深蓝学院
shenlanxueyuan.com

Magic Lab

基于模型的预测方法

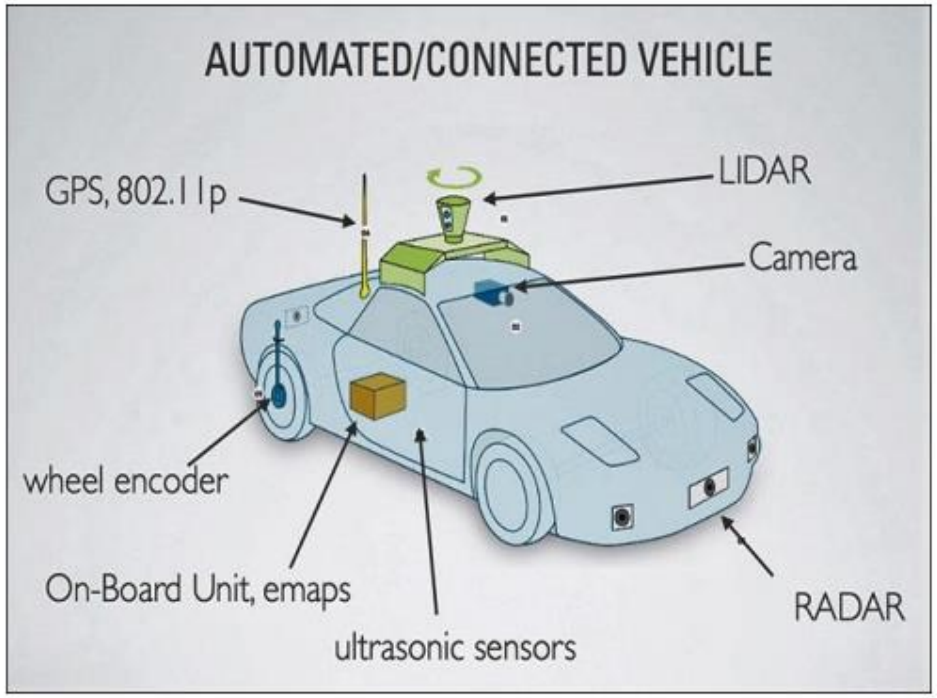
丁文超

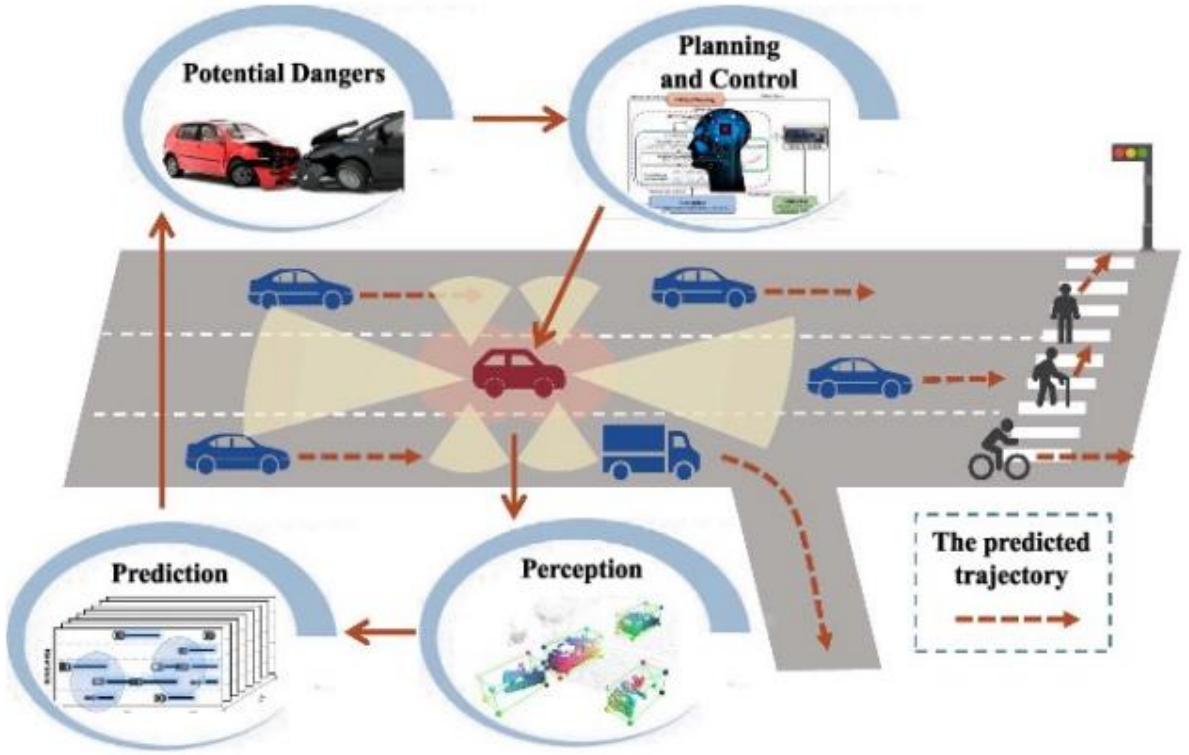
大纲

- 预测系统概述
- 定速度预测
- 定曲率预测
- 基于手工特征的意图预测
- 基于模型的轨迹预测

预测系统概述

自动驾驶系统架构

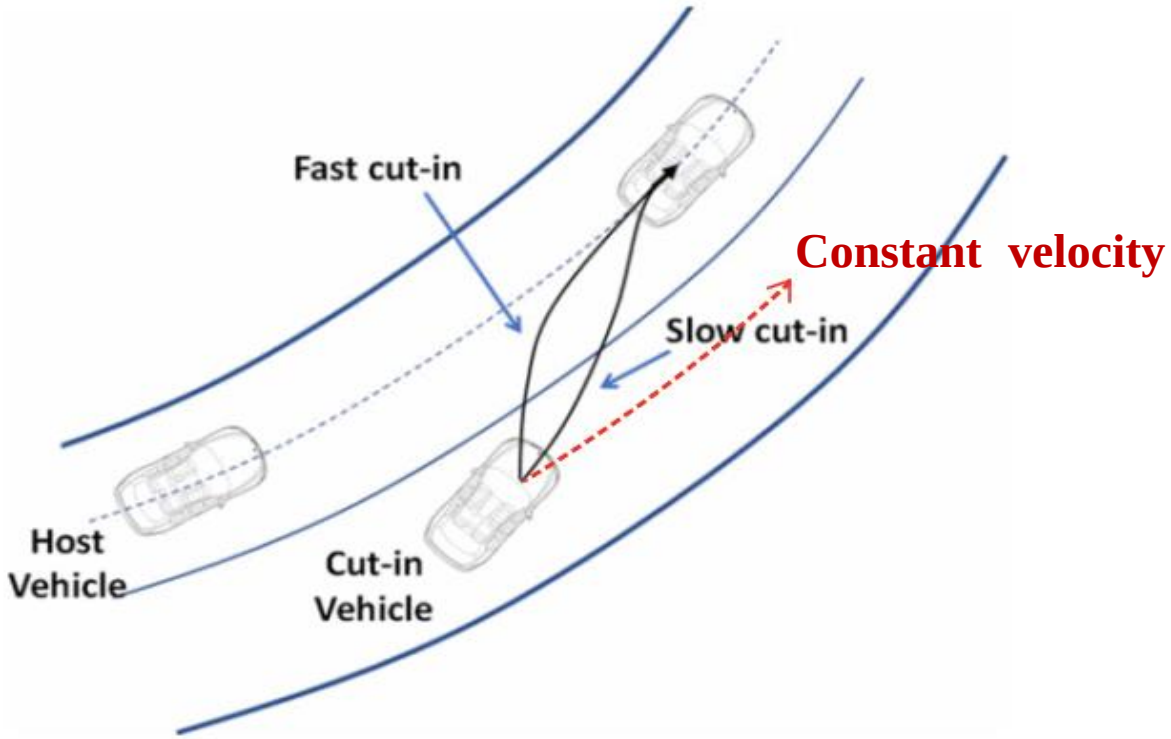




感知 >> **预测** >> 规控

预测系统概述

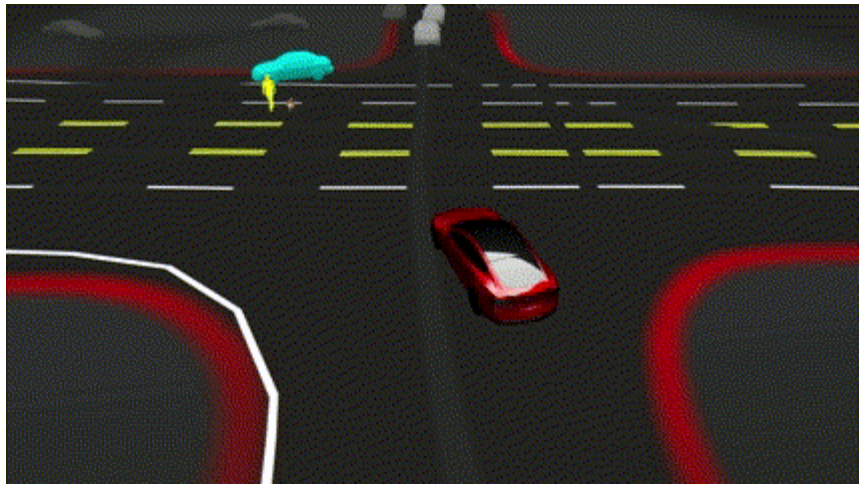
预测系统的必要性



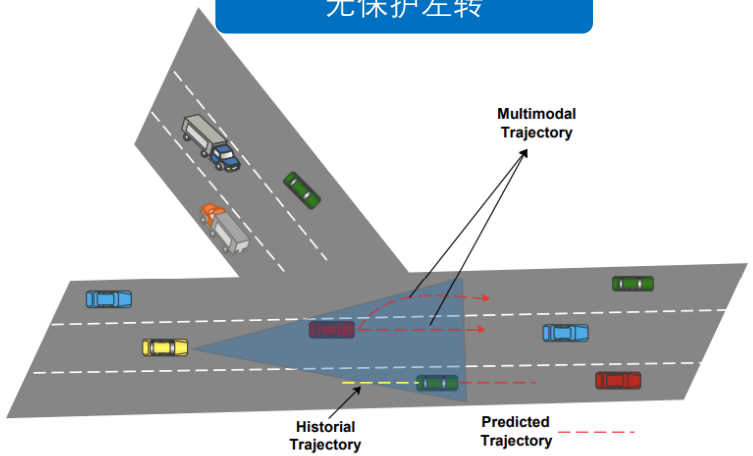
- 预测结果的召回率?
- 预测结果的准确率?
- 对决策规划的影响?

