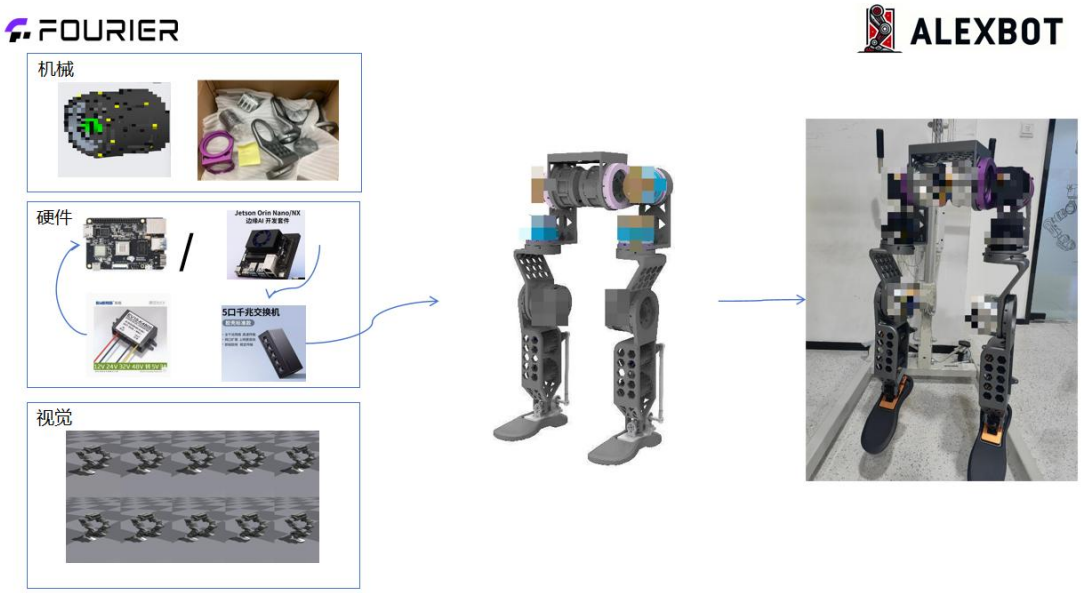
双足机器人简易全技能栈

暨 fftai_alexbot_mini 说明文档

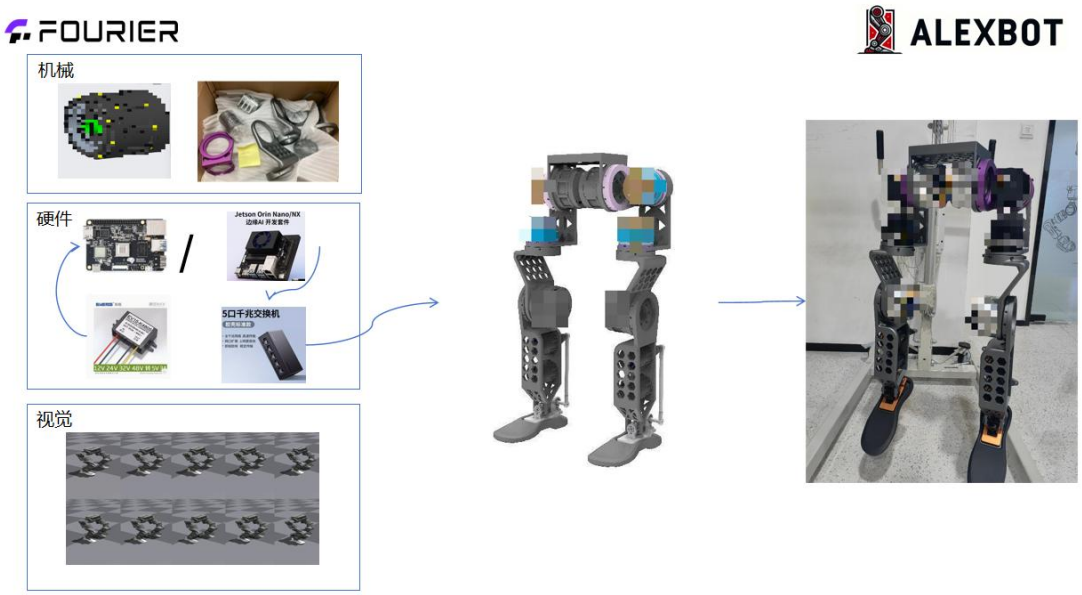


fftai_alexbot_mini 是一款拟打算全开源的双足机器人（包含机械，硬件，控制，算法），重点在于研究步态算法以及轻松部署步态降低 RealityGap 而设计的，整机 fftai_alexbot_mini 下半身长度 700mm，上半身长度 500mm，整体符合人体比例设计。

