技术文档丨交通灯感知

知识点

敲黑板，本文需要学习的知识点有

高精地图　长焦相机

广角相机　Rectifier

Recognizer　Reviser

交通灯感知模块旨在使用相机提供准确和全面的交通灯状态信息。通常，交通灯有三种状态，即红、黄、绿。但是，如果交通灯出现故障，它可能会显示黑色或红灯/黄灯闪烁。如果摄像头的视野中无法找到交通灯，模块将无法识别其状态。

那么，交通灯感知模块是如何在Apollo 2.0中工作的呢？

以下，ENJOY

简介

为了解决交通灯出现故障的不确定情况，交通灯感知模块支持五种状态的输出：

- 红色
- 黄色
- 绿色
- 黑色
- 未知

模块**反复查询HD-Map（高精地图）**以确认车辆前方是否有交通灯。交通灯由其边界上的四个点表示，根据车辆的位置，通过查询HD-Map可以获得交通灯信息。如果车辆前方有交通灯，模块会将它从**世界坐标系**投射到**图像坐标系**。

Apollo 认为只使用具有固定视野的单摄像机**无法**保证看到所有地方的交通灯。**局限**是由于以下因素引起的：

① **感知范围可能超过100米**

② **交通灯的高度或交叉路口的宽度变化很大**

因此，Apollo 2.0使用**两个相机**来扩大感知范围：

安装了一个焦距为25毫米的远距摄像机，用来观察**前方远处的交通灯**。在长焦相机中捕获的交通灯**非常大且易于检测**。然而，长焦相机的**视野非常有限**。如果车道不够直，或者车辆与交通灯比较接近，则交通灯通常将处于图像之外（处于相机视野之外）。

长焦相机检测交通信号灯所获图像

一个**广角相机**，其焦距为6mm，用于**提供足够宽的视野**。该模块根据光线投影自适应地选择使用哪个相机。虽然Apollo汽车上只有两个摄像头，但该算法可以处理多个摄像头。

广角相机检测交通信号灯所获图像

流水线

以下将介绍流水线的**两个主要组成部分**：

> **Pre-process（预处理阶段）**
- 交通灯投影
- 相机选择
- 图像和缓存交通灯的同步

> **Process（处理阶段）**
- 矫正 - 提供准确的交通灯边界框
- 识别 - 提供每个边界框的颜色
- 修订 - 根据时间顺序校正颜色

无需对每一帧图像都进行交通灯检测。因为交通灯变化的频率比较低，并且计算资源也比较有限。通常情况下，来自不同摄像机的图像几乎同时到达，但是只有一张图片会被送到整个流水线的Process部分。因此，**图像的选择和匹配**就显得十分必要。

输入and输出

本节介绍**Pre-process模块的输入和输出**。输入可以通过订阅系统中的**相关主题名称**或直接从**本地存储的文件中读取**获得，输出则直接给后续的**Process模块**。

① 输入

- 通过订阅主题名称获得的来自不同相机的图像，例如：

/apollo/sensor/camera/traffic/image_long
/apollo/sensor/camera/traffic/image_short

- 通过查询主题获得的定位信息:

/tf

- 高精地图
- 标定结果

② 输出

- 所选相机拍摄的图片
- 从世界坐标系投射到图像坐标系的交通灯边界框

摄像头选择

交通灯由**唯一ID**和**其边界上的四个点**表示，每个点在世界坐标系中被描述为3D点。
以下示例是**交通灯signal info**的一种典型表示。根据汽车的位置，通过查询HD地图可以获得**四个边界点**。

```
signal info:
id {
  id: "xxx"
}
boundary {
  point { x: ...  y: ...  z: ...  }
  point { x: ...  y: ...  z: ...  }
  point { x: ...  y: ...  z: ...  }
  point { x: ...  y: ...  z: ...  }
}
```

然后将**三维世界坐标中的边界点**投影到每个相机的**二维图像坐标系**中。对于一个交通灯而言，由长焦相机图片中四个投影点描述的边界框区域比较大。检测效果优于广角相机图像。因此，可以看到**所有交通灯的最长焦距相机的图像**将被选为输出图像。投影在此图像上的**交通灯边界框**将作为**边框输出**。

带有**时间戳**的被选中摄像机ID将被缓存到队列中，如下所示：

```
struct ImageLights {
  CarPose pose;
  CameraId camera_id;
  double timestamp;
  size_t num_signal;
  ... other ...
};
```

到目前为止，我们需要的所有信息包括**定位，标定结果和高精地图**。选择过程可以在任何时间点进行，因为投影是独立于图像内容的。在图像到达时执行选择任务仅仅是为了简单起见。此外，不需要在每个图像到达时执行图像选择，可以为设选择设置一个**时间间隔**。

图像同步

图像到达时带有**时间戳**和**摄像机ID**等信息。时间戳和摄像机ID的配对用于查找适当的**缓存信息**。如果可以找到一个具有相同摄像机ID且时间戳与当前图像时间戳**相差很小的缓存记录**，则可以将图像发布到**“Process”模块**。所有不合适的图像都被废弃。

