楼梯步态说明

对于铁蛋 2 代，具体分为盲走楼梯步态和基于视觉动态选取落足点。前者通过在现有平地步态基础上，估计支撑平面，调整摆腿轨迹和机身姿态实现。后者在盲走步态基础上，避开台阶边缘防止打滑，提供更准确的落足点位置和坡度估计，这里主要介绍下盲走步态的相关要点。

主要算法模块：

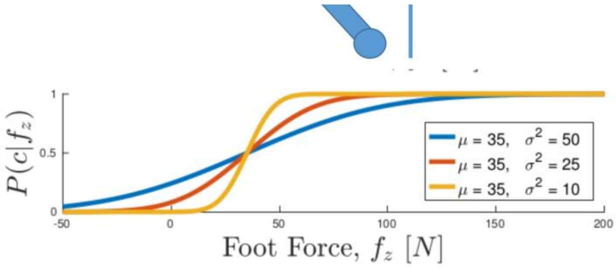
1、触地检测

腿部雅可比矩阵转置的逆乘以关节扭矩，计算足端的受力，根据垂直方向受力变化达到阈值(低速下)，判断是否触地，有视觉的情况可更多依赖视觉

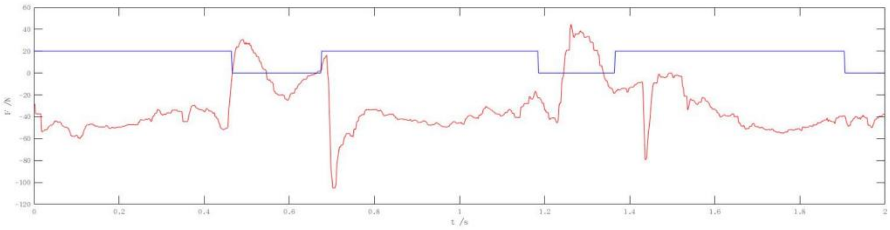
```
datas_[ leg ].jacobian = ComputeLegJacobian( quadruped_, leg, datas_[ leg ].q );
Mat3< T > inv_jacobian_transpose = datas_[ leg ].jacobian.transpose().inverse();
datas_[ leg ].foot_force_actual  = inv_jacobian_transpose *
datas_[ leg ].tau_actual;
```

或者通过触地力变化程度计算概率，一个技巧是将上楼梯和下楼梯步态参数分开设置

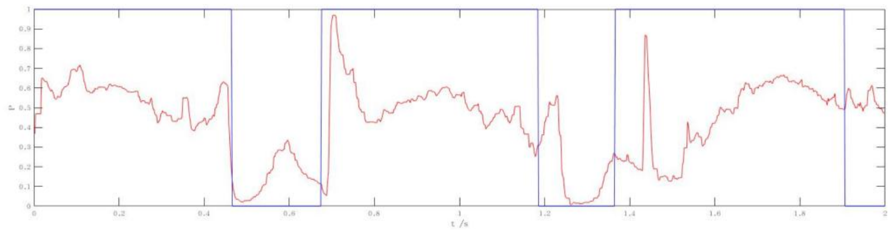
$$P(c|f_z) = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{f_z - \mu_{f_c}}{\sigma_{f_c}\sqrt{2}}\right)\right]$$



Z方向触地力



触地概率



```
JSON
void ConvexMpcStairGaits::CheckContact( int foot_num ) {
    int   i       = foot_num;
    float fc_foot = leg_ctrl_->datas_[ i ].foot_force_actual[ 2 ];
    float u_fc = -40, delta_fc = 35;
    float probability_fc = 0.5 + 0.5 * std::erff( ( -fc_foot +
u_fc ) / ( std::sqrt( 2 ) * delta_fc ) );
    float fc_foot_dt     = ( fc_foot -
actual_last_foot_force_[ i ] ) / 0.002;
    float u_fc_dt = -8000, delta_fc_dt = 1400;
    float probability_fc_dt      = 0.5 + 0.5 * std::erff( ( -
fc_foot_dt + u_fc_dt ) / ( std::sqrt( 2 ) * delta_fc_dt ) );
    actual_last_foot_force_[ i ] = leg_ctrl_-
>datas_[ i ].foot_force_actual[ 2 ];
    if ( swing_states_( i ) > ( gait_cmd_ ==
GaitId::kUpStairsSlowTrot || gait_cmd_ ==
GaitId::kDownStairsSlowTrot ? 0.6 : 1 )
         && ( gait_cmd_ == GaitId::kUpStairsSlowTrot ? 1 : 0 ) *
probability_fc_dt + probability_fc > 0.5 ) {
        contact_states_est_[ i ] = true;
    }
}
```

