摘　要

六足仿生机器人（HexapodRobot），又叫蜘蛛机器人，是足式机器人的一种。因其特殊的结构和运动方式，让六足仿生机器人在复杂地形环境中依旧有着良好的平稳性。在各个领域中有着广泛的应用前景。

本文的主要内容是对六足仿生机器人的运动学分析，解析出其运动学方程。并就运动学方程对六足仿生机器人的步行足运动空间进行分析。再使用 Matlab 的机器人工具箱（Robotics Toolbox）和遗传算法进行步行足的运动轨迹规划。最终在 Adams 中进行整体的运动仿真分析，以检验机器人的结构以及运动的实际效果。且对六足仿生机器的防滑和测距这两项辅助结构进行设计。

关键词：足式机器人；运动学仿真；辅助结构设计

Design of hexapod bionic robot based on stm32

—Auxiliary structural design and dynamic simulation analysis

Abstract

A Hexapod Robot, also known as a spider Robot, is a type of foot Robot.Due to its special structure and movement mode, the hexapod bionic robot still has good stability in the complex terrain environment.It has wide application prospect in various fields.

The main content of this paper is to analyze the kinematics of the hexapod bionic robot and get its kinematic equation.The kinematic equation is used to analyze the walking space of hexapod bionic robot.Then, Robotics Toolbox and genetic algorithm based on Matlab software were used to plan the trajectory of walking feet.Finally, the overall motion simulation analysis was carried out in Adams to verify the structure of the robot and the actual effect of motion.The two auxiliary structures of the six - legged bionic machine are designed.

Keywords: foot robot; kinematics simulation; auxiliary structure design

目录

1 绪论

1.1 六足仿生机器人研究背景及研究意义

近年来机器人的热度不断的提高，吸引着各行各业的关注。其背后显现出的是人们对先进生产力的需求，随着工业化的推进，各种各样的机器人加入我们的生产生活之中，给我们带来了便利。

这些种类繁多的机器人，按照运动方式可以将其划分为轮式、履带式以及足式三类。其中以轮式、履带式为运动方式的机器人对运行环境有一定要求，像是野外这样的复杂地形环境中就不便开展作业。

而足式机器人在此类环境中有着良好的适应能力，其中的一个原因可以归结于设计足类机器人时我们模仿了自然界中一些已有的结构来进行仿生设计，这些结构在自然界的残酷筛选中已经验证了其可行性，此时借用于机器人上自然是一个不错的方案。正是这样的优势让足式机器人在复杂环境下有其他两类机器人不具备的强适应性。让足式机器人可以代替人类进行危险、复杂的任务。

仿生足式机器人可以应用于在复杂地形中的物资运输、巡逻警戒，还可用于军事侦查、地形勘探、玩具娱乐、智能教育等领域，有着广阔的发展应用前景。

1.2 仿生机器人的意义

“仿生学”这一词于1960年在第一届仿生学会议上被提了出来，并通过长久的发展，仿生学与各个学科融合发展成为了一门新兴的学科。仿生学顾名思义就是模仿自然界中已存在的生物进行设计的一门学问。其中机械可以说是与仿生学最早融合的，早先人们通过模仿自然界中的各式结构来制造他们的工具以提高生产效率。

除了简单的模仿，仿生设计还可以给我们许多解决问题的方案。当我们为某个方案而苦苦思索时，说不定在自然界中早已有一种方案已经通过千百年的考验和完善只待我们发现。这种自然界给出的方案无疑大大减轻了人们的设计负担，我们只需要分析学习它即可得到一个完美天成的设计。通过仿生设计所设计出来的机器人，大多都有其独到之处，或是高精度，或是高可靠性，或是高灵活性等等。

综合来说仿生机器人是我们人类对大自然的一种模仿学习，是将自然选择所筛选出来的优秀“设计”融如自身的过程。在六足仿生机器人中，我们就是仿照了自然界中一些昆虫的结构和运动原理。像是六足结构，我们便是仿照自然界中“六足纲”昆虫的结构设计。这些昆虫一般有三对足，对应六足机器人的前、中、后三对足。每个足上我们都仿照昆虫足上的基节、转节、腿节、胫节、跗节和前跗节的结构设计机器人的各个关节，使得六足机器人对环境的适应能力优于其他机器人。在行动步态方面也仿照了昆虫的三角步态，我们将六足分为两组，一为支持足，一为步行足。在行进过程中两组足

部相互交替，每次行进都是三足悬空、三足接地，以三角形支架结构交替前行。让重心保持在三角形中，使得六足机器人在行进中有着良好的稳定性。

1.3 国内外关于六足仿生机器人的研究状况

随着人类科学技术的发展，人们对未知的探索越发热切，探索范围越来越大、越来越深。期间充满危机，科技的快速发展也难以护得探索者周全。这时代替人类担任探索任务的机器人应运而生。

