机器人学课程设计

姓名：朱楷文
学号：520030910178
班级：F2003001

1 机械臂构建

1.1 实现思路

Puma560 机械臂的 D-H 参数如表 1 所示, 其中 $\theta_i,\ i = 1, 2, \cdots, 6$ 为关节变量, a_2, d_3, a_3, d_4 为定值. 本设计中, 各关节变量的取值范围均为 $(-90°, 90°)$, a_2, d_3, a_3, d_4 分别设为 100, 20, 10, 100. 据此建立各连杆并将其连接, 即可得到机械臂.

表 1: Puma560 机械臂的 D-H 参数表

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	$-90°$	0	0	θ_2
3	0	a_2	d_3	θ_3
4	$-90°$	a_3	d_4	θ_4
5	$90°$	0	0	θ_5
6	$-90°$	0	0	θ_6

1.2 结果

构建出的机械臂如图 1 所示.

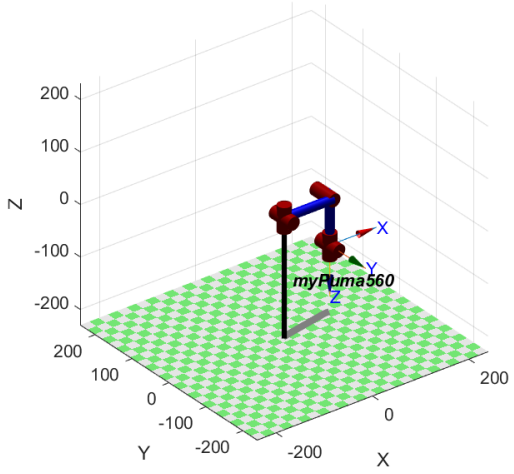


图 1: 机械臂示意图

1.3 代码

构建机械臂的函数代码如下.

```
function my_p560 = create_robot(a2,d3,a3,d4)
    % set the D-H parameters
    qlim = [-pi/2, pi/2];
    theta(1) = 0; d(1) = 0;   a(1) = 0;   alpha(1) = 0;
    theta(2) = 0; d(2) = 0;   a(2) = 0;   alpha(2) = -pi/2;
    theta(3) = 0; d(3) = d3;  a(3) = a2;  alpha(3) = 0;
    theta(4) = 0; d(4) = d4;  a(4) = a3;  alpha(4) = -pi/2;
    theta(5) = 0; d(5) = 0;   a(5) = 0;   alpha(5) = pi/2;
    theta(6) = 0; d(6) = 0;   a(6) = 0;   alpha(6) = -pi/2;
    % create the links
    L(1) = Link([theta(1), d(1), a(1), alpha(1), 0], 'modified');
    L(2) = Link([theta(2), d(2), a(2), alpha(2), 0], 'modified');
    L(3) = Link([theta(3), d(3), a(3), alpha(3), 0], 'modified');
    L(4) = Link([theta(4), d(4), a(4), alpha(4), 0], 'modified');
    L(5) = Link([theta(5), d(5), a(5), alpha(5), 0], 'modified');
    L(6) = Link([theta(6), d(6), a(6), alpha(6), 0], 'modified');
    L(1).qlim = qlim;
    L(2).qlim = qlim;
    L(3).qlim = qlim;
    L(4).qlim = qlim;
    L(5).qlim = qlim;
    L(6).qlim = qlim;
    % create the robot
    my_p560 = SerialLink(L, 'name', 'myPuma560');
end
```

2 工作空间可视化

2.1 实现思路

机械臂的工作空间是末端执行器可达到的空间范围, 即, 对于工作空间中的一个点 $\mathbf{p}$, 在关节空间中存在一个点 $\mathbf{q}$, 使得 $\mathbf{f}(\mathbf{q}) = \mathbf{p}$, 其中 $\mathbf{f}$ 为正运动学函数. 因此, 对关节空间中的每个点应用正运动学函数, 即可得到工作空间.

