SUPER SPIDER 六足机器人

田丰　沈浩瑜　苏汉祺　杨博文　刘迪涵

摘要：基于经典六足机器人模型，分析其腿部结构和行走的三角步态，针对研究者设计的任务，建立并求解正、逆运动学方程。首先通过 SOLIDWORKS、MATLAB 等软件，对六足机器人的步态进行仿真；然后利用实验室的 ROBOTIS BIOLOID Premium 套件，搭建真实的六足机器人，以前面的研究为基础，完成多项具有挑战性的任务，并基于 RoboPlus 平台对机器人编程。项目成果包括，提出了六足机器人前进步态的优化方案并在硬件上实现，完成了六足机器人八个方向的移动、旋转、扭动等基础功能，使六足机器人可以完成爬楼梯、跳舞、翻身等复杂任务，并且实现了自动控制。

关键词：六足机器人；正逆运动学；三角步态；机器人控制

SUPER SPIDER Hexapod Robot

TIAN Feng　SHEN Haoyu　SU Hanqi　YANG Bowen　LIU Dihan

Abstract: Based on the classical hexapod robot model, analyze its leg structure and the triangular gait of walking. Establish and solve the forward and inverse kinematic equations for several tasks designed by the researchers. Firstly, through SOLIDWORKS, MATLAB and other softwares, simulate the gait of the hexapod robot; Then the laboratory's ROBOTIS BIOLOID Premium suite is used to build a real hexapod robot, to complete several challenging tasks based on the previous research, and to program the robot based on the RoboPlus platform. The project results include that the optimization of the pre-progress state of the hexapod robot is proposed, and it is realized in hardware; Basic functions such as movement in eight directions, rotation and twisting are completed; The hexapod robot can complete complex tasks such as climbing the stair, dancing, turning over; And the automatic control of the robot is also realized.

Key words: Hexapod robot; Forward and inverse kinematics; Triangular gait; Robot control

0　研究动机

近年来，仿生学已成为工程学研究的热点。仿生学将生物系统的结构、特质、功能、能量转换、信息控制等各种优异的特征运用到各个领域，极大推进了技术进步和工程发展。在机器人研究领域，也出现了诸如机器狗、机器蝙蝠、机器蜘蛛等仿生机器人[1]。仿生机器人具有较高的灵活性和较大的活动范围，这使它们可以代替人类完成危险的任务，也可以探索人类无法到达的极端环境，对于狭缝、抢险救灾现场等极端场景有更好的适应性。因此仿生机器人具有很高的学习和研究价值[1]。

相关资料指出[2,3]，技术相对成熟的轮式和履带式机器人，运动模式保持连续性。这使其虽然在平坦地形上行驶速度快、性能好，但在松软或崎岖的地面上运动效果不佳。与轮式机器人相比，足式机器人的步态呈现离散的特点。大部分足式机器人的结构和控制基于仿生学设计，其在运动过程中，能在可到达的平面范围内选择结构最优的一组支撑点，增强对崎岖地形的适应性；且其自由度多，运动灵活性更好。

本项目研究的六足机器人就是一种经典的足式机器人，其设计模仿昆虫（如蚂蚁、蜘蛛等）的结构[4]。六足机器人的优势在于，其运动模式与昆虫相似采用三角步态，即以三条腿为一组进行运动，每个步态都可以形成一个三角形的稳定结构。在这样的结构下，机器人也可以随时停止移动，因为重心总是落在稳定的三角形之内[5]。所以六足机器人对坑洼山地的机动性和适应性更强，也具有更高的避障和越障能力，这些特性也使得对机器人的控制更加方便和稳定。在保持平衡性的方面，其腿的数目也可以是冗余的：路面较平坦时，仅需四条腿就可以站立和移动，冗余的两条腿能够在特定状态下充当“手”的功能完成系列任务。因此，六足机器人的步态规划、运动控制、功能应用等方面被学界

关注并进行了大量研究，也成为我们小组感兴趣的项目课题。在机器人研究中，六足机器人有时又被称为“蜘蛛机器人”，所以我们小组将课题名称定为“SUPER SPIDER”，希望能够通过本次项目探索六足机器人更多“超级”功能。

研究动机和目标方面，我们小组希望使用正逆运动学方法推导不同位姿下六足机器人各个关节的角度，设计机器人步态，并在仿真和现实环境中控制机器人朝各方向流畅地运动，以及完成爬楼梯、翻身等复杂度较高的任务。同时，我们小组希望在建立并求解六足机器人的运动学模型、控制机器人运动并完成特定任务的过程中，运用本学期机器人建模与控制课程所学的知识，更深入地理解和掌握学习内容，并为以后自己的科研积累经验。

1 模型建立

1.1 三维模型搭建

为了给理论推导和模型仿真提供基础，首先在 SOLIDWORKS 中搭建六足机器人三维模型。本小组对六足机器人构想如下：设定六足机器人的每条腿拥有三个关节，且均为旋转关节；机器人主体为长方体，六条腿对称且等角度间隔分布在主体周侧。我们从模型官网导出各个部件的.stp 文件，在 SOLIDWORKS 中添加零件之间的配合关系，生成三维模型如图 1。

图 1 六足机器人三维模型

1.2 基于 SIMULINK-URDF 的模型仿真

首先在 SOLIDWORKS 模型上，手动建立世界坐标系、每个旋转关节的坐标系和旋转轴。共生成 25 个坐标系和 18 个基准轴。

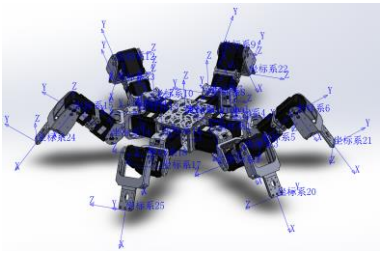


图 2 机器人各关节坐标系

利用 sw_urdf_exporter 插件从 SOLIDWORKS 中导出 urdf 文件及其.stl 格式三维模型数据。修改 urdf 文件中的模型外观读取地址，将其导入 MATLAB-SIMULINK 中，并将各个关节的力控调整为外部输入模式，模型结构树如图 3 所示。

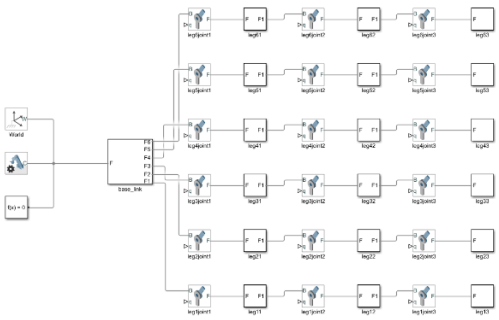


图 3 初始模型结构树

然后添加地面模型和六自由度运动关节模块，此时机器人与地面模块之间的关系不受约束。为了使机器人与地面有仿真接触效果，能够完成前进、旋转等任务，使用地面接触模块加入作用力：在 contact force 文件夹中选择“球与面”接触模块，并设置地面大小与实际地面模块大小相同，以防止穿模。添加地面模块后结构树如图 4。

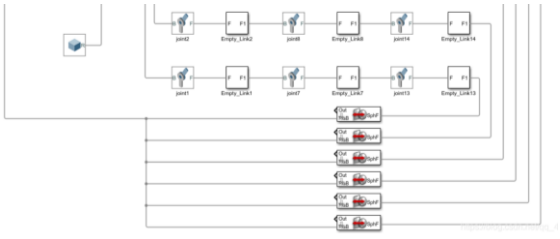


图 4 接触模块结构树

模型配置完成后，六足机器人能够在地面上保持正常站立的姿态，效果如图 5。

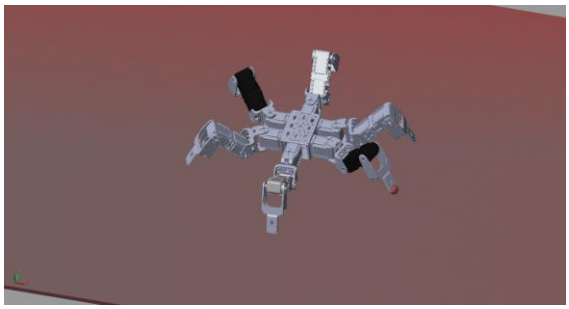


图 5 URDF 模型导入完成

1.3 基于 MATLAB 的棍棒模型仿真

六足机器人棍棒模型的构建主要是在开源代码[6]、教程和其他相关资料的基础上完成的，我们小组进行了复现、优化和拓展工作。

构建模型所需的几个基本运动学求解函数如表 1

所示：

表 1　MATLAB 运动学求解函数表

函数	*功能*
Legs.m	计算关节角度
rot.m	计算旋转矩阵
TrnoP.m	从地面坐标系到主体坐标系的变换矩阵
TrnoP_inv.m	从主体坐标系到地面坐标系的变换转换

上述函数将在 2.3 节中详细介绍，这里主要介绍 Hex_InvKin.m 程序文件。该函数实现了将所有的地面坐标转变到六足本体框架下对应的坐标，根据机器人足部的逆运动学解，计算每个关节点和主体点的地面坐标，并绘制六足机器人的整体图像。其流程图如下所示。

