深蓝学院
shenlanxueyuan.com

第4章 “运动学约束下的运动规划”作业分享

主讲人 haiyan

纲要

- **第一部分：Homework1**

  1、求解推导

  2、求解代码部分

- **第二部分：Homework2**

  1、代码补充

  2、编译运行指令及结果

Homework1

- For the OBVP problem stated in slides p.25-p.29, please get the optimal solution (control, state, and time) for partially free final state case.
- Suppose the position is fixed, velocity and acceleration are free here.

Homework1

1、求解推导

根据已给材料，有如下推导：

$$\begin{bmatrix}\alpha_1\\ \alpha_2\\ \alpha_3\\ \beta_1\\ \beta_2\\ \beta_3\end{bmatrix} = \begin{bmatrix} -\frac{12}{T^3} & 0 & 0 & \frac{6}{T^2} & 0 & 0\\ 0 & -\frac{12}{T^3} & 0 & 0 & \frac{6}{T^2} & 0\\ 0 & 0 & -\frac{12}{T^3} & 0 & 0 & \frac{6}{T^2}\\ \frac{6}{T^2} & 0 & 0 & -\frac{2}{T} & 0 & 0\\ 0 & \frac{6}{T^2} & 0 & 0 & -\frac{2}{T} & 0\\ 0 & 0 & \frac{6}{T^2} & 0 & 0 & -\frac{2}{T}\end{bmatrix}\begin{bmatrix}\Delta p_x - v_{x0}T\\ \Delta p_y - v_{y0}T\\ \Delta p_z - v_{z0}T\\ \Delta v_x\\ \Delta v_y\\ \Delta v_z\end{bmatrix} = \begin{bmatrix} -\frac{12}{T^3}(\Delta p_x - v_{x0}T) + \frac{6\Delta v_x}{T^2}\\ -\frac{12}{T^3}(\Delta p_y - v_{y0}T) + \frac{6\Delta v_y}{T^2}\\ -\frac{12}{T^3}(\Delta p_z - v_{z0}T) + \frac{6\Delta v_z}{T^2}\\ \frac{6}{T^2}(\Delta p_x - v_{x0}T) - \frac{2\Delta v_x}{T}\\ \frac{6}{T^2}(\Delta p_y - v_{y0}T) - \frac{2\Delta v_y}{T}\\ \frac{6}{T^2}(\Delta p_z - v_{z0}T) - \frac{2\Delta v_z}{T}\end{bmatrix} \quad (1)$$

其中，

$\Delta p_x = p_{xf} - p_{x0}$，$\Delta p_y = p_{yf} - p_{y0}$，$\Delta p_z = p_{zf} - p_{z0}$；
$\Delta v_x = v_{xf} - v_{x0}$，$\Delta v_y = v_{yf} - v_{y0}$，$\Delta v_z = v_{zf} - v_{z0}$

Homework1

深蓝学院
shenlanxueyuan.com

1、求解推导

因为

$$J = T + \left(\frac{1}{3}\alpha_1^2 T^3 + \alpha_1\beta_1 T^2 + \beta_1^2 T\right) + \left(\frac{1}{3}\alpha_2^2 T^3 + \alpha_2\beta_2 T^2 + \beta_2^2 T\right) + \left(\frac{1}{3}\alpha_3^2 T^3 + \alpha_3\beta_3 T^2 + \beta_3^2 T\right) \quad (2)$$

令

$$A = \frac{1}{3}\alpha_1^2 T^3 + \alpha_1\beta_1 T^2 + \beta_1^2 T \quad (3)$$

$$B = \frac{1}{3}\alpha_2^2 T^3 + \alpha_2\beta_2 T^2 + \beta_2^2 T \quad (4)$$

$$C = \frac{1}{3}\alpha_3^2 T^3 + \alpha_3\beta_3 T^2 + \beta_3^2 T \quad (5)$$

则，

$$J = T + A + B + C \quad (6)$$

Homework1

深蓝学院 shenlanxueyuan.com

1、求解推导

将(1)中的α_1、α_2、α_3、β_1、β_2、β_3分别代入(3)、(4)、(5)，化简得

$$A = \frac{12}{T^3}\Delta p_x^2 - \frac{12}{T^2}(2\Delta p_x v_{x0} + \Delta p_x \Delta v_x) + \frac{4}{T}(3\Delta v_{x0}^2 + 3\Delta v_x v_{x0} + \Delta v_x^2) \quad (7)$$