https://github.com/Alexhuge1/fftai_alexbotmini

目录

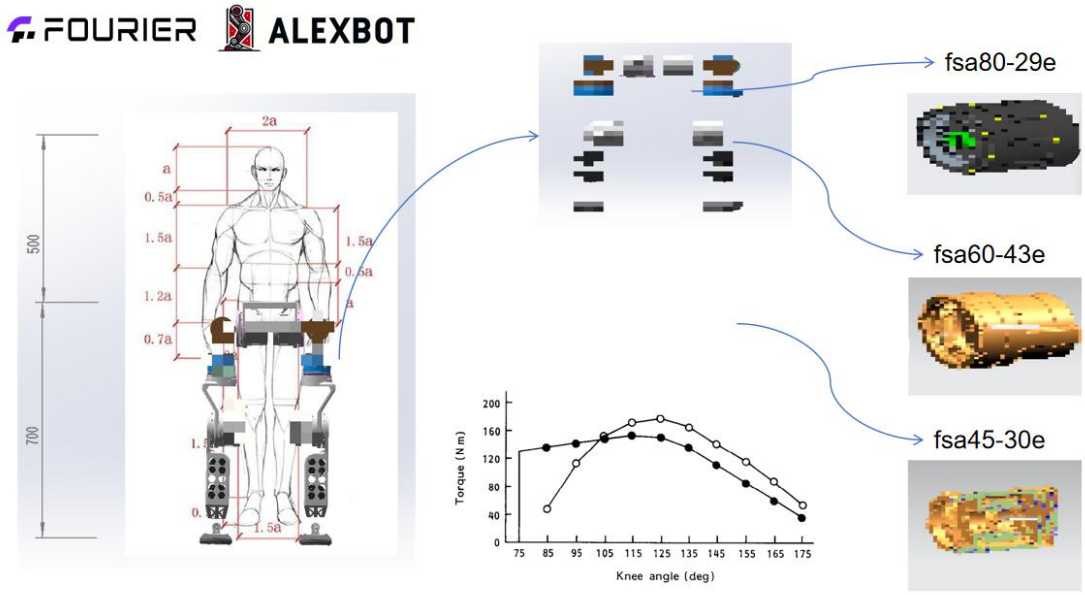
1. 本体整体设计



1.1 机械设计

（1）双腿结构

fftai_alexbot_mini 双腿包含 12 个自由度全长 700mm，预计重量<30kg，符合全身人体结构比。



根据人体的生物学结构以及生物力学知识，正常行走时髋关节的动作平衡且有节奏，耗能最低。双髋轮流负重，重心左右来回移动约 4.0～4.5 cm。髋关节在步态周期过程中会有两个受力波峰，分别在足后跟着地及趾尖离地时。缓慢行走时，惯性力作用可不计，视与静力学相同。但髋关节在快速运动时，受加速和减速的作用，受力会增加。合力等于体重加惯性力，包括地面反冲力、重力、加速度、肌力等，一般认为是体重的 3.9～6.0 倍。在走路时(速度为 1.5m/s)，髋关节最大受力约为 2.5 倍体重，而当跑步时(速度为 3.5m/s)，关节最大受力约为 5～6 倍体重。

Stam 等（见 REF）研究了等速运动对膝伸肌力矩曲线的影响，运用 Cybexl 等速测力仪对 20 名受试者进行主动和被动测试。发现主动模式下，关节角在 119°时产生的平均力矩值为 158Nm；被动模式下，关节角度在 123°时的平均力矩值为 179Nm，因此两种模式下的平均力矩有较大的差异。

性别	发力方向	范围（Nm
男	伸髋	157-409
	屈髋	98-297
女	伸髋	47-277
	屈髋	35-187

性别	发力方向	范围（Nm）
男	髋外展	81-254 ()
	髋内收	68-312
女	髋外展	47-198
	髋内收	39-194

性别	发力方向	范围（Nm）
男	髋外旋	30-83
	髋内旋	25-111
女	髋外旋	15-58
	髋内旋	11-72

性别	发力方向	范围（Nm）
男	踝关节背伸	16-49
	踝关节跖屈	53-184
女	踝关节背伸	11-34
	踝关节跖屈	15-127

性别	发力方向	范围（Nm）
男	伸膝	98-309
	屈膝	57-178
女	伸膝	57-202
	屈膝	31-113

因此，对于本体重量约等于人类 1/2，身高约等于人类的 2/3 的小型双足机器人扭矩要求选用如下：对于扭矩要求最高的胯部和膝关节部分电机选用 FSA80-29E，对于大腿部分选用 FSA60-43E，对于小腿，为了惯量的设计选用 FSA45-30E 以及并联结构，采用双电机朝向