预测系统概述

预测系统的必要性



无保护左转



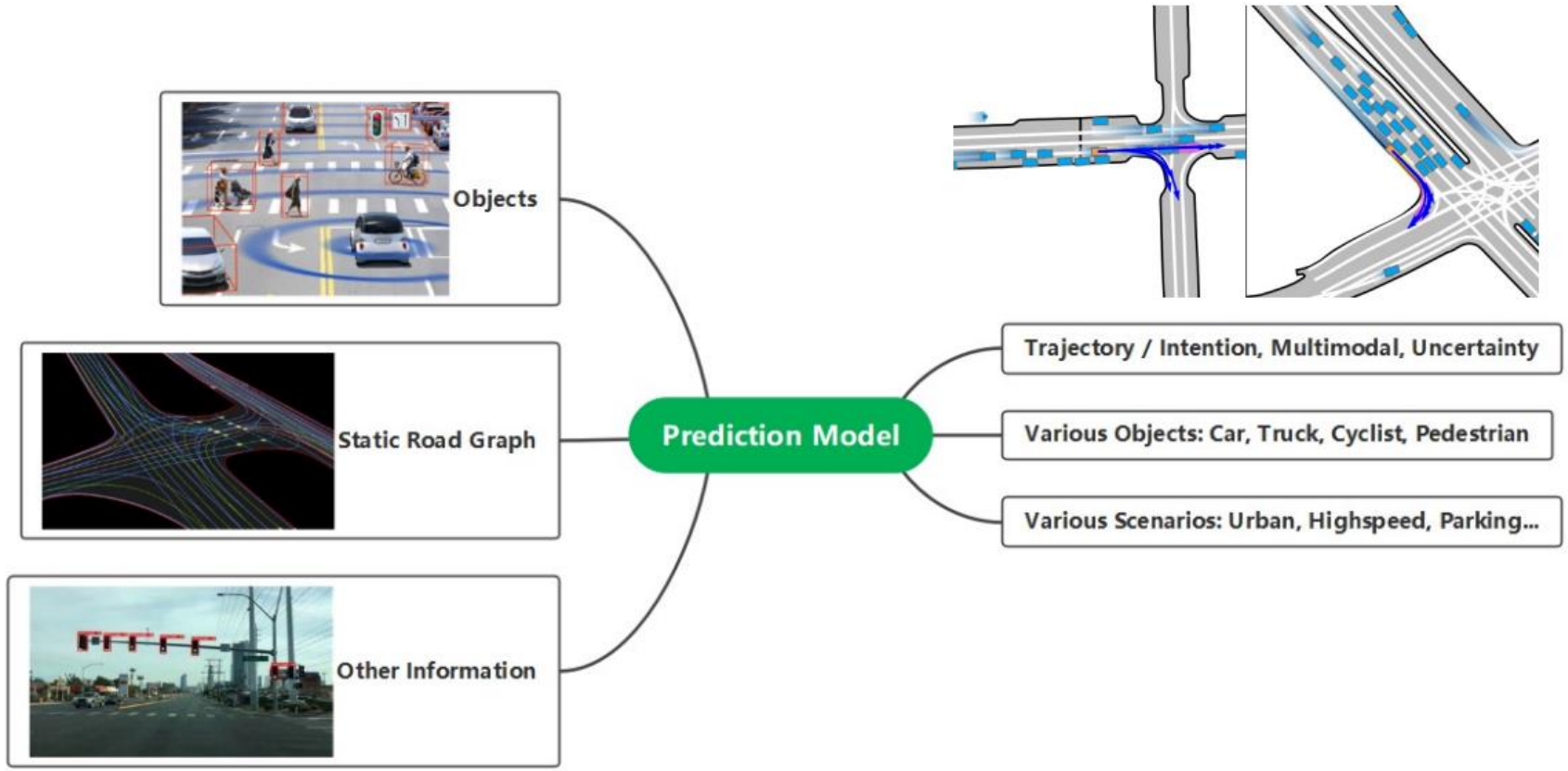
复杂路况

- 预测结果的多模态性
- 预测结果的不确定性

[1] Tesla AI Day 2022

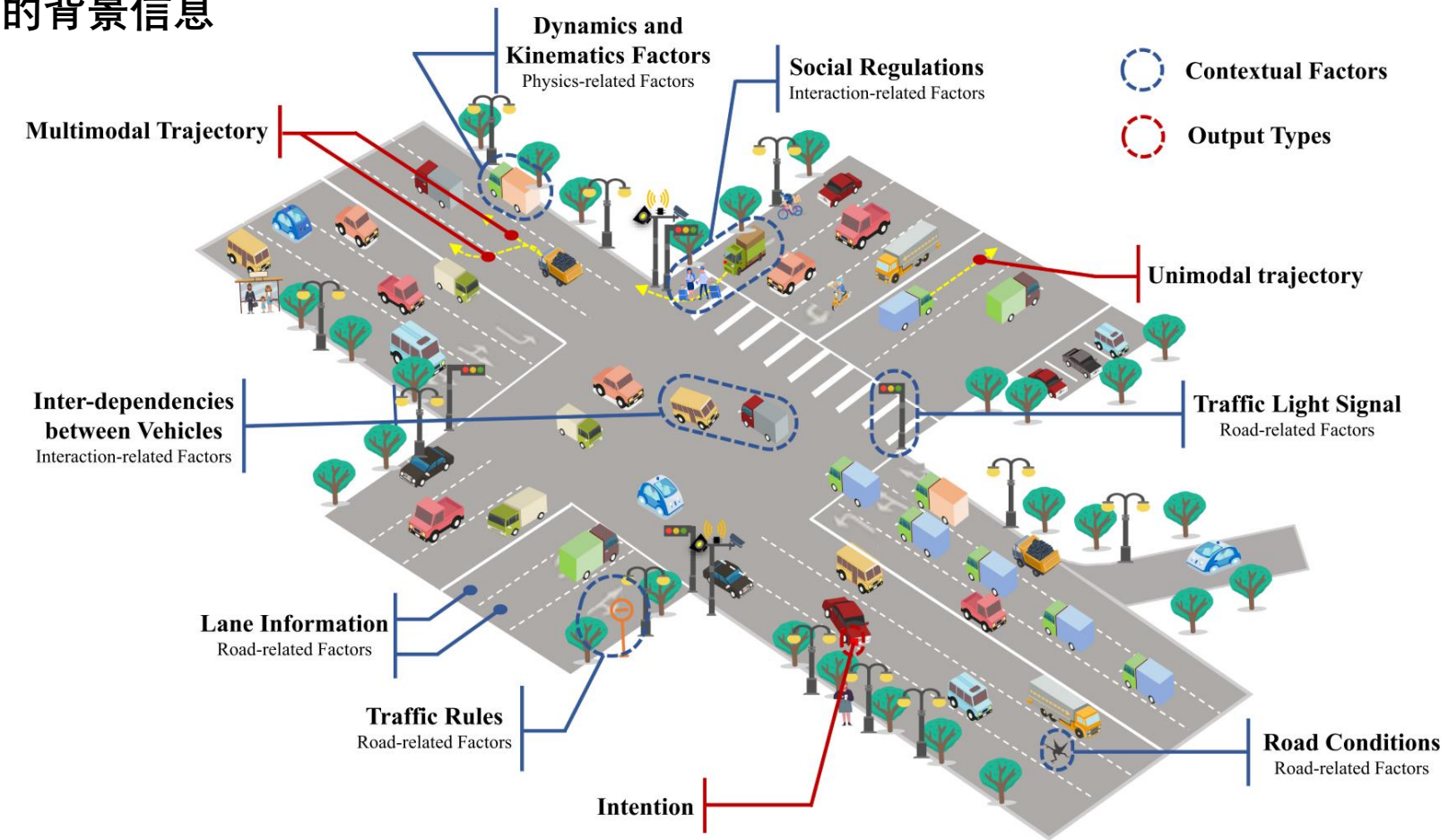
预测系统概述

预测系统的架构



预测系统概述

预测系统的背景信息



轨迹预测的输入输出

预测系统概述

- **主要应用场景：**城镇、高速公路

预测系统发展

- **简单运动模型的预测：定速度预测、定曲率预测**

  计算成本低　　不适合复杂场景，长时预测失准

- **结合场景先验知识的预测：基于手工特征的意图预测**

  融合意图级别信息　　缺少轨迹级别信息

- **意图预测（长时+先验） + 短时预测（趋势） = 长时轨迹预测**

定速度预测

一维匀速运动模型

目标做匀速直线运动，加速度为0。

现实中速度会有轻微扰动变化，可视为具有高斯分布的噪声。

$$\ddot{X}(t) = W(t)$$

一维状态向量：

$$X = [x, \dot{x}]^T$$

定速度预测

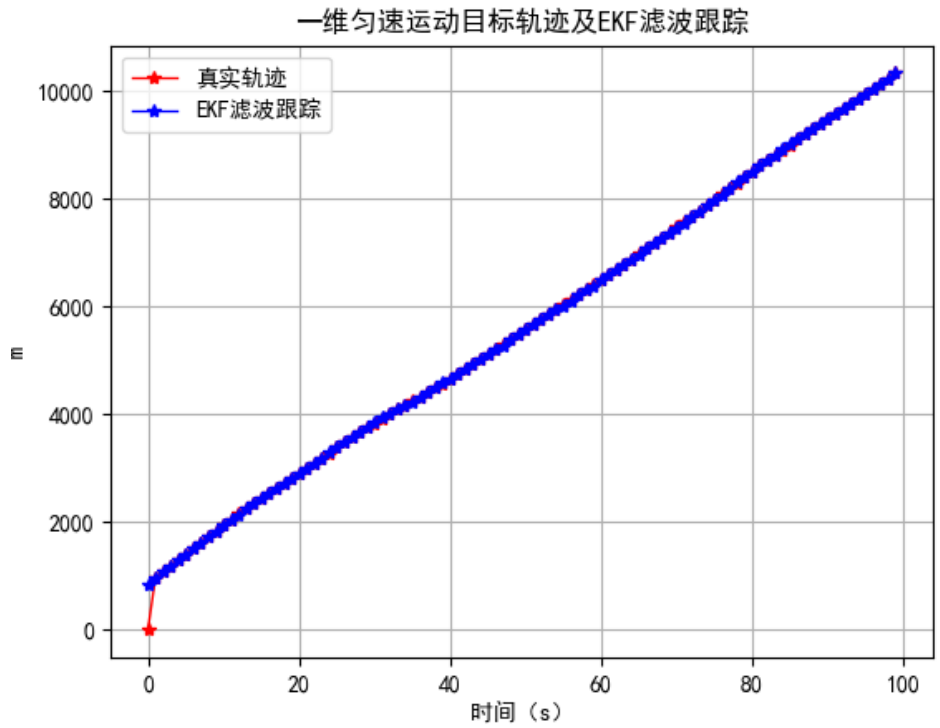
一维连续Constant Velocity模型：

$$\dot{X}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} X(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} W(t)$$

常用离散形式：

$$X_{k+1} = \begin{bmatrix} 1 & \mathrm{T} \\ 0 & 1 \end{bmatrix} X_k + \begin{bmatrix} \mathrm{T}^2/2 \\ \mathrm{T} \end{bmatrix} W_k$$

