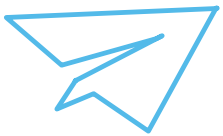
四足机器人步态设计及控制简介

吴畔昊

CONTENTS



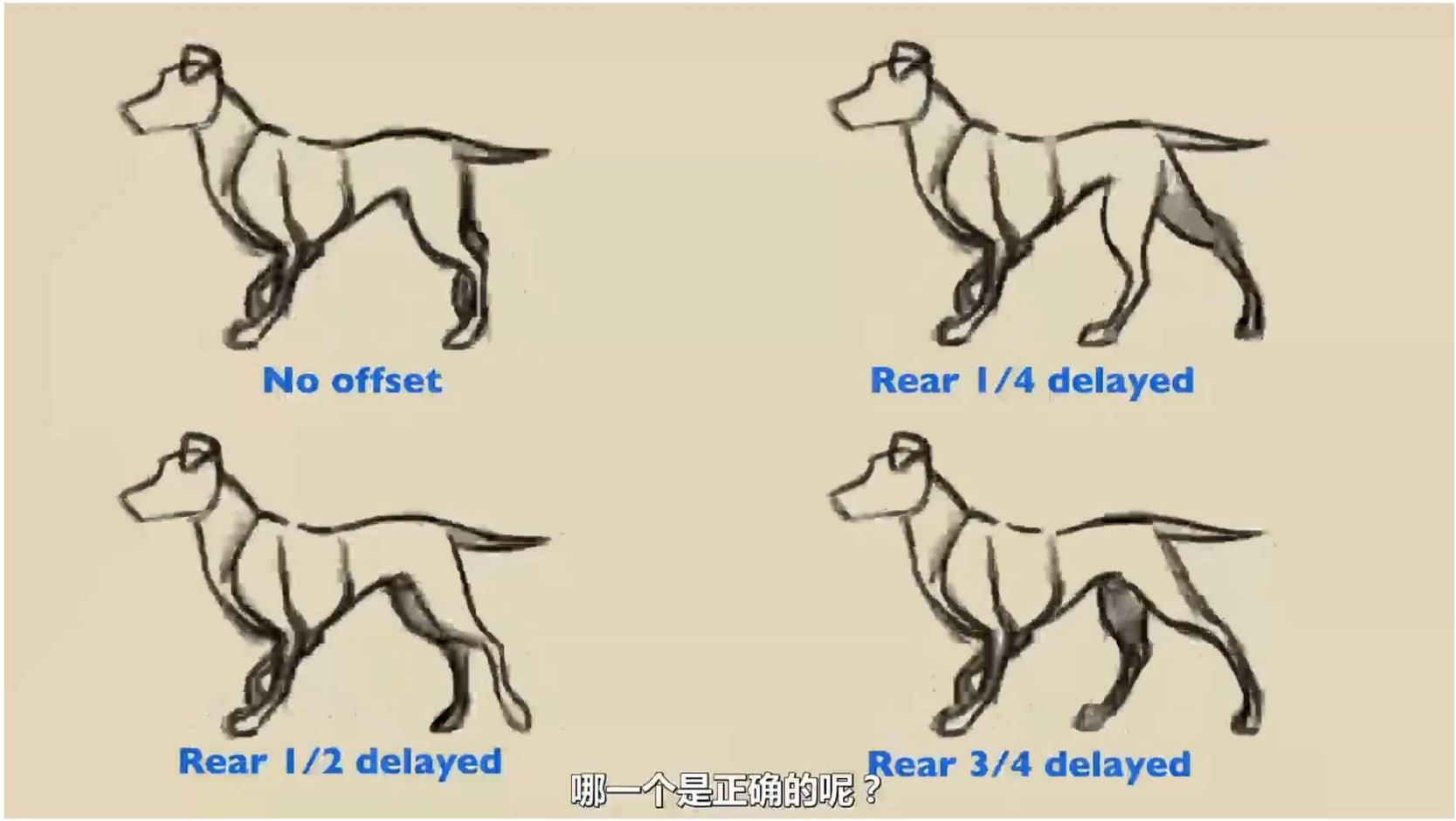
01 步态设计

Gait design

四足步态

- 回顾生活中常见四足动物的行走姿态(猫、狗、牛、马等)
- 如果让你来设计四足机器人的行走方式，你会怎么做?