$$B = \frac{12}{T^3}\Delta p_y^2 - \frac{12}{T^2}(2\Delta p_y v_{y0} + \Delta p_y \Delta v_y) + \frac{4}{T}(3\Delta v_{y0}^2 + 3\Delta v_y v_{y0} + \Delta v_y^2) \quad (8)$$

$$C = \frac{12}{T^3}\Delta p_z^2 - \frac{12}{T^2}(2\Delta p_z v_{z0} + \Delta p_z \Delta v_z) + \frac{4}{T}(3\Delta v_{z0}^2 + 3\Delta v_z v_{z0} + \Delta v_z^2) \quad (9)$$

将(7)、(8)、(9)代入(6)，化简得

$$J = T + \frac{12}{T^3}(\Delta p_x^2 + \Delta p_y^2 + \Delta p_z^2) - \frac{12}{T^2}[2(\Delta p_x v_{x0} + \Delta p_y v_{y0} + \Delta p_z v_{z0}) + (\Delta p_x \Delta v_x + \Delta p_y \Delta v_y + \Delta p_z \Delta v_z)] + \frac{4}{T}[3(\Delta v_{x0}^2 + \Delta v_{y0}^2 + \Delta v_{z0}^2) + 3(\Delta v_x v_{x0} + \Delta v_y v_{y0} + \Delta v_z v_{z0}) + (\Delta v_x^2 + \Delta v_y^2 + \Delta v_z^2)] \quad (10)$$

Homework1

1、求解推导

令

$$m = \Delta p_x^2 + \Delta p_y^2 + \Delta p_z^2 \tag{11}$$

$$n = 2\left(\Delta p_x v_{x0} + \Delta p_y v_{y0} + \Delta p_z v_{z0}\right) + \left(\Delta p_x \Delta v_x + \Delta p_y \Delta v_y + \Delta p_z \Delta v_z\right) \tag{12}$$

$$k = 3(\Delta v_{x0}^2 + \Delta v_{y0}^2 + \Delta v_{z0}^2) + 3(\Delta v_x v_{x0} + \Delta v_y v_{y0} + \Delta v_z v_{z0}) + (\Delta v_x^2 + \Delta v_y^2 + \Delta v_z^2) \tag{13}$$

则，

$$J = T + \frac{12}{T^3} m - \frac{12}{T^2} n + \frac{4}{T} k \tag{14}$$

令 $J' = 0$

则，(14)化简为

$$T^4 - 4kT^2 + 24nT - 36m = 0 \tag{15}$$

Homework1

2、求解代码部分

```
double OBVP(Eigen::Vector3d  start position,Eigen::Vector3d  start velocity,Eigen::Vector3d  target position)
{
   double optimal_cost = 100000.0;
   double deltaPx =  target position(0) -  start position(0);
   double deltaPy =  target position(1) -  start position(1);
   double deltaPz =  target position(2) -  start position(2);
   double deltaVx = 0 -  start velocity(0);
   double deltaVy = 0 -  start velocity(1);
   double deltaVz = 0 -  start velocity(2);
   double Vx0 = _start_velocity(0);
   double Vy0 =  start velocity(1);
   double Vz0 =  start velocity(2);

   double m = deltaPx*deltaPx +deltaPy*deltaPy + deltaPz*deltaPz;
   double n = 2*(deltaPx*Vx0 + deltaPy*Vy0 + deltaPz*Vz0) +
              deltaPx*deltaVx + deltaPy*deltaVy + deltaPz*deltaVz;
   double k = 3*(Vx0*Vx0 + Vy0*Vy0 +Vz0*Vz0 + deltaVx*Vx0 + deltaVy*Vy0 + deltaVz*Vz0) +
              deltaVx*deltaVx + deltaVy*deltaVy + deltaVz*deltaVz;

   Eigen::Matrix<double, 4, 4> CompanionMatrix44;
   Eigen::Matrix<complex<double>, Eigen::Dynamic, Eigen::Dynamic> CompanionMatrix44EigenValues;//复数动态矩阵
```