为了探索外星地貌，人类需要一种适应复杂地形环境的机器人。就这一目标美国于1980 年起便开始了对六足机器人的研究，后来推出了 Genghis、Hannibal（图 1、图 2）等六足机器人。

图 1 Hannibal

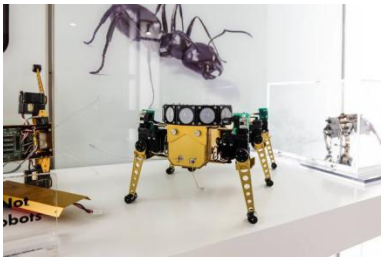


图 2 Genghis

1995 年前后 Rockwell 公司还研发了用以河道排雷的六足机器人 ALUV（图 3），在设计时仿照了螃蟹的结构，且该机器人还可水陆两用，显示出了六足机器人对环境适应性强的特点。

到了二十一世纪 NASA 因为勘探的需求，加大了在六足机器人的研发。作为收获，NASA 在 2002 年和 2005 年前后推出微型仿生蜘蛛机器人和 Lemur 系列（图 4）。其中 Lemur 系列已经可以应用于外太空的勘探和协助宇航员进行空间站的维修建设任务。

图 3 ALUV

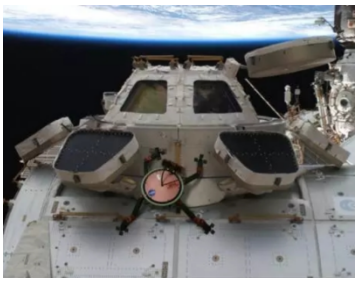


图 4 Lemur

国内的六足机器人的研究大概在上世纪八十年代末，由清华大学、燕山大学、北京航空航天大学、上海交通大学、哈尔滨工程大学、沈阳自动化研究所、哈尔滨工业大学等大学机构进行了有关的研究。中国科学院沈阳自动化研究所与光机所于 1989 年 3 月共同研制了海蟹号六足步行机器人。华中科技大学研制的“4+2”步行机器人。[1]还有

Vincross人工智能科技公司于2017年研发的HEXA（图5），它的头部可360°转动，可朝任意方向前进，可以1.2km/h的速度行进，可攀爬15cm高的障碍物，可适应各种复杂地形，可通过手机进行可视化操作，可进行二次编程开发等等功能。

图5 HEXA

1.4 本文主要研究内容

本文主要内容是六足仿生机器人的仿真分析以及辅助结构的设计。对于六足仿生机器人的仿真分析将会从其运动学的数学模型入手，主要分析步行足在行进过程中足端的位移情况。并且分析足端的可达工作空间，为步行足的轨迹规划提供参考。

建立步行足的数学模型后再利用Matlab中的机器人工具箱（Robotics Toolbox）求解步行足的运动学正逆解，并且利用工具箱对步行足进行初步的轨迹规划。后通过遗传算法进一步得到更加精确、合理的最优时间解，以完成最终的轨迹规划。

完成上述工作后就在Adams中对机器人整体进行仿真分析，用以验证机器人的结构是否合理，运动是否达到规划的目标。最后再对六足仿生机器人的防滑结构、测距结构这两项辅助功能模块进行设计。