四足步态


No offset
Rear 1/4 delayed
Rear 1/2 delayed
Rear 3/4 delayed
哪一个是正确的呢？

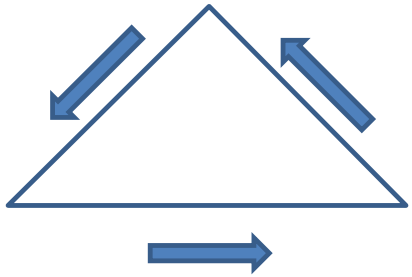
角度 or 轨迹

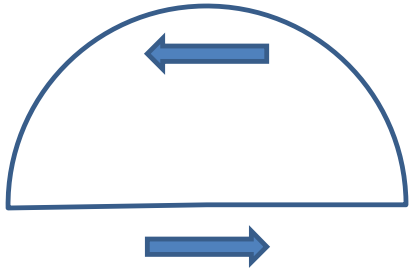
先考虑单只脚的运动，如何从数学层面描述选择的步态?

- 关节的角度变化函数 还是 足端的轨迹函数

角度函数不容易获取也不够直观，先设计位移轨迹函数再求取角度

如何通过位移轨迹函数求得关节角度？建立反向运动学方程





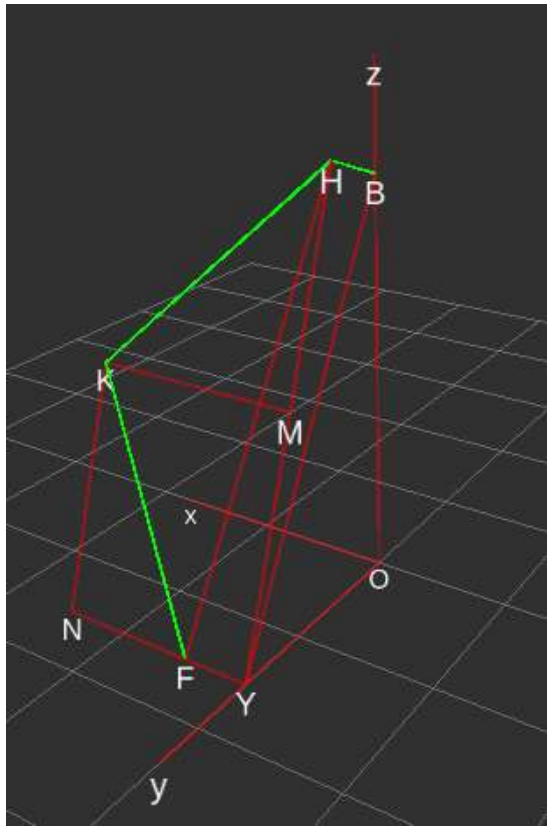
