4 机械臂运动学实验（正运动学）

4.1 预备知识

请复习讲义有关章节。有关连杆、关节、齐次坐标变换等基础知识，在本章的实验讲义里略去，请自行复习。这里只简单再重点突出一下关键知识点。

首先，我们需要搞清姿态的表示方法，主要参考《机器人学导论》(John J.Craig 著 北航贠超老师等翻译)第 2 章有关姿态的描述方法。

我们知道姿态不论在哪个坐标系，其变换矩阵 T 的前面 3X3 的矩阵代表姿态，它有 9 个元素，如果我们有一个坐标点，它的笛卡尔空间（直角坐标系）X/Y/Z 值只代表各坐标系的空间位置，还需要姿态矩阵。如果要表达姿态，用九个元素表达，显然是难以估算的。

能否用比较少的参数来表达呢，这个是基础。书上讲了用三个参量即可表达，具体证明（正交矩阵的凯莱公式）见 P30 页，用三个参量表达的 24 种表示法中有 2 种基本的表达方法，再加上四元数表达法：

1）X—Y—Z 固定角坐标系表达方法，即 RPY 法，用俯仰角（Pitch）、偏转角（Yaw）和回转角（Roll）来表达；

2）Z-Y-X 欧拉角以及 Z-Y-Z 欧拉角；

3）四元数（ quaternion）表达法。

具体描述见教材和讲义，这里不花篇幅讨论。

4.2 D-H 表示法实验

1） 第一步，重温一下课堂所学的 D-H 表示法。

实验课程前面已经涉及 RVIZ 以及 URDF（Unified Robot Description Format）文件 (包括 xacro)、launch 文件等内容，要在 rviz 里面加载一个好看的实物模型，需要使用到 URDF 文件对模型进行描述，所以我们需要知道机械图纸和 URDF 之间的关系。

URDF 翻译出来就是统一机器人描述格式。ROS 中的 URDF 功能包包含一个 URDF 的 C++解析器，URDF 文件使用 XML 格式描述机器人模型。

对于机器人模型，一切的物理结构在 rviz 里都统一抽象成连杆(link)和关节

(joint)，这个我们在机器人学里就学习过，就是 D-H 方法。Denavit 和 Hartenberg 在 1955 年提出一种通用的方法，这种方法在机器人的每个连杆上都固定一个坐标系，然后用 4×4 的齐次变换矩阵来描述相邻两连杆的空间关系。通过依次变换可最终推导出末端执行器相对于基坐标系的位姿，从而建立机器人的运动学方程。

关于 D-H 法，中文教材有蔡自兴老师的《机器人学》这本教材，说起来，我教《机器人原理与技术》这门课有近十年了，刚开始两三年，一直都是用的蔡老师的教材，那时候国内的教材实在太少了，对于机械臂的运动学一直都是一知半解，随着自己的教学和研发，慢慢才发觉原来光是机械臂的运动学都是非常复杂而且值得研究的东西。

机械臂的运动学基础学习推荐大家去翻 John J.Craig 的《机器人学导论》，这本书也是推荐每章的作业用 matlab robotics toolbox 来做作业。实际上，D-H 参数是有两种标定方式的，一种是标准的 D-H(Standard D-H)参数法，还有一种是改进的 D-H(Modified D-H)参数法，大部分书上现在都用到的是改进的 D-H 法，但也有一些书上用的是标准的 D-H 法，这里有两张图（见图 4.1），我们大致简略介绍一下它们的区别。

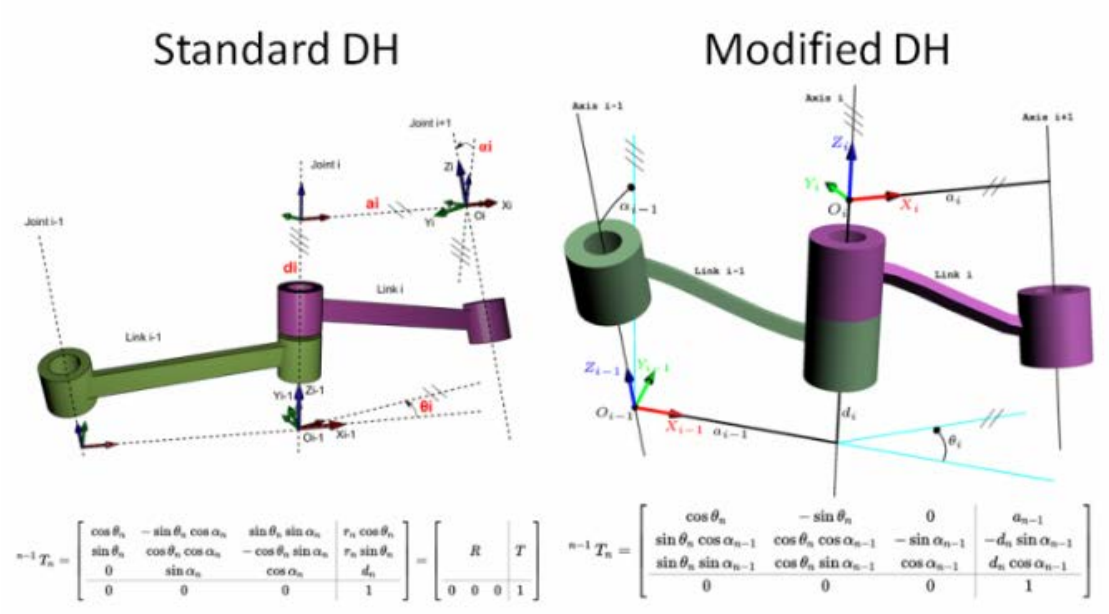

图 4.1 Standard D-H 与 Modified D-H

Modified D-H（MDH）与 Standard D-H（SDH）的区别在于我们将奇形怪状的机械臂物理结构转换为简化连杆与关节时，1）建立的坐标系有所不同（坐标原点不同，X 轴方向不同），2）导致 D-H 参数不一样，3）连杆与坐标系之间的变

换关系不一样。最根本的原因在于标准 D-H 将连杆坐标系建立在连杆输出端，也就是下一个关节上，其坐标系O_{i-1}与关节 *i* 对齐。而改进 D-H 将连杆坐标系建立在**连杆输入端**，也就是上一个关节上，其坐标系O_i与关节 *i* 对齐。设想一下，如果一个连杆有两个输出，那么 SDH 就会产生歧义，采用改进 D-H 最根本的原因就在这。