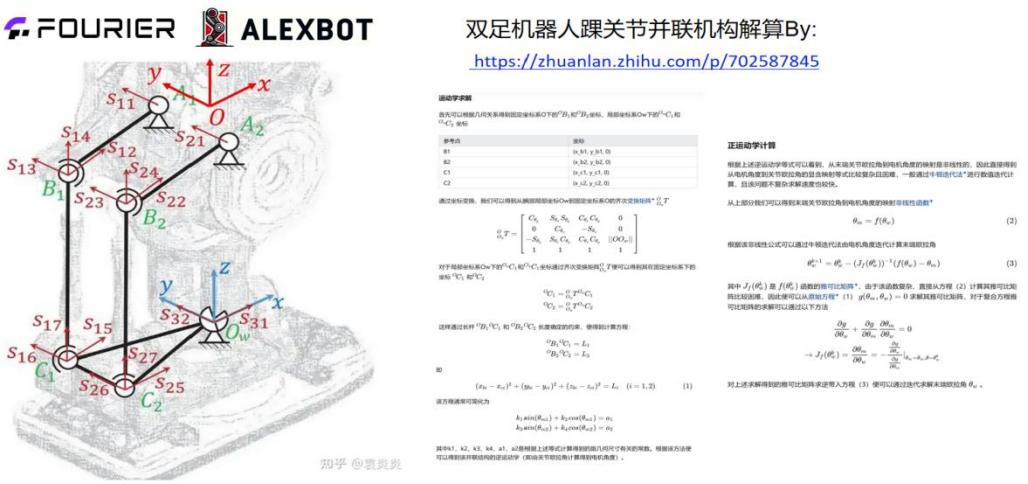
后方，通过板簧连接至脚底结构。

同时关于电机的串并联的选择，在电机扭矩十分金贵的时候，机器人时常采用并联结构从而降低整体惯量从而降低对于电机的扭矩的要求，并链结构不仅能放大力矩输出，还能通过连杆机构将电机上置，减少驱动器末端负载。

但是机器人结构采用并联往往会带来许多麻烦，首先机械上多了许多层级传动从而会增加机器人背隙，由于齿轮间隙或其他组件间隙造成的非预期运动或自由运动，过大的背隙会导致控制系统的精度下降，影响设备的性能和可靠性。

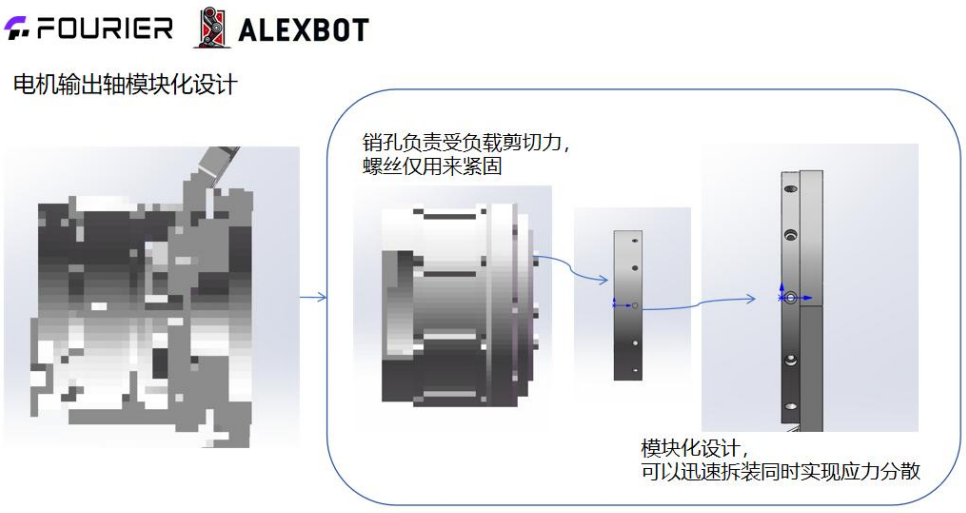
其次在机械导出 URDF 文件中会存在并联需要转移输出轴的做法，由于算法上位机解算的回传数据本质是位置环，所以会存在需要姿态逆解算的问题，而这样的并联机构一方面相比于单个平行四边形难以从几何学直接计算，另一方面其电机角度到空间欧拉角度的映射是非线性的，因此通常逆运动学比较好解（空间欧拉角到电机角度），而正运动学（电机角度到空间欧拉角）大多需要数值迭代求解。往往需要通过数值解去逼近。

