技术文档丨3D 障碍物感知

3D障碍物感知（激光雷达数据+高精度地图数据）：

1.高精度ROI过滤：通过当前位置查找当前位置附近范围内的高精度地图区域 + 激光雷达的区域进行俯视图2D视角进行取交集。

参数：range（-a, a)代表激光雷达的俯视图矩形范围。cell_size 代表这个矩形范围内的分辨率，即分成多少个小格子。

2.#1的输出就是当前区域高精度地图区域内的激光雷达点云数据。然后用2D俯视角范围中每个格子中的尽可能多的信息输入进行深度学习训练，包括：

- Maximum height of points in the cell.
- Intensity of the highest point in the cell.
- Mean height of points in the cell.
- Mean intensity of points in the cell.
- Number of points in the cell.
- Angle of the cell’s center with respect to the origin.
- Distance between the cell’s center and the origin.
- Binary value indicating whether the cell is empty or occupied.

深度学习网络模型FCNN，输出为每个格子的：

- **中心偏移**
- **对象性**
- **积极性**
- **对象高度**

然后根据这些信息进行聚合操作，然后采用MinBox的规则推断出包含六个边的障碍物长方体坐标。

每帧检测出的障碍物会保留一个序列，根据前后时刻追踪到速度方向推断下一刻的位置并对下一刻要检测的位置进行卡尔曼过滤，以此来矫正为更准确的位置。

3.激光雷达障碍物与毫米波雷达障碍物进行融合。

以下，ENJOY

3D障碍感知有三个主要组成部分：

- **LiDAR障碍物感知**
- **雷达障碍物感知**
- **障碍物结果融合**

LiDAR 障碍物感知

以下章节描述了输入为LiDAR传感器的3D点云数据Apollo障碍物感知流程：

- HD-Map ROI Filter（高精地图感兴趣区域滤波器）
- 卷积神经网络（CNN）分割
- MinBox Builder （最小框构建器）
- HM 目标跟踪器
- Sequential TypeFusion（序贯类型融合）

HDMap ROI滤波器

ROI指定了从HDMap中提取的包括路面和路口的可行驶区域。ROI滤波器处理位于ROI区域之外的LiDAR点，移除背景目标，例如道路周围的建筑物和树木。剩余的是交由后续步骤处理在ROI区域之内的点云。

给定HDMap，每个LiDAR点的隶属关系表明它是否在ROI区域内。通过查询汽车周围环境的2D量化查找表（LUT）可获得每个LiDAR点。HDMap ROI滤波器模块的输入和输出如下表所示。

输入	输出
点云：LiDAR传感器捕获的一组3D点。	HDMap定义的在ROI区域内的输入点索引。
HDMap：一组多边形，每个多边形都是一组有序的点。	

Apollo HDMap ROI滤波器通常包含三个步骤：

> - 坐标系转换。
> - ROI LUT构建。
> - 使用ROI LUT进行点查询。

对于HDMap ROI过滤器，HDMap的数据接口由一组多边形定义，每个多边形实际上是世界坐标系中的一组有序点。使用HDMap ROI对点进行查询，要求点云和多边形在相同的坐标系中表示。为此，Apollo将输入点云和HDMap多边形的点转换为以LiDAR传感器位置为原点的局部坐标系。

无论输入点是否在ROI内部，Apollo都采用将ROI区域量化为一个俯视2D网格的LUT来确定输入点。如图1所示，该LUT覆盖了一个矩形区域，该区域由HDMap边界内预先定义的空间范围限定。它描述了每个网格点与ROI的关系（即，1/0表示它在ROI内部/外部）。为了提高计算效率，Apollo使用扫描线算法和位图编码来构建ROI LUT。