02 运动学建模与求解

Kinematics modeling and solution

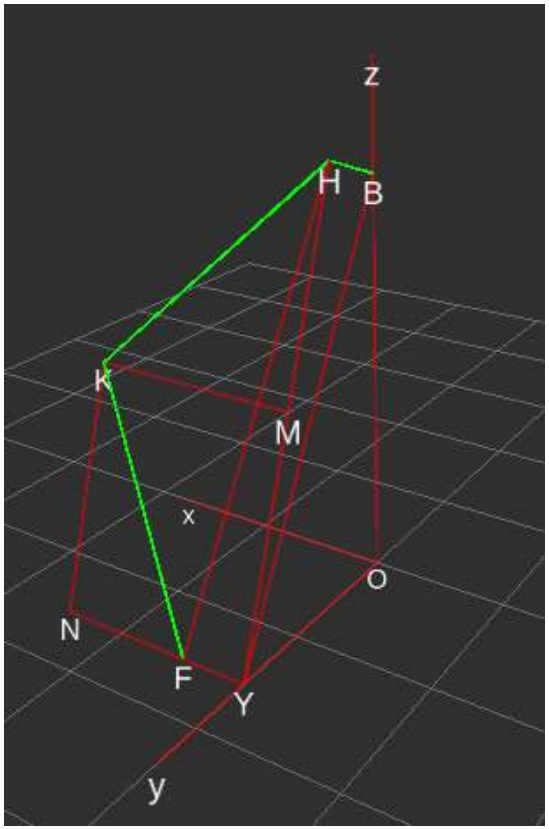
运动学

正向运动学
- 已知每个关节的角度，求出末端的坐标

反向运动学
- 已知末端的坐标，求出每个关节的角度

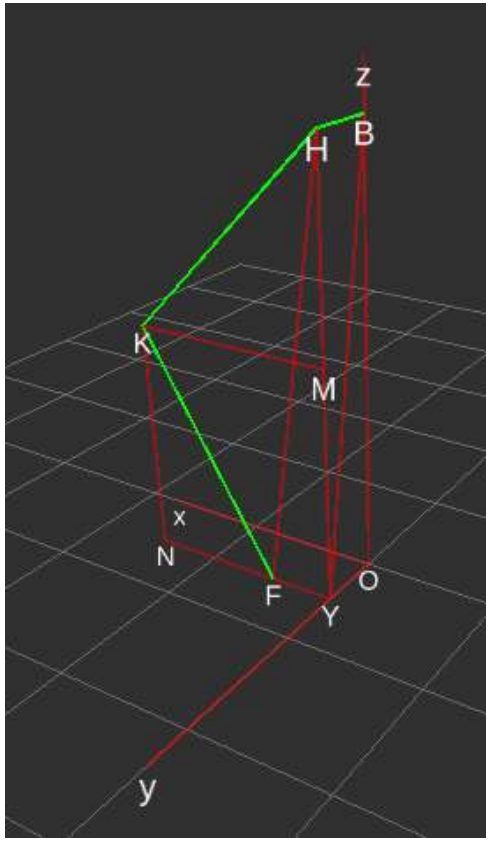


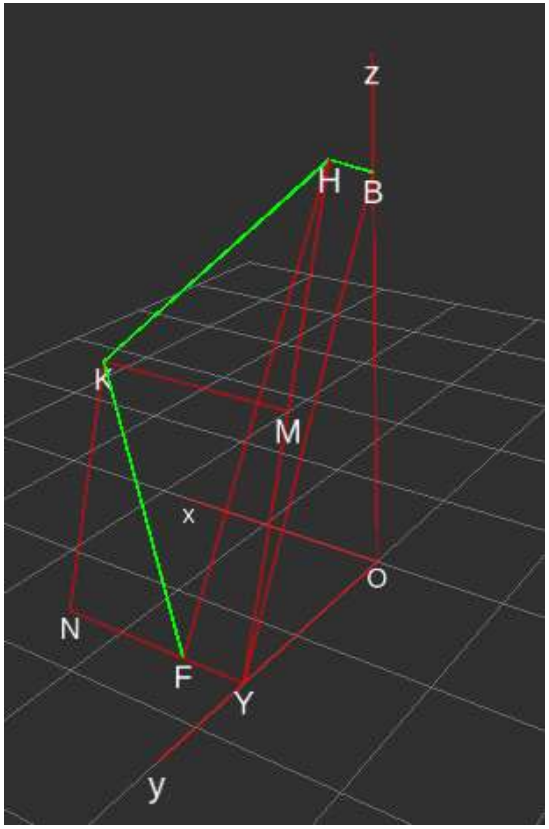
建模



B、H、K分别为舵狗的三个关节
BH⊥x轴，在yOz平面上旋转
HKF平面⊥BH，HK、KF在平面HKF内旋转
Y为F在y轴上的投影，M、N分别是K在HY、FY上的投影
主要考虑yOz平面和HKF平面