处理方法参考：https://zhuanlan.zhihu.com/p/702587845

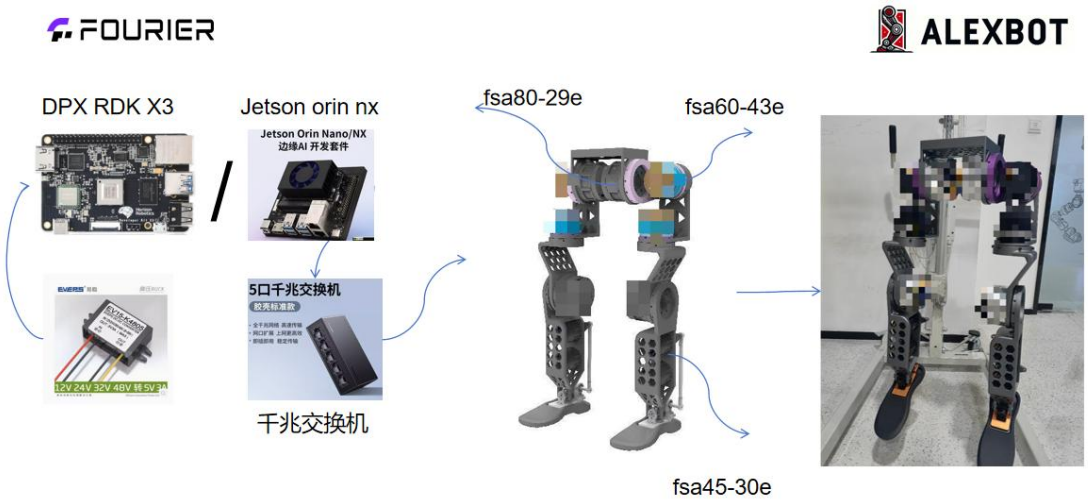


本质上，并联结构的逆解比正解好算，正解通过牛顿法迭代出数值解即可

同时，对于电机的输出侧使用了模块化快拆结构，方便拆卸和组装机器人，采用输出盘的类似设计，使用销结构与螺钉实现快拆，同时更换电机时只需要更换法兰盘即可，以膝关节为例:



1.2 硬件和驱动架构设计



1.2.1 硬件架构设计

硬件整体比较简单，分为信号层和功率层:

(1)信号层面上：主要包含上位机去推理神经网络，然后通过交换机来进行数据交换以实现一（上位机）拖多（电机）的情况，电机采用固定 ip 地址，上位机采用交换机进行数据交换，整体线路连接简单。

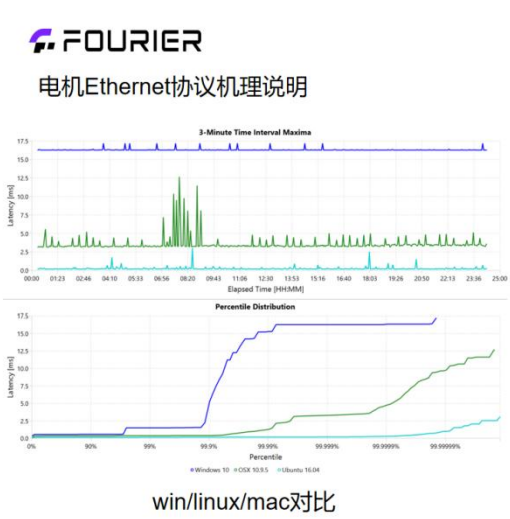
(2)功率层面上：实现硬件隔离，防止电机电流过大反冲烧坏上位机，同时电机采用 48v 供电，上位机采用 12v 供电，交换机采用 5-60v 宽幅输入

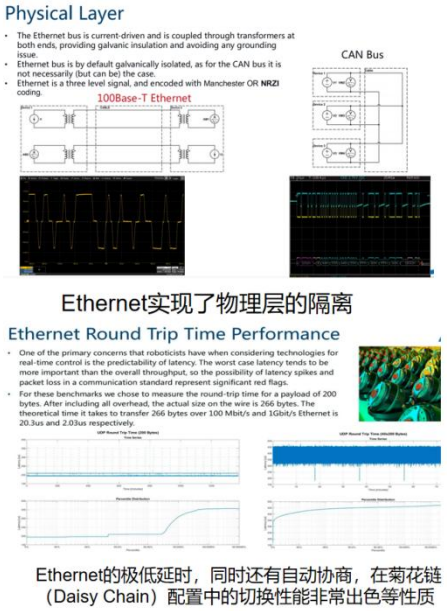
1.2.2 电机 Ethernet 协议说明（用过就回不去了）

得益于简易的电机控制框架，电机走的 Ethernet 协议实现了对于硬件层的封装，实现了简易的操作和快速上手的体验同时降低了电机的 lateancy，如下方论文所述。

左侧图片都包含两个图表。上部显示在给定时间间隔窗口内发生的最严重故障，即第一个数据点显示前 3 分钟内发生的最严重故障，下一个数据点显示后 3 分钟内发生的最严重故障。下图显示了整个持续时间内所有测量值的百分位数。每个 24 小时数据集大约包含 7000-8000 万个样本。以太网还具有诸如低延时，多电机级联，冗余管理链路，无需下位机（现代机器人通常由嵌入式计算机控制以太网端口）。

