基于模型预测控制的四足机器人仿真报告

汇报人：李奥齐 郭俊德 江轶豪 田丰 杨博文

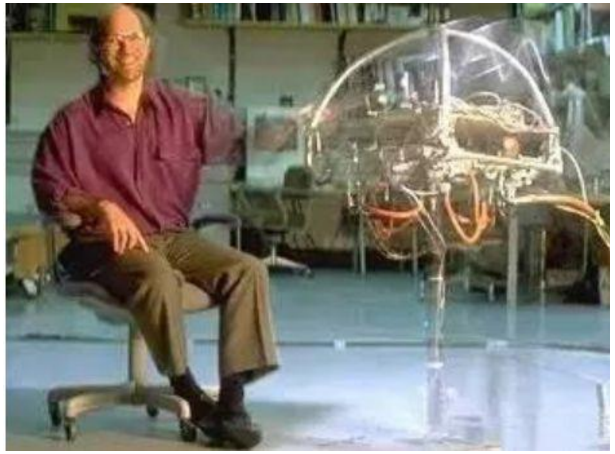
指导教师：付成龙 教授

时间：2022年5月30日

目录 CONTENTS

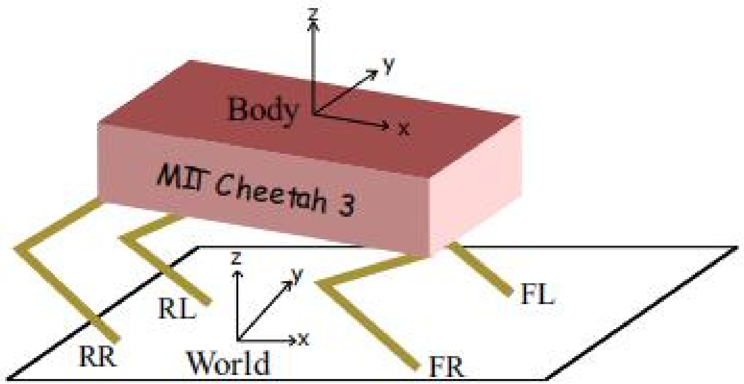
研究背景及意义

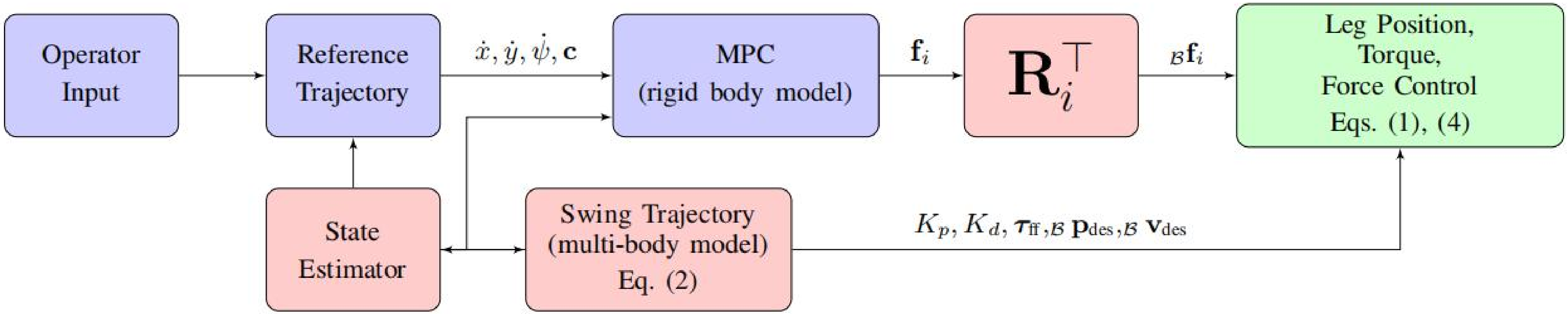
■ 研究四足机器人对于行走机器人控制理论的发展具有重要意义



■ 研究MPC控制对于我们理解现代控制理论的发展具有重要意义

模型预测控制理论





J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 1-9, doi: 10.1109/IROS.2018.8594448.

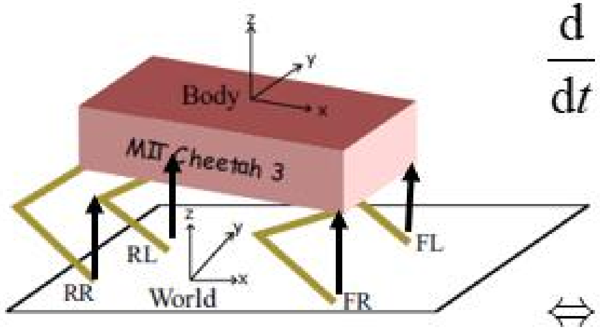
■ 关于质心加速度的牛顿公式

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

■ 旋转矩阵与欧拉角：求欧拉角加速度的近似关系

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

■ 欧拉公式的近似转化与空间惯量张量在坐标系下的变换



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

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

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

J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 1-9, doi: 10.1109/IROS.2018.8594448.

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

$\Big\downarrow$ **Discretization**

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

J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 1-9, doi: 10.1109/IROS.2018.8594448.

模型预测控制理论

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

$$x(\mathrm{k}+1) = A_{\mathrm{k}}x(\mathrm{k}) + B_{\mathrm{k}}f(\mathrm{k})$$

Prediction under given HORIZONS:

$$\begin{bmatrix} x_{k+1} \\ x_{k+2} \\ x_{k+3} \\ \vdots \\ x_{k+h} \end{bmatrix} = \begin{bmatrix} A \\ A^2 \\ A^3 \\ \vdots \\ A^h \end{bmatrix} x_k + \begin{bmatrix} B & 0 & 0 & \cdots & 0 \\ AB & B & 0 & \cdots & 0 \\ A^2B & AB & B & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ A^{h-1}B & A^{h-2}B & A^{h-3}B & \cdots & 0 \end{bmatrix} \begin{bmatrix} f_k \\ f_{k+1} \\ f_{k+2} \\ \vdots \\ f_{k+h} \end{bmatrix}$$

Kim, D., Di Carlo, J., Katz, B., Bledt, G., and Kim, S., “Highly Dynamic Quadruped Locomotion via Whole-Body Impulse Control and Model Predictive Control”, <i>arXiv e-prints</i>, 2019.
J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 1-9, doi: 10.1109/IROS.2018.8594448.

QP Formulation:

$$\begin{bmatrix} x_{k+1} \\ x_{k+2} \\ x_{k+3} \\ \vdots \\ x_{k+h} \end{bmatrix} = \begin{bmatrix} A \\ A^2 \\ A^3 \\ \vdots \\ A^h \end{bmatrix} x_k + \begin{bmatrix} B & 0 & 0 & \cdots & 0 \\ AB & B & 0 & \cdots & 0 \\ A^2B & AB & B & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ A^{h-1}B & A^{h-2}B & A^{h-3}B & \cdots & 0 \end{bmatrix} \begin{bmatrix} f_k \\ f_{k+1} \\ f_{k+2} \\ \vdots \\ f_{k+h} \end{bmatrix}$$

To minimize the cost function:

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

Force Constraints:

$$f_{\min} \le f_z \le f_{\max}$$
$$-\mu f_z \le \pm f_x \le \mu f_z$$
$$-\mu f_z \le \pm f_y \le \mu f_z$$

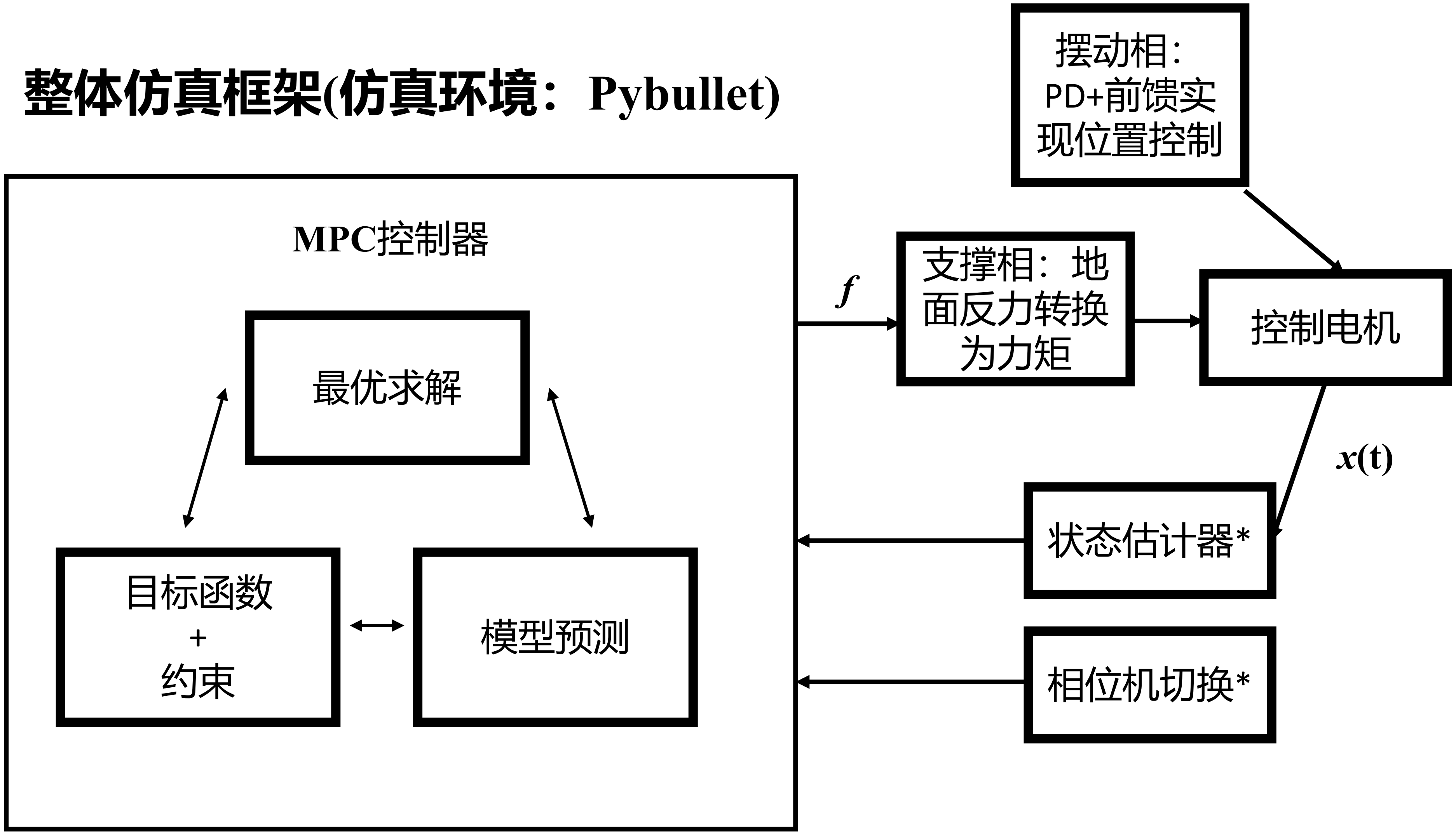
Standard QP function:

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

$$\mathbf{H} = 2\left(\mathbf{B}_{qp}{}^T L \mathbf{B}_{qp} + K \right)$$

$$\mathbf{R} = 2\mathbf{B}_{qp}{}^T L \left(\mathbf{A}_{qp} x_k - \mathbf{X}^{\mathrm{eff}} \right)$$

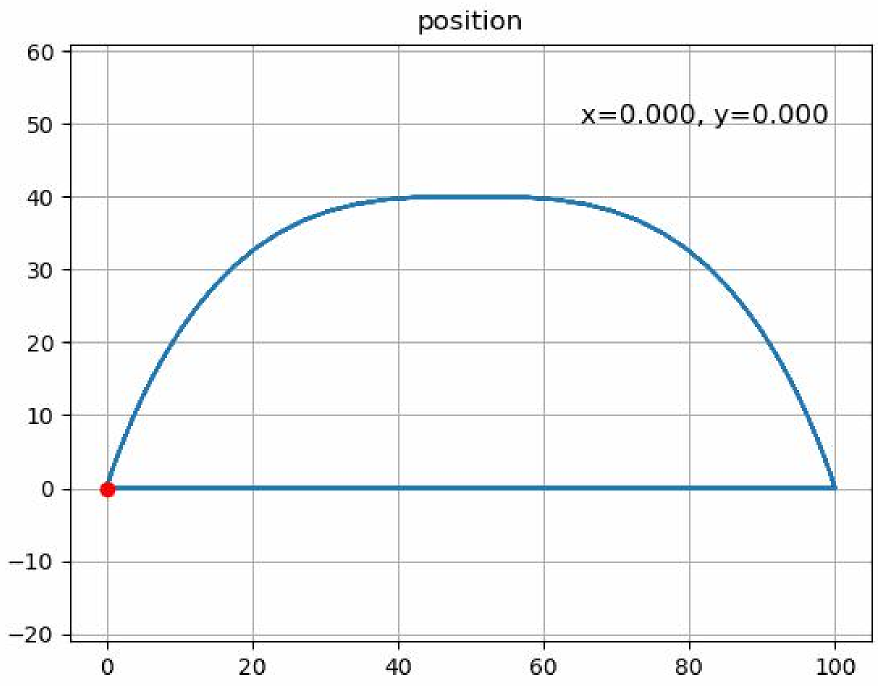
整体仿真框架(仿真环境：Pybullet)



足端轨迹优化

在四足足端轨迹设计中，我们注意了以下几点：

1. 使轨迹曲线光滑无冲击，让腿式机器人在行进过程中更加平稳
2. 在摆动相的抬腿和落足时减小冲击，防止关节速度和加速度项的畸变
3. 摆动高度和步长方便调节，跨步过程迅速平稳
4. 减少足端与地面的接触产生的滑动，避免摆动腿切换时的拖地现象

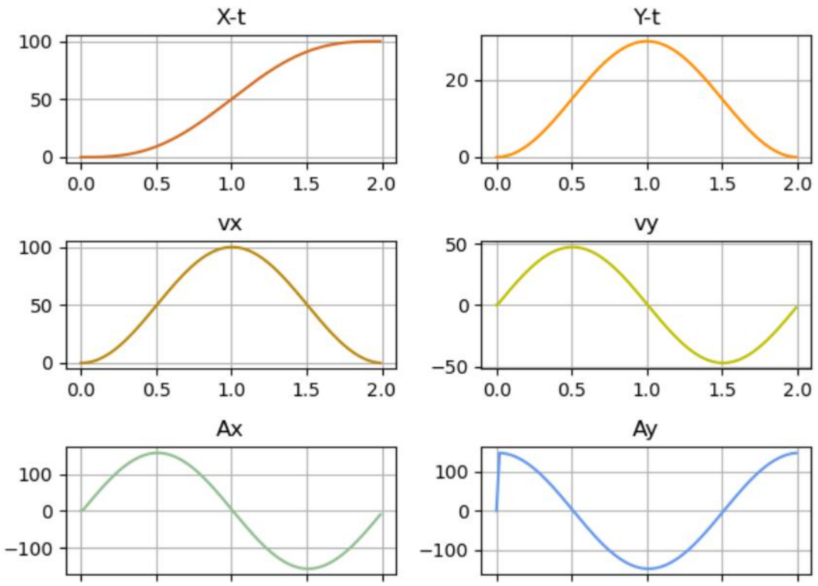



```python
def trot_traj_plan_swing(self, t):
    """
    plan a 'Compound cycloidal trajectories'
    refer to https://blog.csdn.net/weixin_41045354/article/details/105219092
    During T/2: swing phase
    During T/2 ~ T: support phase

    :param S: step length
    :param T: period
    :param H: leg raise height
    :return: a vec3 list (arr)
    """

    t = t % self.T if t > self.T else t
    x = self.S * (t / self.T - np.sin(2 * np.pi * t / self.T) / (2 * np.pi)) - self.S / 2
    fE = t / self.T - np.sin(4 * np.pi * t / self.T) / (4 * np.pi)
    z = self.H * (np.sign(self.T / 2 - t) * (2 * fE - 1) + 1)
    y = 0
    return x, y, z

def trot_traj_plan_support(self, t):
    t = t % self.T if t > self.T else t
    x = self.S * ((2 * self.T - t) / self.T + np.sin(2 * np.pi * t / self.T) / (2 * np.pi) - 1) - self.S / 2
    z = 0
    y = 0
    return x, y, z
```

在“Foot trajectory for a quadruped walking machine”这篇论文中，提出了基于复合摆线的轨迹设计；我们参考《四足机器人动态步行仿真及步行稳定性分析》这篇论文和相关博客对其进行优化，减少加速度项的冲击，并改善了摆动腿的拖地问题等。其中摆动腿轨迹的公式如下：

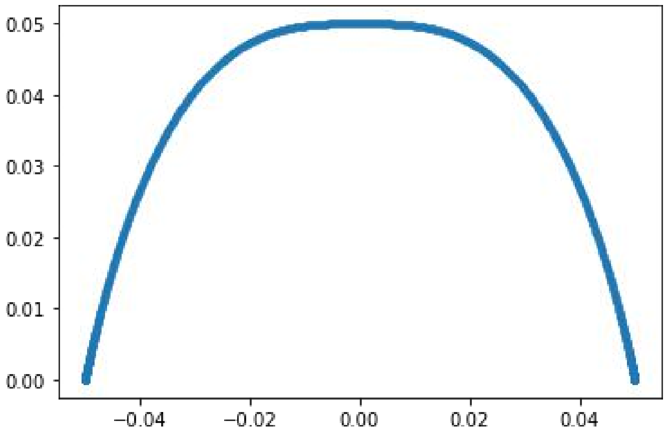
$$\begin{cases} x = S\left[\frac{t}{T_m} - \frac{1}{2\pi}\sin\left(\frac{2\pi t}{T_m}\right)\right] \\ y = H\left[sgn\left(\frac{T_m}{2} - t\right)(2f_E(t) - 1) + 1\right] \end{cases}$$