实际操作中, 为了可视化工作空间, 由于不可能遍历关节空间中的所有点, 我们在其中进行均匀的随机采样, 对采样得到的每个点应用正运动学函数, 绘制出得到的位置. 若样本足够代表整个关节空间, 即可实现工作空间的可视化. 本设计选择样本容量为 20000.

2.2 结果

构建机械臂, 采样并计算后得到的工作空间如图 2 所示, 其中每个红点表示采样并计算后得到的工作空间中的一个点. 工作空间在 X-Y 平面, Y-Z 平面, Z-X 平面上的投影如图 3 所示.

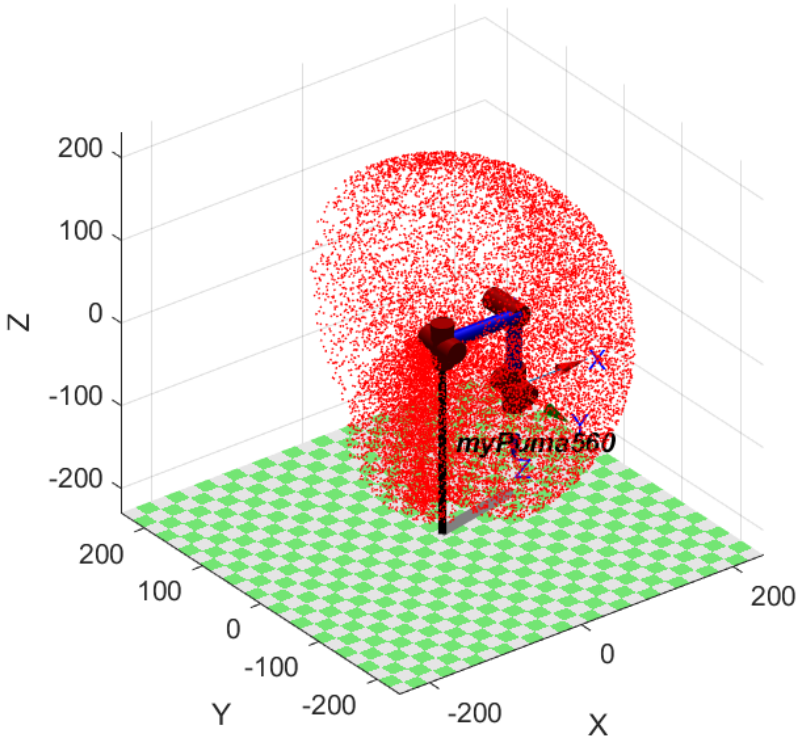
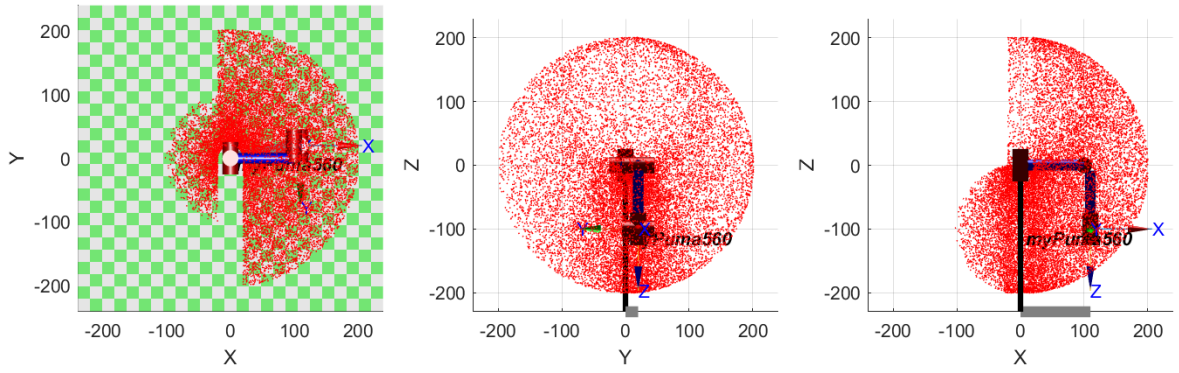


