基于 Cassie 的倒立摆步态和路径规划

夏逸轩 周剑翔 杨博宇 魏锦启 杨昕 张熙芃

摘要：Cassie 机器人是一种基于鸟类腿部关节设计的双足机器人，本文以 Cassie 机器人开源 URDF 模型为基础，首先研究了机器人的结构，并将三维线性倒立摆模型应用于机器人的步态规划，并在 pybullet 仿真环境中实现了 Cassie 基于三维线性倒立摆的直立行走、爬楼梯以及转弯。由于在有限的时间内无法完成步行稳定控制器的设计，所以小组基于 Caltech 的 Gazebo 仿真模型进行了 Cassie 路径规划的算法，并简要介绍了 Caltech 所做的基于 Full order model 的步态设计和稳定控制器。之后，本小组比较了移动机器人的全局路径规划算法以及局部路径规划算法，在利用视觉定位障碍物之后，使用人工势能场法算法自主规划路径，并在 Gazebo 平台下实现了避障算法的应用以及机器人绕桩的演示。

关键词： 双足机器人；倒立摆；人工势能场路径规划；步态设计；Cassie

3D-LIP Gait Planning and Path Planning Based on Cassie Robot

Xia yixuan Zhou jianxiang Yang boyu Wei jinqi Yang xin Zhang xipeng

Abstract： Cassie is a robot which was designed based on the joints that birds usually have. Based on Cassie's open-source robot urdf model on GitHub, this paper first introduces the structure of the robot, and applies the 3d linear inverted pendulum model to the gait planning of the robot, and uses pybullet to demonstrate the robot's ability to walk upright, climb stairs and turn around. In short od the time our group are not able to complete the controller that enable the robot to stand on the ground. So we complete our path planning on the open-source model developed by Caltech. Our group compared the global path planning algorithm and the local path planning algorithm of mobile robot. After locating the obstacles by computer vision, the artificial potential field method is used to plan the path independently, and the application of obstacle avoidance algorithm and robot around the pile are realized on Gazebo platform (Ubuntu).

Key words：Biped robot；Linear Inverted Pendulum Model；Path planning；Gait design；Cassie

1 研究动机

Cassie 是一种仿鸵鸟的双足机器人，由俄勒冈州立大学孵化出来的公司 Agility Robotics 发明并制造，曾获得美国国防部高级研究计划局的 100 万美元拨款。全身具有 20 个自由度，单一腿部有 7 个关节。几乎全铰接的腿部结构能让腿部实现多用途的自由运动。并且当遭受冲击时，膝盖和后跟的板簧能起到吸收冲击的作用，从而体现出机器人的敏捷性。以 Cassie 为主体，研究者们提出了各式各样的控制策略。

本文以加州理工学院所提出的开源控制模型为基础，简化模型，将闭环连机构切开，在组成连杆机构的关节之间添加限制与角度关系，展示了在简化模型下的运动规划以及基于视觉识别的障碍物定位和轨迹规划。本文第二节描述 Cassie 机器人的基本结构、正逆运动学的求解，为后续的仿真提供理论基础；第三节，将从二维线性倒立摆模型开始，介绍三维倒立摆步态规划方法；第四节介绍加州理工学院提出的控制器；在前几节的基础上，第五节介绍路径规划的基本方法，最终利用计算机视觉进行障碍物定位，然后利用人工势场法实现对机器人的路径规划，并完成了绕桩演示；最后将对本文现阶段存在的问题与不足做总结展望。

2 模型建立

2.1 基本模型

Cassie 机器人由 Agility Robotics 公司制作。这个机器人的形态灵感来自食火鸡，一种新几内亚本土的不会飞的鸟；没有生物学背景的人可以把它比作鸵鸟。Cassie 机器人在站立状态下约有 1m 高，总质量为 31 千克，其中位于躯干中的电池约占 4 千克。如下图 1 所示，机器人的浮动基模型具有 20 个自由度。每条腿上有七个关节，其中五

个由电机驱动，另外两个是通过四连杆机构实现的被动关节。每条腿上的三个驱动关节对应于设置腿相对于躯干的滚动、偏转和俯仰角，一个电机设置“膝盖”的角度，最后一个电机调节脚的俯仰角。

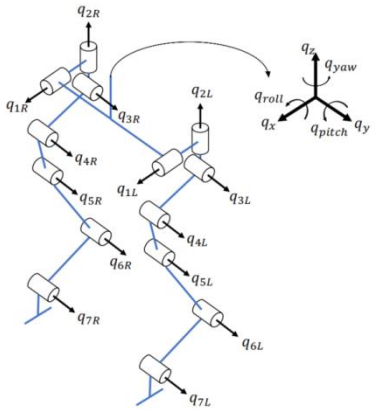


图 1 Cassie 运动学模型

其中，变量 (qx ， qy ， qz) 是躯干的笛卡尔坐标，qyaw，qpitch，qroll 是以 z-y-x 为顺序的欧拉角。关节 q1、q2、q3 分别是臀部的滚动、偏转和俯仰角，q4 是膝盖俯仰角，q5 是小腿俯仰角，q6 是跗骨俯仰角，q7 是脚的俯仰角。

为了方便处理我们将这些关节转换为

关节（q1,q2,q3,q4,q7)是驱动关节。（q5，q6）是欠驱动关节，由引入的的弹簧的构成。当弹簧不压缩时，q5,q6 满足以下关系。

$$q_5 = 0, \text{ and } q_6 = -q_4 + 13^{\circ}.$$

为了方便后期方便建模我们对模型进行简化。

我们假设连杆是一个作用于臀部和脚跟弹簧连接器之间的虚拟的完整距离约束：

$$\eta_{ach}(q_l) := d(q_l) - 0.5012 = 0,$$

我们还假设，当一条腿在摆动时，该腿上的弹簧是刚性固定的：

$$\eta_{sw}(q)^{\mathsf{T}} := [q_s, q_{hs}]^{\mathsf{T}} = 0.$$

2.2 运动学模型

对于 Cassie 机器人来说，关于 h0 控制量的选择，驱动关节与行走模式间有着限制性的物理意义，因此 Caltech 采用了调节躯干的姿态角，触地脚和游动脚的长度，游动脚的姿态角。

$$h_0(q) = \begin{bmatrix} q_{\text{roll}} \\ q_{2\ \text{st}} \\ q_{\text{pitch}} \\ q_{\text{LL st}} \\ q_{\text{LR sw}} \\ q_{2\ \text{sw}} \\ q_{\text{LP sw}} \\ q_{\text{LL sw}} \\ q_{\text{FP sw}} \end{bmatrix} \begin{pmatrix} \text{torso roll} \\ \text{stance hip yaw} \\ \text{torso pitch} \\ \text{stance leg length} \\ \text{swing leg roll} \\ \text{swing hip yaw} \\ \text{swing leg pitch} \\ \text{swing leg length} \\ \text{swing foot pitch} \end{pmatrix}$$