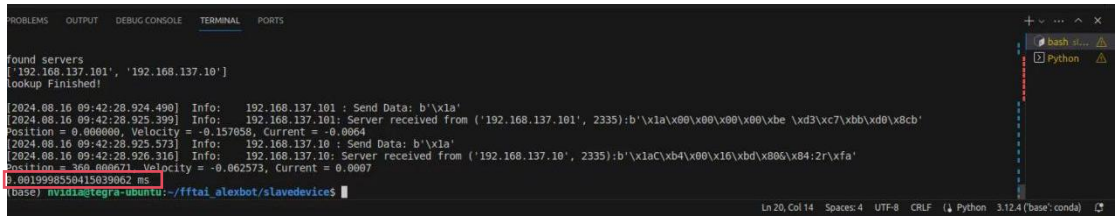
与 CAN 总线的数据帧（只有 8 字节）相比，以太网可以处理更大的数据负载，无需在数据帧中节省字节。同时支持多关节机器人的通信拓扑、网络通信编程的简便性、OTA（Over-The-Air）升级、基于 JSON 的通信协议，方便阅读等等。总之是一个好的协议以及电机控制路线。





https://ennerf.github.io/2016/09/20/A-Practical-Look-at-Latency-in-Robotics-The-Importance-of-Metrics-and-Operating-Systems.html

框架主要是修改了 FFTAI 的电机 sdk：https://gitee.com/FourierIntelligence/wiki-fsa，整体主要采用了 fi_fsa_v2.fast_set_position_control()和 fi_fsa_v2.fast_get_pvc();



fi_fsa_v2.fast_set_position_control()的系统延时

fi_fsa_v2.fast_get_pvc()的系统延时

可以看到，整体的延时还是十分低的，对于 RL 等对控制频率不那么敏感的使用场景来说，我期望的控制频率在 50hz 左右，整体间隔 2ms，这样的控制频率加上电机和 imu 还有模型推理的时间是足够的。