图 2: 机械臂的工作空间



(a) X-Y 平面上的投影　(b) Y-Z 平面上的投影　(c) Z-X 平面上的投影

图 3: 工作空间在各平面上的投影

2.3 代码

可视化工作空间的代码如下.

```m
clear; close all; clc;

% create the robot
my_p560 = create_robot(100,20,10,100);

% plot the robot
my_p560.plot(zeros(1,6));
hold on;

% sample parameters randomly to get the workspace
N = 20000;  % num of samples
for i = 1:N
    T = my_p560.fkine(-pi/2 + rand(1, 6) * pi);
    pos = transl(T);
    x(i) = pos(1);
    y(i) = pos(2);
    z(i) = pos(3);
end

% plot the positions
scatter3(x,y,z, 1, 'r', 'filled');
```

3 避障路径规划

3.1 实现思路

欲控制机械臂避开障碍物从起点到达终点, 只需观察出安全的末端执行器路径, 选取合适的锚点, 利用逆运动学计算出各锚点对应的关节变量, 在这些关节变量之间进行平滑的插值, 最后令机械臂按照插值结果运动即可.

本设计中, 障碍物为长方体, 中心为 (100, 0, 50), 大小为 (100, 30, 150), 路径的起点为 (100, 100, 10), 终点为 (100, -100,10). 因此取锚点为: 起点 (100, 100, 10), 起点正下方 (100, 100, -50), 终点正下方 (100, -100, -50), 终点 (100, -100, 10). 基座位置设为 (0, 0, -130). 但是, 在 Matlab 中调用函数 `ikine` 求解逆运动学时出现了无法收敛的错误. 推测这是由于传入的末端执行器朝向无法达到. 可达的朝向是难以获知的, 而事实上, 这里我们只关心末端执行器的位置而并不关心其朝向, 而位置仅由前三个关节变量决定. 因此考虑推导其前三个关节变量的解析解, 后三个关节变量设为 0 即可.

给定末端位置 (x, y, z), 下面求解其逆运动学.

机械臂前 4 个关节的齐次变换矩阵计算如下, 其中 θ_4 设为 0.

$$
{}^0T_1 = \begin{pmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad {}^1T_2 = \begin{pmatrix} c_2 & -s_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},
$$

$$
{}^2T_3 = \begin{pmatrix} c_3 & -s_3 & 0 & 100 \\ s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 20 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad {}^3T_4 = \begin{pmatrix} 1 & 0 & 0 & 10 \\ 0 & 0 & 1 & 100 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.
$$

进而可得

$$
{}^0T_2 = {}^0\ T_1^1T_2 = \begin{pmatrix} c_1c_2 & -c_1s_2 & -s_1 & 0 \\ s_1c_2 & -s_1s_2 & c_1 & 0 \\ -s_2 & -c_2 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},
$$

$$
{}^0T_3 = {}^0\ T_2^2T_3 = \begin{pmatrix} c_1c_{23} & -c_1s_{23} & -s_1 & 100c_1c_2 - 20s_1 \\ s_1c_{23} & -s_1s_{23} & c_1 & 100s_1c_2 + 20c_1 \\ -s_{23} & -c_{23} & 0 & -100s_2 \\ 0 & 0 & 0 & 1 \end{pmatrix},
$$

$$
{}^0T_4 = {}^0\ T_3^3T_4 = \begin{pmatrix} c_1c_{23} & -c_1s_{23} & -s_1 & 10c_1c_{23} - 100c_1s_{23} + 100c_1c_2 - 20s_1 \\ s_1c_{23} & -s_1s_{23} & c_1 & 10s_1c_{23} - 100s_1s_{23} + 100s_1c_2 + 20c_1 \\ -s_{23} & -c_{23} & 0 & -10s_{23} - 100c_{23} - 100s_2 \\ 0 & 0 & 0 & 1 \end{pmatrix},
$$

