正文

一、求解运动学方程

我们使用的是 Kawasaki ZX165U 机器人，一个六自由度的转动关节机械臂，我们将求解以关节角度为变量的运动学方程。

图 1 Kawasaki ZX165U 机器人

Kawasaki ZX165U 机器人是一个六自由度机器人，所有关节均为转动关节（即这是一个 6R 机构）。

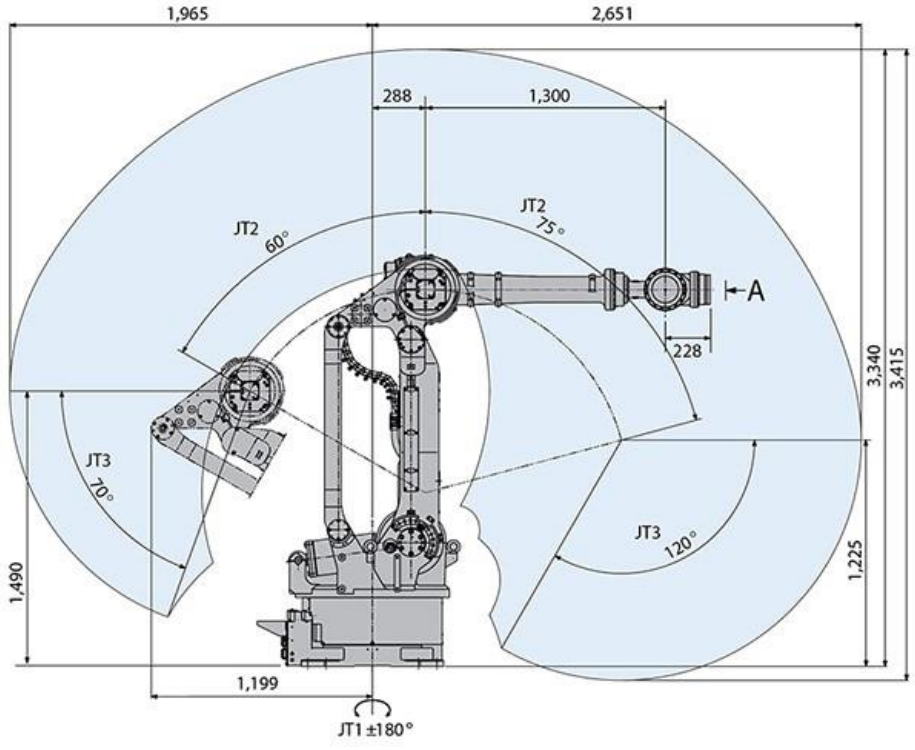

图 2 Kawasaki ZX165U 机器人的工作空间

根据网上查到的该型号机械臂相关参数及实测值，我们可以得到 Kawasaki ZX165U 机器人的 D-H 参数表。

D-H 参数表					
j	q	d	a	α	offset
1	q1	0	288	pi/2	0
2	q2	288	1370	0	pi/2
3	q3	-288	0	pi/2	0
4	q4	288	0	0	0
5	q5	1000	0	pi/2	0
6	q6	228	228	pi/2	pi/2

根据公式

$$ {}^{i-1}_{i}T = \begin{matrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{matrix} $$

可以求出每一个连杆变换矩阵。求得的矩阵如下：

$$ {}^{0}_{1}T = \begin{matrix} \cos q1 & -\sin q1 & 0 & 288 \\ 0 & 0 & -1 & 0 \\ \sin q1 & \cos q1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} $$

$$ {}^{1}_{2}T = \begin{matrix} \cos q2 & -\sin q2 & 0 & 1370 \\ \sin q2 & \cos q2 & 0 & 0 \\ 0 & 0 & 1 & 288 \\ 0 & 0 & 0 & 1 \end{matrix} $$

$$ {}^{2}_{3}T = \begin{matrix} \cos q3 & -\sin q3 & 0 & 0 \\ 0 & 0 & -1 & 288 \\ \sin q3 & \cos q3 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} $$