正向运动学



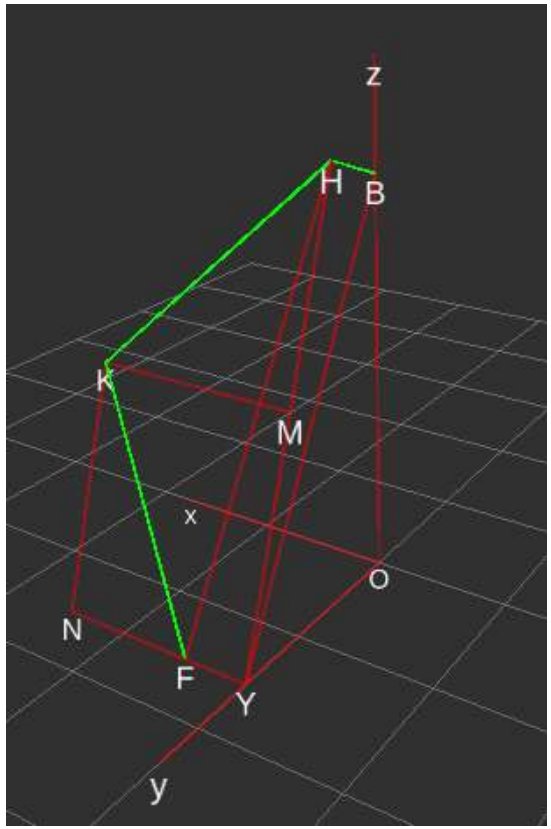


已知三个关节的角度为α，β，γ，求末端F点的坐标

可先假设α为90°，HKF平面平行于xOz面，根据三角函数关系易得F点的坐标

再由平面旋转算出BH旋转α-90°角后F点的坐标

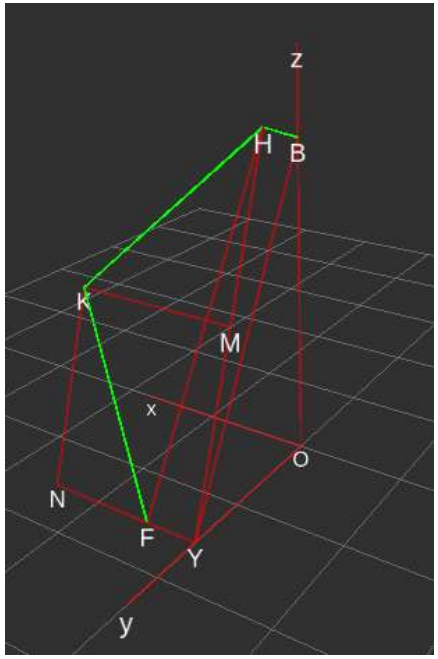
反向运动学

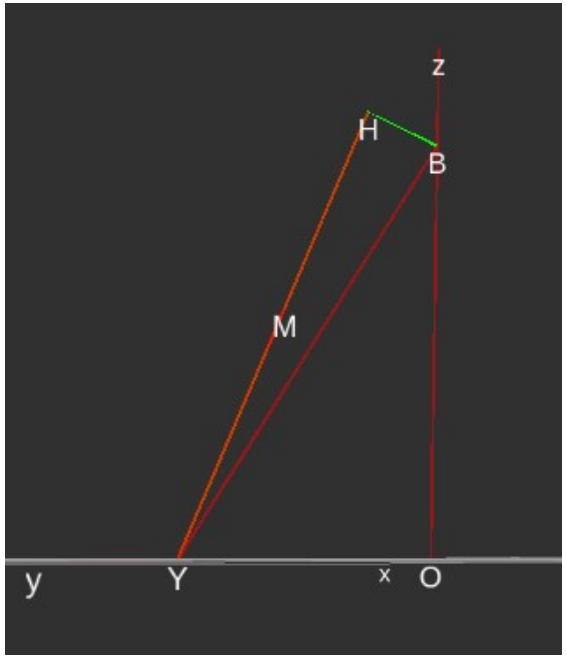


已知末端F点的坐标，BH、HK、KF为已知常量，求关节角度α，β，γ