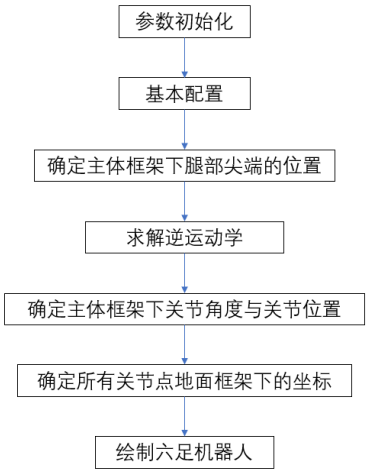


图 6　Hex_InvKin.m 程序结构

(1) 参数初始化：定义一条腿三个连杆的长度和主体部分的半径大小，从而确定六条腿开始的位置。给定一条腿腿部尖端的位置后，通过 rot.m 函数我们就可以确定所有腿部尖端的位置。

```
global l0 l1 l2 lt lf lc
l0=2;%cm
l1=6;%cm
l2=9;%cm
lc=l0;
lf=l1;
lt=l2;
P=zeros(3,1); % center of the body
h=7.6;%cm % Radios of platform
P1=P+[h*sin(pi/6);h*cos(pi/6);0];
P2=P+[h;0;0];
P3=P+[h*sin(pi/6);-h*cos(pi/6);0];
P4=P+[-h*sin(pi/6);-h*cos(pi/6);0];
P5=P+[-h;0;0];
P6=P+[-h*sin(pi/6);h*cos(pi/6);0];
P_MB=[P1,P2,P3,P4,P5,P6]; % construct the
main body, and shows the leg begin position
% The edges of the body where limbs begin in
main bodie's frame

% Giving a sample legs tips position:
%Q1=[8;13.8564;0];% this give the position
of the P1 leg tip
Q1=[7.5000;12.9904;0];% this give the
position of the P1 leg tip
phi=[0,pi/3,2*pi/3,pi,4*pi/3,5*pi/3]; %
the angle between Pi is 60¡ã
for i=1:6
Q2(:,i)=(rot(phi(i))*Q1);
% multiply rotation matrix with the Qi to get
each leg tip position
end
```

(2) 基本配置：设置主体与地面间的距离以及 yaw、roll、pitch 三个角度的大小。

```
OP=[0;0;10]; % the position of the frame of
main body from ground frame.
% distance between main body and ground
Ltip=Q2;  % legs tips position
yaw=0;% Around Z
roll=0;% Around Y
pitch=0;% Around X
```

(3) 确定主体框架下腿部尖端的位置：由于我们需要在主体框架坐标系下计算逆运动学问题。因此，我们利用 TrnoP.m 函数构建变换矩阵，将腿尖位置的坐标进行转换。

```
Trn_op=TrnoP(yaw,roll,pitch,-OP); %
construct the T matrix ( R (rotation)
+P(position))
for i=1:6
   Ltip_MB(:,i)=Trn_op*[Ltip(:,i);1];
end
Ltip_MB=Ltip_MB(1:3,:);   % leg tip
position about main body
```

(4) 求解逆运动学：利用 fslove 函数求解 Legs.m 函数中的方程，得到逆运动学解。

```
% first we should define the X,Y,Z distance
of the leg tip from the leg frame positioned
on the body ( P1,P2...)
global Ltip_JF
Ltip_JF=Ltip_MB-P_MB; % Leg tip in Joint
Frame (distance between leg tip position and
Pi)
[t,fval,exitflag]=fsolve('Legs',zeros(3,6
)); % find the joint 1,2,3 angle
% x = fsolve(fun,x0) starts at x0 and tries
to solve the equations fun(x) = 0, an array
of zeros
if exitflag==1
t1=t(1,1:6)'; % joint 1 angle
t2=t(2,1:6)'; % joint 2 angle
t3=t(3,1:6)'; % joint 3 angle
```

(5) 确定主体框架下关节角度与关节位置：根据上面求出的关节角度和已知的一条腿三个连杆的长度，确定每一个关节的位置坐标。

```
% every 3 rows denote the x,y,z coordinates
of the joints from body to end point on the
ground.
% 4 point -> from body to end point -> 4
columns
% 3 rows -> x y z axis
for i=1:6
Joints_MB(3*i-2:3*i,1:4)=[P_MB(1,i),...
    P_MB(1,i)+lc*cos(t1(i)),...

P_MB(1,i)+lc*cos(t1(i))+lf*cos(t2(i))*cos
```

```
(t1(i)),...
    P_MB(1,i)+cos(t1(i))*(lc+lf*cos(t2(i))
+lt*cos(t2(i)+t3(i)));...
    % x axis
    P_MB(2,i),...
    P_MB(2,i)+lc*sin(t1(i)),...

P_MB(2,i)+sin(t1(i))*(lc+lf*cos(t2(i))),.
..
    P_MB(2,i)+sin(t1(i))*(lc+lf*cos(t2(i))
+lt*cos(t2(i)+t3(i)));...
    % y axis
    P_MB(3,i),...
    P_MB(3,i),...
    P_MB(3,i)-lf*sin(t2(i)),...
    P_MB(3,i)-lf*sin(t2(i))-lt*sin(t3(i)+t
2(i))];
    % z axis
end
```

(7) 确定所有关节在地面坐标系下的坐标：已知在关节坐标系下的坐标，用 TrnoP_inv.m 函数可以把所有坐标点转换到地面坐标系下的坐标大小。

```
Trn_po=TrnoP_inv(yaw,roll,pitch,-OP);
for i=1:6
for j=1:4
    J1=Trn_po*[Joints_MB(3*i-2:3*i,j);1];
    J1=J1(1:3);
    Joints_GF(3*i-2:3*i,j)=J1;  % get the
ground coordinate position
end
end
BJ=[[Joints_GF(1,1),Joints_GF(4,1),Joints
_GF(7,1),...

Joints_GF(10,1),Joints_GF(13,1),Joints_GF
(16,1),Joints_GF(1,1)];...

[Joints_GF(2,1),Joints_GF(5,1),Joints_GF(
8,1),...

Joints_GF(11,1),Joints_GF(14,1),Joints_GF
(17,1),Joints_GF(2,1)];...

[Joints_GF(3,1),Joints_GF(6,1),Joints_GF(
9,1),...

Joints_GF(12,1),Joints_GF(15,1),Joints_GF
(18,1),Joints_GF(3,1)]];% Body joints in
ground frame
```

(8) 绘制六足机器人：使用 plot3 函数进行绘制。因为已知所有关节的位置坐标，所以我们可以轻松地完成模型绘制。

```
for i=1:6

plot3(Joints_GF(3*i-2,:),Joints_GF(3*i-1,
:),Joints_GF(3*i,:),'-o','Linewidth',2,..
.            'MarkerEdgeColor','k',...
      'MarkerFaceColor','c',...
            'MarkerSize',5)
    hold on
end
for i=1:6
plot3([Joints_GF(1,1),Joints_GF(4,1),Join
ts_GF(7,1),...
Joints_GF(10,1),Joints_GF(13,1),Joints_GF
(16,1),Joints_GF(1,1)],...
[Joints_GF(2,1),Joints_GF(5,1),Joints_GF(
8,1),...
Joints_GF(11,1),Joints_GF(14,1),Joints_GF
(17,1),Joints_GF(2,1)],...
[Joints_GF(3,1),Joints_GF(6,1),Joints_GF(
9,1),...
Joints_GF(12,1),Joints_GF(15,1),Joints_GF
(18,1),Joints_GF(3,1)],'k','Linewidth',3)
end
```

至此，我们成功构建出了六足机器人的棍棒模型，如图 7 所示。

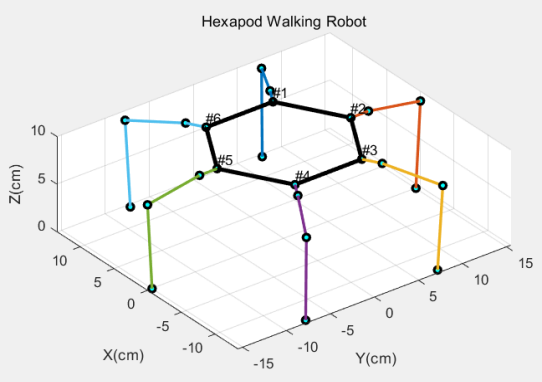


图 7　六足机器人棍棒模型

2　运动学方程及求解

2.1　正向运动学求解——DH 方法

首先分析单腿模型，如下图所示建立各个关节坐标系。

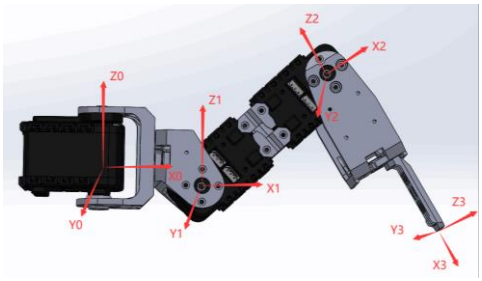


图 6　单腿坐标系

为方便计算，参考平面坐标机构而坐标系建立，保持 y 轴垂直与纸面，使得α与d取值均为 0。设腿三段长度分别为 a_0、a_1、a_2，测量得到 a_0=53mm、a_1=83mm、a_3=89mm。则可得单腿 D-H 参数如表 2。