$$ {}^{3}_{4}T = \begin{matrix} \cos q4 & -\sin q4 & 0 & 0 \\ \sin q4 & \cos q4 & 0 & 0 \\ 0 & 0 & 1 & 288 \\ 0 & 0 & 0 & 1 \end{matrix} $$

$$ {}^{4}_{5}T = \begin{matrix} \cos q5 & -\sin q5 & 0 & 0 \\ 0 & 0 & -1 & -1000 \\ \sin q5 & \cos q5 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} $$

$$ {}^{5}_{6}T = \begin{matrix} \cos q6 & -\sin q6 & 0 & 228 \\ 0 & 0 & -1 & -228 \\ \sin q6 & \cos q6 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{matrix} $$

将各个连杆的矩阵连乘得到${}^{0}_{6}T$（由于计算量较大，我们使用 matlab 进行计算，脚本见附录 2），最后，六个连杆坐标变换矩阵的乘积为

$$ {}^{0}_{6}T = \begin{matrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{matrix} $$

其中 R11= s(q6)*(c(q3)*s(q4)*(c(q1)*c(q2) - s(q1)*s(q2)) + c(q4)*s(q3)*(c(q1)*c(q2) - s(q1)*s(q2))) - c(q6)*(c(q5)*(c(q3)*c(q4)*(c(q1)*c(q2) - s(q1)*s(q2)) - s(q3)*s(q4)*(c(q1)*c(q2) - s(q1)*s(q2))) + s(q5)*(c(q1)*s(q2) + c(q2)*s(q1)))

R21= s(q6)*(c(q3)*c(q4) - s(q3)*s(q4)) + c(q5)*c(q6)*(c(q3)*s(q4) + c(q4)*s(q3))

R31= s(q6)*(c(q3)*s(q4)*(c(q1)*s(q2) + c(q2)*s(q1)) + c(q4)*s(q3)*(c(q1)*s(q2) + c(q2)*s(q1))) - c(q6)*(c(q5)*(c(q3)*c(q4)*(c(q1)*s(q2) + c(q2)*s(q1)) - s(q3)*s(q4)*(c(q1)*s(q2) + c(q2)*s(q1))) - s(q5)*(c(q1)*c(q2) - s(q1)*s(q2)))

R12= c(q6)*(c(q3)*s(q4)*(c(q1)*c(q2) - s(q1)*s(q2)) + c(q4)*s(q3)*(c(q1)*c(q2) - s(q1)*s(q2))) - s(q6)*(c(q5)*(c(q3)*c(q4)*(c(q1)*c(q2) - s(q1)*s(q2)) - s(q3)*s(q4)*(c(q1)*c(q2) - s(q1)*s(q2))) + s(q5)*(c(q1)*s(q2) + c(q2)*s(q1)))

R22= c(q6)*(c(q3)*c(q4) - s(q3)*s(q4)) + c(q5)*s(q6)*(c(q3)*s(q4) + c(q4)*s(q3))

R32= c(q6)*(c(q3)*s(q4)*(c(q1)*s(q2) + c(q2)*s(q1)) + c(q4)*s(q3)*(c(q1)*s(q2) + c(q2)*s(q1))) - s(q6)*(c(q5)*(c(q3)*c(q4)*(c(q1)*s(q2) + c(q2)*s(q1)) - s(q3)*s(q4)*(c(q1)*s(q2) + c(q2)*s(q1))) - s(q5)*(c(q1)*c(q2) - s(q1)*s(q2)))

R13= s(q5)*(c(q3)*c(q4)*(c(q1)*c(q2) - s(q1)*s(q2)) - s(q3)*s(q4)*(c(q1)*c(q2) - s(q1)*s(q2))) - c(q5)*(c(q1)*s(q2) + c(q2)*s(q1))

R23= -s(q5)*(c(q3)*s(q4) + c(q4)*s(q3))

