# 传感器感知融合

**感知融合内容**

* 一般是指运动障碍物（车辆、行人、自行车等）的融合

* 除了感知融合之外，在自定位模块中也会融合IMU、GPS、激光雷达等多种传感器

* 激光+相机+毫米波都有目标检测功能，因此可以融合各自检测结果提升精度

* 其他感知任务（例如红绿灯）只有前向相机负责，其他传感器不参与

**融合重要性**

* 运动目标检测是感知最难的任务，单一传感器很难做到理想精度

* 各种传感器优势是互补的，好的融合算法能充分利用传感器的优势

* 融合能充分利用“**时序一致性**”“**空间一致性**”等先验条件提升精度

## 融合的先决条件

### 空间同步

* 多个传感器必须在同一个坐标系下（**多传感器标定**)

* 车身坐标一般都以车后轴中心投影到地面上的点为原点

* 各自标定后有一定误差，通过**多相机联合标定，相机+雷达联合标定**来减少误差

### 时间同步

* 车辆是运动的，车身坐标时刻在发生改变

* 不同传感器生成数据的时刻并不完全一致，甚至帧率也是不同的（如图像为30帧，雷达一般为10帧）

* 多个传感器融合时，数据都转换到同一时刻的车身坐标（**运动补偿**）——雷达转一圈需要100ms，按照60km/h来算，已经前进了1.6m

## 融合的架构

多传感器融合的体系结构：**分布式、集中式和混合式**

* 分布式：先对各个独立传感器所获得的原始数据进行局部处理，然后再将结果送入信息融合中心进行智能优化组合来获得最终的结果。分布式对通信带宽的需求低、计算速度快、可靠性和延续性好，但跟踪的精度却远没有集中式高

* 集中式：集中式将各传感器获得的原始数据直接送至中央处理器进行融合处理，可以实现实时融合。其数据处理的精度高，算法灵活，缺点是对处理器的要求高，可靠性较低，数据量大，故难于实现

* 混合式：混合式多传感器信息融合框架中，部分传感器采用集中式融合方式，剩余的传感器采用分布式融合方式。混合式融合框架具有较强的适应能力，兼顾了集中式融合和分布式的优点，稳定性强。混合式融合方式的结构比前两种融合方式的结构复杂，这样就加大了通信和计算上的代价