Homework1

深蓝学院
shenlanxueyuan.com

2、求解代码部分

```
double c = - 4*k;
double d = - 24*n;
double e = - 36*m;
vector<double> tmpOptimalT(4 ,0.0);
vector<double> tmpOptimalCost(4, 100000.0);
double optimalT = 0.0;
CompanionMatrix44 << 0, 0, 0, -e,
                     1, 0, 0, -d,
                     0, 1, 0, -c,
                     0, 0, 1, 0 ;

//std::cout<<"CompanionMatrix44: "<<std::endl<<CompanionMatrix44<<std::endl<<std::endl;
CompanionMatrix44EigenValues = CompanionMatrix44.eigenvalues();
//std::cout<<"matrix_eigenvalues: "<<std::endl<<CompanionMatrix44EigenValues<<std::endl;
```

Homework1

深蓝学院
shenlanxueyuan.com

2、求解代码部分

```
for(int i = 0; i < CompanionMatrix44EigenValues.size(); i++)
{
  //TODO:时间不能为负数
   //if(CompanionMatrix44EigenValues(i).imag() == 0)
   if(CompanionMatrix44EigenValues(i).imag() == 0  && CompanionMatrix44EigenValues(i).real() > 0)
   {
      tmpOptimalT[i] = CompanionMatrix44EigenValues(i).real();
      double t =  tmpOptimalT[i];
      double t2 = t*t;
      double t3 = t2*t;
      tmpOptimalCost[i] = t + (12*m)/(t3) - (12*n)/(t2) + (4*k)/t;
   }
   else
   {
      continue;
   }
}
```

Homework1

2、求解代码部分

```
for(int i = 0; i < tmpOptimalCost.size(); i++)
{
    optimal_cost = std::min(tmpOptimalCost[i],optimal_cost);
}
  //可以不用
int flag = 0;
for(int i = 0; i < tmpOptimalCost.size(); i++)
{
    if(optimal_cost == tmpOptimalCost[i])
    flag = i;
    else continue;
}
optimalT = tmpOptimalT[flag];
return optimal_cost;
}
```

以上该部分作业在课程给定资料的基础上，有参考：
https://blog.csdn.net/fb_941219/article/details/102991181?spm=1001.2014.3001.5501

Homework2

- Build an ego-graph of the linear modeled robot.
- Select the best trajectory closest to the planning target.

Homework2

1、代码补充

（1） demo_node.cpp

```
/*
            STEP 1: finish the forward integration, the modelling has been given in the document
            the parameter of forward integration:
_max_input_acc|_discretize_step|_time_interval|_time_step   all have been given
            use the pos and vel to recored the steps in the trakectory
            */
            // The flowing 6 lines of code are supplemented and added.
            pos(0) = pos(0) + vel(0) * delta_time + 0.5 * acc_input(0) * delta_time * delta_time;
            pos(1) = pos(1) + vel(1) * delta_time + 0.5 * acc_input(1) * delta_time * delta_time;
            pos(2) = pos(2) + vel(2) * delta_time + 0.5 * acc_input(2) * delta_time * delta_time;
            vel(0) = vel(0) + acc_input(0) * delta_time;
            vel(1) = vel(1) + acc_input(1) * delta_time;
            vel(2) = vel(2) + acc_input(2) * delta_time;
```