R33= s(q5)*(c(q3)*c(q4)*(c(q1)*s(q2) + c(q2)*s(q1)) - s(q3)*s(q4)*(c(q1)*s(q2) + c(q2)*s(q1))) + c(q5)*(c(q1)*c(q2) - s(q1)*s(q2))

PX=1370*c(q1) + 228*c(q5)*(c(q3)*c(q4)*(c(q1)*c(q2) - s(q1)*s(q2)) - s(q3)*s(q4)*(c(q1)*c(q2) - s(q1)*s(q2))) + 228*s(q5)*(c(q3)*c(q4)*(c(q1)*c(q2) - s(q1)*s(q2)) - s(q3)*s(q4)*(c(q1)*c(q2) - s(q1)*s(q2))) - 228*c(q5)*(c(q1)*s(q2) + c(q2)*s(q1)) + 228*s(q5)*(c(q1)*s(q2) + c(q2)*s(q1)) + 1000*c(q3)*s(q4)*(c(q1)*c(q2) - s(q1)*s(q2)) + 1000*c(q4)*s(q3)*(c(q1)*c(q2) - s(q1)*s(q2)) + 288

PY= 1000*c(q3)*c(q4) - 1000*s(q3)*s(q4) - 228*c(q5)*(c(q3)*s(q4) + c(q4)*s(q3)) - 228*s(q5)*(c(q3)*s(q4) + c(q4)*s(q3)) - 288

PZ= 1370*s(q1) + 228*c(q5)*(c(q3)*c(q4)*(c(q1)*s(q2) + c(q2)*s(q1)) - s(q3)*s(q4)*(c(q1)*s(q2) + c(q2)*s(q1))) + 228*s(q5)*(c(q3)*c(q4)*(c(q1)*s(q2) + c(q2)*s(q1)) - s(q3)*s(q4)*(c(q1)*s(q2) + c(q2)*s(q1))) + 228*c(q5)*(c(q1)*c(q2) - s(q1)*s(q2)) - 228*s(q5)*(c(q1)*c(q2) - s(q1)*s(q2)) + 1000*c(q3)*s(q4)*(c(q1)*s(q2) + c(q2)*s(q1)) + 1000*c(q4)*s(q3)*(c(q1)*s(q2) + c(q2)*s(q1))

二、轨迹规划实例，写一个汉字“利”

由上一部分已知的 D-H 参数表，可以在 matlab 中建立机器人模型。这里需要说明的是，matlab 的机器人工具箱不同的版本语法不一样，这里使用的是 matlab r2016a+机器人工具箱 10.2。

连杆建立如下：

```
L1 = Link('alpha', pi/2, 'a',288, 'd',0);%定义连杆
L2 = Link('alpha', 0, 'a',1370, 'd',288, 'offset',pi/2);
L3 = Link('alpha', pi/2, 'a',0, 'd',-288);
L4 = Link('alpha', 0, 'a',0, 'd',288);
L5 = Link('alpha', pi/2, 'a',0, 'd',1000);
L6 = Link('alpha', 0, 'a',228, 'd',228, 'offset',pi/2);
robot=SerialLink([L1,L2,L3,L4,L5,L6]);   %建立连杆
```

