

原 Apollo代码学习(五)—横纵向控制

2018年11月07日 14:51:48 follow轻尘 阅读数：4739

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/u013914471/article/details/83748571>

Apollo代码学习—横纵向控制

前言

纵向控制

横向控制

前馈控制

注意

反馈控制

总结

补充 2018.11.28

前言

在我的第一篇博文：Apollo代码学习(一)—控制模块概述中，对横纵向控制做了基本概述，现在做一些详细分析。

纵向控制

纵向控制主要为速度控制，通过控制刹车、油门、档位等实现对车速的控制，对于自动挡车辆来说，控制对象其实就是 刹车 和 油门。

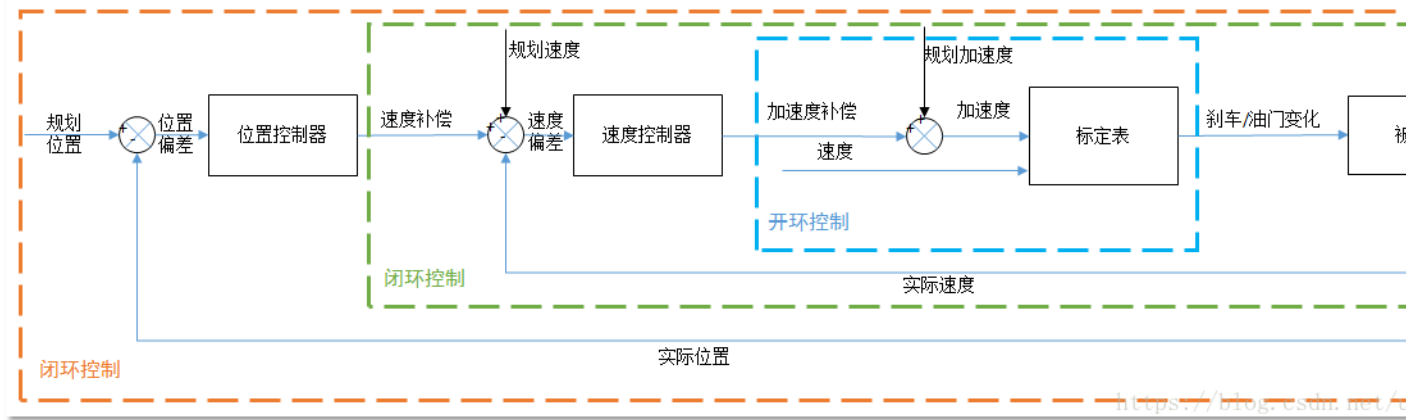


图1 纵向控制

Apollo纵向控制的工作原理如图1所示。它主要由“位移-速度闭环PID控制器”、“速度-加速度闭环PID控制器”和“速度-加速度-刹车/油门”开环控制

位置PID闭环控制器		速度PID闭环控制器		标定表开环控制器	
输入	期望位置+当前实际位置	输入	速度补偿+当前位置速度偏差	输入	加速度补偿+规划加速度，当前速度
输出	速度补偿量	输出	加速度补偿量	输出	油门/刹车控制量

以不加预估的控制为例，apollo纵向控制中计算纵向误差的原理：

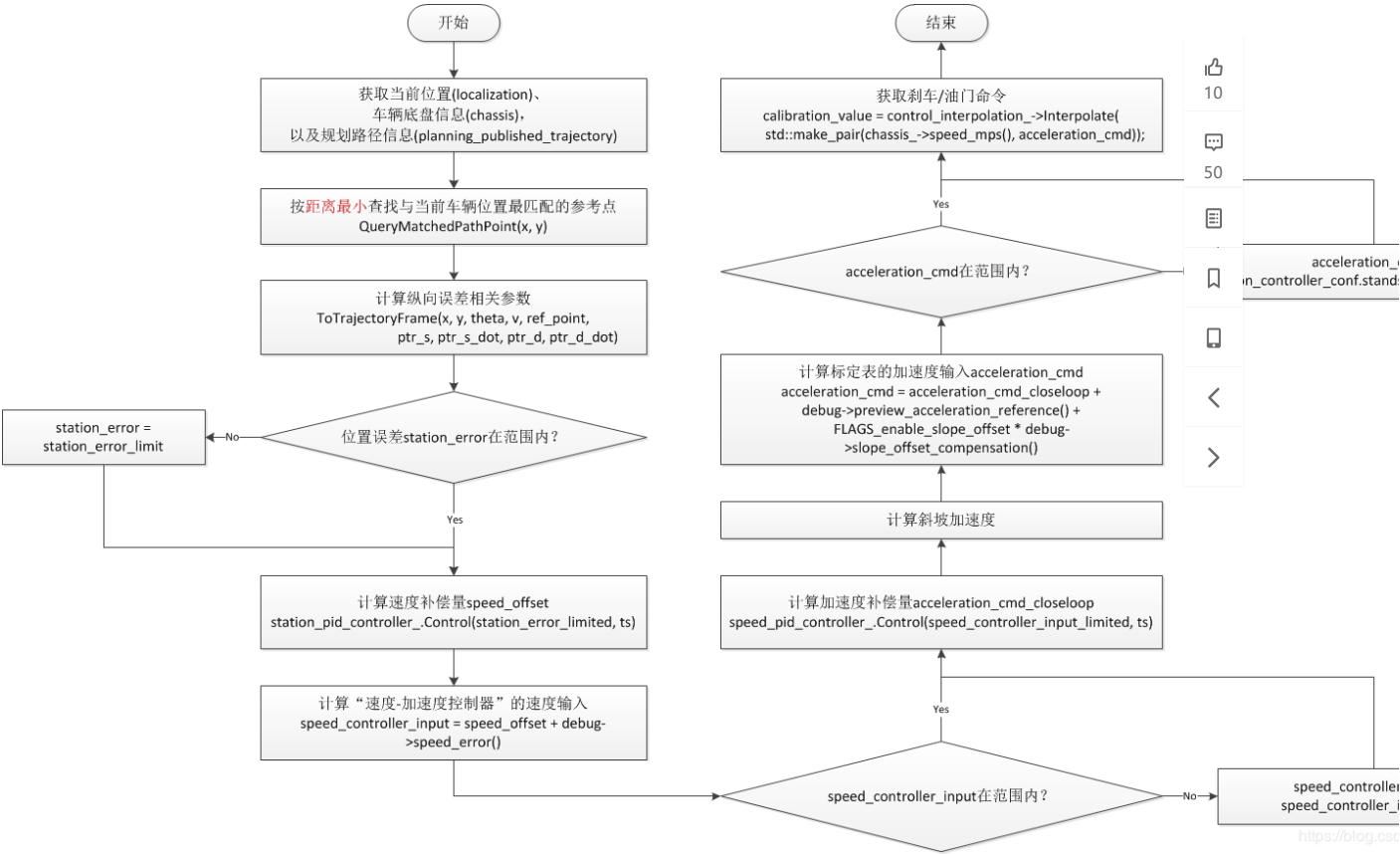


图2 纵向控制计算流程

其中，重要的是纵向误差的计算，纵向误差包含两个状态变量：

- ☑ 速度误差(speed_error)
- ☑ 位置误差(station_error)

跟计算纵向误差相关的函数主要是：

```
1 void TrajectoryAnalyzer::ToTrajectoryFrame(const double x, const double y,
2                                           const double theta, const double v,
3                                           const PathPoint &ref_point,
4                                           double *ptr_s, double *ptr_s_dot,
5                                           double *ptr_d,
6                                           double *ptr_d_dot)
```

计算的原理和Apollo代码学习(三)—车辆动力学模型中的图5类似，由Optimal Trajectory Generation for Dynamic Street Scenarios in a Frenet Fr转化而来：