Homework2

1、代码补充

（2）hw_tool.cpp

```
// The following code is supplemented and added.
double Homeworktool::OptimalBVP(Eigen::Vector3d  start  position,Eigen::Vector3d  start  velocity,Eigen::Vector3d
_target_position)
{
   double optimal  cost = 100000.0;
   double deltaPx =   target  position(0) -   start  position(0);
   double deltaPy =   target  position(1) -   start  position(1);
   double deltaPz =   target  position(2) -   start  position(2);
   double deltaVx = 0 -   start  velocity(0);
   double deltaVy = 0 - _start_velocity(1);
   double deltaVz = 0 -   start  velocity(2);
   double Vx0 =   start  velocity(0);
   double Vy0 =   start  velocity(1);
   double Vz0 =   start  velocity(2);

   double m = deltaPx*deltaPx +deltaPy*deltaPy + deltaPz*deltaPz;
   double n = 2*(deltaPx*Vx0 + deltaPy*Vy0 + deltaPz*Vz0) + deltaPx*deltaVx + deltaPy*deltaVy + deltaPz*deltaVz;
   double k = 3*(Vx0*Vx0 + Vy0*Vy0 +Vz0*Vz0 + deltaVx*Vx0 + deltaVy*Vy0 + deltaVz*Vz0) +  deltaVx*deltaVx +
deltaVy*deltaVy + deltaVz*deltaVz;
```

Homework2

1、代码补充

（2）hw_tool.cpp

```
// Eigen 库相关使用资料 https://blog.csdn.net/shuzfan/article/details/52367329
Eigen::Matrix<double, 4, 4> CompanionMatrix44;
Eigen::Matrix<complex<double>, Eigen::Dynamic, Eigen::Dynamic>
CompanionMatrix44EigenValues;//复数动态矩阵

double c = - 4*k;
double d = - 24*n;
double e = - 36*m;
vector<double> tmpOptimalT(4 ,0.0);
vector<double> tmpOptimalCost(4, 100000.0);
double optimalT = 0.0;
CompanionMatrix44 << 0, 0, 0, -e,
                     1, 0, 0, -d,
                     0, 1, 0, -c,
                     0, 0, 1, 0 ;

//std::cout<<"CompanionMatrix44: "<<std::endl<<CompanionMatrix44<<std::endl<<std::endl;
CompanionMatrix44EigenValues = CompanionMatrix44.eigenvalues();
//std::cout<<"matrix_eigenvalues: "<<std::endl<<CompanionMatrix44EigenValues<<std::endl;

cout << "---------------" << endl;
cout << CompanionMatrix44EigenValues << endl;
```

Homework2

1、代码补充

（2）hw_tool.cpp

```
for(int i = 0; i < CompanionMatrix44EigenValues.size(); i++)
{
  //TODO:时间不能为负数
   // 判断条件怎么来的？ imag() == 0，剔除特征值为复数的情况，real() > 0，时间不能为负数
   // ignoring negative roots and complex roots, if all roots are complex, the function J is monotonous
   if(CompanionMatrix44EigenValues(i).imag() == 0  && CompanionMatrix44EigenValues(i).real() > 0)
   {
      // 为什么特征值可以直接拿过来当解？ 参考：
https://blog.csdn.net/fb_941219/article/details/102984587
      tmpOptimalT[i] = CompanionMatrix44EigenValues(i).real();
      double t =  tmpOptimalT[i];
      double t2 = t*t;
      double t3 = t2*t;
      tmpOptimalCost[i] = t + (12*m)/(t3) + (12*n)/(t2) + (4*k)/t;
   }
   else
   {
      continue;
   }
}
```

Homework2

深蓝学院
shenlanxueyuan.com

1、代码补充

（2）hw_tool.cpp

```
for(auto it = tmpOptimalCost.begin(); it != tmpOptimalCost.end(); it++)
{
    cout << *it << endl;
}
cout << "---------------" << endl;
optimal_cost = *min_element(tmpOptimalCost.begin(), tmpOptimalCost.end());
  //可以不用
int flag = 0;
for(int i = 0; i < tmpOptimalCost.size(); i++)
{
    if(optimal_cost == tmpOptimalCost[i])
    flag = i;
    else continue;
}
optimalT = tmpOptimalT[flag];
return optimal_cost;
}
```

