

对于该类二次型优化问题，可以通过 C++ 的 OSQP 库等计算工具进一步求解。

通过对该类二次型优化问题的求解，我们得到了预测部分 Horizon 时域的输入，即从 k 到 $k+h$ 部分的地面反力作为输入值。通过提取 k 得到的输入值，实际上这就是 MPC 控制器求解得到的最终目标，也就是执行完本次 MPC 计算后真正执行的控制输入。当执行完第 k 次的输入向量之后，MPC 会根据该输入继续进行迭代下去，从而持续的进行控制。

2.2 足端轨迹规划

对机器人足端参考轨迹的设计需要考虑以下四点：

(1) 轨迹曲线需要光滑、没有冲击，使四足机器人在行进过程中保持平稳；

(2) 机器人腿部处于摆动相时，抬腿和落足时的冲击要尽可能减小，防止关节速度和加速度激变；

(3) 选择合适的摆动高度和步长，使行走过程迅速、平稳；

(4) 尽可能减少足端与地面接触时产生的滑动，避免切换摆动腿时出现拖地现象。

基于上述思考，我们参考了文献[6]和[7]。根据其中提出的复合摆线轨迹设计和优化方式，我们设计了基于复合摆线的足端轨迹，减小了加速度的冲击，并改善了摆动腿拖地等问题。具体的设计和优化过程如下：

根据原始摆线轨迹设计结果（如下图所示），我们发现：轨迹在加速度图中出现跳变，因此在抬腿和落足的时候，足端与地面会产生一个冲击力。由于我们使用的 MPC 是通过足端反力来进行控制的，因此该力可能导致控制系统的不稳定。我们基于文献中的方案进行了轨迹的重新设计，改进了轨迹 Y 轴方向的公式。

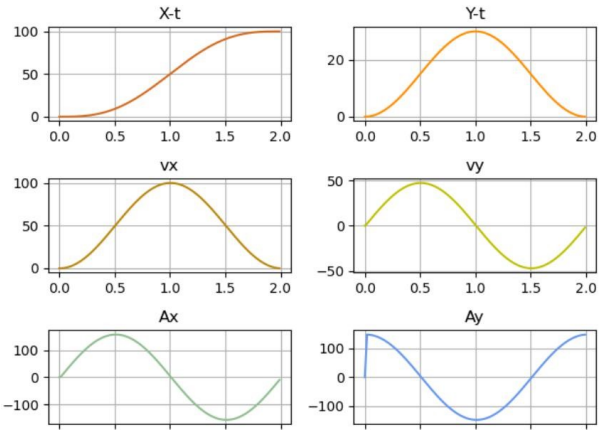

图 3 优化前的摆线轨迹

通过观察上图轨迹的 X 轴方向，发现在轨迹、速度、加速度项上都是我们期望的零冲击状态，因此很自然的思路是通过将两个X轴方向的摆线轨迹拼接，得到一条优化后的摆线轨迹。我们从加速度项开始设计，然后基于待定系数的方法，不断积分，最终得到下面优化后的摆线轨迹。

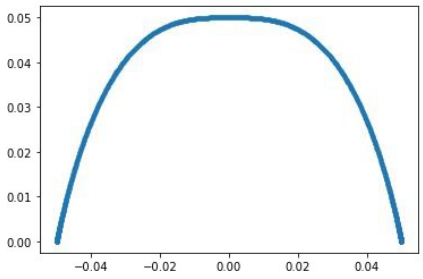

图 4 优化后的复合摆线轨迹

其中，摆动相腿部的轨迹公式为

$$\begin{cases} x = S\left[\dfrac{t}{T_m} - \dfrac{1}{2\pi}\sin\left(\dfrac{2\pi t}{T_m}\right)\right] \\ y = H\left[\mathrm{sgn}\left(\dfrac{T_m}{2} - t\right)\left(2f_E(t) - 1\right) + 1\right] \\ f_E(t) = \dfrac{t}{T_m} - \dfrac{1}{4\pi}\sin\left(\dfrac{4\pi t}{T_m}\right) \end{cases} \tag{14}$$