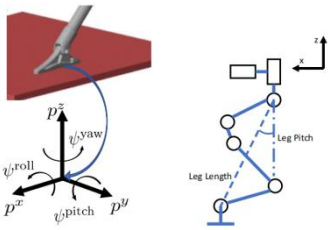


图 2 Cassie 机构控制量

当右腿站立，左腿摆动时，根据机器人的配置，计算了腿部长和腿部螺距的正向运动学：

$$\begin{aligned} q_{\text{LL st}} &= \sqrt{0.5292\cos(q_{4\text{R}} + 0.035) + 0.5301} \\ q_{\text{LR sw}} &= q_{\text{roll}} + q_{1\text{L}} \\ q_{\text{LP sw}} &= -q_{\text{pitch}} + q_{3\text{L}} \\ &- \arccos\left(\frac{0.5(\cos(q_{4\text{L}} + 0.035) + 0.5292)}{\sqrt{0.5292\cos(q_{4\text{L}} + 0.035) + 0.5301}}\right) + 0.1 \\ q_{\text{LL sw}} &= \sqrt{0.5292\cos(q_{4\text{L}} + 0.035) + 0.5301} \\ q_{\text{FP sw}} &= -q_{\text{pitch}} + q_{7\text{L}} + 1.1. \end{aligned}$$

3 基于 3D-LIP 的步态规划

3.1 二维线性倒立摆模型分析

在将三维线性倒立摆模型应用于 Cassie 机器人的步态规划前，需要首先考虑理想模型下双足动态步行规划的基本原理——二维线性倒立摆模型。线性倒立摆模型首先对机器人模型近似化，并做出三个假设。第一、假定机器人的所有质量集中于其质心位置；第二、将机器人的腿简化成一个可伸缩的轻杆，轻杆与地面接触通过一个可转动的支点完成；第三、将机器人的运动约束在铅垂轴与步行方向的纵向轴决定的二维平面内，即约束在纵平面内。在这些约束条件下，Cassie 机器人可简化为一个二维线性倒立摆模型，如图 3 所示。

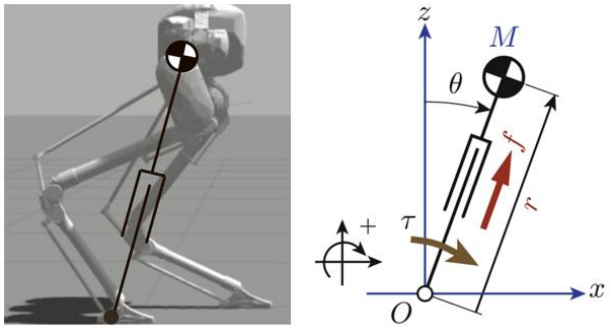


图 3 二维倒立摆：双足步行机器人的最简单模型，由一个点质量和一个可伸缩的无质量的腿构成。其偏转角 θ 以铅垂轴为基准，顺时针方向为正

倒立摆的输入包括作用域支点处的力矩 τ 和沿腿连杆方向伸缩关节上的伸缩力 f。该倒立摆的动

力学可用通过拉格朗日法求得的下列两个微分方程描述。

$$r^2\ddot{\theta} + 2r\dot{r}\dot{\theta} - gr\sin\theta = \tau/M$$
$$\ddot{r} - r\dot{\theta}^2 + g\cos\theta = f/M.$$

假设行走中地面对足部产生的力矩忽略不计，即，τ=0。同时腿的伸缩力 f 可分解为铅垂方向上的重力补偿力和水平方向上的加速度力。直观上可以说倒立摆在摔倒的过程中通过伸展其腿长而保持恒定的质心高度。我们称这种摆为线性倒立摆(linear inverted pendulum)。

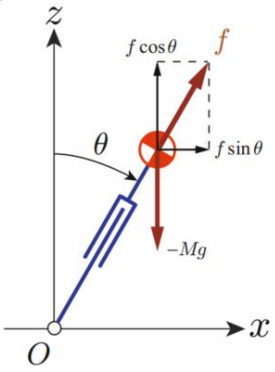


图 4　质心水平运动的解释：重力为伸缩力 f= Mg/cos θ 的铅垂分力所平衡。

考察图 4 ,腿伸缩力的铅垂分力平衡重力之后，水平分力还存在。这一分力使质心沿水平方向加速运动，相应的运动方程为

$$M\ddot{x} = f\sin\theta = Mg\tan\theta = M\frac{g}{z}x$$

其中，x 和 z 为倒立摆质心的位置坐标。整理上面的方程可得到质心水平运动的微分方程为：

$$\ddot{x} = \frac{g}{z}x.$$

当 z 恒定时，从上面的普通微分方程很容易求出水平方向的运动学：

$$x(t) = x(0)\cosh(t/T_c) + T_c\dot{x}(0)\sinh(t/T_c)$$
$$\dot{x}(t) = x(0)/T_c\sinh(t/T_c) + \dot{x}(0)\cosh(t/T_c)$$
$$T_c \equiv \sqrt{z/g}$$

