1 关节空间梯形速度规划

1.1 理论推导

考虑一机器人轨迹 $(t_1,\theta_1),(t_2,\theta_2),\ldots,(t_n,\theta_n)$, 其中 (t_1,θ_1) 与 (t_n,θ_n) 为起止点, 其余点为中间点。

如图 1, 对起点处, 加速度

$$\ddot{\theta_1} = sgn(\theta_2-\theta_1)a_{max},$$

变速时间

$$t_{b1} = t_2 - t_1 - \sqrt{(t_2-t_1)^2 - 2\frac{\theta_2-\theta_1}{a_1}},$$

第一段一次轨迹速度

$$v_1 = \frac{\theta_2-\theta_1}{t_2-t_1-\frac{t_{b1}}{2}}$$

对终点处,

$$\begin{aligned}\ddot{\theta_n} &= sgn(\theta_{n-1}-\theta_n)a_{max},\\ t_{b1} &= t_n - t_{n-1} - \sqrt{(t_n-t_{n-1})^2 + 2\frac{\theta_n-\theta_{n-1}}{a_n}},\\ v_1 &= \frac{\theta_n-\theta_{n-1}}{t_n-t_{n-1}-\frac{t_{bn}}{2}}\end{aligned}$$

对中间点处,

$$\begin{aligned}v_i &= \frac{\theta_{i+1}-\theta_i}{t_{i+1}-t_i}\\ \ddot{\theta_i} &= sgn(v_i-v_{i-1})a_{max},\\ t_{bi} &= \frac{v_i-v_{i-1}}{\theta_i}\end{aligned}$$

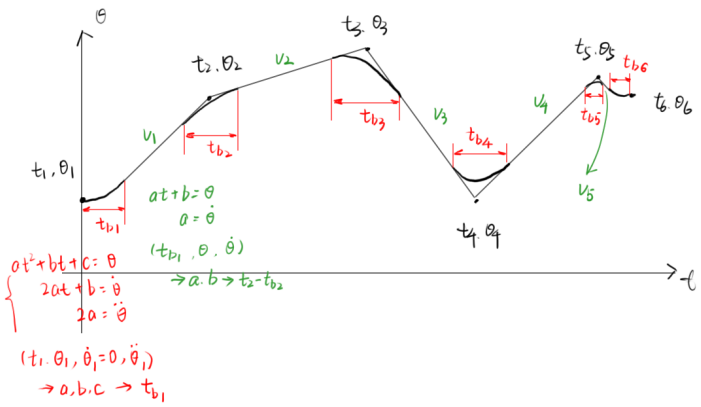

图 1: 轨迹示意图与物理量定义

当确定所有关键点的 $t_{bi}, \ddot{\theta}_i$ 与一次轨迹速度 v_i 后, 运动过程可分为 n 段二次轨迹与 n-1 段一次轨迹。

设二次轨迹为 $\theta(t) = a_{2i}t^2 + b_{2i}t + c_{2i}$, 一次轨迹为 $\theta(t) = a_{1i}t + b_{1i}$。考虑到:

①起点处关节运动状态 $\theta_1, \dot{\theta}_1, \ddot{\theta}_1$ 均已知, 令初始 $t_p = 0$。

②求解出第 i 条二次轨迹后, 令 $t_p = t_i + \frac{t_{bi}}{2}$(i=1 时则为 t_{b1}) 时的关节运动状态, 可求解第 i 条一次轨迹。

③求解出第 i 条一次轨迹后, 令 $t_p = t_{i+1} - \frac{t_{b(i+1)}}{2}$(i=n-1 时则为 $t_6 - t_{b6}$) 时的关节运动状态, 可求解第 i+1 条二次轨迹。

其中, 下标 p 表示用于计算轨迹参数的时间点, 及其对应的关节状态。

由以上条件, 可通过迭代的方式, 对所有轨迹系数进行求解。对二次轨迹:

$$
\begin{aligned}
a_{2i} &= \ddot{\theta}_p/2, \\
b_{2i} &= v_p - 2a_{2i}t_p, \\
c_{2i} &= \theta_p - b_{2i}t_p - a_{2i}t_p^2
\end{aligned}
$$

使用此轨迹更新 t_p, 对一次轨迹:

$$
\begin{aligned}
a_{1i} &= v_p, \\
b_{1i} &= \theta_p - a_{1i}t_p
\end{aligned}
$$