Homework2

深蓝学院
shenlanxueyuan.com

2、编译运行指令及结果

$ catkin_make

$ roscore

Ctrl+Shift+t新建窗口

$ source devel/setup.bash

$ rviz

在RVIZ左上角的的File中打开下述配置文件

……\src\grid_path_searcher\launch\rviz_config\demo.rviz

Ctrl+Shift+t新建窗口

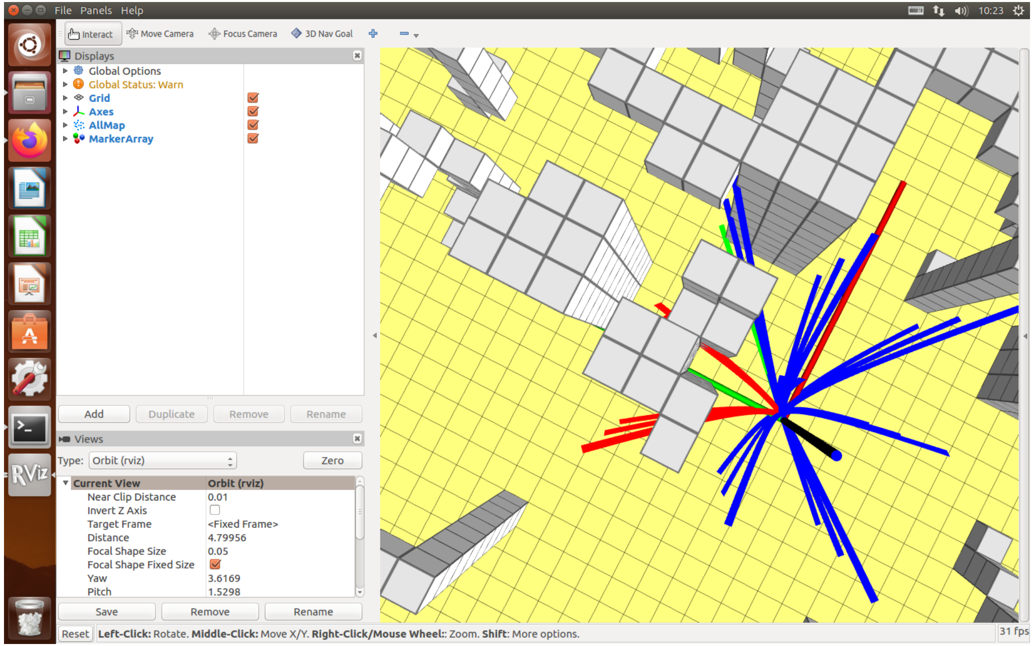
$ source devel/setup.bash

$ roslaunch ……\src\ grid_path_searcher\launch\ demo.launch

Homework2

2、编译运行指令及结果

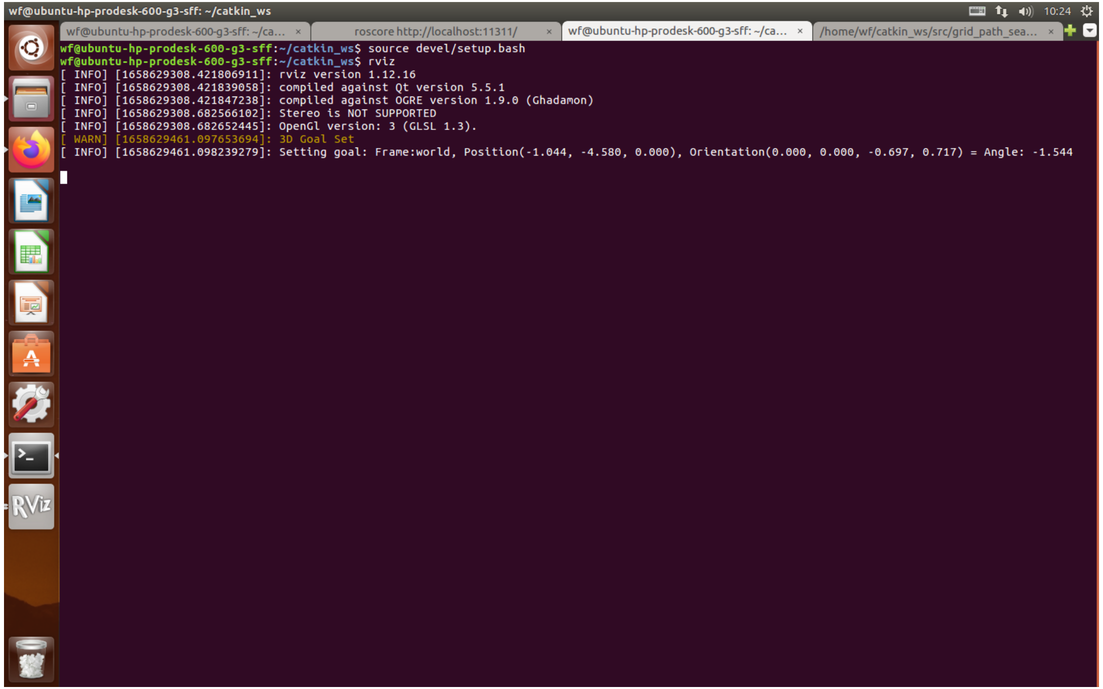
下图中，绿色线表示代价最小轨迹，蓝色线表示不会发生碰撞、非最优的轨迹，红色线表示会发生碰撞的轨迹。