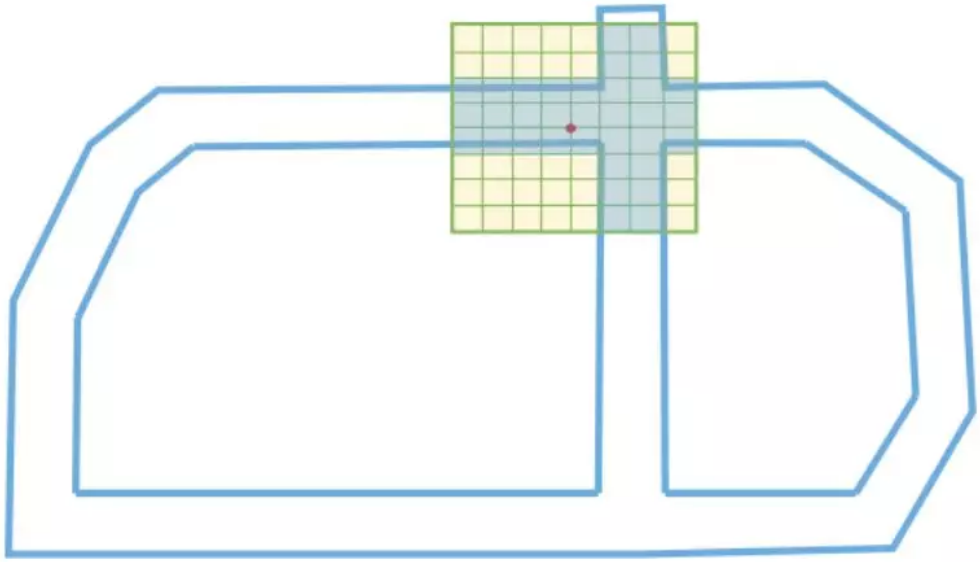

图 1 ROI lookup table (LUT)示意图

蓝线表示HDMap ROI的边界，包括路面和路口。红色实心点表示与LiDAR传感器位置对应的局部坐标系的原点。2D网格由8×8个绿色cell（单元格）组成。

ROI内的cell使用蓝色填充，而ROI外的cell则用黄色填充。

使用ROILUT查询点

基于 ROI LUT，使用两步验证方法来查询每个输入点的隶属关系。然后，Apollo进行数据编译和输出如下所示的结果。对于点的查询过程，Apollo将执行以下步骤：

- 确认该点是在ROI LUT的矩形区域内部还是外部。
- 查询LUT中该点对应的cell，以确定其与ROI的隶属关系。
- 收集所有在ROI内部的点，并输出这些点在点云中索引号。

可在以下配置文件中设置用户自定义参数：

```
modules/perception/production/data/perception/lidar/models/roi_filter/
hdmap_roi_filter/hdmap_roi_filter.conf.
```

下表给出了HDMap ROI Filter 的参数使用说明。

参数名称	用途	默认值
Range（范围）	ROI LUT（2维网格）相对于源（LiDAR传感器）的范围。	120.0米
cell_size（cell尺寸）	量化2D网格的 cell尺寸	0.25米
extend_dist（延伸距离）	ROI区域相对于 HDmap多边形的延伸距离。	0.0米
no_edge_table	开启edge_table 进行多项式掩码生成	False(否)
set_roi_service	开启roi_service 感知激光雷达模块	True(真)

卷积神经网络(CNN) 分割

在使用HDMap ROI滤波器确认周围环境后，Apollo得到过滤后的点云，该点云仅包含在ROI内的点（例如，可行驶的道路和交叉区域）。大部分背景障碍物（例如道路区域周围的建筑物和树木）都已被移除，并且ROI内的点云将被送往分割模块。该模块检测并分割出前景障碍物，例如汽车，卡车，自行车和行人。

输入	输出
点云(一组3D 点)	一组与ROI中障碍物对应的目标
标识点在ROI内部的索引号	

Apollo 使用深度卷积神经网络对障碍物进行精确检测和分割。Apollo CNN 分割算法包括以下四个部分：

- 通道特征提出
- 基于CNN的障碍物预测
- 障碍物聚类
- 后处理

以下章节详细介绍Apollo 中进行分割的深度卷积神经网络：

给定点云数据帧，Apollo构建其鸟瞰视图（即，投影到XY平面），它是局部坐标系中的2D网格。根据x和y坐标，可以将相对于原点（即，LiDAR传感器）的预先定义范围内的每个点都量化到2D网格中的一个cell。在量化之后，Apollo为网格中每个cell中的点计算其8个统计值，这些需要计算的统计值如下所示：