更新 t_p 后, 继续计算下一个二次轨迹。

计算轨迹后, 给定时刻 t, 比较出该点所在的轨迹后, 代入对应表达式即可得到此时的角度, 角速度, 角加速度。

1.2 代码实现

以下代码定义了一个 JointLFPB 类, 调用构造函数时输入所有关键点时刻与角度, 以及此关节的最大加速度, 自动求解各段轨迹的参数。函数 PointCalc 根据输入时刻 t, 返回该时刻对应的关节角度, 角速度, 角加速度。

```
classdef JointLFPB

    properties
        len  % 轨迹点数
        t_begin  % 开始时间
        t_end  % 结束时间
        t  % 关键点时刻，Points
        tb  % 二次段运动时间，Points
        % 二次段轨迹参数，q=a2t^2+b2t+c2，分别为Points维矩阵
        a2
        b2
        c2
        % 一次段轨迹参数，q=a1t+b，分别为(Points-1)*N矩阵
        a1
        b1
    end

```

```
18      methods
19
20          function obj = JointLFPB(t, q, max_acc)
21              len = size(q, 2);  % 点数
22              obj.len = len;
23              tb = zeros(1, len);
24              v = zeros(1, len - 1);
25              a = zeros(1, len);
26
27              % 起点
28              a(1) = sign(q(2) - q(1)) * max_acc;
29              tb(1) = t(2) - t(1) - sqrt((t(2) - t(1))^2 - 2 * (q(2) - q(1)) / a(1))
                    ;
30              v(1) = (q(2) - q(1)) / (t(2) - t(1) - tb(1)/2);
31
32              % 终点
33              a(len) = sign(q(len-1) - q(len)) * max_acc;
34              tb(len) = t(len) - t(len-1) - sqrt((t(len) - t(len-1))^2 + 2 * (q(len)
                     - q(len-1)) / a(len));
35              v(len-1) = (q(len) - q(len-1))/ (t(len) - t(len-1) - tb(len)/2);
36
37              % 中间点
38              for i=2:len-2
39                  v(i) = (q(i+1) - q(i)) / (t(i+1) - t(i));
40              end
41              for i=2:len-1
42                  a(i) = sign(v(i) - v(i-1)) * max_acc;
43                  tb(i) = (v(i) - v(i-1)) / a(i);
44              end
45
46              % 设置起止时间
47              obj.t_begin = t(1);
48              obj.t_end = t(len);
49              obj.tb = tb;
50              obj.t = t;
51
52              % 计算轨迹
53              ti = t(1);
54              qi = q(1);
55              vi = 0;
56              ai = a(1);
57              for i=1:len-1
58                  % 二次轨迹
59                  obj.a2(i) = ai / 2;
60                  obj.b2(i) = vi - 2 * ti * obj.a2(i);
61                  obj.c2(i) = qi - ti * obj.b2(i) - ti^2 * obj.a2(i);
62                  % ti更新为t(i)+tb(i)/2
63                  if i==1
64                      ti = tb(1);
65                  else
66                      ti = t(i) + tb(i) / 2;
67                  end
68                  qi = obj.a2(i) * ti.^2 + obj.b2(i) * ti + obj.c2(i);
69                  vi = 2 * obj.a2(i) * ti + obj.b2(i);
70                  % 一次轨迹
71                  obj.a1(i) = vi;
72                  obj.b1(i) = qi - obj.a1(i) * ti;
```

```
73                  % ti更新为t(i+1)-tb(i+1)/2
74                  ti = t(i+1) - tb(i+1) / 2;
75                  qi = obj.a1(i) * ti + obj.b1(i);
76                  vi = obj.a1(i);
77                  ai = a(i+1);
78              end
79              % 最后一段轨迹，此时ti-t(len)-tb(len)
80              ti = t(len) - tb(len);
81              qi = obj.a1(len-1) * ti + obj.b1(len-1);
82              vi = obj.a1(len-1);
83              ai = a(len);
84
85              obj.a2(len) = ai / 2;
86              obj.b2(len) = vi - 2 * ti * obj.a2(len);
87              obj.c2(len) = qi - ti * obj.b2(len) - ti^2 * obj.a2(len);
88          end
89
90          function [q, v, a] = PointCalc(obj, t)
91              % 若时间点超出边界，则按起止点进行计算
92              if t < obj.t_begin
93                  t = obj.t_begin;
94              elseif t > obj.t_end
95                  t = obj.t_end;
96              end
97              % 计算t所在的区间
98              traj_index = 1;  % 所在二次/一次轨迹的下标
99              traj_times = 2;  % 二次/一次轨迹
100             for i = 1:obj.len-1
101                 if t >= obj.t(i) + obj.tb(i) * (0.75 - 0.25 * sign(i - 1.5))  % i
                        =1时tb(i)系数为1，否则为0.5
102                     traj_times = 1;
103                     if t >= obj.t(i+1) - obj.tb(i+1) * (0.75 + 0.25 * sign(i - obj
                            .len + 1.5))  % i=len-1时tb(i)系数为1，否则为0.5
104                         traj_index = i + 1;
105                         traj_times = 2;
106                     end
107                 else
108                     break
109                 end
110             end
111             % 计算
112             if traj_times==2
113                 q = obj.a2(traj_index) * t^2 + obj.b2(traj_index) * t + obj.c2(
                        traj_index);
114                 v = 2 * obj.a2(traj_index) * t + obj.b2(traj_index);
115                 a = obj.a2(traj_index);
116             elseif traj_times==1
117                 q = obj.a1(traj_index) * t + obj.b1(traj_index);
118                 v = obj.a1(traj_index);
119                 a = 0;
120             end
121         end
122
123         function [q, v, a] = TrajGenerate(obj, t)
124             q = [];
125             v = [];
126             a = [];
```

```
            for i = 1:length(t)
                [qi, vi, ai] = obj.PointCalc(t(i));
                q = [q; qi];
                v = [v; vi];
                a = [a; ai];
            end
            q = q';
            v = v';
            a = a';
        end

    end
end
```