表 4.1 Modified D-H 与 Standard D-H 区别

不同	Standard D-H	Modified D-H
连杆坐标系（原点）建立	以连杆输出端（后一个关节）为固连坐标系	以连杆输入端（前一个关节）为固连坐标系
X 轴方向	Z_{i-1}轴=关节 i 指向 X_{i-1}轴沿关节 i 和关节 i+1 公垂线，或者垂直于关节 i 和关节 i+1 所在的平面 右手法则确定 Y*i-1* 轴	Z_i轴=关节 i 指向 X_i轴沿关节 i 和关节 i+1 公垂线，或者垂直于关节 i 和关节 i+1 所在的平面 右手法则确定 Y*i* 轴
相连关节之间的变换顺序	θ，d，a，α	α，a，θ，d

Standard D-H 方法相连关节转换矩阵如式 4-1。

$$^{i-1}A_i = Rot_{z,\theta_i} Trans_{z,d_i} Trans_{x,a_i} Rot_{x,\alpha_i} \quad 4\text{-}1$$

Modified D-H 方法相连关节转换矩阵如式 4-2。

$$^{n-1}T_n = \mathrm{Rot}_{x_{n-1}}(\alpha_{n-1}) \cdot \mathrm{Trans}_{x_{n-1}}(a_{n-1}) \cdot \mathrm{Rot}_{z_n}(\theta_n) \cdot \mathrm{Trans}_{z_n}(d_n) \quad 4\text{-}2$$

后来有人改进 Modified D-H 表示法（Hayati S, Mirmirani M. Improving the absolute positioning accuracy of robot manipulators[J]. Journal of Robotic Systems, 1985, 2(4): 397-413.），解决机械臂的精度标定问题。本实验教程不详细讲解这个区别，一般来说，我们现在通常使用 Modified D-H 方法，后面的建模以及运动学都是基于 Modified D-H 表示法。上一章图 3.1 中，坐标系的建立采用的就是 Modified D-H 表示法。

根据上一章图 3.1 中建立的 D-H 坐标系求取 D-H 参数，见图 4.2，确定它们

在 rviz 或 gazebo 中的空间位姿关系。

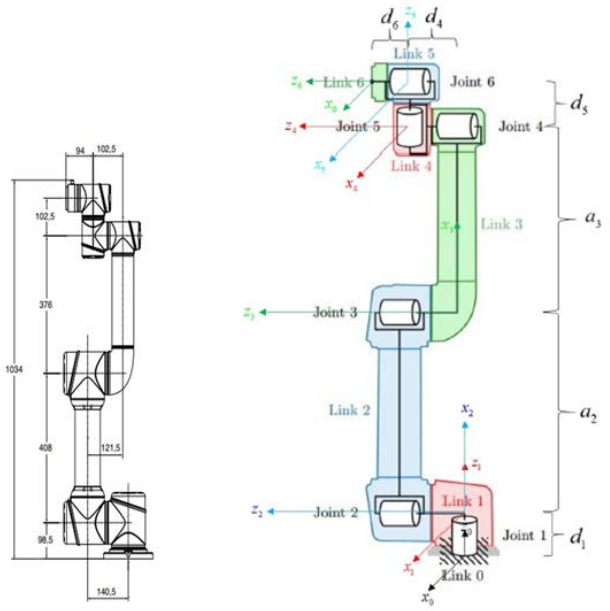

图 4.2 AUBO i5 求取 D-H 参数示意图（右边图是 UR 的，反正形状差不多，这里借用一下）

改进 D-H 四个参数的求法如下：

(1) theta：绕 Zi 轴，从 Xi-1 旋转到 Xi 的角度；

(2) d：沿 Zi 轴，从 Xi-1 移动到 Xi 的距离；

(3) a：沿 Xi 轴，从 Zi 移动到 Zi+1 的距离；

(4) alpha：绕 Xi 轴，从 Zi 旋转到 Zi+1 的角度。

通俗的理解，我们可以认为参数'theta'代表关节角，参数'd'代表横距，参数'a'代表杆件长度，参数'alpha'代表扭转角。

AUBO-i5 的 D-H 参数表如表 4.2 所示。

表 4.2 AUBO-i5 DH 参数表

i	α_{i-1}	a_{i-1}	d_i	θ_i 限制
0	0	0	-	–
1	π /2	0	0.0985	θ_1: -175° 至 175 °

2	0	-0.408	0	θ_2: -175°至 175°
3	0	-0.376	0	$-\theta_3$: -175°至 175°
4	π/2	0	0.1025	θ_4: -175°至 175°
5	-π/2	0	0.1025	θ_5: -175°至 175°
6	-	-	0.094	θ_6: -175°至 175°

Modified D-H 表示法，相连连杆之间的变换矩阵可以通过式 3-2 的变换顺序进行四个齐次变换，得到一个 T 矩阵，其通用表达形式见式 3-3。

$$ {}^{i-1}T_i = \begin{pmatrix} \cos\theta_i & -\sin\theta_i & 0 & a_{i-1} \\ \sin\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -\sin\alpha_{i-1} & -\sin\alpha_{i-1}\,d_i \\ \sin\theta_i\sin\alpha_{i-1} & \cos\theta_i\sin\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}\,d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad 4\text{-}3 $$

那么，对于末端 TCP，它与连杆 i-1 坐标系的关系由 6 个 T 矩阵表达，见式 4-4。

$$ {}^{i\text{-}1}T_6 = T_1T_2T_3T_4T_5T_6 \quad 4\text{-}4 $$

Denavit-Hartenberg parameters

Joint	Type	a	α	d	θ	Offset
1	Revolute	0.00000	π/2	0.089159	q1	0.00
2	Revolute	-0.42500	0.00	0.00000	q2	-π/2
3	Revolute	-0.39225	0.00	0.00000	q3	0.00
4	Revolute	0.00000	π/2	0.10915	q4	-π/2
5	Revolute	0.00000	-π/2	0.09465	q5	0.00
6	Revolute	0.00000	0.00	0.0823	q6	0.00

图 4.3 UR5 的 D-H 参数表（放这里参考一下）

这个D-H参数表怎么来的呢，希望同学们亲自动手画图和计算一下，不要copy一下就完，自己动手加深理解。

第 1 步，画连杆和轴简化图，见图 4.4。

把连杆这样展开便于后面画坐标系以及变换看起来空间感会明朗一些。上一

章图 3.1 是将机械臂整个立起来，画坐标系计算 DH 参数，这样机械臂的各关节零位设置就是立起来的样子，但看起来不是很直观，初学建议如图 4.4 这样展开，空间感更强一些，便于理解。（这样的话，第 2 个关节和遨博 i5 的零位相差 90 度，第 3 个关节方向反向，第 4 个关节也相差 90 度，所以后面需要调整这三个关节角）