表 2　单腿 D-H 参数表

连杆	a	α	d	θ
1	53	0	0	θ_1
2	83	0	0	θ_2
3	89	0	0	θ_3

由 D-H 参数表可得旋转变换矩阵：

$$T_1^0 = Rot_{z,s_1} Trans_{x,a_0} \quad (1)$$

$$T_2^1 = Rot_{y,s_2} Trans_{x,a_1} \quad (2)$$

$$T_3^2 = Rot_{z,s_3} Trans_{x,a_2} \quad (3)$$

将式(1)(2)(3)相乘得到变换矩阵：

$$T_3^0 = T_1^0 T_2^1 T_3^2 = \begin{bmatrix} R & P \\ 0 & 1 \end{bmatrix} \quad (4)$$

用$c_n = cos\theta_n, s_n = sin\theta_n$代换，其中 R、P 矩阵参数为：

$$\mathrm{R} = \begin{bmatrix} c_1c_2c_3 & -s_1 & c_1c_2s_3 \\ s_1c_2c_3 & c_1 & s_1c_2s_3 \\ -s_2c_3-c_2s_3 & 0 & -s_2s_3+c_2c_3 \end{bmatrix}$$

$$\mathrm{P} = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} a_2c_1c_2c_3 + a_1c_1c_2 + a_0c_1 \\ a_2s_1c_2c_3 + a_1s_1c_2 + a_0s_1 \\ -a_2s_2c_3 - a_2c_2c_3 - a_1c_2 \end{bmatrix}$$

至此，我们得到了末端位置与连杆长度和关节角度的关系。

2.2 逆向运动学求解——几何法

首先绘制单腿结构的简易模型如下：

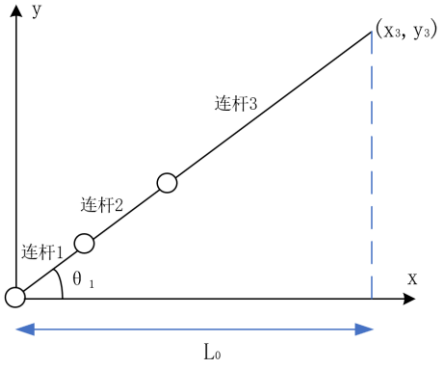


图 7　单腿结构俯视图

以 x 轴为旋转起始位置，末端坐标为(x_3,y_3)。设连杆 1、连杆 2、连杆 3 的长度分别为 L_1、L_2、L_3，Z_{off} 为关节 0 与地面的距离。在图中做出辅助角 q_1、q_2和 Φ：

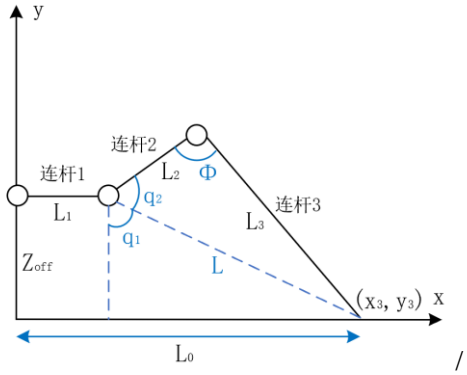


图 8　单腿结构侧视图

由几何关系，利用勾股定理和解三角形公式可以得到辅助角和长度 L：

$$L = \sqrt{Z_{off}{}^2 + (L_0 - L_1)^2}$$

$$q_1 = tan^{-1}\frac{Z_{off}}{(L_1 - L_C)}$$

$$q_2 = cos^{-1}\frac{\sqrt{L_2^2 + L^2 - L_3^2}}{2 \times L_2 \times L}$$

$$\Phi = cos^{-1}\frac{\sqrt{L_2^2 + L_3^2 - L^2}}{2 \times L_2 \times L_3}$$

由图可得，关节旋转角θ_1、θ_2、θ_3为：

$$\theta_1 = tan^{-1}\frac{y_3}{x_3} \quad (5)$$

$$\theta_2 = q_1 + q_2 \quad (6)$$

$$\theta_3 = \Phi - 180 \quad (7)$$

2.3 使用 MATLAB 求解棍棒模型正、逆运动学

这里对表 1 中求解棍棒模型运动学方程用到的几个函数做详细介绍：

(1) Legs.m 函数主要实现如何计算一条腿上三个关节角度值的变化。

```
LEGS_inv(3*i-2)=t1(i)-atan2(y,x);
% calculate angle difference between current
theta 1 and optimization theata 1
c=((x-l0*cos(t1(i)))^2+(y-l0*sin(t1(i)))^
2+z^2)^.5; % calculate the distance between
joint 1 and joint 3
B=acos((-l2^2+l1^2+c^2)/(2*l1*c)); %
calculate angle between l1 and c
LEGS_inv(3*i-1)=t2(i)-(asin(-z/c)-B); %
calculate angle difference between current
theta 2 and optimization theata 2
% asin(-z/c)-B : calculate angle of theta2
C1=pi/2-B;
h=l1*sin(B);
C2=acos(h/l2);
% C1+C2 : angle between l1 and l2
% pi-C1-C2: angle of theta 3
LEGS_inv(3*i)=t3(i)-(pi-C1-C2); %
calculate angle difference between current
theta 3 and optimization theata 3
```

(2) rot.m 函数主要是用于计算旋转矩阵。当已知一条腿的位置信息后，可以根据旋转角来计算其余所有腿的位置坐标。

```
function RT=rot(phi)
RT=[cos(phi),sin(phi),0;
    -sin(phi),cos(phi),0;
    0,0,1];
```

(3) TrnoP.m 函数实现了从地面框架到主体框架的坐标变换矩阵。输入参数为绕 x、y、z 的旋转角度和 OP（即主体框架到地面框架的距离）。

```
function Trn_OP=TrnoP(yaw,roll,pitch,OP)
a=-yaw; %yaw: rotationg around Z axis
b=-roll; %roll: rotation around Y axis
v=-pitch; %pitch: rotation around X axis
Rz=[cos(a),-sin(a),0;sin(a),cos(a),0;0,0,
1];
Ry=[cos(b),0,sin(b);0,1,0;-sin(b),0,cos(b
)];
```

```
Rx=[1,0,0;0,cos(v),-sin(v);0,sin(v),cos(v
)];
Rxyz=Rz*Ry*Rx; % Yaw-Pitch-Roll
Representation
Trn_OP=[Rxyz,OP];
Trn_OP(4,1:4)=[0,0,0,1];
```

(4) TrnoP_inv.m 函数实现了从主体框架到地面框架的坐标变换矩阵。输入参数与 TrnoP.m 相同。

```
function
Trn_OP_inv=TrnoP_inv(yaw,roll,pitch,OP)
a=-yaw; %yaw: rotationg around Z axis
b=-roll; %roll: rotation around Y axis
v=-pitch; %pitch: rotation around X axis
Rz=[cos(a),-sin(a),0;sin(a),cos(a),0;0,0,
1];
Ry=[cos(b),0,sin(b);0,1,0;-sin(b),0,cos(b
)];
Rx=[1,0,0;0,cos(v),-sin(v);0,sin(v),cos(v
)];
Rxyz=Rz*Ry*Rx; % Yaw-Pitch-Roll
Representation
RxyzT=Rxyz';
Trn_OP_inv=[RxyzT,-RxyzT*OP];
Trn_OP_inv(4,1:4)=[0,0,0,1];
```

3 步态仿真

3.1 棍棒模型运动仿真

关于仿真步态实现的程序文件如下表所示。

表 3 棍棒模型运动仿真程序表

程序	功能
Hex_Tra_rotate.m	六足机器人绕 z 轴旋转
Hex_Tra_Trajectory.m	计算六足机器人绕 z 轴旋转的轨迹
Hex_gait_movie.m	展示六足机器人三角步态
MainCode_HexGait_Move_Forward.m	六足机器人向前运动
MainCode_HexGait_Move_Right.m	六足机器人向右平移
MainCode_HexGait_Move_climbing.m	六足机器人爬台阶

通过上述程序，我们实现了如下几种姿态的仿真：向前直行、旋转、向右平移和爬台阶。

(1) 向前直行：MainCode_HexGait_Move_Forward.m 程序实现了六足机器人以三角步态向前运动的功能。在 Hex_InvKin.m 程序中，我们知道，每一条腿的腿尖位置是保持不变的。也就是说，六足机器人实际上是始终处在原地的。那么让机器人运动起来，实际上就是要改变六足机器人腿尖的落点位置，这样就可以使得六足机器人的位置发生变化，从而导致运动的产生。与 Hex_InvKin.m 程序代码的不同之处主要有如下两点：