**在自动驾驶中，较多采用分布式融合方案，即每个传感器自身也有融合算法（例如相机的视觉跟踪），最后再通过统一的异构传感器融合**

按照融合的阶段分为：**前融合、中融合和后融合**

* 前融合：也就是**数据层融合，将不同传感器的数据进行对齐，并将对齐后的数据输入到一个模型中进行处理**。代表算法有环视拼接、点云着色、图像深度补全

* 中融合：也就是特征层融合，将不同传感器的数据先进行一些处理得到统一的特征，再在特征层面上进行融合，代表算法为多模态模型

* **后融合**：也就是决策层融合，是各个传感器分别经过独立的模型，输出的结果在语义层面上进行融合，是**目前自动驾驶主流的融合方式**

按照传感器的种类分为：**同构传感器融合、异构传感器融合**

* 同构传感器融合：所有的传感器规格都是一样的，可以认为有相同的误差范围以及相同的物理特性。自动驾驶中**多相机融合**就是同构传感器融合。

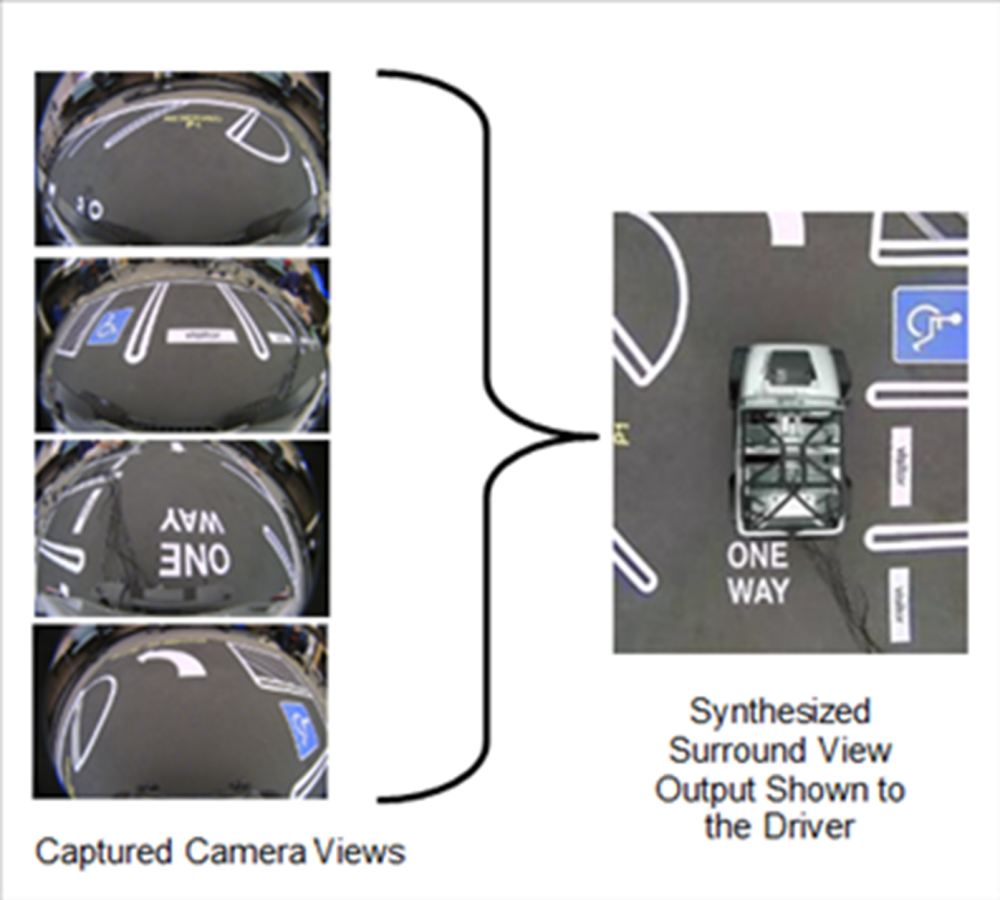
* 异构传感器融合：不同类型的传感器之间做数据融合，或者不同参数的传感器进行融合。自动驾驶中**相机和雷达**的融合属于这种类型