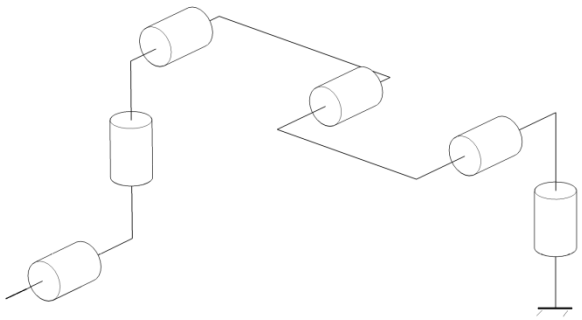

图 4.4 连杆和轴简化图

第 2 步，画第一个坐标系，见图 4.5。

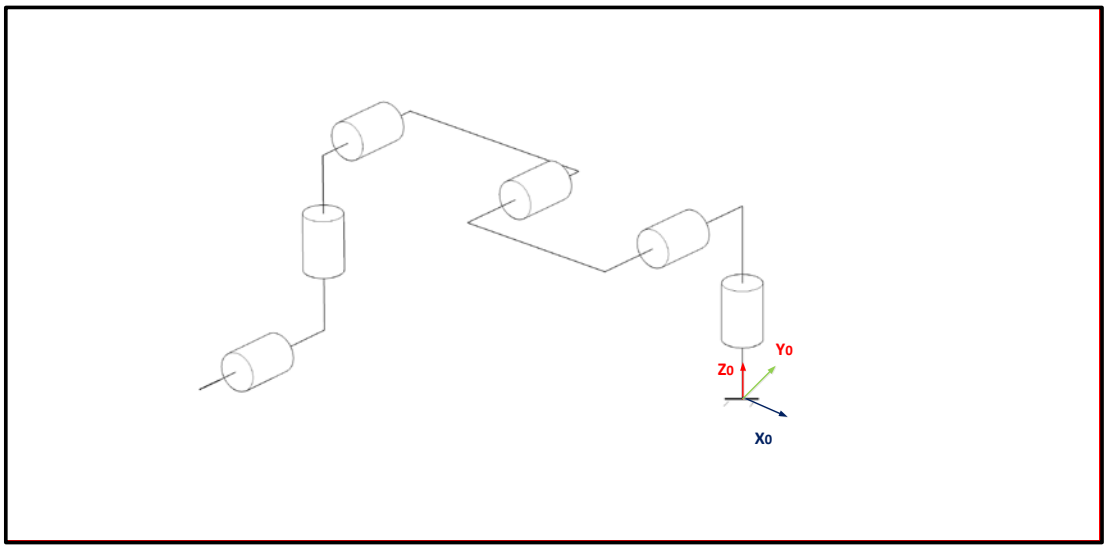

图 4.5 第一个固定坐标系

Modified DH 表示法，**将连杆坐标系的原点建立在关节的输入端，**其第一个连杆坐标系一般和世界坐标系重合。

（1）找出连杆 0 和连杆 1 的共垂线或交点，以连杆 0 和连杆 1 的交点，或共垂线与与连杆 0 的交点作为坐标系{0}的原点；

（2）规定 Z0 沿关节 0 的输出方向；

（3）规定 X0 沿共垂线由关节 0 指向关节 1。如果关节 0 和关节 1 平行，则 X0 垂直于关节 0 和关节 1 所在的平面；

（4）按照右手法则确定 Y0 的方向；

（5）对于坐标系{i}， 其原点和 Xi 的方向是可以任意选取的。但在选取过程中，为了后面的矩阵运算简单化，建议尽量使连杆长度参数 d 为 0。

按此步骤，我们依次建立 6 个连杆坐标系，如图 4.6 所示。该图的特点是将{1}{2}{3}坐标系的原点沿第 1 根轴的中心线与各轴的交点上，这样的方式计算 D-H 参数简单一些。坐标系{4}原点建立在轴 4 与轴 5 的交点上，坐标系{5}原点建立在轴 5 与轴 6 的交点上，坐标系{6}建立在末端工具中心点 TCP 上。

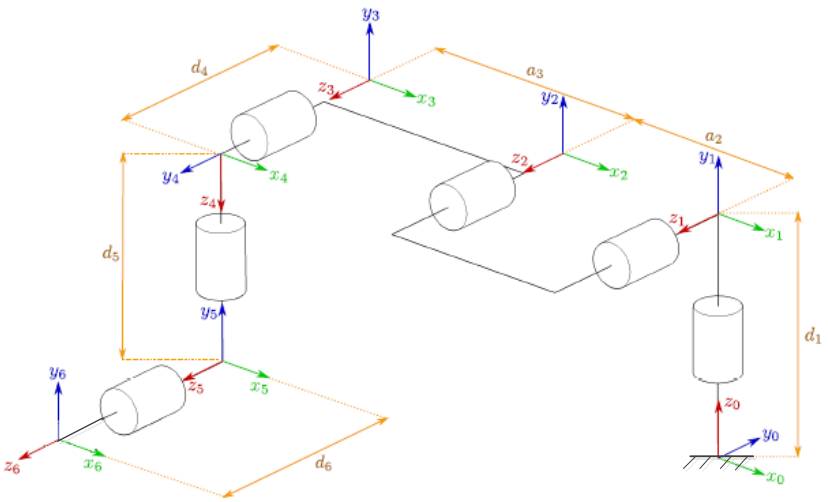

图 4.6 完全坐标系

有了这个图 4.6，我们开始计算四个 DH 参数，坐标系{0}对应世界坐标系，坐标系{1}开始我们计算：

(1) 计算连杆坐标系{1}的 D-H 参数

theta1: 绕 Z0 轴，将 X0 旋转到 X1 的角度，初始角为 0;

d1: 沿 Z1 轴，从 X0 移动到 X1 的距离，本实例中为向上提，d1=0.0985m;

a1: 沿 X1 轴，从 Z0 移动到 Z1 的距离，a1=0;

alpha1: 绕 x1 轴，从 Z0 到 Z1 的旋转角度，需要向左旋转 90 度，alpha1=PI/2;

(2) 计算连杆坐标系{2}的 D-H 参数

Theta2：绕 Z1 轴，将 X1 旋转到 X2 的角度，初始角比直立形态向里旋转了 90 度，设初始角度为 90;

d2：沿 Z2 轴，从 X1 移动到 X2 的距离，d2=0;

a2：沿 X2 轴，从 Z1 移动到 Z2 的距离，a2=0.408m;

alpha2：绕 x2 轴，从 Z1 到 Z2 的旋转角度， alpha2=0;

(3) 计算连杆坐标系{3}的 D-H 参数

theta3：绕 Z2 轴，将 X2 旋转到 X3 的角度，初始角 0;

d3：沿 Z3 轴，从 X2 移动到 X3 的距离，d3=0;

a3：沿 X3 轴，从 Z2 移动到 Z3 的距离，a3=0.376m;

alpha3：绕 x3 轴，从 Z2 到 Z3 的旋转角度， alpha3=0;