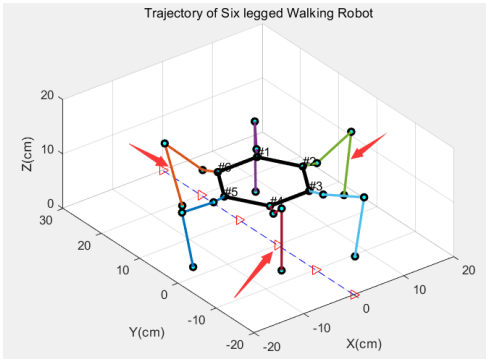
初始化变量增加。我们需要增加六足机器人沿 x、y 方向的速度，确定每一步需要花的时间以及机器人一共要走多少步。

```
Vy=0.2; %cm/sec
Vx=0; %cm/sec
delta_t=2.5;% sec for each step
steps=10;  % 完成一个delta_t时，由10张图片展现，也就是10桢
StepFwd=3; % 所有腿完成一次运动称为一个大循环，那么总共进行3个大循环
Nstep=6*StepFwd; %由于是整体往前走三个大循环，每个循环里面6条腿都要动，所以一个大循环需要6步
```

腿尖落点位置的改变。由于每次只移动三条腿，因此每一步都要考虑某三条腿足部的落点位置。也就是说，足部在每一步的时候的 X、Y、Z 坐标都会发生变化。根据腿尖落点位置的不同，我们可以运用逆运动学公式推算出不同情况下的六足机器人的姿态。其代码如下，仿真结果如图 9。

```
while Nstep>=1
q=0;
LQ=Ltip; % 每个脚的落点位置
qStep=TotStp-Nstep*steps; % 计算出来的值当前对应的脚
while q<steps
    qq=qStep+q; % 获取当前的动画对应的帧数
    time=delta_t*qq/steps; % 计算当前的时间是多少
    % this shows the current center position about main body
    OPPy=Vy*time;
    OPPx=Vx*time;
    OPPz=7;
    % OPPz=10;
    LVy=4*Vy;
    % LVy=4*Vy;
    LVx=2*Vx;
    w=pi/delta_t; % delta_t是完成一小步的时间
    StpTime=delta_t*q/steps; % 在这一步中，第q桢对应的时间
    time_array(k)=time; % 将当前帧的时间存入time_array中

DeltaLtipX=LVx*delta_t*.5*(1-cos(w*StpTime)); % 1-cos(w*StpTime) 1-cos(w*StpTime) 从0-2变化，乘以0.5就是0-1了
    DeltaLtipY=LVy*delta_t*.5*(1-cos(w*StpTime)); % LVy*delta_t 沿y方向运动的变化距离
    DeltaLtipZ=1.5*(1-cos(2*w*StpTime)); % 沿z方向运动的变化距离

DeltaLtip=[DeltaLtipX;DeltaLtipY;DeltaLtipZ];
    % Nstep值为18,取6同余就是计算运动哪个腿
if mod(Nstep,3)==0

  Ltip(1:3,1)=LQ(1:3,1)+DeltaLtip;
  Ltip(1:3,3)=LQ(1:3,3);
  Ltip(1:3,5)=LQ(1:3,5);

Ltip(1:3,2)=LQ(1:3,2);%+[2*Vx*time;2*Vy*time;3*sin(w*time)];

Ltip(1:3,4)=LQ(1:3,4);%+[2*Vx*time;2*Vy*time;3*sin(w*time)];

Ltip(1:3,6)=LQ(1:3,6)+DeltaLtip;%+[2*Vx*t
```

```
ime;2*Vy*time;3*sin(w*time)];
% 这里仅展示一种同余情况，其他同余情况详细请见代码文件
end
yaw=0;% Around Z
roll=0;% Around Y
pitch=0;% Around X
OP=[OPPx;OPPy;OPPz]; % 当前中心点的位置
OPP=OP;
OP_array(OpIn,1:3)=OP; % 记录每一帧中心的位置
OpIn=OpIn+1;
q=q+1;
k=k+1;
end
Nstep=Nstep-1;
end
```



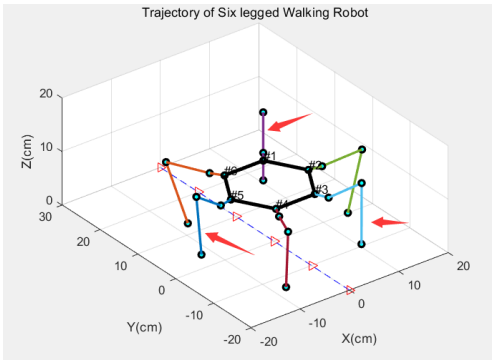
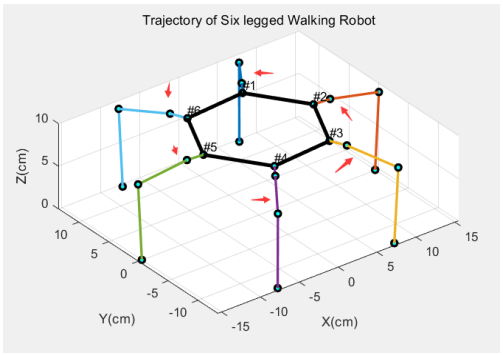


图 9　棍棒模型三角步态

(2) 旋转：Hex_Tra_rotate.m 程序主要用于展示六足机器人绕 z 轴旋转，这是一种基于机器人重心位置不动的旋转。

程序会先调用 Hex_Tra_Trajectory.m 函数，来计算六足机器人绕 z 轴旋转的轨迹。其实这个程序和 Hex_InvKin.m 程序非常相似。唯一的不同点在于配置的时候我们不再是给定一个 OP 和 yaw、roll、pitch 的值，而是连续给出一系列的值，例如假设有 20 个 steps，我们会给出 20 个连续的值。如果让机器人只绕 z 轴转动，那么我们仅仅需要改变 yaw 的值，根据不同的 yaw 值我们可以分别计算出对应关节的角度和位置坐标，从而实现机器人的旋转。仿真结果如图 10 所示。

```
%% TRAJECTORY:
steps=20;
OPPz=10*ones(1,steps);
OPPx=zeros(1,steps);
OPPy=zeros(1,steps);
YAW=zeros(1,steps);% around Z
ROLL=zeros(1,steps);% around Y
PITCH=zeros(1,steps);% around X
LTIP=[Q2,Q2,Q2,Q2,Q2];
for q=1:steps
    YAW(q)=(9/200)*(q*2*pi/steps);
end
OPP=[OPPx;OPPy;OPPz];
k=1;

while k<=steps
% 这里省略其他的代码
k=k+1;
end
```



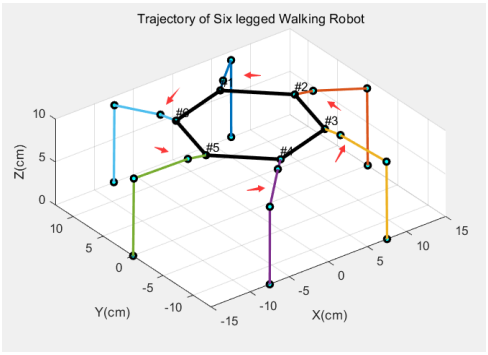
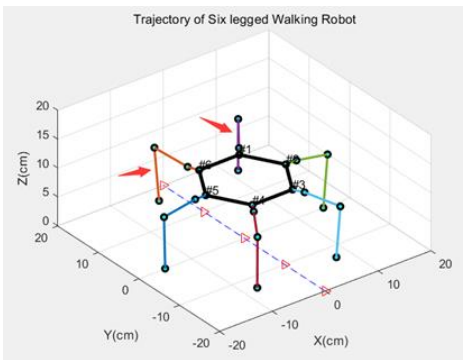


图 10　棍棒模型旋转动作

(3) 向右平移：向右平移的逻辑和向前行走的逻辑本质上基本一致，唯一的变化点在于控制腿的移动次序和行进的方向，故不在此重复放代码，具体代码请见 MainCode_HexGait_Move_Right.m 程序文件。棍棒模型向右平移的步态如下图所示。



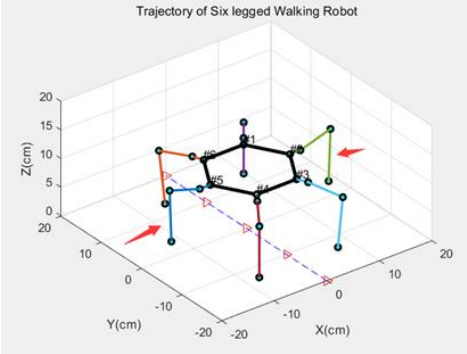
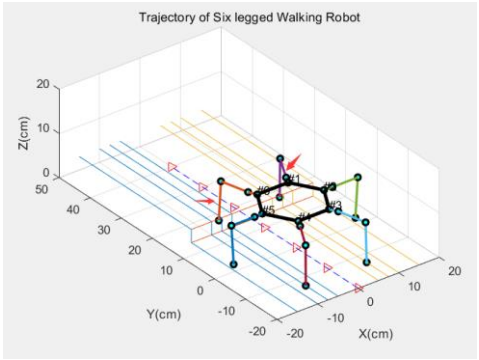