$$f_E(t) = \frac{t}{T_m} - \frac{1}{4\pi}\sin\left(\frac{4\pi t}{T_m}\right)$$



支撑相的轨迹在位置控制的仿真中使用，公式如下：

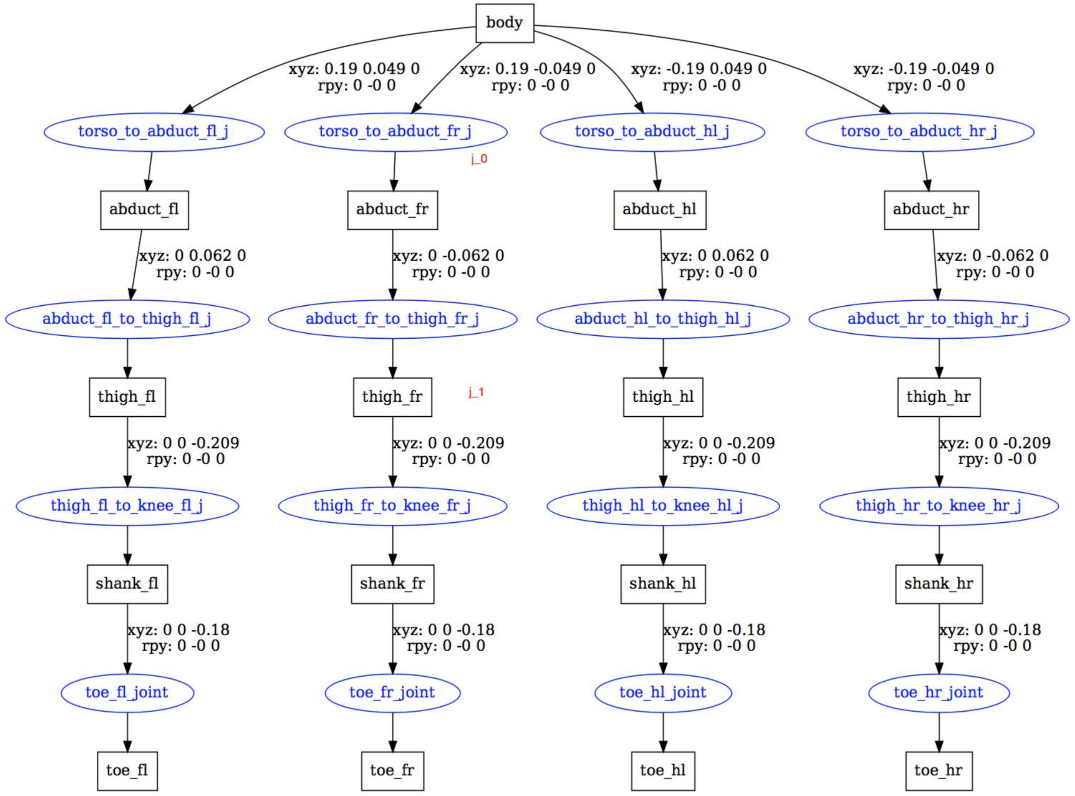
$$\begin{cases} x = S\left(\frac{2T_m - t}{T_m} + \frac{1}{2\pi}\sin\left(\frac{2\pi t}{T_m}\right)\right), T_m \le y \le 2T_m \\ y = 0, T_m \le t \le 2T_m \end{cases}$$

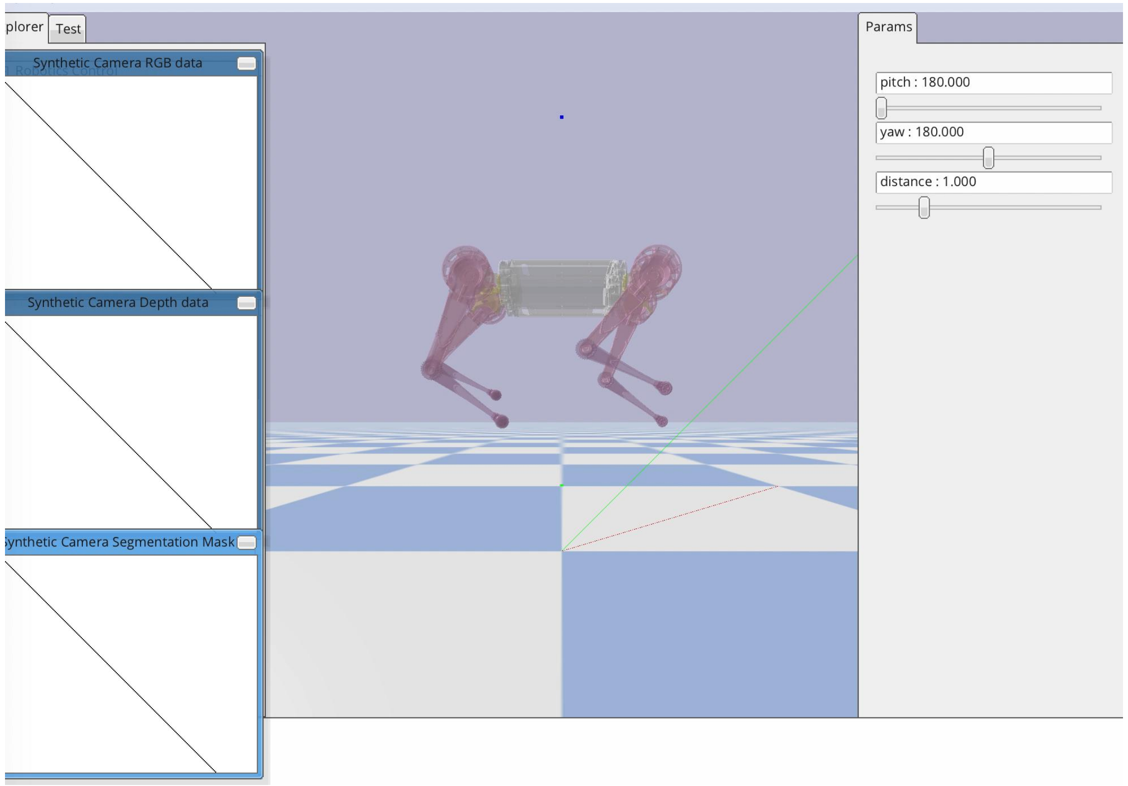


仿真前准备工作

阅读urdf、分析机器人结构及其命名

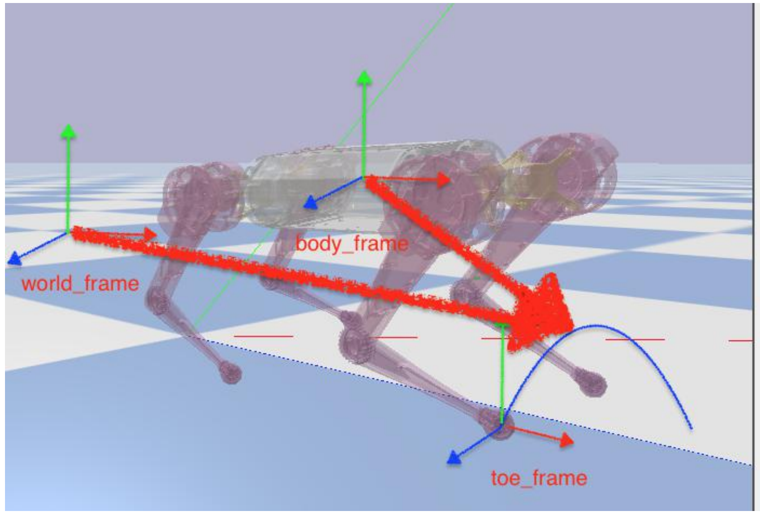
搭建仿真环境、实现简单位置控制

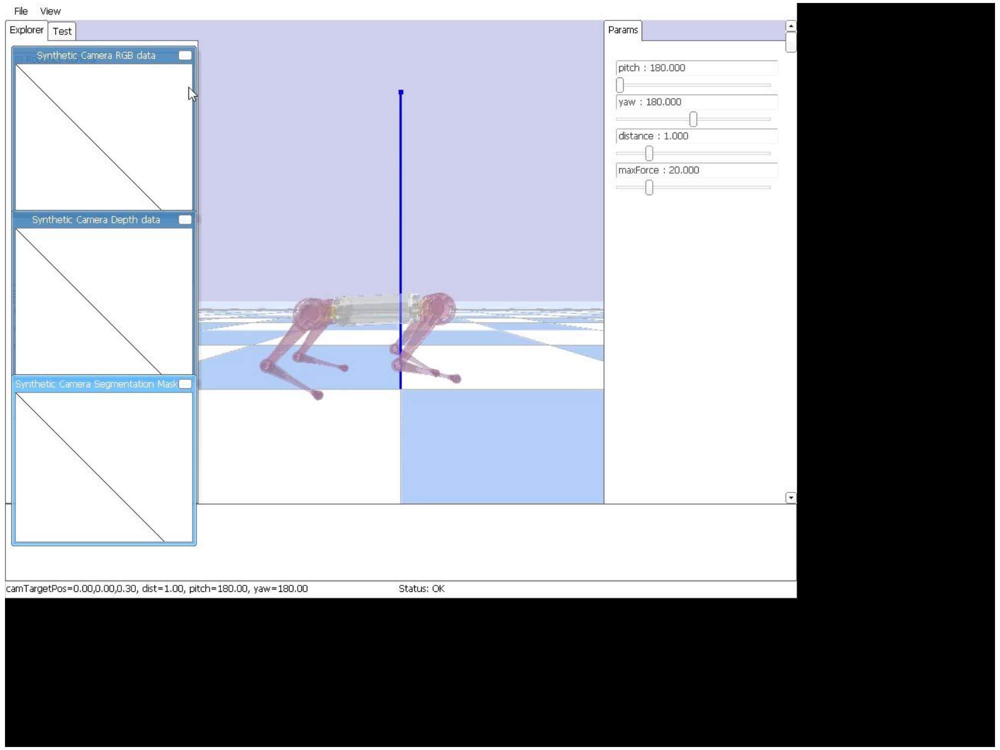


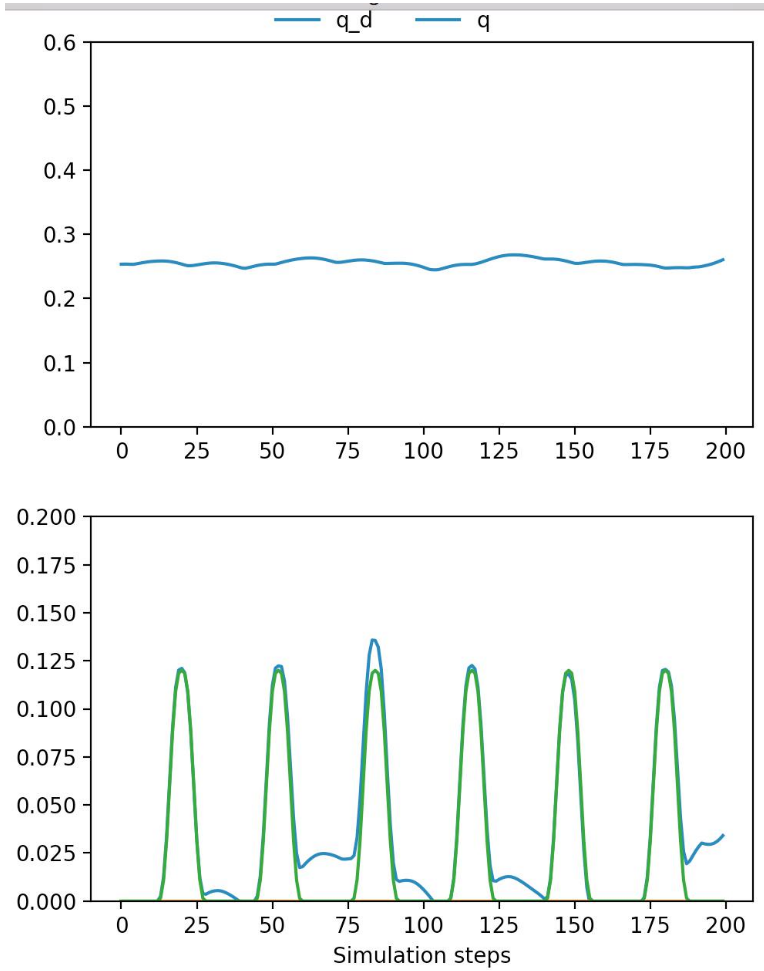


机器人下地行走

Toe trajectory → Position in toe frame $\xrightarrow{x(t)}$ Position in body frame → Position in world frame $\xrightarrow{IK}$ Desired joints position