(4) 计算连杆坐标系{4}的 D-H 参数

theta4：关节角 4，向里转了 90 度，初始角 90;

d4：沿 Z4 轴，从 X3 移动到 X4 的距离，d4=0.1215mm;

a4：沿 X4 轴，从 Z3 移动到 Z4 的距离，a4=0;

alpha4：绕 x4 轴，从 Z3 到 Z4 的旋转角度， alpha4=PI/2;

(5) 计算连杆坐标系{5}的 DH 参数

theta5：关节角 5，初始角 0;

d5：沿 Z5 轴，从 X4 移动到 X5 的距离，d5=0.1025mm;

a5：沿 X5 轴，从 Z4 移动到 Z5 的距离，a5=0;

alpha5：绕 x5 轴，从 Z4 到 Z5 的旋转角度， alpha4=-PI/2;

(6) 计算连杆坐标系{5}的 DH 参数

theta6：关节角 6，初始角 0;

d6：沿 Z6 轴，从 X5 移动到 X6 的距离，d6=0.094mm;

a6：沿 X6 轴，从 Z5 移动到 Z6 的距离，a6=0;

alpha6：绕 x6 轴，从 Z5 到 Z6 的旋转角度， alpha6=0。

求完之后，我们即可得到 aubo-i5 的 MDH 参数。

4.3 AUBO-i5 的正解

针对串联机器人，在我的课程讲义里我们先讲了**3 关节**各种工业机器人：笛卡尔（又称台架，直角）坐标型（PPP）、圆柱坐标型（RPP）、球坐标型（PRR）、平面关节型（RRP，这个构型比较强，在这个构型基础上，改进成三个水平旋转 R 关节，以及一个平动 P 关节，机器人作业更灵活，又称 SCARA：Selective Compliance Assembly Robot Arm， 选择顺应性装配机器手臂机器人）、链式（拟人或全旋转）坐标型（RRR）或称串联关节型的正解与逆解，然后再将到**手腕关节**，从单自由度手腕、两自由度手腕到三自由度手腕的构型，再到通用型的六旋转关节串联机械臂，也是目前用的最多的构型，它的基础实际上就是在三关节全旋转链式机器人基础上加上三个腕部关节构成通用型工业机器人。

三关节机械臂的运动学比较简单，SCARA 的运动学也很简单，所以我们的实验**只讲 6 关节通用串联机器人的运动学实验**。

本节介绍的运动学正解方法涉及**标准 DH 法（SDH）、改进 DH 法（MDH）、旋量法（Screw theory）以及四元数（Quaternion）**等方法。

前面我们学习了 D-H 表示法，核心思想就是将机械人物理形态转换成抽象的连杆(link)和关节(joint)，然后用齐次变换矩阵将关节的转动或拉伸运动，采用旋转矩阵或平移矩阵来进行统一的表达，从一个坐标系变换到了下一个坐标系。

D-H 表达法中，在 n+1 和 n+2 坐标系间严格地按照同样的四个运动顺序（见公式 4-5，4-6）可以将一个坐标变换到下一个坐标系。重复这个步骤，就可以实现一系列相邻坐标系之间的变换。从参考坐标系开始，我们可以将其转换到机器人的基座，然后到第一个关节，第二个关节……，直至末端执行器。这里比较好的一点是，在任何两个坐标系之间的变换均可采用与前面相同的运动步骤。

通过右乘表示四个运动的四个矩阵就可以得到变换矩阵 A，矩阵 A 表示了四个依次的运动。由于所有的变换都是相对于当前坐标系的（即它们都是相对于当前的本地坐标系来测量与执行），因此所有的矩阵都是右乘。以 Standard D-H 为例，可以得到结果如下：

$$^{n}T_{n+1} = A_{n+1} = Rot(z,\theta_{n+1}) \times Tran(0,0,d_{n+1}) \times Tran(a_{n+1},0,0) \times Rot(x,a_{n+1})$$

$$= \begin{bmatrix} C\theta_{n+1} & -S\theta_{n+1} & 0 & 0 \\ S\theta_{n+1} & C\theta_{n+1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{n+1} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\times \begin{bmatrix} 1 & 0 & 0 & a_{n+1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha_{n+1} & -S\alpha_{n+1} & 0 \\ 0 & S\alpha_{n+1} & C\alpha_{n+1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4\text{-}5)$$

$$A_{n+1} = \begin{bmatrix} C\theta_{n+1} & -S\theta_{n+1}C\alpha_{n+1} & S\theta_{n+1}S\alpha_{n+1} & a_{n+1}C\theta_{n+1} \\ S\theta_{n+1} & C\theta_{n+1}C\alpha_{n+1} & -C\theta_{n+1}S\alpha_{n+1} & a_{n+1}S\theta_{n+1} \\ 0 & S\alpha_{n+1} & C\alpha_{n+1} & d_{n+1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4\text{-}6)$$

比如，一般机器人的关节 2 与关节 3 之间的变换可以简化为：

$$^{2}T_3 = A_3 = \begin{bmatrix} C\theta_3 & -S\theta_3C\alpha_3 & S\theta_3S\alpha_3 & a_3C\theta_3 \\ S\theta_3 & C\theta_3C\alpha_3 & -C\theta_3S\alpha_3 & a_3S\theta_3 \\ 0 & S\alpha_3 & C\alpha_3 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4\text{-}7)$$

在机器人的基座上，可以从第一个关节开始变换到第二个关节，然后到第三个……，再到机器人的手，最终到末端执行器。若把每个变换定义为 $^{i-1}T_i$，则可以得到许多表示变换的矩阵。在机器人的基座与手之间的总变换则为：

$$^{R}T_H = {}^{R}T_1\,{}^{1}T_2\,{}^{2}T_3 \cdots {}^{n-1}T_n = A_1A_2A_3 \cdots A_n \quad (4\text{-}8)$$

其中 n 是关节数。对于一个具有六个自由度的机器人而言，有 6 个 A 矩阵，写成一般形式就是式 4-6。

而 Modified D-H，坐标系{i}对应的 D-H 参数从 i-1 开始，变换矩阵有区别，其转换顺序是式 4-2，合成的一般变换矩阵是式 4-3，切记！！！

下面我们开始写 AUBO-i5 的正解。

从上面的 D-H 实验我们已经得到了 AUBO-i5 的 DH 参数，我们把它写入头文件，代码如下：

```
//Modified DH parameters

static double a0=0, a1=0, a2=-0.408, a3=-0.376, a4=0, a5=0, a6=0;

static double alpha0=0, alpha1=M_PI/2, alpha2=0, alpha3=0, alpha4=M_PI/2, alpha5=-M_PI/2, alpha6=0;
```

static double d1=0.0985, d2=0.0, d3=0.0, d4=0.1215, d5=0.1025, d6=0.094;