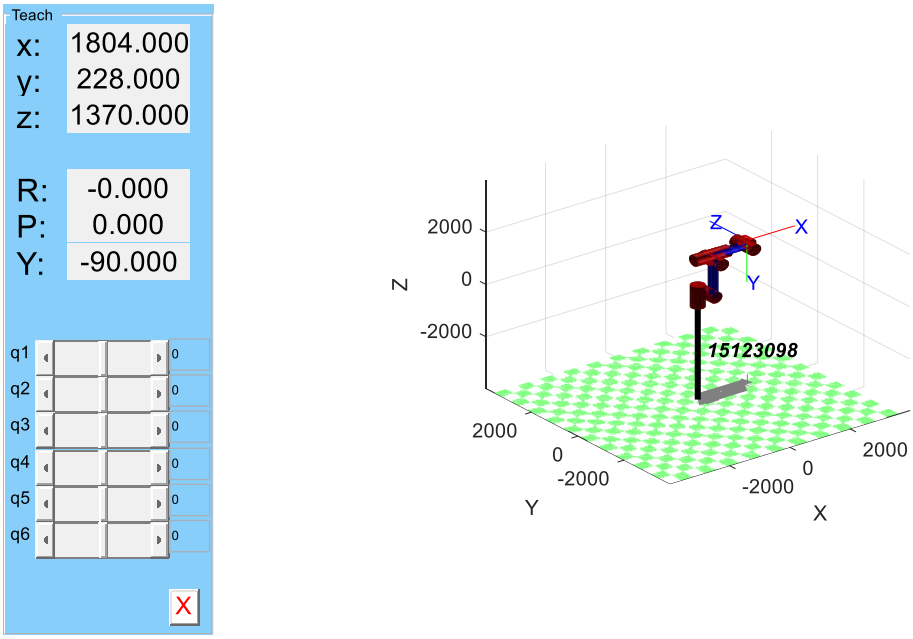

图 3 建立的机器人模型

可以看到建立后的模型如上图所示。接下来可以在示教器中修改角度值使机械臂到需要写的汉字的每个端点位置，记录下此时的角度值。然后就可以编写代码是机械臂运动写下这个汉字。机械臂运动的代码如下（由于机械臂写字的原理相同，在这里仅列举其中一笔的代码，完整的代码见附录）：

```
init_ang=[0 0 0 0 0 0];%p1
targ_ang=[pi/6,-pi/3,pi/9,0,0,0];%p2
step=10;%运动速度
[q,qd,qdd] = jtraj(init_ang, targ_ang, step);
```

```
subplot(1,2,1);%图像位置
robot.plot(q);
```

这段代码设定了机械臂起始处和终点处两个位置的关节角度；step 设置了运动的频率，直观的解释就是机械臂运动的速度；subplot(1,2,1)函数意为将屏幕分为一行两列，此图像位于第一列；robot.plot 函数使机械臂从起始点运动到终点，路径为一条直线。

机械臂生成轨迹的代码如下：

```
p1=robot.fkine(init_ang)
p2=robot.fkine(targ_ang)
Tc=ctraj(p1, p2, step);
Tjtraj=transl(Tc);
subplot(1,2,2);
plot2(Tjtraj,'r');
hold on;%撇
grid on;
```

代码的开头两行将初始位置和重点位置进行逆运动学求解，获得两组位置的矩阵 p1 和 p2，图像位于整个显示框一行两列的第二列，并且显示颜色为“r”，即红色。“hold on”表示将图像保留，下一个图像生成时也不消除，这样一笔一笔显示出来保留在一起，就是一个完整的字。最后完整生成的字如下图。

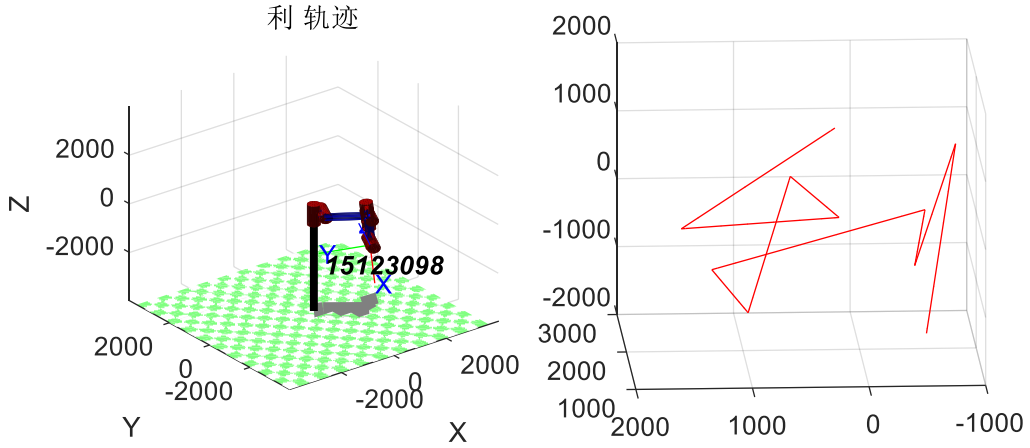

图 4 轨迹规划生成汉字“利”