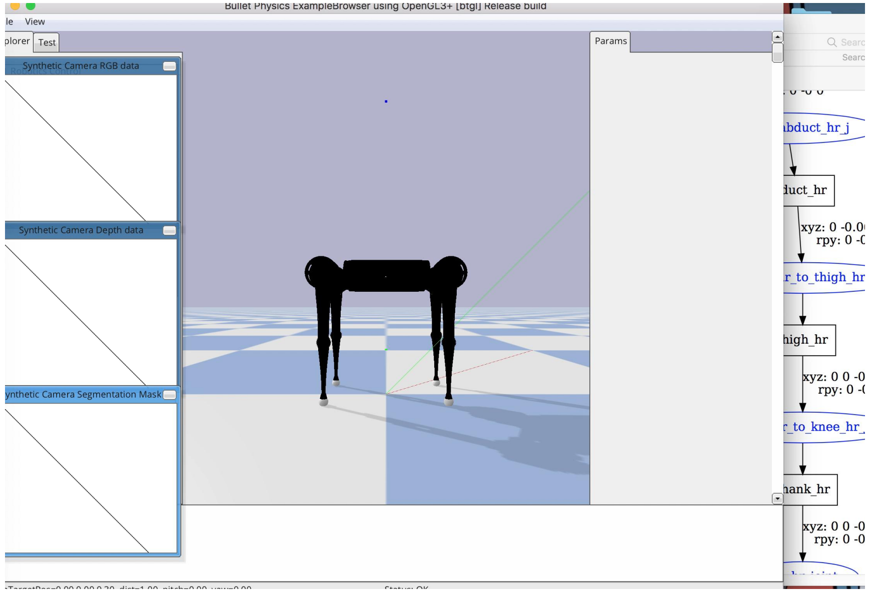


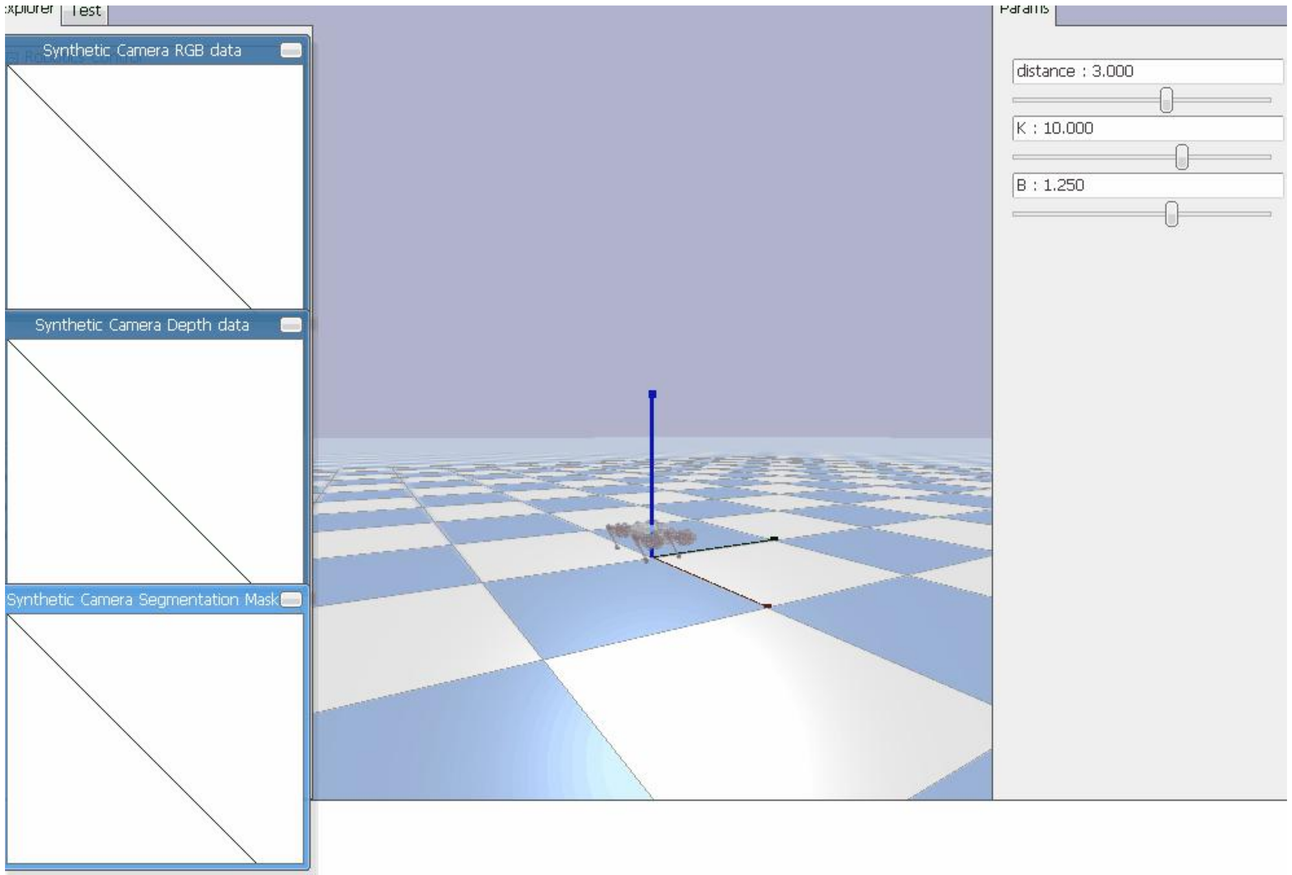


力矩控制行走

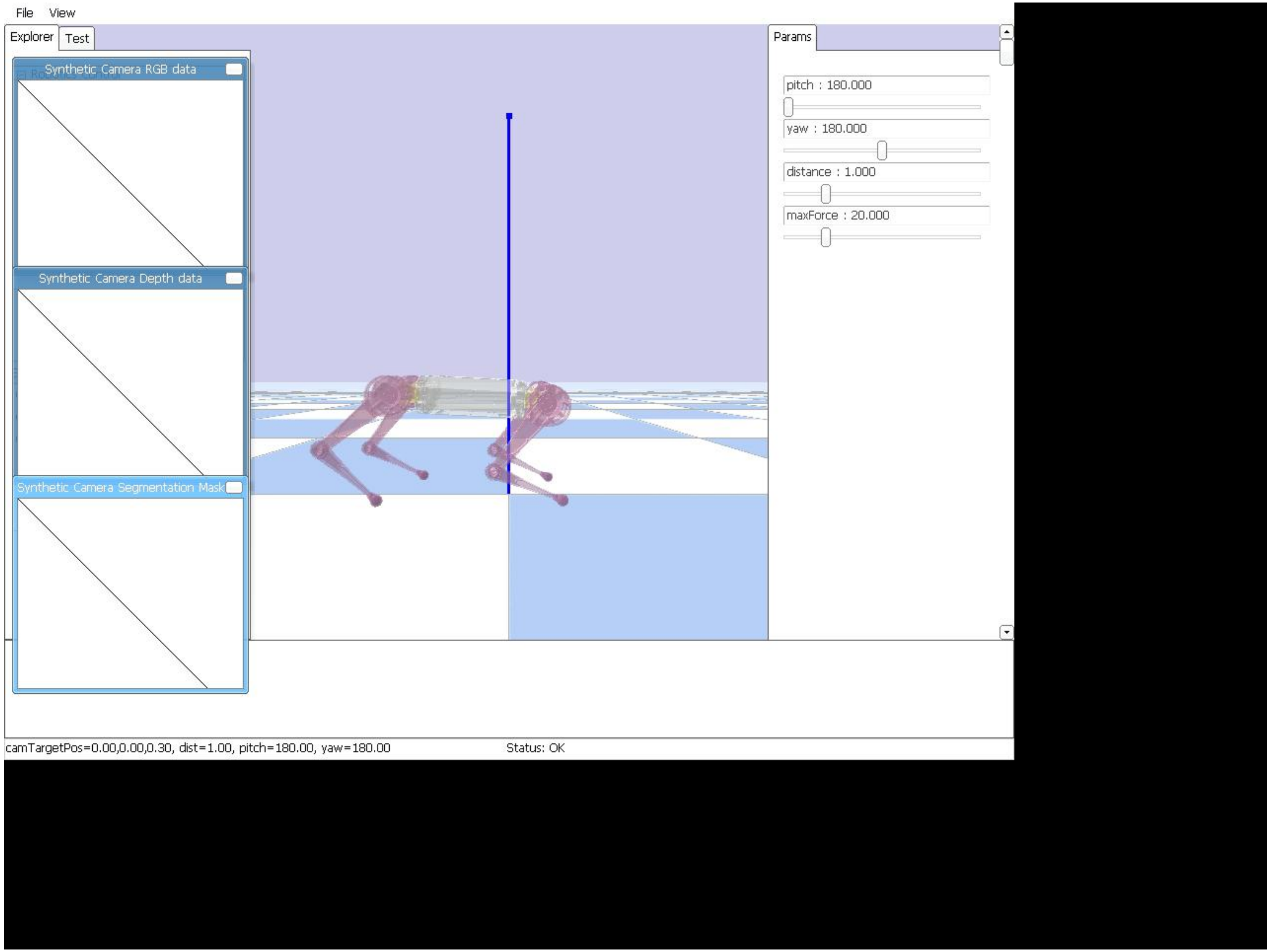
$$\tau_i = J_i^T[K_P(P_{ref} - p_i) + K_d(v_{ref} - v_i)] + \tau_{i,ff}$$