更进一步的框架和代码见 2.2 Simtoreal 的部分

2.软件整体设计

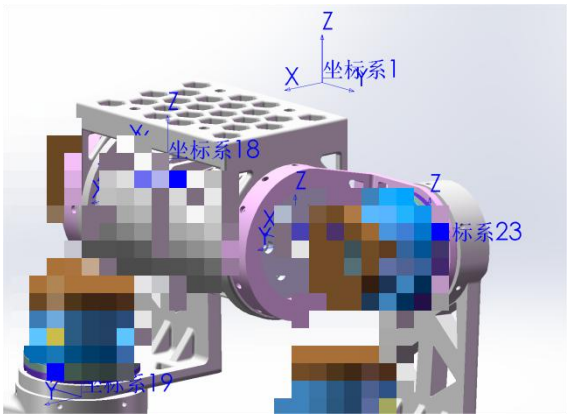
软件整体包含

1. 步态设计
2. Simtosim
3. 3.Simtoreal 的部分

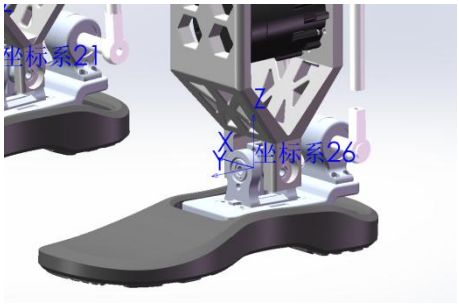
软件设计框图

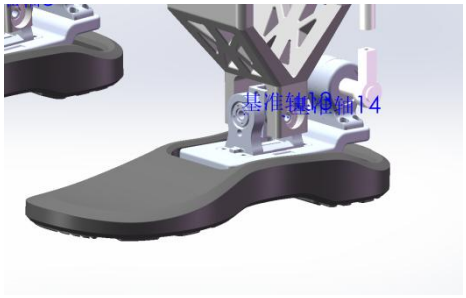
2.1.1 URDF 文件导出注意点

（1）注意导出的 baselink 方向，右手系以及 x 轴朝前，为前进方向。然后 baselink 的坐标系所在点为 imu 所在点！

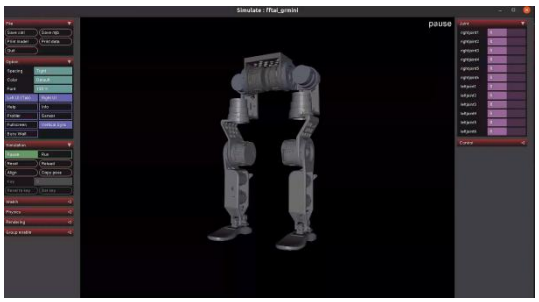


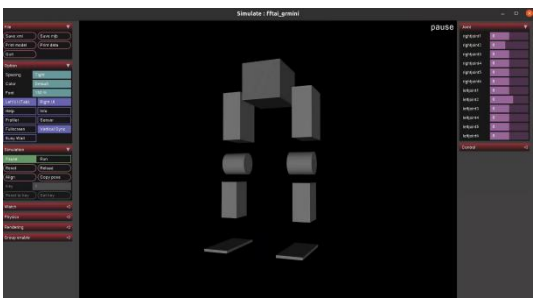
（2）导出脚踝的 urdf 的时候采用虚拟轴设计，使用同一个坐标系，然后双轴，一个 pitch，一个 roll，这里导出的是 leftlink5 是小的中间的那个十字轴，leftlink6 是脚底板。





（3）urdf 在导入仿真环境的时候先使用 mujoco 的 simulate 检查一下，mujoco 的物理引擎比较真实。如果模型文件的 stl 文件面数过多，使用 meshlab 简化一下到 15000 面即可；同时模型 urdf 文件的 collision 文件需要做简化，降低无用的以及自我的碰撞同时并不影响实际输出和 simtoreal 部署。





2.1.2 步态规划设计

步态规划采用 humanoidgym：https://github.com/roboterax/humanoid-gym，实现的步态效果如下（可以见 ppt/github/知乎文档）：
其中主要的 config 设计大体上基于 XBot 的 example，有几个比较重要的参数需要修改一下

（1）这个是机器人的参数设定，包含了维数以及训练参数

```python
class alexbotminiCfg(LeggedRobotCfg):
    """
    Configuration class for the alexbotmini humanoid robot.
    """
    class env(LeggedRobotCfg.env):
        # change the observation dim
        frame_stack = 15
        c_frame_stack = 3
        num_single_obs = 47
        num_observations = int(frame_stack * num_single_obs)
        single_num_privileged_obs = 73
        num_privileged_obs = int(c_frame_stack * single_num_privileged_obs)
        num_actions = 12
        num_envs = 1000
        episode_length_s = 24  # episode length in seconds
        use_ref_actions = False
```

（2）这个是机器人的初始化设定，尽可能使用微微下蹲作为初始状态

```python
class init_state(LeggedRobotCfg.init_state):
    pos = [0.0, 0.0, 0.70]

    default_joint_angles = {  # = target angles [rad] when action = 0.0
        'leftjoint1': -0.2,
        'leftjoint2': 0.,
        'leftjoint3': 0.,
        'leftjoint4': 0.6,
        'leftjoint5': -0.4,
        'leftjoint6': 0.,
        'rightjoint1': 0.2,
        'rightjoint2': 0.,
        'rightjoint3': 0.,
        'rightjoint4': -0.6,
        'rightjoint5': 0.4,
        'rightjoint6': 0.,
    }

class control(LeggedRobotCfg.control):
    # PD Drive parameters:
    stiffness = {'1': 120.0, '2': 80.0, '3': 80.0, '4': 120.0, '5': 15 , '6': 15}
    damping = {'1': 3, '2': 2, '3': 2, '4': 3, '5': 0.3, '6' : 0.3}
    # action scale: target angle = actionScale * action + defaultAngle
    action_scale = 0.25
    # decimation: Number of control action updates @ sim DT per policy DT
    decimation = 10  # 100hz
```

（3）这个是机器人的 reward，细节 reward 代表什么可以看 env 的 reward 的 class

```python
class rewards:
    base_height_target = 0.65
    min_dist = 0.2
    max_dist = 0.5
    # put some settings here for LLM parameter tuning
    target_joint_pos_scale = 0.17    # rad
    target_feet_height = 0.04       # m
    cycle_time = 0.64                # sec
    # if true negative total rewards are clipped at zero (avoids early termination problems)
    only_positive_rewards = False
    # tracking reward = exp(error*sigma)
    tracking_sigma = 5
    max_contact_force = 200  # Forces above this value are penalized

    class scales:
        # reference motion tracking
        joint_pos = 2.0
        feet_clearance = 2.0
        feet_contact_number = 2.5
        # gait
        feet_air_time = 2.5
        foot_slip = -0.12
        feet_distance = 0.2
        knee_distance = 0.2
        # contact
        feet_contact_forces = -0.01
        # vel tracking
        tracking_lin_vel = 1.4
        tracking_ang_vel = 1.1
        vel_mismatch_exp = 0.5  # lin_z; ang x,y
        low_speed = 0.2
        track_vel_hard = 0.5
        # base pos
        default_joint_pos = 0.35
        orientation = 1.
        base_height = 0.2
        base_acc = 0.2
        # energy
        action_smoothness = -0.002
        torques = -1e-5
        dof_vel = -5e-4
        dof_acc = -1e-7
        collision = -1.
```