T：采样时间



定速度预测

二维匀速运动模型

二维状态向量：

$$X = [x, \dot{x}, \quad y, \dot{y}]^{\mathrm{T}}$$

二维连续Constant Velocity模型：

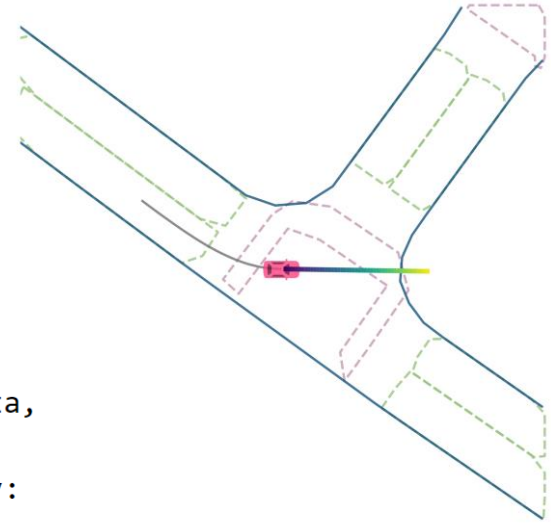
$$\dot{X}(t) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} X(t) + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \mathrm{W}(t)$$

定速度预测

二维离散Constant Velocity模型：

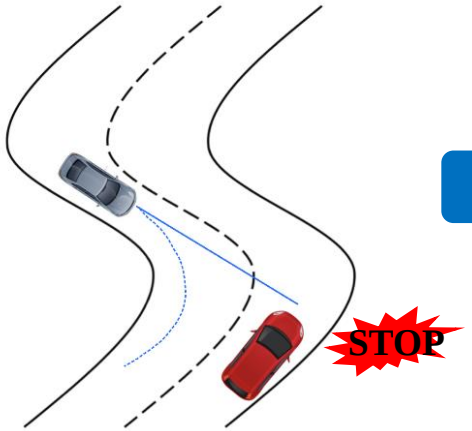
$$X_{k+1} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} X_k + \begin{bmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{bmatrix} W_k$$

T：采样时间



CV预测

```
def _constant_velocity_heading_from_kinematics(kinematics_data: KinematicsData,
                                               sec_from_now: float,
                                               sampled_at: int) -> np.ndarray:
    """
    Computes a constant velocity baseline for given kinematics data, time window
    and frequency.
    :param kinematics_data: KinematicsData for agent.
    :param sec_from_now: How many future seconds to use.
    :param sampled_at: Number of predictions to make per second.
    """
    x, y, vx, vy, _, _, _, _, _, _ = kinematics_data
    preds = []
    time_step = 1.0 / sampled_at
    for time in np.arange(time_step, sec_from_now + time_step, time_step):
        preds.append((x + time * vx, y + time * vy))
    return np.array(preds)
```



Bad Case

定曲率预测

匀速圆周运动状态向量（转弯角速度已知）：

$$X = [x, \dot{x}, \quad y, \dot{y}]^{\mathrm{T}} \qquad v = \sqrt{\dot{x}^2 + \dot{y}^2}$$

匀速圆周运动离散模型：转角变化率δ恒定

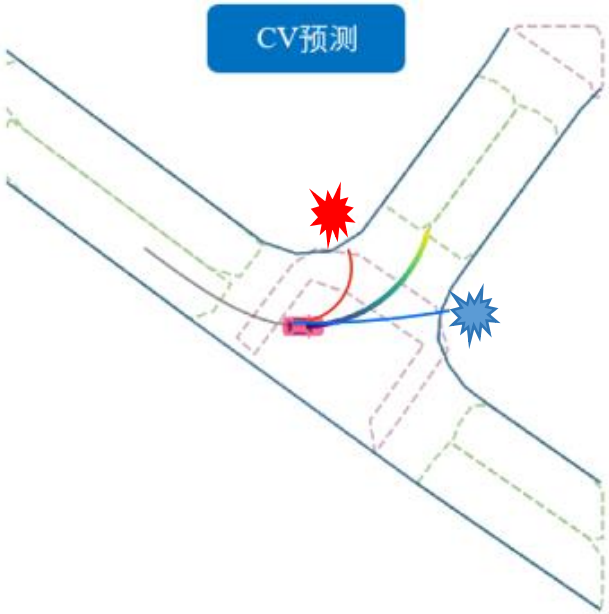
$$X_{k+1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} X_k + \begin{bmatrix} \mathrm{T} * v * \cos(\omega_{\mathrm{k}}) \\ 0 \\ \mathrm{T} * v * \sin(\omega_{\mathrm{k}}) \\ 0 \end{bmatrix} + \begin{bmatrix} \mathrm{T}^2/2 & 0 \\ \mathrm{T} & 0 \\ 0 & \mathrm{T}^2/2 \\ 0 & \mathrm{T} \end{bmatrix} W_k$$

$$\omega_{k+1} = \omega_k + \mathrm{T} * \delta$$

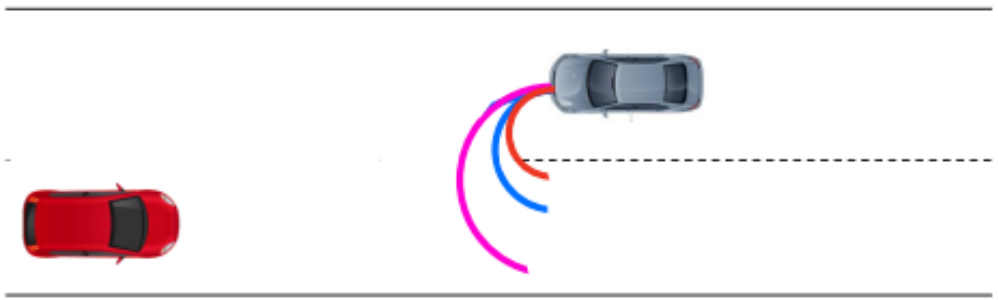
```
"""
Computes a baseline prediction for the given time window and frequency, under
the assumption that the (scalar) speed and yaw rate are constant.
:param kinematics_data: KinematicsData for agent.
:param sec_from_now: How many future seconds to use.
:param sampled_at: Number of predictions to make per second.
"""
```

```python
def _constant_speed_and_yaw_rate(kinematics_data: KinematicsData,
                                 sec_from_now: float, sampled_at: int) -> np.ndarray:
    x, y, vx, vy, _, _, speed, yaw_rate, _, yaw = kinematics_data

    preds = []
    time_step = 1.0 / sampled_at
    distance_step = time_step * speed
    yaw_step = time_step * yaw_rate
    for _ in np.arange(time_step, sec_from_now + time_step, time_step):
        x += distance_step * np.cos(yaw)
        y += distance_step * np.sin(yaw)
        preds.append((x, y))
        yaw += yaw_step
    return np.array(preds)
```

定曲率预测



转向curvature根据历史观测拟合噪声大，
而且容易不符合道路结构



Bad Case

定速度预测 VS 定曲率预测

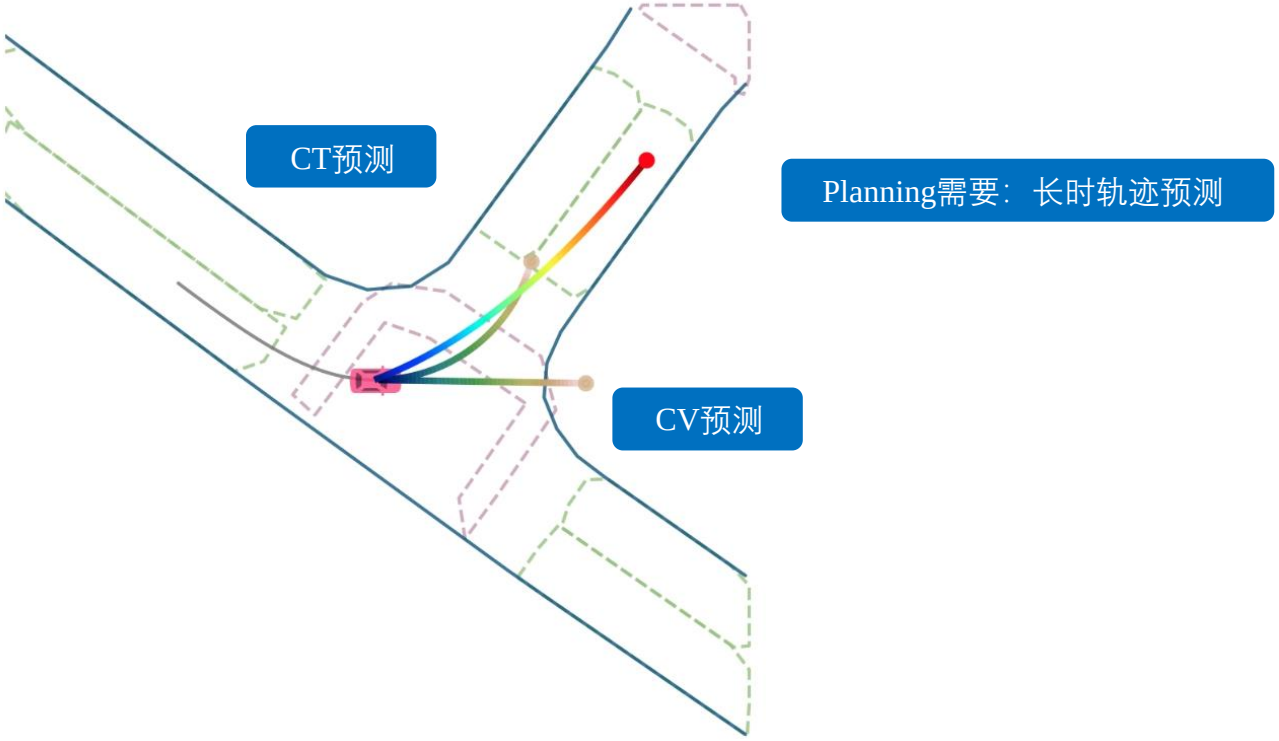
定速度预测	定曲率预测
适用于速度变化不大的场景	适用于曲线路径，汽车转弯的情况
预测精度不高	预测精度稍高
在速度或方向发生显著变化时失准	在直线运动或曲率变化大时失准