J: 第i条腿对应的Jacobian Matrix
Kp: position gain
Kd: velocity gain
Pref: 根据轨迹规划得到的关节位置
Pi: 实际位置
Vref : 位置差分法
Vi: 实际速度

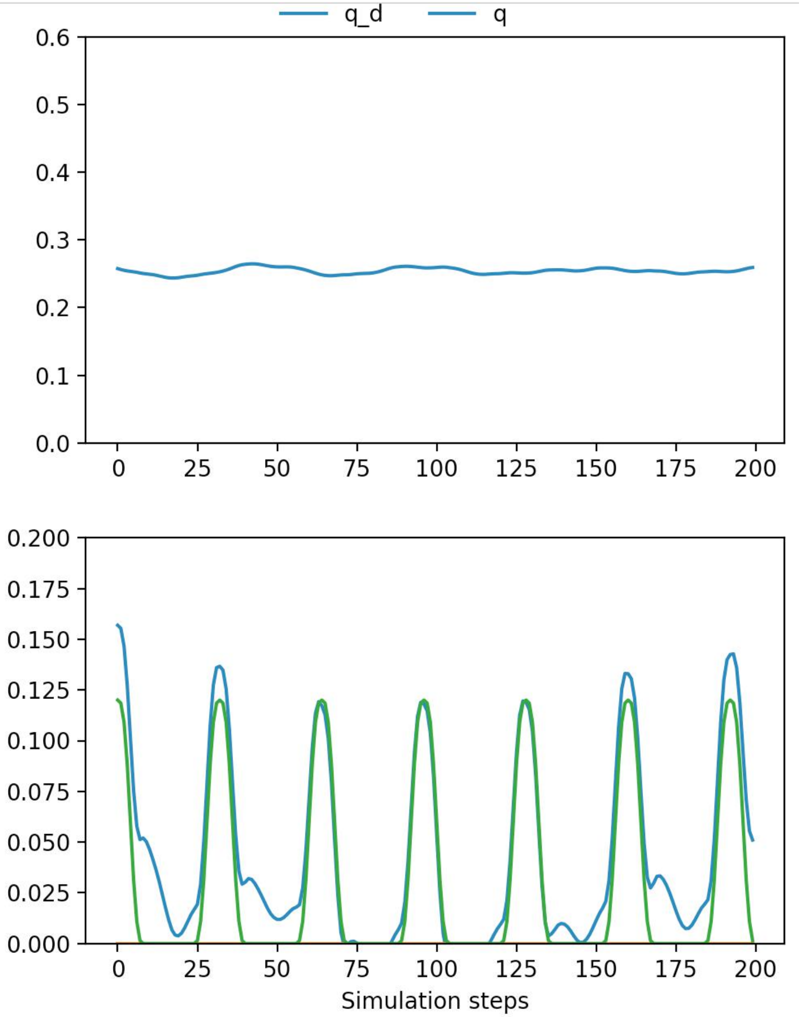




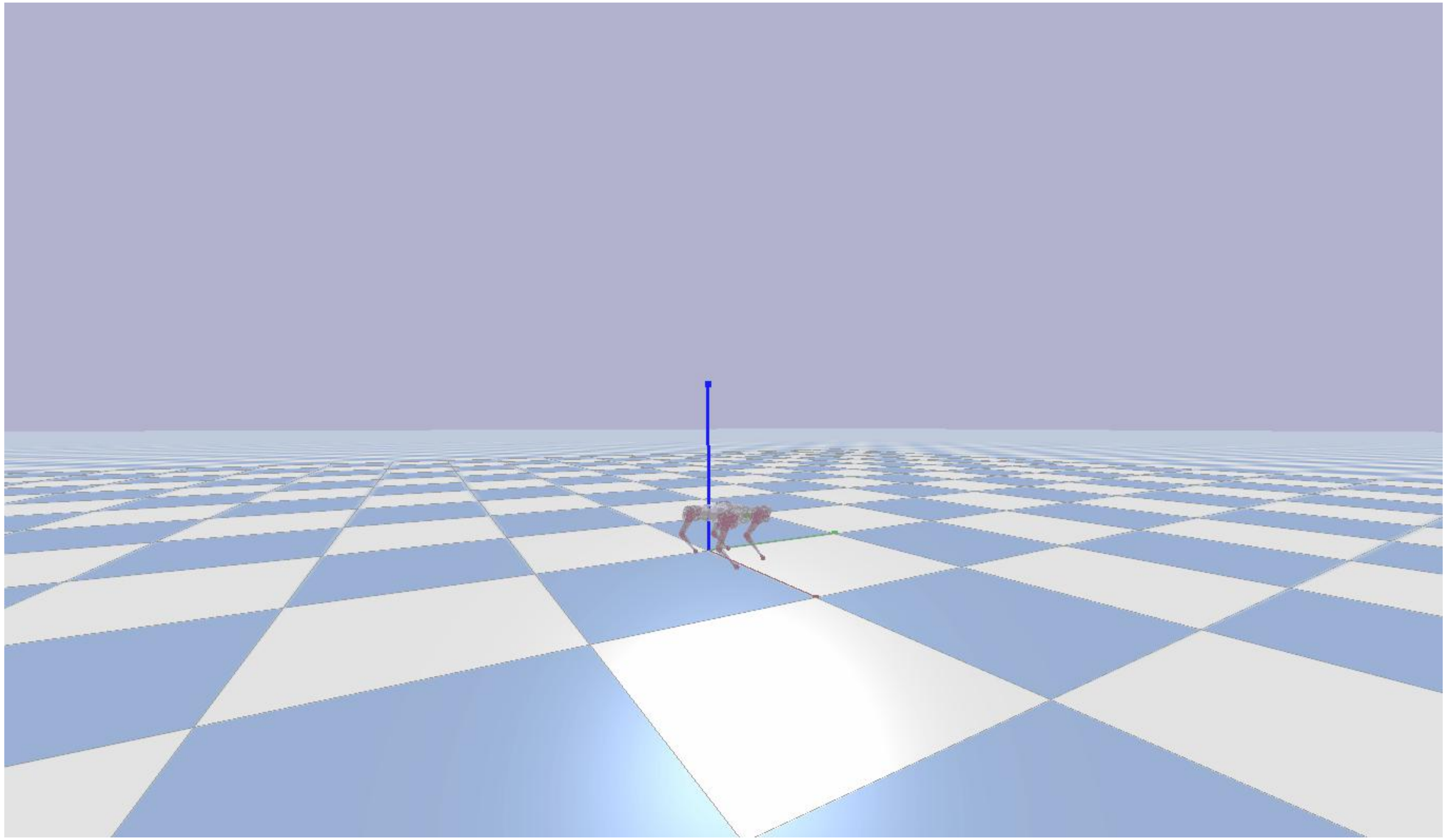
仿真方法与结果



无MPC控制器
摆动相采用PD控制
力矩跟踪轨迹



第一代MPC仿真