5. Rewards设计

5.1 第一部分，涉及 reference motion tracking

函数名	函数功能
_reward_joint_pos	处理当前关节角与参考关节角的差距。包含两个部分，指数形式的正向奖励+线性函数惩罚。当值比较小时，指数函数发挥作用。当值变大后，指数函数变得平坦。这时，线性函数发挥作用。
_reward_feet_clearance	当抬脚高度达到指定高度，并且此时位于摆动相，则获得奖励。
_reward_feet_contact_number	触地的脚的数量是否符合规划的步态相位。如果符合获得奖励，否则获得惩罚。

5.2 第二部分，涉及 gait

函数名	函数功能
_reward_feet_air_time	将脚的滞空时间作为奖励值返回，最大值限制为0.5秒。即鼓励机器人每一步在空中停留0.5秒。
_reward_foot_slip	在脚底板触地的同时，如果在x轴、y轴上有速度，则会受到惩罚。
_reward_feet_distance	计算两脚之间的距离。如果小于下限，或者大于上限，就会受到惩罚。
_reward_knee_distance	控制两个膝盖之间的距离。效果同上。

5.3 第三部分， 涉及 contact

函数名	函数功能
_reward_feet_contact_forces	如果脚底板上的力超过上限，就会受到惩罚。

5.4 第四部分，涉及 velocity tracking

函数名	函数功能
_reward_tracking_lin_vel	对base在x轴、y轴上的线速度进行跟踪。
_reward_tracking_ang_vel	对base在yaw轴上的角速度进行跟踪。
_reward_vel_mismatch_exp	不希望base在Z轴上有线速度。不希望base在roll轴、pitch轴上有角速度。
_reward_low_speed	希望base在x轴上的线速度可以跟上command。如果实际速度 > 50% command且 < 120% command，就可以得到奖励。
_reward_track_vel_hard	同时进行对base在x轴、y轴上的线速度跟踪和在yaw轴上的角速度跟踪。类似于_reward_joint_pos，使用了指数形式的正向奖励+线性函数惩罚。

5.5 第五部分，涉及 base position

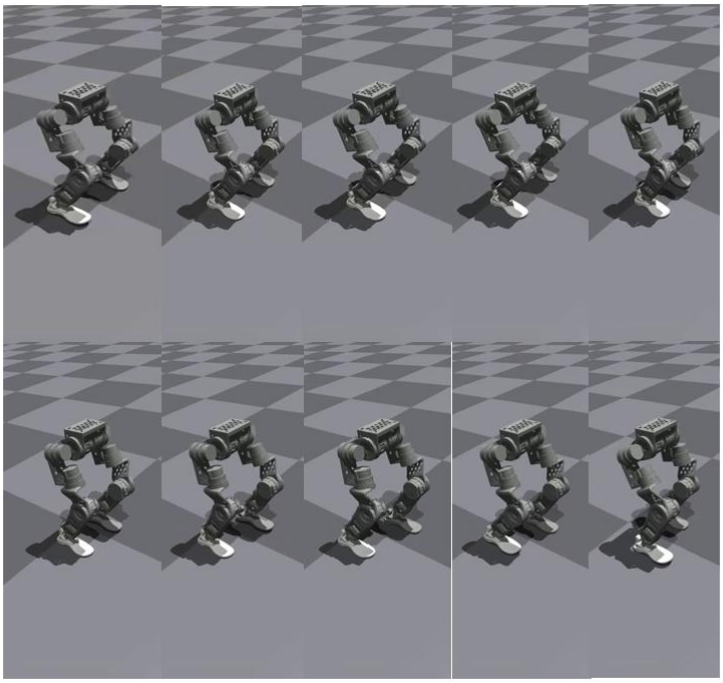
函数名	函数功能
_reward_default_joint_pos	包含两个部分。第一部分，惩罚所有关节位置与默认位置的偏差，权重较小。第二部分，惩罚髋关节的roll轴关节、yaw轴关节与默认位置的偏差，权重较大。
_reward_orientation	确保机器人base在roll轴上、pitch轴上没有转动。包含两个部分。第一部分，计算base在roll轴、pitch轴上的值，如果有值则得到惩罚。第二部分，计算机器人重力方向在xy平面上的投影，如果有值则得到惩罚。两部分功能相同，有冗余，但是可以增加可靠性和强健性。
_reward_base_height	计算base实际高度与目标高度的偏差。计算的过程中考虑到了脚底板的厚度，为5cm。
_reward_base_acc	对base的加速度进行限制，不希望base出现大的加速度。