以下代码生成 6 轴机器人关节空间规划轨迹:

```
clear;

lfpb(1) = JointLFPB([0,1.5,4], [4.577, 9.577, 14.577]*pi/180, 0.2);
lfpb(2) = JointLFPB([0,1.5,4], [15.401, 20.401, 25.401]*pi/180, 0.2);
lfpb(3) = JointLFPB([0,1.5,4], [119.444, 124.444, 129.444]*pi/180, 0.2);
lfpb(4) = JointLFPB([0,1.5,4], [6.290, 1.290, -3.710]*pi/180, 0.2);
lfpb(5) = JointLFPB([0,1.5,4], [53.514, 48.514, 43.514]*pi/180, 0.2);
lfpb(6) = JointLFPB([0,1.5,4], [2.617, 12.617, 22.617]*pi/180, 0.2);
t = 0:0.001:4;

for i = 1:6
    [q(:,i),qd(:,i),qdd(:,i)] = lfpb(i).TrajGenerate(t);
end
t = t';
q = q * 180 / pi;  % 转换角度制
```

1.3 运行结果

各关节目标角度与实际角度如图 2。

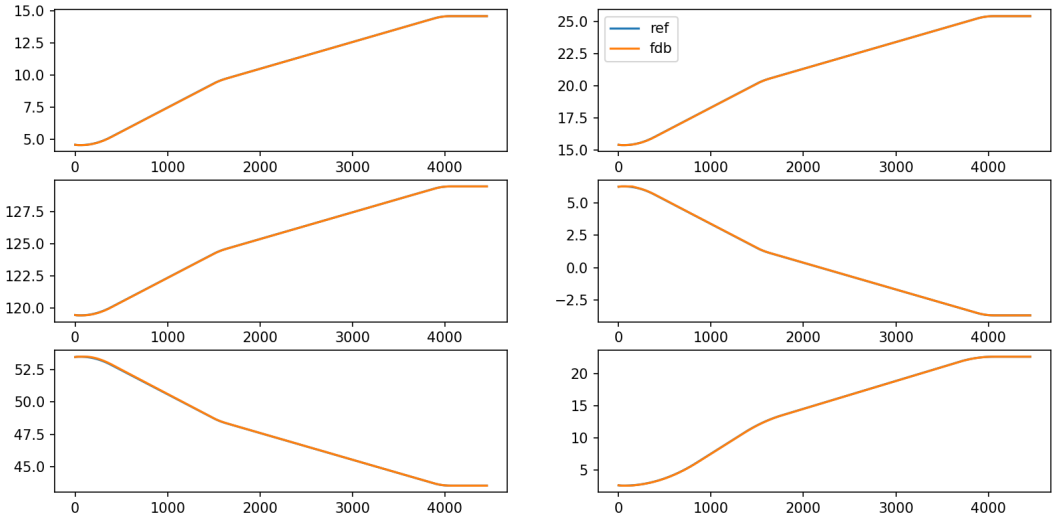

图 2: 各关节目标与实际角度

2 笛卡尔空间直线轨迹梯形速度规划

2.1 理论推导

本节中记笛卡尔空间平移向量为 $q_{1\times3}$, 关节角度为 $p_{1\times6}$.