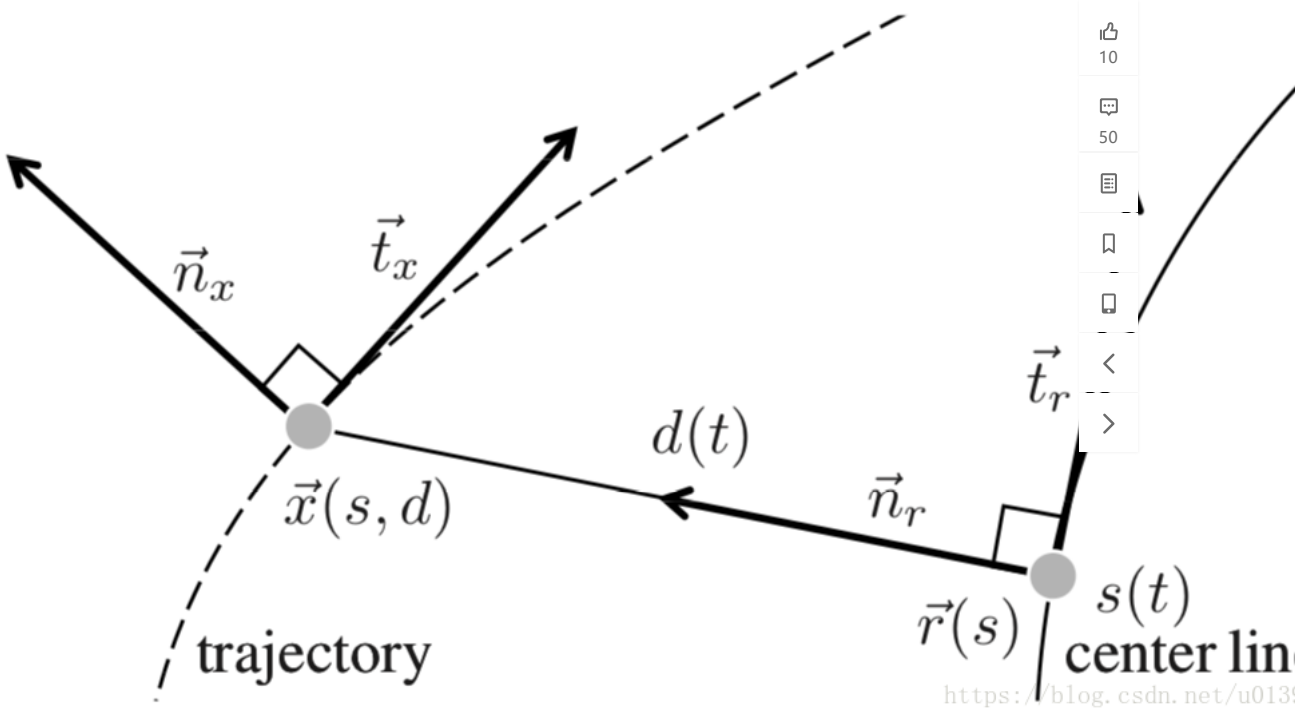


图3 Frenét框架下的轨迹

与图3的区别是，图4不假设由Real_point指向Desire_point的向量和V_x一定垂直，因为在trajectory_analyzer.cc的注释中有：

```
1 // reference: Optimal trajectory generation for dynamic street scenarios in a
2 // Frenet Frame,
3 // Moritz Werling, Julius Ziegler, Sören Kammel and Sebastian Thrun, ICRA 2010
4 // similar to the method in this paper without the assumption the "normal"
5 // vector
6 // (from vehicle position to ref_point position) and reference heading are
7 // perpendicular.
```

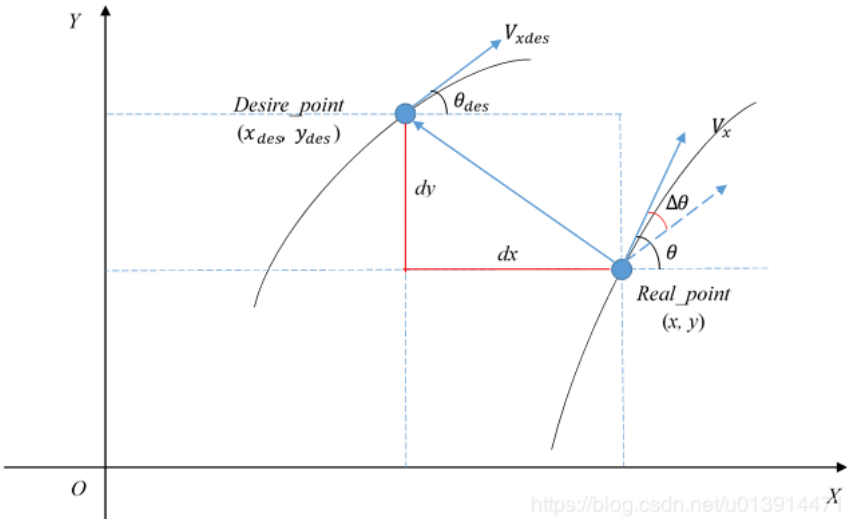


图4 Frenét框架下的误差计算

位置误差的计算：

```
1 // trajectory_analyzer.cc
2 double dx = x - ref_point.x();
3 double dy = y - ref_point.y();
4
5 double dot_rd_nd = dx * cos_ref_theta + dy * sin_ref_theta;
6 *ptr_s = ref_point.s() + dot_rd_nd;
7
```

```

8 // lon_controller.cc
9 // s_matched即上面的ptr_s
10 debug->set_station_error(reference_point.path_point().s() - s_matched);

```



10



50



由此可知，位置误差为：

$$station_error = -(dx * \cos \theta_{des} + dy * \sin \theta_{des})$$

速度误差的计算：

```

1 // trajectory_analyzer.cc
2 double cross_rd_nd = cos_ref_theta * dy - sin_ref_theta * dx;
3 *ptr_d = cross_rd_nd;
4
5 double delta_theta = theta - ref_point.theta();
6 double cos_delta_theta = std::cos(delta_theta);
7 double sin_delta_theta = std::sin(delta_theta);
8
9 double one_minus_kappa_r_d = 1 - ref_point.kappa() * (*ptr_d);
10 if (one_minus_kappa_r_d <= 0.0) {
11     AERROR << "TrajectoryAnalyzer::ToTrajectoryFrame "
12             "found fatal reference and actual difference. "
13             "Control output might be unstable:"
14             << " ref_point.kappa:" << ref_point.kappa()
15             << " ref_point.x:" << ref_point.x()
16             << " ref_point.y:" << ref_point.y() << " car x:" << x
17             << " car y:" << y << " *ptr_d:" << *ptr_d
18             << " one_minus_kappa_r_d:" << one_minus_kappa_r_d;
19     // currently set to a small value to avoid control crash.
20     one_minus_kappa_r_d = 0.01;
21 }
22
23 *ptr_s_dot = v * cos_delta_theta / one_minus_kappa_r_d;
24
25 // lon_controller.cc
26 // s_dot_matched即上面的ptr_s_dot
27 debug->set_speed_error(reference_point.v() - s_dot_matched);

```

由此可知，速度误差为：

$$speed_error = V_{des} - V * \cos \Delta\theta / k$$

其中， V_{des} 为期望车辆线速度， V 为当前车辆线速度， $\Delta\theta$ 为航向误差， k 为系数，即代码中的 $one_minus_kappa_r_d$ 。

求得位移误差和速度误差后，结合“位移-速度闭环PID控制器”和“速度-加速度闭环PID控制器”，求得刹车/油门标定表的两个输入量：速度和加速计算在标定表中查找相应的控制命令值。