通过改变步长来调整速度

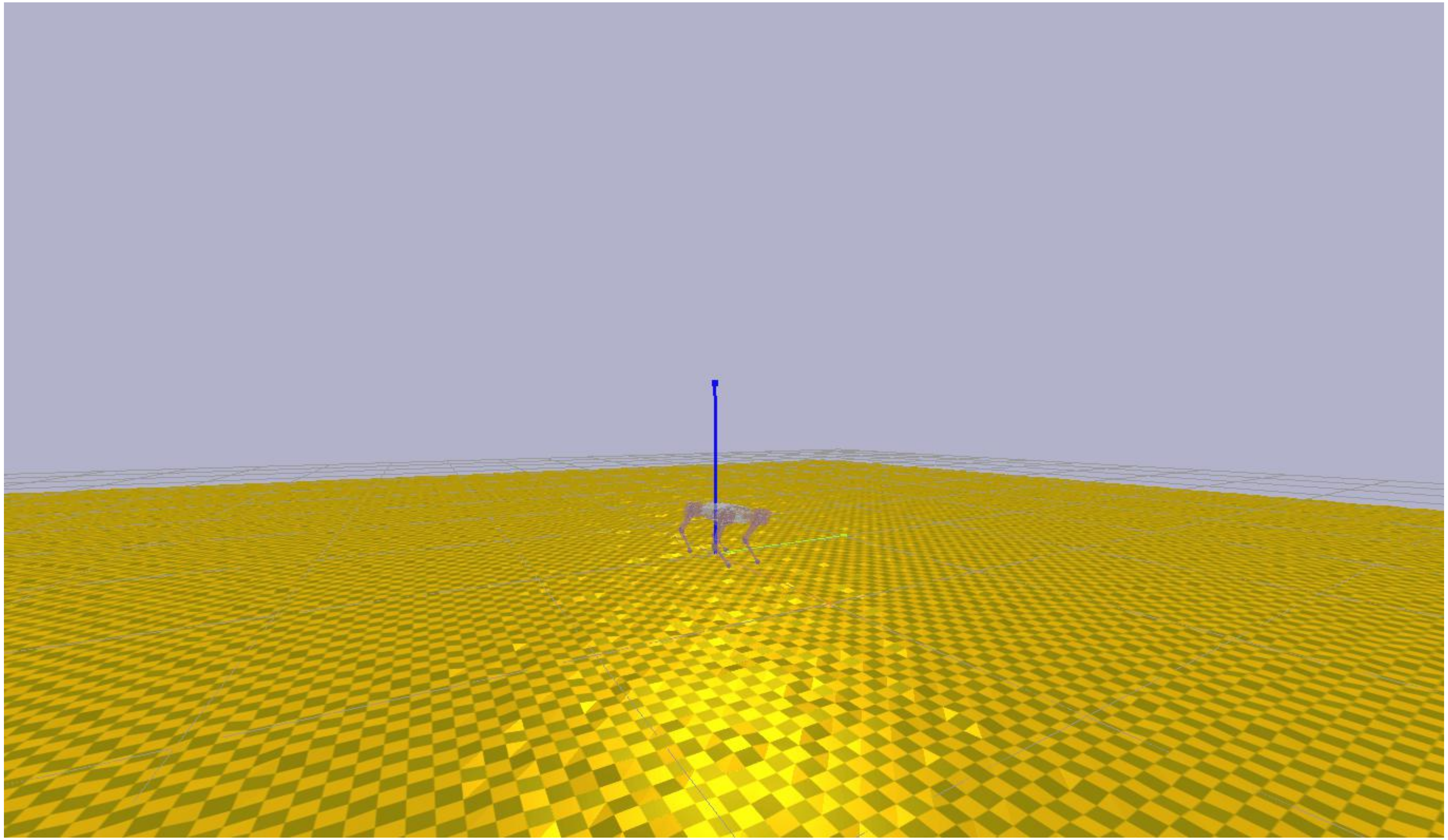
扰动：

（1）在一定时间段施加50N大小的侧向推力

（2）用鼠标任意拖动

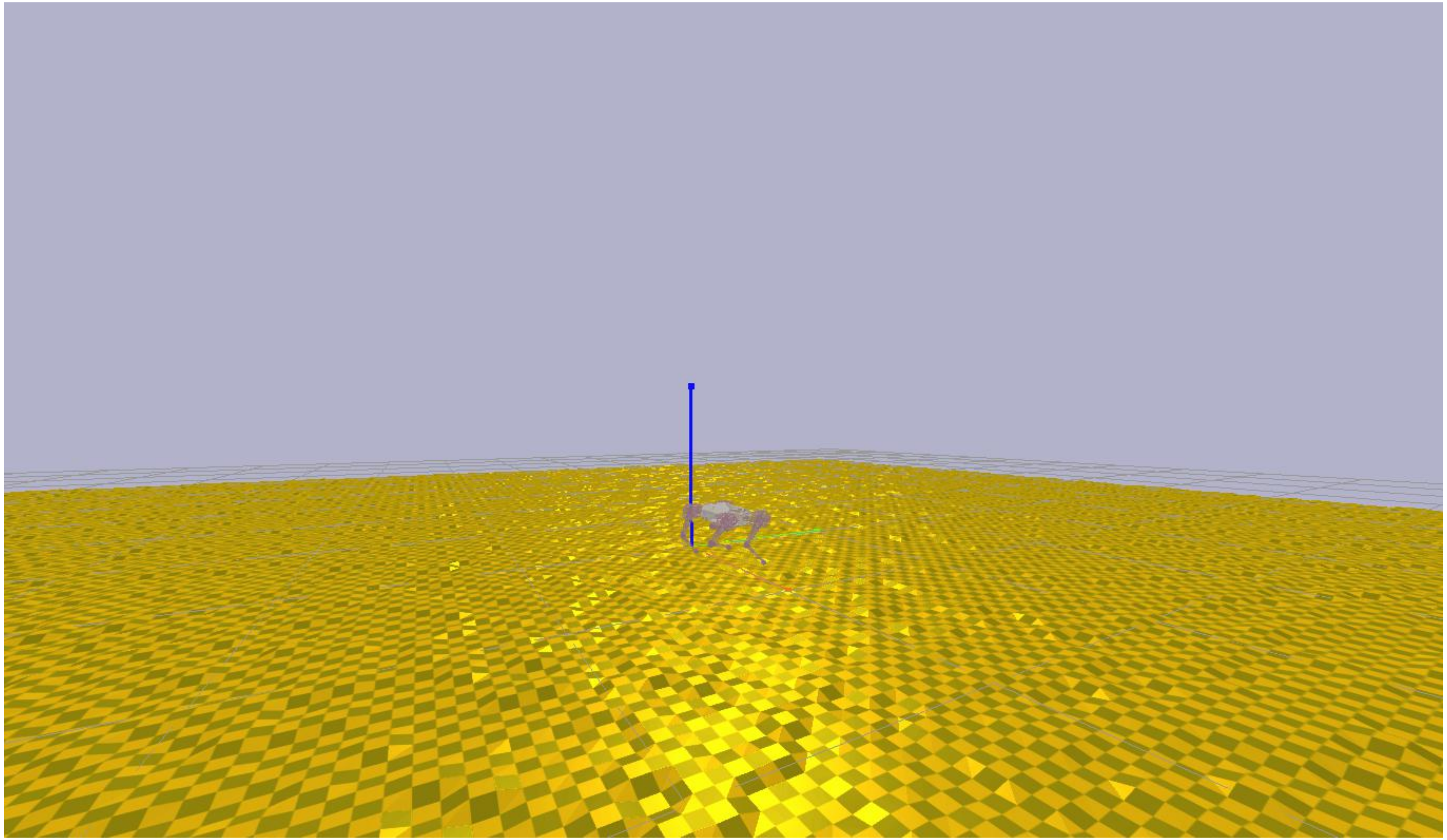
崎岖地形测试

dH = random(0, 0.01):



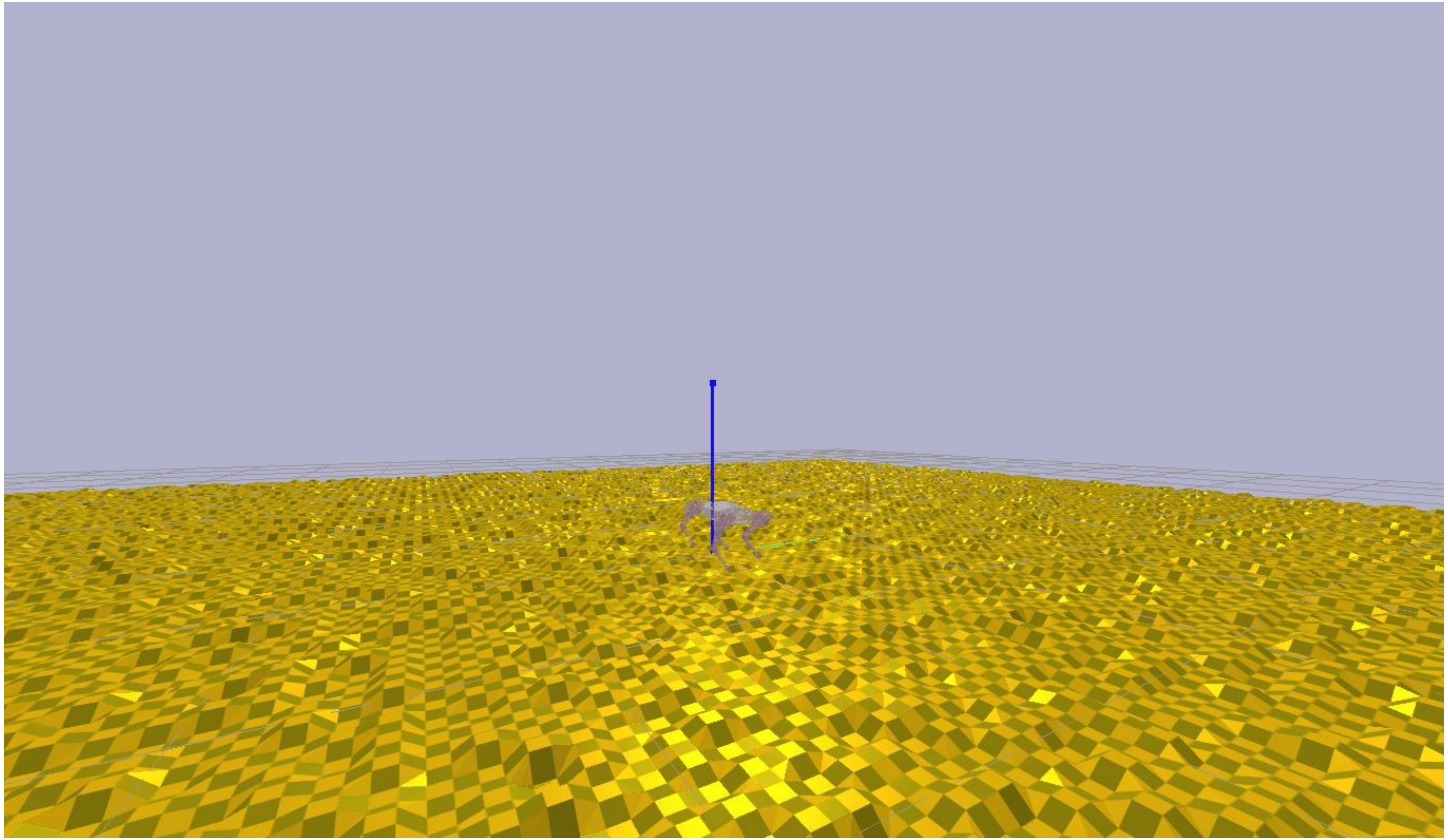
崎岖地形测试

dH = random(0, 0.03):