图 11　棍棒模型向右平移

(4) 爬台阶：爬台阶的本质逻辑与直行的逻辑也是类似的：直行时，每个腿完成一次移动后，其足部在 Z 轴方向上的距离仍然为 0，也就是最终仍然与地面接触。我们设台阶高度为 h，那么让机器人爬上一级台阶，其实就是让机器人足部在完成动作之后的 z 值等于 h，并保持不变（z 值不变），然后继续向前行走。下面展示了让足尖 z 值随时间变化的部分代码和效果图。

```
if qStep==0
    DeltaLtipZ=1*(1-cos(w*StpTime));
end
if qStep==10
    DeltaLtipZ=1*(1-cos(2*w*StpTime));
end
if qStep==20
    DeltaLtipZ=1*(1-cos(2*w*StpTime));% %
第一个大循环
end
if qStep==30
    DeltaLtipZ=1*(1-cos(2*w*StpTime));
end
if qStep==40
    DeltaLtipZ=1*(1-cos(2*w*StpTime));
end
if qStep==50
    DeltaLtipZ=1*(1-cos(2*w*StpTime));% 第
二个大循环
end
if qStep==60
    DeltaLtipZ=1*(1-cos(2*w*StpTime));
end
if qStep==70
    DeltaLtipZ=1*(1-cos(2*w*StpTime));
end
if qStep==80
     DeltaLtipZ=1*(1-cos(2*w*StpTime));%
第三个大循环
end
if qStep==90
    DeltaLtipZ=1*(1-cos(2*w*StpTime));
end
if qStep==100
    DeltaLtipZ=1*(1-cos(2*w*StpTime));
end

...

% 前腿爬上台阶
```



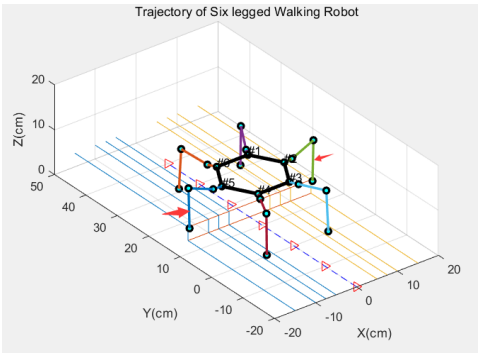


图 12　棍棒模型爬台阶

3.2　SIMULINK-URDF 模型运动仿真

在 SIMULINK 中为关节角设置输入函数，使模型运动。根据资料[6]，机器人关节角输入源函数可以先设置为正弦函数，再在正弦函数之后连接 MATLAB 函数代码块进行调整。由于末端轨迹保持在平面上，z 方向不能为负数，因此用绝对值 abs(u) 进行处理（u 为输入函数）。处理后，y 方向可以得到较为规律的抬腿触地轨迹。根据各条腿与身体坐标系的相对位置，调节输入函数的振幅和偏置，使其在合理的象限运动。运用前面推导的逆运动学公式得到关节角度的旋转值。该部分代码如下。

```
function [s1,s2,s3]= fcn(u)
l1=0.53;
l2=0.83;
l3=0.89;
z =0.3*abs(u)+0.1;
x3=0.5*u+0.1;y3=sin(u)+0.6;l0=X3;
s1= arctan(y3/x3);
q1= arctan(z/(l1-lc));
l= sqrt(z^2+(l0-l1)^2);
q2= arccos(sqrt(l2^2+l^2+l3^2)/(2*l2*l));
s2=q1+q2;
fi =
arccos(sqrt(l2^2+l3^2+l^2)/(2*l2*l3));
s3=fi-180;
```

依据棍棒模型推导和三角步态仿真原理，可知机器人足部的运动方向和起落顺序按组变化。由于不同组开始运动的时间不同，可以给后发生变化的腿添加延时模块。参考棍棒仿真公式原理，修改 SIMULINK 中的输出模型，并与模型的力矩输入连接，如图 13。

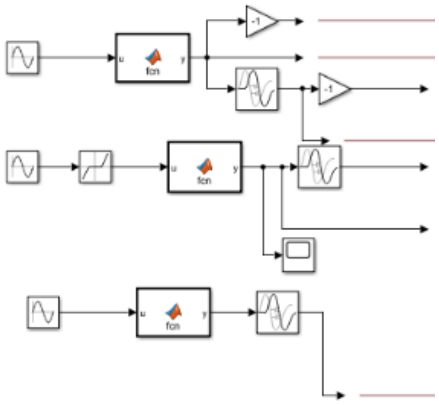
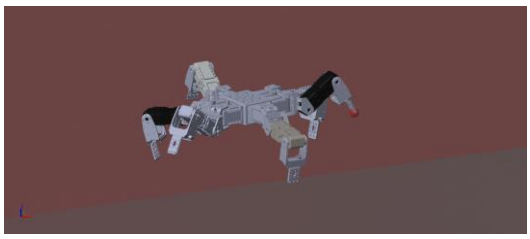


图 13　关节角输出树

完成上述配置之后，在 SIMULINK 中成功仿真了向前直行、原地旋转和跳舞的步态。

(1) 旋转：在原地旋转动作中，同组腿运动方向一致，仿真效果如下图所示。



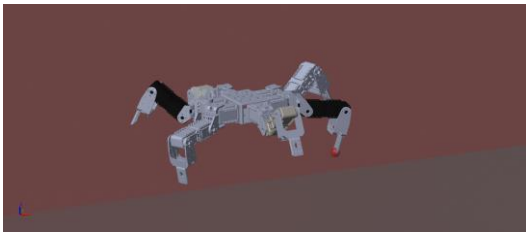


图 14　SIMULINK 旋转仿真

(2) 向前直行：在直行动作中，各组有一条腿的髋关节的运动方向与原地旋转相反，因此需要在树图中给反向变化的腿添加负向增益模块。保持腿的运动分组和开始运动顺序不变，调节输入幅值和增益使得每一步距离相同，并且保持机器人的身体在运动过程中不发生旋转，得到直行效果如图 15。

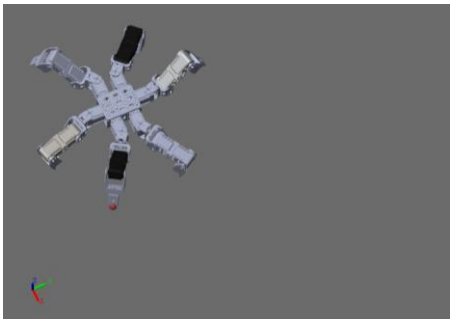


图 15　SIMULINK 直行仿真

(3) 跳舞：在跳舞动作中，每条腿单独为一组，因此分别给每条腿不同的源函数和延时函数，构建输出树如图 16。

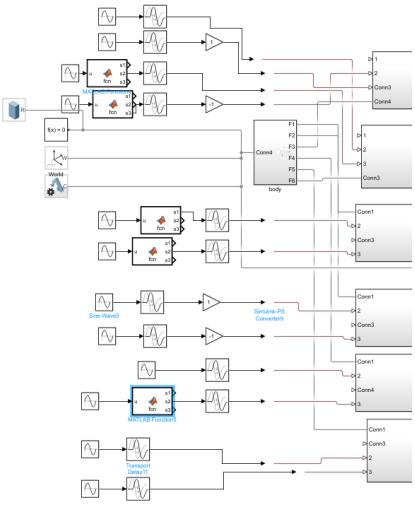
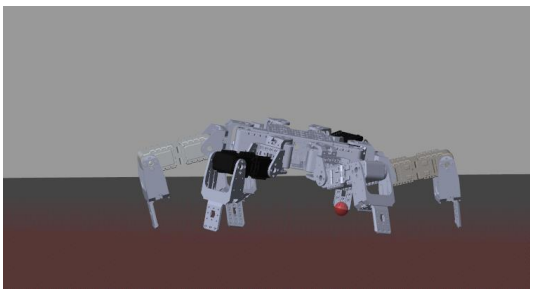


图 16　跳舞动作输出树

六条腿膝关节依次抵达最低点构成完整循环，因此每条腿之间延迟角为 1/6 个源函数周期。得到动作效果如图 17。



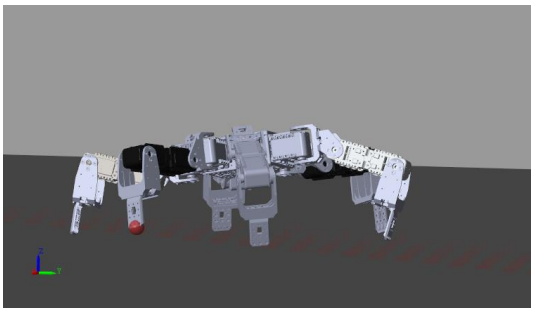


图 17　SIMULINK 跳舞仿真

4 硬件实现——基础动作

借用 ROBOTIS BIOLOID Premium 套件，根据官方教程[8]搭建六足机器人实体。本章介绍在硬件层面，六足机器人实现的基础功能。

4.1 舵机控制原理

Dynamixel AX-12 舵机在位置控制模式下，根据可变脉宽的脉冲进行伺服控制。工作空间为$300°$，12V 供电下堵转扭矩为$1.5N \cdot m$，空载下转速为$59rev \cdot min$。[9]