Apollo所用的PID控制器就是一般的位置型PID控制器：

$$u(k) = K_P e(k) + K_I \sum_{i=0}^k e(i) + K_D [e(k) - e(k-1)] + u(0)$$

```

1 // 微分项
2 diff = (error - previous_error_) / dt;
3 // 积分项
4 integral_ += error * dt * ki_;
5 // PID控制器输出
6 output = error * kp_ + integral_ + diff * kd_;

```

在实际调参过程中先调节P参数，再调节D参数，可能会更快达到想要的效果。感谢Apollo开发者群里哈工大的海洋同学在PID调参过程中给我的指点。

横向控制

横向控制主要控制航向，通过改变方向盘扭矩或角度的大小等，使车辆按照想要的航向行驶。Apollo中的横向控制主要由前馈开环控制器和反馈闭环控

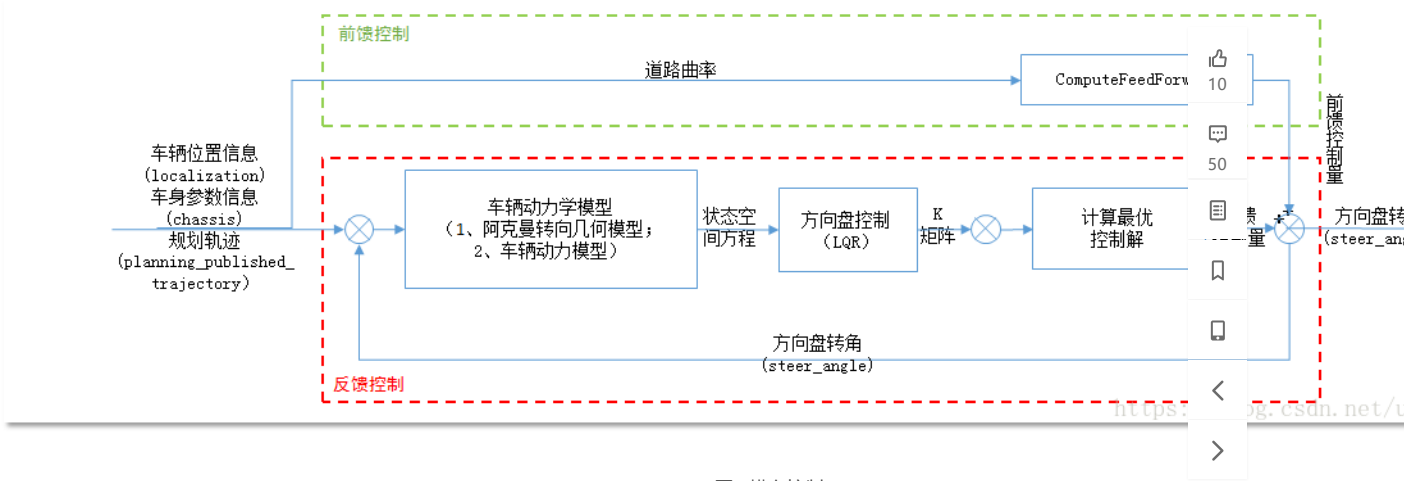


图5 横向控制

前馈控制

在之前的博文中提到，车辆的系统的一般状态方程为：

$$\dot{x} = Ax + Bu$$

其中， u 为前轮转角 δ ，但由于道路曲率 $\dot{\varphi}_{des}$ 的存在，公式4应写为

$$\dot{x} = Ax + B\delta + B_2\dot{\varphi}_{des}$$

当车辆在曲线道路上行驶时，并不能完全消除跟踪误差，因此引入和道路曲率相关的前馈控制器以帮助消除跟踪误差，由代码可以看出：

```
1 | const double steer_angle_feedforward = ComputeFeedForward(debug->curvature());
```

输入量只有一个曲率，计算的依据：

$$\delta_{ff} = \frac{L}{R} + K_V a_y - k_3 \left[\frac{\ell_r}{R} - \frac{\ell_f}{2C_{\alpha r}} \frac{mV_x^2}{R\ell} \right]$$

公式出处：Vehicle Dynamics and Control 第3章，该章节对方向盘控制做了详解。

公式6中 δ_{ff} 为前轮转角，需要乘以传动比，然后转换为方向盘转动率。

```
1 | const double steer_angle_feedforwardterm =
2 |     (wheelbase_ * ref_curvature + kv * v * v * ref_curvature -
3 |      matrix_k_(0, 2) *
4 |      (lr_ * ref_curvature -
5 |       lf_ * mass_ * v * v * ref_curvature / 2 / cr_ / wheelbase_)) *
6 |     180 / M_PI * steer_transmission_ratio_ /
7 |     steer_single_direction_max_degree_ * 100;
```

其中， $ref_curvature$ 为曲率， $ref_curvature = \frac{1}{R}$ ， $k_3 = matrix_k_(0, 2)$ ， $matrix_k_$ 为增益矩阵，由LQR求得。

注意

有博友提示此处前馈控制的代码和公式6有出入，经比较确实有误，参看modules\control\conf\lincoln.pb.txtapollo中的配置信息， $cr_$ 和 $cf_$ 均为155494.663；参看Vehicle Dynamics and Control一书中3.1章节的例子，单侧转动惯量一般为80000左右，二者为2倍的关系；结合代码中其他用到 $cr_$ 和 $cf_$ 的地方，以及书中公式，可能是代码有误，这需要向开发者提交求证。

反馈控制

由于前馈控制用于消除有道路曲率引起的误差，因此对于反馈控制，车辆的状态方程仍可以用公式4表示：

$$\dot{x} = Ax + Bu$$

为了达到性能上的需求，并使控制系统构成全状态反馈系统，需要设计一个反馈控制器 $u = -Kx$ 。基于车辆系统的状态方程，并结合代码进行分析，预估，即 `preview_window=0`。

首先，控制系统的状态变量有四个：

- ☑ 横向误差 *lateral_error*
- ☑ 横向误差率 *lateral_error_rate*
- ☑ 航向误差 *heading_error*
- ☑ 航向误差率 *heading_error_rate*

横向误差的计算参见上篇博文：Apollo代码学习(三)—车辆动力学模型。
代码中，`matrix_state_`为状态变量：

```
1 | matrix_state_ = Matrix::Zero(matrix_size, 1);
2 |
3 | matrix_state_(0, 0) = debug->lateral_error();
4 | matrix_state_(1, 0) = debug->lateral_error_rate();
5 | matrix_state_(2, 0) = debug->heading_error();
6 | matrix_state_(3, 0) = debug->heading_error_rate();
```

对应的状态矩阵A，即代码中的 `matrix_a_`

```
1 | matrix_a_ = Matrix::Zero(basic_state_size_, basic_state_size_);
2 | matrix_ad_ = Matrix::Zero(basic_state_size_, basic_state_size_);
3 | matrix_adc_ = Matrix::Zero(matrix_size, matrix_size);
4 |
5 | matrix_a_(0, 1) = 1.0;
6 | matrix_a_(1, 2) = (cf_ + cr_) / mass_;
7 | matrix_a_(2, 3) = 1.0;
8 | matrix_a_(3, 2) = (lf_ * cf_ - lr_ * cr_) / iz_;
9 |
10 | matrix_a_coeff_ = Matrix::Zero(matrix_size, matrix_size);
11 | matrix_a_coeff_(1, 1) = -(cf_ + cr_) / mass_;
12 | matrix_a_coeff_(1, 3) = (lr_ * cr_ - lf_ * cf_) / mass_;
13 | matrix_a_coeff_(2, 3) = 1.0;
14 | matrix_a_coeff_(3, 1) = (lr_ * cr_ - lf_ * cf_) / iz_;
15 | matrix_a_coeff_(3, 3) = -1.0 * (lf_ * lf_ * cf_ + lr_ * lr_ * cr_) / iz_;
```