由于后三个关节变量不影响末端执行器位置, 因此末端位置为

$$
\mathbf{p} = \begin{pmatrix} 10c_1c_{23} - 100c_1s_{23} + 100c_1c_2 - 20s_1 \\ 10s_1c_{23} - 100s_1s_{23} + 100s_1c_2 + 20c_1 \\ -10s_{23} - 100c_{23} - 100s_2 \end{pmatrix}.
$$

于是可以列出方程组

$$
x = 10c_1c_{23} - 100c_1s_{23} + 100c_1c_2 - 20s_1 \tag{1a}
$$

$$
y = 10s_1c_{23} - 100s_1s_{23} + 100s_1c_2 + 20c_1 \tag{1b}
$$

$$
z = -10s_{23} - 100c_{23} - 100s_2 \tag{1c}
$$

设

$$
A = 10c_{23} - 100s_{23} + 100c_2, \tag{2}
$$

则式 (1a) 和式 (1b) 可写成

$$
x = Ac_1 - 20s_1 \tag{3a}
$$

$$
y = As_1 + 20c_1 \tag{3b}
$$

$(3a)^2 + (3b)^2$, 得

$$
A^2 = x^2 + y^2 - 400, \tag{4}
$$

$(2)^2 + (1c)^2$, 整理可得

$$
\frac{A^2 + z^2}{100} = 201 + 20(c_3 - 10s_3), \tag{5}
$$

将 (4) 代入上式, 并设

$$
\rho^2 = x^2 + y^2 + z^2, \tag{6}
$$

整理可得,

$$c_3 - 10s_3 = \frac{\rho^2 - 20500}{2000}, \tag{7}$$

设

$$B = \frac{\rho^2 - 20500}{2000}, \tag{8}$$

将其代入式 (7), 并应用辅助角公式, 可得

$$\sqrt{101}\sin(\theta_3 + \phi_1) = B, \tag{9}$$

其中 $\phi_1 = \arctan(\frac{1}{-10}) + \pi$. 由此即可解出 θ_3.

将解得的 θ_3 代回式 (1c), 整理可得

$$(B + 10)s_2 + (s_3 + 10c_3)c_2 = \frac{z}{-10}. \tag{10}$$

设

$$C = B + 10, \tag{11}$$

$$D = s_3 + 10c_3. \tag{12}$$

注意到, $C = (\rho^2 - 500)/2000$, 而本设计选取的路径上的点显然都满足到基座的距离大于 $\sqrt{500}$, 因此可以保证 $C > 0$. 从而, 对式 (10) 应用辅助角公式, 并将式 (11) 和式 (12) 代入, 可得

$$\sqrt{C^2 + D^2}\sin(\theta_2 + \phi_2) = \frac{z}{-10}, \tag{13}$$

其中 $\phi_2 = \arctan(D/C)$. 由此即可解出 θ_2.

(3b) $\times A -$ (3a) $\times 20$, 结合式 (4), 整理可得

$$s_1 = \frac{yA - 20x}{x^2 + y^2}. \tag{14}$$

将解得的 θ_2, θ_3 代入式 (2) 即可得到 A 的值. 考虑到 $\theta_1 \in (-\pi/2, \pi/2)$, 由式 (14) 立刻得到

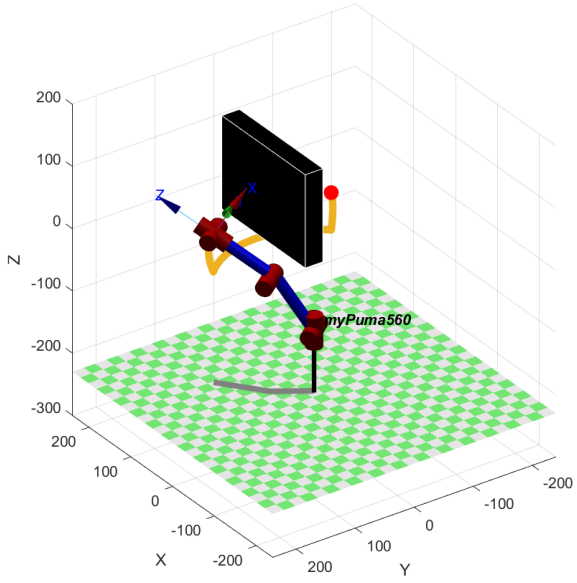
$$\theta_1 = \arcsin(\frac{yA - 20x}{x^2 + y^2}). \tag{15}$$