Homework2

2、编译运行指令及结果

RVIZ设定的终点信息显示：

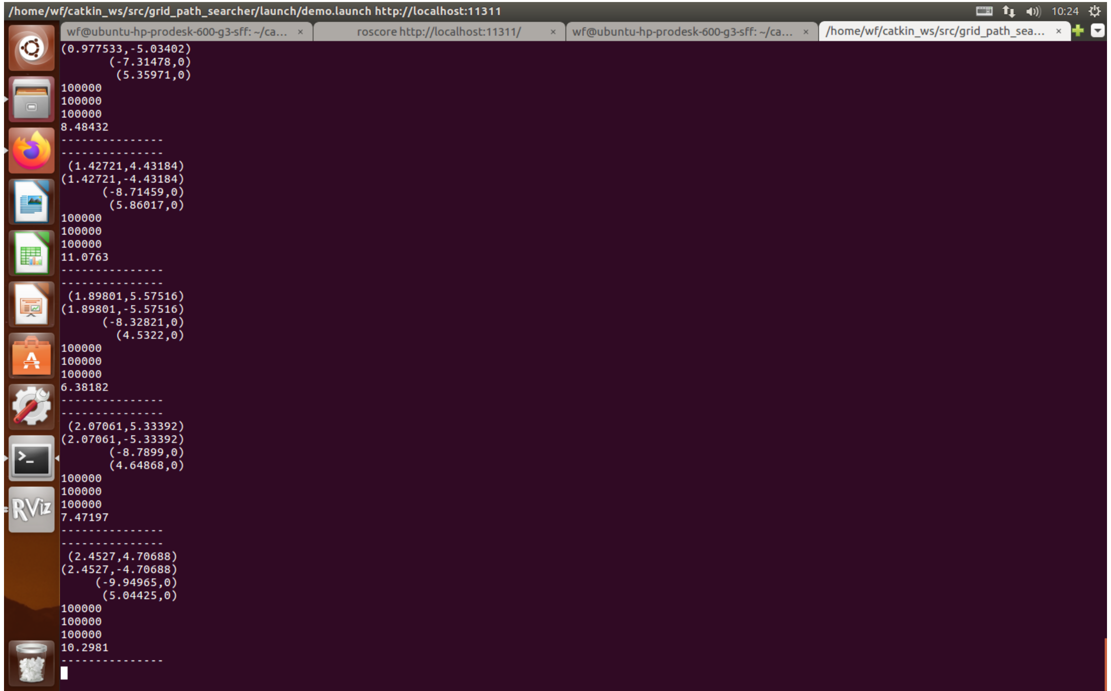


Homework2

2、编译运行指令及结果

以上该部分作业在课程给定代码包的基础上，有参考：
https://blog.csdn.net/weixin_44558122/article/details/115666344

输出的OBVP的解的根和轨迹的cost为：



感谢各位聆听！

Thanks for Listening