共同局限：长时间区间会失准！

短期预测 vs 长时预测

预测不确定性随着预测时间变长显著增大

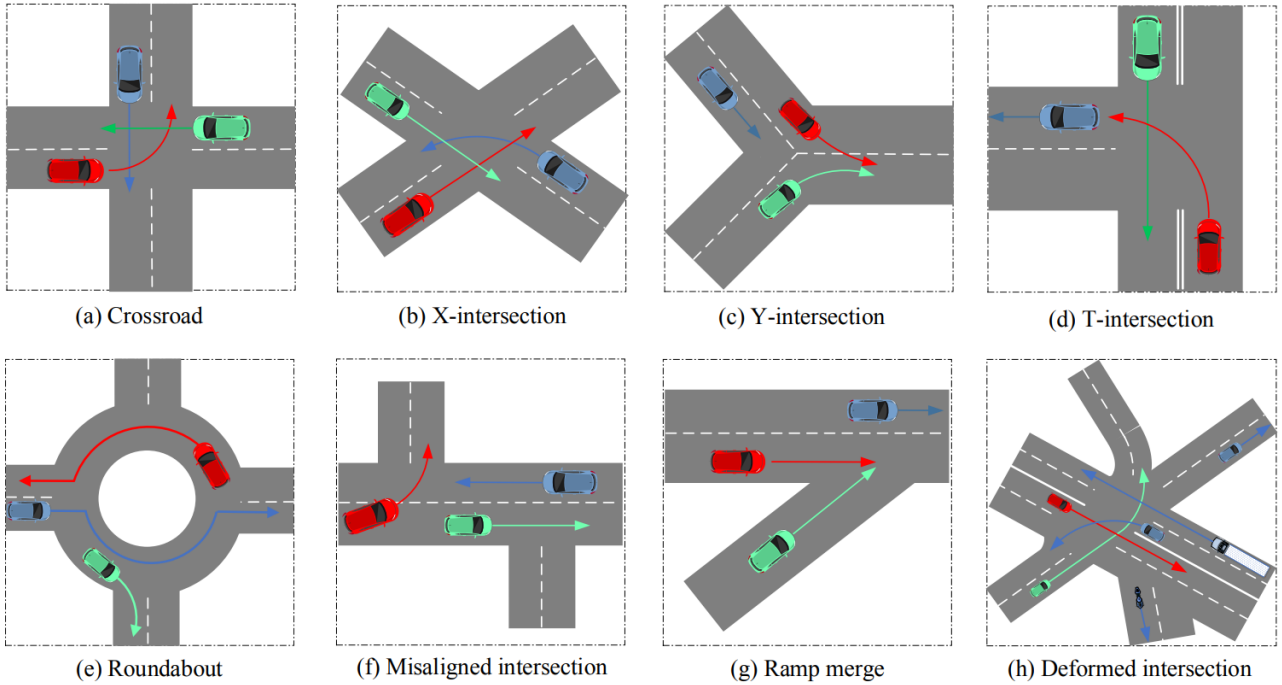
- **短期预测：**基于运动学模型或者预测网络，完成短时推演，一般为**3s**
- **长期预测：**结合**意图**预测，稳定长时预测，避免远端发散，一般为**8s+**



基于手工特征的意图预测

意图：预先定义的车流行为，如变道，左右转等

分类：基于车辆的行为特征，对车辆的意图进行分类



[1] Wei, Lianzhen, et al. "Autonomous driving strategies at intersections: Scenarios, state-of-the-art, and future outlooks." 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). IEEE, 2021.

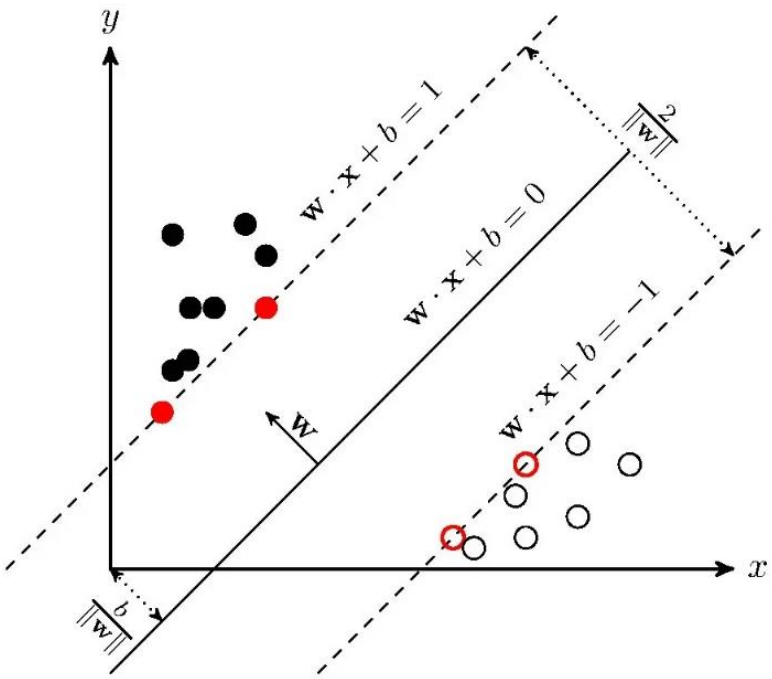
基于SVM的意图分类

支持向量机Support Vector Machine（SVM）

- SVM是用于分类问题的有监督学习算法，其基本思想是找到一个超平面 $\omega \cdot x + b = 0$ ，使不同类别的数据点之间的分离最大化。
- 支持向量是距离超平面最近的数据点，它们决定了超平面的位置：$y_i(\omega \cdot x_i + b) = 1$
- **需要解决优化问题：**

$$min_{\omega,b} \frac{1}{2} \|\omega\|^2,$$

$$\text{s.t. } y_i(\omega \cdot x_i + b) \geq 1, i = 1,2, \dots, N$$

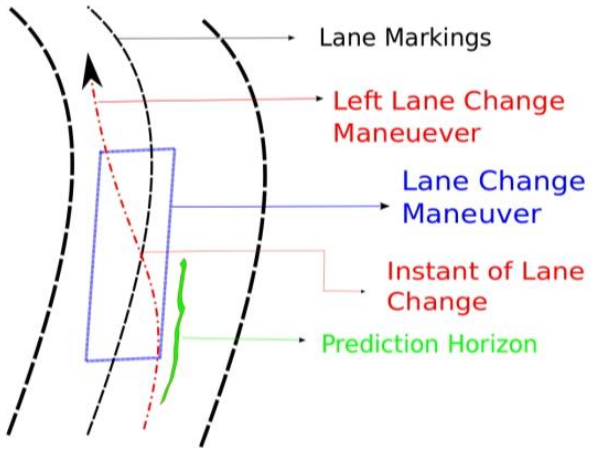


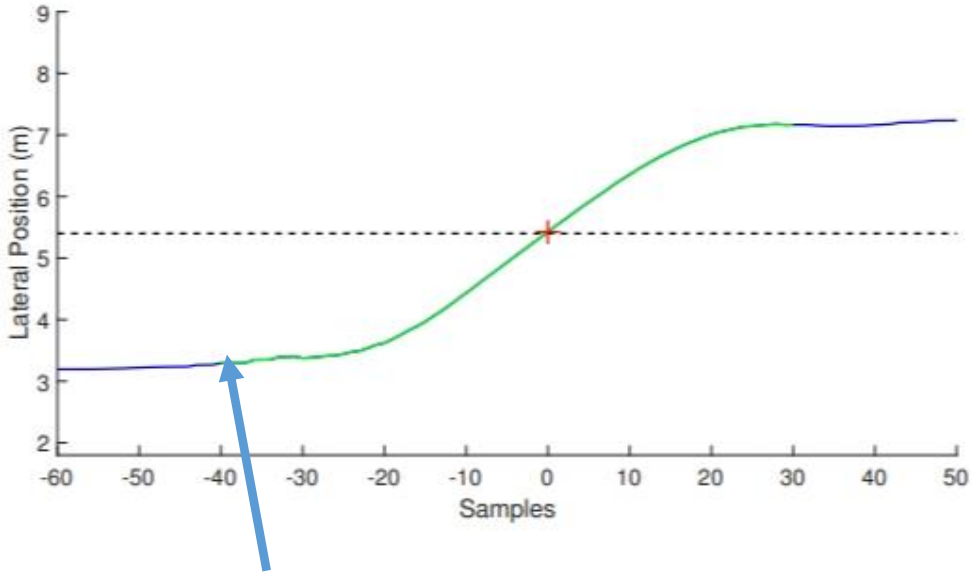
[1] Pisner, Derek A., and David M. Schnyer. "Support vector machine." Machine learning. Academic Press, 2020. 101-121.

基于SVM的意图分类

SVM特征选取

- 手工特征标记：
  - 距离目标车道的横向距离
  - 距离目标车道的横向速度
  - 道路实线、虚线….





在变道前一段时间打上分类标记

基于SVM的意图分类

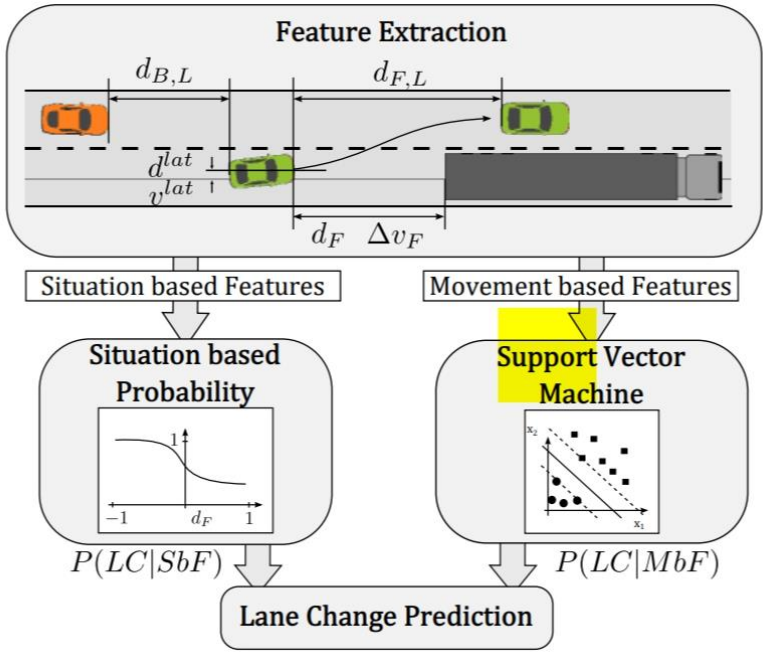
SVM输入输出案例

- **输入：** 历史lateral位置、转向角、速度。
- **输出：** 下一时刻的行为标签∈ {0,1},

  0表示不去该车道，1表示去该车道

- **损失函数：** 交叉熵损失函数

$$l(\theta) = \frac{1}{n}\sum_{i=1}^{n}(-\sum_{m=1}^{q} y^m \ln \hat{y}^m)$$



[1] Kumar, Puneet, et al. "Learning-based approach for online lane change intention prediction." 2013 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2013.