三、运动轨迹规划的发展

机器人可以替代人类完成很多工作，这其中一项关键技术就是运动轨迹规划。对已知静态环境中机器人的轨迹规划的研究已经持续了近 40 年，轨迹规划的研究具有很大的价值，多年的研究过程愈加证明了轨迹规划是一个复杂的难题。轨迹规划的计算量取决于任务、环境的复杂性以及对规划路径质量的要求，一个好的算法应该兼顾对速度和质量的期望。

我国的工业机器人从 80 年代“七五”科技攻关开始起步，目前已基本掌握了机器人设计制造技术、控制系统硬件和软件设计技术、运动学和轨迹规划技术，并能生产部分关键元件，开发出了喷漆、弧焊、点焊、装配、搬运等机器人。但总的来看，我国工业机器人技术及应用水平远低于发达国家水平。

美国是机器人的诞生地，早在 1962 年就研制出世界上第一台工业机器人，比起号称

“机器人王国”的日本起步至少早五六年。1971 年，通用汽车公司第一次用机器人进行点焊。西欧是仅次于日美机器人的生产基地，也是日美机器人的重要市场。德国工业机器人的总数占世界第三，对于一些有危险、有毒、有害的工作岗位，德国要求必须以机器人替代普通人的劳动，同时提出了 1985 年以后要向高级的、带感觉的智能型机器人转移目标。

我们这里要讲的运动轨迹规划（Motion Planning）有别于路径规划（Path Planning），一般来说，路径规划常用于无人车领域，而运动轨迹规划主要用于机械臂。确定 3D 空间的一个姿势(pose)需要 6 个变量，而对于关节数大于 6 的机械臂结构，它的规划空间维度就大于 6，成为冗余系统(redundant system)，从而使规划问题变得更为复杂。所谓冗余系统，就是说，存在多种关节角度配置能够使得终端达到相同的位姿，存在无数的解。这是达到的最终姿势有无数个解，那么如何到达这个最终姿势，整个运动的轨迹，更是存在无数个解。既然存在无数的解，那么很明显，存在两种不同的方向，一种是找到最好的那个解，另一种是快速的找到一个有效的解。前者，大部分算法使用最优规划(Optimization-based Planning)，后者使用采样规划 (Sampling-based Planning)。

接下来介绍一种在 ROS 系统中常用的 OMPL 库。OMPL (Open Motion Planning Library)，开源运动规划库，是一个运动规划的 C++库，其包含了很多运动规划领域的前沿算法。虽然 OMPL 里面提到了最优规划，但总体来说 OMPL 还是一个采样规划算法库。而采样规划算法中，最出名的莫过于 Rapidly-exploring Random Trees (RRT) 和 Probabilistic Roadmap (PRM)。

RRT 通过使用来自搜索空间的随机样本来生长以起始配置为根的树。随着每个样本的绘制，都会尝试与树中最近的状态进行连接。如果连接可行（完全通过空闲空间并服从任何约束），则会导致新状态添加到树中。通过对搜索空间进行均匀采样，扩展现有状态的概率与其 Voronoi 区域的大小成正比。由于最大的 Voronoi 地区属于搜索前沿的 state，这意味着该树优先向大型未探测地区扩展。树与新状态之间的连接长度经常受到增长因素的限制。如果随机样本与树中最近状态的距离超过此限制允许的范围，则使用随机样本沿树的最大距离处的新状态，而不是随机样本本身。随机样本可以被视为控制树木生长的方向，而生长因子决定其速率。RRT 增长可以通过增加特定区域采样状态的可能性而产生偏差。 RRT 的大多数实际应用都利用这一点来指导对规划问题目标的搜索。这是通过在状态抽样过程中引入一个向目标采样的小的概率来实现的。这个概率越高，树越向着目标生长。

