

# 机器人路径规划

作业五 - 小车轨迹 MPC

张 文泰

学号: 21009101463

## 一、程序分析

```
VectorX compensateDelay(const VectorX& x0) {
    VectorX x0_delay = x0;
    if (delay_ == 0)
    {
        return x0_delay;
    }
    Eigen::MatrixXd BB, AA, gg, x0_pred;
    int tau = std::ceil(delay_ / dt_);
    BB.setZero(n * tau, m * tau);
    AA.setZero(n * tau, n);
    gg.setZero(n * tau, 1);
    x0_pred.setZero(n * tau, 1);
    double s0 = s_.findS(x0.head(2));
    double phi, v, delta;
    double last_phi = x0(2);
    for (int i = 0; i < tau; ++i) {
        callinPoint(s0, phi, v, delta);
        if (phi - last_phi > M_PI) {
            phi -= 2 * M_PI;
        } else if (phi - last_phi < -M_PI) {
            phi += 2 * M_PI;
        }
        last_phi = phi;
        linearization(phi, v, delta);
        if (i == 0) {
            BB.block(0, 0, n, m) = Bd_;
            AA.block(0, 0, n, n) = Ad_;
            gg.block(0, 0, n, 1) = gd_;
        } else {
            BB.block(i * n, i * m, n, m) = Bd_;
            for (int j = i - 1; j >= 0; --j) {
                BB.block(i * n, j * m, n, m) = Ad_ * BB.block((i - 1) * n, j * m, n, m);
            }
            AA.block(i * n, 0, n, n) = Ad_ * AA.block((i - 1) * n, 0, n, n);
            gg.block(i * n, 0, n, 1) = Ad_ * gg.block((i - 1) * n, 0, n, 1) + gd_;
        }
    }
    Eigen::VectorXd uu(m * tau, 1);
    for (double t = delay_; t > 0; t -= dt_) {
        int i = std::ceil(t / dt_);
        uu.coeffRef((tau - i) * m + 0, 0) = historyInput_[i - 1][0];
        uu.coeffRef((tau - i) * m + 1, 0) = historyInput_[i - 1][1];
    }
    x0_pred = BB * uu + AA * x0 + gg;
    x0_delay = x0_pred.block((tau - 1) * n, 0, n, 1);
    return x0_delay;
}
```

compensateDelay 函数

该函数实现了一个延时补偿功能，用于根据系统延迟来估计当前状态的补偿状态。函数的主要思路框架如下：

1. **函数参数：**x0 为当前状态的向量。如果延迟 delay\_ 为 0，则直接返回当前状态 x0。
2. **初始化矩阵和变量：**初始化矩阵 BB、AA、gg 和 x0\_pred，它们用于存储状态空间模型的线性化信息和延迟预测。计算延迟对应的时间步数 tau，并根据时间步数初始化相关的矩阵。
3. **计算线性化信息：**循环 tau 次，计算每个时间步对应的线性化点，并

使用 `linearization` 函数计算对应的线性化矩阵，将这些矩阵填充到 `BB`、`AA` 和 `gg` 中，以便后续使用。

4. **历史输入重构**: 从历史输入 `historyInput_` 中重构输入向量 `uu`，其中 `historyInput_` 是记录了历史输入的容器，然后根据时间步长 `dt_` 和延迟 `delay_`，将历史输入插入到 `uu` 中。
5. **状态预测**: 使用补偿前的状态 `x0`、历史输入 `uu`，以及线性化信息 `BB`、`AA` 和 `gg`，通过状态空间模型预测延迟期间的状态变化，将预测得到的状态更新到 `x0_delay` 中，并返回。

```
void linearization(const double &phi, const double &v, const double &delta) {
    Ad_ << 0, 0, -v*sin(phi), cos(phi),
           0, 0, v*cos(phi), sin(phi),
           0, 0, 0, tan(delta) / ll_,
           0, 0, 0, 0;
    Bd_ << 0, 0,
           0, 0,
           0, v/(ll_*pow(cos(delta),2)),
           1, 0;
    gd_ << v*phi*sin(phi), -v*phi*cos(phi), -v*delta/(ll_*pow(cos(delta),2)), 0;

    Ad_ = MatrixA::Identity() + dt_ * Ad_;
    Bd_ = dt_ * Bd_;
    gd_ = dt_ * gd_;
    return;
}
```