`matrix_a_coeff_` 用于更新状态矩阵 `matrix_a_`

```
1 | void LatController::UpdateMatrix() {
2 |     const double v =
3 |         std::max(VehicleStateProvider::instance()->linear_velocity(), 0.2);
4 |     matrix_a_(1, 1) = matrix_a_coeff_(1, 1) / v;
5 |     matrix_a_(1, 3) = matrix_a_coeff_(1, 3) / v;
6 |     matrix_a_(3, 1) = matrix_a_coeff_(3, 1) / v;
7 |     matrix_a_(3, 3) = matrix_a_coeff_(3, 3) / v;
8 |     Matrix matrix_i = Matrix::Identity(matrix_a_.cols(), matrix_a_.cols());
9 |     matrix_ad_ = (matrix_i + ts_ * 0.5 * matrix_a_) *
10 |        (matrix_i - ts_ * 0.5 * matrix_a_).inverse();
11 | }
12 |
13 | matrix_adc_.block(0, 0, basic_state_size_, basic_state_size_) = matrix_ad_;
```

则状态矩阵 `matrix_a_` 为：

$$\mathbf{matrix_a_} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{C_f+C_r}{mV} & \frac{C_f+C_r}{m} & \frac{\ell_r C_r - \ell_f C_f}{mV} \\ 0 & 0 & 0 & 1 \\ 0 & \frac{\ell_r C_r - \ell_f C_f}{I_z V} & \frac{\ell_f C_f - \ell_r C_r}{I_z} & \frac{\ell_r^2 C_r - \ell_f^2 C_f}{I_z V} \end{bmatrix}$$

由于动力学模型基于单车模型，它将左右两侧轮胎合并为一个轮胎，因此式中的转角刚度 C_f 、 C_r 应为单边转角刚度的2倍，文中均如此。

与动力学模型中的公式24的首项系数吻合。在没有预估的情况下，`matrix_adc_`=`matrix_ad_`，`matrix_ad_` 为离散时间下的状态矩阵，采用的是**双线性法**。感谢博主LLCCCJ的友情提示。

控制矩阵B，即代码中的 `matrix_b_`，`matrix_bd_` 为离散时间下的控制矩阵：

```
1 | matrix_b_ = Matrix::Zero(basic_state_size_, 1);
2 | matrix_bd_ = Matrix::Zero(basic_state_size_, 1);
3 | matrix_bdc_ = Matrix::Zero(matrix_size, 1);
4 | matrix_b_(1, 0) = cf_ / mass_;
```

```

5 | matrix_b_(3, 0) = lf_ * cf_ / iz_;
6 | matrix_bd_ = matrix_b_ * ts_;
7 |
8 | matrix_bdc_.block(0, 0, basic_state_size_, 1) = matrix_bd_;

```

$$matrix_b_ = \begin{bmatrix} 0 \\ \frac{cf}{m} \\ 0 \\ \frac{\ell_f C_f}{I_z} \end{bmatrix}$$

与动力学模型中公式24的 δ 的系数吻合。在没有预估的情况下, $matrix_bdc_ = matrix_bd_$ 。

对于控制系统, 求其最优控制解的方法有很多, apollo用的是LQR调节器(可参考一下官方讲解视频: [线性二次调节器](#)), 它求解控制轨迹应该使得该能量函数最小。能量函数的一般形式为:

$$J = \frac{1}{2} \int_0^{\infty} (x^T Q x + u^T R u) dt$$

其中, Q 是半正定矩阵, R 为正定矩阵, 可自行设计。

设计LQR控制器, 需要求取 K 矩阵, 计算反馈矩阵 K 的过程:

1. 选择参数矩阵 Q , R ;
2. 求解Riccati 方程得到矩阵 P ;
3. 计算 $K = R^{-1} B^T P$ 。

LQR的求解过程可参看LQR, 感兴趣的可以研究一下, matlab、python等也提供的都有用于求解LQR模型的函数。

Apollo中对 Q 、 R 矩阵的定义如下:

```

1 | lqr_eps_ = control_conf->lat_controller_conf().eps();
2 | lqr_max_iteration_ = control_conf->lat_controller_conf().max_iteration();
3 |
4 | query_relative_time_ = control_conf->query_relative_time();
5 |
6 | matrix_k_ = Matrix::Zero(1, matrix_size);
7 | matrix_r_ = Matrix::Identity(1, 1);
8 | matrix_q_ = Matrix::Zero(matrix_size, matrix_size);
9 |
10 | int q_param_size = control_conf->lat_controller_conf().matrix_q_size();
11 | for (int i = 0; i < q_param_size; ++i) {
12 |     matrix_q_(i, i) = control_conf->lat_controller_conf().matrix_q(i);
13 | }
14 |
15 | matrix_q_updated_ = matrix_q_;

```

求取 K 矩阵的代码:

```


1 | if (FLAGS_enable_gain_scheduler) {
2 |     matrix_q_updated_(0, 0) =
3 |         matrix_q_(0, 0) *
4 |         lat_err_interpolation->Interpolate(
5 |             VehicleStateProvider::instance()->linear_velocity());
6 |     matrix_q_updated_(2, 2) =
7 |         matrix_q_(2, 2) *
8 |         heading_err_interpolation->Interpolate(
9 |             VehicleStateProvider::instance()->linear_velocity());
10 |     common::math::SolveLQRProblem(matrix_adc_, matrix_bdc_, matrix_q_updated_,
11 |                                     matrix_r_, lqr_eps_, lqr_max_iteration_,
12 |                                     &matrix_k_);
13 | } else {
14 |     common::math::SolveLQRProblem(matrix_adc_, matrix_bdc_, matrix_q_,
15 |                                     matrix_r_, lqr_eps_, lqr_max_iteration_,
16 |                                     &matrix_k_);
17 | }


```





其中，状态变量 `matrix_state_`、混合状态矩阵 `matrix_adc_`、混合控制矩阵 `matrix_bdc_`、增益矩阵 `matrix_k_` 是时变的，状态权重矩阵 `matrix_` 变的，控制权重矩阵 `matrix_r_`、阈值 `lqr_eps_`、最大迭代次数 `lqr_max_iteration_` 是固定的。


求出K矩阵后，即可求最优控制解：


10


50











```
1 // feedback = - K * state
2 // Convert vehicle steer angle from rad to degree and then to steer degree
3 // then to 100% ratio
4 const double steer_angle_feedback = -(matrix_k_ * matrix_state_)(0, 0) * 180 /
5                                     M_PI * steer_transmission_ratio_ /
6                                     steer_single_direction_max_degree_ * 100;
7
8 const double steer_angle_feedforward = ComputeFeedForward(debug->curvature());
9
10 // Clamp the steer angle to -100.0 to 100.0
11 double steer_angle = common::math::Clamp(
12     steer_angle_feedback + steer_angle_feedforward, -100.0, 100.0);
```

总结

本文详述了Apollo的横纵向控制，纵向控制的主体是PID控制，横向控制的主体是前馈控制器加反馈控制器，反馈控制器的设计依赖于LQR模型；Apollo基于横纵向控制设计。

由于对线性化、离散化等知识理解有限，故在此篇中未对LQR的求解做展开说明，下篇文章将尝试对非线性系统的线性化、求最优解等进行分析。

补充 2018.11.28

很多人（包括我）对于横向误差`lateral_error`和纵向误差中的位误差`station_error`的计算不太明白，我就试着去解释一下吧。

首先，对于坐标系下任一点 (x, y) ，假设如图6所示，将坐标轴绕原点向左旋转 θ 角度后的坐标值 (x', y') 满足以下等式：

$$\begin{cases} x' = OA + BC = x \cos \theta + y \sin \theta \\ y' = PC - AB = y \cos \theta - x \sin \theta \end{cases}$$