支撑相的参考轨迹公式为

$$\begin{cases} x = S\left(\dfrac{2T_m - t}{T_m} + \dfrac{1}{2\pi}\sin\left(\dfrac{2\pi t}{T_m}\right)\right), T_m \le y \le 2T_m \\ y = 0, T_m \le t \le 2T_m \end{cases} \tag{15}$$

3 四足机器人仿真

3.1 仿真平台构建

目前机器人仿真常用的机器人仿真平台有 Gazebo，Webots，MuJoCo，Pybullet。Gazebo 依托于 ROS 的发展，具有很强的仿真能力，是目前最广泛使用的仿真环境，但是 Ubuntu 环境下环境问题比较复杂；Webots 功能齐全，支撑多语言多系统，但是运算速度比较慢，对电脑配置比较高；MuJoCo 侧重控制与接触相关的仿真与优化，适合机械臂仿真；Pybullet 基于 Python 的 Bullet 物理引擎，API 函数和库丰富，且调用十分方便，非常适用于多关节移动机器人的仿真。因此在权衡之下，我们选择使用 Pybullet 作为仿真平台。

关于四足机器人模型的选择，目前网上常见的开源四足机器人模型包括 Mini Cheetah、Open Dynamic Robot、Laikago 等，相比于后面几款，Mini Cheetah 具有体积小，灵活易控制的优点，有利于后续 MPC 算法的实现。我们下载了开源的 Mini Cheetah 模型，并通过解读 urdf 源文件，对整个机器人的关节连杆信息进行了整理，以便于之后仿真代码的编写，具体信息如下：

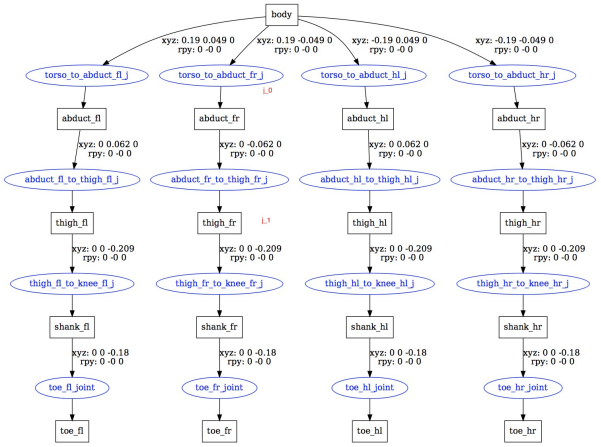

图 5 Mini Cheetah 仿真结构树

3.2 轨迹和步态验证

在控制的代码中，我们单独编写了一个 Planner 类，用于输出上述轨迹规划计算所得的位置(x,y,z)_toe，该位置是相对于足端坐标系而言。由于 Pybullet 自带的逆运动学求解函数需要的是末端执行器在世界坐标系下的位置，因此我们进行了两次坐标系变化，将相对于足端的坐标转换到世界坐标系下的坐标。

足端相对于机身的位姿可以通过正运动学求解，借助 Pybullet 自带的传感器，可以获取每一个关节实时的角度信息，通过串联机械臂求解正运动学的方法，我们可以得到没一条腿足端相对于机身的齐次变换矩阵 T1。机身相对于世界坐标系的位置和四元数也可以通过 Pybullet 自带的 IMU 传感器直接获取，通过计算可以得到机身相对于世界坐标系的齐次变换矩阵 T2，这样即可得到足端与世界坐标系的变换关系：

$$T = T_1 T_2 \tag{16}$$

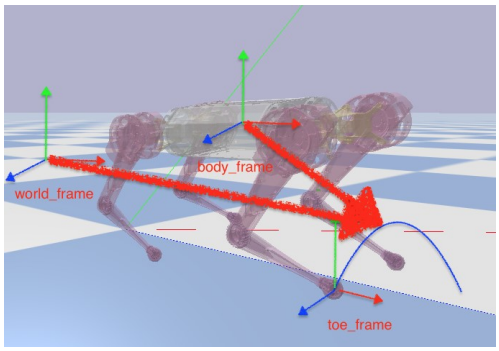

图 6 世界坐标系、机器人身体坐标系和足端坐标系变换

搭建好仿真平台后，我们首先将机器人悬空放置，避免支反力对轨迹的影响了，将机器人的 12 个电机设置为简单的位置控制模式，将 Planner 计算的坐标通过坐标变化后，逆运动学求解得到预期关节的位置，并依次传给对应电机，观察机器人足端能走出预期轨迹和步态。