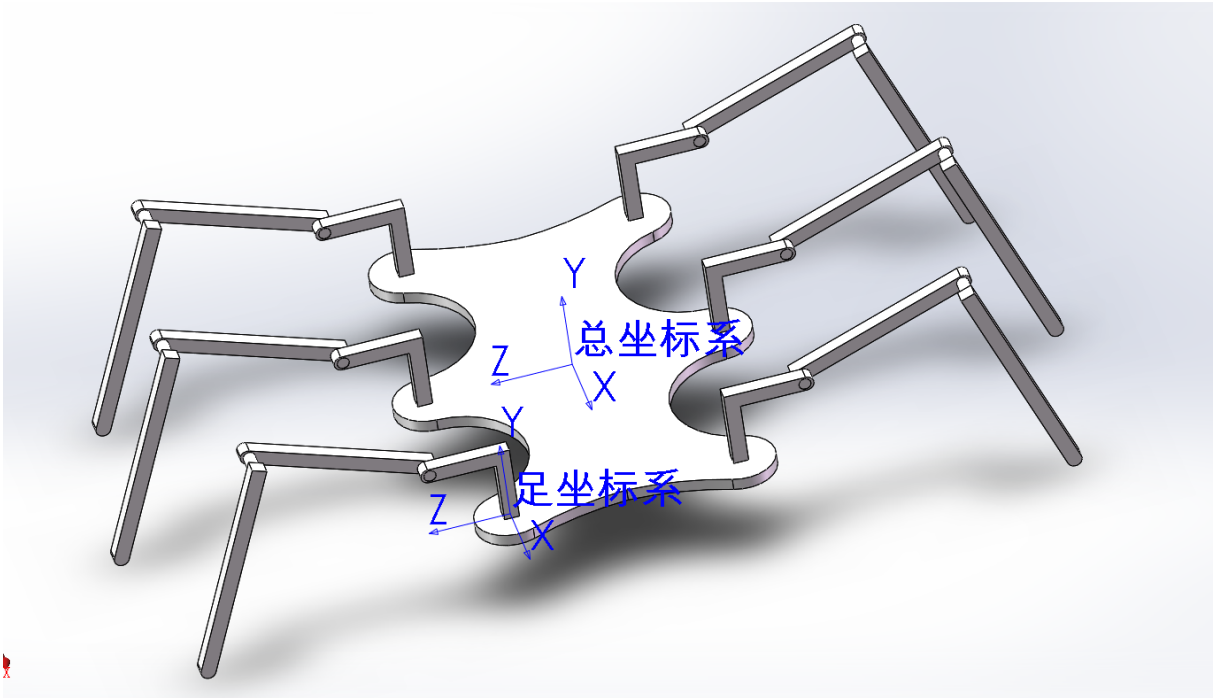
2 六足仿生机器人的运动分析

2.1 机体坐标系到足坐标系的建立

运动学分析在研究六足仿生机器人中有着重要的地位，它是后面进行步态规划、运动控制以及仿真的基础。而要进行运动学分析之前需要进行机体坐标系到足坐标系的建立，这可以为后来的运动学分析提供便利以及表达得更加清晰直观。

首先我们将六足仿生机器人的重心在地面上的投影设为机器人的机体总坐标系的原点建立坐标系，足部坐标系则以腿根关节初旋转中心为坐标原点建立足坐标系。

如下图所示。



2.2 步行足的运动分析

在运动学的研究中我们只探究运动规律，如物体的位置、速度、加速度间的关系。在六足仿生机器人的运动过程中，六只结构完全一致的足分为两组，一组为步行足，一组为支撑足。因为支撑足不运动所以不对其进行运动学分析。步行足组因其运动情况都是一样的，因此只需对其中一个进行分析即可。

对步行足的运动分析主要是要得到足尖在运动中的情况变化。本质上，六足机器人中的腿就是一个三自由度的机械手，我们使用D-H法建立步行足局部坐标系。再通过对某轴平移和旋转得到相邻轴的坐标系可得到转换矩阵。其转换矩阵 ${}^{i}_{i+1}T$ 为：

$$
{}^{i}_{i+1}T=\begin{bmatrix}
\cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\
\cos\alpha_{i-1}\sin\theta_i & \cos\alpha_{i-1}\cos\theta_i & -\sin\alpha_{i-1} & -d_i\sin\alpha_{i-1} \\
\sin\alpha_{i-1}\sin\theta_i & \sin\alpha_{i-1}\cos\theta_i & \cos\alpha_{i-1} & d_i\cos\alpha_{i-1} \\
0 & 0 & 0 & 1
\end{bmatrix}
$$

a_{i-1} ：前一个关节 i-1 的长度　　d_i ：关节 i-1 和关节 i 的偏置

θ_i ：两个关节的夹角　　α_{i-1} ：关节 i-1 的扭角

步行足的 D-H 表

杆号	α_{i-1}	d_i	θ_i	a_{i-1}
1	0	0	θ_1	0
2	90°	0	θ_2	l_1=27
3	0	0	θ_3	l_2=60.5
4	0	0	0	l_3=75

将 D-H 表中的参数带入转换矩阵 ${}^{i}_{i+1}T$ 后得：

$$
{}^{0}_{1}T=\begin{bmatrix}
\cos\theta_1 & -\sin\theta_1 & 0 & 0 \\
\sin\theta_1 & \cos\theta_1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\qquad
{}^{1}_{2}T=\begin{bmatrix}
\cos\theta_2 & -\sin\theta_2 & 0 & l_1 \\
0 & 0 & -1 & 0 \\
\sin\theta_2 & \cos\theta_2 & 0 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
$$

$$
{}^{2}_{3}T=\begin{bmatrix}
\cos\theta_3 & -\sin\theta_3 & 0 & l_2 \\
\sin\theta_3 & \cos\theta_3 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\qquad
{}^{3}_{4}T=\begin{bmatrix}
1 & 0 & 0 & l_3 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
$$

从足端坐标系到步行足坐标系的转换矩阵为：

$$
{}^{0}_{4}T={}^{0}_{1}T+{}^{1}_{2}T+{}^{2}_{3}T+{}^{3}_{4}T
$$

代入后有：

$$
{}^{0}_{4}T=\begin{bmatrix} c_1c_{23} & -c_1s_{23} & s_1 & l_3c_1c_{23}+c_1(l_2c_2+l_1) \\ s_1c_{23} & -s_1s_{23} & -c_1 & l_3s_1c_{23}+s_1(l_2c_2+l_1) \\ s_{23} & c_{23} & 0 & l_3s_{23}+l_2s_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