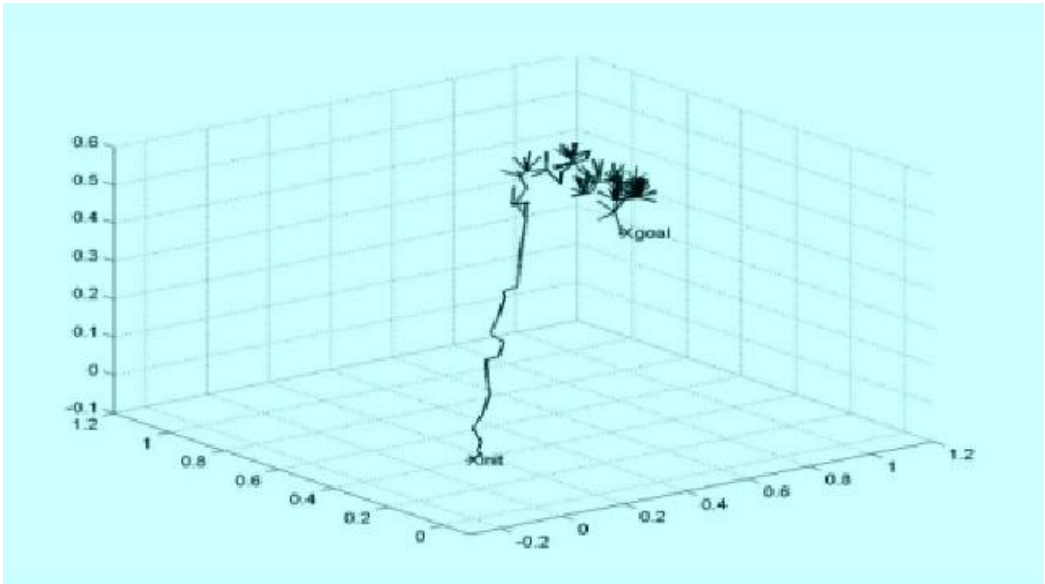

图 5 基于 RRT 算法的扩展树

PRM 随机路径图法由 Lydia E. 等人在 1996 年提出，它的优点在于：1）克服了以往一些路径规划方法易于陷入局部极小的缺点；2）可应用于多自由度的机器人的路径规划中；3）计算量小。PRM 是一种基于图搜索的方法，它将连续空间转换成离散空间，再利用 A*等搜索算法在路线图上寻找路径，以提高搜索效率。这种方法能用相对少的随机采样点来找到一个解，对多数问题而言，相对少的样本足以覆盖大部分可行的空间，并且找到路径的概率为 1（随着采样数增加，P（找到一条路径）指数的趋向于 1）。显然，当采样点太少，或者分布不合理时，PRM 算法是不完备的，但是随着采用点的增加，也可以达到完备。

