6. Robot trajectory tracking

6.1 Program function description

After the program runs, the car will follow the trajectory given in the.csv file. Starting rviz on the virtual machine side allows for visualization of trajectory tracking.

In addition, the [L1] button on the handle locks/turns on the car's motion controls. When motion control is enabled, the function is locked; This function can be turned on when the motion control is locked.

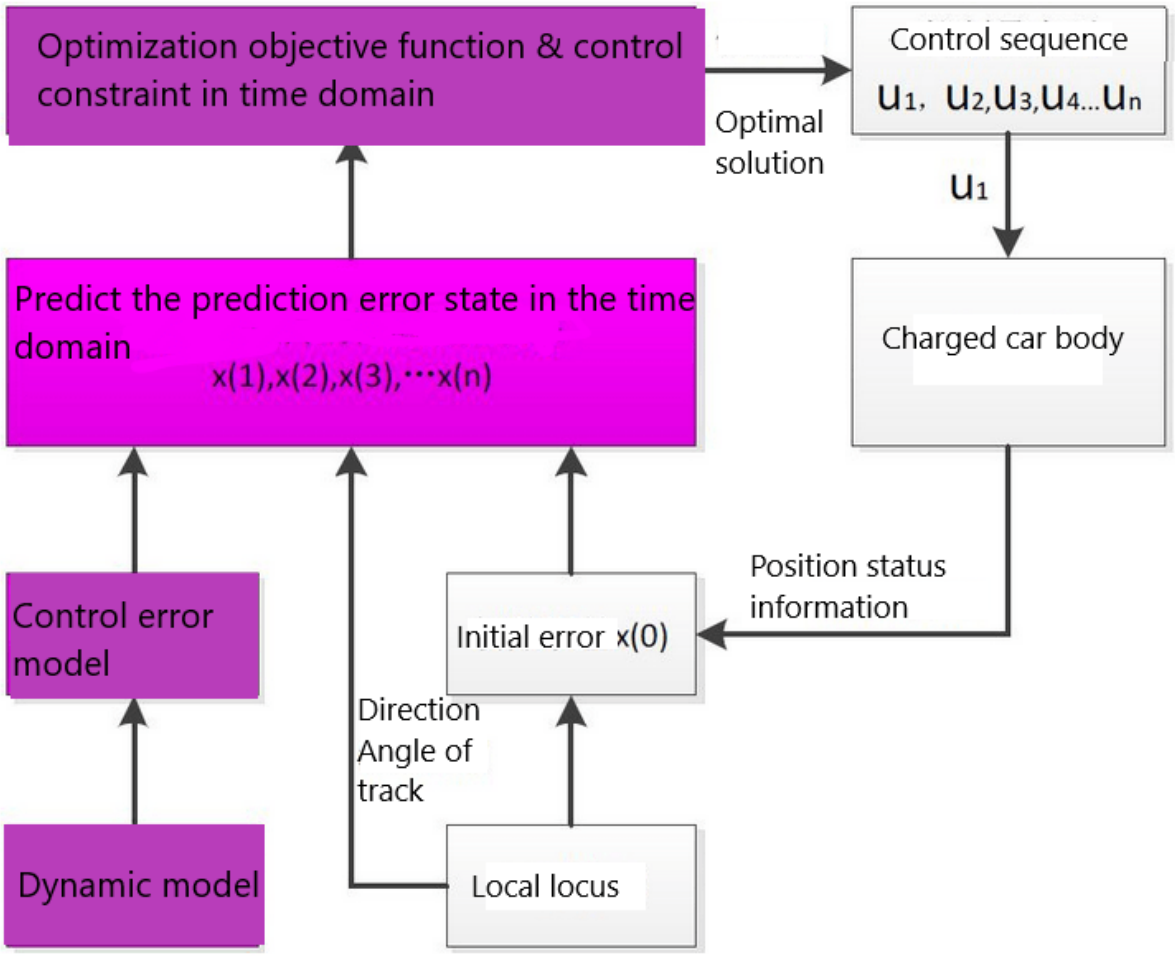
6.2 Introduction to Principle

PurePuesuit and MPC algorithms are commonly used control algorithms in automatic driving, aiming to smooth the trajectory points at the planning point and satisfy the kinematic trajectory of the vehicle itself.