其中:

$$
c_i=\cos\theta_i \qquad c_{ij}=\cos(\theta_i+\theta_j) \qquad s_i=\sin\theta_i \qquad s_{ij}=\sin(\theta_i+\theta_j)
$$

综上所述，我们可以得到步行足的末端位置为：

$$
\begin{bmatrix} P_X \\ P_Y \\ P_Z \end{bmatrix}=\begin{bmatrix} l_3c_1c_{23}+c_1(l_2c_2+l_1) \\ l_3s_1c_{23}+s_1(l_2c_2+l_1) \\ l_3s_{23}+l_2s_2 \end{bmatrix}
$$

2.3 基于 MATLABT 机器人工具箱的步行足仿真分析

机器人的运动仿真分析是研究机器人的重要步骤，我们可以通过计算机来建立一个虚拟的机器人样机，对其进行仿真分析，可以直观可视的得到一些我们需要的数据例如：正逆运动学分析、求解机器人的工作空间等。我们将使用 Matlab 中的 Robotics Toolbox 进行建模与求解。

机器人工具箱（RoboticsToolbox）是学者 PeterCorke 为了方便教学而开发的一款自定义工具箱。此处我将利用机器人工具箱进行 Matlab 中的步行足建模（link），以及计算出步行足运动学方程的正逆解（ikine 和 fkine）和初步的轨迹规划（jtraj）。

2.3.1 步行足的建模

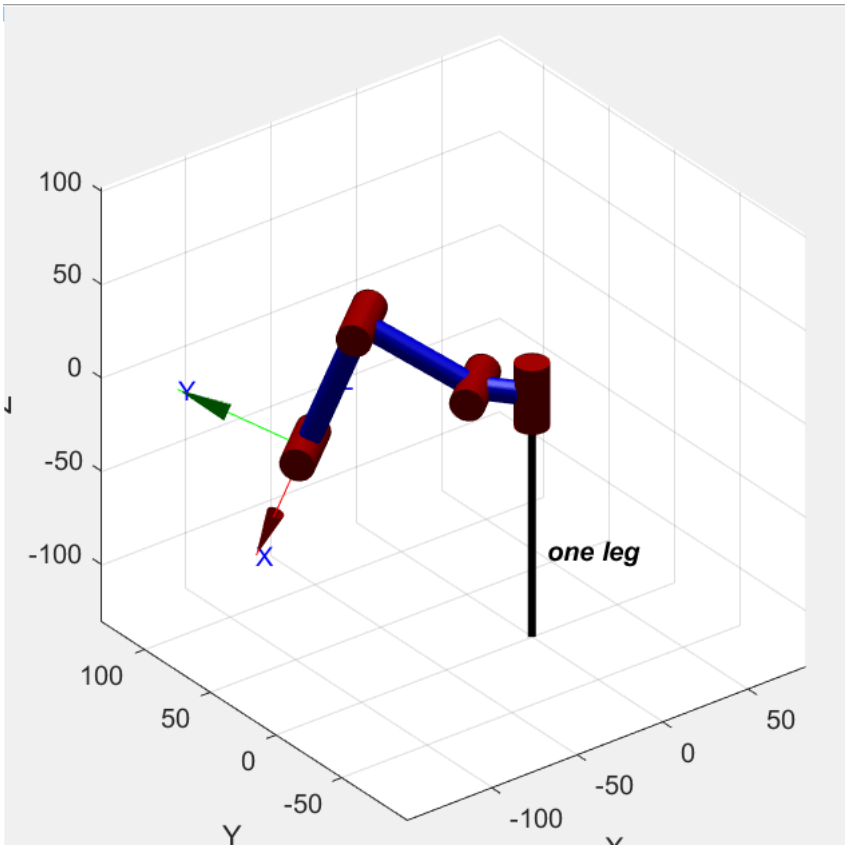
Denavit 与 Hartenberg 提出的 D-H 参数法是描述机器人的常用方法，D-H 参数法通过齐次变换来构建各个坐标系的姿态和相对位置的关系。从而使得各个坐标系可以与参考坐标系联系起来,让机器人每个部件运动位置都可以轻易的在参考坐标系中表示出来。

在分析六足仿生机器人的步行足时我们可以将其视为一个三自由度的机器手，其 D-H 表如上所示。我们通过 Robotics Toolbox 的 link（）函数按 D-H 表来构建步行足的模型。Link（）函数需要我们确定各个连杆间的夹角、偏置、扭角以及其自身的长度，因为 RoboticsToolbox 中默认的是六轴工业机器人的建模，而我们的步行足只相当于一个三轴机器手，故还需要将多出的三杆置零。

各个连杆建模完成后使用 SerialLink（）函数进行连接并设置设置各个关节的活动范围。至此步行足的模型基本建立完毕，之后再使用 Teach 可直观观察到模型。