- cell中点的最大高度。
- cell中最高点的亮度。
- cell中点的平均高度。
- cell中点的平均亮度。
- cell中点的数量。
- 相对于原点，cell中点的角度。
- cell中点到原点的距离。
- 表示单元是空还是以被占据的二进制数值。

基于上述通道特征，Apollo使用一个全卷积神经网络来预测cell中障碍物的属性，包括相对于物体中心的偏移量-称为中心偏移，objectness(是否是目标的概率)， positiveness（有效目标概率）以及物体高度。如图2所示，网络的输入是一个W×H×C通道的图形，这里：

- W 表示网格的列数。
- H 表示网格的行数。
- C 表示特征通道的数量。

Apollo使用的全卷积神经网络包括三层，分别为：

- 下游编码层 (特征编码器)。
- 上游解码层(特征解码器decoder)。
- 障碍物属性预测层(预测器)。

特征编码器将通道特征图像作为输入，并连续地对其空间分辨率进行下采样以提高特征抽象程度。之后，特征解码器对编码的特征图像逐渐上采样，使其大小与到输入的2D网格的空间分辨率一致，这可以恢复特征图像的空间细节，以便于cell障碍物属性预测。下采样和上采样是以带非线性激活层(例如, ReLu)的卷积/反卷积层堆栈实现的。

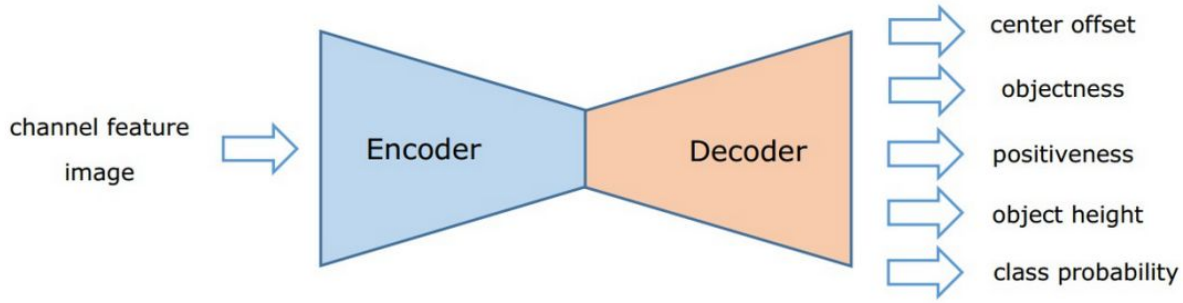

图2 进行网格单元障碍物预测的全卷积神经网络

障碍物聚类

在基于CNN的预测之后，Apollo 可获得每个cell的预测信息。Apollo 使用包含以下5个属性的cell目标属性图：

- 中心偏移量
- objectness（是否包含目标）
- positiveness（有效目标概率）
- 目标高度
- 类别概率

为了生成障碍物，Apollo基于cell的中心偏移预测结果构建有向图，并搜索连接的组件。如图3所示，每个cell由图中的一个节点表示，图的有向边则基于cell的预测中心偏移量构建，指向单元格对应的父节点。

根据此有向图，Apollo使用压缩Union Find（并查集）算法来高效地查找连接的组件，每个组件都是候选障碍物目标簇。Objectness是描述一个cell是一个有效目标的概率。因此，Apollo将objectness值小于0.5的cell定义为非目标cell。因此，Apollo为每个候选目标簇过滤掉空cell和非目标cell。