Before talking about the implementation of the two algorithms in detail, it is necessary to understand that the two mainstream trajectory tracking algorithms are divided into geometric tracking and model-based tracking methods.

In this implementation, it can be basically regarded as known coordinate information, including (x,y,yaw) and curvature Kappa, and the steering Angle of the control quantity can be calculated, that is, the lateral amount of movement, so that the car can run smoothly in the track.

Take the MPC as an example, as shown below:



PurePursuit

The pure tracking algorithm puts forward the concept of "pre-sight distance", and finds the qualified target waypoint in the target trajectory according to the pre-sight distance. The judgment logic is to find which point on the target trajectory and the relative distance between the current vehicle position is equal to the pre-sight distance, then the point is the target point at the current moment. The control target is to calculate how much front wheel deflection Angle can make the current car position move to the target position. On this basis, may wish to briefly look at the core point, what is the pre-sight distance.

In simple terms, the pre-sight distance is like finding a tracking reference point when driving, for example, when driving on a straight road, we will choose a higher speed, and we are used to considering the point far ahead as a tracking reference point; When driving on a curve, it will choose to slow down, and it is customary to choose a closer point as a tracking reference point. So this value is a value that we can set ourselves. The setting of this value will also greatly affect the movement of the car. For example, in the findCloestindex function in this example, we find the nearest point based on the preview distance:

```
//公式 : ld=l+kvld=l+kv (过于依赖前视距离的选取，可以采用动态前视距离ld=l+kvld=l+kv其中l、kl、k为系数，根据速度调整前视距离)
// Formula: ld=l+kvld=l+kv (too dependent on the selection of forward looking distance, can use dynamic forward looking distance ld=l+kvld=l+kv, where l, kl, k are the coefficients, adjust the forward looking distance according to the speed)
double ld = kv*linear_x+ld0;
double ld_now = 0;
while(ld_now<ld && index<=(int)xr.size()) {
    double dx_ref = xr[index+1] - xr[index];
    double dy_ref = yr[index+1] - yr[index];
    ld_now += sqrt(pow(dx_ref,2)+pow(dy_ref,2));
    index++;
}
```

After obtaining this point, calculate the Angle according to the model limitations of the car itself:

```
double PurePursuit::calSteeringAngle(double alpha,double ld)
{
    double steer = atan2(2*track*sin(alpha),ld);
    if (steer > M_PI) {
        steer -= M_PI*2;
        } else if (steer < (-M_PI)) {
        steer += M_PI*2;
        }
    return steer;
}
```

MPC

The MPC is more complex to solve. Show the code frame directly:

1. Obtain car body parameters.
2. Select state quantity and control quantity.
3. The AB matrix is discretized, and the perturbation matrix is added to Apollo.
4. Set the QR matrix.
5. Set the number of predicted steps and augment the ABQR.
6. Set the constraints of relevant parameters and obtain the values of each sensor.
7. Solve the control quantity u, and finally get u[0].

> For more specific logic, see this link:https://blog.csdn.net/u013914471/article/details/83824490

6.3 Application Code Reference path

After SSH connection car, the location of the function source code is located at,

```
#Purepursuit
/userdata/yahboomcar_ws/src/yahboomcar_autonomous/Purepursuit/src/purepursuit.cc
#MPC
/userdata/yahboomcar_ws/src/yahboomcar_autonomous/MPC/launch/yahboomcar_mpc.launch.py
```

Where the trace's.csv file is located,

```
/userdata/yahboomcar_ws/src/yahboomcar_autonomous/waypoints/path.csv
```

6.4. Program Startup

After SSH connects to the car, terminal input,

```
#启动底盘# Start chassis
ros2 launch yahboomcar_bringup yahboomcar_bringup_launch.py
#Purepursuit
ros2 run yahboomcar_autonomous purepursuit_node
#MPC
ros2 launch yahboomcar_autonomous yahboomcar_mpc.launch.py
```

Enter on the VM.

```
rviz2
```