其中，Tc 为一个取决于质心高度和重力加速度的常数。x(0)与 x’(0)分别为零时刻水平方向质心的初始位置和初始速度，即初始条件。

3.2　三维线性倒立摆模型分析

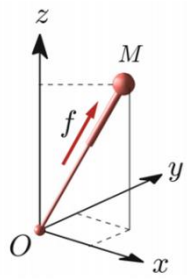


图 5　三维线性倒立摆模型

我们将三维空间的机器人近似为一个由集中了所有质量的点和连接该质点与支撑点的无质量的腿组成的倒立摆，如图 5 所示。支撑点处的力矩为零，因而可以自由转动。在伸缩力 f 的作用下腿可以伸缩，相当于机器人膝关节的弯曲或伸展而产生质心的上下运动。伸缩力 f 可以分解为 x，y 和 z 三个方向上的分量：

$$f_x = (x/r)f$$
$$f_y = (y/r)f$$
$$f_z = (z/r)f$$

其中，r 为支撑点和质心之间的距离，质心只受伸缩力和重力的作用，故其运动方程为：

$$M\ddot{x} = (x/r)f$$
$$M\ddot{y} = (y/r)f$$
$$M\ddot{z} = (z/r)f - Mg$$

类似于二维倒立摆的情况，考虑质心约束，对三维倒立摆定义约束面如下：

$$z = k_x x + k_y y + z_c$$

其中k_x, k_y决定平面的倾斜，z_c决定其高度。

为了让质心在约束面上运动，其加速度应该与约束面的法矢量垂直，即：

$$\begin{bmatrix} f\left(\frac{x}{r}\right) & f\left(\frac{y}{r}\right) & f\left(\frac{z}{r}\right) - Mg \end{bmatrix} \begin{bmatrix} -k_x \\ -k_y \\ 1 \end{bmatrix} = 0$$

根据上式解出f，得：

$$f = \frac{Mgr}{z_c}$$

通过施加与腿长r成正比的伸缩力f就可以把质心控制在约束面上运动，如下图 6 所示。

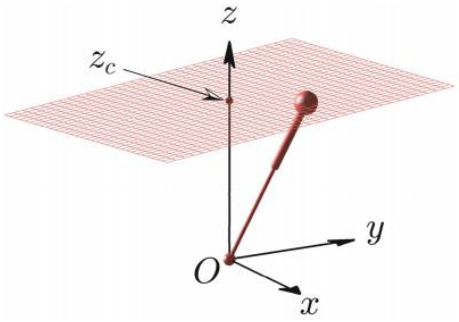


图 6 三维线性倒立摆。通过控制伸缩力可使质心在约束面上运动，约束面的斜度不影响质心的水平运动

同时可以求解获得质心在水平方向的运动方程为：

$$\ddot{x} = \frac{g}{z_c} x$$

$$\ddot{y} = \frac{g}{z_c} y$$

可以看出，约束面的参数k_x, k_y不影响质心的水平运动，这样的倒立摆称为三维线性倒立摆（3D-LIP）。其水平方向的运动方程与二维线性倒立摆模型相同。

3.3 三维步行模式的生成

步行单元

对于三维步行，需要在x轴和y轴方向上同时切换支撑脚，在后面的讨论中，设支撑脚的切换周期固定，每步的支撑时间为T_{sup}。

在时间段$[0 \quad T_{sup}]$，三维线性倒立摆轨迹如图7，是关于y轴对称的双曲线，我们称之为步行单元（Walking Primitive）。

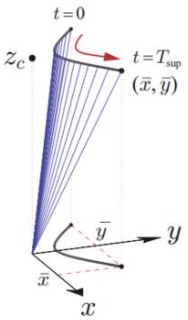


图 7 步行单元：三维步行的基本模式

在支撑时间T_{sup}和约束面的z轴截距给定后，由其对称性可知，步行单元由其终止位置$(\bar{x}, \bar{y})$唯一确定，终止速度$(\bar{v}_x, \bar{v}_y)$可计算如下。

设沿x轴的初始条件为$(-\bar{x}, \bar{v}_x)$，终止位置为$\bar{x}$，根据线性倒立摆的解析式，有：

$$\bar{x} = -\bar{x}C + T_c \bar{v}_x S$$

其中，

$$T_c = \sqrt{\frac{z_c}{g}}, C \equiv \cosh\frac{T_{sup}}{T}, S \equiv \sinh\frac{T_{sup}}{T_c}$$

同时由位置解析式可以求出终止速度为，

$$\bar{v}_x = \bar{x}(C+1)/(T_c S)$$

类似地，对步行单元的 y 轴分量，终止位置和速度为，

$$\bar{y} = \bar{y}C + T_c(-\bar{v}_y)S$$

$$\bar{v}_y = \bar{y}(C-1)/(T_c S)$$

步行参数

在很多诸如上下楼梯或避障等实际情况下，需要直接指定脚的着地位置，常用步长和步宽来定义描述，如下表。

n	1	2	3	4	5
$s_x^{(n)}$	0	0.3	0.3	0.3	0
$s_y^{(n)}$	0.2	0.2	0.2	0.2	0.2

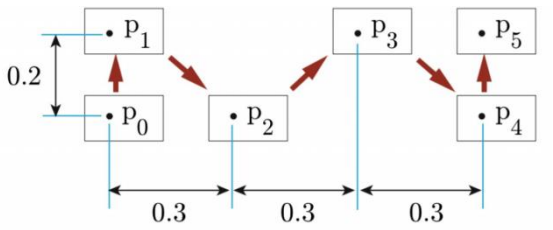


图 8 步行参数

其中$s_x^{(n)}$是前进方向的步长，$s_y^{(n)}$是左右方向的步宽，上标(n)表示第n步，表中的数据称为步行参数，他们描述图 8 所示的脚步情况。第n个落脚位置$(p_x^{(n)}, p_y^{(n)})$可表示如下。

$$\begin{bmatrix} p_x^{(n)} \\ p_y^{(n)} \end{bmatrix} = \begin{bmatrix} p_x^{(n-1)} + s_x^{(n)} \\ p_y^{(n-1)} - (-1)^n s_y^{(n)} \end{bmatrix}$$

此处是假设开始的支撑脚为右脚，如果是左脚需要用 $+(-1)^n$ 替代 $-(-1)^n$。第n步的步行参数确定如下：

$$\begin{bmatrix} \bar{x}^{(n)} \\ \bar{y}^{(n)} \end{bmatrix} = \begin{bmatrix} s_x^{(n+1)}/2 \\ (-1)^n s_y^{(n+1)}/2 \end{bmatrix}$$

第n步的步行参数是由第$n+1$步的步长和步宽决定的，这对于脚部位置和步行运动的适当协调是必要的。

步行单元的终止速度为，

$$\begin{bmatrix} \bar{v}_x^{(n)} \\ \bar{v}_y^{(n)} \end{bmatrix} = \begin{bmatrix} (C+1)/(T_c S)\bar{x}^{(n)} \\ (C-1)/(T_c S)\bar{y}^{(n)} \end{bmatrix}$$