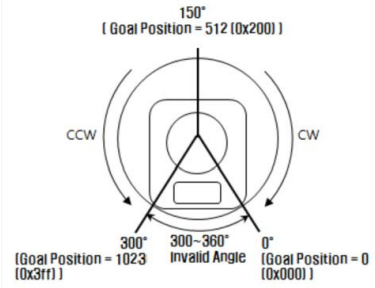


图 18 Dynamixel AX-12 舵机前视图

舵机控制通过 TTL 线串联实现总线控制，各个舵机通过不同 ID 进行区别。上位机控制配合 Dynamixel SDK，通过设置各个变量地址实现与舵机的通信并对其输出进行控制。[10]

通过 RoboPlus 平台进行动作 motion 以及任务 task 的编写。如图 19 所示，根据舵机位置的编码，输入对应的值即可控制舵机到达指定位置。根据 AX-12 舵机从 0 到 300 度对应值十进制编码值 0 到 1023 可知，角度值与编码值的换算为 θ=300/1023α。其中复位值即正中编码值为 512，顺时针运动值减少，逆时针运动值增大。通过增加 step 可以在一个 motion 中实现多次连续运动以及反复运动。通过编辑左侧步骤动作一栏为各个 ID 的舵机赋值，通过右向箭头即可赋值到右侧机器人的姿势一栏实现姿态。同时可以修改停止时间，执行时间，循环次数，执行速度以及调节惯性力等选项改变动作的执行周期。

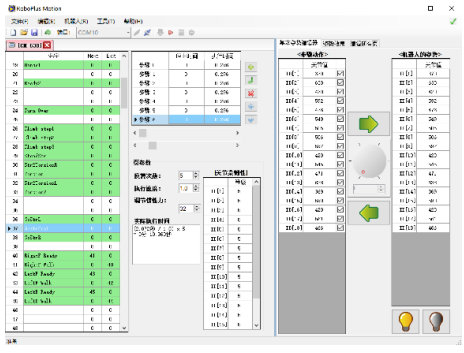


图 19 RoboPlus 动作编辑界面

将模型设置为与官方模型相同，可以通过姿态效果选项卡查看当前模型的大致姿态。编辑完成后保存 motion 在控制器中，点击执行动作按钮即可操纵整体机器人实现指定动作。

4.2 前进步态优化

前进步态以及之后的动作设计所采用的关节 ID 编号均与官方设定的各个关节 ID 一致。如图 20 所示，3，4 号电机一侧为前侧，六足机器人每个足由三个关节组成，分别为与身体连接实现左右摆动的肩关节，以及两个腿关节实现上下摆动。其实，官方提供了一个六足机器人前进的步态，但是经过观察，我们小组认为官方提供的步态连贯性不强、动作幅度较大。因此，我们重新对六足机器人前进的三角步态进行了优化设计。

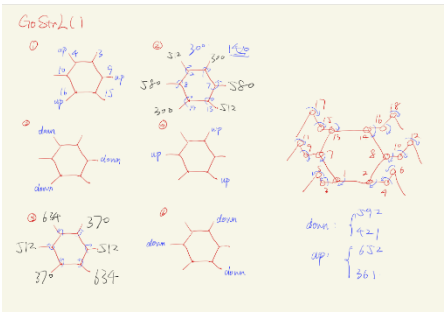


图 20 前进步态关节值推导

根据仿真设计，前进通过三角步态实现，即组成三角的三足统一移动，与另一个三角交替移动实现前进。具体执行中三足抬起并向前摆动，另外三足向后摆动，而后抬起三足落下同时另三足抬起并相似的向前摆动，以此循环实现前进，效果如下图。

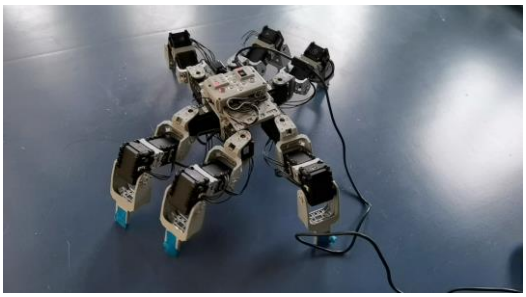
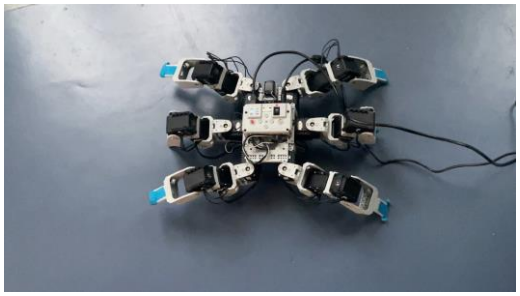


图 21 六足机器人以三角步态前进

4.3 左右平移

与前进类似，左右平移同样采取三角步态，依次抬起放下三足并向一侧步进以实现单侧平动。



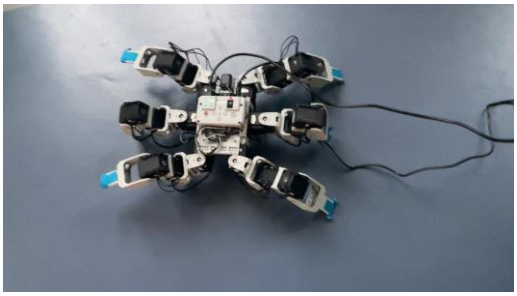
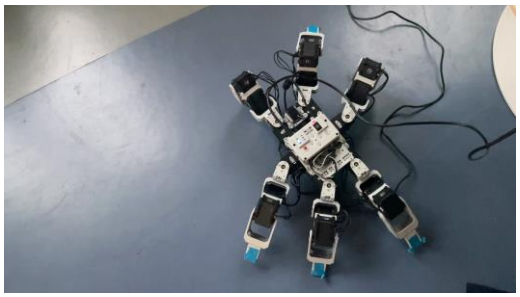


图 22　六足机器人以三角步态左右平移

4.4　斜向步进

在前进和左右平移的基础上，让机器人以三角步态向左前、右前方移动。



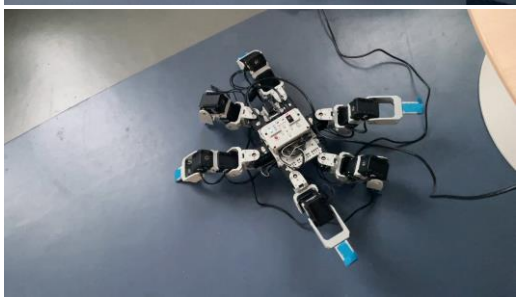


图 23　六足机器人向斜前方前进

4.5　原地旋转

根据仿真步态设计，基于三角步态，三足抬起同时肩关节向同侧摆动，下一步对应另外三足抬起并继续向同侧摆动，即可实现原地旋转。通过这样的旋转，六足机器人可以在原地调整前进方向。

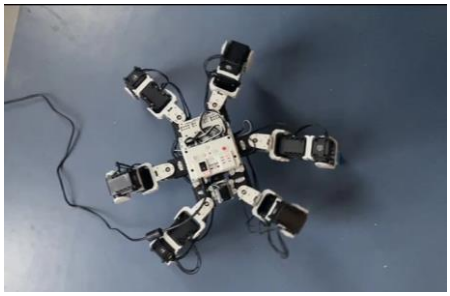


图 24　六足机器人旋转

4.6　夹取物体

在动作设计时我们发现，有时机器人靠四足也能保持平衡，此时前方的两足是冗余的。所以如图 25 所示将姿态调整为四足，余下的两足可以通过肩关节的摆动实现对部分轻小物体的夹取，设计思路类似龙虾、螃蟹等动物的猎食方式。

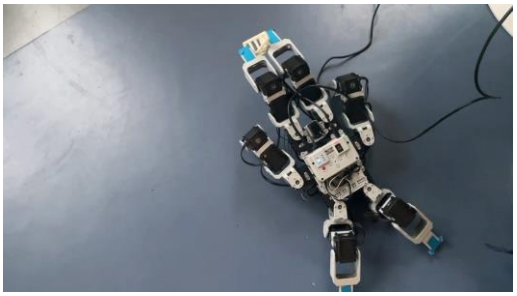


图 25　机器人通过前两足夹取物体

5　硬件实现——复杂动作

通过对基础动作进行组合、调整，我们小组实现了一系列动作复杂的机器人功能。本章介绍部分让六足机器人完成的复杂动作。

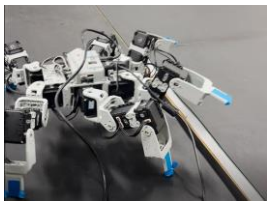
5.1　爬台阶

爬台阶的动作设计采用对称腿抬起，然后通过支撑的四足将抬起的腿逐渐移动到台阶平面上放之后，落下进行支撑的方法，依次将前端双足，中端双足以及末端双足移动到台阶上放最终实现爬台阶的动作。