解除机器人悬空固定状态，设定初始状态蹲伏在地面上，并规划出一个从地面站起的轨迹，使机器人行走前将质心高度 h 维持在 0.25m 后，开始让机器人足端执行预期的轨迹和步态。通过调节摆动周期 Tm、步高 H、步长 S，使机器人在开环控制下稳定行走，以用于后续 MPC 控制。绘制质心高度和足端高度轨迹如下图，得到质心高度较为平稳的步幅参数 Tm = 1.6s, H = 0.125m, S = 0.1m。

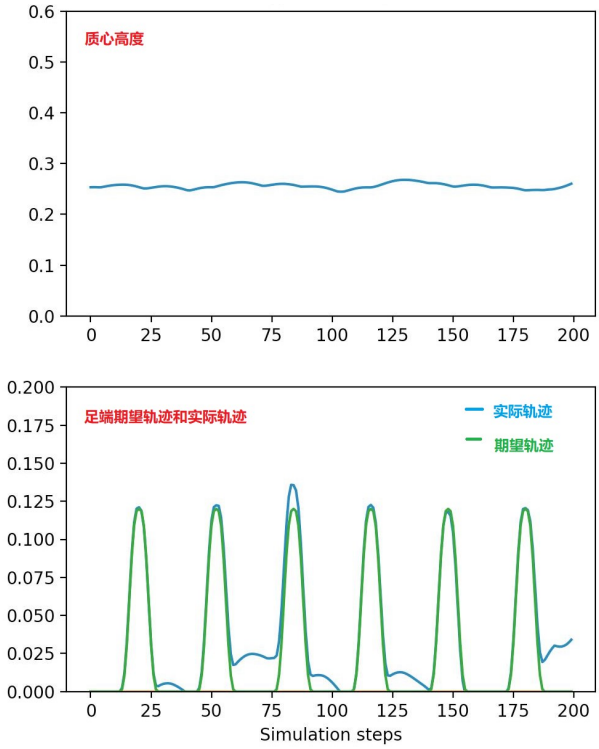

图 7 机器人质心高度和足端高度

3.3 摆动相力矩控制

在位置控制的基础上，我们只需将电机的位置控制模式修改为力矩控制模式，并添加一个 PD 控制器，依照如下公式，即可实现摆动相的力矩控制：

$$\tau_i = J_i^T \left[K_P \left(P_{ref} - p_i \right) + K_d \left(v_{ref} - v_i \right) \right] + \tau_{i,ff} \tag{17}$$

其中 Ji 为第 i 条腿对应的雅可比矩阵，可以通过 pybullet 自带的函数进行计算，Kp 和 Kd 分别对应位置和速度增益，通过调节参数，最终确定为

$$K_p = 10 \ , \ K_d = 2.5 \tag{18}$$

P_{ref} 是 3.2 中计算得到的预期关节位置，Pi 是传感器返回的实际位置，通过对两个位置参数进行差分，可以得到预期速度参数 V_ref，对两个时刻传感器测得位置 进行差分得到实际速度 Vi。