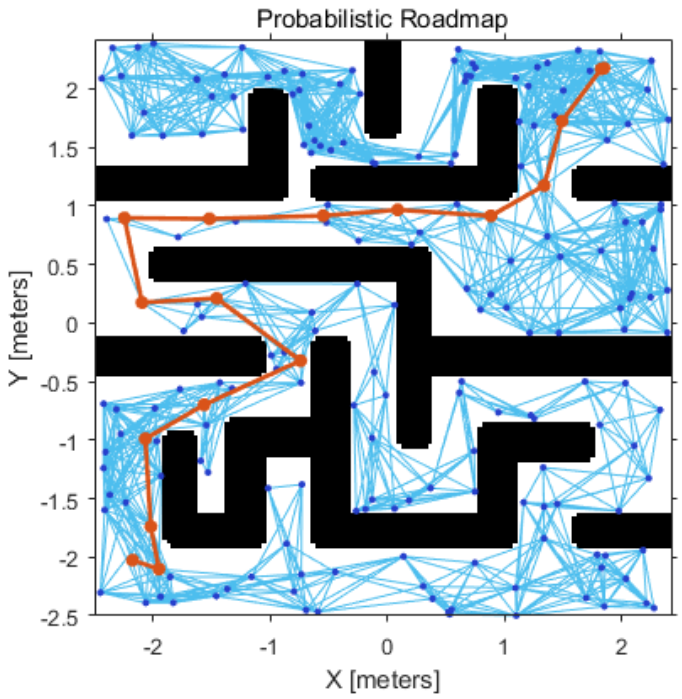

图 6 PRM 算法

参考文献

【1】. 熊有伦，机器人技术基础，第一版，华中科技大学出版社，2011-01-01

【2】. (美)John J. Craig 等，机器人学导论，原书第三版，机械工业出版社，2006-6-1

【3】. Marc-Pony ， Matlab Robotic Toolbox 工具箱学习笔记（二），http://blog.sina.com.cn/s/blog_a16714bf0101hyn2.html，2014 年 3 月 20 日。

【4】. chauncygu，PRM 路径规划算法，https://blog.csdn.net/chauncygu/article/details/78032283，2017 年 09 月 19 日

【5】. 啊冷 cold，RRT 基本概念，https://blog.csdn.net/baidu_36645485/article/details/79090718，2018 年 01 月 18 日