基于神经网络的意图预测

通过多层感知机预测障碍车会选择哪一条车道行驶，输出每个车道线的概率

- **输入：**每个车道线序列障碍物特征和车道线特征组成的62维特征向量。
- **输出：**每条车道线序列的概率
- **模型（MLP_EVALUATOR）：**4层全连接神经网络，每层之间用线性层进行连接

```
Linear(62,30)->Linear(30,15)->Relu()->Linear(15,5)->Relu()->Linear(5,1)->Sigmoid()
```

```
layer {
  layer_input_dim: 62
  layer_output_dim: 30
  layer_input_weight {}
  layer_bias {}
  layer_activation_func: RELU
}
```

```
layer {
  layer_input_dim: 30
  layer_output_dim: 15
  layer_input_weight {}
  layer_bias {}
  layer_activation_func: RELU
}
```

```
layer {
  layer_input_dim: 15
  layer_output_dim: 5
  layer_input_weight {}
  layer_bias {}
  layer_activation_func: RELU
}
```

```
layer {
  layer_input_dim: 5
  layer_output_dim: 1
  layer_input_weight {}
  layer_bias {}
  layer_activation_func: SIGMOID
}
dim_output: 1
```

- **COST_EVALUATOR：**计算障碍物对每个车道线序列的概率：主要计算车道半宽与车辆距离车道中心线的横向距离做差，然后传入sigmoid函数，如果小于0说明车在车道外侧，概率为0；大于0车在车道内，概率为1

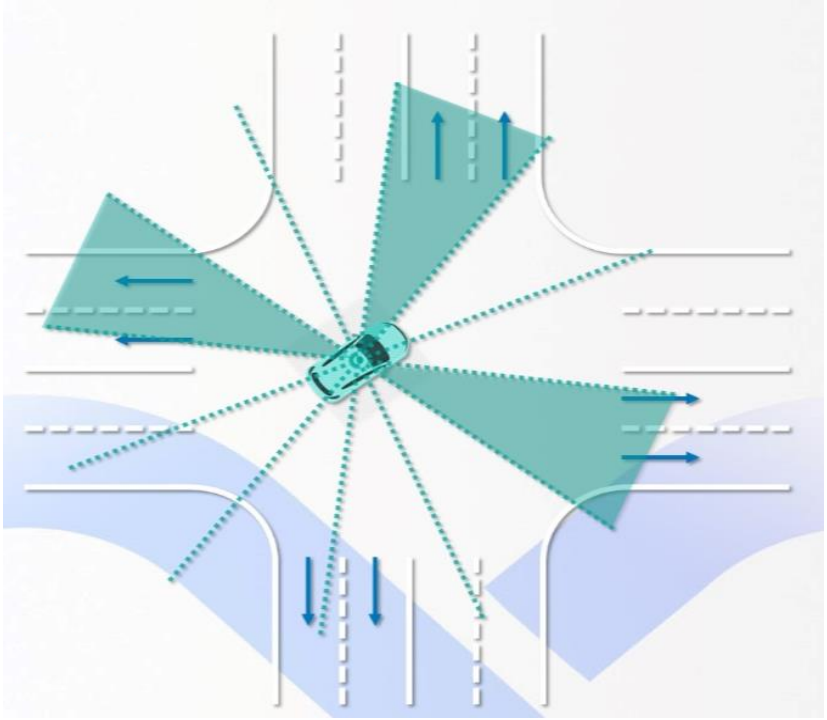
```
double CostEvaluator::ComputeProbability(const double obstacle_length,
                                         const double obstacle_width,
                                         const LaneSequence& lane_sequence) {
  double front_lateral_distance_cost =
      FrontLateralDistanceCost(obstacle_length, obstacle_width, lane_sequence);
  return apollo::common::math::Sigmoid(front_lateral_distance_cost);
}
```

[1] Apollo 5.0

其他的输出建模方法

通过将路口进行扇区划分进行交通路口的变道意图预测

- **输入：**障碍物自身运动历史，路口车道信息，周围其他障碍物信息。
- **模型：**以障碍物车朝向为参考方向，划分为12个扇形区域；记录每个扇形区域内是否有离开该路口的车道；将问题转化为12元分类问题。
- **输出：**选择对应扇区的概率。



[1] Apollo 5.0

其他的输出建模方法

通过将路口进行扇区划分进行交通路口的变道意图预测

- 根据障碍车车辆heading将周围区域分为12个扇形，如果扇形内存在驶离路口，则将其mask置1
- 输出每一个扇形区域行驶的概率，然后对扇形区域内的所有lane segment赋上概率

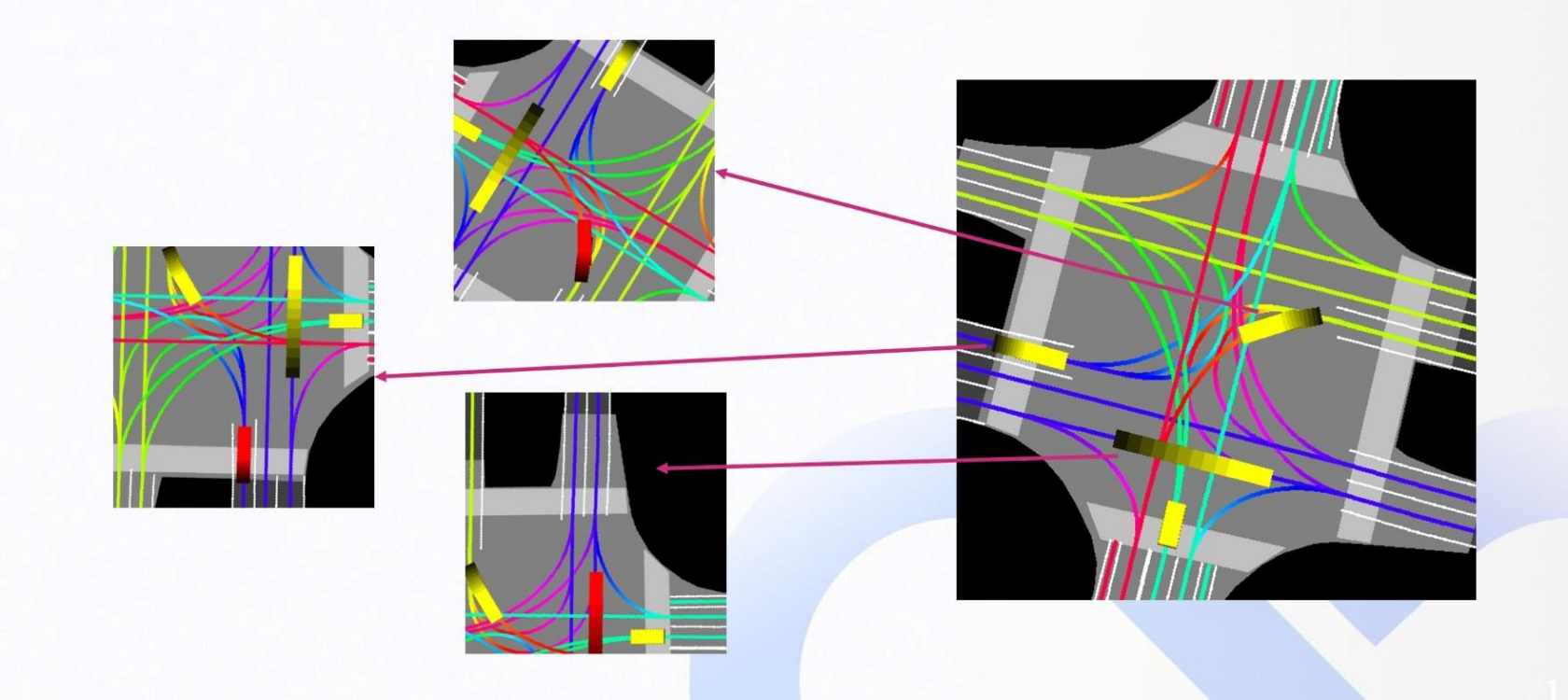
```
bool JunctionMapEvaluator::ExtractFeatureValues(Obstacle* obstacle_ptr, std::vector<double>* feature_values)
{ feature_values->clear();
  feature_values->resize(JUNCTION_FEATURE_SIZE, 0.0);
  Feature* feature_ptr = obstacle_ptr->mutable_latest_feature();
  if (!feature_ptr->has_position())
  {
    ADEBUG << "Obstacle [" << obstacle_ptr->id() << "] has no position.";
    return false;
  }
  double heading = std::atan2(feature_ptr->raw_velocity().y(),
                              feature_ptr->raw_velocity().x());
  if (!feature_ptr->has_junction_feature()) {
    AERROR << "Obstacle [" << obstacle_ptr->id()
           << "] has no junction_feature.";
    return false;}

  int num_junction_exit = feature_ptr->junction_feature().junction_exit_size();
  for (int i = 0; i < num_junction_exit; ++i) {
    const JunctionExit& junction_exit =
        feature_ptr->junction_feature().junction_exit(i);
    double x = junction_exit.exit_position().x() - feature_ptr->position().x();
    double y = junction_exit.exit_position().y() - feature_ptr->position().y();
    double diff_x = std::cos(-heading) * x - std::sin(-heading) * y;
    double diff_y = std::sin(-heading) * x + std::cos(-heading) * y;
    double angle = std::atan2(diff_y, diff_x);
    double d_idx = (angle / (2.0 * M_PI) + 1.0 / 24.0) * 12.0;
    int idx = static_cast<int>(floor(d_idx >= 0 ? d_idx : d_idx + 12));
    feature_values->at(idx) = 1.0;
  }
  return true;
}
```

[1] Apollo 5.0

其他的输入建模方法

人工构造的输入特征有天然局限性

- **输入：**语义地图渲染的方式，将障碍车的历史运动状态、车道的形状与连接关系，以及其他车的运动状态和历史，都转化为图像信息。



[1] Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks, Cui et al.

其他的分类模型

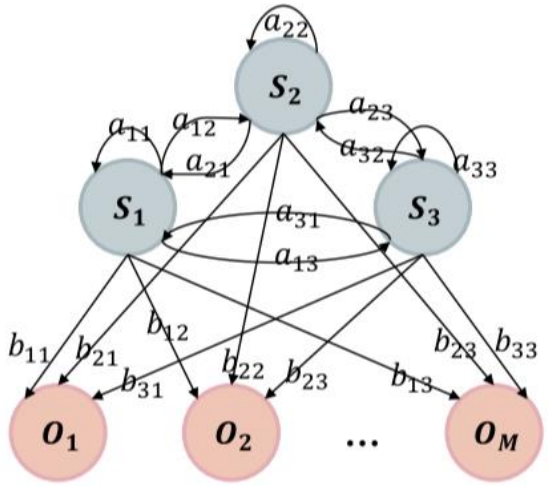
基于隐马尔可夫模型的意图估计

- 隐马尔可夫模型通过计算给定序列数据的概率分布解决轨迹等时间序列预测问题，可以表示成(S,O,A,B,π)。
- 交通参与者的历史状态表示为观测序列O

a. 在训练阶段，研究和挖掘历史轨迹信息构建隐马尔可夫模型；

b. 在预测阶段，基于训练模型，输入查询车辆的驱动轨迹，采用Viterbi等算法确定可能的隐藏序列来对应已知的观测序列；

c. 最后，根据隐马尔可夫模型的参数计算概率转移矩阵，分析和预测运动轨迹。



[1] Mathew, Wesley, Ruben Raposo, and Bruno Martins. "Predicting future locations with hidden Markov models." Proceedings of the 2012 ACM conference on ubiquitous computing. 2012.

[2] Ye, Ning, et al. "Vehicle trajectory prediction based on Hidden Markov Model." (2016).