落脚点的调整

在步行周期和落脚时间固定的情况下，可以通

过调整落脚点来控制步行速度。定性地说,下一步在较近处落脚时速度加大，而下一步在较远处落脚时速度减小。

记调整后的落脚位置为 p*,我们来定量地分析它对步行状态的影响。参见图 9,相对于地面上的固定坐标系,线性倒立摆的运动方程为：

$$\ddot{x} = \frac{g}{z_c}(x - p_x^*).$$

其解析解为：

$$x(t) = (x_i^{(n)} - p_x^*)\cosh(t/T_c) + T_c\dot{x}_i^{(n)}\sinh(t/T_c) + p_x^*$$

$$\dot{x}(t) = \frac{x_i^{(n)} - p_x^*}{T_c}\sinh(t/T_c) + \dot{x}_i^{(n)}\cosh(t/T_c),$$

因此，落脚点 p*与第 n 步的最终状态之间的关系如下：

$$\begin{bmatrix} x_f^{(n)} \\ \dot{x}_f^{(n)} \end{bmatrix} = \begin{bmatrix} C & T_cS \\ S/T_c & C \end{bmatrix} \begin{bmatrix} x_i^{(n)} \\ \dot{x}_i^{(n)} \end{bmatrix} + \begin{bmatrix} 1-C \\ -S/T_c \end{bmatrix} p_x^*.$$

至于最终状态的目标值,可用地面参考坐标系中最终步行单元的终止状态表示:

$$\begin{bmatrix} x^d \\ \dot{x}^d \end{bmatrix} = \begin{bmatrix} p_x^{(n)} + \bar{x}^{(n)} \\ \bar{v}_x^{(n)} \end{bmatrix}.$$

为了计算落脚点导致的最终状态与目标状态之间的接近程度,定义误差评估函数如下：

$$N \equiv a(x^d - x_f^{(n)})^2 + b(\dot{x}^d - \dot{x}_f^{(n)})^2$$

其中 a，b 为取值大于 0 的加权因子。根据条件$\partial N/\partial p_x^* = 0$，可求得使误差评估函数值 N 最小的落脚点：

$$p_x^* = -\frac{a(C-1)}{D}(x^d - Cx_i^{(n)} - T_cS\dot{x}_i^{(n)}) - \frac{bS}{T_cD}(\dot{x}^d - \frac{S}{T_c}x_i^{(n)} - C\dot{x}_i^{(n)})$$

$$D \equiv a(C-1)^2 + b(S/T_c)^2.$$

上述步行模式的生成方法可以归纳为如下所示的算法：

第一步	设定步行周期T_{sup}和步行参数(s_x, s_y)，质心的初始位置(x, y)以及初始落脚点$(p_x{}^*, p_x{}^*) = (p_x^{(0)}, p_y^{(0)})$；
第二步	$T:=0, n:=0$；
第三步	从时刻T到$T+T_{sup}$，对线性倒立摆方程式积分；
第四步	$T:=T+T_{sup}, n:=n+1$；
第五步	计算下一个落脚点$(p_x^{(n)}, p_x^{(n)})$；
第六步	计算下一个步行单元$(\bar{x}^{(n)}, \bar{y}^{(n)})$；
第七步	计算目标状态$(x^d, \dot{x}^d)$和$(y^d, \dot{y}^d)$
第八步	调整落脚点$(p_x{}^*, p_y{}^*)$的位置
第九步	返回第三步

表 1 基于三维线性倒立摆的步行模式生成算法

步行方向的改变

如图 9 所示，在步行中增添方向的信息，就可以改变行走方向。

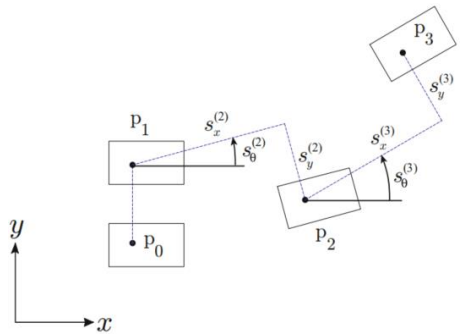


图 9 包含转向信息的落脚点设定方法。方向由相对于 x 轴逆时针方向为正的转角s_θ定义

第 n 步的落脚点由下式确定：

$$\begin{bmatrix} p_x^{(n)} \\ p_y{}^{(n)} \end{bmatrix} = \begin{bmatrix} p_x^{(n-1)} \\ p_y{}^{(n-1)} \end{bmatrix} + \begin{bmatrix} \cos s_\theta{}^{(n)} & -\sin s_\theta{}^{(n)} \\ \sin s_\theta{}^{(n)} & \cos s_\theta{}^{(n)} \end{bmatrix} \begin{bmatrix} s_x{}^{(n)} \\ -(-1)^n s_y{}^{(n)} \end{bmatrix}$$

第 n 步的步行单元更新如下：

$$\begin{bmatrix} \bar{x}^{(n)} \\ \bar{y}^{(n)} \end{bmatrix} = \begin{bmatrix} \cos s_\theta^{(n+1)} & -\sin s_\theta^{(n+1)} \\ \sin s_\theta^{(n+1)} & \cos s_\theta^{(n+1)} \end{bmatrix} \begin{bmatrix} s_x^{(n+1)}/2 \\ (-1)^n s_y^{(n+1)}/2 \end{bmatrix}$$

步行单元的速度式子更新如下：

$$\begin{bmatrix} \bar{v}_x^{(n)} \\ \bar{v}_y^{(n)} \end{bmatrix} = \begin{bmatrix} \cos s_\theta^{(n+1)} & -\sin s_\theta^{(n+1)} \\ \sin s_\theta^{(n+1)} & \cos s_\theta^{(n+1)} \end{bmatrix} \begin{bmatrix} (1+C)/(T_cS)\bar{x}^{(n)} \\ (C-1)/(T_cS)\bar{y}^{(n)} \end{bmatrix}$$

从线性倒立摆模型到多连杆模型

基于线性倒立摆模型作实际机器人的运动规划的最简单方法是使其腰部跟踪倒立摆的质心运动。首先机器人起始状态的质心在腰部坐标系中的实际位置根据多连杆模型求得。然后假定质心在腰部的相对位置不变，根据倒立摆直接求得腰部的位置。最后计算游脚的轨迹，使之在规定的着地时间到达期望的着地点。

一旦腰部和双足的运动确定后,根据拟运动学就可以求得腿的各关节角度。

3.4 基于 3D-LIP 步态规划的仿真演示

根据以上理论分析，我们完成了基于 3D-LIP 的直线行走、转弯和爬楼梯三个任务的演示，首先我们使用 python 的 matplotlib 库绘制三维空间中的步态动画演示和投影到 xy 平面方向上的质心轨迹图，之后在 pybullet 仿真环境中将质心轨迹和落脚点轨迹通过逆运动学求解关节角度，输入到 cassie 模型中进行位置控制。