然后写一个准备运动学函数的 cpp 文件，定义一个将 D-H 参数变成转换矩阵的通用函数，**使用式 4-3**。这里我们直接使用 Eigen 库来完成矩阵运算，Eigen 库非常好用，它是一个高层次的 C++ 库，有效支持线性代数、矩阵和矢量运算、数值分析及其相关的算法，只需要包含头文件，不需要编译成库文件，即可在项目里使用，速度也很快，ROS 下的相关数学运算推荐使用 **Eigen3**，速度很快，效率高，也比较易于我们理解。这里不讲 Eigen 库如何加入 ROS 的包开发，本实验使用的 Eigen3，Eigen3 的使用教程请自行查阅文档学习。

```
Eigen::Matrix<double,4,4> TransMatrixMDH(double a, double alpha, double d, double theta) {

    Eigen::Matrix<double,4,4> T;

    T.row(0)<<cos(theta),-sin(theta), 0, a;

    T.row(1)<<sin(theta)*cos(alpha),cos(theta)*cos(alpha),-sin(alpha),-sin(alpha)*d;

    T.row(2)<<sin(theta)*sin(alpha),cos(theta)*sin(alpha),cos(alpha),cos(alpha)*d;

    T.row(3)<<0,0,0,1;


    return T;

}
```

接下来，我们写正运动学函数，代码如下：

```
Eigen::Matrix<double,4,4> ForwardKinematics(double theta1,double theta2,double theta3,double theta4,double theta5,double theta6){

    Eigen::Matrix<double,4,4> T1, T2, T3, T4, T5, T6;

    Eigen::Matrix<double,4,4> T = Eigen::Matrix<double,4,4>::Identity();

    T1=TransMatrixMDH(a0,alpha0,d1,theta1);

    T2=TransMatrixMDH(a1,alpha1,d2,theta2);

    T3=TransMatrixMDH(a2,alpha2,d3,theta3);

    T4=TransMatrixMDH(a3,alpha3,d4,theta4);
```

```
    T5=TransMatrixMDH(a4,alpha4,d5,theta5);

    T6=TransMatrixMDH(a5,alpha5,d6,theta6);

    T=T6; T=T5*T; T=T4*T; T=T3*T; T=T2*T; T=T1*T;


    return T;

}
```

这里我们使用的都是 double 数据类型，根据精度要求，如果不需要这么高的精度，可以使用 float 数据类型。

接下来我们在应用程序里写一个测试，来验证看看是否正确。

主函数里面测试代码如下：

```
            printf("Please input 6 joint angle:\n");

            scanf("%lf%lf%lf%lf%lf%lf",&j0,&j1,&j2,&j3,&j4,&j5);

            q0 = j0*M_PI/180.0;

            q1 = j1*M_PI/180.0;

            q2 = j2*M_PI/180.0;

            q3 = j3*M_PI/180.0;

            q4 = j4*M_PI/180.0;

            q5 = j5*M_PI/180.0;


            printf("%lf %lf %lf %lf %lf %lf\n", q0,q1,q2,q3,q4,q5);

            moveJ(q0,q1,q2,q3,q4,q5);


    TcpCenter=ForwardKinematics(q0,q1-M_PI/2.0,-q2,q3-M_PI/2.0,q4,q5);

            std::cout<<TcpCenter<<std::endl;

            pointX=TcpCenter(0,3);

            pointY=TcpCenter(1,3);
```

```
pointZ=TcpCenter(2,3);
printf("%lf %lf %lf\n",pointX,pointY,pointZ);
points.header.stamp = ros::Time::now();
points.action = visualization_msgs::Marker::DELETE;//delete not work
points.points.clear();//work to clear
marker_pub.publish(points);

points.action = visualization_msgs::Marker::ADD;
points.lifetime=ros::Duration();
points.scale.x = 0.01f;
points.scale.y = 0.01f;
points.scale.z = 0.01f;
points.color.r = 1.0f;
points.color.g = 0.0f;
points.color.b = 0.0f;
points.color.a = 1.0;
DisplayPoint(pointX,pointY,pointZ);
```

在测试代码中，我们让用户输入 6 个关节的角度，转换成弧度，然后使用上一章中我们编写的 moveJ 函数将机械臂按 6 个关节运动，然后我们用正解函数解算出来一个 4x4 的矩阵，也就是末端 TCP 的位姿矩阵，将 x/y/z 值取出来，用 visualization_msgs::Marker 发布一个画点的消息给 rviz，看看我们解算的点和模型直接驱动的运动是否一致。测试结果如图 4.7 所示。

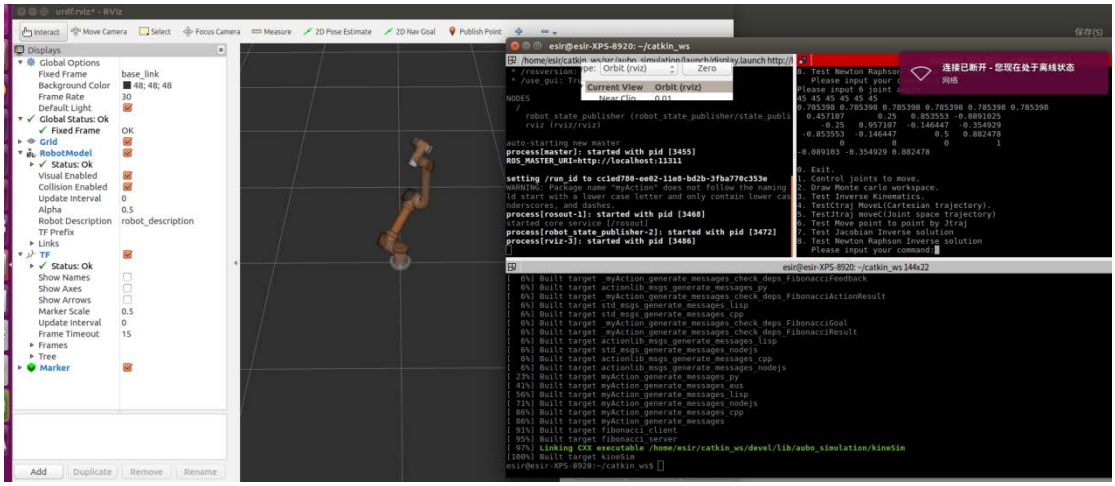

图 4.7 正解测试图

放大看看，可以看到正解出来的标记红点始终在 TCP 上（由于遨博 github 上给的图纸模型有一点小小的尺寸误差，所以那个红点看起来始终不在 TCP 中心，见图 4.8，但经过反复验证，这个不是正解的错误，是机械图纸标定的误差，关系不大）。