由于笛卡尔空间内三平移自由度解耦, 因此给定目标点, 可分别对 x,y,z 方向进行速度规划, 可采用上述提出中的算法。

对于机械臂姿态, 本实验中保持与初始状态相同, 因此需给定机械臂初始状态, 通过计算得到 $g_1 = \begin{bmatrix} R_1 & q_1 \\ 0 & 1 \end{bmatrix}$, 而后续关键点直接给出 p_i。

通过计算得到 $q(t)$ 后, t 时刻机械臂末端位姿即 $g(t) = \begin{bmatrix} R_1 & q(t) \\ 0 & 1 \end{bmatrix}$。

接下来分别对每一时刻进行逆运动学。由于给定末端位姿, 机械臂关节角度 q 最多存在 8 组解, 因此需要对解进行选择。通过子问题方法得到所有解后, 首先将所有角度转换到 $[-\pi, \pi]$ 内, 并检验对应关节角是否处于机器人限位范围内。

记初始时间为 t_1, 步长 Δt, t 时刻选取" 上一时刻关节角"

$$p_l = \begin{cases} p_1, t = t_1 \\ p(t - \Delta t), t \neq t_1 \end{cases}$$

对每一组解计算 $Cost = \sum_{i=1}^{6} w_i (p_{(i)} - p_{l(i)})^2$, 其中 w_i 为各关节的权重系数。

最后选取 $Cost$ 最小的解作为最优解。若关节空间轨迹连续且步长足够小, 理论上最优解有 $Cost \approx 0$。

得到关节空间轨迹后, 还需计算各关节角速度与角加速度, 验证其在机械臂允许的范围内。

2.2 代码实现

笛卡尔空间规划代码:

```
% INPUTS: p0            机器人初始关节角度
%         q             关键点对应[x,y,z]坐标, Points*3矩阵, (不包括起点,包括终点)
%         t             关键点时间, 1*Points矩阵
%         max_acc       笛卡尔空间每个方向最大加速度
%         step          时间步长
function p = CartesianTransition(p0, q, t, max_acc, step)
    % 机器人参数
    gst0 = [
        -1 0 0 0; ...
        0 -1 0 0; ...
        0 0 1 1475; ...
        0 0 0 1
    ];

    wq = [0   0   0   0   0   0; ...
          0   0   0   0   0   0; ...
          0   491   941   941   1391   1391];
    w = [0   0   0   0   0   0; ...
         0   1   1   0   1   0; ...
         1   0   0   1   0   1];

    for i=1:6
        v(:,i) = cross(wq(1:3,i), w(1:3,i));
    end

    for i=1:6
        Xi(:,i) = [v(:,i); w(:,i)];
    end

    % 正运动学求初始姿态
    g1 = Fkine(Xi, p0, gst0);
    % 初始点加入关键点
    q = [g1(1:3, 4)'; q];
    t = [0 t];

    % 平移三轴解耦, 因此可类似关节空间LFPB插值进行规划
    lfpb(1) = JointLFPB(t, q(:,1)', max_acc);  % x
    lfpb(2) = JointLFPB(t, q(:,2)', max_acc);  % y
    lfpb(3) = JointLFPB(t, q(:,3)', max_acc);  % z

    % 生成轨迹
    t = 0:step:t(length(t));
    for i = 1:3
        [cart_q(:,i),cart_v(:,i),cart_a(:,i)] = lfpb(i).TrajGenerate(t);
    end