主要考虑yOz平面和HKF平面

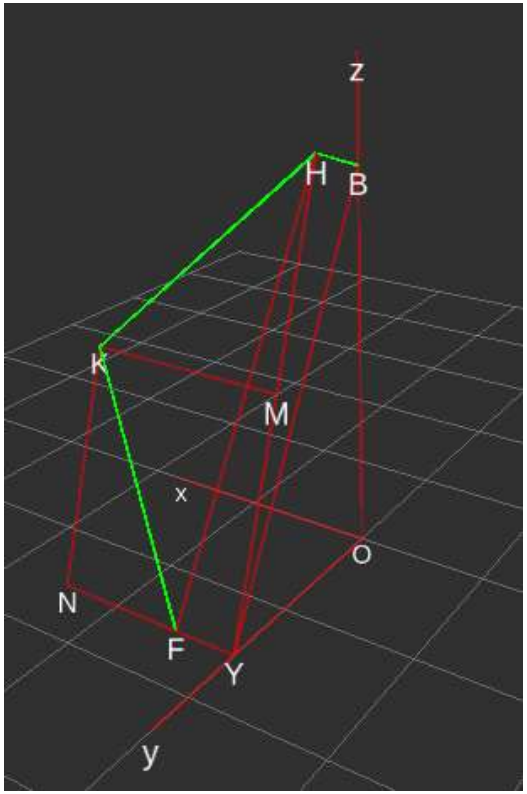
反向运动学

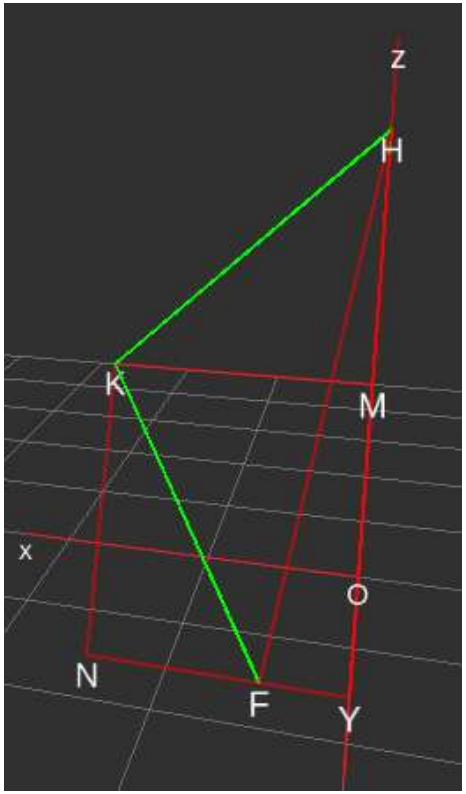




侧视图(yOz平面)
α = ∠HBO = ∠HBY+∠OBY
BH⊥平面HKF，由勾股定理可得HY长度，由三角函数可得∠HBY和∠OBY

反向运动学





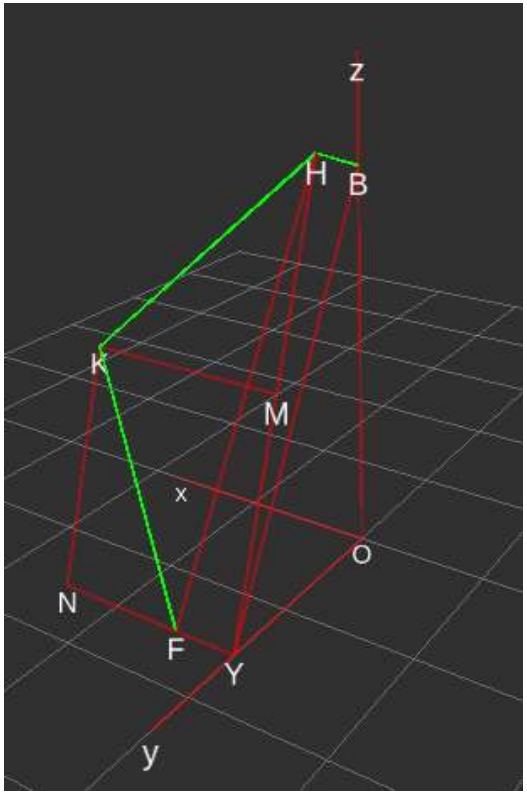