直线行走：

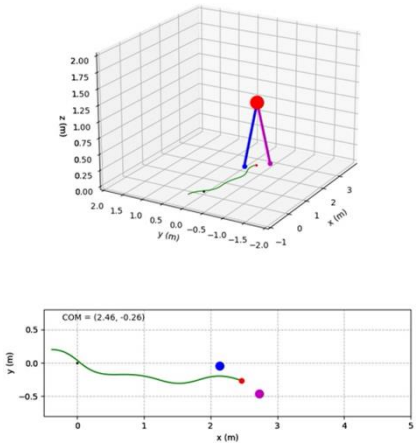


图 10 上图为在 matplotlib 中的三维空间中的直走步态演示，下图为投影到 xy 平面上的质心轨迹和足端轨迹。

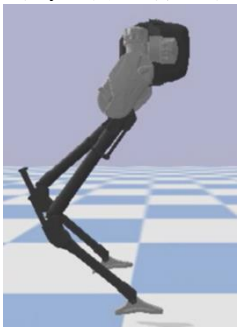


图 11 上图为在 pybullet 中的 cassie 直走步态演示。

转弯行走：

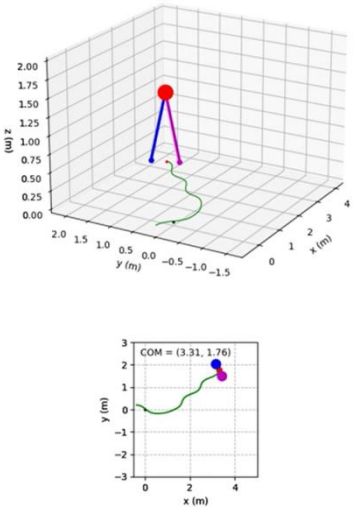


图 12 上图为在 matplotlib 中的三维空间中的转弯步态演示，下图为投影到 xy 平面上的质心轨迹和足端轨迹。

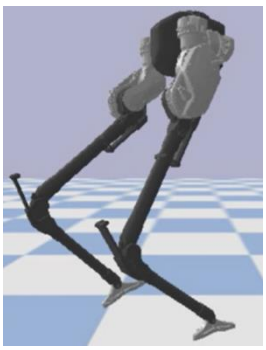
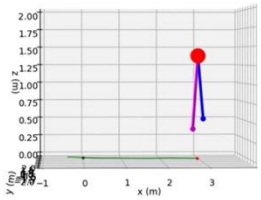


图 13 上图为在 pybullet 中的 cassie 转弯步态演示。

爬楼梯：



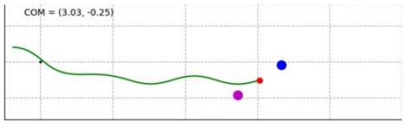


图 14 上图为在 matplotlib 中的三维空间中的爬楼梯步态演示，下图为投影到 xy 平面上的质心轨迹和足端轨迹。

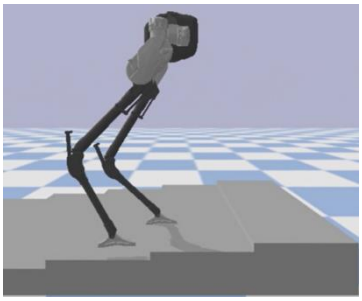


图 15 上图为在 pybullet 中的 cassie 爬楼梯演示。

4 基于 Full order model 的步态规划

在后续路径规划任务上，我们采用了 Caltech 开源的一套基于 Full order model 的 Cassie 行走模型来完成的。

4.1 Full order model vs Reduced order model

虽然基于三维线性倒立摆模型的步态规划在运算上更快，效率更高，但是这种只考虑到系统的

局部会限制到整体运动行为的灵活性并影响到能量消耗效率，并且可能需要到额外的优化以确保可行的步行模式[8]。同时，采用线性倒立摆模型，也会面临着如下的众多挑战，利用机器人的全部能力，考虑电机和关节的限制，以及决定如何将全阶系统的状态与线性倒立摆的运动相关联。

但线性倒立摆模型的一个重要特征就是质心速度可以根据落脚点进行调节，更准确的来说，是通过改变游动脚与地面接触时的角度。事实上，线性倒立摆 (LIP)的质心速度的开环中心在纵向和横向方向上为一个积分器，因此可以用比例控制器稳定。

4.2 Virtual constraint

通过虚约束[10]实现在 Cassie 上单独的周期性步态，这些虚约束又根据机器人的当前速度进行“增益调节”，从而创建连续稳定的步态。由此产生的闭环系统具有这样的特性，即其质心速度在纵向和横向方向上近似演变为一个积分器，因此类似于倒立摆模型，行走速度可以通过“落脚点的位置”来调节到一个期望值。

虚约束是机器人广义坐标间的函数关系，用于系统反馈控制中。虚约束定义为如下形式：

$$y = h(q, \tau, \alpha) = h_0(q) - h_d(\tau, \alpha),$$

其中 h0 为被调节的物理量，hd 为期望值。控制器被设计来使得虚约束输出 y 等于 0。

对于 Cassie 机器人来说，关于 h0 控制量的选择，驱动关节与行走模式间有着限制性的物理意义，因此 Caltech 采用了调节躯干的姿态角，触地脚和游动脚的长度，游动脚的姿态角，如下图所示。并且这些物理量是与步态行为直接相关的。

$$h_0(q) = \begin{bmatrix} q_{\text{roll}} \\ q_{2\ \text{st}} \\ q_{\text{pitch}} \\ q_{\text{LL st}} \\ q_{\text{LR sw}} \\ q_{2\ \text{sw}} \\ q_{\text{LP sw}} \\ q_{\text{LL sw}} \\ q_{\text{FP sw}} \end{bmatrix} \begin{pmatrix} \text{torso roll} \\ \text{stance hip yaw} \\ \text{torso pitch} \\ \text{stance leg length} \\ \text{swing leg roll} \\ \text{swing hip yaw} \\ \text{swing leg pitch} \\ \text{swing leg length} \\ \text{swing foot pitch} \end{pmatrix}$$