至此, $\theta_1, \theta_2, \theta_3$ 的解析解已全部可以给出. 需要说明的是, θ_2, θ_3 的值分别由式 (13) 和式 (9) 给出, 但这两个式子各自可能存在两个解. 因此, 在计算出可能的解后, 应将结果代回正运动学公式 (1) 进行验证. 此外, 还需要考虑 θ_2, θ_3 的值应在 $(-\pi/2, \pi/2)$ 范围内. 具体的实施细节见下文列出的 `my_ikine` 函数代码.

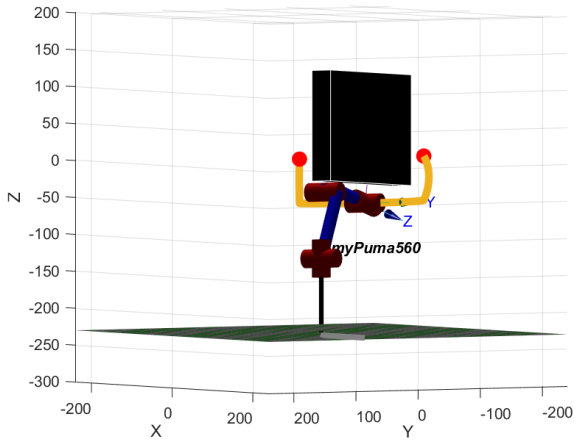
3.2 结果

按照以上思路, 计算出的机械臂从起点到终点的工作空间路径如图 4 所示, 各子图展示了机械臂在路径的起点、中点、终点时的姿态. 图中, 两个红色的点代表起点和终点, 橙色的曲线代表路径.

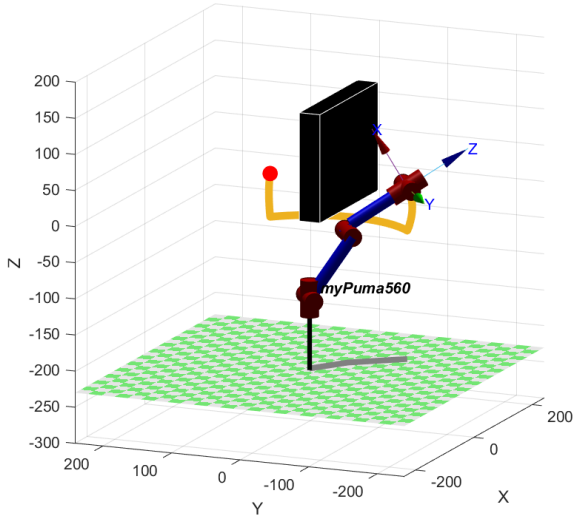
机械臂运动过程中, 其各关节角度的变化曲线如图 5 所示. 注意 $\theta_4, \theta_5, \theta_6$ 始终为 0, 因而其曲线重合.