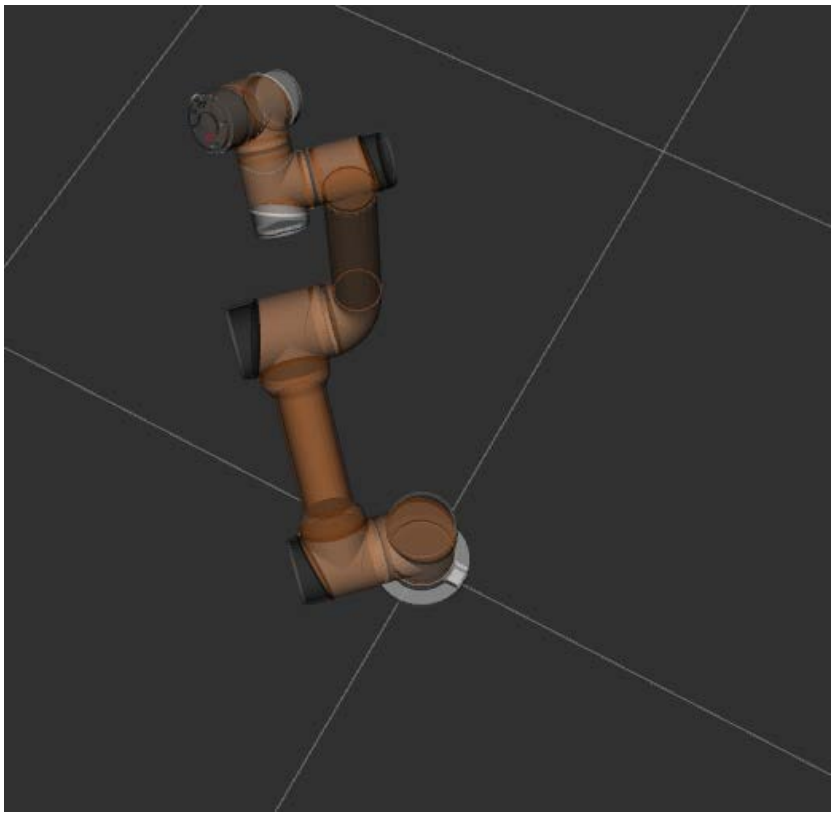

图 4.8 正解红点标记放大图

反复做实验，红点始终标记在正确位置，可以看到我们的正解函数和 rviz 的

joint_state 消息驱动模型是完全一致的，证明了我们的这个正解是完全正确的。

上面讲的是一般教科书上的正解，使用 SDH 还是 MDH 我在前面已经讲过，如果机械臂满足 Pieper 准则时（机器人的三个相邻关节轴交于一点或三轴线平行），会涉及到后面我们讲的逆运动学是否有封闭解，但正运动学没有关系。对于单运动链的串联机械臂，MDH 和 SDH 都适用，读者可以参照自行编写使用 SDH 表示法的正解，如果机械臂的基坐标系和全局坐标系一致的情况下，这两种矩阵变换的结果实际是一样的。

除了上面两种求正解的方法，这里再介绍一种旋量法求正解的实验。

如图 4.9 所示，我们建立 6 个旋转轴(screw axis)，假设图 4.9 为每个旋转轴的零位位置，那么终端工具坐标系{b}在零位状态下的初始矩阵见式 4-9。

$$M = \begin{bmatrix} 1 & 0 & 0 & -(L1+L2) \\ 0 & 0 & -1 & -(W1+W2) \\ 0 & 1 & 0 & H1-H2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad 4\text{-}9$$

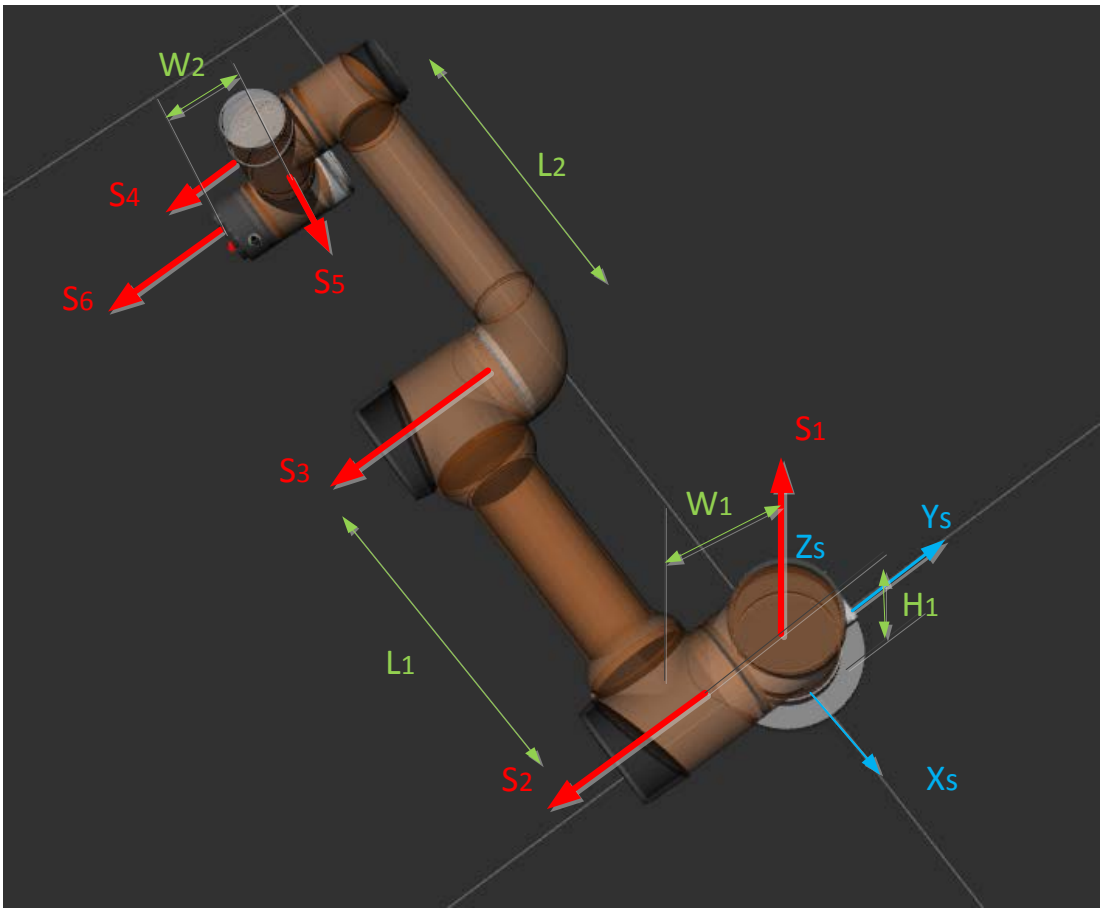

图 4.9 AUBO-i5 初始零位，右手法则表示正转，W1 为 S1 旋转轴与 S5 旋转轴之间的距离，L1=0.408m，L2=0.376m，W1=121.5mm，W2=94mm，H1=98.5mm