在六足机器人本身的步态设计上，如果把三角步态的髋关节抬腿高度设高，可以在前进过程中跨越一定的障碍，自然也可以跨上台阶，但存在一个明显的问题就是采用三角步态跨越的过程中，会出现大量的腿部连杆和台阶垂直面的碰撞，这不仅会影响三角步态前进的稳定性（前进方向可能会发生偏移），还会对机器人的连杆、关节等造成摩擦损耗和冲击磨损。因此，需要一种更加完善的爬台阶的步态设计。

由于六足机器人的机身比较长，需要多次移动机身才能实现抬起的两足能够平稳的落到台阶之上，同时要保证四足支撑时机器人不会放生重心不稳倒下的情况，而且这个过程是无法循环的，最后通过六组，共计 42 个 step 实现了爬台阶的动作。

首先抬起前端双足，将后端双足各自向前移动移动距离，然后略向后滑动中端双足和大幅度向后滑动后端双足来将前肢移置台阶之上。



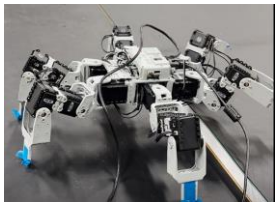


图 26.1　爬台阶步态 1

然后落下前肢，抬起中端双足，通过前肢和后肢进行支撑，重复两次类似于第一步的移动方法，将中端双足转移到台阶之上。

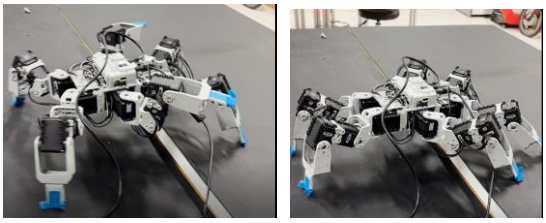


图 26.1　爬台阶步态 2

当中端两足落到台阶上之后，可以发现，此时后端双足距离台阶有较长的一段距离，机身的大部分也并没有移动到台阶之上，此时不能够直接抬起后端双足，一是机器人会重心偏移倒下，二是后端双足会承受很大的力矩，舵机会启动自我保护机制，导致断电。我们需要通过此时的六足支撑，再次采用两次前面提到的足端滑动的方法来使机器人向前移动一定的距离，然后当机身和六足的位置合适的时候，使用一次三角步态大幅度移动一次机身距离，使后端双足能够直接跨到台阶之上。

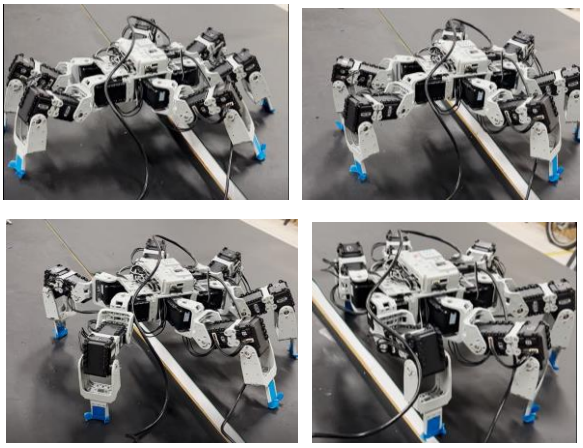
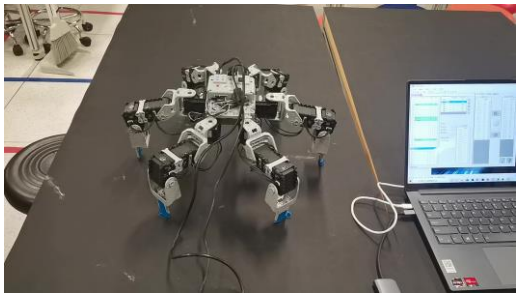


图 26.3　爬台阶步态 3

5.2　跳舞

跳舞姿态设计采取保持六足落点不变，通过膝关节和踝关节摆动，实现中心平面的波动并展现一种原地波浪式跳舞的姿态。

根据仿真步态模型，通过腿部关节的弯曲实现中心平面的倾斜，然后依次将由 3 个舵机组成的腿部姿态向相邻腿赋值，执行 6 次即可实现机器人的波浪跳舞。图 27 展示了机器人的跳舞动作。



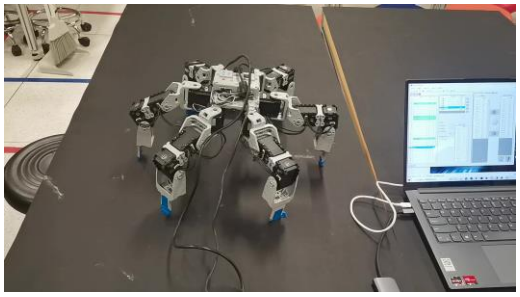


图 27　六足机器人在原地跳舞

5.3　自主翻身

在自然界中，昆虫（如蜘蛛）在完全翻倒之后，是很难完成自主翻身的，大多数情况下都是借助地形或者通过弹跳翻身。但是本项目研究的六足机器人，无法进行弹跳，那么如何让六足机器人在平面地形上不借助任何工具地完成翻身就成为了一个兼具挑战性和趣味性的任务。

经过实验我们发现：通过机器人前部和中部的四条腿，逐渐将六足机器人撑起；在达到垂直角度后，后端的两条腿调整到合适的关节角度，就可以实现翻身。在设计翻身动作的过程中，我们同时注意所有足端的触地角度，尽可能减少冲击载荷对关节处的冲击。翻身过程如下图所示。

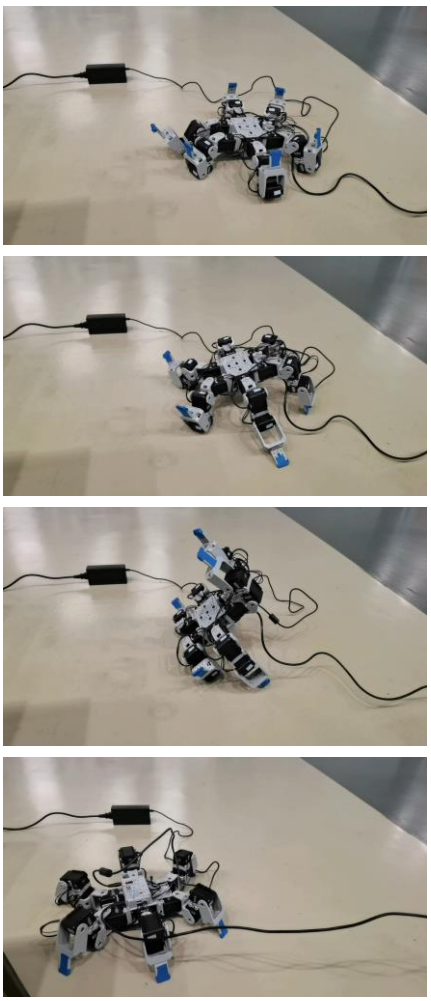


图 28　六足机器人翻身

6 硬件实现——自动控制

基于 RoboPlus 平台，对六足机器人编程，使机器人可以根据接收到的不同指令和环境信息，执行不同的动作。

6.1 前进和转向

按控制器或遥控器上的 U 键，控制六足机器人以三角步态向前移动；按 L 键，控制机器人向左转向；按 R 键，控制机器人向右转向。由于动作均采用前几章介绍的步态，所以不在这里展示，仅展示部分关键代码，如下图。

```
10:         如果 ( 按键 == U )
11:         {
12:             调用 MoveForward
13:             时钟 = 32.000秒
14:
15: CheckIR :
16:             如果 ( 按键 == U )
17:                 调用 MoveForward
77:         如果 ( 按键 == L )
78:         {
79:             调用 Stop
80:             动作编号 = 8
81:             调用 WaitMotionComplete
82:         }
83:         如果 ( 按键 == R )
84:         {
85:             调用 Stop
86:             动作编号 = 6
87:             调用 WaitMotionComplete
88:         }
```

图 29　控制前进和转向的代码

6.2 自动避障

借助套件中的红外传感器，机器人可以感知外部环境信息。在向前运动的过程中，机器人如果感知到前方存在物体，则会先退后一定距离，再旋转以寻找合适的前进方向；重复这个过程，直到能够正常前进。

```
 99: 函数 ObjectReaction
100: {
101:     调用 Stop
102:     动作编号 = 60
103:     调用 WaitMotionComplete
104: }
```