HKF平面，M、N分别是K在HY、YF上的投影

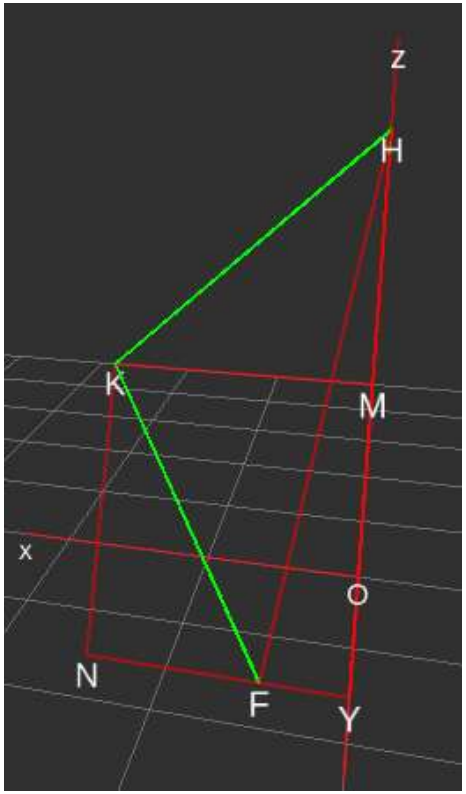
β = ∠YHK = ∠YHF+∠KHF

HY长度前面已求得，FY长度已知，可得HF长度与∠ YHF

已知三边长度由余弦定理可得∠KHF

反向运动学





HKF平面，M、N分别是K在HY、YF上的投影

γ = ∠NKF = ∠NKH - ∠HKF

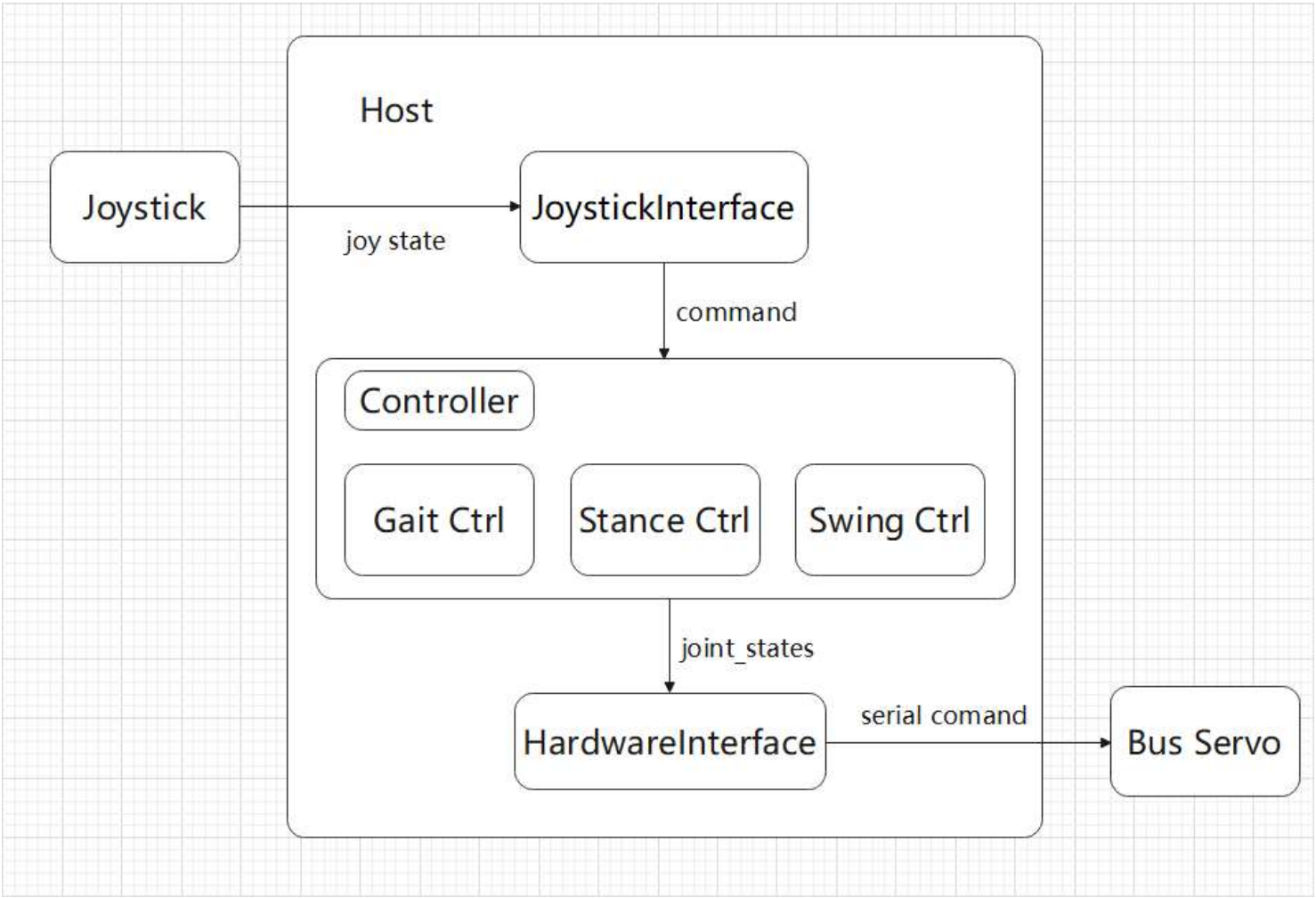
∠NKH=180° - ∠YHK(β)