4.3 Gait library

Caltech 所开源的模型控制器是基于 gait library[9]。它允许在保证关节角度、地面作用力范围、电机功率的限制下，使用全身动力学模型来设计周期行走步态，并且在给定的速度下优化每一步的能量消耗。

而在上文中所提到的虚约束中的目标值 hd，是通过对离散的 gait library 线性插值得到的，其中每一个 gait library 都对应于一个特定的行走速度。在求取 gait library 时是通过最小化以下代价函数得到的。

$$\text{Domain Cost} = \int_{\tau=0}^{\tau=1} \left(||u||^2 + c\,|q_{pitch}|^2 + c\,|q_{roll}|^2 + c\,|q_{1\text{L}}|^2 + c\,|q_{2\text{L}}|^2 + c\,|q_{1\text{R}}|^2 + c\,|q_{2\text{R}}|^2 \right) d\tau.$$

Average sagittal velocity, $\bar{v}_x$	$= v_i$ m/s
Average lateral velocity, $\bar{v}_y$	= 0 m/s
Step time	= 0.4 s
Torque for stance foot pitch	= 0 Nm
Friction cone, μ	< 0.6
Mid-step swing foot clearance	> 0.15 m
Absolute swing foot pitch	= 0 rad
Distance between feet	> 0.2 m
Distance between pelvis and stance foot	$\in$ (0.5, 1) m
Swing foot velocity on impact (x and y)	= 0 m/s
Swing foot velocity on impact (z)	$\in$ (-1, 0) m/s

图 16 gait optimization 的代价函数和限制约束

4.4 Controller

控制器框架图如下所示，

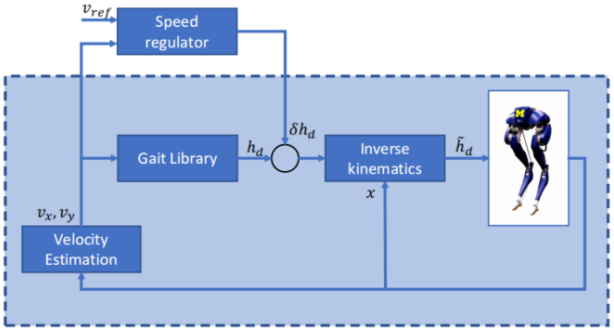


图 17 行走模型的控制器

从图中可看出，反馈控制环由虚约束和步态库组成，以维持机器人的稳定行走。同时整理来看，整个闭环系统在调节质心速度时与线性倒立摆模型相似，可以通过调节游动腿的 pitch 和 roll angle 来调节速度。

5 路径规划

5.1 不同路径规划算法比较

本小组主要对比了所使用的局部规划算法——人工势能场算法与传统全局规划，图搜索算法 A*算法进行比较。

参考移动机器人路径规划算法的分类。大致可以分为全局规划与局部规划两大类型，主要是根据是否已知全局情况来分类，实际上许多算法既可以作为全局规划也可以作为局部规划，取决于算法的设计。这里本小组选择了全局规划的图搜索算法中较为经典的 A*算法作为比较的对象。A*算法主要原理是使用运动启发式函数在静态地图中寻找避障

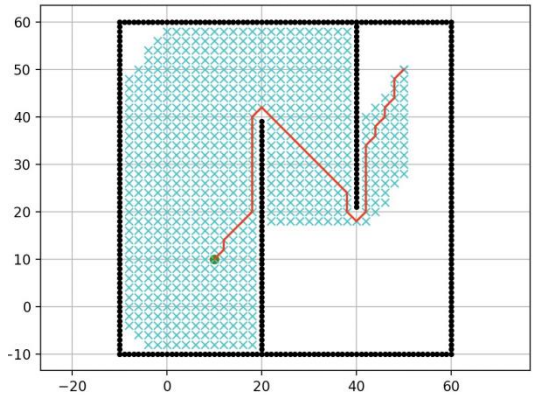
的路径。从原理上可以被认为是 Dijkstra 算法的拓展，借助启发函数的引导，A*算法通常拥有更好的性能。A-Star 算法是一种静态路网中求解最短路径最有效的直接搜索算法，公式表达为：

$$f^*(n) = g^*(n) + h^*(n)$$

其中， f*(n) 是从初始状态经由状态 n 到目标状态的最小代价估计，

g*(n) 是在状态空间中从初始状态到状态 n 的最小代价，

h*(n) 是从状态 n 到目标状态的路径的最小估计代价。



A*算法运行结果示意

如果 h(n) <= n 到终点的实际距离，A*算法可以找到最短路径，但是搜索的点数多，搜索范围大，效率低。

如果 h(n) > n 到终点的实际距离，搜索的点数少，搜索范围小，效率高，但是得到的路径并不一定是最短的。

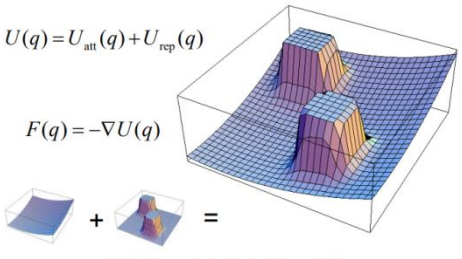
h(n) 越接近 n 到终点的实际距离，那么 A*算法越完美。（个人理解是如果用曼哈顿距离，那么只需要找到一条长度小于等于该距离的路径就算完成任务了。而使用对角线距离就要找到一条长度大于等于对角线距离且最短的路径才行。）

若 h(n)=0，即 f(n)=g(n)，A*算法就变为了 Dijkstra 算法。

若 h(n) 远远大于 g(n) ，那么 f(n) 的值就主要取决于 h(n)，A*算法就演变成了 BFS 算法。[2]

A*算法原理简单，对环境反应迅速，但是计算量较大、占用内存多，搜索出现的路径可能并非最有。

与 A*算法进行比较，人工势能场算法采用的原理是让机器人在目标点引力场和周边障碍物斥力场的共同作用下进行运动，每次均在对应的势能场中寻找梯度最低的方向，这样寻找出来的路径平滑，从算法的角度上而言方法简单，易于实现。但是容易出现局部极小值以及在对角线方向上或者大型障碍物的情况下容易出现无法避让的情况。



人工势能场算法主要公式为势能场函数以及梯度算法。

$$U(q) = U_{att}(q) + U_{rep}(q)$$

Combined Potential

$$U_{att}(q) = \begin{cases} \frac{1}{2}\zeta d^2(q, q_{goal}), & d(q, q_{goal}) \le d^*_{goal}, \\ d^*_{goal}\zeta d(q, q_{goal}) - \frac{1}{2}\zeta (d^*_{goal})^2, & d(q, q_{goal}) > d^*_{goal}. \end{cases}$$