旋转轴的$S_i = (\omega_i, \nu_i)$如表 4.3 所示：

表 4.3 各旋转轴向量表

i	ω_i	ν_i
1	（0，0，1）	（0，0，0）
2	（0，-1，0）	（H1，0，0）
3	（0，-1，0）	（H1，0，L1）
4	（0，-1，0）	（H1，0，L1+L2）
5	（0，0，-1）	（W1，-（L1+L2），0）
6	（0，-1，0）	（H1-H2，0，L1+L2）

AUBO-i5 每个关节都是旋转关节，其运动旋量(twist)坐标为：

$$V = -\omega x q, \xi = \begin{bmatrix} \upsilon \\ \omega \end{bmatrix} \in R^{6x1} \quad 4\text{-}11$$

AUBO-i5 有六个旋转轴，假设六个关节的旋转角为θ1，θ2，θ3，θ4，θ5，θ6，则末端工具坐标系的姿态矩阵 T可以由 POE 指数积公式 4-10 表达。

$$T(\theta) = e^{[S1]\theta 1} e^{[S2]\theta 2} e^{[S3]\theta 3} e^{[S4]\theta 4} e^{[S5]\theta 5} e^{[S6]\theta 6} M \quad 4\text{-}10$$

这个 POE 指数积公式就是正解，将每个关节的旋转角度代入 POE 公式，即可得到末端关节的姿态矩阵。

下面我们来编程实现，主要函数如下：

Exp_twist 函数将每个旋转轴的转动角代入，求出相连关节之间的运动旋量 twist，旋转轴向量 Ti={ν_i，ω_i}，根据表 4.3 其定义如下：

```
Eigen::Matrix<double, 1, 6> T0, T1, T2, T3, T4, T5;

T0 << 0, 0, 0, 0, 0, 1;

T1 << 98.5, 0, 0, 0, -1, 0;

T2 << 98.5, 0, 408,0, -1, 0;

T3 << 98.5, 0, 784,0, -1, 0;

T4 << 121.5, -784, 0, 0, 0, -1;
```

```
T5 << -4, 0,784, 0, -1, 0;
```

初始姿态矩阵定义如下：

```
Eigen::Matrix4d gz;

gz(0,0)=1; gz(0,1)=0; gz(0,2)=0; gz(0,3)= -784;

gz(1,0)=0; gz(1,1)=0; gz(1,2)=-1; gz(1,3)= -215.5;

gz(2,0)=0; gz(2,1)=1; gz(2,2)=0; gz(2,3)= -4;

gz(3,0)=0; gz(3,1)=0; gz(3,2)=0; gz(3,3)=1;
```

相连旋转轴之间的运动旋转 twist 函数如下：

```
Eigen::Matrix4d Exp_twist(Eigen::Matrix<double,1,6> T, double theta){

    Eigen::Matrix4d R;

    Eigen::Matrix3d what,eye3;

    eye3.setIdentity(3,3);

    double p[3];

    Eigen::Matrix<double,3,1> v, w,z;


    v.block<3,1>(0,0) = T.block<1,3>(0,0);

    w.block<3,1>(0,0) = T.block<1,3>(0,3);


    p[0] = w(0,0);

    p[1] = w(1,0);

    p[2] = w(2,0);

    what = w_hat(p);

    z = Eigen::Matrix<double,3,1>::Zero();


    if(sum3v(w,z)!=3){
```

```
            R.block<3,1>(0,3)                                                    =
((eye3-exp_rot(theta,what))*(w.cross(v))+(w*(w.transpose())*v*theta));
        }else{
            R.block<3,1>(0,3) = theta*v;
        }

        R.block<3,3>(0,0) = exp_rot(theta,what);
        R.block<1,3>(3,0) = z.transpose();
        R(3,3) = 1;

        return R;
    }
    Eigen::Matrix3d exp_rot(double theta, Eigen::Matrix3d what){
        Eigen::Matrix3d R;
        Eigen::Matrix3d eye3;
        eye3.setIdentity(3,3);
        R = eye3+what*sin(theta)+what*what*(1-cos(theta));

        return R;
    }

    int sum3v(Eigen::Matrix<double,3,1> w, Eigen::Matrix<double,3,1> z){
        int same=0;
        if(w(0)==z(0))same=1;
        if(w(1)==z(1))same=same+1;
```

```
        if(w(2)==z(2))same=same+1;
        return same;
    }
    Eigen::Matrix<double,3,3> w_hat( double *p){
        Eigen::Matrix3d what;
        what = Eigen::Matrix3d::Zero();
        what(0,1) = - p[2];
        what(0,2) = p[1];
        what(1,2) = -p[0];
        what(1,0) = p[2];
        what(2,0) = -p[1];
        what(2,1) = p[0];


        return what;
    }
```

测试一下,可以使用如下语句来得到这个旋量法正解：

```
T = Exp_twist(T0,q0) * Exp_twist(T1,q1-M_PI/2.0) * Exp_twist(T2,-q2) * Exp_twist(T3,q3-M_PI/2.0) * Exp_twist(T4,q4) * Exp_twist(T5,q5)) * gz;
```