此时腿部实现了一定程度的闭环控制，但是由于没有加入支撑相控制，只能在轻微扰动下恢复正常，当冲击大于 0.7 kg·m/s 时有可能会发生倾倒。

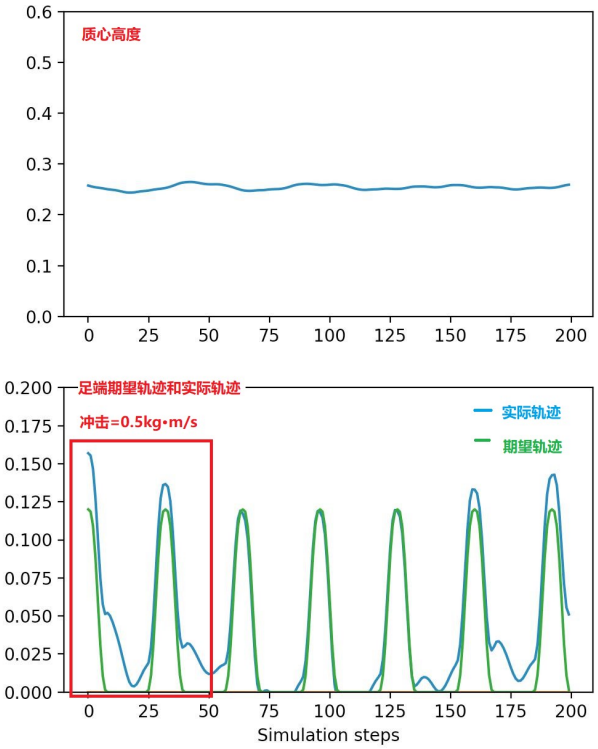

图 8 机器人受到冲击时的质心高度和足端高度

4 MPC 控制方法实现

4.1 MPC 控制框架

MPC 控制框架整体上由轨迹规划模块、腿部状态估计模块和相位切换模块、地面反力优化问题求解模块、支撑相控制器模块和摆动相控制器模块组成。具体的流程图如下所示：

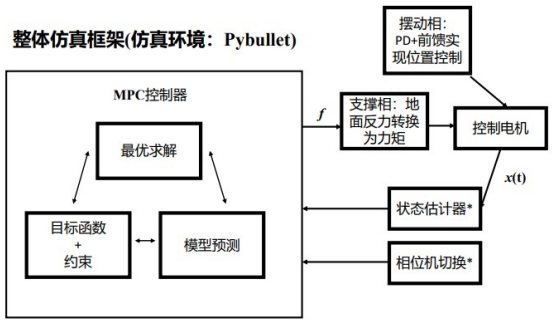

图 9 四足机器人 MPC 整体仿真框架

其中，地面反力求解和轨迹规划已经在第 2 章中详细介绍，这里不再赘述。下文将详细介绍腿部状态估计模块和相位切换模块、支撑相和摆动相的控制模块的实现方法，并展示 MPC 控制四足机器人的效果。

4.2 腿部状态估计和相位切换

我们面对的第一个问题是：如何实时切换摆动腿和支撑腿的控制？为此，我们设置了一个相位切换模块：设摆动腿的相位为 0，支撑腿的相位为 1。每次我们通过逆运动学求解到关节位置信息之后，都需要将关节的位置信息输入给关节控制器。我们在关节控制器中进行了相位的识别功能，遍历每一条腿识别相位，相位是 0 则通过上面的转换公式得到关节力矩，相位是 1 则调用支撑腿的地面反力求解器计算地面反力，并进一步将其映射为关节力矩。

那么如何获取或估计足端是否是触地状态呢？我们参考了 Google 团队关于腿部状态估计的代码，并且根据我们需要实现的具体功能进行了修改与优化。现实的四足机器人中，以宇树科技的 aliengo 为例，在足端的气垫中装有力传感器，可以监控触地状态，但这往往是不够的，还需要使用基于事件的足端状态估计器来预测足端触地状态（检测足端的速度、加速度、位置信息等），这也是 Google 代码中触底检测的一个重要性的部分。

除此之外，我们还需要针对足端实时的触地情况进行提前触地和滞后触地不同情况的分类控制，这类似于我们在 ZMP 控制理论中所提及的通过控制触地点来改变加速度的方式。但是我们的项目中不需要进行速度和加速度的控制，所以我们简化了足端触地检测，通过 Pybullet 自带的触地检测函数，得到了我们的四足机器人模型和地面接触的信息，这里返回的信息包括触地的杆件号以及在世界坐标系中的位置等，值得提及的是我们的触地杆件号不仅仅包括足端，还包括小腿杆件，针对 Mini Cheetah 模型，我们发现小腿关节也有一定的可能性与地面接触，因此需要将小腿触地也作为足端触地的情况进行判断。下图展示了实现腿部状态估计和相位切换功能的核心代码。

```
def update_foot_contact_state(self):
    """
    get contact situations by `p.getContactPoints()`, then update contact situation
    Body 0: plane;
    Body 1: robot
    """
    foot_contact_test = [0,0,0,0]
    self.foot_contact = [0,0,0,0]
    for cp in p.getContactPoints(self.robot):
        print(cp)
        if cp[3] in self.toe_link_IDs:
            self.foot_contact[cp[3] // 4] = 1
        print(self.foot_contact)
```