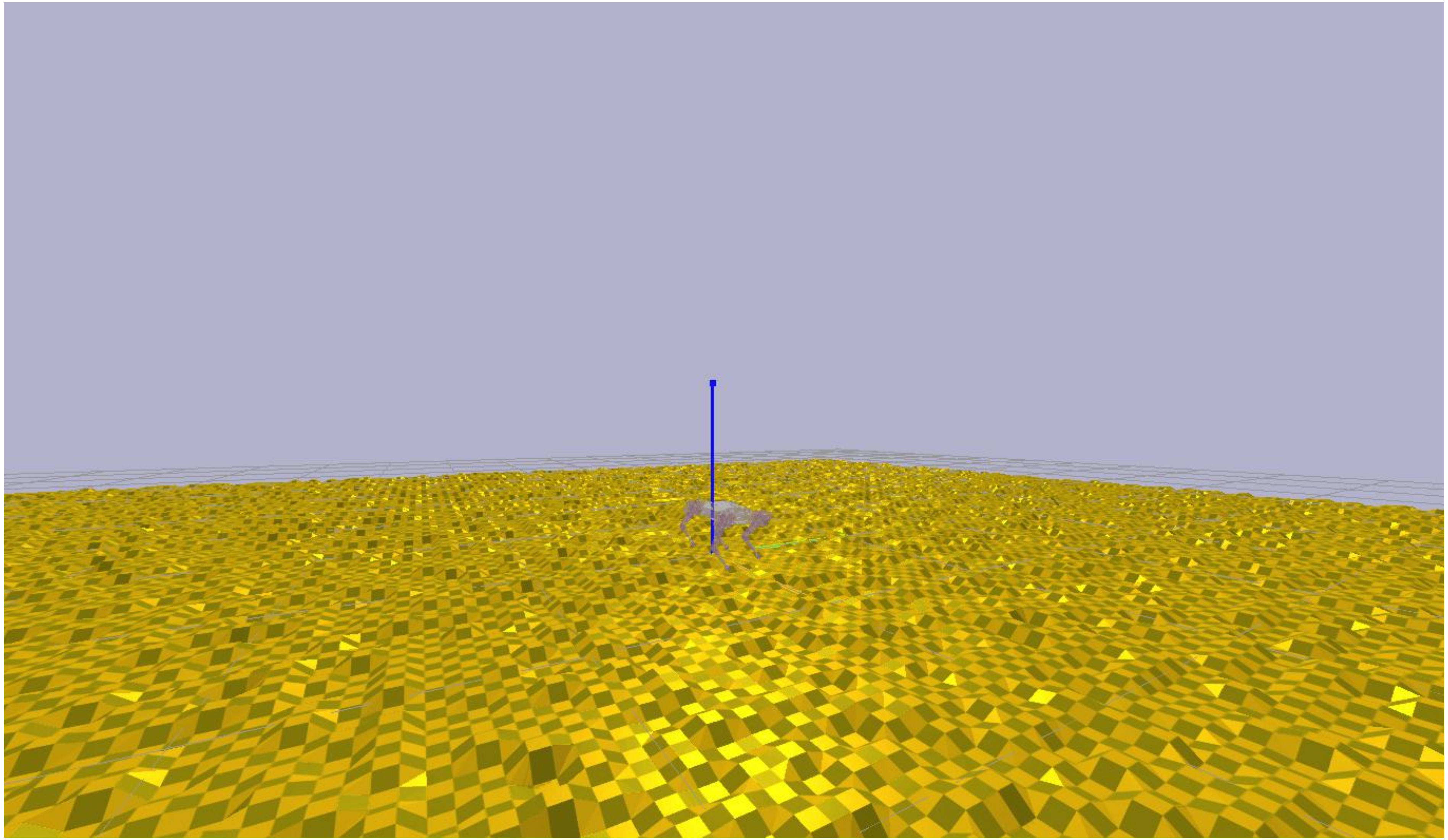


崎岖地形测试

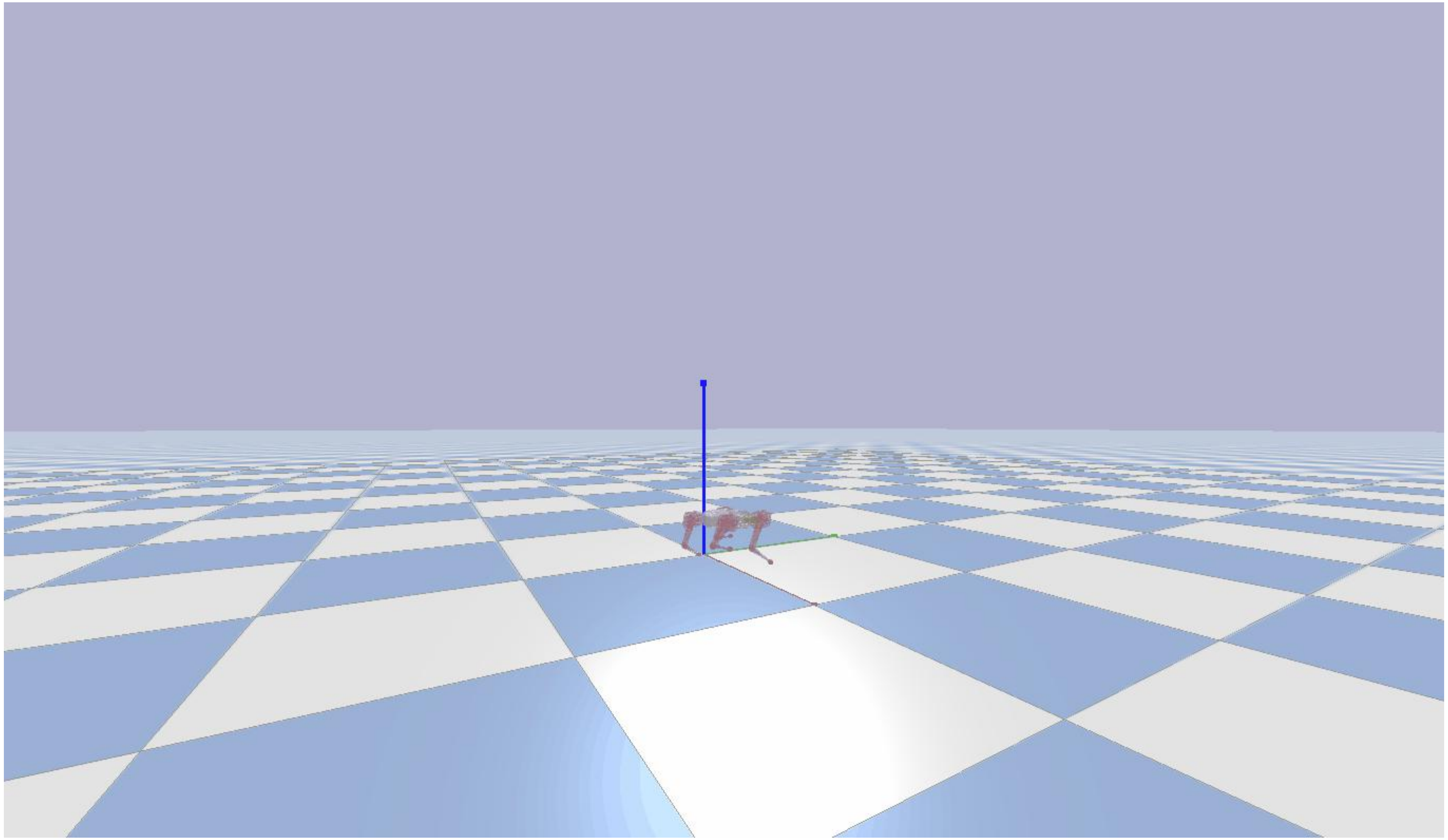
dH = random(0, 0.05):



崎岖地形测试 dH = random(0, 0.07):