* 混合式融合：既存在同构传感器，也存在异构传感器，融合时通常会采用分布式融合的策略

## 前融合

### 数据层融合

#### 环视拼接



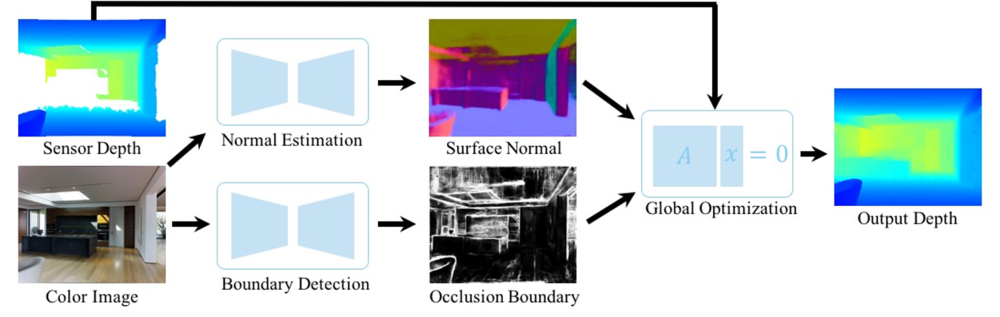
* 使用**IPM投影**的方式，将每一张图片都转换为俯视图（BEV）

* 根据相机位置关系，将多个相机（ 3-4张鱼眼相机）的俯视图拼接到一起

* 地面元素可真实还原，但有高度的物体会产生严重的变形

* 常用于**自主泊车场景的停车位检测**，也可用于近距离障碍物、freespace检测

#### 深度补全

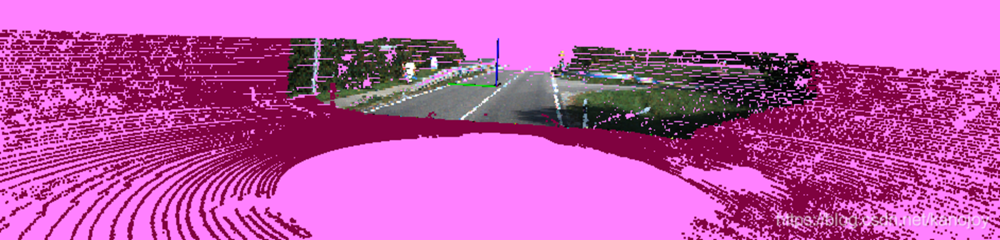


* **深度补全，是通过lidar获取的稀疏点云，通过算法来获得稠密点云，从而给图像的每一个像素一个深度信息，生成RGBD图像**

* 通过深度补全获得的RGBD图像可以作为感知模型的输入，**实现精度更高的3D目标检测**

* 深度补全可以基于传统视觉方法来做，也可以用CNN来做，目前基于深度学习方法是主流做法。一般是输入带有部分像素深度的图片到CNN网络，预测出所有像素深度。

#### 点云染色



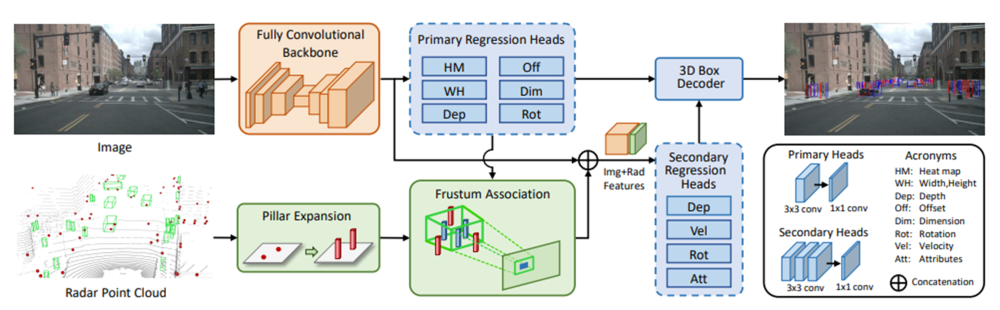
* 通过将点云投影至图像，并将图**像对应的颜色附加到点云数据上，可以得到染色之后的点云**

* 染色点云可以丰富点云的语义信息，可作为点云模型的输入，也可用于**3D标注**

* 由于图像与点云的FOV并不完全一致，因此会出现某些点云没有对应颜色的情况

### 特征层融合

#### CenterFusion



* CenterFusion部分借鉴了**Frustum-PointNet**的特征融合思想，将radar点云融合进目标检测网络，是为数不多的camera+radar融合方法

* 图像特征提取部分，使用的是CenterNet来预测2D的bounding-box

* 雷达特征提取部分，用的是类似pointpillars的方法将雷达点云扩展成柱子

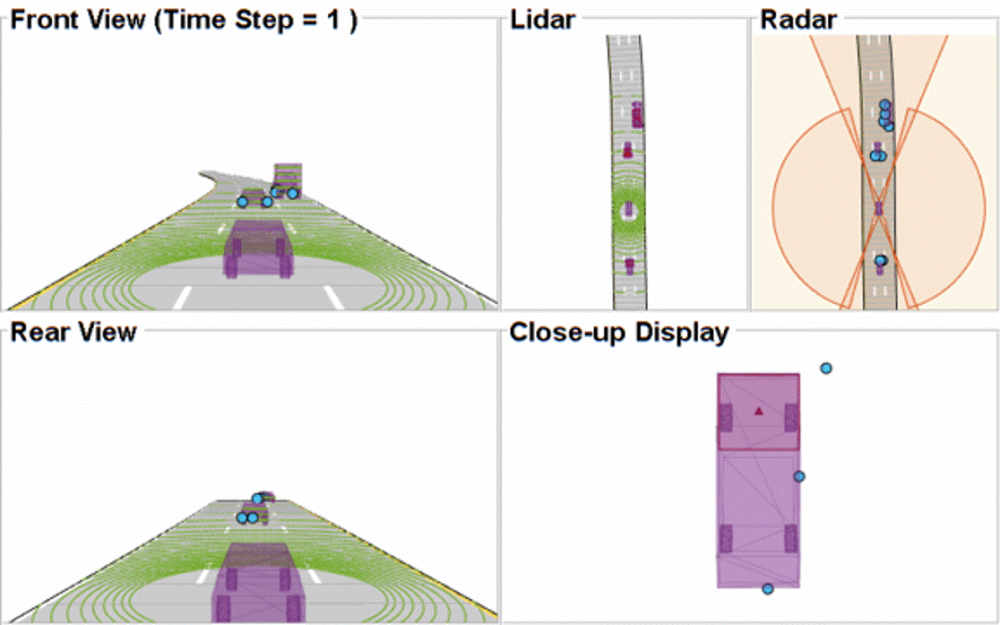
* 通过**2D的box生成视锥，并将最前面的这个柱子取出来，得到其速度和距离，**与图像特征融合进入第二个阶段预测速度、距离、朝向角、属性等。