程序如下：

```
startup_rvc                                  %调用机器人工具箱
clc ;clear;
L1=Link([0 0 27 pi/2]);                      %建立各个连杆
L2=Link([0 0 60.5 0]);
L3=Link([0 0 75 0]);
L4=Link([0 0 0 0]);
L5=Link([0 0 0 0]);
L6=Link([0 0 0 0]);
L1.qlim = [deg2rad(45) deg2rad(135)];        %设置关节活动范围
L2.qlim = [deg2rad(-45) deg2rad(90)];
L3.qlim = [deg2rad(-120) deg2rad(0)];
leg=SerialLink([L1 L2 L3 L4 L5 L6]);
leg.name='one leg';                          %命名
leg.display();
leg.teach;                                   %显示三维模型
```



2.3.2 步行足的正逆解

正解是已知步行足各个关节的变量，来求足端的空间姿态。逆解是已知步行足足端的空间姿态，来求步行足各个关节的角度变量。通过正逆运动解我们以及进行对步行足的轨迹进行规划的基础。在 Robotics Toolbox 中我们可以使用函数 fkine（）轻易获得步行足的正运动解，免除繁杂的计算。同样，我们可以使用 ikine（）函数进行逆运动解。

程序如下：

```
theta =[pi/4 pi/6 -pi/2 0 0 0];                    %起始关节变量
theta1=[pi/2 pi/4 -pi/2 0 0 0];                    %中点关节变量
theta2 = [pi*3/4 pi/6 -pi/2 0 0 0];                %终点关节变量
p0 = leg.fkine(theta)                              %求 p0 下的正运动学
p1 = leg.fkine(theta1)                             %求 p1 下的正运动学
p2 = leg.fkine(theta2)                             %求 p2 下的正运动学
M = [1 1 1 0 0 0]                                  %设置自由度
q0 = leg.ikine(p0,'q0',theta,'mask',M)             %求 p0 下的逆运动学
q1 = leg.ikine(p1,'q0',theta1,'mask',M)            %求 p1 下的逆运动学
q2=leg.ikine(p2,'q0',theta2,'mask',M)              %求 p2 下的逆运动学
```

得到结果如下：

```
p0 =
    +0.3536    +0.6124    +0.7071    +82.66
    +0.3536    +0.6124    -0.7071    +82.66
    -0.8660    +0.5000    +0         -34.7
    +0         +0         +0         +1

p1 =
    +0         +0         +1         +0
    +0.7071    +0.7071    +0         +122.8
    -0.7071    +0.7071    +0         -10.25
    +0         +0         +0         +1

p2 =
    -0.3536    -0.6124    +0.7071    -82.66
    +0.3536    +0.6124    +0.7071    +82.66
    -0.8660    +0.5000    +0         -34.7
    +0         +0         +0         +1


q0 =
    0.7854    0.5236   -1.5708         0         0         0

q1 =
```

1.5708	0.7854	-1.5708	0	0	0

q2 =

2.3562　　0.5236　　-1.5708　　0　　0　　0

2.3.3 步行足的运动空间云图

在规划轨迹前，为了进一步的了解清楚步行足的运动范围，我们可以通过上述足端坐标的表达式和各个关节的运动约束条件，来进行步行足足端可到达空间的云图绘制。

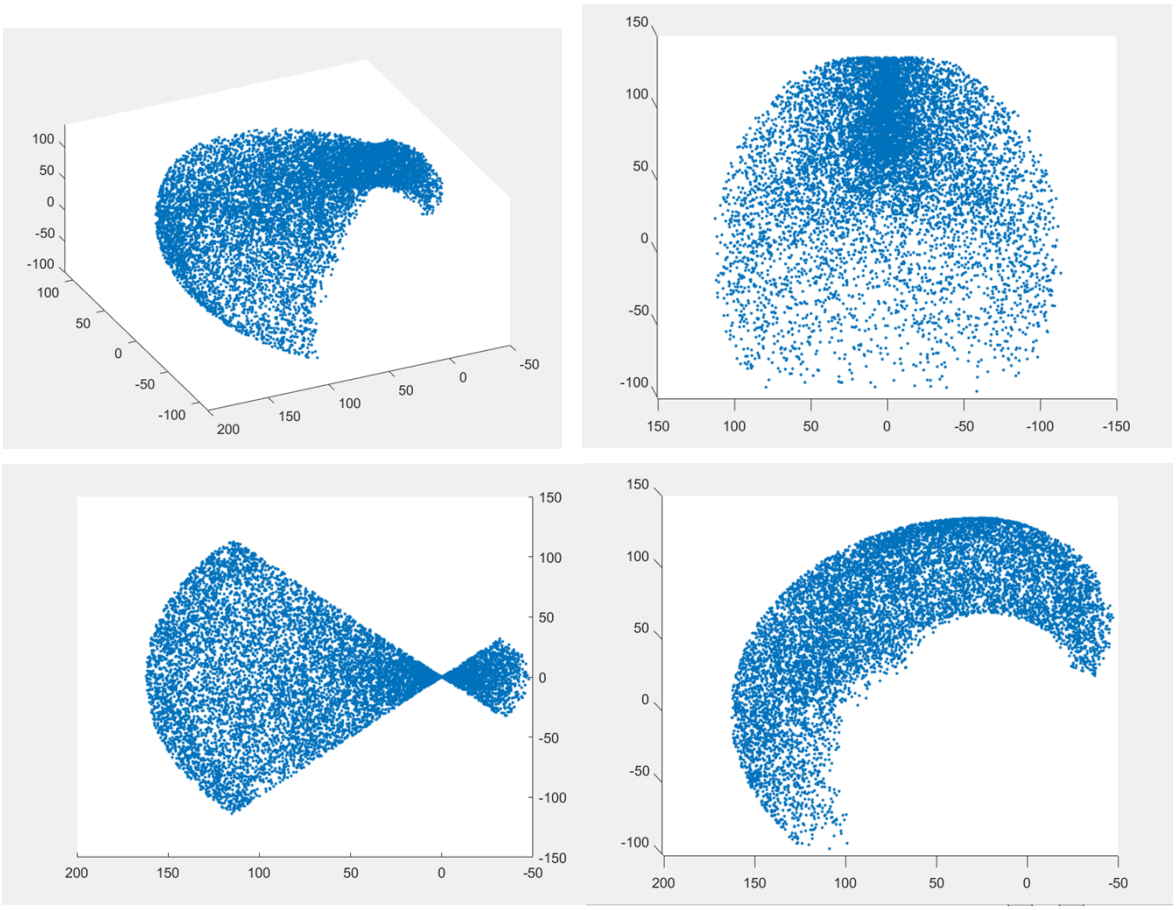
通过步行足的运动空间云图，我们可以直观的了解到步行足运动空间的形状分布和大小。可为后续的轨迹规划提供一定的参考。