图 10 足端触地检测和相位切换代码

4.3 支撑相和摆动相的控制

在完成腿部的触地检测和相位切换之后，我们分别用两种不同的方法控制支撑相和摆动相。

对于处在支撑相的腿，我们首先通过地面反力优化问题求解模块，求得使支撑相轨迹接近参考轨迹且所需能量较小的期望地面反力（在 2.1 中详细介绍），对于优化问题的求解我们使用了 C++的 OSQP 库。地面反力求解模块的输入项包括期望线速度、期望角速度、期望高度、机身重量、惯量矩阵、腿的数量和摩擦系数等。通过以上参数的输入

我们构建了质心的期望线速度、期望角速度和期望欧拉角等信息以及当前质心的状态。我们将质心位置、质心速度、质心俯仰角、质心角速度、足端触地情况、足端基坐标位置、足端摩擦系数、质心期望位置、期望速度、期望俯仰角、期望角速度等信息作为输入参数，给到我们先前引入的优化问题求解库的函数中，就可以得到我们需要的地面反力。该部分的代码如下。

```
predicted_contact_forces = self._cpp_mpc.compute_contact_forces(
    [0],  #com_position
    np.asarray(self._robot.com_velocity_body_frame,
               dtype=np.float64),  #com_velocity
    np.array(com_roll_pitch_yaw, dtype=np.float64),  #com_roll_pitch_yaw
    # Angular velocity in the yaw aligned world frame is actually different
    # from rpy rate. We use it here as a simple approximation.
    np.asarray(self._robot.GetBaseRollPitchYawRate(),
               dtype=np.float64),  #com_angular_velocity
    foot_contact_state,  #foot_contact_states
    np.array(self._robot.GetFootPositionsInBaseFrame().flatten(),
             dtype=np.float64),  #foot_positions_base_frame
    self._friction_coeffs,  #foot_friction_coeffs
    desired_com_position,  #desired_com_position
    desired_com_velocity,  #desired_com_velocity
    desired_com_roll_pitch_yaw,  #desired_com_roll_pitch_yaw
    desired_com_angular_velocity  #desired_com_angular_velocity
)
```

图 11 地面反力求解代码

得到期望的地面反力之后，我们通过如下公式将地面反力映射为控制支撑相关节的力矩。

$$\tau_i = J_i^T R_i^T f_i \tag{19}$$

其中，τ_i 表示第 i 条腿的关节扭矩，是 3×1 的向量，J 表示足端的雅可比矩阵，R 表示从机器人身体坐标系到世界坐标系的旋转矩阵，f_i 表示世界坐标系下控制第 i 条腿期望的地面反力。

完成地面反力到关节扭矩映射的代码如下图所示。

```
def MapContactForceToJointTorques(self, leg_id, contact_force):
    """Maps the foot contact force to the leg joint torques."""
    jv = self.ComputeJacobian(leg_id)  # 计算目标控制腿的雅可比矩阵
    all_motor_torques = np.matmul(contact_force, jv)
    motor_torques = {}
    motors_per_leg = 3
    # 将计算得到的力矩传给关节
    com_dof = 6
    for joint_id in range(leg_id * motors_per_leg,
                          (leg_id + 1) * motors_per_leg):
        motor_torques[joint_id] = all_motor_torques[
                                  com_dof + joint_id] * self._motor_direction[joint_id]

    return motor_torques
```

图 12 关节扭矩计算代码

对于摆动相，我们也是通过关节力矩控制其按照参考轨迹运动。与支撑相不同的是，我们直接采用与 3.3 相同的 PD 控制。

4.4 机器人控制流程和效果

为方便整理思路，在展示我们使用自己搭建的 MPC 框架控制四足机器人的效果之前，先对我们仿真控制的整体流程进行总结：在加载完成世界和四足机器人的 URDF 模型之后，我们先让机器人执行站立程序，当能够成功站起的时候，我们先用时序控制生成腿部的初始相位和轨迹，通过 Planner.py 文件得到期望轨迹后进行坐标转换得到世界坐标系下的位置信息，然后通过逆运动学求解关节位置，将关节位置信息作为输入给到关节控制器，进行摆动腿和支撑腿的判断并执行对应的控制器，摆动腿和支撑腿的控制器输出的都是关节力矩，由电机进行控制。整个运动过程实际上是对上述过程的循环，每次循环结束都进行腿部的触地检测和状态更新。