#### BEVFusion

参照语雀笔记

此处为语雀内容卡片，点击链接查看：<https://yuque.antfin.com/zppdqz/sigipx/ylr5by>

## 后融合

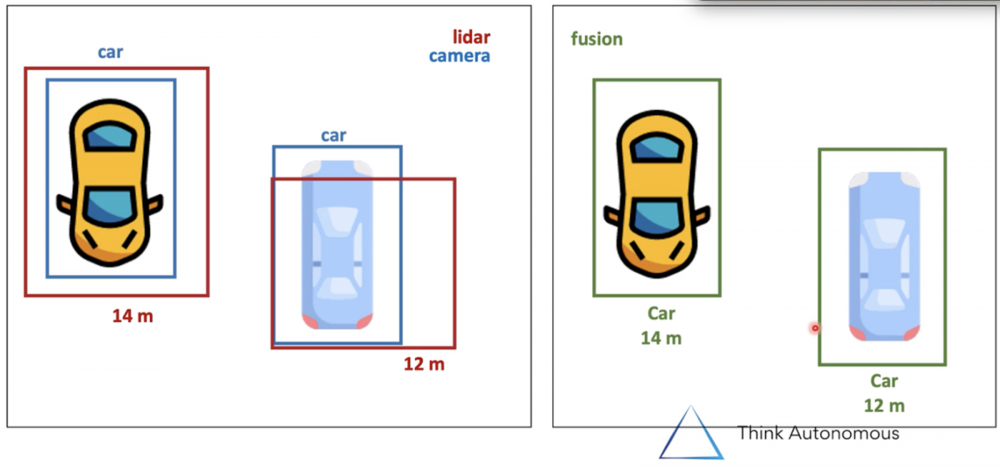


* 后融合是将各传感器给出障碍物检测结果进行融合，一般情况下**只能拿到3D框信息**，不使用传感器原始数据或者特征。

* 在自动驾驶中，后融合能够在感知不能达到100%精度的情况下，提供较高的融合精度。融合模块也经常会根据一些先验知识加入过滤规则，将不正确的检测结果过滤掉（例如**速度超过10m/s的行人，超过2米高的行人**），因此被认为是“兜底模块”

* 后融合能解决感知偶尔失效的情况（检测断断续续），是一种灵活的冗余方案

### 目标关联



* 目标关联会用在**时序目标关联、多传感器关联**等情况，用法都是一样的

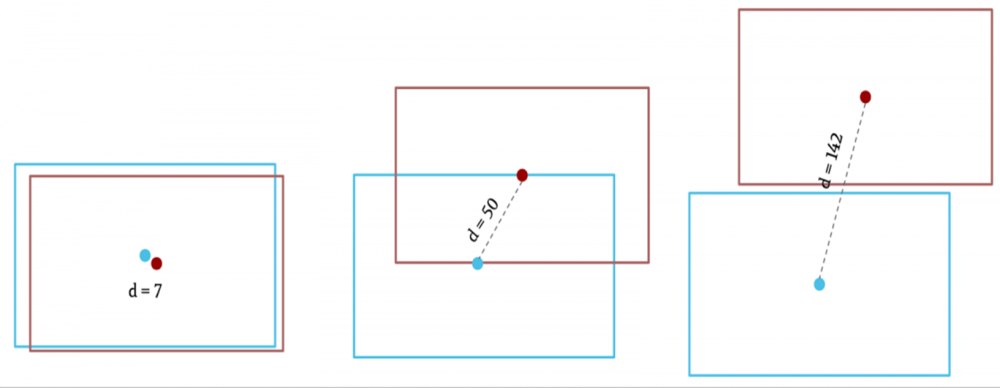
* 一般仅根据目标3D信息进行关联，也有些系统中会使用模型的embedding特征

* 对**3D位置估计要求很高**，关联失败的情况基本上都是由于3D信息偏差太大导致

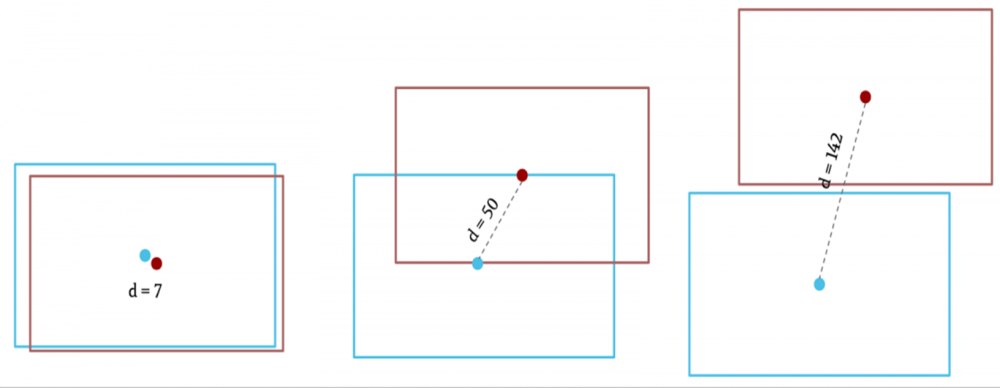
* 目标关联最常用的算法是**匈牙利匹配**，很多研究集中在如何改进相似度计算方法