#### Linearization 函数

这段函数用于状态空间系统线性化。它通过接收车辆的航向角(`phi`)、速度(`v`)和转向角(`delta`)计算状态空间模型的离散化矩阵、输入矩阵和扰动项。

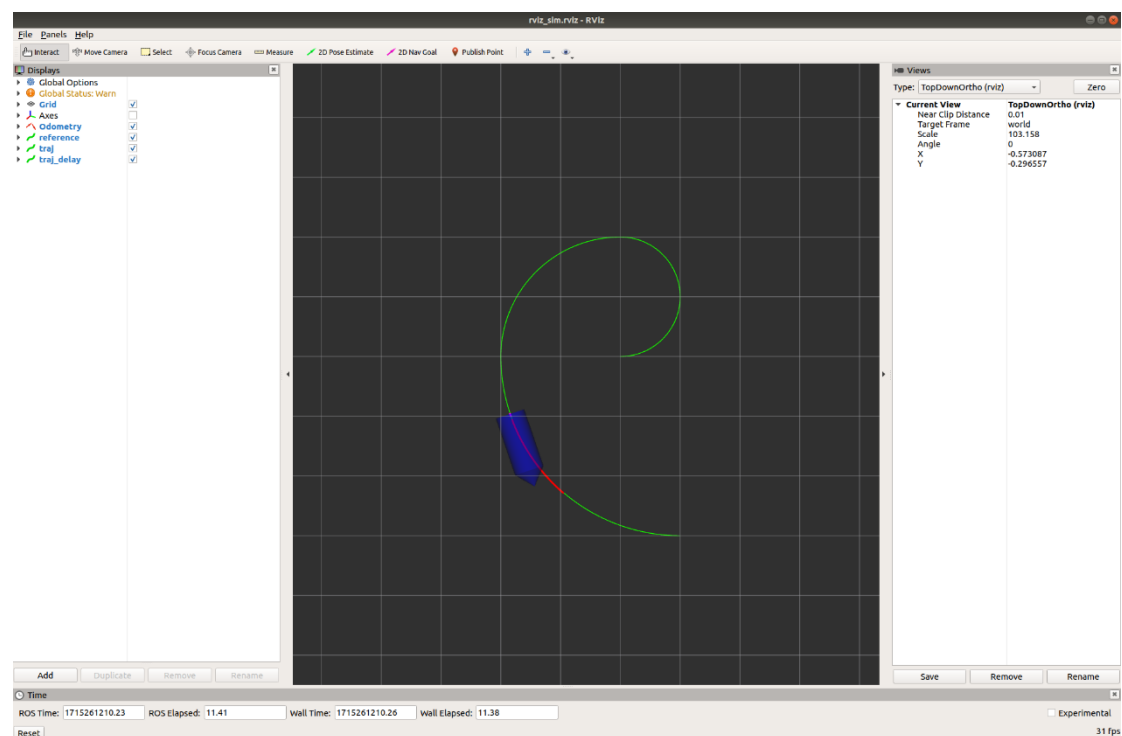
在函数内部，首先初始化了三个矩阵 `Ad`、`Bd` 和 `gd`，根据输入的参数，计算并填充了这些矩阵的值。然后对离散化的矩阵 `Ad`、`Bd` 和 `gd` 进行了时间步长的更新，因为在实际控制中，系统模型需要离散化来进行数字控制。

## 二、实验过程

```
carpent@carpent-virtual-machine: ~/ROS_car
File Edit View Search Terminal Help
[ 45%] Generating EusLisp manifest code for car_msgs
[ 45%] Built target car_msgs_generate_messages_nodejs
Scanning dependencies of target car_msgs_generate_messages_lisp
[ 54%] Generating Lisp code from car_msgs/CarCmd.msg
[ 54%] Built target car_msgs_generate_messages_lisp
[ 54%] Built target car_msgs_generate_messages_cpp
[ 63%] Generating Python msg __init__.py for car_msgs
[ 63%] Built target car_msgs_generate_messages_py
[ 63%] Built target car_msgs_generate_messages_eus
Scanning dependencies of target car_msgs_generate_messages
Scanning dependencies of target mpc_car_nodelet
Scanning dependencies of target car_simulator_nodelet
[ 63%] Built target car_msgs_generate_messages
[ 72%] Building CXX object car_simulator/CMakeFiles/car_simulator_nodelet.dir/src/car_simulator_nodelet.cpp.o
[ 81%] Building CXX object mpc_car/CMakeFiles/mpc_car_nodelet.dir/src/mpc_car_nodelet.cpp.o
[ 90%] Linking CXX shared library /home/carpent/ROS_car/devel/lib/libcar_simulator_nodelet.so
[ 90%] Built target car_simulator_nodelet
[100%] Linking CXX shared library /home/carpent/ROS_car/devel/lib/libmpc_car_nodelet.so
[100%] Built target mpc_car_nodelet
carpent@carpent-virtual-machine: ~/ROS_car$
```

环境配置成功

## 三、实验结果



程序运行结果

程序成功运行，地图显示在 rviz 中并开始自动规划路径。