已知三边长度由余弦定理可得∠HKF

03 代码说明

Code explanation

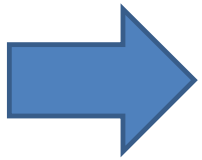
代码架构


Host
Joystick
joy state
JoystickInterface
command
Controller
Gait Ctrl
Stance Ctrl
Swing Ctrl
joint_states
HardwareInterface
serial comand
Bus Servo

JoystickInterface

/JoystickInterface.py

```
msg = {
    "ly": left_y,
    "lx": left_x,
    "rx": right_x,
    "ry": right_y,
    "L2": L2,
    "R2": R2,
    "R1": R1,
    "L1": L1,
    "dpady": dpady,
    "dpadx": dpadx,
    "x": x,
    "square": square,
    "circle": circle,
    "triangle": triangle,
    "message_rate": MESSAGE_RATE,
}
```



/src/Command.py

```
class Command:
    """Stores movement command
    """

    def __init__(self):
        self.horizontal_velocity = np.array([0, 0])
        self.yaw_rate = 0.0
        self.height = -0.16
        self.pitch = 0.0
        self.roll = 0.0
        self.activation = 0

        self.hop_event = False
        self.trot_event = False
        self.activate_event = False
```

以把手柄摇杆和按键数据转换成线速度和角速度

State

/src/State.py

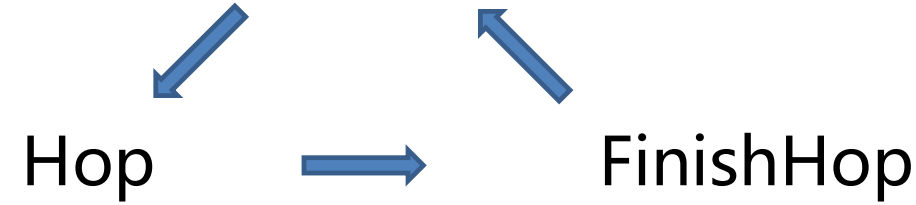
```
class BehaviorState(Enum):
    DEACTIVATED = -1
    REST = 0
    TROT = 1
    HOP = 2
    FINISHHOP = 3
```

状态机:

L1: Active <----> Deactive

L2: Rest <----> Trot

X: Rest

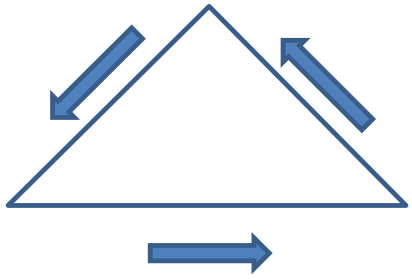


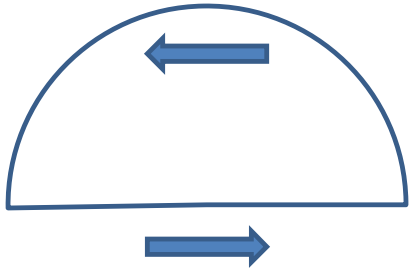
Controller

GaitController
- 确定每个控制周期每条腿的状态(抬腿or着地)