此处我们使用蒙特卡洛数值分析法来对步行足足端可达空间进行仿真分析。使用randperm（1000：1）函数来随机生成一个 1 到 1000 内的一个整数，在将其映射到各个关节的活动区间中得到一个随机的关节角度变量，代入足端坐标表达式得其在该种随机角度下的坐标，重复足够多的次数后即可得到步行足足端可达范围的空间云图。

程序如下：

```
x = [];y = [];z = [];                    %足端 xyz
l1 = 27;l2 = 60.5;l3 = 75;               %步行足各个杆长
fai1 = 45:90/10000:135;                  %细分活动区间为 10000
fai2 = -45:135/10000:90;
fai3 = 0:120/10000:120;
f1 = fai1*pi/180;                        %转为弧度制
f2 = fai2*pi/180;
f3 = fai3*pi/180;
c1 = cos(f1);c2 = cos(f2);c3 = cos(f3);s1 = sin(f1);s2 = sin(f2);s3 = sin(f3);          %简化
for a = 1:10000                          %循环 10000
i1 = randperm(10000,1);                  %产生随机数
i2 = randperm(10000,1);
i3 = randperm(10000,1);
x0 = l3*c1(i1)*cos(f2(i2)+f3(i3))+c1(i1)*l1+c1(i1)*c2(i2)*l2;          %带入足端表达式
y0 = l3*s1(i1)*cos(f2(i2)+f3(i3))+s1(i1)*l1+s1(i1)*c2(i2)*l2;
z0 = l3*sin(f2(i2)+f3(i3))+l2*s2(i2);
x = [x,x0];y = [y,y0];z = [z,z0];                                          %储存
end
plot3(x,y,z,'.')                                                          %显示云图
```

结果如下图：



通过空间云图我们可以清晰的看出，其活动范围大约是在一个 200mmX200mmX200mm 的空间之中。接下来的轨迹规划应该在这个空间范围之内才能保证机器人的正常运行，如果超出了这个空间范围机器人的机械机构就可能出现干涉的情况。

3 基于遗传算法的步行足轨迹规划

3.1 轨迹规划的重要意义

轨迹规划就是对机器人各个关节在运行时的位移、速度、加速度等参数进行控制。合理的轨迹规划不仅可以增加机器的运行效率，而且也保证了其曲线的平滑，有利于机器的保护。所以轨迹规划在机器人的研究中占有重要地位。