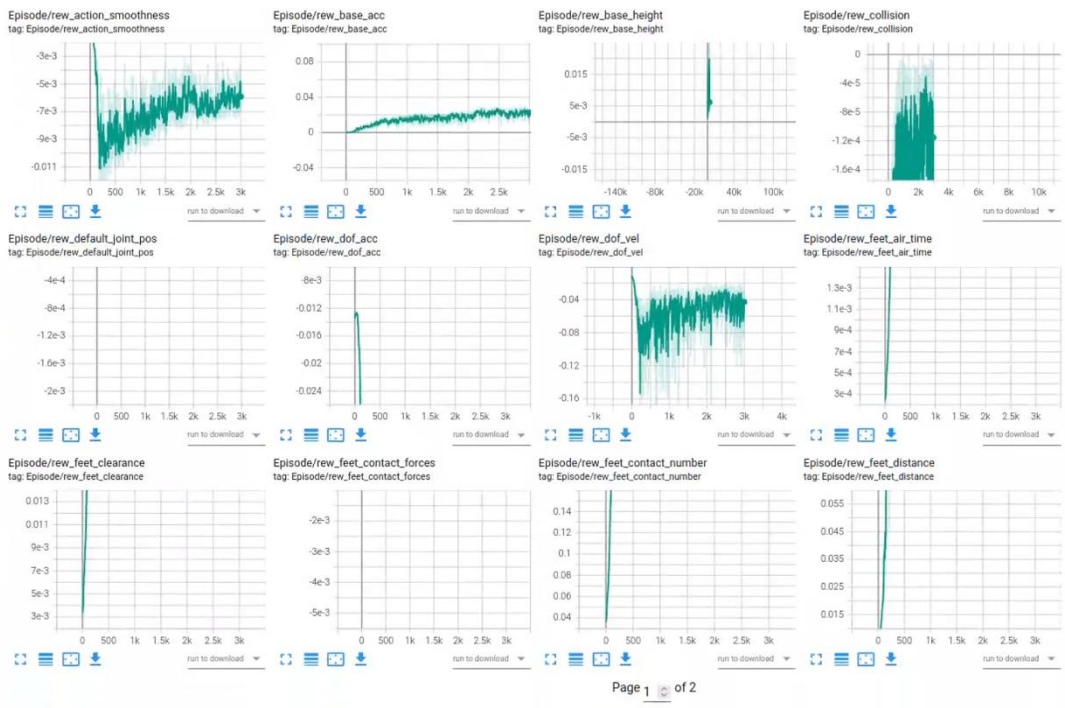
5.6 第六部分，涉及 energy

函数名	函数功能
_reward_action_smoothness	用来降低连续step的actions之间的差距，以产生连续的动作。分为三个部分。第一部分，一阶差分。当前action与上一个action之间的差距。效果类似于dof_vel。第二部分，二阶差分。 (action - last_action) - (last_acton - last_last_action)。效果类似于dof_acc。第三部分，action的绝对值。惩罚过大的动作幅度，鼓励小幅度、平滑的动作。
_reward_torques	惩罚过大的关节力矩。
_reward_dof_vel	惩罚过大的关节速度。
_reward_dof_acc	惩罚过大的关节加速度。
_reward_collision	避免base link发生碰撞，比如base link与手臂的碰撞。

2.1.2 步态规划效果

已开源：https://github.com/Alexhuge1/Alexbotmini_gait





2.1.3 Simtosim 迁移（From isaacgym to mujoco）

对于模型参数文件（.pt）的迁移来说，模型的 Reality gap 极为重要，通过不同仿真软件的迁移可以起到模型验证以及降低 gap 的作用，其中 isaacgym 所使用的 physicx 是支持并行运算的，非常适合模型训练，然后在 simtosim 迁移到 mujoco 中，如下图可以看到 mujoco 与实际环境的 gap 相对 isaacgym 小（参考：[2404.05695] Humanoid-Gym: Reinforcement Learning for Humanoid Robot with Zero-Shot Sim2Real Transfer (arxiv.org)）

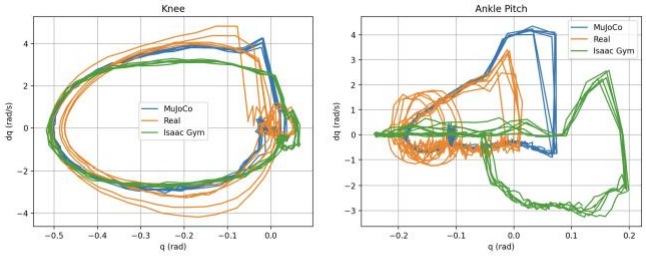


Fig. 4: Phase Portrait for MuJoCo, Real-World Environment, and Isaac Gym.