(a) 起点处



(b) 中点处



(c) 终点处

图 4: 机械臂从起点到终点的工作空间路径及在运动中的姿态

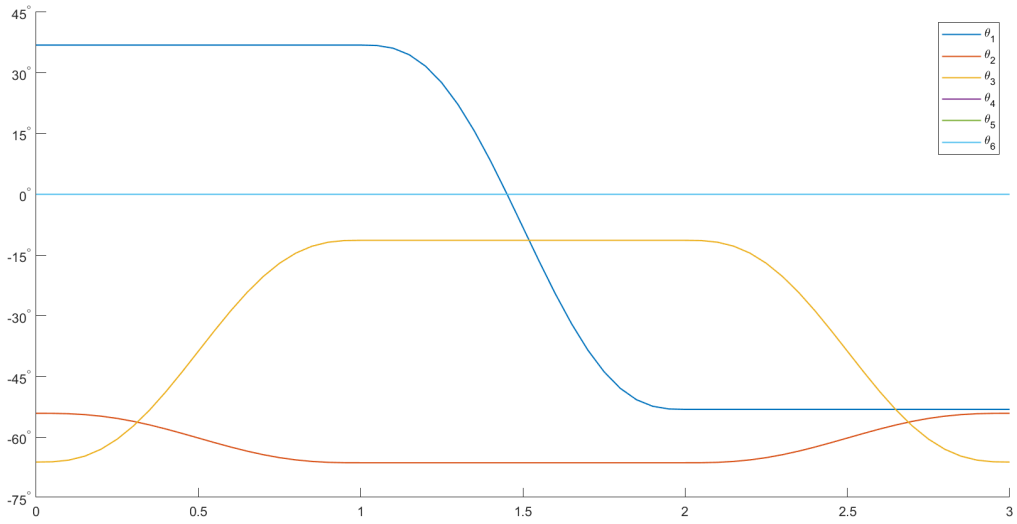


图 5: 机械臂各关节角度的变化

3.3 代码

计算逆运动学的代码如下.

```
function [q1,q2,q3] = my_ikine(x,y,z)
    % determine whether two floating numbers are equal
    eq = @(a,b) abs(a-b) < 1e-5;

    function [x,y,z] = my_fkine(q1,q2,q3)
        % calculate the forward kinematics
        x = 10 * cos(q1) * cos(q2+q3) - 100 * cos(q1) * sin(q2+q3) ...
                        + 100 * cos(q1) * cos(q2) - 20 * sin(q1);
        y = 10 * sin(q1) * cos(q2+q3) - 100 * sin(q1) * sin(q2+q3) ...
            + 100 * sin(q1) * cos(q2) + 20 * cos(q1);
        z = -10 * sin(q2+q3) - 100 * cos(q2+q3) - 100 * sin(q2);
    end

    function [q3_1, q3_2] = get_q3(x,y,z)
       rho2 = x*x + y*y + z*z;
       B = (rho2 - 20500) / 2000;
       phi1 = atan(1/(-10)) + pi;
       q3_p_phi1_m_pi = asin(-B/sqrt(101));  % q3 + phi1 - pi
       if tan(q3_p_phi1_m_pi) < -10
           q3_p_phi1_m_pi_2 = -pi - q3_p_phi1_m_pi;
       else
           q3_p_phi1_m_pi_2 = 1453;
       end
       q3_1 = q3_p_phi1_m_pi - phi1 + pi;
       q3_2 = q3_p_phi1_m_pi_2 - phi1 + pi;
```

```
26      end
27
28      function [q2_1,q2_2] = get_q2(z,q3)
29          C = cos(q3) - 10 * sin(q3) + 10;
30          D = sin(q3) + 10 * cos(q3);
31          phi2 = atan(D / C);
32          if (-1 <= -z/10/sqrt(C*C + D*D)) && (-z/10/sqrt(C*C + D*D) <= 1)
33              q2_p_phi2 = asin(-z/10/sqrt(C*C + D*D));  % q2 + phi2
34              if q2_p_phi2 > 0
35                  q2_p_phi2_2 = pi - q2_p_phi2;
36              else
37                  q2_p_phi2_2 = -pi - q2_p_phi2;
38              end
39              q2_1 = q2_p_phi2 - phi2;
40              q2_2 = q2_p_phi2_2 - phi2;
41          else
42              q2_1 = 1453;
43              q2_2 = 1453;
44          end
45      end
46
47      function q1 = get_q1(x,y,q2,q3)
48          A = 10 * cos(q2+q3) - 100 * sin(q2+q3) + 100 * cos(q2);
49          q1 = asin((y*A - 20*x) / (x*x + y*y));
50      end
51
52      [q3_1,q3_2] = get_q3(x,y,z);
53      for q3 = [q3_1,q3_2]
54          if (q3 > -pi/2) && (q3 < pi/2)
55              [q2_1,q2_2] = get_q2(z,q3);
56              for q2 = [q2_1,q2_2]
57                  if (q2 > -pi/2) && (q2 < pi/2)
58                      q1 = get_q1(x,y,q2,q3);
59                      [x1,y1,z1] = my_fkine(q1,q2,q3);
60                      if eq(x,x1) && eq(y,y1) && eq(z,z1)
61                          % [q1,q2,q3] is indeed a solution
62                          return
63                      end
64                  end
65              end
66          end
67      end
```

```
68
69      % fail to solve
70      error("Solution not found!");
71  end
```