```

```
47      % 逆解，N*6矩阵
48      p = [];
49      % 根据上一个时间点，选取当前所有解中与之最相近且满足范围的一组解
50      for i = 1:size(cart_q,1)
51          % 机器人上一步关节角
52          if i == 1
53              last_p = p0;
54              sol_ = Ikine6s(g1);  % 8组解
55          else
56              last_p = p(i-1,:);
57              sol_ = Ikine6s([g1(1:3, 1:3), cart_q(i,1:3)'; 0 0 0 1]);
58          end
59          sol = [];
60
61          for sol_index = 1:size(sol_,1)
62              for joint_index = 1:6
63                  % 多圈校正
64                  while sol_(sol_index,joint_index) > pi
65                      sol_(sol_index,joint_index) = sol_(sol_index,joint_index) - 2*
                            pi;
66                  end
67                  while sol_(sol_index,joint_index) < -pi
68                      sol_(sol_index,joint_index) = sol_(sol_index,joint_index) + 2*
                            pi;
69                  end
70              end
71          end
72
73          % 关节角限制
74          for sol_index = 1:size(sol_,1)
75              min_val = -[170, 120, 140, 170, 120]*pi/180;
76              min_judge = sol_(sol_index, 1:5) > min_val;
77              max_val = [170, 120, 140, 170, 120]*pi/180;
78              max_judge = sol_(sol_index, 1:5) < max_val;
79              if(sum(min_judge) == 5 && sum(max_judge) == 5)
80                  sol = [sol; sol_(sol_index, :)];
81              end
82          end
83
84          % 计算角度差
85          del_p = abs(sol(:,:) - last_p);
86          for sol_index = 1:size(sol,1)
87              for joint_index = 1:6
88                  % 多圈校正
89                  while del_p(sol_index,joint_index) > pi
90                      del_p(sol_index,joint_index) = del_p(sol_index,joint_index) -
                            2*pi;
91                  end
92              end
93          end
94          del_p = del_p .^ 2;  % 角度差的平方
95
96          % 根据不同角度权重，选取代价函数最小的一组解
97          weight = [2,3,3,1,1,0.1];
98          cost = sum((del_p .* weight)');
99          p = [p; sol(find(cost == min(cost)), :)];
100     end
```

```
101
102 end
```

以下代码生成实际轨迹:

```
1  clear;
2
3  t = 0:0.001:10;
4
5  q = CartesianTransition([13.846, 34.767, 109.040, 9.122, 43.836, 8.990]*pi/180,
       ...
6  [494.688, 131.422, 444.683; ...
7  494.687, 41.423, 444.682; ...
8  494.686, 41.424, 414.683], ...
9  [2,8,10], 50, 0.001);
10
11 qd = [zeros(1,6); diff(q, 1, 1)] / 0.001;
12 qdd = [zeros(1,6); diff(qd, 1, 1)] / 0.001;
13
14 t = t';
15 q = q * 180 / pi;
```

2.3 运行结果

各关节目标角度与实际角度如图 3。图中除目标门形轨迹外, 也包括末端从当前姿态到达第一关键点 (点位文件中的起点) 的运动过程, 即机械臂实际沿矩形轨迹运动。

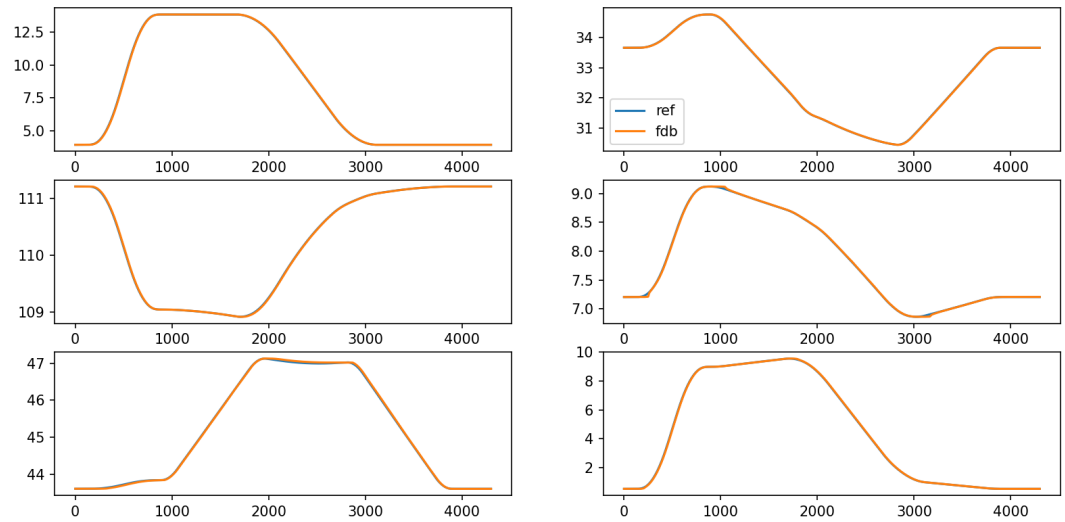

图 3: 各关节目标与实际角度

3 对齐时间

为使各关节尽可能同时启停, 所有关节设定关键点的时间相同。例如关节 1 的关键点为 $(t_1, \theta_{11}), (t_2, \theta_{12}), \ldots, (t_n, \theta_{1n})$, 则关节 i(i=2,...,6) 的关键点为 $(t_1, \theta_{i1}), (t_2, \theta_{i2}), \ldots, (t_n, \theta_{in})$。