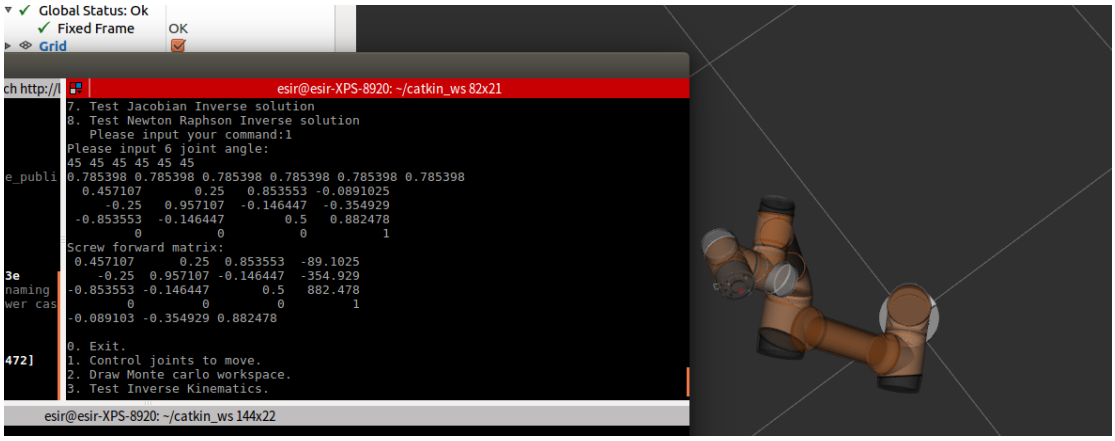

图 4.10 测试旋量法正解

从图 4.10 中，我们可以看到，除了坐标一个用的 mm，一个用的 m 之外，旋量法和 MDH 求得的正解完全一样，位置和姿态完全一致。

除了上面我们介绍的三种求正解方法，还有采用四元数求运动学的方法，本实验不讲这个，可以根据讲义自行实现。

4.4 AUBO-i5 的工作空间（Work space）

机器人的工作空间有三种类型：

可达工作空间(reachable workspace)，即机器人末端可达位置点的集合；

灵巧工作空间(dexterous workspace)，即在满足给定位姿（除坐标可达外，姿态必须满足工艺要求）范围时机器人末端可达点的集合；

全工作空间(global workspace)，即给定所有位姿时机器人末端可达点的集合。

有关机器人工作空间，我在讲义里在机械臂正解这一章只讲笛卡尔空间，但在机械臂逆解那一章还讲了笛卡尔空间以及关节空间。在移动机器人路径规划中还介绍了 C-space，这些概念请仔细阅读讲义，加以区分。

这里涉及的机械臂工作空间求解，只涉及全局坐标系下（笛卡尔）的位姿求解，针对的是末端工具 TCP 的可达位姿或灵巧位姿。工作空间的求解及分析，可以有助于机械臂构型优化设计，但本实验只讲基本原理实验，具体如何优化请查阅相关研究。

机械臂的工作空间求解算法最基本的方法有几何图解法、数值法以及解析法等，目前计算机的计算与绘图能力强大，所以一般几何图解法、复杂的解析法都不需要了，我们可以直接使用合适的运动学正解，采用蒙特卡洛法直接绘制出各种工作空间。计算机目前计算速度快，存储足够大，工作空间中的姿态和坐标都可以快速求取，也可以将数据存取起来（同时具有坐标和姿态），进行相关分析，去进行作业分析或指导结构优化。

蒙特卡洛法利用随机数原理得到生成关节角公式 $\theta=(\theta_{\max}-\theta_{\min})R\text{and}（\text{N},1）+\theta_{\min}$ 获得有限个关节角的随机值，进而求出末端的位置点。将这些点在空间中描绘出来，即为机器人的工作空间。

下面给出简单的蒙特卡洛法代码：

```
int Monte_carlo_Workspace(){
```

```
int N=150000;
double t1[N],t2[N],t3[N],t4[N],t5[N],t6[N];
Eigen::Matrix<double,4,4>T1,T2,T3,T4,T5,T6,T;
srand(time(0));
for(int i=0;i<N;i++){
    t1[i]=joint1Lim_min+(joint1Lim_max-joint1Lim_min)*random(N)/N;
    t2[i]=joint2Lim_min+(joint2Lim_max-joint2Lim_min)*random(N)/N;
    t3[i]=joint3Lim_min+(joint3Lim_max-joint3Lim_min)*random(N)/N;
    t4[i]=joint4Lim_min+(joint4Lim_max-joint4Lim_min)*random(N)/N;
    t5[i]=joint5Lim_min+(joint5Lim_max-joint5Lim_min)*random(N)/N;
    t6[i]=joint6Lim_min+(joint6Lim_max-joint6Lim_min)*random(N)/N;
}

for(int i=0;i<N;i++){
    T1=TransMatrixMDH(a0,alpha0,d1,t1[i]);
    T2=TransMatrixMDH(a1,alpha1,d2,t2[i]-M_PI/2.0);
    T3=TransMatrixMDH(a2,alpha2,d3,-t3[i]);
    T4=TransMatrixMDH(a3,alpha3,d4,t4[i]-M_PI/2.0);
    T5=TransMatrixMDH(a4,alpha4,d5,t5[i]);
    T6=TransMatrixMDH(a5,alpha5,d6,t6[i]);
    T=T1*T2*T3*T4*T5*T6;

    double pointX=T(0,3);
    double pointY=T(1,3);
    double pointZ=T(2,3);
```

```
            DisplayPoint(pointX,pointY,pointZ);
        }
        return 0;
    }

    int DisplayPoint(double x,double y,double z){
        geometry_msgs::Point p;
        p.x = x;
        p.y = y;
        p.z = z;
        points.header.stamp = ros::Time::now();
        points.points.push_back(p);

        //发布 point marker
        marker_pub.publish(points);

        return 0;
    }
```

我在我的机器上测试了 N=150000 共 15 万个 6 关节角的随机组合，采用 MDH 法求正解，并在 rviz 下绘制点云图，大概耗费时间在 7 分钟左右。（机器参数一般，这里就不贴出来了）

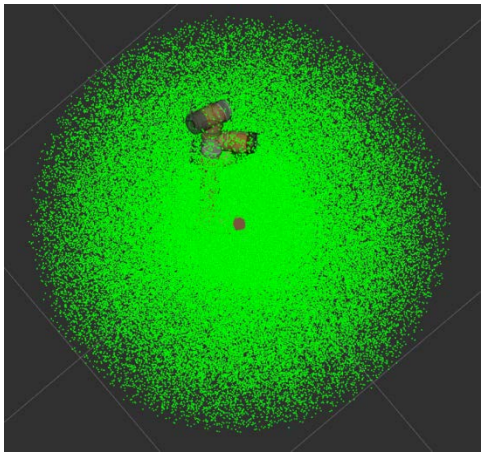

图 4.11 蒙特卡洛法绘制工作空间

从图 4.11 中，我们可以看出，AUBO-i5 在六个关节角的-175℃-175℃的限制范围内，其工作空间为除去底部圆柱体的一个球形工作空间。当然这里我们没有考虑关节连杆的物理干涉，也没有考虑姿态。可以对这个点云数据进行分析，加以末端姿态约束以及连杆干涉约束，即可得到灵巧工作空间。利用这个点云数据，后面也可以进行查表或者智能学习算法，快速得出满足各种约束要求的逆解或工作空间分析。

正运动学的实验介绍到此，希望同学们完成这些实验后，有能力的同学能运用这些基本原理，进行机械臂的结构优化分析。

4.5 本章版权及声明

本章内容由闵华松(mhuasong@wust.edu.cn)编写。