#### 目标相似度

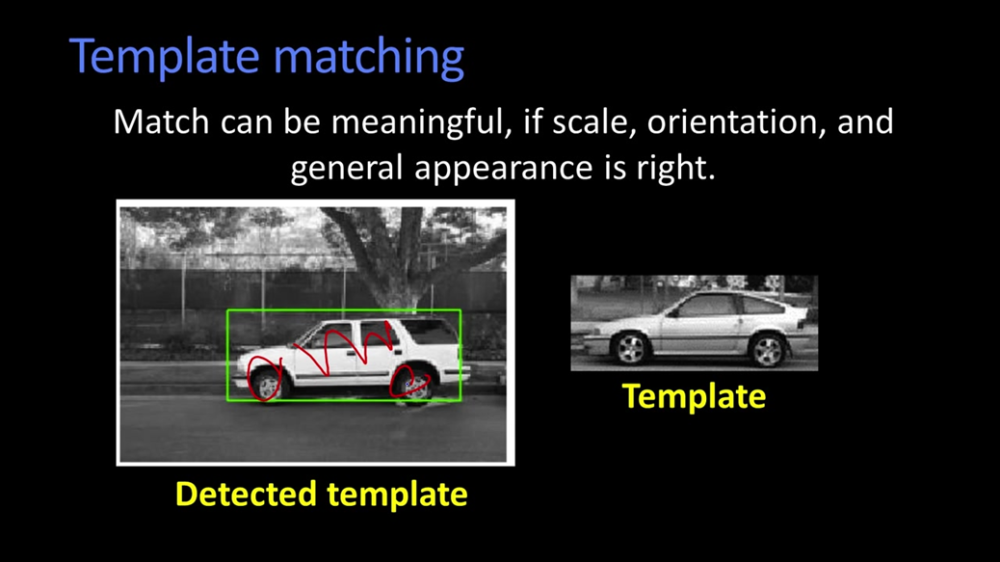
* 中心点距离



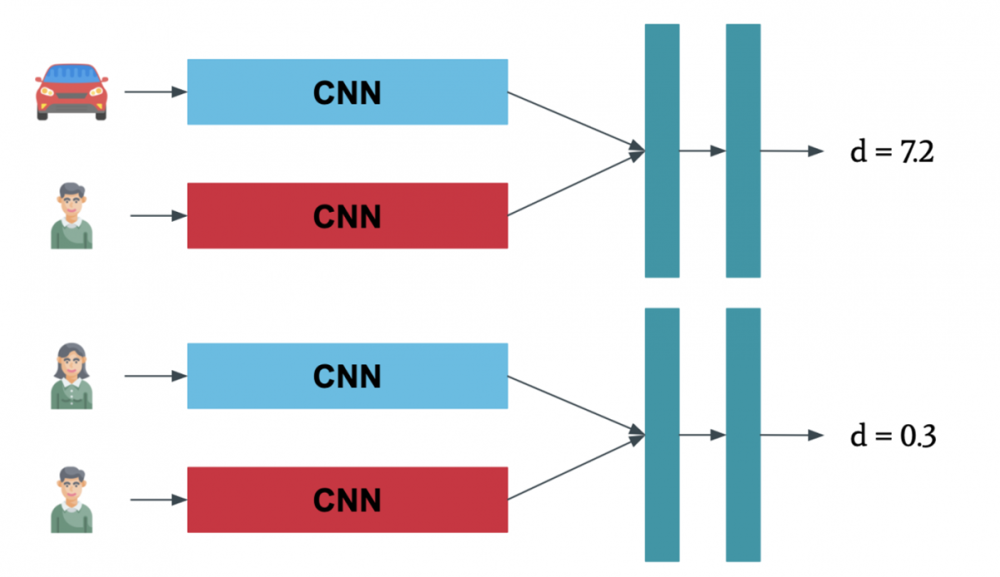
* 3D IOU



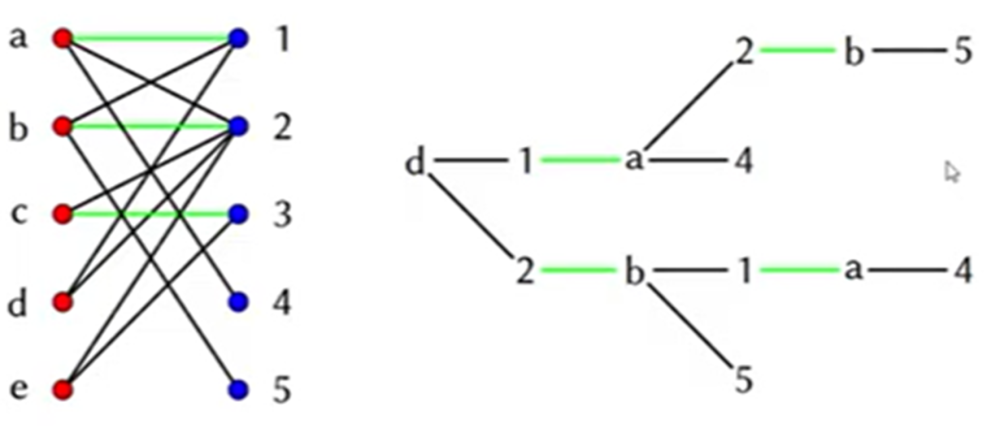
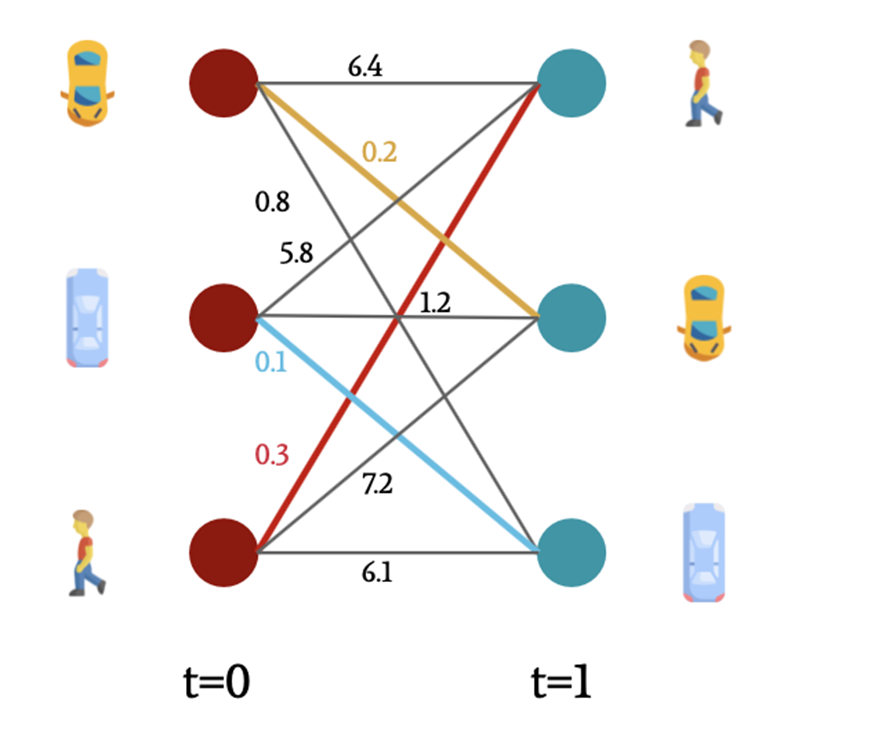
* 外观相似度



* CNN特征相似度



#### 匈牙利匹配

* 