2、支撑平面估计

当机器人前进方向正对斜面或台阶（盲上时可简化成斜坡）时，期望俯仰角可根据侧视图中前后支撑足端两点连线同水平面夹角设置，可适当简化问题

完整地形估计可参考于宪元-《基于稳定性的仿生四足机器人控制系统设计》相关章

节，或铁蛋开源代码中 USE_TERRAIN_DETECTER 宏关联代码，同盲走楼梯步态相同

当机器人在斜坡上行走时，摆动相的落足点在z方向坐标不能简单规划为 0。否则，在坡下的腿就会表现出如同踩到坑里，在坡上的腿就会表现出如同踩到石头上（如图22所示）这对机身控制的干扰非常明显。因此，有必要估算机器人所在地面的坡度，从而合理规划机器人的落足点高度坐标。

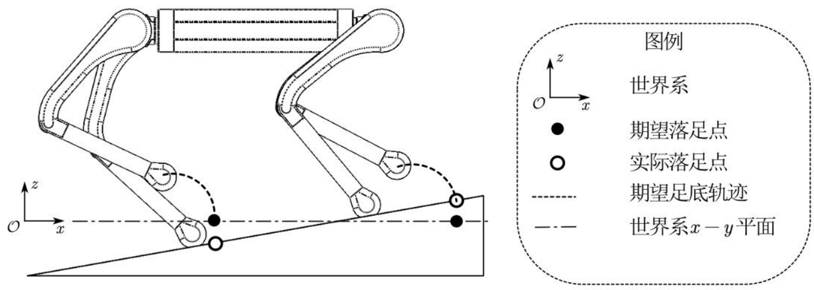


图22　机器人在斜坡上行走，期望落足点与实际落足点偏差较大

机器人不会在垂直于地面的墙面上行走，因此可以设地面的平面方程为：

$$z = a + bx + cy = [1 \quad x \quad y] A_{pla} \tag{2.66}$$

机器人每条腿都会交替成为支撑相和摆动相。记 ${}^{O}_{i}\boldsymbol{p}_{st\cdot end} = [x \quad y \quad z]^T$ 为i号腿最后一次成为支撑腿时的足底坐标。即：若i号腿是支撑腿，则 ${}^{O}_{i}\boldsymbol{p}_{st\cdot end}$ 为当前足底坐标；若i号腿是摆动腿，则 ${}^{O}_{i}\boldsymbol{p}_{st\cdot end}$ 为最后一次支撑相切换为摆动相瞬间的足底坐标。${}^{O}_{i}\boldsymbol{p}_{st\cdot end}$ 将由以下递推方法获得：

$$\begin{cases} {}^{O}_{i}\boldsymbol{p}^{0}_{st\cdot end} = {}^{O}_{i}\boldsymbol{p}^{t=0} \\ {}^{O}_{i}\boldsymbol{p}^{k}_{st\cdot end} = {}^{O}_{i}\boldsymbol{p} \quad , \quad {}_{i}s_{\Phi} = 1 \\ {}^{O}_{i}\boldsymbol{p}^{k}_{st\cdot end} = {}^{O}_{i}\boldsymbol{p}^{k-1}_{st\cdot end}, \quad {}_{i}s_{\Phi} = 0 \end{cases} \tag{2.67}$$