可自行推广到其他角度或象限的旋转。坐标系旋转可参考博客：[二维坐标系的转换](#)。

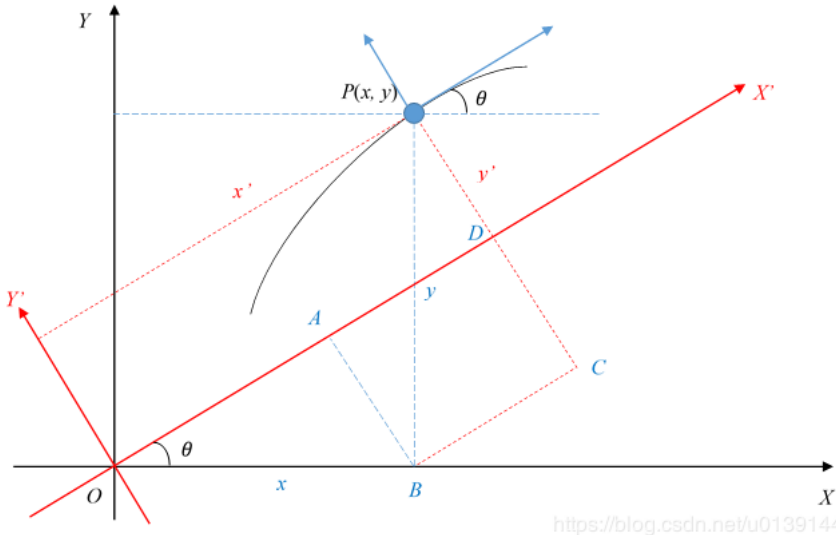


图6 坐标旋转

结合图3和代码中的注释，可得图7，其中 V_{des} 垂直于 e_1 ， e_s 为位置误差`station_error`， e_l 为横向误差`lateral_error`。

为什么 dx, dy 是实际点坐标减去参考点坐标?

个人理解：控制的目的是为了将车辆从当前点移动到期望点，分别在 x, y 方向上移动 dx, dy 的距离能到参考点，所以 dx, dy 是实际点坐标减去参考点

而怎么从规划轨迹中找对应的参考点，有两种方法，根据绝对时间或者位置最近进行匹配，

10

50

```
1 | TrajectoryPoint target_point;
2 | // 绝对时间
3 | target_point = trajectory_analyzer.QueryNearestPointByAbsoluteTime(
4 |     current_timestamp + query_relative_time_);
5 | // 相对位置
6 | target_point = trajectory_analyzer.QueryNearestPointByPosition(x, y);
```

按照最近位置寻找匹配点的话，找的其实就是参考点与实际点间 **连线距离最短** 的参考点。但这个 **连线距离最短** 中的 **距离**，指的不是纵向误差，也不是纵两点间连线的长度

$$distance = \sqrt{dx^2 + dy^2}$$

计算横纵向误差是在S-L坐标系下，而 dx, dy 是在笛卡尔坐标系下计算的，因此需要进行坐标系转换，简单的来说就是将原始坐标系进行旋转，使其与合，如图6所示。且坐标系旋转后，参考点与实际点两点间距离 $distance$ 应是不变的。

为什么将参考点进行旋转，而不是将实际点进行旋转?

个人理解：控制的目的是为了让汽车按照规划轨迹行驶，当前时刻的误差，肯定是基于参考点得来的，也就是我要找到现在实际的点相较于参考点，它的误差各是多少。因此横纵向误差的计算中的角度使用的期望航向角 θ_{des} 。

对于纵向误差的计算，可见代码[trajectory_analyzer.cc](#)以及[lon_controller.cc](#):

```
1 | // the sin of diff angle between vector (cos_ref_theta, sin_ref_theta) and
2 | // (dx, dy)
3 | double cross_rd_nd = cos_ref_theta * dy - sin_ref_theta * dx;
4 | *ptr_d = cross_rd_nd;
5 |
6 | // the cos of diff angle between vector (cos_ref_theta, sin_ref_theta) and
7 | // (dx, dy)
8 | double dot_rd_nd = dx * cos_ref_theta + dy * sin_ref_theta;
9 | *ptr_s = ref_point.s() + dot_rd_nd;
10 |
11 | double delta_theta = theta - ref_point.theta();
12 | double cos_delta_theta = std::cos(delta_theta);
13 | double sin_delta_theta = std::sin(delta_theta);
14 |
15 | *ptr_d_dot = v * sin_delta_theta;
16 |
17 | double one_minus_kappa_r_d = 1 - ref_point.kappa() * (*ptr_d);
18 | if (one_minus_kappa_r_d <= 0.0) {
19 |     AERROR << "TrajectoryAnalyzer::ToTrajectoryFrame "
20 |         "found fatal reference and actual difference. "
21 |         "Control output might be unstable:"
22 |         << " ref_point.kappa:" << ref_point.kappa()
23 |         << " ref_point.x:" << ref_point.x()
24 |         << " ref_point.y:" << ref_point.y() << " car x:" << x
25 |         << " car y:" << y << " *ptr_d:" << *ptr_d
26 |         << " one_minus_kappa_r_d:" << one_minus_kappa_r_d;
27 |     // currently set to a small value to avoid control crash.
28 |     one_minus_kappa_r_d = 0.01;
29 | }
30 |
31 | *ptr_s_dot = v * cos_delta_theta / one_minus_kappa_r_d;
32 |
33 | // 位置误差, 其中s_matched = ptr_s
34 | debug->set_station_error(reference_point.path_point().s() - s_matched);
35 | // 速度误差, 其中s_dot_matched = ptr_s_dot
36 | debug->set_speed_error(reference_point.v() - s_dot_matched);
```

纵向位置误差 $station_error$ 为

$$station_error = -(dx * \cos \theta_{des} + dy * \sin \theta_{des})$$

对于横向误差的计算，可见代码[lat_controller.cc](#):

```
1 | const double cos_target_heading = std::cos(target_point.path_point().theta());
2 | const double sin_target_heading = std::sin(target_point.path_point().theta());
3 |
4 | double lateral_error = cos_target_heading * dy - sin_target_heading * dx;
5 | if (FLAGS_enable_navigation_mode_error_filter) {
6 |     lateral_error = lateral_error_filter_.Update(lateral_error);
7 | }
8 |
9 | debug->set_lateral_error(lateral_error);
```

👍 10

💬 50

📖

🔖

📱

⏪

⏩

也就是：

$$lateral_error = dy * \cos \theta_{des} - dx * \sin \theta_{des}$$

结合公式10、公式11、公式12, $station_error$, $lateral_error$ 分别和 x' , y' 的绝对值相等。且满足坐标系旋转过后, 参实际点间的直线规律:

$$distance = \sqrt{dx^2 + dy^2} = \sqrt{station_error^2 + lateral_error^2}$$

🔍 想对作者说点什么

weixin_42939036: 你看前馈里面的代码啊, 我知道你说的那个两倍。公式6是两倍的Car, 跟代码里能对起来? 代码里是2Cr。代码前面算AB矩阵时是用单倍的Cr, 到了算公式6, 为什么变成2Cr了? (1个月前 #18楼) [查看回复\(8\)](#)