**原理：**

* 匈牙利匹配的目标，在于找到一个全局最优匹配，也就是所有的源和目标的匹配代价之和（框的距离）最小

* 核心原理在于**迭代寻找当前匹配关系的“改进”，也就是尝试找到比当前的匹配方法更好的匹配**

**步骤：**

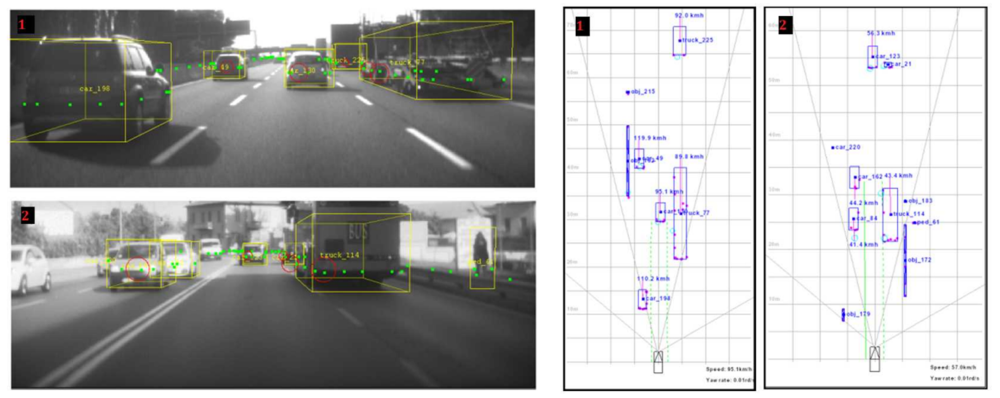
* 先随意建立一个匹配关系（为每一个源找到任意一个未被占用的目标）——**计算出当前的匹配的代码并保存**

* 选择一个源，然后尝试改变一下匹配的目标，同时将与这个目标匹配的源做改变，直到完成所有的“腾挪”