在测试中，我们以四足机器人在 MPC 控制下保持自稳定的能力作为指标，在 Pybullet 中设置多种环境，包括外部横向冲击和崎岖地形，以及我们可以通过用鼠标拉动的方式给机身一个扰动，来观察机器人的自稳定效果。

机器人的控制效果如下图所示，我们发现，当给机器人一定范围内的上述扰动后，机器人可以对扰动做出反应，维持稳定，并逐渐恢复我们给定的参考轨迹。

（机器人控制的详细动态效果见附件视频）

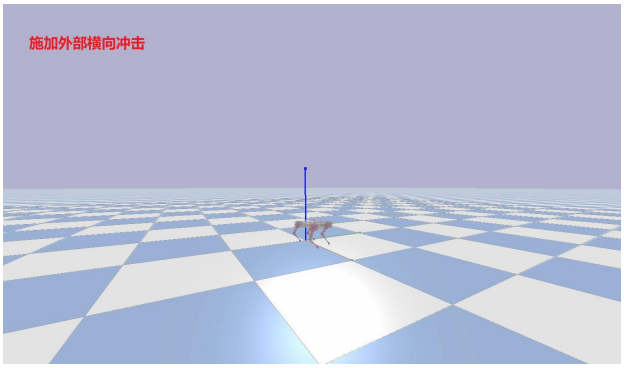

图 13 通过 MPC 控制使机器人在外部横向冲击下保持稳定

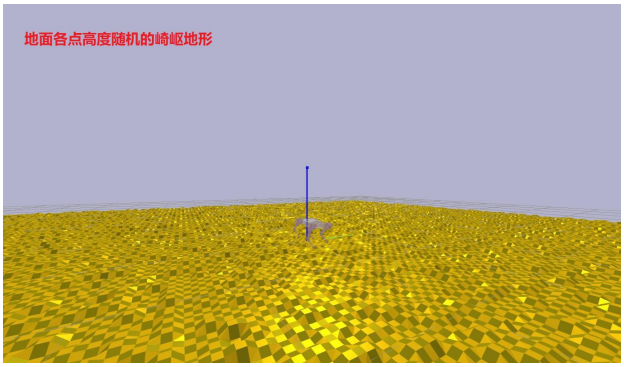

图 14 通过 MPC 控制使机器人在崎岖地形上行走

5 系统参数对机器人控制的影响

由上文可知，摆动周期 Tm、步高 H、步长 S、摆动相力矩 PD 控制中的 Kp 和 Kd 以及地形崎岖程度 dH 等系统参数，均会对机器人的运动和控制产生影响。即使其中一个参数不合适，机器人也有可

能发生倾倒等问题。

为了方便描述，我们将系统参数分为机器人内参和外参：内参即和步态、控制有关的机器人内部参数，外参即外部冲击、地形崎岖程度等环境参数。

经过我们小组的计算和反复的调试，我们确定了合适的内参，如下表所示。

机器人内参	数值
质心高度 h	0.25m
摆动周期 Tm	1.6s
步高 H	0.125m
步长 S	0.1m
摆动相力矩控制 Kp	10
摆动相力矩控制 Kd	2.5

表 1 机器人合适的内参

对于外参中的外部冲击，随着控制方法的升级，机器人所能承受的外部冲击不断增加：当只用 PD 控制摆动相力矩时，机器人仅受到大于 0.7 kg·m/s 的冲击时就有可能发生倾倒；加入 MPC 控制后，机器人能在大于 50N 的冲击力下保持稳定（如 4.3 所示）。

我们通过程序生成地面每点高度不同的地形，高度为 0~dH 之间的随机值。我们用 dH 表示地形崎岖程度，dH 越大，地面相邻两点之间可能的高度落差就越大，即地形越崎岖。关于机器人能够承受的地形崎岖程度，我们做了大量测试，发现：当只用 PD 控制时，机器人只能在完全平坦的地面运动；用 MPC 控制时，机器人可以在 dH = 0.07m 的地形上保持稳定。