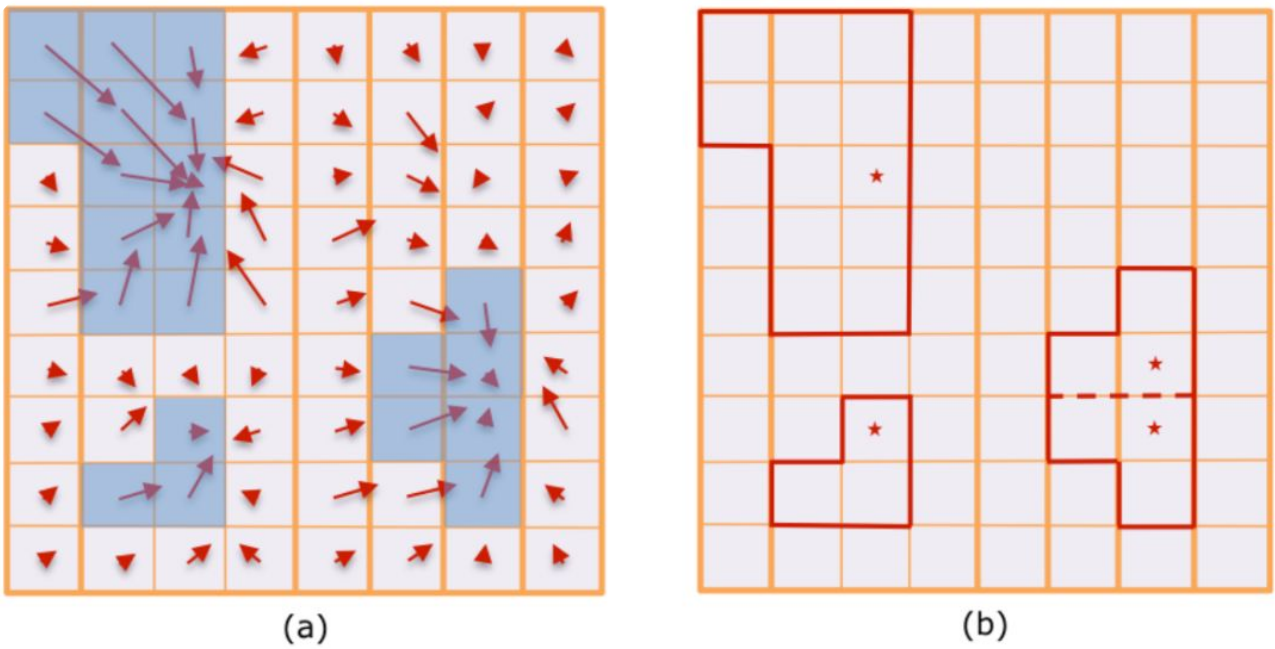

图3 障碍物聚类示意图

- 红色箭头表示针对每个cell预测的目标中心偏移方向。
- 使用蓝色覆盖的区域表示objectness 概率不小于0.5的目标cell。

> - 红色实线多边形内的cell组成一个候选目标簇。
> - 红色五角星表示连接组件对应子图的根结点（cell）。

一个候选目标簇可以由多个相邻的组件组成，这些组件的根节点彼此相邻。

Apollo对候选目标簇内各个cell的类别概率求和来计算各类候选障碍物的概率，包括车辆，行人，自行车和未知的障碍物。具有最大平均概率的障碍物类型即为该目标簇的最终分类结果。

聚类之后，Apollo获得一组候选目标簇，每个候选目标簇包括几个cell。在后处理中，通过对计算该簇所包含的cell的positiveness和目标高度的均值，Apollo首先计算出每个候选簇的检测置信度。然后，Apollo移除比预测目标高度高很多的点，并收集每个候选簇有效cell内的点。最后，Apollo移除置信度非常低或者包含点数比较少的候选簇，输出最终障碍簇/分割。

可以在以下配置文件中设置用户自定义的参数

```
1   modules/perception/production/data/perception/lidar/models/cnnseg/
2   velodyne128/cnnseg_param.conf.
```

下表给出了CNN 分割子模块的参数及其说明和默认值。

参数名	用途	默认值
objectness_thresh	障碍物聚类阶段过滤没有目标的cell的objectness参数的阈值	0.5
model_type	网络类型，例如，RTNet 意味使用tensorRT加速网络	RTNet
confidence_thresh	在后处理阶段用于过滤候选簇的检测置信度阈值	0.1
confidence_range	相对于原点（LiDAR 传感器）	85.0米