【6】. 郑秀娟，移动机械臂的运动控制与轨迹规划算法研究，武汉科技大学，2012.05

附录：机械臂轨迹规划生成汉字“利”代码

```
clear;
clc;
L1 = Link('alpha', pi/2, 'a',288, 'd',0);%定义连杆
L2 = Link('alpha', 0, 'a',1370, 'd',288, 'offset',pi/2);
L3 = Link('alpha', pi/2, 'a',0, 'd',-288);
L4 = Link('alpha', 0, 'a',0, 'd',288);
L5 = Link('alpha', pi/2, 'a',0, 'd',1000);
L6 = Link('alpha', 0, 'a',228, 'd',228, 'offset',pi/2);
robot=SerialLink([L1,L2,L3,L4,L5,L6]);   %建立连杆
robot.name ='15123098';      %名字

init_ang=[0 0 0 0 0 0];%p1
targ_ang=[pi/6,-pi/3,pi/9,0,0,0];%p2
targ_ang2=[0,0,-pi/4,0,0,0];%p3
targ_ang3=[pi/9,pi/9,-pi/6,0,0,0];%p4
targ_ang4=[pi/9,-pi/2,pi/9,0,0,0];%p5
targ_ang5=[5*pi/36,-5*pi/12,pi/9,0,0,0];%p6
targ_ang6=[-pi/9,-pi/4,0,0,0,0];%p7
targ_ang7=[-pi/10,-pi/2.5,pi/18,0,0,0];%p8
targ_ang8=[-pi/6,-pi/18,0,0,0,0];%p9
targ_ang9=[-pi/7.2,-pi/1.9,pi/18,0,0,0];%p10
step=10;%运动速度
[q,qd,qdd] = jtraj(init_ang, targ_ang, step);
[q2,qd,qdd] = jtraj(targ_ang, targ_ang2, step);
[q3,qd,qdd] = jtraj(targ_ang2, targ_ang3, step);
[q4,qd,qdd] = jtraj(targ_ang3, targ_ang4, step);
[q5,qd,qdd] = jtraj(targ_ang4, targ_ang5, step);
[q6,qd,qdd] = jtraj(targ_ang5, targ_ang6, step);
[q7,qd,qdd] = jtraj(targ_ang6, targ_ang7, step);
[q8,qd,qdd] = jtraj(targ_ang7, targ_ang8, step);
[q9,qd,qdd] = jtraj(targ_ang8, targ_ang9, step);%连线
subplot(1,2,1);%图像位置
robot.plot(q);
robot.plot(q2);
robot.plot(q3);
robot.plot(q4);
robot.plot(q5);
robot.plot(q6);
robot.plot(q7);
robot.plot(q8);
robot.plot(q9);%运动

%"利"字轨迹
```

```
title('利 轨迹');
p1=robot.fkine(init_ang)
p2=robot.fkine(targ_ang)
Tc=ctraj(p1, p2, step);
Tjtraj=transl(Tc);
subplot(1,2,2);
plot2(Tjtraj,'r');
hold on;%撇
grid on;

p2=robot.fkine(targ_ang)
p3=robot.fkine(targ_ang2)
Tc=ctraj(p2, p3, step);
Tjtraj=transl(Tc);
subplot(1,2,2);
plot2(Tjtraj,'r');
hold on;%横
grid on;

p3=robot.fkine(targ_ang2)
p4=robot.fkine(targ_ang3)
Tc=ctraj(p3, p4, step);
Tjtraj=transl(Tc);
subplot(1,2,2);
plot2(Tjtraj,'r');
hold on;%斜上
grid on;

p4=robot.fkine(targ_ang3)
p5=robot.fkine(targ_ang4)
Tc=ctraj(p4, p5, step);
Tjtraj=transl(Tc);
subplot(1,2,2);
plot2(Tjtraj,'r');
hold on;%竖
grid on;

p5=robot.fkine(targ_ang4)
p6=robot.fkine(targ_ang5)
Tc=ctraj(p5, p6, step);
Tjtraj=transl(Tc);
subplot(1,2,2);
plot2(Tjtraj,'r');
hold on;%斜上
```

```
grid on;

p6=robot.fkine(targ_ang5)
p7=robot.fkine(targ_ang6)
Tc=ctraj(p6, p7, step);
Tjtraj=transl(Tc);
subplot(1,2,2);
plot2(Tjtraj,'r');
hold on;%斜上
grid on;

p7=robot.fkine(targ_ang6)
p8=robot.fkine(targ_ang7)
Tc=ctraj(p7, p8, step);
Tjtraj=transl(Tc);
subplot(1,2,2);
plot2(Tjtraj,'r');
hold on;%短竖
grid on;

p8=robot.fkine(targ_ang7)
p9=robot.fkine(targ_ang8)
Tc=ctraj(p8, p9, step);
Tjtraj=transl(Tc);
subplot(1,2,2);
plot2(Tjtraj,'r');
hold on;%斜上
grid on;

p9=robot.fkine(targ_ang8)
p10=robot.fkine(targ_ang9)
Tc=ctraj(p9, p10, step);
Tjtraj=transl(Tc);
subplot(1,2,2);
plot2(Tjtraj,'r');
hold on;%长竖
grid on;
```

附录 2：连杆矩阵连乘计算

```
clear;
clc;
syms theta5 theta6 theta4 theta3 theta2 theta1;
T54=[cos(theta5) -sin(theta5) 0 0
    0 0 -1 -1000
    sin(theta5) cos(theta5) 0 0
    0 0 0 1];

T65=[-cos(theta6) -sin(theta6) 0 228
    0 0 -1 -228
    sin(theta6) cos(theta6) 0 0
    0 0 0 1];

T43=[cos(theta4) -sin(theta4) 0 0
    sin(theta4) cos(theta4) 0 0
    0 0 1 288
    0 0 0 1];

T32=[cos(theta3) -sin(theta3) 0 0
    0 0 -1 288
    sin(theta3) cos(theta3) 0 0
    0 0 0 1];

T21=[cos(theta2) -sin(theta2) 0 1370
    sin(theta2) cos(theta2) 0 0
    0 0 1 288
    0 0 0 1];

T10=[cos(theta1) -sin(theta1) 0 288
    0 0 -1 0
    sin(theta1) cos(theta1) 0 0
    0 0 0 1];

T60=T10*T21*T32*T43*T54*T65
```