将 ${}^{O}_{i}\boldsymbol{p}_{st\cdot end}$ 的z坐标与xy坐标分离，并分别填充到两个矩阵中：

$${}^{O}\boldsymbol{z}_f = [\ {}^{O}_{1}\boldsymbol{p}_{st\cdot end}(3) \quad {}^{O}_{2}\boldsymbol{p}_{st\cdot end}(3) \quad {}^{O}_{3}\boldsymbol{p}_{st\cdot end}(3) \quad {}^{O}_{4}\boldsymbol{p}_{st\cdot end}(3)]^T \tag{2.68}$$

$$W_{pla} = \begin{bmatrix} 1 & {}^{O}_{1}\boldsymbol{p}_{st\cdot end}(1) & {}^{O}_{1}\boldsymbol{p}_{st\cdot end}(2) \\ 1 & {}^{O}_{2}\boldsymbol{p}_{st\cdot end}(1) & {}^{O}_{2}\boldsymbol{p}_{st\cdot end}(2) \\ 1 & {}^{O}_{3}\boldsymbol{p}_{st\cdot end}(1) & {}^{O}_{3}\boldsymbol{p}_{st\cdot end}(2) \\ 1 & {}^{O}_{4}\boldsymbol{p}_{st\cdot end}(1) & {}^{O}_{4}\boldsymbol{p}_{st\cdot end}(2) \end{bmatrix} \tag{2.69}$$

利用最小二乘法计算地面的平面方程：

$$A_{pla} = W^{+}_{pla} \cdot {}^{O}\boldsymbol{z}_f \tag{2.70}$$

其中W^{+}_{pla}表示W_{pla}的伪逆矩阵，为A_{pla}设计一个低通滤波器：

$$\hat{A}^{k}_{pla} = [\hat{a} \quad \hat{b} \quad \hat{c}]^T = \eta A_{pla} + (1-\eta)\hat{A}^{k-1}_{pla} \tag{2.71}$$

其中η为低通滤波系数，本文中$\eta = 0.2$。根据平面方程的性质，地面的法向量为

$\boldsymbol{n}=[-\hat{b}\ \ -\hat{c}\ \ 1]^T$，将其单位化：

$$\boldsymbol{n}_e=\frac{\boldsymbol{n}}{|\boldsymbol{n}|} \tag{2.72}$$

根据公式(2.9)，$\boldsymbol{n}_e$是机体期望旋转矩阵 ${}^{O}_{d}R_B$ 的第三列，结合公式(2.20)：

$$\boldsymbol{n}_e=\begin{bmatrix} s_\phi s_\psi+c_\phi c_\psi s_\theta \\ c_\phi s_\theta s_\psi-c_\psi s_\phi \\ c_\phi c_\theta \end{bmatrix}=\underbrace{\begin{bmatrix} c_\psi & s_\psi & 0 \\ s_\psi & -c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\Upsilon}\underbrace{\begin{bmatrix} c_\phi s_\theta \\ s_\phi \\ c_\phi c_\theta \end{bmatrix}}_{\xi}=\Upsilon\xi \tag{2.73}$$

在上一小节中，期望偏航角${}_d\psi^1$已经求出，因此矩阵Υ已知，可解方程 (2.73)：

$$\xi=\Upsilon^{+}\boldsymbol{n}_e \tag{2.74}$$

根据公式(2.73)中ξ的定义，可得机器人期望俯仰角与期望横滚角：

$$\begin{cases} {}_d\phi^1=\arcsin(\xi(2)) \\ {}_d\theta^1=\arctan(\xi(1)/\xi(3)) \end{cases} \tag{2.75}$$

3、摆动轨迹和落足点调整

根据触地判断和地形估计，调整摆动腿的摆动曲线，调整落足点，这里需要注意世界坐标系和机身坐标系之间的变换

```
foot_swing_trajectory_[ i ].SetHeight( step_height_des_ );
foot_swing_trajectory_[ i ].SetDepth( user_params_->downstairs_depth );
foot_swing_trajectory_[ i ].SetFinalPosition( landing_pos_[ i ] );
```

4、前后腿换向对齐

假设足端只会在摆动提前触地，不会在支撑相位打滑，因提前触地导致腾空项提前结束时，可保持位置不变直到下一个支撑相，从而进行前后腿相位切换对齐

5、其他

提前触地时，将 WBC pd 减小，减小震荡，将上楼梯的支撑相占比调大，上下楼梯步态分开设置参数，对于盲走步态，里程计高度估计可选择相对高度而非世界系下的绝对进行简化