哈库拉Matata: 大神你好, Apollo的纵向控制是PID控制, 横向控制是LQR, 还有一个横纵向控制的MPC, 也是控制方向盘转角和速度, 请问MPC跟着两个模块有什么关系吗? 是做冗余用的吗? (1周前 #17楼)

renaomin: 先感谢一下楼主这样大篇幅的分析。我打算从数学的角度进行剖析, 先Mark一下。再回来 (2周前 #16楼)

weixin_41622120: 你好, 我想问个问题, 在计算纵向速度误差时, 会计算一个one minus kappa * r * d - 1 - ref.point.kappa() * (*ntr.d) 这个的物理意义是什 (1周前 #15楼) [查看 50 条热评](#)

Apollo代码学习(六)—模型预测控制(MPC) 阅读数 7055

Apollo代码学习—模型预测控制前言模型预测控制预测模型滚动优化反馈矫正前言查看Apollo中关于M... [博文](#) 来自: [follow轻尘的...](#)

无人驾驶中的决策规划控制技术 阅读数 9585

作者: 李力耘, 刘少山责编: 何永灿, 欢迎人工智能领域技术投稿、约稿、给文章纠错, 请发送邮件至... [博文](#) 来自: [CSDN 人工智能](#)

百度Apollo 中纵向控制源码解析 阅读数 1436

严正声明: 本文系作者yue597215286原创, 未经允许, 严禁转载! 上篇博文中, 我们主要介绍了co... [博文](#) 来自: [yue59721528...](#)

apollo代码学习2.2——深度解析(control) 阅读数 1805

每次遇见复杂的事情总是在先寻找一种简单明了的方式进行研究, 用一种浅显易懂的方式来表达。今天... [博文](#) 来自: [xiaolangwj的...](#)

百度apollo - Apollo代码解析: 4. control模块 阅读数 2183

0.简介: 阅读本章之前默认已经阅读了: 百度apollo-Apollo代码解析: 3.命令行参数传递googleflag... [博文](#) 来自: [DinnerHowe...](#)

【Baidu Apollo】控制 阅读数 494

文章来源: Apollo阿波罗自动驾驶开发者社区1.简介控制是驱使车辆前行的策略。对于汽车而言, 最基... [博文](#) 来自: [笑扬轩逸的博客](#)

apollo代码学习2.4——深度解析(control) 阅读数 2316

前段时间总结了一下control模块工作的大致流程, 但是还有很多遗留的问题, 上次博客也有提及到, 像... [博文](#) 来自: [xiaolangwj的...](#)

百度Apollo(一): 代码模块 阅读数 3376

百度Apollo: <https://github.com/ApolloAuto/apollo/tree/master/modules>一级目录结构重要文件... [博文](#) 来自: [听说你爱吃芒果](#)

Python从入门到实战 基础入门视频教程 (讲解超细致)

Python基础入门视频教程: 本课程从Python入门到纯Python项目实战。超100以上课时, 内容非常详... [学院](#) 讲师: [黄勇](#)

Apollo代码学习 (二) 入门 阅读数 147

百度Apollo代码学习 (二) 昨天的学习中, 我们看完了apollo代码的localization/common文件夹中的... [博文](#) 来自: [fizz.bob的博客](#)

无人驾驶汽车横向控制模型

阅读数 419

无人驾驶横向控制模型1.车辆单车二自由度模式车辆二自由度模型如下： 如对具体推导感兴趣，可以查... 博文 来自： weixin_37614...



ali是个小太阳

59篇文章

关注 排名:千里之外



skt1127

6篇文章

关注 排名:千里之外



xianxianCho

62篇文章

关注 排名:千里之外



IKE-zi

28篇文章

关注 排名:千里之外

10

50

目

🔖

📱

<

>

Apollo代码学习(四)—Windows下编译Apollo并与Carsim和Simulink联调

阅读数 2819

Apollo代码学习—Carsim+Simulink联调前言准备所需工具安装说明前言Apollo的仿真主要在Linux下... 博文 来自： follow轻尘的...

Apollo 2.5版导航模式的使用方法

阅读数 3247

Apollo2.5版导航模式的使用方法严正声明：本文系作者davidhopper原创，未经许可，不得转载。说... 博文 来自： davidhopper...

Apollo代码学习(三)—车辆动力学模型

阅读数 5643

Apollo代码学习—车辆动力学模型车辆动力学模型车辆动力学模型动力学主要研究作用于物体的力与物... 博文 来自： follow轻尘的...

无人驾驶及Apollo开源平台技术教程

美国软件是如何最终装备在中国攻击直升机上的（一）

阅读数 4332

译自：ArsTechnica，作者：SeanGallagher当西方的公司愿意卖给你所需要的技术时何必去盗取或偷窃... 博文 来自： 极光译站

汽车双移线稳定性控制

04-18

通过汽车双移线实验进行车辆横纵向稳定性控制，代码可用

下载

控制律设计详细介绍

05-19

详细介绍控制律，包括横侧向控制律设计、纵向控制律设计，硬件和软件设计等等，对有意学习的人是很好的一份资源

下载

Apollo控制

阅读数 153

控制就是驱动车辆前行的策略——转向、加速、制动 最大限度地降低与目标轨迹的偏差（cost误差最... 博文 来自： Ali_start的博客

Apollo代码学习(二)—车辆运动学模型

阅读数 5336

Apollo代码学习—车辆运动学与动力学模型前言车辆模型车辆运动学模型车辆动力学模型Apollo(阿波... 博文 来自： follow轻尘的...

coursera | Aerial Robotics 四旋翼导航与控制 笔记4

阅读数 406

写在前面这门课上完花了挺久，晒一个证书，感觉这证书还挺帅的。本节的内容主要是对四旋翼的很多... 博文 来自： sn1979c的博客

无人驾驶汽车系统入门（二）——高级运动模型和扩展卡尔曼滤波

阅读数 1万+

前言：上一篇文章的最后我们提到卡尔曼滤波存在着一个非常大的局限性——它仅能对线性的处理模型... 博文 来自： AdamShan的...

基于动力学模型的智能车辆横、纵向及综合控制策略研究

11-09

以智能车辆横、纵向动力学模型为基础，以设计可行的、稳定的智能车辆横、纵向及综合控制系统为目的进行研究

下载

java学习：什么是横向越权？什么是纵向越权？

阅读数 636

优秀文章参考：横向越权与纵向越权 博文 来自： ali是个小太阳...

window.print打印时设置打印布局（纵向、横向）

阅读数 4703

使用css的@page可设置，纵向：@page{size:portrait;}横向：@page{size:landscape;} 博文 来自： skt1127的博客

Vehicle Dynamics and Control

10-23

Rajamani R. Vehicle Dynamics and Control[M]. Springer Science, 2006 主要讲述了车辆横纵向控制原理，对车辆运动学模型、动力学模型...