对于笛卡尔空间, 则 x,y,z 方向上设定关键点的时间相同。

4 码垛功能实现

图 4采用以下方式实现码垛功能, 需提供起点到目标点处的点位文件。程序执行开始, 机械臂回到物体上方一定距离的 ① 处。打开气泵, 机械臂运动到 ② 处, 获取物体, 此后沿给定直线规划轨迹到达 ④。最后关闭气泵, 放置物体, 并运动到物体放置目标上方的 ③ 处。

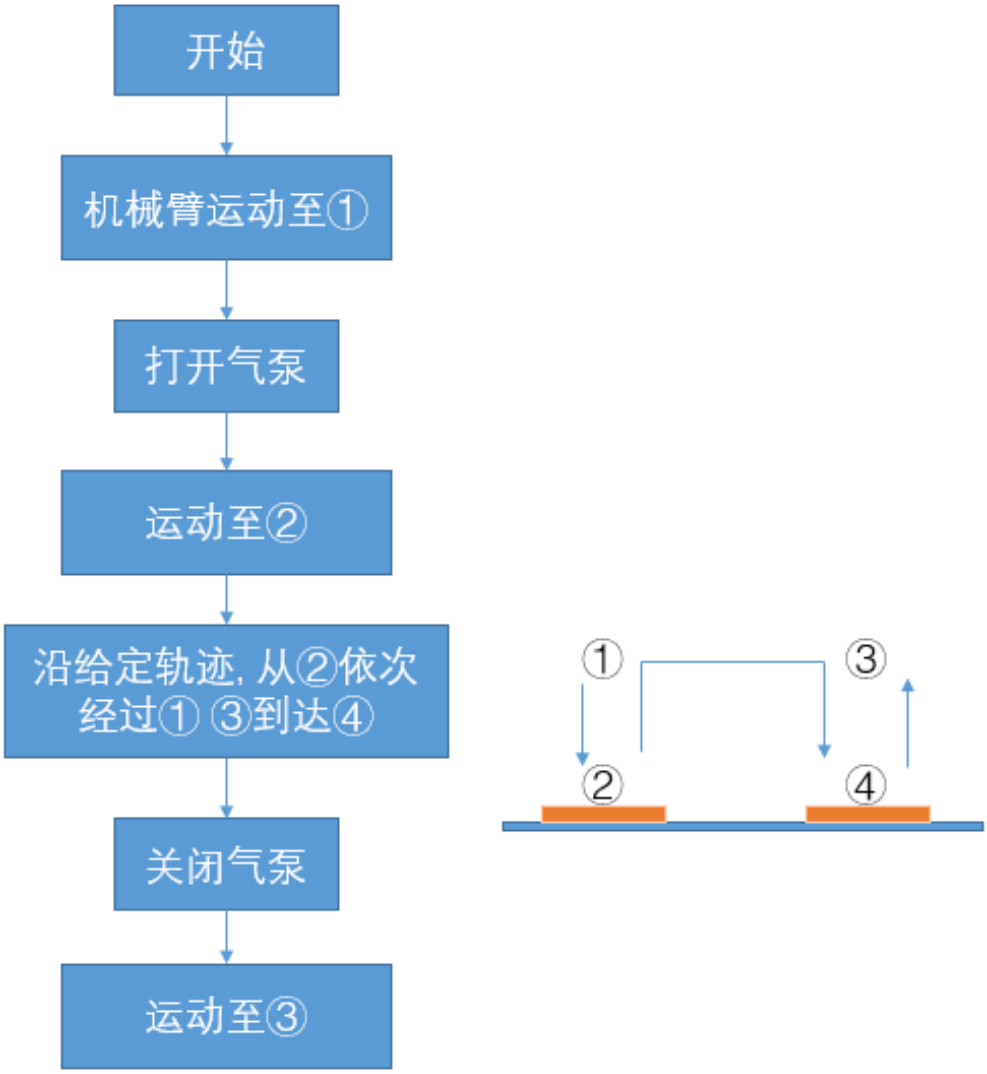

图 4: 码垛流程图