规划避障路径的代码如下.

```
1   clear; close all; clc;
2
3   % create the robot
4   my_p560 = create_robot(100,20,10,100);
5
6   % change the base
7   offset = 130;
8   my_p560.base = [0, 0, -offset];
9
10  % plot the obstacle
11  center = [100, 0, 50];
12  size = [200, 30, 150];
13  origin = center - size/2;
14  vertex_index = [0,0,0; 0,0,1; 0,1,0; 0,1,1; 1,0,0; 1,0,1; 1,1,0; 1,1,1];
15  % compute the positions of eight vertices
16  vertex_pos = origin + vertex_index.*size;
17  % specify the six facets
18  facet=[1,2,4,3; 1,2,6,5; 1,3,7,5; 2,4,8,6; 3,4,8,7; 5,6,8,7];
19  patch('Vertices',vertex_pos, 'Faces',facet, 'EdgeColor','white');
20  hold on;
21
22  % compute the joint variables corresponding to the four anchor points
23  pini = [100, 100, 10];  % the start point
24  pmid1 = [100, 100, -50];  % the upper middle point
25  pmid2 = [100, -100, -50];  % the lower middle point
26  pend = [100, -100, 10];  % the end point
27  qini = zeros(1,6); qend = zeros(1,6);
28  qmid1 = zeros(1,6); qmid2 = zeros(1,6);
29  [qini(1),qini(2),qini(3)] = my_ikine(pini(1),pini(2),pini(3)+offset);
30  [qmid1(1),qmid1(2),qmid1(3)] = my_ikine(pmid1(1),pmid1(2),pmid1(3)+offset);
31  [qmid2(1),qmid2(2),qmid2(3)] = my_ikine(pmid2(1),pmid2(2),pmid2(3)+offset);
32  [qend(1),qend(2),qend(3)] = my_ikine(pend(1),pend(2),pend(3)+offset);
33
34  % interpolate the joint variables smoothly
35  t0 = [0 : 0.05 : 1]';
36  q1 = mtraj(@tpoly, qini, qmid1, t0);
```

```
q2 = mtraj(@tpoly, qmid1, qmid2, t0);
q3 = mtraj(@tpoly, qmid2, qend, t0);
q = [q1; q2(2:end,:); q3(2:end,:)];

% plot the path of the end effector
T = my_p560.fkine(q);
p = transl(T);
scatter3(pini(1), pini(2), pini(3), 100, 'r', 'filled');
scatter3(pend(1), pend(2), pend(3), 100, 'r', 'filled');
% my_p560.teach();
plot3(p(:,1), p(:,2), p(:,3), 'LineWidth',5);
my_p560.plot(q);

waitfor(gcf);

% plot the joint variables
hold on;
t = [0 : 0.05 : 3]';
for i = 1:6
    plot(t, q(:,i), ...
    'DisplayName',strcat('\theta_',num2str(i)), 'LineWidth',1);
end
legend;
ylim([-5*pi/12, pi/4]);
yticks(-5*pi/12 : pi/12 : pi/4);
yticklabels({'-75^{\circ}','-60^{\circ}','-45^{\circ}','-30^{\circ}', ...
        '-15^{\circ}','0^{\circ}','15^{\circ}','30^{\circ}','45^{\circ}'});
```