下载

Apollo自动驾驶入门课程第⑧讲 — 规划（下）

阅读数 974

目录1、路径-速度解耦规划2、路径生成与选择3、ST图4、速度规划5、优化6、路径-速度规划的轨迹... 博文 来自： 10点43的博客

Apollo配置中心


阅读数 631


更多干货 分布式实战（干货） springcloud实战（干货） mybatis实战（干货） springboot实战（干... 博文 来自： 架构师的成长...


android4.4增加ethernet 文章 的补充内容


阅读数 5183


原文网址:<http://blog.csdn.net/jingxia2008/article/details/26591005#增加ethernet主要是三部分>: ... 博文 来自: [xiaoyaofriend...](#)


10


50











apollo框架学习

阅读数 995

目标: 使用apollo无人驾驶框架搭建无人驾驶车辆, 汽车为哈弗H8, 系统为ubuntu16.04, 环境... 博文 来自: [林小川的博客](#)

apollo代码学习1

阅读数 2289

接触到百度无人驾驶开源代码apollo是导师推荐学习的, 在摸索了一个多月后, 终于有了一些眉目, 对... 博文 来自: [xiaolangwj的...](#)

Apollo感知模块算法详解

阅读数 3384

原文出处: https://github.com/ApolloAuto/apollo/blob/master/docs/specs/perception_apollo_... 博文 来自: [xiangxianghe...](#)

apollo 运动规划算法解析

阅读数 2689

参考:apollo坐标系说明<https://blog.csdn.net/davidhopper/article/details/79162385apollo运动规...> 博文 来自: [yangfan111的...](#)

对Github中Apollo项目进行版本控制的方法

阅读数 1777

本文介绍对Github中Apollo项目进行版本控制涉及到的各方面内容。本文使用的操作系统为Ubuntu16... 博文 来自: [davidhopper...](#)

Apollo计划：源码分析一：模块功能

阅读数 324

1.概括Apollo源码主要是c++实现的, 也有少量python, 主要程序在apollo/modules目录中, 共18个... 博文 来自: [微笑passking ...](#)

无人驾驶汽车系统入门（二十一）——基于Frenet优化轨迹的无人车动作规划方法

阅读数 8810

动作规划动作在无人车规划模块的最底层, 它负责根据当前配置和目标配置生成一序列的动作, 我们前... 博文 来自: [AdamShan的...](#)

无人驾驶之控制

阅读数 398

智能驾驶汽车的车辆控制技术旨在环境感知技术的基础之上, 根据决策规划出目标轨迹, 通过纵向和... 博文 来自: [He3he3he的博...](#)

前馈控制+PID


阅读数 917

转: <https://blog.csdn.net/u013528298/article/details/80435009> 博文 来自: [eric_e的博客](#)

自动控制原理01 基本概念

阅读数 172

一自动控制系统的组成自动控制装置: 即控制器被控对象: 常见的有电机、锅炉二 自动控制系统的基... 博文 来自: [m0_37196973...](#)



真传奇不是靠充的！装备全靠捡，0付费复古传奇！

PB 打印机打印横纵向设置

阅读数 346

`dw_print.settransobject(sqlca)dw_print.Object.DataWindow.Print.Orientation = 1 //默认横向 edi... 博文 来自: xianxianCho的...`

charts框架 横向 纵向柱状图

阅读数 4317

实现效果如下, 除了基础的柱状图实现, 主要增加了, x,y轴的自定义title, 可以显示汉字而不是数值。... 博文 来自: [lee727n的博客](#)

PB关于打印机纵向横向打印的设置

阅读数 7523

`dw_2.Object.DataWindow.Print.Orientation = 0 横向dw_2.Object.DataWindow.Print.Orientati... 博文 来自: 我的博客我作主`

apollo3.0 canbus部分源码解析

阅读数 1210

最近一直想要调整下底层can通信部分的架构, 想了一些适配的方案, 细节部分还是有些模糊, 想试着... 博文 来自: [liu3612162的...](#)

横向越权和纵向越权

阅读数 708

横向越权: 横向越权指的是攻击者尝试访问与他拥有相同权限的用户的资源 纵向越权: 纵向越权指的是... 博文 来自: [韩帅的博客](#)



开局一把刀，神装全靠秒！传奇我就服这款

apollo代码学习2.3——深度解析(control)

阅读数 2992

每次遇见复杂的事情总是在先寻找一种简单明了的方式进行研究, 用一种浅显易懂的方式来表达。今天... 博文 来自: [xiaolangwj的...](#)

Apollo control 模块源码解析 阅读数 1849

Apollocontrol模块接收车辆的底盘(chassis),定位(localization),行驶轨迹(planning)信息,输出车辆的控... [博文](#) 来自: [yue59721528...](#)

携程 Apollo 配置中心 | 学习笔记 序章 阅读数 3825


Apollo 携程Apollo配置中心 目录导航 携程Apollo配置中心|学习笔记（一）|Apollo配置中心简... [博文](#) 来自: [Coder编程的...](#)

Apollo学习（一）Apollo初学入门 阅读数 8221

前言最近公司项目开始用微服务相关的技术设计apollo，自己也是初学者就边学边总结，部分内容来自... [博文](#) 来自: [千年似流水的...](#)

sql语句纵向转横向 阅读数 7776

mysql中的casewhen语句查询结果问题数据库中的测试数据现在我想让他显示的结果是：姓名语文数... [博文](#) 来自: [犀利虫的专栏](#)



我的天！一刀一怪，全屏掉落，神装爆不停！

一本非常好的车辆动力学方面的书，车辆动力学及其控制

这是国内少有的写的比较好的有关车辆动力学及其控制方面的书，适合于学习车辆工程的本科、研究生等学习和研究。

Apollo自动驾驶入门课程第⑩讲 — 控制（下） 阅读数 771

目录1、线性二次调节器2、模型控制预测3、总结本文转自微信公众号：Apollo开发者社区原创：阿波... [博文](#) 来自: [10点43的博客](#)

如何用adb连接android手机？（我的亲自经历）----- 顺便说说unable to connect... 阅读数 16万+

adb是什么呢？我就不多说了，对于搞android开发的人来说，一定不陌生。本文讲述如何用adb来... [博文](#) 来自: [stpeace的专栏](#)

Python下Opencv使用笔记（七）（图像梯度与边缘检测） 阅读数 1万+

梯度简单来说就是求导，在图像上表现出来的就是提取图像的边缘（不管是横向的、纵向的、斜方向的... [博文](#) 来自: [我爱智能](#)

thymeleaf模板实现html5标签的非严格检查 阅读数 3万+

一、概述最近在springboot项目引入thymeleaf模板时，使用非严格标签时，运行会报错。默认thymel... [博文](#) 来自: [Luck_ZZ的博客](#)

搭建图片服务器《二》-linux安装nginx 阅读数 3万+

nginx是个好东西，Nginx (engine x) 是一个高性能的HTTP和反向代理服务器，也是一个IMAP/POP3/... [博文](#) 来自: [maoyuanmin...](#)

frp配置本地服务端口到服务器80端口 阅读数 1万+

搭建环境： ubuntu 16.04 LTS （本地服务计算机） ubuntu 14.04 LTS(阿里云服务器) apache tomca... [博文](#) 来自: [Anteoy的博客](#)

OPENVPN拨入后给不同的用户分配不同的访问权限 阅读数 5481

需求： 我们想要给不同部门的或者级别的用户单独设置访问策略，比如 系统管理员：允许访问公司内... [博文](#) 来自: [linuxkai](#)

kingofark关于学习C++和编程的另外35个观点 阅读数 1420

作者：kingofarkRevision 1.01.编写小程序，请画程序流程框图；编写大程序，请画分段程序流程框图... [博文](#) 来自: [方舟 K \[N G ...](#)

三菱FX系列PLC与PC通讯的实现之专有协议（计算机联接）的程序设计之一 阅读数 1万+

阅读内容为：FX系列微型可编程控制器用户手册（通讯篇）中计算机链接功能章节。采用本方法通信... [博文](#) 来自: [pengjc2001的...](#)

强连通分量及缩点tarjan算法解析 阅读数 56万+

强连通分量： 简言之 就是找环（每条边只走一次，两两可达）孤立的一个点也是一个连通分量 使用t... [博文](#) 来自: [九野的博客](#)

mybatis一级缓存(session cache)引发的问题 阅读数 2万+

mybatis一级缓存(session cache)引发的问题 [博文](#) 来自: [flysharkym的...](#)

魔兽争霸3冰封王座1.24e 多开联机补丁 信息发布与收集点 阅读数 2万+


畅所欲言！ [博文](#) 来自: [Smile_qiqi的专...](#)