* 看看新的匹配关系是否更优，如果更优就接收这个匹配；遍历这个源可选的所有目标，用腾挪法建立新的匹配关系，找到对于这个源的最优匹配，将当前匹配关系调整为这个匹配

* 遍历其他的源，重复2-3步为每一个源找到当前情况下最优匹配路径

### 目标关联-相机+毫米波



* 相机主要输出障碍物的bounding box、类别和不太精确的距离，而毫米波雷达能够输出类别、距离和速度和方位角

* 通过类别、距离、方位角，可以对相机和毫米波输出的障碍物进行关联——**分别有对应权重信息，毫米波的类别不是很准**

* 一旦关联成功之后，后续的帧都会通过**时序跟踪**来锁定关联关系，如**相机图像中已经对目标进行跟踪了**

#### 时序目标关联

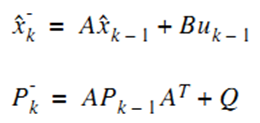


* 需要根据目标的运动情况，来计算出前一帧的目标在当前帧的预估位置来实现准确关联，并由关联关系得到目标的准确位置，从而可以预测出下一帧的位置

#### 卡尔曼滤波

根据连续输入的测量值，结合使用先验假设得到的估计值，经过卡尔曼滤波后可得到更加准确的修正测量值（预估值和测量值的融合）

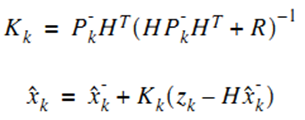
首先要计算**预测值**、**预测值和真实值之间误差协方差矩阵**



* 预测(先验)值由上一时刻的估计值和当前时刻的输入量得到的

* 预测值与真实值之间误差协方差矩阵是由估计值与真实值之间误差协方差矩阵推导古来，**Q是过程噪声协方差，表示估计值有多么不准**确

根据这两个就可以计算卡尔曼增益K，然后再得到估计值，其中H是状态转测量矩阵，**R是测量噪声协方差，表示测量值有多么不准确**



最后还要计算估计值和真实值之间的误差协方差矩阵，为下次递推做准备



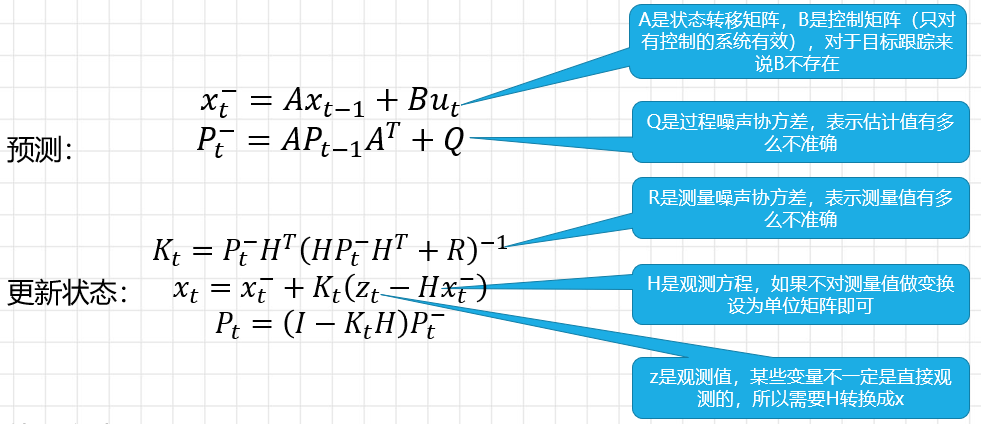
**总结:**

* 首先根据上一时刻的估计值来计算当前时刻的预测值，因此当前预测值的误差跟上一时刻估计值的误差有关系

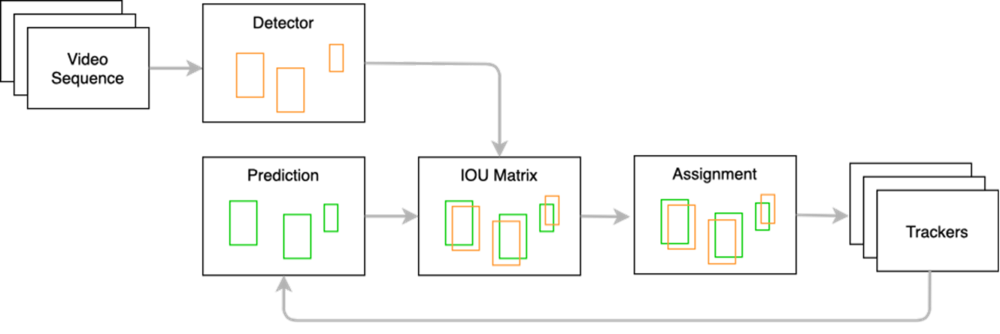
* 因此接下来就需要根据上一时刻估计值的误差协方差矩阵来计算当前时刻预测值的误差协方差矩阵，同时引入Q来表示估计值有多么不准

* 接下来就需要根据已经得到当前时刻的预测先验值和对应预测值得误差协方差矩阵，再加上当前时刻得测量值来计算当前状态得估计值，这里得K就是为了使当前时刻估计值得误差协方差最小而推导出来得，根据K得公式，可以看出主要与预测值得协方差矩阵，状态观测矩阵H有关系，也很好理解，如果当前时刻预测值得误差越小，模型更倾向于从预测值多学一些