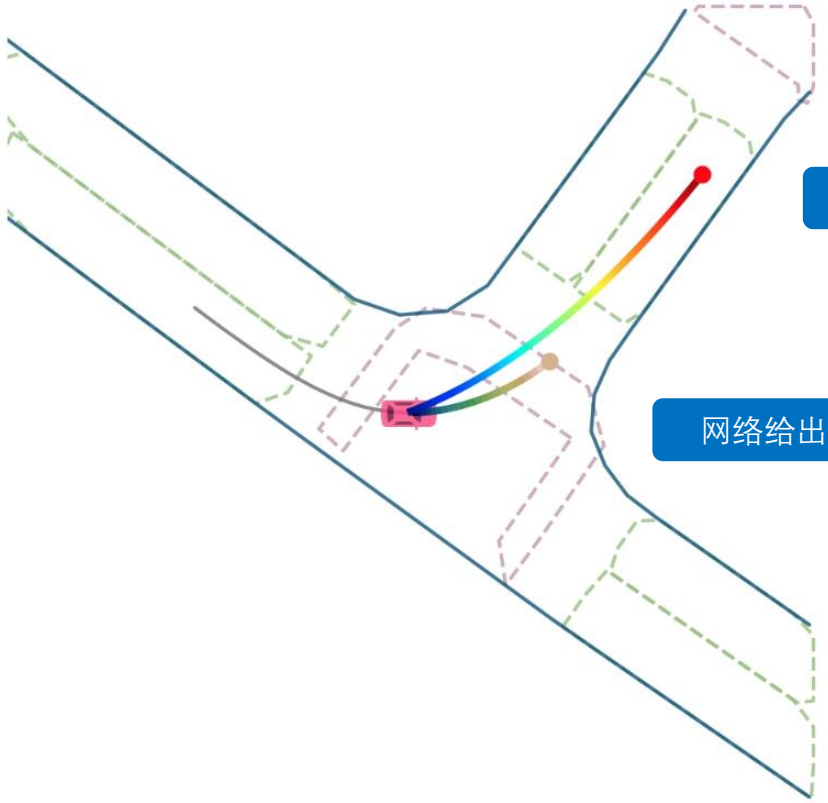
[3] Huang, Yanjun, et al. "A survey on trajectory-prediction methods for autonomous driving." IEEE Transactions on Intelligent Vehicles 7.3 (2022): 652-674.

如何将短时轨迹结合长时意图?

长时轨迹生成 -> 轻量化的planner

核心思想：

- ✓ 基于意图估计，结合地图拿到参考
- ✓ 基于短时预测，确定局部运动趋势
- ✓ 结合意图估计和短时预测，补全长时轨迹



Planning需要：长时轨迹预测

网络给出：短时轨迹预测

轨迹生成方法

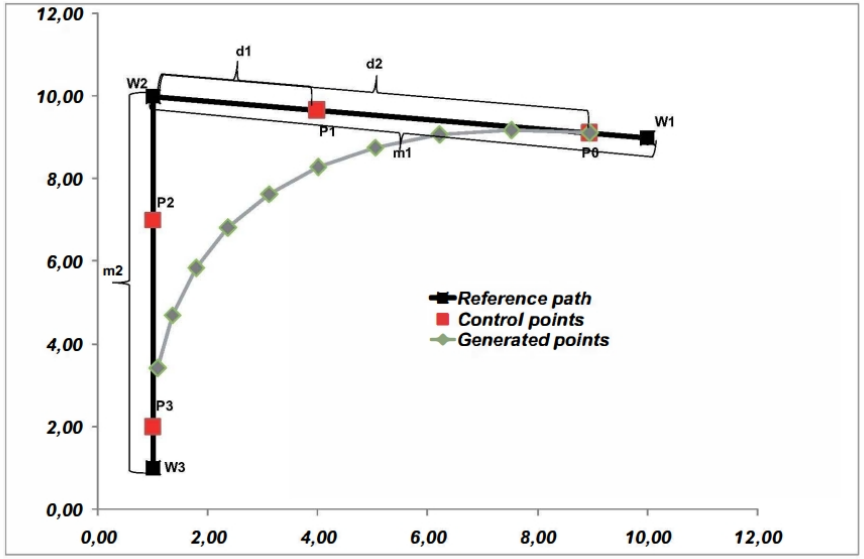
曲线类型	优点	缺点
贝塞尔曲线	可以通过少量的控制点来描述曲线， 而且计算简单	难以修改， 一个控制点的变化会影响整条曲线
B样条曲线	具有局部性和一定程度的平滑性， 可以自定义阶次， 移动控制点仅仅改变曲线的部分形状	在复杂情况下计算难度大， 对于拥有很多控制点的复杂曲线， B样条曲线需要更高阶的多项式， 合并两条曲线也比较困难
卡尔曼滤波	可以处理非线性系统， 同时可以处理多维度的数据	需要对系统建立数学模型， 而且对于非线性系统， 需要使用扩展卡尔曼滤波等变种算法

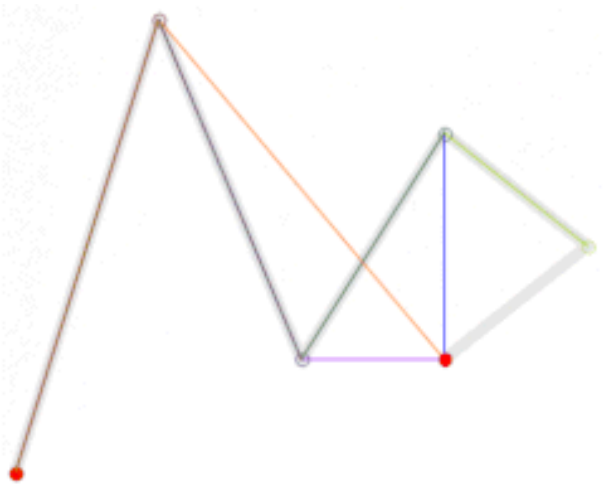
Bezier曲线生成长时轨迹

意图预测判断出车道后，可以灵活基于地图抽取控制点。

Bezier曲线基本原理

- n阶Bezier曲线可以表示成$B_{(t)} = \sum_{i=0}^{n}\binom{n}{i}(i-t)^{n-i}t * P_j$
- 给定参考路径W_1, W_2, W_3和控制距离，可以计算出平滑曲线的生成点。





五次Bezier曲线

[1] Pérez, Joshué, et al. "Trajectory generator for autonomous vehicles in urban environments." 2013 IEEE international conference on robotics and automation. IEEE, 2013.

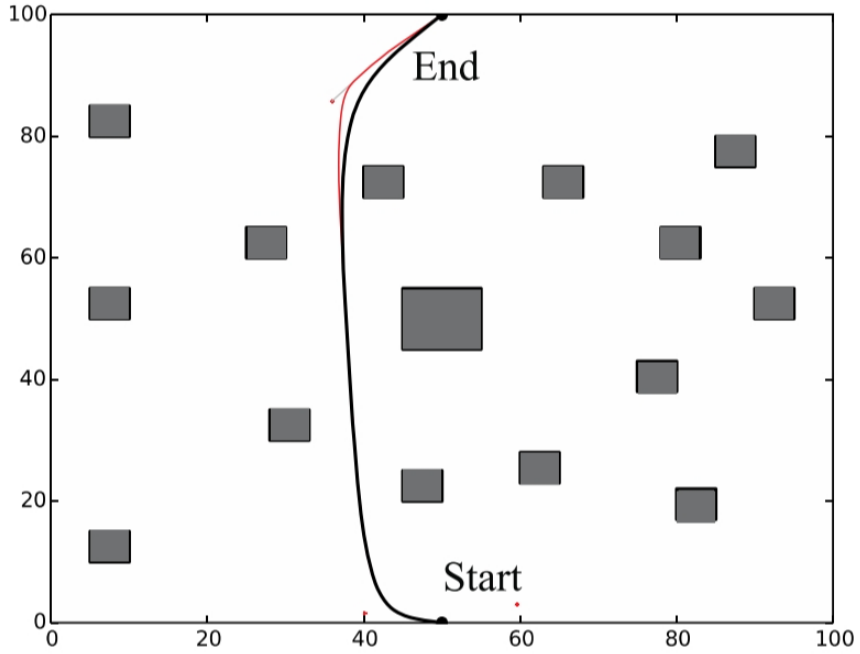
Bezier曲线生成长时轨迹

Bezier曲线数学方法

- 结合状态信息位置、航向角、速度、加速度等，作为待拟合路径曲线的初始和约束条件。
- 利用曲线系数，根据时间间隔对曲线进行采样，得到的采样点即为拟合后的路径点。
- p次B样条曲线$c(u) = \sum_{i=1}^{n} N_{i,p}(u)\, P_i$

$$N_{i,0}(u) = \begin{cases} 1 & u \in [\hat{u}_i, \hat{u}_{i+1}) \\ 0 & else \end{cases}$$

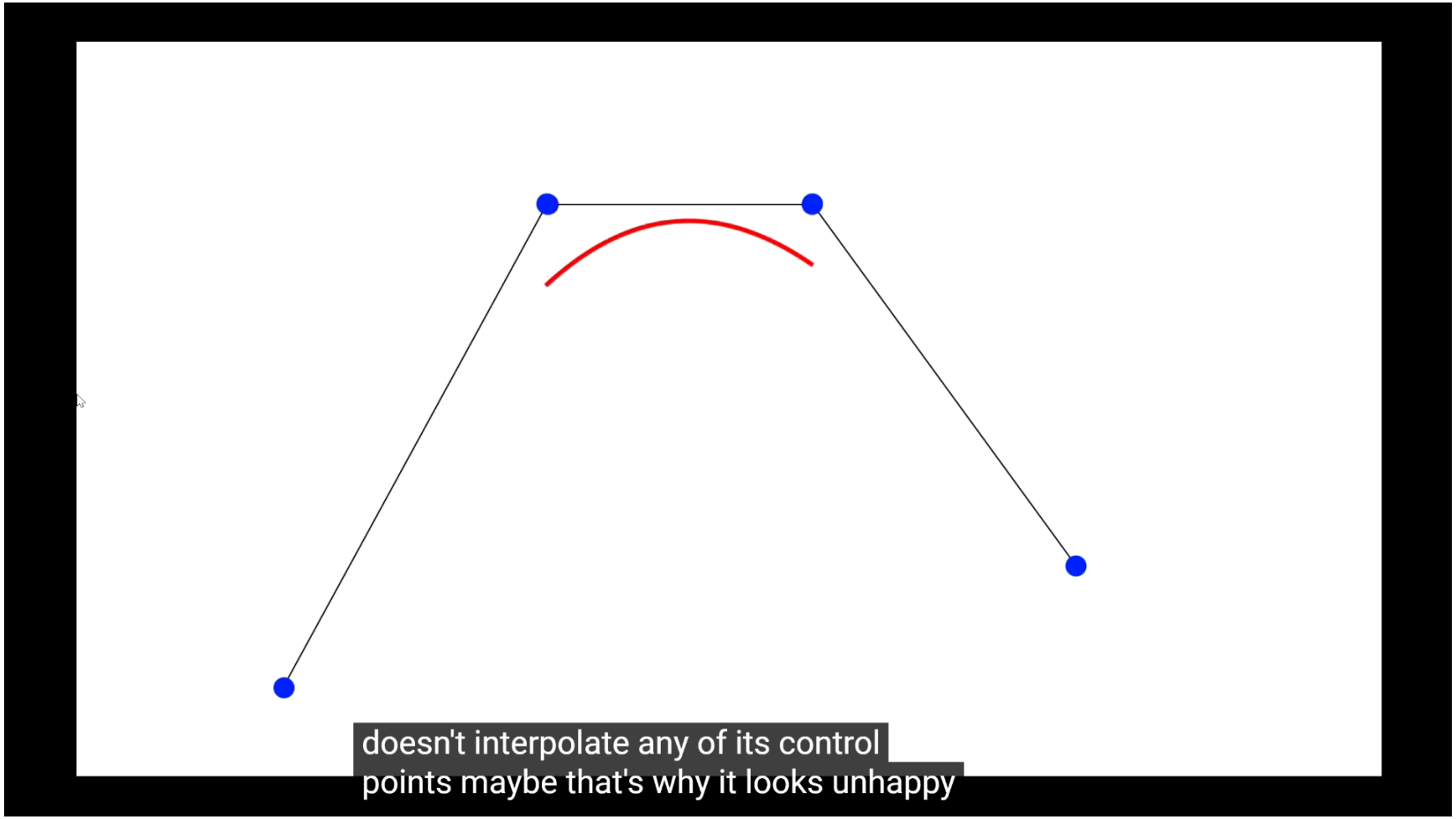
$$N_{i,p}(u) = \frac{u - \hat{u}_i}{\hat{u}_{i+p} - \hat{u}_i} N_{i,p-1}(u) + \frac{\hat{u}_{i+p+1} - u}{\hat{u}_{i+p+1} - \hat{u}_{i+1}} N_{i+1,p-1}(u)$$