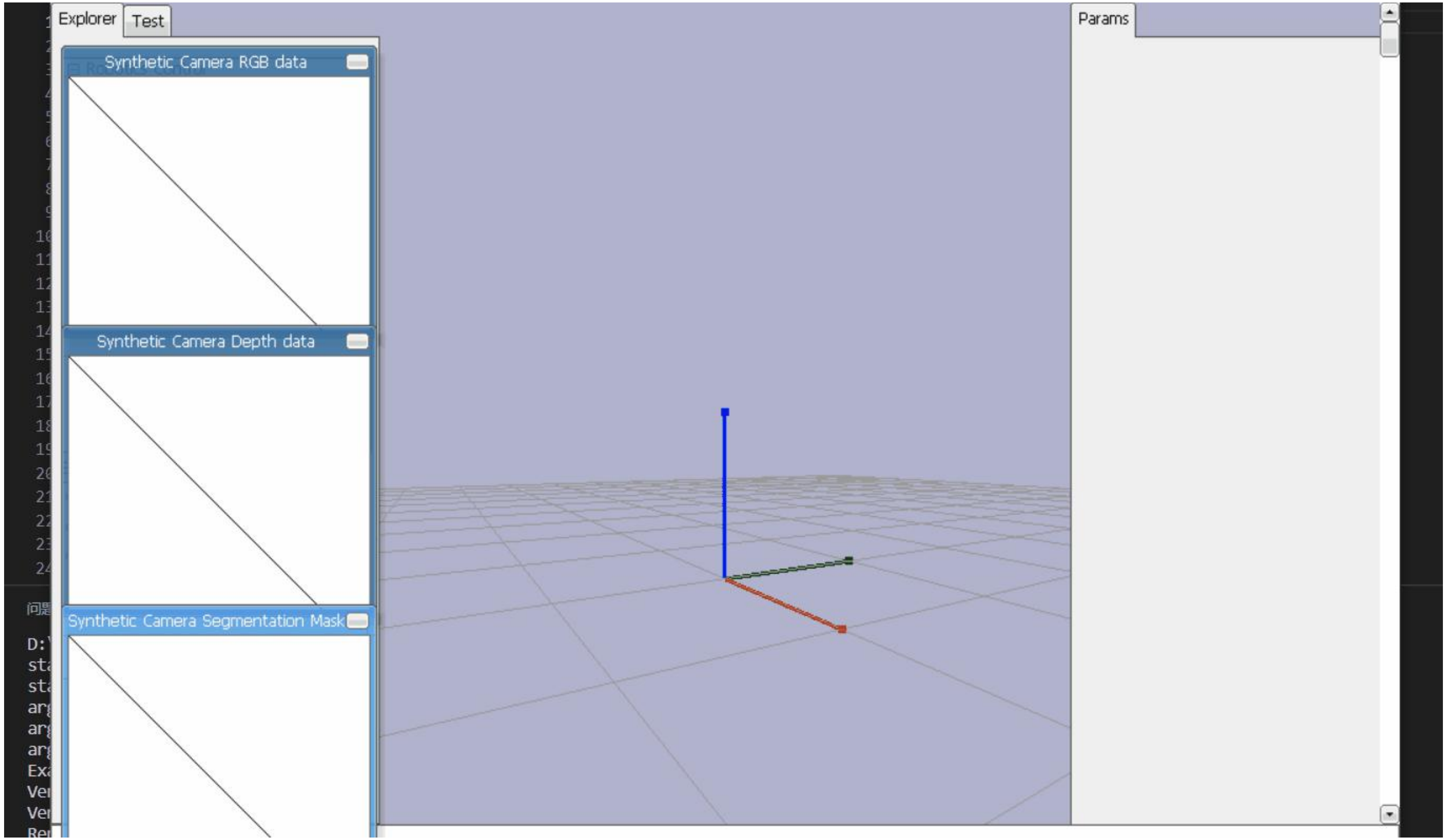


第二代MPC仿真



崎岖地形测试

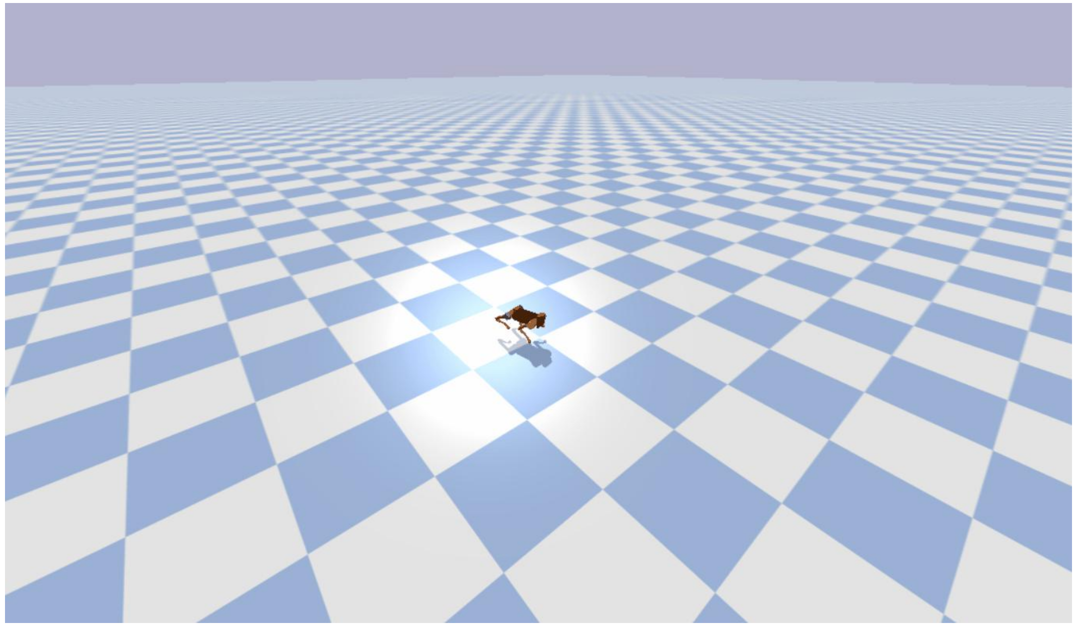
dH = random(0, 0.01):

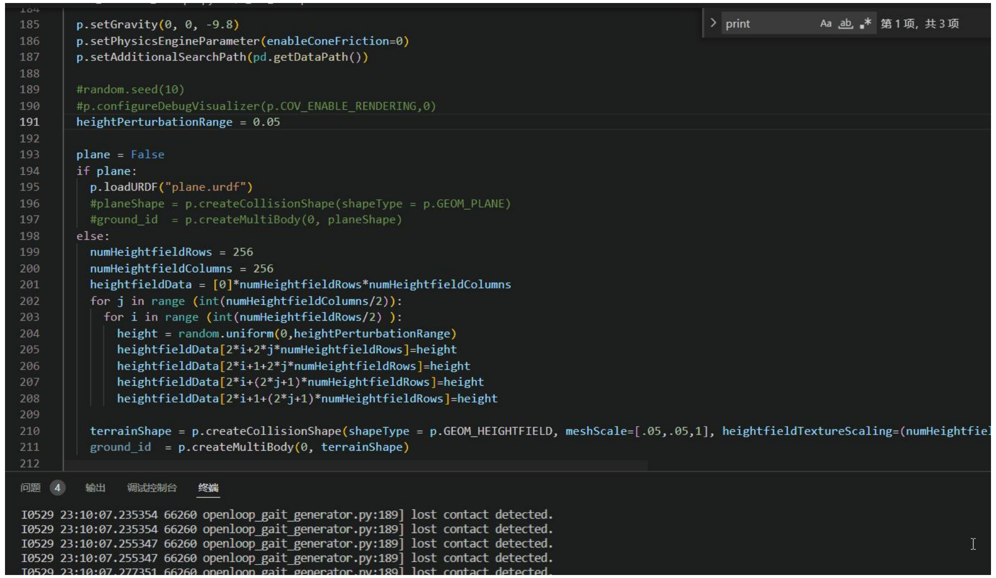


问题总结

Google开源代码：平地+崎岖地形测试

https://github.com/erwincoumans/motion_imitation



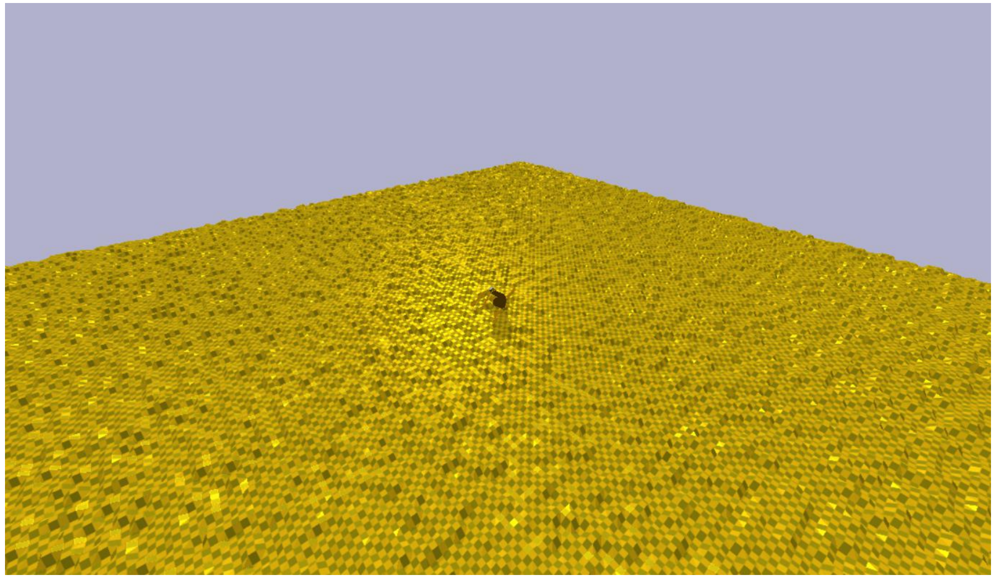


```python
p.setGravity(0, 0, -9.8)
p.setPhysicsEngineParameter(enableConeFriction=0)
p.setAdditionalSearchPath(pd.getDataPath())

#random.seed(10)
#p.configureDebugVisualizer(p.COV_ENABLE_RENDERING,0)
heightPerturbationRange = 0.05

plane = False
if plane:
  p.loadURDF("plane.urdf")
  #planeShape = p.createCollisionShape(shapeType = p.GEOM_PLANE)
  #ground_id  = p.createMultiBody(0, planeShape)
else:
  numHeightfieldRows = 256
  numHeightfieldColumns = 256
  heightfieldData = [0]*numHeightfieldRows*numHeightfieldColumns
  for j in range (int(numHeightfieldColumns/2)):
    for i in range (int(numHeightfieldRows/2) ):
      height = random.uniform(0,heightPerturbationRange)
      heightfieldData[2*i+2*j*numHeightfieldRows]=height
      heightfieldData[2*i+1+2*j*numHeightfieldRows]=height
      heightfieldData[2*i+(2*j+1)*numHeightfieldRows]=height
      heightfieldData[2*i+1+(2*j+1)*numHeightfieldRows]=height

  terrainShape = p.createCollisionShape(shapeType = p.GEOM_HEIGHTFIELD, meshScale=[.05,.05,1], heightfieldTextureScaling=(numHeightfiel
  ground_id  = p.createMultiBody(0, terrainShape)
```



```
I0529 23:10:07.235354 66260 openloop_gait_generator.py:189] lost contact detected.
I0529 23:10:07.235354 66260 openloop_gait_generator.py:189] lost contact detected.
I0529 23:10:07.255347 66260 openloop_gait_generator.py:189] lost contact detected.
I0529 23:10:07.255347 66260 openloop_gait_generator.py:189] lost contact detected.
```

dH = random(0, 0.05):



dH = random(0, 0.07):

问题总结

1. 缺少基于卡尔曼滤波或滑窗滤波的状态估计器

2. 缺少关于在理论上的被近似小量（如roll和pitch）的PD控制

3. 缺少关于基于事件触发的状态机切换机制（EARLY_CONTACT or LOSE_CONTACT）

4. 关于摆动腿的摆动控制机制，应参考Marc Raibert的相关文献

5. 关于摆动腿的位置控制前馈项不准确

李奥齐：负责仿真前准备工作和机器人仿真环境的搭建

江轶豪：负责理论支撑和模型预测控制代码框架的编写搭建

郭俊德：负责轨迹设计和Pybullet函数接口的实现

杨博文：负责代码调试和多类型仿真环境的测试

田丰：负责代码编写和多类型仿真环境的测试