	的可认为具有较好检测效果的置信范围	
height_thresh	如果height_thresh非负，比预测目标高度还要高height_thresh 的点将在后处理阶段被过滤掉。	0.5米
min_pts_num	在后处理阶段，包含点数小于min_pts_num 的候选簇将被过滤掉	3
ground_detector	地面检测器类型	SpatioTemporalGroundDetector
gpu_id	基于CNN的障碍物预测模块使用的GPU的ID号	0
roi_filter	ROI filter类型	HdmapROIFilter

network_param {instance_pt_blob, etc}	不同的caffe输入和输出层blob的类型。	layer predefined(预定义层）
feature_param {width}	2D网格的列数，即X轴包含的cell数目	864
feature_param {height}	2D网格的行数，即Y轴包含的cell数目	864
feature_param {min_height}	相对于原点（LiDAR sensor）的最小高度	-5.0米
feature_param {max_height}	相对于原点（LiDAR sensor）的最大高度	5.0米
feature_param {use_intensity_feature}	启用输入通道强度特征	false
feature_param {use_constant_feature}	启用输入通道常量特征	false
feature_param {point_cloud_range}	相对于原点（LiDARsensor），2D网格区域的范围	90 米

注意: 提供的模型例子仅用于实验。

MinBox 构建器

目标构建器组件会为检测到的障碍物建立bounding box（边界框）。由于遮挡或距离LiDAR传感器太远，形成障碍物的点云可能比较稀疏并且仅能覆盖表面的一部分。

因此，box builder 的主要目的是基于给定的多边形点来恢复完整的bounding box。bounding box的主要用途是在点云比较稀疏的情况下也能估计障碍物（例如，车辆）的航向。同样，bounding box也可用于可视化障碍物。

该算法的思想是根据多边形点的一条边找到整个区域。在下面的示例中，如果AB是边，Apollo会将其他多边形点投影到AB上，并建立具有最大距离的交叉点对。这是bounding box的边之一。

然后可直接得到bounding box的另一边。通过迭代处理多边形中的所有边，如图4所示，Apollo构建出一个具有6条边的bounding box。最后，Apollo选择具有最小面积的方案作为最终bounding box。