[1] Elbanhawi, Mohamed, Milan Simic, and Reza Jazar. "Randomized bidirectional B-spline parameterization motion planning." IEEE Transactions on intelligent transportation systems 17.2 (2015): 406-419.

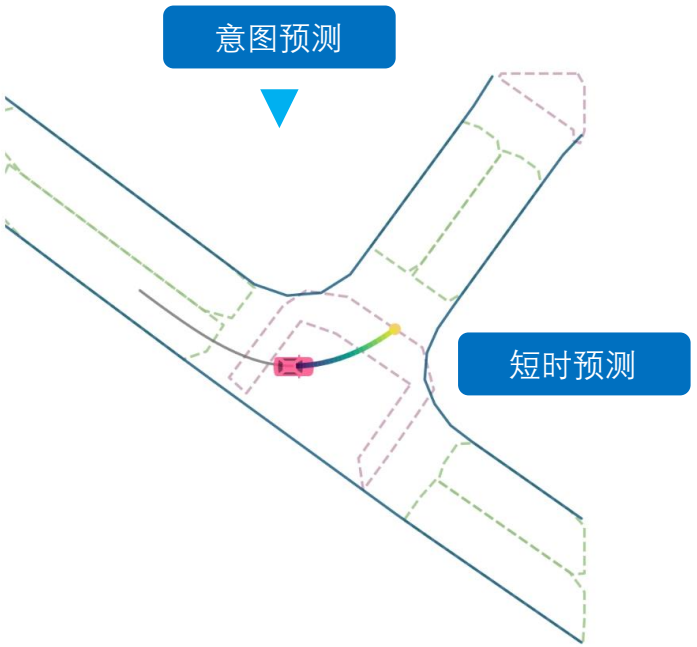
Bezier曲线生成长时轨迹

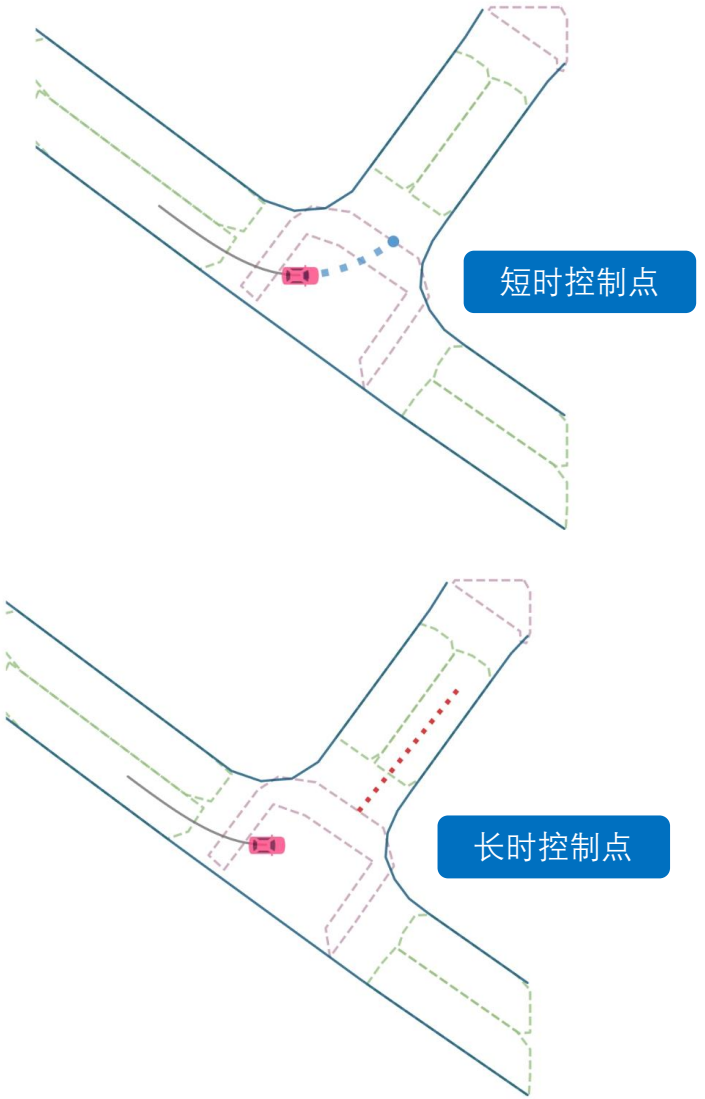
Bezier曲线案例

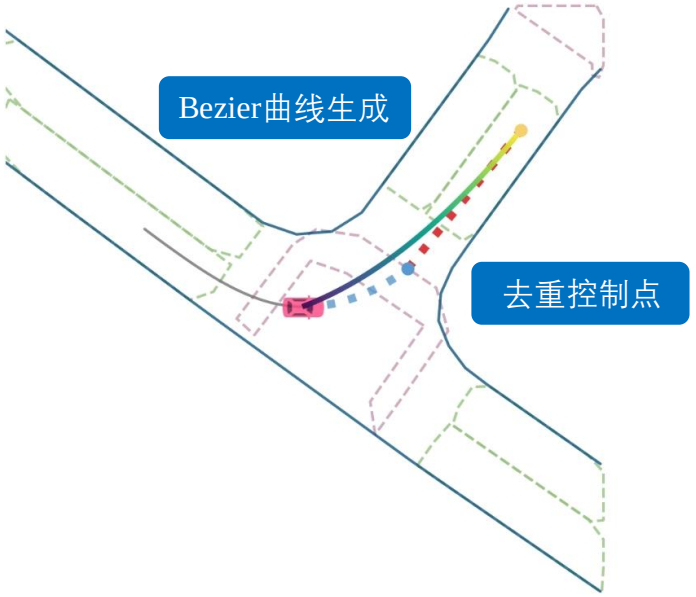


Bezier曲线生成长时轨迹

Bezier曲线生成效果

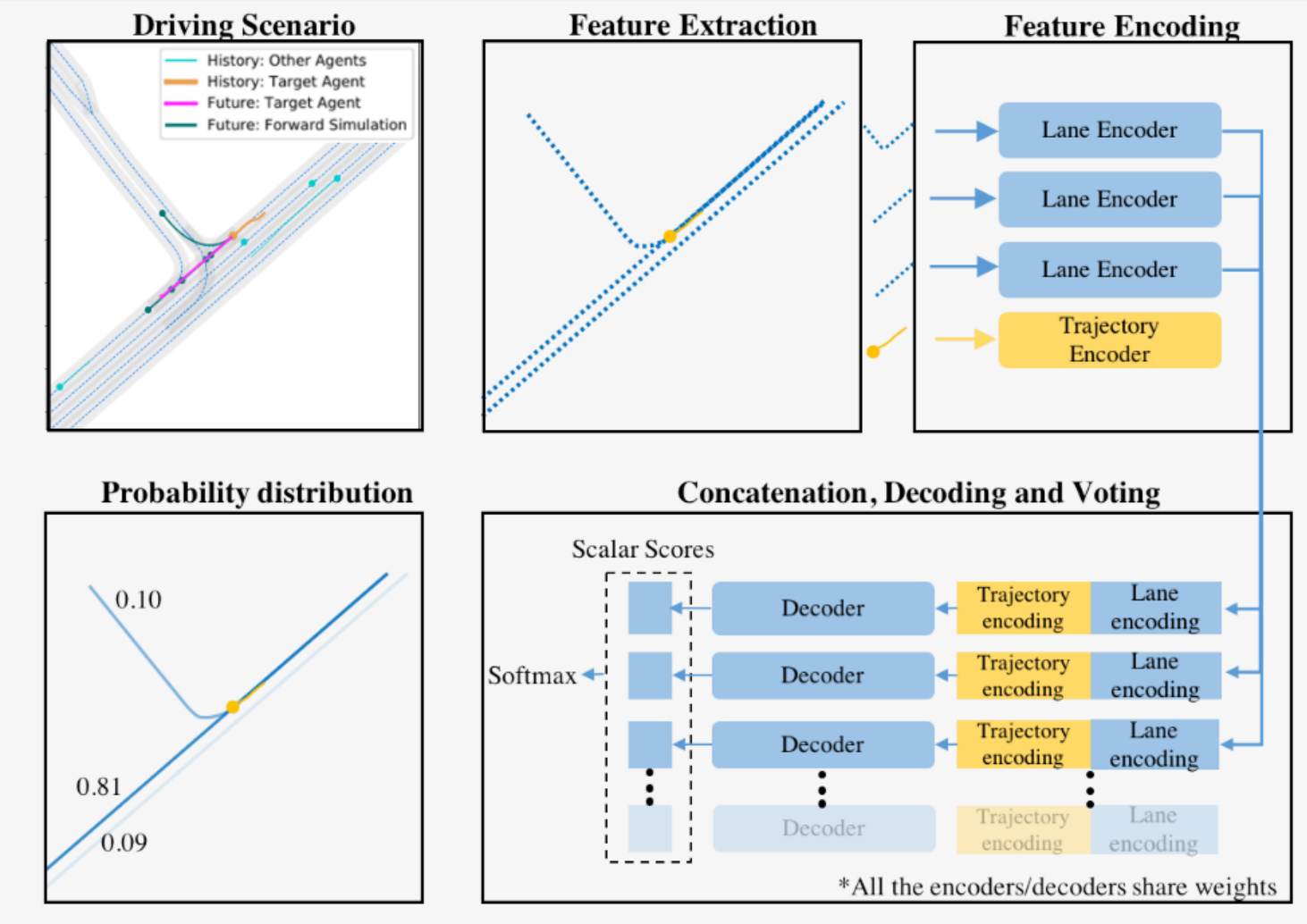






EPSILON: Intention Prediction Network

Intention Prediction Network Pipeline



输入：环境信息和历史轨迹

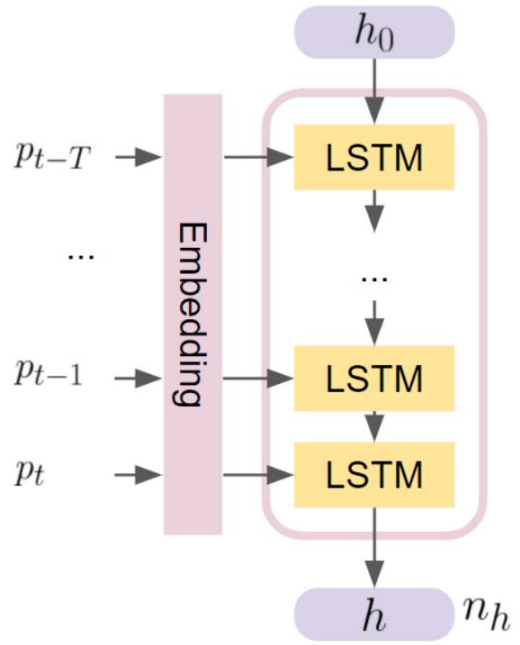
输出：候选路线的分数

EPSILON: An Efficient Planning System for Automated Vehicles in Highly Interactive Environments, https://arxiv.org/pdf/2108.07993.pdf