6 合作分工与组员贡献情况

(1) 李奥齐（20%）：完成四足机器人的仿真，实现机器人腿部关节的位置控制和关节力矩的 PD 控制，为 MPC 控制提供了基础。

(2) 郭骏德（20%）：搜集并分析足端轨迹规划的文献，设计四足机器人足端轨迹；与李奥齐合作完成了部分仿真接口的实现。

(3) 江轶豪（20%）：搜集并分析 MPC 相关文献和代码，并向小组做文献综述，帮助成员理解 MPC 理论；完成 MPC 框架的搭建。

(4) 田丰（20%）：分析 MPC 相关文献和代码；完成 MPC 控制器中地面反力求解部分的代码编写；与杨博文合作完成 MPC 代码的调试和修改。

(5) 杨博文（20%）：分析 MPC 相关代码；完成 MPC 控制器中腿部摆动相和支撑相不同控制方法的切换；与田丰合作完成 MPC 代码的调试；完成四足机器人在多种地形环境中的测试。

7 研究总结

(1) 本项目基于 Pybullet 仿真环境和麻省理工学院团队开发的 Mini Cheetah 实现了对四足机器人的仿真控制：我们小组首先根据相关文献，完成了对四足机器人动力学模型、状态空间模型等模型的理论分析和建模工作；然后我们完成了足端复合摆线轨迹的优化设计；完成理论分析之后，我们搭建了四足机器人仿真平台，并完成了腿部关节力矩的 PD 控制；在以上工作的基础上，我们独立编写了 MPC 控制器，在对目前开源的 MPC 框架进行了一系列简化和优化之后，我们实现了四足机器人的 MPC 控制。实验结果表明，我们编写的 MPC 控制器能够使四足机器人具有一定的自稳定能力，机器人在受到外部干扰时可以通过调整步态和位姿保持站立，并逐渐恢复预先指定的参考轨迹。

(2) 通过本项目，我们加深了对坐标变换、机器人动力学建模等行走机器人课上所学内容的理解，学习并掌握了 MPC 控制理论，接触了二次型优化问题求解、足端复合摆线轨迹规划、机器人状态估计等知识，提高了自己的文献分析、数学建模和代码编写等能力。

(3) 回顾我们小组在本项目中的工作，我们发现相比于完整的 MPC 体系，我们还有一些不足之处，例如：我们直接忽略了 Roll、Pitch 等理论小量，而没有对其进行控制；我们直接将摆动腿的力矩控制前馈项设为机器人静止站立所需的力矩，这个简化是不准确的；我们也缺少基于事件触发的足端状态估计机制，即我们只能判断一条腿是否和地面接触，而无法判断其接触是否过早或过晚，这使我们的控制在一些特殊情况下（如外部干扰过大）可能会失效。这些缺陷为我们提供了未来进一步研究和改进的方向。

(4) 本项目的研究结果对其他研究者设计并搭建 MPC 控制器有一定的借鉴作用，我们小组将开源本项目中的所有代码，以供参考。

参 考 文 献

[1] 汪世庆,单鑫,刘逸驰.四足机器人的发展现状及趋势[J].造纸装备及材料,2020,49(04):227-228.

[2] 孟健,刘进长,荣学文,李贻斌.四足机器人发展现状与展望[J].科技导报,2015,33(21):59-63.

[3] W. Bosworth, S. Kim and N. Hogan, The MIT super mini cheetah: A small, low-cost quadrupedal robot for dynamic locomotion[C]// 2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2015: 1-8.

[4] 王存强. 四足机器人运动控制的研究与实现[D].太原理

工大学,2020.DOI:10.27352/d.cnki.gylgu.2020.000036.

[5] Carlo J D， Wensing P M， Katz B , et al. Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control[C]// 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018.

[6] SAKAKIBARA Y，KAN K，HOSODA Y，et al. Foot trajectory for a quadruped walking machine[C]// Proceedings IROS '90. IEEE International Workshop on，July 3-6，1990，Ibaraki，Japan. New York，NY，USA：IEEE，1990: 315-322.

[7] 何冬青，马培荪. 四足机器人动态步行仿真及步行稳定性分析[J]. 计算机仿真，2005(2)：146-149. HE Dongqing ， MA Peisun. Simulation of dynamic walking of quadruped robot and analysis of walking stability[J]. Computer Simulation，2005(2): 146-149.