精仿今日头条 阅读数 1万+


转载请注明出处：http://blog.csdn.net/Chay_Chun/article/details/75319452 由来篇 当初的梦想 ... [博文](#) 来自: [Chay_Chun的...](#)


keras实现常用深度学习模型LeNet, AlexNet, ZFNet, VGGNet, GoogleNet, ... 阅读数 8270


LeNet #coding=utf-8 from keras.models import Sequential from keras.layers import Dense,Flat... [博文](#) 来自: [wmy199216的...](#)


 10


 50











04-08
下载

<div>ODAC (odp.net) 从开发到部署</div> <div>test</div>	阅读数 1万+	博文	来自: 我想我是海 冬...	<div><div>10</div><div>50</div><div></div><div></div><div><</div><div>></div></div>
<div>Consumer 接口 、 Predicate 接口初使用</div> <div>Consumer 接口 源码 package java.util.function; import java.util.Objects; @FunctionalInterface p...</div>	阅读数 312	博文	来自: kaven	
<div>SNMP协议详解<二></div> <div>上一篇文章讲解了SNMP的基本架构，本篇文章将重点分析SNMP报文，并对不同版本（SNMPv1、v2...</div>	阅读数 12万+	博文	来自: 假装在纽约	
<div>Java设计模式18——状态模式</div> <div>一、定义状态(State)模式又称为状态对象模式(Pattern of Objects for State),状态模式是对象的行为模...</div>	阅读数 1万+	博文	来自: 小小本科生成...	
<div>jquery/js实现一个网页同时调用多个倒计时(最新的)</div> <div>jquery/js实现一个网页同时调用多个倒计时(最新的) 最近需要网页添加多个倒计时. 查阅网络,基本上都...</div>	阅读数 43万+	博文	来自: Websites	
<div>机器学习(周志华西瓜书) 参考答案 总目录</div> <div>机器学习(周志华西瓜书)参考答案总目录 从刚开始学习机器学习到现在也有几个月了，期间看过PDF， ...</div>	阅读数 11万+	博文	来自: 我的博客	
<div>Spark2学习1之基本环境搭建（win）问题</div> <div>更多代码请见：https://github.com/xubo245/SparkLearning 版本：Spark-2.0.01解释 从【2】中下...</div>	阅读数 5万+	博文	来自: Keep Learning	
<div>linux上安装Docker(非常简单的安装方法)</div> <div>最近比较有空，大四出来实习几个月了，作为实习狗的我，被叫去研究Docker了，汗汗！ Docker的三...</div>	阅读数 19万+	博文	来自: 我走小路的博客	
<div>免费的天气接口（满足你的大部分需求）</div> <div>项目中有需要抓取当地天气的需求，在网上找了很多的接口,要么是接口请求次数有限制,要么是数据不...</div>	阅读数 3987	博文	来自: wanghao9401...	
<div>配置简单功能强大的excel工具类搞定excel导入导出工具类(一)</div> <div>对于J2EE项目导入导出Excel是最普通和实用功能,本工具类使用步骤简单,功能强大,只需要对实体类进行...</div>	阅读数 4万+	博文	来自: 李坤 大米时代 ...	
<div>人脸检测工具face_recognition的安装与应用</div> <div>人脸检测工具face_recognition的安装与应用</div>	阅读数 4万+	博文	来自: roguesir的博客	
<div>Docx4j 简单操作文字图片（包含页眉页脚和主体内容）</div> <div>docx4j官方提供了一些例子，本文只是其中一部分应用的简单例子。需要注意的地方是页眉和页脚，必...</div>	阅读数 9588	博文	来自: 偶尔记一下	
<div>06世界杯结果预测</div> <div>16强A 德国 波兰B 英格兰 巴拉圭C 阿根廷 荷兰D 墨西哥 葡萄牙E 意大利 捷克F 巴西 克罗地亚G 法国 ...</div>	阅读数 984	博文	来自: 我的测试时代	
<div>编写C语言版本的卷积神经网络CNN之一：前言与Minst数据集</div> <div>卷积神经网络是深度学习的基础，但是学习CNN却不是那么简单，虽然网络上关于CNN的相关代码很...</div>	阅读数 2万+	博文	来自: tostq的专栏	
<div>关于SpringBoot bean无法注入的问题（与文件包位置有关）</div> <div>问题场景描述整个项目通过Maven构建，大致结构如下： 核心Spring框架一个module spring-boot-b...</div>	阅读数 16万+	博文	来自: 开发随笔	
<div>DirectX修复工具增强版</div> <div>最后更新：2018-12-20 DirectX修复工具最新版：DirectX Repair V3.8 增强版 NEW! 版本号：V3.8.0...</div>	阅读数 191万+	博文	来自: VBcom的专栏	
<div>caffe下fcnn数据集制作的简化</div> <div>前面一直跑caffe下的fcnn，同时上一篇博客，也说明了关于fcnn的数据集的制作。但是过程还是显的很复...</div>	阅读数 7278	博文	来自: supe_king的博...	
<div>ubuntu在系统启动logo过后无法进入桌面的处理方法</div> <div>1) ubuntu在系统启动logo过后无法进入桌面的处理方法 一般情况下，无法显示桌面，然后/var/log/...</div>	阅读数 9964	博文	来自: 四维空间	
NLP学习学习 Linux进程生命周期控制 深度学习学习 软件设计学习 Ngnix学习				
ios获取idfa server的安全控制模型是什么 sql android title搜索 ios 动态修改约束 java控制台五子棋学习 学习java的纵向拓展				



follow轻尘

关注

原创9

粉丝353

喜欢59

评论205

等级：

博客 3

访问：3万+

积分：502

排名：13万+

最新文章

Git fatal: unable to auto-detect email address

电动车结构及其工作原理

Apollo代码学习(七)—MPC与LQR比较

Apollo代码学习(六)—模型预测控制(MPC)

Apollo代码学习(四)—Windows下编译

Apollo并与Carsim和Simulink联调

个人分类

Apollo7篇

Control4篇

Markdown1篇

电动汽车1篇

Git1篇

归档

2019年3月1篇

2018年12月1篇

2018年11月4篇

2018年10月3篇

2018年9月1篇

热门文章

Apollo代码学习(六)—模型预测控制(MPC)
阅读数 7013

Apollo代码学习(一)—控制模块概述
阅读数 5705

Apollo代码学习(三)—车辆动力学模型
阅读数 5611

Apollo代码学习(二)—车辆运动学模型
阅读数 5300

Apollo代码学习(五)—横纵向控制
阅读数 4718


最新评论

Apollo代码学习(二)—车辆运...
u013914471: 1.16只是一个简单的例子，比实际的模型要简单，反馈控制那些主要是基于动力...


Apollo代码学习(二)—车辆运...
weixin_39927213: 楼主您好，这篇博客最后的公式1.16和上篇博客的公式反馈控制器的公式什...

Apollo代码学习(五)—横纵向...
weixin_39927213: [reply]weixin_42939036[/reply] 您好，看了很多遍没看出来您说的问题呢： ...


Apollo代码学习(六)—模型预...





10





50











weixin_40215443: 博主您好, 公式13由于C的存在, 是不是就应该不是线性模型了, 困惑中! ...

Apollo代码学习(一)—控制模...

u013914471: [reply]a08003104[/reply] 输入输出我是根据代码来的, 您再看一下代码对照一~ ...



CSDN学院



CSDN企业招聘

 QQ客服

 kefu@csdn.net

 客服论坛

 400-660-0108

工作时间 8:30-22:00

关于我们

招聘

广告服务

网站地图

 百度提供站内搜索 京ICP证19004658号

©1999-2019 北京创新乐知网络技术有限公司


公司


网络110报警服务

经营性网站备案信息

北京互联网违法和不良信息举报中心

中国互联网举报中心

 10

 50