EPSILON: Intention Prediction Network

Intention Prediction Network Structure

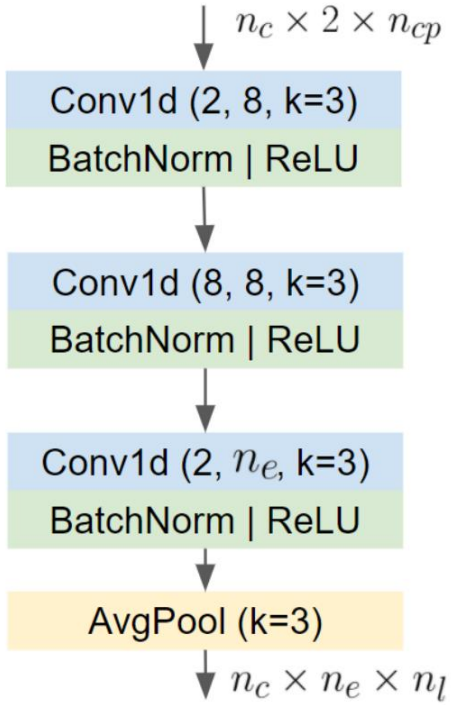
Trajectory Encoder



输入：车辆历史轨迹

输出：提取时序特征后的隐式向量

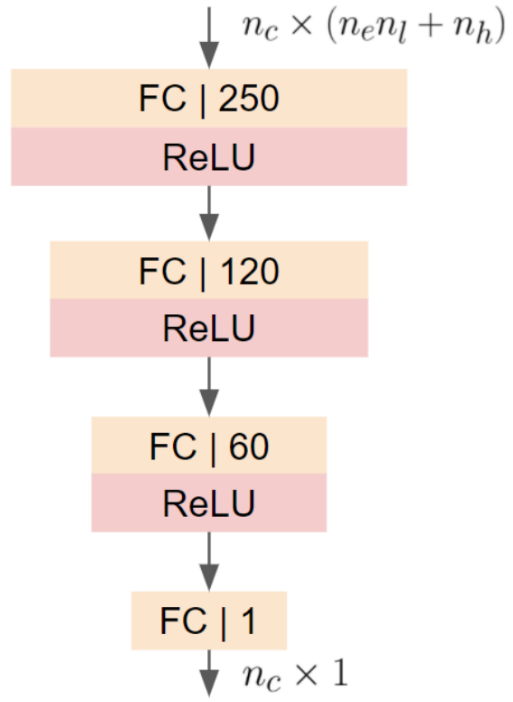
Lane Encoder



输入：目标车辆周围车道中心线采样点

输出：卷积池化后的道路信息特征

Intention Decoder



输入：道路信息特征 + 历史轨迹特征

输出：候选道路的归一化分数

LSTM是一种用于处理序列数据的神经网络，本质上也是一种特征提取器（CNN, Transformer也是）。了解LSTM可以阅读这份资料：人人都能看懂的LSTM

EPSILON: Intention Prediction Network

Intention Prediction Network Application

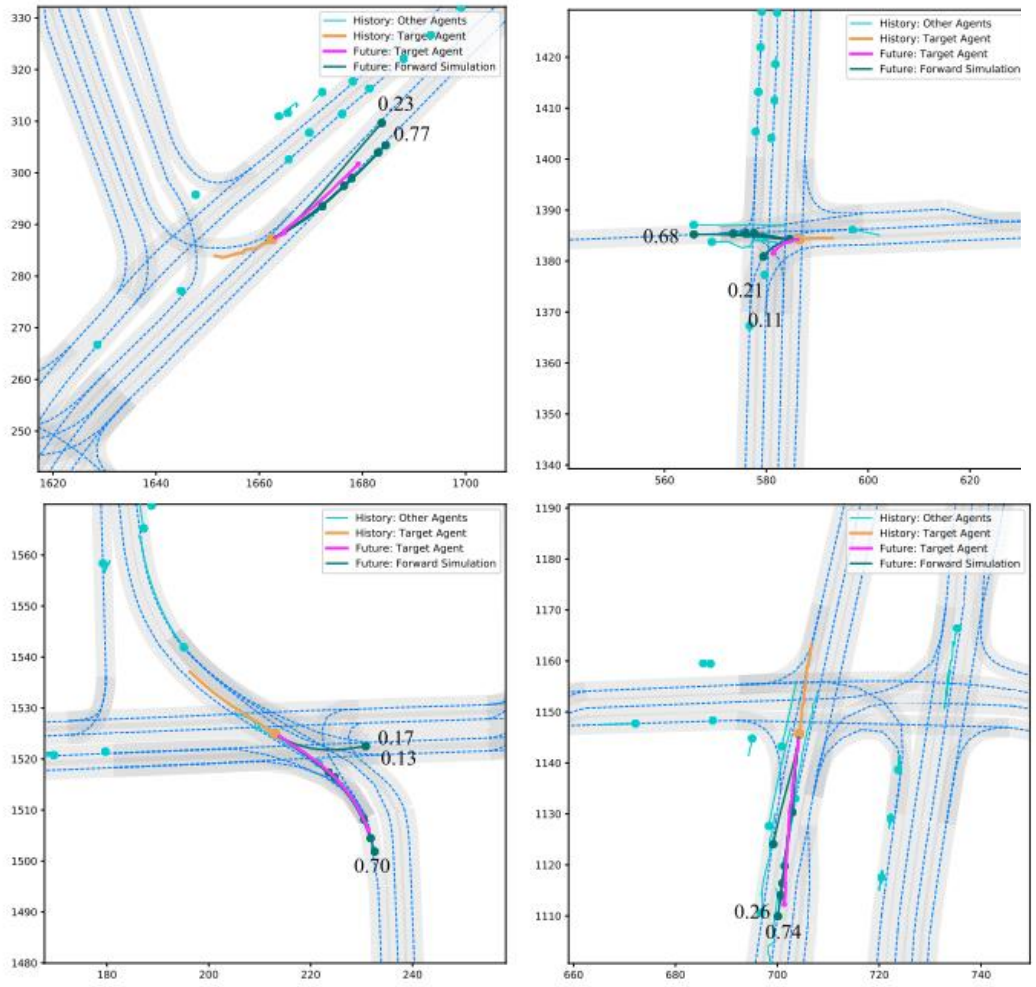
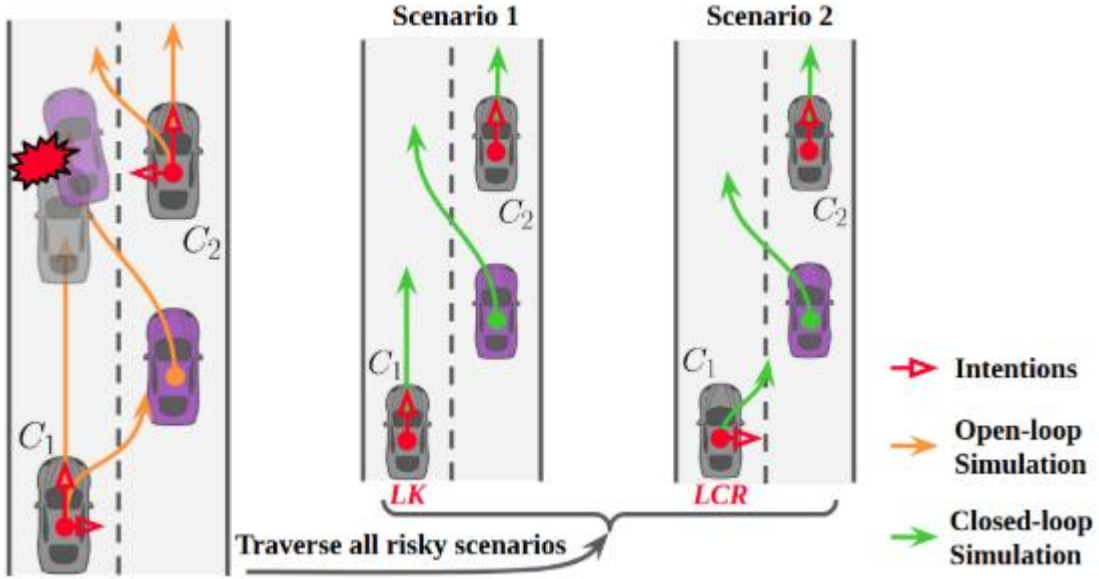
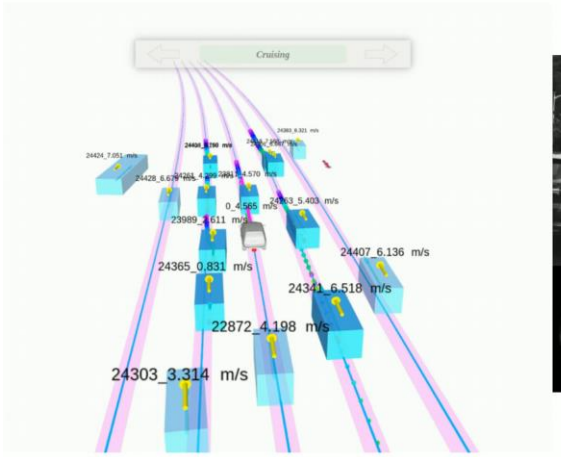


Fig. 9: Intention estimation and forward simulation on Argoverse validation set.





深蓝学院
shenlanxueyuan.com

Magic Lab

感谢聆听!

Thanks for Listening

注：更多预测与决策规划的经验分享，在官方课程答疑群内进行。请大家及时关注！