* 得到K之后就可以计算当前时刻估计值，同时也要更新当前时刻估计值的误差协方差矩阵，用于下一次迭代



#### 基于卡尔曼滤波的跟踪-SORT

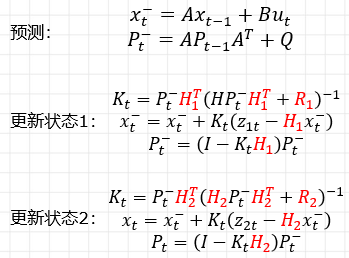


* SORT (Simple Online Realtime Tracking)是一个简单的多目标跟踪框架，结合了**匈牙利匹配和卡尔曼滤波算法**

* 由目标检测网络给出非实时的框，然后使用匈牙利匹配算法对相邻检测结果进行匹配给出每一个框的ID，最后使用卡尔曼滤波对运动轨迹进行平滑，并利用卡尔曼滤波的预测功能，给出无检测结果帧的目标位置

* **SORT的匹配使用的是IOU作为匹配距离**，升级版DeepSORT使用了**目标检测网络输出的embedding特征距离+目标的实际距离作为匹配标准**

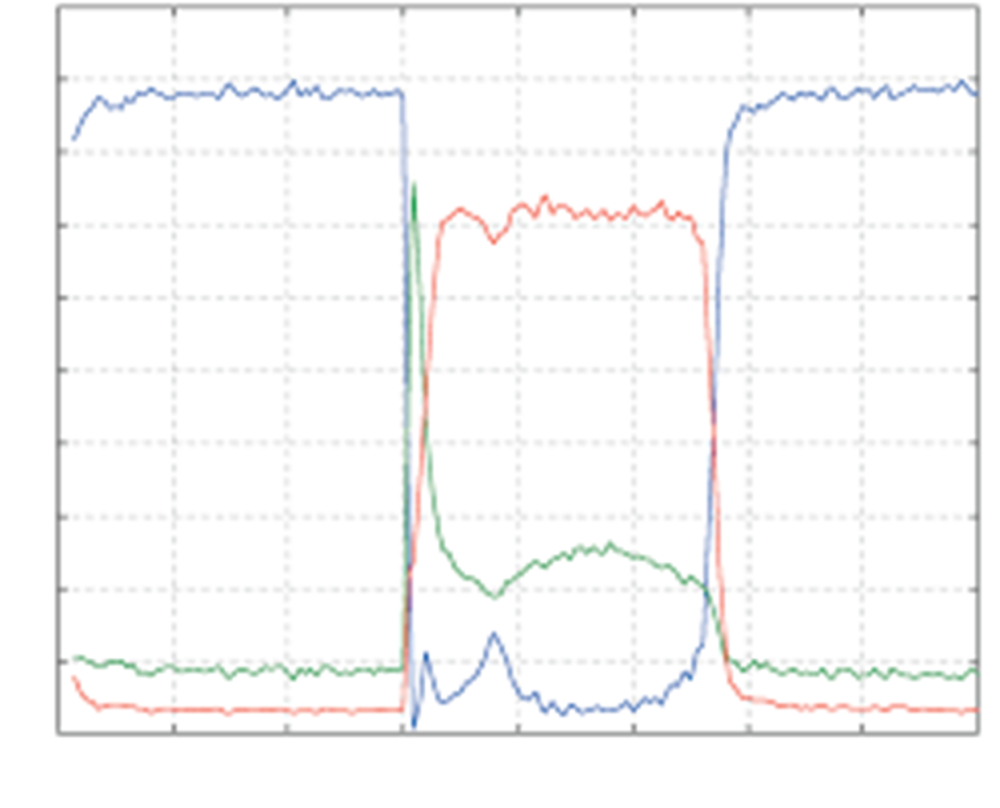
#### 卡尔曼滤波融合多传感器



* 使用卡尔曼滤波融合多传感器观测，其实就是一次预测，多次更新！**——误差存在传递**

* 需要确定两个R矩阵以及两个H矩阵，分别代表两个传感器观测方程和误差大小

#### IMM（ Interacting Multiple Model）模型



* 当使用卡尔曼滤波时，对物体的运动有较强假设，例如匀速直线运动、角速度恒定运动等，但实际情况中车辆的行为在不同阶段是会变化的，例如直行的时候就是匀速直线，而拐弯或通过环岛的时候变为角速度恒定运动

* 为了解决这个问题，提出了IMM模型，也就是同时使用多个运动模型对状态进行预估，哪一个更准确就用哪个（实际是通过评估不同的模型对于真实测量值的误差，来确定模型的权重进行融合，可以认为是对融合的融合）