图 30　避障的部分代码

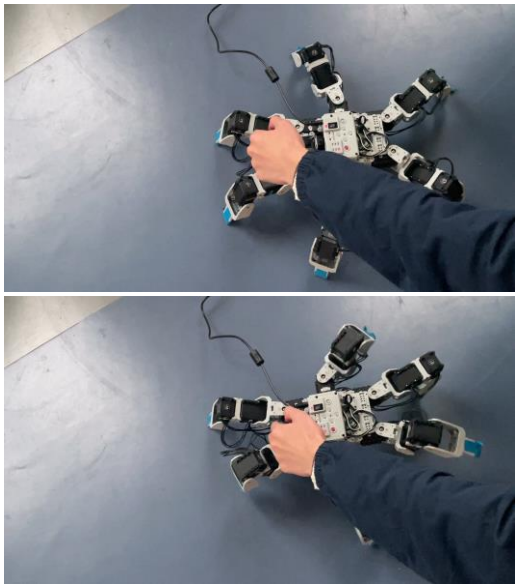


图 31　避障演示

6.3 危险感知和攻击

同样借助红外传感器，机器人感知到上方存在物体（即危险）时，会蜷缩身体；蜷缩状态下，又感知到前方存在物体，则会进行攻击。

按控制器或遥控器上的 D 键，可以强制机器人退出危险感知状态，继续运动。

```
23:                 如果 ( PORT[3]:红外线传感器 >= 100 )
24:                     调用 Fear
25:                 如果 ( PORT[4]:红外线传感器 >= 20 )
26:                 {
27:                     调用 Attack
28:                     无条件循环
29:                     {
30:                         如果 ( PORT[4]:红外线传感器 >= :
32:                             调用 WaitMotionComplete
33:                             动作编号 = 80
34:                             调用 WaitMotionComplete
35:                         }
36:                         如果 ( PORT[3]:红外线传感器 >=
   100 )
37:                         {
38:                             调用 WaitMotionComplete
39:                             动作编号 = 12
40:                             调用 WaitMotionComplete
41:                         }
42:                         如果 ( 按键 == D )
43:                             终止循环
```

图 32　感知危险和攻击的代码

6.4 休眠模式

在给定时间内，如果机器人没有收到任何指令或者环境信息，则会进入休眠；借助套件中的声音传感器，可以实现在机器人上方拍手或发出其他声音使其退出休眠状态。相关代码和演示如图 33 和图 34。

```
58: Sleep :
59:             调用 Stop
60:             动作编号 = 10
61:             调用 WaitMotionComplete
62:             最终声音识别次数 = 0
```

```
63: SleepAgain :
64:         如果 ( 最终声音识别次数 > 0 )
65:         {
66:             动作编号 = 1
67:             调用 WaitMotionComplete
68:             时钟 = 32.000秒
69:             跳跃 CheckIR
```

图 33　控制休眠模式的代码

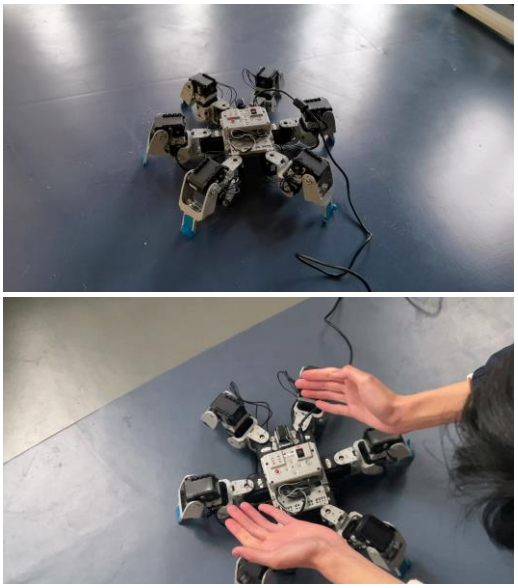


图 34　休眠状态和拍手使其退出休眠

7　系统参数对功能的影响

本项目中对仿真和实体机器人动作具有影响的参数众多，经过分析、比较，我们认为动作执行时间 t_e 和指令延迟时间 t_d 两个参数较有代表性。

(1) 动作执行时间 t_e：指控制机器人完成任务时，为执行每个动作所预设的时间。

通常，如果 t_e 设置得太大，每个动作之间间隔的时间会过长，导致动作不连贯；而且外部环境在动作间隔中变化的可能性也会增加，可能会使机器人无法完成预期任务。

如果 t_e 设置得太小，则可能出现机器人无法在预设时间内完成动作，就有可能发生动作指令之间的冲突；而且如果动作执行得过快，舵机负荷会加大，可能会对舵机造成损害，甚至使舵机控制失灵。

(2) 指令延迟时间 t_d：指对机器人编程时，在不同任务指令之间预设的时间间隔。设置延迟时间是必要的，如果没有延迟时间，那么控制器会在上一个任务执行过程中发送下一个任务的指令，而之后发送的指令通常并不会打断前面指令的执行，就会发生后面发送的指令没有执行的情况。

如果 t_d 设置得太大，那么同样会有每个动作之间不连贯、无法适应快速变化的环境的问题；此外，在编程中，我们通常会给控制器设置总运行时间，如果 t_d 过大，那么机器人来不及执行所有任务，就会停止运行。

如果 t_d 设置得太小，那么可能相当于没有设置延迟时间，后面任务的指令还是会在上一个任务执行完成之前发送。

因此需要设定合适的 t_d，程序才能正常控制机器人。

8　合作分工与组员贡献情况

(1) 田丰（22%）：几乎独立完成六足机器人的硬件搭建，并解决部件存在的问题；完成六足机器人左右平动、斜向步进；完成机器人翻身动作；完成自动控制部分代码的编写和调试，调试遥控功能（最终未实现）。

(2) 沈浩瑜（20%）：配合六足机器人硬件调试；优化机器人直线前进、扭转、跳舞等运动；完成基于 Dynamixel SDK 的 C 和 python 代码控制模块（最终版本未应用）。

(3) 苏汉祺（20%）：完成六足机器人部分运动学分析；使用 MATLAB 构建相应模型与运动学方程；实现向前直行（多种步态方式）、旋转、向右平移、爬台阶等动作仿真。

(4) 杨博文（20%）：独立完成实体机器人爬台阶动作；合作完成六足机器人左右平移、斜向步进等操作；合作完成 RoboPlus 平台的调试、机器舵机调试等。

(5) 刘迪涵（18%）：推导六足机器人部分逆运动学公式；完成机器人 URDF 模型仿真；配置 MATLAB SIMULINK 模块，完成向前直行、转弯及原地旋转的动作仿真。

9　研究总结

(1) 在本项目中，我们小组建立了六足机器人模型，完成了基础正、逆运动学方程的建立和求解，并在此基础上计算得到前进、原地旋转等基本步态。

(2) 我们在 MATLAB 中进一步求解逆运动学方程，完成了六足机器人 SIMULINK-URDF 模型和棍棒模型的仿真，在两种模型上实现了六足机器人前进、旋转、爬台阶等步态的仿真。

(3) 我们用 ROBOTIS BIOLOID Premium 套件搭建了六足机器人的实体模型。完成了前进、斜向运动、旋转、夹取物体等基本动作，爬台阶、跳舞、自主翻身（不借助外部工具）等复杂动作，以及通过编程使机器人可以对外界信号做出反应。

(4) 本项目的研究结果对其他同学之后设计六足机器人步态、在 MATLAB 中仿真六足机器人模型或用 ROBOTIS BIOLOID Premium 套件搭建六足机器

人并完成相似任务，有较强的借鉴和指导作用。

参 考 文 献

[1] 张秀丽,郑浩峻,陈恳,段广洪.(2002).机器人仿生学研究综述. 机器人(02),188-192.

[2] 王晓芸,崔培 & 陈晓.(2019).轮式移动机器人文献综述. 石家庄铁路职业技术学院学报(02),66-70. doi:

[3] 周小淞,张亚,连云飞 & 刘廷.(2015).履带式机器人运动平台控制系统的设计. 机电技术(06),47-50. doi:

[4] 陈甫.(2009).六足仿生机器人的研制及其运动规划研究(博士学位论文,哈尔滨工业大学).

[5] 李胜铭,朱碧珂,王秉奇 & 吴振宇.(2019).基于 D-H 矩阵的六足机器人运动仿真与分析. 物联网技术(12),78-82. doi:10.16667/j.issn.2095-1302.2019.12.021.

[6] Mobile-Robotics. https://github.com/dallyria/Mobile-RoboticS

[7] Beaber, SI, Zaghloul, AS, Kamel, MA, & Hussein, WM. "Dynamic Modeling and Control of the Hexapod Robot Using Matlab SimMechanics." Proceedings of the ASME 2018 International Mechanical Engineering Congress and Exposition. Volume 4A: Dynamics, Vibration, and Control. Pittsburgh, Pennsylvania, USA. November 9–15, 2018. V04AT06A036. ASME.

[8] Robotis Bioloid Premium Manual. https://emanual.robotis.com/docs/en/edu/bioloid/premium/

[9] Dynamixel. Dynamixel AX-12A Introduction. https://emanual.robotis.com/docs/en/dxl/ax/ax-12a/

[10] Dynamixel.Dynamixel SDK. https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/overview/#supported-languages