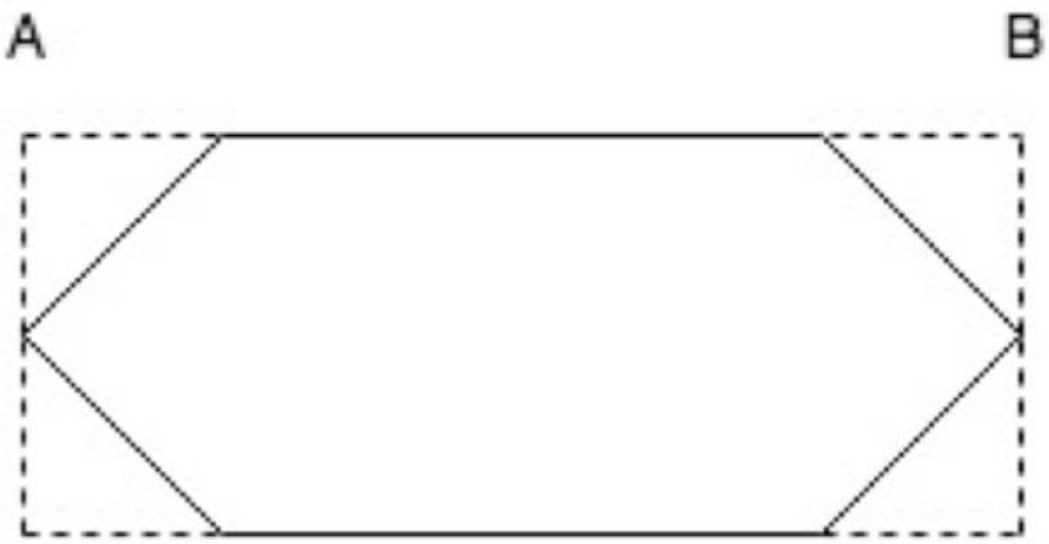

图 4 MinBox Object 构建器演示

HM目标跟踪器

HM目标跟踪器的主要功能是跟踪分割步骤检测到的障碍物。通常，它通过将当前检测结果与现有跟踪列表相关联的方式，形成和更新跟踪列表。如果原来的目标都不再出现则删除旧跟踪列表，在确认新的检测结果之后会生成新的跟踪列表。

关联之后，将会估计更新后的跟踪列表的运动状态。HM目标跟踪器使用Hungarian算法（匈牙利算法）对检测和跟踪（detection-to-track）进行关联，使用Robust Kalman Filter（鲁棒卡尔曼滤波器）进行运动估计。

检测到跟踪关联

将检测与现有的跟踪列表进行关联时，Apollo构建了一个二分图并使用Hungarian算法找到具有最小代价（距离）的最佳检测-跟踪匹配。

计算关联距离矩阵

首先建立一个关联距离矩阵。一个给定的检测和跟踪之间的距离可以通过一系列关联属性进行计算，这些关联属性包括运动一致性和外观一致性。HM跟踪器中距离计算中使用的一些属性如下所示：

关联属性名称	评估一致性的说明

location_distance	运行
direction_distance	运动
bbox_size_distance	外观
point_num_distance	外观
histogram_distance	外观

另外，有一些距离**权重**的重要参数，结合上述关联特征用于最终的距离测量。

根据**关联距离矩阵**，Apollo构造了一个二分图并使用Hungarian算法，基于最小化距离成本找到最佳检测-跟踪的匹配，如图5所示。它在时间复杂度小于O（n ^ 3）。为了提高计算性能，通过删除距离大于最大距离阈值的顶点将原始二分图切割成子图之后再来实现Hungarian算法。

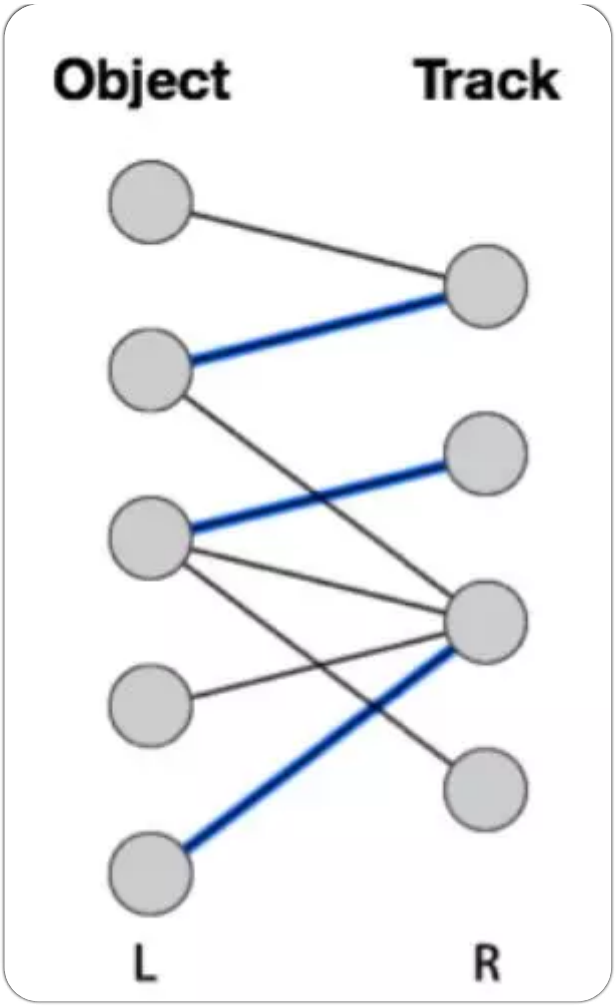

图 5 二分图匹配示例

跟踪运动估计

在检测到**跟踪关联**之后，HM 目标跟踪器使用Robust Kalman Filter并基于恒定速度运动模型来估计当前跟踪列表的运动状态。运动状态包括belief anchor point and belief velocity，它们分别对应于3D位置和3D速度。为了克服由检测引起的分散问题，在跟踪器的过滤算法中采用了鲁棒统计技术。