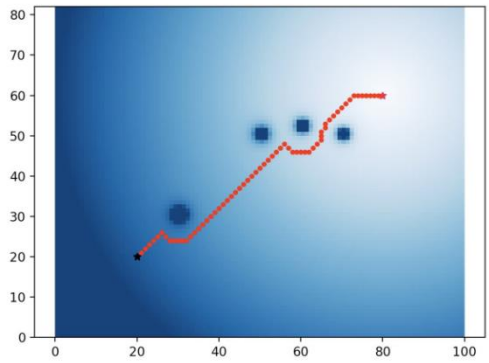
$$\nabla U_{att}(q) = \begin{cases} \zeta(q - q_{goal}), & d(q, q_{goal}) \le d^*_{goal}, \\ \frac{d^*_{goal}\zeta(q - q_{goal})}{d(q, q_{goal})}, & d(q, q_{goal}) > d^*_{goal}, \end{cases}$$

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{D(q)} - \frac{1}{Q^*})^2, & D(q) \le Q^*, \\ 0, & D(q) > Q^*, \end{cases}$$

whose gradient is

$$\nabla U_{rep}(q) = \begin{cases} \eta\left(\frac{1}{Q^*} - \frac{1}{D(q)}\right)\frac{1}{D^2(q)}\nabla D(q), & D(q) \le Q^*, \\ 0, & D(q) > Q^*, \end{cases}$$

人工势能场避障算法运行结果示意：



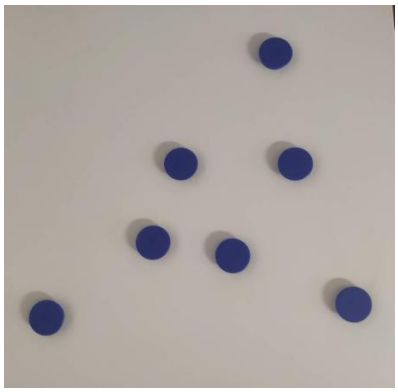
针对人工势能场算法的局部最优解问题，一般有两种应对方案，其一是通过调整步长与其他势能场参数躲避最优解位置，其二是通过局部最优解检测函数，在发现陷于局部最优解时采用某种方法使机器人走出最优解位置。

本小组采用人工势能场算法参考轮式机器人运动规划，实际上针对行走机器人使用的路径规划算法会更加复杂，因为腿式机器人可以在跨越或者踏上一定高度的障碍物或者通过腿部轨迹规划避障，所以会有更加复杂的算法用于腿式机器人路径规划，由于所学知识的限制小组目前没有使用这样的算法，而是参考轮式机器人的路径规划算法。

[2] https://zhuanlan.zhihu.com/p/385733813

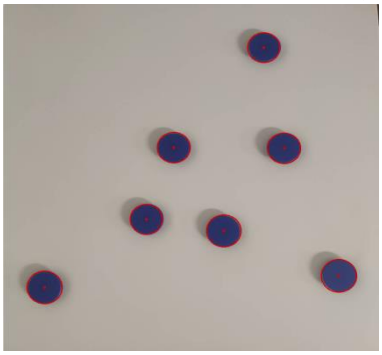
1） 障碍物视觉识别定位与路径规划

在完成路径规划算法的选取之后，小组计划进行障碍物的视觉识别与定位。由于 Cassie 模型上并没有携带摄像头，所以小组采用全局摄像头对障碍物进行拍摄，小组使用圆形物品来代表障碍物，这是由于圆形物品在计算势能场时对称性更强，有利于简化势能场的计算。



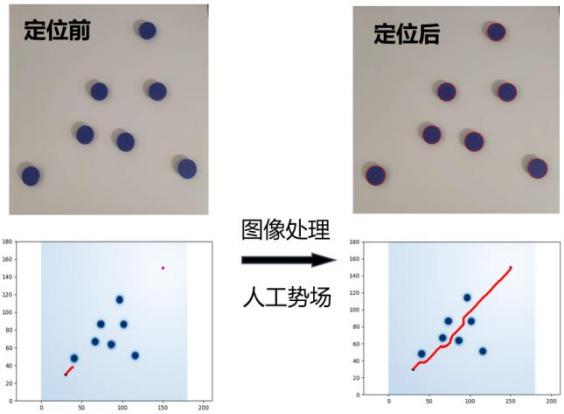
障碍物图片

对图片进行视觉处理，小组使用 Python 进行图像处理，转化成灰度图使用 opencv 提取特征轮廓并寻找障碍物轮廓与中心点，由于使用的是灰度，可能需要对障碍物产生的光影效果进行处理，通过筛选光影的面积与形状去除光影的影响。从而对障碍物本身进行定位与面积大小的计算。



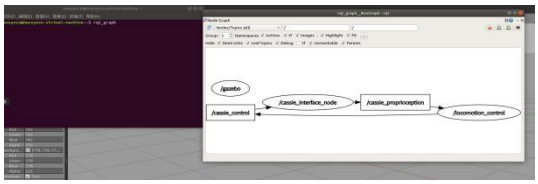
处理后的图片

将生成的障碍物位置与大小导入到路径规划的代码中，设置起始点与结束位置，之后进行路径规划，整体完整运行结果如下：



5.2 Ubuntu + Gazebo 绕桩演示与路径仿真

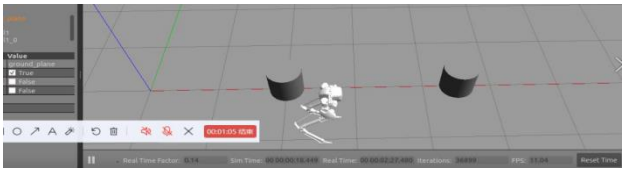
由于小组所做的 3D-LIP 在 Pybullet 环境中没有添加运动控制器，没有办法下地走动，因此在路径规划的算法上采用了基于 Caltech 的 Gazebo 运动模型。在仿真环境上选择了 Ubuntu，Gazebo 的仿真环境，使用 ROS 进行节点通信控制 Gazebo 中模型的运动参数。