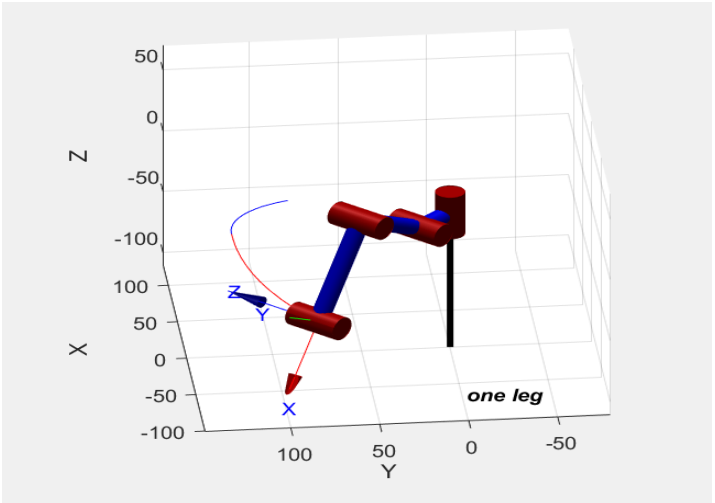
3.2 与 Robotics Toolbox 轨迹规划的对比

在 Robotics Toolbox 中就可以通过 jtraj（）函数来进行 5 次多项式法来进行轨迹规划和关节空间的规划。

程序如下：

```
init_ang = [pi/4 pi/6 -pi/2 0 0 0];                    %起始角变量
targ_ang = [pi/2 pi/4 -pi/2 0 0 0];                    %中点角变量
xxxx_xxx=[pi*3/4 pi/6 -pi/2 0 0 0];                    %终点角变量

t = 0:0.1:2;                                           %设置步数
step = 150;
[q qd qdd] = jtraj(init_ang,targ_ang,step);            %进行五次多项式轨迹规划
[q1 qd1 qdd1] = jtraj(targ_ang,xxxx_xxx,step);
traj_1 = jtraj(init_ang,targ_ang,t);                   %获取角变量
xxxx_1 = jtraj(targ_ang,xxxx_xxx,t);
JTA = transl(leg.fkine(traj_1));                       %求出足端姿态
JTA1 = transl(leg.fkine(xxxx_1));
```



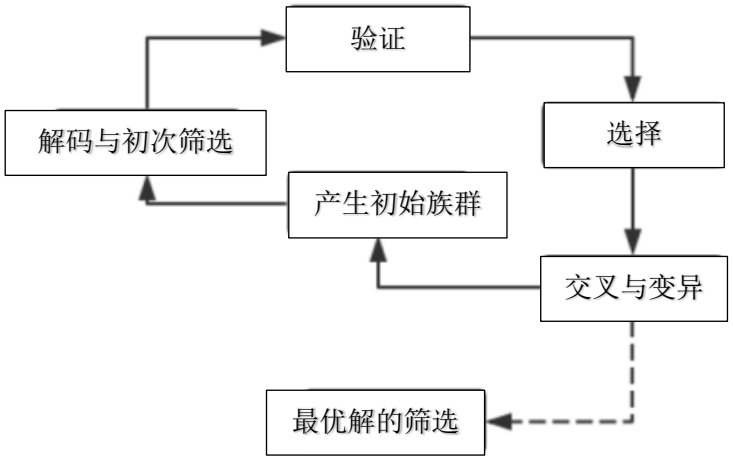
在这种规划方法中，我们是以已知其时间为前提的。这样的规划未必就是最优，我

们可以通过遗传算法来进行对时间进行推算，推出一个最优的时间，完成最优的轨迹规划。

3.3 基于遗传算法的轨迹规划的基本步骤

我使用的遗传算法分为六部分：生产原始族群部分、解码器与初次筛选部分、验证部分、选择部分、交叉变异部分、最优解筛选部分。

即先随机生成一个族群，在将其中的各个个体代入目标函数，按其适应度进行淘汰筛选。将符合我们要求的个体留下，不合格的就剔除，以此往复数代。最终将在我们的控制之下，族群整体会越来越符合我们的目标函数，从中我们可以得到目标函数最优解。



3.4 生成初始族群

生成初始种群，顾名思义就是产生一群数组来代表我们的时间。我们只需要一个时间值。故我以一个 10（chromlength）位的二进制数组作为本次族群个体的表现形式，族群内个体数量为 40（popsize）。

代码如下：

```
function pop=initpop(popsize,chromlength)
%    生成种群
pop=round(rand(popsize,chromlength));
end
```

3.5 解码与初次筛选

在上一步骤中我们已经得到初始种群，但还不能直接进行使用。因为我们的族群是通过 rand（）函数生产 40 个 10 位的数组再通过 round（）四舍五入得到的。所以存在可能时间为 0 的情况，但是时间是不可能为 0。所以我们需要先筛选一遍，以保证时间不为 0。而且每一代都要经过筛选，防止在交叉变异中产生时间为 0 的错误。

代码如下：

```
for i=1:40                                          %杜绝时间为0
    if pop(i,:)==[0 0 0 0 0 0 0 0 0 0]
       pop(i,:)=[1 1 1 1 1 1 1 1 1 1];
    end
end
```

经过初次筛选后就可以进行二进制转十进制的解码过程了。

代码如下：

```
for i=1:40                                          %转换为十进制
    t1(i)=pop(i,1)*1+pop(i,2)*2+pop(i,3)*4+pop(i,4)*8+pop(i,5)*16.
    +pop(i,6)*32+pop(i,7)*64+pop(i,8)*128+pop(i,9)*256+pop(i,10)*512;
end
```