冗余观测

用作滤波算法输入的测量速度将在一系列冗余观测中选择，包括**锚点移位**，**边界框中心移位**和**边界框角点移位**。冗余观测为过滤测量带来了额外的鲁棒性，因为所有观测失败的概率远小于单次观测失败的概率。

Breakdown(衰减)

Gaussian Filter（**高斯滤波**） 算法假设噪音是服从高斯分布的 。然而，该假设可能在运动估计问题中失效，因为其测量的噪声可能来自于肥尾效应分布。Apollo在滤波过程中使用breakdown阈值来抵消更新增益的过高估计。

根据关联质量更新

原始卡尔曼滤波器更新状态时是不区分测量质量的。但是，测量质量是过滤噪声的有益指标并且也可以估算。例如，在关联步骤中计算的距离可以作为测量质量的合理估计。根据关联质量更新滤波算法的状态可以增强运动估计问题的鲁棒性和平滑性。

HM目标跟踪器的高级工作流程如图6所示。

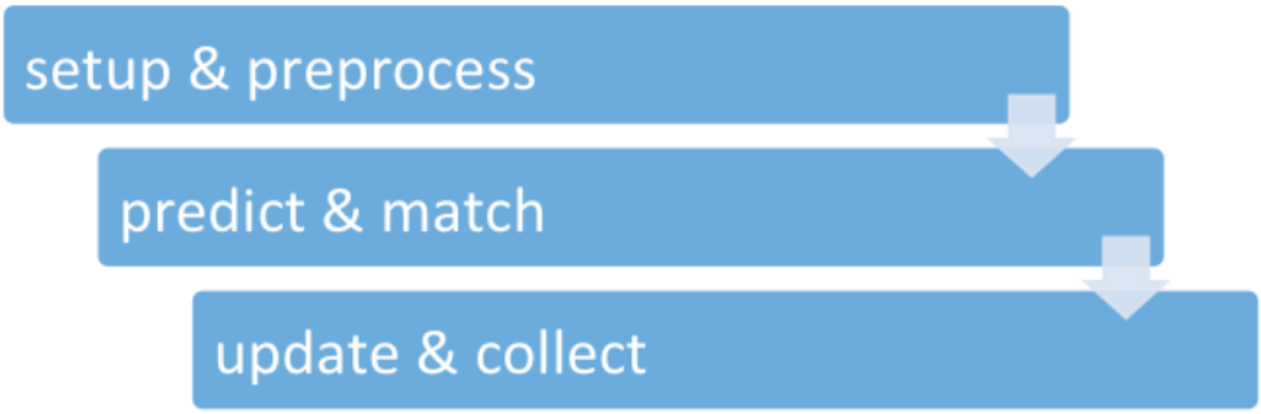

图 6 HM 目标跟踪器的工作流程

HM目标跟踪器工作流程的要点如下：

1. 构造跟踪目标并将其转换到世界坐标系。
2. 预测现有跟踪列表的状态并与检测结果进行匹配。
3. 更新最新跟踪列表的动作状态并收集跟踪结果。

Sequential Type Fusion（序贯类型融合）

Apollo利用**基于线性链条件随机场（CRF）的序贯算法**，来平滑障碍物类型并减少整个跟踪轨迹过程中的类型切换，该算法可以使用如下方式描述：

$$P(y|x) = \frac{1}{Z_x}\prod_t \varphi_t(y_t, x)\cdot\theta_t(y_t, y_{t-1}, x)$$