ROS 节点示意图

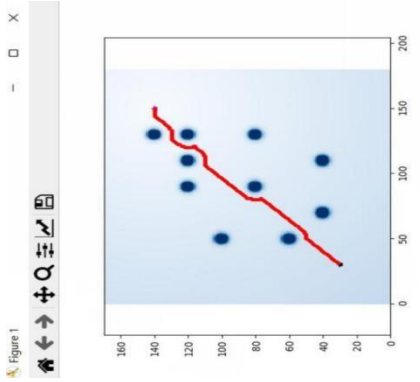
小组主要修改/Cassie_interface_node 中的参数，直接调整控制 Cassie 运动摇杆的位置，映射到 Cassie 机器人在矢状面与冠状面上的运动速度，从而进一步控制机器人的运动轨迹。

对于控制逻辑进行验证，小组制作了一个简单的绕桩模型，让 Cassie 在仿真环境中行走一个 Sin 的波形，冠状面速度采用 cos 函数，矢状面匀速运动。

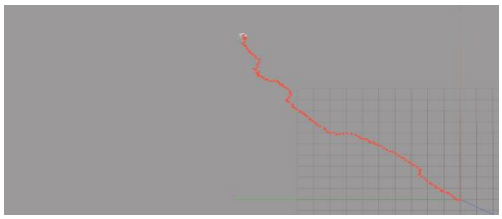


Cassie 机器人绕桩运动示意

验证控制逻辑可行之后，将对应的路径规划数据导出为每一小段路径运动的方向，此处代码设计为每次向周围八个方向运动，将运动方向导出为由 -1，0，1 组合表示的八个方向并导入 C++，Cassie_interface_node.cpp 文件，按照顺序读取设定好的每一小段 ROS::Time 内的运动速度，输出给对应的 Subscriber，最终映射到 Gazebo 内 Cassie 机器人的矢状面与冠状面运动速度。



所规划的路径



Gazebo 中实际运动路径

可以看出这样的控制算法较好的重现了规划的路径。虽然由于直接控制的是控制 Cassie 的摇杆值而不是直接映射到速度上，可能 Gazebo 中运动路径会与所规划的路径存在一点区别，控制器对于输出的速度进行了一定的插值平滑处理。本次小组使用的仿真没有使用 Caltech 设计的 QP 控制器，行走步态也较为稳定，行走的路径也可以保持大体一致。

6 合作分工与组员贡献情况

22% 夏逸轩 ：路径规划、gazebo 运动仿真与行走轨迹规划、ppt 制作、组长
22% 周剑翔：3D-LIP、Pybullet 仿真、ppt 制作
20% 杨博宇：3D-LIP、Pybullet 仿真、ppt 制作
12% 张熙芃：3D-LIP
12% 杨昕：路径规划各算法分析与对比、ppt 制作
12% 魏锦启：环境设置、障碍物视觉识别、路径规划

7 问题及难点总结(解决方案)

问题及难点总结：

1) Gazebo 模型仿真时间长，Linux 系统与 Windows 系统在文件编码与程序运行上具有较大的差异。虚拟机与双系统导致 Ubuntu 运行受硬件限制。
2) Gazebo 模型运动采用摇杆控制，自带速度的平滑处理，调整需要一定的时间。导致速度的切换频率不能过快且与实际规划出的路径可能存在一定的差异。
3) Ubuntu 仿真环境的配置以及 Caltech 开源仿真运行存在一些环境配置上的繁琐之处，在项目过程中对项目进度产生了一定的影响。
4) 仿真环境中，cassie 模型的足端与地面为线基础，其动态行走过程中难以保持稳定。在基于 3D-LIP 的步态规划中，因自身知识水平限制，难以搭建出一套步行稳定控制器来保证在行走过程中跟踪预期轨迹并且保证不倾倒。

解决方案：

1） 在有条件的情况下使用独立系统，直接在单个系统环境中进行编程。

2） 通过尽可能地选择合适的速度更新时间以及比例来使整体运动较为协调，可能通过直接修改控制器参数来跳过摇杆控制的环节。

3） 在基于 3D-LIP 的步态规划演示中，每个关节采用了位置控制而非力矩控制，以保证在仿真环境中 cassie 能按照预期轨迹运动，同时也对 cassie 的 base 进行了自由度约束，保证其不会倾覆。

8 总结

本文以开源 Cassie 机器人 URDF 模型为基础，首先描述机器人关节的自由度以及全身机械机构，然后介绍了基于线性倒立摆模型的控制方法，并在 Pybullet 中展示了机器人的直立行走、爬楼梯以及转弯能力。此外，我们还在 Gazebo 平台上基于 Caltech 的控制器进行了机器人的路径规划，通过不同算法之间的对比，得出最优算法策略。在视觉定位障碍物之后，使用人工势场法自主规划路径。

参 考 文 献

[1]https://zhuanlan.zhihu.com/p/385733813
[2]https://www.cs.cmu.edu/~motionplanning/lecture/Chap4-Potential-Field_howie.pdf
[3] Reher, Jenna and Aaron D. Ames. "Inverse Dynamics Control of Compliant Hybrid Zero Dynamic Walking." Submitted to 2021 IEEE ICRA and Robotics and Automation Letters (RA-L).
[4] Reher, Jenna, Wen-Loong Ma, and Aaron D. Ames. "Dynamic walking with compliance on a Cassie bipedal robot." 2019 18th European Control Conference (ECC). IEEE, 2019.
[5] Gong, Y. , Hartley, R. , X Da, Hereid, A. , Harib, O. , & Huang, J. K. , et al. (2018). Feedback control of a cassie bipedal robot: walking, standing, and riding a segway.
[6] Open-sourced repository for the C++ controller code used in this work on hardware and for use in a Gazebo simulation.https://github.com/jpreher/cassie_documentation
[7]https://www.researchgate.net/publication/298659076_Introduction_to_Humanoid_Robotics
[8] Sebastien Dalibard, Antonio El Khoury, Florent Lamiraux, Alireza Nakhaei, Michel Ta¨ıx, and Jean-Paul Laumond. Dynamic walking and whole-body motion planning for humanoid robots: an integrated approach. The International Journal of Robotics Research, 32(9-10):1089–1103, 2013.
[9] Xingye Da, Omar Harib, Ross Hartley, Brent Griffifin, and Jessy W Grizzle. From 2D design of underactuated bipedal

gaits to 3D implementation: Walking with speed tracking. IEEE Access, 4:3469– 3478, 2016.

[10] Carlos Canudas-de-Wit. On the concept of virtual constraints as a tool for walking robot control and balancing. Annual Reviews in Control, 28(2):157–166, 2004.