Process模块分为如下三个步骤，每一步聚焦一个任务上

- 矫正 – 在ROI中检测交通灯边框
- 识别 – 对边界框的颜色进行分类
- 修订 - 根据时序信息更正颜色

输入and输出

本节介绍Process阶段的输入和输出数据。**输入从Pre-process模块获得，输出将作为交通灯主题发布。**

① 输入

- 所选摄像机拍摄的图像
- 一组边框数据

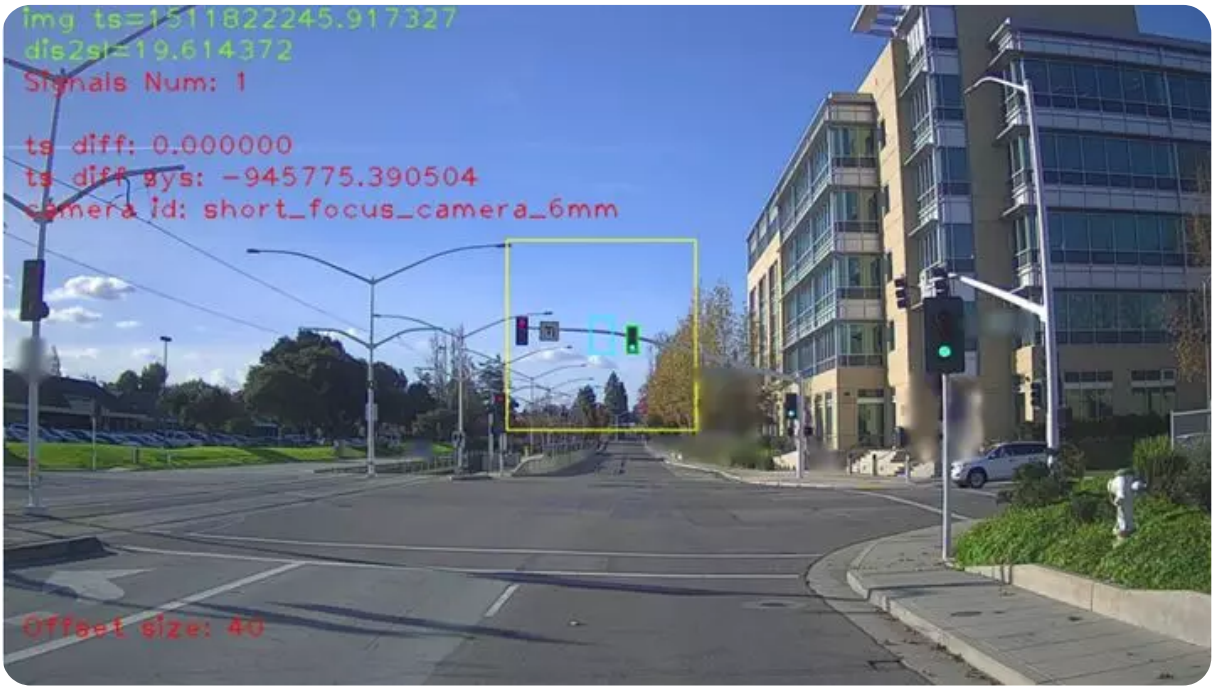
② 输出

- 一组带颜色标签的边框

Rectifier （矫正器）

受标定，定位和高精地图标签的影响，**投影位置并不完全可靠**。基于交通灯投影位置计算得到的**较大感兴趣区域（ROI）**将被用于找到更加精确的交通灯边界框。

在下面的照片中，**蓝色矩形**表示投影交通灯边框，它与实际交通灯的位置还有很大偏差。范围更大的**黄色矩形框**是ROI。



交通灯检测被当成一个**常规卷积神经网络（CNN）检测任务**来实现。它接收一个带ROI的图形作为输入，输出一系列边框。ROI中的交通灯可能比输入中的交通灯多。

Apollo需要根据**检测分数、输入的交通灯位置和形状**来选择合适的交通灯。如果CNN网络无法在ROI中找到任何交通灯，则输入交通灯的状态将标记为**未知**，并跳过剩余的两个步骤（识别器和修订器）。

Recognizer（识别器）

交通灯识别被当成一个典型的**CNN分类任务**实现。该神经网络以一个**带ROI的图形**和一个**边框列表**作为输入。具有ROI和边界框列表作为输入的图像。网络的输出一个**4*n的向量**，表示每个**候选框**的分别为**黑色，红色，黄色和绿色的概率**。

有且只有概率足够大时，具有**最大概率的类**才被视为交通灯的状态。否则，交通灯的状态将设置为**黑色**，这意味着**状态不确定**。

由于红绿灯可能闪烁或被遮挡，并且识别器不是完美的，因此当前状态可能无法表示真实状态。一个可以**纠正状态的修订器**就显得十分必要。

如果修订器收到一个确定的状态（如红色或绿色），"修订器"将直接保存并输出该状态。如果接收到的状态为黑色或未知，则**Reviser将查找保存的地图**。如果交通灯的状态在一段时间内都是确定，**Reviser将输出保存的状态**。否则，黑色或未知的状态将作为输出。

由于时间顺序的影响，黄灯只存在于绿灯之后和红灯之前。为了安全起见，红灯之后的任何黄灯都将重置为红灯， 直到绿灯亮起。