$$\varphi_t(y_t, x) = \alpha\cdot\exp(P(y_t|x))$$

$$\theta_t(y_t, y_{t-1}, x) = \beta\cdot\exp(P(y_t|y_{t-1}, x))$$

其中一元项表示单个节点，而二元项表示每个边缘。

一元项中概率是基于CNN的预测输出的类概率，二元项中的状态转移概率可以通过从时间t-1到时间t的障碍类型转换来建模，这是根据大量的障碍物轨迹统计学习得到的。

特别的，Apollo还使用学习到的混淆矩阵来表示预测类型到真正类型的变化概率，以此来优化原始的基于**CNN预测**的类概率。
使用Viterbi（维特比）算法，通过求解以下问题来优化序贯障碍物类型：

$$y^* = arg\max_y \mathrm{P}(y|\mathrm{x})$$

IV 雷达探测器

给定来自雷达传感器的数据，遵循如下所述的基本处理过程。

首先，要扩展跟踪ID，因为Apollo需要一个用于全局跟踪ID来进行ID关联。原始雷达传感器提供仅由8位表示的ID，因此难以确定在两个相邻帧中具有相同ID的两个目标是否表示跟踪历史中的单个目标，尤其在存在丢帧问题的情况下。Apollo使用雷达传感器提供的测量状态来处理这个问题。

同时，如果目标与前序帧中具有相同跟踪ID的目标相距很远，Apollo将为其分配一个新的跟踪ID。

其次，使用误报滤波器去除噪音。Apollo对雷达数据设置一些阈值来过滤可能是噪声的结果。然后，根据雷达数据以统一的目标格式构建目标。Apollo通过标定结果将目标转换到世界坐标系。原始雷达传感器提供物体的相对速度，因此Apollo使用来自定位模块的**主车速度**。Apollo将这两个速度相加来表示检测到的物体的绝对速度。

最后，使用HDMap ROI 滤波器来获得感兴趣的目标。只有在ROI区域内的目标才能用在传感器融合算法中。

V 障碍物结果融合

设计传感器融合模块的目的是融合 LiDAR 的跟踪结果和毫米波雷达的检测结果。Apollo首先基于跟踪融合项的ID将传感器结果与融合项进行匹配。然后，为不匹配的传感器结果和不匹配的融合项构建关联矩阵，以获得最佳匹配结果。

融合项管理

在Apollo系统中，有发布传感器的概念。给定的雷达结果将被缓存，给定的LiDAR结果会触发融合动作。传感器融合输出的频率与发布传感器的频率相同。Apollo的发布传感器是LiDAR。传感器结果按照传感器时间戳排序输入到融合流水线中。Apollo保留了所有传感器结果。

在Apollo中，不同传感器目标的生命周期不同。如果至少有一个传感器结果保存着，则目标将依旧处于活动状态。Apollo感知模块提供汽车周围的短程区域的LiDAR和雷达的融合结果，而远距离区域只提供雷达结果。

传感器结果到融合列表的关联

当要将传感器结果与融合列表进行关联时，Apollo首先匹配与传感器具有相同跟踪ID的融合项，然后构建二分图并使用Hungarian算法，基于最小距离代价为不匹配的传感器结果和融合列表找到最佳的结果-融合匹配。Hungarian算法与HM 目标跟踪器使用的算法相同。距离代价可以计算传感器结果的锚点与融合项之间的欧几里德距离来求解。

运动融合

Apollo使用自适应**卡尔曼滤波器**（Adaptive Kalman filter），基于等加速度运动模型来预测当前项目的运动状态。运动状态包括其的belief anchor point, belief velocity and belief acceleration，分别对应于3D位置，其3D速度和加速度。Apollo仅使用来自传感器结果的位置和速度。

在运动融合中，Apollo缓存所有传感器结果的状态，并通过卡尔曼滤波器计算加速度。Apollo提供的LiDAR跟踪器和**雷达探测器**数据中位置和速度具有不确定性。Apollo将所有状态和不确定项都送到自适应**卡尔曼滤波器**中以获得融合结果。Apollo在滤波过程中使用breakdown阈值来抵消更新增益的过高估计。