StanceController
- 控制着地时的运动
- z不变,xy按速度-horizontal_velocity变化

SwingController
- 控制抬腿时的运动
- Z先升高再降低,xy按速度horizontal_velocity变化





horizontal_velocity：左手摇杆横纵分别控制vy和vx
抬腿的时候向horizontal_velocity方向移动，着地后向相反方向移动

GaitController

```
################### GAIT ######################
self.dt = 0.01
self.num_phases = 4
self.contact_phases = np.array(
    [[1, 1, 1, 0],
     [1, 0, 1, 1],
     [1, 0, 1, 1],
     [1, 1, 1, 0]]
)

self.overlap_time = (
    0.10  # duration of the phase where all four feet are on the ground
)
self.swing_time = (
    0.15  # duration of the phase when only two feet are on the ground
)
```

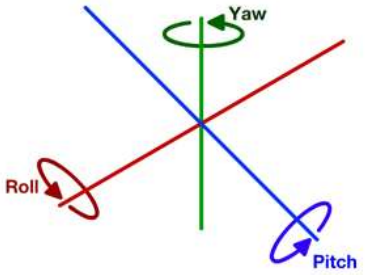
10毫秒为一个时间周期
控制循环分为四个阶段

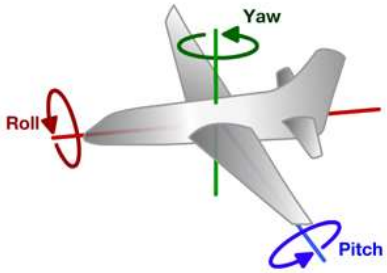
- 四条腿落地(后退)
- 2、3号腿抬起(前进)
- 四条腿落地(后退)
- 1、4号腿抬起(前进)

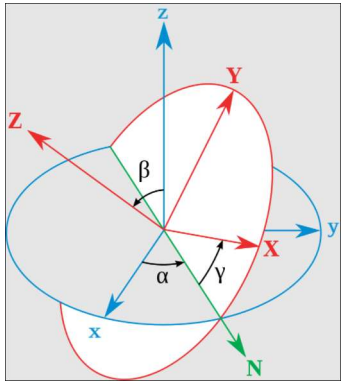
overlap：四条腿落地
swing：两条腿落地
对单条腿而言，每个循环只有一个阶段离地

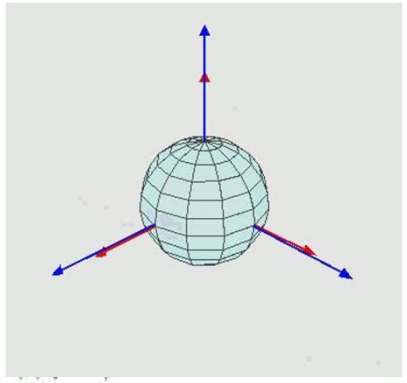
速度乘以时间算出位移得到每只脚的坐标，再根据反向运动学求出每只脚各个关节的角度

考虑旋转









α = yaw : 偏航角
β = roll :俯仰角
γ = pitch : 横滚角

$$\mathbf{R}_1 = Z(\gamma) \circ X(\beta) \circ Z(\alpha) \circ \mathbf{r}_1 \text{ 。}$$

$$Z(\alpha) = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$X(\beta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & \sin\beta \\ 0 & -\sin\beta & \cos\beta \end{bmatrix},$$

$$Z(\gamma) = \begin{bmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}。$$

欧拉角构造旋转矩阵就直接把三个旋转矩阵乘在一起

考虑旋转

```
# Apply the desired body rotation
rotated_foot_locations = (
    euler2mat(
        command.roll,
        command.pitch,
        self.smoothed_yaw,
    ).dot(state.foot_locations)
)
```

command.roll 左手横向十字
command.pitch 右手纵向摇杆
command.yaw 右手横向摇杆

根据欧拉角构造旋转矩阵，乘上原本的坐标，就得到旋转变换后的坐标

04 优化改进

Optimization and improvement

优化改进

- 轨迹函数优化，抬腿高度，前进速度
- 控制循环优化，着地时间、抬腿时间、抬腿周期
- 执行器优化，空心杯电机，无刷电机
- 机器人结构优化，减轻配重，增加强度
- 控制算法优化，考虑力的作用，实现三足、双足行走

THANKS FOR WATCHING

吴畔昊