运动规划第 4 章作业 ROS 版

1.第一步是建立前向的路径方程，即使用物理知识，建立目标位置，起始位置，起始速度，目标速度，加速度三者之间的关系。

```
Pf=P0+v0*t+(1/2)a*t**2,vf=v0+a*t
```

2. 第二步是在路径中选择 cost 最小的，即作业中的目标函数 J 最小的

J 函数的极小值就是求解对于 T 一阶导为 0 的情况，为此先要求解 J 函数相对于 T 的一阶导。依据作业指导里面的 J 的表达式，使用 Python 的 sympy 库进行求导（也可以手推）。

求解完之后使用 simplify 函数对表达式进行整理

```
import sympy
from sympy import symbols
from sympy.abc import *

dpx = symbols('dpx')
dpy = symbols('dpy')
dpz = symbols('dpz')
dvx = symbols('dvx')
dvy = symbols('dvy')
dvz = symbols('dvz')
T = symbols('T')
a1 = (-12) * dpx / (T ** 3) + 6 * dvx / (T ** 2)
a2 = (-12) * dpy / (T ** 3) + 6 * dvy / (T ** 2)
a3 = (-12) * dpz / (T ** 3) + 6 * dvz / (T ** 2)
b1 = 6 * dpx / (T ** 2) + (-2) * dvx / T
b2 = 6 * dpy / (T ** 2) + (-2) * dvy / T
b3 = 6 * dpz / (T ** 2) + (-2) * dvz / T
J = T + ((1 / 3) * (a1 ** 2) * (T ** 3) + a1 * b1 * (T ** 2) + (b1 ** 2) * T) + (
        (1 / 3) * (a2 ** 2) * (T ** 3) + a2 * b2 * (T ** 2) + (b2 ** 2) * T) + (
            (1 / 3) * (a3 ** 2) * (T ** 3) + a3 * b3 * (T ** 2) + (b3 ** 2) * T)
dJ = sympy.diff(J, T)
print(dJ)
dj_simple = sympy.simplify(dJ)
print(dj_simple)
```

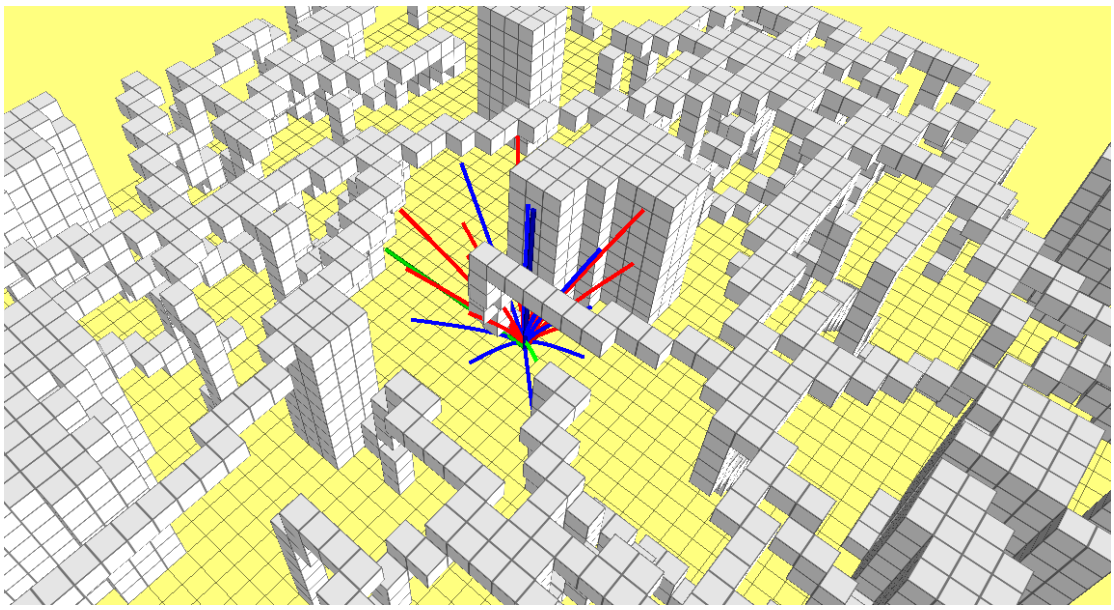
```
0.333333333333333*T**3*(-24*dvx/T**3 + 72*dpx/T**4)*(6*dvx/T**2 - 12*dpx/T**3) + 0.333333333333333*T**3*(-24*dvy/T**3 + 72*dpy/T**4)*(6*dvy/T**2 - 12*dpy/T**3) + 0.333333333333333*
1.0*(1.0*T**4 - 4.0*T**2*dvx**2 - 4.0*T**2*dvy**2 - 4.0*T**2*dvz**2 + 24.0*T*dpx*dvx + 24.0*T*dpy*dvy + 24.0*T*dpz*dvz - 36.0*dpx**2 - 36.0*dpy**2 - 36.0*dpz**2)/T**4
```

现在问题变成了求解下方的一元四次方程组的根，使用多项式的伴随矩阵进行求解，参考 [(68 条消息) 一元四次方程的求解_古路的博客-CSDN 博客_一元四次方程求解]
为了简化求解的过程，将目标速度设置为 0

```
T^4 - 4*(vx0^2 + vy0^2 + vz0^2) * T^2 + 24 * (dx * vx0 + dy * vy0 + dz * vz0) * T - 36 * (dx^2 + dy^2 +dz^2)
```

根求解完之后，重新带回到 J 的表达式，就得到了最优的 cost

3. 按照第一章编译和启动程序的方式进行相关操作，然后在 3D 地图上设置目标点，结果如下



/home/jeff/catkin_ws/src/grid_path_searcher/launch/demo.launch http://localhost:11311

roscore http://ubuntu:11... × | jeff@ubuntu: ~/catkin_ws × | /home/jeff/catkin_ws/src... ×

```
matrix_eigenvalues:
 (-2.41762,2.71151)
(-2.41762,-2.71151)
        (6.99316,0)
       (-2.15793,0)
optimal cost: 43.8074
_start_velocity:
 1.475
 1.475
2.6775

matrix_44:
      0       0       0 275.263
      1       0       0 216.369
      0       1       0  46.081
      0       0       1       0

matrix_eigenvalues:
 (-3.23394,2.03575)
(-3.23394,-2.03575)
       (-2.1798,0)
       (8.64768,0)
optimal cost: 56.4888
```