解码后还需映射回规划时间的大概范围，我们这里取 2。即规划时间范围为（0，2）。

代码如下：

```
t=t1/1023*2;                                        %映射回区间
```

3.6 验证

得到时间后代入到五次多项式曲线公式中，得出每一段轨迹的速度、加速度表达式，验证得到的时间下产生的运动是否满足我们的运动约束条件。

公式如下：

$$v(p)=30*60/t(i)^3*o^2-60*60/t(i)^4*o^3+30*60/t(i)^5*o^4;$$

$$a(p)=60*60/t(i)^3*o-180*60/t(i)^4*o^2+120*60/t(i)^5*o^3;$$

将时间代入到速度、加速度公式后，还需再通过一个 for（）进行遍历，寻找期间的最大速度、加速度：

代码如下：

```
for o=0:0.0001:t(i)
    p=p+1;
    v(p)=30*60/t(i)^3*o^2-60*60/t(i)^4*o^3+30*60/t(i)^5*o^4;
    a(p)=60*60/t(i)^3*o-180*60/t(i)^4*o^2+120*60/t(i)^5*o^3;
end
v=abs(v);a=abs(a);
vmax(i)=max(v);amax(i)=max(a);
```

3.7 选择

选择开始的步骤就是进行适应度的计算，通过适应度的高低可以有效的辨别各组数据的情况，是否满足我们的模型，是否与我们的运动约束条件有冲突。我们通过添加罚函数的方式来进行区分。对不满足我们运动约束条件的个体适应的减去 20，从而进行区分。

主要程序如下：

```
syd=[];
for i=1:40
   pd=vmax(i)<480&&amax(i)<134;
   if pd==1
       syd(i)=100-t(i);
   else
       syd(i)=80-t(i);
   end
end
```

得到各个个体的适应度后，我们需要进行排序。通过 sort()函数进行从小到大的排序，再通过 find（）函数找到排序中的某些值在原来的数组中的位置，从而得到了在原数组中的那些是最大，那些是最小。在这里，我的筛选策略是将适应度低的替换为适应度高的个体以提高合适基因的占比。另外，在不同的时期的筛选压力也设置得不一样，在前四十代只替换适应度后 2 位的个体（5%），在后四十代则将后 5 位替换（12.5%）。这样设置有利于在前期让族群提高多样性，以提高算法搜索的范围。后期加强筛选力度有利于让族群收敛，使族群整体向符合我们要求的区域靠拢。

主要程序如下：

```
pk=syd;
pkpx=sort(pk);
[m,n1]=find(pk==pkpx(1));
[m,n2]=find(pk==pkpx(2));
[m,n3]=find(pk==pkpx(3));
[m,n4]=find(pk==pkpx(4));
[m,n5]=find(pk==pkpx(5));
[m,n21]=find(pk==pkpx(40));
[m,n22]=find(pk==pkpx(39));
[m,n23]=find(pk==pkpx(38));
[m,n24]=find(pk==pkpx(36));
[m,n25]=find(pk==pkpx(35));
```

```
if k<=40                          %替换
    pop(n1(1),:)=pop(n21(1),:);
    pop(n2(1),:)=pop(n22(1),:);
    pop(n3(1),:)=pop(n23(1),:);
end
if k>40
    pop(n1(1),:)=pop(n21(1),:);
    pop(n2(1),:)=pop(n22(1),:);
    pop(n3(1),:)=pop(n23(1),:);
    pop(n4(1),:)=pop(n24(1),:);
    pop(n5(1),:)=pop(n25(1),:);
end
```

3.8 交叉变异

通过淘汰适应度差的个体的基因，提高适应度高的个体的基因。来提升族群的总体适应度，让族群的发展朝着我们所期待的方向发展。进行淘汰后已经是开始进行下一代族群的建立了，我们还需要对现有的个体进行交叉，以丰富族群的多样性。在这里我们随机对族群中相邻个体的随机 3 位进行交换。且交叉发生概率 pc 在不同时期都有所差别，其变化可以表达为：

$$pc = 0.8-k/200$$

即随族群的发展，交叉的机会将越来越少。后期交叉的减少有利于族群的收敛，且前期的较大交叉概率也能一定程度上的保证族群多样性。

除了交叉外还有变异，通过变异可以拓宽族群的多样性，让族群的探索范围更大，避免陷入局部最优解的尴尬境地。尤其是在后期交叉概率下降后，变异成了多样性的保证。变异的概率 pm 同上述的 pc 一样，都是随代数 k 变化的。这里变异概率 pm 可表达为：

$$pm = 0.1+k/200$$

且我们对变异个体有两种变异方案，对适应度已经满足要求即大于八十的个体来说，就没有必要进行大的变异。我们就对其的低位数进行变异即可。而对适应度本来就不达预期的个体就对它进行较大的变异，让它去拓展算法的搜索范围，来增加族群的多样性。

主要程序如下：